intel®

Embedded Microprocessors

1996

Intel486™ Processors

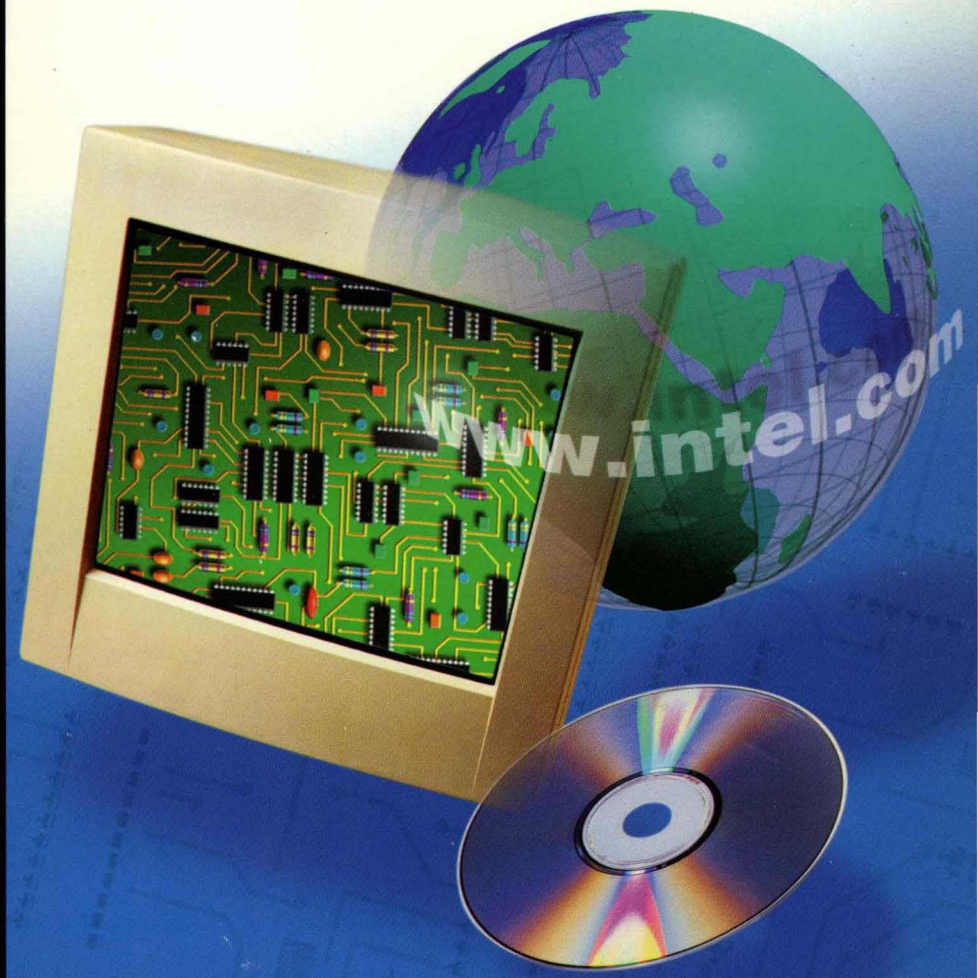
Intel386™ Processors

Intel 376 Processors and Peripherals

80186/80188 Family

intel®

Embedded Microprocessors



LITERATURE

For additional information on Intel products in the U.S. or Canada, call Intel's Literature Center at (800) 548-4725 or write to:

INTEL LITERATURE SALES
P. O. Box 7641
Mt. Prospect, IL 60056-7641

To order literature outside of the U.S. and Canada contact your local sales office.

Additional information about Intel products is available on Intel's web site: http://www.intel.com.

CURRENT DATABOOKS

Product line databooks contain datasheets, application notes, article reprints, and other design information. All databooks can be ordered individually, and most are available in a pre-packaged set in the U.S. and Canada. Databooks can be ordered in the U.S. and Canada by calling TAB/McGraw-Hill at 1-800-822-8158; outside of the U.S. and Canada contact your local sales office.

Title	Intel Order Number	ISBN
SET OF NINE DATABOOKS (Available in U.S. and Canada)	**231003**	**N/A**
CONTENTS LISTED BELOW FOR INDIVIDUAL ORDERING:		
EMBEDDED MICROCONTROLLERS	270646	1-55512-248-5
EMBEDDED MICROPROCESSORS	272396	1-55512-249-3
FLASH MEMORY (2 volume set)	210830	1-55512-250-7
i960® PROCESSORS AND RELATED PRODUCTS	272084	1-55512-252-3
NETWORKING	297360	1-55512-256-6
OEM BOARDS, SYSTEMS AND SOFTWARE	280407	1-55512-253-1
PACKAGING	240800	1-55512-254-X
PENTIUM® AND PENTIUM PRO PROCESSORS AND RELATED PRODUCTS	241732	1-55512-251-5
PERIPHERAL COMPONENTS	296467	1-55512-255-8
ADDITIONAL LITERATURE: (Not included in handbook set)		
AUTOMOTIVE PRODUCTS	231792	1-55512-257-4
COMPONENTS QUALITY/RELIABILITY	210997	1-55512-258-2
EMBEDDED APPLICATIONS (1995/96)	270648	1-55512-179-9
MILITARY	210461	N/A
SYSTEMS QUALITY/RELIABILITY	231762	1-55512-046-6

A complete set of this information is available on CD-ROM through Intel's Data on Demand program, order number 240897. For information about Intel's Data on Demand ask for item number 240952.

December 1995
Order Number: 000900-001

Intel Application Support Services

World Wide Web [URL: http://www.intel.com/]

Intel's Web site now contains technical and product information that is available 24-hours a day! Also visit Intel's site for financials, history, current news and events, job opportunities, educational news and much, much more!

FaxBack*

Technical and product information are available 24-hours a day! Order documents containing:

- Product Announcements
- Product Literature
- Intel Device Characteristics
- Design/Application Recommendations
- Stepping/Change Notifications
- Quality and Reliability Information

Information on the following subjects are available:

- Microcontroller and Flash
- OEM Branded Systems
- Multibus and iRMX Software/BBS listing
- Multimedia
- Development Tools
- Quality and Reliability/Change Notification
- Microprocessor/PCI/Peripheral
- Intel Architecture Labs

To use FaxBack (for Intel components and systems), dial **(800) 628-2283** or 916-356-3105 (U.S./Canada/APAC/Japan) *or +44{0} 1793-496646 (Europe)* and follow the automated voice-prompt. Document orders will be faxed to the fax number you specify. For information on how the Intel Application Support team can help you, order our Customer Service Agreement, document #1201. Catalogs are updated as needed, so call for the latest information!

Bulletin Board System (BBS)

To use the Intel Application BBS (components and systems), dial **(503) 264-7999** or **(916) 356-3600** (U.S./Canada/APAC/Japan) *or +44{0} 1793-432955 (Europe).* The BBS will support 1200-19200 baud rate modem. *Typical modem configuration: 14.4K baud rate, No Parity, 8 Data Bits, 1 Stop Bit.*

CompuServe *Just type* 'Go Intel'

Intel maintains several forums where people come together to meet their peers, gather information, share discoveries and debate issues. For more information about service fees and access, call CompuServe at 1-800-848-8199 or 614-529-1340 (outside the U.S.). The INTELC forum is setup to support designers using various Intel components.

General Information Help Desk

Dial 1-800-628-8686 or 916-356-7599 (U.S. and Canada) between 5 a.m. and 5 p.m. PST for help with Intel products. For customers not in the U.S. or Canada, please contact your local distributor.

Intel Literature Centers

U.S.	+1-800-548-4725	France	+44{0} 1793 421777
U.S. (from overseas)	+1-708-296-9333	Germany	+44{0} 1793 421333
England	+44{0} 1793 431 155	Japan (fax only)	+81{0} 120 47 88 32

Intel Distributors

Check the back of an Intel data book or request one of the following distributor listing FaxBack documents: #4083 (U.S. Eastern Time Zone), #4084 (U.S. Central Time Zone), #4085 (Mountain Time Zone), #4086 (U.S. Alaska/Pacific Time Zone), #4209 (Europe) or #4403 (Canada).

December 1995
Order Number: 000901-001

intel®

Embedded Microprocessors

Intel486™ Processors, Intel386™ Processors, Intel376 Processors and Peripherals, 80186/80188 Family

1996

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

DATASHEET DESIGNATIONS

Intel uses various datasheet markings to designate each phase of the document as it relates to the product. The markings appear in the lower inside corner of each datasheet page. Following are the definitions of each marking:

Datasheet Marking	Description
Product Preview	Contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product becomes available.
Advanced Information	Contains information on products being sampled or in the initial production phase of development.*
Preliminary	Contains preliminary information on new products in production.*
No Marking	Contains information on products in full production.*

* Specifications within these datasheets are subject to change without notice. Verify with your local Intel sales office that you have the latest datasheet before finalizing a design.

November 1995
Order Number: 000902-001

Table of Contents

Table of Contents (Continued)

Alphanumeric Index

intel®

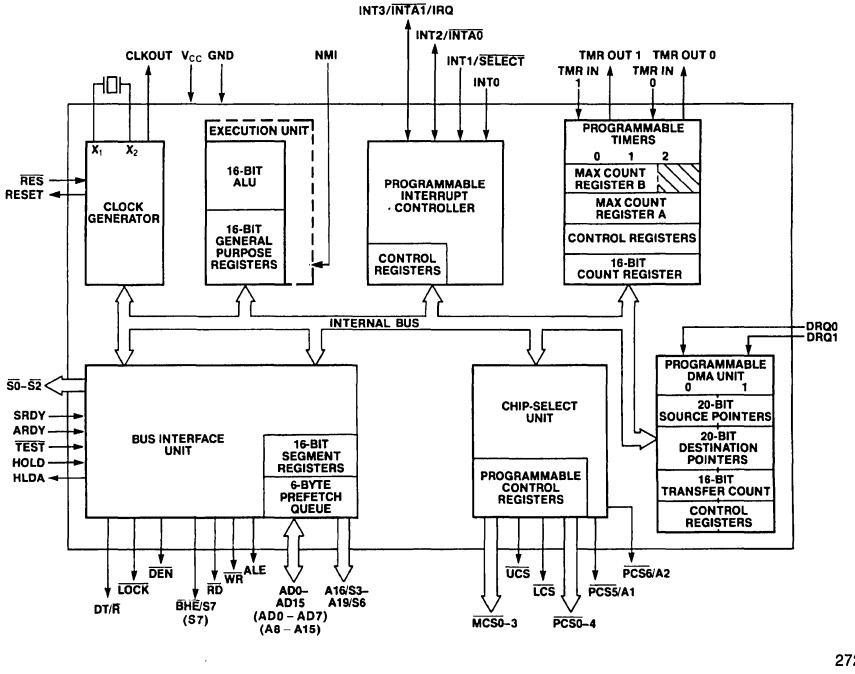
1

80186/80188 Family

80186/80188
HIGH-INTEGRATION 16-BIT MICROPROCESSORS

- **Integrated Feature Set**
  - — Enhanced 8086-2 CPU
  - — Clock Generator
  - — 2 Independent DMA Channels
  - — Programmable Interrupt Controller
  - — 3 Programmable 16-bit Timers
  - — Programmable Memory and Peripheral Chip-Select Logic
  - — Programmable Wait State Generator
  - — Local Bus Controller
- **Available in 10 MHz and 8 MHz Versions**
- **High-Performance Processor**
  - — 4 Mbyte/Sec Bus Bandwidth Interface @ 8 MHz (80186)
  - — 5 Mbyte/Sec Bus Bandwidth Interface @ 10 MHz (80186)
- **Direct Addressing Capability to 1 Mbyte of Memory and 64 Kbyte I/O**
- **Completely Object Code Compatible with All Existing 8086, 8088 Software**
  - — 10 New Instruction Types
- **Numerics Coprocessing Capability Through 8087 Interface**
- **Available in 68 Pin:**
  - — Plastic Leaded Chip Carrier (PLCC)
  - — Ceramic Pin Grid Array (PGA)
  - — Ceramic Leadless Chip Carrier (LCC)
- **Available in EXPRESS**
  - — Standard Temperature with Burn-In
  - — Extended Temperature Range (−40°C to +85°C)

1



272430-1

Figure 1. Block Diagram

80186/80188 High-Integration 16-Bit Microprocessors

CONTENTS

CONTENTS

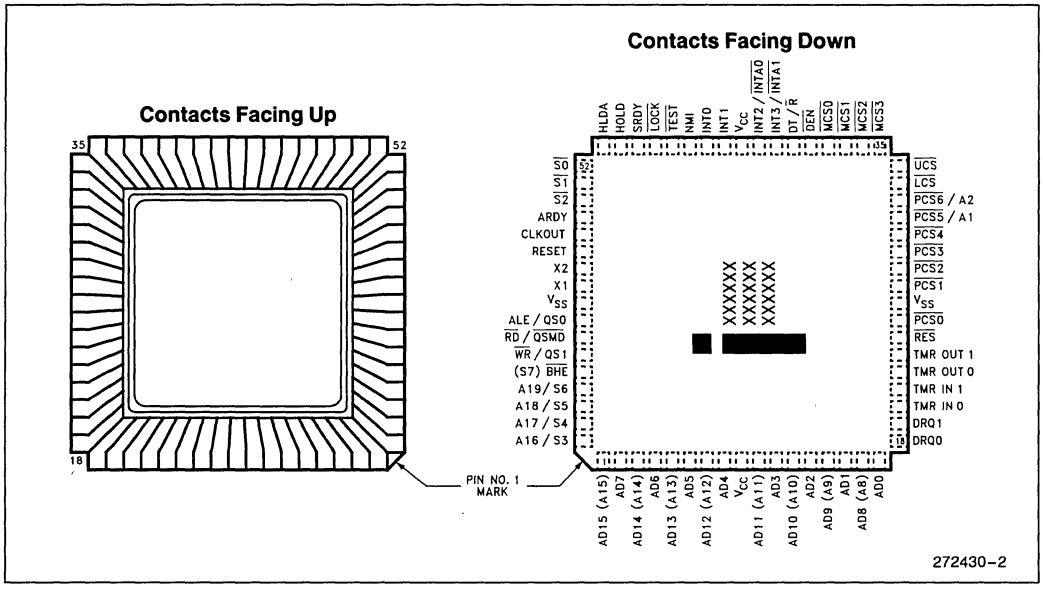


Figure 2. Ceramic Leadless Chip Carrier (JEDEC Type A)

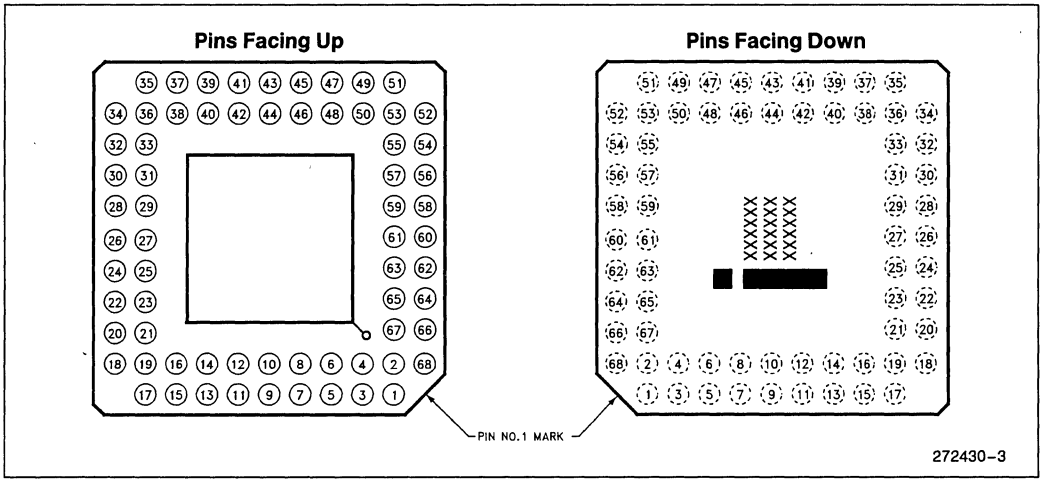


Figure 3. Ceramic Pin Grid Array

NOTE:
Pin names in parentheses apply to the 80188.

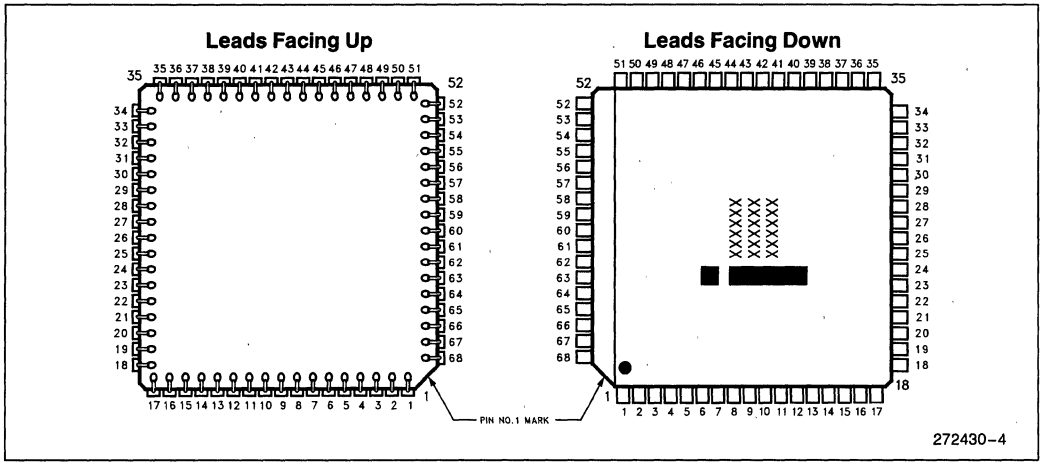


Figure 4. Plastic Leaded Chip Carrier

NOTE:
Pin names in parentheses apply to the 80188.

Table 1. Pin Descriptions

Symbol	Pin No.	Type	Name and Function
V_{CC}	9 43	I	**SYSTEM POWER:** +5 volt power supply.
V_{SS}	26 60	I	System Ground.
RESET	57	O	Reset Output indicates that the CPU is being reset, and can be used as a system reset. It is active HIGH, synchronized with the processor clock, and lasts an integer number of clock periods corresponding to the length of the $\overline{RES}$ signal.
X1 X2	59 58	I O	Crystal Inputs X1 and X2 provide external connections for a fundamental mode parallel resonant crystal for the internal oscillator. Instead of using a crystal, an external clock may be applied to X1 while minimizing stray capacitance on X2. The input or oscillator frequency is internally divided by two to generate the clock signal (CLKOUT).
CLKOUT	56	O	Clock Output provides the system with a 50% duty cycle waveform. All device pin timings are specified relative to CLKOUT.
$\overline{RES}$	24	I	An active $\overline{RES}$ causes the processor to immediately terminate its present activity, clear the internal logic, and enter a dormant state. This signal may be asynchronous to the processor clock. The processor begins fetching instructions approximately 6½ clock cycles after $\overline{RES}$ is returned HIGH. For proper initialization, V_{CC} must be within specifications and the clock signal must be stable for more than 4 clocks with $\overline{RES}$ held LOW. $\overline{RES}$ is internally synchronized. This input is provided with a Schmitt-trigger to facilitate power-on $\overline{RES}$ generation via an RC network.
$\overline{TEST}$	47	I/O	$\overline{TEST}$ is examined by the WAIT instruction. If the $\overline{TEST}$ input is HIGH when "WAIT" execution begins, instruction execution will suspend. $\overline{TEST}$ will be resampled until it goes LOW, at which time execution will resume. If interrupts are enabled while the processor is waiting for $\overline{TEST}$, interrupts will be serviced. During power-up, active $\overline{RES}$ is required to configure $\overline{TEST}$ as an input. This pin is synchronized internally.
TMR IN 0 TMR IN 1	20 21	I I	Timer Inputs are used either as clock or control signals, depending upon the programmed timer mode. These inputs are active HIGH (or LOW-to-HIGH transitions are counted) and internally synchronized.
TMR OUT 0 TMR OUT 1	22 23	O O	Timer outputs are used to provide single pulse or continous waveform generation, depending upon the timer mode selected.
DRQ0 DRQ1	18 19	I I	DMA Request is asserted HIGH by an external device when it is ready for DMA Channel 0 or 1 to perform a transfer. These signals are level-triggered and internally synchronized.
NMI	46	I	The Non-Maskable Interrupt input causes a Type 2 interrupt. An NMI transition from LOW to HIGH is latched and synchronized internally, and initiates the interrupt at the next instruction boundary. NMI must be asserted for at least one clock. The Non-Maskable Interrupt cannot be avoided by programming.
INT0 INT1/$\overline{SELECT}$ INT2/$\overline{INTA0}$ INT3/$\overline{INTA1}$/IRQ	45 44 42 41	I I I/O I/O	Maskable Interrupt Requests can be requested by activating one of these pins. When configured as inputs, these pins are active HIGH. Interrupt Requests are synchronized internally. INT2 and INT3 may be configured to provide active-LOW interrupt-acknowledge output signals. All interrupt inputs may be configured to be either edge- or level-triggered. To ensure recognition, all interrupt requests must remain active until the interrupt is acknowledged. When Slave Mode is selected, the function of these pins changes (see Interrupt Controller section of this data sheet).

NOTE:
Pin names in parentheses apply to the 80188.

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function
A19/S6 A18/S5 A17/S4 A16/S3	65 66 67 68	O O O O	Address Bus Outputs (16–19) and Bus Cycle Status (3–6) indicate the four most significant address bits during T_1. These signals are active HIGH. During T_2, T_3, T_W, and T_4, the S6 pin is LOW to indicate a CPU-initiated bus cycle or HIGH to indicate a DMA-initiated bus cycle. During the same T-states, S3, S4, and S5 are always LOW. The status pins float during bus HOLD or RESET.
AD15 (A15) AD14 (A14) AD13 (A13) AD12 (A12) AD11 (A11) AD10 (A10) AD9 (A9) AD8 (A8) AD7 AD6 AD5 AD4 AD3 AD2 AD1 AD0	1 3 5 7 10 12 14 16 2 4 6 8 11 13 15 17	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	Address/Data Bus signals constitute the time multiplexed memory or I/O address (T_1) and data (T_2, T_3, T_W, and T_4) bus. The bus is active HIGH. A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7 through D_0. It is LOW during T_1 when a byte is to be transferred onto the lower portion of the bus in memory or I/O operations. $\overline{BHE}$ does not exist on the 80188, as the data bus is only 8 bits wide.
$\overline{BHE}$/S7 (S7)	64	O	During T_1 the Bus High Enable signal should be used to determine if data is to be enabled onto the most significant half of the data bus; pins D_{15}–D_8. $\overline{BHE}$ is LOW during T_1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the higher half of the bus. The S_7 status information is available during T_2, T_3, and T_4. S_7 is logically equivalent to $\overline{BHE}$. $\overline{BHE}$/S7 floats during HOLD. On the 80188, S7 is high during normal operation.

$\overline{BHE}$ and A0 Encodings (80186 Only)

$\overline{BHE}$ Value	A0 Value	Function
0	0	Word Transfer
0	1	Byte Transfer on upper half of data bus (D15–D8)
1	0	Byte Transfer on lower half of data bus (D_7–D_0)
1	1	Reserved

ALE/QS0	61	O	Address Latch Enable/Queue Status 0 is provided by the processor to latch the address. ALE is active HIGH. Addresses are guaranteed to be valid on the trailing edge of ALE. The ALE rising edge is generated off the rising edge of the CLKOUT immediately preceding T_1 of the associated bus cycle, effectively one-half clock cycle earlier than in the 8086. The trailing edge is generated off the CLKOUT rising edge in T_1 as in the 8086. Note that ALE is never floated.
$\overline{WR}$/QS1	63	O	Write Strobe/Queue Status 1 indicates that the data on the bus is to be written into a memory or an I/O device. $\overline{WR}$ is active for T_2, T_3, and T_W of any write cycle. It is active LOW, and floats during HOLD. When the processor is in queue status mode, the ALE/QS0 and $\overline{WR}$/QS1 pins provide information about processor/instruction queue interaction.

QS1	QS0	Queue Operation
0	0	No queue operation
0	1	First opcode byte fetched from the queue
1	1	Subsequent byte fetched from the queue
1	0	Empty the queue

NOTE:
Pin names in parentheses apply to the 80188.

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{RD}/\overline{QSMD}$	62	I/O	Read Strobe is an active LOW signal which indicates that the processor is performing a memory or I/O read cycle. It is guaranteed not to go LOW before the A/D bus is floated. An internal pull-up ensures that $\overline{RD}$ is HIGH during RESET. Following RESET the pin is sampled to determine whether the processor is to provide ALE, $\overline{RD}$, and $\overline{WR}$, or queue status information. To enable Queue Status Mode, $\overline{RD}$ must be connected to GND. $\overline{RD}$ will float during bus HOLD.
ARDY	55	I	Asynchronous Ready informs the processor that the addressed memory space or I/O device will complete a data transfer. The ARDY pin accepts a rising edge that is asynchronous to CLKOUT, and is active HIGH. The falling edge of ARDY must be synchronized to the processor clock. Connecting ARDY HIGH will always assert the ready condition to the CPU. If this line is unused, it should be tied LOW to yield control to the SRDY pin.
SRDY	49	I	Synchronous Ready informs the processor that the addressed memory space or I/O device will complete a data transfer. The SRDY pin accepts an active-HIGH input synchronized to CLKOUT. The use of SRDY allows a relaxed system timing over ARDY. This is accomplished by elimination of the one-half clock cycle required to internally synchronize the ARDY input signal. Connecting SRDY high will always assert the ready condition to the CPU. If this line is unused, it should be tied LOW to yield control to the ARDY pin.
$\overline{LOCK}$	48	O	$\overline{LOCK}$ output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is requested by the LOCK prefix instruction and is activated at the beginning of the first data cycle associated with the instruction following the LOCK prefix. It remains active until the completion of that instruction. No instruction prefetching will occur while $\overline{LOCK}$ is asserted. When executing more than one LOCK instruction, always make sure there are 6 bytes of code between the end of the first LOCK instruction and the start of the second LOCK instruction. $\overline{LOCK}$ is driven HIGH for one clock during RESET and then floated.
$\overline{S0}$ $\overline{S1}$ $\overline{S2}$	52 53 54	O O O	Bus cycle status $\overline{S0}$–$\overline{S2}$ are encoded to provide bus-transaction information: **Bus Cycle Status Information** $\overline{S2}$ $\overline{S1}$ $\overline{S0}$ — Bus Cycle Initiated 0 0 0 — Interrupt Acknowledge 0 0 1 — Read I/O 0 1 0 — Write I/O 0 1 1 — Halt 1 0 0 — Instruction Fetch 1 0 1 — Read Data from Memory 1 1 0 — Write Data to Memory 1 1 1 — Passive (no bus cycle) The status pins float during HOLD. $\overline{S2}$ may be used as a logical M/$\overline{IO}$ indicator, and $\overline{S1}$ as a DT/$\overline{R}$ indicator.

NOTE:
Pin names in parentheses apply to the 80188.

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function
HOLD HLDA	50 51	I O	HOLD indicates that another bus master is requesting the local bus. The HOLD input is active HIGH. HOLD may be asynchronous with respect to the processor clock. The processor will issue a HLDA (HIGH) in response to a HOLD request at the end of T_4 or T_i. Simultaneous with the issuance of HLDA, the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will lower HLDA. When the processor needs to run another bus cycle, it will again drive the local bus and control lines.
$\overline{UCS}$	34	O	Upper Memory Chip Select is an active LOW output whenever a memory reference is made to the defined upper portion (1K–256K block) of memory. This line is not floated during bus HOLD. The address range activating $\overline{UCS}$ is software programmable.
$\overline{LCS}$	33	O	Lower Memory Chip Select is active LOW whenever a memory reference is made to the defined lower portion (1K–256K) of memory. This line is not floated during bus HOLD. The address range activating $\overline{LCS}$ is software programmable.
$\overline{MCS0}$ $\overline{MCS1}$ $\overline{MCS2}$ $\overline{MCS3}$	38 37 36 35	O O O O	Mid-Range Memory Chip Select signals are active LOW when a memory reference is made to the defined mid-range portion of memory (8K–512K). These lines are not floated during bus HOLD. The address ranges activating $\overline{MCS0}$–3 are software programmable.
$\overline{PCS0}$ $\overline{PCS1}$ $\overline{PCS2}$ $\overline{PCS3}$ $\overline{PCS4}$	25 27 28 29 30	O O O O O	Peripheral Chip Select signals 0–4 are active LOW when a reference is made to the defined peripheral area (64 Kbyte I/O space). These lines are not floated during bus HOLD. The address ranges activating $\overline{PCS0}$–4 are software programmable.
$\overline{PCS5}$/A1	31	O	Peripheral Chip Select 5 or Latched A1 may be programmed to provide a sixth peripheral chip select, or to provide an internally latched A1 signal. The address range activating $\overline{PCS5}$ is software-programmable. $\overline{PCS5}$/A1 does not float during bus HOLD. When programmed to provide latched A1, this pin will retain the previously latched value during HOLD.
$\overline{PCS6}$/A2	32	O	Peripheral Chip Select 6 or Latched A2 may be programmed to provide a seventh peripheral chip select, or to provide an internally latched A2 signal. The address range activating $\overline{PCS6}$ is software programmable. $\overline{PCS6}$/A2 does not float during bus HOLD. When programmed to provide latched A2, this pin will retain the previously latched value during HOLD.
DT/$\overline{R}$	40	O	Data Transmit/Receive controls the direction of data flow through an external data bus transceiver. When LOW, data is transferred to the processsor. When HIGH, the processor places write data on the data bus.
$\overline{DEN}$	39	O	Data Enable is provided as a data bus transceiver output enable. $\overline{DEN}$ is active LOW during each memory and I/O access. $\overline{DEN}$ is HIGH whenever DT/$\overline{R}$ changes state. During RESET, $\overline{DEN}$ is driven HIGH for one clock, then floated. $\overline{DEN}$ also floats during HOLD.

NOTE:
Pin names in parentheses apply to the 80188.

FUNCTIONAL DESCRIPTION

Introduction

The following Functional Description describes the base architecture of the 80186. The 80186 is a very high integration 16-bit microprocessor. It combines 15–20 of the most common microprocessor system components onto one chip while providing twice the performance of the standard 8086. The 80186 is object code compatible with the 8086/8088 microprocessors and adds 10 new instruction types to the 8086/8088 instruction set.

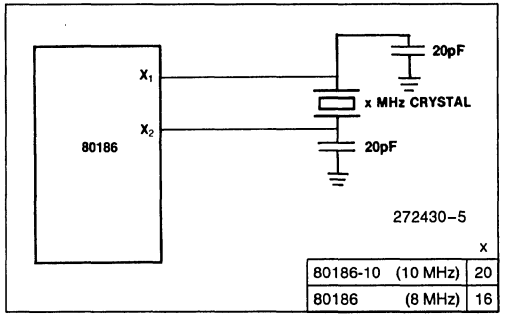
For more detailed information on the architecture, please refer to the 80C186XL/80C188XL User's Manual. The 80186 and the 80186XL devices are functionally and register compatible.

CLOCK GENERATOR

The processor provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter, synchronous and asynchronous ready inputs, and reset circuitry.

Oscillator

The oscillator circuit is designed to be used with a parallel resonant fundamental mode crystal. This is used as the time base for the processor. The crystal frequency selected will be double the CPU clock frequency. Use of an LC or RC circuit is not recommended with this oscillator. If an external oscillator is used, it can be connected directly to the input pin X1 in lieu of a crystal. The output of the oscillator is not directly available outside the processor. The recommended crystal configuration is shown in Figure 5.



		x
80186-10	(10 MHz)	20
80186	(8 MHz)	16

Figure 5. Recommended Crystal Configuration

Intel recommends the following values for crystal selection parameters:

Temperature Range:	0 to 70°C
ESR (Equivalent Series Resistance):	30Ω max
C_0 (Shunt Capacitance of Crystal):	7.0 pf max
C_1 (Load Capacitance):	20 pf ± 2 pf
Drive Level:	1 mW max

Clock Generator

The clock generator provides the 50% duty cycle processor clock for the processor. It does this by dividing the oscillator output by 2 forming the symmetrical clock. If an external oscillator is used, the state of the clock generator will change on the falling edge of the oscillator signal. The CLKOUT pin provides the processor clock signal for use outside the device. This may be used to drive other system components. All timings are referenced to the output clock.

READY Synchronization

The processor provides both synchronous and asynchronous ready inputs. In addition, the processor, as part of the integrated chip-select logic, has the capability to program WAIT states for memory and peripheral blocks.

RESET Logic

The processor provides both a $\overline{RES}$ input pin and a synchronized RESET output pin for use with other system components. The $\overline{RES}$ input pin is provided with hysteresis in order to facilitate power-on Reset generation via an RC network. RESET output is guaranteed to remain active for at least five clocks given a $\overline{RES}$ input of at least six clocks.

LOCAL BUS CONTROLLER

The processor provides a local bus controller to generate the local bus control signals. In addition, it employs a HOLD/HLDA protocol for relinquishing the local bus to other bus masters. It also provides outputs that can be used to enable external buffers and to direct the flow of data on and off the local bus.

Memory/Peripheral Control

The processor provides ALE, $\overline{RD}$, and $\overline{WR}$ bus control signals. The $\overline{RD}$ and $\overline{WR}$ signals are used to strobe data from memory or I/O to the processor or to strobe data from the processor to memory or I/O. The ALE line provides a strobe to latch the address when it is valid. The local bus controller does not provide a memory/$\overline{I/O}$ signal. If this is required, use the $\overline{S2}$ signal (which will require external latching), make the memory and I/O spaces nonoverlapping, or use only the integrated chip-select circuitry.

Local Bus Arbitration

The processor uses a HOLD/HLDA system of local bus exchange. This provides an asynchronous bus exchange mechanism. This means multiple masters utilizing the same bus can operate at separate clock frequencies. The processor provides a single HOLD/HLDA pair through which all other bus masters may gain control of the local bus. External circuitry must arbitrate which external device will gain control of the bus when there is more than one alternate local bus master. When the processor relinquishes control of the local bus, it floats $\overline{DEN}$, $\overline{RD}$, $\overline{WR}$, $\overline{S0}$–$\overline{S2}$, $\overline{LOCK}$, AD0–AD15 (AD0–AD7), A16–A19 (A8–A19), $\overline{BHE}$ (S7), and DT/$\overline{R}$ to allow another master to drive these lines directly.

Local Bus Controller and Reset

During RESET the local bus controller will perform the following action:

- Drive $\overline{DEN}$, $\overline{RD}$, and $\overline{WR}$ HIGH for one clock cycle, then float.

NOTE:
$\overline{RD}$ is also provided with an internal pull-up device to prevent the processor from inadvertently entering Queue Status Mode during RESET.

- Drive $\overline{S0}$–$\overline{S2}$ to the inactive state (all HIGH) and then float.
- Drive $\overline{LOCK}$ HIGH and then float.
- Float AD0–15 (AD0–AD7), A16–19 (A8–A19), $\overline{BHE}$ (S7), DT/$\overline{R}$.
- Drive ALE LOW (ALE is never floated).
- Drive HLDA LOW.

PERIPHERAL ARCHITECTURE

All of the integrated peripherals are controlled by 16-bit registers contained within an internal 256-byte control block. The control block may be mapped into either memory or I/O space. Internal logic will recognize control block addresses and respond to bus cycles. During bus cycles to internal registers, the bus controller will signal the operation externally (i.e., the $\overline{RD}$, $\overline{WR}$, status, address, data, etc., lines will be driven as in a normal bus cycle), but D_{15-0} (D_{7-0}), SRDY, and ARDY will be ignored. The base address of the control block must be on an even 256-byte boundary (i.e., the lower 8 bits of the base address are all zeros).

The control block base address is programmed by a 16-bit relocation register contained within the control block at offset FEH from the base address of the control block. It provides the upper 12 bits of the base address of the control block.

In addition to providing relocation information for the control block, the relocation register contains bits which place the interrupt controller into Slave Mode, and cause the CPU to interrupt upon encountering ESC instructions.

Chip-Select/Ready Generation Logic

The processor contains logic which provides programmable chip-select generation for both memories and peripherals. In addition, it can be programmed to provide READY (or WAIT state) generation. It can also provide latched address bits A1 and A2. The chip-select lines are active for all memory and I/O cycles in their programmed areas, whether they be generated by the CPU or by the integrated DMA unit.

MEMORY CHIP SELECTS

The processor provides 6 memory chip select outputs for 3 address areas; upper memory, lower memory, and midrange memory. One each is provided for upper memory and lower memory, while four are provided for midrange memory.

UPPER MEMORY $\overline{CS}$

The processor provides a chip select, called $\overline{UCS}$, for the top of memory. The top of memory is usually used as the system memory because after reset the processor begins executing at memory location FFFF0H.

LOWER MEMORY $\overline{CS}$

The processor provides a chip select for low memory called $\overline{LCS}$. The bottom of memory contains the interrupt vector table, starting at location 00000H.

The lower limit of memory defined by this chip select is always 0H, while the upper limit is programmable. By programming the upper limit, the size of the memory block is defined.

MID-RANGE MEMORY $\overline{CS}$

The processor provides four $\overline{MCS}$ lines which are active within a user-locatable memory block. This block can be located within the 1-Mbyte memory address space exclusive of the areas defined by $\overline{UCS}$ and $\overline{LCS}$. Both the base address and size of this memory block are programmable.

PERIPHERAL CHIP SELECTS

The processor can generate chip selects for up to seven peripheral devices. These chip selects are active for seven contiguous blocks of 128 bytes above a programmable base address. The base address may be located in either memory or I/O space. Seven $\overline{CS}$ lines called $\overline{PCS0}$–6 are generated by the processor. $\overline{PCS5}$ and $\overline{PCS6}$ can also be programmed to provide latched address bits A1 and A2. If so programmed, they cannot be used as peripheral selects. These outputs can be connected directly to the A0 and A1 pins used for selecting internal registers of 8-bit peripheral chips.

READY GENERATION LOGIC

The processor can generate a READY signal internally for each of the memory or peripheral $\overline{CS}$ lines. The number of WAIT states to be inserted for each peripheral or memory is programmable to provide 0–3 wait states for all accesses to the area for which the chip select is active. In addition, the processor may be programmed to either ignore external READY for each chip-select range individually or to factor external READY with the integrated ready generator.

CHIP SELECT/READY LOGIC AND RESET

Upon RESET, the Chip-Select/Ready Logic will perform the following actions:

- All chip-select outputs will be driven HIGH.
- Upon leaving RESET, the $\overline{UCS}$ line will be programmed to provide chip selects to a 1K block with the accompanying READY control bits set at 011 to insert 3 wait states in conjunction with external READY (i.e., UMCS resets to FFFBH).
- No other chip select or READY control registers have any predefined values after RESET. They will not become active until the CPU accesses their control registers. Both the PACS and MPCS registers must be accessed before the $\overline{PCS}$ lines will become active.

DMA Channels

The DMA controller provides two independent DMA channels. Data transfers can occur between memory and I/O spaces (e.g., Memory to I/O) or within the same space (e.g., Memory to Memory or I/O to I/O). Data can be transferred either in bytes or in words (80186 only) to or from even or odd addresses. Each DMA channel maintains both a 20-bit source and destination pointer which can be optionally incremented or decremented after each data transfer (by one or two depending on byte or word transfers). Each data transfer consumes 2 bus cycles (a minimum of 8 clocks), one cycle to fetch data and the other to store data. This provides a maximum data transfer rate of 1.25 Mword/sec or 2.5 Mbytes/sec at 10 MHz (half of this rate for the 80188).

DMA CHANNELS AND RESET

Upon RESET, the DMA channels will perform the following actions:

- The Start/Stop bit for each channel will be reset to STOP.
- Any transfer in progress is aborted.

Timers

The processor provides three internal 16-bit programmable timers. Two of these are highly flexible and are connected to four external pins (2 per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real-time coding and time delay applications. In addition, the third timer can be used as a prescaler to the other two, or as a DMA request source.

TIMERS AND RESET

Upon RESET, the Timers will perform the following actions:

- All EN (Enable) bits are reset preventing timer counting.
- For Timers 0 and 1, the RIU bits are reset to zero and the ALT bits are set to one. This results in the Timer Out pins going high.

Interrupt Controller

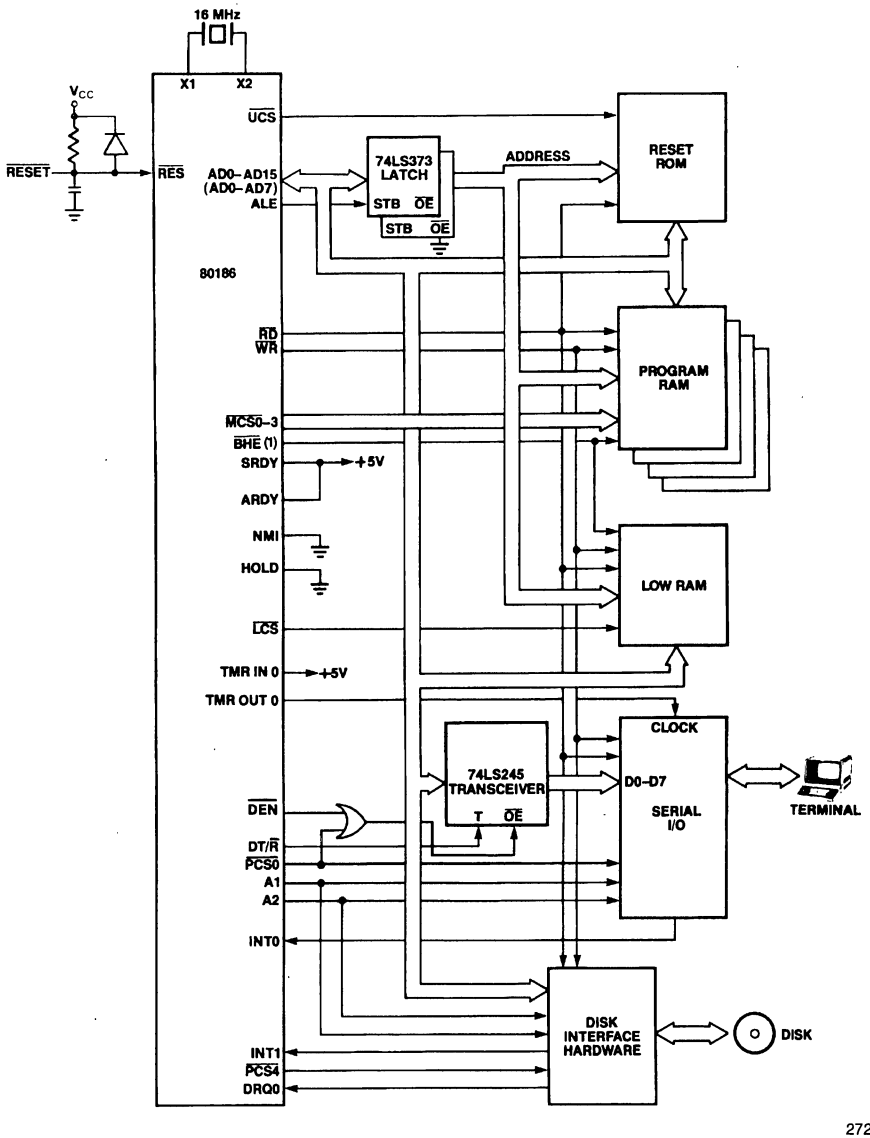
The processor can receive interrupts from a number of sources, both internal and external. The internal interrupt controller serves to merge these requests on a priority basis, for individual service by the CPU.

Internal interrupt sources (Timers and DMA channels) can be disabled by their own control registers or by mask bits within the interrupt controller. The interrupt controller has its own control register that sets the mode of operation for the controller.

INTERRUPT CONTROLLER AND RESET

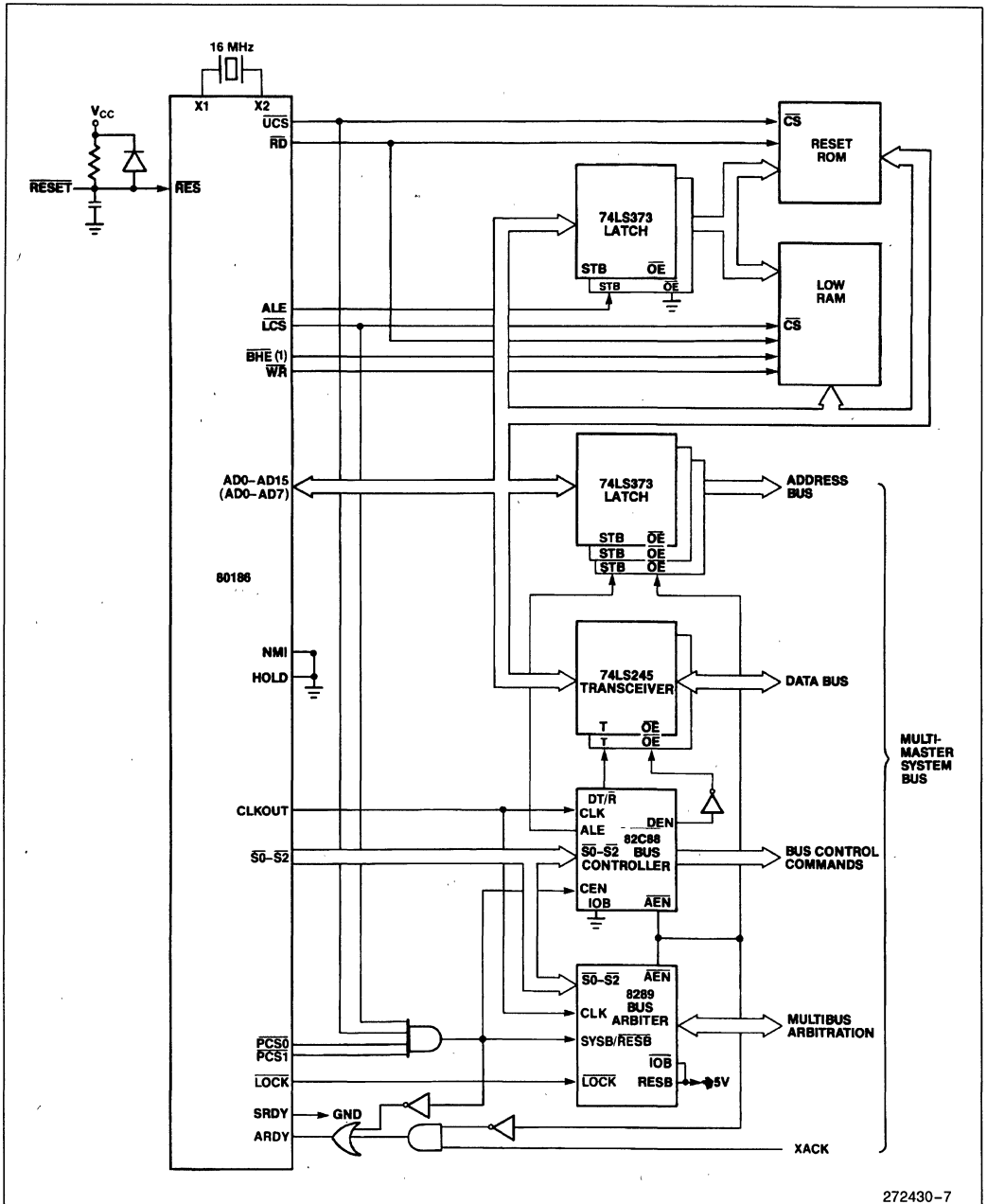
Upon RESET, the interrupt controller will perform the following actions:

- All SFNM bits reset to 0, implying Fully Nested Mode.
- All PR bits in the various control registers set to 1. This places all sources at lowest priority (level 111).
- All LTM bits reset to 0, resulting in edge-sense mode.
- All Interrupt Service bits reset to 0.
- All Interrupt Request bits reset to 0.
- All MSK (Interrupt Mask) bits set to 1 (mask).
- All C (Cascade) bits reset to 0 (non-Cascade).
- All PRM (Priority Mask) bits set to 1, implying no levels masked.
- Initialized to Master Mode.



NOTE:
Pin names in parenthesis apply to 80188.
(1) $\overline{BHE}$ does not exist on the 80188, this is only required for a 16-bit data bus.

Figure 6. Typical 80186/80188 Computer



NOTE:
Pin names in parentheses apply to 80188.
(1) $\overline{BHE}$ does not exist on the 80188, this is only required for a 16-bit data bus.

Figure 7. Typical 80186/80188 Multi-Master Bus 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.....−1.0V to +7V
Power Dissipation 3W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

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

Applicable to 8 MHz and 10 MHz devices.

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage (All except X1 and $\overline{RES}$)	2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage ($\overline{RES}$)	3.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_a = 2.5 mA for $\overline{S0}$–$\overline{S2}$ I_a = 2.0 mA for all other Outputs
V_{OH}	Output High Voltage	2.4		V	I_{oa} = −400 μA
I_{CC}	Power Supply Current		600*	mA	T_A = −40°C
			550	mA	T_A = 0°C
			415	mA	T_A = +70°C
I_{LI}	Input Leakage Current		±10	μA	0V < V_{IN} < V_{CC}
I_{LO}	Output Leakage Current		±10	μA	0.45V < V_{OUT} < V_{CC}
V_{CLO}	Clock Output Low		0.6	V	I_a = 4.0 mA
V_{CHO}	Clock Output High	4.0		V	I_{oa} = −200 μA
V_{CLI}	Clock Input Low Voltage	−0.5	0.6	V	
V_{CHI}	Clock Input High Voltage	3.9	V_{CC} + 1.0	V	
C_{IN}	Input Capacitance		10	pF	
C_{IO}	I/O Capacitance		20	pF	

*For extended temperature parts only.

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

Timing Requirements All Timings Measured At 1.5V Unless Otherwise Noted.

Symbol	Parameter	8 MHz		10 MHz		Units	Test Conditions
		Min	Max	Min	Max		
T_{DVCL}	Data in Setup (A/D)	20		15		ns	
T_{CLDX}	Data in Hold (A/D)	10		8		ns	
T_{ARYHCH}	Asynchronous Ready (ARDY) Active Setup Time(1)	20		15		ns	
T_{ARYLCL}	ARDY Inactive Setup Time	35		25		ns	
T_{CLARX}	ARDY Hold Time	15		15		ns	
T_{ARYCHL}	Asynchronous Ready Inactive Hold Time	15		15		ns	
T_{SRYCL}	Synchronous Ready (SRDY) Transition Setup Time(2)	20		20		ns	
T_{CLSRY}	SRDY Transition Hold Time(2)	15		15		ns	
T_{HVCL}	HOLD Setup(1)	25		20		ns	
T_{INVCH}	INTR, NMI, $\overline{TEST}$, TIM IN, Setup(1)	25		25		ns	
T_{INVCL}	DRQ0, DRQ1, Setup(1)	25		20		ns	

Master Interface Timing Responses

T_{CLAV}	Address Valid Delay	5	55	5	44	ns	C_L = 20 pF–200 pF all Outputs (Except T_{CLTMV}) @ 8 MHz and 10 MHz
T_{CLAX}	Address Hold	10		10		ns	
T_{CLAZ}	Address Float Delay	T_{CLAX}	35	T_{CLAX}	30	ns	
T_{CHCZ}	Command Lines Float Delay		45		40	ns	
T_{CHCV}	Command Lines Valid Delay (after Float)		55		45	ns	
T_{LHLL}	ALE Width	$T_{CLCL}-35$		$T_{CLCL}-30$		ns	
T_{CHLH}	ALE Active Delay		35		30	ns	
T_{CHLL}	ALE Inactive Delay		35		30	ns	
T_{LLAX}	Address Hold from ALE Inactive	$T_{CHCL}-25$		$T_{CHCL}-20$		ns	
T_{CLDV}	Data Valid Delay	10	44	10	40	ns	
T_{CLDOX}	Data Hold Time	10		10		ns	
T_{WHDX}	Data Hold after WR	$T_{CLCL}-40$		$T_{CLCL}-34$		ns	
T_{CVCTV}	Control Active Delay 1	5	50	5	40	ns	
T_{CHCTV}	Control Active Delay 2	10	55	10	44	ns	
T_{CVCTX}	Control Inactive Delay	5	55	5	44	ns	
T_{CVDEX}	$\overline{DEN}$ Inactive Delay (Non-Write Cycle)	10	70	10	56	ns	

1. To guarantee recognition at next clock.
2. To guarantee proper operation.

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

Master Interface Timing Responses (Continued)

Symbol	Parameter	8 MHz		10 MHz		Units	Test Conditions
		Min	Max	Min	Max		
T_{AZRL}	Address Float to $\overline{RD}$ Active	0		0		ns	
T_{CLRL}	$\overline{RD}$ Active Delay	10	70	10	56	ns	
T_{CLRH}	$\overline{RD}$ Inactive Delay	10	55	10	44	ns	
T_{RHAV}	$\overline{RD}$ Inactive to Address Active	$T_{CLCL}-40$		$T_{CLCL}-40$		ns	
T_{CLHAV}	HLDA Valid Delay	5	50	5	40	ns	
T_{RLRH}	$\overline{RD}$ Width	$2T_{CLCL}-50$		$2T_{CLCL}-46$		ns	
T_{WLWH}	$\overline{WR}$ Width	$2T_{CLCL}-40$		$2T_{CLCL}-34$		ns	
T_{AVLL}	Address Valid to ALE Low	$T_{CLCH}-25$		$T_{CLCH}-19$		ns	
T_{CHSV}	Status Active Delay	10	55	10	45	ns	
T_{CLSH}	Status Inactive Delay	10	65	10	50	ns	
T_{CLTMV}	Timer Output Delay		60		48	ns	100 pF max @ 8 & 10 MHz
T_{CLRO}	Reset Delay		60		48	ns	
T_{CHQSV}	Queue Status Delay		35		28	ns	
T_{CHDX}	Status Hold Time	10		10		ns	
T_{AVCH}	Address Valid to Clock High	10		10		ns	
T_{CLLV}	$\overline{LOCK}$ Valid/Invalid Delay	5	65	5	60	ns	
Chip-Select Timing Responses							
T_{CLCSV}	Chip-Select Active Delay		66		45	ns	
T_{CXCSX}	Chip-Select Hold from Command Inactive	35		35		ns	
T_{CHCSX}	Chip-Select Inactive Delay	5	35	5	32	ns	
CLKIN Requirements							
T_{CKIN}	CLKIN Period	62.5	250	50	250	ns	
T_{CKHL}	CLKIN Fall Time		10		10	ns	3.5 to 1.0V
T_{CKLH}	CLKIN Rise Time		10		10	ns	1.0 to 3.5V
T_{CLCK}	CLKIN Low Time	25		20		ns	1.5V
T_{CHCK}	CLKIN High Time	25		20		ns	1.5V
CLKOUT Timing (200 pF load)							
T_{CICO}	CLKIN to CLKOUT Skew		50		25	ns	
T_{CLCL}	CLKOUT Period	125	500	100	500	ns	
T_{CLCH}	CLKOUT Low Time	$\frac{1}{2}T_{CLCL}-7.5$		$\frac{1}{2}T_{CLCL}-6.0$		ns	1.5V
T_{CHCL}	CLKOUT High Time	$\frac{1}{2}T_{CLCL}-7.5$		$\frac{1}{2}T_{CLCL}-6.0$		ns	1.5V
T_{CH1CH2}	CLKOUT Rise Time		15		12	ns	1.0 to 3.5V
T_{CL2CL1}	CLKOUT Fall Time		15		12	ns	3.5 to 1.0V

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has from 5 to 7 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.

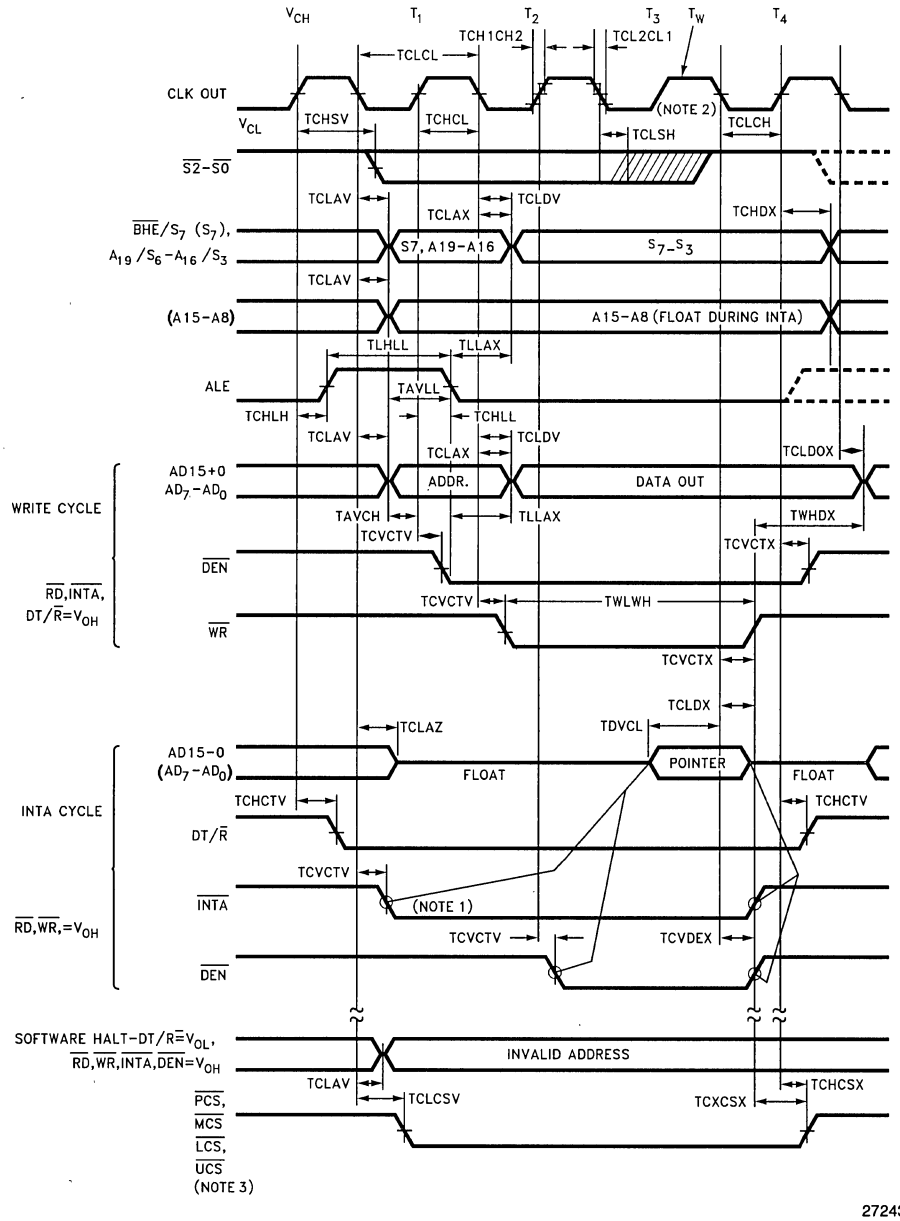
A: Address
ARY: Asynchronous Ready Input
C: Clock Output
CK: Clock Input
CS: Chip Select
CT: Control (DT/$\overline{R}$, $\overline{DEN}$, . . .)
D: Data Input
DE: $\overline{DEN}$
H: Logic Level High
IN: Input (DRQ0, TIM0, . . .)
L: Logic Level Low or ALE
O: Output
QS: Queue Status (QS1, QS2)
R: $\overline{RD}$ signal, $\overline{RESET}$ signal
S: Status ($\overline{S0}$, $\overline{S1}$, $\overline{S2}$)
SRY: Synchronous Ready Input
V: Valid
W: WR Signal
X: No Longer a Valid Logic Level
Z: Float

Examples:

T_{CLAV} — Time from Clock low to Address valid
T_{CHLH} — Time from Clock high to ALE high
T_{CLCSV} — Time from Clock low to Chip Select valid

WAVEFORMS

MAJOR CYCLE TIMING

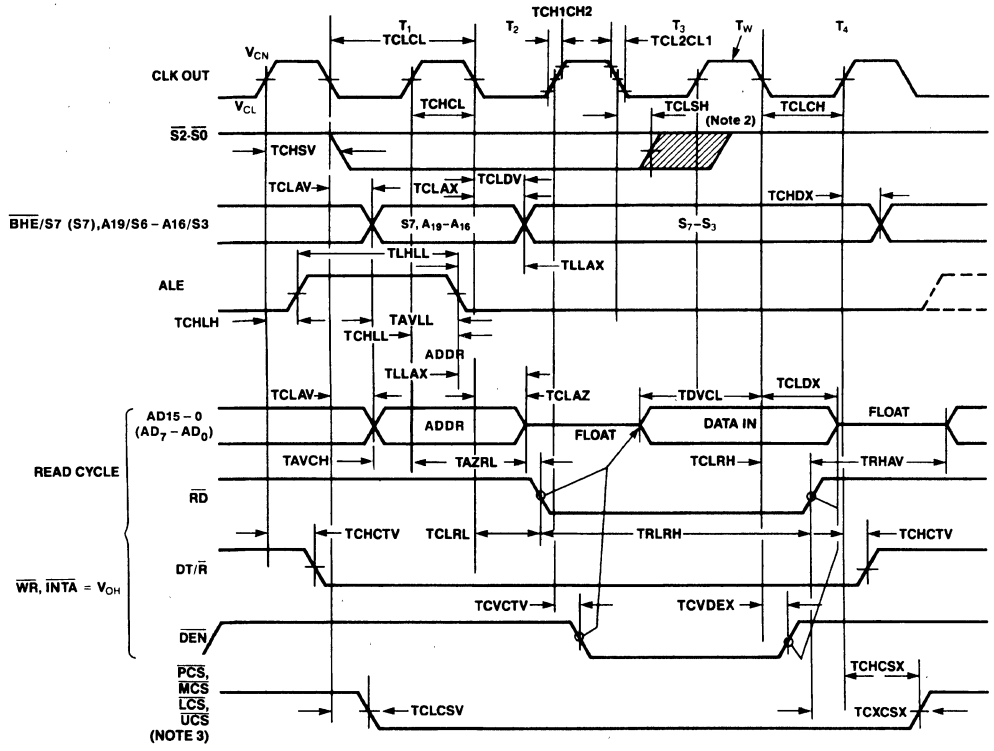


272430-8

NOTE:
Pin names in parentheses apply to the 80188.

WAVEFORMS (Continued)

MAJOR CYCLE TIMING (Continued)

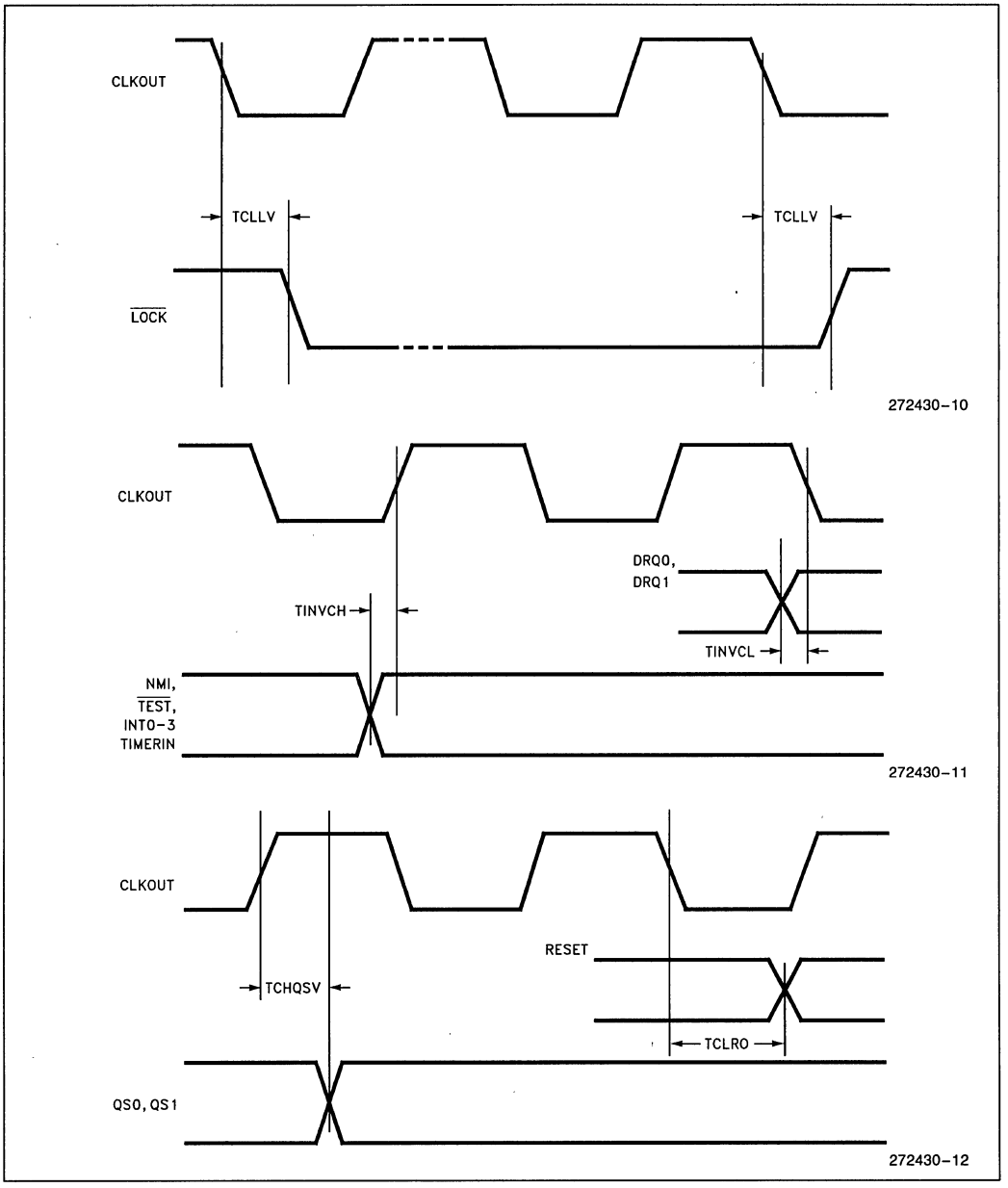


272430-9

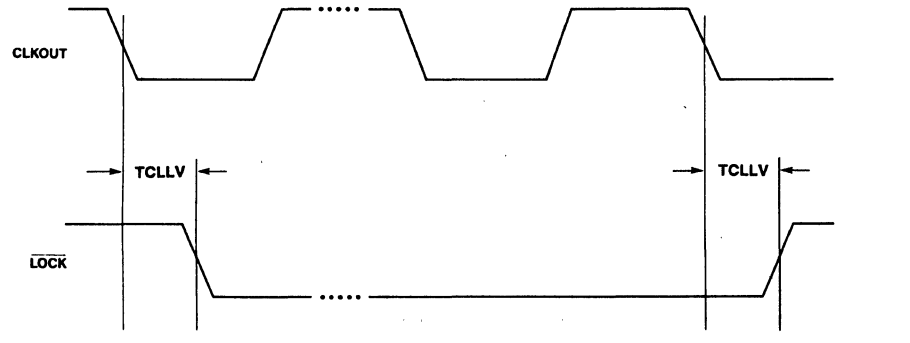
NOTES:

1. INTA occurs one clock later in slave mode.
2. Status inactive just prior to T_4.
3. If latched A1 and A2 are selected instead of $\overline{PCS5}$ and $\overline{PCS6}$, only T_{CLCSV} is applicable.
4. Pin names in parentheses apply to the 80188.

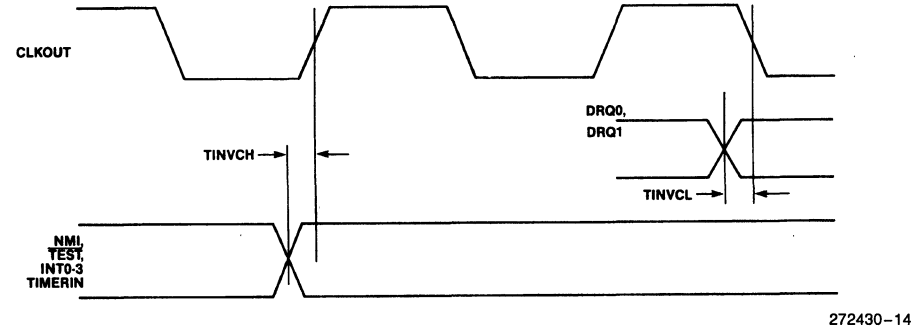
WAVEFORMS (Continued)



WAVEFORMS (Continued)



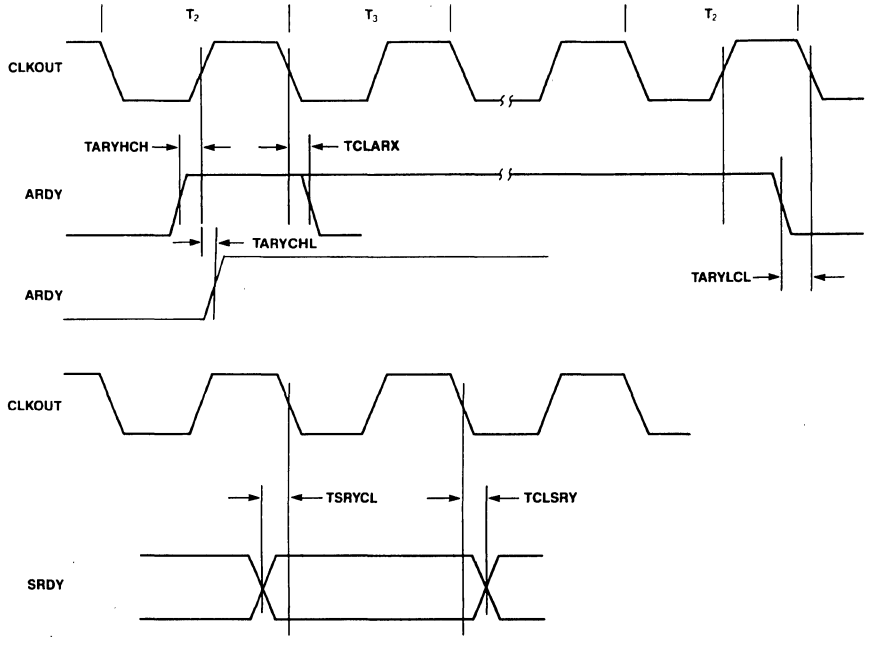
272430-13



272430-14

WAVEFORMS (Continued)

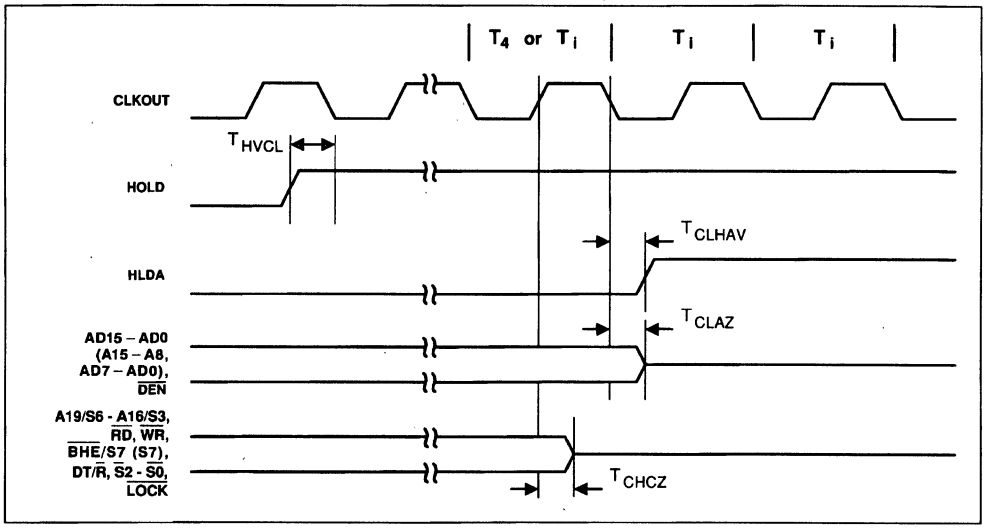
READY TIMING



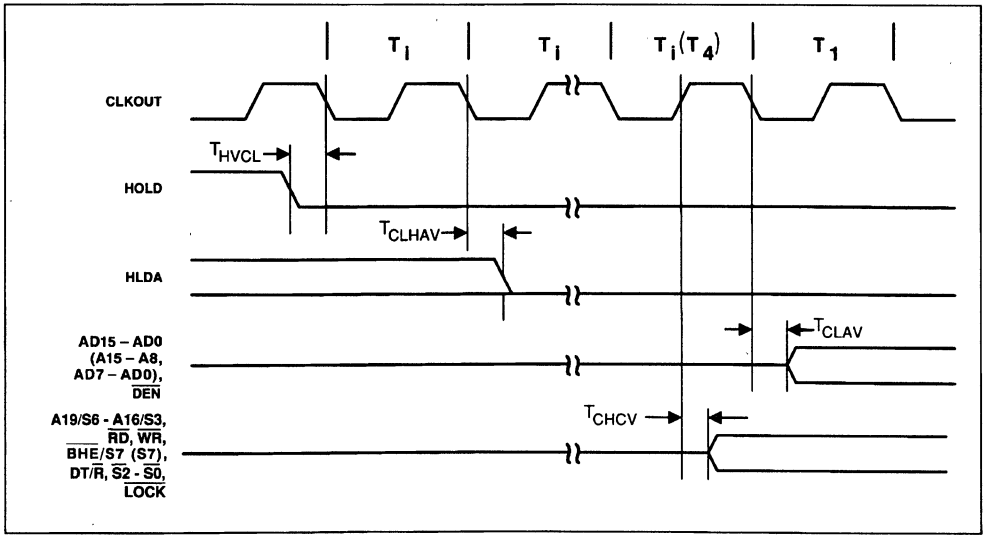
272430-15

WAVEFORMS (Continued)

HOLD/HLDA TIMING (Entering Hold)



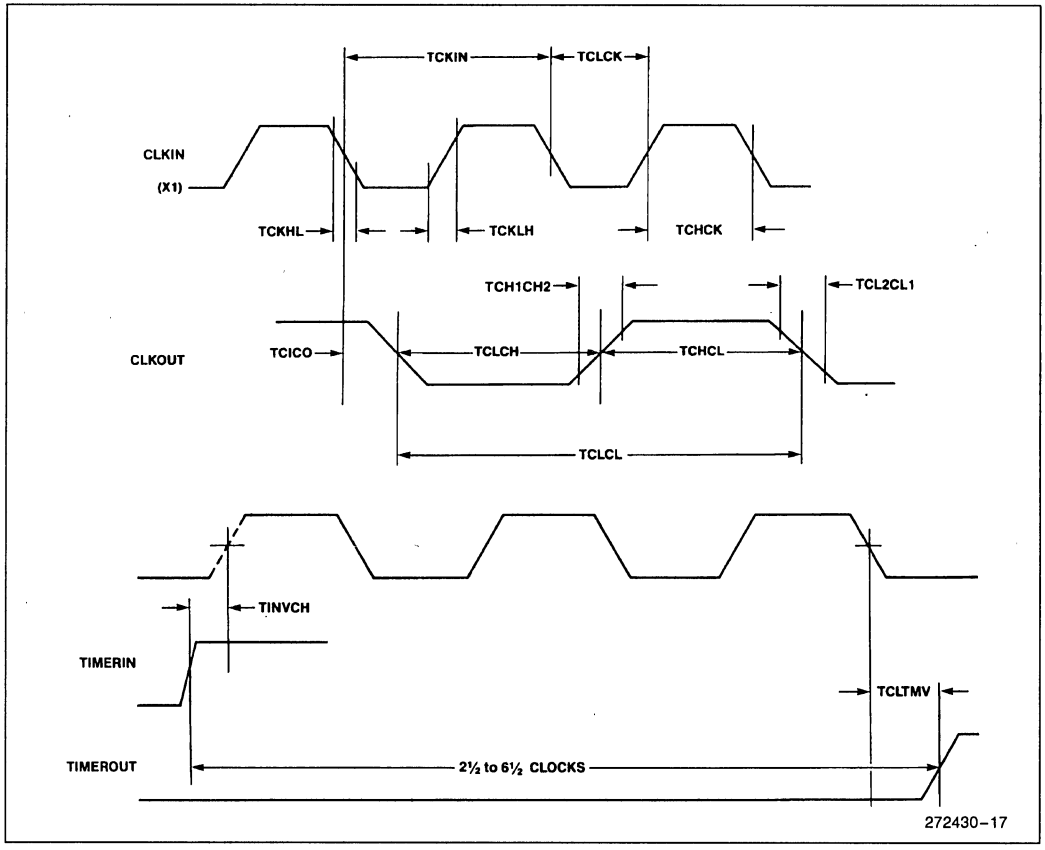
HOLD/HLDA TIMING (Leaving Hold)



272430-16

NOTE:
Pin names in parentheses apply to the 80188.

WAVEFORMS (Continued)



EXPRESS

The Intel EXPRESS system offers enhancements to the operational specifications of the microprocessor. EXPRESS products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

The EXPRESS program includes the commercial standard temperature range with burn-in and an extended temperature range without burn-in.

With the commercial standard temperature range operational characteristics are guaranteed over the temperature range of 0°C to +70°C. With the extended temperature range option, operational characteristics are guaranteed over the range of −40°C to +85°C.

The optional burn-in is dynamic, for a minimum time of 160 hours at +125°C with V_{CC} = 5.5V ±0.25V, following guidelines in MIL-STD-883, Method 1015.

Package types and EXPRESS versions are identified by a one- or two-letter prefix to the part number. The prefixes are listed in Table 2. All A.C. and D.C. specifications not mentioned in this section are the same for both commercial and EXPRESS parts.

Table 2. Prefix Identification

Prefix	Package Type	Temperature Range	Burn-In
A	PGA	Commercial	No
N	PLCC	Commercial	No
R	LCC	Commercial	No
TA	PGA	Extended	No
QA	PGA	Commercial	Yes
QR	LCC	Commercial	Yes

NOTE:
Not all package/temperature range/speed combinations are available.

EXECUTION TIMINGS

A determination of program execution timing must consider the bus cycles necessary to prefetch instructions as well as the number of execution unit cycles necessary to execute instructions. The following instruction timings represent the minimum execution time in clock cycles for each instruction. The timings given are based on the following assumptions:

- The opcode, along with any data or displacement required for execution of a particular instruction, has been prefetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDS occur.
- All word-data is located on even-address boundaries.

All instructions which involve memory accesses can also require one or two additional clocks above the minimum timings shown due to the asynchronous handshake between the bus interface unit (BIU) and execution unit.

All jumps and calls include the time required to fetch the opcode of the next instruction at the destination address.

The 80186 has sufficient bus performance to ensure that an adequate number of prefetched bytes will reside in the queue (6 bytes) most of the time. Therefore, actual program execution time will not be substantially greater than that derived from adding the instruction timings shown.

The 80188 is noticeably limited in its performance relative to the execution unit. A sufficient number of prefetched bytes may not reside in the prefetch queue (4 bytes) much of the time. Therefore, actual program execution time may be substantially greater than that derived from adding the instruction timings shown.

INSTRUCTION SET SUMMARY

Function	Format				80186 Clock Cycles	80188 Clock Cycles	Comments
DATA TRANSFER							
MOV = Move:							
Register to Register/Memory	1000100w	mod reg r/m			2/12	2/12*	
Register/memory to register	1000101w	mod reg r/m			2/9	2/9*	
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1	12/13	12/13	8/16-bit
Immediate to register	1011w reg	data	data if w=1		3/4	3/4	8/16-bit
Memory to accumulator	1010000w	addr-low	addr-high		8	8*	
Accumulator to memory	1010001w	addr-low	addr-high		9	9*	
Register/memory to segment register	10001110	mod 0 reg r/m			2/9	2/13	
Segment register to register/memory	10001100	mod 0 reg r/m			2/11	2/15	
PUSH = Push:							
Memory	11111111	mod 1 1 0 r/m			16	20	
Register	01010 reg				10	14	
Segment register	000reg110				9	13	
Immediate	011010s0	data	data if s=0		10	14	
PUSHA = Push All	01100000				36	68	
POP = Pop:							
Memory	10001111	mod 0 0 0 r/m			20	24	
Register	01011 reg				10	14	
Segment register	000reg111	(reg≠01)			8	12	
POPA = Pop All	01100001				51	83	
XCHG = Exchange:							
Register/memory with register	1000011w	mod reg r/m			4/17	4/17*	
Register with accumulator	10010 reg				3	3	
IN = Input from:							
Fixed port	1110010w	port			10	10*	
Variable port	1110110w				8	8*	
OUT = Output to:							
Fixed port	1110011w	port			9	9*	
Variable port	1110111w				7	7*	
XLAT = Translate byte to AL	11010111				11	15	
LEA = Load EA to register	10001101	mod reg r/m			6	6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod≠11)		18	26	
LES = Load pointer to ES	11000100	mod reg r/m	(mod≠11)		18	26	
LAHF = Load AH with flags	10011111				2	2	
SAHF = Store AH into flags	10011110				3	3	
PUSHF = Push flags	10011100				9	13	
POPF = Pop flags	10011101				8	12	

Shaded areas indicate instructions not available in 8086, 8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for each memory transfer.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80186 Clock Cycles	80188 Clock Cycles	Comments
DATA TRANSFER (Continued)				
SEGMENT = Segment Override:				
CS	00101110	2	2	
SS	00110110	2	2	
DS	00111110	2	2	
ES	00100110	2	2	
ARITHMETIC				
ADD = Add:				
Reg/memory with register to either	000000dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	100000sw | mod 000 r/m | data | data if s w=01	4/16	4/16*	
Immediate to accumulator	0000010w | data | data if w=1	3/4	3/4	8/16-bit
ADC = Add with carry:				
Reg/memory with register to either	000100dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	100000sw | mod 010 r/m | data | data if s w=01	4/16	4/16*	
Immediate to accumulator	0001010w | data | data if w=1	3/4	3/4	8/16-bit
INC = Increment:				
Register/memory	1111111w | mod 000 r/m	3/15	3/15*	
Register	01000 reg	3	3	
SUB = Subtract:				
Reg/memory and register to either	001010dw | mod reg r/m	3/10	3/10*	
Immediate from register/memory	100000sw | mod 101 r/m | data | data if s w=01	4/16	4/16*	
Immediate from accumulator	0010110w | data | data if w=1	3/4	3/4	8/16-bit
SBB = Subtract with borrow:				
Reg/memory and register to either	000110dw | mod reg r/m	3/10	3/10*	
Immediate from register/memory	100000sw | mod 011 r/m | data | data if s w=01	4/16	4/16*	
Immediate from accumulator	0001110w | data | data if w=1	3/4	3/4	8/16-bit
DEC = Decrement				
Register/memory	1111111w | mod 001 r/m	3/15	3/15*	
Register	01001 reg	3	3	
CMP = Compare:				
Register/memory with register	0011101w | mod reg r/m	3/10	3/10*	
Register with register/memory	0011100w | mod reg r/m	3/10	3/10*	
Immediate with register/memory	100000sw | mod 111 r/m | data | data if s w=01	3/10	3/10*	
Immediate with accumulator	0011110w | data | data if w=1	3/4	3/4	8/16-bit
NEG = Change sign register/memory	1111011w | mod 011 r/m	3/10	3/10*	
AAA = ASCII adjust for add	00110111	8	8	
DAA = Decimal adjust for add	00100111	4	4	
AAS = ASCII adjust for subtract	00111111	7	7	
DAS = Decimal adjust for subtract	00101111	4	4	
MUL = Multiply (unsigned):	1111011w | mod 100 r/m			
Register-Byte		26–28	26–28	
Register-Word		35–37	35–37	
Memory-Byte		32–34	32–34	
Memory-Word		41–43	41–43*	

Shaded areas indicate instructions not available in 8086, 8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for each memory transfer.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80186 Clock Cycles	80188 Clock Cycles	Comments
ARITHMETIC (Continued)				
IMUL = Integer multiply (signed):	1 1 1 1 0 1 1 w | mod 1 0 1 r/m			
Register-Byte		25–28	25–28	
Register-Word		34–37	34–37	
Memory-Byte		31–34	31–34	
Memory-Word		40–43	40–43*	
IMUL = Integer immediate multiply (signed)	0 1 1 0 1 0 s 1 | mod reg r/m | data | data if s=0	22–25/ 29–32	22–25/ 29–32	
DIV = Divide (unsigned):	1 1 1 1 0 1 1 w | mod 1 1 0 r/m			
Register-Byte		29	29	
Register-Word		38	38	
Memory-Byte		35	35	
Memory-Word		44	44*	
IDIV = Integer divide (signed):	1 1 1 1 0 1 1 w | mod 1 1 1 r/m			
Register-Byte		44–52	44–52	
Register-Word		53–61	53–61	
Memory-Byte		50–58	50–58	
Memory-Word		59–67	59–67*	
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0 | 0 0 0 0 1 0 1 0	19	19	
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1 | 0 0 0 0 1 0 1 0	15	15	
CBW = Convert byte to word	1 0 0 1 1 0 0 0	2	2	
CWD = Convert word to double word	1 0 0 1 1 0 0 1	4	4	
LOGIC				
Shift/Rotate Instructions:				
Register/Memory by 1	1 1 0 1 0 0 0 w | mod TTT r/m	2/15	2/15	
Register/Memory by CL	1 1 0 1 0 0 1 w | mod TTT r/m	5+n/17+n	5+n/17+n	
Register/Memory by Count	1 1 0 0 0 0 0 w | mod TTT r/m | count	5+n/17+n	5+n/17+n	
	TTT Instruction 0 0 0 ROL 0 0 1 ROR 0 1 0 RCL 0 1 1 RCR 1 0 0 SHL/SAL 1 0 1 SHR 1 1 1 SAR			
AND = **And:**				
Reg/memory and register to either	0 0 1 0 0 0 d w | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1 0 0 0 0 0 0 w | mod 1 0 0 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0 0 1 0 0 1 0 w | data | data if w=1	3/4	3/4	8/16-bit
TEST = And function to flags, no result:				
Register/memory and register	1 0 0 0 0 1 0 w | mod reg r/m	3/10	3/10*	
Immediate data and register/memory	1 1 1 1 0 1 1 w | mod 0 0 0 r/m | data | data if w=1	4/10	4/10*	
Immediate data and accumulator	1 0 1 0 1 0 0 w | data | data if w=1	3/4	3/4	8/16-bit
OR = Or:				
Reg/memory and register to either	0 0 0 0 1 0 d w | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1 0 0 0 0 0 0 w | mod 0 0 1 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0 0 0 0 1 1 0 w | data | data if w=1	3/4	3/4	8/16-bit

Shaded areas indicate instructions not available in 8086, 8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for each memory transfer.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80186 Clock Cycles	80188 Clock Cycles	Comments
LOGIC (Continued)				
XOR = **Exclusive or:**				
Reg/memory and register to either	001100dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod 110 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0011010w | data | data if w=1	3/4	3/4	8/16-bit
NOT = Invert register/memory	1111011w | mod 010 r/m	3/10	3/10*	
STRING MANIPULATION				
MOVS = Move byte/word	1010010w	14	14*	
CMPS = Compare byte/word	1010011w	22	22*	
SCAS = Scan byte/word	1010111w	15	15*	
LODS = Load byte/wd to AL/AX	1010110w	12	12*	
STOS = Store byte/wd from AL/AX	1010101w	10	10*	
INS = Input byte/wd from DX port	0110110w	14	14	
OUTS = Output byte/wd to DX port	0110111w	14	14	
Repeated by count in CX (REP/REPE/REPZ/REPNE/REPNZ)				
MOVS = Move string	11110010 | 1010010w	8+8n	8+8n*	
CMPS = Compare string	1111001z | 1010011w	5+22n	5+22n*	
SCAS = Scan string	1111001z | 1010111w	5+15n	5+15n*	
LODS = Load string	11110010 | 1010110w	6+11n	6+11n*	
STOS = Store string	11110010 | 1010101w	6+9n	6+9n*	
INS = Input string	11110010 | 0110110w	8+8n	8+8n*	
OUTS = Output string	11110010 | 0110111w	8+8n	8+8n*	
CONTROL TRANSFER				
CALL = **Call:**				
Direct within segment	11101000 | disp-low | disp-high	15	19	
Register/memory indirect within segment	11111111 | mod 010 r/m	13/19	17/27	
Direct intersegment	10011010 | segment offset | segment selector	23	31	
Indirect intersegment	11111111 | mod 011 r/m | (mod ≠ 11)	38	54	
JMP = **Unconditional jump:**				
Short/long	11101011 | disp-low	14	14	
Direct within segment	11101001 | disp-low | disp-high	14	14	
Register/memory indirect within segment	11111111 | mod 100 r/m	11/17	11/21	
Direct intersegment	11101010 | segment offset | segment selector	14	14	
Indirect intersegment	11111111 | mod 101 r/m | (mod ≠ 11)	26	34	

Shaded areas indicate instructions not available in 8086, 8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for each memory transfer.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80186 Clock Cycles	80188 Clock Cycles	Comments
CONTROL TRANSFER (Continued)				
RET = Return from CALL:				
Within segment	11000011	16	20	
Within seg adding immed to SP	11000010 | data-low | data-high	18	22	
Intersegment	11001011	22	30	
Intersegment adding immediate to SP	11001010 | data-low | data-high	25	33	
JE/JZ = Jump on equal/zero	01110100 | disp	4/13	4/13	JMP not taken/JMP taken
JL/JNGE = Jump on less/not greater or equal	01111100 | disp	4/13	4/13	
JLE/JNG = Jump on less or equal/not greater	01111110 | disp	4/13	4/13	
JB/JNAE = Jump on below/not above or equal	01110010 | disp	4/13	4/13	
JBE/JNA = Jump on below or equal/not above	01110110 | disp	4/13	4/13	
JP/JPE = Jump on parity/parity even	01111010 | disp	4/13	4/13	
JO = Jump on overflow	01110000 | disp	4/13	4/13	
JS = Jump on sign	01111000 | disp	4/13	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101 | disp	4/13	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101 | disp	4/13	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111 | disp	4/13	4/13	
JNB/JAE = Jump on not below/above or equal	01110011 | disp	4/13	4/13	
JNBE/JA = Jump on not below or equal/above	01110111 | disp	4/13	4/13	
JNP/JPO = Jump on not par/par odd	01111011 | disp	4/13	4/13	
JNO = Jump on not overflow	01110001 | disp	4/13	4/13	
JNS = Jump on not sign	01111001 | disp	4/13	4/13	
JCXZ = Jump on CX zero	11100011 | disp	5/15	5/15	
LOOP = Loop CX times	11100010 | disp	6/16	6/16	LOOP not taken/LOOP taken
LOOPZ/LOOPE = Loop while zero/equal	11100001 | disp	6/16	6/16	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000 | disp	6/16	6/16	
ENTER = Enter Procedure	11001000 | data-low | data-high | L			
L = 0		15	19	
L = 1		25	29	
L > 1		22+16(n−1)	26+20(n−1)	
LEAVE = Leave Procedure	11001001	8	8	
INT = Interrupt:				
Type specified	11001101 | type	47	47	
Type 3	11001100	45	45	if INT. taken/ if INT. not taken
INTO = Interrupt on overflow	11001110	48/4	48/4	
IRET = Interrupt return	11001111	28	28	
BOUND = Detect value out of range	01100010 | mod reg r/m	33–35	33–35	

Shaded areas indicate instructions not available in 8086, 8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for each memory transfer.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80186 Clock Cycles	80188 Clock Cycles	Comments
PROCESSOR CONTROL				
CLC = Clear carry	11111000	2	2	
CMC = Complement carry	11110101	2	2	
STC = Set carry	11111001	2	2	
CLD = Clear direction	11111100	2	2	
STD = Set direction	11111101	2	2	
CLI = Clear interrupt	11111010	2	2	
STI = Set interrupt	11111011	2	2	
HLT = Halt	11110100	2	2	
WAIT = Wait	10011011	6	6	if $\overline{TEST}$ = 0
LOCK = Bus lock prefix	11110000	2	3	
ESC = Processor Extension Escape	11011TTT mod LLL r/m (TTT LLL are opcode to processor extension)	6	6	
NOP = No Operation	10010000	3	3	

Shaded areas indicate instructions not available in 8086, 8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for each memory transfer.

FOOTNOTES

The Effective Address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

EA calculation time is 4 clock cycles for all modes, and is included in the execution times given whenever appropriate.

Segment Override Prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

REVISION HISTORY

This data sheet replaces the following data sheets:

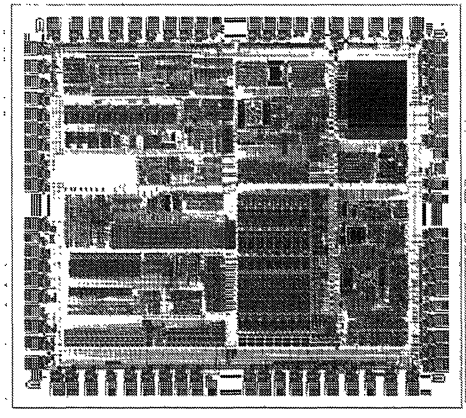
210706-011 80188

210451-011 80186

80C186XL/80C188XL
16-BIT HIGH-INTEGRATION EMBEDDED PROCESSORS

- Low Power, Fully Static Versions of 80C186/80C188
- Operation Modes:
  - Enhanced Mode
    - DRAM Refresh Control Unit
    - Power-Save Mode
    - Direct Interface to 80C187 (80C186XL Only)
  - Compatible Mode
    - NMOS 80186/80188 Pin-for-Pin Replacement for Non-Numerics Applications
- Integrated Feature Set
  - Static, Modular CPU
  - Clock Generator
  - 2 Independent DMA Channels
  - Programmable Interrupt Controller
  - 3 Programmable 16-Bit Timers
  - Dynamic RAM Refresh Control Unit
  - Programmable Memory and Peripheral Chip Select Logic
  - Programmable Wait State Generator
  - Local Bus Controller
  - Power-Save Mode
  - System-Level Testing Support (High Impedance Test Mode)
- Completely Object Code Compatible with Existing 8086/8088 Software and Has 10 Additional Instructions over 8086/8088
- Speed Versions Available
  - 25 MHz (80C186XL25/80C188XL25)
  - 20 MHz (80C186XL20/80C188XL20)
  - 12 MHz (80C186XL12/80C188XL12)
- Direct Addressing Capability to 1 MByte Memory and 64 Kbyte I/O
- Available in 68-Pin:
  - Plastic Leaded Chip Carrier (PLCC)
  - Ceramic Pin Grid Array (PGA)
  - Ceramic Leadless Chip Carrier (JEDEC A Package)
- Available in 80-Pin:
  - Quad Flat Pack (EIAJ)
  - Shrink Quad Flat Pack (SGFP)
- Available in Extended Temperature Range (−40°C to +85°C)

The Intel 80C186XL is a Modular Core re-implementation of the 80C186 microprocessor. It offers higher speed and lower power consumption than the standard 80C186 but maintains 100% clock-for-clock functional compatibility. Packaging and pinout are also identical.



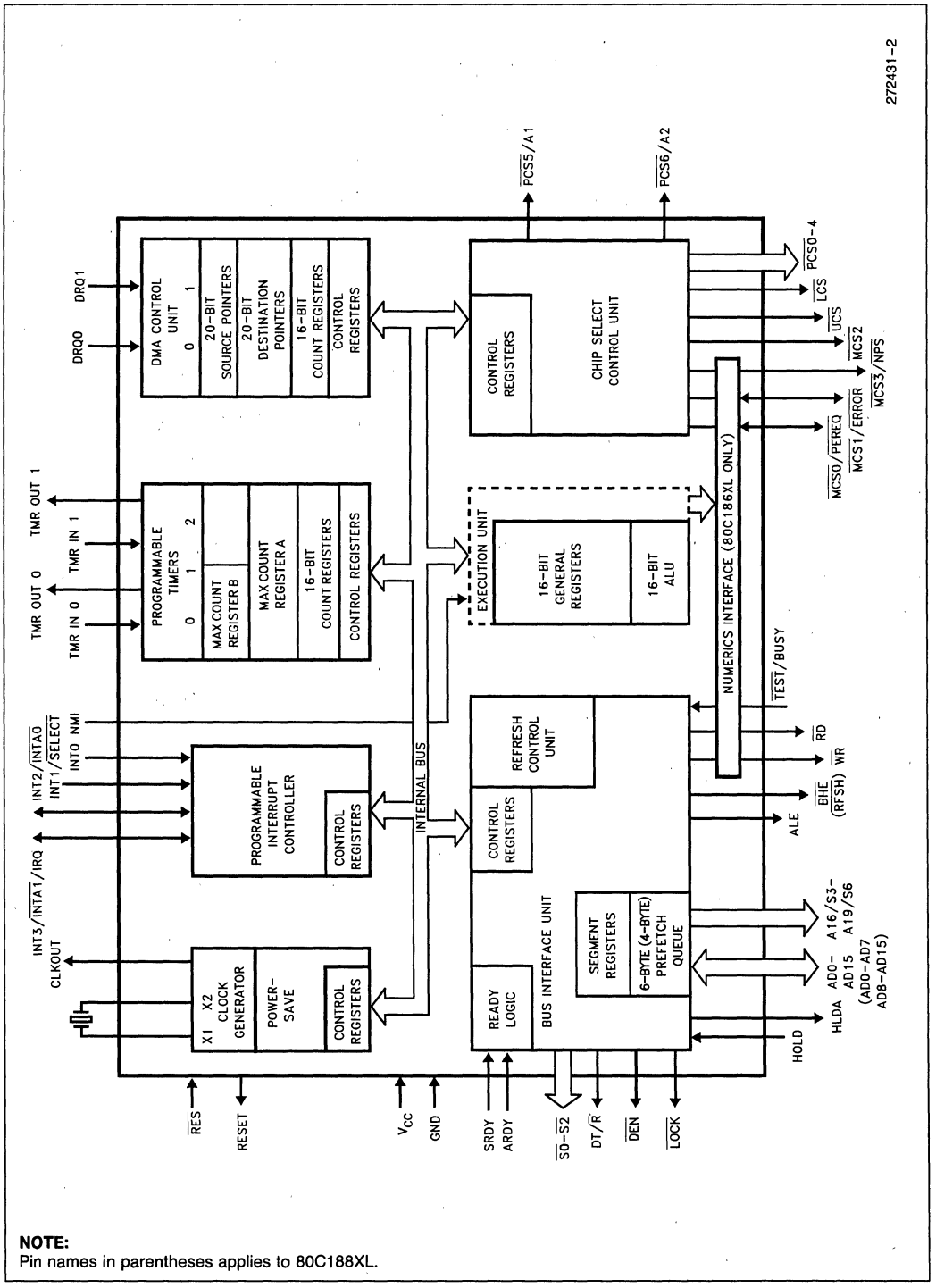
272431-1

October 1995
Order Number: 272431-004

80C186XL/80C188XL
16-Bit High-Integration Embedded Processors

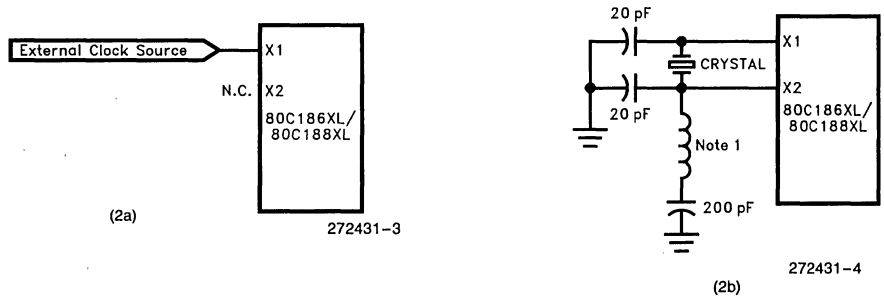
CONTENTS

CONTENTS



NOTE:
Pin names in parentheses applies to 80C188XL.

Figure 1. 80C186XL/80C188XL Block Diagram



Note 1:

XTAL Frequency	L1 Value
20 MHz	12.0 μH ±20%
25 MHz	8.2 μH ±20%
32 MHz	4.7 μH ±20%
40 MHz	3.0 μH ±20%

LC network is only required when using a third overtone crystal.

Figure 2. Oscillator Configurations (see text)

INTRODUCTION

Unless specifically noted, all references to the 80C186XL apply to the 80C188XL. References to pins that differ between the 80C186XL and the 80C188XL are given in parentheses.

The following Functional Description describes the base architecture of the 80C186XL. The 80C186XL is a very high integration 16-bit microprocessor. It combines 15–20 of the most common microprocessor system components onto one chip. The 80C186XL is object code compatible with the 8086/8088 microprocessors and adds 10 new instruction types to the 8086/8088 instruction set.

The 80C186XL has two major modes of operation, Compatible and Enhanced. In Compatible Mode the 80C186XL is completely compatible with NMOS 80186, with the exception of 8087 support. The Enhanced mode adds three new features to the system design. These are Power-Save control, Dynamic RAM refresh, and an asynchronous Numerics Coprocessor interface (80C186XL only).

80C186XL CORE ARCHITECTURE

80C186XL Clock Generator

The 80C186XL provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter, synchronous and asynchronous ready inputs, and reset circuitry.

The 80C186XL oscillator circuit is designed to be used either with a parallel resonant fundamental or third-overtone mode crystal, depending upon the frequency range of the application. This is used as the time base for the 80C186XL.

The output of the oscillator is not directly available outside the 80C186XL. The recommended crystal configuration is shown in Figure 2b. When used in third-overtone mode, the tank circuit is recommended for stable operation. Alternately, the oscillator may be driven from an external source as shown in Figure 2a.

The crystal or clock frequency chosen must be twice the required processor operating frequency due to the internal divide by two counter. This counter is used to drive all internal phase clocks and the external CLKOUT signal. CLKOUT is a 50% duty cycle processor clock and can be used to drive other system components. All AC Timings are referenced to CLKOUT.

Intel recommends the following values for crystal selection parameters.

Temperature Range:	Application Specific
ESR (Equivalent Series Resistance):	60Ω max
C_0 (Shunt Capacitance of Crystal):	7.0 pF max
C_1 (Load Capacitance):	20 pF ±2 pF
Drive Level:	2 mW max

Bus Interface Unit

The 80C186XL provides a local bus controller to generate the local bus control signals. In addition, it employs a HOLD/HLDA protocol for relinquishing the local bus to other bus masters. It also provides outputs that can be used to enable external buffers and to direct the flow of data on and off the local bus.

The bus controller is responsible for generating 20 bits of address, read and write strobes, bus cycle status information and data (for write operations) information. It is also responsible for reading data from the local bus during a read operation. Synchronous and asynchronous ready input pins are provided to extend a bus cycle beyond the minimum four states (clocks).

The 80C186XL bus controller also generates two control signals ($\overline{DEN}$ and DT/$\overline{R}$) when interfacing to external transceiver chips. This capability allows the addition of transceivers for simple buffering of the multiplexed address/data bus.

During RESET the local bus controller will perform the following action:

- Drive $\overline{DEN}$, $\overline{RD}$ and $\overline{WR}$ HIGH for one clock cycle, then float them.
- Drive $\overline{S0}$–$\overline{S2}$ to the inactive state (all HIGH) and then float.
- Drive $\overline{LOCK}$ HIGH and then float.
- Float AD0–15 (AD0–8), A16–19 (A9–A19), $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$.
- Drive ALE LOW
- Drive HLDA LOW.

$\overline{RD}$/$\overline{QSMD}$, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$/PEREQ, $\overline{MCS1}$/$\overline{ERROR}$ and $\overline{TEST}$/BUSY pins have internal pullup devices which are active while $\overline{RES}$ is applied. Excessive loading or grounding certain of these pins causes the 80C186XL to enter an alternative mode of operation:

- $\overline{RD}$/$\overline{QSMD}$ low results in Queue Status Mode.
- $\overline{UCS}$ and $\overline{LCS}$ low results in ONCE Mode.
- $\overline{TEST}$/BUSY low (and high later) results in Enhanced Mode.

80C186XL PERIPHERAL ARCHITECTURE

All the 80C186XL integrated peripherals are controlled by 16-bit registers contained within an internal 256-byte control block. The control block may be mapped into either memory or I/O space. Internal logic will recognize control block addresses and respond to bus cycles. An offset map of the 256-byte control register block is shown in Figure 3.

Chip-Select/Ready Generation Logic

The 80C186XL contains logic which provides programmable chip-select generation for both memories and peripherals. In addition, it can be programmed to provide READY (or WAIT state) generation. It can also provide latched address bits A1 and A2. The chip-select lines are active for all memory and I/O cycles in their programmed areas, whether they be generated by the CPU or by the integrated DMA unit.

The 80C186XL provides 6 memory chip select outputs for 3 address areas; upper memory, lower memory, and midrange memory. One each is provided for upper memory and lower memory, while four are provided for midrange memory.

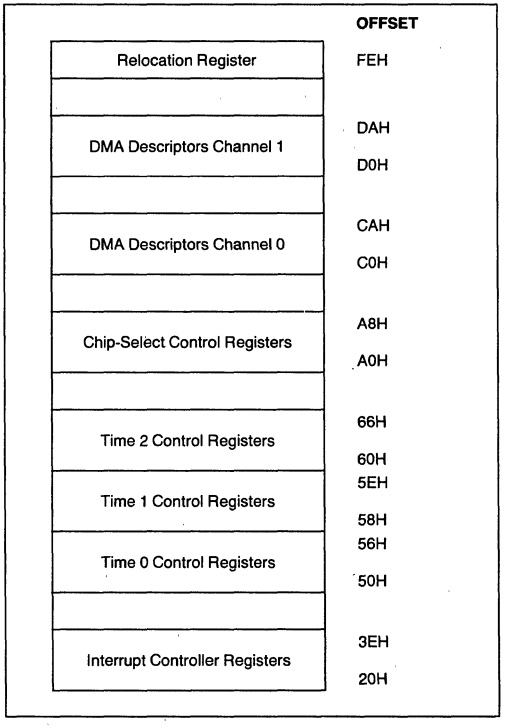


Figure 3. Internal Register Map

The 80C186XL provides a chip select, called $\overline{UCS}$, for the top of memory. The top of memory is usually used as the system memory because after reset the 80C186XL begins executing at memory location FFFF0H.

The 80C186XL provides a chip select for low memory called $\overline{LCS}$. The bottom of memory contains the interrupt vector table, starting at location 00000H.

The 80C186XL provides four $\overline{MCS}$ lines which are active within a user-locatable memory block. This block can be located within the 80C186XL 1 Mbyte memory address space exclusive of the areas defined by $\overline{UCS}$ and $\overline{LCS}$. Both the base address and size of this memory block are programmable.

The 80C186XL can generate chip selects for up to seven peripheral devices. These chip selects are active for seven contiguous blocks of 128 bytes above a programmable base address. The base address may be located in either memory or I/O space.

The 80C186XL can generate a READY signal internally for each of the memory or peripheral $\overline{CS}$ lines. The number of WAIT states to be inserted for each peripheral or memory is programmable to provide 0–3 wait states for all accesses to the area for which the chip select is active. In addition, the 80C186XL may be programmed to either ignore external READY for each chip-select range individually or to factor external READY with the integrated ready generator.

Upon RESET, the Chip-Select/Ready Logic will perform the following actions:

- All chip-select outputs will be driven HIGH.
- Upon leaving RESET, the $\overline{UCS}$ line will be programmed to provide chip selects to a 1K block with the accompanying READY control bits set at 011 to insert 3 wait states in conjunction with external READY (i.e., UMCS resets to FFFBH).
- No other chip select or READY control registers have any predefined values after RESET. They will not become active until the CPU accesses their control registers.

DMA Unit

The 80C186XL DMA controller provides two independent high-speed DMA channels. Data transfers can occur between memory and I/O spaces (e.g., Memory to I/O) or within the same space (e.g., Memory to Memory or I/O to I/O). Data can be transferred either in bytes (8 bits) or in words (16 bits) to or from even or odd addresses.

NOTE:
Only byte transfers are possible on the 80C188XL.

Each DMA channel maintains both a 20-bit source and destination pointer which can be optionally incremented or decremented after each data transfer (by one or two depending on byte or word transfers). Each data transfer consumes 2 bus cycles (a minimum of 8 clocks), one cycle to fetch data and the other to store data.

Timer/Counter Unit

The 80C186XL provides three internal 16-bit programmable timers. Two of these are highly flexible and are connected to four external pins (2 per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real-time coding and time delay applications. In addition, the third timer can be used as a prescaler to the other two, or as a DMA request source.

Interrupt Control Unit

The 80C186XL can receive interrupts from a number of sources, both internal and external. The 80C186XL has 5 external and 2 internal interrupt sources (Timer/Couners and DMA). The internal interrupt controller serves to merge these requests on a priority basis, for individual service by the CPU.

Enhanced Mode Operation

In Compatible Mode the 80C186XL operates with all the features of the NMOS 80186, with the exception of 8087 support (i.e. no math coprocessing is possible in Compatible Mode). Queue-Status information is still available for design purposes other than 8087 support.

All the Enhanced Mode features are completely masked when in Compatible Mode. A write to any of the Enhanced Mode registers will have no effect, while a read will not return any valid data.

In Enhanced Mode, the 80C186XL will operate with Power-Save, DRAM refresh, and numerics coprocessor support (80C186XL only) in addition to all the Compatible Mode features.

If connected to a math coprocessor (80C186XL only), this mode will be invoked automatically. Without an NPX, this mode can be entered by tying the RESET output signal from the 80C186XL to the $\overline{TEST}$/BUSY input.

Queue-Status Mode

The queue-status mode is entered by strapping the $\overline{RD}$ pin low. $\overline{RD}$ is sampled at RESET and if LOW, the 80C186XL will reconfigure the ALE and $\overline{WR}$ pins to be QS0 and QS1 respectively. This mode is available on the 80C186XL in both Compatible and Enhanced Modes.

DRAM Refresh Control Unit

The Refresh Control Unit (RCU) automatically generates DRAM refresh bus cycles. The RCU operates only in Enhanced Mode. After a programmable period of time, the RCU generates a memory read request to the BIU. If the address generated during a refresh bus cycle is within the range of a properly programmed chip select, that chip select will be activated when the BIU executes the refresh bus cycle.

Power-Save Control

The 80C186XL, when in Enhanced Mode, can enter a power saving state by internally dividing the processor clock frequency by a programmable factor. This divided frequency is also available at the CLKOUT pin.

All internal logic, including the Refresh Control Unit and the timers, have their clocks slowed down by the division factor. To maintain a real time count or a fixed DRAM refresh rate, these peripherals must be re-programmed when entering and leaving the power-save mode.

Interface for 80C187 Math Coprocessor (80C186XL Only)

In Enhanced Mode, three of the mid-range memory chip selects are redefined according to Table 1 for use with the 80C187. The fourth chip select, $\overline{MCS2}$ functions as in compatible mode, and may be programmed for activity with ready logic and wait states accordingly. As in Compatible Mode, $\overline{MCS2}$ will function for one-fourth a programmed block size.

Table 1. $\overline{MCS}$ Assignments

Compatible Mode	Enhanced Mode	
$\overline{MCS0}$	PEREQ	Processor Extension Request
$\overline{MCS1}$	$\overline{ERROR}$	NPX Error
$\overline{MCS2}$	$\overline{MCS2}$	Mid-Range Chip Select
$\overline{MCS3}$	$\overline{NPS}$	Numeric Processor Select

ONCE Test Mode

To facilitate testing and inspection of devices when fixed into a target system, the 80C186XL has a test mode available which allows all pins to be placed in a high-impedance state. ONCE stands for "ON Circuit Emulation". When placed in this mode, the 80C186XL will put all pins in the high-impedance state until RESET.

The ONCE mode is selected by tying the $\overline{UCS}$ and the $\overline{LCS}$ LOW during RESET. These pins are sampled on the low-to-high transition of the $\overline{RES}$ pin. The $\overline{UCS}$ and the $\overline{LCS}$ pins have weak internal pull-up resistors similar to the $\overline{RD}$ and $\overline{TEST}$/BUSY pins to guarantee ONCE Mode is not entered inadvertently during normal operation. $\overline{LCS}$ and $\overline{UCS}$ must be held low at least one clock after $\overline{RES}$ goes high to guarantee entrance into ONCE Mode.

PACKAGE INFORMATION

This section describes the pin functions, pinout and thermal characteristics for the 80C186XL in the Quad Flat Pack (QFP), Plastic Leaded Chip Carrier (PLCC), Leadless Chip Carrier (LCC) and the Shrink Quad Flat Pack (SQFP). For complete package specifications and information, see the Intel Packaging Outlines and Dimensions Guide (Order Number: 231369).

Pin Descriptions

Each pin or logical set of pins is described in Table 3. There are four columns for each entry in the Pin Description Table. The following sections describe each column.

Column 1: Pin Name

In this column is a mnemonic that describes the pin function. Negation of the signal name (i.e., $\overline{\text{RESIN}}$) implies that the signal is active low.

Column 2: Pin Type

A pin may be either power (P), ground (G), input only (I), output only (O) or input/output (I/O). Please note that some pins have more than one function.

Column 3: Input Type (for I and I/O types only)

These are two different types of input pins on the 80C186XL: asynchronous and synchronous. **Asynchronous** pins require that setup and hold times be met only to *guarantee recognition.* **Synchronous** input pins require that the setup and hold times be met to *guarantee proper operation.* Stated simply, missing a setup or hold on an asynchronous pin will result in something minor (i.e., a timer count will be missed) whereas missing a setup or hold on a synchronous pin result in system failure (the system will "lock up").

An input pin may also be edge or level sensitive.

Column 4: Output States (for O and I/O types only)

The state of an output or I/O pin is dependent on the operating mode of the device. There are four modes of operation that are different from normal active mode: Bus Hold, Reset, Idle Mode, Powerdown Mode. This column describes the output pin state in each of these modes.

The legend for interpreting the information in the Pin Descriptions is shown in Table 2.

As an example, please refer to the table entry for AD7:0. The "I/O" signifies that the pins are bidirectional (i.e., have both an input and output function). The "S" indicates that, as an input the signal must be synchronized to CLKOUT for proper operation. The "H(Z)" indicates that these pins will float while the processor is in the Hold Acknowledge state. R(Z) indicates that these pins will float while $\overline{\text{RESIN}}$ is low.

All pins float while the processor is in the ONCE Mode (with the exception of X2).

Table 2. Pin Description Nomenclature

Symbol	Description
P	Power Pin (apply + V_{CC} voltage)
G	Ground (connect to V_{SS})
I	Input only pin
O	Output only pin
I/O	Input/Output pin
S(E)	Synchronous, edge sensitive
S(L)	Synchronous, level sensitive
A(E)	Asynchronous, edge sensitive
A(L)	Asynchronous, level sensitive
H(1)	Output driven to V_{CC} during bus hold
H(0)	Output driven to V_{SS} during bus hold
H(Z)	Output floats during bus hold
H(Q)	Output remains active during bus hold
H(X)	Output retains current state during bus hold
R(WH)	Output weakly held at V_{CC} during reset
R(1)	Output driven to V_{CC} during reset
R(0)	Output driven to V_{SS} during reset
R(Z)	Output floats during reset
R(Q)	Output remains active during reset
R(X)	Output retains current state during reset

Table 3. Pin Descriptions

Pin Name	Pin Type	Input Type	Output States	Pin Description
V_{CC}	P			System Power: +5 volt power supply.
V_{SS}	G			System Ground.
RESET	O		H(0) R(1)	RESET Output indicates that the CPU is being reset, and can be used as a system reset. It is active HIGH, synchronized with the processor clock, and lasts an integer number of clock periods corresponding to the length of the $\overline{RES}$ signal. Reset goes inactive 2 clockout periods after $\overline{RES}$ goes inactive. When tied to the $\overline{TEST}$/BUSY pin, RESET forces the processor into enhanced mode. RESET is not floated during bus hold.
X1	I	A(E)		Crystal Inputs X1 and X2 provide external connections for a fundamental mode or third overtone parallel resonant crystal for the internal oscillator. X1 can connect to an external clock instead of a crystal. In this case, minimize the capacitance on X2. The input or oscillator frequency is internally divided by two to generate the clock signal (CLKOUT).
X2	O		H(Q) R(Q)	
CLKOUT	O		H(Q) R(Q)	Clock Output provides the system with a 50% duty cycle waveform. All device pin timings are specified relative to CLKOUT. CLKOUT is active during reset and bus hold.
$\overline{RES}$	I	A(L)		An active $\overline{RES}$ causes the processor to immediately terminate its present activity, clear the internal logic, and enter a dormant state. This signal may be asynchronous to the clock. The processor begins fetching instructions approximately 6½ clock cycles after $\overline{RES}$ is returned HIGH. For proper initialization, V_{CC} must be within specifications and the clock signal must be stable for more than 4 clocks with $\overline{RES}$ held LOW. $\overline{RES}$ is internally synchronized. This input is provided with a Schmitt-trigger to facilitate power-on $\overline{RES}$ generation via an RC network.
$\overline{TEST}$/BUSY ($\overline{TEST}$)	I	A(E)		The $\overline{TEST}$ pin is sampled during and after reset to determine whether the processor is to enter Compatible or Enhanced Mode. Enhanced Mode requires $\overline{TEST}$ to be HIGH on the rising edge of $\overline{RES}$ and LOW four CLKOUT cycles later. Any other combination will place the processor in Compatible Mode. During power-up, active $\overline{RES}$ is required to configure $\overline{TEST}$/BUSY as an input. A weak internal pullup ensures a HIGH state when the input is not externally driven. $\overline{TEST}$—In Compatible Mode this pin is configured to operate as $\overline{TEST}$. This pin is examined by the WAIT instruction. If the $\overline{TEST}$ input is HIGH when WAIT execution begins, instruction execution will suspend. $\overline{TEST}$ will be resampled every five clocks until it goes LOW, at which time execution will resume. If interrupts are enabled while the processor is waiting for $\overline{TEST}$, interrupts will be serviced. BUSY (80C186XL Only)—In Enhanced Mode, this pin is configured to operate as BUSY. The BUSY input is used to notify the 80C186XL of Math Coprocessor activity. Floating point instructions executing in the 80C186XL sample the BUSY pin to determine when the Math Coprocessor is ready to accept a new command. BUSY is active HIGH.

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
TMR IN 0 TMR IN 1	I	A(L) A(E)		Timer Inputs are used either as clock or control signals, depending upon the programmed timer mode. These inputs are active HIGH (or LOW-to-HIGH transitions are counted) and internally synchronized. Timer Inputs must be tied HIGH when not being used as clock or retrigger inputs.
TMR OUT 0 TMR OUT 1	O		H(Q) R(1)	Timer outputs are used to provide single pulse or continuous waveform generation, depending upon the timer mode selected. These outputs are not floated during a bus hold.
DRQ0 DRQ1	I	A(L)		DMA Request is asserted HIGH by an external device when it is ready for DMA Channel 0 or 1 to perform a transfer. These signals are level-triggered and internally synchronized.
NMI	I	A(E)		The Non-Maskable Interrupt input causes a Type 2 interrupt. An NMI transition from LOW to HIGH is latched and synchronized internally, and initiates the interrupt at the next instruction boundary. NMI must be asserted for at least one CLKOUT period. The Non-Maskable Interrupt cannot be avoided by programming.
INT0 INT1/$\overline{\text{SELECT}}$	I	A(E) A(L)		Maskable Interrupt Requests can be requested by activating one of these pins. When configured as inputs, these pins are active HIGH. Interrupt Requests are synchronized internally. INT2 and INT3 may be configured to provide active-LOW interrupt-acknowledge output signals. All interrupt inputs may be configured to be either edge- or level-triggered. To ensure recognition, all interrupt requests must remain active until the interrupt is acknowledged. When Slave Mode is selected, the function of these pins changes (see Interrupt Controller section of this data sheet).
INT2/$\overline{\text{INTA0}}$ INT3/$\overline{\text{INTA1}}$/IRQ	I/O	A(E) A(L)	H(1) R(Z)	
A19/S6 A18/S5 A17/S4 A16/S3 (A8–A15)	O		H(Z) R(Z)	Address Bus Outputs and Bus Cycle Status (3–6) indicate the four most significant address bits during T_1. These signals are active HIGH. During T_2, T_3, T_W and T_4, the S6 pin is LOW to indicate a CPU-initiated bus cycle or HIGH to indicate a DMA-initiated or refresh bus cycle. During the same T-states, S3, S4 and S5 are always LOW. On the 80C188XL, A15–A8 provide valid address information for the entire bus cycle.
AD0–AD15 (AD0–AD7)	I/O	S(L)	H(Z) R(Z)	Address/Data Bus signals constitute the time multiplexed memory or I/O address (T_1) and data (T_2, T_3, T_W and T_4) bus. The bus is active HIGH. For the 80C186XL, A_0 is analogous to $\overline{\text{BHE}}$ for the lower byte of the data bus, pins D_7 through D_0. It is LOW during T_1 when a byte is to be transferred onto the lower portion of the bus in memory or I/O operations.

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
$\overline{BHE}$ ($\overline{RFSH}$)	O		H(Z) R(Z)	The $\overline{BHE}$ (Bus High Enable) signal is analogous to A0 in that it is used to enable data on to the most significant half of the data bus, pins D15–D8. $\overline{BHE}$ will be LOW during T_1 when the upper byte is transferred and will remain LOW through T_3 and T_W. $\overline{BHE}$ does not need to be latched. On the 80C188XL, $\overline{RFSH}$ is asserted LOW to indicate a refresh bus cycle. In Enhanced Mode, $\overline{BHE}$ ($\overline{RFSH}$) will also be used to signify DRAM refresh cycles. A refresh cycle is indicated by both $\overline{BHE}$ ($\overline{RFSH}$) and A0 being HIGH.
				80C186XL $\overline{BHE}$ and A0 Encodings
				$\overline{BHE}$ Value / **A0 Value** / **Function** 0 / 0 / Word Transfer 0 / 1 / Byte Transfer on upper half of data bus (D15–D8) 1 / 0 / Byte Transfer on lower half of data bus (D_7–D_0) 1 / 1 / Refresh
ALE/QS0	O		H(0) R(0)	Address Latch Enable/Queue Status 0 is provided by the processor to latch the address. ALE is active HIGH, with addresses guaranteed valid on the trailing edge.
$\overline{WR}$/QS1	O		H(Z) R(Z)	Write Strobe/Queue Status 1 indicates that the data on the bus is to be written into a memory or an I/O device. It is active LOW. When the processor is in Queue Status Mode, the ALE/QS0 and $\overline{WR}$/QS1 pins provide information about processor/instruction queue interaction.
				QS1 / **QS0** / **Queue Operation** 0 / 0 / No queue operation 0 / 1 / First opcode byte fetched from the queue 1 / 1 / Subsequent byte fetched from the queue 1 / 0 / Empty the queue
$\overline{RD}$/$\overline{QSMD}$	O		H(Z) R(1)	Read Strobe is an active LOW signal which indicates that the processor is performing a memory or I/O read cycle. It is guaranteed not to go LOW before the A/D bus is floated. An internal pull-up ensures that $\overline{RD}$/$\overline{QSMD}$ is HIGH during RESET. Following RESET the pin is sampled to determine whether the processor is to provide ALE, $\overline{RD}$, and $\overline{WR}$, or queue status information. To enable Queue Status Mode, $\overline{RD}$ must be connected to GND.
ARDY	I	A(L) S(L)		Asynchronous Ready informs the processor that the addressed memory space or I/O device will complete a data transfer. The ARDY pin accepts a rising edge that is asynchronous to CLKOUT and is active HIGH. The falling edge of ARDY must be synchronized to the processor clock. Connecting ARDY HIGH will always assert the ready condition to the CPU. If this line is unused, it should be tied LOW to yield control to the SRDY pin.

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
SRDY	I	S(L)	—	Synchronous Ready informs the processor that the addressed memory space or I/O device will complete a data transfer. The SRDY pin accepts an active-HIGH input synchronized to CLKOUT. The use of SRDY allows a relaxed system timing over ARDY. This is accomplished by elimination of the one-half clock cycle required to internally synchonize the ARDY input signal. Connecting SRDY high will always assert the ready condition to the CPU. If this line is unused, it should be tied LOW to yield control to the ARDY pin.
$\overline{LOCK}$	O	—	H(Z) R(Z)	$\overline{LOCK}$ output indicates that other system bus masters are not to gain control of the system bus. $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is requested by the LOCK prefix instruction and is activated at the beginning of the first data cycle associated with the instruction immediately following the LOCK prefix. It remains active until the completion of that instruction. No instruction prefetching will occur while $\overline{LOCK}$ is asserted.
$\overline{S0}$ $\overline{S1}$ $\overline{S2}$	O	—	H(Z) R(1)	Bus cycle status $\overline{S0}$–$\overline{S2}$ are encoded to provide bus-transaction information: **Bus Cycle Status Information** $\overline{S2}$ $\overline{S1}$ $\overline{S0}$ — Bus Cycle Initiated 0 0 0 — Interrupt Acknowledge 0 0 1 — Read I/O 0 1 0 — Write I/O 0 1 1 — Halt 1 0 0 — Instruction Fetch 1 0 1 — Read Data from Memory 1 1 0 — Write Data to Memory 1 1 1 — Passive (no bus cycle) $\overline{S2}$ may be used as a logical M/$\overline{IO}$ indicator, and $\overline{S1}$ as a DT/$\overline{R}$ indicator.
HOLD	I	A(L)	—	HOLD indicates that another bus master is requesting the local bus. The HOLD input is active HIGH. The processor generates HLDA (HIGH) in response to a HOLD request. Simultaneous with the issuance of HLDA, the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will lower HLDA. When the processor needs to run another bus cycle, it will again drive the local bus and control lines. In Enhanced Mode, HLDA will go low when a DRAM refresh cycle is pending in the processor and an external bus master has control of the bus. It will be up to the external master to relinquish the bus by lowering HOLD so that the processor may execute the refresh cycle.
HLDA	O	—	H(1) R(0)	

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

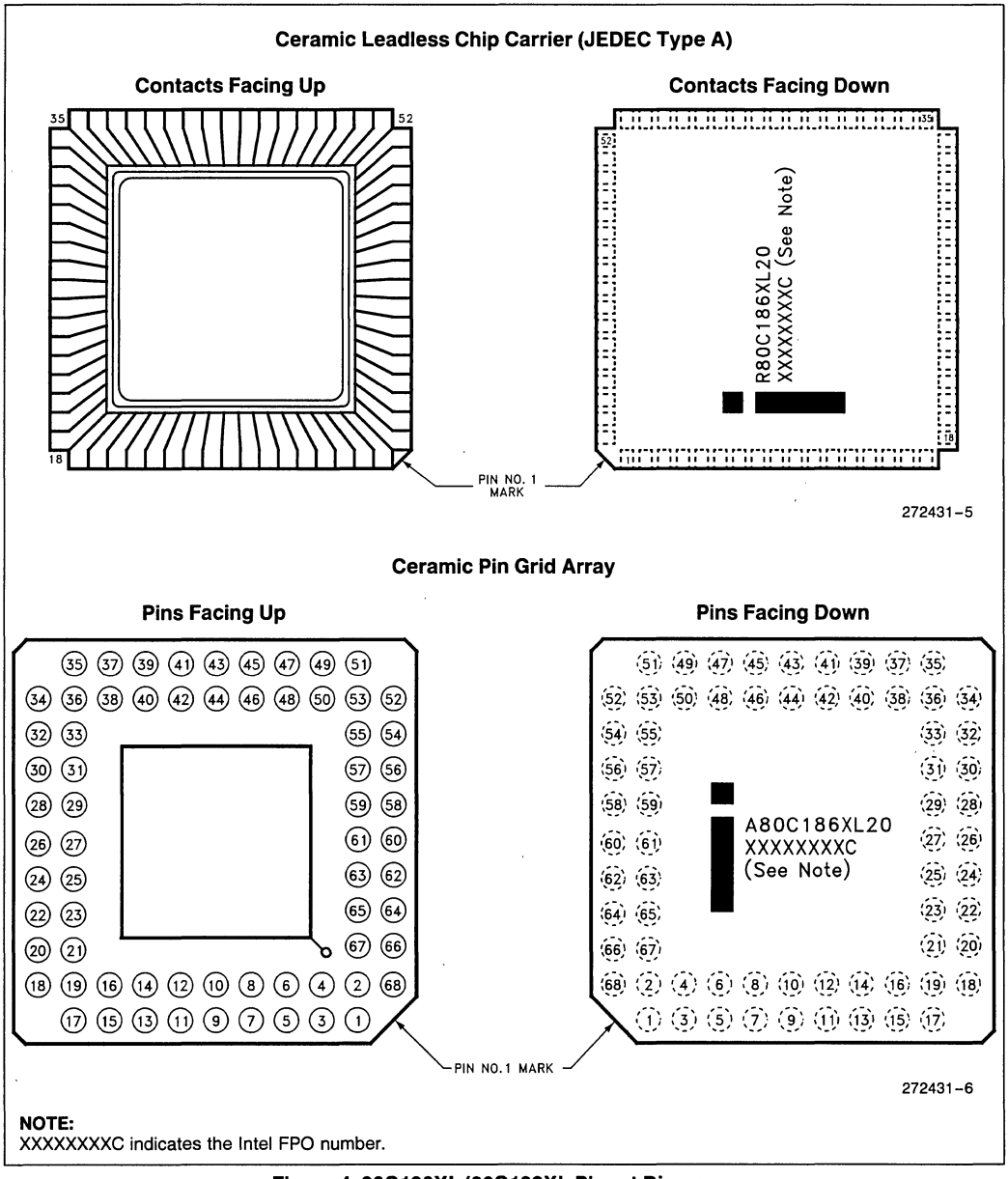
Pin Name	Pin Type	Input Type	Output States	Pin Description
$\overline{UCS}$	I/O	A(L)	H(1) R(WH)	Upper Memory Chip Select is an active LOW output whenever a memory reference is made to the defined upper portion (1K–256K block) of memory. The address range activating $\overline{UCS}$ is software programmable. $\overline{UCS}$ and $\overline{LCS}$ are sampled upon the rising edge of $\overline{RES}$. If both pins are held low, the processor will enter ONCE Mode. In ONCE Mode all pins assume a high impedance state and remain so until a subsequent RESET. $\overline{UCS}$ has a weak internal pullup that is active during RESET to ensure that the processor does not enter ONCE Mode inadvertently.
$\overline{LCS}$	I/O	A(L)	H(1) R(WH)	Lower Memory Chip Select is active LOW whenever a memory reference is made to the defined lower portion (1K–256K) of memory. The address range activating $\overline{LCS}$ is software programmable. $\overline{UCS}$ and $\overline{LCS}$ are sampled upon the rising edge of $\overline{RES}$. If both pins are held low, the processor will enter ONCE Mode. In ONCE Mode all pins assume a high impedance state and remain so until a subsequent RESET. $\overline{LCS}$ has a weak internal pullup that is active only during RESET to ensure that the processor does not enter ONCE mode inadvertently.
$\overline{MCS0}$/PEREQ $\overline{MCS1}$/$\overline{ERROR}$	I/O	A(L)	H(1) R(WH)	Mid-Range Memory Chip Select signals are active LOW when a memory reference is made to the defined mid-range portion of memory (8K–512K). The address ranges activating $\overline{MCS0–3}$ are software programmable. On the 80C186XL, in Enhanced Mode, $\overline{MCS0}$ becomes a PEREQ input (Processor Extension Request). When connected to the Math Coprocessor, this input is used to signal the 80C186XL when to make numeric data transfers to and from the coprocessor. $\overline{MCS3}$ becomes $\overline{NPS}$ (Numeric Processor Select) which may only be activated by communication to the 80C187. $\overline{MCS1}$ becomes $\overline{ERROR}$ in Enhanced Mode and is used to signal numerics coprocessor errors.
$\overline{MCS2}$ $\overline{MCS3}$/$\overline{NPS}$	O		H(1) R(1)	
$\overline{PCS0}$ $\overline{PCS1}$ $\overline{PCS2}$ $\overline{PCS3}$ $\overline{PCS4}$	O		H(1) R(1)	Peripheral Chip Select signals 0–4 are active LOW when a reference is made to the defined peripheral area (64 Kbyte I/O or 1 MByte memory space). The address ranges activating $\overline{PCS0–4}$ are software programmable.
$\overline{PCS5}$/A1	O		H(1)/H(X) R(1)	Peripheral Chip Select 5 or Latched A1 may be programmed to provide a sixth peripheral chip select, or to provide an internally latched A1 signal. The address range activating $\overline{PCS5}$ is software-programmable. $\overline{PCS5}$/A1 does not float during bus HOLD. When programmed to provide latched A1, this pin will retain the previously latched value during HOLD.

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

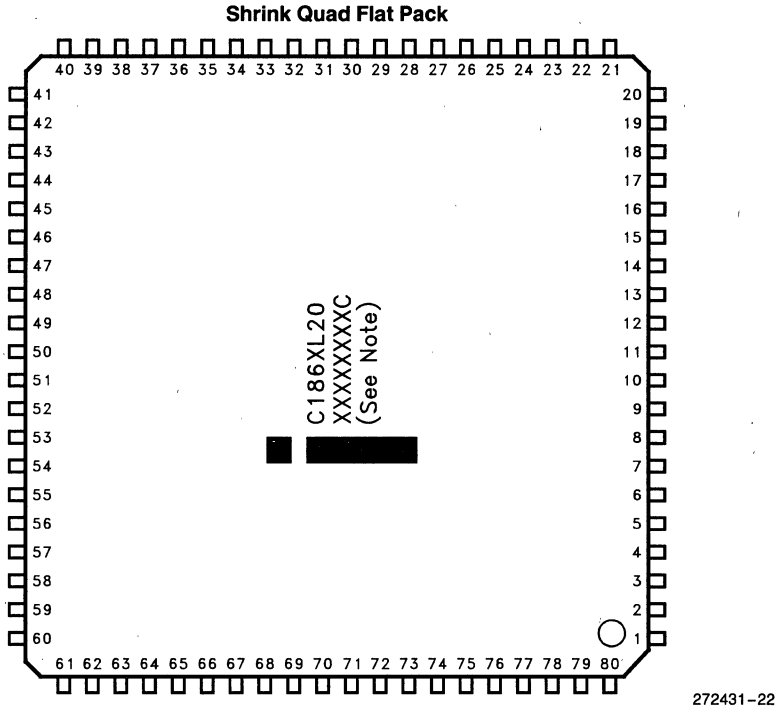
Pin Name	Pin Type	Input Type	Output States	Pin Description
$\overline{\text{PCS6}}$/A2	O	—	H(1)/H(X) R(1)	Peripheral Chip Select 6 or Latched A2 may be programmed to provide a seventh peripheral chip select, or to provide an internally latched A2 signal. The address range activating $\overline{\text{PCS6}}$ is software-programmable. $\overline{\text{PCS6}}$/A2 does not float during bus HOLD. When programmed to provide latched A2, this pin will retain the previously latched value during HOLD.
DT/$\overline{\text{R}}$	O	—	H(Z) R(Z)	Data Transmit/Receive controls the direction of data flow through an external data bus transceiver. When LOW, data is transferred to the procesor. When HIGH the processor places write data on the data bus.
$\overline{\text{DEN}}$	O	—	H(Z) R(1,Z)	Data Enable is provided as a data bus transceiver output enable. $\overline{\text{DEN}}$ is active LOW during each memory and I/O access (including 80C187 access). $\overline{\text{DEN}}$ is HIGH whenever DT/$\overline{\text{R}}$ changes state. During RESET, $\overline{\text{DEN}}$ is driven HIGH for one clock, then floated.
N.C.	—	—	—	Not connected. To maintain compatibility with future products, do not connect to these pins.

NOTE:
Pin names in parentheses apply to the 80C188XL.



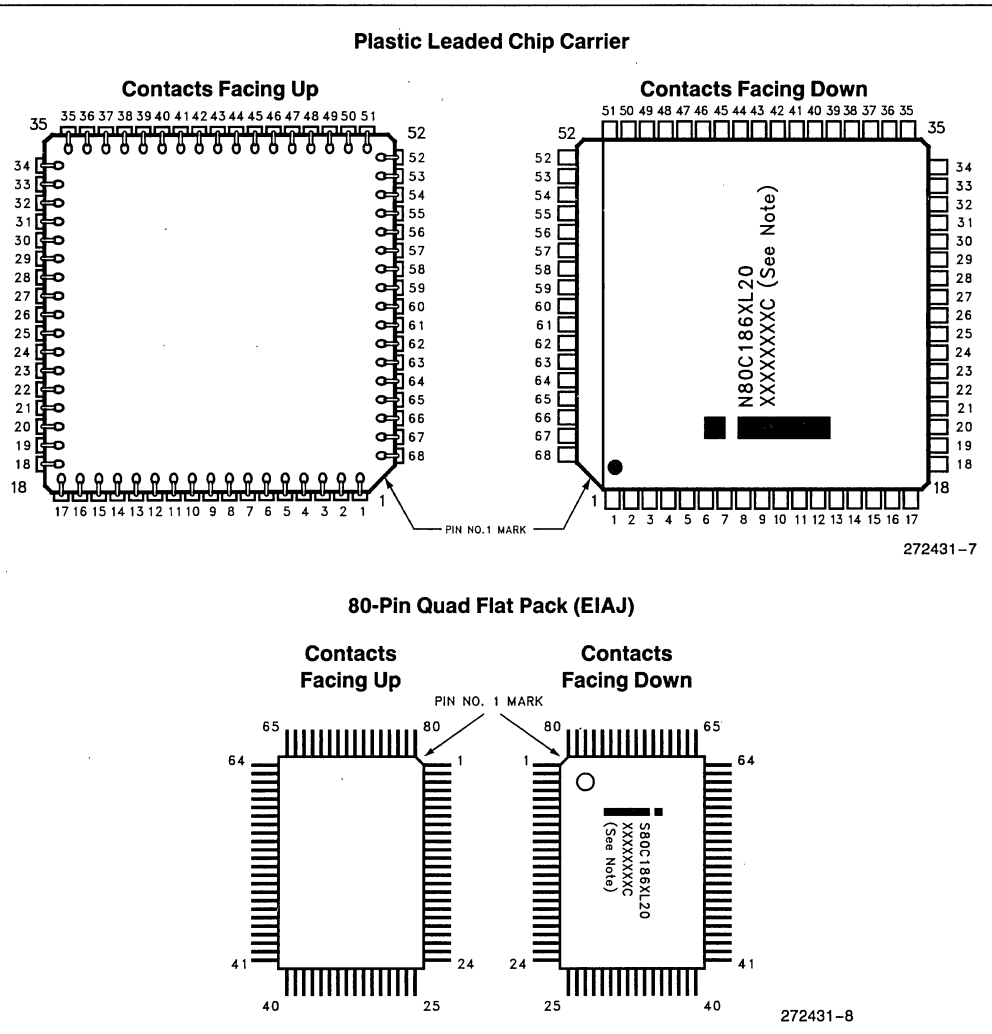
NOTE:
XXXXXXXXC indicates the Intel FPO number.

Figure 4. 80C186XL/80C188XL Pinout Diagrams



NOTE:
XXXXXXXXC indicates the Intel FPO number.

Figure 4. 80C186XL/80C188XL Pinout Diagrams (Continued)



NOTE:
XXXXXXXXA indicates the Intel FPO number.

Figure 4. 80C186XL/80C288XL Pinout Diagrams (Continued)

Table 4. LCC/PLCC Pin Functions with Location

AD Bus	
AD0	17
AD1	15
AD2	13
AD3	11
AD4	8
AD5	6
AD6	4
AD7	2
AD8 (A8)	16
AD9 (A9)	14
AD10 (A10)	12
AD11 (A11)	10
AD12 (A12)	7
AD13 (A13)	5
AD14 (A14)	3
AD15 (A15)	1
A16/S3	68
A17/S4	67
A18/S5	66
A19/S6	65

Bus Control	
ALE/QS0	61
$\overline{BHE}$ ($\overline{RFSH}$)	64
$\overline{S0}$	52
$\overline{S1}$	53
$\overline{S2}$	54
$\overline{RD}$/$\overline{QSMD}$	62
$\overline{WR}$/QS1	63
ARDY	55
SRDY	49
$\overline{DEN}$	39
DT/$\overline{R}$	40
$\overline{LOCK}$	48
HOLD	50
HLDA	51

Processor Control	
$\overline{RES}$	24
RESET	57
X1	59
X2	58
CLKOUT	56
$\overline{TEST}$/BUSY	47
NMI	46
INT0	45
INT1/$\overline{SELECT}$	44
INT2/$\overline{INTA0}$	42
INT3/$\overline{INTA1}$	41

Power and Ground	
V_{CC}	9
V_{CC}	43
V_{SS}	26
V_{SS}	60

I/O	
$\overline{UCS}$	34
$\overline{LCS}$	33
$\overline{MCS0}$/PEREQ	38
$\overline{MCS1}$/$\overline{ERROR}$	37
$\overline{MCS2}$	36
$\overline{MCS3}$/$\overline{NPS}$	35
$\overline{PCS0}$	25
$\overline{PCS1}$	27
$\overline{PCS2}$	28
$\overline{PCS3}$	29
$\overline{PCS4}$	30
$\overline{PCS5}$/A1	31
$\overline{PCS6}$/A2	32
TMR IN 0	20
TMR IN 1	21
TMR OUT 0	22
TMR OUT 1	23
DRQ0	18
DRQ1	19

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 5. LCC/PGA/PLCC Pin Locations with Pin Names

1	AD15 (A15)
2	AD7
3	AD14 (A14)
4	AD6
5	AD13 (A13)
6	AD5
7	AD12 (A12)
8	AD4
9	V_{CC}
10	AD11 (A11)
11	AD3
12	AD10 (A10)
13	AD2
14	AD9 (A9)
15	AD1
16	AD8 (A8)
17	AD0

18	DRQ0
19	DRQ1
20	TMR IN 0
21	TMR IN 1
22	TMR OUT 0
23	TMR OUT 1
24	$\overline{RES}$
25	$\overline{PCS0}$
26	V_{SS}
27	$\overline{PCS1}$
28	$\overline{PCS2}$
29	$\overline{PCS3}$
30	$\overline{PCS4}$
31	$\overline{PCS5}$/A1
32	$\overline{PCS6}$/A2
33	$\overline{LCS}$
34	$\overline{UCS}$

35	$\overline{MCS3}$/$\overline{NPS}$
36	$\overline{MCS2}$
37	$\overline{MCS1}$/$\overline{ERROR}$
38	$\overline{MCS0}$/PEREQ
39	$\overline{DEN}$
40	DT/$\overline{R}$
41	INT3/$\overline{INTA1}$
42	INT2/$\overline{INTA0}$
43	V_{CC}
44	INT1/$\overline{SELECT}$
45	INT0
46	NMI
47	$\overline{TEST}$/BUSY
48	$\overline{LOCK}$
49	SRDY
50	HOLD
51	HLDA

52	$\overline{S0}$
53	$\overline{S1}$
54	$\overline{S2}$
55	ARDY
56	CLKOUT
57	RESET
58	X2
59	X1
60	V_{SS}
61	ALE/QS0
62	$\overline{RD}$/$\overline{QSMD}$
63	$\overline{WR}$/QS1
64	$\overline{BHE}$ ($\overline{RFSH}$)
65	A19/S2
66	A18/S3
67	A17/S4
68	A16/S3

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 6. QFP Pin Functions with Location

AD Bus	
AD0	64
AD1	66
AD2	68
AD3	70
AD4	74
AD5	76
AD6	78
AD7	80
AD8 (A8)	65
AD9 (A9)	67
AD10 (A10)	69
AD11 (A11)	71
AD12 (A12)	75
AD13 (A13)	77
AD14 (A14)	79
AD15 (A15)	1
A16/S3	3
A17/S4	4
A18/S5	5
A19/S6	6

Bus Control	
ALE/QS0	10
$\overline{BHE}$ ($\overline{RFSH}$)	7
$\overline{S0}$	23
$\overline{S1}$	22
$\overline{S2}$	21
$\overline{RD}$/QSMD	9
$\overline{WR}$/QS1	8
ARDY	20
SRDY	27
$\overline{DEN}$	38
DT/$\overline{R}$	37
$\overline{LOCK}$	28
HOLD	26
HLDA	25

No Connection	
N.C.	2
N.C.	11
N.C.	14
N.C.	15
N.C.	24
N.C.	43
N.C.	44
N.C.	62
N.C.	63

Processor Control	
$\overline{RES}$	55
RESET	18
X1	16
X2	17
CLKOUT	19
$\overline{TEST}$/BUSY	29
NMI	30
INT0	31
INT1/$\overline{SELECT}$	32
INT2/$\overline{INTA0}$	35
INT3/$\overline{INTA1}$	36

Power and Ground	
V_{CC}	33
V_{CC}	34
V_{CC}	72
V_{CC}	73
V_{SS}	12
V_{SS}	13
V_{SS}	53

I/O	
$\overline{UCS}$	45
$\overline{LCS}$	46
$\overline{MCS0}$/PEREQ	39
$\overline{MCS1}$/ERROR	40
$\overline{MCS2}$	41
$\overline{MCS3}$/$\overline{NPS}$	42
$\overline{PCS0}$	54
$\overline{PCS1}$	52
$\overline{PCS2}$	51
$\overline{PCS3}$	50
$\overline{PCS4}$	49
$\overline{PCS5}$/A1	48
$\overline{PCS6}$/A2	47
TMR IN 0	59
TMR IN 1	58
TMR OUT 0	57
TMR OUT 1	56
DRQ0	61
DRQ1	60

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 7. QFP Pin Locations with Pin Names

1	AD15 (A15)
2	N.C.
3	A16/S3
4	A17/S4
5	A18/S5
6	A19/S6
7	$\overline{BHE}$/($\overline{RFSH}$)
8	$\overline{WR}$/QS1
9	$\overline{RD}$/QSMD
10	ALE/QS0
11	N.C.
12	V_{SS}
13	V_{SS}
14	N.C.
15	N.C.
16	X1
17	X2
18	RESET
19	CLKOUT
20	ARDY

21	$\overline{S2}$
22	$\overline{S1}$
23	$\overline{S0}$
24	N.C.
25	HLDA
26	HOLD
27	SRDY
28	$\overline{LOCK}$
29	$\overline{TEST}$/BUSY
30	NMI
31	INT0
32	INT1/$\overline{SELECT}$
33	V_{CC}
34	V_{CC}
35	INT2/$\overline{INTA0}$
36	INT3/$\overline{INTA1}$
37	DT/$\overline{R}$
38	$\overline{DEN}$
39	$\overline{MCS0}$/PEREQ
40	$\overline{MCS1}$/ERROR

41	$\overline{MCS2}$
42	$\overline{MCS3}$/$\overline{NPS}$
43	N.C.
44	N.C.
45	$\overline{UCS}$
46	$\overline{LCS}$
47	$\overline{PCS6}$/A2
48	$\overline{PCS5}$/A1
49	$\overline{PCS4}$
50	$\overline{PCS3}$
51	$\overline{PCS2}$
52	$\overline{PCS1}$
53	V_{SS}
54	$\overline{PCS0}$
55	$\overline{RES}$
56	TMR OUT 1
57	TMR OUT 0
58	TMR IN 1
59	TMR IN 0
60	DRQ1

61	DRQ0
62	N.C.
63	N.C.
64	AD0
65	AD8 (A8)
66	AD1
67	AD9 (A9)
68	AD2
69	AD10 (A10)
70	AD3
71	AD11 (A11)
72	V_{CC}
73	V_{CC}
74	AD4
75	AD12 (A12)
76	AD5
77	AD13 (A13)
78	AD6
79	AD14 (A14)
80	AD7

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 8. SQFP Pin Functions with Location

AD Bus	
AD0	1
AD1	3
AD2	6
AD3	8
AD4	12
AD5	14
AD6	16
AD7	18
AD8 (A8)	2
AD9 (A9)	5
AD10 (A10)	7
AD11 (A11)	9
AD12 (A12)	13
AD13 (A13)	15
AD14 (A14)	17
AD15 (A15)	19
A16/S3	21
A17/S4	22
A18/S5	23
A19/S6	24

Bus Control	
ALE/QS0	29
$\overline{BHE}$ ($\overline{RFSH}$)	26
$\overline{S0}$	40
$\overline{S1}$	39
$\overline{S2}$	38
$\overline{RD}$/$\overline{QSMD}$	28
$\overline{WR}$/QS1	27
ARDY	37
SRDY	44
$\overline{DEN}$	56
DT/$\overline{R}$	54
$\overline{LOCK}$	45
HOLD	43
HLDA	42

No Connection	
N.C.	4
N.C.	25
N.C.	35
N.C.	55
N.C.	72

Processor Control	
$\overline{RES}$	73
RESET	34
X1	32
X2	33
CLKOUT	36
$\overline{TEST}$/BUSY	46
NMI	47
INT0	48
INT1/$\overline{SELECT}$	49
INT2/$\overline{INTA0}$	52
INT3/$\overline{INTA1}$	53

Power and Ground	
V_{CC}	10
V_{CC}	11
V_{CC}	20
V_{CC}	50
V_{CC}	51
V_{CC}	61
V_{SS}	30
V_{SS}	31
V_{SS}	41
V_{SS}	70
V_{SS}	80

I/O	
$\overline{UCS}$	62
$\overline{LCS}$	63
$\overline{MCS0}$/PEREQ	57
$\overline{MCS1}$/$\overline{ERROR}$	58
$\overline{MCS2}$	59
$\overline{MCS3}$/$\overline{NPS}$	60
$\overline{PCS0}$	71
$\overline{PCS1}$	69
$\overline{PCS2}$	68
$\overline{PCS3}$	67
$\overline{PCS4}$	66
$\overline{PCS5}$/A1	65
$\overline{PCS6}$/A2	64
TMR IN 0	77
TMR IN 1	76
TMR OUT 0	75
TMR OUT 1	74
DRQ0	79
DRQ1	78

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 9. SQFP Pin Locations with Pin Names

1	AD0
2	AD8 (A8)
3	AD1
4	N.C.
5	AD9 (A9)
6	AD2
7	AD10 (A10)
8	AD3
9	AD11 (A11)
10	V_{CC}
11	V_{CC}
12	AD4
13	AD12 (A12)
14	AD5
15	AD13 (A13)
16	AD6
17	AD14 (A14)
18	AD7
19	AD15 (A15)
20	V_{CC}

21	A16/S3
22	A17/S4
23	A18/S5
24	A19/S6
25	N.C.
26	$\overline{BHE}$ ($\overline{RFSH}$)
27	$\overline{WR}$/QS1
28	$\overline{RD}$/$\overline{QSMD}$
29	ALE/QS0
30	V_{SS}
31	V_{SS}
32	X1
33	X2
34	RESET
35	N.C.
36	CLKOUT
37	ARDY
38	$\overline{S2}$
39	$\overline{S1}$
40	$\overline{S0}$

41	V_{SS}
42	HLDA
43	HOLD
44	SRDY
45	$\overline{LOCK}$
46	$\overline{TEST}$/BUSY
47	NMI
48	INT0
49	INT1/$\overline{SELECT}$
50	V_{CC}
51	V_{CC}
52	INT2/$\overline{INTA0}$
53	INT3/$\overline{INTA1}$
54	DT/$\overline{R}$
55	N.C.
56	$\overline{DEN}$
57	$\overline{MCS0}$/PEREQ
58	$\overline{MCS1}$/$\overline{ERROR}$
59	$\overline{MCS2}$
60	$\overline{MCS3}$/$\overline{NPS}$

61	V_{CC}
62	$\overline{UCS}$
63	$\overline{LCS}$
64	$\overline{PCS6}$/A2
65	$\overline{PCS5}$/A1
66	$\overline{PCS4}$
67	$\overline{PCS3}$
68	$\overline{PCS2}$
69	$\overline{PCS1}$
70	V_{SS}
71	$\overline{PCS0}$
72	N.C.
73	$\overline{RES}$
74	TMR OUT 1
75	TMR OUT 0
76	TMR IN 1
77	TMR IN 0
78	DRQ1
79	DRQ0
80	V_{SS}

NOTE:
Pin names in parentheses apply to the 80C188XL.

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings*

Ambient Temperature under Bias ....0°C to +70°C

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

Voltage on Any Pin with Respect to Ground .....−1.0V to +7.0V

Package Power Dissipation 1W
Not to exceed the maximum allowable die temperature based on thermal resistance of the package.

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

NOTICE: The specifications are subject to change without notice.

1

DC SPECIFICATIONS T_A = 0°C to +70°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (Except X1)	−0.5	0.2 V_{CC} − 0.3	V	
V_{IL1}	Clock Input Low Voltage (X1)	−0.5	0.6	V	
V_{IH}	Input High Voltage (All except X1 and $\overline{RES}$)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage ($\overline{RES}$)	3.0	V_{CC} + 0.5	V	
V_{IH2}	Clock Input High Voltage (X1)	3.9	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.5 mA (S0, 1, 2) I_{OL} = 2.0 mA (others)
V_{OH}	Output High Voltage	2.4	V_{CC}	V	I_{OH} = −2.4 mA @ 2.4V (4)
		V_{CC} − 0.5	V_{CC}	V	I_{OH} = −200 µA @ V_{CC} −0.5(4)
I_{CC}	Power Supply Current		100	mA	@ 25 MHz, 0°C V_{CC} = 5.5V(3)
			90	mA	@ 20 MHz, 0°C V_{CC} = 5.5V(3)
			62.5	mA	@ 12 MHz, 0°C V_{CC} = 5.5V (3)
			100	µA	@ DC 0°C V_{CC} = 5.5V
I_{LI}	Input Leakage Current		±10	µA	@ 0.5 MHz, 0.45V ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±10	µA	@ 0.5 MHz, 0.45V ≤ V_{OUT} ≤ V_{CC}(1)
V_{CLO}	Clock Output Low		0.45	V	I_{CLO} = 4.0 mA

DC SPECIFICATIONS (Continued) $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{CHO}	Clock Output High	$V_{CC} - 0.5$		V	$I_{CHO} = -500\ \mu A$
C_{IN}	Input Capacitance		10	pF	@ 1 MHz(2)
C_{IO}	Output or I/O Capacitance		20	pF	@ 1 MHz(2)

NOTES:
1. Pins being floated during HOLD or by invoking the ONCE Mode.
2. Characterization conditions are a) Frequency = 1 MHz; b) Unmeasured pins at GND; c) V_{IN} at + 5.0V or 0.45V. This parameter is not tested.
3. Current is measured with the device in RESET with X1 and X2 driven and all other non-power pins open.
4. $\overline{RD}/QSMD$, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$/PEREQ, $\overline{MCS1}/\overline{ERROR}$ and $\overline{TEST}$/BUSY pins have internal pullup devices. Loading some of these pins above $I_{OH} = -200\ \mu A$ can cause the processor to go into alternative modes of operation. See the section on Local Bus Controller and Reset for details.

Power Supply Current

Current is linearly proportional to clock frequency and is measured with the device in RESET with X1 and X2 driven and all other non-power pins open.

Maximum current is given by $I_{CC} = 5\ mA \times$ freq. (MHz) + I_{QL}.

I_{QL} is the quiescent leakage current when the clock is static. I_{QL} is typically less than 100 μA.

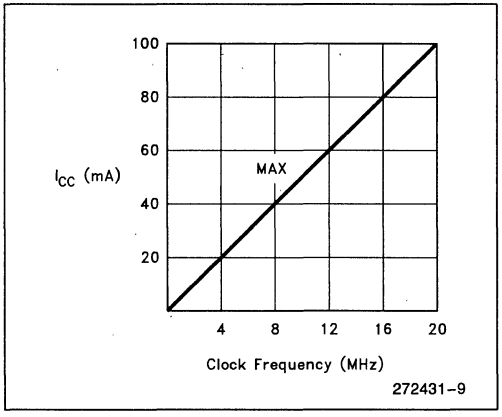


Figure 5. I_{CC} vs Frequency

AC SPECIFICATIONS

MAJOR CYCLE TIMINGS (READ CYCLE)

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where V_{IH} = V_{CC} − 0.5V.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL GENERAL TIMING REQUIREMENTS (Listed More Than Once)									
T_{DVCL}	Data in Setup (A/D)	8		10		15		ns	
T_{CLDX}	Data in Hold (A/D)	3		3		3		ns	
80C186XL GENERAL TIMING RESPONSES (Listed More Than Once)									
T_{CHSV}	Status Active Delay	3	20	3	25	3	35	ns	
T_{CLSH}	Status Inactive Delay	3	20	3	25	3	35	ns	
T_{CLAV}	Address Valid Delay	3	20	3	27	3	36	ns	
T_{CLAX}	Address Hold	0		0		0		ns	
T_{CLDV}	Data Valid Delay	3	20	3	27	3	36	ns	
T_{CHDX}	Status Hold Time	10		10		10		ns	
T_{CHLH}	ALE Active Delay		20		20		25	ns	
T_{LHLL}	ALE Width	T_{CLCL} − 15		T_{CLCL} − 15		T_{CLCL} − 15		ns	
T_{CHLL}	ALE Inactive Delay		20		20		25	ns	
T_{AVLL}	Address Valid to ALE Low	T_{CLCH} − 10		T_{CLCH} − 10		T_{CLCH} − 15		ns	Equal Loading
T_{LLAX}	Address Hold from ALE Inactive	T_{CHCL} − 8		T_{CHCL} − 10		T_{CHCL} − 15		ns	Equal Loading
T_{AVCH}	Address Valid to Clock High	0		0		0		ns	
T_{CLAZ}	Address Float Delay	T_{CLAX}	20	T_{CLAX}	20	T_{CLAX}	25	ns	
T_{CLCSV}	Chip-Select Active Delay	3	20	3	25	3	33	ns	
T_{CXCSX}	Chip-Select Hold from Command Inactive	T_{CLCH} − 10		T_{CLCH} − 10		T_{CLCH} − 10		ns	Equal Loading
T_{CHCSX}	Chip-Select Inactive Delay	3	17	3	20	3	30	ns	
T_{DXDL}	$\overline{DEN}$ Inactive to DT/$\overline{R}$ Low	0		0		0		ns	Equal Loading
T_{CVCTV}	Control Active Delay 1	3	17	3	22	3	37	ns	
T_{CVDEX}	$\overline{DEN}$ Inactive Delay	3	17	3	22	3	37	ns	
T_{CHCTV}	Control Active Delay 2	3	20	3	22	3	37	ns	
T_{CLLV}	$\overline{LOCK}$ Valid/Invalid Delay	3	17	3	22	3	37	ns	

AC SPECIFICATIONS (Continued)

MAJOR CYCLE TIMINGS (READ CYCLE) (Continued)

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where $V_{IH} = V_{CC} - 0.5V$.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL TIMING RESPONSES (Read Cycle)									
T_{AZRL}	Address Float to $\overline{RD}$ Active	0		0		0		ns	
T_{CLRL}	$\overline{RD}$ Active Delay	3	20	3	27	3	37	ns	
T_{RLRH}	$\overline{RD}$ Pulse Width	$2T_{CLCL} - 15$		$2T_{CLCL} - 20$		$2T_{CLCL} - 25$		ns	
T_{CLRH}	$\overline{RD}$ Inactive Delay	3	20	3	27	3	37	ns	
T_{RHLH}	$\overline{RD}$ Inactive to ALE High	$T_{CLCH} - 14$		$T_{CLCH} - 14$		$T_{CLCH} - 14$		ns	Equal Loading
T_{RHAV}	$\overline{RD}$ Inactive to Address Active	$T_{CLCL} - 15$		$T_{CLCL} - 15$		$T_{CLCL} - 15$		ns	Equal Loading

AC SPECIFICATIONS (Continued)

MAJOR CYCLE TIMINGS (WRITE CYCLE)

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where V_{IH} = V_{CC} − 0.5V.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL GENERAL TIMING RESPONSES (Listed More Than Once)									
T_{CHSV}	Status Active Delay	3	20	3	25	3	35	ns	
T_{CLSH}	Status Inactive Delay	3	20	3	25	3	35	ns	
T_{CLAV}	Address Valid Delay	3	20	3	27	3	36	ns	
T_{CLAX}	Address Hold	0		0		0		ns	
T_{CLDV}	Data Valid Delay	3	20	3	27	3	36	ns	
T_{CHDX}	Status Hold Time	10		10		10		ns	
T_{CHLH}	ALE Active Delay		20		20		25	ns	
T_{LHLL}	ALE Width	T_{CLCL} − 15		T_{CLCL} − 15		T_{CLCL} − 15		ns	
T_{CHLL}	ALE Inactive Delay		20		20		25	ns	
T_{AVLL}	Address Valid to ALE Low	T_{CLCH} − 10		T_{CLCH} − 10		T_{CLCH} − 15		ns	Equal Loading
T_{LLAX}	Address Hold from ALE Inactive	T_{CHCL} − 10		T_{CHCL} − 10		T_{CHCL} − 15		ns	Equal Loading
T_{AVCH}	Address Valid to Clock High	0		0		0		ns	
T_{CLDOX}	Data Hold Time	3		3		3		ns	
T_{CVCTV}	Control Active Delay 1	3	20	3	25	3	37	ns	
T_{CVCTX}	Control Inactive Delay	3	17	3	25	3	37	ns	
T_{CLCSV}	Chip-Select Active Delay	3	20	3	25	3	33	ns	
T_{CXCSX}	Chip-Select Hold from Command Inactive	T_{CLCH} − 10		T_{CLCH} − 10		T_{CLCH} − 10		ns	Equal Loading
T_{CHCSX}	Chip-Select Inactive Delay	3	17	3	20	3	30	ns	
T_{DXDL}	$\overline{DEN}$ Inactive to DT/$\overline{R}$ Low	0		0		0		ns	Equal Loading
T_{CLLV}	$\overline{LOCK}$ Valid/Invalid Delay	3	17	3	22	3	37	ns	
80C186XL TIMING RESPONSES (Write Cycle)									
T_{WLWH}	$\overline{WR}$ Pulse Width	$2T_{CLCL}$ − 15		$2T_{CLCL}$ − 20		$2T_{CLCL}$ − 25		ns	
T_{WHLH}	$\overline{WR}$ Inactive to ALE High	T_{CLCH} − 14		T_{CLCH} − 14		T_{CLCH} − 14		ns	Equal Loading
T_{WHDX}	Data Hold after $\overline{WR}$	T_{CLCL} − 10		T_{CLCL} − 15		T_{CLCL} − 20		ns	Equal Loading
T_{WHDEX}	$\overline{WR}$ Inactive to $\overline{DEN}$ Inactive	T_{CLCH} − 10		T_{CLCH} − 10		T_{CLCH} − 10		ns	Equal Loading

AC SPECIFICATIONS (Continued)

MAJOR CYCLE TIMINGS (INTERRUPT ACKNOWLEDGE CYCLE)

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where V_{IH} = V_{CC} − 0.5V.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL GENERAL TIMING REQUIREMENTS (Listed More Than Once)									
T_{DVCL}	Data in Setup (A/D)	8		10		15		ns	
T_{CLDX}	Data in Hold (A/D)	3		3		3		ns	
80C186XL GENERAL TIMING RESPONSES (Listed More Than Once)									
T_{CHSV}	Status Active Delay	3	20	3	25	3	35	ns	
T_{CLSH}	Status Inactive Delay	3	20	3	25	3	35	ns	
T_{CLAV}	Address Valid Delay	3	20	3	27	3	36	ns	
T_{AVCH}	Address Valid to Clock High	0		0		0		ns	
T_{CLAX}	Address Hold	0		0		0		ns	
T_{CLDV}	Data Valid Delay	3	20	3	27	3	36	ns	
T_{CHDX}	Status Hold Time	10		10		10		ns	
T_{CHLH}	ALE Active Delay		20		20		25	ns	
T_{LHLL}	ALE Width	T_{CLCL} − 15		T_{CLCL} − 15		T_{CLCL} − 15		ns	
T_{CHLL}	ALE Inactive Delay		20		20		25	ns	
T_{AVLL}	Address Valid to ALE Low	T_{CLCH} − 10		T_{CLCH} − 10		T_{CLCH} − 15		ns	Equal Loading
T_{LLAX}	Address Hold to ALE Inactive	T_{CHCL} − 10		T_{CHCL} − 10		T_{CHCL} − 15		ns	Equal Loading
T_{CLAZ}	Address Float Delay	T_{CLAX}	20	T_{CLAX}	20	T_{CLAX}	25	ns	
T_{CVCTV}	Control Active Delay 1	3	17	3	25	3	37	ns	
T_{CVCTX}	Control Inactive Delay	3	17	3	25	3	37	ns	
T_{DXDL}	$\overline{DEN}$ Inactive to DT/$\overline{R}$ Low	0		0		0		ns	Equal Loading
T_{CHCTV}	Control Active Delay 2	3	20	3	22	3	37	ns	
T_{CVDEX}	$\overline{DEN}$ Inactive Delay (Non-Write Cycles)	3	17	3	22	3	37	ns	
T_{CLLV}	$\overline{LOCK}$ Valid/Invalid Delay	3	17	3	22	3	37	ns	

AC SPECIFICATIONS (Continued)

SOFTWARE HALT CYCLE TIMINGS

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where V_{IH} = V_{CC} − 0.5V.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL GENERAL TIMING REQUIREMENTS (Listed More Than Once)									
T_{CHSV}	Status Active Delay	3	20	3	25	3	35	ns	
T_{CLSH}	Status Inactive Delay	3	20	3	25	3	35	ns	
T_{CLAV}	Address Valid Delay	3	20	3	27	3	36	ns	
T_{CHLH}	ALE Active Delay		20		20		25	ns	
T_{LHLL}	ALE Width	T_{CLCL} − 15		T_{CLCL} − 15		T_{CLCL} − 15		ns	
T_{CHLL}	ALE Inactive Delay		20		20		25	ns	
T_{DXDL}	$\overline{DEN}$ Inactive to DT/$\overline{R}$ Low		0		0		0	ns	Equal Loading
T_{CHCTV}	Control Active Delay 2	3	20	3	22	3	37	ns	

AC SPECIFICATIONS (Continued)

CLOCK TIMINGS

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where V_{IH} = V_{CC} − 0.5V.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL CLKIN REQUIREMENTS(1)									
T_{CKIN}	CLKIN Period	20	∞	25	∞	40	∞	ns	
T_{CLCK}	CLKIN Low Time	8	∞	10	∞	16	∞	ns	1.5V(2)
T_{CHCK}	CLKIN High Time	8	∞	10	∞	16	∞	ns	1.5V(2)
T_{CKHL}	CLKIN Fall Time		5		5		5	ns	3.5 to 1.0V
T_{CKLH}	CLKIN Rise Time		5		5		5	ns	1.0 to 3.5V
80C186XL CLKOUT TIMING									
T_{CICO}	CLKIN to CLKOUT Skew		17		17		21	ns	
T_{CLCL}	CLKOUT Period	40	∞	50		80	∞	ns	
T_{CLCH}	CLKOUT Low Time	0.5 T_{CLCL} − 5		0.5 T_{CLCL} − 5		0.5 T_{CLCL} − 5		ns	C_L = 100 pF(3)
T_{CHCL}	CLKOUT High Time	0.5 T_{CLCL} − 5		0.5 T_{CLCL} − 5		0.5 T_{CLCL} − 5		ns	C_L = 100 pF(4)
T_{CH1CH2}	CLKOUT Rise Time		6		8		10	ns	1.0 to 3.5V
T_{CL2CL1}	CLKOUT Fall Time		6		8		10	ns	3.5 to 1.0V

NOTES:
1. External clock applied to X1 and X2 not connected.
2. T_{CLCK} and T_{CHCK} (CLKIN Low and High times) should not have a duration less than 40% of T_{CKIN}.
3. Tested under worst case conditions: V_{CC} = 5.5V. T_A = 70°C.
4. Tested under worst case conditions: V_{CC} = 4.5V. T_A = 0°C.

AC SPECIFICATIONS (Continued)

READY, PERIPHERAL AND QUEUE STATUS TIMINGS

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where V_{IH} = V_{CC} − 0.5V.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL READY AND PERIPHERAL TIMING REQUIREMENTS (Listed More Than Once)									
T_{SRYCL}	Synchronous Ready (SRDY) Transition Setup Time(1)	8		10		15		ns	
T_{CLSRY}	SRDY Transition Hold Time(1)	8		10		15		ns	
T_{ARYCH}	ARDY Resolution Transition Setup Time(2)	8		10		15		ns	
T_{CLARX}	ARDY Active Hold Time(1)	8		10		15		ns	
T_{ARYCHL}	ARDY Inactive Holding Time	8		10		15		ns	
T_{ARYLCL}	Asynchronous Ready (ARDY) Setup Time(1)	10		15		25		ns	
T_{INVCH}	INTx, NMI, $\overline{TEST}$/BUSY, TMR IN Setup Time(2)	8		10		15		ns	
T_{INVCL}	DRQ0, DRQ1 Setup Time(2)	8		10		15		ns	
80C186XL PERIPHERAL AND QUEUE STATUS TIMING RESPONSES									
T_{CLTMV}	Timer Output Delay		17		22		33	ns	
T_{CHQSV}	Queue Status Delay		22		27		32	ns	

NOTES:
1. To guarantee proper operation.
2. To guarantee recognition at clock edge.

AC SPECIFICATIONS (Continued)

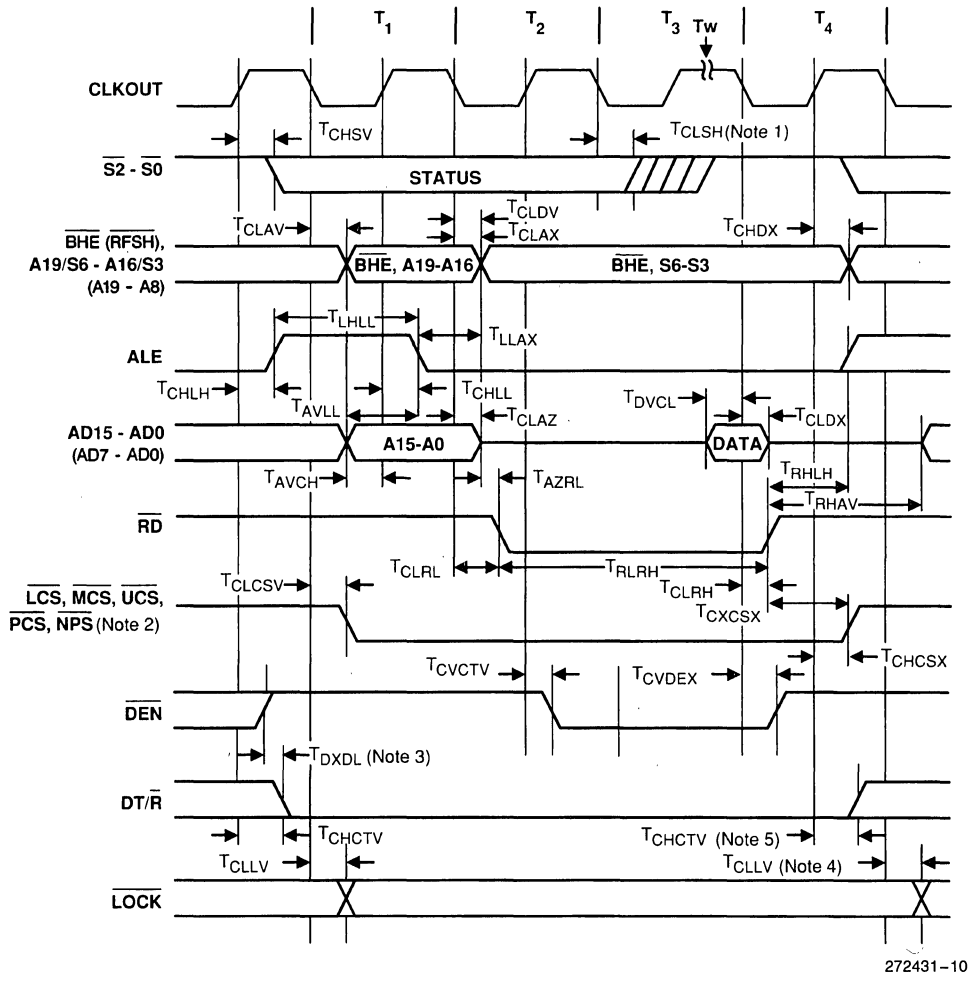
RESET AND HOLD/HLDA TIMINGS

T_A = 0°C to +70°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V and 50 pF loading on CLKOUT unless otherwise noted.
All output test conditions are with C_L = 50 pF.
For AC tests, input V_{IL} = 0.45V and V_{IH} = 2.4V except at X1 where V_{IH} = V_{CC} − 0.5V.

Symbol	Parameter	Values						Unit	Test Conditions
		80C186XL25		80C186XL20		80C186XL12			
		Min	Max	Min	Max	Min	Max		
80C186XL RESET AND HOLD/HLDA TIMING REQUIREMENTS									
T_{RESIN}	$\overline{RES}$ Setup	15		15		15		ns	
T_{HVCL}	HOLD Setup(1)	8		10		15		ns	
80C186XL GENERAL TIMING RESPONSES (Listed More Than Once)									
T_{CLAZ}	Address Float Delay	T_{CLAX}	20	T_{CLAX}	20	T_{CLAX}	25	ns	
T_{CLAV}	Address Valid Delay	3	20	3	22	3	36	ns	
80C186XL RESET AND HOLD/HLDA TIMING RESPONSES									
T_{CLRO}	Reset Delay		17		22		33	ns	
T_{CLHAV}	HLDA Valid Delay	3	17	3	22	3	33	ns	
T_{CHCZ}	Command Lines Float Delay		22		25		33	ns	
T_{CHCV}	Command Lines Valid Delay (after Float)		20		26		36	ns	

NOTE:
1. To guarantee recognition at next clock.

AC SPECIFICATIONS (Continued)

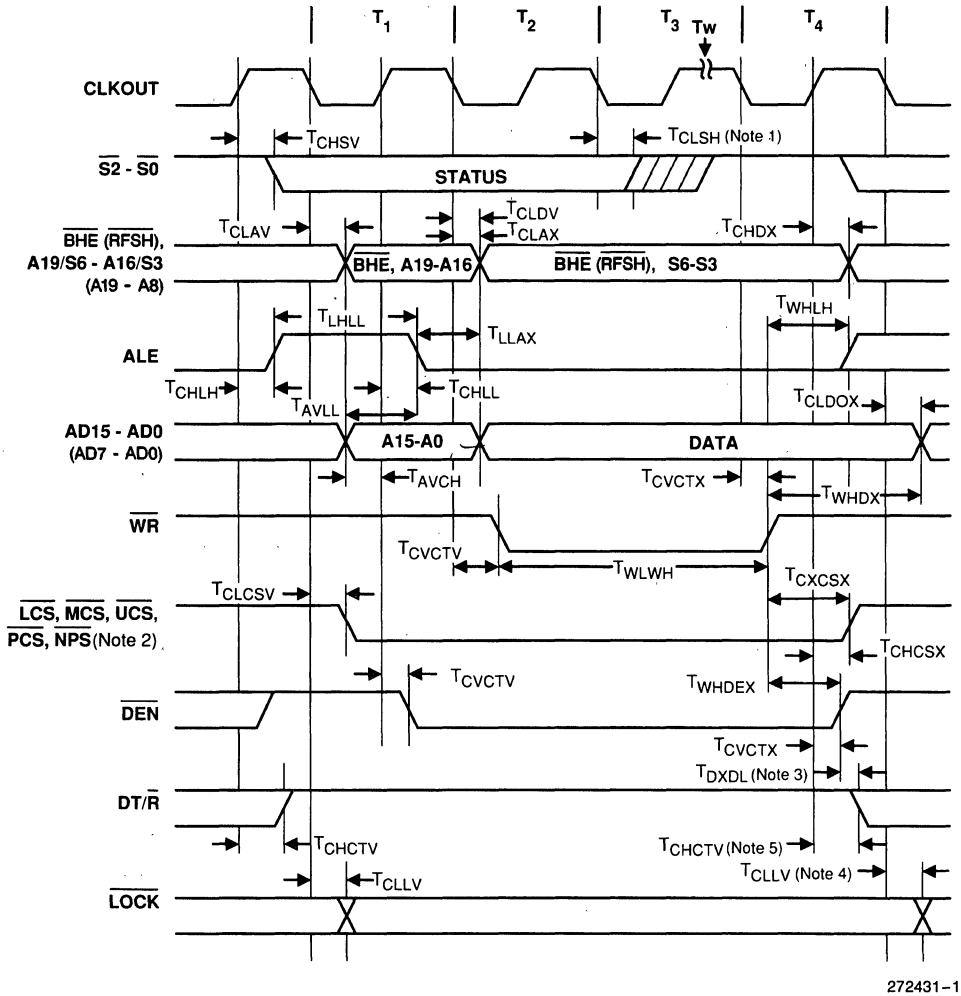


NOTES:
1. Status inactive in state preceding T_4.
2. If latched A_1 and A_2 are selected instead of $\overline{PCS5}$ and $\overline{PCS6}$, only T_{CLCSV} is applicable.
3. For write cycle followed by read cycle.
4. T_1 of next bus cycle.
5. Changes in T-state preceding next bus cycle if followed by write.

Pin names in parentheses apply to the 80C188XL.

Figure 6. Read Cycle Waveforms

AC SPECIFICATIONS (Continued)

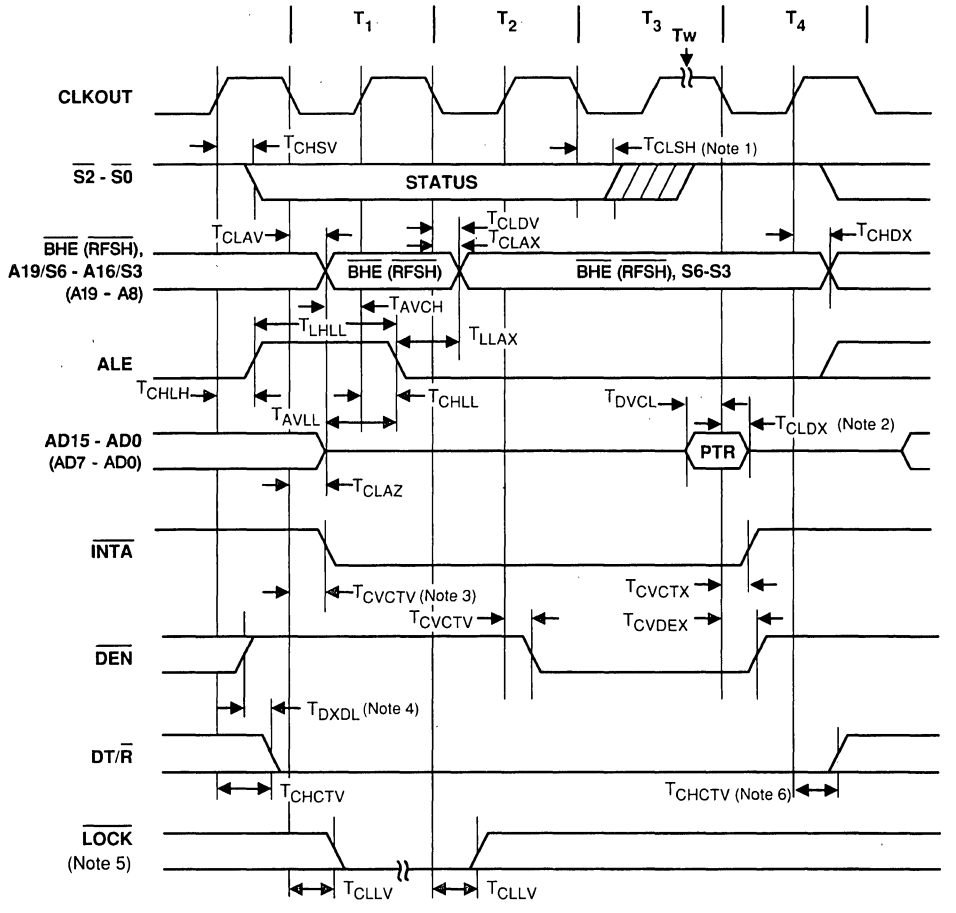


NOTES:
1. Status inactive in state preceding T_4.
2. If latched A_1 and A_2 are selected instead of $\overline{PCS5}$ and $\overline{PCS6}$, only T_{CLCSV} is applicable.
3. For write cycle followed by read cycle.
4. T_1 of next bus cycle.
5. Changes in T-state preceding next bus cycle if followed by read, INTA, or halt.

Pin names in parentheses apply to the 80C188XL.

Figure 7. Write Cycle Waveforms

AC SPECIFICATIONS (Continued)

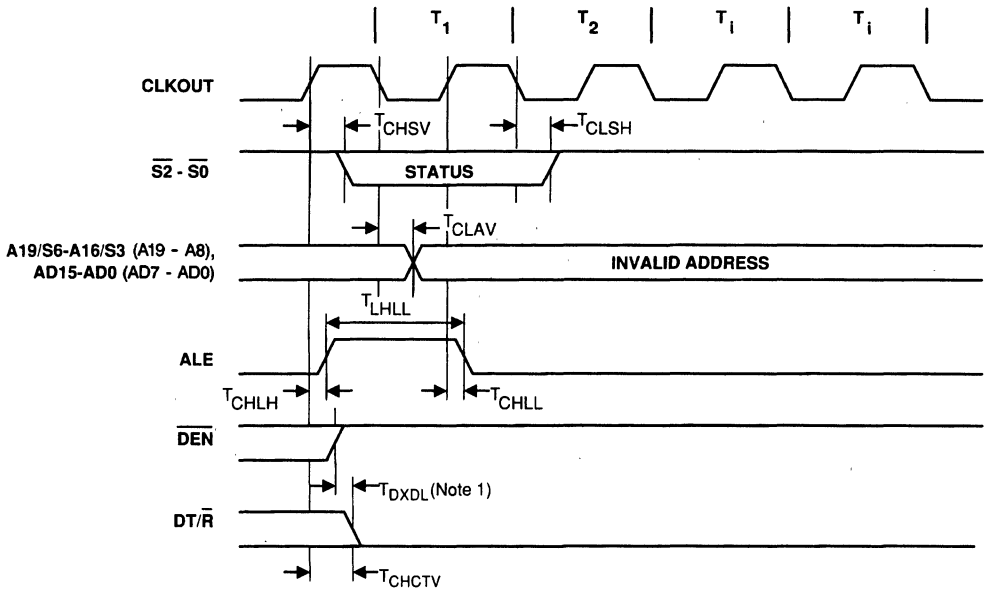


NOTES:
1. Status inactive in state preceding T_4.
2. The data hold time lasts only until $\overline{INTA}$ goes inactive, even if the $\overline{INTA}$ transition occurs prior to T_{CLDX} (min).
3. $\overline{INTA}$ occurs one clock later in Slave Mode.
4. For write cycle followed by interrupt acknowledge cycle.
5. $\overline{LOCK}$ is active upon T_1 of the first interrupt acknowledge cycle and inactive upon T_2 of the second interrupt acknowledge cycle.
6. Changes in T-state preceding next bus cycle if followed by write.

Pin names in parentheses apply to the 80C188XL.

Figure 8. Interrupt Acknowledge Cycle Waveforms

AC SPECIFICATIONS (Continued)



NOTE:
1. For write cycle followed by halt cycle.
Pin names in parentheses apply to the 80C188XL.

Figure 9. Software Halt Cycle Waveforms

WAVEFORMS

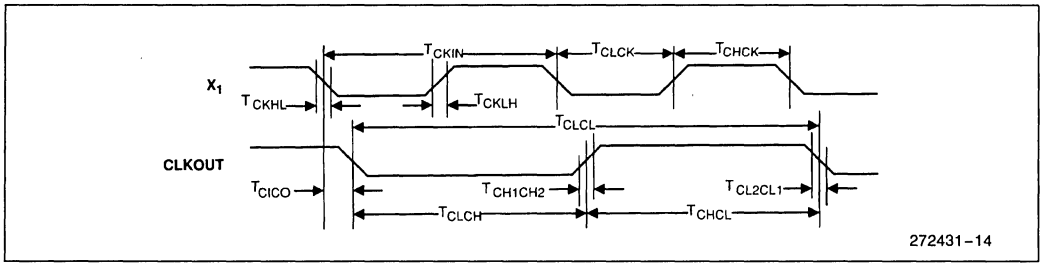


Figure 10. Clock Waveforms

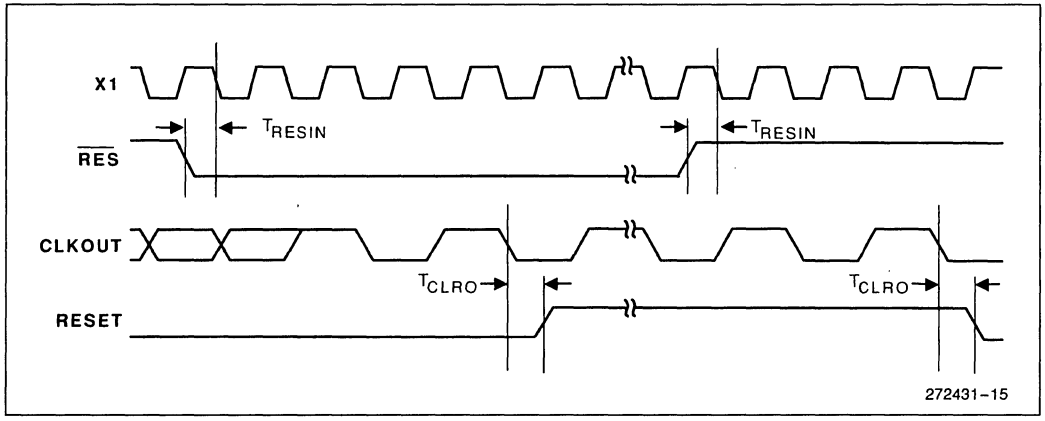


Figure 11. Reset Waveforms

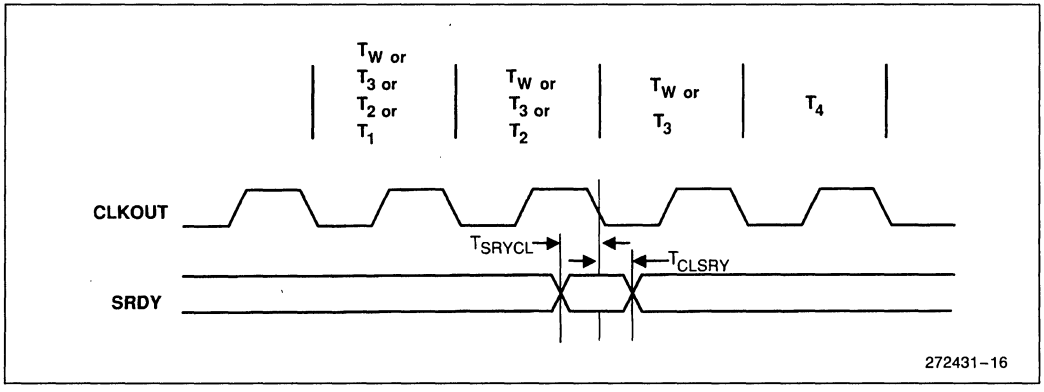


Figure 12. Synchronous Ready (SRDY) Waveforms

AC CHARACTERISTICS

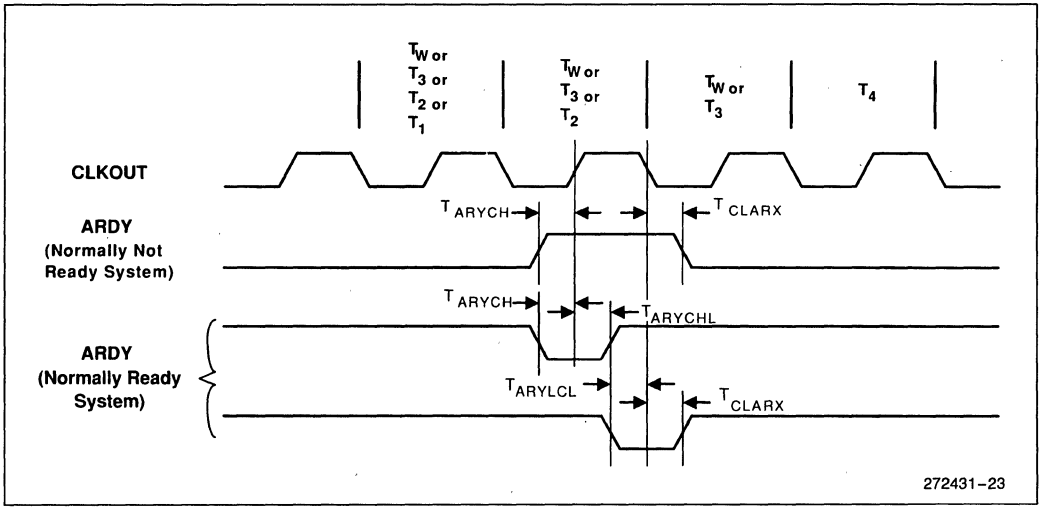


Figure 13. Asynchronous Ready (ARDY) Waveforms

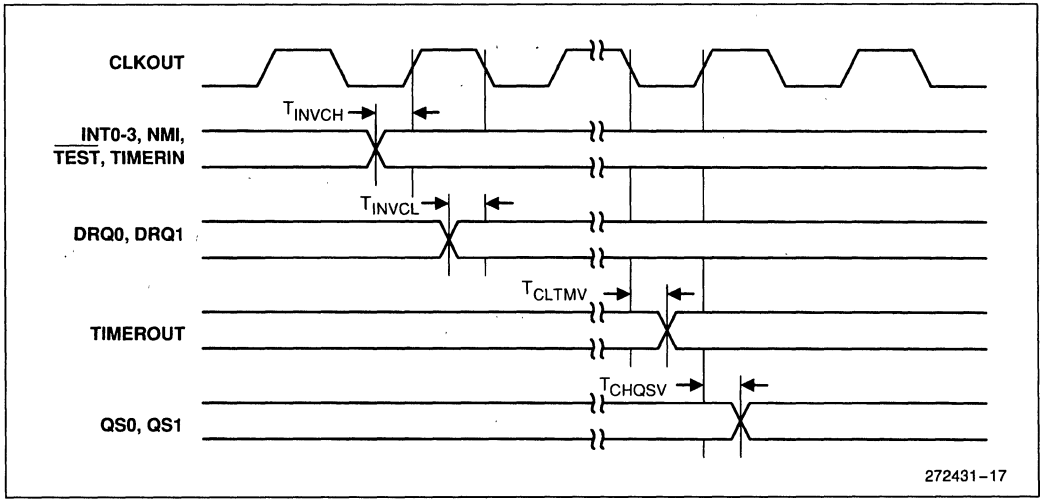


Figure 14. Peripheral and Queue Status Waveforms

AC CHARACTERISTICS (Continued)

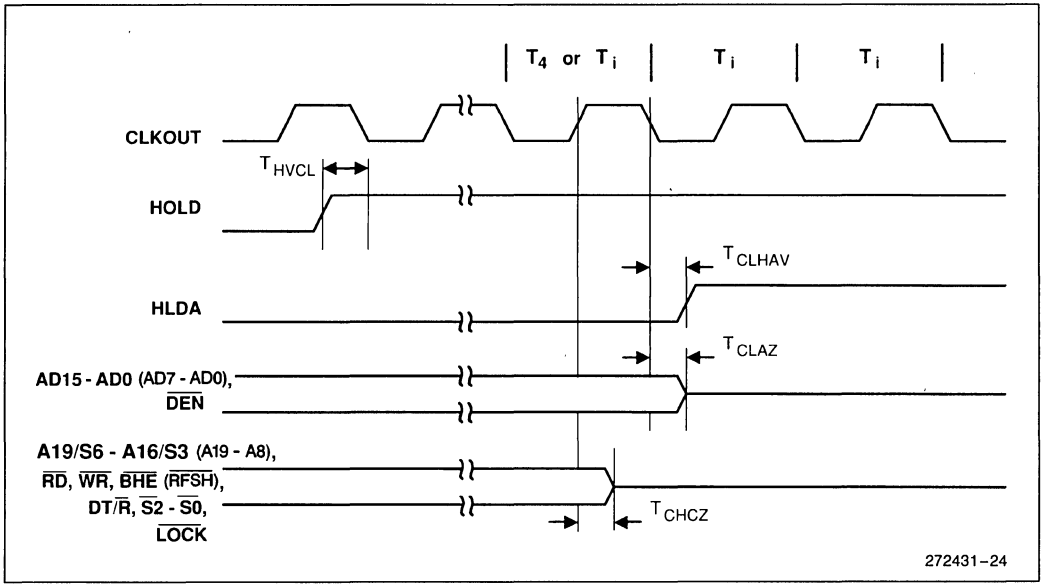


Figure 15. HOLDA/HLDA Waveforms (Entering Hold)

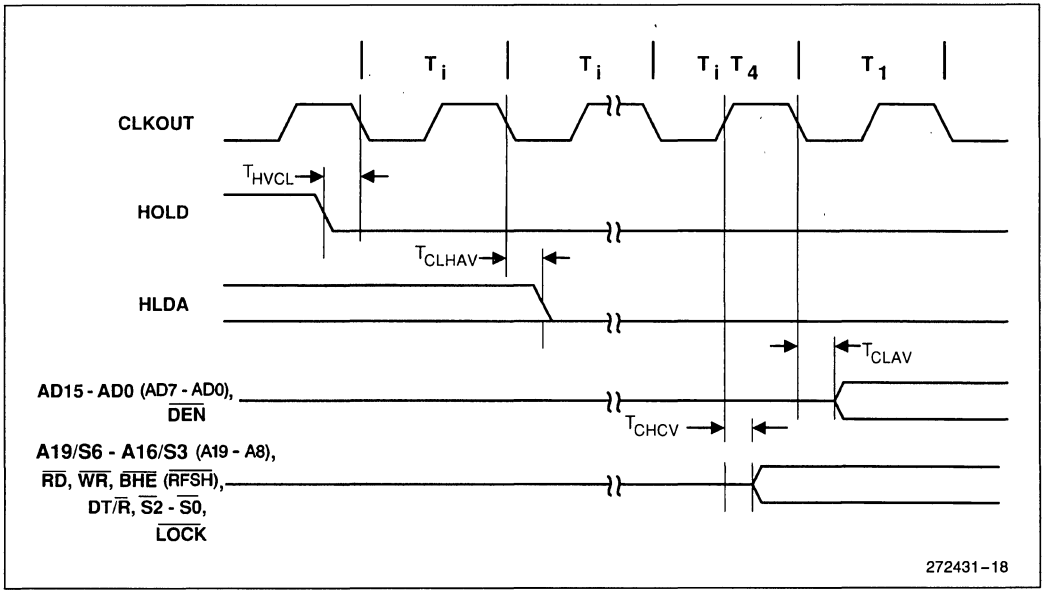


Figure 16. HOLD/HLDA Waveforms (Leaving Hold)

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has from 5 to 7 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
ARY: Asynchronous Ready Input
C: Clock Output
CK: Clock Input
CS: Chip Select
CT: Control (DT/$\overline{R}$, $\overline{DEN}$, . . .)
D: Data Input
DE: $\overline{DEN}$
H: Logic Level High
OUT: Input (DRQ0, TIM0, . . .)
L: Logic Level Low or ALE
O: Output
QS: Queue Status (QS1, QS2)
R: $\overline{RD}$ Signal, RESET Signal
S: Status ($\overline{S0}$, $\overline{S1}$, $\overline{S2}$)
SRY: Synchronous Ready Input
V: Valid
W: WR Signal
X: No Longer a Valid Logic Level
Z: Float

Examples:

T_{CLAV} — Time from Clock low to Address valid
T_{CHLH} — Time from Clock high to ALE high
T_{CLCSV} — Time from Clock low to Chip Select valid

DERATING CURVES

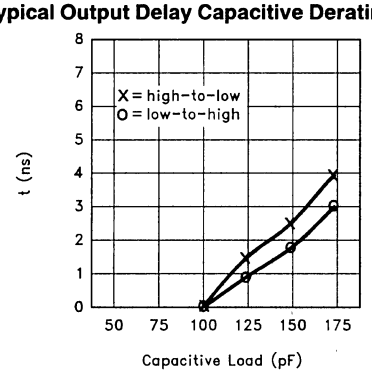


Figure 17. Capacitive Derating Curve

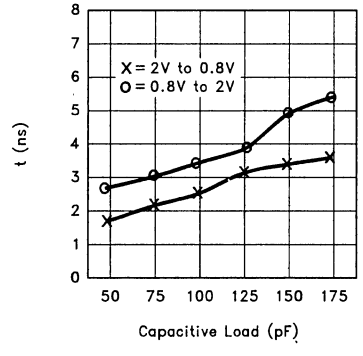


Figure 18. TTL Level Rise and Fall Times for Output Buffers

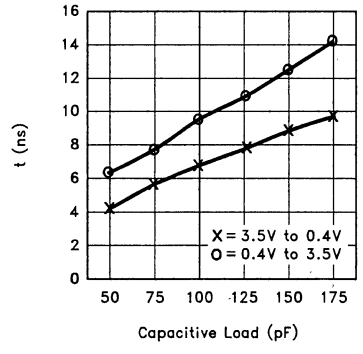


Figure 19. CMOS Level Rise and Fall Times for Output Buffers

80C186XL/80C188XL EXPRESS

The Intel EXPRESS system offers enhancements to the operational specifications of the 80C186XL microprocessor. EXPRESS products are designed to meet the needs of those applications whose operating requirements exceed commercial standards.

The 80C186XL EXPRESS program includes an extended temperature range. With the commercial standard temperature range, operational characteristics are guaranteed over the temperature range of 0°C to +70°C. With the extended temperature range option, operational characteristics are guaranteed over the range of −40°C to +85°C.

Package types and EXPRESS versions are identified by a one or two-letter prefix to the part number. The prefixes are listed in Table 10. All AC and DC specifications not mentioned in this section are the same for both commercial and EXPRESS parts.

Table 10. Prefix Identification

Prefix	Package Type	Temperature Range
A	PGA	Commercial
N	PLCC	Commercial
R	LCC	Commercial
S	QFP	Commercial
SB	SQFP	Commercial
TA	PGA	Extended
TN	PLCC	Extended
TR	LCC	Extended
TS	QFP	Extended

80C186XL/80C188XL EXECUTION TIMINGS

A determination of program execution timing must consider the bus cycles necessary to prefetch instructions as well as the number of execution unit cycles necessary to execute instructions. The following instruction timings represent the minimum execution time in clock cycles for each instruction. The timings given are based on the following assumptions:

- The opcode, along with any data or displacement required for execution of a particular instruction, has been prefetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDs occur.
- All word-data is located on even-address boundaries (80C186XL only).

All jumps and calls include the time required to fetch the opcode of the next instruction at the destination address.

All instructions which involve memory accesses can require one or two additional clocks above the minimum timings shown due to the asynchronous handshake between the bus interface unit (BIU) and execution unit.

With a 16-bit BIU, the 80C186XL has sufficient bus performance to ensure that an adequate number of prefetched bytes will reside in the queue (6 bytes) most of the time. Therefore, actual program execution time will not be substantially greater than that derived from adding the instruction timings shown.

The 80C188XL 8-bit BIU is limited in its performance relative to the execution unit. A sufficient number of prefetched bytes may not reside in the prefetch queue (4 bytes) much of the time. Therefore, actual program execution time will be substantially greater than that derived from adding the instruction timings shown.

INSTRUCTION SET SUMMARY

Function	Format				80C186XL Clock Cycles	80C188XL Clock Cycles	Comments
DATA TRANSFER							
MOV = Move:							
Register to Register/Memory	1000100w	mod reg r/m			2/12	2/12*	
Register/memory to register	1000101w	mod reg r/m			2/9	2/9*	
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1	12/13	12/13	8/16-bit
Immediate to register	1011w reg	data	data if w=1		3/4	3/4	8/16-bit
Memory to accumulator	1010000w	addr-low	addr-high		8	8*	
Accumulator to memory	1010001w	addr-low	addr-high		9	9*	
Register/memory to segment register	10001110	mod 0 reg r/m			2/9	2/13	
Segment register to register/memory	10001100	mod 0 reg r/m			2/11	2/15	
PUSH = Push:							
Memory	11111111	mod 1 1 0 r/m			16	20	
Register	01010 reg				10	14	
Segment register	000 reg 1 1 0				9	13	
Immediate	011010s0	data	data if s=0		10	14	
PUSHA = Push All	01100000				36	68	
POP = Pop:							
Memory	10001111	mod 0 0 0 r/m			20	24	
Register	01011 reg				10	14	
Segment register	000 reg 1 1 1	(reg≠01)			8	12	
POPA = Pop All	01100001				51	83	
XCHG = Exchange:							
Register/memory with register	1000011w	mod reg r/m			4/17	4/17*	
Register with accumulator	10010 reg				3	3	
IN = Input from:							
Fixed port	1110010w	port			10	10*	
Variable port	1110110w				8	8*	
OUT = Output to:							
Fixed port	1110011w	port			9	9*	
Variable port	1110111w				7	7*	
XLAT = Translate byte to AL	11010111				11	15	
LEA = Load EA to register	10001101	mod reg r/m			6	6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod≠11)		18	26	
LES = Load pointer to ES	11000100	mod reg r/m	(mod≠11)		18	26	
LAHF = Load AH with flags	10011111				2	2	
SAHF = Store AH into flags	10011110				3	3	
PUSHF = Push flags	10011100				9	13	
POPF = Pop flags	10011101				8	12	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186XL Clock Cycles	80C188XL Clock Cycles	Comments
DATA TRANSFER (Continued)							
SEGMENT = Segment Override:							
CS	00101110				2	2	
SS	00110110				2	2	
DS	00111110				2	2	
ES	00100110				2	2	
ARITHMETIC							
ADD = Add:							
Reg/memory with register to either	000000dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 000 r/m	data	data if s w=01	4/16	4/16*	
Immediate to accumulator	0000010w	data	data if w=1		3/4	3/4	8/16-bit
ADC = Add with carry:							
Reg/memory with register to either	000100dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 010 r/m	data	data if s w=01	4/16	4/16*	
Immediate to accumulator	0001010w	data	data if w=1		3/4	3/4	8/16-bit
INC = Increment:							
Register/memory	1111111w	mod 000 r/m			3/15	3/15*	
Register	01000 reg				3	3	
SUB = Subtract:							
Reg/memory and register to either	001010dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 101 r/m	data	data if s w=01	4/16	4/16*	
Immediate from accumulator	0010110w	data	data if w=1		3/4	3/4	8/16-bit
SBB = Subtract with borrow:							
Reg/memory and register to either	000110dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 011 r/m	data	data if s w=01	4/16	4/16*	
Immediate from accumulator	0001110w	data	data if w=1		3/4	3/4*	8/16-bit
DEC = Decrement							
Register/memory	1111111w	mod 001 r/m			3/15	3/15*	
Register	01001 reg				3	3	
CMP = Compare:							
Register/memory with register	0011101w	mod reg r/m			3/10	3/10*	
Register with register/memory	0011100w	mod reg r/m			3/10	3/10*	
Immediate with register/memory	100000sw	mod 111 r/m	data	data if s w=01	3/10	3/10*	
Immediate with accumulator	0011110w	data	data if w=1		3/4	3/4	8/16-bit
NEG = Change sign register/memory	1111011w	mod 011 r/m			3/10	3/10*	
AAA = ASCII adjust for add	00110111				8	8	
DAA = Decimal adjust for add	00100111				4	4	
AAS = ASCII adjust for subtract	00111111				7	7	
DAS = Decimal adjust for subtract	00101111				4	4	
MUL = Multiply (unsigned):	1111011w	mod 100 r/m					
Register-Byte					26-28	26-28	
Register-Word					35-37	35-37	
Memory-Byte					32-34	32-34	
Memory-Word					41-43	41-43*	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186XL Clock Cycles	80C188XL Clock Cycles	Comments
ARITHMETIC (Continued)							
IMUL = Integer multiply (signed):	1 1 1 1 0 1 1 w	mod 1 0 1 r/m					
Register-Byte					25–28	25–28	
Register-Word					34–37	34–37	
Memory-Byte					31–34	32–34	
Memory-Word					40–43	40–43*	
IMUL = Integer Immediate multiply (signed)	0 1 1 0 1 0 s 1	mod reg r/m	data	data if s=0	22–25/ 29–32	22–25/ 29–32	
DIV = Divide (unsigned):	1 1 1 1 0 1 1 w	mod 1 1 0 r/m					
Register-Byte					29	29	
Register-Word					38	38	
Memory-Byte					35	35	
Memory-Word					44	44*	
IDIV = Integer divide (signed):	1 1 1 1 0 1 1 w	mod 1 1 1 r/m					
Register-Byte					44–52	44-52	
Register-Word					53–61	53–61	
Memory-Byte					50–58	50–58	
Memory-Word					59–67	59–67*	
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0			19	19	
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0			15	15	
CBW = Convert byte to word	1 0 0 1 1 0 0 0				2	2	
CWD = Convert word to double word	1 0 0 1 1 0 0 1				4	4	
LOGIC							
Shift/Rotate Instructions:							
Register/Memory by 1	1 1 0 1 0 0 0 w	mod TTT r/m			2/15	2/15	
Register/Memory by CL	1 1 0 1 0 0 1 w	mod TTT r/m			5+n/17+n	5+n/17+n	
Register/Memory by Count	1 1 0 0 0 0 0 w	mod TTT r/m	count		5+n/17+n	5+n/17+n	
	TTT Instruction 0 0 0 ROL 0 0 1 ROR 0 1 0 RCL 0 1 1 RCR 1 0 0 SHL/SAL 1 0 1 SHR 1 1 1 SAR						
AND = **And:**							
Reg/memory and register to either	0 0 1 0 0 0 d w	mod reg r/m			3/10	3/10*	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1	4/16	4/16*	
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1		3/4	3/4*	8/16-bit
TEST = **And function to flags, no result:**							
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m			3/10	3/10*	
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1	4/10	4/10*	
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1		3/4	3/4	8/16-bit
OR = **Or:**							
Reg/memory and register to either	0 0 0 0 1 0 d w	mod reg r/m			3/10	3/10*	
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1	4/16	4/16*	
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1		3/4	3/4*	8/16-bit

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186XL Clock Cycles	80C188XL Clock Cycles	Comments
LOGIC (Continued)				
XOR = Exclusive or:				
Reg/memory and register to either	001100dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod 110 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0011010w | data | data if w=1	3/4	3/4	8/16-bit
NOT = Invert register/memory	1111011w | mod 010 r/m	3/10	3/10*	
STRING MANIPULATION				
MOVS = Move byte/word	1010010w	14	14*	
CMPS = Compare byte/word	1010011w	22	22*	
SCAS = Scan byte/word	1010111w	15	15*	
LODS = Load byte/wd to AL/AX	1010110w	12	12*	
STOS = Store byte/wd from AL/AX	1010101w	10	10*	
INS = Input byte/wd from DX port	0110110w	14	14	
OUTS = Output byte/wd to DX port	0110111w	14	14	
Repeated by count in CX (REP/REPE/REPZ/REPNE/REPNZ)				
MOVS = Move string	11110010 | 1010010w	8+8n	8+8n*	
CMPS = Compare string	1111001z | 1010011w	5+22n	5+22n*	
SCAS = Scan string	1111001z | 1010111w	5+15n	5+15n*	
LODS = Load string	11110010 | 1010110w	6+11n	6+11n*	
STOS = Store string	11110010 | 1010101w	6+9n	6+9n*	
INS = Input string	11110010 | 0110110w	8+8n	8+8n*	
OUTS = Output string	11110010 | 0110111w	8+8n	8+8n*	
CONTROL TRANSFER				
CALL = Call:				
Direct within segment	11101000 | disp-low | disp-high	15	19	
Register/memory indirect within segment	11111111 | mod 010 r/m	13/19	17/27	
Direct intersegment	10011010 | segment offset segment selector	23	31	
Indirect intersegment	11111111 | mod 011 r/m (mod ≠ 11)	38	54	
JMP = Unconditional jump:				
Short/long	11101011 | disp-low	14	14	
Direct within segment	11101001 | disp-low | disp-high	14	14	
Register/memory indirect within segment	11111111 | mod 100 r/m	11/17	11/21	
Direct intersegment	11101010 | segment offset segment selector	14	14	
Indirect intersegment	11111111 | mod 101 r/m (mod ≠ 11)	26	34	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186XL Clock Cycles	80C188XL Clock Cycles	Comments
CONTROL TRANSFER (Continued)							
RET = **Return from CALL:**							
Within segment	11000011				16	20	
Within seg adding immed to SP	11000010	data-low	data-high		18	22	
Intersegment	11001011				22	30	
Intersegment adding immediate to SP	11001010	data-low	data-high		25	33	
JE/JZ = Jump on equal/zero	01110100	disp			4/13	4/13	JMP not taken/JMP taken
JL/JNGE = Jump on less/not greater or equal	01111100	disp			4/13	4/13	
JLE/JNG = Jump on less or equal/not greater	01111110	disp			4/13	4/13	
JB/JNAE = Jump on below/not above or equal	01110010	disp			4/13	4/13	
JBE/JNA = Jump on below or equal/not above	01110110	disp			4/13	4/13	
JP/JPE = Jump on parity/parity even	01111010	disp			4/13	4/13	
JO = Jump on overflow	01110000	disp			4/13	4/13	
JS = Jump on sign	01111000	disp			4/13	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101	disp			4/13	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101	disp			4/13	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111	disp			4/13	4/13	
JNB/JAE = Jump on not below/above or equal	01110011	disp			4/13	4/13	
JNBE/JA = Jump on not below or equal/above	01110111	disp			4/13	4/13	
JNP/JPO = Jump on not par/par odd	01111011	disp			4/13	4/13	
JNO = Jump on not overflow	01110001	disp			4/13	4/13	
JNS = Jump on not sign	01111001	disp			4/13	4/13	
JCXZ = Jump on CX zero	11100011	disp			5/15	5/15	
LOOP = Loop CX times	11100010	disp			6/16	6/16	LOOP not taken/LOOP taken
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp			6/16	6/16	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp			6/16	6/16	
ENTER = Enter Procedure	11001000	data-low	data-high	L			
L = 0					15	19	
L = 1					25	29	
L > 1					22+16(n−1)	26+20(n−1)	
LEAVE = Leave Procedure	11001001				8	8	
INT = **Interrupt:**							
Type specified	11001101	type			47	47	
Type 3	11001100				45	45	if INT. taken/ if INT. not taken
INTO = Interrupt on overflow	11001110				48/4	48/4	
IRET = Interrupt return	11001111				28	28	
BOUND = Detect value out of range	01100010	mod reg r/m			33–35	33–35	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186XL Clock Cycles	80C188XL Clock Cycles	Comments
PROCESSOR CONTROL				
CLC = Clear carry	11111000	2	2	
CMC = Complement carry	11110101	2	2	
STC = Set carry	11111001	2	2	
CLD = Clear direction	11111100	2	2	
STD = Set direction	11111101	2	2	
CLI = Clear interrupt	11111010	2	2	
STI = Set interrupt	11111011	2	2	
HLT = Halt	11110100	2	2	
WAIT = Wait	10011011	6	6	if $\overline{TEST}$ = 0
LOCK = Bus lock prefix	11110000	2	2	
NOP = No Operation	10010000	3	3	
	(TTT LLL are opcode to processor extension)			

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

The Effective Address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

EA calculation time is 4 clock cycles for all modes, and is included in the execution times given whenever appropriate.

Segment Override Prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

REVISION HISTORY

This data sheet replaces the following data sheets:

- 272031-002 80C186XL
- 270975-002 80C188XL
- 272309-001 SB80C186XL
- 272310-001 SB80C188XL

ERRATA

An A or B step 80C186XL/80C188XL has the following errata. The A or B step 80C186XL/80C188XL can be identified by the presence of an "A" or "B" alpha character, respectively, next to the FPO number. The FPO number location is shown in Figure 4.

1. An internal condition with the interrupt controller can cause no acknowledge cycle on the INTA1 line in response to INT1. This errata only occurs when Interrupt 1 is configured in cascade mode and a higher priority interrupt exists. This errata will not occur consistently, it is dependent on interrupt timing.

The C step 80C186XL/80C188XL has no known errata. The C step can be identified by the presence of a "C" or "D" alpha character next to the FPO number. The FPO number location is shown in Figure 4.

PRODUCT IDENTIFICATION

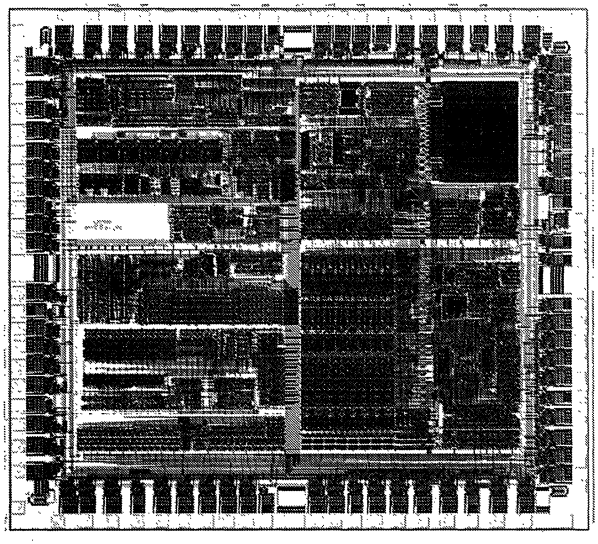
Intel 80C186XL devices are marked with a 9-character alphanumeric Intel FPO number underneath the product number. This data sheet (272431-001) is valid for devices with an "A", "B", "C", or "D" as the ninth character in the FPO number, as illustrated in Figure 4.

80C186EA/80C188EA AND 80L186EA/80L188EA 16-BIT HIGH-INTEGRATION EMBEDDED PROCESSORS

- 80C186 Upgrade for Power Critical Applications
- Fully Static Operation
- True CMOS Inputs and Outputs

- Integrated Feature Set
  - Static 186 CPU Core
  - Power Save, Idle and Powerdown Modes
  - Clock Generator
  - 2 Independent DMA Channels
  - 3 Programmable 16-Bit Timers
  - Dynamic RAM Refresh Control Unit
  - Programmable Memory and Peripheral Chip Select Logic
  - Programmable Wait State Generator
  - Local Bus Controller
  - System-Level Testing Support (High Impedance Test Mode)
- Speed Versions Available (5V):
  - 25 MHz (80C186EA25/80C188EA25)
  - 20 MHz (80C186EA20/80C188EA20)
  - 13 MHz (80C186EA13/80C188EA13)
- Speed Versions Available (3V):
  - 13 MHz (80L186EA13/80L188EA13)
  - 8 MHz (80L186EA8/80L188EA8)
- Direct Addressing Capability to 1 Mbyte Memory and 64 Kbyte I/O
- Supports 80C187 Numeric Coprocessor Interface (80C186EA only)
- Available in the Following Packages:
  - 68-Pin Plastic Leaded Chip Carrier (PLCC)
  - 80-Pin EIAJ Quad Flat Pack (QFP)
  - 80-Pin Shrink Quad Flat Pack (SQFP)
- Available in Extended Temperature Range (−40°C to +85°C)

The 80C186EA is a CHMOS high integration embedded microprocessor. The 80C186EA includes all of the features of an "Enhanced Mode" 80C186 while adding the additional capabilities of Idle and Powerdown Modes. In Numerics Mode, the 80C186EA interfaces directly with an 80C187 Numerics Coprocessor.



272432-1

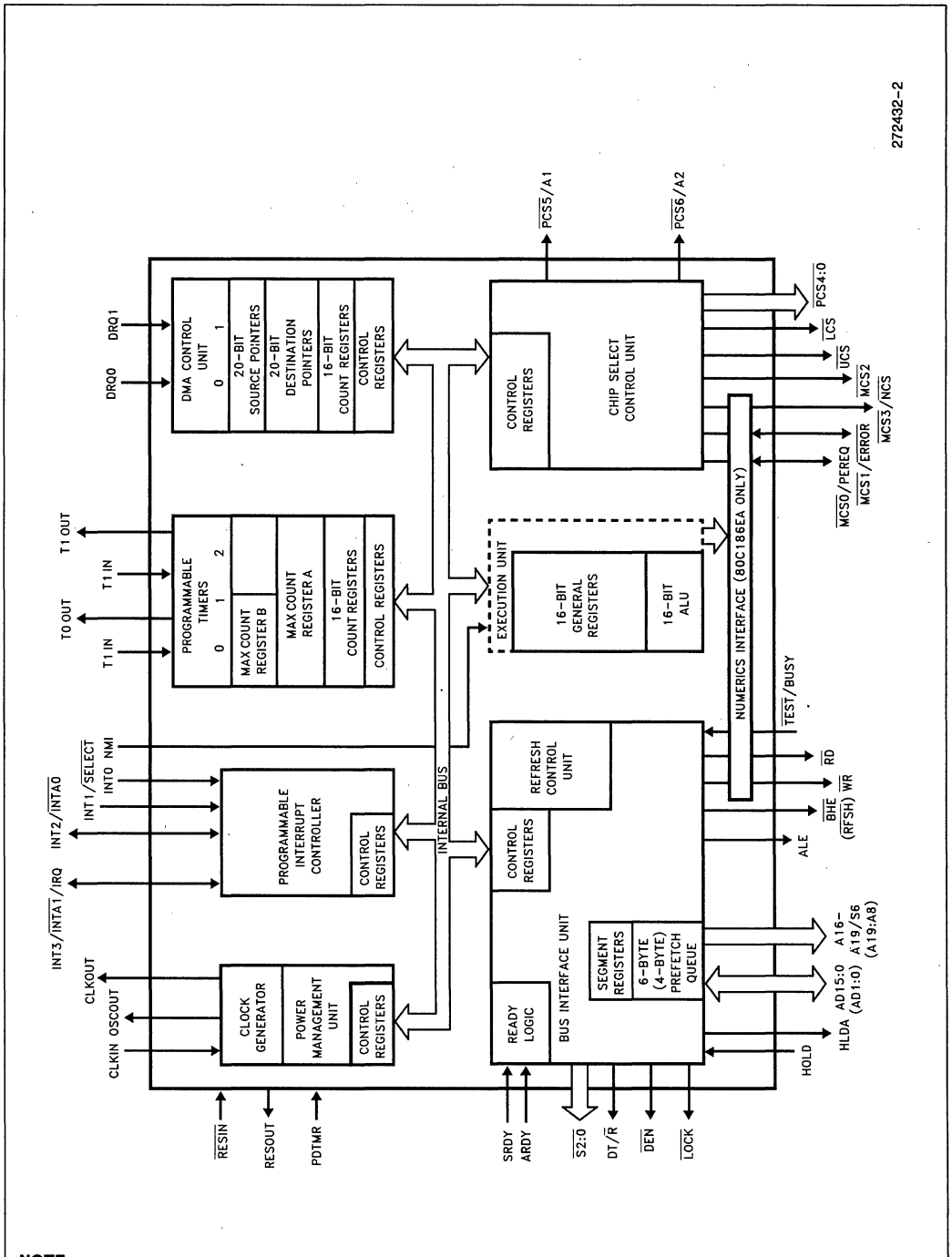
October 1995
Order Number: 272432-003

80C186EA/80C188EA AND 80L186EA/80L188EA 16-Bit High Integration Embedded Processor

CONTENTS

CONTENTS

1



NOTE:
Pin names in parentheses apply to the 80C186EA/80L188EA

Figure 1. 80C186EA/80C188EA Block Diagram

INTRODUCTION

Unless specifically noted, all references to the 80C186EA apply to the 80C188EA, 80L186EA, and 80L188EA. References to pins that differ between the 80C186EA/80L186EA and the 80C188EA/80L188EA are given in parentheses. The "L" in the part number denotes low voltage operation. Physically and functionally, the "C" and "L" devices are identical.

The 80C186EA is the second product in a new generation of low-power, high-integration microprocessors. It enhances the existing 80C186XL family by offering new features and operating modes. The 80C186EA is object code compatible with the 80C186XL embedded processor.

The 80L186EA is the 3V version of the 80C186EA. The 80L186EA is functionally identical to the 80C186EA embedded processor. Current 80C186EA customers can easily upgrade their designs to use the 80L186EA and benefit from the reduced power consumption inherent in 3V operation.

The feature set of the 80C186EA/80L186EA meets the needs of low-power, space-critical applications. Low-power applications benefit from the static design of the CPU core and the integrated peripherals as well as low voltage operation. Minimum current consumption is achieved by providing a Powerdown Mode that halts operation of the device, and freezes the clock circuits. Peripheral design enhancements ensure that non-initialized peripherals consume little current.

Space-critical applications benefit from the integration of commonly used system peripherals. Two flexible DMA channels perform CPU-independent data transfers. A flexible chip select unit simplifies memory and peripheral interfacing. The interrupt unit provides sources for up to 128 external interrupts and will prioritize these interrupts with those generated from the on-chip peripherals. Three general purpose timer/counters round out the feature set of the 80C186EA.

Figure 1 shows a block diagram of the 80C186EA/80C188EA. The Execution Unit (EU) is an enhanced 8086 CPU core that includes: dedicated hardware to speed up effective address calculations, enhance execution speed for multiple-bit shift and rotate instructions and for multiply and divide instructions, string move instructions that operate at full bus bandwidth, ten new instructions, and static operation. The Bus Interface Unit (BIU) is the same as that found on the original 80C186 family products. An independent internal bus is used to allow communication between the BIU and internal peripherals.

80C186EA CORE ARCHITECTURE

Bus Interface Unit

The 80C186EA core incorporates a bus controller that generates local bus control signals. In addition, it employs a HOLD/HLDA protocol to share the local bus with other bus masters.

The bus controller is responsible for generating 20 bits of address, read and write strobes, bus cycle status information and data (for write operations) information. It is also responsible for reading data off the local bus during a read operation. SRDY and ARDY input pins are provided to extend a bus cycle beyond the minimum four states (clocks).

The local bus controller also generates two control signals ($\overline{DEN}$ and DT/$\overline{R}$) when interfacing to external transceiver chips. This capability allows the addition of transceivers for simple buffering of the mulitplexed address/data bus.

Clock Generator

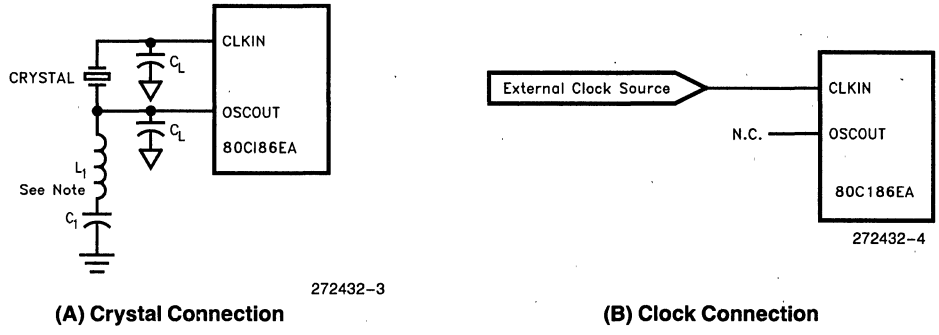
The processor provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter, and two low-power operating modes.

The oscillator circuit is designed to be used with either a **parallel resonant** fundamental or third-overtone mode crystal network. Alternatively, the oscillator circuit may be driven from an external clock source. Figure 2 shows the various operating modes of the oscillator circuit.

The crystal or clock frequency chosen must be twice the required processor operating frequency due to the internal divide-by-two counter. This counter is used to drive all internal phase clocks and the external CLKOUT signal. CLKOUT is a 50% duty cycle processor clock and can be used to drive other system components. All AC timings are referenced to CLKOUT.

The following parameters are recommended when choosing a crystal:

Temperature Range:	Application Specific
ESR (Equivalent Series Resistance):	60Ω max
C0 (Shunt Capacitance of Crystal):	7.0 pF max
C_L (Load Capacitance):	20 pF ± 2 pF
Drive Level:	2 mW max



(A) Crystal Connection **(B) Clock Connection**

NOTE:
The L_1C_1 network is **only** required when using a third-overtone crystal.

Figure 2. Clock Configurations

80C186EA PERIPHERAL ARCHITECTURE

The 80C186EA has integrated several common system peripherals with a CPU core to create a compact, yet powerful system. The integrated peripherals are designed to be flexible and provide logical interconnections between supporting units (e.g., the interrupt control unit supports interrupt requests from the timer/counters or DMA channels).

The list of integrated peripherals include:

- 4-Input Interrupt Control Unit
- 3-Channel Timer/Counter Unit
- 2-Channel DMA Unit
- 13-Output Chip-Select Unit
- Refresh Control Unit
- Power Management logic

The registers associated with each integrated peripheral are contained within a 128 x 16 register file called the Peripheral Control Block (PCB). The PCB can be located in either memory or I/O space on any 256 byte address boundary.

Figure 3 provides a list of the registers associated with the PCB when the processor's Interrupt Control Unit is in Master Mode. In Slave Mode, the definitions of some registers change. Figure 4 provides register definitions specific to Slave Mode.

Interrupt Control Unit

The 80C186EA can receive interrupts from a number of sources, both internal and external. The Interrupt Control Unit (ICU) serves to merge these requests on a priority basis, for individual service by the CPU. Each interrupt source can be independently masked by the Interrupt Control Unit or all interrupts can be globally masked by the CPU.

Internal interrupt sources include the Timers and DMA channels. External interrupt sources come from the four input pins INT3:0. The NMI interrupt pin is not controlled by the ICU and is passed directly to the CPU. Although the timers only have one request input to the ICU, separate vector types are generated to service individual interrupts within the Timer Unit.

Timer/Counter Unit

The 80C186EA Timer/Counter Unit (TCU) provides three 16-bit programmable timers. Two of these are highly flexible and are connected to external pins for control or clocking. A third timer is not connected to any external pins and can only be clocked internally. However, it can be used to clock the other two timer channels. The TCU can be used to count external events, time external events, generate non-repetitive waveforms, generate timed interrupts, etc.

PCB Offset	Function
00H	Reserved
02H	Reserved
04H	Reserved
06H	Reserved
08H	Reserved
0AH	Reserved
0CH	Reserved
0EH	Reserved
10H	Reserved
12H	Reserved
14H	Reserved
16H	Reserved
18H	Reserved
1AH	Reserved
1CH	Reserved
1EH	Reserved
20H	Reserved
22H	End of Interrupt
24H	Poll
26H	Poll Status
28H	Interrupt Mask
2AH	Priority Mask
2CH	In-Service
2EH	Interrupt Request
30H	Interrupt Status
32H	Timer Control
34H	DMA0 Int. Control
36H	DMA1 Int. Control
38H	INT0 Control
3AH	INT1 Control
3CH	INT2 Control
3EH	INT3 Control

PCB Offset	Function
40H	Reserved
42H	Reserved
44H	Reserved
46H	Reserved
48H	Reserved
4AH	Reserved
4CH	Reserved
4EH	Reserved
50H	Timer 0 Count
52H	Timer 0 Compare A
54H	Timer 0 Compare B
56H	Timer 0 Control
58H	Timer 1 Count
5AH	Timer 1 Compare A
5CH	Timer 1 Compare B
5EH	Timer 1 Control
60H	Timer 2 Count
62H	Timer 2 Compare
64H	Reserved
66H	Timer 2 Control
68H	Reserved
6AH	Reserved
6CH	Reserved
6EH	Reserved
70H	Reserved
72H	Reserved
74H	Reserved
76H	Reserved
78H	Reserved
7AH	Reserved
7CH	Reserved
7EH	Reserved

PCB Offset	Function
80H	Reserved
82H	Reserved
84H	Reserved
86H	Reserved
88H	Reserved
8AH	Reserved
8CH	Reserved
8EH	Reserved
90H	Reserved
92H	Reserved
94H	Reserved
96H	Reserved
98H	Reserved
9AH	Reserved
9CH	Reserved
9EH	Reserved
A0H	UMCS
A2H	LMCS
A4H	PACS
A6H	MMCS
A8H	MPCS
AAH	Reserved
ACH	Reserved
AEH	Reserved
B0H	Reserved
B2H	Reserved
B4H	Reserved
B6H	Reserved
B8H	Reserved
BAH	Reserved
BCH	Reserved
BEH	Reserved

PCB Offset	Function
C0H	DMA0 Src. Lo
C2H	DMA0 Src. Hi
C4H	DMA0 Dest. Lo
C6H	DMA0 Dest. Hi
C8H	DMA0 Count
CAH	DMA0 Control
CCH	Reserved
CEH	Reserved
D0H	DMA1 Src. Lo
D2H	DMA1 Src. Hi
D4H	DMA1 Dest. Lo
D6H	DMA1 Dest. Hi
D8H	DMA1 Count
DAH	DMA1 Control
DCH	Reserved
DEH	Reserved
E0H	Refresh Base
E2H	Refresh Time
E4H	Refresh Control
E6H	Reserved
E8H	Reserved
EAH	Reserved
ECH	Reserved
EEH	Reserved
F0H	Power-Save
F2H	Power Control
F4H	Reserved
F6H	Step ID
F8H	Reserved
FAH	Reserved
FCH	Reserved
FEH	Relocation

Figure 3. Peripheral Control Block Registers

PCB Offset	Function
20H	Interrupt Vector
22H	Specific EOI
24H	Reserved
26H	Reserved
28H	Interrupt Mask
2AH	Priority Mask
2C	In-Service
2E	Interrupt Request
30	Interrupt Status
32	TMR0 Interrupt Control
34	DMA0 Interrupt Control
36	DMA1 Interrupt Control
38	TMR1 Interrupt Control
3A	TMR2 Interrupt Control
3C	Reserved
3E	Reserved

Figure 4. 80C186EA Slave Mode Peripheral Control Block Registers

DMA Control Unit

The 80C186EA DMA Contol Unit provides two independent high-speed DMA channels. Data transfers can occur between memory and I/O space in any combination: memory to memory, memory to I/O, I/O to I/O or I/O to memory. Data can be transferred either in bytes or words. Transfers may proceed to or from either even or odd addresses, but even-aligned word transfers proceed at a faster rate. Each data transfer consumes two bus cycles (a minimum of eight clocks), one cycle to fetch data and the other to store data. The chip-select/ready logic may be programmed to point to the memory or I/O space subject to DMA transfers in order to provide hardware chip select lines. DMA cycles run at higher priority than general processor execution cycles.

Chip-Select Unit

The 80C186EA Chip-Select Unit integrates logic which provides up to 13 programmable chip-selects to access both memories and peripherals. In addition, each chip-select can be programmed to automatically terminate a bus cycle independent of the condition of the SRDY and ARDY input pins. The chip-select lines are available for all memory and I/O bus cycles, whether they are generated by the CPU, the DMA unit, or the Refresh Control Unit.

Refresh Control Unit

The Refresh Control Unit (RCU) automatically generates a periodic memory read bus cycle to keep dynamic or pseudo-static memory refreshed. A 9-bit counter controls the number of clocks between refresh requests.

A 9-bit address generator is maintained by the RCU with the address presented on the A9:1 address lines during the refresh bus cycle. Address bits A19:13 are programmable to allow the refresh address block to be located on any 8 Kbyte boundary.

Power Management

The 80C186EA has three operational modes to control the power consumption of the device. They are Power Save Mode, Idle Mode, and Powerdown Mode.

Power Save Mode divides the processor clock by a programmable value to take advantage of the fact that current is linearly proportional to frequency. An unmasked interrupt, NMI, or reset will cause the 80C186EA to exit Power Save Mode.

Idle Mode freezes the clocks of the Execution Unit and the Bus Interface Unit at a logic zero state while all peripherals operate normally.

Powerdown Mode freezes all internal clocks at a logic zero level and disables the crystal oscillator. All internal registers hold their values provided V_{CC} is maintained. Current consumption is reduced to transistor leakage only.

80C187 Interface (80C186EA Only)

The 80C187 Numerics Coprocessor may be used to extend the 80C186EA instruction set to include floating point and advanced integer instructions. Connecting the 80C186EA RESOUT and $\overline{TEST}$/BUSY pins to the 80C187 enables Numerics Mode operation. In Numerics Mode, three of the four Mid-Range Chip Select ($\overline{MCS}$) pins become handshaking pins for the interface. The exchange of data and control information proceeds through four dedicated I/O ports.

If an 80C187 is not present, the 80C186EA configures itself for regular operation at reset.

NOTE:
The 80C187 is not specified for 3V operation and therefore does not interface directly to the 80L186EA.

ONCE Test Mode

To facilitate testing and inspection of devices when fixed into a target system, the 80C186EA has a test mode available which forces all output and input/output pins to be placed in the high-impedance state. ONCE stands for "ON Circuit Emulation". The ONCE mode is selected by forcing the $\overline{UCS}$ and $\overline{LCS}$ pins LOW (0) during a processor reset (these pins are weakly held to a HIGH (1) level) while $\overline{RESIN}$ is active.

DIFFERENCES BETWEEN THE 80C186XL AND THE 80C186EA

The 80C186EA is intended as a direct functional upgrade for 80C186XL designs. In many cases, it will be possible to replace an existing 80C186XL with little or no hardware redesign. The following sections describe differences in pinout, operating modes, and AC and DC specifications to keep in mind.

Pinout Compatibility

The 80C186EA requires a PDTMR pin to time the processor's exit from Powerdown Mode. The original pin arrangement for the 80C186XL in the PLCC package did not have any spare leads to use for PDTMR, so the DT/$\overline{R}$ pin was sacrificed. The arrangement of all the other leads in the 68-lead PLCC is identical between the 80C186XL and the 80C186EA. DT/$\overline{R}$ may be synthesized by latching the $\overline{S1}$ status output. Therefore, upgrading a PLCC 80C186XL to PLCC 80C186EA is straightforward.

The 80-lead QFP (EIAJ) pinouts are different between the 80C186XL and the 80C186EA. In addition to the PDTMR pin, the 80C186EA has more power and ground pins and the overall arrangement of pins was shifted. A new circuit board layout for the 80C186EA is required.

Operating Modes

The 80C186XL has two operating modes, Compatible and Enhanced. Compatible Mode is a pin-to-pin replacement for the NMOS 80186, except for numerics coprocessing. In Enhanced Mode, the processor has a Refresh Control Unit, the Power-Save feature and an interface to the 80C187 Numerics Coprocessor. The $\overline{MCS0}$, $\overline{MCS1}$, and $\overline{MCS3}$ pins change their functions to constitute handshaking pins for the 80C187.

The 80C186EA allows all non-80C187 users to use all the $\overline{MCS}$ pins for chip-selects. In regular operation, all 80C186EA features (including those of the Enhanced Mode 80C186) are present except for the interface to the 80C187. Numerics Mode disables the three chip-select pins and reconfigures them for connection to the 80C187.

TTL vs CMOS Inputs

The inputs of the 80C186EA are rated for CMOS switching levels for improved noise immunity, but the 80C186XL inputs are rated for TTL switching levels. In particular, the 80C186EA requires a minimum V_{IH} of 3.5V to recognize a logic one while the 80C186XL requires a minimum V_{IH} of only 1.9V (assuming 5.0V operation). The solution is to drive the 80C186EA with true CMOS devices, such as those from the HC and AC logic families, or to use pullup resistors where the added current draw is not a problem.

Timing Specifications

80C186EA timing relationships are expressed in a simplified format over the 80C186XL. The AC performance of an 80C186EA at a specified frequency will be very close to that of an 80C186XL at the same frequency. Check the timings applicable to your design prior to replacing the 80C186XL.

PACKAGE INFORMATION

This section describes the pins, pinouts, and thermal characteristics for the 80C186EA in the Plastic Leaded Chip Carrier (PLCC) package, Shrink Quad Flat Pack (SQFP), and Quad Flat Pack (QFP) package. For complete package specifications and information, see the Intel Packaging Outlines and Dimensions Guide (Order Number: 231369).

With the extended temperature range operational characteristics are guaranteed over a temperature range corresponding to −40°C to +85°C ambient. Package types are identified by a two-letter prefix to the part number. The prefixes are listed in Table 1.

Table 1. Prefix Identification

Prefix	Note	Package Type	Temperature Range
TN		PLCC	Extended
TS		QFP (EIAJ)	Extended
SB	1	SQFP	Extended/Commercial
N	1	PLCC	Commercial
S	1	QFP (EIAJ)	Commercial

NOTE:
1. The 25 MHz version is only available in commercial temperature range corresponding to 0°C to +70°C ambient.

Pin Descriptions

Each pin or logical set of pins is described in Table 3. There are three columns for each entry in the Pin Description Table.

The **Pin Name** column contains a mnemonic that describes the pin function. Negation of the signal name (for example, $\overline{\text{RESIN}}$) denotes a signal that is active low.

The **Pin Type** column contains two kinds of information. The first symbol indicates whether a pin is power (P), ground (G), input only (I), output only (O) or input/output (I/O). Some pins have multiplexed functions (for example, A19/S6). Additional symbols indicate additional characteristics for each pin. Table 3 lists all the possible symbols for this column.

The **Input Type** column indicates the type of input (asynchronous or synchronous).

Asynchronous pins require that setup and hold times be met only in order to guarantee *recognition* at a particular clock edge. Synchronous pins require that setup and hold times be met to guarantee proper *operation.* For example, missing the setup or hold time for the SRDY pin (a synchronous input) will result in a system failure or lockup. Input pins may also be edge- or level-sensitive. The possible characteristics for input pins are S(E), S(L), A(E) and A(L).

The **Output States** column indicates the output state as a function of the device operating mode. Output states are dependent upon the current activity of the processor. There are four operational states that are different from regular operation: bus hold, reset, Idle Mode and Powerdown Mode. Appropriate characteristics for these states are also indicated in this column, with the legend for all possible characteristics in Table 2.

The **Pin Description** column contains a text description of each pin.

As an example, consider AD15:0. I/O signifies the pins are bidirectional. S(L) signifies that the input function is synchronous and level-sensitive. H(Z) signifies that, as outputs, the pins are high-impedance upon acknowledgement of bus hold. R(Z) signifies that the pins float during reset. P(X) signifies that the pins retain their states during Powerdown Mode.

Table 2. Pin Description Nomenclature

Symbol	Description
P	Power Pin (Apply $+V_{CC}$ Voltage)
G	Ground (Connect to V_{SS})
I	Input Only Pin
O	Output Only Pin
I/O	Input/Output Pin
S(E)	Synchronous, Edge Sensitive
S(L)	Synchronous, Level Sensitive
A(E)	Asynchronous, Edge Sensitive
A(L)	Asynchronous, Level Sensitive
H(1)	Output Driven to V_{CC} during Bus Hold
H(0)	Output Driven to V_{SS} during Bus Hold
H(Z)	Output Floats during Bus Hold
H(Q)	Output Remains Active during Bus Hold
H(X)	Output Retains Current State during Bus Hold
R(WH)	Output Weakly Held at V_{CC} during Reset
R(1)	Output Driven to V_{CC} during Reset
R(0)	Output Driven to V_{SS} during Reset
R(Z)	Output Floats during Reset
R(Q)	Output Remains Active during Reset
R(X)	Output Retains Current State during Reset
I(1)	Output Driven to V_{CC} during Idle Mode
I(0)	Output Driven to V_{SS} during Idle Mode
I(Z)	Output Floats during Idle Mode
I(Q)	Output Remains Active during Idle Mode
I(X)	Output Retains Current State during Idle Mode
P(1)	Output Driven to V_{CC} during Powerdown Mode
P(0)	Output Driven to V_{SS} during Powerdown Mode
P(Z)	Output Floats during Powerdown Mode
P(Q)	Output Remains Active during Powerdown Mode
P(X)	Output Retains Current State during Powerdown Mode

Table 3. Pin Descriptions

Pin Name	Pin Type	Input Type	Output States	Description
V_{CC}	P			**POWER** connections consist of six pins which must be shorted externally to a V_{CC} board plane.
V_{SS}	G			**GROUND** connections consist of five pins which must be shorted externally to a V_{SS} board plane.
CLKIN	I	A(E)		**CLocK INput** is an input for an external clock. An external oscillator operating at two times the required processor operating frequency can be connected to CLKIN. For crystal operation, CLKIN (along with OSCOUT) are the crystal connections to an internal Pierce oscillator.
OSCOUT	O		H(Q) R(Q) P(Q)	**OSCillator OUTput** is only used when using a crystal to generate the external clock. OSCOUT (along with CLKIN) are the crystal connections to an internal Pierce oscillator. This pin is not to be used as 2X clock output for non-crystal applications (i.e., this pin is N.C. for non-crystal applications). OSCOUT does not float in ONCE mode.
CLKOUT	O		H(Q) R(Q) P(Q)	**CLocK OUTput** provides a timing reference for inputs and outputs of the processor, and is one-half the input clock (CLKIN) frequency. CLKOUT has a 50% duty cycle and transistions every falling edge of CLKIN.
$\overline{\text{RESIN}}$	I	A(L)		**RESet IN** causes the processor to immediately terminate any bus cycle in progress and assume an initialized state. All pins will be driven to a known state, and RESOUT will also be driven active. The rising edge (low-to-high) transition synchronizes CLKOUT with CLKIN before the processor begins fetching opcodes at memory location 0FFFF0H.
RESOUT	O		H(0) R(1) P(0)	**RESet OUTput** that indicates the processor is currently in the reset state. RESOUT will remain active as long as $\overline{\text{RESIN}}$ remains active. When tied to the $\overline{\text{TEST}}$/BUSY pin, RESOUT forces the 80C186EA into Numerics Mode.
PDTMR	I/O	A(L)	H(WH) R(Z) P(1)	**Power-Down TiMeR** pin (normally connected to an external capacitor) that determines the amount of time the processor waits after an exit from power down before resuming normal operation. The duration of time required will depend on the startup characteristics of the crystal oscillator.
NMI	I	A(E)		**Non-Maskable Interrupt** input causes a Type 2 interrupt to be serviced by the CPU. NMI is latched internally.
$\overline{\text{TEST}}$/BUSY ($\overline{\text{TEST}}$)	I	A(E)		**$\overline{\text{TEST}}$/BUSY** is sampled upon reset to determine whether the 80C186EA is to enter Numerics Mode. In regular operation, the pin is **$\overline{\text{TEST}}$.** $\overline{\text{TEST}}$ is used during the execution of the WAIT instruction to suspend CPU operation until the pin is sampled active (low). In Numerics Mode, the pin is **BUSY**. BUSY notifies the 80C186EA of 80C187 Numerics Coprocessor activity.
AD15:0 (AD7:0)	I/O	S(L)	H(Z) R(Z) P(X)	These pins provide a multiplexed **Address** and **Data** bus. During the address phase of the bus cycle, address bits 0 through 15 (0 through 7 on the 8-bit bus versions) are presented on the bus and can be latched using ALE. 8- or 16-bit data information is transferred during the data phase of the bus cycle.

NOTE:
Pin names in parentheses apply to the 80C188EA and 80L188EA.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Description
A18:16 A19/S6–A16 (A19–A8)	O		H(Z) R(Z) P(X)	These pins provide multiplexed **Address** during the address phase of the bus cycle. Address bits 16 through 19 are presented on these pins and can be latched using ALE. A18:16 are driven to a logic 0 during the data phase of the bus cycle. On the 8-bit bus versions, A15–A8 provide valid address information for the entire bus cycle. Also during the data phase, S6 is driven to a logic 0 to indicate a CPU-initiated bus cycle or logic 1 to indicate a DMA-initiated bus cycle or a refresh cycle.
$\overline{S2:0}$	O		H(Z) R(Z) P(1)	Bus cycle **Status** are encoded on these pins to provide bus transaction information. $\overline{S2:0}$ are encoded as follows:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Processor HALT
1	0	0	Queue Instruction Fetch
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive (no bus activity)

ALE/QS0	O		H(0) R(0) P(0)	**Address Latch Enable** output is used to strobe address information into a transparent type latch during the address phase of the bus cycle. In Queue Status Mode, QS0 provides queue status information along with QS1.
$\overline{BHE}$ ($\overline{RFSH}$)	O		H(Z) R(Z) P(X)	**Byte High Enable** output to indicate that the bus cycle in progress is transferring data over the upper half of the data bus. $\overline{BHE}$ and A0 have the following logical encoding:

A0	$\overline{BHE}$	Encoding (For 80C186EA/80L186EA Only)
0	0	Word Transfer
0	1	Even Byte Transfer
1	0	Odd Byte Transfer
1	1	Refresh Operation

On the 80C188EA/80L188EA, $\overline{RFSH}$ is asserted low to indicate a Refresh bus cycle.

$\overline{RD}$/$\overline{QSMD}$	O		H(Z) R(WH) P(1)	**ReaD** output signals that the accessed memory or I/O device must drive data information onto the data bus. Upon reset, this pin has an alternate function. As $\overline{QSMD}$, it enables **Queue Status Mode** when grounded. In Queue Status Mode, the ALE/QS0 and $\overline{WR}$/QS1 pins provide the following information about processor/instruction queue interaction:

QS1	QS0	Queue Operation
0	0	No Queue Operation
0	1	First Opcode Byte Fetched from the Queue
1	1	Subsequent Byte Fetched from the Queue
1	0	Empty the Queue

NOTE:
Pin names in parentheses apply to the 80C188EA and 80L188EA.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Description
$\overline{WR}$/QS1	O		H(Z) R(Z) P(1)	**WRite** output signals that data available on the data bus are to be written into the accessed memory or I/O device. In Queue Status Mode, QS1 provides queue status information along with QS0.
ARDY	I	A(L) S(L)		**Asychronous ReaDY** is an input to signal for the end of a bus cycle. ARDY is asynchronous on rising CLKOUT and synchronous on falling CLKOUT. ARDY or SRDY must be active to terminate any processor bus cycle, unless they are ignored due to correct programming of the Chip Select Unit.
SRDY	I	S(L)		**Synchronous ReaDY** is an input to signal for the end of a bus cycle. ARDY or SRDY must be active to terminate any processor bus cycle, unless they are ignored due to correct programming of the Chip Select Unit.
$\overline{DEN}$	O	H(Z) R(Z) P(1)		**Data ENable** output to control the enable of bidirectional transceivers when buffering a system. $\overline{DEN}$ is active only when data is to be transferred on the bus.
DT/$\overline{R}$	O		H(Z) R(Z) P(X)	**Data Transmit/Receive** output controls the direction of a bi-directional buffer in a buffered system. DT/$\overline{R}$ is only available on the QFP (EIAJ) package and the SQFP package.
$\overline{LOCK}$	O		H(Z) R(WH) P(1)	**LOCK** output indicates that the bus cycle in progress is not to be interrupted. The processor will not service other bus requests (such as HOLD) while $\overline{LOCK}$ is active. This pin is configured as a weakly held high input while $\overline{RESIN}$ is active and must not be driven low.
HOLD	I	A(L)		**HOLD** request input to signal that an external bus master wishes to gain control of the local bus. The processor will relinquish control of the local bus between instruction boundaries not conditioned by a LOCK prefix.
HLDA	O		H(1) R(0) P(0)	**HoLD Acknowledge** output to indicate that the processor has relinquished control of the local bus. When HLDA is asserted, the processor will (or has) floated its data bus and control signals allowing another bus master to drive the signals directly.
$\overline{UCS}$	O		H(1) R(1) P(1)	**Upper Chip Select** will go active whenever the address of a memory or I/O bus cycle is within the address limitations programmed by the user. After reset, $\overline{UCS}$ is configured to be active for memory accesses between 0FFC00H and 0FFFFFH. During a processor reset, $\overline{UCS}$ and $\overline{LCS}$ are used to enable ONCE Mode.
$\overline{LCS}$	O		H(1) R(1) P(1)	**Lower Chip Select** will go active whenever the address of a memory bus cycle is within the address limitations programmed by the user. $\overline{LCS}$ is inactive after a reset. During a processor reset, $\overline{UCS}$ and $\overline{LCS}$ are used to enable ONCE Mode.

NOTE:
Pin names in parentheses apply to the 80C188EA and 80L188EA.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Description
MCS0/PEREQ MCS1/ERROR MCS2 MCS3/NCS	I/O	A(L)	H(1) R(1) P(1)	These pins provide a multiplexed function. If enabled, these pins normally comprise a block of **Mid-Range Chip Select** outputs which will go active whenever the address of a memory bus cycle is within the address limitations programmed by the user. In Numerics Mode (80C186EA only), three of the pins become handshaking pins for the 80C187. The **CoProcessor REQuest** input signals that a data transfer is pending. **ERROR** is an input which indicates that the previous numerics coprocessor operation resulted in an exception condition. An interrupt Type 16 is generated when ERROR is sampled active at the beginning of a numerics operation. **Numerics Coprocessor Select** is an output signal generated when the processor accesses the 80C187.
PCS4:0	O		H(1) R(1) P(1)	**Peripheral Chip Selects** go active whenever the address of a memory or I/O bus cycle is within the address limitations programmed by the user.
PCS5/A1 PCS6/A2	O		H(1)/H(X) R(1) P(1)	These pins provide a multiplexed function. As additional **Peripheral Chip Selects,** they go active whenever the address of a memory or I/O bus cycle is within the address limitations by the user. They may also be programmed to provide latched **Address** A2:1 signals.
T0OUT T1OUT	O		H(Q) R(1) P(Q)	**Timer OUTput** pins can be programmed to provide a single clock or continuous waveform generation, depending on the timer mode selected.
T0IN T1IN	I	A(L) A(E)		**Timer INput** is used either as clock or control signals, depending on the timer mode selected.
DRQ0 DRQ1	I	A(L)		**DMA ReQuest** is asserted by an external request when it is prepared for a DMA transfer.
INT0 INT1/SELECT	I	A(E,L)		Maskable **INTerrupt** input will cause a vector to a specific Type interrupt routine. To allow interrupt expansion, INT0 and/or INT1 can be used with INTA0 and INTA1 to interface with an external slave controller. INT1 becomes SELECT when the ICU is configured for Slave Mode.
INT2/INTA0 INT3/INTA1/IRQ	I/O	A(E,L)	H(1) R(Z) P(1)	These pins provide multiplexed functions. As inputs, they provide a maskable **INTerrupt** that will cause the CPU to vector to a specific Type interrupt routine. As outputs, each is programmatically controlled to provide an **INTerrupt Acknowledge** handshake signal to allow interrupt expansion. INT3/INTA1 becomes **IRQ** when the ICU is configured for Slave Mode.
N.C.				**No Connect.** For compatibility with future products, do not connect to these pins.

NOTE:
Pin names in parentheses apply to the 80C188EA and 80L188EA.

80C186EA PINOUT

Tables 4 and 5 list the 80C186EA pin names with package location for the 68-pin Plastic Leaded Chip Carrier (PLCC) component. Figure 9 depicts the complete 80C186EA/80L186EA pinout (PLCC package) as viewed from the top side of the component (i.e., contacts facing down).

Tables 6 and 7 list the 80C186EA pin names with package location for the 80-pin Quad Flat Pack (EIAJ) component. Figure 6 depicts the complete 80C186EA/80C188EA (EIAJ QFP package) as viewed from the top side of the component (i.e., contacts facing down).

Tables 8 and 9 list the 80C186EA/80C188EA pin names with package location for the 80-pin Shrink Quad Flat Pack (SQFP) component. Figure 7 depicts the complete 80C186EA/80C188EA (SQFP) as viewed from the top side of the component (i.e., contacts facing down).

Table 4. PLCC Pin Names with Package Location

Address/Data Bus	
Name	**Location**
AD0	17
AD1	15
AD2	13
AD3	11
AD4	8
AD5	6
AD6	4
AD7	2
AD8 (A8)	16
AD9 (A9)	14
AD10 (A10)	12
AD11 (A11)	10
AD12 (A12)	7
AD13 (A13)	5
AD14 (A14)	3
AD15 (A15)	1
A16	68
A17	67
A18	66
A19/S6	65

Bus Control	
Name	**Location**
ALE/QS0	61
$\overline{\text{BHE}}$ ($\overline{\text{RFSH}}$)	64
$\overline{\text{S0}}$	52
$\overline{\text{S1}}$	53
$\overline{\text{S2}}$	54
$\overline{\text{RD}}$/$\overline{\text{QSMD}}$	62
$\overline{\text{WR}}$/QS1	63
ARDY	55
SRDY	49
$\overline{\text{DEN}}$	39
$\overline{\text{LOCK}}$	48
HOLD	50
HLDA	51

Power	
Name	**Location**
V_{SS}	26, 60
V_{CC}	9, 43

Processor Control	
Name	**Location**
$\overline{\text{RESIN}}$	24
RESOUT	57
CLKIN	59
OSCOUT	58
CLKOUT	56
$\overline{\text{TEST}}$/BUSY	47
PDTMR	40
NMI	46
INT0	45
INT1/SELECT	44
INT2/$\overline{\text{INTA0}}$	42
INT3/$\overline{\text{INTA1}}$/IRQ	41

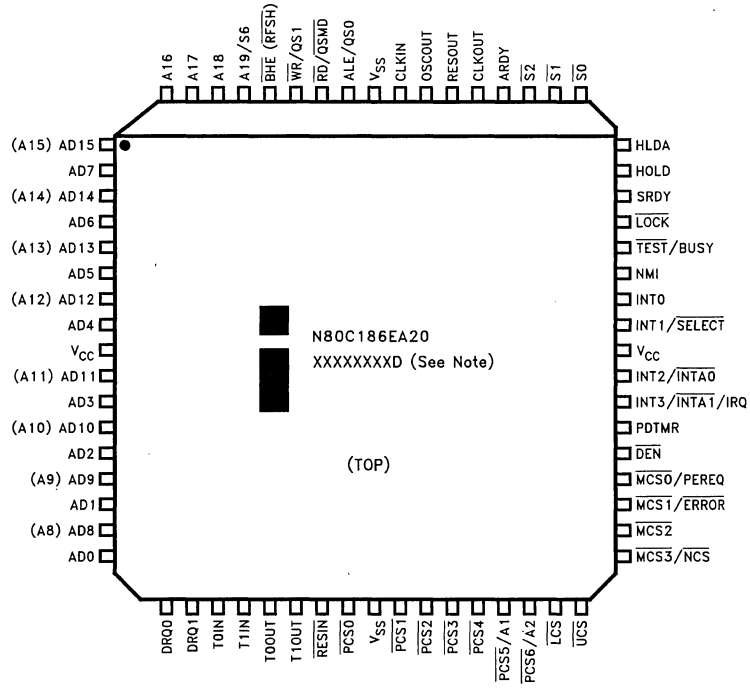
I/O	
Name	**Location**
$\overline{\text{UCS}}$	34
$\overline{\text{LCS}}$	33
$\overline{\text{MCS0}}$/PEREQ	38
$\overline{\text{MCS1}}$/$\overline{\text{ERROR}}$	37
$\overline{\text{MCS2}}$	36
$\overline{\text{MCS3}}$/$\overline{\text{NCS}}$	35
$\overline{\text{PCS0}}$	25
$\overline{\text{PCS1}}$	27
$\overline{\text{PCS2}}$	28
$\overline{\text{PCS3}}$	29
$\overline{\text{PCS4}}$	30
$\overline{\text{PCS5}}$/A1	31
$\overline{\text{PCS6}}$/A2	32
T0OUT	22
T0IN	20
T1OUT	23
T1IN	21
DRQ0	18
DRQ1	19

NOTE:
Pin names in parentheses apply to the 80C188EA/80L188EA.

Table 5. PLCC Package Location with Pin Names

Location	Name	Location	Name	Location	Name	Location	Name
1	AD15 (A15)	18	DRQ0	35	$\overline{MCS3}/\overline{NCS}$	52	$\overline{S0}$
2	AD7	19	DRQ1	36	$\overline{MCS2}$	53	$\overline{S1}$
3	AD14 (A14)	20	T0IN	37	$\overline{MCS1}/\overline{ERROR}$	54	$\overline{S2}$
4	AD6	21	T1IN	38	$\overline{MCS0}$/PEREQ	55	ARDY
5	AD13 (A13)	22	T0OUT	39	$\overline{DEN}$	56	CLKOUT
6	AD5	23	T1OUT	40	PDTMR	57	RESOUT
7	AD12 (A12)	24	$\overline{RESIN}$	41	INT3/$\overline{INTA1}$/	58	OSCOUT
8	AD4	25	$\overline{PCS0}$		IRQ	59	CLKIN
9	V_{CC}	26	V_{SS}	42	INT2/$\overline{INTA0}$	60	V_{SS}
10	AD11 (A11)	27	$\overline{PCS1}$	43	V_{CC}	61	ALE/QS0
11	AD3	28	$\overline{PCS2}$	44	INT1/SELECT	62	$\overline{RD}/\overline{QSMD}$
12	AD10 (A10)	29	$\overline{PCS3}$	45	INT0	63	$\overline{WR}$/QS1
13	AD2	30	$\overline{PCS4}$	46	NMI	64	$\overline{BHE}$ ($\overline{RFSH}$)
14	AD9 (A9)	31	$\overline{PCS5}$/A1	47	$\overline{TEST}$/BUSY	65	A19/S6
15	AD1	32	$\overline{PCS6}$/A2	48	LOCK	66	A18
16	AD8 (A8)	33	$\overline{LCS}$	49	SRDY	67	A17
17	AD0	34	$\overline{UCS}$	50	HOLD	68	A16
				51	HLDA		

NOTE:
Pin names in parentheses apply to the 80C186EA/80L188EA.



NOTES:
1. The nine-character alphanumeric code (XXXXXXXXD) underneath the product number is the Intel FPO number.
2. Pin names in parentheses apply to the 80C186EA/80L188EA.

272432-5

Figure 5. 68-Lead PLCC Pinout Diagram

Table 6. QFP (EIAJ) Pin Names with Package Location

Address/Data Bus		Bus Control		Processor Control		I/O	
Name	**Location**	**Name**	**Location**	**Name**	**Location**	**Name**	**Location**
AD0	64	ALE/QS0	10	$\overline{\text{RESIN}}$	55	$\overline{\text{UCS}}$	45
AD1	66	$\overline{\text{BHE}}$ ($\overline{\text{RFSH}}$)	7	RESOUT	18	$\overline{\text{LCS}}$	46
AD2	68	$\overline{\text{S0}}$	23	CLKIN	16	$\overline{\text{MCS0}}$/PEREQ	40
AD3	70	$\overline{\text{S1}}$	22	OSCOUT	17	$\overline{\text{MCS1}}$/$\overline{\text{ERROR}}$	41
AD4	74	$\overline{\text{S2}}$	21	CLKOUT	19	$\overline{\text{MCS2}}$	42
AD5	76	$\overline{\text{RD}}$/$\overline{\text{QSMD}}$	9	$\overline{\text{TEST}}$/BUSY	29	$\overline{\text{MCS3}}$/$\overline{\text{NCS}}$	43
AD6	78	$\overline{\text{WR}}$/QS1	8	PDTMR	38	$\overline{\text{PCS0}}$	54
AD7	80	ARDY	20	NMI	30	$\overline{\text{PCS1}}$	52
AD8 (A8)	65	SRDY	27	INT0	31	$\overline{\text{PCS2}}$	51
AD9 (A9)	67	DT/$\overline{\text{R}}$	37	INT1/$\overline{\text{SELECT}}$	32	$\overline{\text{PCS3}}$	50
AD10 (A10)	69	$\overline{\text{DEN}}$	39	INT2/$\overline{\text{INTA0}}$	35	$\overline{\text{PCS4}}$	49
AD11 (A11)	71	$\overline{\text{LOCK}}$	28	INT3/$\overline{\text{INTA1}}$/	36	$\overline{\text{PCS5}}$/A1	48
AD12 (A12)	75	HOLD	26	IRQ		$\overline{\text{PCS6}}$/A2	47
AD13 (A13)	77	HLDA	25	N.C.	11, 14, 15, 63	T0OUT	57
AD14 (A14)	79					T0IN	59
AD15 (A15)	1					T1OUT	56
A16	3					T1IN	58
A17	4					DRQ0	61
A18	5					DRQ1	60
A19/S6	6						

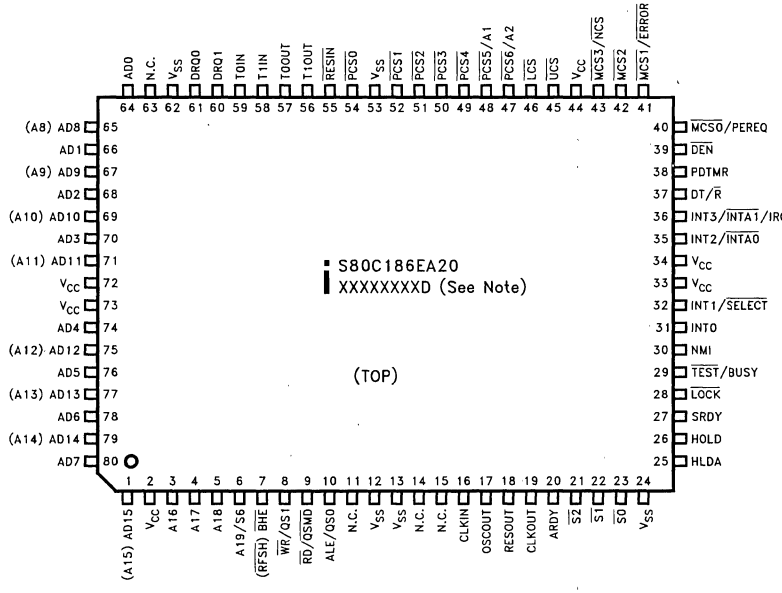
Power	
Name	**Location**
V_{SS}	12, 13, 24, 53,62
V_{CC}	2, 33, 34, 44, 72, 73

NOTE:
Pin names in parentheses apply to the 80C186EA/80L188EA.

Table 7. QFP (EIAJ) Package Location with Pin Names

Location	Name	Location	Name	Location	Name	Location	Name
1	AD15 (A15)	21	$\overline{S2}$	41	$\overline{MCS1}/\overline{ERROR}$	61	DRQ0
2	V_{CC}	22	$\overline{S1}$	42	$\overline{MCS2}$	62	V_{SS}
3	A16	23	$\overline{S0}$	43	$\overline{MCS3}/\overline{NCS}$	63	N.C.
4	A17	24	V_{SS}	44	V_{CC}	64	AD0
5	A18	25	HLDA	45	$\overline{UCS}$	65	AD8 (A8)
6	A19/S6	26	HOLD	46	$\overline{LCS}$	66	AD1
7	$\overline{BHE}$ ($\overline{RFSH}$)	27	SRDY	47	$\overline{PCS6}$/A2	67	AD9 (A9)
8	$\overline{WR}$/QS1	28	$\overline{LOCK}$	48	$\overline{PCS5}$/A1	68	AD2
9	$\overline{RD}/\overline{QSMD}$	29	$\overline{TEST}$/BUSY	49	$\overline{PCS4}$	69	AD10 (A10)
10	ALE/QS0	30	NMI	50	$\overline{PCS3}$	70	AD3
11	N.C.	31	INT0	51	$\overline{PCS2}$	71	AD11 (A11)
12	V_{SS}	32	INT1/$\overline{SELECT}$	52	$\overline{PCS1}$	72	V_{CC}
13	V_{SS}	33	V_{CC}	53	V_{SS}	73	V_{CC}
14	N.C.	34	V_{CC}	54	$\overline{PCS0}$	74	AD4
15	N.C.	35	INT2/$\overline{INTA0}$	55	$\overline{RESIN}$	75	AD12 (A12)
16	CLKIN	36	INT3/$\overline{INTA1}$/	56	T1OUT	76	AD5
17	OSCOUT		IRQ	57	T0OUT	77	AD13 (A13)
18	RESOUT	37	DT/$\overline{R}$	58	T1IN	78	AD6
19	CLKOUT	38	PDTMR	59	T0IN	79	AD14 (A14)
20	ARDY	39	$\overline{DEN}$	60	DRQ1	80	AD7
		40	$\overline{MCS0}$/PEREQ				

NOTE:
Pin names in parentheses apply to the 80C186EA/80L188EA.



NOTES:
1. The nine-character alphanumeric code (XXXXXXXXD) underneath the product number is the Intel FPO number.
2. Pin names in parentheses apply to the 80C186EA/80L188EA.

272432-6

Figure 6. Quad Flat Pack (EIAJ) Pinout Diagram

Table 8. SQFP Pin Functions with Package Location

AD Bus	
AD0	1
AD1	3
AD2	6
AD3	8
AD4	12
AD5	14
AD6	16
AD7	18
AD8 (A8)	2
AD9 (A9)	5
AD10 (A10)	7
AD11 (A11)	9
AD12 (A12)	13
AD13 (A13)	15
AD14 (A14)	17
AD15 (A15)	19
A16/S3	21
A17/S4	22
A18/S5	23
A19/S6	24

Bus Control	
ALE/QS0	29
$\overline{BHE}$/($\overline{RFSH}$)	26
$\overline{S0}$	40
$\overline{S1}$	39
$\overline{S2}$	38
$\overline{RD}$/$\overline{QSMD}$	28
$\overline{WR}$/QS1	27
ARDY	37
SRDY	44
$\overline{DEN}$	56
DT/$\overline{R}$	54
$\overline{LOCK}$	45
HOLD	43
HLDA	42

No Connection	
N.C.	4
N.C.	25
N.C.	35
N.C.	72

Processor Control	
$\overline{RESIN}$	73
RESOUT	34
CLKIN	32
OSCOUT	33
CLKOUT	36
$\overline{TEST}$/BUSY	46
NMI	47
INT0	48
INT1/$\overline{SELECT}$	49
INT2/$\overline{INTA0}$	52
INT3/$\overline{INTA1}$	53
PDTMR	55

Power and Ground	
V_{CC}	10
V_{CC}	11
V_{CC}	20
V_{CC}	50
V_{CC}	51
V_{CC}	61
V_{SS}	30
V_{SS}	31
V_{SS}	41
V_{SS}	70
V_{SS}	80

I/O	
$\overline{UCS}$	62
$\overline{LCS}$	63
$\overline{MCS0}$/PEREQ	57
$\overline{MCS1}$/$\overline{ERROR}$	58
$\overline{MCS2}$	59
$\overline{MCS3}$/$\overline{NPS}$	60
$\overline{PCS0}$	71
$\overline{PCS1}$	69
$\overline{PCS2}$	68
$\overline{PCS3}$	67
$\overline{PCS4}$	66
$\overline{PCS5}$/A1	65
$\overline{PCS6}$/A2	64
TMR IN 0	77
TMR IN 1	76
TMR OUT 0	75
TMR OUT 1	74
DRQ0	79
DRQ1	78

NOTE:
Pin names in parentheses apply to the 80C186EA/80L188EA.

Table 9. SQFP Pin Locations with Pin Names

1	AD0
2	AD8 (A8)
3	AD1
4	N.C.
5	AD9 (A9)
6	AD2
7	AD10 (A10)
8	AD3
9	AD11 (A11)
10	V_{CC}
11	V_{CC}
12	AD4
13	AD12 (A12)
14	AD5
15	AD13 (A13)
16	AD6
17	AD14 (A14)
18	AD7
19	AD15 (A15)
20	V_{CC}

21	A16/S3
22	A17/S4
23	A18/S5
24	A19/S6
25	N.C.
26	$\overline{BHE}$/($\overline{RFSH}$)
27	$\overline{WR}$/QS1
28	RD/$\overline{QSMD}$
29	ALE/QS0
30	V_{SS}
31	V_{SS}
32	X1
33	X2
34	RESET
35	N.C.
36	CLKOUT
37	ARDY
38	$\overline{S2}$
39	$\overline{S1}$
40	$\overline{S0}$

41	V_{SS}
42	HLDA
43	HOLD
44	SRDY
45	$\overline{LOCK}$
46	$\overline{TEST}$/BUSY
47	NMI
48	INT0
49	INT1/$\overline{SELECT}$
50	V_{CC}
51	V_{CC}
52	INT2/$\overline{INTA0}$
53	INT3/$\overline{INTA1}$
54	DT/$\overline{R}$
55	PDTMR
56	$\overline{DEN}$
57	$\overline{MCS0}$/PEREQ
58	$\overline{MCS1}$/$\overline{ERROR}$
59	$\overline{MCS2}$
60	$\overline{MCS3}$/$\overline{NPS}$

61	V_{CC}
62	$\overline{UCS}$
63	$\overline{LCS}$
64	$\overline{PCS6}$/A2
65	$\overline{PCS5}$/A1
66	$\overline{PCS4}$
67	$\overline{PCS3}$
68	$\overline{PCS2}$
69	$\overline{PCS1}$
70	V_{SS}
71	$\overline{PCS0}$
72	N.C.
73	$\overline{RES}$
74	TMR OUT 1
75	TMR OUT 0
76	TMR IN 1
77	TMR IN 0
78	DRQ1
79	DRQ0
80	V_{SS}

NOTE:
Pin names in parentheses apply to the 80C186EA/80L188EA.

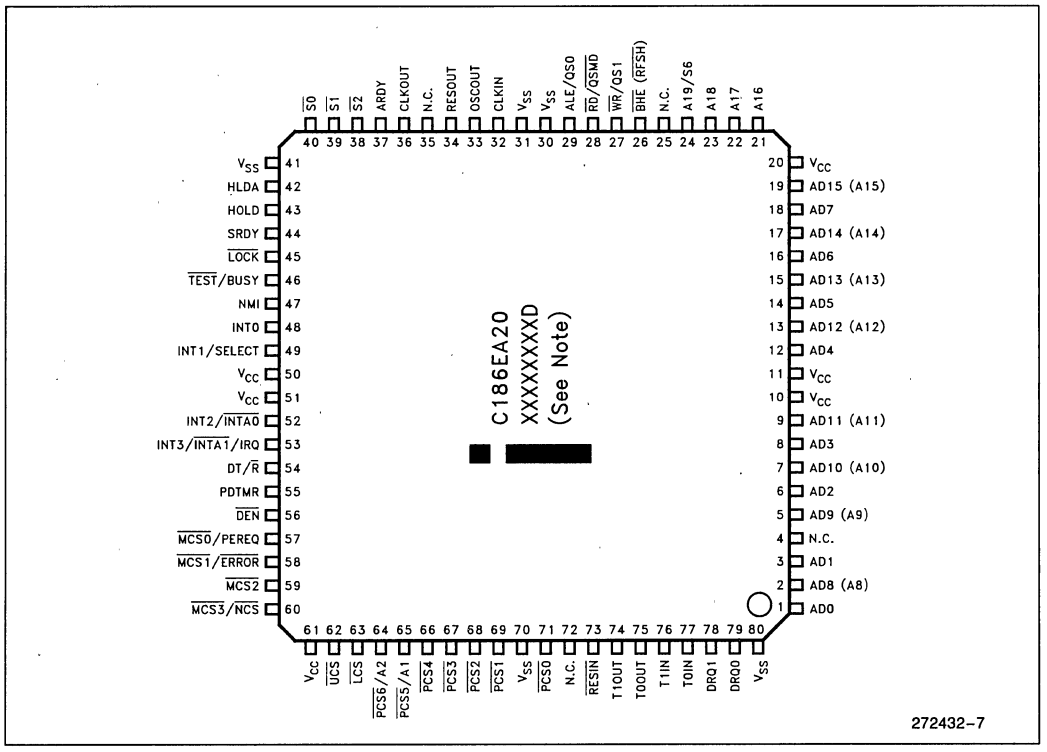


Figure 7. Shrink Quad Flat Pack (SQFP) Pinout Diagram

NOTES:
1. XXXXXXXXD indicates the Intel FPO number.
2. Pin names in parentheses apply to the 80C188EA.

PACKAGE THERMAL SPECIFICATIONS

The 80C186EA/80L186EA is specified for operation when T_C (the case temperature) is within the range of 0°C to 85°C (PLCC package) or 0°C to 106°C (QFP-EIAJ) package. T_C may be measured in any environment to determine whether the processor is within the specified operating range. The case temperature must be measured at the center of the top surface.

T_A (the ambient temperature) can be calculated from θ_{CA} (thermal resistance from the case to ambient) with the following equation:

$$T_A = T_C - P \times \theta_{CA}$$

Typical values for θ_{CA} at various airflows are given in Table 10.

P (the maximum power consumption, specified in watts) is calculated by using the maximum ICC as tabulated in the DC specifications and V_{CC} of 5.5V.

Table 10. Thermal Resistance (θ_{CA}) at Various Airflows (in °C/Watt)

	Airflow Linear ft/min (m/sec)					
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
θ_{CA} (PLCC)	29	25	21	19	17	16.5
θ_{CA} (QFP)	66	63	60.5	59	58	57
θ_{CA} (SQFP)	70					

ELECTRICAL SPECIFICATIONS

NOTICE: This data sheet contains preliminary information on new products in production. It is valid for the devices indicated in the revision history. The specifications are subject to change without notice.

Absolute Maximum Ratings*

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

Case Temperature under Bias ... −65°C to +150°C

Supply Voltage with Respect to V_{SS} ..... −0.5V to +6.5V

Voltage on Other Pins with Respect to V_{SS} ..... −0.5V to V_{CC} + 0.5V

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

Recommended Connections

Power and ground connections must be made to multiple V_{CC} and V_{SS} pins. Every 80C186EA based circuit board should contain separate power (V_{CC}) and ground (V_{SS}) planes. All V_{CC} and V_{SS} pins **must** be connected to the appropriate plane. Pins identified as "N.C." must not be connected in the system. Decoupling capacitors should be placed near the processor. The value and type of decoupling capacitors is application and board layout dependent. The processor can cause transient power surges when its output buffers transition, particularly when connected to large capacitive loads.

Always connect any unused input pins to an appropriate signal level. In particular, unused interrupt pins (NMI, INT3:0) should be connected to V_{SS} to avoid unwanted interrupts. **Leave any unused output pin or any "N.C." pin unconnected.**

DC SPECIFICATIONS (80C186EA/80C188EA)

Symbol	Parameter	Min	Max	Units	Conditions
V_{CC}	Supply Voltage	4.5	5.5	V	
V_{IL}	Input Low Voltage for All Pins	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage for All Pins	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 3 mA (min)
V_{OH}	Output High Voltage	V_{CC} − 0.5		V	I_{OH} = −2 mA (min)
V_{HYR}	Input Hysterisis on $\overline{RESIN}$	0.30		V	
I_{IL1}	Input Leakage Current (except $\overline{RD}$/QSMD, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$/PEREQ, $\overline{MCS1}$/$\overline{ERROR}$, $\overline{LOCK}$ and $\overline{TEST}$/BUSY)		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{IL2}	Input Leakage Current ($\overline{RD}$/QSMD, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$/PEREQ, $\overline{MCS1}$, $\overline{ERROR}$, $\overline{LOCK}$ and $\overline{TEST}$/BUSY	−275		μA	V_{IN} = 0.7 V_{CC} (Note 1)
I_{OL}	Output Leakage Current		±10	μA	0.45 ≤ V_{OUT} ≤ V_{CC} (Note 2)
I_{CC}	Supply Current Cold (RESET) 80C186EA25/80C188EA25 80C186EA20/80C188EA20 80C186EA13/80C188EA13		 105 90 65	 mA mA mA	 (Notes 3, 5)
I_{ID}	Supply Current In Idle Mode 80C186EA25/80C188EA25 80C186EA20/80C188EA20 80C186EA13/80C188EA13		 90 70 46	 mA mA mA	 (Note 5)
I_{PD}	Supply Current In Powerdown Mode 80C186EA25/80C188EA25 80C186EA20/80C188EA20 80C186EA13/80C188EA13		 100 100 100	 μA μA μA	 (Note 5)
C_{OUT}	Output Pin Capacitance	0	15	pF	T_F = 1 MHz (Note 4)
C_{IN}	Input Pin Capacitance	0	15	pF	T_F = 1 MHz

NOTES:
1. $\overline{RD}$/QSMD, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$/PEREQ, $\overline{MCS1}$/$\overline{ERROR}$, $\overline{LOCK}$ and $\overline{TEST}$/BUSY have internal pullups that are only activated during RESET. Loading these pins above I_{OL} = −275 μA will cause the processor to enter alternate modes of operation.
2. Output pins are floated using HOLD or ONCE Mode.
3. Measured at worst case temperature and V_{CC} with all outputs loaded as specified in the AC Test Conditions, and with the device in RESET ($\overline{RESIN}$ held low). RESET is worst case for I_{CC}.
4. Output capacitance is the capacitive load of a floating output pin.
5. Operating conditions for 25 MHz are 0°C to +70°C, V_{CC} = 5.0V ±10%.

DC SPECIFICATIONS (80L186EA/80L188EA)

Symbol	Parameter	Min	Max	Units	Conditions
V_{CC}	Supply Voltage	2.7	5.5	V	
V_{IL}	Input Low Voltage for All Pins	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage for All Pins	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 1.6 mA (min)
V_{OH}	Output High Voltage	V_{CC} − 0.5		V	I_{OH} = −1 mA (min)
V_{HYR}	Input Hysterisis on $\overline{RESIN}$	0.30		V	
I_{IL1}	Input Leakage Current (except $\overline{RD}$/QSMD, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$/PEREQ, $\overline{MCS1}$, $\overline{LOCK}$ and $\overline{TEST}$)		±10	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{IL2}	Input Leakage Current ($\overline{RD}$/QSMD, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$, $\overline{MCS1}$, $\overline{LOCK}$ and $\overline{TEST}$)	−275		μA	V_{IN} = 0.7 V_{CC} (Note 1)
I_{OL}	Output Leakage Current		±10	μA	0.45 ≤ V_{OUT} ≤ V_{CC} (Note 2)
I_{CC5}	Supply Current (RESET, 5.5V) 80L186EA-13 80L186EA-8		 65 40	 mA mA	 (Note 3) (Note 3)
I_{CC3}	Supply Current (RESET, 2.7V) 80L186EA-13 80L186EA-8		 34 20	 mA mA	 (Note 3) (Note 3)
I_{ID5}	Supply Current Idle (5.5V) 80L186EA-13 80L186EA-8		 46 28	 mA mA	
I_{ID5}	Supply Current Idle (2.7V) 80L186EA-13 80L186EA-8		 24 14	 mA mA	
I_{PD5}	Supply Current Powerdown (5.5V) 80L186EA-13 80L186EA-8		 100 100	 μA μA	
I_{PD3}	Supply Current Powerdown (2.7V) 80L186EA-13 80L186EA-8		 50 50	 μA μA	
C_{OUT}	Output Pin Capacitance	0	15	pF	T_F = 1 MHz (Note 4)
C_{IN}	Input Pin Capacitance	0	15	pF	T_F = 1 MHz

NOTES:
1. $\overline{RD}$/QSMD, $\overline{UCS}$, $\overline{LCS}$, $\overline{MCS0}$, $\overline{MCS1}$, $\overline{LOCK}$ and $\overline{TEST}$ have internal pullups that are only activated during RESET. Loading these pins above I_{OL} = −275 μA will cause the processor to enter alternate modes of operation.
2. Output pins are floated using HOLD or ONCE Mode.
3. Measured at worst case temperature and V_{CC} with all outputs loaded as specified in the AC Test Conditions, and with the device in RESET ($\overline{RESIN}$ held low).
4. Output capacitance is the capacitive load of a floating output pin.

I_{CC} VERSUS FREQUENCY AND VOLTAGE

The current (I_{CC}) consumption of the processor is essentially composed of two components; I_{PD} and I_{CCS}.

I_{PD} is the **quiescent** current that represents internal device leakage, and is measured with all inputs or floating outputs at GND or V_{CC} (no clock applied to the device). I_{PD} is equal to the Powerdown current and is typically less than 50 μA.

I_{CCS} is the **switching** current used to charge and discharge parasitic device capacitance when changing logic levels. Since I_{CCS} is typically much greater than I_{PD}, I_{PD} can often be ignored when calculating I_{CC}.

I_{CCS} is related to the voltage and frequency at which the device is operating. It is given by the formula:

$$\text{Power} = V \times I = V^2 \times C_{DEV} \times f$$

$$\therefore I = I_{CC} = I_{CCS} = V \times C_{DEV} \times f$$

Where: V = Device operating voltage (V_{CC})

C_{DEV} = Device capacitance

f = Device operating frequency

$I_{CCS} = I_{CC}$ = Device current

Measuring C_{DEV} on a device like the 80C186EA would be difficult. Instead, C_{DEV} is calculated using the above formula by measuring I_{CC} at a known V_{CC} and frequency (see Table 11). Using this C_{DEV} value, I_{CC} can be calculated at any voltage and frequency within the specified operating range.

EXAMPLE: Calculate the typical I_{CC} when operating at 20 MHz, 4.8V.

$$I_{CC} = I_{CCS} = 4.8 \times 0.515 \times 20 \approx 49 \text{ mA}$$

PDTMR PIN DELAY CALCULATION

The PDTMR pin provides a delay between the assertion of NMI and the enabling of the internal clocks when exiting Powerdown. A delay is required only when using the on-chip oscillator to allow the crystal or resonator circuit time to stabilize.

NOTE:
The PDTMR pin function does not apply when $\overline{\text{RESIN}}$ is asserted (i.e., a device reset during Powerdown is similar to a cold reset and $\overline{\text{RESIN}}$ must remain active until after the oscillator has stabilized).

To calculate the value of capacitor required to provide a desired delay, use the equation:

$$440 \times t = C_{PD} \quad (5V, 25°C)$$

Where: t = desired delay in **seconds**

C_{PD} = capacitive load on PDTMR in **microfarads**

EXAMPLE: To get a delay of 300 μs, a capacitor value of $C_{PD} = 440 \times (300 \times 10^{-6}) = 0.132$ μF is required. Round up to standard (available) capacitive values.

NOTE:
The above equation applies to delay times greater than 10 μs and will compute the **TYPICAL** capacitance needed to achieve the desired delay. A delay variance of +50% or −25% can occur due to temperature, voltage, and device process extremes. In general, higher V_{CC} and/or lower temperature will decrease delay time, while lower V_{CC} and/or higher temperature will increase delay time.

Table 11. C_{DEV} Values

Parameter	Typ	Max	Units	Notes
C_{DEV} (Device in Reset)	0.515	0.905	mA/V*MHz	1, 2
C_{DEV} (Device in Idle)	0.391	0.635	mA/V*MHz	1, 2

1. Max C_{DEV} is calculated at −40°C, all floating outputs driven to V_{CC} or GND, and all outputs loaded to 50 pF (including CLKOUT and OSCOUT).
2. Typical C_{DEV} is calculated at 25°C with all outputs loaded to 50 pF except CLKOUT and OSCOUT, which are not loaded.

AC SPECIFICATIONS

AC Characteristics—80C186EA25/80C186EA20/80C186EA13

Symbol	Parameter	Min	Max	Min	Max	Min	Max	Units	Notes
INPUT CLOCK		**25 MHz(12)**		**20 MHz**		**13 MHz**			
T_F	CLKIN Frequency	0	50	0	40	0	26	MHz	1
T_C	CLKIN Period	20	∞	25	∞	38.5	∞	ns	1
T_{CH}	CLKIN High Time	10	∞	10	∞	12	∞	ns	1, 2
T_{CL}	CLKIN Low Time	10	∞	10	∞	12	∞	ns	1, 2
T_{CR}	CLKIN Rise Time	1	8	1	8	1	8	ns	1, 3
T_{CF}	CLKIN Fall Time	1	8	1	8	1	8	ns	1, 3
OUTPUT CLOCK									
T_{CD}	CLKIN to CLKOUT Delay	0	15	0	17	0	23	ns	1, 4
T	CLKOUT Period		$2T_C$		$2*T_C$		$2*T_C$	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5		(T/2) − 5		(T/2) − 5		ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5		(T/2) − 5		(T/2) − 5		ns	1
T_{PR}	CLKOUT Rise Time	1	6	1	6	1	6	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	6	1	6	1	6	ns	1, 5
OUTPUT DELAYS									
T_{CHOV1}	ALE, $\overline{S2:0}$, $\overline{DEN}$, DT/$\overline{R}$, $\overline{BHE}$, ($\overline{RFSH}$), $\overline{LOCK}$, A19:16	3	20	3	22	3	25	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{MCS3:0}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{PCS6:0}$, $\overline{NCS}$, $\overline{RD}$, $\overline{WR}$	3	25	3	27	3	30	ns	1, 4, 6, 8
T_{CLOV1}	$\overline{BHE}$ ($\overline{RFSH}$), $\overline{DEN}$, $\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT, A19:16	3	20	3	22	3	25	ns	1, 4, 6
T_{CLOV2}	$\overline{RD}$, $\overline{WR}$, $\overline{MCS3:0}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{PCS6:0}$, AD15:0 (A15:8, AD7:0), $\overline{NCS}$, $\overline{INTA1:0}$, $\overline{S2:0}$	3	25	3	27	3	30	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	25	0	25	0	25	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (A15:8, AD7:0)	0	25	0	25	0	25	ns	1

AC SPECIFICATIONS (Continued)

AC Characteristics—80C186EA25/80C186EA20/80C186EA13

Symbol	Parameter	Min	Max	Min	Max	Min	Max	Units	Notes
SYNCHRONOUS INPUTS		**25 MHz(12)**		**20 MHz**		**13 MHz**			
T_{CHIS}	$\overline{TEST}$, NMI, INT3:0, T1:0IN, ARDY	8		10		10		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, INT3:0, T1:0IN, ARDY	3		3		3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), ARDY, SRDY, DRQ1:0	10		10		10		ns	1, 10
T_{CLIH}	AD15:0 (AD7:0), ARDY, SRDY, DRQ1:0	3		3		3		ns	1, 10
T_{CLIS}	HOLD, PEREQ, $\overline{ERROR}$ (80C186EA Only)	10		10		10		ns	1, 9
T_{CLIH}	HOLD, PEREQ, $\overline{ERROR}$ (80C186EA Only)	3		3		3		ns	1, 9
T_{CLIS}	$\overline{RESIN}$ (to CLKIN)	10		10		10		ns	1, 9
T_{CLIH}	$\overline{RESIN}$ (from CLKIN)	3		3		3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measured at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.
11. T_{CHOVS} applies to $\overline{BHE}$ ($\overline{RFSH}$) and A19:16 only after a HOLD release.
12. Operating conditions for 25 MHz are 0°C to +70°C, V_{CC} = 5.0V ±10%.

Pin names in parentheses apply to the 80C188EA/80L188EA.

AC SPECIFICATIONS

AC Characteristics—80L186EA13/80L186EA8

Symbol	Parameter	Min	Max	Min	Max	Units	Notes
INPUT CLOCK		**13 MHz**		**8 MHz**			
T_F	CLKIN Frequency	0	26	0	16	MHz	1
T_C	CLKIN Period	38.5	∞	62.5	∞	ns	1
T_{CH}	CLKIN High Time	12	∞	12	∞	ns	1, 2
T_{CL}	CLKIN Low Time	12	∞	12	∞	ns	1, 2
T_{CR}	CLKIN Rise Time	1	8	1	8	ns	1, 3
T_{CF}	CLKIN Fall Time	1	8	1	8	ns	1, 3
OUTPUT CLOCK							
T_{CD}	CLKIN to CLKOUT Delay	0	45	0	95	ns	1, 4
T	CLKOUT Period		$2*T_C$		$2*T_C$	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5		(T/2) − 5		ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5		(T/2) − 5		ns	1
T_{PR}	CLKOUT Rise Time	1	12	1	12	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	12	1	12	ns	1, 5
OUTPUT DELAYS							
T_{CHOV1}	ALE, $\overline{LOCK}$	3	27	3	27	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{MCS3:0}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{PCS6:0}$, $\overline{RD}$, $\overline{WR}$	3	32	3	32	ns	1, 4, 6, 8
T_{CHOV3}	$\overline{S2:0}$ ($\overline{DEN}$), DT/$\overline{R}$, $\overline{BHE}$ ($\overline{RFSH}$), A19:16	3	30	3	30	ns	1
T_{CLOV1}	$\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT	3	27	3	27	ns	1, 4, 6
T_{CLOV2}	$\overline{RD}$, $\overline{WR}$, $\overline{MCS3:0}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{PCS6:0}$, $\overline{INTA1:0}$	3	32	3	35	ns	1, 4, 6
T_{CLOV3}	$\overline{BHE}$ ($\overline{RFSH}$), $\overline{DEN}$, A19:16	3	30	3	30	ns	1, 4, 6
T_{CLOV4}	AD15:0 (A15:8, AD7:0)	3	34	3	35	ns	1, 4, 6
T_{CLOV5}	$\overline{S2:0}$	3	38	3	40	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	27	0	27	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (A15:8, AD7:0)	0	27	0	27	ns	1

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measured at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.
11. T_{CHOVS} applies to $\overline{BHE}$ ($\overline{RFSH}$) and A19:16 only after a HOLD release.
12. Pin names in parentheses apply to the 80C188EA/80L188EA.

AC SPECIFICATIONS

AC Characteristics—80L186EA13/80L186EA8

Symbol	Parameter	Min	Max	Min	Max	Units	Notes
SYNCHRONOUS INPUTS		**13 MHz**		**8 MHz**			
T_{CHIS}	$\overline{TEST}$, NMI, INT3:0, T1:0IN, ARDY	22		22		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, INT3:0, T1:0IN, ARDY	3		3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), ARDY, SRDY, DRQ1:0	22		22		ns	1, 10
T_{CLIH}	AD15:0 (AD7:0), ARDY, SRDY, DRQ1:0	3		3		ns	1, 10
T_{CLIS}	HOLD	22		22		ns	1, 9
T_{CLIH}	HOLD	3		3		ns	1, 9
T_{CLIS}	$\overline{RESIN}$ (to CLKIN)	22		22		ns	1, 9
T_{CLIH}	$\overline{RESIN}$ (from CLKIN)	3		3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measured at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.
11. T_{CHOVS} applies to $\overline{BHE}$ ($\overline{RFSH}$) and A19:16 only after a HOLD release.
12. Pin names in parentheses apply to the 80C188EA/80L188EA.

AC SPECIFICATIONS (Continued)

Relative Timings (80C186EA25/20/13, 80L186EA13/8)

Symbol	Parameter	Min	Max	Unit	Notes
RELATIVE TIMINGS					
T_{LHLL}	ALE Rising to ALE Falling	T − 15		ns	
T_{AVLL}	Address Valid to ALE Falling	½T − 10		ns	
T_{PLLL}	Chip Selects Valid to ALE Falling	½T − 10		ns	1
T_{LLAX}	Address Hold from ALE Falling	½T − 10		ns	
T_{LLWL}	ALE Falling to $\overline{WR}$ Falling	½T − 15		ns	1
T_{LLRL}	ALE Falling to $\overline{RD}$ Falling	½T − 15		ns	1
T_{RHLH}	$\overline{RD}$ Rising to ALE Rising	½T − 10		ns	1
T_{WHLH}	$\overline{WR}$ Rising to ALE Rising	½T − 10		ns	1
T_{AFRL}	Address Float to $\overline{RD}$ Falling	0		ns	
T_{RLRH}	$\overline{RD}$ Falling to $\overline{RD}$ Rising	(2*T) − 5		ns	2
T_{WLWH}	$\overline{WR}$ Falling to $\overline{WR}$ Rising	(2*T) − 5		ns	2
T_{RHAV}	$\overline{RD}$ Rising to Address Active	T − 15		ns	
T_{WHDX}	Output Data Hold after $\overline{WR}$ Rising	T − 15		ns	
T_{WHDEX}	$\overline{WR}$ Rising to $\overline{DEN}$ Rising	½T − 10		ns	1
T_{WHPH}	$\overline{WR}$ Rising to Chip Select Rising	½T − 10		ns	1, 4
T_{RHPH}	$\overline{RD}$ Rising to Chip Select Rising	½T − 10		ns	1, 4
T_{PHPL}	$\overline{CS}$ Inactive to $\overline{CS}$ Active	½T − 10		ns	1
T_{DXDL}	$\overline{DEN}$ Inactive to DT/$\overline{R}$ Low	0		ns	5
T_{OVRH}	ONCE ($\overline{UCS}$, $\overline{LCS}$) Active to $\overline{RESIN}$ Rising	T		ns	3
T_{RHOX}	ONCE ($\overline{UCS}$, $\overline{LCS}$) to $\overline{RESIN}$ Rising	T		ns	3

NOTES:
1. Assumes equal loading on both pins.
2. Can be extended using wait states.
3. Not tested.
4. Not applicable to latched A2:1. These signals change only on falling T_1.
5. For write cycle followed by read cycle.
6. Operating conditions for 25 MHz are 0°C to +70°C, V_{CC} = 5.0V ±10%.

AC TEST CONDITIONS

The AC specifications are tested with the 50 pF load shown in Figure 8. See the Derating Curves section to see how timings vary with load capacitance.

Specifications are measured at the $V_{CC}/2$ crossing point, unless otherwise specified. See AC Timing Waveforms, for AC specification definitions, test pins, and illustrations.

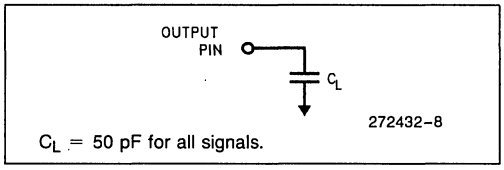


Figure 8. AC Test Load

AC TIMING WAVEFORMS

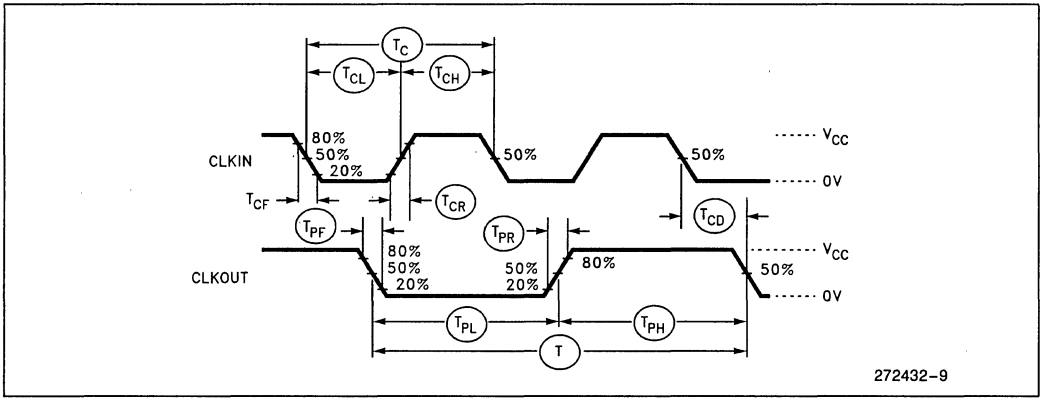
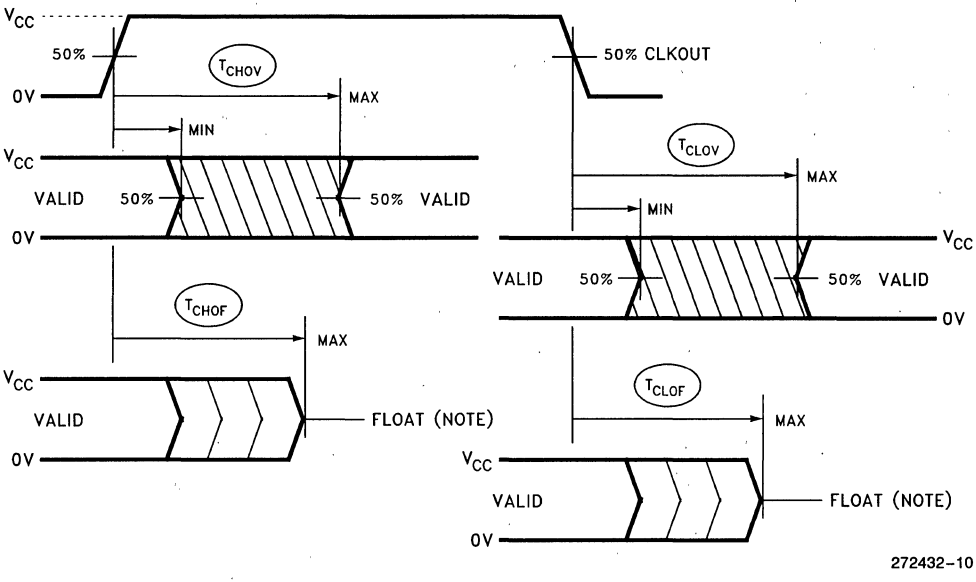
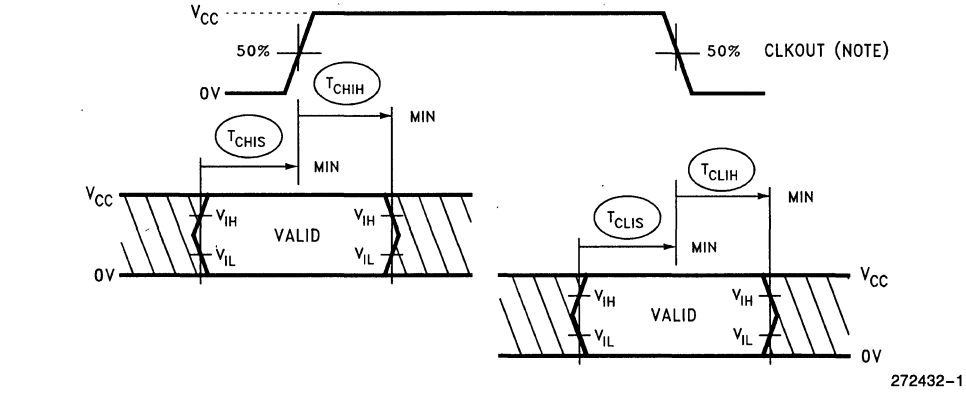


Figure 9. Input and Output Clock Waveform



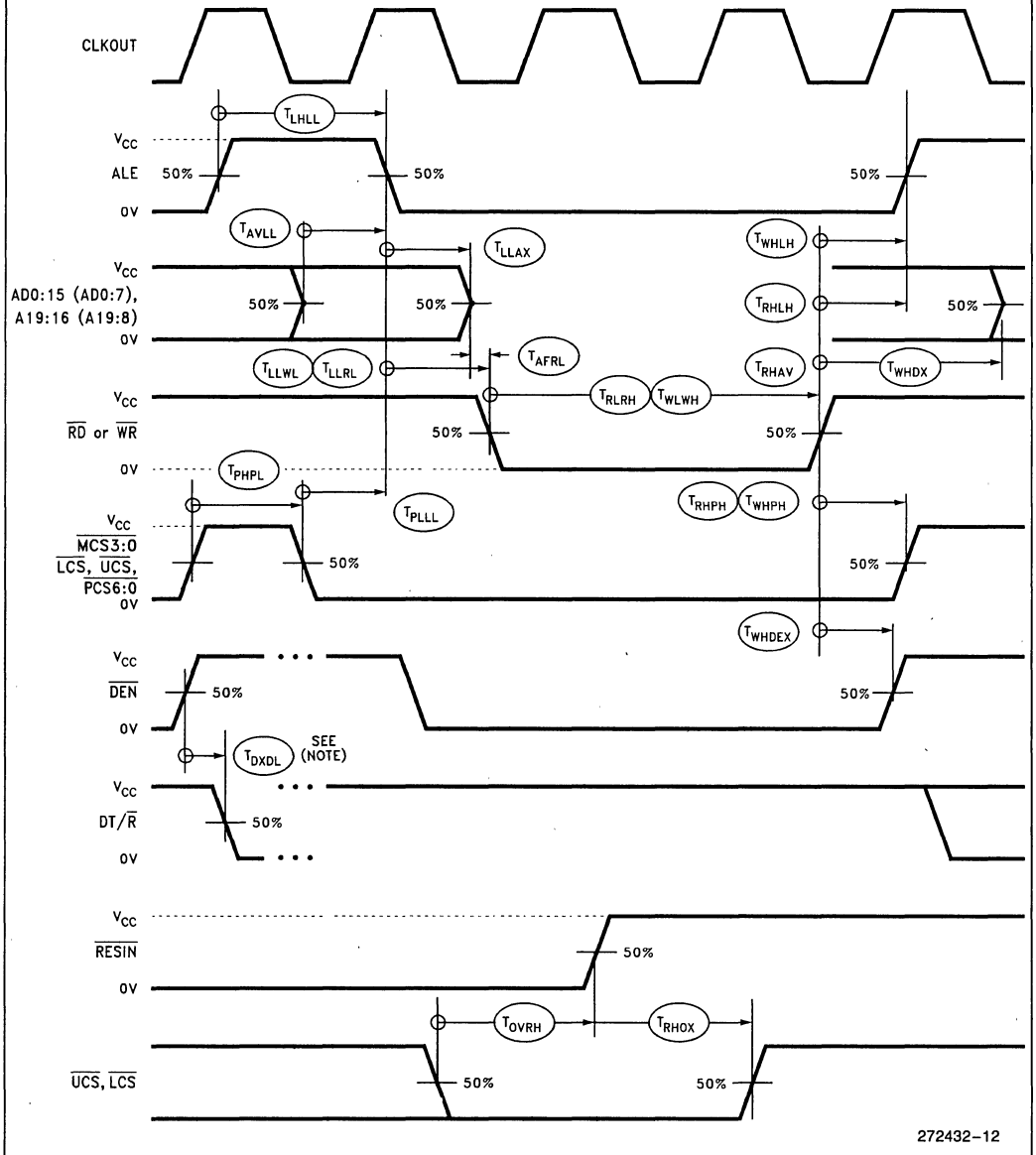
NOTE:
20% V_{CC} < Float < 80% V_{CC}

Figure 10. Output Delay and Float Waveform



NOTE:
$\overline{RESIN}$ measured to CLKIN, not CLKOUT

Figure 11. Input Setup and Hold



NOTES:
1. T_{DXDL} for write cycle followed by read cycle.
2. Pin names in parentheses apply to tthe 80C188EA.

Figure 12. Relative Signal Waveform

DERATING CURVES

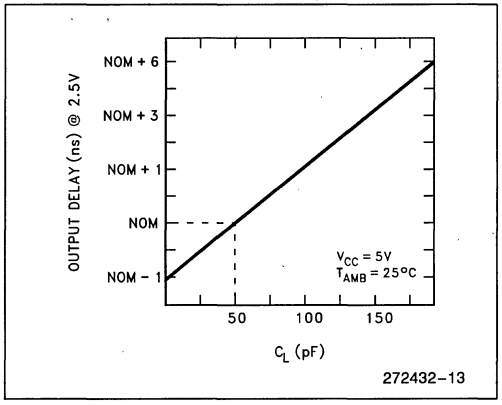


Figure 13. Typical Output Delay Variations Versus Load Capacitance

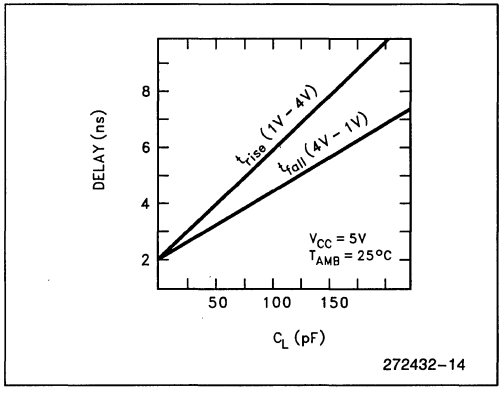


Figure 14. Typical Rise and Fall Variations Versus Load Capacitance

RESET

The processor performs a reset operation any time the $\overline{\text{RESIN}}$ pin is active. The $\overline{\text{RESIN}}$ pin is actually synchronized before it is presented internally, which means that the clock must be operating before a reset can take effect. From a power-on state, $\overline{\text{RESIN}}$ must be held active (low) in order to guarantee correct initialization of the processor. **Failure to provide $\overline{\text{RESIN}}$ while the device is powering up will result in unspecified operation of the device.**

Figure 15 shows the correct reset sequence when first applying power to the processor. An external clock connected to CLKIN must not exceed the V_{CC} threshold being applied to the processor. This is normally not a problem if the clock driver is supplied with the same V_{CC} that supplies the processor. When attaching a crystal to the device, $\overline{\text{RESIN}}$ must remain active until both V_{CC} and CLKOUT are stable (the length of time is application specific and depends on the startup characteristics of the crystal circuit). The $\overline{\text{RESIN}}$ pin is designed to operate correctly using an RC reset circuit, but the designer must ensure that the ramp time for V_{CC} is not so long that $\overline{\text{RESIN}}$ is never really sampled at a logic low level when V_{CC} reaches minimum operating conditions.

Figure 16 shows the timing sequence when $\overline{\text{RESIN}}$ is applied after V_{CC} is stable and the device has been operating. Note that a reset will terminate all activity and return the processor to a known operating state. Any bus operation that is in progress at the time $\overline{\text{RESIN}}$ is asserted will terminate immediately (note that most control signals will be driven to their inactive state first before floating).

While $\overline{\text{RESIN}}$ is active, signals $\overline{\text{RD}}$/QSMD, $\overline{\text{UCS}}$, $\overline{\text{LCS}}$, $\overline{\text{MCS0}}$/PEREQ, $\overline{\text{MCS1}}$/$\overline{\text{ERROR}}$, $\overline{\text{LOCK}}$, and $\overline{\text{TEST}}$/BUSY are configured as inputs and weakly held high by internal pullup transistors. Forcing $\overline{\text{UCS}}$ and $\overline{\text{LCS}}$ low selects ONCE Mode. Forcing QSMD low selects Queue Status Mode. Forcing $\overline{\text{TEST}}$/BUSY high at reset and low four clocks later enables Numerics Mode. Forcing $\overline{\text{LOCK}}$ low is prohibited and results in unspecified operation.

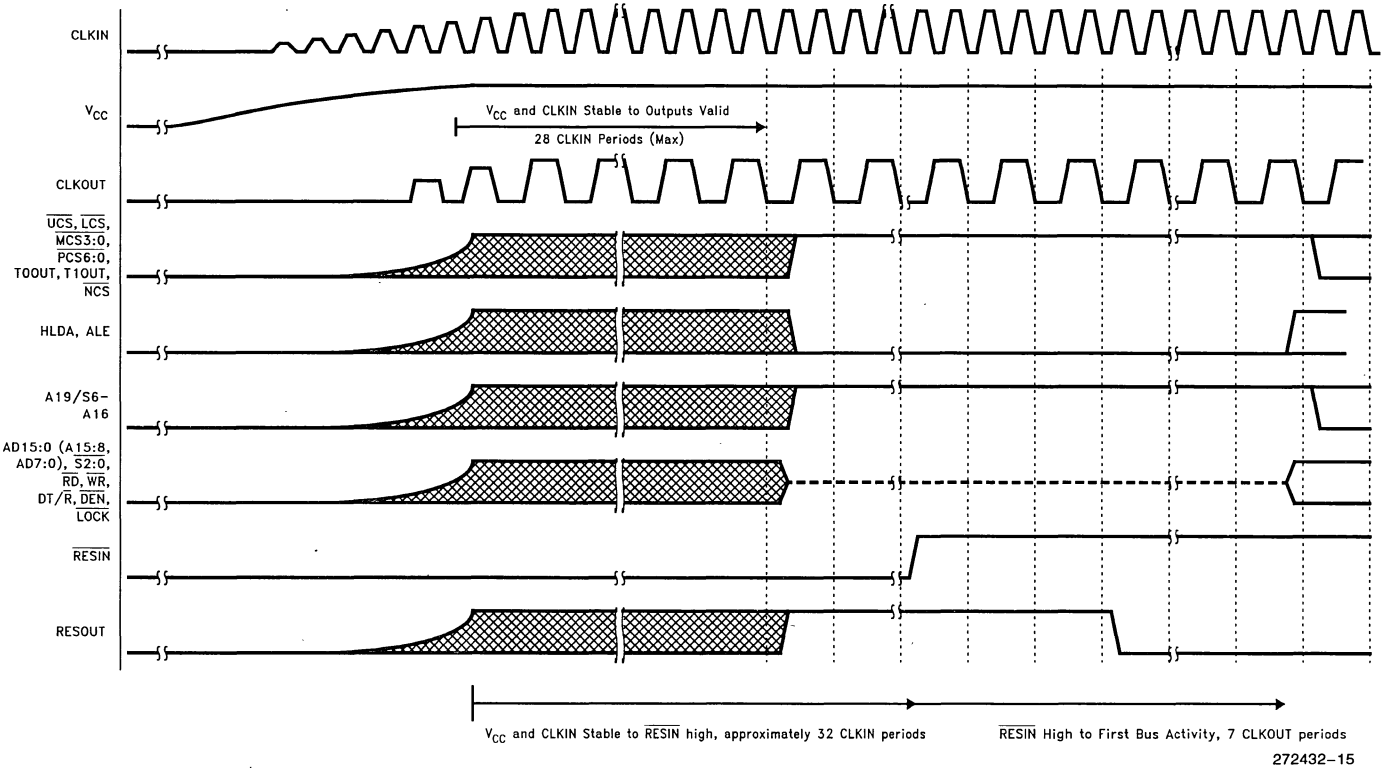
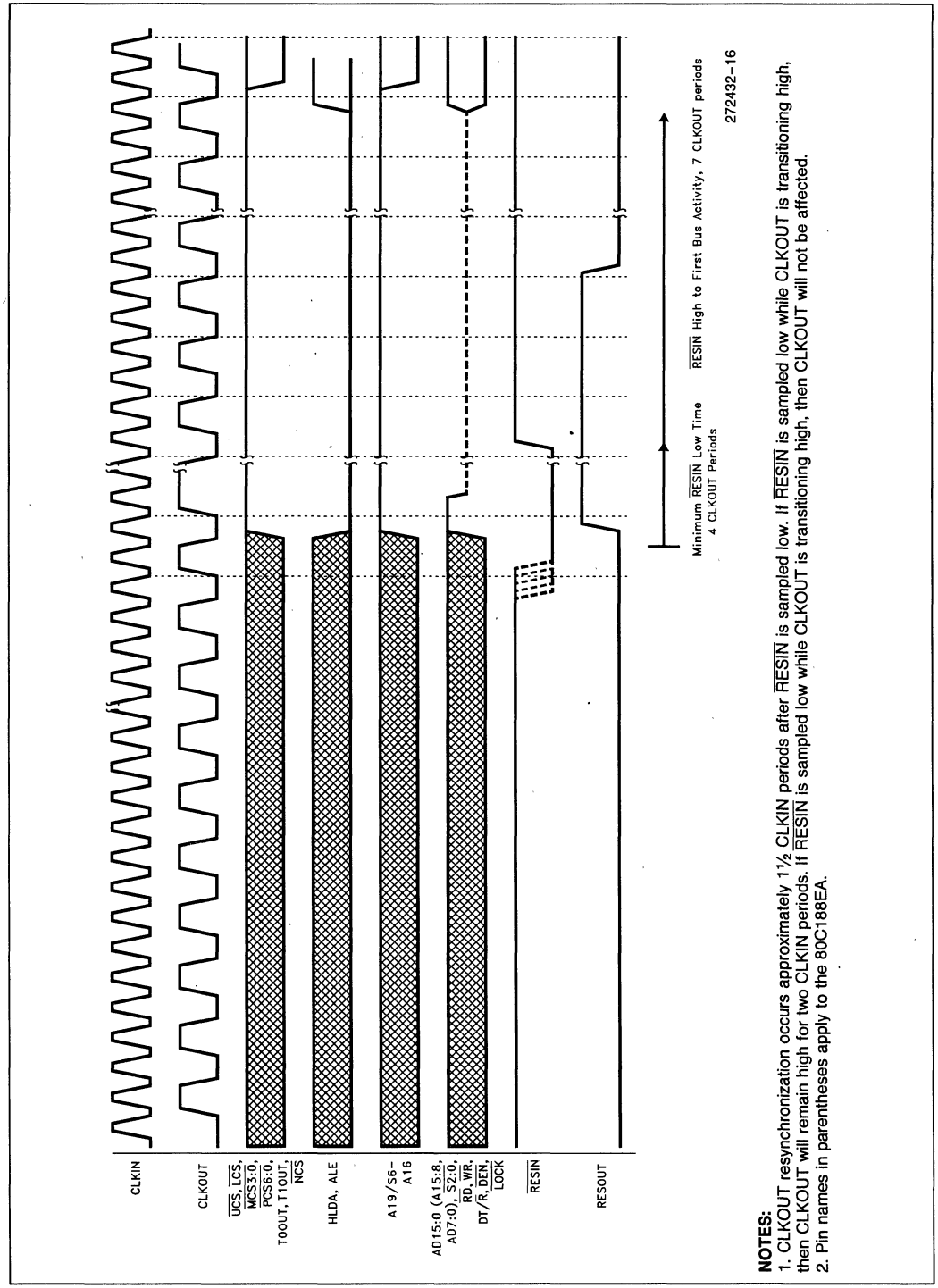


Figure 15. Powerup Reset Waveforms

NOTES:
1. CLKOUT synchronization occurs approximately 1½ CLKIN periods after $\overline{RESIN}$ is sampled low.
2. Pin names in parentheses apply to the 80C188EA.

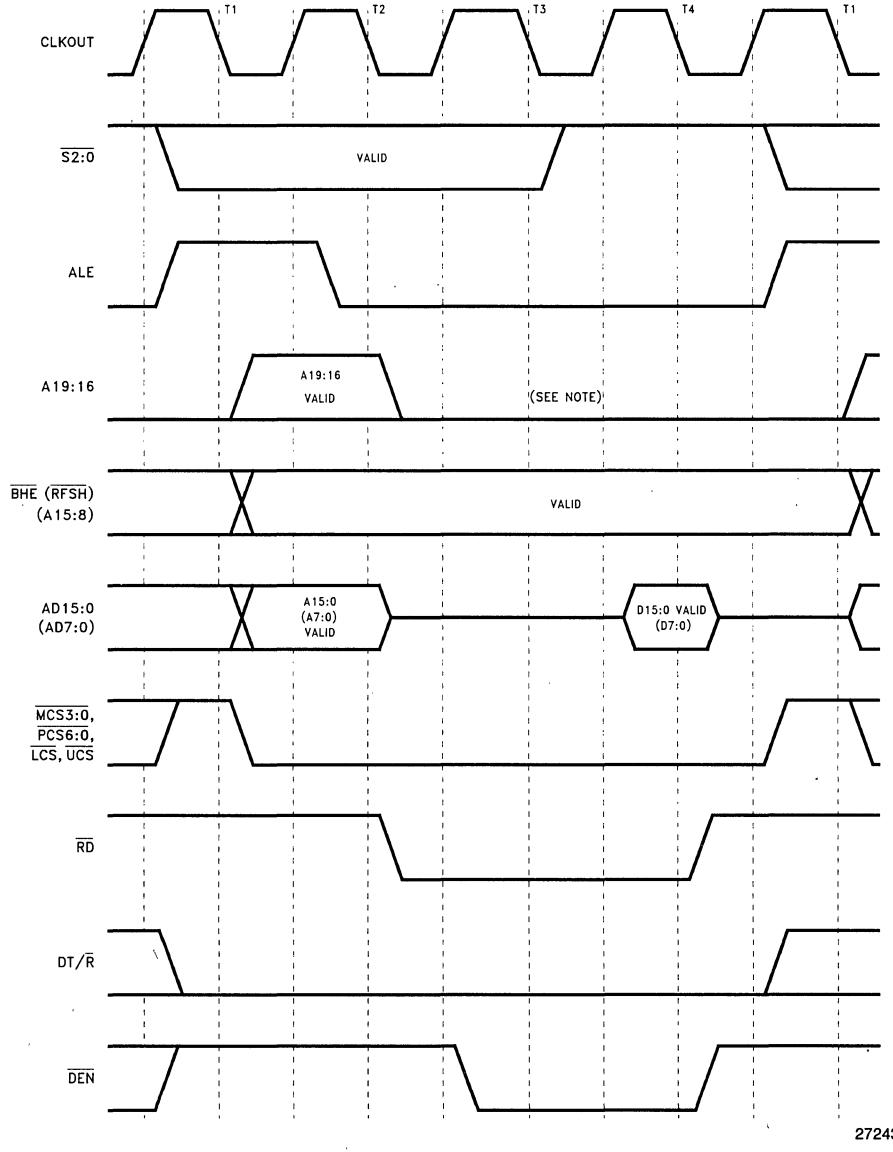


NOTES:
1. CLKOUT resynchronization occurs approximately 1½ CLKIN periods after $\overline{\text{RESIN}}$ is sampled low. If $\overline{\text{RESIN}}$ is sampled low while CLKOUT is transitioning high, then CLKOUT will remain high for two CLKIN periods. If $\overline{\text{RESIN}}$ is sampled low while CLKOUT is transitioning high, then CLKOUT will not be affected.
2. Pin names in parentheses apply to the 80C188EA.

Figure 16. Warm Reset Waveforms

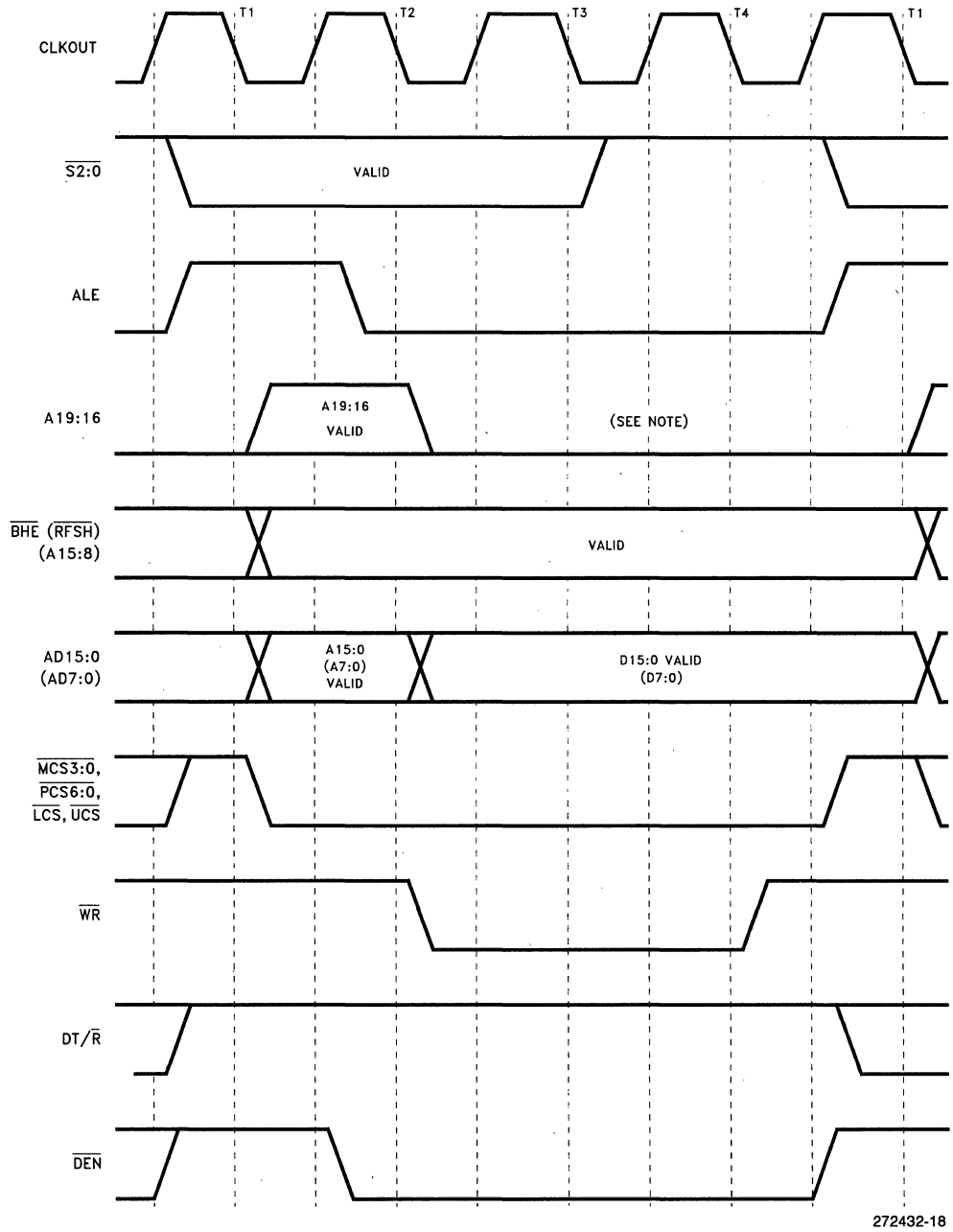
BUS CYCLE WAVEFORMS

Figures 17 through 23 present the various bus cycles that are generated by the processor. What is shown in the figure is the relationship of the various bus signals to CLKOUT. These figures along with the information present in **AC Specifications** allow the user to determine all the critical timing analysis needed for a given application.



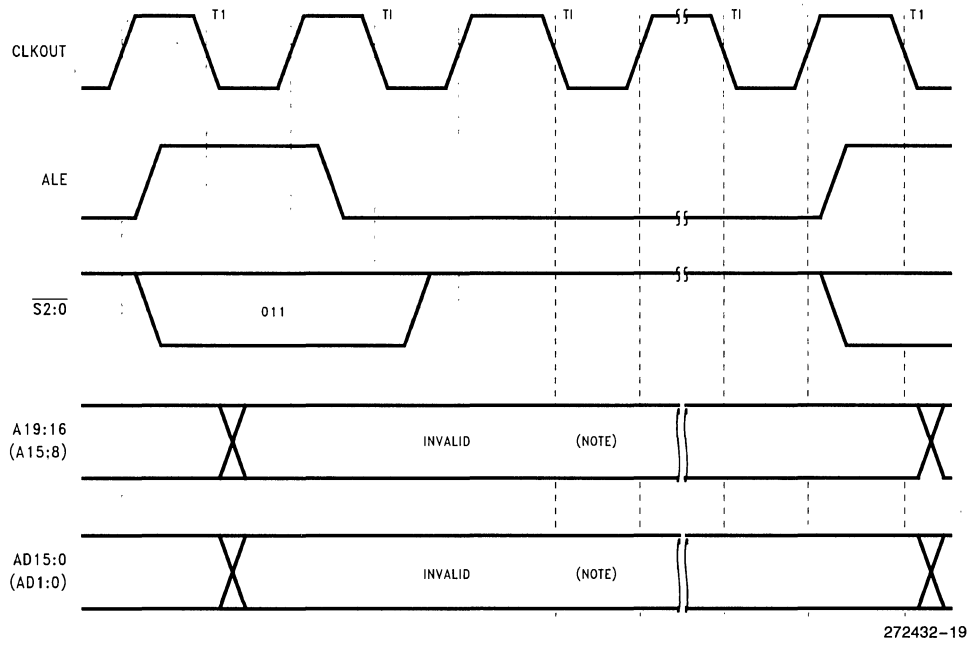
NOTES:
1. During the data phase of the bus cycle, A19/S6 is driven high for a DMA or refresh cycle.
2. Pin names in parentheses apply to the 80C188EA.

Figure 17. Read, Fetch and Refresh Cycle Waveform



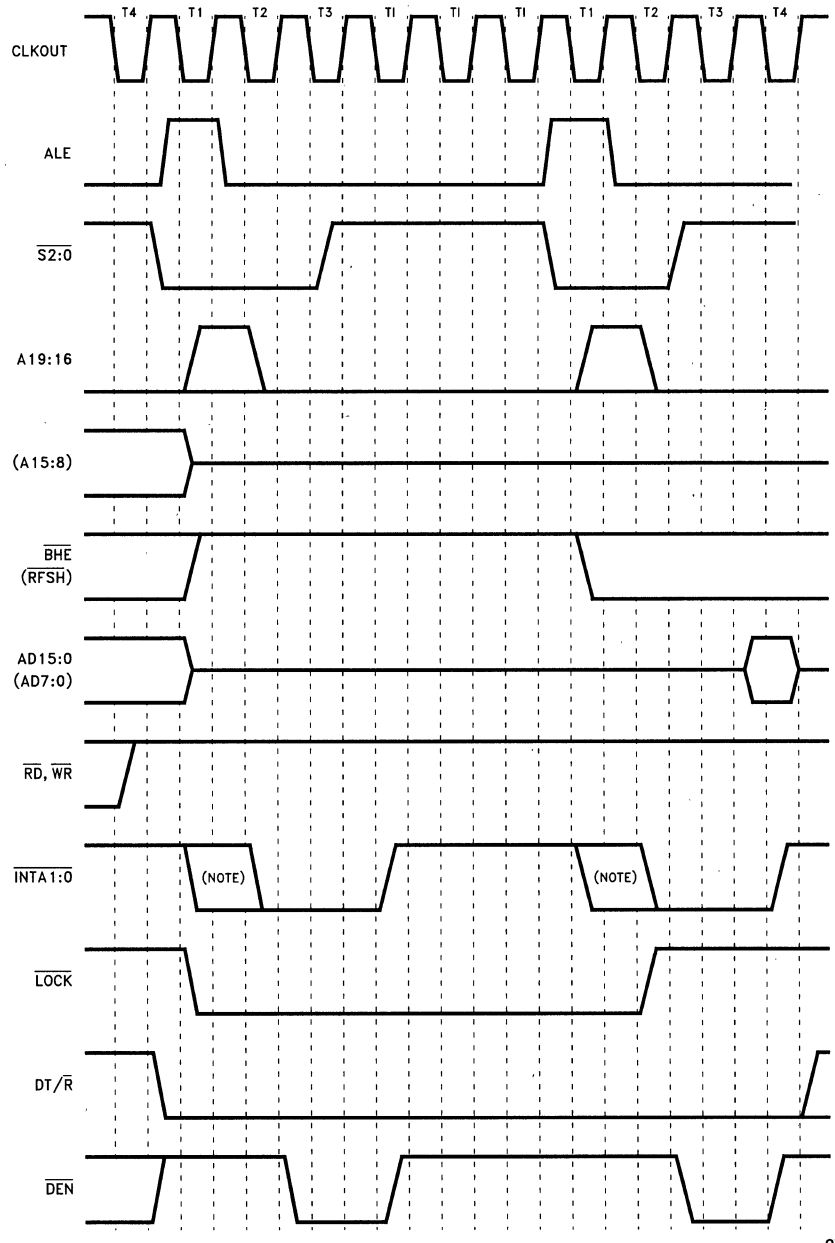
NOTES:
1. During the data phase of the bus cycle, A19/S6 is driven high for a DMA cycle.
2. Pin names in parentheses apply to the 80C188EA.

Figure 18. Write Cycle Waveform



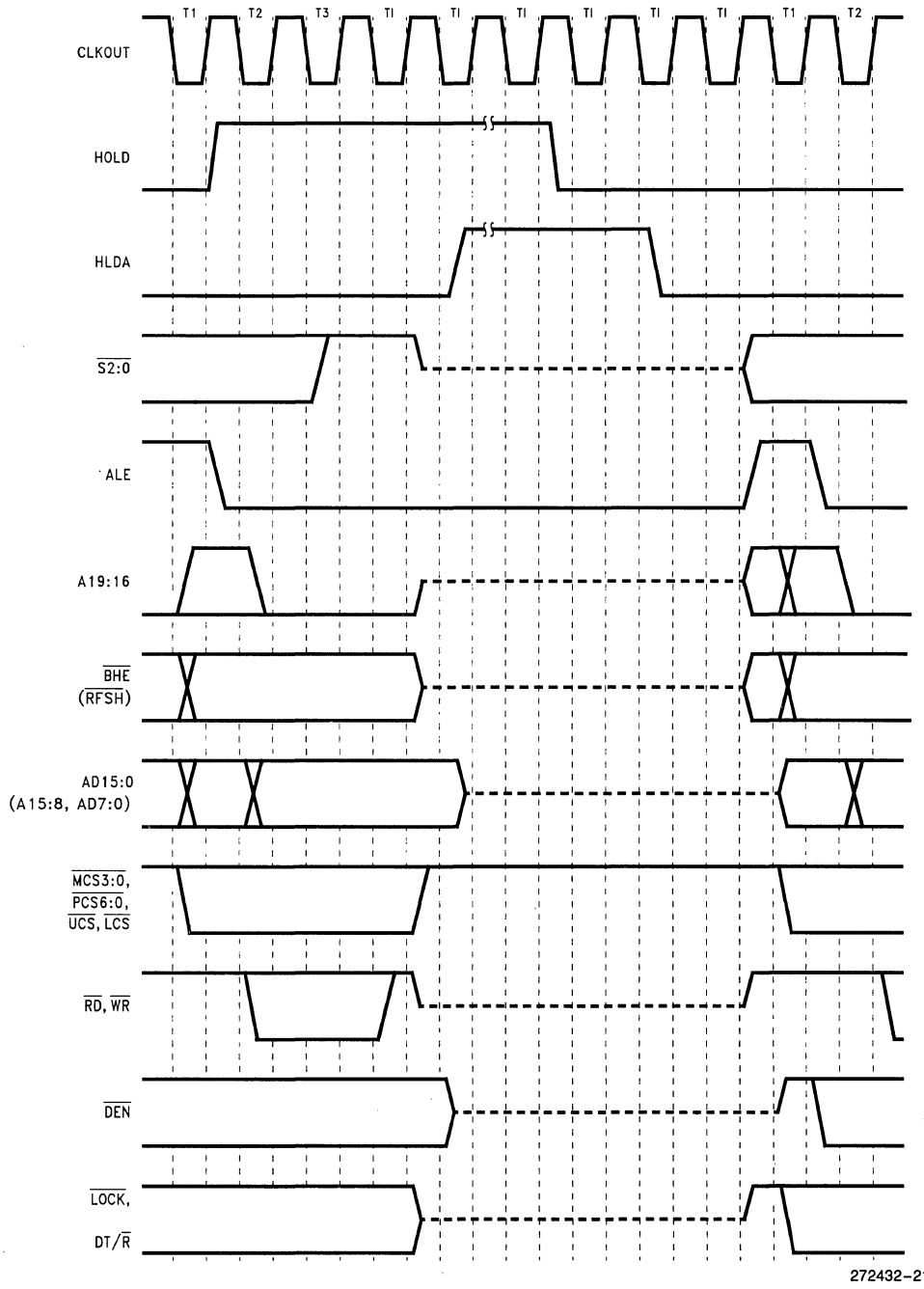
NOTES:
1. The processor drives these pins to 0 during Idle and Powerdown Modes.
2. Pin names in parentheses apply to the 80C188EA.

Figure 19. Halt Cycle Waveform



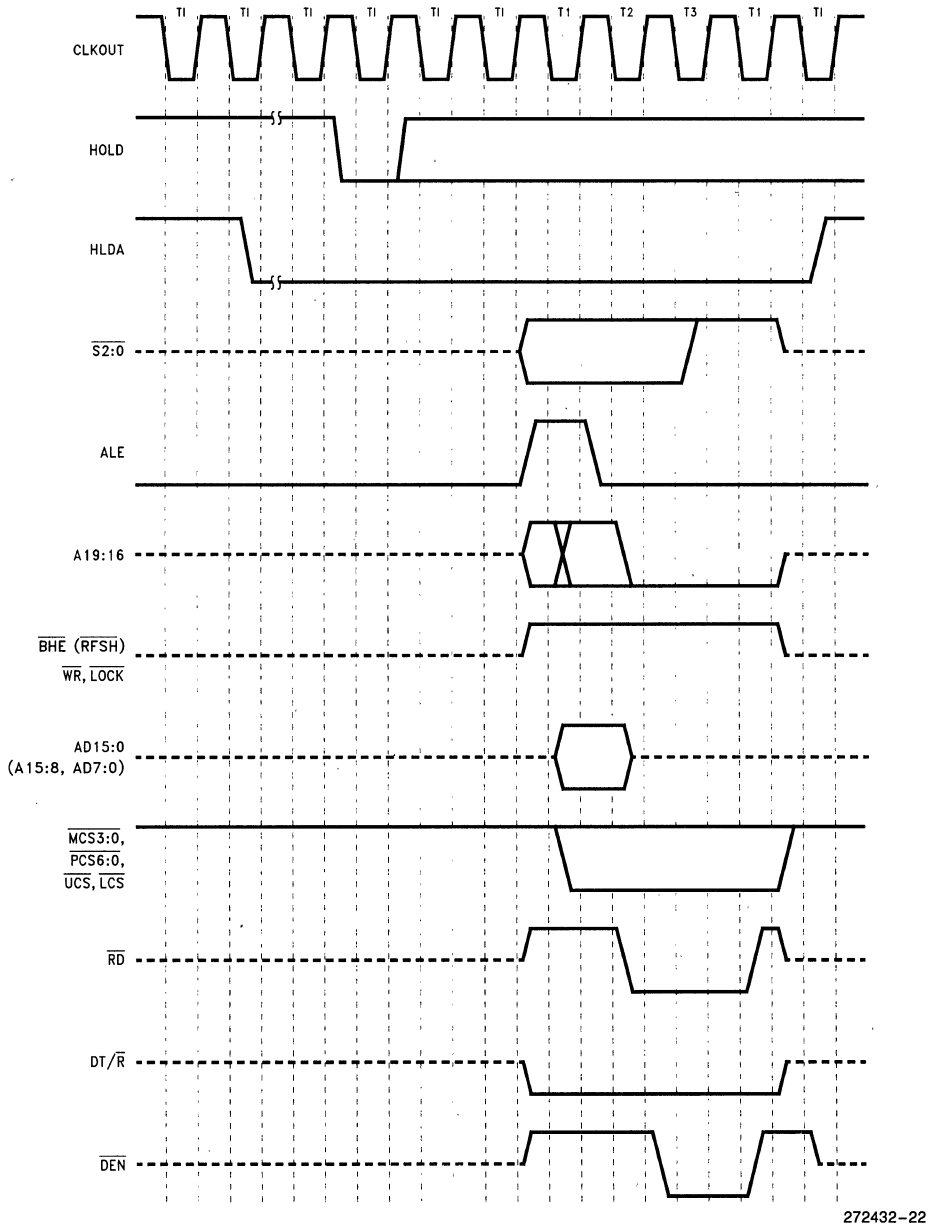
NOTES:
1. INTA occurs one clock later in Slave Mode.
2. Pin names in parentheses apply to the 80C188EA.

Figure 20. INTA Cycle Waveform



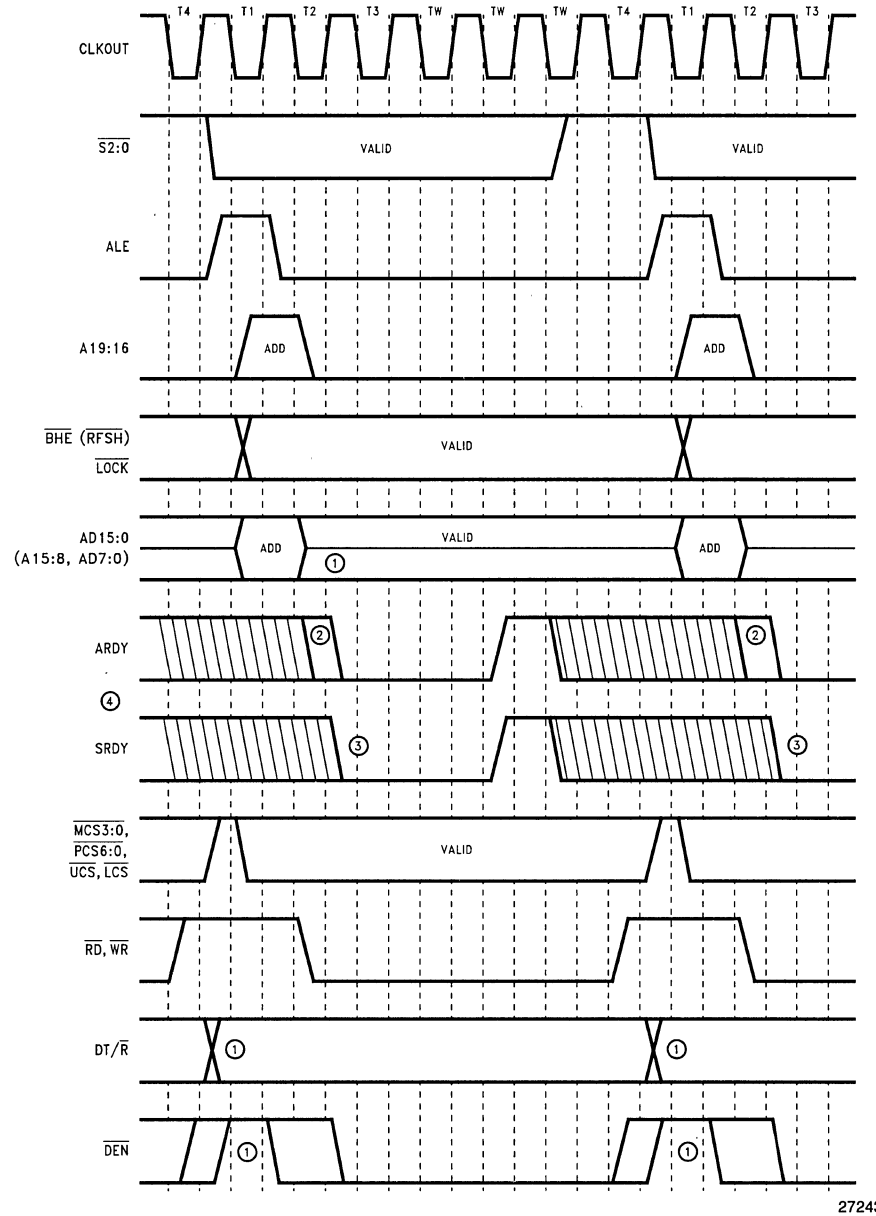
NOTE:
1. Pin names in parentheses apply to the 80C188EA.

Figure 21. HOLD/HLDA Waveform



NOTE:
1. Pin names in parentheses apply to the 80C188EA.

Figure 22. DRAM Refresh Cycle During Hold Acknowledge



NOTES:
1. Generalized diagram for READ or WRITE.
2. ARDY low by either edge causes a wait state. Only rising ARDY is fully synchronized.
3. SRDY low causes a wait state. SRDY must meet setup and hold times to ensure correct device operation.
4. Either ARDY or SRDY active high will terminate a bus cycle.
5. Pin names in parentheses apply to the 80C188EA.

Figure 23. Ready Waveform

80C186EA/80C188EA EXECUTION TIMINGS

A determination of program exeuction timing must consider the bus cycles necessary to prefetch instructions as well as the number of execution unit cycles necessary to execute instructions. The following instruction timings represent the **minimum** execution time in clock cycle for each instruction. The timings given are based on the following assumptions:

- The opcode, along with any data or displacement required for execution of a particular instruction, has been prefetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDs occur.
- All word-data is located on even-address boundaries. (80C186EA only)

All jumps and calls include the time required to fetch the opcode of the next instruction at the destination address.

All instructions which involve memory accesses can require one or two additional clocks above the minimum timings shown due to the asynchronous handshake between the bus interface unit (BIU) and execution unit.

With a 16-bit BIU, the 80C186EA has sufficient bus performance to endure that an adequate number of prefetched bytes will reside in the queue (6 bytes) most of the time. Therefore, actual program exeuction time will not be substanially greater than that derived from adding the instruction timings shown.

The 80C188EA 8-bit BIU is limited in its performance relative to the execution unit. A sufficient number of prefetched bytes may not reside in the prefetch queue (4 bytes) much of the time. Therefore, actual program execution time will be substantially greater than that derived from adding the instruction timings shown.

INSTRUCTION SET SUMMARY

Function	Format				80C186EA Clock Cycles	80C188EA Clock Cycles	Comments
DATA TRANSFER							
MOV = Move:							
Register to Register/Memory	1000100w	mod reg r/m			2/12	2/12*	
Register/memory to register	1000101w	mod reg r/m			2/9	2/9	
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1	12–13	12–13	8/16-bit
Immediate to register	1011w reg	data	data if w=1		3–4	3–4	8/16-bit
Memory to accumulator	1010000w	addr-low	addr-high		8	8*	
Accumulator to memory	1010001w	addr-low	addr-high		9	9*	
Register/memory to segment register	10001110	mod 0 reg r/m			2/9	2/13	
Segment register to register/memory	10001100	mod 0 reg r/m			2/11	2/15	
PUSH = Push:							
Memory	11111111	mod 1 1 0 r/m			16	20	
Register	01010 reg				10	14	
Segment register	000reg110				9	13	
Immediate	011010s0	data	data if s=0		10	14	
PUSHA = Push All	01100000				36	68	
POP = Pop:							
Memory	10001111	mod 0 0 0 r/m			20	24	
Register	01011 reg				10	14	
Segment register	000reg111	(reg≠01)			8	12	
POPA = Pop All	01100001				51	83	
XCHG = Exchange:							
Register/memory with register	1000011w	mod reg r/m			4/17	4/17*	
Register with accumulator	10010 reg				3	3	
IN = Input from:							
Fixed port	1110010w	port			10	10*	
Variable port	1110110w				8	7*	
OUT = Output to:							
Fixed port	1110011w	port			9	9*	
Variable port	1110111w				7	7*	
XLAT = Translate byte to AL	11010111				11	15	
LEA = Load EA to register	10001101	mod reg r/m			6	6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod≠11)		18	26	
LES = Load pointer to ES	11000100	mod reg r/m	(mod≠11)		18	26	
LAHF = Load AH with flags	10011111				2	2	
SAHF = Store AH into flags	10011110				3	3	
PUSHF = Push flags	10011100				9	13	
POPF = Pop flags	10011101				8	12	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EA Clock Cycles	80C188EA Clock Cycles	Comments
DATA TRANSFER (Continued)							
SEGMENT = Segment Override:							
CS	00101110				2	2	
SS	00110110				2	2	
DS	00111110				2	2	
ES	00100110				2	2	
ARITHMETIC							
ADD = Add:							
Reg/memory with register to either	000000dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 000 r/m	data	data if s w=01	4/16	4/16*	
Immediate to accumulator	0000010w	data	data if w=1		3/4	3/4	8/16-bit
ADC = Add with carry:							
Reg/memory with register to either	000100dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 010 r/m	data	data if s w=01	4/16	4/16*	
Immediate to accumulator	0001010w	data	data if w=1		3/4	3/4	8/16-bit
INC = Increment:							
Register/memory	1111111w	mod 000 r/m			3/15	3/15*	
Register	01000 reg				3	3	
SUB = Subtract:							
Reg/memory and register to either	001010dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 101 r/m	data	data if s w=01	4/16	4/16*	
Immediate from accumulator	0010110w	data	data if w=1		3/4	3/4	8/16-bit
SBB = Subtract with borrow:							
Reg/memory and register to either	000110dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 011 r/m	data	data if s w=01	4/16	4/16*	
Immediate from accumulator	0001110w	data	data if w=1		3/4	3/4*	8/16-bit
DEC = Decrement							
Register/memory	1111111w	mod 001 r/m			3/15	3/15*	
Register	01001 reg				3	3	
CMP = Compare:							
Register/memory with register	0011101w	mod reg r/m			3/10	3/10*	
Register with register/memory	0011100w	mod reg r/m			3/10	3/10*	
Immediate with register/memory	100000sw	mod 111 r/m	data	data if s w=01	3/10	3/10*	
Immediate with accumulator	0011110w	data	data if w=1		3/4	3/4	8/16-bit
NEG = Change sign register/memory	1111011w	mod 011 r/m			3/10*	3/10*	
AAA = ASCII adjust for add	00110111				8	8	
DAA = Decimal adjust for add	00100111				4	4	
AAS = ASCII adjust for subtract	00111111				7	7	
DAS = Decimal adjust for subtract	00101111				4	4	
MUL = Multiply (unsigned):	1111011w	mod 100 r/m					
Register-Byte					26–28	26–28	
Register-Word					35–37	35–37	
Memory-Byte					32–34	32–34	
Memory-Word					41–43	41–48*	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186EA Clock Cycles	80C188EA Clock Cycles	Comments
ARITHMETIC (Continued)				
IMUL = Integer multiply (signed):	1111011w | mod101 r/m			
Register-Byte		25–28	25–28	
Register-Word		34–37	34–37	
Memory-Byte		31–34	32–34	
Memory-Word		40–43	40–43*	
IMUL = Integer Immediate multiply (signed)	011010s1 | mod reg r/m | data | data if s=0	22–25 29–32	22-25 29–32	
DIV = Divide (unsigned):	1111011w | mod110 r/m			
Register-Byte		29	29	
Register-Word		38	38	
Memory-Byte		35	35	
Memory-Word		44	44*	
IDIV = Integer divide (signed):	1111011w | mod111 r/m			
Register-Byte		44–52	44–52	
Register-Word		53–61	53–61	
Memory-Byte		50–58	50–58	
Memory-Word		59–67	59–67*	
AAM = ASCII adjust for multiply	11010100 | 00001010	19	19	
AAD = ASCII adjust for divide	11010101 | 00001010	15	15	
CBW = Convert byte to word	10011000	2	2	
CWD = Convert word to double word	10011001	4	4	
LOGIC				
Shift/Rotate Instructions:				
Register/Memory by 1	1101000w | mod TTT r/m	2/15	2/15	
Register/Memory by CL	1101001w | mod TTT r/m	5+n/17+n	5+n/17+n	
Register/Memory by Count	1100000w | mod TTT r/m | count	5+n/17+n	5+n/17+n	
	TTT Instruction 000 ROL 001 ROR 010 RCL 011 RCR 100 SHL/SAL 101 SHR 111 SAR			
AND = **And:**				
Reg/memory and register to either	001000dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod100 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0010010w | data | data if w=1	3/4	3/4*	8/16-bit
TEST = **And function to flags, no result:**				
Register/memory and register	1000010w | mod reg r/m	3/10	3/10*	
Immediate data and register/memory	1111011w | mod000 r/m | data | data if w=1	4/10	4/10*	
Immediate data and accumulator	1010100w | data | data if w=1	3/4	3/4	8/16-bit
OR = **Or:**				
Reg/memory and register to either	000010dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod001 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0000110w | data | data if w=1	3/4	3/4*	8/16-bit

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186EA Clock Cycles	80C188EA Clock Cycles	Comments
LOGIC (Continued)				
XOR = Exclusive or:				
Reg/memory and register to either	001100dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod 110 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0011010w | data | data if w=1	3/4	3/4	8/16-bit
NOT = Invert register/memory	1111011w | mod 010 r/m	3/10	3/10*	
STRING MANIPULATION				
MOVS = Move byte/word	1010010w	14	14*	
CMPS = Compare byte/word	1010011w	22	22*	
SCAS = Scan byte/word	1010111w	15	15*	
LODS = Load byte/wd to AL/AX	1010110w	12	12*	
STOS = Store byte/wd from AL/AX	1010101w	10	10*	
INS = Input byte/wd from DX port	0110110w	14	14	
OUTS = Output byte/wd to DX port	0110111w	14	14	
Repeated by count in CX (REP/REPE/REPZ/REPNE/REPNZ)				
MOVS = Move string	11110010 | 1010010w	8+8n	8+8n*	
CMPS = Compare string	1111001z | 1010011w	5+22n	5+22n	
SCAS = Scan string	1111001z | 1010111w	5+15n	5+15n*	
LODS = Load string	11110010 | 1010110w	6+11n	6+11n*	
STOS = Store string	11110010 | 1010101w	6+9n	6+9n*	
INS = Input string	11110010 | 0110110w	8+8n	8+8n*	
OUTS = Output string	11110010 | 0110111w	8+8n	8+8n*	
CONTROL TRANSFER				
CALL = Call:				
Direct within segment	11101000 | disp-low | disp-high	15	19	
Register/memory indirect within segment	11111111 | mod 010 r/m	13/19	17/27	
Direct intersegment	10011010 | segment offset | segment selector	23	31	
Indirect intersegment	11111111 | mod 011 r/m | (mod ≠ 11)	38	54	
JMP = Unconditional jump:				
Short/long	11101011 | disp-low	14	14	
Direct within segment	11101001 | disp-low | disp-high	14	14	
Register/memory indirect within segment	11111111 | mod 100 r/m	11/17	11/21	
Direct intersegment	11101010 | segment offset | segment selector	14	14	
Indirect intersegment	11111111 | mod 101 r/m | (mod ≠ 11)	26	34	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EA Clock Cycles	80C188EA Clock Cycles	Comments
CONTROL TRANSFER (Continued)							
RET = Return from CALL:							
Within segment	11000011				16	20	
Within seg adding immed to SP	11000010	data-low	data-high		18	22	
Intersegment	11001011				22	30	
Intersegment adding immediate to SP	11001010	data-low	data-high		25	33	
JE/JZ = Jump on equal/zero	01110100	disp			4/13	4/13	JMP not taken/JMP taken
JL/JNGE = Jump on less/not greater or equal	01111100	disp			4/13	4/13	
JLE/JNG = Jump on less or equal/not greater	01111110	disp			4/13	4/13	
JB/JNAE = Jump on below/not above or equal	01110010	disp			4/13	4/13	
JBE/JNA = Jump on below or equal/not above	01110110	disp			4/13	4/13	
JP/JPE = Jump on parity/parity even	01111010	disp			4/13	4/13	
JO = Jump on overflow	01110000	disp			4/13	4/13	
JS = Jump on sign	01111000	disp			4/13	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101	disp			4/13	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101	disp			4/13	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111	disp			4/13	4/13	
JNB/JAE = Jump on not below/above or equal	01110011	disp			4/13	4/13	
JNBE/JA = Jump on not below or equal/above	01110111	disp			4/13	4/13	
JNP/JPO = Jump on not par/par odd	01111011	disp			4/13	4/13	
JNO = Jump on not overflow	01110001	disp			4/13	4/13	
JNS = Jump on not sign	01111001	disp			4/13	4/13	
JCXZ = Jump on CX zero	11100011	disp			5/15	5/15	
LOOP = Loop CX times	11100010	disp			6/16	6/16	LOOP not taken/LOOP taken
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp			6/16	6/16	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp			6/16	6/16	
ENTER = Enter Procedure	11001000	data-low	data-high	L			
L = 0					15	19	
L = 1					25	29	
L > 1					22+16(n−1)	26+20(n−1)	
LEAVE = Leave Procedure	11001001				8	8	
INT = Interrupt:							
Type specified	11001101	type			47	47	
Type 3	11001100				45	45	if INT. taken/ if INT. not taken
INTO = Interrupt on overflow	11001110				48/4	48/4	
IRET = Interrupt return	11001111				28	28	
BOUND = Detect value out of range	01100010	mod reg r/m			33–35	33–35	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186EA Clock Cycles	80C188EA Clock Cycles	Comments
PROCESSOR CONTROL				
CLC = Clear carry	11111000	2	2	
CMC = Complement carry	11110101	2	2	
STC = Set carry	11111001	2	2	
CLD = Clear direction	11111100	2	2	
STD = Set direction	11111101	2	2	
CLI = Clear interrupt	11111010	2	2	
STI = Set interrupt	11111011	2	2	
HLT = Halt	11110100	2	2	
WAIT = Wait	10011011	6	6	if $\overline{TEST}$ = 0
LOCK = Bus lock prefix	11110000	2	2	
NOP = No Operation	10010000	3	3	
	(TTT LLL are opcode to processor extension)			

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

The Effective Address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

EA calculation time is 4 clock cycles for all modes, and is included in the execution times given whenever appropriate.

Segment Override Prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

REVISION HISTORY

Intel 80C186EA/80L186EA devices are marked with a 9-character alphanumeric Intel FPO number underneath the product number. This data sheet update is valid for devices with an "A", "B", "C", "D", or "E" as the ninth character in the FPO number, as illustrated in Figure 5 for the 68-lead PLCC package, Figure 6 for the 84-lead QFP (EIAJ) package, and Figure 7 for the 80-lead SQFP device. Such devices may also be identified by reading a value of 01H, 02H, 03H from the STEPID register.

This data sheet replaces the following data sheets:

272019-002—80C186EA
272020-002—80C188EA
272021-002—80L186EA
272022-002—80L188EA
272307-001—SB80C186EA/SB80L186EA
272308-001—SB80C188EA/SB80L188EA

ERRATA

An 80C186EA/80L186EA with a STEPID value of 01H or 02H has the following known errata. A device with a STEPID of 01H or 02H can be visually identified by noting the presence of an "A", "B", or "C" alpha character, next to the FPO number. The FPO number location is shown in Figures 5, 6, and 7.

1. An internal condition with the interrupt controller can cause no acknowledge cycle on the INTA1 line in response to INT1. This errata only occurs when Interrupt 1 is configured in cascade mode and a higher priority interrupt exists. This errata will not occur consistantly, it is dependent on interrupt timing.

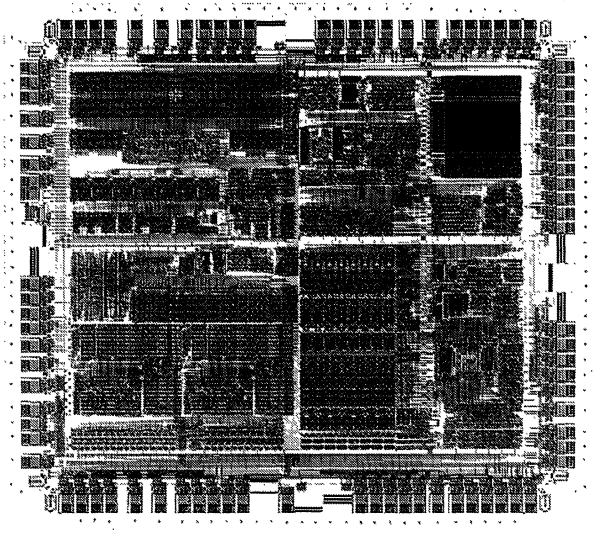
An 80C186EA/80L186EA with a STEPID value of 03H has no known errata. A device with a STEPID of 03H can be visually identified by noting the presence of a "D" or "E" alpha character next to the FPO number. The FPO number location is shown in Figures 5, 6, and 7.

80C186EB/80C188EB AND 80L186EB/80L188EB 16-BIT HIGH-INTEGRATION EMBEDDED PROCESSORS

- Full Static Operation
- True CMOS Inputs and Outputs

- Integrated Feature Set
  - Low-Power Static CPU Core
  - Two Independent UARTs each with an Integral Baud Rate Generator
  - Two 8-Bit Multiplexed I/O Ports
  - Programmable Interrupt Controller
  - Three Programmable 16-Bit Timer/Counters
  - Clock Generator
  - Ten Programmable Chip Selects with Integral Wait-State Generator
  - Memory Refresh Control Unit
  - System Level Testing Support (ONCE Mode)
- Direct Addressing Capability to 1 Mbyte Memory and 64 Kbyte I/O
- Speed Versions Available (5V):
  - 25 MHz (80C186EB25/80C188EB25)
  - 20 MHz (80C186EB20/80C188EB20)
  - 13 MHz (80C186EB13/80C188EB13)
- Available in Extended Temperature Range (−40°C to +85°C)
- Speed Versions Available (3V):
  - 16 MHz (80L186EB16/80L188EB16)
  - 13 MHz (80L186EB13/80L188EB13)
  - 8 MHz (80L186EB8/80L188EB8)
- Low-Power Operating Modes:
  - Idle Mode Freezes CPU Clocks but keeps Peripherals Active
  - Powerdown Mode Freezes All Internal Clocks
- Supports 80C187 Numeric Coprocessor Interface (80C186EB PLCC Only)
- Available In:
  - 80-Pin Quad Flat Pack (QFP)
  - 84-Pin Plastic Leaded Chip Carrier (PLCC)
  - 80-Pin Shrink Quad Flat Pack (SQFP)

The 80C186EB is a second generation CHMOS High-Integration microprocessor. It has features that are new to the 80C186 family and include a STATIC CPU core, an enhanced Chip Select decode unit, two independent Serial Channels, I/O ports, and the capability of Idle or Powerdown low power modes.



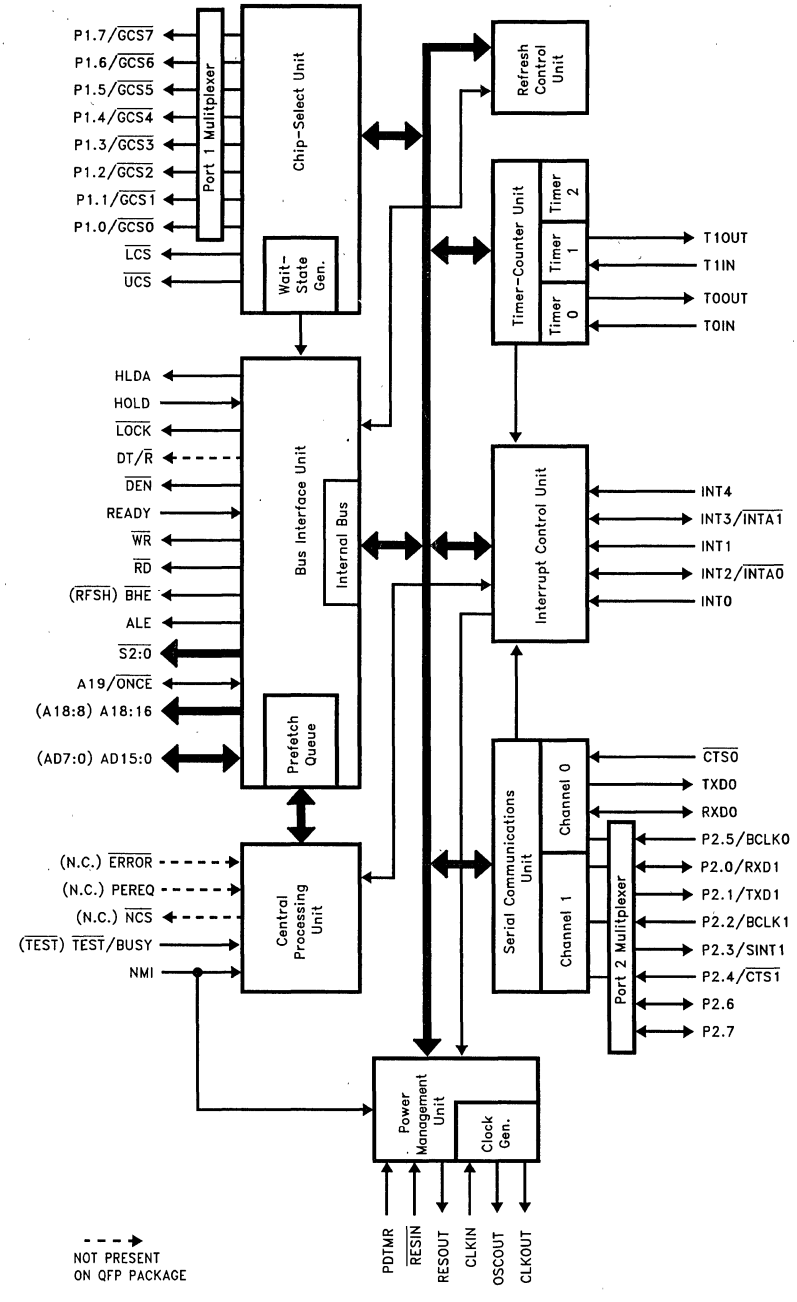
272433-1

October 1995
Order Number: 272433-004

80C186EB/80C188EB and 80L186EB/80L188EB 16-Bit High-Integration Embedded Processors

CONTENTS

CONTENTS



NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB

Figure 1. 80C186EB/80C188EB Block Diagram

INTRODUCTION

Unless specifically noted, all references to the 80C186EB apply to the 80C188EB, 80L186EB, and 80L188EB. References to pins that differ between the 80C186EB/80L186EB and the 80C188EB/80L188EB are given in parentheses. The "L" in the part number denotes low voltage operation. Physically and functionally, the "C" and "L" devices are identical.

The 80C186EB is the first product in a new generation of low-power, high-integration microprocessors. It enhances the existing 186 family by offering new features and new operating modes. The 80C186EB is object code compatible with the 80C186XL/80C188XL microprocessors.

The 80L186EB is the 3V version of the 80C186EB. The 80L186EB is functionally identical to the 80C186EB embedded processor. Current 80C186EB users can easily upgrade their designs to use the 80L186EB and benefit from the reduced power consumption inherent in 3V operation.

The feature set of the 80C186EB meets the needs of low power, space critical applications. Low-Power applications benefit from the static design of the CPU core and the integrated peripherals as well as low voltage operation. Minimum current consumption is achieved by providing a Powerdown mode that halts operation of the device, and freezes the clock circuits. Peripheral design enhancements ensure that non-initialized peripherals consume little current.

Space critical applications benefit from the integration of commonly used system peripherals. Two serial channels are provided for services such as diagnostics, inter-processor communication, modem interface, terminal display interface, and many others. A flexible chip select unit simplifies memory and peripheral interfacing. The interrupt unit provides sources for up to 129 external interrupts and will prioritize these interrupts with those generated from the on-chip peripherals. Three general purpose timer/counters and sixteen multiplexed I/O port pins round out the feature set of the 80C186EB.

Figure 1 shows a block diagram of the 80C186EB/80C188EB. The Execution Unit (EU) is an enhanced 8086 CPU core that includes: dedicated hardware to speed up effective address calculations, enhance execution speed for multiple-bit shift and rotate instructions and for multiply and divide instructions, string move instructions that operate at full bus bandwidth, ten new instruction, and fully static operation. The Bus Interface Unit (BIU) is the same as that found on the original 186 family products, except the queue status mode has been deleted and buffer interface control has been changed to ease system design timings. An independent internal bus is used to allow communication between the BIU and internal peripherals.

CORE ARCHITECTURE

Bus Interface Unit

The 80C186EB core incorporates a bus controller that generates local bus control signals. In addition, it employs a HOLD/HLDA protocol to share the local bus with other bus masters.

The bus controller is responsible for generating 20 bits of address, read and write strobes, bus cycle status information, and data (for write operations) information. It is also responsible for reading data off the local bus during a read operation. A READY input pin is provided to extend a bus cycle beyond the minimum four states (clocks).

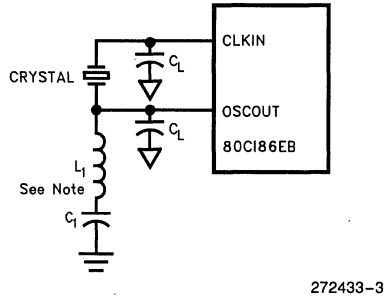
The local bus controller also generates two control signals ($\overline{DEN}$ and DT/$\overline{R}$) when interfacing to external transceiver chips. (Both $\overline{DEN}$ and DT/$\overline{R}$ are available on the PLCC devices, only $\overline{DEN}$ is available on the QFP and SQFP devices.) This capability allows the addition of transceivers for simple buffering of the multiplexed address/data bus.

Clock Generator

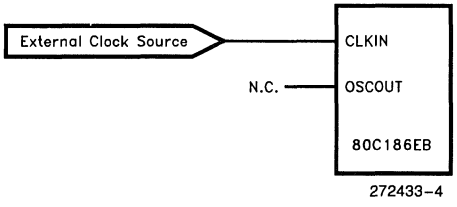
The processor provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter, and two low-power operating modes.

The oscillator circuit is designed to be used with either a **parallel resonant** fundamental or third-overtone mode crystal network. Alternatively, the oscillator circuit may be driven from an external clock source. Figure 2 shows the various operating modes of the oscillator circuit.

The crystal or clock frequency chosen must be twice the required processor operating frequency due to the internal divide-by-two counter. This counter is used to drive all internal phase clocks and the external CLKOUT signal. CLKOUT is a 50% duty cycle processor clock and can be used to drive other system components. All AC timings are referenced to CLKOUT.



(A) Crystal Connection



(B) Clock Connection

NOTE:
The L_1C_1 network is **only** required when using a third-overtone crystal.

Figure 2. Clock Configurations

The following parameters are recommended when choosing a crystal:

Temperature Range:	Application Specific
ESR (Equivalent Series Resistance):	40Ω max
C0 (Shunt Capacitance of Crystal):	7.0 pF max
C_L (Load Capacitance):	20 pF ± 2 pF
Drive Level:	1 mW max

80C186EB PERIPHERAL ARCHITECTURE

The 80C186EB has integrated several common system peripherals with a CPU core to create a compact, yet powerful system. The integrated peripherals are designed to be flexible and provide logical interconnections between supporting units (e.g., the interrupt control unit supports interrupt requests from the timer/counters or serial channels).

The list of integrated peripherals includes:

- 7-Input Interrupt Control Unit
- 3-Channel Timer/Counter Unit
- 2-Channel Serial Communications Unit
- 10-Output Chip-Select Unit
- I/O Port Unit
- Refresh Control Unit
- Power Management Unit

The registers associated with each integrated peripheral are contained within a 128 x 16 register file called the Peripheral Control Block (PCB). The PCB can be located in either memory or I/O space on any 256 Byte address boundary.

Figure 3 provides a list of the registers associated with the PCB. The Register Bit Summary at the end of this specification individually lists all of the registers and identifies each of their programming attributes.

Interrupt Control Unit

The 80C186EB can receive interrupts from a number of sources, both internal and external. The interrupt control unit serves to merge these requests on a priority basis, for individual service by the CPU. Each interrupt source can be independently masked by the Interrupt Control Unit (ICU) or all interrupts can be globally masked by the CPU.

Internal interrupt sources include the Timers and Serial channel 0. External interrupt sources come from the five input pins INT4:0. The NMI interrupt pin is not controlled by the ICU and is passed directly to the CPU. Although the Timer and Serial channel each have only one request input to the ICU, separate vector types are generated to service individual interrupts within the Timer and Serial channel units.

Timer/Counter Unit

The 80C186EB Timer/Counter Unit (TCU) provides three 16-bit programmable timers. Two of these are highly flexible and are connected to external pins for control or clocking. A third timer is not connected to any external pins and can only be clocked internally. However, it can be used to clock the other two timer channels. The TCU can be used to count external events, time external events, generate non-repetitive waveforms, generate timed interrupts. etc.

PCB Offset	Function
00H	Reserved
02H	End Of Interrupt
04H	Poll
06H	Poll Status
08H	Interrupt Mask
0AH	Priority Mask
0CH	In-Service
0EH	Interrupt Request
10H	Interrupt Status
12H	Timer Control
14H	Serial Control
16H	INT4 Control
18H	INT0 Control
1AH	INT1 Control
1CH	INT2 Control
1EH	INT3 Control
20H	Reserved
22H	Reserved
24H	Reserved
26H	Reserved
28H	Reserved
2AH	Reserved
2CH	Reserved
2EH	Reserved
30H	Timer0 Count
32H	Timer0 Compare A
34H	Timer0 Compare B
36H	Timer0 Control
38H	Timer1 Count
3AH	Timer1 Compare A
3CH	Timer1 Compare B
3EH	Timer1 Control

PCB Offset	Function
40H	Timer2 Count
42H	Timer2 Compare
44H	Reserved
46H	Timer2 Control
48H	Reserved
4AH	Reserved
4CH	Reserved
4EH	Reserved
50H	Port 1 Direction
52H	Port 1 Pin
54H	Port 1 Control
56H	Port 1 Latch
58H	Port 2 Direction
5AH	Port 2 Pin
5CH	Port 2 Control
5EH	Port 2 Latch
60H	Serial0 Baud
62H	Serial0 Count
64H	Serial0 Control
66H	Serial0 Status
68H	Serial0 RBUF
6AH	Serial0 TBUF
6CH	Reserved
6EH	Reserved
70H	Serial1 Baud
72H	Serial1 Count
74H	Serial1 Control
76H	Serial1 Status
78H	Serial1 RBUF
7AH	Serial1 TBUF
7CH	Reserved
7EH	Reserved

PCB Offset	Function
80H	GCS0 Start
82H	GCS0 Stop
84H	GCS1 Start
86H	GCS1 Stop
88H	GCS2 Start
8AH	GCS2 Stop
8CH	GCS3 Start
8EH	GCS3 Stop
90H	GCS4 Start
92H	GCS4 Stop
94H	GCS5 Start
96H	GCS5 Stop
98H	GCS6 Start
9AH	GCS6 Stop
9CH	GCS7 Start
9EH	GCS7 Stop
A0H	LCS Start
A2H	LCS Stop
A4H	UCS Start
A6H	UCS Stop
A8H	Relocation
AAH	Reserved
ACH	Reserved
AEH	Reserved
B0H	Refresh Base
B2H	Refresh Time
B4H	Refresh Control
B6H	Reserved
B8H	Power Control
BAH	Reserved
BCH	Step ID
BEH	Reserved

PCB Offset	Function
C0H	Reserved
C2H	Reserved
C4H	Reserved
C6H	Reserved
C8H	Reserved
CAH	Reserved
CCH	Reserved
CEH	Reserved
D0H	Reserved
D2H	Reserved
D4H	Reserved
D6H	Reserved
D8H	Reserved
DAH	Reserved
DCH	Reserved
DEH	Reserved
E0H	Reserved
E2H	Reserved
E4H	Reserved
E6H	Reserved
E8H	Reserved
EAH	Reserved
ECH	Reserved
EEH	Reserved
F0H	Reserved
F2H	Reserved
F4H	Reserved
F6H	Reserved
F8H	Reserved
FAH	Reserved
FCH	Reserved
FEH	Reserved

Figure 3. Peripheral Control Block Registers

Serial Communications Unit

The Serial Control Unit (SCU) of the 80C186EB contains two independent channels. Each channel is identical in operation except that only channel 0 is supported by the integrated interrupt controller (channel 1 has an external interrupt pin). Each channel has its own baud rate generator that is independent of the Timer/Counter Unit, and can be internally or externally clocked at up to one half the 80C186EB operating frequency.

Independent baud rate generators are provided for each of the serial channels. For the asynchronous modes, the generator supplies an 8x baud clock to both the receive and transmit register logic. A 1x baud clock is provided in the synchronous mode.

Chip-Select Unit

The 80C186EB Chip-Select Unit (CSU) integrates logic which provides up to ten programmable chip-selects to access both memories and peripherals. In addition, each chip-select can be programmed to automatically insert additional clocks (wait-states) into the current bus cycle and automatically terminate a bus cycle independent of the condition of the READY input pin.

I/O Port Unit

The I/O Port Unit (IPU) on the 80C186EB supports two 8-bit channels of input, output, or input/output operation. Port 1 is multiplexed with the chip select pins and is output only. Most of Port 2 is multiplexed with the serial channel pins. Port 2 pins are limited to either an output or input function depending on the operation of the serial pin it is multiplexed with.

Refresh Control Unit

The Refresh Control Unit (RCU) automatically generates a periodic memory read bus cycle to keep dynamic or pseudo-static memory refreshed. A 9-bit counter controls the number of clocks between refresh requests.

A 12-bit address generator is maintained by the RCU and is presented on the A12:1 address lines during the refresh bus cycle. Address bits A19:13 are programmable to allow the refresh address block to be located on any 8 Kbyte boundary.

Power Management Unit

The 80C186EB Power Management Unit (PMU) is provided to control the power consumption of the device. The PMU provides three power modes: Active, Idle, and Powerdown.

Active Mode indicates that all units on the 80C186EB are functional and the device consumes maximum power (depending on the level of peripheral operation). Idle Mode freezes the clocks of the Execution and Bus units at a logic zero state (all peripherals continue to operate normally).

The Powerdown mode freezes all internal clocks at a logic zero level and disables the crystal oscillator. All internal registers hold their values provided V_{CC} is maintained. Current consumption is reduced to just transistor junction leakage.

80C187 Interface (80C186EB Only)

The 80C186EB (PLCC package only) supports the direct connection of the 80C187 Numerics Coprocessor.

ONCE Test Mode

To facilitate testing and inspection of devices when fixed into a target system, the 80C186EB has a test mode available which forces all output and input/output pins to be placed in the high-impedance state. ONCE stands for "ON Circuit Emulation". The ONCE mode is selected by forcing the A19/$\overline{\text{ONCE}}$ pin LOW (0) during a processor reset (this pin is weakly held to a HIGH (1) level) while $\overline{\text{RESIN}}$ is active.

PACKAGE INFORMATION

This section describes the pins, pinouts, and thermal characteristics for the 80C186EB in the Plastic Leaded Chip Carrier (PLCC) package, Shrink Quad Flat Pack (SQFP), and Quad Flat Pack (QFP) package. For complete package specifications and information, see the Intel Packaging Outlines and Dimensions Guide (Order Number: 231369).

Prefix Identification

With the extended temperature range, operational characteristics are guaranteed over the temperature range corresponding to −40°C to +85°C ambient. Package types are identified by a two-letter prefix to the part number. The prefixes are listed in Table 1.

Table 1. Prefix Identification

Prefix	Note	Package Type	Temperature Type
TN		PLCC	Extended
TS		QFP (EIAJ)	Extended
SB	1	SQFP	Extended/Commercial
N	1	PLCC	Commercial
S	1	QFP (EIAJ)	Commercial

NOTE:
1. The 5V 25 MHz and 3V 16 MHz versions are only available in commercial temperature range corresponding to 0°C to +70°C ambient.

Pin Descriptions

Each pin or logical set of pins is described in Table 3. There are three columns for each entry in the Pin Description Table.

The **Pin Name** column contains a mnemonic that describes the pin function. Negation of the signal name (for example, $\overline{\text{RESIN}}$) denotes a signal that is active low.

The **Pin Type** column contains two kinds of information. The first symbol indicates whether a pin is power (P), ground (G), input only (I), output only (O) or input/output (I/O). Some pins have multiplexed functions (for example, A19/S6). Additional symbols indicate additional characteristics for each pin. Table 2 lists all the possible symbols for this column.

The **Input Type** column indicates the type of input (Asynchronous or Synchronous).

Asynchronous pins require that setup and hold times be met only in order to guarantee *recognition* at a particular clock edge. Synchronous pins require that setup and hold times be met to guarantee proper *operation.* For example, missing the setup or hold time for the SRDY pin (a synchronous input) will result in a system failure or lockup. Input pins may also be edge- or level-sensitive. The possible characteristics for input pins are S(E), S(L), A(E) and A(L).

The **Output States** column indicates the output state as a function of the device operating mode. Output states are dependent upon the current activity of the processor. There are four operational states that are different from regular operation: bus hold, reset, Idle Mode and Powerdown Mode. Appropriate characteristics for these states are also indicated in this column, with the legend for all possible characteristics in Table 2.

The **Pin Description** column contains a text description of each pin.

As an example, consider AD15:0. I/O signifies the pins are bidirectional. S(L) signifies that the input function is synchronous and level-sensitive. H(Z) signifies that, as outputs, the pins are high-impedance upon acknowledgement of bus hold. R(Z) signifies that the pins float during reset. P(X) signifies that the pins retain their states during Powerdown Mode.

Table 2. Pin Description Nomenclature

Symbol	Description
P	Power Pin (Apply $+V_{CC}$ Voltage)
G	Ground (Connect to V_{SS})
I	Input Only Pin
O	Output Only Pin
I/O	Input/Output Pin
S(E)	Synchronous, Edge Sensitive
S(L)	Synchronous, Level Sensitive
A(E)	Asynchronous, Edge Sensitive
A(L)	Asynchronous, Level Sensitive
H(1)	Output Driven to V_{CC} during Bus Hold
H(0)	Output Driven to V_{SS} during Bus Hold
H(Z)	Output Floats during Bus Hold
H(Q)	Output Remains Active during Bus Hold
H(X)	Output Retains Current State during Bus Hold
R(WH)	Output Weakly Held at V_{CC} during Reset
R(1)	Output Driven to V_{CC} during Reset
R(0)	Output Driven to V_{SS} during Reset
R(Z)	Output Floats during Reset
R(Q)	Output Remains Active during Reset
R(X)	Output Retains Current State during Reset
I(1)	Output Driven to V_{CC} during Idle Mode
I(0)	Output Driven to V_{SS} during Idle Mode
I(Z)	Output Floats during Idle Mode
I(Q)	Output Remains Active during Idle Mode
I(X)	Output Retains Current State during Idle Mode
P(1)	Output Driven to V_{CC} during Powerdown Mode
P(0)	Output Driven to V_{SS} during Powerdown Mode
P(Z)	Output Floats during Powerdown Mode
P(Q)	Output Remains Active during Powerdown Mode
P(X)	Output Retains Current State during Powerdown Mode

Table 3. Pin Descriptions

Pin Name	Pin Type	Input Type	Output States	Description
V_{CC}	P	—	—	**POWER** connections consist of four pins which must be shorted externally to a V_{CC} board plane.
V_{SS}	G	—	—	**GROUND** connections consist of six pins which must be shorted externally to a V_{SS} board plane.
CLKIN	I	A(E)	—	**CLocK INput** is an input for an external clock. An external oscillator operating at two times the required processor operating frequency can be connected to CLKIN. For crystal operation, CLKIN (along with OSCOUT) are the crystal connections to an internal Pierce oscillator.
OSCOUT	O	—	H(Q) R(Q) P(Q)	**OSCillator OUTput** is only used when using a crystal to generate the external clock. OSCOUT (along with CLKIN) are the crystal connections to an internal Pierce oscillator. This pin is not to be used as 2X clock output for non-crystal applications (i.e., this pin is N.C. for non-crystal applications). OSCOUT does not float in ONCE mode.
CLKOUT	O	—	H(Q) R(Q) P(Q)	**CLocK OUTput** provides a timing reference for inputs and outputs of the processor, and is one-half the input clock (CLKIN) frequency. CLKOUT has a 50% duty cycle and transistions every falling edge of CLKIN.
$\overline{\text{RESIN}}$	I	A(L)	—	**RESet IN** causes the processor to immediately terminate any bus cycle in progress and assume an initialized state. All pins will be driven to a known state, and RESOUT will also be driven active. The rising edge (low-to-high) transition synchronizes CLKOUT with CLKIN before the processor begins fetching opcodes at memory location 0FFFF0H.
RESOUT	O	—	H(0) R(1) P(0)	**RESet OUTput** that indicates the processor is currently in the reset state. RESOUT will remain active as long as $\overline{\text{RESIN}}$ remains active.
PDTMR	I/O	A(L)	H(WH) R(Z) P(1)	**Power-Down TiMeR** pin (normally connected to an external capacitor) that determines the amount of time the processor waits after an exit from power down before resuming normal operation. The duration of time required will depend on the startup characteristics of the crystal oscillator.
NMI	I	A(E)	—	**Non-Maskable Interrupt** input causes a TYPE-2 interrupt to be serviced by the CPU. NMI is latched internally.
$\overline{\text{TEST}}$/BUSY ($\overline{\text{TEST}}$)	I	A(E)	—	**TEST** is used during the execution of the WAIT instruction to suspend CPU operation until the pin is sampled active (LOW). $\overline{\text{TEST}}$ is alternately known as BUSY when interfacing with an 80C187 numerics coprocessor (80C186EB only).
AD15:0 (AD7:0)	I/O	S(L)	H(Z) R(Z) P(X)	These pins provide a multiplexed **Address** and **Data** bus. During the address phase of the bus cycle, address bits 0 through 15 (0 through 7 on the 80C188EB) are presented on the bus and can be latched using ALE. 8- or 16-bit data information is transferred during the data phase of the bus cycle.

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Description
A18:16 A19/ONCE (A15:A8) (A18:16) (A19/ONCE)	I/O	A(L)	H(Z) R(WH) P(X)	These pins provide multiplexed **Address** during the address phase of the bus cycle. Address bits 16 through 19 are presented on these pins and can be latched using ALE. These pins are driven to a logic 0 during the data phase of the bus cycle. On the 80C188EB, A15–A8 provide valid address information for the entire bus cycle. During a processor reset (RESIN active), A19/ONCE is used to enable ONCE mode. A18:16 must not be driven low during reset or improper operation may result.
S2:0	O	—	H(Z) R(Z) P(1)	Bus cycle **Status** are encoded on these pins to provide bus transaction information. S2:0 are encoded as follows:

S2	S1	S0	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Processor HALT
1	0	0	Queue Instruction Fetch
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive (no bus activity)

ALE	O	—	H(0) R(0) P(0)	**Address Latch Enable** output is used to strobe address information into a transparent type latch during the address phase of the bus cycle.
BHE (RFSH)	O	—	H(Z) R(Z) P(X)	**Byte High Enable** output to indicate that the bus cycle in progress is transferring data over the upper half of the data bus. BHE and A0 have the following logical encoding

A0	BHE	**Encoding** (for the 80C186EB/80L186EB only)
0	0	Word Transfer
0	1	Even Byte Transfer
1	0	Odd Byte Transfer
1	1	Refresh Operation

On the 80C188EB/80L188EB, RFSH is asserted low to indicate a refresh bus cycle.

RD	O	—	H(Z) R(Z) P(1)	**ReaD** output signals that the accessed memory or I/O device must drive data information onto the data bus.
WR	O	—	H(Z) R(Z) P(1)	**WRite** output signals that data available on the data bus are to be written into the accessed memory or I/O device.
READY	I	A(L) S(L)	—	**READY** input to signal the completion of a bus cycle. READY must be active to terminate any bus cycle, unless it is ignored by correctly programming the Chip-Select Unit.
DEN	O	—	H(Z) R(Z) P(1)	**Data ENable** output to control the enable of bi-directional transceivers in a buffered system. DEN is active only when data is to be transferred on the bus.

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Description
DT/$\overline{R}$	O	—	H(Z) R(Z) P(X)	**Data Transmit/Receive** output controls the direction of a bi-directional buffer in a buffered system. DT/$\overline{R}$ is only available for the PLCC package.
$\overline{LOCK}$	O	—	H(Z) R(WH) P(1)	**LOCK** output indicates that the bus cycle in progress is not to be interrupted. The processor will not service other bus requests (such as HOLD) while $\overline{LOCK}$ is active. This pin is configured as a weakly held high input while $\overline{RESIN}$ is active and must not be driven low.
HOLD	I	A(L)	—	**HOLD** request input to signal that an external bus master wishes to gain control of the local bus. The processor will relinquish control of the local bus between instruction boundaries not conditioned by a LOCK prefix.
HLDA	O	—	H(1) R(0) P(0)	**HoLD Acknowledge** output to indicate that the processor has relinquished control of the local bus. When HLDA is asserted, the processor will (or has) floated its data bus and control signals allowing another bus master to drive the signals directly.
$\overline{NCS}$ (N.C.)	O	—	H(1) R(1) P(1)	**Numerics Coprocessor Select** output is generated when accessing a numerics coprocessor. $\overline{NCS}$ is not provided on the QFP or SQFP packages. This signal does not exist on the 80C188EB/80L188EB.
$\overline{ERROR}$ (N.C.)	I	A(L)	—	**ERROR** input that indicates the last numerics coprocessor operation resulted in an exception condition. An interrupt TYPE 16 is generated if $\overline{ERROR}$ is sampled active at the beginning of a numerics operation. $\overline{ERROR}$ is not provided on the QFP or SQFP packages. This signal does not exist on the 80C188EB/80L188EB.
PEREQ (N.C.)	I	A(L)	—	**CoProcessor REQuest** signals that a data transfer between an External Numerics Coprocessor and Memory is pending. PEREQ is not provided on the QFP or SQFP packages. This signal does not exist on the 80C188EB/80L188EB.
$\overline{UCS}$	O	—	H(1) R(1) P(1)	**Upper Chip Select** will go active whenever the address of a memory or I/O bus cycle is within the address limitations programmed by the user. After reset, $\overline{UCS}$ is configured to be active for memory accesses between 0FFC00H and 0FFFFFH.
$\overline{LCS}$	O	—	H(1) R(1) P(1)	**Lower Chip Select** will go active whenever the address of a memory bus cycle is within the address limitations programmed by the user. $\overline{LCS}$ is inactive after a reset.
P1.0/$\overline{GCS0}$ P1.1/$\overline{GCS1}$ P1.2/$\overline{GCS2}$ P1.3/$\overline{GCS3}$ P1.4/$\overline{GCS4}$ P1.5/$\overline{GCS5}$ P1.6/$\overline{GCS6}$ P1.7/$\overline{GCS7}$	O	—	H(X)/H(1) R(1) P(X)/P(1)	These pins provide a multiplexed function. If enabled, each pin can provide a **Generic Chip Select** output which will go active whenever the address of a memory or I/O bus cycle is within the address limitations programmed by the user. When not programmed as a Chip-Select, each pin may be used as a general purpose output **Port.** As an output port pin, the value of the pin can be read internally.

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Description
T0OUT T1OUT	O	—	H(Q) R(1) P(Q)	**Timer OUTput** pins can be programmed to provide a single clock or continuous waveform generation, depending on the timer mode selected.
T0IN T1IN	I	A(L) A(E)	—	**Timer INput** is used either as clock or control signals, depending on the timer mode selected.
INT0 INT1 INT4	I	A(E,L)	—	Maskable **INTerrupt** input will cause a vector to a specific Type interrupt routine. To allow interrupt expansion, INT0 and/or INT1 can be used with $\overline{\text{INTA0}}$ and $\overline{\text{INTA1}}$ to interface with an external slave controller.
INT2/$\overline{\text{INTA0}}$ INT3/$\overline{\text{INTA1}}$	I/O	A(E,L)	H(1) R(Z) P(1)	These pins provide a multiplexed function. As inputs, they provide a maskable **INTerrupt** that will cause the CPU to vector to a specific Type interrupt routine. As outputs, each is programmatically controlled to provide an INTERRUPT ACKNOWLEDGE handshake signal to allow interrupt expansion.
P2.7 P2.6	I/O	A(L)	H(X) R(Z) P(X)	BI-DIRECTIONAL, open-drain **Port** pins.
$\overline{\text{CTS0}}$ P2.4/$\overline{\text{CTS1}}$	I	A(L)	—	**Clear-To-Send** input is used to prevent the transmission of serial data on their respective TXD signal pin. $\overline{\text{CTS1}}$ is multiplexed with an input only port function.
TXD0 P2.1/TXD1	O	—	H(X)/H(Q) R(1) P(X)/P(Q)	**Transmit Data** output provides serial data information. TXD1 is multiplexed with an output only **Port** function. During synchronous serial communications, TXD will function as a clock output.
RXD0 P2.0/RXD1	I/O	A(L)	R(Z) H(Q) P(X)	**Receive Data** input accepts serial data information. RXD1 is multiplexed with an input only **Port** function. During synchronous serial communications, RXD is bi-directional and will become an output for transmission or data (TXD becomes the clock).
P2.5/BCLK0 P2.2/BCLK1	I	A(L)/A(E)	—	**Baud CLocK** input can be used as an alternate clock source for each of the integrated serial channels. BCLKx is multiplexed with an input only **Port** function, and cannot exceed a clock rate greater than one-half the operating frequency of the processor.
P2.3/SINT1	O	—	H(X)/H(Q) R(0) P(X)/P(X)	**Serial INTerrupt** output will go active to indicate serial channel 1 requires service. SINT1 is multiplexed with an output only **Port** function.

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.

80C186EB PINOUT

Tables 4 and 5 list the 80C186EB/80C188EB pin names with package location for the 84-pin Plastic Leaded Chip Carrier (PLCC) component. Figure 5 depicts the complete 80C186EB/80C188EB pinout (PLCC package) as viewed from the top side of the component (i.e., contacts facing down).

Tables 6 and 7 list the 80C186EB/80C188EB pin names with package location for the 80-pin Quad Flat Pack (QFP) component. Figure 6 depicts the complete 80C186EB/80C188EB (QFP package) as viewed from the top side of the component (i.e., contacts facing down).

Tables 8 and 9 list the 80186EB/80188EB pin names with package location for the 80-pin Shrink Quad Flat Pack (SQFP) component. Figure 7 depicts the complete 80C186EB/80C188EB (SQFP package) as viewed from the top side of the component (i.e., contacts facing down).

Table 4. PLCC Pin Names with Package Location

Address/Data Bus	
Name	**Location**
AD0	61
AD1	66
AD2	68
AD3	70
AD4	72
AD5	74
AD6	76
AD7	78
AD8 (A8)	62
AD9 (A9)	67
AD10 (A10)	69
AD11 (A11)	71
AD12 (A12)	73
AD13 (A13)	75
AD14 (A14)	77
AD15 (A15)	79
A16	80
A17	81
A18	82
A19/$\overline{\text{ONCE}}$	83

Bus Control	
Name	**Location**
ALE	6
$\overline{\text{BHE}}$ ($\overline{\text{RFSH}}$)	7
$\overline{\text{S0}}$	10
$\overline{\text{S1}}$	9
$\overline{\text{S2}}$	8
$\overline{\text{RD}}$	4
$\overline{\text{WR}}$	5
READY	18
$\overline{\text{DEN}}$	11
DT/$\overline{\text{R}}$	16
$\overline{\text{LOCK}}$	15
HOLD	13
HLDA	12

Power	
Name	**Location**
V_{SS}	2, 22, 43 63, 65, 84
V_{CC}	1, 23 42, 64

Processor Control	
Name	**Location**
$\overline{\text{RESIN}}$	37
RESOUT	38
CLKIN	41
OSCOUT	40
CLKOUT	44
$\overline{\text{TEST}}$/BUSY	14
$\overline{\text{NCS}}$ (N.C.)	60
PEREQ (N.C.)	39
$\overline{\text{ERROR}}$ (N.C.)	3
PDTMR	36
NMI	17
INT0	31
INT1	32
INT2/$\overline{\text{INTA0}}$	33
INT3/$\overline{\text{INTA1}}$	34
INT4	35

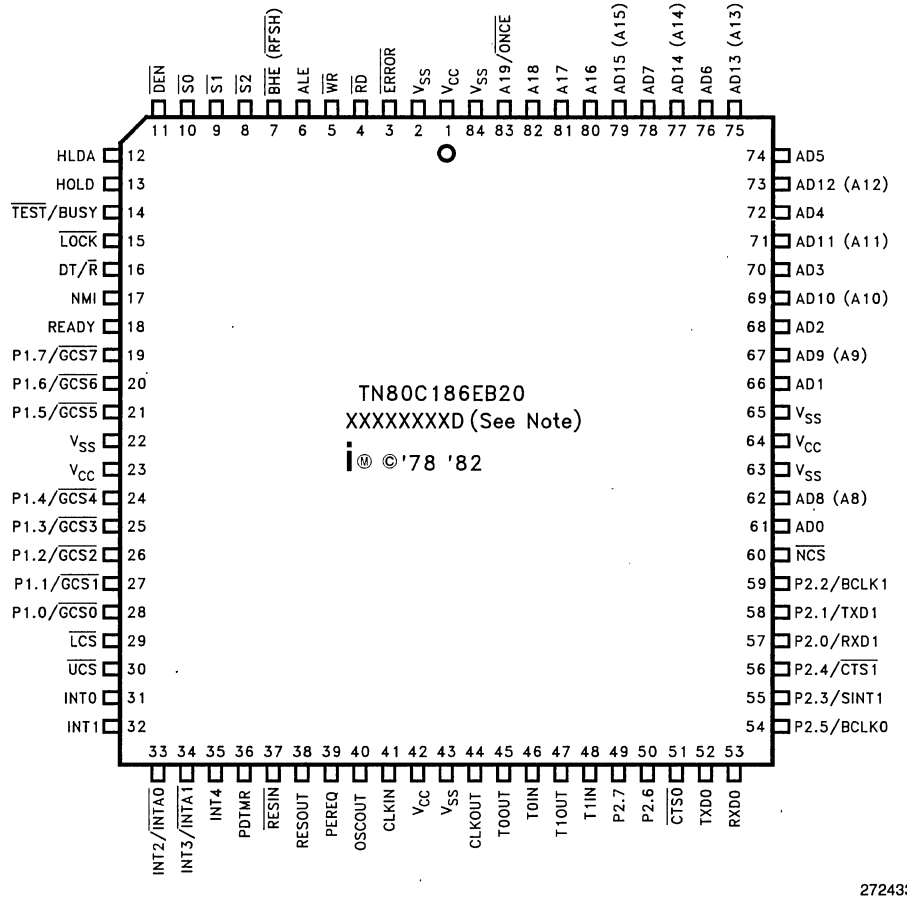
I/O	
Name	**Location**
$\overline{\text{UCS}}$	30
$\overline{\text{LCS}}$	29
P1.0/$\overline{\text{GCS0}}$	28
P1.1/$\overline{\text{GCS1}}$	27
P1.2/$\overline{\text{GCS2}}$	26
P1.3/$\overline{\text{GCS3}}$	25
P1.4/$\overline{\text{GCS4}}$	24
P1.5/$\overline{\text{GCS5}}$	21
P1.6/$\overline{\text{GCS6}}$	20
P1.7/$\overline{\text{GCS7}}$	19
T0OUT	45
T0IN	46
T1OUT	47
T1IN	48
RXD0	53
TXD0	52
P2.5/BCLK0	54
$\overline{\text{CTS0}}$	51
P2.0/RXD1	57
P2.1/TXD1	58
P2.2/BCLK1	59
P2.3/SINT1	55
P2.4/$\overline{\text{CTS1}}$	56
P2.6	50
P2.7	49

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.

Table 5. PLCC Package Locations with Pin Name

Location	Name	Location	Name	Location	Name	Location	Name
1	V_{CC}	22	V_{SS}	43	V_{SS}	64	V_{CC}
2	V_{SS}	23	V_{CC}	44	CLKOUT	65	V_{SS}
3	$\overline{\text{ERROR}}$ (N.C.)	24	P1.4/$\overline{\text{GCS4}}$	45	T0OUT	66	AD1
4	$\overline{\text{RD}}$	25	P1.3/$\overline{\text{GCS3}}$	46	T0IN	67	AD9 (A9)
5	$\overline{\text{WR}}$	26	P1.2/$\overline{\text{GCS2}}$	47	T1OUT	68	AD2
6	ALE	27	P1.1/$\overline{\text{GCS1}}$	48	T1IN	69	AD10 (A10)
7	$\overline{\text{BHE}}$ ($\overline{\text{RFSH}}$)	28	P1.0/$\overline{\text{GCS0}}$	49	P2.7	70	AD3
8	$\overline{\text{S2}}$	29	$\overline{\text{LCS}}$	50	P2.6	71	AD11 (A11)
9	$\overline{\text{S1}}$	30	$\overline{\text{UCS}}$	51	$\overline{\text{CTS0}}$	72	AD4
10	$\overline{\text{S0}}$	31	INT0	52	TXD0	73	AD12 (A12)
11	$\overline{\text{DEN}}$	32	INT1	53	RXD0	74	AD5
12	HLDA	33	INT2/$\overline{\text{INTA0}}$	54	P2.5/BCLK0	75	AD13 (A13)
13	HOLD	34	INT3/$\overline{\text{INTA1}}$	55	P2.3/SINT1	76	AD6
14	$\overline{\text{TEST}}$/BUSY	35	INT4	56	P2.4/$\overline{\text{CTS1}}$	77	AD14 (A14)
15	$\overline{\text{LOCK}}$	36	PDTMR	57	P2.0/RXD1	78	AD7
16	DT/$\overline{\text{R}}$	37	$\overline{\text{RESIN}}$	58	P2.1/TXD1	79	AD15 (A15)
17	NMI	38	RESOUT	59	P2.2/BCLK1	80	A16
18	READY	39	PEREQ (N.C.)	60	$\overline{\text{NCS}}$ (N.C.)	81	A17
19	P1.7/$\overline{\text{GCS7}}$	40	OSCOUT	61	AD0	82	A18
20	P1.6/$\overline{\text{GCS6}}$	41	CLKIN	62	AD8 (A8)	83	A19/$\overline{\text{ONCE}}$
21	P1.5/$\overline{\text{GCS5}}$	42	V_{CC}	63	V_{SS}	84	V_{SS}

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.



NOTE:
This is the FPO number location (indicated by X's).
Pin names in parentheses apply to the 80C188EB/80L188EB.

Figure 4. 84-Pin Plastic Leaded Chip Carrier Pinout Diagram

Table 6. QFP Pin Name with Package Location

Address/Data Bus	
Name	**Location**
AD0	10
AD1	15
AD2	17
AD3	19
AD4	21
AD5	23
AD6	25
AD7	27
AD8 (A8)	11
AD9 (A9)	16
AD10 (A10)	18
AD11 (A11)	20
AD12 (A12)	22
AD13 (A13)	24
AD14 (A14)	26
AD15 (A15)	28
A16	29
A17	30
A18	31
A19/$\overline{\text{ONCE}}$	32

Bus Control	
Name	**Location**
ALE	38
$\overline{\text{BHE}}$ ($\overline{\text{RFSH}}$)	39
$\overline{\text{S0}}$	42
$\overline{\text{S1}}$	41
$\overline{\text{S2}}$	40
$\overline{\text{RD}}$	36
$\overline{\text{WR}}$	37
READY	49
$\overline{\text{DEN}}$	43
$\overline{\text{LOCK}}$	47
HOLD	45
HLDA	44

Power	
Name	**Location**
V_{SS}	12, 14, 33
	35, 53, 73
V_{CC}	13, 34
	54, 72

Processor Control	
Name	**Location**
$\overline{\text{RESIN}}$	68
RESOUT	69
CLKIN	71
OSCOUT	70
CLKOUT	74
$\overline{\text{TEST}}$	46
PDTMR	67
NMI	48
INT0	62
INT1	63
INT2/$\overline{\text{INTA0}}$	64
INT3/$\overline{\text{INTA1}}$	65
INT4	66

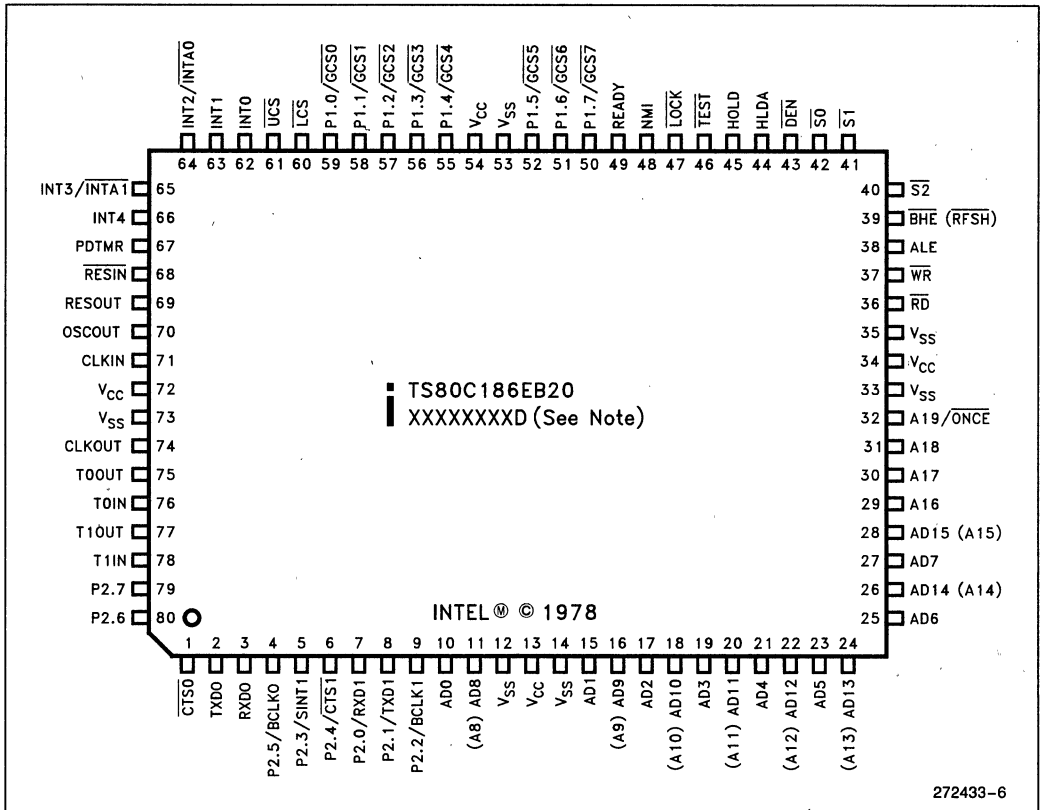
I/O	
Name	**Location**
$\overline{\text{UCS}}$	61
$\overline{\text{LCS}}$	60
P1.0/$\overline{\text{GCS0}}$	59
P1.1/$\overline{\text{GCS1}}$	58
P1.2/$\overline{\text{GCS2}}$	57
P1.3/$\overline{\text{GCS3}}$	56
P1.4/$\overline{\text{GCS4}}$	55
P1.5/$\overline{\text{GCS5}}$	52
P1.6/$\overline{\text{GCS6}}$	51
P1.7/$\overline{\text{GCS7}}$	50
T0OUT	75
T0IN	76
T1OUT	77
T1IN	78
RXD0	3
TXD0	2
P2.5/BCLK0	4
$\overline{\text{CTS0}}$	1
P2.0/RXD1	7
P2.1/TXD1	8
P2.2/BCLK1	9
P2.3/SINT1	5
P2.4/$\overline{\text{CTS1}}$	6
P2.6	80
P2.7	79

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.

Table 7. QFP Package Location with Pin Names

Location	Name	Location	Name	Location	Name	Location	Name
1	$\overline{CTS0}$	21	AD4	41	$\overline{S1}$	61	$\overline{UCS}$
2	TXD0	22	AD12 (A12)	42	$\overline{S0}$	62	INT0
3	RXD0	23	AD5	43	$\overline{DEN}$	63	INT1
4	P2.5/BCLK0	24	AD13 (A13)	44	HLDA	64	INT2/$\overline{INTA0}$
5	P2.3/SINT1	25	AD6	45	HOLD	65	INT3/$\overline{INTA1}$
6	P2.4/$\overline{CTS1}$	26	AD14 (A14)	46	$\overline{TEST}$	66	INT4
7	P2.0/RXD1	27	AD7	47	$\overline{LOCK}$	67	PDTMR
8	P2.1/TXD1	28	AD15 (A15)	48	NMI	68	$\overline{RESIN}$
9	P2.2/BCLK1	29	A16	49	READY	69	RESOUT
10	AD0	30	A17	50	P1.7/$\overline{GCS7}$	70	OSCOUT
11	AD8 (A8)	31	A18	51	P1.6/$\overline{GCS6}$	71	CLKIN
12	V_{SS}	32	A19/$\overline{ONCE}$	52	P1.5/$\overline{GCS5}$	72	V_{CC}
13	V_{CC}	33	V_{SS}	53	V_{SS}	73	V_{SS}
14	V_{SS}	34	V_{CC}	54	V_{CC}	74	CLKOUT
15	AD1	35	V_{SS}	55	P1.4/$\overline{GCS4}$	75	T0OUT
16	AD9 (A9)	36	$\overline{RD}$	56	P1.3/$\overline{GCS3}$	76	T0IN
17	AD2	37	$\overline{WR}$	57	P1.2/$\overline{GCS2}$	77	T1OUT
18	AD10 (A10)	38	ALE	58	P1.1/$\overline{GCS1}$	78	T1IN
19	AD3	39	$\overline{BHE}$ ($\overline{RFSH}$)	59	P1.0/$\overline{GCS0}$	79	P2.7
20	AD11 (A11)	40	$\overline{S2}$	60	$\overline{LCS}$	80	P2.6

NOTE:
Pin names in parentheses apply to the 80C188EB/80L188EB.



NOTE:
This is the FPO number location (indicated by X's).
Pin names in parentheses apply to the 80C188EB/80L188EB.

Figure 5. Quad Flat Pack Pinout Diagram

Table 8. SQFP Pin Functions with Location

AD Bus	
AD0	47
AD1	52
AD2	54
AD3	56
AD4	58
AD5	60
AD6	62
AD7	64
AD8 (A8)	48
AD9 (A9)	53
AD10 (A10)	55
AD11 (A11)	57
AD12 (A12)	59
AD13 (A13)	61
AD14 (A14)	63
AD15 (A15)	65
A16	66
A17	67
A18	68
A19/ONCE	69

Bus Control	
ALE	75
BHE# (RFSH#)	76
S0#	79
S1#	78
S2#	77
RD#	73
WR#	74
READY	6
DEN#	80
LOCK#	4
HOLD	2
HLDA	1

Processor Control	
RESIN#	25
RESOUT	26
CLKIN	28
OSCOUT	27
CLKOUT	31
TEST#/BUSY	3
NMI	5
INT0	19
INT1	20
INT2/INTA0#	21
INT3/INTA1#	22
INT4	23
PDTMR	24

Power and Ground	
V_{CC}	11
V_{CC}	29
V_{CC}	50
V_{CC}	71
V_{SS}	10
V_{SS}	30
V_{SS}	49
V_{SS}	51
V_{SS}	70
V_{SS}	72

I/O	
UCS#	18
LCS#	17
P1.0/GCS0#	16
P1.1/GCS1#	15
P1.2/GCS2#	14
P1.3/GCS3#	13
P1.4/GCS4#	12
P1.5/GCS5#	9
P1.6/GCS6#	8
P1.7/GCS7#	7
P2.0/RXD1	44
P2.1/TXD1	45
P2.2/BCLK1	46
P2.3/SINT1	42
P2.4/CTS1#	43
P2.5/BCLK0	41
P2.6	37
P2.7	36
CTS0#	38
TXD0	39
RXD0	40
T0IN	33
T1IN	35
T0OUT	32
T1OUT	34

Table 9. SQFP Pin Locations with Pin Names

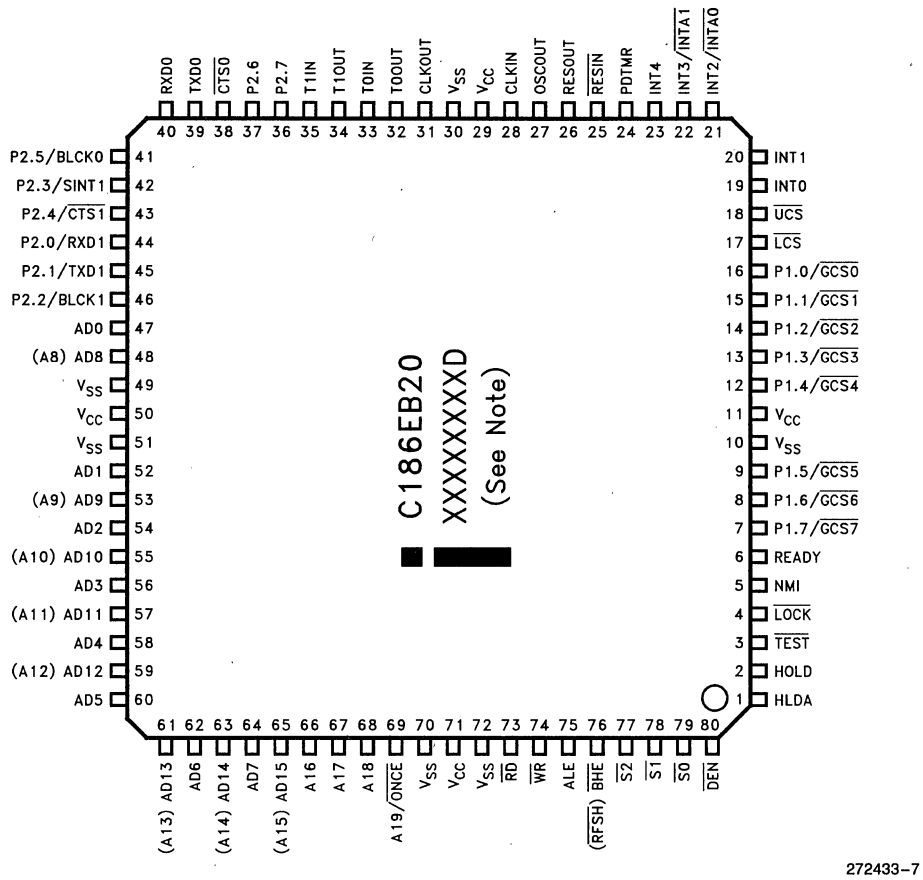
1	HLDA
2	HOLD
3	TEST#
4	LOCK#
5	NMI
6	READY
7	P1.7/GCS7#
8	P1.6/GCS6#
9	P1.5/GCS5#
10	V_{SS}
11	V_{CC}
12	P1.4/GCS4#
13	P1.3/GCS3#
14	P1.2/GCS2#
15	P1.1/GCS1#
16	P1.0/GCS0#
17	LCS#
18	UCS#
19	INT0
20	INT1

21	INT1/INTA0#
22	INT3/INTA1#
23	INT4
24	PDTMR
25	RESIN#
26	RESOUT
27	OSCOUT
28	CLKIN
29	V_{CC}
30	V_{SS}
31	CLKOUT
32	T0OUT
33	T0IN
34	T1OUT
35	T1IN
36	P2.7
37	P2.6
38	CTS0#
39	TXD0
40	RXD0

41	P2.5/BCLK0
42	P2.3/SINT1
43	P2.4/CTS1#
44	P2.0/RXD1
45	P2.1/TXD1
46	P2.2/BCLK1
47	AD0
48	AD8 (A8)
49	V_{SS}
50	V_{CC}
51	V_{SS}
52	AD1
53	AD9 (A9)
54	AD2
55	AD10 (A10)
56	AD3
57	AD11 (A11)
58	AD4
59	AD12 (A12)
60	AD5

61	AD13 (A13)
62	AD6
63	AD14 (A14)
64	AD7
65	AD15 (A15)
66	A16
67	A17
68	A18
69	A19/ONCE
70	V_{SS}
71	V_{CC}
72	V_{SS}
73	RD#
74	WR#
75	ALE
76	BHE# (RFSH#)
77	S2#
78	S1#
79	S0#
80	DEN#

NOTE:
Pin names in parentheses appiy to the 80C188EB/80L188EB.



272433-7

NOTE:
XXXXXXXC indicates Intel FPO number.
Pin names in parentheses apply to the 80C188EB/80L188EB.

Figure 6. SQFP Package

PACKAGE THERMAL SPECIFICATIONS

The 80C186EB/80L186EB is specified for operation when T_C (the case temperature) is within the range of −40°C to +100°C (PLCC package) or −40°C to +114°C (QFP package). T_C may be measured in any environment to determine whether the processor is within the specified operating range. The case temperature must be measured at the center of the top surface.

T_A (the ambient temperature) can be calculated from θ_{CA} (thermal resistance from the case to ambient) with the following equation:

$$T_A = T_C - P*\theta_{CA}$$

Typical values for θ_{CA} at various airflows are given in Table 10. P (the maximum power consumption, specified in watts) is calculated by using the maximum ICC as tabulated in the DC specifications and V_{CC} of 5.5V.

Table 10. Thermal Resistance (θ_{CA}) at Various Airflows (in °C/Watt)

	Airflow Linear ft/min (m/sec)					
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
θ_{CA} (PLCC)	30	24	21	19	17	16.5
θ_{CA} (QFP)	58	47	43	40	38	36
θ_{CA} (SQFP)	70	TBD	TBD	TBD	TBD	TBD

ELECTRICAL SPECIFICATIONS

NOTICE: This data sheet contains preliminary information on new products in production. It is valid for the devices indicated in the revision history. The specifications are subject to change without notice.

Absolute Maximum Ratings

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

Case Temp under Bias ..... −65°C to +120°C

Supply Voltage
with Respect to V_{SS} ..... −0.5V to + 6.5V

Voltage on other Pins
with Respect to V_{SS} ..... −0.5V to V_{CC} + 0.5V

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

Recommended Connections

Power and ground connections must be made to multiple V_{CC} and V_{SS} pins. Every 80C186EB-based circuit board should include separate power (V_{CC}) and ground (V_{SS}) planes. Every V_{CC} pin must be connected to the power plane, and every V_{SS} pin must be connected to the ground plane. Pins identified as "NC" must not be connected in the system. Liberal decoupling capacitance should be placed near the processor. The processor can cause transient power surges when its output buffers transition, particularly when connected to large capacitive loads.

Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance is reduced by placing the decoupling capacitors as close as possible to the processor V_{CC} and V_{SS} package pins.

Always connect any unused input to an appropriate signal level. In particular, unused interrupt inputs (INT0:4) should be connected to V_{CC} through a pull-up resistor (in the range of 50 KΩ). **Leave any unused output pin or any NC pin unconnected.**

DC SPECIFICATIONS (80C186EB/80C188EB)

Symbol	Parameter	Min	Max	Units	Notes
V_{CC}	Supply Voltage	4.5	5.5	V	
V_{IL}	Input Low Voltage	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 3 mA (Min)
V_{OH}	Output High Voltage	V_{CC} − 0.5		V	I_{OH} = −2 mA (Mln)
V_{HYR}	Input Hysterisis on $\overline{RESIN}$	0.50		V	
I_{LI1}	Input Leakage Current for Pins: AD15:0 (AD7:0), READY, HOLD, $\overline{RESIN}$, CLKIN, $\overline{TEST}$, NMI, INT4:0, T0IN, T1IN, RXD0, $\overline{BCLK0}$, $\overline{CTS0}$, RXD1, $\overline{BCLK1}$, $\overline{CTS1}$, P2.6, P2.7		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LI2}	Input Leakage Current for Pins: $\overline{ERROR}$, PEREQ	±0.275	±7	mA	0V ≤ V_{IN} < V_{CC}
I_{LI3}	Input Leakage Current for Pins: A19/$\overline{ONCE}$, A18:16, $\overline{LOCK}$	−0.275	−5.0	mA	V_{IN} = 0.7 V_{CC} (Note 1)
I_{LO}	Output Leakage Current		±15	μA	0.45 ≤ V_{OUT} ≤ V_{CC} (Note 2)
I_{CC}	Supply Current Cold (RESET)				
	80C186EB25		115	mA	(Notes 3, 7)
	80C186EB20		108	mA	(Note 3)
	80C186EB13		73	mA	(Note 3)
I_{ID}	Supply Current Idle				
	80C186EB25		91	mA	(Notes 4, 7)
	80C186EB20		76	mA	(Note 4)
	80C186EB13		48	mA	(Note 4)
I_{PD}	Supply Current Powerdown				
	80C186EB25		100	μA	(Notes 5, 7)
	80C186EB20		100	μA	(Note 5)
	80C186EB13		100	μA	(Note 5)
C_{IN}	Input Pin Capacitance	0	15	pF	T_F = 1 MHz
C_{OUT}	Output Pin Capacitance	0	15	pF	T_F = 1 MHz (Note 6)

NOTES:
1. These pins have an internal pull-up device that is active while $\overline{RESIN}$ is low and ONCE Mode is not active. Sourcing more current than specified (on any of these pins) may invoke a factory test mode.
2. Tested by outputs being floated by invoking ONCE Mode or by asserting HOLD.
3. Measured with the device in RESET and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
4. Measured with the device in HALT (IDLE Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
5. Measured with the device in HALT (Powerdown Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
6. Output Capacitance is the capacitive load of a floating output pin.
7. Operating temperature for 25 MHz is 0°C to 70°C, V_{CC} = 5.0 ±10%.

DC SPECIFICATIONS (80L186EB16) (operating temperature, 0°C to 70°C)

Symbol	Parameter	Min	Max	Units	Notes
V_{CC}	Supply Voltage	3.0	5.5	V	
V_{IL}	Input Low Voltage	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 1.6 mA (Min) (Note 1)
V_{OH}	Output High Voltage	V_{CC} − 0.5		V	I_{OH} = −1 mA (Min) (Note 1)
V_{HYR}	Input Hysterisis on $\overline{\text{RESIN}}$	0.50		V	
I_{LI1}	Input Leakage Current for pins: AD15:0 (AD7:0), READY, HOLD, $\overline{\text{RESIN}}$, CLKIN, $\overline{\text{TEST}}$, NMI, INT4:0, T0IN, T1IN, RXD0, $\overline{\text{BCLK0}}$, $\overline{\text{CTS0}}$, RXD1, $\overline{\text{BCLK1}}$, $\overline{\text{CTS1}}$, SINT1, P2.6, P2.7		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LI2}	Input Leakage Current for Pins: A19/$\overline{\text{ONCE}}$, A18:16, $\overline{\text{LOCK}}$	−0.275	−2	mA	V_{IN} = 0.7 V_{CC} (Note 2)
I_{LO}	Output Leakage Current		±15	μA	0.45 ≤ V_{OUT} ≤ V_{CC} (Note 3)
I_{CC3}	Supply Current (RESET, 3.3V) 80L186EB16		54	mA	(Note 4)
I_{ID3}	Supply Current Idle (3.3V) 80L186EB16		38	mA	(Note 5)
I_{PD3}	Supply Current Powerdown (3.3V) 80L186EB16		40	μA	(Note 6)
C_{IN}	Input Pin Capacitance	0	15	pF	T_F = 1 MHz
C_{OUT}	Output Pin Capacitance	0	15	pF	T_F = 1 MHz (Note 7)

NOTES:
1. I_{OL} and I_{OH} measured at V_{CC} = 3.0V.
2. These pins have an internal pull-up device that is active while $\overline{\text{RESIN}}$ is low and ONCE Mode is not active. Sourcing more current than specified (on any of these pins) may invoke a factory test mode.
3. Tested by outputs being floated by invoking ONCE Mode or by asserting HOLD.
4. Measured with the device in RESET and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
5. Measured with the device in HALT (IDLE Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
6. Measured with the device in HALT (Powerdown Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
7. Output Capacitance is the capacitive load of a floating output pin.

DC SPECIFICATIONS (80L186EB13/80L188EB13, 80L186EB8/80L188EB8)

Symbol	Parameter	Min	Max	Units	Notes
V_{CC}	Supply Voltage	2.7	5.5	V	
V_{IL}	Input Low Voltage	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 1.6 mA (Min) (Note 1)
V_{OH}	Output High Voltage	V_{CC} − 0.5		V	I_{OH} = −1 mA (Min) (Note 1)
V_{HYR}	Input Hysterisis on $\overline{RESIN}$	0.50		V	
I_{LI1}	Input Leakage Current for pins: AD15:0 (AD7:0), READY, HOLD, $\overline{RESIN}$, CLKIN, $\overline{TEST}$, NMI, INT4:0, T0IN, T1IN, RXD0, $\overline{BCLK0}$, $\overline{CTS0}$, RXD1, $\overline{BCLK1}$, $\overline{CTS1}$, SINT1, P2.6, P2.7		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LI2}	Input Leakage Current for Pins: A19/$\overline{ONCE}$, A18:16, $\overline{LOCK}$	−0.275	−2	mA	V_{IN} = 0.7 V_{CC} (Note 2)
I_{LO}	Output Leakage Current		±15	μA	0.45 ≤ V_{OUT} ≤ V_{CC} (Note 3)
I_{CC5}	Supply Current (RESET, 5.5V) 80L186EB13 80L186EB8		 73 45	 mA mA	 (Note 4) (Note 4)
I_{CC3}	Supply Current (RESET, 2.7V) 80L186EB13 80L186EB8		 36 22	 mA mA	 (Note 4) (Note 4)
I_{ID5}	Supply Current Idle (5.5V) 80L186EB13 80L186EB8		 48 31	 mA mA	 (Note 5) (Note 5)
I_{ID3}	Supply Current Idle (2.7V) 80L186EB13 80L186EB8		 24 15	 mA mA	 (Note 5) (Note 5)
I_{PD5}	Supply Current Powerdown (5.5V) 80L186EB13 80L186EB8		 100 100	 μA μA	 (Note 6) (Note 6)
I_{PD3}	Supply Current Powerdown (2.7V) 80L186EB13 80L186EB8		 30 30	 μA μA	 (Note 6) (Note 6)
C_{IN}	Input Pin Capacitance	0	15	pF	T_F = 1 MHz
C_{OUT}	Output Pin Capacitance	0	15	pF	T_F = 1 MHz (Note 7)

NOTES:
1. I_{OL} and I_{OH} measured at V_{CC} = 2.7V.
2. These pins have an internal pull-up device that is active while $\overline{RESIN}$ is low and ONCE Mode is not active. Sourcing more current than specified (on any of these pins) may invoke a factory test mode.
3. Tested by outputs being floated by invoking ONCE Mode or by asserting HOLD.
4. Measured with the device in RESET and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
5. Measured with the device in HALT (IDLE Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
6. Measured with the device in HALT (Powerdown Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
7. Output Capacitance is the capacitive load of a floating output pin.

I_{CC} VERSUS FREQUENCY AND VOLTAGE

The current (I_{CC}) consumption of the processor is essentially composed of two components; I_{PD} and I_{CCS}.

I_{PD} is the **quiescent** current that represents internal device leakage, and is measured with all inputs or floating outputs at GND or V_{CC} (no clock applied to the device). I_{PD} is equal to the Powerdown current and is typically less than 50 μA.

I_{CCS} is the **switching** current used to charge and discharge parasitic device capacitance when changing logic levels. Since I_{CCS} is typically much greater than I_{PD}, I_{PD} can often be ignored when calculating I_{CC}.

I_{CCS} is related to the voltage and frequency at which the device is operating. It is given by the formula:

$$\text{Power} = V \times I = V^2 \times C_{DEV} \times f$$

$$\therefore I = I_{CC} = I_{CCS} = V \times C_{DEV} \times f$$

Where: V = Device operating voltage (V_{CC})

C_{DEV} = Device capacitance

f = Device operating frequency

$I_{CCS} = I_{CC}$ = Device current

Measuring C_{DEV} on a device like the 80C186EB would be difficult. Instead, C_{DEV} is calculated using the above formula by measuring I_{CC} at a known V_{CC} and frequency (see Table 11). Using this C_{DEV} value, I_{CC} can be calculated at any voltage and frequency within the specified operating range.

EXAMPLE: Calculate the typical I_{CC} when operating at 10 MHz, 4.8V.

$$I_{CC} = I_{CCS} = 4.8 \times 0.583 \times 10 \approx 28 \text{ mA}$$

PDTMR PIN DELAY CALCULATION

The PDTMR pin provides a delay between the assertion of NMI and the enabling of the internal clocks when exiting Powerdown. A delay is required only when using the on-chip oscillator to allow the crystal or resonator circuit time to stabilize.

NOTE:
The PDTMR pin function does not apply when $\overline{\text{RESIN}}$ is asserted (i.e., a device reset during Powerdown is similar to a cold reset and $\overline{\text{RESIN}}$ must remain active until after the oscillator has stabilized).

To calculate the value of capacitor required to provide a desired delay, use the equation:

$$440 \times t = C_{PD} \quad (5V, 25°C)$$

Where: t = desired delay in **seconds**

C_{PD} = capacitive load on PDTMR in **microfarads**

EXAMPLE: To get a delay of 300 μs, a capacitor value of $C_{PD} = 440 \times (300 \times 10^{-6}) = 0.132$ μF is required. Round up to standard (available) capacitive values.

NOTE:
The above equation applies to delay times greater than 10 μs and will compute the **TYPICAL** capacitance needed to achieve the desired delay. A delay variance of +50% or −25% can occur due to temperature, voltage, and device process extremes. In general, higher V_{CC} and/or lower temperature will decrease delay time, while lower V_{CC} and/or higher temperature will increase delay time.

Table 11. Device Capacitance (C_{DEV}) Values

Parameter	Typ	Max	Units	Notes
C_{DEV} (Device in Reset)	0.583	1.02	mA/V*MHz	1, 2
C_{DEV} (Device in Idle)	0.408	0.682	mA/V*MHz	1, 2

1. Max C_{DEV} is calculated at −40°C, all floating outputs driven to V_{CC} or GND, and all outputs loaded to 50 pF (including CLKOUT and OSCOUT).
2. Typical C_{DEV} is calculated at 25°C with all outputs loaded to 50 pF except CLKOUT and OSCOUT, which are not loaded.

AC SPECIFICATIONS

AC Characteristics—80C186EB25

Symbol	Parameter	25 MHz		Units	Notes
		Min	Max		
INPUT CLOCK					
T_F	CLKIN Frequency	0	50	MHz	1
T_C	CLKIN Period	20	∞	ns	1
T_{CH}	CLKIN High Time	8	∞	ns	1, 2
T_{CL}	CLKIN Low Time	8	∞	ns	1, 2
T_{CR}	CLKIN Rise Time	1	7	ns	1, 3
T_{CF}	CLKIN Fall Time	1	7	ns	1, 3
OUTPUT CLOCK					
T_{CD}	CLKIN to CLKOUT Delay	0	16	ns	1, 4
T	CLKOUT Period		$2*T_C$	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PR}	CLKOUT Rise Time	1	6	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	6	ns	1, 5
OUTPUT DELAYS					
T_{CHOV1}	ALE, $\overline{S2:0}$, $\overline{DEN}$, DT/$\overline{R}$, $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$, A19:16	3	17	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{GCS0:7}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{NCS}$, $\overline{RD}$, $\overline{WR}$	3	20	ns	1, 4, 6, 8
T_{CLOV1}	$\overline{BHE}$ ($\overline{RFSH}$), $\overline{DEN}$, $\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT, A19:16	3	17	ns	1, 4, 6
T_{CLOV2}	$\overline{RD}$, $\overline{WR}$, $\overline{GCS7:0}$, $\overline{LCS}$, $\overline{UCS}$, AD15:0 (AD7:0, A15:8), $\overline{NCS}$, $\overline{INTA1:0}$, $\overline{S2:0}$	3	20	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	20	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (AD7:0, A15:8)	0	20	ns	1

AC SPECIFICATIONS

AC Characteristics—80C186EB25 (Continued)

Symbol	Parameter	25 MHz		Units	Notes
		Min	Max		
SYNCHRONOUS INPUTS					
T_{CHIS}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY, $\overline{CTS1:0}$, P2.6, P2.7	10		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY, $\overline{CTS1:0}$	3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), READY	10		ns	1, 10
T_{CLIH}	READY, AD15:0 (AD7:0)	3		ns	1, 10
T_{CLIS}	HOLD, PEREQ, $\overline{ERROR}$	10		ns	1, 9
T_{CLIH}	HOLD, PEREQ, $\overline{ERROR}$	3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measure at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.

AC SPECIFICATIONS

AC Characteristics—80C186EB20/80C186EB13

Symbol	Parameter	20 MHz		13 MHz		Units	Notes
		Min	Max	Min	Max		
INPUT CLOCK							
T_F	CLKIN Frequency	0	40	0	26	MHz	1
T_C	CLKIN Period	25	∞	38.5	∞	ns	1
T_{CH}	CLKIN High Time	10	∞	12	∞	ns	1, 2
T_{CL}	CLKIN Low Time	10	∞	12	∞	ns	1, 2
T_{CR}	CLKIN Rise Time	1	8	1	8	ns	1, 3
T_{CF}	CLKIN Fall Time	1	8	1	8	ns	1, 3
OUTPUT CLOCK							
T_{CD}	CLKIN to CLKOUT Delay	0	17	0	23	ns	1, 4
T	CLKOUT Period		$2*T_C$		$2*T_C$	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5	(T/2) + 5	(T/2) − 5	(T/2) + 5	ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5	(T/2) + 5	(T/2) − 5	(T/2) + 5	ns	1
T_{PR}	CLKOUT Rise Time	1	6	1	6	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	6	1	6	ns	1, 5
OUTPUT DELAYS							
T_{CHOV1}	ALE, S2:0, DEN, DT/R, BHE (RFSH), LOCK, A19:16	3	22	3	25	ns	1, 4, 6, 7
T_{CHOV2}	GCS0:7, LCS, UCS, NCS, RD, WR	3	27	3	30	ns	1, 4, 6, 8
T_{CLOV1}	BHE (RFSH), DEN, LOCK, RESOUT, HLDA, T0OUT, T1OUT, A19:16	3	22	3	25	ns	1, 4, 6
T_{CLOV2}	RD, WR, GCS7:0, LCS, UCS, AD15:0 (AD7:0, A15:8), NCS, INTA1:0, S2:0	3	27	3	30	ns	1, 4, 6
T_{CHOF}	RD, WR, BHE (RFSH), DT/R, LOCK, S2:0, A19:16	0	25	0	25	ns	1
T_{CLOF}	DEN, AD15:0 (AD7:0, A15:8)	0	25	0	25	ns	1

AC SPECIFICATIONS

AC Characteristics—80C186EB20/80C186EB13 (Continued)

Symbol	Parameter	20 MHz		13 MHz		Units	Notes
		Min	Max	Min	Max		
SYNCHRONOUS INPUTS							
T_{CHIS}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY, $\overline{CTS1:0}$, P2.6, P2.7	10		10		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY, $\overline{CTS1:0}$	3		3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), READY	10		10		ns	1, 10
T_{CLIH}	READY, AD15:0 (AD7:0)	3		3		ns	1, 10
T_{CLIS}	HOLD, PEREQ, $\overline{ERROR}$	10		10		ns	1, 9
T_{CLIH}	HOLD, PEREQ, $\overline{ERROR}$	3		3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measure at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.

AC SPECIFICATIONS

AC Characteristics—80L186EB16

Symbol	Parameter	16 MHz		Units	Notes
		Min	Max		
INPUT CLOCK					
T_F	CLKIN Frequency	0	32	MHz	1
T_C	CLKIN Period	31.25	∞	ns	1
T_{CH}	CLKIN High Time	13	∞	ns	1, 2
T_{CL}	CLKIN Low Time	13	∞	ns	1, 2
T_{CR}	CLKIN Rise Time	1	8	ns	1, 3
T_{CF}	CLKIN Fall Time	1	8	ns	1, 3
OUTPUT CLOCK					
T_{CD}	CLKIN to CLKOUT Delay	0	30	ns	1, 4
T	CLKOUT Period		$2*T_C$	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PR}	CLKOUT Rise Time	1	9	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	9	ns	1, 5
OUTPUT DELAYS					
T_{CHOV1}	DT/$\overline{R}$, $\overline{LOCK}$, A19:16, $R_{\overline{FSH}}$	3	22	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{GCS0:7}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{NCS}$, $\overline{RD}$, $\overline{WR}$	3	27	ns	1, 4, 6, 8
T_{CHOV3}	$\overline{BHE}$, $\overline{DEN}$	3	25	ns	1, 4
T_{CHOV4}	ALE	3	30	ns	1, 4
T_{CHOV5}	$\overline{S2:0}$	3	33	ns	1, 4
T_{CLOV1}	$\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT, A19:16	3	22	ns	1, 4, 6
T_{CLOV2}	$\overline{RD}$, $\overline{WR}$, $\overline{GCS7:0}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{NCS}$, $\overline{INTA1:0}$, AD15:0 (AD7:0, A15:8)	3	27	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	25	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (AD7:0, A15:8)	0	25	ns	1
T_{CLOV3}	$\overline{BHE}$, $\overline{DEN}$	3	25	ns	1, 4, 6
T_{CLOV5}	$\overline{S2:0}$	3	33	ns	1, 4, 6

AC SPECIFICATIONS

AC Characteristics—80L186EB16 (Continued)

Symbol	Parameter	16 MHz		Units	Notes
		Min	Max		
SYNCHRONOUS INPUTS					
T_{CHIS}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY, $\overline{CTS1:0}$, P2.6, P2.7	15		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, INT4:0, T1:0IN, BCLK1:0, READY, $\overline{CTS1:0}$	3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), READY	15		ns	1, 10
T_{CLIH}	READY, AD15:0 (AD7:0)	3		ns	1, 10
T_{CLIS}	HOLD	15		ns	1, 9
T_{CLIH}	HOLD	3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measure at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.

AC SPECIFICATIONS

AC Characteristics—80L186EB13/80L186EB8

Symbol	Parameter	13 MHz		8 MHz		Units	Notes
		Min	Max	Min	Max		
INPUT CLOCK							
T_r	CLKIN Frequency	0	26	0	16	MHz	1
T_C	CLKIN Period	38.5	∞	62.5	∞	ns	1
T_{CH}	CLKIN High Time	15	∞	15	∞	ns	1, 2
T_{CL}	CLKIN Low Time	15	∞	15	∞	ns	1, 2
T_{CR}	CLKIN Rise Time	1	8	1	8	ns	1, 3
T_{CF}	CLKIN Fall Time	1	8	1	8	ns	1, 3
OUTPUT CLOCK							
T_{CD}	CLKIN to CLKOUT Delay	0	10	0	50	ns	1, 4
T	CLKOUT Period		$2*T_C$		$2*T_C$	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5	(T/2) +5	(T/2) − 5	(T/2) +5	ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5	(T/2) + 5	(T/2) − 5	(T/2) + 5	ns	1
T_{PR}	CLKOUT Rise Time	1	10	1	15	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	10	1	15	ns	1, 5
OUTPUT DELAYS							
T_{CHOV1}	ALE, $\overline{S2\text{-}0}$, $\overline{DEN}$, DT/$\overline{R}$, $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$, A19:16	3	25	3	30	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{GCS0:7}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{NCS}$, $\overline{RD}$, $\overline{WR}$	3	30	3	35	ns	1, 4,6, 8
T_{CLOV1}	$\overline{BHE}$ ($\overline{RFSH}$), $\overline{DEN}$, $\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT, A19:16	3	25	3	30	ns	1, 4, 6
T_{CLOV2}	$\overline{S2:0}$, $\overline{RD}$, $\overline{WR}$, $\overline{GCS7:0}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{NCS}$, $\overline{INTA1:0}$, AD15:0 (AD7:0, A15:8)	3	30	3	35	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	30	0	30	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (AD7:0, A15:8)	0	30	0	35	ns	1

AC SPECIFICATIONS

AC Characteristics—80L186EB13/80L186EB8 (Continued)

Symbol	Parameter	13 MHz		8 MHz		Units	Notes
		Min	Max	Min	Max		
SYNCHRONOUS INPUTS							
T_{CHIS}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY $\overline{CTS1:0}$, P2.6, P2.7	20		25		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, INT4:0, T1:0IN, BCLK1:0, READY, $\overline{CTS1:0}$	3		3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), READY	20		25		ns	1, 10
T_{CLIH}	READY, AD15:0 (AD7:0)	3		3		ns	1, 10
T_{CLIS}	HOLD	20		25		ns	1, 9
T_{CLIH}	HOLD	3		3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measured at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.

AC SPECIFICATIONS (Continued)

Relative Timings (80C186EB25, 20, 13/80L186EB16, 13, 8)

Symbol	Parameter	Min	Max	Units	Notes
RELATIVE TIMINGS					
T_{LHLL}	ALE Rising to ALE Falling	T − 15		ns	
T_{AVLL}	Address Valid to ALE Falling	½T − 10		ns	
T_{PLLL}	Chip Selects Valid to ALE Falling	½T − 10		ns	1
T_{LLAX}	Address Hold from ALE Falling	½T − 10		ns	
T_{LLWL}	ALE Falling to $\overline{WR}$ Falling	½T − 15		ns	1
T_{LLRL}	ALE Falling to $\overline{RD}$ Falling	½T − 15		ns	1
T_{WHLH}	$\overline{WR}$ Rising to ALE Rising	½T − 10		ns	1
T_{AFRL}	Address Float to $\overline{RD}$ Falling	0		ns	
T_{RLRH}	$\overline{RD}$ Falling to $\overline{RD}$ Rising	(2*T) − 5		ns	2
T_{WLWH}	$\overline{WR}$ Falling to $\overline{WR}$ Rising	(2*T) − 5		ns	2
T_{RHAV}	$\overline{RD}$ Rising to Address Active	T − 15		ns	
T_{WHDX}	Output Data Hold after $\overline{WR}$ Rising	T − 15		ns	
T_{WHPH}	$\overline{WR}$ Rising to Chip Select Rising	½T − 10		ns	1
T_{RHPH}	$\overline{RD}$ Rising to Chip Select Rising	½T − 10		ns	1
T_{PHPL}	$\overline{CS}$ Inactive to $\overline{CS}$ Active	½T − 10		ns	1
T_{OVRH}	$\overline{ONCE}$ Active to $\overline{RESIN}$ Rising	T		ns	3
T_{RHOX}	$\overline{ONCE}$ Hold from $\overline{RESIN}$ Rising	T		ns	3

NOTES:
1. Assumes equal loading on both pins.
2. Can be extended using wait states.
3. Not tested

AC SPECIFICATIONS (Continued)

Serial Port Mode 0 Timings (80C186EB25, 20, 13/80L186EB16, 13, 8)

Symbol	Parameter	Min	Max	Unit	Notes
T_{XLXL}	TXD Clock Period	T (n + 1)		ns	1, 2
T_{XLXH}	TXD Clock Low to Clock High (n > 1)	2T − 35	2T + 35	ns	1
T_{XLXH}	TXD Clock Low to Clock High (n = 1)	T − 35	T + 35	ns	1
T_{XHXL}	TXD Clock High to Clock Low (n > 1)	(n − 1) T − 35	(n − 1) T + 35	ns	1, 2
T_{XHXL}	TXD Clock High to Clock Low (n = 1)	T − 35	T + 35	ns	1
T_{QVXH}	RXD Output Data Setup to TXD Clock High (n > 1)	(n − 1) T − 35		ns	1, 2
T_{QVXH}	RXD Output Data Setup to TXD Clock High (n = 1)	T − 35		ns	1
T_{XHQX}	RXD Output Data Hold after TXD Clock High (n > 1)	2T − 35		ns	1
T_{XHQX}	RXD Output Data Hold after TXD Clock High (n = 1)	T − 35		ns	1
T_{XHQZ}	RXD Output Data Float after Last TXD Clock High		T + 20	ns	1
T_{DVXH}	RXD Input Data Setup to TXD Clock High	T + 20		ns	1
T_{XHDX}	RXD Input Data Hold after TXD Clock High	0		ns	1

NOTES:
1. See Figure 12 for waveforms.
2. n is the value of the BxCMP register ignoring the ICLK Bit (i.e., ICLK = 0).

AC TEST CONDITIONS

The AC specifications are tested with the 50 pF load shown in Figure 7. See the Derating Curves section to see how timings vary with load capacitance.

Specifications are measured at the $V_{CC}/2$ crossing point, unless otherwise specified. See AC Timing Waveforms, for AC specification definitions, test pins, and illustrations.

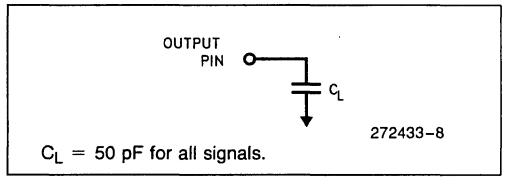


Figure 7. AC Test Load

AC TIMING WAVEFORMS

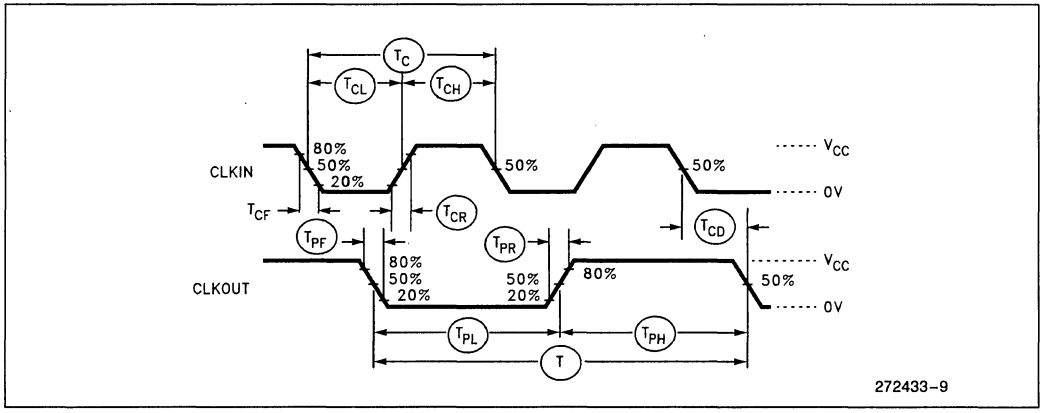
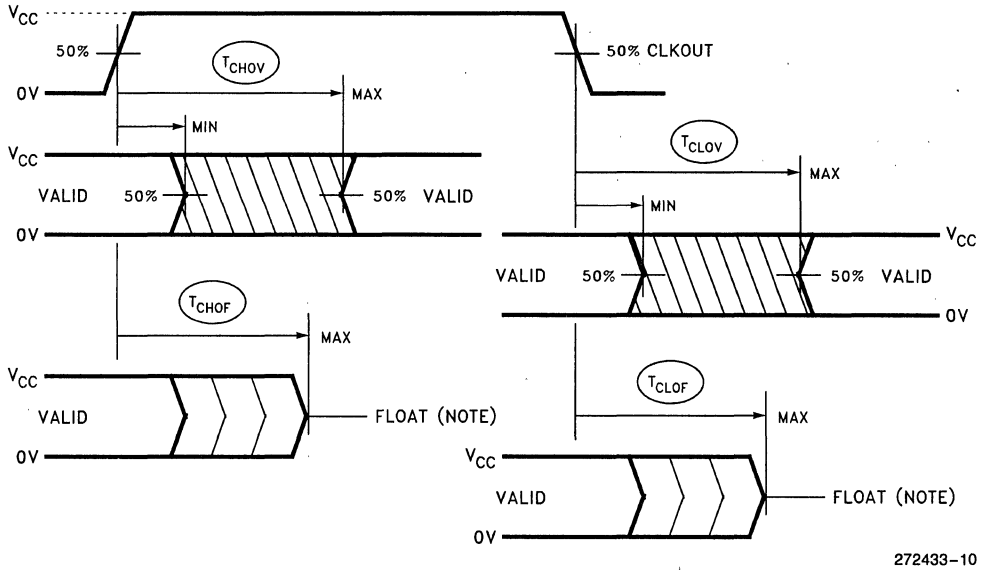


Figure 8. Input and Output Clock Waveform



NOTE:
20% V_{CC} < Float < 80% V_{CC}

Figure 9. Output Delay and Float Waveform

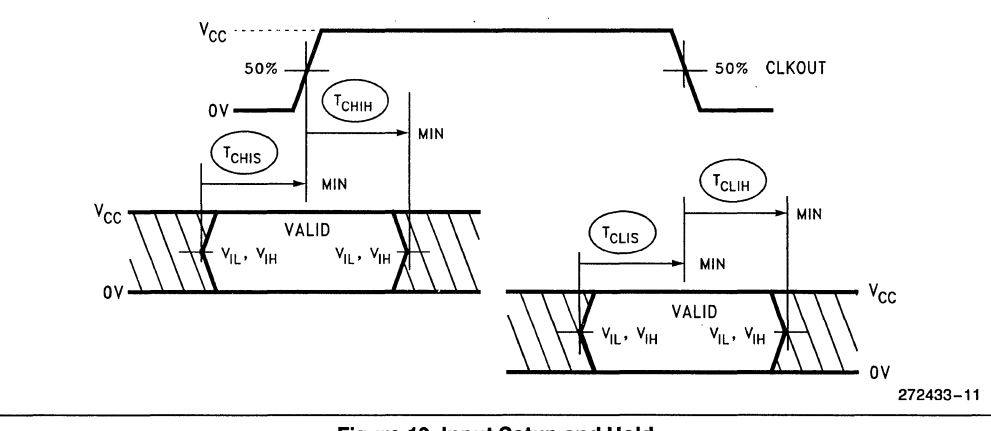
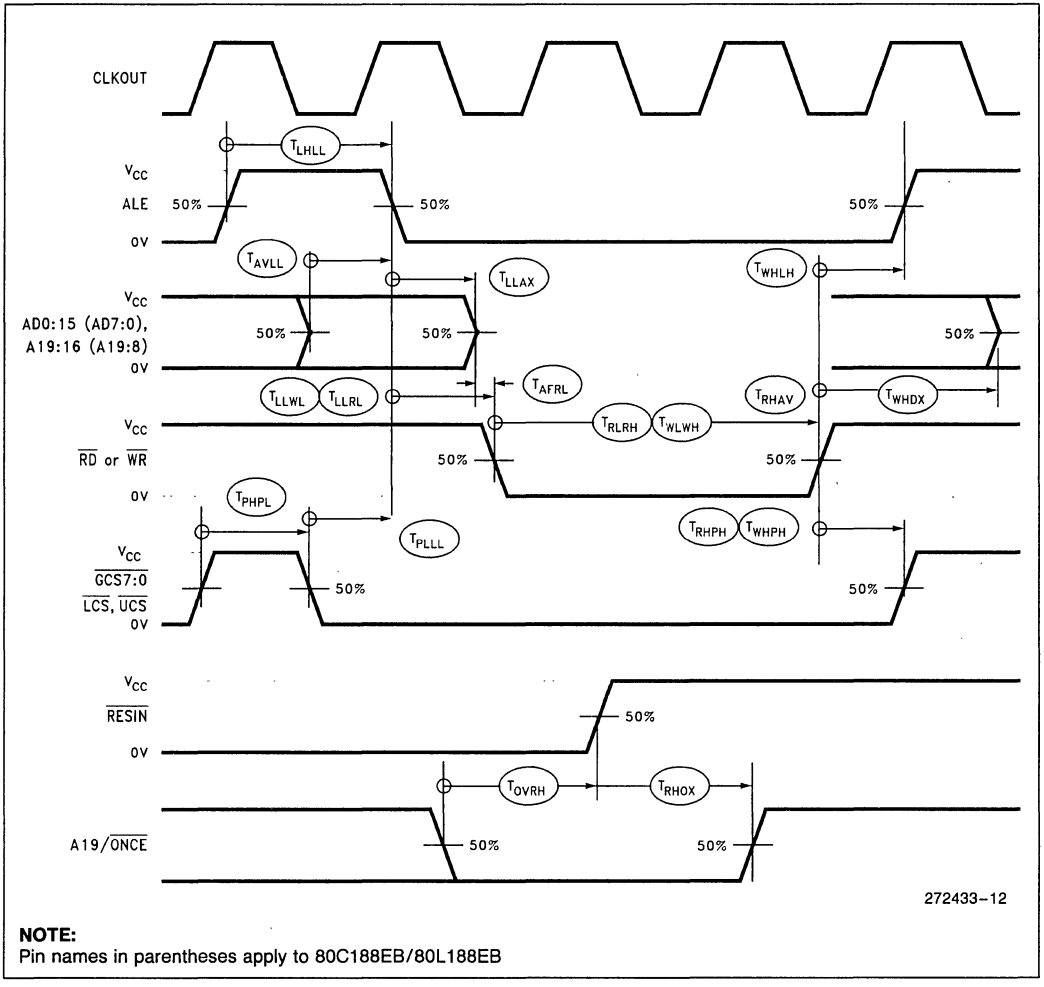


Figure 10. Input Setup and Hold



NOTE:
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 11. Relative Signal Waveform

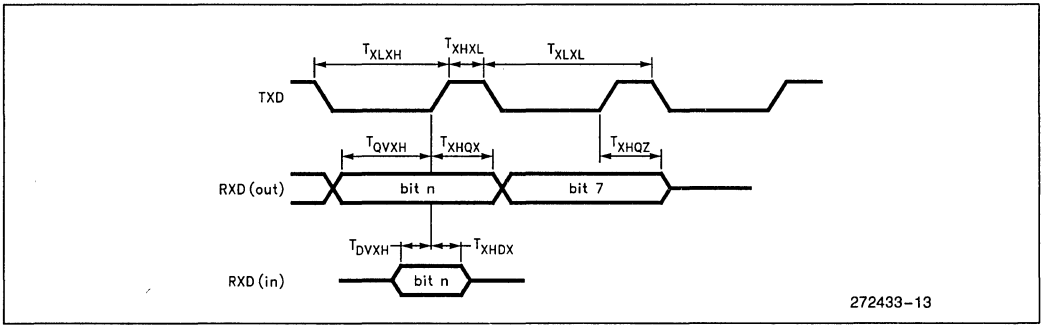


Figure 12. Serial Port Mode 0 Waveform

DERATING CURVES

TYPICAL OUTPUT DELAY VARIATIONS VERSUS LOAD CAPACITANCE

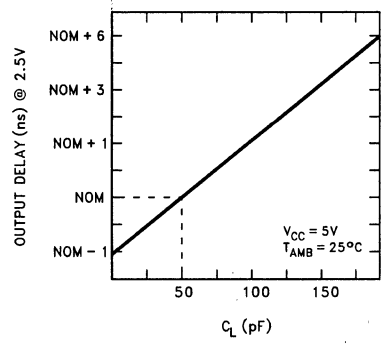


Figure 13

TYPICAL RISE AND FALL VARIATIONS VERSUS LOAD CAPACITANCE

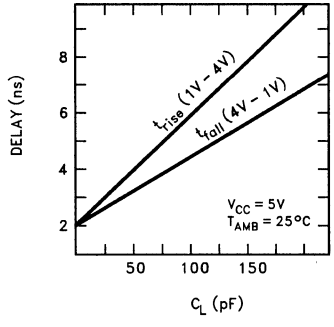


Figure 14

RESET

The processor will perform a reset operation any time the RESIN pin active. The RESIN pin is actually synchronized before it is presented internally, which means that the clock must be operating before a reset can take effect. From a power-on state, RESIN must be held active (low) in order to guarantee correct initialization of the processor. **Failure to provide RESIN while the device is powering up will result in unspecified operation of the device.**

Figure 14 shows the correct reset sequence when first applying power to the processor. An external clock connected to CLKIN must not exceed the V_{CC} threshold being applied to the processor. This is normally not a problem if the clock driver is supplied with the same V_{CC} that supplies the processor. When attaching a crystal to the device, RESIN must remain active until both V_{CC} and CLKOUT are stable (the length of time is application specific and depends on the startup characteristics of the crystal circuit). The RESIN pin is designed to operate correctly using an RC reset circuit, but the designer must ensure that the ramp time for V_{CC} is not so long that RESIN is never really sampled at a logic low level when V_{CC} reaches minimum operating conditions.

Figure 16 shows the timing sequence when RESIN is applied after V_{CC} is stable and the device has been operating. Note that a reset will terminate all activity and return the processor to a known operating state. Any bus operation that is in progress at the time RESIN is asserted will terminate immediately (note that most control signals will be driven to their inactive state first before floating).

While RESIN is active, bus signals LOCK, A19/ONCE, and A18:16 are configured as inputs and weakly held high by internal pullup transistors. Only 19/ONCE can be overdriven to a low and is used to enable ONCE Mode. Forcing LOCK or A18:16 low at any time while RESIN is low is prohibited and will cause unspecified device operation.

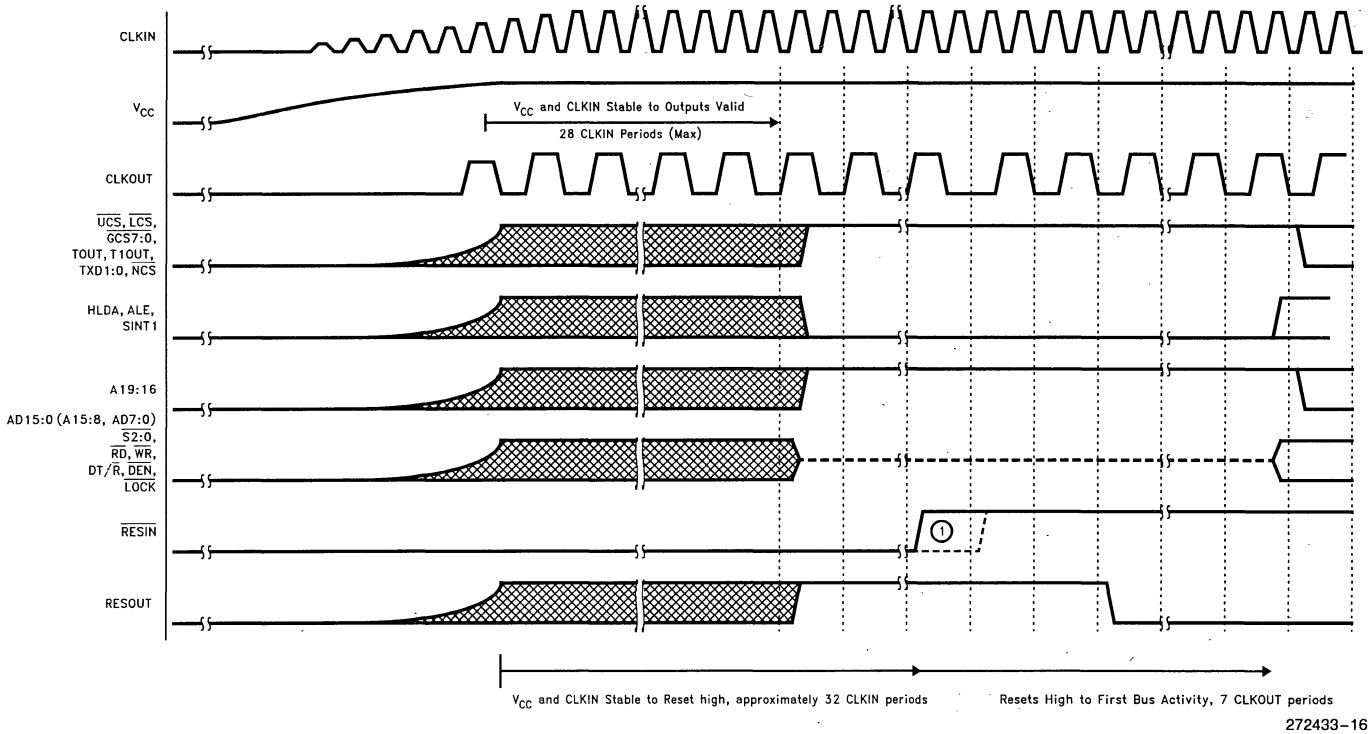


Figure 15. Cold Reset Waveforms

NOTE:
CLKOUT synchronization occurs on the rising edge of $\overline{RESIN}$. If $\overline{RESIN}$ is sampled high while CLKOUT is high (solid line), then CLKOUT will remain low for two CLKIN periods. If $\overline{RESIN}$ is sampled high while CLKOUT is low (dashed line), then CLKOUT will not be affected.
Pin names in parentheses apply to 80C188EB/80L188EB

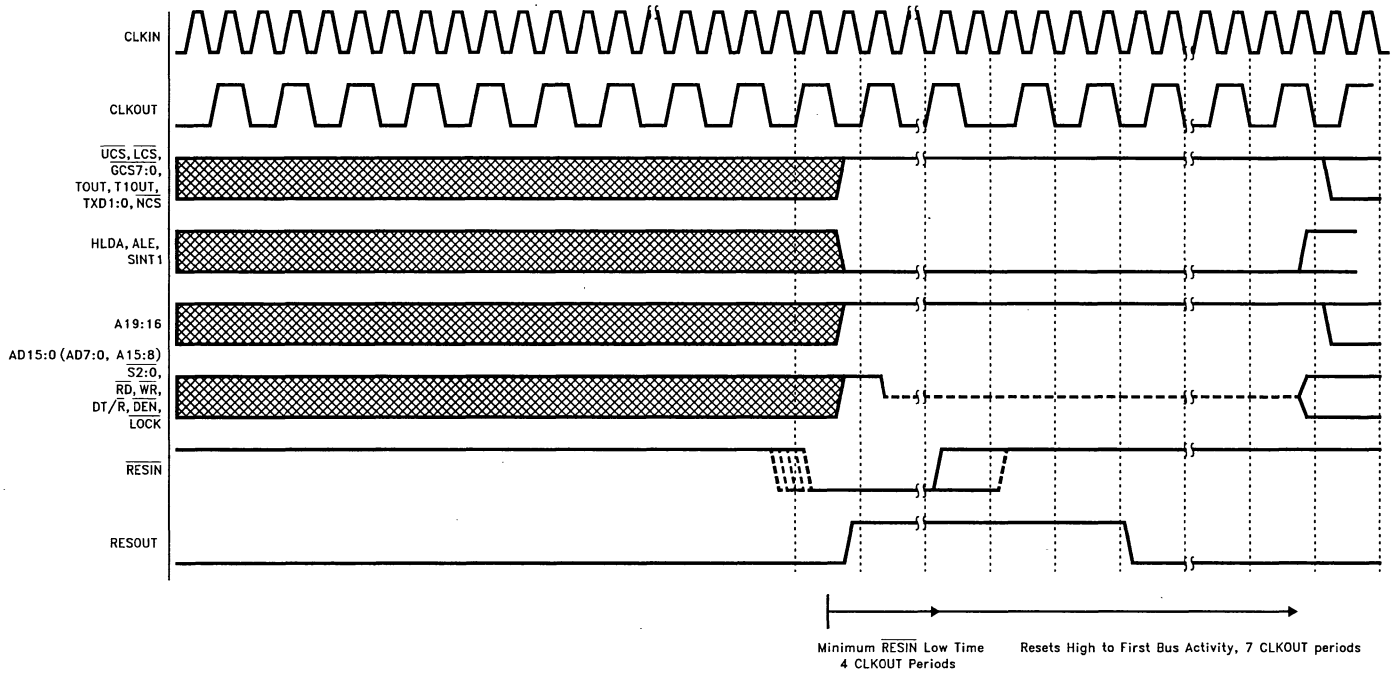
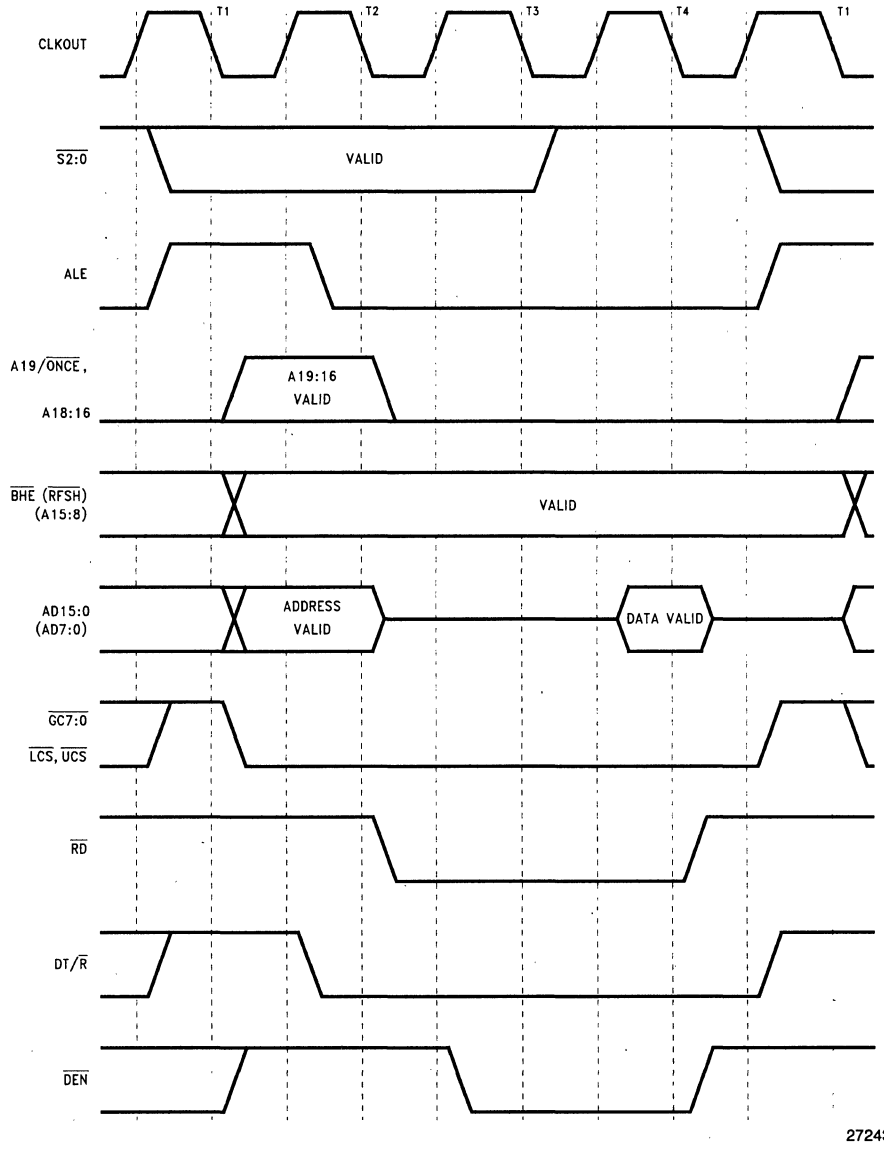


Figure 16. Warm Reset Waveforms

NOTE:
CLKOUT synchronization occurs on the rising edge of RESIN. If RESIN is sampled high while CLKOUT is high (solid line), then CLKOUT will remain low for two CLKIN periods. If RESIN is sampled high while CLKOUT is low (dashed line), then CLKOUT will not be affected.
Pin names in parentheses apply to 80C188EB/80L188EB

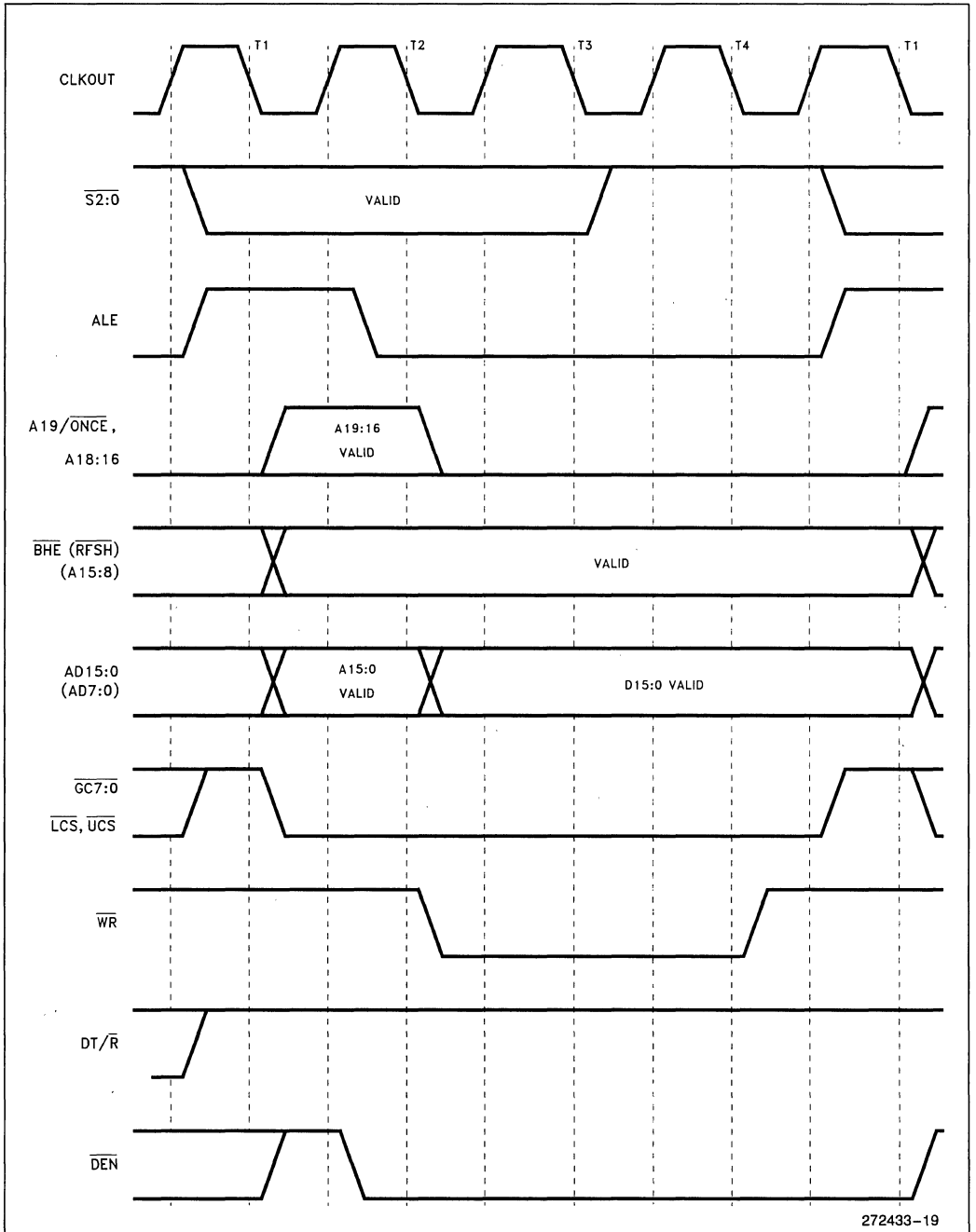
BUS CYCLE WAVEFORMS

Figures 17 through 23 present the various bus cycles that are generated by the processor. What is shown in the figure is the relationship of the various bus signals to CLKOUT. These figures along with the information present in **AC Specifications** allow the user to determine all the critical timing analysis needed for a given application.



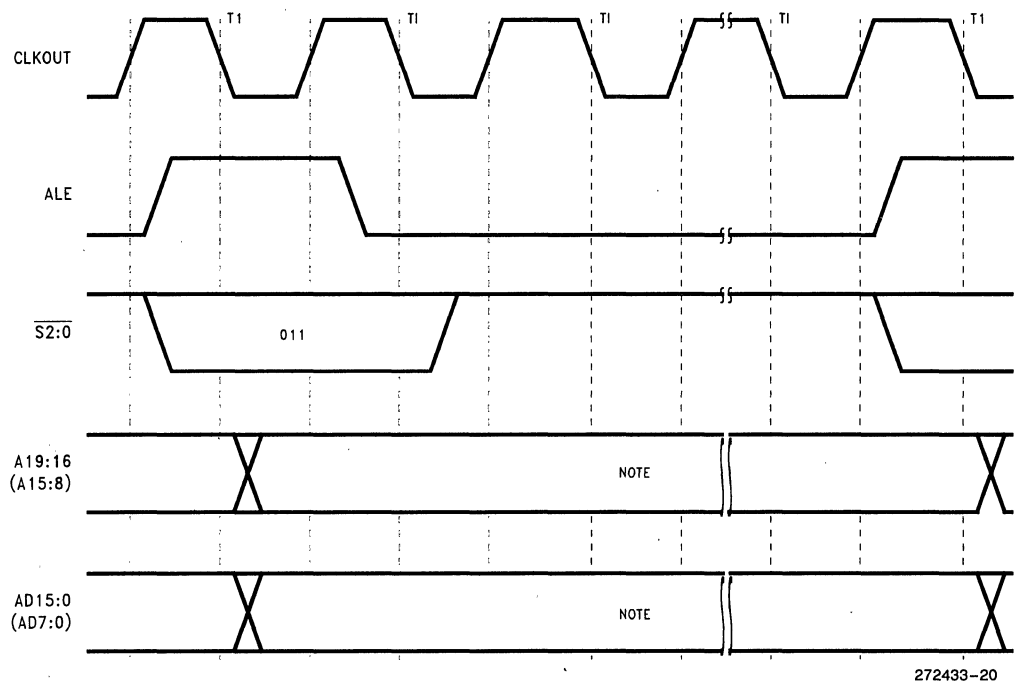
NOTE:
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 17. Read, Fetch, and Refresh Cycle Waveforms



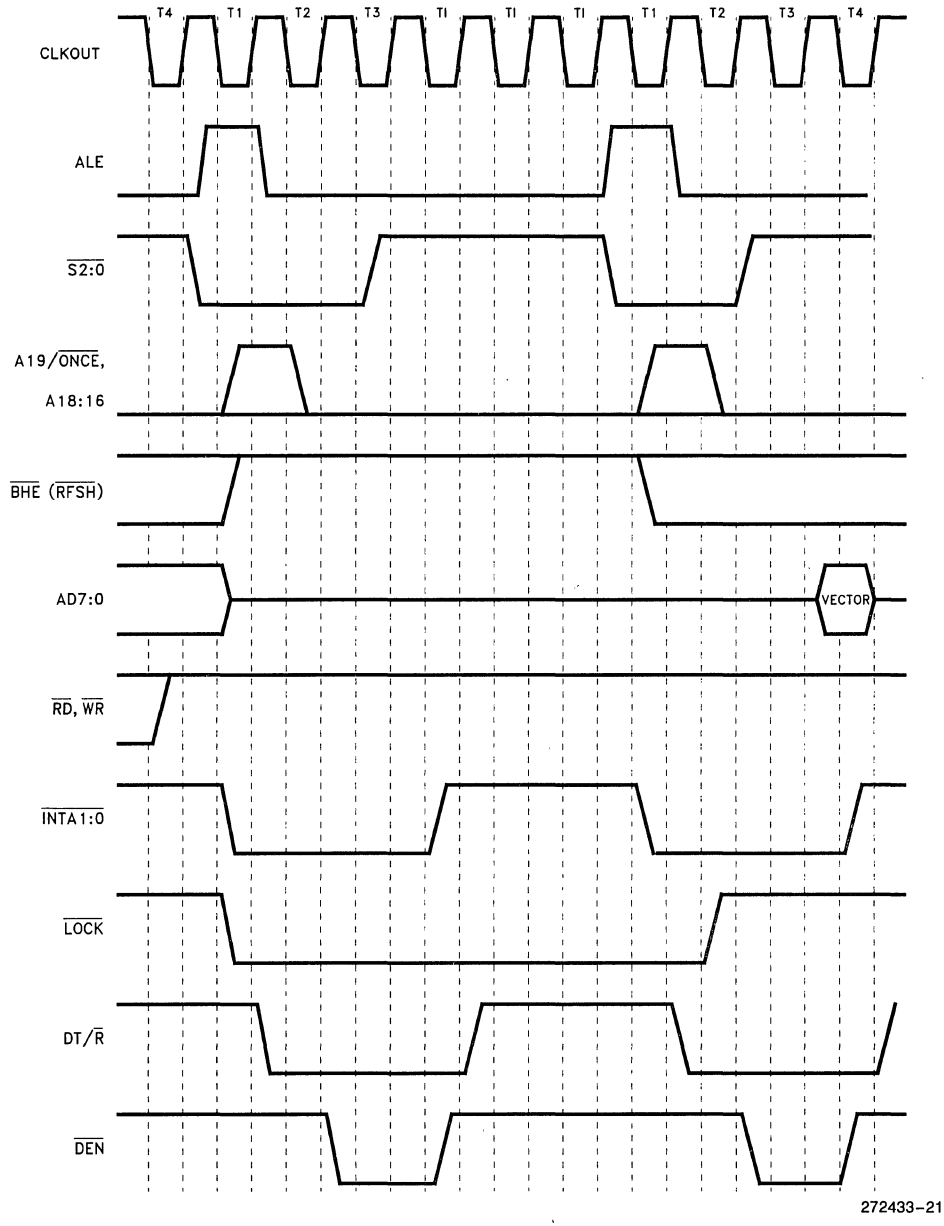
NOTE:
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 18. Write Cycle Waveforms



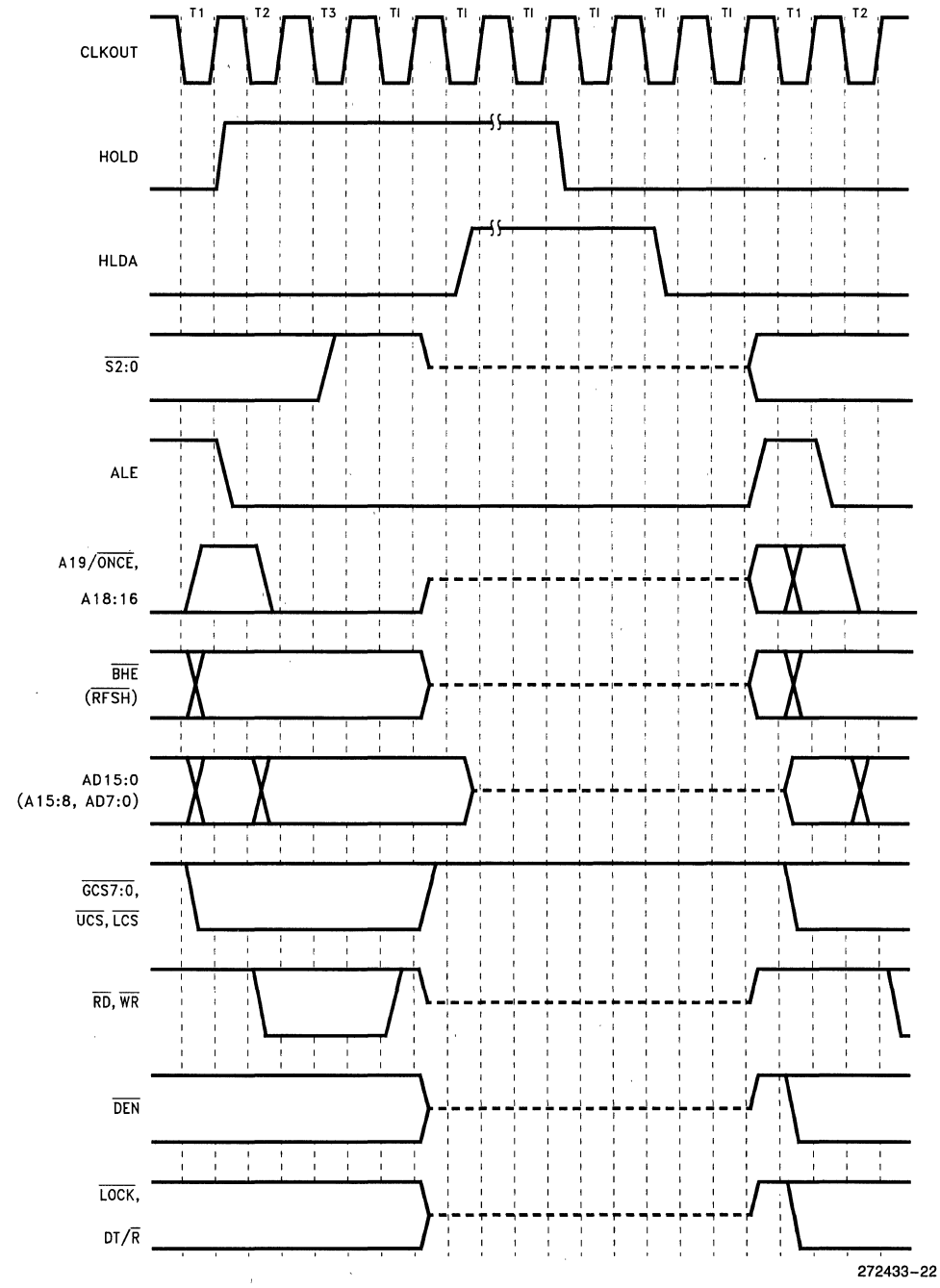
NOTE:
The address driven is typically the location of the next instruction prefetch. Under a majority of instruction sequences the AD15:0 (AD7:0) bus will float, while the A19:16 (A19:8) bus remains driven and all bus control signals are driven to their inactive state.
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 19. Halt Cycle Waveforms



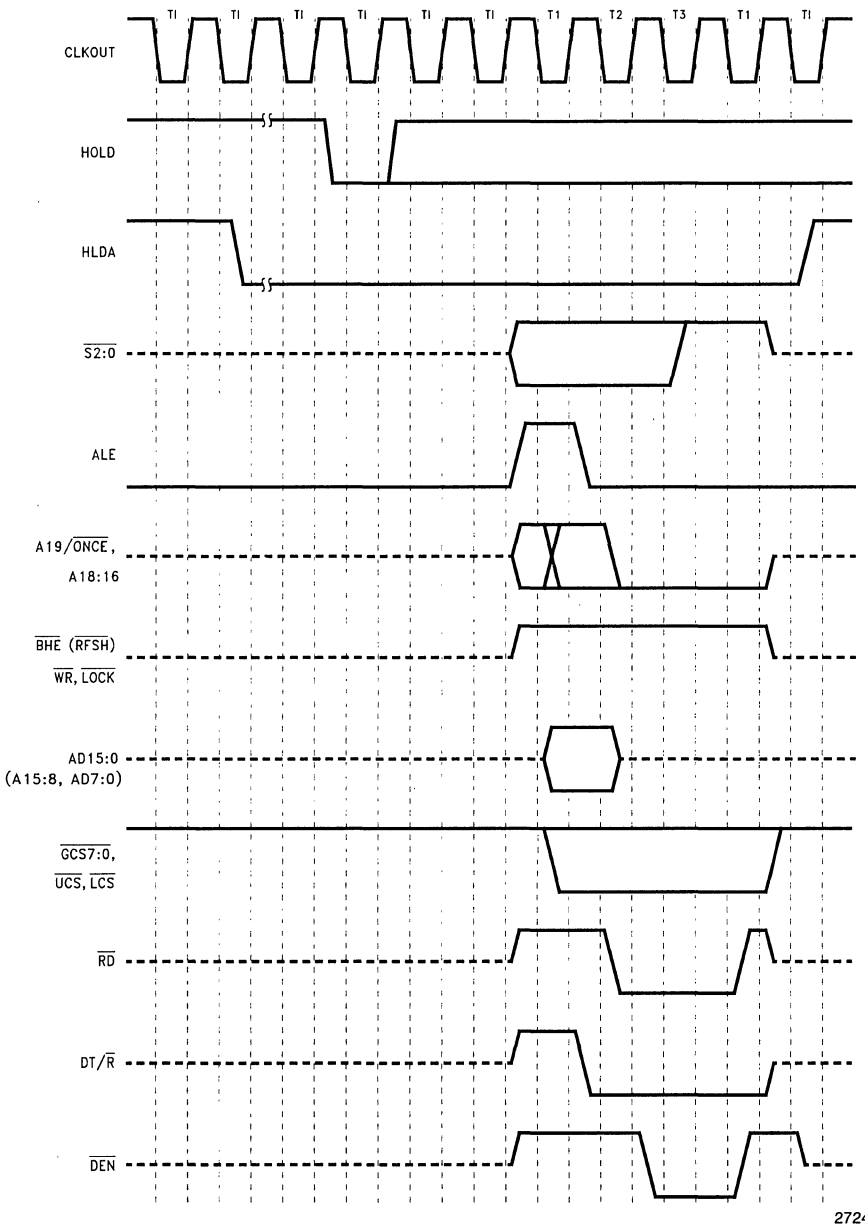
NOTE:
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 20. Interrupt Acknowledge Cycle Waveform



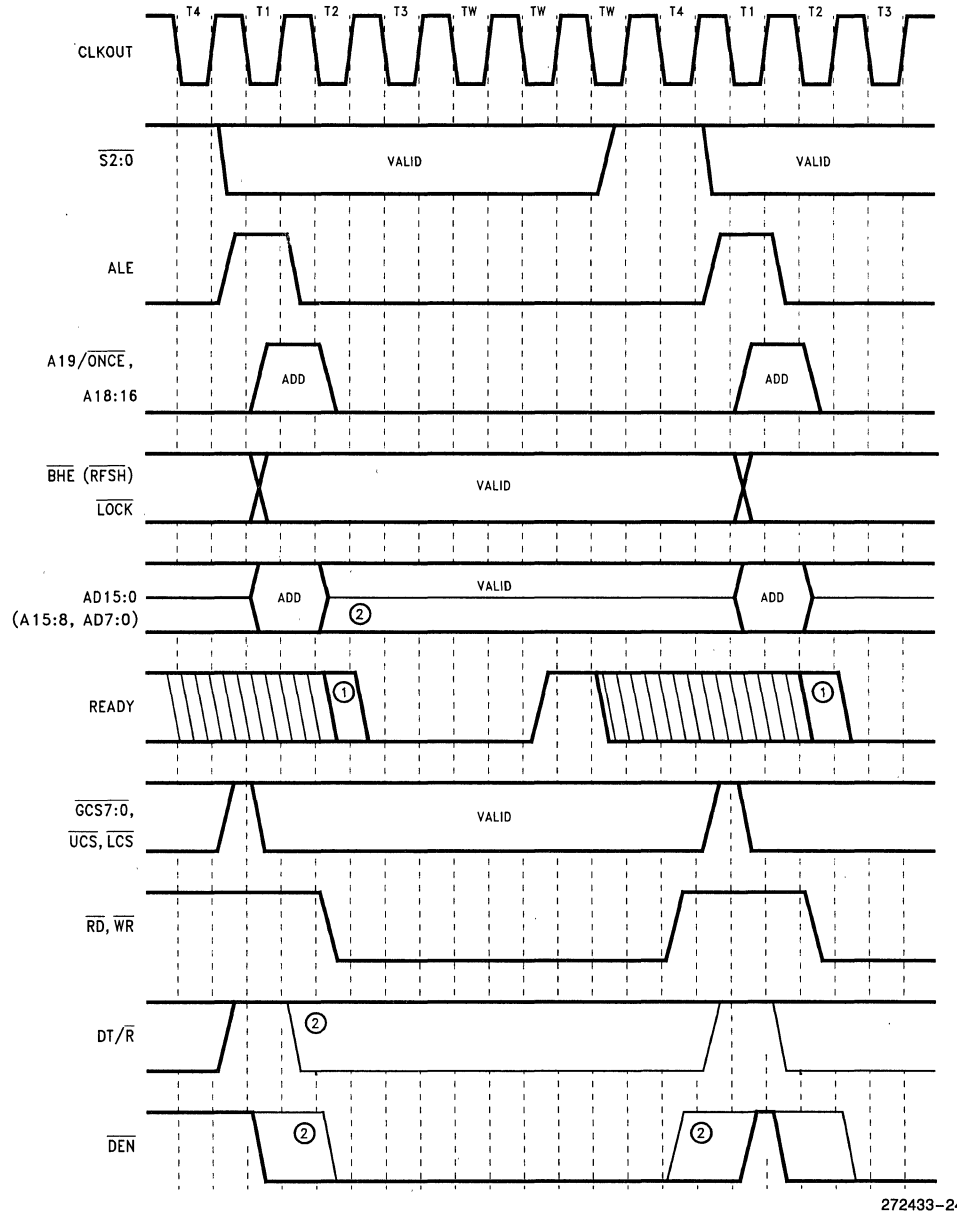
NOTE:
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 21. HOLD/HLDA Waveforms



NOTES:
1. READY must be low by either edge to cause a wait state.
2. Lighter lines indicate READ cycles, darker lines indicate WRITE cycles.
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 22. Refresh during Hold Acknowledge



NOTES:
1. READY must be low by either edge to cause a wait state.
2. Lighter lines indicate READ cycles, darker lines indicate WRITE cycles.
Pin names in parentheses apply to 80C188EB/80L188EB

Figure 23. Ready Waveforms

EXECUTION TIMINGS

A determination of program execution timing must consider the bus cycles necessary to prefetch instructions as well as the number of execution unit cycles necessary to execute instructions. The following instruction timings represent the **minimum** execution time in clock cycles for each instruction. The timings given are based on the following assumptions:

- The opcode, along with any data or displacement required for execution of a particular instruction, has been prefetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDs occur.
- All word-data is located on even-address boundaries (80C186EB only).

All jumps and calls include the time required to fetch the opcode of the next instruction at the destination address.

All instructions which involve memory accesses can require one or two additional clocks above the minimum timings shown due to the asynchronous handshake between the bus interface unit (BIU) and execution unit.

With a 16-bit BIU, the 80C186EB has sufficient bus performance to ensure that an adequate number of prefetched bytes will reside in the queue (6 bytes) most of the time. Therefore, actual program execution time will not be substantially greater than that derived from adding the instruction timings shown.

The 80C188EB 8-bit BIU is limited in its performance relative to the execution unit. A sufficient number of prefetched bytes may not reside in the prefetch queue (4 bytes) much of the time. Therefore, actual program execution time will be substantially greater than that derived from adding the instruction timings shown.

INSTRUCTION SET SUMMARY

Function	Format				80C186EB Clock Cycles	80C188EB Clock Cycles	Comments
DATA TRANSFER							
MOV = Move:							
Register to Register/Memory	1000100w	mod reg r/m			2/12	2/12*	
Register/memory to register	1000101w	mod reg r/m			2/9	2/9*	
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1	12/13	12/13	8/16-bit
Immediate to register	1011w reg	data	data if w=1		3/4	3/4	8/16-bit
Memory to accumulator	1010000w	addr-low	addr-high		8	8*	
Accumulator to memory	1010001w	addr-low	addr-high		9	9*	
Register/memory to segment register	10001110	mod 0 reg r/m			2/9	2/13	
Segment register to register/memory	10001100	mod 0 reg r/m			2/11	2/15	
PUSH = Push:							
Memory	11111111	mod 1 1 0 r/m			16	20	
Register	01010 reg				10	14	
Segment register	000 reg 1 1 0				9	13	
Immediate	011010s0	data	data if s=0		10	14	
PUSHA = Push All	01100000				36	68	
POP = Pop:							
Memory	10001111	mod 0 0 0 r/m			20	24	
Register	01011 reg				10	14	
Segment register	000 reg 1 1 1	(reg≠01)			8	12	
POPA = Pop All	01100001				51	83	
XCHG = Exchange:							
Register/memory with register	1000011w	mod reg r/m			4/17	4/17*	
Register with accumulator	10010 reg				3	3	
IN = Input from:							
Fixed port	1110010w	port			10	10*	
Variable port	1110110w				8	8*	
OUT = Output to:							
Fixed port	1110011w	port			9	9*	
Variable port	1110111w				7	7*	
XLAT = Translate byte to AL	11010111				11	15	
LEA = Load EA to register	10001101	mod reg r/m			6	6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod≠11)		18	26	
LES = Load pointer to ES	11000100	mod reg r/m	(mod≠11)		18	26	
LAHF = Load AH with flags	10011111				2	2	
SAHF = Store AH into flags	10011110				3	3	
PUSHF = Push flags	10011100				9	13	
POPF = Pop flags	10011101				8	12	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EB Clock Cycles	80C188EB Clock Cycles	Comments
DATA TRANSFER (Continued)							
SEGMENT = Segment Override:							
CS	00101110				2	2	
SS	00110110				2	2	
DS	00111110				2	2	
ES	00100110				2	2	
ARITHMETIC							
ADD = Add:							
Reg/memory with register to either	000000dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 000 r/m	data	data if s w=01	4/16	4/16*	
Immediate to accumulator	0000010w	data	data if w=1		3/4	3/4	8/16-bit
ADC = Add with carry:							
Reg/memory with register to either	000100dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 010 r/m	data	data if s w=01	4/16	4/16*	
Immediate to accumulator	0001010w	data	data if w=1		3/4	3/4	8/16-bit
INC = Increment:							
Register/memory	1111111w	mod 000 r/m			3/15	3/15*	
Register	01000 reg				3	3	
SUB = Subtract:							
Reg/memory and register to either	001010dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 101 r/m	data	data if s w=01	4/16	4/16*	
Immediate from accumulator	0010110w	data	data if w=1		3/4	3/4	8/16-bit
SBB = Subtract with borrow:							
Reg/memory and register to either	000110dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 011 r/m	data	data if s w=01	4/16	4/16*	
Immediate from accumulator	0001110w	data	data if w=1		3/4	3/4*	8/16-bit
DEC = Decrement							
Register/memory	1111111w	mod 001 r/m			3/15	3/15*	
Register	01001 reg				3	3	
CMP = Compare:							
Register/memory with register	0011101w	mod reg r/m			3/10	3/10*	
Register with register/memory	0011100w	mod reg r/m			3/10	3/10*	
Immediate with register/memory	100000sw	mod 111 r/m	data	data if s w=01	3/10	3/10*	
Immediate with accumulator	0011110w	data	data if w=1		3/4	3/4	8/16-bit
NEG = Change sign register/memory	1111011w	mod 011 r/m			3/10	3/10*	
AAA = ASCII adjust for add	00110111				8	8	
DAA = Decimal adjust for add	00100111				4	4	
AAS = ASCII adjust for subtract	00111111				7	7	
DAS = Decimal adjust for subtract	00101111				4	4	
MUL = Multiply (unsigned):	1111011w	mod 100 r/m					
Register-Byte					26-28	26-28	
Register-Word					35-37	35-37	
Memory-Byte					32-34	32-34	
Memory-Word					41-43	41-43*	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186EB Clock Cycles	80C188EB Clock Cycles	Comments
ARITHMETIC (Continued)				
IMUL = Integer multiply (signed):	1111011w | mod101 r/m			
Register-Byte		25–28	25–28	
Register-Word		34–37	34–37	
Memory-Byte		31–34	31–34	
Memory-Word		40–43	40–43*	
IMUL = Integer Immediate multiply (signed)	011010s1 | mod reg r/m | data | data if s=0	22–25/ 29–32	22–25/ 29–32	
DIV = Divide (unsigned):	1111011w | mod110 r/m			
Register-Byte		29	29	
Register-Word		38	38	
Memory-Byte		35	35	
Memory-Word		44	44*	
IDIV = Integer divide (signed):	1111011w | mod111 r/m			
Register-Byte		44–52	44–52	
Register-Word		53–61	53–61	
Memory-Byte		50–58	50–58	
Memory-Word		59–67	59–67*	
AAM = ASCII adjust for multiply	11010100 | 00001010	19	19	
AAD = ASCII adjust for divide	11010101 | 00001010	15	15	
CBW = Convert byte to word	10011000	2	2	
CWD = Convert word to double word	10011001	4	4	
LOGIC				
Shift/Rotate Instructions:				
Register/Memory by 1	1101000w | mod TTT r/m	2/15	2/15	
Register/Memory by CL	1101001w | mod TTT r/m	5+n/17+n	5+n/17+n	
Register/Memory by Count	1100000w | mod TTT r/m | count	5+n/17+n	5+n/17+n	
	TTT Instruction 000 ROL 001 ROR 010 RCL 011 RCR 100 SHL/SAL 101 SHR 111 SAR			
AND = **And:**				
Reg/memory and register to either	001000dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod100 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0010010w | data | data if w=1	3/4	3/4*	8/16-bit
TEST = **And function to flags, no result:**				
Register/memory and register	1000010w | mod reg r/m	3/10	3/10*	
Immediate data and register/memory	1111011w | mod000 r/m | data | data if w=1	4/10	4/10*	
Immediate data and accumulator	1010100w | data | data if w=1	3/4	3/4	8/16-bit
OR = **Or:**				
Reg/memory and register to either	000010dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod001 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0000110w | data | data if w=1	3/4	3/4*	8/16-bit

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EB Clock Cycles	80C188EB Clock Cycles	Comments
LOGIC (Continued)							
XOR = Exclusive or:							
Reg/memory and register to either	001100dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	1000000w	mod 110 r/m	data	data if w=1	4/16	4/16*	
Immediate to accumulator	0011010w	data	data if w=1		3/4	3/4	8/16-bit
NOT = Invert register/memory	1111011w	mod 010 r/m			3/10	3/10*	
STRING MANIPULATION							
MOVS = Move byte/word	1010010w				14	14*	
CMPS = Compare byte/word	1010011w				22	22*	
SCAS = Scan byte/word	1010111w				15	15*	
LODS = Load byte/wd to AL/AX	1010110w				12	12*	
STOS = Store byte/wd from AL/AX	1010101w				10	10*	
INS = Input byte/wd from DX port	0110110w				14	14	
OUTS = Output byte/wd to DX port	0110111w				14	14	
Repeated by count in CX (REP/REPE/REPZ/REPNE/REPNZ)							
MOVS = Move string	11110010	1010010w			8+8n	8+8n*	
CMPS = Compare string	1111001z	1010011w			5+22n	5+22n*	
SCAS = Scan string	1111001z	1010111w			5+15n	5+15n*	
LODS = Load string	11110010	1010110w			6+11n	6+11n*	
STOS = Store string	11110010	1010101w			6+9n	6+9n*	
INS = Input string	11110010	0110110w			8+8n	8+8n*	
OUTS = Output string	11110010	0110111w			8+8n	8+8n*	
CONTROL TRANSFER							
CALL = Call:							
Direct within segment	11101000	disp-low	disp-high		15	19	
Register/memory indirect within segment	11111111	mod 010 r/m			13/19	17/27	
Direct intersegment	10011010	segment offset			23	31	
		segment selector					
Indirect intersegment	11111111	mod 011 r/m	(mod ≠ 11)		38	54	
JMP = Unconditional jump:							
Short/long	11101011	disp-low			14	14	
Direct within segment	11101001	disp-low	disp-high		14	14	
Register/memory indirect within segment	11111111	mod 100 r/m			11/17	11/21	
Direct intersegment	11101010	segment offset			14	14	
		segment selector					
Indirect intersegment	11111111	mod 101 r/m	(mod ≠ 11)		26	34	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EB Clock Cycles	80C188EB Clock Cycles	Comments
CONTROL TRANSFER (Continued)							
RET = **Return from CALL:**							
Within segment	11000011				16	20	
Within seg adding immed to SP	11000010	data-low	data-high		18	22	
Intersegment	11001011				22	30	
Intersegment adding immediate to SP	11001010	data-low	data-high		25	33	
JE/JZ = Jump on equal/zero	01110100	disp			4/13	4/13	JMP not taken/JMP taken
JL/JNGE = Jump on less/not greater or equal	01111100	disp			4/13	4/13	
JLE/JNG = Jump on less or equal/not greater	01111110	disp			4/13	4/13	
JB/JNAE = Jump on below/not above or equal	01110010	disp			4/13	4/13	
JBE/JNA = Jump on below or equal/not above	01110110	disp			4/13	4/13	
JP/JPE = Jump on parity/parity even	01111010	disp			4/13	4/13	
JO = Jump on overflow	01110 000	disp			4/13	4/13	
JS = Jump on sign	01111000	disp			4/13	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101	disp			4/13	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101	disp			4/13	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111	disp			4/13	4/13	
JNB/JAE = Jump on not below/above or equal	01110011	disp			4/13	4/13	
JNBE/JA = Jump on not below or equal/above	01110111	disp			4/13	4/13	
JNP/JPO = Jump on not par/par odd	01111011	disp			4/13	4/13	
JNO = Jump on not overflow	01110001	disp			4/13	4/13	
JNS = Jump on not sign	01111001	disp			4/13	4/13	
JCXZ = Jump on CX zero	11100011	disp			5/15	5/15	
LOOP = Loop CX times	11100010	disp			6/16	6/16	LOOP not taken/LOOP taken
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp			6/16	6/16	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp			6/16	6/16	
ENTER = Enter Procedure	11001000	data-low	data-high	L			
L = 0					15	19	
L = 1					25	29	
L > 1					22+16(n−1)	26+20(n−1)	
LEAVE = Leave Procedure	11001001				8	8	
INT = **Interrupt:**							
Type specified	11001101	type			47	47	
Type 3	11001100				45	45	if INT. taken/ if INT. not taken
INTO = Interrupt on overflow	11001110				48/4	48/4	
IRET = Interrupt return	11001111				28	28	
BOUND = Detect value out of range	01100010	mod reg r/m			33–35	33–35	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186EB Clock Cycles	80C188EB Clock Cycles	Comments
PROCESSOR CONTROL				
CLC = Clear carry	11111000	2	2	
CMC = Complement carry	11110101	2	2	
STC = Set carry	11111001	2	2	
CLD = Clear direction	11111100	2	2	
STD = Set direction	11111101	2	2	
CLI = Clear interrupt	11111010	2	2	
STI = Set interrupt	11111011	2	2	
HLT = Halt	11110100	2	2	
WAIT = Wait	10011011	6	6	if $\overline{TEST}$ = 0
LOCK = Bus lock prefix	11110000	2	2	
NOP = No Operation	10010000	3	3	
	(TTT LLL are opcode to processor extension)			

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers. For word operations, add 4 clock cycles for all memory transfers.

FOOTNOTES

The Effective Address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

EA calculation time is 4 clock cycles for all modes, and is included in the execution times given whenever appropriate.

Segment Override Prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

ERRATA

An 80C186EB/80L186EB with a STEPID value of 0001H has the following known errata. A device with a STEPID of 0001H can be visually identified by the **presence** of an **"A"** alpha character next to the FPO number. The FPO number location is shown in Figures 4, 5 and 6.

1. A19/$\overline{\text{ONCE}}$ is not latched by the rising edge of $\overline{\text{RESIN}}$. A19/$\overline{\text{ONCE}}$ must remain active (LOW) at all times to remain in the ONCE Mode. Removing A19/$\overline{\text{ONCE}}$ after $\overline{\text{RESIN}}$ is high will return all output pins to a driving state, however, the 80C186EB will remain in a reset state.
2. During interrupt acknowledge (INTA) bus cycles, the bus controller will ignore the state of the READY pin if the previous bus cycle ignored the state of the READY pin. This errata can only occur if the Chip-Select Unit is being used. All active chip-selects must be programmed to use READY (RDY bit must be programmed to a 1) if wait-states are required for INTA bus cycles.
3. CLKOUT will transition off the **rising** edge of CLKIN rather than the falling edge of CLKIN. This does not affect any bus timings other than T_{CD}.
4. $\overline{\text{RESIN}}$ has a hysterisis of only 130 mV. It is recommended that $\overline{\text{RESIN}}$ be driven by a Schmitt triggered device to avoid processor lockup during reset using an RC circuit.
5. SINT1 will only go active for one clock period when a receive or transmit interrupt is pending (i.e., it does not remain active until the S1STS register is read). If SINT1 is to be connected to any of the processor interrupt lines (INT0–INT4), then it must be latched by user logic.

An 80C186EB/80L186EB with a STEPID value of 0001H or 0002H has the following known errata. A device with a STEPID of 0002H can be visually identified by noting the presence of a "B", "C", "D", or "E" alpha character next to the FPO number. The FPO number location is shown in Figures 4, 5 and 6.

1. An internal condition with the interrupt controller can cause no acknowledge cycle on the $\overline{\text{INTA1}}$ line in response to INT1. This errata only occurs when Interrupt 1 is configured in cascade mode and a higher priority interrupt exists. This errata will not occur consistantly, it is dependent on interrupt timing.

REVISION HISTORY

This data sheet replaces the following data sheets:

270803-004	80C186EB
270885-003	80C188EB
270921-003	80L186EB
270920-003	80L188EB
272311-001	SB80C188EB/SB80L188EB
272312-001	SB80C186EB/SB80L186EB

80C186EC/80C188EC AND 80L186EC/80L188EC 16-BIT HIGH-INTEGRATION EMBEDDED PROCESSORS

- Fully Static Operation
- True CMOS Inputs and Outputs

- Integrated Feature Set:
  - Low-Power, Static, Enhanced 8086 CPU Core
  - Two Independent DMA Supported UARTs, each with an Integral Baud Rate Generator
  - Four Independent DMA Channels
  - 22 Multiplexed I/O Port Pins
  - Two 8259A Compatible Programmable Interrupt Controllers
  - Three Programmable 16-Bit Timer/ Counters
  - 32-Bit Watchdog Timer
  - Ten Programmable Chip Selects with Integral Wait-State Generator
  - Memory Refresh Control Unit
  - Power Management Unit
  - On-Chip Oscillator
  - System Level Testing Support (ONCE Mode)
- Direct Addressing Capability to 1 Mbyte Memory and 64 Kbyte I/O
- Low-Power Operating Modes:
  - Idle Mode Freezes CPU Clocks but Keeps Peripherals Active
  - Powerdown Mode Freezes All Internal Clocks
  - Powersave Mode Divides All Clocks by Programmable Prescalar
- Available in Extended Temperature Range (−40°C to +85°C)
- Supports 80C187 Numerics Processor Extension (80C186EC only)
- Package Types:
  - 100-Pin EIAJ Quad Flat Pack (QFP)
  - 100-Pin Plastic Quad Flat Pack (PQFP)
  - 100-Pin Shrink Quad Flat Pack (SQFP)
- Speed Versions Available (5V):
  - 25 MHz (80C186EC25/80C188EC25)
  - 20 MHz (80C186EC20/80C188EC20)
  - 13 MHz (80C186EC13/80C188EC13)
- Speed Version Available (3V):
  - 13 MHz (80L186EC13/80L188EC13)
  - 8 MHz (80L186EC8/80L188EC8)

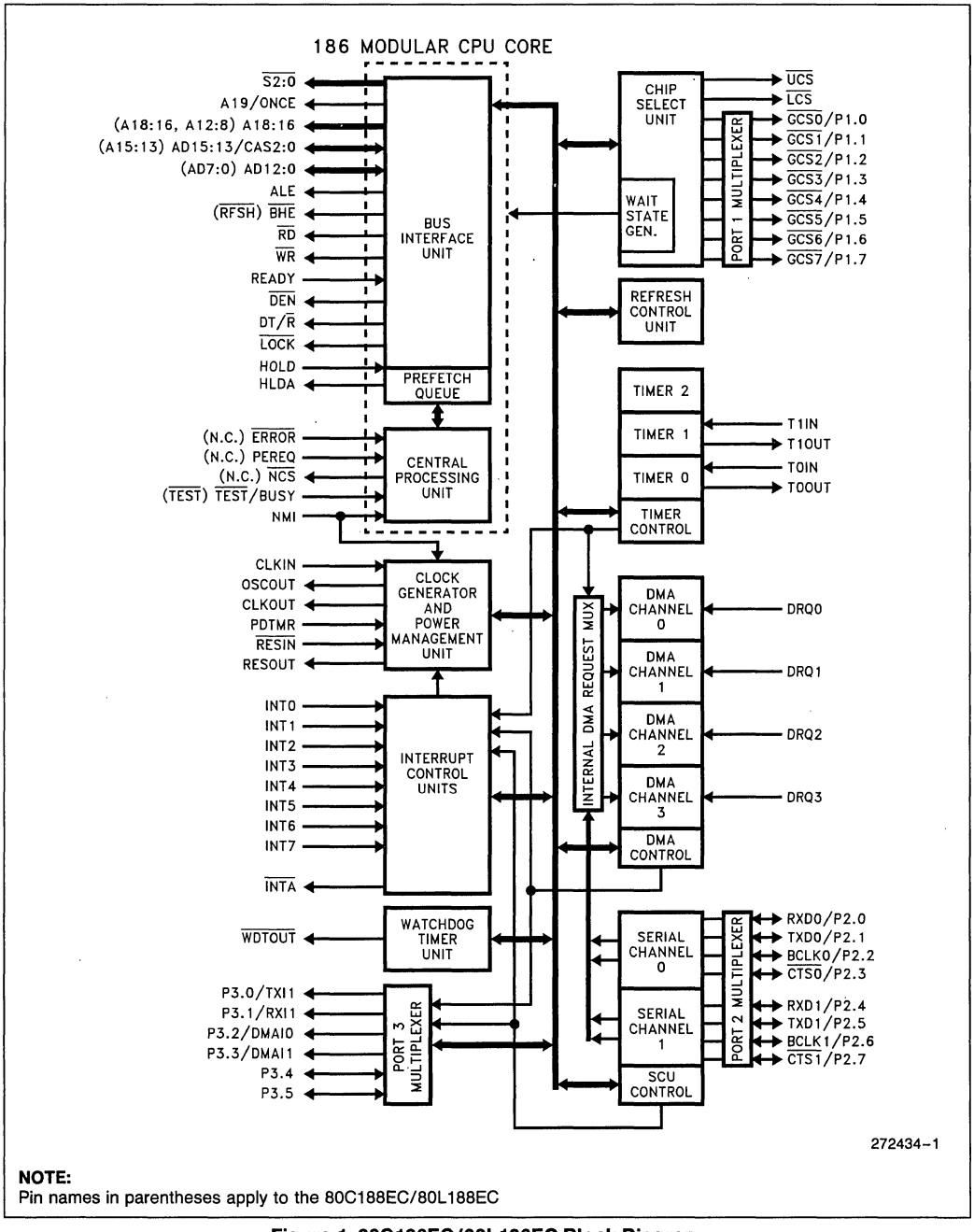
The 80C186EC is a member of the 186 Integrated Processor Family. The 186 Integrated Processor Family incorporates several different VLSI devices all of which share a common CPU architecture: the 8086/8088. The 80C186EC uses the latest high density CHMOS technology to integrate several of the most common system peripherals with an enhanced 8086 CPU core to create a powerful system on a single monolithic silicon die.

January 1995
Order Number: 272434-003

80C186EC/80C188EC and 80L186EC/80L188EC 16-Bit High-Integration Embedded Processor

CONTENTS

CONTENTS



NOTE:
Pin names in parentheses apply to the 80C188EC/80L188EC

Figure 1. 80C186EC/80L186EC Block Diagram

INTRODUCTION

Unless specifically noted, all references to the 80C186EC apply to the 80C188EC, 80L186EC, and 80L188EC. References to pins that differ between the 80C186EC/80L186EC and the 80C188EC/80L188EC are given in parentheses. The "L" in the part number denotes low voltage operation. Physically and functionally, the "C" and "L" devices are identical.

The 80C186EC is one of the highest integration members of the 186 Integrated Processor Family. Two serial ports are provided for services such as interprocessor communication, diagnostics and modem interfacing. Four DMA channels allow for high speed data movement as well as support of the onboard serial ports. A flexible chip select unit simplifies memory and peripheral interfacing. The three general purpose timer/counters can be used for a variety of time measurement and waveform generation tasks. A watchdog timer is provided to insure system integrity even in the most hostile of environments. Two 8259A compatible interrupt controllers handle internal interrupts, and, up to 57 external interrupt requests. A DRAM refresh unit and 24 multiplexed I/O ports round out the feature set of the 80C186EC.

The future set of the 80C186EC meets the needs of low-power, space-critical applications. Low-power applications benefit from the static design of the CPU and the integrated peripherals as well as low voltage operation. Minimum current consumption is achieved by providing a powerdown mode that halts operaton of the device and freezes the clock circuits. Peripheral design enhancements ensure that non-initialized peripherals consume little current.

The 80L186EC is the 3V version of the 80C186EC. The 80L186EC is functionally identical to the 80C186EC embedded processor. Current 80C186EC users can easily upgrade their designs to use the 80L186EC and benefit from the reduced power consumption inherent in 3V operation.

Figure 1 shows a block diagram of the 80C186EC/80C188EC. The execution unit (EU) is an enhanced 8086 CPU core that includes: dedicated hardware to speed up effective address calculations, enhanced execution speed for multiple-bit shift and rotate instructions and for multiply and divide instructions, string move instructions that operate at full bus bandwidth, ten new instructions and fully static operation. The bus interface unit (BIU) is the same as that found on the original 186 family products, except the queue-status mode has been deleted and buffer interface control has been changed to ease system design timings. An independent internal bus is used for communication between the BIU and on-chip peripherals.

80C186EC CORE ARCHITECTURE

Bus Interface Unit

The 80C186EC core incorporates a bus controller that generates local bus control signals. In addition, it employs a HOLD/HLDA protocol to share the local bus with other bus masters.

The bus controller is responsible for generating 20 bits of address, read and write strobes, bus cycle status information and data (for write operations) information. It is also responsible for reading data from the local bus during a read operation. A ready input pin is provided to extend a bus cycle beyond the minimum four states (clocks).

The bus controller also generates two control signals ($\overline{DEN}$ and DT/$\overline{R}$) when interfacing to external transceiver chips. This capability allows the addition of transceivers for simple buffering of the multiplexed address/data bus.

Clock Generator

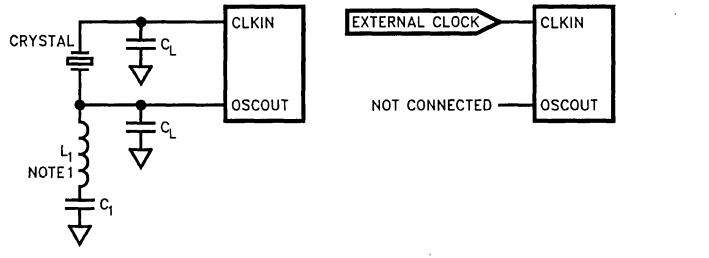
The 80C186EC provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter and three low-power operating modes.

The oscillator circuit is designed to be used with either a parallel resonant fundamental or third-overtone mode crystal network. Alternatively, the oscillator circuit may be driven from an external clock source. Figure 2 shows the various operating modes of the oscillator circuit.

The crystal or clock frequency chosen must be twice the required processor operating frequency due to the internal divide-by-two counter. This counter is used to drive all internal phase clocks and the external CLKOUT signal. CLKOUT is a 50% duty cycle processor clock and can be used to drive other system components. All AC timings are referenced to CLKOUT.

The following parameters are recommended when choosing a crystal:

Temperature Range:	Application Specific
ESR (Equivalent Series Res.):	40Ω max
C0 (Shunt Capacitance of Crystal):	7.0 pF max
C_L (Load Capacitance):	20 pF ±2 pF
Drive Level:	1 mW (max)



NOTE:
1. The LC network is only required when using a third overtone crystal.

Figure 2. 80C186EC Clock Connections

80C186EC PERIPHERAL ARCHITECTURE

The 80C186EC integrates several common system peripherals with a CPU core to create a compact, yet powerful system. The integrated peripherals are designed to be flexbile and provide logical interconnections between supporting units (e.g., the DMA unit can accept requests from the Serial Communications Unit).

The list of integrated peripherals includes:

— Two cascaded, 8259A compatible, Programmable Interrupt Controllers
— 3-Channel Timer/Counter Unit
— 2-Channel Serial Communications Unit
— 4-Channel DMA Unit
— 10-Output Chip-Select Unit
— 32-bit Watchdog Timer Unit
— I/O Port Unit
— Refresh Control Unit
— Power Management Unit

The registers associated with each integrated peripheral are contained within a 128 x 16-bit register file called the Peripheral Control Block (PCB). The base address of the PCB is programmable and can be located on any 256 byte address boundary in either memory or I/O space.

Figure 3 provides a list of the registers associated with the PCB. The Register Bit Summary individually lists all of the registers and identifies each of their programming attributes.

PCB Offset	Function
00H	Master PIC Port 0
02H	Master PIC Port 1
04H	Slave PIC Port 0
06H	Slave PIC Port 1
08H	Reserved
0AH	SCU Int. Req. Ltch.
0CH	DMA Int. Req. Ltch.
0EH	TCU Int. Req. Ltch.
10H	Reserved
12H	Reserved
14H	Reserved
16H	Reserved
18H	Reserved
1AH	Reserved
1CH	Reserved
1EH	Reserved
20H	WDT Reload High
22H	WDT Reload Low
24H	WDT Count High
26H	WDT Count Low
28H	WDT Clear
2AH	WDT Disable
2CH	Reserved
2EH	Reserved
30H	T0 Count
32H	T0 Compare A
34H	T0 Compare B
46H	T0 Control
38H	T1 Count
3AH	T1 Compare A
3CH	T1 Compare B
3EH	T1 Control

PCB Offset	Function
40H	T2 Count
42H	T2 Compare
44H	Reserved
46H	T2 Control
48H	Port 3 Direction
4AH	Port 3 Pin State
4CH	Port 3 Mux Control
4EH	Port 3 Data Latch
50H	Port 1 Direction
52H	Port 1 Pin State
54H	Port 1 Mux Control
56H	Port 1 Data Latch
58H	Port 2 Direction
5AH	Port 2 Pin State
5CH	Port 2 Mux Control
5EH	Port 2 Data Latch
60H	SCU 0 Baud
62H	SCU 0 Count
64H	SCU 0 Control
66H	SCU 0 Status
68H	SCU 0 RBUF
6AH	SCU 0 TBUF
6CH	Reserved
6EH	Reserved
70H	SCU 1 Baud
72H	SCU 1 Count
74H	SCU 1 Control
76H	SCU 1 Status
78H	SCU 1 RBUF
7AH	SCU 1 TBUF
7CH	Reserved
7EH	Reserved

PCB Offset	Function
80H	GCS0 Start
82H	GCS0 Stop
84H	GCS1 Start
86H	GCS1 Stop
88H	GCS2 Start
8AH	GCS2 Stop
8CH	GCS3 Start
8EH	GCS3 Stop
90H	GCS4 Start
92H	GCS4 Stop
94H	GCS5 Start
96H	GCS5 Stop
98H	GCS6 Start
9AH	GCS6 Stop
9CH	GCS7 Start
9EH	GCS7 Stop
A0H	LCS Start
A2H	LCS Stop
A4H	UCS Start
A6H	UCS Stop
A8H	Relocation Register
AAH	Reserved
ACH	Reserved
AEH	Reserved
B0H	Refresh Base Addr.
B2H	Refresh Time
B4H	Refresh Control
B6H	Refresh Address
B8H	Power Control
BAH	Reserved
BCH	Step ID
BEH	Powersave

PCB Offset	Function
C0H	DMA 0 Source Low
C2H	DMA 0 Source High
C4H	DMA 0 Dest. Low
C6H	DMA 0 Dest. High
C8H	DMA 0 Count
CAH	DMA 0 Control
CCH	DMA Module Pri.
CEH	DMA Halt
D0H	DMA 1 Source Low
D2H	DMA 1 Source High
D4H	DMA 1 Dest. Low
D6H	DMA 1 Dest. High
D8H	DMA 1 Count
DAH	DMA 1 Control
DCH	Reserved
DEH	Reserved
E0H	DMA 2 Source Low
E2H	DMA 2 Source High
E4H	DMA 2 Dest. Low
E6H	DMA 2 Dest. High
E8H	DMA 2 Count
EAH	DMA 2 Control
ECH	Reserved
EEH	Reserved
F0H	DMA 3 Source Low
F2H	DMA 3 Source High
F4H	DMA 3 Dest. Low
F6H	DMA 3 Dest. High
F8H	DMA 3 Count
FAH	DMA 3 Control
FCH	Reserved
FEH	Reserved

Figure 3. Peripheral Control Block Registers

Programmable Interrupt Controllers

The 80C186EC utilizes two 8259A compatible Programmable Interrupt Controllers (PIC) to manage both internal and external interrupts. The 8259A modules are configured in a master/slave arrangement.

Seven of the external interrupt pins, INT0 through INT6, are connected to the master 8259A module. The eighth external interrupt pin, INT7, is connected to the slave 8259A module.

There are a total of 11 internal interrupt sources from the integrated peripherals: 4 Serial, 4 DMA and 3 Timer/Counter.

Timer/Counter Unit

The 80C186EC Timer/Counter Unit (TCU) provides three 16-bit programmable timers. Two of these are highly flexible and are connected to external pins for external control or clocking. The third timer is not connected to any external pins and can only be clocked internally. However, it can be used to clock the other two timer channels. The TCU can be used to count external events, time external events, generate non-repetitive waveforms or generate timed interrupts.

Serial Communications Unit

The 80C186EC Serial Communications Unit (SCU) contains two independent channels. Each channel is identical in operation except that only channel 0 is directly supported by the integrated interrupt controller (the channel 1 interrupts are routed to external interrupt pins). Each channel has its own baud rate generator and can be internally or externally clocked up to one half the processor operating frequency. Both serial channels can request service from the DMA unit thus providing block reception and transmission without CPU intervention.

Independent baud rate generators are provided for each of the serial channels. For the asynchronous modes, the generator supplies an 8x baud clock to both the receive and transmit shifting register logic. A 1x baud clock is provided in the synchronous mode.

DMA Unit

The four channel Direct Memory Access (DMA) Unit is comprised of two modules with two channels each. All four channels are identical in operation. DMA transfers can take place from memory to memory, I/O to memory, memory to I/O or I/O to I/O. DMA requests can be external (on the DRQ pins), internal (from Timer 2 or a serial channel) or software initiated.

The DMA Unit transfers data as bytes only. Each data transfer requires at least two bus cycles, one to fetch data and one to deposit. The minimum clock count for each transfer is 8, but this will vary depending on synchronization and wait states.

Chip-Select Unit

The 80C186EC Chip-Select Unit (CSU) integrates logic which provides up to ten programmable chip-selects to access both memories and peripherals. In addition, each chip-select can be programmed to automatically insert additional clocks (wait states) into the current bus cycle, and/or automatically terminate a bus cycle independent of the condition of the READY input pin.

I/O Port Unit

The I/O Port Unit on the 80C186EC supports two 8-bit channels and one 6-bit channel of input, output or input/output operation. Port 1 is multiplexed with the chip select pins and is output only. Port 2 is multiplexed with the pins for serial channels 1 and 2. All Port 2 pins are input/output. Port 3 has a total of 6 pins: four that are multiplexed with DMA and serial port interrupts and two that are non-multiplexed, open drain I/O.

Refresh Control Unit

The Refresh Control Unit (RCU) automatically generates a periodic memory read bus cycle to keep dynamic or pseudo-static memory refreshed. A 9-bit counter controls the number of clocks between refresh requests.

A 12-bit address generator is maintained by the RCU and is presented on the A12:1 address lines during the refresh bus cycle. Address bits A19:13 are programmable to allow the refresh address block to be located on any 8 Kbyte boundary.

Watchdog Timer Unit

The Watchdog Timer Unit (WDT) allows for graceful recovery from unexpected hardware and software upsets. The WDT consists of a 32-bit counter that decrements every clock cycle. If the counter reaches zero before being reset, the $\overline{\text{WDTOUT}}$ pin is

pulled low for four clock cycles. Logically ANDing the WDTOUT pin with the power-on reset signal allows the WDT to reset the device in the event of a WDT timeout. If a less drastic method of recovery is desired, WDTOUT can be connected directly to NMI or one of the INT input pins. The WDT may also be used as a general purpose timer.

Power Management Unit

The 80C186EC Power Management Unit (PMU) is provided to control the power consumption of the device. The PMU provides four power management modes: Active, Powersave, Idle and Powerdown.

Active Mode indicates that all units on the 80C186EC are operating at ½ the CLKIN frequency.

Idle Mode freezes the clocks of the Execution and Bus units at a logic zero state (all peripherals continue to operate normally).

The Powerdown Mode freezes all internal clocks at a logic zero level and disables the crystal oscillator.

In Powersave Mode, all internal clock signals are divided by a programmable prescalar (up to 1/64 the normal frequency). Powersave Mode can be used with Idle Mode as well as during normal (Active Mode) operation.

80C187 Interface (80C186EC only)

The 80C186EC supports the direct connection of the 80C187 Numerics Processor Extension. The 80C187 can dramatically improve the performance of calculation intensive applications.

ONCE Test Mode

To facilitate testing and inspection of devices when fixed into a target system, the 80C186EC has a test mode available which forces all output and input/output pins to be placed in the high-impedance state. ONCE stands for "ON Circuit Emulation". The ONCE mode is selected by forcing the A19/S6/ONCE pin low during a processor reset (this pin is weakly held high during reset to prevent inadvertant entrance into ONCE Mode).

PACKAGE INFORMATION

This section describes the pin functions, pinout and thermal characteristics for the 80C186EC in the Plastic Quad Flat Pack (JEDEC PQFP), the EIAJ Quad Flat Pack (QFP) and the Shrink Quad Flat Pack (SQFP). For complete package specifications and information, see the Intel Packaging Outlines and Dimensions Guide (Order Number: 231369).

Prefix Identification

Table 1 lists the prefix identifications.

Table 1. Prefix Identification

Prefix	Note	Package Type	Temperature Range
TS		QFP (EIAJ)	Extended
KU	1	PQFP	Extended/Commercial
SB	1	SQFP	Extended/Commercial
S	1	QFP (EIAJ)	Commercial

NOTE:
1. The 5V 25 MHz version is only available in commercial temperature range corresponding to 0°C to +70°C ambient.

Pin Descriptions

Each pin or logical set of pins is described in Table 2. There are four columns for each entry in the Pin Description Table. The following sections describe each column.

Column 1: Pin Name
In this column is a mnemonic that describes the pin function. Negation of the signal name (i.e. RESIN) implies that the signal is active low.

Column 2: Pin Type
A pin may be either power (P), ground (G), input only (I), output only (O) or input/output (I/O). Please note that some pins have more than 1 function. A19/S6/ONCE, for example, is normally an output but functions as an input during reset. For this reason A19/S6/ONCE is classified as an input/output pin.

Column 3: Input Type (for I and I/O types only)
There are two different types of input pins on the 80C186EC: asynchronous and synchronous. **Asynchronous** pins require that setup and hold times be met only to *guarantee recognition*. **Synchronous** input pins require that the setup and hold times be met to *guarantee proper operation*. Stated simply, missing a setup or hold on an asynchronous pin will result in something minor (i.e. a timer count will be missed) whereas missing a setup or hold on a synchronous pin will result in system failure (the system will "lock up").

An input pin may also be edge or level sensitive.

Column 4: Output States (for O and I/O types only)

The state of an output or I/O pin is dependent on the operating mode of the device. There are four modes of operation that are different from normal active mode: Bus Hold, Reset, Idle Mode, Powerdown Mode. This column describes the output pin state in each of these modes.

The legend for interpreting the information in the Pin Descriptions is shown in Table 1.

As an example, please refer to the table entry for AD12:0. The "I/O" signifies that the pins are bidirectional (i.e. have both an input and output function). The "S" indicates that, as an input the signal must be synchronized to CLKOUT for proper operation. The "H(Z)" indicates that these pins will float while the processor is in the Hold Acknowledge state. R(Z) indicates that these pins will float while $\overline{RESIN}$ is low. P(0) and I(0) indicate that these pins will drive 0 when the device is in either Powerdown or Idle Mode.

Some pins, the I/O Ports for example, can be programmed to perform more than one function. Multifunction pins have a "/" in their signal name between the different functions (i.e. P3.0/RXI1). If the input pin type or output pin state differ between functions, then that will be indicated by separating the state (or type) with a "/" (i.e. H(X)/H(Q)). In this example when the pin is configured as P3.0 then its hold output state is H(X); when configured as RXI1 its output state is H(Q).

All pins float while the processor is in the ONCE Mode (with the exception of OSCOUT).

Table 1. Pin Description Nomenclature

Symbol	Description
P	Power Pin (apply + V_{CC} voltage)
G	Ground (connect to V_{SS})
I	Input only pin
O	Output only pin
I/O	Input/Output pin
S(E)	Synchronous, edge sensitive
S(L)	Synchronous, level sensitive
A(E)	Asynchronous, edge sensitive
A(L)	Asynchronous, level sensitive
H(1)	Output driven to V_{CC} during bus hold
H(0)	Output driven to V_{SS} during bus hold
H(Z)	Output floats during bus hold
H(Q)	Output remains active during bus hold
H(X)	Output retains current state during bus hold
R(WH)	Output weakly held at V_{CC} during reset
R(1)	Output driven to V_{CC} during reset
R(0)	Output driven to V_{SS} during reset
R(Z)	Output floats during reset
R(Q)	Output remains active during reset
R(X)	Output retains current state during reset
I(1)	Output driven to V_{CC} during Idle Mode
I(0)	Output driven to V_{SS} during Idle Mode
I(Z)	Output floats during Idle Mode
I(Q)	Output remains active during Idle Mode
I(X)	Output retains current state during Idle Mode
P(1)	Output driven to V_{CC} during Powerdown Mode
P(0)	Output driven to V_{SS} during Powerdown Mode
P(Z)	Output floats during Powerdown Mode
P(Q)	Output remains active during Powerdown Mode
P(X)	Output retains current state during Powerdown Mode

Table 2. Pin Descriptions

Pin Name	Pin Type	Input Type	Output States	Pin Description
V_{CC}	P	—	—	**POWER** +5V ±10% power supply connection
V_{SS}	G	—	—	**GROUND**
CLKIN	I	A(E)	—	**CLocK INput** is the external clock input. An external oscillator operating at two times the required processor operating frequency can be connected to CLKIN. For crystal operation, CLKIN (along with OSCOUT) are the crystal connections to an internal Pierce oscillator.
OSCOUT	O	—	H(Q) R(Q) I(Q) P(X)	**OSCillator OUTput** is only used when using a crystal to generate the internal clock. OSCOUT (along with CLKIN) are the crystal connections to an internal Pierce oscillator. This pin can not be used as 2X clock output for non-crystal applications (i.e. this pin is not connected for non-crystal applications).
CLKOUT	O	—	H(Q) R(Q) I(Q) P(X)	**CLocK OUTput** provides a timing reference for inputs and outputs of the processor, and is one-half the input clock (CLKIN) frequency. CLKOUT has a 50% duty cycle and transitions every falling edge of CLKIN.
$\overline{\text{RESIN}}$	I	A(L)	—	**RESet IN** causes the processor to immediately terminate any bus cycle in progress and assume an initialized state. All pins will be driven to a known state, and RESOUT will also be driven active. The rising edge (low-to-high) transition synchronizes CLKOUT with CLKIN before the processor begins fetching opcodes at memory location 0FFFF0H.
RESOUT	O	—	H(0) R(1) I(0) P(0)	**RESet OUTput** that indicates the processor is currently in the reset state. RESOUT will remain active as long as $\overline{\text{RESIN}}$ remains active.
PDTMR	I/O	A(L)	H(WH) R(Z) P(WH) I(WH)	**Power-Down TiMeR** pin (normally connected to an external capacitor) that determines the amount of time the processors waits after an exit from Powerdown before resuming normal operation. The duration of time required will depend on the startup characteristics of the crystal oscillator.
NMI	I	A(E)	—	**Non-Maskable Interrupt** input causes a TYPE-2 interrupt to be serviced by the CPU. NMI is latched internally.
$\overline{\text{TEST}}$/BUSY (TEST)	I	A(E)	—	**TEST** is used during the execution of the WAIT instruction to suspend CPU operation until the pin is sampled active (LOW). $\overline{\text{TEST}}$ is alternately known as **BUSY** when interfacing with an 80C187 numerics coprocessor (80C186EC only).
A19/S6/$\overline{\text{ONCE}}$	I/O	A(L)	H(Z) R(WH) I(0) P(0)	This pin drives address bit 19 during the address phase of the bus cycle. During T2 and T3 this pin functions as status bit 6. S6 is low to indicate CPU bus cycles and high to indicate DMA or refresh bus cycles. During a processor reset ($\overline{\text{RESIN}}$ active) this pin becomes the ONCE input pin. Holding this pin low during reset will force the part into ONCE Mode.

NOTE:
Pin names in parentheses apply to the 80C188EC/80L188EC.

Table 2. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
A18/S5 A17/S4 A16/S3 (A15:8)	I/O	A(L)	H(Z) R(WH) I(0) P(0)	These pins drive address information during the address phase of the bus cycle. During T2 and T3 these pins drive status information (which is always 0 on the 80C186EC). These pins are used as inputs during factory test; driving these pins low during reset will cause unspecified operation. On the 80C188EC, A15:8 provide valid address information for the entire bus cycle.
AD15/CAS2 AD14/CAS1 AD13/CAS0	I/O	S(L)	H(Z) R(Z) I(0) P(0)	These pins are part of the multiplexed ADDRESS and DATA bus. During the address phase of the bus cycle, address bits 15 through 13 are presented on these pins and can be latched using ALE. Data information is transferred during the data phase of the bus cycle. Pins AD15:13/CAS2:0 drive the 82C59 slave address information during interrupt acknowledge cycles.
AD12:0 (AD7:0)	I/O	S(L)	H(Z) R(Z) I(0) P(0)	These pins provide a multiplexed ADDRESS and DATA bus. During the address phase of the bus cycle, address bits 0 through 12 (0 through 7 on the 80C188EC) are presented on the bus and can be latched using ALE. Data information is transferred during the data phase of the bus cycle.
$\overline{S2:0}$	O	—	H(Z) R(1) I(1) P(1)	Bus cycle Status are encoded on these pins to provide bus transaction information. $\overline{S2:0}$ are encoded as follows:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Processor HALT
1	0	0	Instruction Queue Fetch
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive (No bus activity)

ALE	O	—	H(0) R(0) I(0) P(0)	**Address Latch Enable** output is used to strobe address information into a transparent type latch during the address phase of the bus cycle.
$\overline{BHE}$ ($\overline{RFSH}$)	O	—	H(Z) R(Z) I(1) P(1)	**Byte High Enable** output to indicate that the bus cycle in progress is transferring data over the upper half of the data bus. $\overline{BHE}$ and A0 have the following logical encoding:

A0	$\overline{BHE}$	**Encoding** (for 80C186EC/80L186EC only)
0	0	Word transfer
0	1	Even Byte transfer
1	0	Odd Byte transfer
1	1	Refresh operation

On the 80C188EC/80L188EC, $\overline{RFSH}$ is asserted low to indicate a refresh bus cycle.

NOTE:
Pin names in parentheses apply to the 80C188EC/80L188EC.

Table 2. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
$\overline{RD}$	O	—	H(Z) R(Z) I(1) P(1)	**ReaD** output signals that the accessed memory or I/O device should drive data information onto the data bus.
$\overline{WR}$	O	—	H(Z) R(Z) I(1) P(1)	**WRite** output signals that data available on the data bus are to be written into the accessed memory or I/O device.
READY	I	A(L) S(L) (Note 1)	—	**READY** input to signal the completion of a bus cycle. READY must be active to terminate any 80C186EC bus cycle, unless it is ignored by correctly programming the Chip-Select unit.
$\overline{DEN}$	O	—	H(Z) R(Z) I(1) P(1)	**Data ENable** output to control the enable of bi-directional transceivers in a buffered system. $\overline{DEN}$ is active only when data is to be transferred on the bus.
DT/$\overline{R}$	O	—	H(Z) R(Z) I(X) P(X)	**Data Transmit/Receive** output controls the direction of a bi-directional buffer in a buffered system.
$\overline{LOCK}$	I/O	A(L)	H(Z) R(Z) I(X) P(X)	**LOCK** output indicates that the bus cycle in progress is not interruptable. The processor will not service other bus requests (such as HOLD) while $\overline{LOCK}$ is active. This pin is configured as a weakly held high input while $\overline{RESIN}$ is active and must not be driven low.
HOLD	I	A(L)	—	**HOLD** request input to signal that an external bus master wishes to gain control of the local bus. The processor will relinquish control of the local bus between instruction boundaries that are not LOCKed.
HLDA	O	—	H(1) R(0) I(0) P(0)	**HoLD Acknowledge** output to indicate that the processor has relinquished control of the local bus. When HLDA is asserted, the processor will (or has) floated its data bus and control signals allowing another bus master to drive the signals directly.
$\overline{NCS}$	O	—	H(1) R(1) I(1) P(1)	**Numerics Coprocessor Select** output is generated when acessing a numerics coprocessor. This signal does not exist on the 80C188EC/80L188EC.
$\overline{ERROR}$	I	A(L)	—	**ERROR** input that indicates the last numerics processor extension operation resulted in an exception condition. An interrupt TYPE 16 is generated if $\overline{ERROR}$ is sampled active at the beginning of a numerics operation. Systems not using an 80C187 must tie $\overline{ERROR}$ to V_{CC}. This signal does not exist on the 80C188EC/80L188EC.

NOTE:
Pin names in parentheses apply to the 80C188EC/80L188EC.

Table 2. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
PEREQ	I	A(L)	—	**Processor Extension REQuest** signals that a data transfer between an 80C187 Numerics Processor Extension and Memory is pending. Systems not using an 80C187 must tie this pin to V_{SS}. This signal does not exist on the 80C188EC/80L188EC.
$\overline{UCS}$	O	—	H(1) R(1) I(1) P(1)	**Upper Chip Select** will go active whenever the address of a memory or I/O bus cycle is within the address range programmed by the user. After reset, $\overline{UCS}$ is configured to be active for memory accesses between 0FFC00H and 0FFFFFH.
$\overline{LCS}$	O	—	H(1) R(1) I(1) P(1)	**Lower Chip Select** will go active whenever the address of a memory or I/O bus cycle is within the address range programmed by the user. $\overline{LCS}$ is inactive after a reset.
P1.0/$\overline{GCS0}$ P1.1/$\overline{GCS1}$ P1.2/$\overline{GCS2}$ P1.3/$\overline{GCS3}$ P1.4/$\overline{GCS4}$ P1.5/$\overline{GCS5}$ P1.6/$\overline{GCS6}$ P1.7/$\overline{GCS7}$	O	—	H(X)/H(1) R(1) I(X)/I(1) P(X)/P(1)	These pins provide a multiplexed function. If enabled, each pin can provide a **General purpose Chip Select** output which will go active whenever the address of a memory or I/O bus cycle is within the address limitations programmed by the user. When not programmed as a Chip-Select, each pin may be used as a general purpose output port.
T0OUT T1OUT	O	—	H(Q) R(1) I(Q) P(X)	**Timer OUTput** pins can be programmed to provide single clock or continuous waveform generation, depending on the timer mode selected.
T0IN T1IN	I	A(L) A(E)	—	**Timer INput** is used either as clock or control signals, depending on the timer mode selected. This pin may be either level or edge sensitive depending on the programming mode.
INT7:0	I	A(L) A(E)	—	**Maskable INTerrupt** input will cause a vector to a specific Type interrupt routine. The INT6:0 pins can be used as cascade inputs from slave 8259A devices. The INT pins can be configured as level or edge sensitive.
$\overline{INTA}$	O	—	H(1) R(1) I(1) P(1)	**INTerrupt Acknowledge** output is a handshaking signal used by external 82C59A Programmable Interrupt Controllers.
P3.5 P3.4	I/O	A(L)	H(X) R(Z) I(X) H(X)	Bidirectional, open-drain port pins.
P3.3/DMAI1 P3.2/DMAI0	O	—	H(X) R(0) I(Q) P(X)	**DMA Interrupt** output goes active to indicate that the channel has completed a transfer. DMAI1 and DMAI0 are multiplexed with output only port functions.

NOTE:
Pin names in parentheses apply to the 80C188EC/80L188EC.

Table 2. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
P3.1/TXI1	O	—	H(X)/H(Q) R(0) I(Q) P(X)	**Transmit Interrupt** output goes active to indicate that serial channel 1 has completed a transfer. TXI1 is multiplexed with an output only Port function.
P3.0/RXI1	O	—	H(X)/H(Q) R(0) I(Q) P(X)	**Receive Interrupt** output goes active to indicate that serial channel 1 has completed a reception. RXI1 is multiplexed with an output only port function.
$\overline{\text{WDTOUT}}$	O	—	H(Q) R(1) I(Q) P(X)	**WatchDog Timer OUTput** is driven low for four clock cycles when the watchdog timer reaches zero. $\overline{\text{WDTOUT}}$ may be ANDed with the power-on reset signal to reset the processor when the watchdog timer is not properly reset.
P2.7/$\overline{\text{CTS1}}$ P2.3/$\overline{\text{CTS0}}$	I/O	A(L)	H(X) R(Z) I(X) P(X)	**Clear-To-Send** input is used to prevent the transmission of serial data on the TXD signal pin. $\overline{\text{CTS1}}$ and $\overline{\text{CTS0}}$ are multiplexed with an I/O Port function.
P2.6/BCLK1 P2.2/BCLK0	I/O	A(L)/ A(E)	H(X) R(Z) I(X) P(X)	**Baud CLocK** input can be used as an alternate clock source for each of the integrated serial channels. The BCLK inputs are multiplexed with I/O Port functions. The BCLK input frequency cannot exceed ½ the operating frequency of the processor .
P2.5/TXD1 P2.1/TXD0	I/O	A(L)	H(Q) R(Z) I(X)/I(Q) P(X)	**Transmit Data** output provides serial data information. The TXD outputs are multiplexed with I/O Port functions. During synchronous serial communications, TXD will function as a clock output.
P2.4/RXD1 P2.0/RXD0	I/O	A(L)	H(X)/H(Q) R(Z) I(X)/I(Q) P(X)	**Receive Data** input accepts serial data information. The RXD pins are multiplexed with I/O Port functions. During synchronous serial communications, RXD is bi-directional and will become an output for transmission of data (TXD becomes the clock).
DRQ3:0	I	A(L)	—	**DMA ReQuest** input pins are used to request a DMA transfer. The timing of the request is dependent on the programmed synchronization mode.

NOTES:
1. READY is A(E) for the rising edge of CLKOUT, S(E) for the falling edge of CLKOUT.
2. Pin names in parentheses apply to the 80C188EC/80L188EC.

Pinout

Tables 3 and 4 list the pin names with package location for the 100-pin Plastic Quad Flat Pack (PQFP) component. Figure 4 depicts the PQFP package as viewed from the top side of the component (i.e. contacts facing down).

Tables 5 and 6 list the pin names with package location for the 100-pin EIAJ Quad Flat Pack (QFP) component. Figure 5 depicts the QFP package as viewed from the top side of the component (i.e. contacts facing down).

Tables 7 and 8 list the pin names with package location for the 100-pin Shrink Quad Flat Pack (SQFP) component. Figure 6 depicts the SQFP package as viewed from the top side of the component (i.e., contacts facing down).

Table 3. PQFP Pin Functions with Location

AD Bus	
Name	**Pin**
AD0	73
AD1	72
AD2	71
AD3	70
AD4	66
AD5	65
AD6	64
AD7	63
AD8 (A8)	60
AD9 (A9)	59
AD10 (A10)	58
AD11 (A11)	57
AD12 (A12)	56
AD13/CAS0 (A13/CAS0)	55
AD14/CAS1 (A14/CAS1)	54
AD15/CAS2 (A15/CAS2)	53
A16/S3	77
A17/S4	76
A18/S5	75
A19/S6/$\overline{\text{ONCE}}$	74

Bus Control	
Name	**Pin**
ALE	52
$\overline{\text{BHE}}$ ($\overline{\text{RFSH}}$)	51
$\overline{\text{S0}}$	78
$\overline{\text{S1}}$	79
$\overline{\text{S2}}$	80
$\overline{\text{RD}}$	50
$\overline{\text{WR}}$	49
READY	85
$\overline{\text{DEN}}$	47
DT/$\overline{\text{R}}$	46
$\overline{\text{LOCK}}$	48
HOLD	44
HLDA	45
$\overline{\text{INTA}}$	34

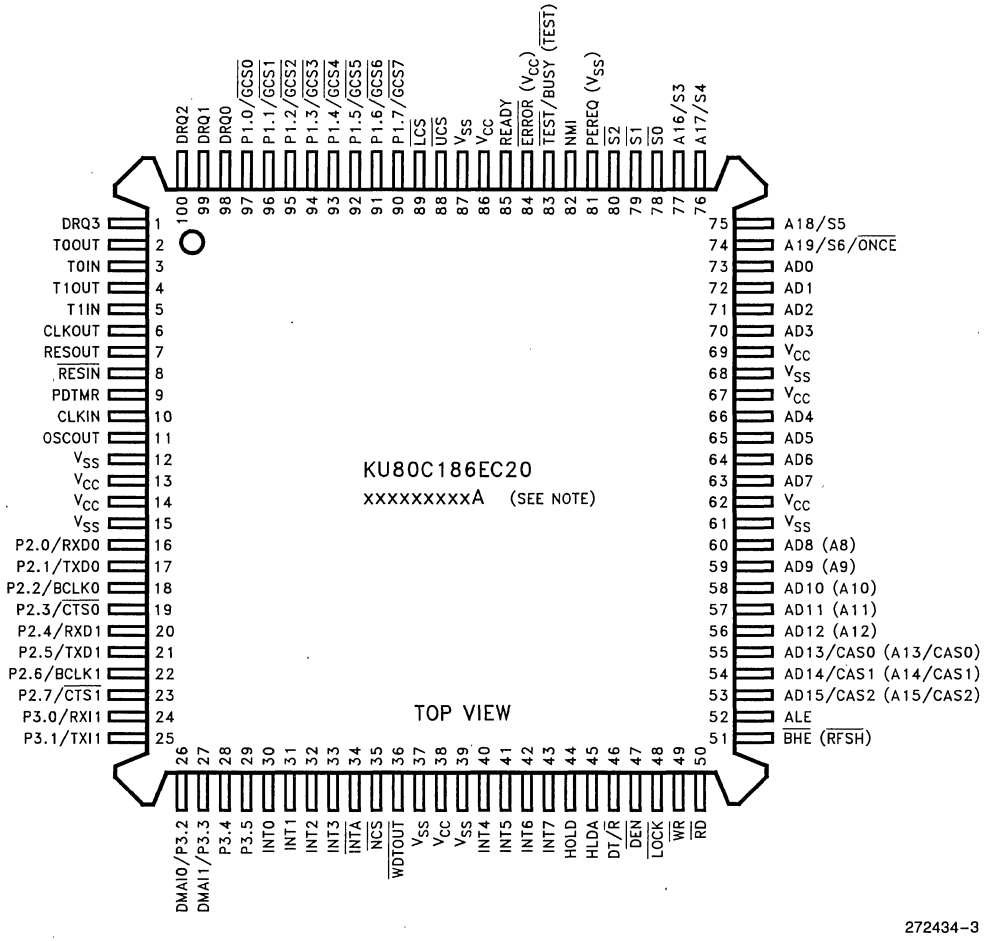
Power and Ground	
Name	**Pin**
V_{CC}	13
V_{CC}	14
V_{CC}	38
V_{CC}	62
V_{CC}	67
V_{CC}	69
V_{CC}	86
V_{SS}	12
V_{SS}	15
V_{SS}	37
V_{SS}	39
V_{SS}	61
V_{SS}	68
V_{SS}	87

Processor Control	
Name	**Pin**
$\overline{\text{RESIN}}$	8
RESOUT	7
CLKIN	10
OSCOUT	11
CLKOUT	6
$\overline{\text{TEST}}$/BUSY ($\overline{\text{TEST}}$)	83
PEREQ (V_{SS})	81
$\overline{\text{NCS}}$ (N.C.)	35
$\overline{\text{ERROR}}$ (V_{CC})	84
PDTMR	9
NMI	82
INT0	30
INT1	31
INT2	32
INT3	33
INT4	40
INT5	41
INT6	42
INT7	43

I/O	
Name	**Pin**
$\overline{\text{UCS}}$	88
$\overline{\text{LCS}}$	89
P1.7/$\overline{\text{GCS7}}$	90
P1.6/$\overline{\text{GCS6}}$	91
P1.5/$\overline{\text{GCS5}}$	92
P1.4/$\overline{\text{GCS4}}$	93
P1.3/$\overline{\text{GCS3}}$	94
P1.2/$\overline{\text{GCS2}}$	95
P1.1/$\overline{\text{GCS1}}$	96
P1.0/$\overline{\text{GCS0}}$	97
P2.7/$\overline{\text{CTS1}}$	23
P2.6/BCLK1	22
P2.5/TXD1	21
P2.4/RXD1	20
P2.3/$\overline{\text{CTS0}}$	19
P2.2/BCLK0	18
P2.1/TXD0	17
P2.0/RXD0	16
P3.5	29
P3.4	28
P3.3/DMAI1	27
P3.2/DMAI0	26
P3.1/TXI1	25
P3.0/RXI1	24
T0IN	3
T0OUT	2
T1IN	5
T1OUT	4
DRQ0	98
DRQ1	99
DRQ2	100
DRQ3	1
$\overline{\text{WDTOUT}}$	36

Table 4. PQFP Pin Locations with Pin Name

Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	DRQ3	26	DMAI0/P3.2	51	$\overline{BHE}$ ($\overline{RFSH}$)	76	A17/S4
2	T0OUT	27	DMAI1/P3.3	52	ALE	77	A16/S3
3	T0IN	28	P3.4	53	AD15 (A15)	78	$\overline{S0}$
4	T1OUT	29	P3.5	54	AD14 (A14)	79	$\overline{S1}$
5	T1IN	30	INT0	55	AD13 (A13)	80	$\overline{S2}$
6	CLKOUT	31	INT1	56	AD12 (A12)	81	PEREQ (V_{SS})
7	RESOUT	32	INT2	57	AD11 (A11)	82	NMI
8	$\overline{RESIN}$	33	INT3	58	AD10 (A10)	83	$\overline{TEST}$
9	PDTMR	34	$\overline{INTA}$	59	AD9 (A9)	84	$\overline{ERROR}$ (V_{CC})
10	CLKIN	35	$\overline{NCS}$ (N.C.)	60	AD8 (A8)	85	READY
11	OSCOUT	36	$\overline{WDTOUT}$	61	V_{SS}	86	V_{CC}
12	V_{SS}	37	V_{SS}	62	V_{CC}	87	V_{SS}
13	V_{CC}	38	V_{CC}	63	AD7	88	$\overline{UCS}$
14	V_{CC}	39	V_{SS}	64	AD6	89	$\overline{LCS}$
15	V_{SS}	40	INT4	65	AD5	90	P1.7/$\overline{GCS7}$
16	P2.0/RXD0	41	INT5	66	AD4	91	P1.6/$\overline{GCS6}$
17	P2.1/TXD0	42	INT6	67	V_{CC}	92	P1.5/$\overline{GCS5}$
18	P2.2/BCLK0	43	INT7	68	V_{SS}	93	P1.4/$\overline{GCS4}$
19	P2.3/$\overline{CTS0}$	44	HOLD	69	V_{CC}	94	P1.3/$\overline{GCS3}$
20	P2.4/RXD1	45	HLDA	70	AD3	95	P1.2/$\overline{GCS2}$
21	P2.5/TXD1	46	DT/$\overline{R}$	71	AD2	96	P1.1/$\overline{GCS1}$
22	P2.6/BCLK1	47	$\overline{DEN}$	72	AD1	97	P1.0/$\overline{GCS0}$
23	P2.7/$\overline{CTS1}$	48	$\overline{LOCK}$	73	AD0	98	DRQ0
24	P3.0/RXI1	49	$\overline{WR}$	74	A19/S6/$\overline{ONCE}$	99	DRQ1
25	P3.1/TXI1	50	$\overline{RD}$	75	A18/S5	100	DRQ2



NOTE:
This is the FPO number location (indicated by X's).

Figure 4. 100-Pin Plastic Quad Flat Pack Package (PQFP)

Table 5. QFP Pin Names with Package Location

AD Bus	
Name	**Pin**
AD0	76
AD1	75
AD2	74
AD3	73
AD4	69
AD5	68
AD6	67
AD7	66
AD8 (A8)	63
AD9 (A9)	62
AD10 (A10)	61
AD11 (A11)	60
AD12 (A12)	59
AD13/CAS0 (A13/CAS0)	58
AD14/CAS1 (A14/CAS1)	57
AD15/CAS2 (A15/CAS2)	56
A16/S3	80
A17/S4	79
A18/S5	78
A19/S6/$\overline{\text{ONCE}}$	77

Bus Control	
Name	**Pin**
ALE	55
$\overline{\text{BHE}}$ ($\overline{\text{RFSH}}$)	54
$\overline{\text{S0}}$	81
$\overline{\text{S1}}$	82
$\overline{\text{S2}}$	83
$\overline{\text{RD}}$	53
$\overline{\text{WR}}$	52
READY	88
$\overline{\text{DEN}}$	50
DT/$\overline{\text{R}}$	49
$\overline{\text{LOCK}}$	51
HOLD	47
HLDA	48
$\overline{\text{INTA}}$	37

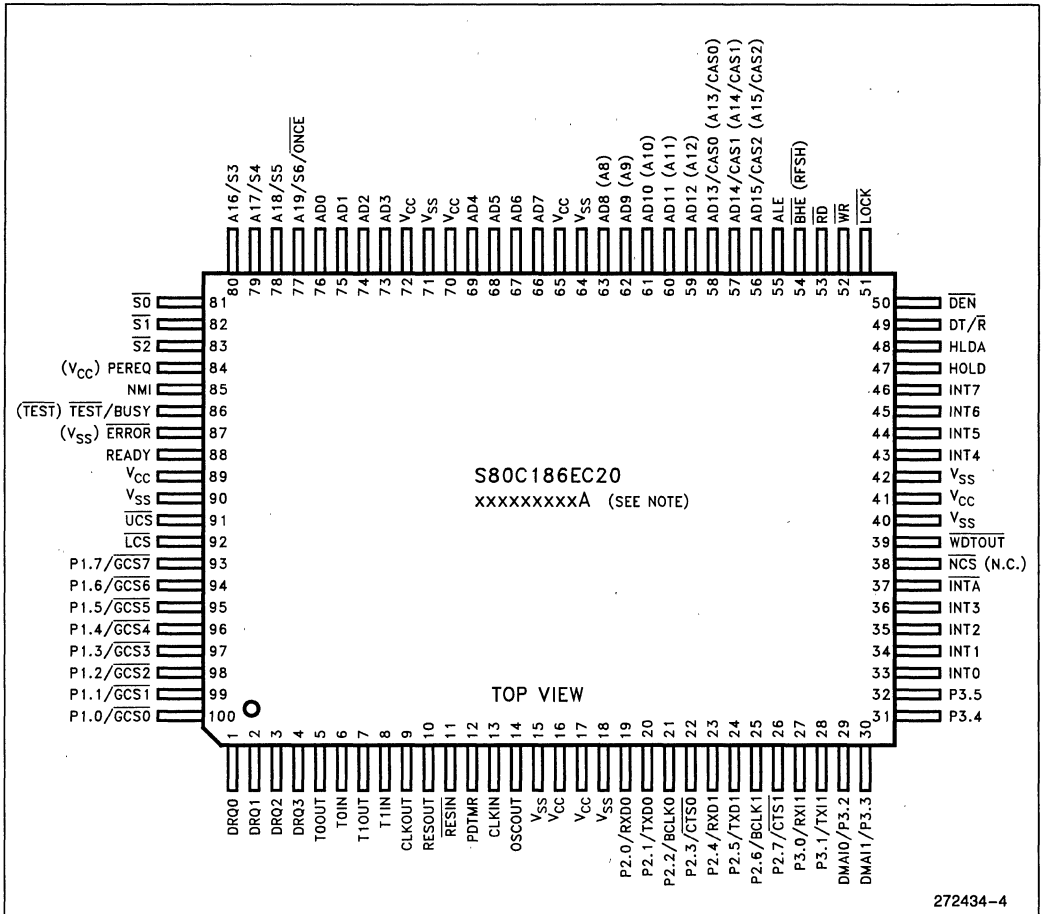
Power and Ground	
Name	**Pin**
V_{CC}	16
V_{CC}	17
V_{CC}	41
V_{CC}	65
V_{CC}	70
V_{CC}	72
V_{CC}	89
V_{SS}	15
V_{SS}	18
V_{SS}	40
V_{SS}	42
V_{SS}	64
V_{SS}	71
V_{SS}	90

Processor Control	
Name	**Pin**
$\overline{\text{RESIN}}$	11
RESOUT	10
CLKIN	13
OSCOUT	14
CLKOUT	9
$\overline{\text{TEST}}$/BUSY ($\overline{\text{TEST}}$)	86
PEREQ (V_{SS})	84
$\overline{\text{NCS}}$ (N.C.)	38
$\overline{\text{ERROR}}$ (V_{CC})	87
PDTMR	12
NMI	85
INT0	33
INT1	34
INT2	35
INT3	36
INT4	43
INT5	44
INT6	45
INT7	46

I/O	
Name	**Pin**
$\overline{\text{UCS}}$	91
$\overline{\text{LCS}}$	92
P1.7/$\overline{\text{GCS7}}$	93
P1.6/$\overline{\text{GCS6}}$	94
P1.5/$\overline{\text{GCS5}}$	95
P1.4/$\overline{\text{GCS4}}$	96
P1.3/$\overline{\text{GCS3}}$	97
P1.2/$\overline{\text{GCS2}}$	98
P1.1/$\overline{\text{GCS1}}$	99
P1.0/$\overline{\text{GCS0}}$	100
P2.7/$\overline{\text{CTS1}}$	26
P2.6/BCLK1	25
P2.5/TXD1	24
P2.4/RXD1	23
P2.3/$\overline{\text{CTS0}}$	22
P2.2/BCLK0	21
P2.1/TXD0	20
P2.0/RXD0	19
P3.5	32
P3.4	31
P3.3/DMAI1	30
P3.2/DMAI0	29
P3.1/TXI1	28
P3.0/RXI1	27
T0IN	6
T0OUT	5
T1IN	8
T1OUT	7
DRQ0	1
DRQ1	2
DRQ2	3
DRQ3	4
$\overline{\text{WDTOUT}}$	39

Table 6. QFP Package Location with Pin Names

Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	DRQ0	26	P2.7/$\overline{CTS1}$	51	$\overline{LOCK}$	76	AD0
2	DRQ1	27	P3.0/RXI1	52	$\overline{WR}$	77	A19/S6/$\overline{ONCE}$
3	DRQ2	28	P3.1/TXI1	53	$\overline{RD}$	78	A18/S5
4	DRQ3	29	DMAI0/P3.2	54	$\overline{BHE}$ ($\overline{RFSH}$)	79	A17/S4
5	T0OUT	30	DMAI1/P3.3	55	ALE	80	A16/S3
6	T0IN	31	P3.4	56	AD15 (A15)	81	$\overline{S0}$
7	T1OUT	32	P3.5	57	AD14 (A14)	82	$\overline{S1}$
8	T1IN	33	INT0	58	AD13 (A13)	83	$\overline{S2}$
9	CLKOUT	34	INT1	59	AD12 (A12)	84	PEREQ (V_{SS})
10	RESOUT	35	INT2	60	AD11 (A11)	85	NMI
11	$\overline{RESIN}$	36	INT3	61	AD10 (A10)	86	$\overline{TEST}$
12	PDTMR	37	$\overline{INTA}$	62	AD9 (A9)	87	$\overline{ERROR}$ (V_{CC})
13	CLKIN	38	$\overline{NCS}$ (N.C.)	63	AD8 (A8)	88	READY
14	OSCOUT	39	$\overline{WDTOUT}$	64	V_{SS}	89	V_{CC}
15	V_{SS}	40	V_{SS}	65	V_{CC}	90	V_{SS}
16	V_{CC}	41	V_{CC}	66	AD7	91	$\overline{UCS}$
17	V_{CC}	42	V_{SS}	67	AD6	92	$\overline{LCS}$
18	V_{SS}	43	INT4	68	AD5	93	P1.7/$\overline{GCS7}$
19	P2.0/RXD0	44	INT5	69	AD4	94	P1.6/$\overline{GCS6}$
20	P2.1/TXD0	45	INT6	70	V_{CC}	95	P1.5/$\overline{GCS5}$
21	P2.2/BCLK0	46	INT7	71	V_{SS}	96	P1.4/$\overline{GCS4}$
22	P2.3/$\overline{CTS0}$	47	HOLD	72	V_{CC}	97	P1.3/$\overline{GCS3}$
23	P2.4/RXD1	48	HLDA	73	AD3	98	P1.2/$\overline{GCS2}$
24	P2.5/TXD1	49	DT/$\overline{R}$	74	AD2	99	P1.1/$\overline{GCS1}$
25	P2.6/BCLK1	50	$\overline{DEN}$	75	AD1	100	P1.0/$\overline{GCS0}$



NOTE:
This is the FPO number location (indicated by X's).

Figure 5. Quad Flat Pack (EIAJ) Pinout Diagram

Table 7. SQFP Pin Functions with Location

AD Bus	
AD0	73
AD1	72
AD2	71
AD3	70
AD4	66
AD5	65
AD6	64
AD7	63
AD8 (A8)	60
AD9 (A9)	59
AD10 (A10)	58
AD11 (A11)	57
AD12 (A12)	56
AD13 (A13)	55
AD14 (A14)	54
AD15 (A15)	53
A16	77
A17	76
A18	75
A19/ONCE	74

Bus Control	
ALE	52
$\overline{BHE}$ ($\overline{RFSH}$)	51
$\overline{S0}$	78
$\overline{S1}$	79
$\overline{S2}$	80
$\overline{RD}$	50
$\overline{WR}$	49
READY	85
DT/$\overline{R}$	46
$\overline{DEN}$	47
$\overline{LOCK}$	48
HOLD	44
HLDA	45

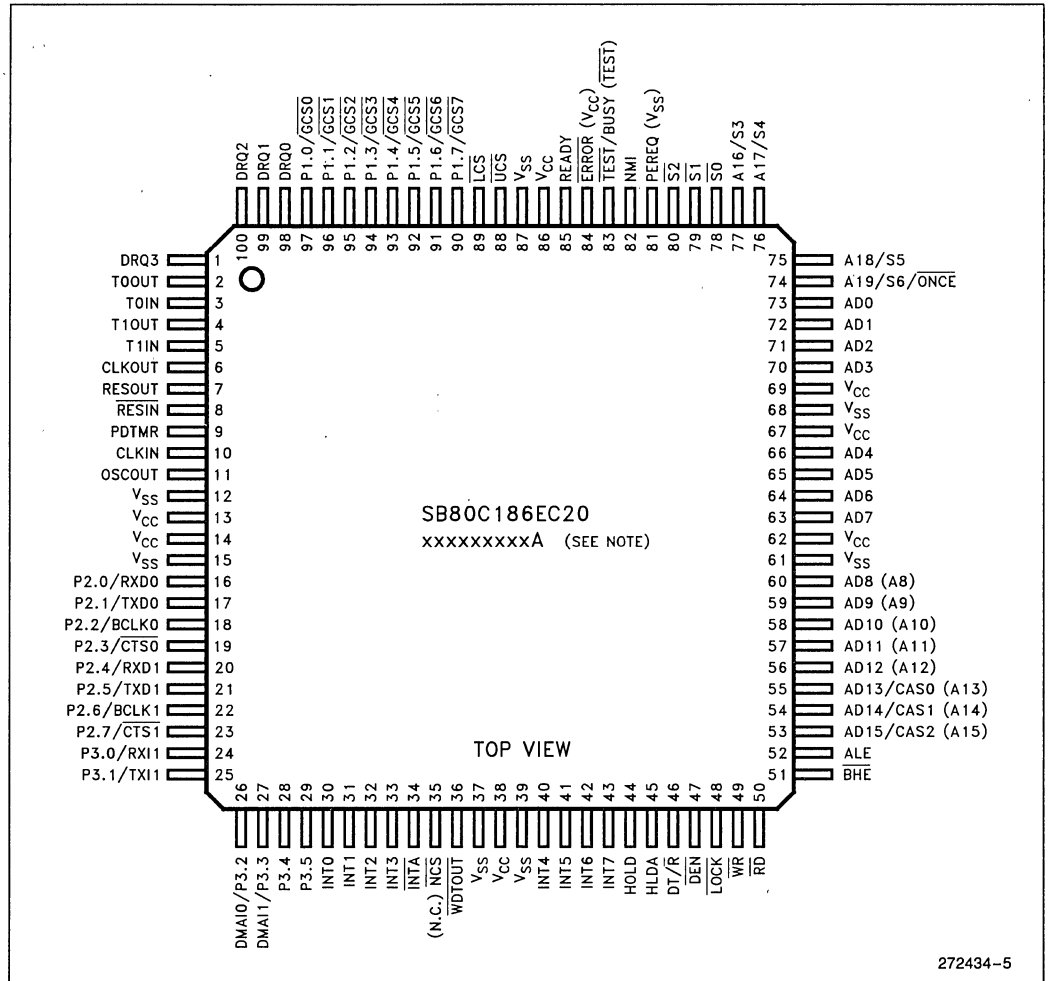
Processor Control	
$\overline{RESIN}$	8
RESOUT	7
CLKIN	10
OSCOUT	11
CLKOUT	6
$\overline{TEST}$/BUSY	83
NMI	82
INT0	30
INT1	31
INT2	32
INT3	33
INT4	40
INT5	41
INT6	42
INT7	43
$\overline{INTA}$	34
PEREQ (V_{SS})	81
$\overline{ERROR}$ (V_{CC})	84
$\overline{NCS}$ (N.C.)	35
PDTMR	9

Power and Ground	
V_{CC}	13
V_{CC}	14
V_{CC}	38
V_{CC}	62
V_{CC}	67
V_{CC}	69
V_{CC}	86
V_{SS}	12
V_{SS}	15
V_{SS}	37
V_{SS}	39
V_{SS}	61
V_{SS}	68
V_{SS}	87

I/O	
$\overline{UCS}$	88
$\overline{LCS}$	89
P1.0/$\overline{GCS0}$	97
P1.1/$\overline{GCS1}$	96
P1.2/$\overline{GCS2}$	95
P1.3/$\overline{GCS3}$	94
P1.4/$\overline{GCS4}$	93
P1.5/$\overline{GCS5}$	92
P1.6/$\overline{GCS6}$	91
P1.7/$\overline{GCS7}$	90
P2.0/RXD0	16
P2.1/TXD0	17
P2.2/BCLK0	18
P2.3/$\overline{CTS0}$	19
P2.4/RXD1	20
P2.5/TXD1	21
P2.6/BCLK1	22
P2.7/$\overline{CTS1}$	23
P3.0/RXI1	24
P3.1/TXI1	25
P3.2/DMAI0	26
P3.3/DMAI1	27
P3.4	28
P3.5	29
DRQ0	98
DRQ1	99
DRQ2	100
DRQ3	1
T0IN	3
T0OUT	2
T1IN	5
T1OUT	4
$\overline{WDTOUT}$	36

Table 8. SQFP Pin Locations with Pin Names

Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	DRQ3	26	P3.2/DMAI0	51	$\overline{BHE}$ ($\overline{RFSH}$)	76	A17
2	T0OUT	27	P3.3/DMAI1	52	ALE	77	A16
3	T0IN	28	P3.4	53	AD15 (A15)	78	$\overline{S0}$
4	T1OUT	29	P3.5	54	AD14 (A14)	79	$\overline{S1}$
5	T1IN	30	INT0	55	AD13 (A13)	80	$\overline{S2}$
6	CLKOUT	31	INT1	56	AD12 (A12)	81	PEREQ (V_{SS})
7	RESOUT	32	INT2	57	AD11 (A11)	82	MNI
8	$\overline{RESIN}$	33	INT3	58	AD10 (A10)	83	$\overline{TEST}$/BUSY ($\overline{TEST}$)
9	PDTMR	34	$\overline{INTA}$	59	AD9 (A9)	84	$\overline{ERROR}$ (V_{CC})
10	CLKIN	35	$\overline{NSC}$ (N.C.)	60	AD8 (A8)	85	READY
11	OSCOUT	36	$\overline{WDTOUT}$	61	V_{SS}	86	V_{CC}
12	V_{SS}	37	V_{SS}	62	V_{CC}	87	V_{SS}
13	V_{CC}	38	V_{CC}	63	AD7 (A7)	88	$\overline{UCS}$
14	V_{CC}	39	V_{SS}	64	AD6 (A6)	89	$\overline{LCS}$
15	V_{SS}	40	INT4	65	AD5	90	P1.7/$\overline{GCS7}$
16	P2.0/RXD0	41	INT5	66	AD4	91	P1.6/$\overline{GS6}$
17	P2.1/TXD0	42	INT6	67	V_{CC}	92	P1.5/$\overline{GCS5}$
18	P2.2/BCLK0	43	INT7	68	V_{SS}	93	P1.4/$\overline{GCS4}$
19	P2.3/$\overline{CTS0}$	44	HOLD	69	V_{CC}	94	P1.3/$\overline{GCS3}$
20	P2.4/RXD1	45	HLDA	70	AD3	95	P1.2/$\overline{GCS2}$
21	P2.5/TXD1	46	DT/$\overline{R}$	71	AD2	96	P1.1/$\overline{GCS1}$
22	P2.6/BCLK1	47	$\overline{DEN}$	72	AD1	97	P1.0/$\overline{GCS0}$
23	P2.7/$\overline{CTS1}$	48	$\overline{LOCK}$	73	AD0	98	DRQ0
24	P3.0/RXI1	49	$\overline{WR}$	74	A19/ONCE	99	DRQ1
25	P3.1/TXI1	50	$\overline{RD}$	75	AD18	100	DRQ2



NOTE:
This is the FPO number location (indicated by X's)

Figure 6. 100-Pin Shrink Quad Flat Pack Package (SQFP)

Package Thermal Specifications

The 80C186EC/80L186EC is specified for operation when T_C (the case temperature) is within the range of −40°C to +100°C. T_C may be measured in any environment to determine whether the processor is within the specified operating range. The case temperature must be measured at the center of the top surface.

T_A (the ambient temperature) can be calculated from θ_{CA} (thermal resistance from the case to ambient) with the following equation:

$$T_A = T_C - P * \theta_{CA}$$

Typical values for θ_{CA} at various airflows are given in Table 9. P (the maximum power consumption—specified in Watts) is calculated by using the maximum I_{CC} and V_{CC} of 5.5V.

Table 9. Thermal Resistance (θ_{CA}) at Various Airflows (in °C/Watt)

	Airflow in ft/min (m/sec)					
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
θ_{CA} (PQFP)	27.0	22.0	18.0	15.0	14.0	13.5
θ_{CA} (QFP)	64.5	55.5	51.0	TBD	TBD	TBD
θ_{CA} (SQFP)	62.0	TBD	TBD	TBD	TBD	TBD

ELECTRICAL SPECIFICATIONS

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

Absolute Maximum Ratings

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

Case Temperature Under Bias... −65°C to +100°C

Supply Voltage
with Respect to V_{SS} ..... −0.5V to +6.5V

Voltage on Other Pins
with Respect to V_{SS} ..... −0.5V to V_{CC} + 0.5V

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

Recommended Connections

Power and ground connections must be made to multiple V_{CC} and V_{SS} pins. Every 80C186EC-based circuit board should include separate power (V_{CC}) and ground (V_{SS}) planes. Every V_{CC} pin must be connected to the power plane, and every V_{SS} pin must be connected to the ground plane. Liberal decoupling capacitance should be placed near the processor. The processor can cause transient power surges when its output buffers transition, particularly when connected to large capacitive loads.

Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance is reduced by placing the decoupling capacitors as close as possible to the processor V_{CC} and V_{SS} package pins.

Always connect any unused input to an appropriate signal level. In particular, unused interrupt inputs (NMI, INT0:7) should be connected to V_{SS} through a pull-down resistor. Leave any unused output pin unconnected.

DC SPECIFICATIONS (80C186EC/80C188EC)

Symbol	Parameter	Min	Max	Units	Notes
V_{CC}	Supply Voltage	4.5	5.5	V	
V_{IL}	Input Low Voltage	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 3 mA (Min)
V_{OH}	Output High Voltage	V_{CC} − 0.5		V	I_{OH} = −2 mA (Min)
V_{HYR}	Input Hysteresis on $\overline{RESIN}$	0.5		V	
I_{LI}	Input Leakage Current for Pins: AD15:0 (AD7:0, A15:8), READY, HOLD, $\overline{RESIN}$, CLKIN, $\overline{TEST}$/BUSY, NMI, INT7:0, T0IN, T1IN, P2.7–P2.0, P3.5–P3.0, DRQ3:0, PEREQ, $\overline{ERROR}$		±15	μA	0 ≤ V_{IN} ≤ V_{CC}
I_{LIU}	Input Leakage for Pins with Pullups Active During Reset: A19:16, $\overline{LOCK}$	−0.275	−5	mA	V_{IN} = 0.7 V_{CC} (Note 1)
I_{LO}	Output Leakage for Floated Output Pins		±15	μA	0.45 ≤ V_{OUT} ≤ V_{CC} (Note 2)
I_{CC}	Supply Current Cold (in RESET) 80C186EC25 80C186EC20 80C186EC13		 125 100 70	 mA mA mA	 (Notes 3, 7) (Note 3) (Note 3)
I_{ID}	Supply Current in Idle Mode 80C186EC25 80C186EC20 80C186EC13		 92 76 50	 mA mA mA	 (Notes 4, 7) (Note 4) (Note 4)
I_{PD}	Supply Current in Powerdown Mode 80C186EC25 80C186EC20 80C186EC13		 100 100 100	 μA μA μA	 (Notes 5, 7) (Note 5) (Note 5)
C_{IN}	Input Pin Capacitance	0	15	pF	T_F = 1 MHz
C_{OUT}	Output Pin Capacitance	0	15	pF	T_F = 1 MHz (Note 6)

NOTES:
1. These pins have an internal pull-up device that is active while $\overline{RESIN}$ is low and ONCE Mode is not active. Sourcing more current than specified (on any of these pins) may invoke a factory test mode.
2. Tested by outputs being floated by invoking ONCE Mode or by asserting HOLD.
3. Measured with the device in RESET and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
4. Measured with the device in HALT (IDLE Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
5. Measured with the device in HALT (Powerdown Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
6. Output Capacitance is the capacitive load of a floating output pin.
7. Operating conditions for 25 MHz is 0°C to +70°C, V_{CC} = 5.0 ±10%.

DC SPECIFICATIONS (80L186EC/80L188EC)

Symbol	Parameter	Min	Max	Units	Notes
V_{CC}	Supply Voltage	2.7	5.5	V	
V_{IL}	Input Low Voltage	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 3 mA (Min)
V_{OH}	Output High Voltage	V_{CC} − 0.5		V	I_{OH} = −2 mA (Min)
V_{HYR}	Input Hysteresis on $\overline{RESIN}$	0.5		V	
I_{LI}	Input Leakage Current for Pins: AD15:0 (AD7:0, A15:8), READY, HOLD, $\overline{RESIN}$, CLKIN, $\overline{TEST}$/BUSY, NMI, INT7:0, T0IN, T1IN, P2.7–P2.0, P3.5–P3.0, DRQ3:0, PEREQ, $\overline{ERROR}$		±15	μA	0 ≤ V_{IN} ≤ V_{CC}
I_{LIU}	Input Leakage for Pins with Pullups Active During Reset: A19:16, $\overline{LOCK}$	−0.275	−5	mA	V_{IN} = 0.7 V_{CC} (Note 1)
I_{LO}	Output Leakage for Floated Output Pins		±15	μA	0.45 ≤ V_{OUT} ≤ V_{CC} (Note 2)
I_{CC}	Supply Current Cold (in RESET) 80L186EC-13		36	mA	(Note 3)
I_{ID}	Supply Current in Idle Mode 80L186EC-13		24	mA	(Note 4)
I_{PD}	Supply Current in Powerdown Mode 80L186EC-13		30	μA	(Note 5)
C_{IN}	Input Pin Capacitance	0	15	pF	T_F = 1 MHz
C_{OUT}	Output Pin Capacitance	0	15	pF	T_F = 1 MHz (Note 6)

NOTES:
1. These pins have an internal pull-up device that is active while $\overline{RESIN}$ is low and ONCE Mode is not active. Sourcing more current than specified (on any of these pins) may invoke a factory test mode.
2. Tested by outputs being floated by invoking ONCE Mode or by asserting HOLD.
3. Measured with the device in RESET and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
4. Measured with the device in HALT (IDLE Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
5. Measured with the device in HALT (Powerdown Mode active) and at worst case frequency, V_{CC}, and temperature with **ALL** outputs loaded as specified in AC Test Conditions, and all floating outputs driven to V_{CC} or GND.
6. Output Capacitance is the capacitive load of a floating output pin.

I_{CC} versus Frequency and Voltage

The I_{CC} consumed by the processor is composed of two components:

1. I_{PD}—The quiescent current that represents internal device leakage. Measured with all inputs at either V_{CC} or ground and no clock applied.
2. I_{CCS}—The switching current used to charge and discharge internal parasitic capacitance when changing logic levels. I_{CCS} is related to both the frequency of operation and the device supply voltage (V_{CC}). I_{CCS} is given by the formula:

$$\text{Power} = V * I = V^2 * C_{DEV} * f$$
$$\therefore I_{CCS} = V * C_{DEV} * f$$

Where:

V = Supply Voltage (V_{CC})

C_{DEV} = Device Capacitance

f = Operating Frequency

Measuring C_{PD} on a device like the 80C186EC would be difficult. Instead, C_{PD} is calculated using the above formula with I_{CC} values measured at known V_{CC} and frequency. Using the C_{PD} value, the user can calculate I_{CC} at any voltage and frequency within the specified operating range.

Example. Calculate typical I_{CC} at 14 MHz, 5.2V V_{CC}.

$I_{CC} = I_{PD} + I_{CCS}$
$= 0.1\text{ mA} + 5.2V * 0.77 * 14\text{ MHz}$
$= 56.2\text{ mA}$

PDTMR Pin Delay Calculation

The PDTMR pin provides a delay between the assertion of NMI and the enabling of the internal clocks when exiting Powerdown Mode. A delay is required only when using the on chip oscillator to allow the crystal or resonator circuit to stabilize.

NOTE:
The PDTMR pin function does not apply when $\overline{\text{RESIN}}$ is asserted (i.e. a device reset while in Powerdown is similar to a cold reset and $\overline{\text{RESIN}}$ must remain active until after the oscillator has stabilized.

To calculate the value of capacitor to use to provide a desired delay, use the equation:

$$440 \times t = C_{PD}\ (5V, 25°C)$$

Where:

t = desired delay in **seconds**
C_{PD} = capacitive load on PDTMR in **microfarads**

Example. For a delay of 300 μs, a capacitor value of $C_{PD} = 440 \times (300 \times 10^{-6} = 0.132\ \mu F$ is required. Round up to a standard (available) capacitor value.

NOTE:
The above equation applies to delay time longer than 10 μs and will compute the **TYPICAL** capacitance needed to achieve the desired delay. A delay variance of +50% to −25% can occur due to temperature, voltage, and device process extremes. In general, higher V_{CC} and/or lower temperatures will decrease delay time, while lower V_{CC} and/or higher temperature will increase delay time.

Parameter	Typical	Max	Units	Notes
CPD	0.77	1.37	mA/V*MHz	1, 2
CPD (Idle Mode)	0.55	0.96	mA/V*MHz	1, 2

NOTES:
1. Maximum C_{PD} is measured at −40°C with all outputs loaded as specified in the AC test conditions and the device in reset (or Idle Mode). Due to tester limitations, CLKOUT and OSCOUT also have 50 pF loads that increase I_{CC} by V*C*F.
2. Typical C_{PD} is calculated at 25°C assuming no loads on CLKOUT or OSCOUT and the device in reset (or Idle Mode).

AC SPECIFICATIONS

AC Characteristics—80C186EC25

Symbol	Parameter	25 MHz		Units	Notes
		Min	Max		
INPUT CLOCK					
T_F	CLKIN Frequency	0	50	MHz	1
T_C	CLKIN Period	20	∞	ns	1
T_{CH}	CLKIN High Time	8	∞	ns	1, 2
T_{CL}	CLKIN Low Time	8	∞	ns	1, 2
T_{CR}	CLKIN Rise Time	1	10	ns	1, 3
T_{CF}	CLKIN Fall Time	1	10	ns	1, 3
OUTPUT CLOCK					
T_{CD}	CLKIN to CLKOUT Delay	0	17	ns	1, 4
T	CLKOUT Period		$2*T_C$	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PR}	CLKOUT Rise Time	1	6	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	6	ns	1, 5
OUTPUT DELAYS					
T_{CHOV1}	ALE, $\overline{S2:0}$, $\overline{DEN}$, DT/$\overline{R}$, $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$, A19:16	3	17	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{GCS0:7}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{NCS}$, $\overline{RD}$, $\overline{WR}$	3	20	ns	1, 4, 6, 8
T_{CLOV1}	$\overline{BHE}$ ($\overline{RFSH}$), $\overline{DEN}$, $\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT, A19:16	3	17	ns	1, 4, 6
T_{CLOV2}	$\overline{RD}$, $\overline{WR}$, $\overline{GCS7:0}$, $\overline{LCS}$, $\overline{UCS}$, AD15:0 (AD7:0, A15:8), $\overline{NCS}$, $\overline{INTA1:0}$, $\overline{S2:0}$	3	20	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	20	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (AD7:0, A15:8)	0	20	ns	1

AC SPECIFICATIONS

AC Characteristics—80C186EC25 (Continued)

Symbol	Parameter	25 MHz		Units	Notes
		Min	Max		
SYNCHRONOUS INPUTS					
T_{CHIS}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY, $\overline{CTS1:0}$, P2.6, P2.7	10		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, INT4:0, BCLK1:0, T1:0IN, READY, $\overline{CTS1:0}$	3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), READY	10		ns	1, 10
T_{CLIH}	READY, AD15:0 (AD7:0)	3		ns	1, 10
T_{CLIS}	HOLD, PEREQ, $\overline{ERROR}$	10		ns	1, 9
T_{CLIH}	HOLD, PEREQ, $\overline{ERROR}$	3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measure at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 13 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 14 for rise and fall times outside 50 pF.
6. See Figure 14 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.

AC SPECIFICATIONS

AC Characteristics—80C186EC-20/80C186EC-13

Symbol	Parameter	Min	Max	Min	Max	Unit	Notes
INPUT CLOCK		**20 MHz**		**13 MHz**			
TF	CLKIN Frequency	0	40	0	26	MHz	1
TC	CLKIN Period	25	∞	38.5	∞	ns	1
TCH	CLKIN High Time	10	∞	12	∞	ns	1, 2
TCL	CLKIN Low Time	10	∞	12	∞	ns	1, 2
TCR	CLKIN Rise Time	1	10	1	10	ns	1, 3
TCF	CLKIN Fall Time	1	10	1	10	ns	1, 3
OUTPUT CLOCK							
T_{CD}	CLKIN to CLKOUT Delay	0	17	0	23	ns	1, 4
T	CLKOUT Period		2 * TC		2 * TC	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5	(T/2) + 5	(T/2) − 5	(T/2) + 5	ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5	(T/2) + 5	(T/2) − 5	(T/2) + 5	ns	1
T_{PR}	CLKOUT Rise Time	1	6	1	6	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	6	1	6	ns	1, 5
OUTPUT DELAYS							
T_{CHOV1}	ALE, $\overline{S2:0}$, $\overline{DEN}$, DT/$\overline{R}$, $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$, A19:16	3	20	3	25	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{GCS7:0}$, $\overline{LCS}$, $\overline{UCS}$, $\overline{RD}$, $\overline{WR}$, $\overline{NCS}$, $\overline{WDTOUT}$	3	23	3	30	ns	1, 4, 6, 8
T_{CLOV1}	$\overline{BHE}$ ($\overline{RFSH}$), $\overline{DEN}$, $\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT	3	20	3	25	ns	1, 4, 6
T_{CLOV2}	$\overline{RD}$, $\overline{WR}$, $\overline{GSC7:0}$, $\overline{LCS}$, $\overline{UCS}$, AD15:0 (AD7:0, A15:8), $\overline{NCS}$, $\overline{INTA}$, $\overline{S2:0}$, A19:16	3	23	3	30	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	25	0	30	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (AD7:0, A15:8)	0	25	0	30	ns	1
INPUT REQUIREMENTS							
T_{CHIS}	$\overline{TEST}$, NMI, T1IN, T0IN, READY, $\overline{CTS1:0}$, BCLK1:0, P3.4, P3.5	10		10		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, T1IN, T0IN, READY, $\overline{CTS1:0}$, BCLK1:0, P3.4, P3.5	3		3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), READY	10		10		ns	1, 10
T_{CLIH}	AD15:0 (AD7:0), READY	3		3		ns	1, 10
T_{CLIS}	HOLD, $\overline{RESIN}$, PEREQ, $\overline{ERROR}$, DRQ3:0	10		10		ns	1, 9
T_{CLIH}	HOLD, $\overline{RESIN}$, REREQ, $\overline{ERROR}$, DRQ3:0	3		3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measure at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 14 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 15 for rise and fall times outside 50 pF.
6. See Figure 15 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.

AC Characteristics—80L186EC13

Symbol	Parameter	Min	Max	Unit	Notes
INPUT CLOCK		**13 MHz**			
TF	CLKIN Frequency	0	26	MHz	1
TC	CLKIN Period	38.5	∞	ns	1
TCH	CLKIN High Time	15	∞	ns	1, 2
TCL	CLKIN Low Time	15	∞	ns	1, 2
TCR	CLKIN Rise Time	1	10	ns	1, 3
TCF	CLKIN Fall Time	1	10	ns	1, 3
OUTPUT CLOCK					
T_{CD}	CLKIN to CLKOUT Delay	0	20	ns	1, 4
T	CLKOUT Period		2 * TC	ns	1
T_{PH}	CLKOUT High Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PL}	CLKOUT Low Time	(T/2) − 5	(T/2) + 5	ns	1
T_{PR}	CLKOUT Rise Time	1	10	ns	1, 5
T_{PF}	CLKOUT Fall Time	1	10	ns	1, 5
OUTPUT DELAYS					
T_{CHOV1}	$\overline{S2:0}$, DT/$\overline{R}$, $\overline{BHE}$, $\overline{LOCK}$	3	28	ns	1, 4, 6, 7
T_{CHOV2}	$\overline{LCS}$, $\overline{UCS}$, $\overline{DEN}$, A19:16, $\overline{RD}$, $\overline{WR}$, $\overline{NCS}$, $\overline{WDTOUT}$, ALE	3	32	ns	1, 4, 6, 8
T_{CHOV3}	$\overline{GCS7:0}$	3	34	ns	1, 4, 6
T_{CLOV1}	$\overline{LOCK}$, RESOUT, HLDA, T0OUT, T1OUT	3	28	ns	1, 4, 6
T_{CLOV2}	$\overline{RD}$, $\overline{WR}$, AD15:0 (AD7:0, A15:8), $\overline{BHE}$ ($\overline{RFSH}$), $\overline{NCS}$, $\overline{INTA}$, $\overline{DEN}$	3	32	ns	1, 4, 6
T_{CLOV3}	$\overline{GSC7:0}$, $\overline{LCS}$, $\overline{UCS}$	3	34	ns	1, 4, 6
T_{CLOV4}	$\overline{S2:0}$, A19:16	3	37	ns	1, 4, 6
T_{CHOF}	$\overline{RD}$, $\overline{WR}$, $\overline{BHE}$ ($\overline{RFSH}$), DT/$\overline{R}$, $\overline{LOCK}$, $\overline{S2:0}$, A19:16	0	30	ns	1
T_{CLOF}	$\overline{DEN}$, AD15:0 (AD7:0, A15:8)	0	35	ns	1
INPUT REQUIREMENTS					
T_{CHIS}	$\overline{TEST}$, NMI, T1IN, T0IN, READY, $\overline{CTS1:0}$, BCLK1:0, P3.4, P3.5	20		ns	1, 9
T_{CHIH}	$\overline{TEST}$, NMI, T1IN, T0IN, READY, $\overline{CTS1:0}$, BCLK1:0, P3.4, P3.5	3		ns	1, 9
T_{CLIS}	AD15:0 (AD7:0), READY	20		ns	1, 10
T_{CLIH}	AD15:0 (AD7:0), READY	3		ns	1, 10
T_{CLIS}	HOLD, $\overline{RESIN}$, PEREQ, $\overline{ERROR}$, DRQ3:0	20		ns	1, 9
T_{CLIH}	HOLD, $\overline{RESIN}$, REREQ, $\overline{ERROR}$, DRQ3:0	3		ns	1, 9

NOTES:
1. See **AC Timing Waveforms**, for waveforms and definition.
2. Measure at V_{IH} for high time, V_{IL} for low time.
3. Only required to guarantee I_{CC}. Maximum limits are bounded by T_C, T_{CH} and T_{CL}.
4. Specified for a 50 pF load, see Figure 14 for capacitive derating information.
5. Specified for a 50 pF load, see Figure 15 for rise and fall times outside 50 pF.

AC Characteristics—80L186EC (Continued)

NOTES:
6. See Figure 15 for rise and fall times.
7. T_{CHOV1} applies to $\overline{BHE}$ ($\overline{RFSH}$), $\overline{LOCK}$ and A19:16 only after a HOLD release.
8. T_{CHOV2} applies to $\overline{RD}$ and $\overline{WR}$ only after a HOLD release.
9. Setup and Hold are required to guarantee recognition.
10. Setup and Hold are required for proper operation.

Relative Timings (80C186EC-25/20/13, 80L186EC13)

Symbol	Parameter	Min	Max	Unit	Notes
RELATIVE TIMINGS					
T_{LHLL}	ALE Active Pulse Width	T − 15		ns	
T_{AVLL}	AD Valid Setup before ALE Falls	½T − 10		ns	
T_{PLLL}	Chip Select Valid before ALE Falls	½T − 10		ns	1
T_{LLAX}	AD Hold after ALE Falls	½T − 10		ns	
T_{LLWL}	ALE Falling to $\overline{WR}$ Falling	½T − 15		ns	1
T_{LLRL}	ALE Falling to $\overline{RD}$ Falling	½T − 15		ns	1
T_{WHLH}	$\overline{WR}$ Rising to Next ALE Rising	½T − 10		ns	1
T_{AFRL}	AD Float to $\overline{RD}$ Falling	0		ns	
T_{RLRH}	$\overline{RD}$ Active Pulse Width	2T − 5		ns	2
T_{WLWH}	$\overline{WR}$ Active Pulse Width	2T − 5		ns	2
T_{RHAX}	$\overline{RD}$ Rising to Next Address Active	T − 15		ns	
T_{WHDX}	Output Data Hold after $\overline{WR}$ Rising	T − 15		ns	
T_{WHPH}	$\overline{WR}$ Rise to Chip Select Rise	½T − 10		ns	1
T_{RHPH}	$\overline{RD}$ Rise to Chip Select Rise	½T − 10		ns	1
T_{PHPL}	Chip Select Inactive to Next Chip Select Active	½T − 10		ns	1
T_{OVRH}	$\overline{ONCE}$ Active Setup to $\overline{RESIN}$ Rising	T		ns	
T_{RHOX}	$\overline{ONCE}$ Hold after $\overline{RESIN}$ Rise	T		ns	
T_{IHIL}	$\overline{INTA}$ High to Next $\overline{INTA}$ Low during INTA Cycle	4T − 5		ns	4
T_{ILIH}	$\overline{INTA}$ Active Pulse Width	2T − 5		ns	2, 4
T_{CVIL}	CAS2:0 Setup before 2nd $\overline{INTA}$ Pulse Low	8T		ns	2, 4

Relative Timings (80C186EC-25/20/13, 80L186EC13) (Continued)

Symbol	Parameter	Min	Max	Unit	Notes
RELATIVE TIMINGS					
T_{ILCX}	CAS2:0 Hold after 2nd $\overline{INTA}$ Pulse Low	4T		ns	2, 4
T_{IRES}	Interrupt Resolution Time		150	ns	3
T_{IRLH}	IR Low Time to Reset Edge Detector	50		ns	
T_{IRHIF}	IR Hold Time after 1st $\overline{INTA}$ Falling	25		ns	4, 5

NOTES:
1. Assumes equal loading on both pins.
2. Can be extended using wait states.
3. Interrupt resolution time is the delay between an unmasked interrupt request going active and the interrupt output of the 8259A module going active. This is not directly measureable by the user. For interrupt pin INT7 the delay from an active signal to an active input to the CPU would actually be twice the T_{IRES} value since the signal must pass through two 8259A modules.
4. See INTA Cycle Waveforms for definition.
5. To guarantee interrupt is not spurious.

Serial Port Mode 0 Timings (80C186EC-25/20/13, 80L186EC13)

Symbol	Parameter	Min	Max	Unit	Notes
RELATIVE TIMINGS					
T_{XLXL}	TXD Clock Period	T (n + 1)		ns	1, 2
T_{XLXH}	TXD Clock Low to Clock High (N > 1)	2T − 35	2T + 35	ns	1
T_{XLXH}	TXD Clock Low to Clock High (N = 1)	T − 35	T + 35	ns	1
T_{XHXL}	TXD Clock High to Clock Low (N > 1)	(n − 1) T − 35	(n − 1) T + 35	ns	1, 2
T_{XHXL}	TXD Clock High to Clock Low (N = 1)	T − 35	T + 35	ns	1
T_{QVXH}	RXD Output Data Setup to TXD Clock High (N > 1)	(n − 1)T − 35		ns	1, 2
T_{QVXH}	RXD Output Data Setup to TXD Clock High (N = 1)	T − 35		ns	1
T_{XHQX}	RXD Output Data Hold after TXD Clock High (N > 1)	2T − 35		ns	1
T_{XHQX}	RXD Output Data Hold after TXD Clock High (N = 1)	T − 35		ns	1
T_{XHQZ}	RXD Output Data Float after Last TXD Clock High		T + 20	ns	1
T_{DVXH}	RXD Input Data Setup to TXD Clock High	T + 20		ns	1
T_{XHDX}	RXD Input Data Setup after TXD Clock High	0		ns	1

NOTES:
1. See Figure 13 for Waveforms.
2. n is the value in the BxCMP register ignoring the ICLK bit.

AC TEST CONDITIONS

The AC specifications are tested with the 50 pF load shown in Figure 7. See the Derating Curves section to see how timings vary with load capacitance.

Specifications are measured at the $V_{CC}/2$ crossing point, unless otherwise specified. See AC Timing Waveforms for AC specification definitions, test pins and illustrations.

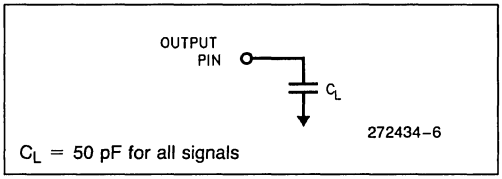


Figure 7. AC Test Load

AC TIMING WAVEFORMS

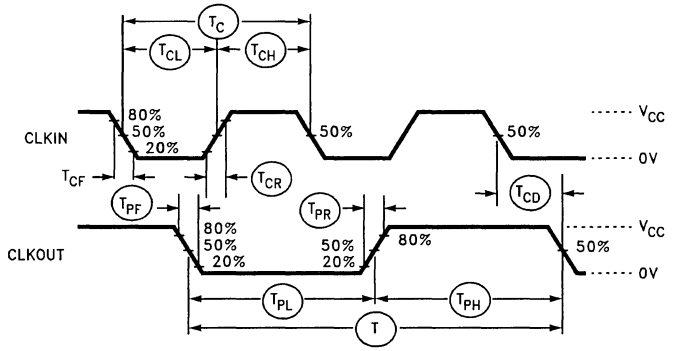


Figure 8. Input and Output Clock Waveforms

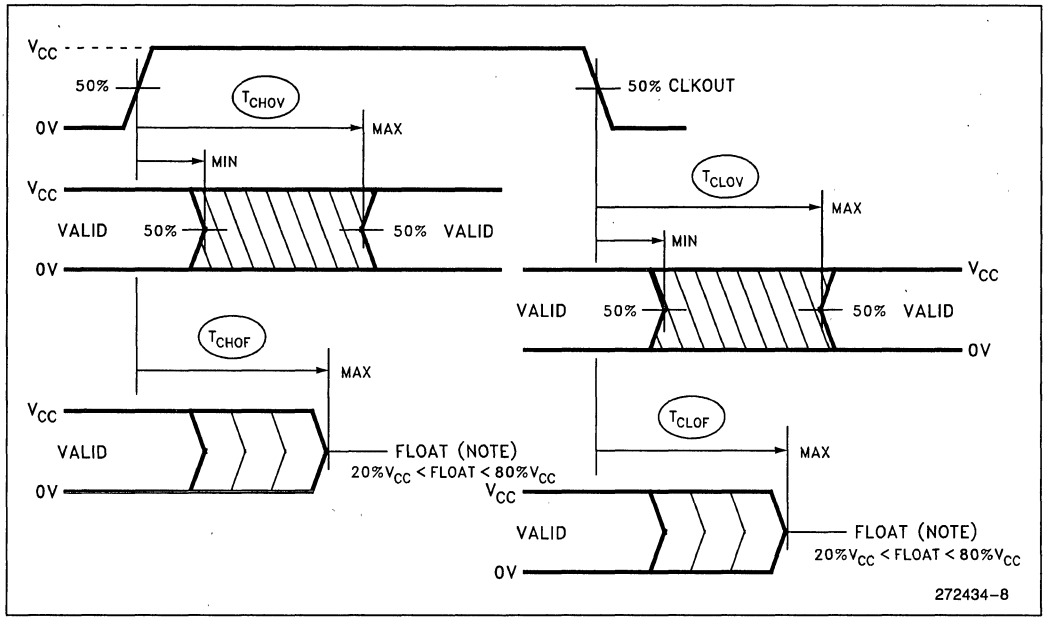


Figure 9. Output Delay and Float Waveforms

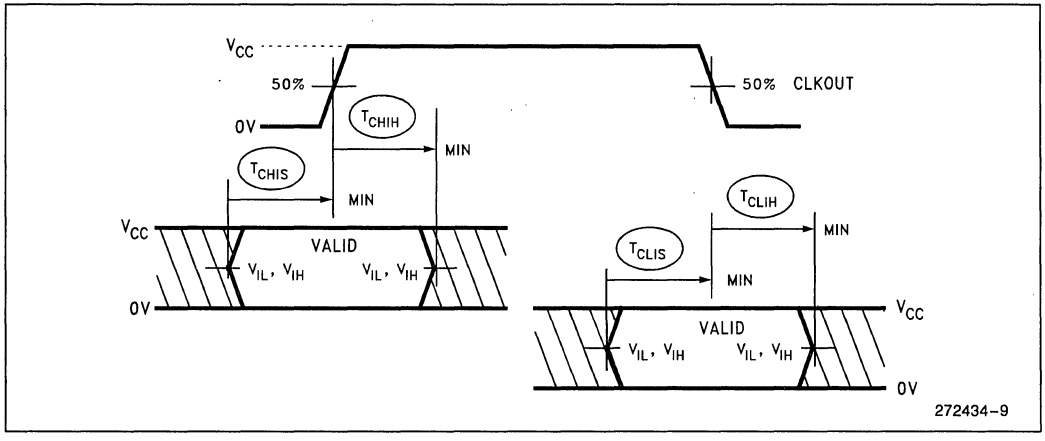


Figure 10. Input Setup and Hold

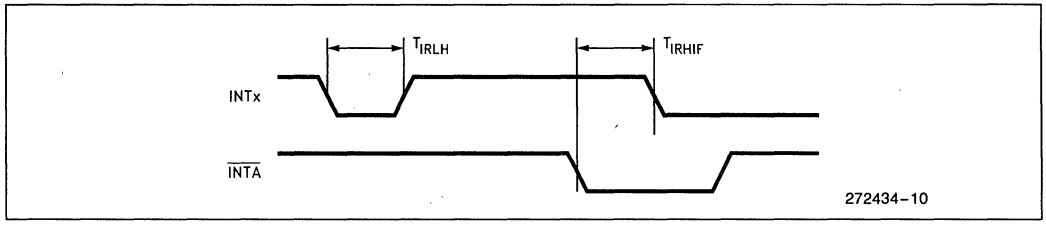


Figure 11. Relative Interrupt Signal Timings

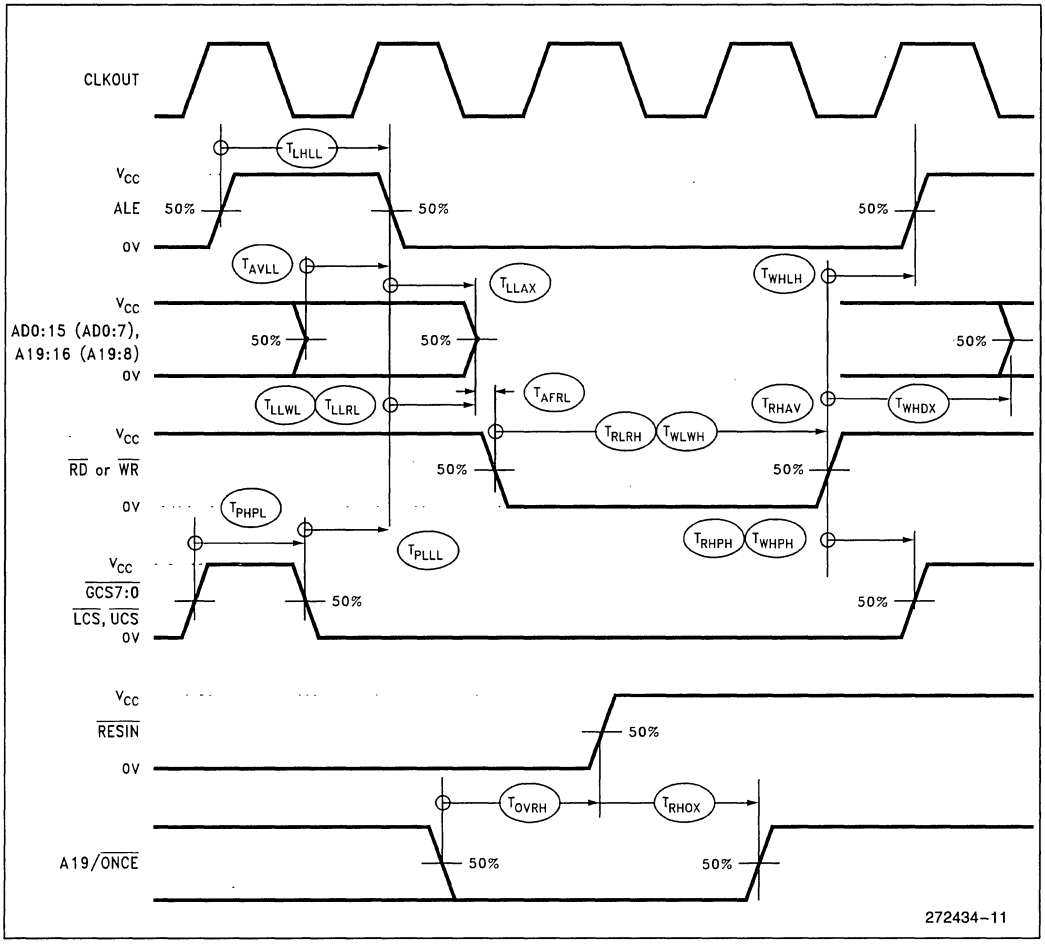


Figure 12. Relative Signal Waveform

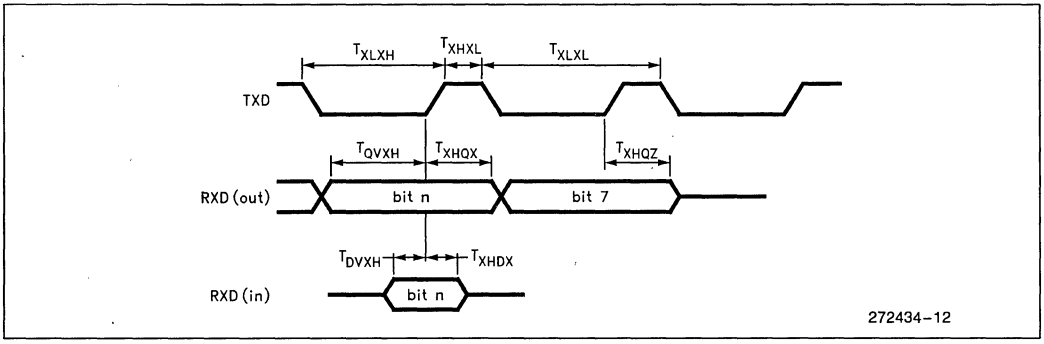


Figure 13. Serial Port Mode 0 Waveform

DERATING CURVES

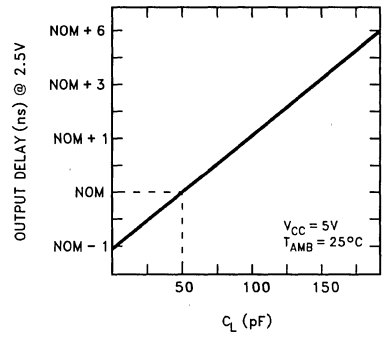


Figure 14. Typical Output Delay Variations versus Load Capacitance

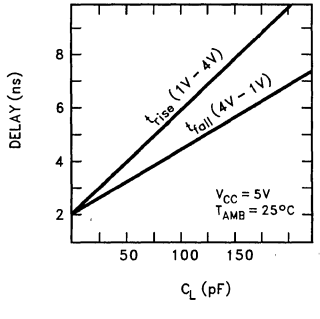


Figure 15. Typical Rise and Fall Variations versus Load Capacitance

RESET

The processor will perform a reset operation any time the $\overline{RESIN}$ pin is active. The $\overline{RESIN}$ pin is synchronized before it is presented internally, which means that the clock must be operating before a reset can take effect. From a power-on state, $\overline{RESIN}$ must be held active (low) in order to guarantee correct initialization of the processor. **Failure to provide $\overline{RESIN}$ while the device is powering up will result in unspecified operation of the device.**

Figure 16 shows the correct reset sequence when first applying power to the processor. An external clock connected to CLKIN must not exceed the V_{CC} threshold being applied to the processor. This is normally not a problem if the clock driver is supplied with the same V_{CC} that supplies the processor. When attaching a crystal to the device, $\overline{RESIN}$ must remain active until both V_{CC} and CLKOUT are stable (the length of time is application specific and depends on the startup characteristics of the crystal circuit). The $\overline{RESIN}$ pin is designed to operate correctly using a RC reset circuit, but the designer must ensure that the ramp time for V_{CC} is not so long that $\overline{RESIN}$ is never sampled at a logic low level when V_{CC} reaches minimum operating conditions.

Figure 17 shows the timing sequence when $\overline{RESIN}$ is applied after V_{CC} is stable and the device has been operating. Note that a reset will terminate all activity and return the processor to a known operating state. Any bus operation that is in progress at the time $\overline{RESIN}$ is asserted will terminate immediately (note that most control signals will be driven to their inactive state first before floating).

While $\overline{RESIN}$ is active, bus signals $\overline{LOCK}$, A19/S16/$\overline{ONCE}$ and A18:16 are configured as inputs and weakly held high by internal pullup transistors. Only A19/$\overline{ONCE}$ can be overdriven to a low and is used to enable the ONCE Mode. Forcing $\overline{LOCK}$ or A18:16 low at any time while $\overline{RESIN}$ is low is prohibited and will cause unspecified device operation.

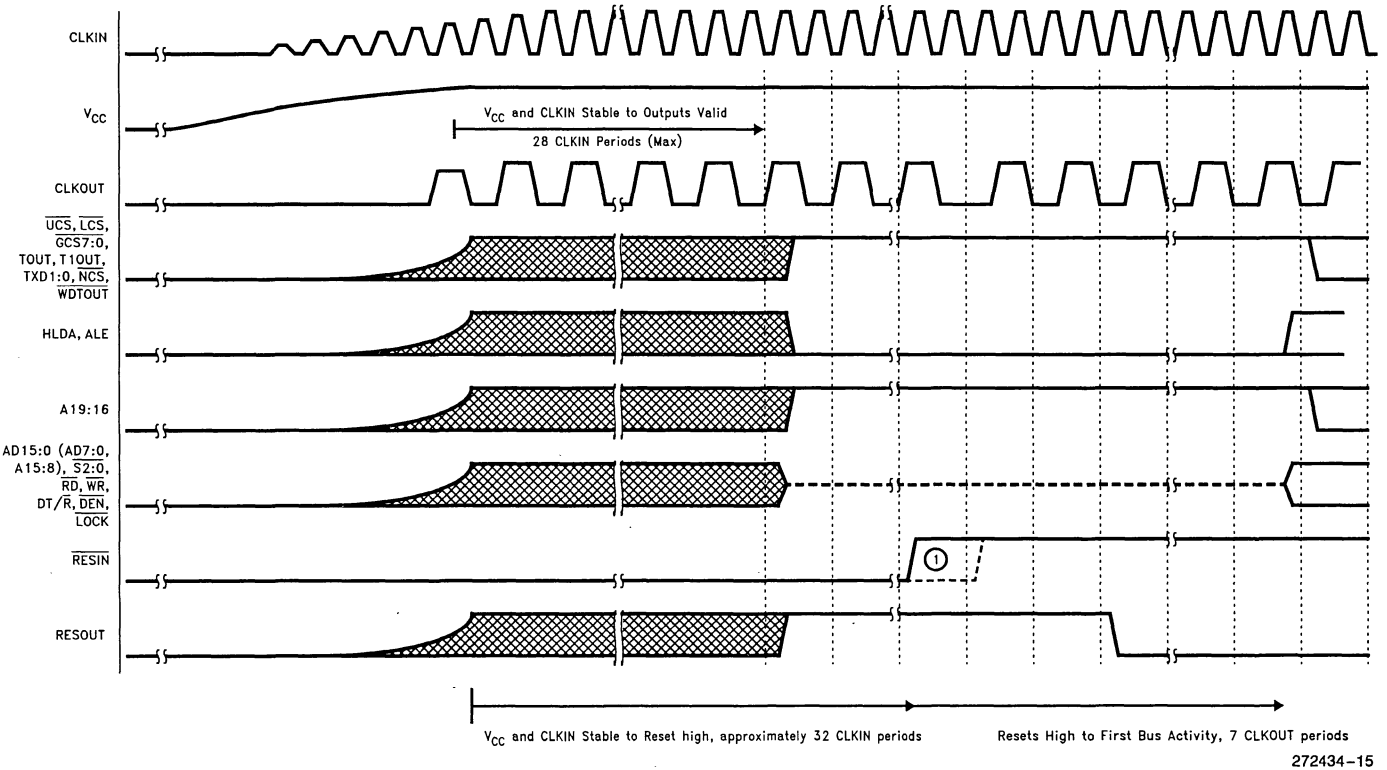
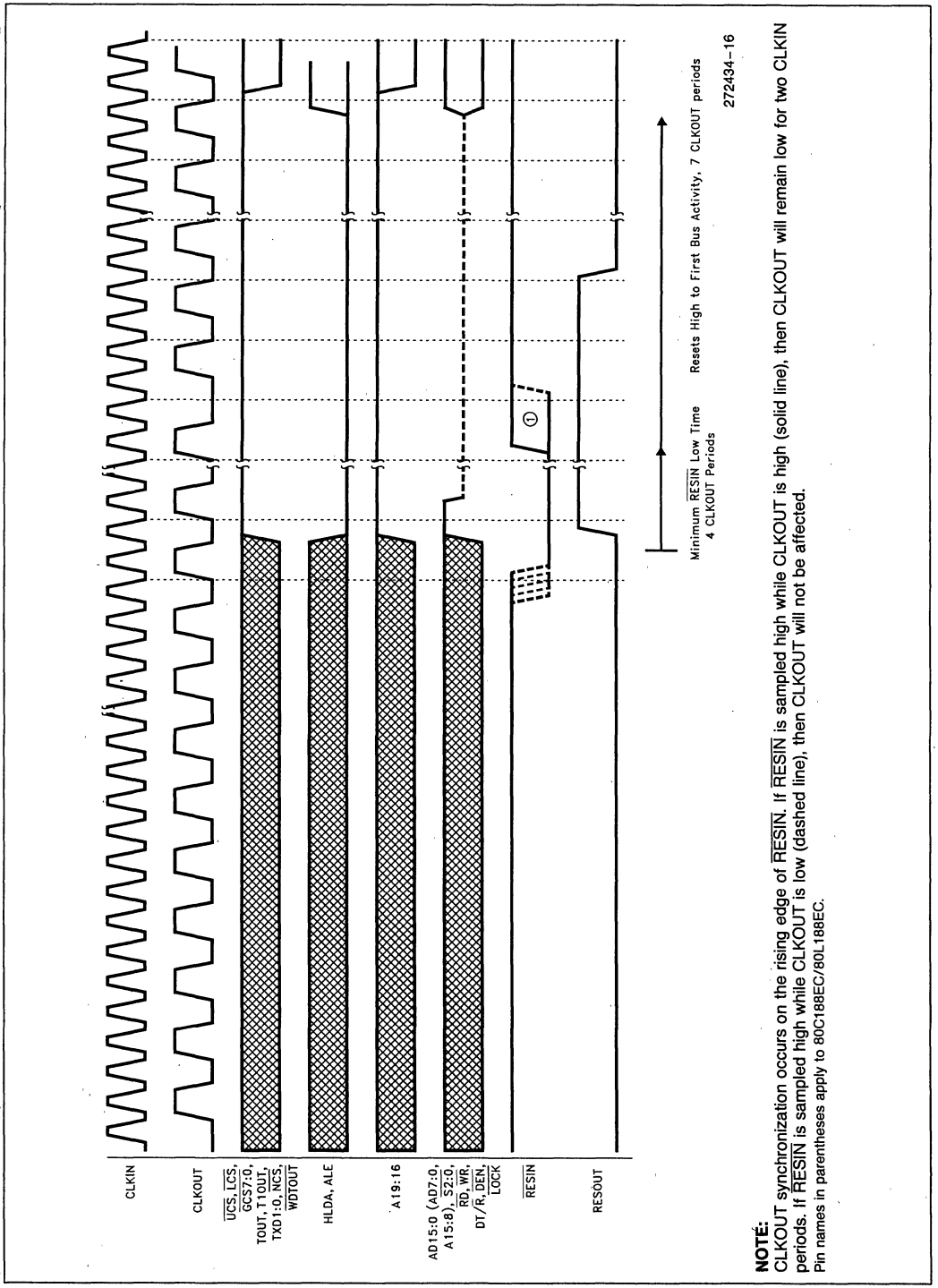


Figure 16. Cold RESET Waveforms

NOTE:
CLKOUT synchronization occurs on the rising edge of RESIN. If RESIN is sampled high while CLKOUT is high (solid line), then CLKOUT will remain low for two CLKIN periods. If RESIN is sampled high while CLKOUT is low (dashed line), then CLKOUT will not be affected.
Pin names in parentheses apply to 80C188EC/80L188EC.

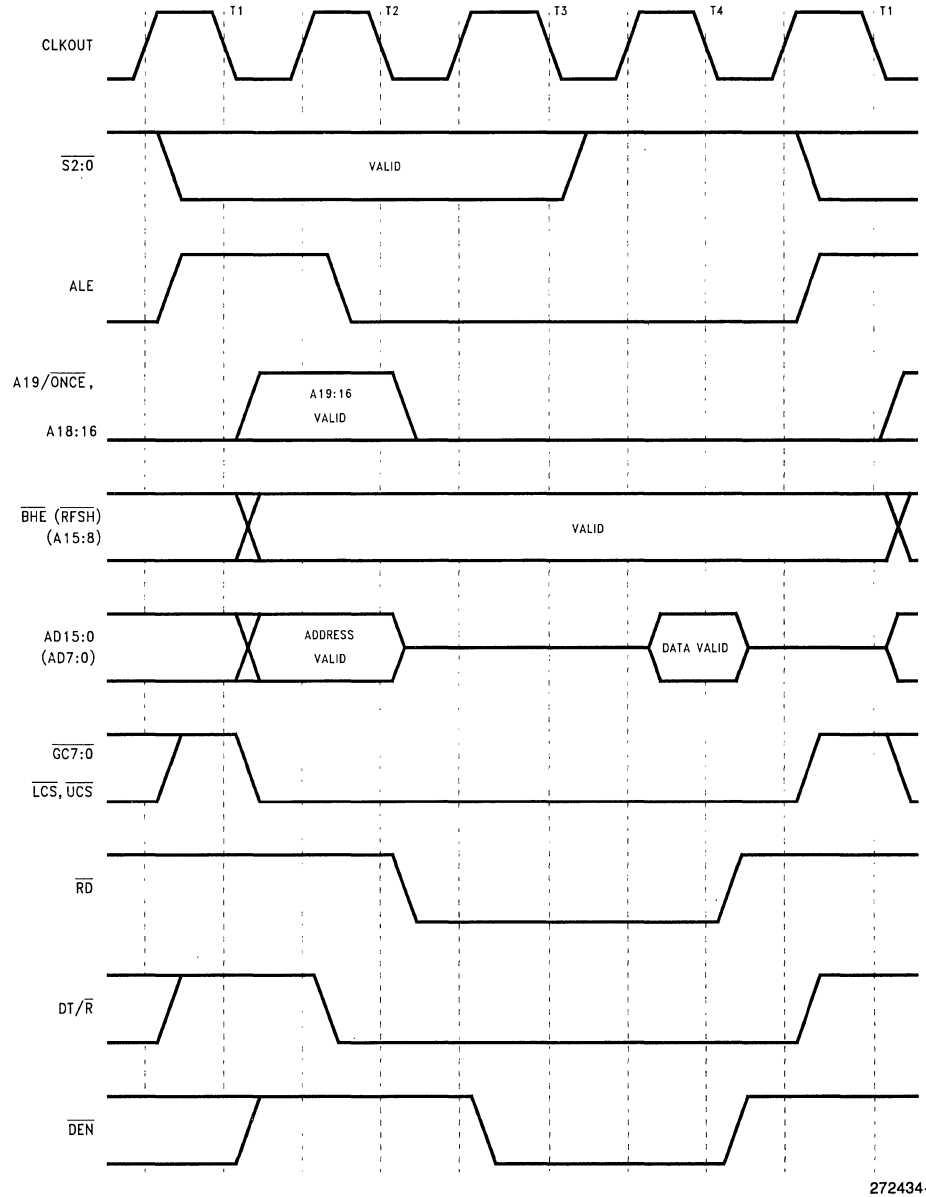


NOTE:
CLKOUT synchronization occurs on the rising edge of RESIN. If RESIN is sampled high while CLKOUT is high (solid line), then CLKOUT will remain low for two CLKIN periods. If RESIN is sampled high while CLKOUT is low (dashed line), then CLKOUT will not be affected.
Pin names in parentheses apply to 80C188EC/80L188EC.

Figure 17. Warm RESET Waveforms

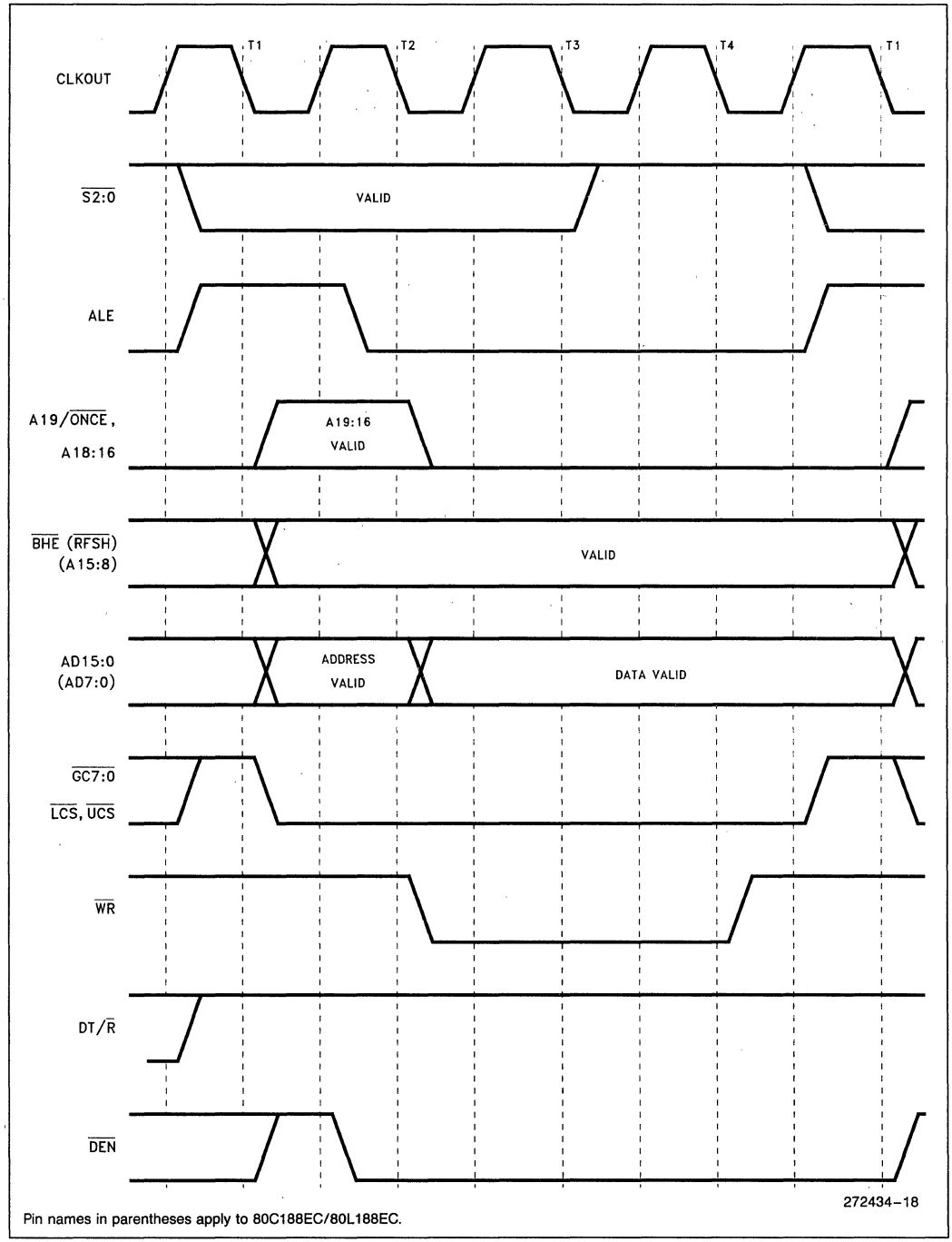
BUS CYCLE WAVEFORMS

Figures 18 through 24 present the various bus cycles that are generated by the processor. What is shown in the figure is the relationship of the various bus signals to CLKOUT. These figures along with the information present in AC Specifications allow the user to determine all the critical timing analysis needed for a given application.



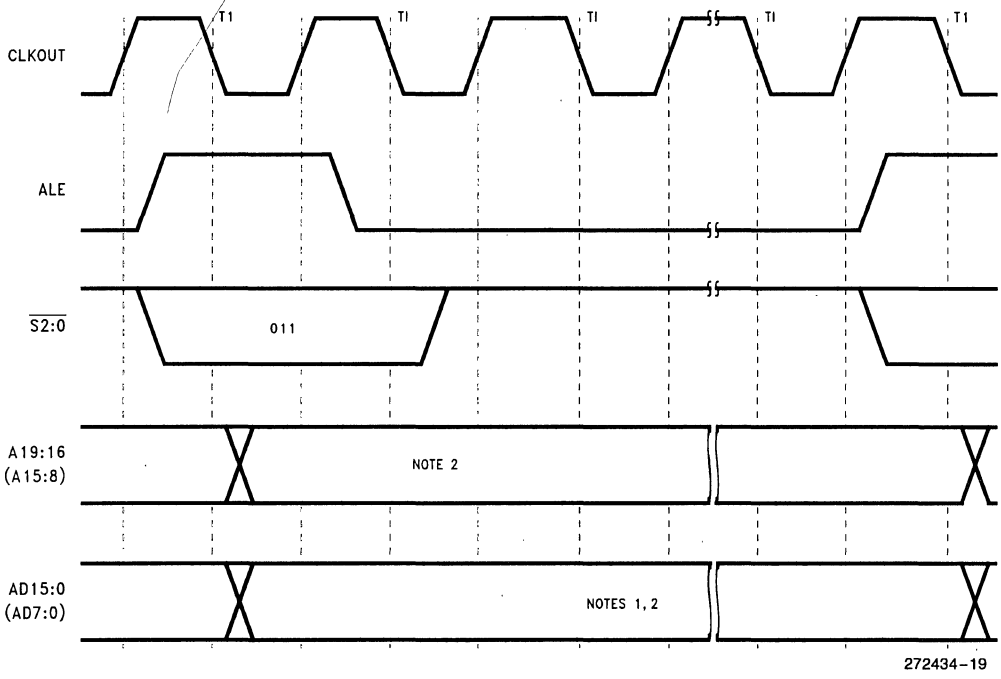
Pin names in parentheses apply to 80C188EC/80L188EC.

Figure 18. Memory Read, I/O Read, Instruction Fetch and Refresh Waveforms



Pin names in parentheses apply to 80C188EC/80L188EC.

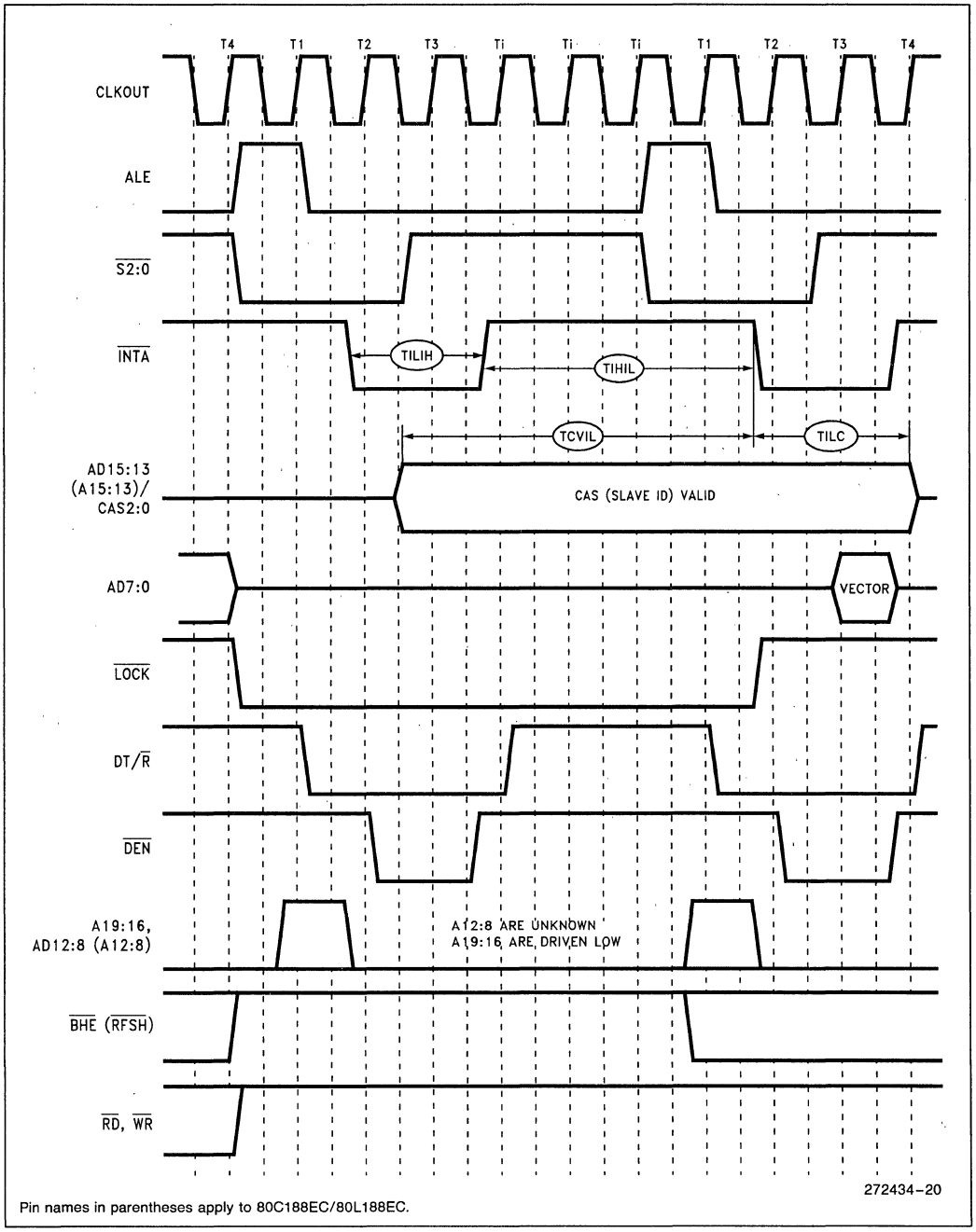
Figure 19. Memory Write and I/O Write Cycle Waveforms



NOTES:
1. Address information is invalid. If previous bus cycle was a read, then the AD15:0 (AD7:0) lines will float during T1. Otherwise, the AD15:0 (AD7:0) lines will continue to drive during T1 (data is invalid). All other control lines are in their inactive state.
2. All address lines drive zeros while in Powerdown or Idle Mode.

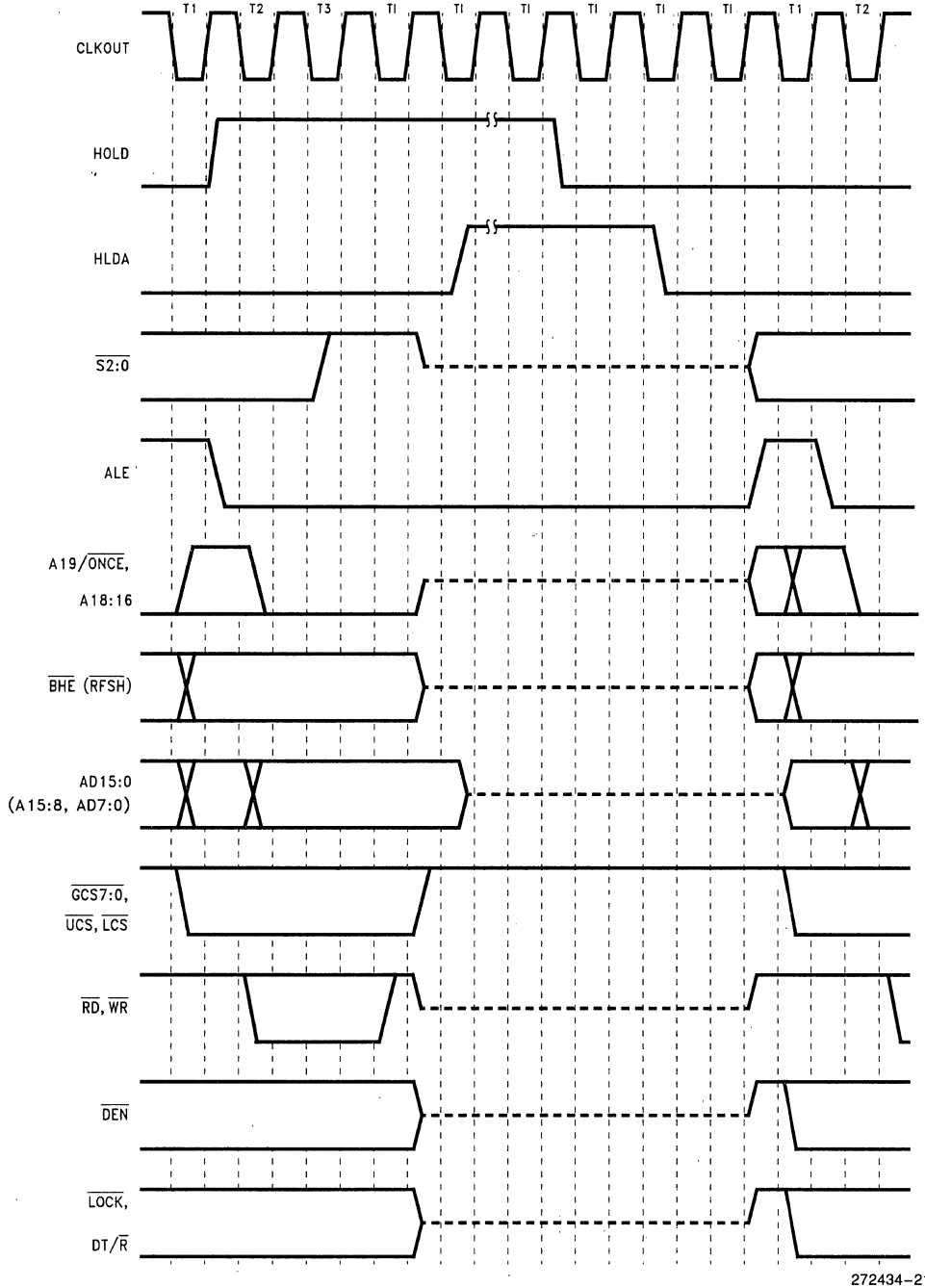
Pin names in parentheses apply to 80C188EC/80L188EC.

Figure 20. Halt Cycle Waveforms



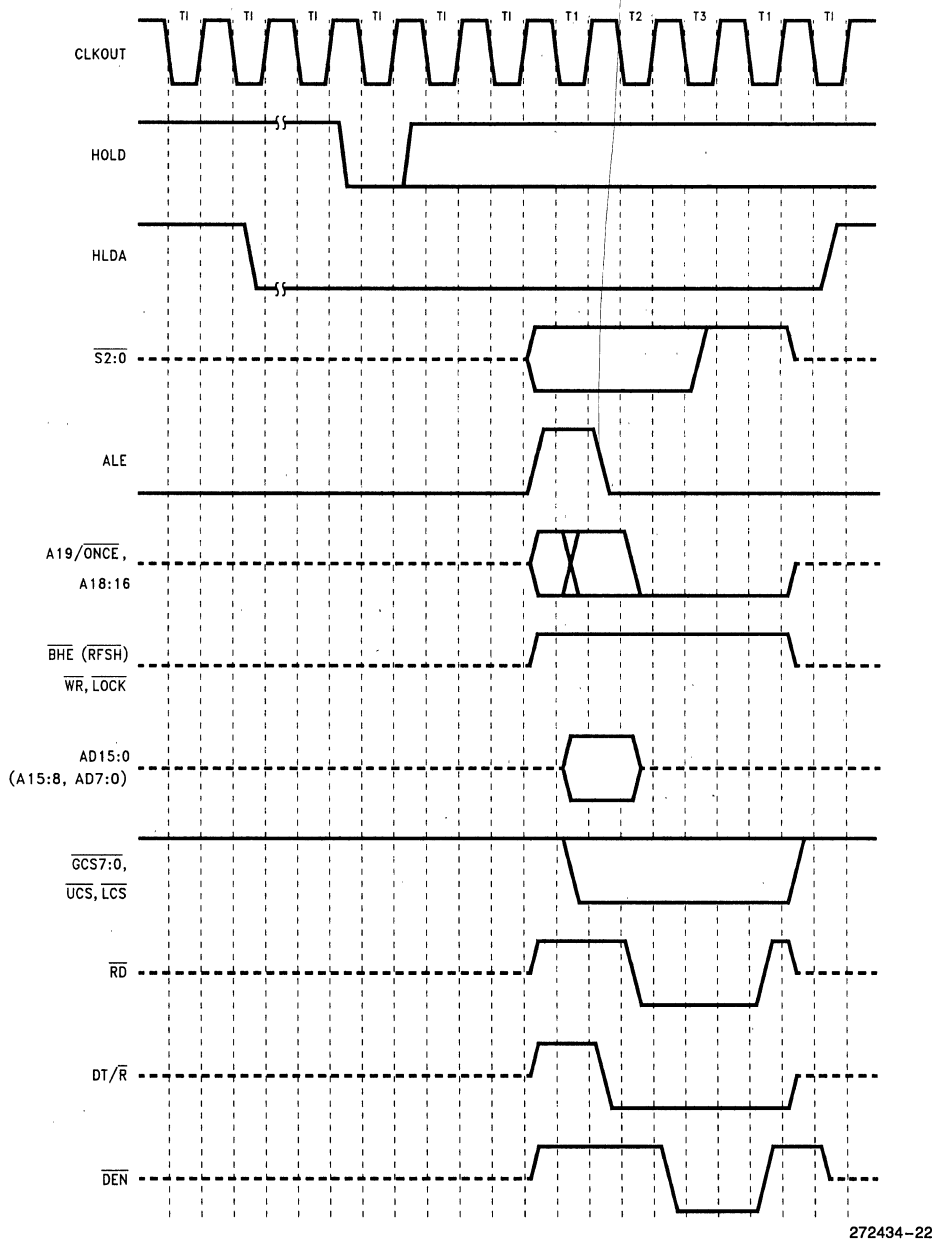
Pin names in parentheses apply to 80C188EC/80L188EC.

Figure 21. Interrupt Acknowledge Cycle Waveforms



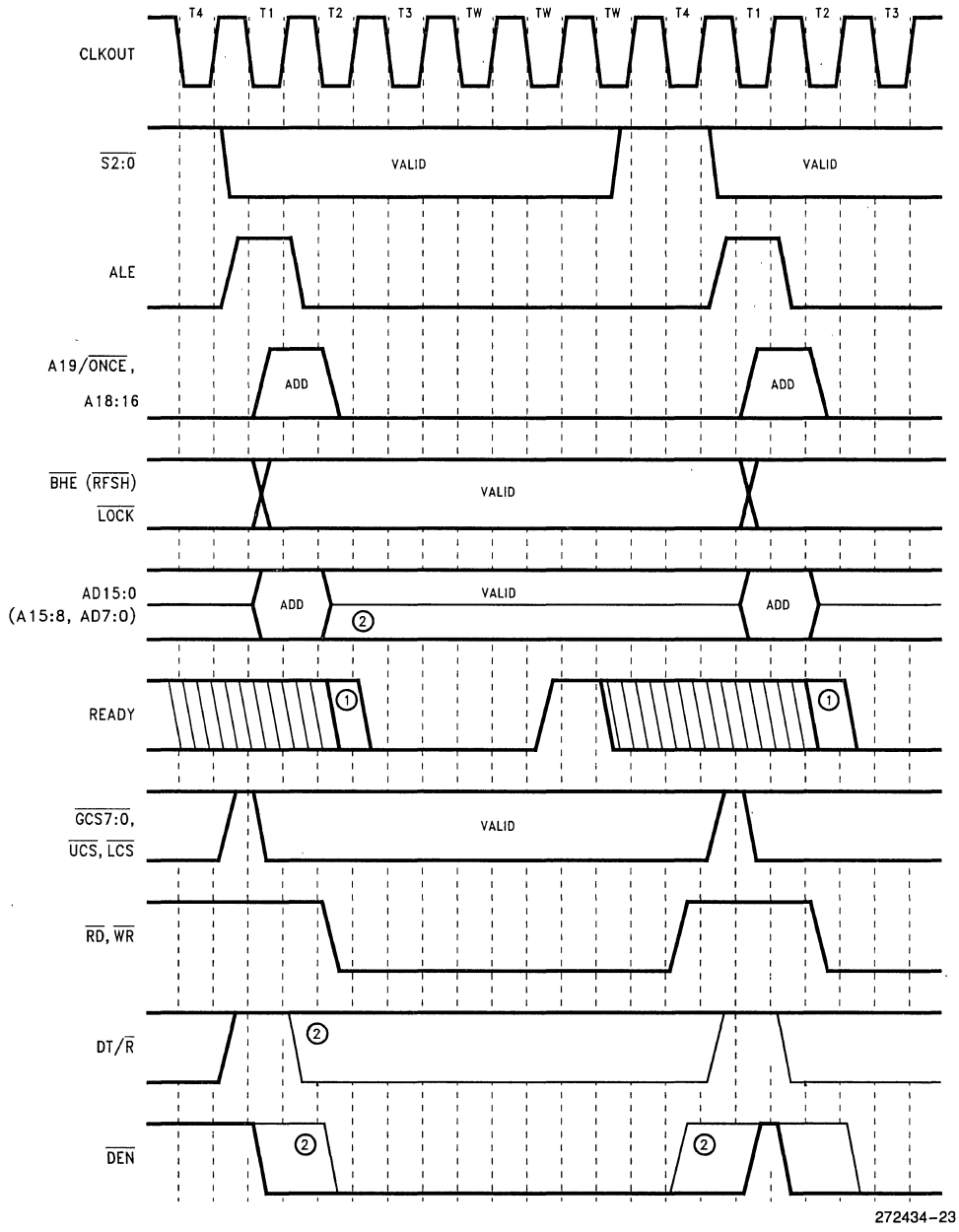
Pin names in parentheses apply to 80C188EC/80L188EC.

Figure 22. HOLD/HLDA Cycle Waveforms



Pin names in parentheses apply to 80C188EC/80L188EC.

Figure 23. Refresh during HLDA Waveforms



NOTES:
1. READY must be low by either edge to cause a wait state.
2. Lighter lines indicate READ cycles, darker lines indicate WRITE cycles.
Pin names in parentheses apply to 80C188EC/80L188EC.

Figure 24. READY Cycle Waveforms

80C186EC/80C188EC EXECUTION TIMINGS

A determination of program execution timing must consider the bus cycles necessary to prefetch instructions as well as the number of execution unit cycles necessary to execute instructions. The following instruction timings represent the **minimum** execution time in clock cycles for each instruction. The timings given are based on the following assumptions:

- The opcode, along with any data or displacement required for execution of a particular instruction, has been prefetched and resides in the queue at the time it is needed.
- No wait states or bus HOLDs occur.
- All word-data is located on even-address boundaries (80C186EC only).

All jumps and calls include the time required to fetch the opcode of the next instruction at the destination address.

All instructions which involve memory accesses can require one or two additional clocks above the minimum timings shown due to the asynchronous handshake between the bus interface unit (BIU) and execution unit.

With a 16-bit BIU, the 80C186EC has sufficient bus performance to ensure that an adequate number of prefetched bytes will reside in the queue (6 bytes) most of the time. Therefore, actual program execution time will not be substantially greater than that derived from adding the instruction timings shown.

The 80C188EC 8-bit BIU is limited in its performance relative to the execution unit. A sufficient number of prefetched bytes may not reside in the prefetch queue (4 bytes) much of the time. Therefore, actual program execution time will be substantially greater than that derived from adding the instruction timings shown.

INSTRUCTION SET SUMMARY

Function	Format				80C186EC Clock Cycles	80C188EC Clock Cycles	Comments
DATA TRANSFER							
MOV = Move:							
Register to Register/Memory	1000100w	mod reg r/m			2/12	2/12*	
Register/memory to register	1000101w	mod reg r/m			2/9	2/9*	
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1	12/13	12/13	8/16-bit
Immediate to register	1011w reg	data	data if w=1		3/4	3/4	8/16-bit
Memory to accumulator	1010000w	addr-low	addr-high		8	8*	
Accumulator to memory	1010001w	addr-low	addr-high		9	9*	
Register/memory to segment register	10001110	mod 0 reg r/m			2/9	2/13	
Segment register to register/memory	10001100	mod 0 reg r/m			2/11	2/15	
PUSH = Push:							
Memory	11111111	mod 110 r/m			16	20	
Register	01010 reg				10	14	
Segment register	000reg110				9	13	
Immediate	011010s0	data	data if s=0		10	14	
PUSHA = Push All	01100000				36	68	
POP = Pop:							
Memory	10001111	mod 000 r/m			20	24	
Register	01011 reg				10	14	
Segment register	000reg111	(reg≠01)			8	12	
POPA = Pop All	01100001				51	83	
XCHG = Exchange:							
Register/memory with register	1000011w	mod reg r/m			4/17	4/17*	
Register with accumulator	10010 reg				3	3	
IN = Input from:							
Fixed port	1110010w	port			10	10*	
Variable port	1110110w				8	8*	
OUT = Output to:							
Fixed port	1110011w	port			9	9*	
Variable port	1110111w				7	7*	
XLAT = Translate byte to AL	11010111				11	15	
LEA = Load EA to register	10001101	mod reg r/m			6	6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod≠11)		18	26	
LES = Load pointer to ES	11000100	mod reg r/m	(mod≠11)		18	26	
LAHF = Load AH with flags	10011111				2	2	
SAHF = Store AH into flags	10011110				3	3	
PUSHF = Push flags	10011100				9	13	
POPF = Pop flags	10011101				8	12	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EC Clock Cycles	80C188EC Clock Cycles	Comments
DATA TRANSFER (Continued)							
SEGMENT = Segment Override:							
CS	00101110				2	2	
SS	00110110				2	2	
DS	00111110				2	2	
ES	00100110				2	2	
ARITHMETIC							
ADD = Add:							
Reg/memory with register to either	000000dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 000 r/m	data	data if sw=01	4/16	4/16*	
Immediate to accumulator	0000010w	data	data if w=1		3/4	3/4	8/16-bit
ADC = Add with carry:							
Reg/memory with register to either	000100dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	100000sw	mod 010 r/m	data	data if sw=01	4/16	4/16*	
Immediate to accumulator	0001010w	data	data if w=1		3/4	3/4	8/16-bit
INC = Increment:							
Register/memory	1111111w	mod 000 r/m			3/15	3/15*	
Register	01000 reg				3	3	
SUB = Subtract:							
Reg/memory and register to either	001010dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 101 r/m	data	data if sw=01	4/16	4/16*	
Immediate from accumulator	0010110w	data	data if w=1		3/4	3/4*	8/16-bit
SBB = Subtract with borrow:							
Reg/memory and register to either	000110dw	mod reg r/m			3/10	3/10*	
Immediate from register/memory	100000sw	mod 011 r/m	data	data if sw=01	4/16	4/16*	
Immediate from accumulator	0001110w	data	data if w=1		3/4	3/4*	8/16-bit
DEC = Decrement							
Register/memory	1111111w	mod 001 r/m			3/15	3/15*	
Register	01001 reg				3	3	
CMP = Compare:							
Register/memory with register	0011101w	mod reg r/m			3/10	3/10*	
Register with register/memory	0011100w	mod reg r/m			3/10	3/10*	
Immediate with register/memory	100000sw	mod 111 r/m	data	data if sw=01	3/10	3/10*	
Immediate with accumulator	0011110w	data	data if w=1		3/4	3/4	8/16-bit
NEG = Change sign register/memory	1111011w	mod 011 r/m			3/10	3/10*	
AAA = ASCII adjust for add	00110111				8	8	
DAA = Decimal adjust for add	00100111				4	4	
AAS = ASCII adjust for subtract	00111111				7	7	
DAS = Decimal adjust for subtract	00101111				4	4	
MUL = Multiply (unsigned):	1111011w	mod 100 r/m					
Register-Byte					26–28	26–28	
Register-Word					35–37	35–37	
Memory-Byte					32–34	32–34	
Memory-Word					41–43	41–43*	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:

*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186EC Clock Cycles	80C188EC Clock Cycles	Comments
ARITHMETIC (Continued)				
IMUL = Integer multiply (signed):	1111011w | mod101 r/m			
Register-Byte		25–28	25–28	
Register-Word		34–37	34–37	
Memory-Byte		31–34	32–34	
Memory-Word		40–43	40–43*	
IMUL = Integer Immediate multiply (signed)	011010s1 | mod reg r/m | data | data if s=0	22–25/ 29–32	22–25/ 29–32	
DIV = Divide (unsigned):	1111011w | mod110 r/m			
Register-Byte		29	29	
Register-Word		38	38	
Memory-Byte		35	35	
Memory-Word		44	44*	
IDIV = Integer divide (signed):	1111011w | mod111 r/m			
Register-Byte		44–52	44–52	
Register-Word		53–61	53–61	
Memory-Byte		50–58	50–58	
Memory-Word		59–67	59–67*	
AAM = ASCII adjust for multiply	11010100 | 00001010	19	19	
AAD = ASCII adjust for divide	11010101 | 00001010	15	15	
CBW = Convert byte to word	10011000	2	2	
CWD = Convert word to double word	10011001	4	4	
LOGIC				
Shift/Rotate Instructions:				
Register/Memory by 1	1101000w | mod TTT r/m	2/15	2/15	
Register/Memory by CL	1101001w | mod TTT r/m	5+n/17+n	5+n/17+n	
Register/Memory by Count	1100000w | mod TTT r/m | count	5+n/17+n	5+n/17+n	
	TTT Instruction 000 ROL 001 ROR 010 RCL 011 RCR 100 SHL/SAL 101 SHR 111 SAR			
AND = **And:**				
Reg/memory and register to either	001000dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod100 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0010010w | data | data if w=1	3/4	3/4*	8/16-bit
TEST = **And function to flags, no result:**				
Register/memory and register	1000010w | mod reg r/m	3/10	3/10	
Immediate data and register/memory	1111011w | mod000 r/m | data | data if w=1	4/10	4/10*	
Immediate data and accumulator	1010100w | data | data if w=1	3/4	3/4	8/16-bit
OR = **Or:**				
Reg/memory and register to either	000010dw | mod reg r/m	3/10	3/10*	
Immediate to register/memory	1000000w | mod001 r/m | data | data if w=1	4/16	4/16*	
Immediate to accumulator	0000110w | data | data if w=1	3/4	3/4*	8/16-bit

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EC Clock Cycles	80C188EC Clock Cycles	Comments
LOGIC (Continued)							
XOR = Exclusive or:							
Reg/memory and register to either	001100dw	mod reg r/m			3/10	3/10*	
Immediate to register/memory	1000000w	mod 110 r/m	data	data if w=1	4/16	4/16*	
Immediate to accumulator	0011010w	data	data if w=1		3/4	3/4	8/16-bit
NOT = Invert register/memory	1111011w	mod 010 r/m			3/10	3/10*	
STRING MANIPULATION							
MOVS = Move byte/word	1010010w				14	14*	
CMPS = Compare byte/word	1010011w				22	22*	
SCAS = Scan byte/word	1010111w				15	15*	
LODS = Load byte/wd to AL/AX	1010110w				12	12*	
STOS = Store byte/wd from AL/AX	1010101w				10	10*	
INS = Input byte/wd from DX port	0110110w				14	14	
OUTS = Output byte/wd to DX port	0110111w				14	14	
Repeated by count in CX (REP/REPE/REPZ/REPNE/REPNZ)							
MOVS = Move string	11110010	1010010w			8+8n	8+8n*	
CMPS = Compare string	1111001z	1010011w			5+22n	5+22n*	
SCAS = Scan string	1111001z	1010111w			5+15n	5+15n*	
LODS = Load string	11110010	1010110w			6+11n	6+11n*	
STOS = Store string	11110010	1010101w			6+9n	6+9n*	
INS = Input string	11110010	0110110w			8+8n	8+8n*	
OUTS = Output string	11110010	0110111w			8+8n	8+8n*	
CONTROL TRANSFER							
CALL = Call:							
Direct within segment	11101000	disp-low	disp-high		15	19	
Register/memory indirect within segment	11111111	mod 010 r/m			13/19	17/27	
Direct intersegment	10011010	segment offset			23	31	
		segment selector					
Indirect intersegment	11111111	mod 011 r/m	(mod ≠ 11)		38	54	
JMP = Unconditional jump:							
Short/long	11101011	disp-low			14	14	
Direct within segment	11101001	disp-low	disp-high		14	14	
Register/memory indirect within segment	11111111	mod 100 r/m			11/17	11/21	
Direct intersegment	11101010	segment offset			14	14	
		segment selector					
Indirect intersegment	11111111	mod 101 r/m	(mod ≠ 11)		26	34	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format				80C186EC Clock Cycles	80C188EC Clock Cycles	Comments
CONTROL TRANSFER (Continued)							
RET = **Return from CALL:**							
Within segment	11000011				16	20	
Within seg adding immed to SP	11000010	data-low	data-high		18	22	
Intersegment	11001011				22	30	
Intersegment adding immediate to SP	11001010	data-low	data-high		25	33	
JE/JZ = Jump on equal/zero	01110100	disp			4/13	4/13	JMP not taken/JMP taken
JL/JNGE = Jump on less/not greater or equal	01111100	disp			4/13	4/13	
JLE/JNG = Jump on less or equal/not greater	01111110	disp			4/13	4/13	
JB/JNAE = Jump on below/not above or equal	01110010	disp			4/13	4/13	
JBE/JNA = Jump on below or equal/not above	01110110	disp			4/13	4/13	
JP/JPE = Jump on parity/parity even	01111010	disp			4/13	4/13	
JO = Jump on overflow	01110 000	disp			4/13	4/13	
JS = Jump on sign	01111000	disp			4/13	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101	disp			4/13	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101	disp			4/13	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111	disp			4/13	4/13	
JNB/JAE = Jump on not below/above or equal	01110011	disp			4/13	4/13	
JNBE/JA = Jump on not below or equal/above	01110111	disp			4/13	4/13	
JNP/JPO = Jump on not par/par odd	01111011	disp			4/13	4/13	
JNO = Jump on not overflow	01110001	disp			4/13	4/13	
JNS = Jump on not sign	01111001	disp			4/13	4/13	
JCXZ = Jump on CX zero	11100011	disp			5/15	5/15	
LOOP = Loop CX times	11100010	disp			6/16	6/16	LOOP not taken/LOOP taken
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp			6/16	6/16	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp			6/16	6/16	
ENTER = Enter Procedure	11001000	data-low	data-high	L			
L = 0					15	19	
L = 1					25	29	
L > 1					22+16(n−1)	26+20(n−1)	
LEAVE = Leave Procedure	11001001				8	8	
INT = **Interrupt:**							
Type specified	11001101	type			47	47	
Type 3	11001100				45	45	if INT. taken/ if INT. not taken
INTO = Interrupt on overflow	11001110				48/4	48/4	
IRET = Interrupt return	11001111				28	28	
BOUND = Detect value out of range	01100010	mod reg r/m			33–35	33–35	

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for all memory transfers.

INSTRUCTION SET SUMMARY (Continued)

Function	Format	80C186EC Clock Cycles	80C188EC Clock Cycles	Comments
PROCESSOR CONTROL				
CLC = Clear carry	11111000	2	2	
CMC = Complement carry	11110101	2	2	
STC = Set carry	11111001	2	2	
CLD = Clear direction	11111100	2	2	
STD = Set direction	11111101	2	2	
CLI = Clear interrupt	11111010	2	2	
STI = Set interrupt	11111011	2	2	
HLT = Halt	11110100	2	2	
WAIT = Wait	10011011	6	6	if $\overline{TEST}$ = 0
LOCK = Bus lock prefix	11110000	2	2	
NOP = No Operation	10010000	3	3	
	(TTT LLL are opcode to processor extension)			

Shaded areas indicate instructions not available in 8086/8088 microsystems.

NOTE:
*Clock cycles shown for byte transfers, for word operations, add 4 clock cycles for all memory transfers.

The Effective Address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

EA calculation time is 4 clock cycles for all modes, and is included in the execution times given whenever appropriate.

Segment Override Prefix

0	0	1	reg	1	1	0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

ERRATA

An 80C186EC/80L186EC with a STEPID value of 0002H has no known errata. A device with a STEPID of 0002H can be visually identified by noting the **presence** of an "A" or "B" alpha character next to the FPO number or the absence of any alpha character. The FPO number location is shown in Figures 4, 5 and 6.

REVISION HISTORY

This data sheet replaces the following data sheets:

272072-003	80C186EC
272076-003	80C188EC
272332-001	80L186EC
272333-001	80L188EC
272373-001	SB80C188EC/SB80L188EC
272372-001	SB80C186EC/SB80L186EC

80C187
80-BIT MATH COPROCESSOR

- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Upward Object-Code Compatible from 8087**
- **Fully Compatible with 387DX and 387SX Math Coprocessors. Implements all 387 Architectural Enhancements over 8087**
- **Directly Interfaces with 80C186 CPU**
- **80C186/80C187 Provide a Software/Binary Compatible Upgrade from 80186/82188/8087 Systems**
- **Expands 80C186's Data Types to Include 32-, 64-, 80-Bit Floating-Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Directly Extends 80C186's Instruction Set to Trigonometric, Logarithmic, Exponential, and Arithmetic Instructions for All Data Types**
- **Full-Range Transcendental Operations for SINE, COSINE, TANGENT, ARCTANGENT, and LOGARITHM**
- **Built-In Exception Handling**
- **Eight 80-Bit Numeric Registers, Usable as Individually Addressable General Registers or as a Register Stack**
- **Available in 40-Pin CERDIP and 44-Pin PLCC Package**

  (See Packaging Outlines and Dimensions, Order #231369)

The Intel 80C187 is a high-performance math coprocessor that extends the architecture of the 80C186 with floating-point, extended integer, and BCD data types. A computing system that includes the 80C187 fully conforms to the IEEE Floating-Point Standard. The 80C187 adds over seventy mnemonics to the instruction set of the 80C186, including support for arithmetic, logarithmic, exponential, and trigonometric mathematical operations. The 80C187 is implemented with 1.5 micron, high-speed CHMOS III technology and packaged in both a 40-pin CERDIP and a 44-pin PLCC package. The 80C187 is upward object-code compatible from the 8087 math coprocessor and will execute code written for the 80387DX and 80387SX math coprocessors.

November 1992
Order Number: 270640-004

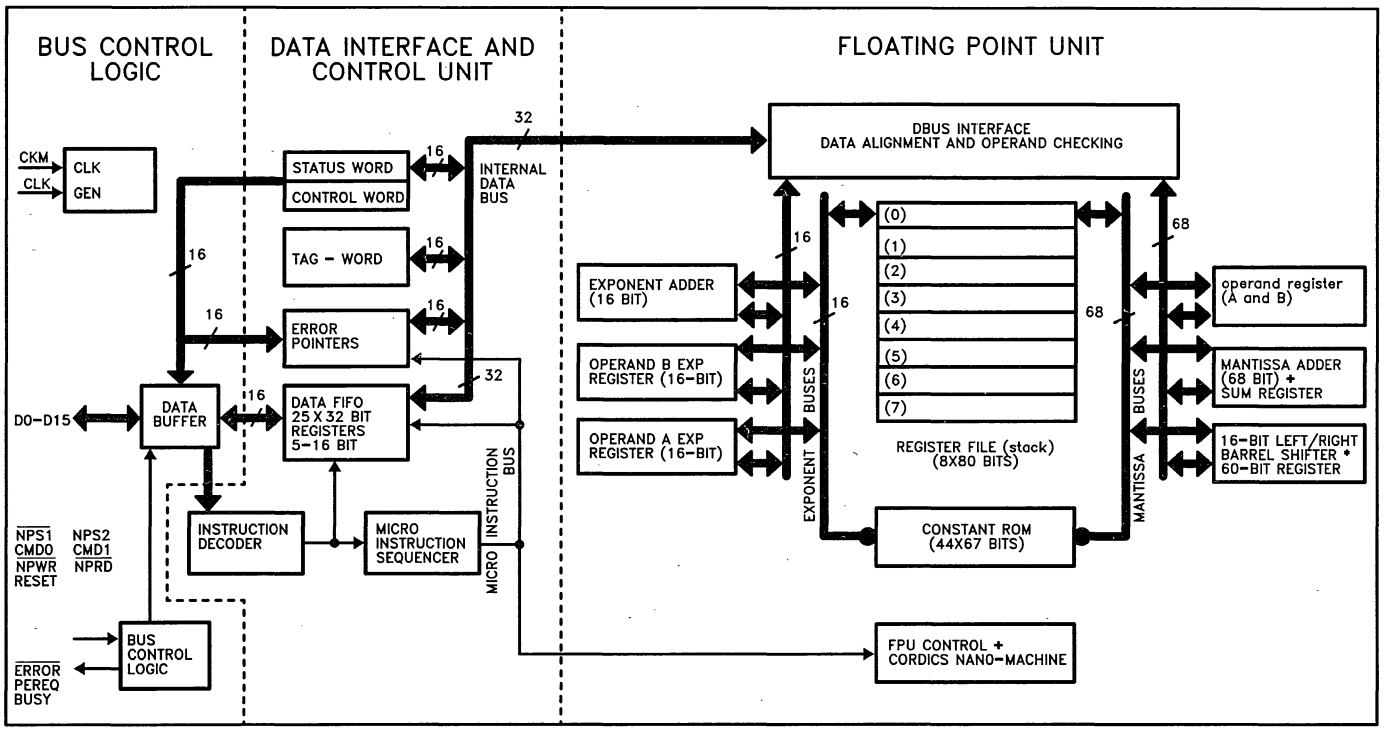


Figure 1. 80C187 Block Diagram

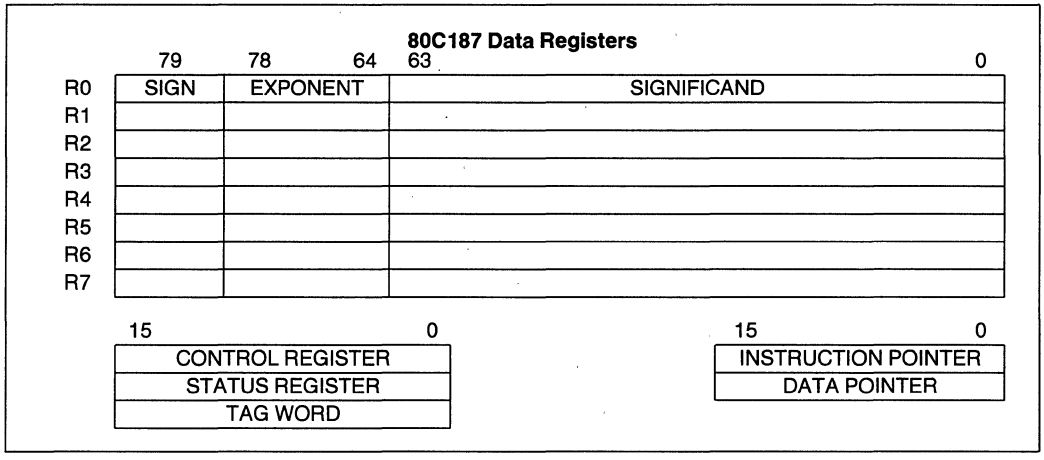


Figure 2. Register Set

FUNCTIONAL DESCRIPTION

The 80C187 Math Coprocessor provides arithmetic instructions for a variety of numeric data types. It also executes numerous built-in transcendental functions (e.g. tangent, sine, cosine, and log functions). The 80C187 effectively extends the register and instruction set of the 80C186 CPU for existing data types and adds several new data types as well. Figure 2 shows the additional registers visible to programs in a system that includes the 80C187. Essentially, the 80C187 can be treated as an additional resource or an extension to the CPU. The 80C186 CPU together with an 80C187 can be used as a single unified system.

A 80C186 system that includes the 80C187 is completely upward compatible with software for the 8086/8087.

The 80C187 interfaces only with the 80C186 CPU. The interface hardware for the 80C187 is not implemented on the 80C188.

PROGRAMMING INTERFACE

The 80C187 adds to the CPU additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. All new instructions and data types are directly supported by the assembler and compilers for high-level languages. The 80C187 also supports the full 80387DX instruction set.

All communication between the CPU and the 80C187 is transparent to applications software. The CPU automatically controls the 80C187 whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the 80C187. All memory addressing modes are available for addressing numerics operands.

The end of this data sheet lists by class the instructions that the 80C187 adds to the instruction set.

NOTE:
The 80C187 Math Coprocessor is also referred to as a Numeric Processor Extension (NPX) in this document.

Data Types

Table 1 lists the seven data types that the 80C187 supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at even physical-memory addresses; operands may begin at odd addresses, but will require extra memory cycles to access the entire operand.

Internally, the 80C187 holds all numbers in the extended-precision real format. Instructions that load operands from memory automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating-point numbers, or 18-digit packed BCD numbers into extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

Numeric Operands

A typical NPX instruction accepts one or two operands and produces one (or sometimes two) results. In two-operand instructions, one operand is the contents of an NPX register, while the other may be a memory location. The operands of some instructions are predefined; for example, FSQRT always takes the square root of the number in the top stack element (refer to the section on Data Registers).

Register Set

Figure 2 shows the 80C187 register set. When an 80C187 is present in a system, programmers may use these registers in addition to the registers normally available on the CPU.

DATA REGISTERS

80C187 computations use the extended-precision real data type.

Table 1. Data Type Representation in Memory

Data Formats	Range	Precision	Most Significant Byte ... HIGHEST ADDRESSED BYTE
Word Integer	$\pm 10^4$	16 Bits	(TWO S COMPLEMENT)
Short Integer	$\pm 10^9$	32 Bits	(TWO S COMPLEMENT)
Long Integer	$\pm 10^{18}$	64 Bits	(TWO S COMPLEMENT)
Packed BCD	$\pm 10^{18}$	18 Digits	S, X, MAGNITUDE
Single Precision	$\pm 10^{\pm 38}$	24 Bits	S, BIASED EXPONENT, SIGNIFICAND
Double Precision	$\pm 10^{\pm 308}$	53 Bits	S, BIASED EXPONENT, SIGNIFICAND
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits	S, BIASED EXPONENT, I, SIGNIFICAND

270640-2

NOTES:
1. S = Sign bit (0 = Positive, 1 = Negative)
2. d_n = Decimal digit (two per byte)
3. X = Bits have no significance; 80C187 ignores when loading, zeros when storing
4. ▲ = Position of implicit binary point
5. I = Integer bit of significand; stored in temporary real, implicit in single and double precision
6. Exponent Bias (normalized values):
   Single: 127 (7FH)
   Double: 1023 (3FFH)
   Extended Real: 16383 (3FFFH)
7. Packed BCD: $(-1)^S (D_{17} \ldots D_0)$
8. Real: $(-1)^S (2^{E\text{-BIAS}}) (F_0, F_1 \ldots)$

The 80C187 register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as individually addressable registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by one. The 80C187 register stack grows "down" toward lower-addressed registers.

Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to use. This explicit addressing is also relative to TOP.

TAG WORD

The tag word marks the content of each numeric data register, as Figure 3 shows. Each two-bit tag represents one of the eight data registers. The principal function of the tag word is to optimize the NPX's performance and stack handling by making it possible to distinguish between empty and nonempty register locations. It also enables exception handlers to identify special values (e.g. NaNs or denormals) in the contents of a stack location without the need to perform complex decoding of the actual data.

STATUS WORD

The 16-bit status word (in the status register) shown in Figure 4 reflects the overall state of the 80C187. It may be read and inspected by programs.

Bit 15, the B-bit (busy bit) is included for 8087 compatibility only. It always has the same value as the ES bit (bit 7 of the status word); it does **not** indicate the status of the BUSY output of 80C187.

Bits 13–11 (TOP) point to the 80C187 register that is the current top-of-stack.

The four numeric condition code bits (C_3–C_0) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of these instructions on the condition code are summarized in Tables 2 through 5.

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the $\overline{\text{ERROR}}$ signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C_1) distinguishes between stack overflow (C_1 = 1) and underflow (C_1 = 0).

Figure 4 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the 80C187 has detected an exception while executing an instruction. A later section entitled "Exception Handling" explains how they are set and used.

Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the value of ES (bit 7) and its reflection in the B-bit (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the $\overline{\text{ERROR}}$ output of the 80C187 is activated immediately.

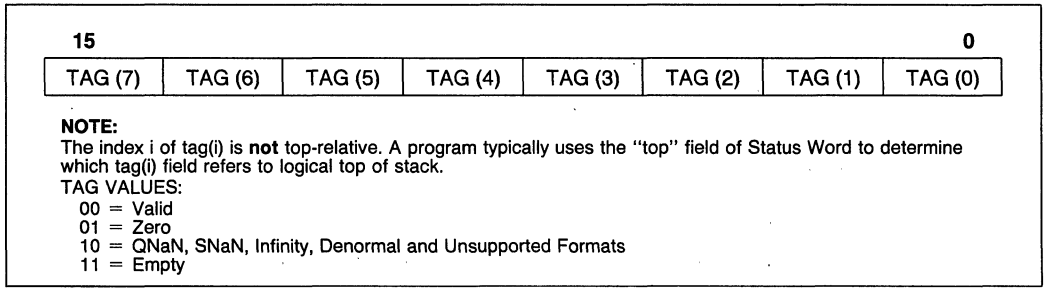
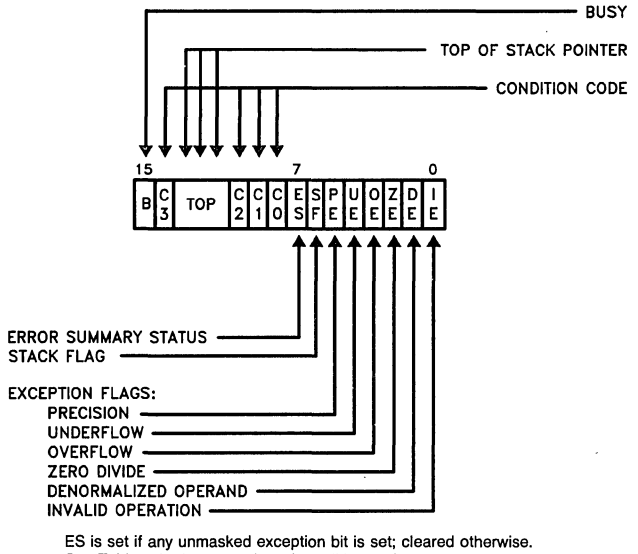


Figure 3. Tag Word



270640-3

ES is set if any unmasked exception bit is set; cleared otherwise.
See Table 2 for interpretation of condition code.
TOP values:
000 = Register 0 is Top of Stack
001 = Register 1 is Top of Stack
•
•
•
111 = Register 7 is Top of Stack
For definitions of exceptions, refer to the section entitled, "Exception Handling"

Figure 4. Status Word

CONTROL WORD

The NPX provides several processing options that are selected by loading a control word from memory into the control register. Figure 5 shows the format and encoding of fields in the control word.

Table 2. Condition Code Interpretation

Instruction	C0(S)	C3(Z)	C1(A)	C2(C)
FPREM, FPREM1 (See Table 3)	Three Least Significant Bits of Quotient Q2 Q0		Q1 or O/$\overline{U}$	Reduction 0 = Complete 1 = Incomplete
FCOM, FCOMP, FCOMPP, FTST FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of Comparison (See Table 4)		Zero or O/$\overline{U}$	Operand is not Comparable (Table 4)
FXAM	Operand Class (See Table 5)		Sign or O/$\overline{U}$	Operand Class (Table 5)
FCHS, FABS, FXCH, FINCSTP, FDECSTP, Constant Loads, FXTRACT, FLD, FILD, FBLD, FSTP (Ext Real)	UNDEFINED		Zero or O/$\overline{U}$	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/$\overline{U}$	UNDEFINED
FPTAN, FSIN, FCOS, FSINCOS	UNDEFINED		Roundup or O/$\overline{U}$, Undefined if C2 = 1	Reduction 0 = Complete 1 = Incomplete
FLDENV, FRSTOR	Each Bit Loaded from Memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

O/$\overline{U}$ When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).

Reduction If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.

Roundup When the PE bit of the status word is set, this bit indicates whether one was added to the least significant bit of the result during the last rounding.

UNDEFINED Do not rely on finding any specific value in these bits.

The low-order byte of this control word configures exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the 80C187 recognizes.

The high-order byte of the control word configures the 80C187 operating mode, including precision, rounding, and infinity control.

- The "infinity control bit" (bit 12) is not meaningful to the 80C187, and programs must ignore its value. To maintain compatibility with the 8087, this bit can be programmed; however, regardless of its value, the 80C187 always treats infinity in the affine sense ($-\infty < +\infty$). This bit is initialized to zero both after a hardware reset and after the FINIT instruction.
- The rounding control (RC) bits (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS, and FCHS), and all transcendental instructions.
- The precision control (PC) bits (bits 9–8) can be used to set the 80C187 internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

Table 3. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: Further Iteration Required for Complete Reduction	
	Q1	Q0	Q2	Q MOD 8	
0	0	0	0	0	Complete Reduction: C0, C3, C1 Contain Three Least Significant Bits of Quotient
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 4. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 5. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	1	− Denormal

INSTRUCTION AND DATA POINTERS

Because the NPX operates in parallel with the CPU, any exceptions detected by the NPX may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the failing numerics instruction, the 80C187 contains registers that aid in diagnosis. These registers supply the opcode of the failing numerics instruction, the address of the instruction, and the address of its numerics memory operand (if appropriate).

The instruction and data pointers are provided for user-written exception handlers. Whenever the 80C187 executes a new ESC instruction, it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if present), and the opcode.

The instruction and data pointers appear in the format shown by Figure 6. The ESC instruction FLDENV, FSTENV, FSAVE and FRSTOR are used to transfer these values between the registers and memory. Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

Interrupt Description

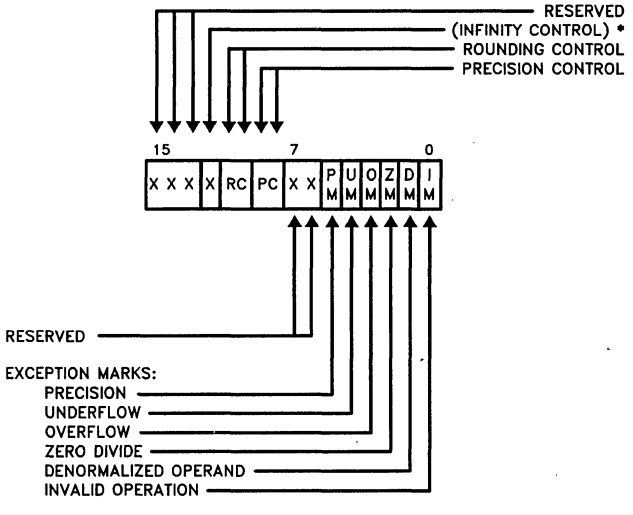
CPU interrupt 16 is used to report exceptional conditions while executing numeric programs. Interrupt 16 indicates that the previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC instructions can cause this interrupt. The CPU return address pushed onto the stack of the exception handler points to an ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the NPX. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.

Exception Handling

The 80C187 detects six different exception conditions that can occur during instruction execution. Table 6 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the 80C187 if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the $\overline{\text{ERROR}}$ signal. When the CPU attempts to execute another ESC instruction, interrupt 16 occurs. The exception condition must be resolved via an interrupt service routine. The return address pushed onto the CPU stack upon entry to the service routine does not necessarily point to the failing instruction nor to the following instruction. The 80C187 saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

If error trapping is required at the end of a series of numerics instructions (specifically, when the last ESC instruction modifies memory data and that data is used in subsequent nonnumerics instructions), it is necessary to insert the FNOP instruction to force the 80C187 to check its $\overline{\text{ERROR}}$ input.



Precision Control
00— 24 Bits (Single Precision)
01— (Reserved)
10— 53 Bits (Double Precision)
11— 64 Bits (Extended Precision)

Rounding Control
00— Round to Nearest or Even
01— Round Down (toward $-\infty$)
10— Round Up (toward $+\infty$)
11— Chop (Truncate toward Zero)

*The "infinity control" bit is not meaningful to the 80C187. To maintain compatibility with the 8087, this bit can be programmed; however, regardless of its value, the 80C187 treats infinity in the affine sense ($-\infty < +\infty$).

Figure 5. Control Word

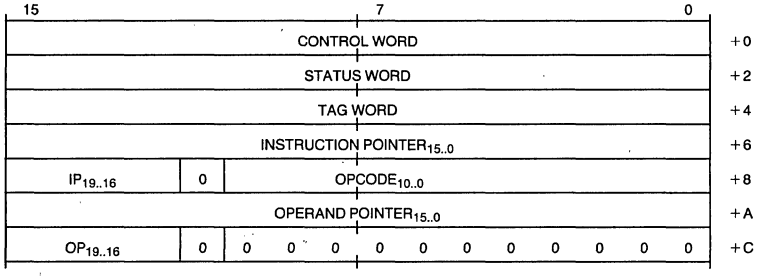


Figure 6. Instruction and Data Pointer Image in Memory

Table 6. Exceptions

Exception	Cause	Default Action (If Exception is Masked)
Invalid Operation	Operation on a signalling NaN, unsupported format, indeterminate form (0*∞, 0/0), (+∞) + (−∞), etc.), or stack overflow/underflow (SF is also set)	Result is a quiet NaN, integer indefinite, or BCD indefinite
Denormalized Operand	At least one of the operands is denormalized, i.e. it has the smallest exponent but a nonzero significand	The operand is normalized, and normal processing continues
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format	Result is largest finite value or ∞
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g. 1/3); the result is rounded according to the rounding mode	Normal processing continues

Initialization

After FNINIT or RESET, the control word contains the value 037FH (all exceptions masked, precision control 64 bits, rounding to nearest) the same values as in an 8087 after RESET. For compatibility with the 8087, the bit that used to indicate infinity control (bit 12) is set to zero; however, regardless of its setting, infinity is treated in the affine sense. After FNINIT or RESET, the status word is initialized as follows:

- All exceptions are set to zero.
- Stack TOP is zero, so that after the first push the stack top will be register seven (111B).
- The condition code C_3–C_0 is **undefined**.
- The B-bit is zero.

The tag word contains FFFFH (all stack locations are empty).

80C186/80C187 initialization software should execute an FNINIT instruction (i.e. an FINIT without a preceding WAIT) after RESET. The FNINIT is not strictly required for 80C187 software, but Intel recommends its use to help ensure upward compatibility with other processors.

8087 Compatibility

This section summarizes the differences between the 80C187 and the 8087. Many changes have been designed into the 80C187 to directly support the IEEE standard in hardware. These changes result in increased performance by elminating the need for software that supports the standard.

GENERAL DIFFERENCES

The 8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the 80C187 Numeric Processor Extension. They do not alter the state of the 80C187 Numeric Processor Extension. (They are treated similarly to FNOP, except that $\overline{\text{ERROR}}$ is not checked.) While 8086/8087 code containing these instructions can be executed on the 80C186/80C187, it is unlikely that the exception-handling routines containing these instructions will be completely portable to the 80C187 Numeric Processor Extension.

The 80C187 differs from the 8087 with respect to instruction, data, and exception synchronization. Except for the processor control instructions, all of the 80C187 numeric instructions are automatically synchronized by the 80C186 CPU. When necessary, the

80C186 automatically tests the BUSY line from the 80C187 Numeric Processor Extension to ensure that the 80C187 Numeric Processor Extension has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the 8087 used with 8086 and 8088 CPUs, explicit WAITs are required before each numeric instruction to ensure synchronization. Although 8086/8087 programs having explicit WAIT instructions will execute on the 80C186/80C187, these WAIT instructions are unnecessary.

The 80C187 supports only affine closure for infinity arithmetic, not projective closure.

Operands for FSCALE and FPATAN are no longer restricted in range (except for $\pm\infty$); F2XM1 and FPTAN accept a wider range of operands.

Rounding control is in effect for FLD *constant*.

Software cannot change entries of the tag word to values (other than empty) that differ from actual register contents.

After reset, FINIT, and incomplete FPREM, the 80C187 resets to zero the condition code bits C_3–C_0 of the status word.

In conformance with the IEEE standard, the 80C187 does not support the special data formats pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

The denormal exception has a different purpose on the 80C187. A system that uses the denormal-exception handler solely to normalize the denormal operands, would better mask the denormal exception on the 80C187. The 80C187 automatically normalizes denormal operands when the denormal exception is masked.

EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the 80C186/80C187:

1. The 80C186/80C187 traps exceptions only on the next ESC instruction; i.e. the 80C186 does not notice unmasked 80C187 exceptions on the 80C186 $\overline{\text{ERROR}}$ input line until a later numerics instruction is executed. Because the 80C186 does not sample $\overline{\text{ERROR}}$ on WAIT and FWAIT instructions, programmers should place an FNOP instruction at the end of a sequence of numerics instructions to force the 80C186 to sample its $\overline{\text{ERROR}}$ input.
2. The 80C187 Numeric Processor Extension signals exceptions through a dedicated $\overline{\text{ERROR}}$ line to the CPU. The 80C187 error signal does not pass through an interrupt controller (the 8087 INT signal does). Therefore, any interrupt-controller-oriented instructions in numerics exception handlers for the 8086/8087 should be deleted.
3. Interrupt vector 16 must point to the numerics exception handling routine.
4. The ESC instruction address saved in the 80C187 Numeric Processor Extension includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
5. When the overflow or underflow exception is masked, the 80C187 differs from the 8087 in rounding when overflow or underflow occurs. The 80C187 produces results that are consistent with the rounding mode.
6. When the underflow exception is masked, the 80C187 sets its underflow flag only if there is also a loss of accuracy during denormalization.
7. Fewer invalid-operation exceptions due to denormal operands, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
8. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
9. The denormal exception can occur during the transcendental instructions and the FXTRACT instruction.
10. The denormal exception no longer takes precedence over all other exceptions.
11. When the denormal exception is masked, the 80C187 automatically normalizes denormal operands. The 8087 performs unnormal arithmetic, which might produce an unnormal result.
12. When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
13. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
14. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
15. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized oper-

and exception. When loading a signalling NaN, FLD *single/double precision* signals an invalid-operand exception.

16. The 80C187 only generates quiet NaNs (as on the 8087); however, the 80C187 distinguishes between quiet NaNs and signalling NaNs. Signalling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
17. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 8087 leaves the original operand in ST(1) intact.
18. When the scaling factor is $\pm\infty$, the FSCALE (ST(0), ST(1) instruction behaves as follows (ST(0) and ST(1) contain the scaled and scaling operands respectively):
    - FSCALE (0, ∞) generates the invalid operation exception.
    - FSCALE (finite, $-\infty$) generates zero with the same sign as the scaled operand.
    - FSCALE (finite, $+\infty$) generates ∞ with the same sign as the scaled operand.

    The 8087 returns zero in the first case and raises the invalid-operation exception in the other cases.
19. The 80C187 returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 supports a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

Table 7. Pin Summary

Pin Name	Function	Active State	Input/ Output
CLK	CLocK		I
CKM	ClocKing Mode		I
RESET	System reset	High	I
PEREQ	Processor Extension REQuest	High	O
BUSY	Busy status	High	O
$\overline{\text{ERROR}}$	Error status	Low	O
D_{15}–D_0	Data pins	High	I/O
$\overline{\text{NPRD}}$	Numeric Processor ReaD	Low	I
$\overline{\text{NPWR}}$	Numeric Processor WRite	Low	I
$\overline{\text{NPS1}}$	NPX select #1	Low	I
NPS2	NPX select #2	High	I
CMD0	CoMmanD 0	High	I
CMD1	CoMmanD 1	High	I
V_{CC}	System power		I
V_{SS}	System ground		I

HARDWARE INTERFACE

In the following description of hardware interface, an overbar above a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no overbar is present above the signal name, the signal is asserted when at the high voltage level.

Signal Description

In the following signal descriptions, the 80C187 pins are grouped by function as follows:

1. Execution Control— CLK, CKM, RESET
2. NPX Handshake— PEREQ, BUSY, $\overline{\text{ERROR}}$
3. Bus Interface Pins— D_{15}–D_0, $\overline{\text{NPWR}}$, $\overline{\text{NPRD}}$
4. Chip/Port Select— $\overline{\text{NPS1}}$, NPS2, CMD0, CMD1
5. Power Supplies— V_{CC}, V_{SS}

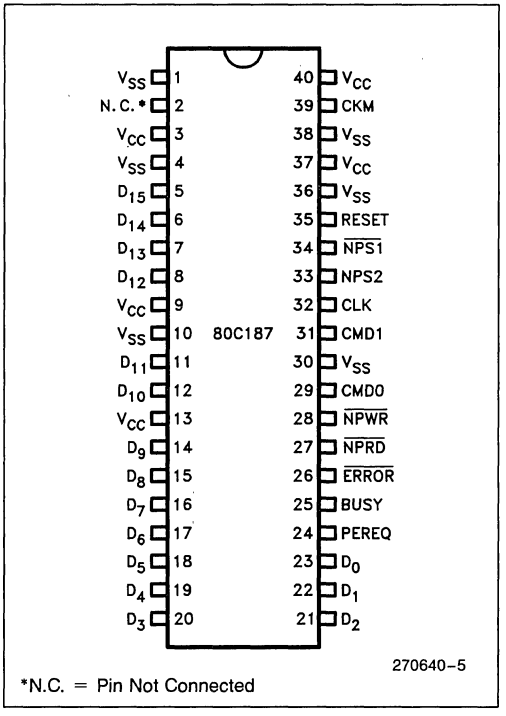
Table 7 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics. Figure 7 shows the locations of pins on the CERDIP package, while Figure 8 shows the locations of pins on the PLCC package. Table 8 helps to locate pin identifiers in Figures 7 and 8.

Clock (CLK)

This input provides the basic timing for internal operation. This pin does not require MOS-level input; it will operate at either TTL or MOS levels up to the maximum allowed frequency. A minimum frequency must be provided to keep the internal logic properly functioning. Depending on the signal on CKM, the signal on CLK can be divided by two to produce the internal clock signal (in which case CLK may be up to 32 MHz in frequency), or can be used directly (in which case CLK may be up to 12.5 MHz).

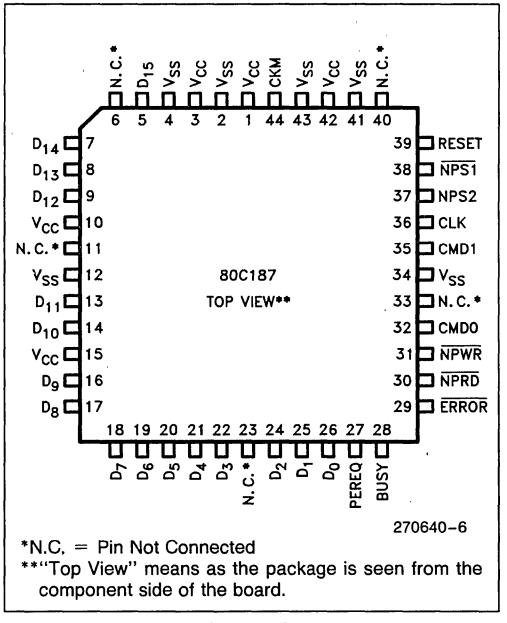
Clocking Mode (CKM)

This pin is a strapping option. When it is strapped to V_{CC} (HIGH), the CLK input is used directly; when strapped to V_{SS} (LOW), the CLK input is divided by two to produce the internal clock signal. During the RESET sequence, this input must be stable at least four internal clock cycles (i.e. CLK clocks when CKM is HIGH; 2 × CLK clocks when CKM is LOW) before RESET goes LOW.



*N.C. = Pin Not Connected

Figure 7. CERDIP Pin Configuration



*N.C. = Pin Not Connected
**"Top View" means as the package is seen from the component side of the board.

Figure 8. PLCC Pin Configuration

Table 8. PLCC Pin Cross-Reference

Pin Name	CERDIP Package	PLCC Package
BUSY	25	28
CKM	39	44
CLK	32	36
CMD0	29	32
CMD1	31	35
D_0	23	26
D_1	22	25
D_2	21	24
D_3	20	22
D_4	19	21
D_5	18	20
D_6	17	19
D_7	16	18
D_8	15	17
D_9	14	16
D_{10}	12	14
D_{11}	11	13
D_{12}	8	9
D_{13}	7	8
D_{14}	6	7
D_{15}	5	5
$\overline{\text{ERROR}}$	26	29
No Connect	2	6, 11, 23, 33, 40
$\overline{\text{NPRD}}$	27	30
$\overline{\text{NPS1}}$	34	38
NPS2	33	37
$\overline{\text{NPWR}}$	28	31
PEREQ	24	27
RESET	35	39
V_{CC}	3, 9, 13, 37, 40	1, 3, 10, 15, 42
V_{SS}	1, 4, 10, 30, 36, 38	2, 4, 12, 34, 41, 43

System Reset (RESET)

A LOW to HIGH transition on this pin causes the 80C187 to terminate its present activity and to enter a dormant state. RESET must remain active (HIGH) for at least four internal clock periods. (The relation of the internal clock period to CLK depends on CLKM; the internal clock may be different from that of the CPU.) Note that the 80C187 is active internally for 25 clock periods after the termination of the RESET signal (the HIGH to LOW transition of RESET); therefore, the first instruction should not be written to the 80C187 until 25 internal clocks after the falling edge of RESET. Table 9 shows the status of the output pins during the reset sequence. After a reset, all output pins return to their inactive states.

Table 9. Output Pin Status during Reset

Output Pin Name	Value during Reset
BUSY	HIGH
$\overline{\text{ERROR}}$	HIGH
PEREQ	LOW
D_{15}–D_0	TRI-STATE OFF

Processor Extension Request (PEREQ)

When active, this pin signals to the CPU that the 80C187 is ready for data transfer to/from its data FIFO. When there are more than five data transfers,

PEREQ is deactivated after the first three transfers and subsequently after every four transfers. This signal always goes inactive before BUSY goes inactive.

Busy Status (BUSY)

When active, this pin signals to the CPU that the 80C187 is currently executing an instruction. This pin is active HIGH. It should be connected to the 80C186's $\overline{\text{TEST}}$/BUSY pin. During the RESET sequence this pin is HIGH. The 80C186 uses this HIGH state to detect the presence of an 80C187.

Error Status ($\overline{\text{ERROR}}$)

This pin reflects the ES bit of the status register. When active, it indicates that an unmasked exception has occurred. This signal can be changed to inactive state only by the following instructions (without a preceding WAIT): FNINIT, FNCLEX, FNSTENV, FNSAVE, FLDCW, FLDENV, and FRSTOR. This pin should be connected to the $\overline{\text{ERROR}}$ pin of the CPU. $\overline{\text{ERROR}}$ can change state only when BUSY is active.

Data Pins (D_{15}–D_0)

These bidirectional pins are used to transfer data and opcodes between the CPU and 80C187. They are normally connected directly to the corresponding CPU data pins. Other buffers/drivers driving the local data bus must be disabled when the CPU reads from the NPX. High state indicates a value of one. D_0 is the least significant data bit.

Numeric Processor Write ($\overline{\text{NPWR}}$)

A signal on this pin enables transfers of data from the CPU to the NPX. This input is valid only when $\overline{\text{NPS1}}$ and NPS2 are both active.

Numeric Processor Read ($\overline{\text{NPRD}}$)

A signal on this pin enables transfers of data from the NPX to the CPU. This input is valid only when $\overline{\text{NPS1}}$ and NPS2 are both active.

Numeric Processor Selects ($\overline{\text{NPS1}}$ and NPS2)

Concurrent assertion of these signals indicates that the CPU is performing an escape instruction and enables the 80C187 to execute that instruction. No data transfer involving the 80C187 occurs unless the device is selected by these lines.

Command Selects (CMD0 and CMD1)

These pins along with the select pins allow the CPU to direct the operation of the 80C187.

System Power (V_{CC})

System power provides the +5V ±10% DC supply input. All V_{CC} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

System Ground (V_{SS})

All V_{SS} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

Processor Architecture

As shown by the block diagram (Figure 1), the 80C187 NPX is internally divided into three sections: the bus control logic (BCL), the data interface and control unit, and the floating-point unit (FPU). The FPU (with the support of the control unit which contains the sequencer and other support units) executes all numerics instructions. The data interface and control unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, and sequencing the microinstructions, and for handling some of the administrative instructions. The BCL is responsible for CPU bus tracking and interface.

BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from memory to the 80C187 and transferring outputs from the 80C187 to memory. A dedicated communication protocol makes possible high-speed transfer of opcodes and operands between the CPU and 80C187.

Table 10. Bus Cycles Definition

$\overline{NPS1}$	NPS2	CMD0	CMD1	$\overline{NPRD}$	$\overline{NPWR}$	Bus Cycle Type
x	0	x	x	x	x	80C187 Not Selected
1	x	x	x	x	x	80C187 Not Selected
0	1	0	0	1	0	Opcode Write to 80C187
0	1	0	0	0	1	CW or SW Read from 80C187
0	1	1	0	0	1	Read Data from 80C187
0	1	1	0	1	0	Write Data to 80C187
0	1	0	1	1	0	Write Exception Pointers
0	1	0	1	0	1	Reserved
0	1	1	1	0	1	Read Opcode Status
0	1	1	1	1	0	Reserved

DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the FIFO or the instruction decoder. The instruction decoder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, FSTCW, FSETPM, or FRSTPM, the control executes it independently of the FPU and the sequencer. The data interface and control unit is the one that generates the BUSY, PEREQ, and $\overline{ERROR}$ signals that synchronize 80C187 activities with the CPU.

FLOATING-POINT UNIT

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

Bus Cycles

The pins $\overline{NPS1}$, NPS2, CMD0, CMD1, $\overline{NPRD}$ and $\overline{NPWR}$ identify bus cycles for the NPX. Table 10 defines the types of 80C187 bus cycles.

80C187 ADDRESSING

The $\overline{NPS1}$, NPS2, CMD0, and CMD1 signals allow the NPX to identify which bus cycles are intended for the NPX. The NPX responds to I/O cycles when the I/O address is 00F8H, 00FAH, 00FCH, or 00FEH. The correspondence betwen I/O addresses and control signals is defined by Table 11. To guarantee correct operation of the NPX, programs must not perform any I/O operations to these reserved port addresses.

Table 11. I/O Address Decoding

I/O Address (Hexadecimal)	80C187 Select and Command Inputs			
	NPS2	$\overline{NPS1}$	CMD1	CMD0
00F8	1	0	0	0
00FA	1	0	0	1
00FC	1	0	1	0
00FE	1	0	1	1

CPU/NPX SYNCHRONIZATION

The pins BUSY, PEREQ, and $\overline{\text{ERROR}}$ are used for various aspects of synchronization between the CPU and the NPX.

BUSY is used to synchronize instruction transfer from the CPU to the 80C187. When the 80C187 recognizes an ESC instruction, it asserts BUSY. For most ESC instructions, the CPU waits for the 80C187 to deassert BUSY before sending the new opcode.

The NPX uses the PEREQ pin of the CPU to signal that the NPX is ready for data transfer to or from its data FIFO. The NPX does not directly access memory; rather, the CPU provides memory access services for the NPX.

Once the CPU initiates an 80C187 instruction that has operands, the CPU waits for PEREQ signals that indicate when the 80C187 is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the CPU continues program execution while the 80C187 executes the ESC instruction.

In 8086/8087 systems, WAIT instructions are required to achieve synchronization of both commands and operands. The 80C187, however, does not require WAIT instructions. The WAIT or FWAIT instruction commonly inserted by high-level compilers and assembly-language programmers for exception synchronization is not treated as an instruction by the 80C186 and does not provide exception trapping. (Refer to the section "System Configuration for 8087-Compatible Exception Trapping".)

Once it has started to execute a numerics instruction and has transferred the operands from the CPU, the 80C187 can process the instruction in parallel with and independent of the host CPU. When the NPX detects an exception, it asserts the $\overline{\text{ERROR}}$ signal, which causes a CPU interrupt.

OPCODE INTERPRETATION

The CPU and the NPX use a bus protocol that adapts to the numerics opcode being executed. Only the NPX directly interprets the opcode. Some of the results of this interpretation are relevant to the CPU. The NPX records these results (opcode status information) in an internal 16-bit register. The 80C186 accesses this register only via reads from NPX port 00FEH. Tables 10 and 11 define the signal combinations that correspond to each of the following steps.

1. The CPU writes the opcode to NPX port 00F8H. This write can occur even when the NPX is busy or is signalling an exception. The NPX does not necessarily begin executing the opcode immediately.
2. The CPU reads the opcode status information from NPX port 00FEH.
3. The CPU initiates subsequent bus cycles according to the opcode status information. The opcode status information specifies whether to wait until the NPX is not busy, when to transfer exception pointers to port 00FCH, when to read or write operands and results at port 00FAH, etc.

For most instructions, the NPX does not start executing the previously transferred opcode until the CPU (guided by the opcode status information) first writes exception pointer information to port 00FCH of the NPX. This protocol is completely transparent to programmers.

Bus Operation

With respect to bus interface, the 80C187 is fully asynchronous with the CPU, even when it operates from the same clock source as the CPU. The CPU initiates a bus cycle for the NPX by activating both $\overline{\text{NPS1}}$ and NPS2, the NPX select signals. During the CLK period in which $\overline{\text{NPS1}}$ and NPS2 are activated, the 80C187 also examines the $\overline{\text{NPRD}}$ and $\overline{\text{NPRW}}$

input signals to determine whether the cycle is a read or a write cycle and examines the CMD0 and CMD1 inputs to determine whether an opcode, operand, or control/status register transfer is to occur. The 80C187 activates its BUSY output some time after the leading edge of the $\overline{NPRD}$ or $\overline{NPRW}$ signal. Input and ouput data are referenced to the trailing edges of the $\overline{NPRD}$ and $\overline{NPRW}$ signals.

The 80C187 activates the PEREQ signal when it is ready for data transfer. The 80C187 deactivates PEREQ automatically.

System Configuration

The 80C187 can be connected to the 80C186 CPU as shown by Figure 9. (Refer to the 80C186 Data Sheet for an explanation of the 80C186's signals.) This interface has the following characteristics:

- The 80C187's $\overline{NPS1}$, $\overline{ERROR}$, PEREQ, and BUSY pins are connected directly to the corresponding pins of the 80C186.
- The 80C186 pin $\overline{MCS3}/\overline{NPS}$ is connected to $\overline{NPS1}$; NPS2 is connected to V_{CC}. Note that if the 80C186 CPU's $\overline{DEN}$ signal is used to gate external data buffers, it must be combined with the $\overline{NPS}$ signal to insure numeric accesses will not activate these buffers.
- The $\overline{NPRD}$ and $\overline{NPRW}$ pins are connected to the $\overline{RD}$ and $\overline{WR}$ pins of the 80C186.
- CMD1 and CMD0 come from the latched A_2 and A_1 of the 80C186, respectively.
- The 80C187 BUSY output connects to the 80C186 $\overline{TEST}$/BUSY input. During RESET, the signal at the 80C187 BUSY output automatically programs the 80C186 to use the 80C187.
- The 80C187 can use the CLKOUT signal of the 80C186 to conserve board space when operating at 12.5 MHz or less. In this case, the 80C187 CKM input must be pulled HIGH. For operation in excess of 12.5 MHz, a double-frequency external oscillator for CLK input is needed. In this case, CKM must be pulled LOW.

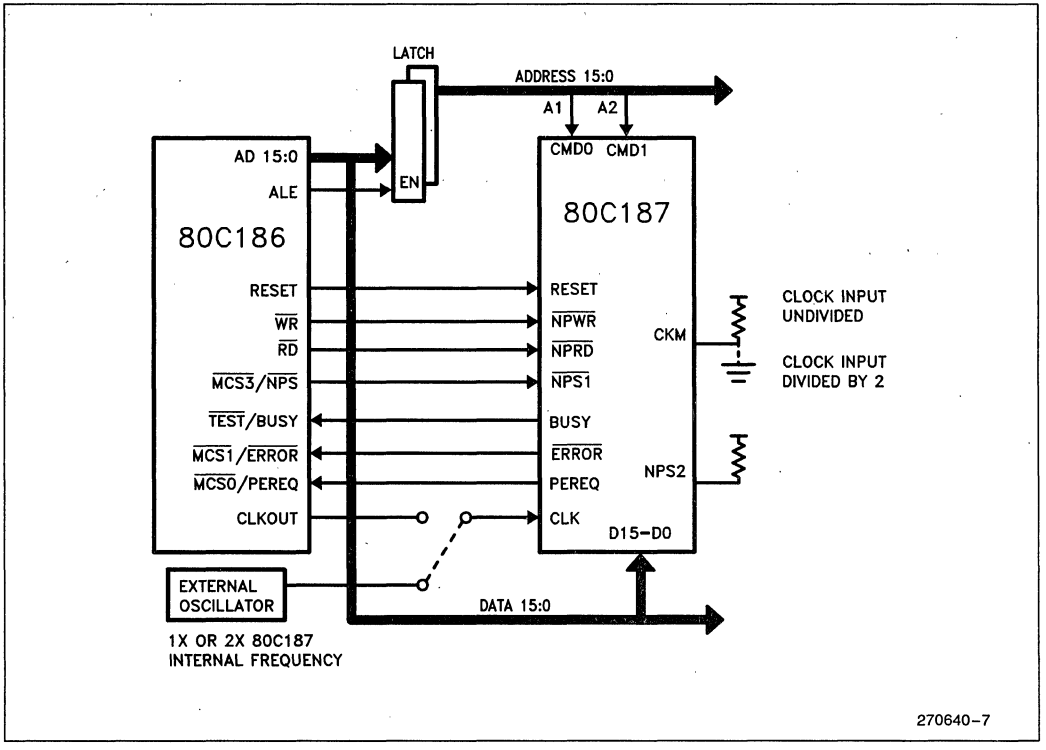
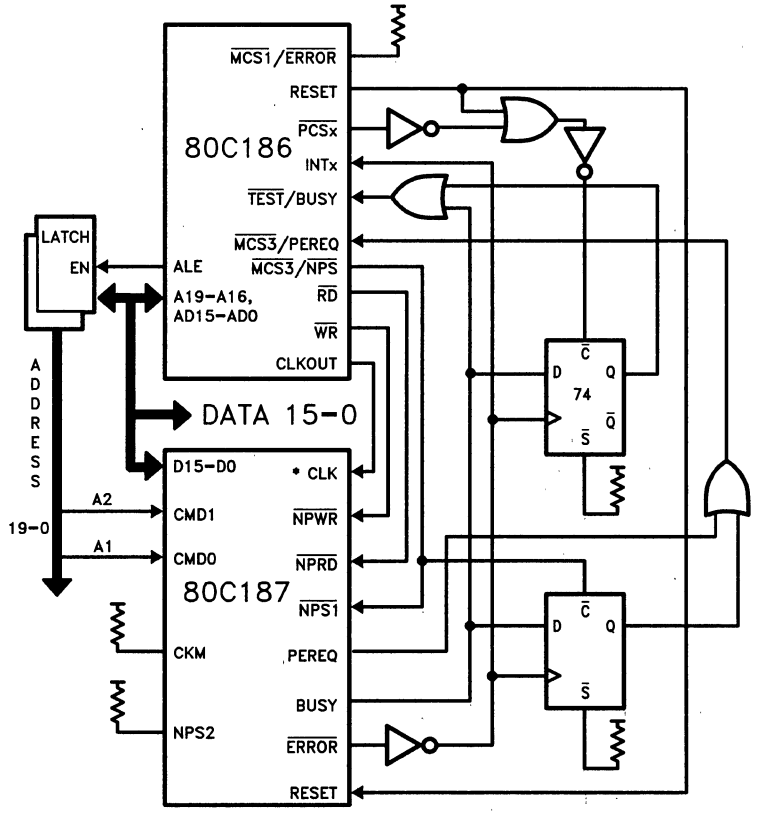


Figure 9. 80C186/80C187 System Configuration

System Configuration for 80186/ 80187-Compatible Exception Trapping

When the 80C187 $\overline{\text{ERROR}}$ output signal is connected directly to the 80C186 $\overline{\text{ERROR}}$ input, floating-point exceptions cause interrupt #16. However, existing software may be programmed to expect floating-point exceptions to be signalled over an external interrupt pin via an interrupt controller.

For exception handling compatible with the 80186/ 82188/8087, the 80C186 can be wired to recognize exceptions through an external interrupt pin, as Figure 10 shows. (Refer to the 80C186 Data Sheet for an explanation of the 80C186's signals.) With this arrangement, a flip-flop is needed to latch BUSY upon assertion of $\overline{\text{ERROR}}$. The latch can then be cleared during the exception-handler routine by forcing a $\overline{\text{PCS}}$ pin active. The latch must also be cleared at RESET in order for the 80C186 to work with the 80C187.



*For input clocking options, refer to Figure 9.

Figure 10. System Configuration for 8087-Compatible Exception Trapping

intel

ELECTRICAL DATA

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

Absolute Maximum Ratings*

Case Temperature Under Bias (T_C) . . . 0°C to +85°C

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

Voltage on Any Pin
with Respect to Ground . . . . −0.5V to V_{CC} +0.5V

Power Dissipation 1.5W

WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

Power and Frequency Requirements

The typical relationship between I_{CC} and the frequency of operation F is as follows:

$I_{CC_{typ}} = 55 + 5 \cdot F$ mA where F is in MHz.

When the frequency is reduced below the minimum operating frequency specified in the AC Characteristics table, the internal states of the 80C187 may become indeterminate. The 80C187 clock cannot be stopped; otherwise, I_{CC} would increase significantly beyond what the equation above indicates.

DC Characteristics T_C = 0°C to +85°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} +0.5	V	
V_{ICL}	Clock Input LOW Voltage	−0.5	+0.8	V	
V_{ICH}	Clock Input HIGH Voltage	2.0	V_{CC} +0.5	V	
V_{OL}	Output LOW Voltage		0.45	V	I_{OL} = 3.0 mA
V_{OH}	Output HIGH Voltage	2.4		V	I_{OH} = −0.4 mA
I_{CC}	Power Supply Current		156 135	mA mA	16 MHz 12.5 MHz
I_{LI}	Input Leakage Current		±10	μA	$0V \le V_{IN} \le V_{CC}$
I_{LO}	I/O Leakage Current		±10	μA	$0.45V \le V_{OUT} \le V_{CC} - 0.45V$
C_{IN}	Input Capacitance		10	pF	F_C = 1 MHz
C_O	I/O or Output Capacitance		12	pF	F_C = 1 MHz
C_{CLK}	Clock Capacitance		20	pF	F_C = 1 MHz

AC Characteristics

T_C = 0°C to +85°C, V_{CC} = 5V ±10%
All timings are measured at 1.5V unless otherwise specified

Symbol	Parameter	12.5 MHz		16 MHz		Test Conditions
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	
T_{dvwh} (t6)	Data Setup to $\overline{NPWR}$	43		33		
T_{whdx} (t7)	Data Hold from $\overline{NPWR}$	14		14		
T_{rlrh} (t8)	$\overline{NPRD}$ Active Time	59		54		
T_{wlwh} (t9)	$\overline{NPWR}$ Active Time	59		54		
T_{avwl} (t10)	Command Valid to $\overline{NPWR}$	0		0		
T_{avrl} (t11)	Command Valid to $\overline{NPRD}$	0		0		
T_{mhrl} (t12)	Min Delay from PEREQ Active to $\overline{NPRD}$ Active	40		30		
T_{whax} (t18)	Command Hold from $\overline{NPWR}$	12		8		
T_{rhax} (t19)	Command Hold from $\overline{NPRD}$	12		8		
T_{ivcl} (t20)	$\overline{NPRD}$, $\overline{NPWR}$, RESET to CLK Setup Time	46		38		Note 1
T_{clih} (t21)	$\overline{NPRD}$, $\overline{NPWR}$, RESET from CLK Hold Time	26		18		Note 1
T_{rscl} (t24)	RESET to CLK Setup	21		19		Note 1
T_{clrs} (t25)	RESET from CLK Hold	14		9		Note 1
T_{cmdi} (t26)	Command Inactive Time					
	Write to Write	69		59		
	Read to Read	69		59		
	Read to Write	69		59		
	Write to Read	69		59		

NOTE:
1. This is an asynchronous input. This specification is given for testing purposes only, to assure recognition at a specific CLK edge.

Timing Responses

All timings are measured at 1.5V unless otherwise specified

Symbol	Parameter	12.5 MHz		16 MHz		Test Conditions
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	
T_{rhqz} (t27)	$\overline{NPRD}$ Inactive to Data Float*		18		18	Note 2
T_{rlqv} (t28)	$\overline{NPRD}$ Active to Data Valid		50		45	Note 3
T_{ilbh} (t29)	$\overline{ERROR}$ Active to Busy Inactive	104		104		Note 4
T_{wlbv} (t30)	$\overline{NPWR}$ Active to Busy Active		80		60	Note 4
T_{klml} (t31)	$\overline{NPRD}$ or $\overline{NPWR}$ Active to PEREQ Inactive		80		60	Note 5
T_{rhqh} (t32)	Data Hold from $\overline{NPRD}$ Inactive	2		2		Note 3
T_{rlbh} (t33)	RESET Inactive to BUSY Inactive		80		60	

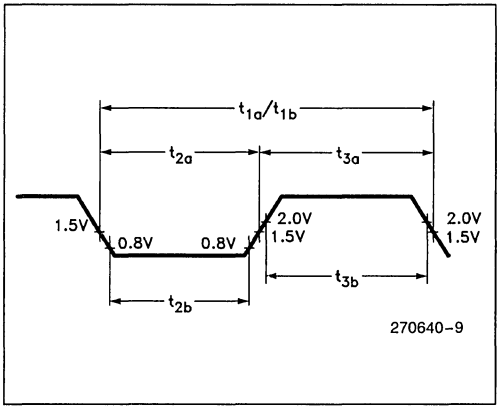
NOTES:
*The data float delay is not tested.
2. The float condition occurs when the measured output current is less than I_{OL} on D_{15}–D_0.
3. D_{15}–D_0 loading: C_L = 100 pF.
4. BUSY loading: C_L = 100 pF.
5. On last data transfer of numeric instruction.

Clock Timings

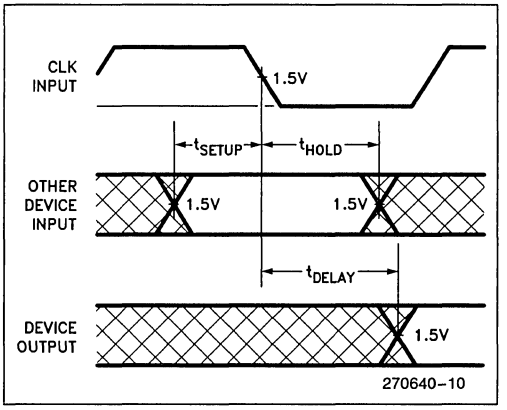
Symbol	Parameter		12.5 MHz		16 MHz*		Test Conditions
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	
T_{clcl} (t1a)	CLK Period	CKM = 1	80	250	N/A	N/A	Note 6
(t1B)		CKM = 0	40	125	31.25	125	Note 6
T_{clch} (t2a)	CLK Low Time	CKM = 1	35		N/A		Note 6
(t2b)		CKM = 0	9		7		Note 7
T_{chcl} (t3a)	CLK High Time	CKM = 1	35		N/A		Note 6
(t3b)		CKM = 0	13		9		Note 8
T_{ch2ch1}(t4)				10		8	Note 9
T_{ch1ch2}(t5)				10		8	Note 10

NOTES:
*16 MHz operation is available only in divide-by-2 mode (CKM strapped LOW).
6. At 1.5V
7. At 0.8V
8. At 2.0V
9. CKM = 1: 3.7V to 0.8V at 16 MHz, 3.5V to 1.0V at 12.5 MHz
10. CKM = 1: 0.8V to 3.7V at 16 MHz, 1.0V to 3.5V at 12.5 MHz

AC DRIVE AND MEASUREMENT POINTS—CLK INPUT

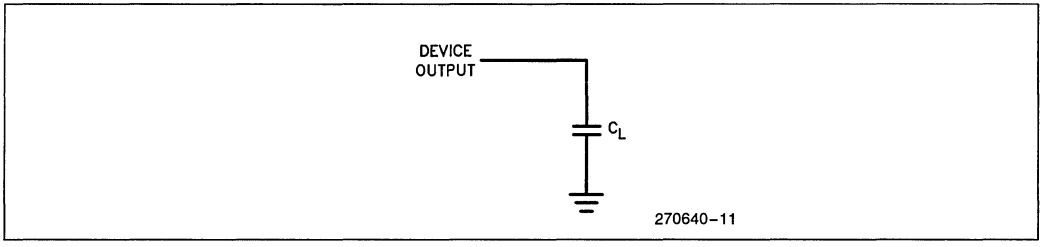


AC SETUP, HOLD, AND DELAY TIME MEASUREMENTS—GENERAL

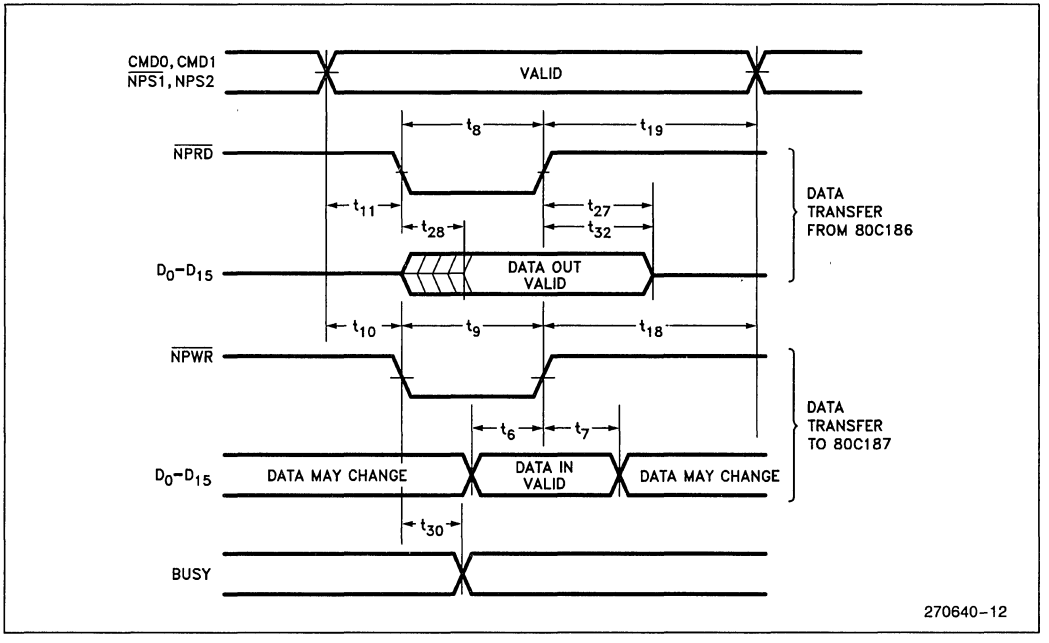


1

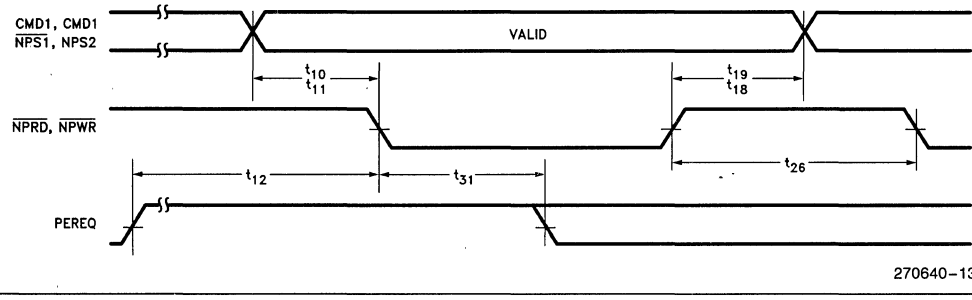
AC TEST LOADING ON OUTPUTS



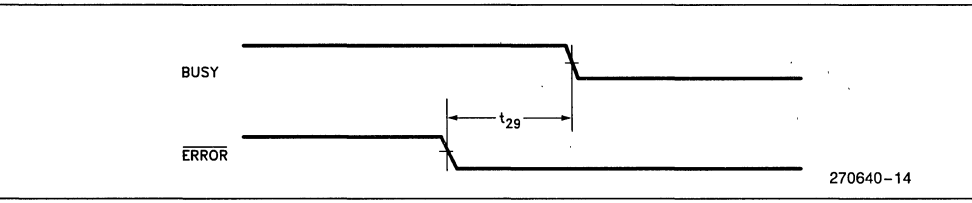
DATA TRANSFER TIMING (INITIATED BY CPU)



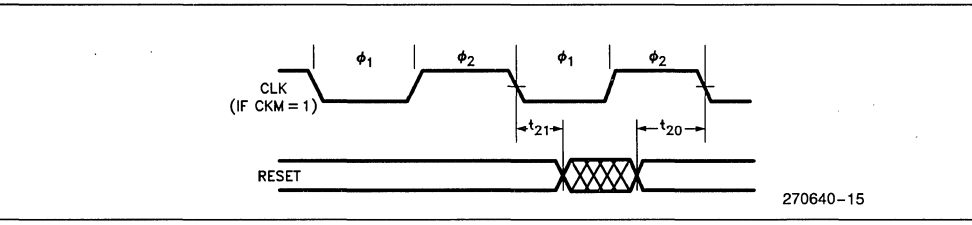
DATA CHANNEL TIMING (INITIATED BY 80C187)



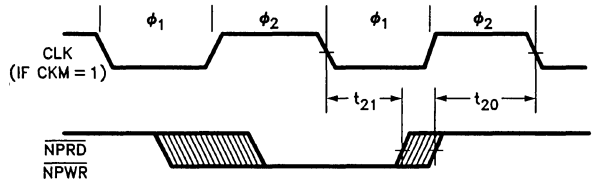
ERROR OUTPUT TIMING



CLK, RESET TIMING (CKM = 1)

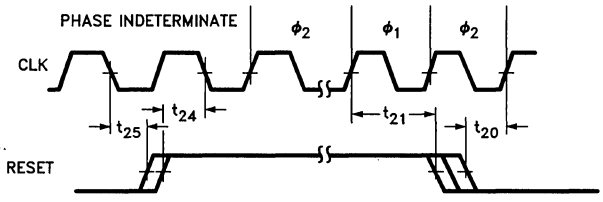


CLK, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ TIMING (CKM = 1)



270640-16

CLK, RESET TIMING (CKM = 0)

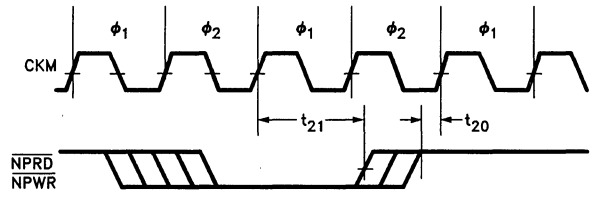


270640-17

RESET must meet timing shown to guarantee known phase of internal divide by 2 circuits.

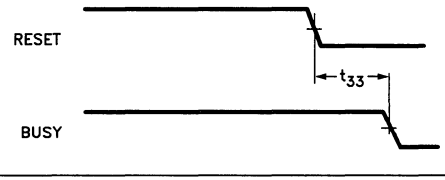
NOTE:
RESET, $\overline{\text{NPWR}}$, $\overline{\text{NPRD}}$ inputs are asynchronous to CLK. Timing requirements are given for testing purposes only, to assure recognition at a specific CLK edge.

CLK, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$ TIMING (CKM = 0)



270640-18

RESET, BUSY TIMING



270640-19

80C187 EXTENSIONS TO THE CPU's INSTRUCTION SET

Instructions for the 80C187 assume one of the five forms shown in Table 12. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the CPU's addressing modes.

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of CPU instructions (refer to Programmer's Reference Manual for the CPU). The DISP (displacement) is optionally present in instructions that have MOD and R/M fields. Its presence depends on the values of MOD and R/M, as for instructions of the CPU.

The instruction summaries that follow assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD requests delaying processor access to the bus; and that no exceptions are detected during instruction execution. Timings are given in internal 80C187 clocks and include the time for opcode and data transfer between the CPU and the NPX. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

Table 12. Instruction Formats

	Instruction									Optional Field
	First Byte				Second Byte					
1	11011	OPA		1	MOD		1	OPB	R/M	DISP
2	11011	MF		OPA	MOD		OPB *		R/M	DISP
3	11011	d	P	OPA	1	1	OPB *		ST (i)	
4	11011	0	0	1	1	1	1	OP		
5	11011	0	1	1	1	1	1	OP		
	15-11	10	9	8	7	6	5	4 3	2 1 0	

NOTES:

OP = Instruction opcode, possibly split into two fields OPA and OPB

MF = Memory Format
- 00— 32-Bit Real
- 01— 32-Bit Integer
- 10— 64-Bit Real
- 11— 16-Bit Integer

d = Destination
- 0— Destination is ST(0)
- 0— Destination is ST(i)

R XOR d = 0— Destination (op) Source
R XOR d = 1— Source (op) Destination

*In FSUB and FDIV, the low-order bit of OPB is the R (reversed) bit

P = Pop
- 0— Do not pop stack
- 1— Pop stack after operation

ESC = 11011

ST(i) = Register Stack Element *i*
- 000 = Stack Top
- 001 = Second Stack Element
- •
- •
- •
- 111 = Eighth Stack Element

80C187 Extensions to the 80C186 Instruction Set

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2-3	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
DATA TRANSFER							
FLD = Load[a]							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	DISP	40	65-72	59	67-71
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	DISP		90-101		
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	DISP		74		
BCD memory to ST(0)	ESC 111	MOD 100 R/M	DISP		296-305		
ST(i) to ST(0)	ESC 001	11000 ST(i)			16		
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	DISP	58	93-107	73	80-93
ST(0) to ST(i)	ESC 101	11010 ST(i)			13		
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	DISP	58	93-107	73	80-93
ST(0) to long integer memory	ESC 111	MOD 111 R/M	DISP		116-133		
ST(0) to extended real	ESC 011	MOD 111 R/M	DISP		83		
ST(0) to BCD memory	ESC 111	MOD 110 R/M	DISP		542-564		
ST(0) to ST(i)	ESC 101	11001 ST (i)			14		
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)			20		
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	DISP	48	78-85	67	77-81
ST(i) to ST(0)	ESC 000	11010 ST(i)			26		
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	DISP	48	78-85	67	77-81
ST(i) to ST(0)	ESC 000	11011 ST(i)			28		
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001			28		
FTST = Test ST(0)	ESC 001	1110 0100			30		
FUCOM = Unordered compare	ESC 101	11100 ST(i)			26		
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)			28		
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001			28		
FXAM = Examine ST(0)	ESC 001	11100101			32-40		
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110			22		
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000			26		
FLDPI = Load pi into ST(0)	ESC 001	1110 1011			42		
FLDL2T = Load $\log_2(10)$ into ST(0)	ESC 001	1110 1001			42		

Shaded areas indicate instructions not available in 8087.

NOTE:
a. When loading single- or double-precision zero from memory, add 5 clocks.

80C187 Extensions to the 80C186 Instruction Set (Continued)

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–3	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
CONSTANTS (Continued)							
FLDL2E = Load $\log_2(e)$ into ST(0)	ESC 001	1110 1010			42		
FLDLG2 = Load $\log_{10}(2)$ into ST(0)	ESC 001	1110 1100			43		
FLDLN2 = Load $\log_e(2)$ into ST(0)	ESC 001	1110 1101			43		
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	DISP	44–52	77–92	65–73	77–91
ST(i) and ST(0)	ESC d P 0	11000 ST(i)			25–33[b]		
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	DISP	44–52	77–92	65–73	77–91[c]
ST(i) and ST(0)	ESC d P 0	1110 R R/M			28–36[d]		
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	DISP	47–57	81–102	68–93	82–93
ST(i) and ST(0)	ESC d P 0	1100 1 R/M			31–59[e]		
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	DISP	108	140–147[f]	128	142–146[g]
ST(i) and ST(0)	ESC d P 0	1111 R R/M			90[h]		
FSQRT[i] = Square root	ESC 001	1111 1010			124–131		
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101			69–88		
FPREM = Partial remainder of ST(0) ÷ ST(1)	ESC 001	1111 1000			76–157		
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101			97–187		
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100			66–82		
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100			72–78		
FABS = Absolute value of ST(0)	ESC 001	1110 0001			24		
FCHS = Change sign of ST(0)	ESC 001	1110 0000			26–27		

Shaded areas indicate instructions not available in 8087.

NOTES:
b. Add 3 clocks to the range when d = 1.
c. Add 1 clock to **each** range when R = 1.
d. Add 3 clocks to the range when d = 0.
e. typical = 54 (When d = 0, 48–56, typical = 51).
f. Add 1 clock to the range when R = 1.
g. 153–159 when R = 1.
h. Add 3 clocks to the range when d = 1.
i. $-0 \leq ST(0) \leq +\infty$.

80C187 Extensions to the 80C186 Instruction Set (Continued)

Instruction	Encoding			Clock Count Range
	Byte 0	Byte 1	Optional Bytes 2-3	
TRANSCENDENTAL				
FCOS = Cosine of ST(0)	ESC 001	1111 1111		125-774[j]
FPTAN[k] = Partial tangent of ST(0)	ESC 001	1111 0010		193-499[j]
FPATAN = Partial arctangent	ESC 001	1111 0011		316-489
FSIN = Sine of ST(0)	ESC 001	1111 1110		124-773[j]
FSINCOS = Sine and cosine of ST(0)	ESC 001	1111 1011		196-811[j]
F2XM1[l] = $2^{ST(0)} - 1$	ESC 001	1111 0000		213-478
FYL2X[m] = ST(1) * $\log_2(ST(0))$	ESC 001	1111 0001		122-540
FYL2XP1[n] = ST(1) * $\log_2(ST(0) + 1.0)$	ESC 001	1111 1001		259-549
PROCESSOR CONTROL				
FINIT = Initialize NPX	ESC 011	1110 0011		35
FSTSW AX = Store status word	ESC 111	1,110 0000		17
FLDCW = Load control word	ESC 001	MOD 101 R/M	DISP	23
FSTCW = Store control word	ESC 001	MOD 111 R/M	DISP	21
FSTSW = Store status word	ESC 101	MOD 111 R/M	DISP	21
FCLEX = Clear exceptions	ESC 011	1110 0010		13
FSTENV = Store environment	ESC 001	MOD 110 R/M	DISP	146
FLDENV = Load environment	ESC 001	MOD 100 R/M	DISP	113
FSAVE = Save state	ESC 101	MOD 110 R/M	DISP	550
FRSTOR = Restore state	ESC 101	MOD 100 R/M	DISP	482
FINCSTP = Increment stack pointer	ESC 001	1111 0111		23
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		24
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)		20
FNOP = No operations	ESC 001	1101 0000		14

Shaded areas indicate instructions not available in 8087.

NOTES:
j. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 78 clocks may be needed to reduce the operand.
k. $0 \leq |ST(0)| < 2^{63}$.
l. $-1.0 \leq ST(0) \leq 1.0$.
m. $0 \leq ST(0) < \infty$, $-\infty < ST(1) < +\infty$.
n. $0 \leq |ST(0)| < (2 - \sqrt{(2)})/2$, $-\infty < ST(1) < +\infty$.

DATA SHEET REVISION REVIEW

The following list represents the key differences between the -002 and the -001 version of the 80C187 data sheet. Please review this summary carefully.

1. Figure 10, titled "System Configuration for 8087—Compatible Exception Trapping", was replaced with a revised schematic. The previous configuration was faulty. Updated timing diagrams on Data Transfer Timing, Error Output, and RESET/BUSY.

intel®

2

376 Embedded Processors

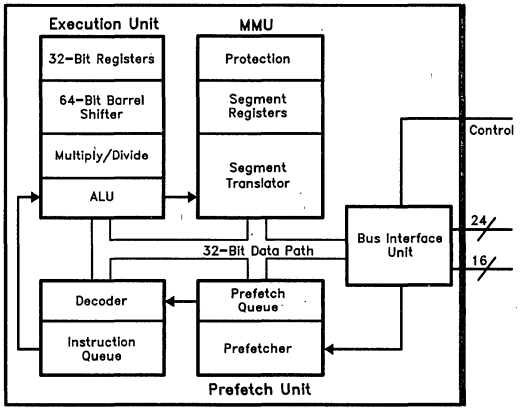
376™ HIGH PERFORMANCE 32-BIT EMBEDDED PROCESSOR

- **Full 32-Bit Internal Architecture**
  - **— 8-, 16-, 32-Bit Data Types**
  - **— 8 General Purpose 32-Bit Registers**
  - **— Extensive 32-Bit Instruction Set**
- **High Performance 16-Bit Data Bus**
  - **— 16 or 20 MHz CPU Clock**
  - **— Two-Clock Bus Cycles**
  - **— 16 Mbytes/Sec Bus Bandwidth**
- **16 Mbyte Physical Memory Size**
- **High Speed Numerics Support with the 80387SX**
- **Low System Cost with the 82370 Integrated System Peripheral**
- **On-Chip Debugging Support Including Break Point Registers**
- **Complete Intel Development Support**
  - **— C, PL/M, Assembler**
  - **— ICE™-376, In-Circuit Emulator**
  - **— iRMK Real Time Kernel**
  - **— iSDM Debug Monitor**
  - **— DOS Based Debug**
- **Extensive Third-Party Support:**
  - **— Languages: C, Pascal, FORTRAN, BASIC and ADA***
  - **— Hosts: VMS*, UNIX*, MS-DOS*, and Others**
  - **— Real-Time Kernels**
- **High Speed CHMOS IV Technology**
- **Available in 100 Pin Plastic Quad Flat-Pack Package and 88-Pin Pin Grid Array**
  (See Packaging Outlines and Dimensions #231369)

2

INTRODUCTION

The 376 32-bit embedded processor is designed for high performance embedded systems. It provides the performance benefits of a highly pipelined 32-bit internal architecture with the low system cost associated with 16-bit hardware systems. The 80376 processor is based on the 80386 and offers a high degree of compatibility with the 80386. All 80386 32-bit programs not dependent on paging can be executed on the 80376 and all 80376 programs can be executed on the 80386. All 32-bit 80386 language translators can be used for software development. With proper support software, any 80386-based computer can be used to develop and test 80376 programs. In addition, any 80386-based PC-AT* compatible computer can be used for hardware prototyping for designs based on the 80376 and its companion product the 82370.



240182-48

80376 Microarchitecture

Intel, iRMK, ICE, 376, 386, Intel386, iSDM, Intel1376 are trademarks of Intel Corp.
*UNIX is a registered trademark of AT&T.
ADA is a registered trademark of the U.S. Government, Ada Joint Program Office.
PC-AT is a registered trademark of IBM Corporation.
VMS is a trademark of Digital Equipment Corporation.
MS-DOS is a trademark of MicroSoft Corporation.

December 1990
Order Number: 240182-004

1.0 PIN DESCRIPTION

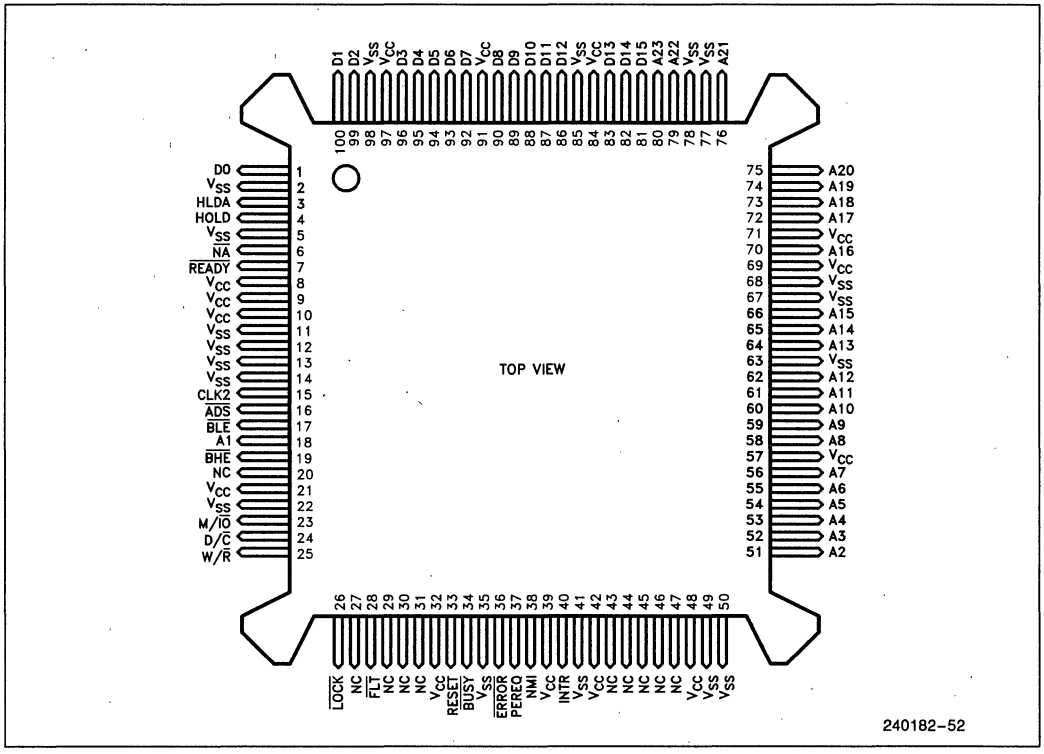
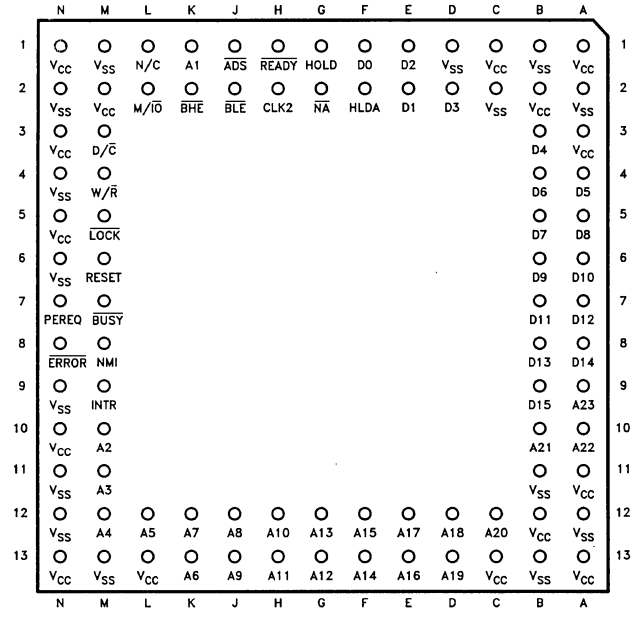


Figure 1.1. 80376 100-Pin Quad Flat-Pack Pin Out (Top View)

Table 1.1. 100-Pin Plastic Quad Flat-Pack Pin Assignments

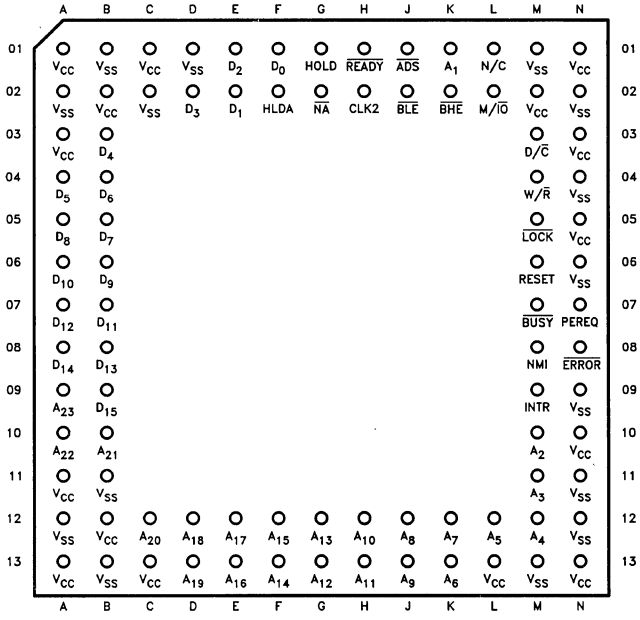
Address		Data		Control		N/C	V_{CC}	V_{SS}
A_1	18	D_0	1	$\overline{ADS}$	16	20	8	2
A_2	51	D_1	100	$\overline{BHE}$	19	27	9	5
A_3	52	D_2	99	$\overline{BLE}$	17		10	11
A_4	53	D_3	96	$\overline{BUSY}$	34	29	21	12
A_5	54	D_4	95	CLK2	15	30	32	13
A_6	55	D_5	94	D/$\overline{C}$	24	31	39	14
A_7	56	D_6	93	$\overline{ERROR}$	36	43	42	22
A_8	58	D_7	92	$\overline{FLT}$	28	44	48	35
A_9	59	D_8	90	HLDA	3	45	57	41
A_{10}	60	D_9	89	HOLD	4	46	69	49
A_{11}	61	D_{10}	88	INTR	40	47	71	50
A_{12}	62	D_{11}	87	$\overline{LOCK}$	26		84	63
A_{13}	64	D_{12}	86	M/$\overline{IO}$	23		91	67
A_{14}	65	D_{13}	83	$\overline{NA}$	6		97	68
A_{15}	66	D_{14}	82	NMI	38			77
A_{16}	70	D_{15}	81	PEREQ	37			78
A_{17}	72			$\overline{READY}$	7			85
A_{18}	73			RESET	33			98
A_{19}	74			W/$\overline{R}$	25			
A_{20}	75							
A_{21}	76							
A_{22}	79							
A_{23}	80							

Top View
(Component Side)



240182-49

Bottom View
(Pin Side)



240182-2

Figure 1.2. 80376 88-Pin Grid Array Pin Out

Table 1.2. 88-Pin Grid Array Pin Assignments

Pin	Label	Pin	Label	Pin	Label	Pin	Label
2H	CLK2	12D	A_{18}	2L	M/$\overline{IO}$	11A	V_{CC}
9B	D_{15}	12E	A_{17}	5M	$\overline{LOCK}$	13A	V_{CC}
8A	D_{14}	13E	A_{16}	1J	$\overline{ADS}$	13C	V_{CC}
8B	D_{13}	12F	A_{15}	1H	$\overline{READY}$	13L	V_{CC}
7A	D_{12}	13F	A_{14}	2G	$\overline{NA}$	1N	V_{CC}
7B	D_{11}	12G	A_{13}	1G	HOLD	13N	V_{CC}
6A	D_{10}	13G	A_{12}	2F	HLDA	11B	V_{SS}
6B	D_9	13H	A_{11}	7N	PEREQ	2C	V_{SS}
5A	D_8	12H	A_{10}	7M	$\overline{BUSY}$	1D	V_{SS}
5B	D_7	13J	A_9	8N	$\overline{ERROR}$	1M	V_{SS}
4B	D_6	12J	A_8	9M	INTR	4N	V_{SS}
4A	D_5	12K	A_7	8M	NMI	9N	V_{SS}
3B	D_4	13K	A_6	6M	RESET	11N	V_{SS}
2D	D_3	12L	A_5	2B	V_{CC}	2A	V_{SS}
1E	D_2	12M	A_4	12B	V_{CC}	12A	V_{SS}
2E	D_1	11M	A_3	1C	V_{CC}	1B	V_{SS}
1F	D_0	10M	A_2	2M	V_{CC}	13B	V_{SS}
9A	A_{23}	1K	A_1	3N	V_{CC}	13M	V_{SS}
10A	A_{22}	2J	$\overline{BLE}$	5N	V_{CC}	2N	V_{SS}
10B	A_{21}	2K	$\overline{BHE}$	10N	V_{CC}	6N	V_{SS}
12C	A_{20}	4M	W/$\overline{R}$	1A	V_{CC}	12N	V_{SS}
13D	A_{19}	3M	D/$\overline{C}$	3A	V_{CC}	1L	N/C

The following table lists a brief description of each pin on the 80376. The following definitions are used in these descriptions:

— The named signal is active LOW.
I Input signal.
O Output signal.
I/O Input and Output signal.
— No electrical connection.

Symbol	Type	Name and Function
CLK2	I	**CLK2** provides the fundamental timing for the 80376. For additional information see **Clock** in Section 4.1.
RESET	I	**RESET** suspends any operation in progress and places the 80376 in a known reset state. See **Interrupt Signals** in Section 4.1 for additional information.
D_{15}–D_0	I/O	**DATA BUS** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles. See **Data Bus** in Section 4.1 for additional information.
A_{23}–A_1	O	**ADDRESS BUS** outputs physical memory or port I/O addresses. See **Address Bus** in Section 4.1 for additional information.
W/$\overline{R}$	O	**WRITE/READ** is a bus cycle definition pin that distinguishes write cycles from read cycles. See **Bus Cycle Definition Signals** in Section 4.1 for additional information.
D/$\overline{C}$	O	**DATA/CONTROL** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and instruction fetching. See **Bus Cycle Definition Signals** in Section 4.1 for additional information.
M/$\overline{IO}$	O	**MEMORY I/O** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles. See **Bus Cycle Definition Signals** in Section 4.1 for additional information.
$\overline{LOCK}$	O	**BUS LOCK** is a bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active. See **Bus Cycle Definition Signals** in Section 4.1 for additional information.
$\overline{ADS}$	O	**ADDRESS STATUS** indicates that a valid bus cycle definition and address (W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{BHE}$, $\overline{BLE}$ and A_{23}–A_1) are being driven at the 80376 pins. See **Bus Control Signals** in Section 4.1 for additional information.
$\overline{NA}$	I	**NEXT ADDRESS** is used to request address pipelining. See **Bus Control Signals** in Section 4.1 for additional information.
$\overline{READY}$	I	**BUS READY** terminates the bus cycle. See **Bus Control Signals** in Section 4.1 for additional information.
$\overline{BHE}$, $\overline{BLE}$	O	**BYTE ENABLES** indicate which data bytes of the data bus take part in a bus cycle. See **Address Bus** in Section 4.1 for additional information.
HOLD	I	**BUS HOLD REQUEST** input allows another bus master to request control of the local bus. See **Bus Arbitration Signals** in Section 4.1 for additional information.

Symbol	Type	Name and Function
HLDA	O	**BUS HOLD ACKNOWLEDGE** output indicates that the 80376 has surrendered control of its local bus to another bus master. See **Bus Arbitration Signals** in Section 4.1 for additional information.
INTR	I	**INTERRUPT REQUEST** is a maskable input that signals the 80376 to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** in Section 4.1 for additional information.
NMI	I	**NON-MASKABLE INTERRUPT REQUEST** is a non-maskable input that signals the 80376 to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** in Section 4.1 for additional information.
$\overline{BUSY}$	I	**BUSY** signals a busy condition from a processor extension. See **Coprocessor Interface Signals** in Section 4.1 for additional information.
$\overline{ERROR}$	I	**ERROR** signals an error condition from a processor extension. See **Coprocessor Interface Signals** in Section 4.1 for additional information.
PEREQ	I	**PROCESSOR EXTENSION REQUEST** indicates that the processor extension has data to be transferred by the 80376. See **Coprocessor Interface Signals** in Section 4.1 for additional information.
$\overline{FLT}$	I	**FLOAT,** when active, forces all bidirectional and output signals, including HLDA, to the float condition. FLOAT is not available on the PGA package. See **Float** for additional information.
N/C	—	**NO CONNECT** should always remain unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the 80376.
V_{CC}	I	**SYSTEM POWER** provides the +5V nominal D.C. supply input.
V_{SS}	I	**SYSTEM GROUND** provides 0V connection from which all inputs and outputs are measured.

2.0 ARCHITECTURE OVERVIEW

The 80376 supports the protection mechanisms needed by sophisticated multitasking embedded systems and real-time operating systems. The use of these protection mechanisms is completely optional. For embedded applications not needing protection, the 80376 can easily be configured to provide a 16 Mbyte physical address space.

Instruction pipelining, high bus bandwidth, and a very high performance ALU ensure short average instruction execution times and high system throughput. The 80376 is capable of execution at sustained rates of 2.5–3.0 million instructions per second.

The 80376 offers on-chip testability and debugging features. Four break point registers allow conditional or unconditional break point traps on code execution or data accesses for powerful debugging of even ROM based systems. Other testability features include self-test and tri-stating of output buffers during RESET.

The Intel 80376 embedded processor consists of a central processing unit, a memory management unit and a bus interface. The central processing unit consists of the execution unit and instruction unit. The execution unit contains the eight 32-bit general registers which are used for both address calculation and data operations and a 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations. The instruction unit decodes the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

The Memory Management Unit (MMU) consists of a segmentation and protection unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability, and efficient sharing.

The protection unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of integrity and simplifies debugging.

Finally, to facilitate high performance system hardware designs, the 80376 bus interface offers address pipelining and direct Byte Enable signals for each byte of the data bus.

2.1 Register Set

The 80376 has twenty-nine registers as shown in Figure 2.1. These registers are grouped into the following six categories:

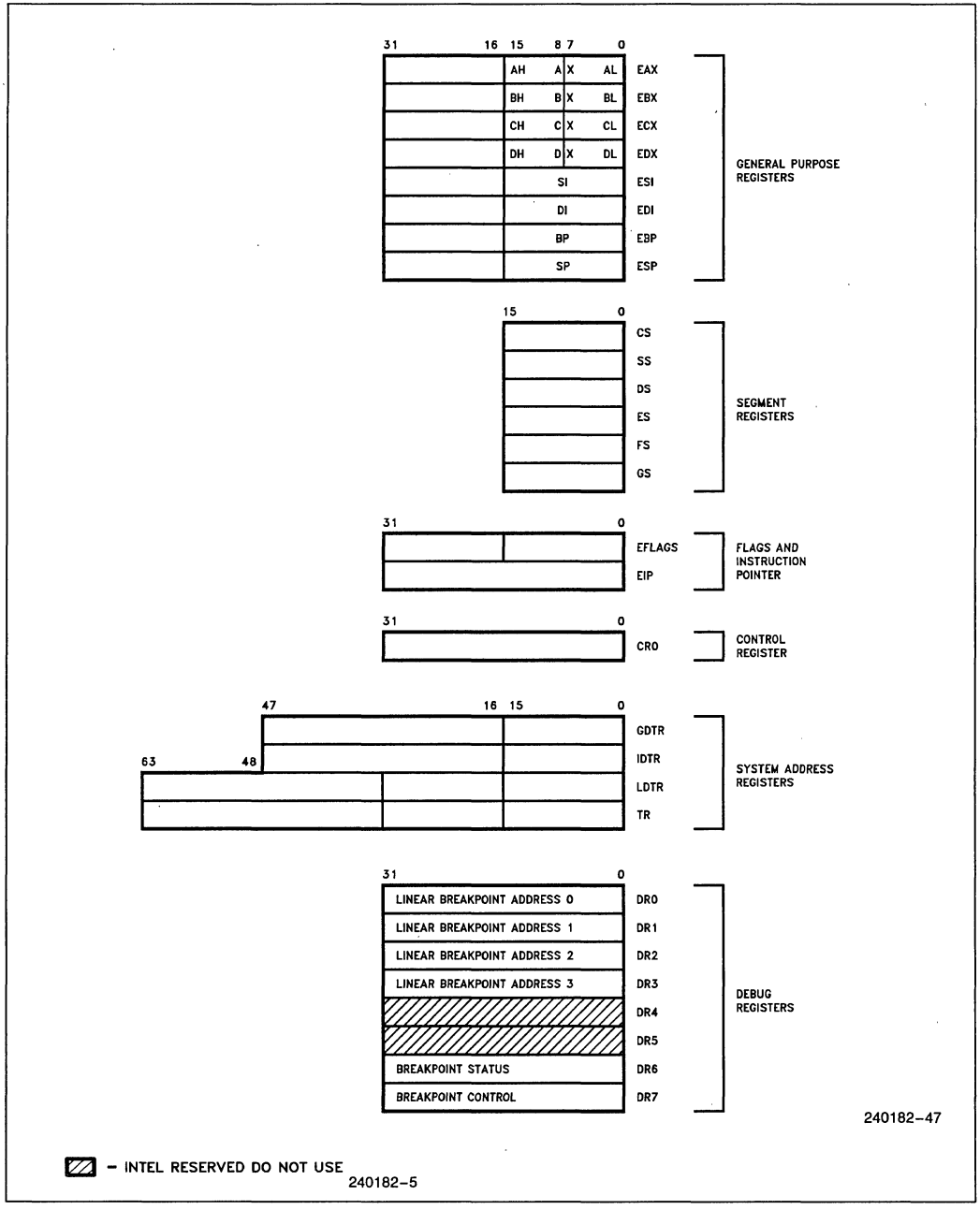


Figure 2.1. 80376 Base Architecture Registers

General Registers: The eight 32-bit general purpose registers are used to contain arithmetic and logical operands. Four of these (EAX, EBX, ECX and EDX) can be used either in their entirety as 32-bit registers, as 16-bit registers, or split into pairs of separate 8-bit registers.

Segment Registers: Six 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data.

Flags and Instruction Pointer Registers: These two 32-bit special purpose registers in Figure 2.1 record or control certain aspects of the 80376 processor state. The EFLAGS register includes status and control bits that are used to reflect the outcome of many instructions and modify the semantics of some instructions. The Instruction Pointer, called EIP, is 32 bits wide. The Instruction Pointer controls instruction fetching and the processor automatically increments it after executing an instruction.

Control Register: The 32-bit control register, CR0, is used to control Coprocessor Emulation.

System Address Registers: These four special registers reference the tables or segments supported by the 80376/80386 protection model. These tables or segments are:

GDTR (Global Descriptor Table Register),
IDTR (Interrupt Descriptor Table Register),
LDTR (Local Descriptor Table Register),
TR (Task State Segment Register).

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. The use of the debug registers is described in Section 2.11 **Debugging Support**.

EFLAGS REGISTER

The flag Register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS, shown in Figure 2.2, control certain operations and indicate the status of the 80376 processor. The function of the flag bits is given in Table 2.1.

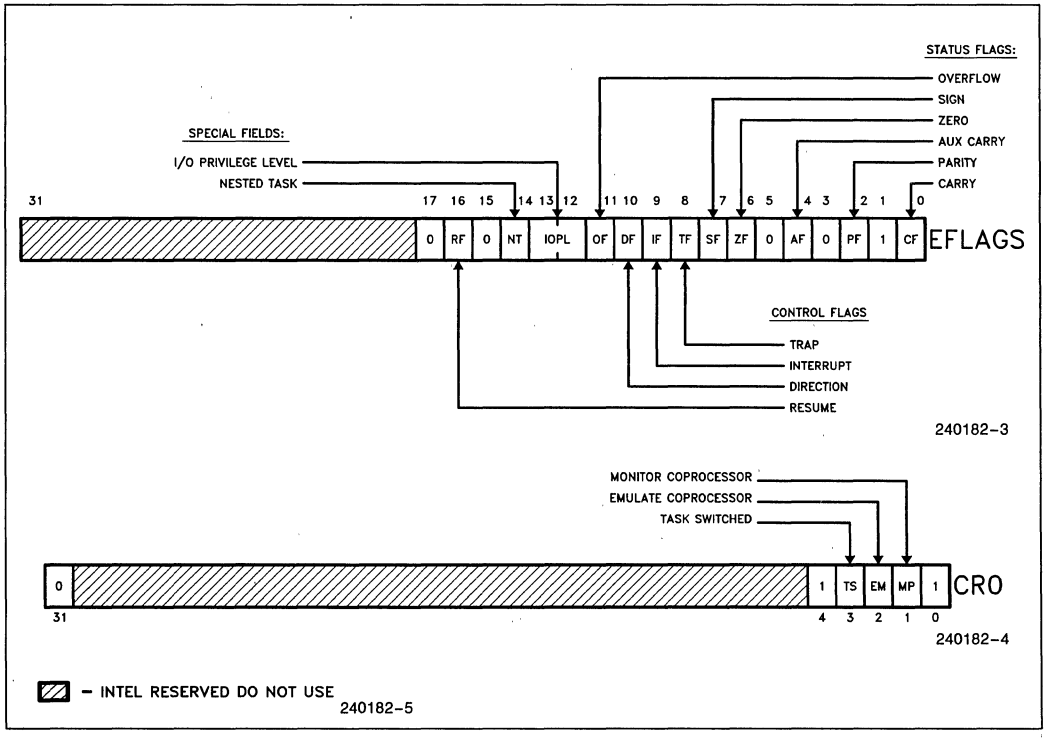


Figure 2.2. Status and Control Register Bit Functions

Table 2.1. Flag Definitions

Bit Position	Name	Function
0	CF	**Carry Flag**—Set on high-order bit carry or borrow; cleared otherwise.
2	PF	**Parity Flag**—Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise.
4	AF	**Auxiliary Carry Flag**—Set on carry from or borrow to the low order four bits of AL; cleared otherwise.
6	ZF	**Zero Flag**—Set if result is zero; cleared otherwise.
7	SF	**Sign Flag**—Set equal to high-order bit of result (0 if positive, 1 if negative).
8	TF	**Single Step Flag**—Once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt.
9	IF	**Interrupt-Enable Flag**—When set, external interrupts signaled on the INTR pin will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	**Direction Flag**—Causes string instructions to auto-increment (default) the appropriate index registers when cleared. Setting DF causes auto-decrement.
11	OF	**Overflow Flag**—Set if the operation resulted in a carry/borrow into the sign bit (high-order bit) of the result but did not result in a carry/borrow out of the high-order bit or vice-versa.
12, 13	IOPL	**I/O Privilege Level**—Indicates the maximum CPL permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O permission bit map. It also indicates the maximum CPL value allowing alteration of the IF bit.
14	NT	**Nested Task**—Indicates that the execution of the current task is nested within another task (see **Task Switching**).
16	RF	**Resume Flag**—Used in conjunction with debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. If set, any debug fault is ignored on the next instruction. It is reset at the successful completion of any instruction except IRET, POPF, and those instructions causing task switches.

CONTROL REGISTER

The 80376 has a 32-bit control register called CR0 that is used to control coprocessor emulation. This register is shown in Figures, 2.1 and 2.2. The defined CR0 bits are described in Table 2.2. Bits 0, 4 and 31 of CR0 have fixed values in the 80376. These values cannot be changed. Programs that load CR0 should always load bits 0, 4 and 31 with values previously there to be compatible with the 80386.

Table 2.2. CR0 Definitions

Bit Position	Name	Function
1	MP	**Monitor Coprocessor Extension**—Allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	**Emulate Processor Extension**—When set, this bit causes a processor extension not present exception (number 7) on ESC instructions to allow processor extension emulation.
3	TS	**Task Switched**—When set, this bit indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task (see **Task Switching**).

2.2 Instruction Set

The instruction set is divided into nine categories of operations:

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High Level Language Support
Operating System Support
Processor Control

These 80376 processor instructions are listed in Table 8.1 **80376 Instruction Set and Clock Count Summary**.

All 80376 processor instructions operate on either 0, 1, 2 or 3 operands; an operand resides in a register, in the instruction itself, or in memory. Most zero operand instructions (e.g. CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2 bytes long. Since the 80376 has a 16-byte prefetch instruction queue an average of 5 instructions can be prefetched. The use of two operands permits the following types of common instructions:

Register to Register
Memory to Register
Immediate to Register
Memory to Memory
Register to Memory
Immediate to Memory

The operands are either 8-, 16- or 32-bit long.

2.3 Memory Organization

Memory on the 80376 is divided into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address. The address of a word or Dword is the byte address of the low-order byte. For maximum performance word and dword values should be at even physical addresses.

In addition to these basic data types the 80376 processor supports segments. Memory can be divided up into one or more variable length segments, which can be shared between programs.

ADDRESS SPACES

The 80376 has three types of address spaces: **logical, linear,** and **physical.** A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, and DISPLACEMENT), discussed in Section 2.4 **Addressing Modes,** into an effective address.

Every selector has a **logical base** address associated with it that can be up to 32 bits in length. This 32-bit **logical base** address is added to either a 32-bit offset address or a 16-bit offset address (by using the ***address length prefix***)to form a final 32-bit **linear** address. This final **linear** address is then truncated so that only the lower 24 bits of this address are used to address the 16 Mbytes physical memory address space. The **logical base** address is stored in one of two operating system tables (i.e. the Local Descriptor Table or Global Descriptor Table).

Figure 2.3 shows the relationship between the various address spaces.

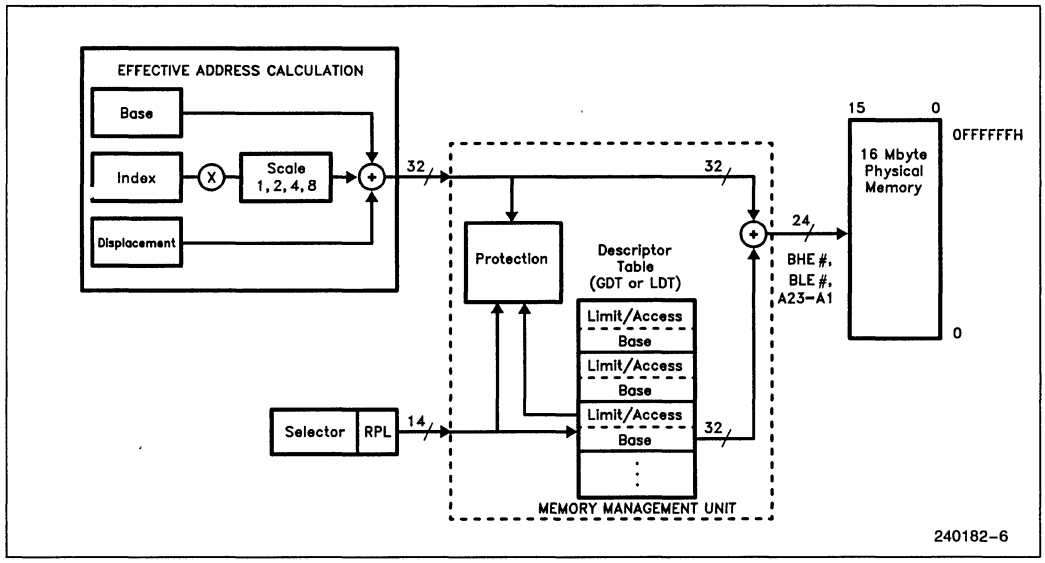


Figure 2.3. Address Translation

SEGMENT REGISTER USAGE

The main data structure used to organize memory is the segment. On the 80376, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments, code and data. The simplest use of segments is to have one code and data segment. Each segment is 16 Mbytes in size overlapping each other. This allows code and data to be directly addressed by the same offset.

In order to provide compact instruction encoding and increase processor performance, instructions do not need to explicitly specify which segment register is used. The segment register is automatically chosen according to the rules of Table 2.3 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register, stack references use the SS register and instruction fetches use the CS register. The contents of the Instruction Pointer provide the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 2.3. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero. Further details of segmentation are discussed in Section 3.0 Architecture.

Table 2.3. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET Instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References, with Effective Address Using Base Register of:		
[EAX]	DS	CS, SS, ES, FS, GS
[EBX]	DS	CS, SS, ES, FS, GS
[ECX]	DS	CS, SS, ES, FS, GS
[EDX]	DS	CS, SS, ES, FS, GS
[ESI]	DS	CS, SS, ES, FS, GS
[EDI]	DS	CS, SS, ES, FS, GS
[EBP]	SS	CS, SS, ES, FS, GS
[ESP]	SS	CS, SS, ES, FS, GS

2.4 Addressing Modes

The 80376 provides a total of 8 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

Two of the addressing modes provide for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8-, 16- or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the opcode.

The remaining 6 modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by summing any combination of the following three address elements (see Figure 2.3):

DISPLACEMENT: an 8-, 16- or 32-bit immediate value following the instruction.

BASE: The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area. Note that if the ***Address Length Prefix*** is used, only BX and BP can be used as a BASE register.

INDEX: The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters. The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. The scaled index is especially useful for accessing arrays or structures. Note that if the ***Address Length Prefix*** is used, no Scaling is available and only the registers SI and DI can be used to INDEX.

Combinations of these 3 components make up the 6 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of BASE and INDEX components which requires one additional clock.

As shown in Figure 2.4, the effective address (EA) of an operand is calculated according to the following formula:

$$EA = BASE_{Register} + (INDEX_{Register} \times scaling) + DISPLACEMENT$$

1. **Direct Mode:** The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit DISPLACEMENT.
2. **Register Indirect Mode:** A BASE register contains the address of the operand.
3. **Based Mode:** A BASE register's contents is added to a DISPLACEMENT to form the operand's offset.
4. **Scaled Index Mode:** An INDEX register's contents is multiplied by a SCALING factor which is added to a DISPLACEMENT to form the operand's offset.
5. **Based Scaled Index Mode:** The contents of an INDEX register is multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operand's offset.
6. **Based Scaled Index Mode with Displacement:** The contents of an INDEX register are multiplied by a SCALING factor, and the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.

2

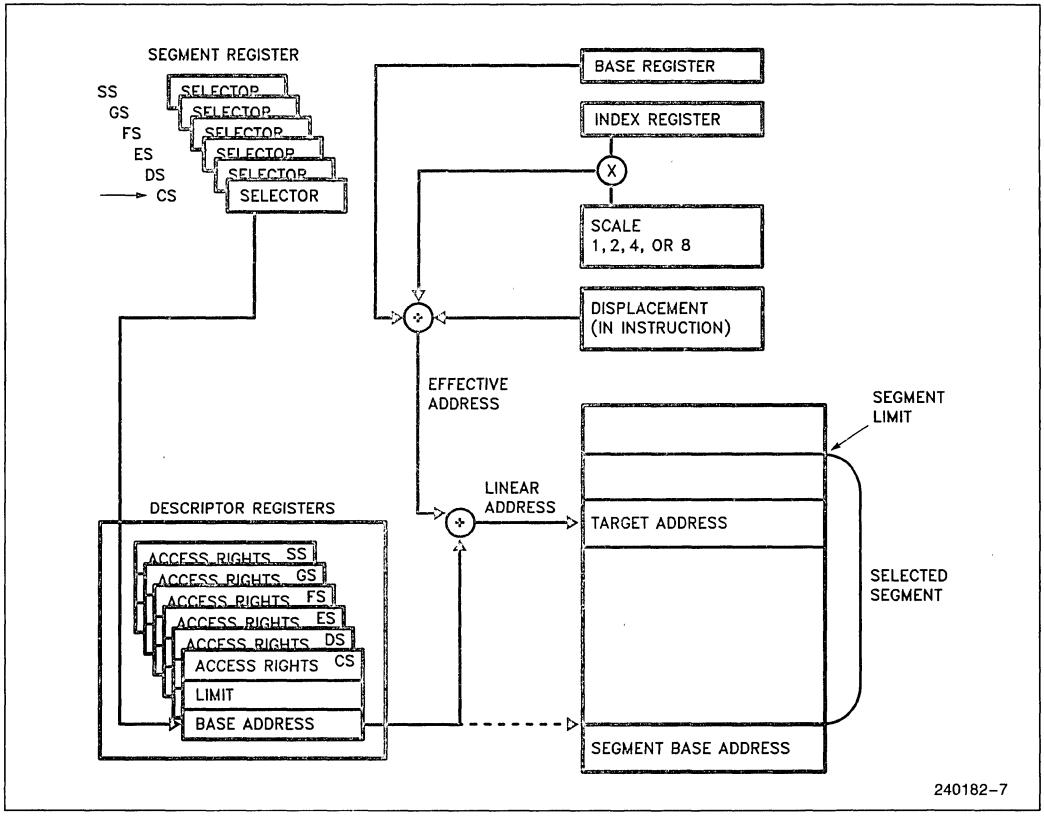


Figure 2.4. Addressing Mode Calculations

GENERATING 16-BIT ADDRESSES

The 80376 executes code with a default length for operands and addresses of 32 bits. The 80376 is also able to execute operands and addresses of 16 bits. This is specified through the use of override prefixes. Two prefixes, the **Operand Length Prefix** and the **Address Length Prefix,** override the default 32-bit length on an individual instruction basis. These prefixes are automatically added by assemblers. The Operand Length and Address Length Prefixes can be applied separately or in combination to any instruction.

The 80376 normally executes 32-bit code and uses either 8- or 32-bit displacements, and any register can be used as based or index registers. When executing 16-bit code (by prefix overrides), the displacements are either 8 or 16 bits, and the base and index register conform to the 16-bit model. Table 2.4 illustrates the differences.

Table 2.4. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX, BP	Any 32-Bit GP Register
INDEX REGISTER	SI, DI	Any 32-Bit GP Register except ESP
SCALE FACTOR	None	1, 2, 4, 8
DISPLACMENT	0, 8, 16 Bits	0, 8, 32 Bits

2.5 Data Types

The 80376 supports all of the data types commonly used in high level languages:

Bit:	A single bit quantity.
Bit Field:	A group of up to 32 contiguous bits, which spans a maximum of four bytes.
Bit String:	A set of contiguous bits, on the 80376 bit strings can be up to 16 Mbits long.
Byte:	A signed 8-bit quantity.
Unsigned Byte:	An unsigned 8-bit quantity.
Integer (Word):	A signed 16-bit quantity.
Long Integer (Double Word):	A signed 32-bit quantity. All operations assume a 2's complement representation.
Unsigned Integer (Word):	An unsigned 16-bit quantity.
Unsigned Long Integer (Double Word):	An unsigned 32-bit quantity.
Signed Quad Word:	A signed 64-bit quantity.
Unsigned Quad Word:	An unsigned 64-bit quantity.
Pointer:	A 16- or 32-bit offset only quantity which indirectly references another memory location.
Long Pointer:	A full pointer which consists of a 16-bit segment selector and either a 16- or 32-bit offset.
Char:	A byte representation of an ASCII Alphanumeric or control character.
String:	A contiguous sequence of bytes, words or dwords. A string may contain between 1 byte and 16 Mbytes.
BCD:	A byte (unpacked) representation of decimal digits 0–9.
Packed BCD:	A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

When the 80376 is coupled with a numerics Coprocessor such as the 80387SX then the following common Floating Point types are supported.

Floating Point: A signed 32-, 64- or 80-bit real number representation. Floating point numbers are supported by the 80387SX numerics coprocessor.

Figure 2.5 illustrates the data types supported by the 80376 processor and the 80387SX coprocessor.

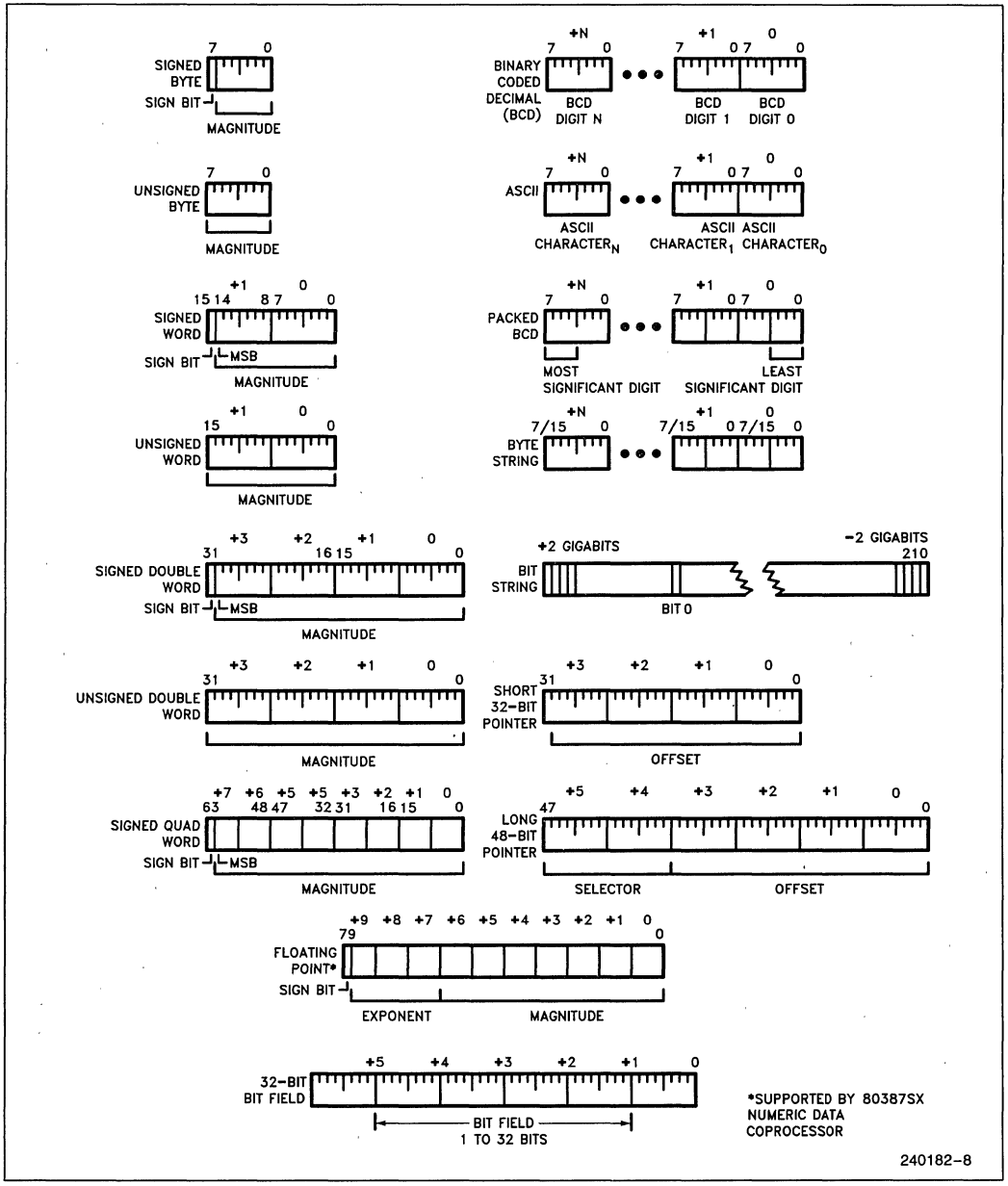


Figure 2.5. 80376 Supported Data Types

2.6 I/O Space

The 80376 has two distinct physical address spaces: physical memory and I/O. Generally, peripherals are placed in I/O space although the 80376 also supports memory-mapped peripherals. The I/O space consists of 64 Kbytes which can be divided into 64K 8-bit ports, 32K 16-bit ports, or any combination of ports which add to no more than 64 Kbytes. The M/$\overline{IO}$ pin acts as an additional address line, thus allowing the system designer to easily determine which address space the processor is accessing. Note that the I/O address refers to a physical address.

The I/O ports are accessed by the IN and OUT instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8-bit and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/$\overline{IO}$ pin to be driven LOW. I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

2.7 Interrupts and Exceptions

Interrupts and exceptions alter the normal program flow in order to handle external events, report errors or exceptional conditons. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction.

Exceptions are classified as faults, traps, or aborts depending on the way they are reported, and whether or not restart of the instruction causing the exception is suported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction which caused the problem. **Aborts** are exceptions which do not permit the precise location of the instruction causing the exception to be determined. Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand the return address from an exception/fault routine will always point at the instruction causing the exception and include any leading instruction prefixes. Table 2.5 summarizes the possible interrupts for the 80376 and shows where the return address points to.

The 80376 has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. The interrupt vectors are 8-byte quantities, which are put in an Interrupt Descriptor Table. Of the 256 possible interrupts, 32 are reserved for use by Intel and the remaining 224 are free to be used by the system designer.

INTERRUPT PROCESSING

When an interrupt occurs the following actions happen. First, the current program address and the Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the 80376 which identifies the appropriate entry in the interrupt table. The table contains either an Interrupt Gate, a Trap Gate or a Task Gate that will point to an interrupt procedure or task. The user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the 80376 in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

Maskable Interrupt

Maskable interrupts are the most common way to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled HIGH and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions (string instructions have an "interrupt window" between memory moves which allows interrupts during long string moves). When an interrupt occurs the processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt (one of 224 user defined interrupts).

Table 2.5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	Yes	FAULT
Debug Exception	1	Any Instruction	Yes	TRAP*
NMI Interrupt	2	INT 2 or NMI	No	NMI
One-Byte Interrupt	3	INT	No	TRAP
Interrupt on Overflow	4	INTO	No	TRAP
Array Bounds Check	5	BOUND	Yes	FAULT
Invalid OP-Code	6	Any Illegal Instruction	Yes	FAULT
Device Not Available	7	ESC, WAIT	Yes	FAULT
Double Fault	8	Any Instruction That Can Generate an Exception		ABORT
Coprocessor Segment Overrun	9	ESC	No	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	Yes	FAULT
Segment Not Present	11	Segment Register Instructions	Yes	FAULT
Stack Fault	12	Stack References	Yes	FAULT
General Protection Fault	13	Any Memory Reference	Yes	FAULT
Intel Reserved	14–15	—	—	—
Coprocessor Error	16	ESC, WAIT	Yes	FAULT
Intel Reserved	17–32			
Two-Byte Interrupt	0–255	INT n	No	TRAP

*Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

Interrupts through Interrupt Gates automatically reset IF, disabling INTR requests. Interrupts through Trap Gates leave the state of the IF bit unchanged. Interrupts through a Task Gate change the IF bit according to the image of the EFLAGs register in the task's Task State Segment (TSS). When an IRET instruction is executed, the original state of the IF bit is restored.

Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. When the NMI input is pulled HIGH it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt no interrupt acknowledgement sequence is performed for an NMI.

While executing the NMI servicing procedure, the 80376 will not service any further NMI request, or INT requests, until an interrupt return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The disabling of INTR requests depends on the gate in IDT location 2.

Software Interrupts

A third type of interrupt/exception for the 80376 is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the n^{th} vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt, is the single step interrupt. It is discussed in **Single-Step Trap** (page 22).

INTERRUPT AND EXCEPTION PRIORITIES

Interrupts are externally-generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are **both** recognized at the **same** instruction boundary, the 80376 invokes the NMI service routine first. If, after the NMI service routine has been invoked, maskable interrupts are still enabled, then the 80376 will invoke the appropriate interrupt service routine.

As the 80376 executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 2.6. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution.

INSTRUCTION RESTART

The 80376 fully supports restarting all instructions after faults. If an exception is detected in the instruction to be executed (exception categories 4 through 9 in Table 2.6), the 80376 device invokes the appropriate exception service routine. The 80376 is in a state that permits restart of the instruction.

DOUBLE FAULT

A Double fault (exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13), but in the process of doing so, detects an exception.

2.8 Reset and Initialization

When the processor is Reset the registers have the values shown in Table 2.7. The 80376 will then start executing instructions near the top of physical memory, at location 0FFFFF0H. A short JMP should be executed within the segment defined for power-up (see Table 2.7). The GDT should then be initialized for a start-up data and code segment followed by a far JMP that will load the segment descriptor cache with the new descriptor values. The IDT table, after reset, is located at physical address 0H, with a limit of 256 entries.

RESET forces the 80376 to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350 and 450 CLK2 periods after Reset becomes inactive, the 80376 will start executing instructions at the top of physical memory.

Table 2.6. Sequence of Exception Checking

Consider the case of the 80376 having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2. Check for external NMI and INTR.
3. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
4. Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5. Check for Faults decoding the next instruction (exception 6 if illegal opcode; or exception 13 if instruction is longer than 15 bytes, or privilege violation (i.e. not at IOPL or at CPL = 0).
6. If WAIT opcode, check if TS = 1 and MP = 1 (exception 7 if both are 1).
7. If ESCape opcode for numeric coprocessor, check if EM = 1 or TS = 1 (exception 7 if either are 1).
8. If WAIT opcode or ESCape opcode for numeric coprocessor, check $\overline{\text{ERROR}}$ input signal (exception 16 if $\overline{\text{ERROR}}$ input is asserted).
9. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13).

Table 2.7. Register Values after Reset

Flag Word (EFLAGS)	uuuu0002H	(Note 1)
Machine Status Word (CR0)	uuuuuuu1H	(Note 2)
Instruction Pointer (EIP)	0000FFF0H	
Code Segment (CS)	F000H	(Note 3)
Data Segment (DS)	0000H	(Note 4)
Stack Segment (SS)	0000H	
Extra Segment (ES)	0000H	(Note 4)
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
EAX Register	0000H	(Note 5)
EDX Register	Component and Stepping ID	(Note 6)
All Other Registers	Undefined	(Note 7)

NOTES:
1. EFLAG Register. The upper 14 bits of the EFLAGS register are undefined, all defined flag bits are zero.
2. CR0: The defined 4 bits in the CR0 is equal to 1H.
3. The Code Segment Register (CS) will have its Base Address set to 0FFFF0000H and Limit set to 0FFFFH.
4. The Data and Extra Segment Registers (DS and ES) will have their Base Address set to 000000000H and Limit set to 0FFFFH.
5. If self-test is selected, the EAX should contain a 0 value. If a value of 0 is not found the self-test has detected a flaw in the part.
6. EDX register always holds component and stepping identifier.
7. All unidentified bits are Intel Reserved and should not be used.

2.9 Initialization

Because the 80376 processor starts executing in protected mode, certain precautions need be taken during initialization. Before any far jumps can take place the GDT and/or LDT tables need to be setup and their respective registers loaded. Before interrupts can be initialized the IDT table must be setup and the IDTR must be loaded. The example code is shown below:

```
; ***************************************************************
;
; This is an example of startup code to put either an 80376,
; 80386SX or 80386 into flat mode. All of memory is treated as
; simple linear RAM. There are no interrupt routines. The
; Builder creates the GDT-alias and IDT-alias and places them,
; by default, in GDT[1] and GDT[2].  Other entries in the GDT
; are specified in the Build file.  After initialization it jumps
; to a C startup routine. To use this template, change this jmp
; address to that of your code, or make the label of your code
; "c_startup".
;
; This code was assembled and built using version 1.2 of the
; Intel RLL utilities and Intel 386ASM assembler.
;
;             ***    This code was tested    ***
;
; ***************************************************************
```

```
NAME FLAT                ; name of the object module

EXTRN    c_startup:near ; this is the label jmped to after init

pe_flag       equ 1
data_selc     equ 20h    ; assume code is GDT[3], data GDT[4]

INIT_CODE    SEGMENT ER PUBLIC USE32     ; Segment base at 0ffffff80h

PUBLIC GDT_DESC

gdt_desc    dq  ?

PUBLIC    START

start:
    cld                          ; clear direction flag
    smsw bx                      ; check for processor (80376) at reset
    test bl,1                    ; use SMSW rather than MOV for speed
    jnz pestart
realstart                        ; is an 80386 and in real mode
    db 66h                       ; force the next operand into 32-bit mode.
    mov eax,offset gdt_desc      ; move address of the GDT descriptor into eax
    xor ebx,ebx                  ; clear ebx
    mov bh,ah                    ; load 8 bits of address into bh
    move bl,al                   ; load 8 bits of address into bl
    db 67h
    db 66h                       ; use the 32-bit form of LGDT to load
    lgdt cs:[ebx]                ; the 32-bits of address into the GDTR
    smsw ax                      ; go into protected mode (set PE bit)
    or al,pe_flag
    lmsw ax
    jmp next                     ; flush prefetch queue
pestart:
    mov ebx,offset gdt_desc
    xor eax,eax
    mov ax,bx                    ; lower portion of address only
    lgdt cs:[eax]
    xor ebx,ebx                  ; initialize data selectors
    mov bl,data_selc             ; GDT[3]
    mov ds,bx
    mov ss,bx
    mov es,bx
    mov fs,bx
    mov gs,bx
    jmp pejump
next:
    xor ebx,ebx                  ; initialize data selectors
    mov bl,data_selc             ; GDT[3]
    mov ds,bx
    mov ss,bx
    mov es,bx
    mov fs,bx
    mov gs,bx
    db 66h                       ; for the 80386, need to make a 32-bit jump
pejump:
    jmp far ptr c_startup        ; but the 80376 is already 32-bit.

    org 70h                      ; only if segment base is at 0ffffff80h
    jmp short start
INIT_CODE ENDS
END
```

This code should be linked into your application for boot loadable code. The following build file illustrates how this is accomplished.

```
FLAT; -- build program id

SEGMENT
    *segments (dpl=0),              -- Give all user segments a DPL of 0.
    _phantom_code_ (dpl=0),         -- These two segments are created by
    _phantom_data_ (dpl=0),         -- the builder when the FLAT control is used.
    init_code  (base=0ffffff80h); -- Put startup code at the reset vector area.

GATE
    gl3 (entry=l3, dpl=0, trap),        -- trap gate disables interrupts
    i32 (entry=32, dpl=0, interrupt),   -- interrupt gates doesn't

TABLE
    -- create GDT

    GDT (LOCATION = GDT_DESC,       -- In a buffer starting at GDT_DESC,
                                    -- BLD386 places the GDT base and
                                    -- GDT limit values. Buffer must be
                                    -- 6 bytes long. The base and limit
                                    -- values are places in this buffer
                                    -- as two bytes of limit plus
                                    -- four bytes of base in the format
                                    -- required for use by the LGDT
                                    -- instruction.
        ENTRY = (3:_phantom_code_,      -- Explicitly place segment
                 4:_phantom_data_,      -- entries into the GDT.
                 5:code32,
                 6:data,
                 7:init_code)
        );
TASK

    MAIN_TASK
        (
         DPL  = 0,                  -- Task privilege level is 0.
         DATA = DATA,               -- Points to a segment that
                                    -- indicates initial DS value.
         CODE = main,               -- Entry point is main, which
                                    -- must be a public id.

         STACKS = (DATA),           -- Segment id points to stack
                                    -- segment. Sets the initial SS:ESP.
         NO INTENABLED,             -- Disable interrupts.
         PRESENT                    -- Present bit in TSS set to 1.
        );

    MEMORY
        (RANGE = (EPROM = ROM(0ffff8000h..0ffffffffh),
                  DRAM = RAM(0..0ffffh)),
         ALLOCATE = (EPROM = (MAIN_TASK)));

END
```

```
asm386 flatsim.a38 debug
asm386 application.a38 debug
bnd386 application.obj,flatsim.obj nolo debug oj (application.bnd)
bld386 application.bnd bf (flatsim.bld) bl flat
```

Commands to assemble and build a boot-loadable application named "application.a38". The initialization code is called "flatsim.a38", and build file is called "application.bld".

2.10 Self-Test

The 80376 has the capability to perform a self-test. The self-test checks the function of all of the Control ROM and most of the non-random logic of the part. Approximately one-half of the 80376 can be tested during self-test.

Self-Test is initiated on the 80376 when the RESET pin transitions from HIGH to LOW, and the $\overline{BUSY}$ pin is LOW. The self-test takes about 2^{20} clocks, or approximately 33 ms with a 16 MHz 80376 processor. At the completion of self-test the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX register is zero. If the EAX register is not zero then the self-test has detected a flaw in the part. If self-test is not selected after reset, EAX may be non-zero after reset.

2.11 Debugging Support

The 80376 provides several features which simplify the debugging process. The three categories of on-chip debugging aids are:

1. The code execution breakpoint opcode (0CCH).
2. The single-step capability provided by the TF bit in the flag register, and
3. The code and data breakpoint capability provided by the Debug Registers DR0–3, DR6, and DR7.

BREAKPOINT INSTRUCTION

A single-byte software interrupt (Int 3) breakpoint instruction is available for use by software debuggers. The breakpoint opcode is 0CCh, and generates an exception 3 trap when executed.

DEBUG REGISTERS

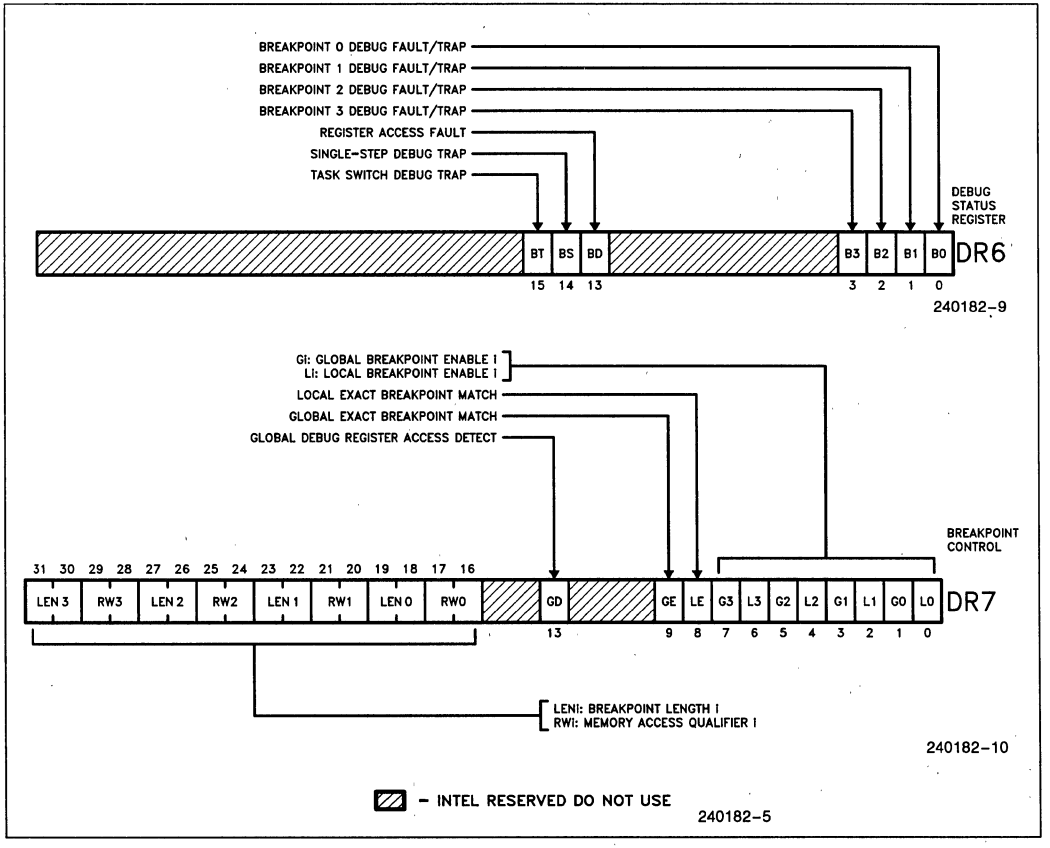


Figure 2.6. Debug Registers

SINGLE-STEP TRAP

If the single-step flag (TF, bit 8) in the EFLAG register is found to be set at the end of an instruction, a single-step exception occurs. The single-step exception is auto vectored to exception number 1.

The Debug Registers are an advanced debugging feature of the 80376. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT 3 breakpoint opcode.

The 80376 contains six Debug Registers, consisting of four breakpoint address registers and two breakpoint control registers. Initially after reset, breakpoints are in the disabled state; therefore, no breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are auto-vectored to exception 1. Figure 2.6 shows the breakpoint status and control registers.

3.0 ARCHITECTURE

The Intel 80376 Embedded Processor has a physical address space of 16 Mbytes (2^{24} bytes) and allows the running of virtual memory programs of almost unlimited size (16 Kbytes × 16 Mbytes or 256 Gbytes (2^{38} bytes)). In addition the 80376 provides a sophisticated memory management and a hardware-assisted protection mechanism.

3.1 Addressing Mechanism

The 80376 uses two components to form the logical address, a 16-bit selector which determines the linear base address of a segment, and a 32-bit effective address. The selector is used to specify an index into an operating system defined table (see Figure 3.1). The table contains the 32-bit base address of a given segment. The linear address is formed by adding the base address obtained from the table to the 32-bit effective address. This value is truncated to 24 bits to form the physical address, which is then placed on the address bus.

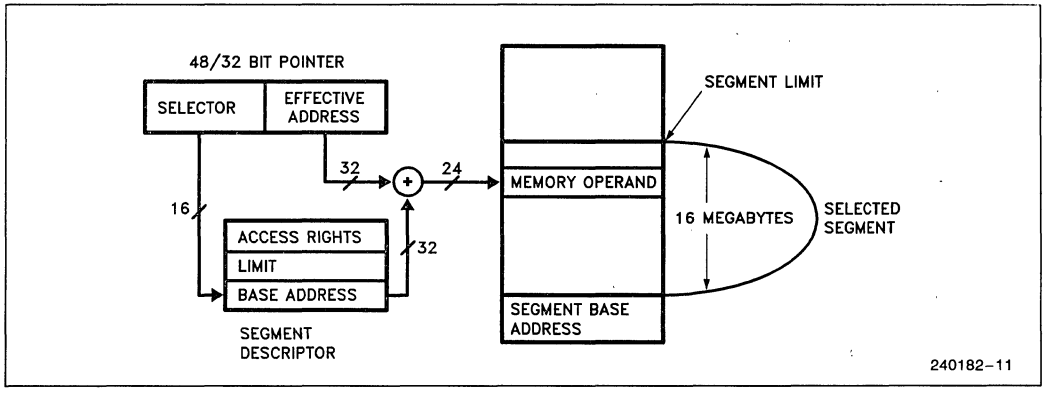


Figure 3.1. Address Calculation

3.2 Segmentation

Segmentation is one method of memory management and provides the basis for protection in the 80376. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about each segment, is stored in an 8-byte data structure called a descriptor. All of the descriptors in a system are contained in tables recognized by hardware.

TERMINOLOGY

The following terms are used throughout the discussion of descriptors, privilege levels and protection:

PL: **Privilege Level**—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged.

RPL: **Requestor Privilege Level**—The privilege level of the original supplier of the selector. RPL is determined by the least two significant bits of a selector.

DPL: **Descriptor Privilege Level**—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

CPL: **Current Privilege Level**—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: **Effective Privilege Level**—The effective privilege level is the least privileged of the RPL and the DPL. EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

DESCRIPTOR TABLES

The descriptor tables define all of the segments which are used in an 80376 system. There are three types of tables on the 80376 which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Decriptor Table. All of the tables are variable length memory arrays, they can range in size between 8 bytes and 64 Kbytes. Each table can hold up to 8192 8-byte descriptors. The upper 13 bits of a selector are used as an index into the descriptor table. The tables have registers associated with them which hold the 32-bit linear base address, and the 16-bit limit of each table.

Each of the tables have a register associated with it: GDTR, LDTR and IDTR; see Figure 3.2. The LGDT, LLDT and LIDT instructions load the base and limit of the Global, Local and Interrupt Descriptor Tables into the appropriate register. The SGDT, SLDT and SIDT store these base and limit values. These are privileged instructions.

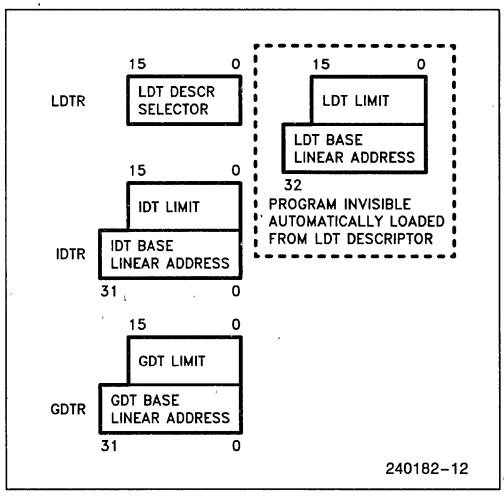


Figure 3.2. Descriptor Table Registers

Global Descriptor Table

The Global Descriptor Table (GDT) contains descriptors which are possibly available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for interrupt and trap descriptors. Every 80376 system contains a GDT. A simple 80376 system contains only 2 entries in the GDT; a code and a data descriptor. For maximum performance, descriptor tables should begin on even addresses.

The first slot of the Global Descriptor Table corresponds to the null selector and is not used. The null selector defines a null pointer value.

Local Descriptor Table

LDTs contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This pro-

vides both isolation and protection for a task's segments, while still allowing global data to be shared among tasks.

Unlike the 6-byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a Local Descriptor Table descriptor in the GDT (see Figure 2.1).

INTERRUPT DESCRIPTOR TABLE

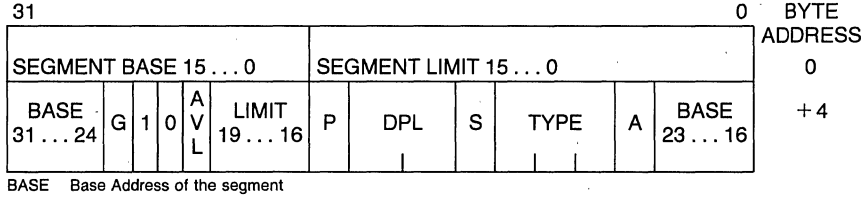
The third table needed for 80376 systems is the Interrupt Descriptor Table. The IDT contains the descriptors which point to the location of up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 Intel Reserved Interrupts. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced by INT instructions, external interrupt vectors, and exceptions.

DESCRIPTORS

The object to which the segment selector points to is called a descriptor. Descriptors are eight-byte quantities which contain attributes about a given region of linear address space. These attributes include the 32-bit logical base address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write or execute privileges, and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. Figure 3.3 shows the general format of a descriptor. All segments on the the 80376 have three attribute fields in common: the Present bit (P), the Descriptor Privilege Level bits (DPL) and the Segment bit (S). P=1 if the segment is loaded in physical memory, if P = 0 then any attempt to access the segment causes a not present exception (exception 11). The DPL is a two-bit field which specifies the protection level, 0–3, associated with a segment.

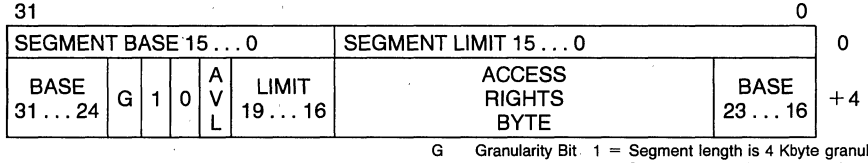
The 80376 has two main categories of segments: system segments, and non-system segments (for code and data). The segment bit, S, determines if a given segment is a system segment, a code segment or a data segment. If the S bit is 1 then the segment is either a code or data segment, if it is 0 then the segment is a system segment.

Note that although the 80376 is limited to a 16-Mbyte Physical address space (2^{24}), its base address allows a segment to be placed anywhere in a 4-Gbyte linear address space. When writing code for the 80376, users should keep code portability to an 80386 processor (or other processors with a larger physical address space) in mind. A segment base address can be placed anywhere in this 4-Gbyte linear address space, but a physical address will be



BASE Base Address of the segment
LIMIT The length of the segment
P Present Bit 1 = Present 0 = Not Present
DPL Descriptor Privilege Level 0–3
S Segment Descriptor: 0 = System Descriptor, 1 = Code or Data Descriptor
TYPE Type of Segment
A Accessed Bit
G Granularity Bit 1 = Segment length is 4 Kbyte Granular
0 = Segment length is byte granular
0 Bit must be zero (0) for compatibility with future processors
AVL Available field for user or OS

Figure 3.3. Segment Descriptors



G Granularity Bit 1 = Segment length is 4 Kbyte granular
0 = Segment length is byte granular
0 Bit must be zero (0) for compatibility with future processors
AVL Available field for user or OS

Figure 3.4. Code and Data Descriptors

Table 3.1. Access Rights Byte Definition for Code and Data Descriptors

Bit Position	Name	Function	
7	Present (P)	P = 1 Segment is mapped into physical memory. P = 0 No mapping to physical memory exits	
6-5	Descriptor Privilege Level (DPL)	Segment privilege attribute used in privilege tests.	
4	Segment Descriptor (S)	S = 1 Code or Data (includes stacks) segment descriptor S = 0 System Segment Descriptor or Gate Descriptor	
3	Executable (E)	E = 0 Descriptor type is data segment:	If Data Segment (S = 1, E = 0)
2	Expansion Direction (ED)	ED = 0 Expand up segment, offsets must be ≤ limit. ED = 1 Expand down segment, offsets must be > limit.	
1	Writable (W)	W = 0 Data segment may not be written into. W = 1 Data segment may be written into.	
3	Executable (E)	E = 1 Descriptor type is code segment:	If Code Segment (S = 1, E = 1)
2	Conforming (C)	C = 1 Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
1	Readable (R)	R = 0 Code segment may not be read. R = 1 Code segment may be read.	
0	Accessed (A)	A = 0 Segment has not been accessed. A = 1 Segment selector has been loaded into segment register or used by selector test instructions.	

generated that is a truncated version of this linear address. Truncation will be to the maximum number of address bits. It is recommended to place EPROM at the highest physical address and DRAM at the lowest physical addresses.

Code and Data Descriptors (S = 1)

Figure 3.4 shows the general format of a code and data descriptor and Table 3.1 illustrates how the bits in the Access Right Byte are interpreted.

Code and data segments have several descriptor fields in common. The accessed bit, A, is set whenever the processor accesses a descriptor. The granularity bit, G, specifies if a segment length is 1-byte-granular or 4-Kbyte-granular. Base address bits 31–24, which are normally found in 80386 descriptors, are not made externally available on the 80376. They do not affect the operation of the 80376. The A_{31}–A_{24} field should be set to allow an 80386 to correctly execute with EPROM at the upper 4096 Mbytes of physical memory.

System Descriptor Formats (S = 0)

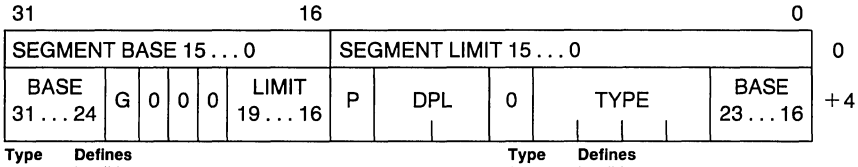
System segments describe information about operating system tables, tasks, and gates. Figure 3.5 shows the general format of system segment descriptors, and the various types of system segments. 80376 system descriptors (which are the same as 80386 descriptor types 2, 5, 9, B, C, E and F) contain a 32-bit logical base address and a 20-bit segment limit.

Selector Fields

A selector has three fields: Local or Global Descriptor Table Indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL) as shown in Figure 3.6. The TI bit selects either the Global Descriptor Table or the Local Descriptor Table. The Index selects one of 8K descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

Segment Descriptor Cache

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs which modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.



Type	Defines	Type	Defines
0	Invalid	8	Invalid
1	Reserved	9	Available 80376/80386 TSS
2	LDT	A	Undefined (Intel Reserved)
3	Reserved	B	Busy 80376/80386 TSS
4	Reserved	C	80376/80386 Call Gate
5	Task Gate (80376/80386 Task)	D	Undefined (Intel Reserved)
6	Reserved	E	80376/80386 Interrupt Gate
7	Reserved	F	80376/80386 Trap Gate

Figure 3.5. System Descriptors

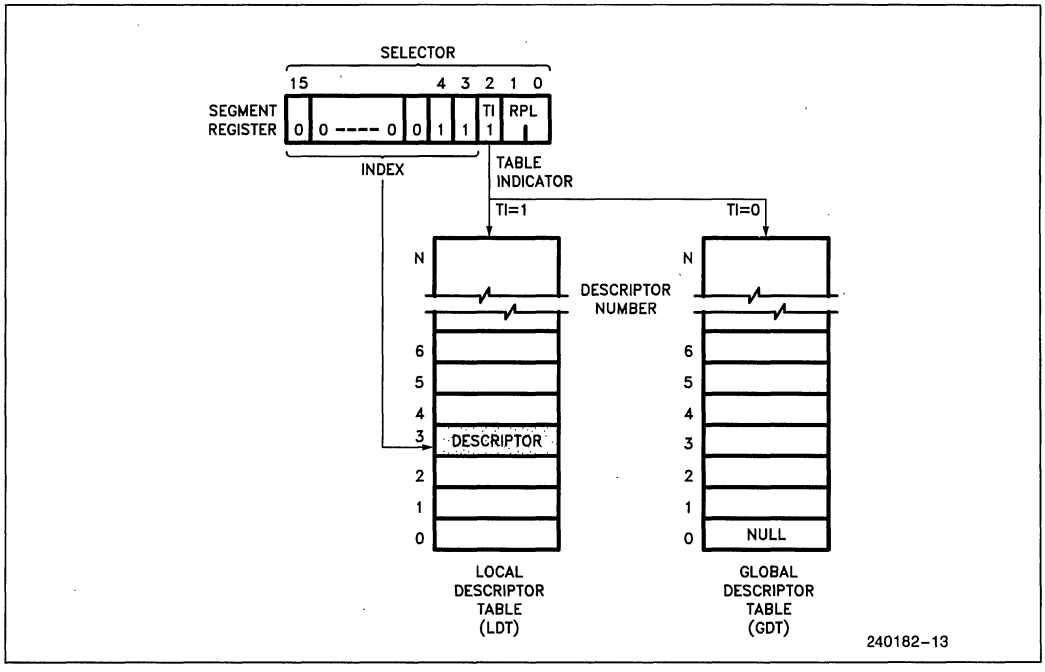


Figure 3.6. Example Descriptor Selection

3.3 Protection

The 80376 offers extensive protection features. These protection features are particularly useful in sophisticated embedded applications which use multitasking real-time operating systems. For simpler embedded applications these protection capabilities can be easily bypassed by making all applications run at privilege level (PL) 0.

RULES OF PRIVILEGE

The 80376 controls access to both data and procedures between levels of a task, according to the following rules.

—Data stored in a segment with privilege level **p** can be accessed only by code executing at a privilege level at least as privileged as **p.**

—A code segment/procedure with privilege level **p** can only be called by a task executing at the same or a lesser privilege level than **p.**

PRIVILEGE LEVELS

At any point in time, a task on the 80376 always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies what the task's privilege level is. A task's CPL may only be changed

by control transfers through gate descriptors to a code segment with a different privilege level. Thus, an application program running at PL = 3 may call an operating system routine at PL = 1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

Selector Privilege (RPL)

The privilege level of a selector is specified by the RPL field. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level of the task for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (numerically larger) level of a task's CPL and a selector's RPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

I/O Privilege

The I/O privilege level (IOPL) lets the operating system code executing at CPL = 0 define the least privileged level at which I/O instructions can be used. An exception 13 (General Protection Violation) is generated if an I/O instruction is attempted when the CPL of the task is less privileged than the IOPL. The IOPL is stored in bits 13 and 14 of the EFLAGS register. The following instructions cause an exception 13 if the CPL is greater than IOPL: IN, INS, OUT, OUTS, STI, CLI and LOCK prefix.

Descriptor Access

There are basically two types of segment accessess: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL as described above.

Any time an instruction loads a data segment register (DS, ES, FS, GS) the 80376 makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segment or readable code segments.

Finally the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL, an exception 13 (general protection fault) is generated.

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL of all other descriptor types or a privilege level violation will cause an exception 13. A stack not present fault causes an exception 12.

PRIVILEGE LEVEL TRANSFERS

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call or a jump to another routine. There are five types of control transfers which are summarized in Table 3.2. Many of these transfers result in a privilege level transfer. Changing privilege levels is done only by control transfers, using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an exception 13.

CALL GATES

Gates provide protected indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures.

Table 3.2. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL	CALL	Call Gate	GDT/LDT
	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0
**NT (Nested Task bit of flag register) = 1

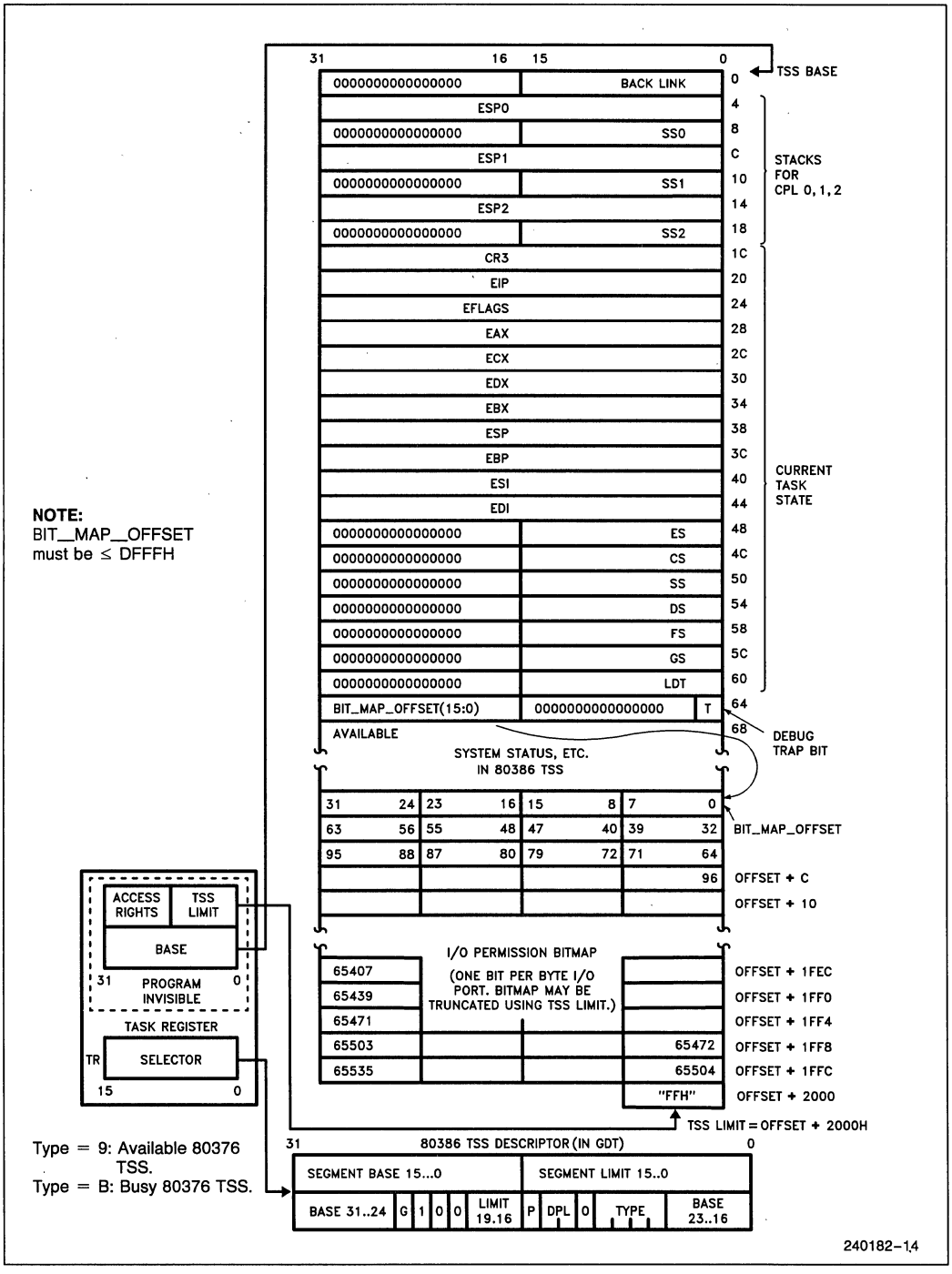


Figure 3.7. 80376 TSS And TSS Registers

TASK SWITCHING

A very important attribute of any multi-tasking operating system is its ability to rapidly switch between tasks or processes. The 80376 directly supports this operation by providing a task switch instruction in hardware. The 80376 task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task. Like transfer of control by gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot. For simple applications, the TSS and task switching may not be used. The TSS or task switch will not be used or occur if no task gates are present in the GDT, LDT or IDT.

The TSS descriptor points to a segment (see Figure 3.7) containing the entire 80376 execution state. A task gate descriptor contains a TSS selector. The limit of an 80376 TSS must be greater than 64H, and can be as large as 16 Mbytes. In the additional TSS space, the operating system is free to store additional information as the reason the task is inactive, the time the task has spent running, and open files belonging to the task. For maximum performance, TSS should start on an even address.

Each Task must have a TSS associated with it. The current TSS is identified by a special register in the 80376 called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with the TSS descriptor is loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The current executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and CR0 register give information about the state of a task which is useful to the operating system. The Nested Task bit, NT, controls the function of the IRET instruction. If NT = 0 the IRET instruction performs the regular return. If NT = 1, IRET performs a task switch operation back to the previous task. The NT bit is set or reset in the following fashion:

> When a CALL or INT instruction initiates a task switch, the new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT (The NT bit will be restored after execution of the interrupt handler). NT may also be set or cleared by POPF or IRET instructions.

The 80376 task state segment is marked busy by changing the descriptor type field from TYPE 9 to TYPE 0BH. Use of a selector that references a busy task state segment causes an exception 13.

The coprocessor's state is not automatically saved when a task switch occurs. The Task Switched Bit, TS, in the CR0 register helps deal with the coprocessor's state in a multi-tasking environment. Whenever the 80376 switches tasks, it sets the TS bit. The 80376 detects the first use of a processor extension instruction after a task switch and causes the processor extension not available exception 7. The exception handler for exception 7 may then decide whether to save the state of the coprocessor.

The T bit in the 80376 TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1 then upon entry to a new task a debug exception 1 will be generated.

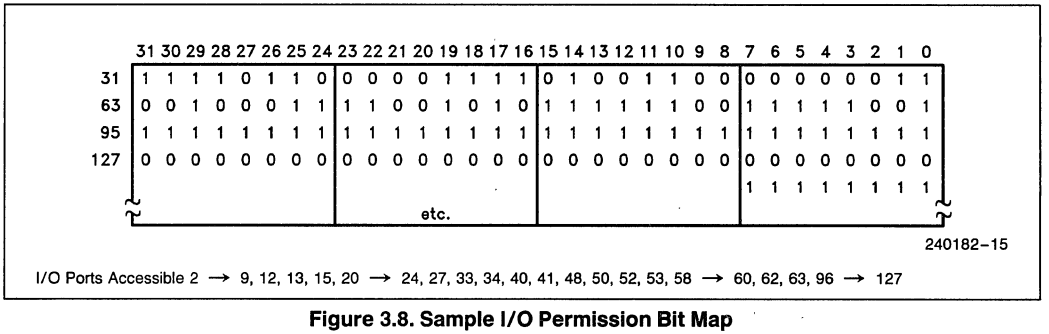


Figure 3.8. Sample I/O Permission Bit Map

PROTECTION AND I/O PERMISSION BIT MAP

The I/O instructions that directly refer to addresses in the processor's I/O space are IN, INS, OUT and OUTS. The 80376 has the ability to selectively trap references to specific I/O addresses. The structure that enables selective trapping is the ***I/O Permission Bit Map*** in the TSS segment (see Figures 3.7 and 3.8). The I/O permission map is a bit vector. The size of the map and its location in the TSS segment are variable. The processor locates the I/O permission map by means of the **I/O map base** field in the fixed portion of the TSS. The **I/O map base** field is 16 bits wide and contains the offset of the beginning of the I/O permission map.

If an I/O instruction (IN, INS, OUT or OUTS) is encountered, the processor first checks whether CPL ≤ IOPL. If this condition is true, the I/O operation may proceed. If not true, the processor checks the I/O permission map.

Each bit in the map corresponds to an I/O port byte address; for example, the bit for port 41 is found at **I/O map base** +5 linearly, (5 × 8 = 40), bit offset 1. The processor tests all the bits that correspond to the I/O addresses spanned by an I/O operation; for example, a double word operation tests four bits corresponding to four adjacent byte addresses. If any tested bit is set, the processor signals a general protection exception. If all the tested bits are zero, the I/O operations may proceed.

It is not necessary for the I/O permission map to represent all the I/O addresses. I/O addresses not spanned by the map are treated as if they had one-bits in the map. The **I/O map base** should be at least one byte less than the TSS limit and the last byte beyond the I/O mapping information must contain all 1's.

Because the I/O permission map is in the TSS segment, different tasks can have different maps. Thus, the operating system can allocate ports to a task by changing the I/O permission map in the task's TSS.

IMPORTANT IMPLEMENTATION NOTE:
Beyond the last byte of I/O mapping information in the I/O permission bit map **must** be a byte containing all 1's. The byte of all 1's must be within the limit of the 80376's TSS segment (see Figure 3.7).

4.0 FUNCTIONAL DATA

The Intel 80376 embedded processor features a straightforward functional interface to the external hardware. The 80376 has separate parallel buses for data and address. The data bus is 16 bits in width, and bidirectional. The address bus outputs 24-bit address values using 23 address lines and two-byte enable signals.

The 80376 has two selectable address bus cycles: pipelined and non-pipelined. The pipelining option allows as much time as possible for data access by

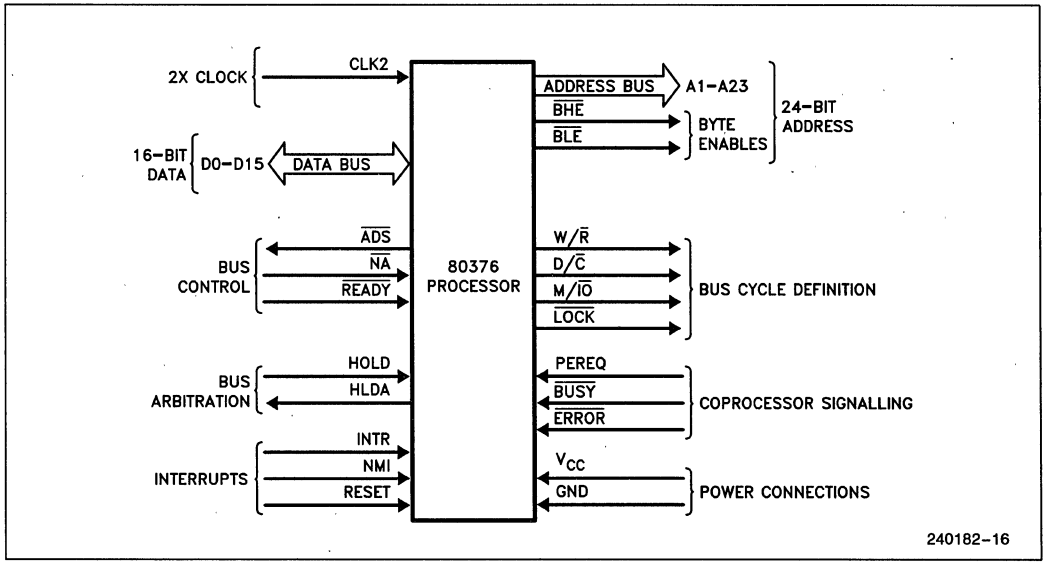


Figure 4.1. Functional Signal Groups

starting the pending bus cycle before the present bus cycle is finished. A non-pipelined bus cycle gives the highest bus performance by executing every bus cycle in two processor clock cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor, or from processor to system. 80376 bus cycles perform data transfer in a minimum of only two clock periods. On a 16-bit data bus, the maximum 80376 transfer bandwidth at 16 MHz is therefore 16 Mbytes/sec. However, any bus cycle will be extended for more than two clock periods if external hardware withholds acknowledgement of the cycle.

The 80376 can relinquish control of its local buses to allow mastership by other devices, such as direct memory access (DMA) channels. When relinquished, HLDA is the only output pin driven by the 80376, providing near-complete isolation of the processor from its system (all other output pins are in a float condition).

4.1 Signal Description Overview

Ahead is a brief description of the 80376 input and output signals arranged by functional groups.

The signal descriptions sometimes refer to A.C. timing parameters, such as "t_{25} Reset Setup Time" and "t_{26} Reset Hold Time." The values of these parameters can be found in Tables 6.4 and 6.5.

CLOCK (CLK2)

CLK2 provides the fundamental timing for the 80376. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two

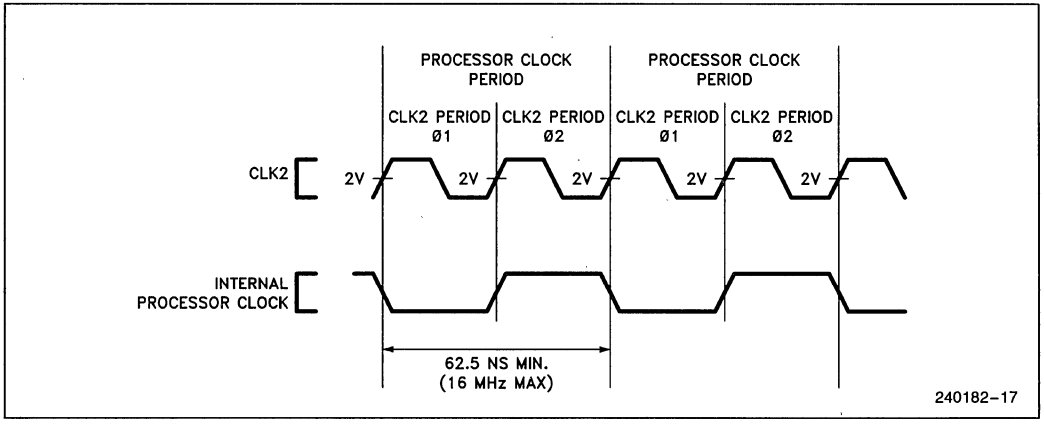


Figure 4.2. CLK2 Signal and Internal Processor Clock

phases, "phase one" and "phase two". Each CLK2 period is a phase of the internal clock. Figure 4.2 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the falling edge of the RESET signal meets the applicable setup and hold times t_{25} and t_{26}.

DATA BUS (D_{15}–D_0)

These three-state bidirectional signals provide the general purpose data path between the 80376 and other devices. The data bus outputs are active HIGH and will float during bus hold acknowledge. Data bus reads require that read-data setup and hold times t_{21} and t_{22} be met relative to CLK2 for correct operation.

ADDRESS BUS ($\overline{BHE}$, $\overline{BLE}$, A_{23}–A_1)

These three-state outputs provide physical memory addresses or I/O port addresses. A_{23}–A_{16} are LOW during I/O transfers except for I/O transfers automatically generated by coprocessor instructions. During coprocessor I/O transfers, A_{22}–A_{16} are driven LOW, and A_{23} is driven HIGH so that this address line can be used by external logic to generate the coprocessor select signal. Thus, the I/O address driven by the 80376 for coprocessor commands is 8000F8H, and the I/O address driven by the 80376 processor for coprocessor data is 8000FCH or 8000FEH.

The address bus is capable of addressing 16 Mbytes of physical memory space (000000H through 0FFFFFFH), and 64 Kbytes of I/O address space (000000H through 00FFFFH) for programmed I/O. The address bus is active HIGH and will float during bus hold acknowledge.

The Byte Enable outputs $\overline{BHE}$ and $\overline{BLE}$ directly indicate which bytes of the 16-bit data bus are involved with the current transfer. $\overline{BHE}$ applies to D_{15}–D_8 and $\overline{BLE}$ applies to D_7–D_0. If both $\overline{BHE}$ and $\overline{BLE}$ are asserted, then 16 bits of data are being transferred. See Table 4.1 for a complete decoding of these signals. The byte enables are active LOW and will float during bus hold acknowledge.

Table 4.1. Byte Enable Definitions

$\overline{BHE}$	$\overline{BLE}$	Function
0	0	Word Transfer
0	1	Byte Transfer on Upper Byte of the Data Bus, D_{15}–D_8
1	0	Byte Transfer on Lower Byte of the Data Bus, D_7–D_0
1	1	Never Occurs

BUS CYCLE DEFINITION SIGNALS (W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$)

These three-state outputs define the type of bus cycle being performed: W/$\overline{R}$ distinguishes between write and read cycles, D/$\overline{C}$ distinguishes between data and control cycles, M/$\overline{IO}$ distinguishes between memory and I/O cycles, and $\overline{LOCK}$ distinguishes between locked and unlocked bus cycles. All of these signals are active LOW and will float during bus acknowledge.

The primary bus cycle definition signals are W/$\overline{R}$, D/$\overline{C}$ and M/$\overline{IO}$, since these are the signals driven valid as $\overline{ADS}$ (Address Status output) becomes active. The $\overline{LOCK}$ signal is driven valid at the same time the bus cycle begins, which due to address pipelining, could be after $\overline{ADS}$ becomes active. Exact bus cycle definitions, as a function of W/$\overline{R}$, D/$\overline{C}$ and M/$\overline{IO}$ are given in Table 4.2.

$\overline{LOCK}$ indicates that other system bus masters are not to gain control of the system bus while it is active. $\overline{LOCK}$ is activated on the CLK2 edge that begins the first locked bus cycle (i.e., it is not active at the same time as the other bus cycle definition pins) and is deactivated when ready is returned to the end of the last bus cycle which is to be locked. The beginning of a bus cycle is determined when $\overline{READY}$ is returned in a previous bus cycle and another is pending ($\overline{ADS}$ is active) or the clock in which $\overline{ADS}$ is driven active if the bus was idle. This means that it follows more closely with the write data rules when it is valid, but may cause the bus to be locked longer than desired. The $\overline{LOCK}$ signal may be explicitly activated by the LOCK prefix on certain instructions. $\overline{LOCK}$ is always asserted when executing the XCHG instruction, during descriptor updates, and during the interrupt acknowledge sequence.

BUS CONTROL SIGNALS ($\overline{ADS}$, $\overline{READY}$, $\overline{NA}$)

The following signals allow the processor to indicate when a bus cycle has begun, and allow other system hardware to control address pipelining and bus cycle termination.

Address Status ($\overline{ADS}$)

This three-state output indicates that a valid bus cycle definition and address (W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{BHE}$, $\overline{BLE}$ and A_{23}–A_1) are being driven at the 80376 pins. $\overline{ADS}$ is an active LOW output. Once $\overline{ADS}$ is driven active, valid address, byte enables, and definition signals will not change. In addition, $\overline{ADS}$ will remain active until its associated bus cycle begins (when $\overline{READY}$ is returned for the previous bus cycle when running pipelined bus cycles). $\overline{ADS}$ will float during bus hold acknowledge. See sections **Non-Pipelined Bus Cycles** and **Pipelined Bus Cycles** for additional information on how $\overline{ADS}$ is asserted for different bus states.

Transfer Acknowledge ($\overline{READY}$)

This input indicates the current bus cycle is complete, and the active bytes indicated by $\overline{BHE}$ and $\overline{BLE}$ are accepted or provided. When $\overline{READY}$ is sampled active during a read cycle or interrupt acknowledge cycle, the 80376 latches the input data and terminates the cycle. When $\overline{READY}$ is sampled active during a write cycle, the processor terminates the bus cycle.

Table 4.2. Bus Cycle Definition

M/$\overline{IO}$	D/$\overline{C}$	W/$\overline{R}$	Bus Cycle Type	Locked?
0	0	0	INTERRUPT ACKNOWLEDGE	Yes
0	0	1	Does Not Occur	—
0	1	0	I/O DATA READ	No
0	1	1	I/O DATA WRITE	No
1	0	0	MEMORY CODE READ	No
1	0	1	HALT: Address = 2, $\overline{BHE}$ = 1, $\overline{BLE}$ = 0 — SHUTDOWN: Address = 0, $\overline{BHE}$ = 1, $\overline{BLE}$ = 0	No
1	1	0	MEMORY DATA READ	Some Cycles
1	1	1	MEMORY DATA WRITE	Some Cycles

$\overline{READY}$ is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. $\overline{READY}$ must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, $\overline{READY}$ must always meet setup and hold times t_{19} and t_{20} for correct operation.

Next Address Request ($\overline{NA}$)

This is used to request pipelining. This input indicates the system is prepared to accept new values of $\overline{BHE}$, $\overline{BLE}$, A_{23}–A_1, W/$\overline{R}$, D/$\overline{C}$ and M/$\overline{IO}$ from the 80376 even if the end of the current cycle is not being acknowledged on $\overline{READY}$. If this input is active when sampled, the next bus cycle's address and status signals are driven onto the bus, provided the next bus request is already pending internally. $\overline{NA}$ is ignored in clock cycles in which $\overline{ADS}$ or $\overline{READY}$ is activated. This signal is active LOW and must satisfy setup and hold times t_{15} and t_{16} for correct operation. See **Pipelined Bus Cycles** and **Read and Write Cycles** for additional information.

BUS ARBITRATION SIGNALS (HOLD, HLDA)

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See **Entering and Exiting Hold Acknowledge** for additional information.

Bus Hold Request (HOLD)

This input indicates some device other than the 80376 requires bus mastership. When control is granted, the 80376 floats A_{23}–A_1, $\overline{BHE}$, $\overline{BLE}$, D_{15}–D_0, $\overline{LOCK}$, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$ and $\overline{ADS}$, and then activates HLDA, thus entering the bus hold acknowledge state. The local bus will remain granted to the requesting master until HOLD becomes inactive. When HOLD becomes inactive, the 80376 will deactivate HLDA and drive the local bus (at the same time), thus terminating the hold acknowledge condition.

HOLD must remain asserted as long as any other device is a local bus master. External pull-up resistors may be required when in the hold acknowledge state since none of the 80376 floated outputs have internal pull-up resistors. See **Resistor Recommendations** for additional information. HOLD is not recognized while RESET is active but is recognized during the time between the high-to-low transistion of RESET and the first instruction fetch. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the hold acknowledge (high-impedance) state.

HOLD is a level-sensitive, active HIGH, synchronous input. HOLD signals must always meet setup and hold times t_{23} and t_{24} for correct operation.

Bus Hold Acknowledge (HLDA)

When active (HIGH), this output indicates the 80376 has relinquished control of its local bus in response to an asserted HOLD signal, and is in the bus Hold Acknowledge state.

The Bus Hold Acknowledge state offers near-complete signal isolation. In the Hold Acknowledge state, HLDA is the only signal being driven by the 80376. The other output signals or bidirectional signals (D_{15}–D_0, $\overline{BHE}$, $\overline{BLE}$, A_{23}–A_1, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$ and $\overline{ADS}$) are in a high-impedance state so the requesting bus master may control them. These pins remain OFF throughout the time that HLDA remains active (see Table 4.3). Pull-up resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See **Resistor Recommendations** for additional information.

When the HOLD signal is made inactive, the 80376 will deactivate HLDA and drive the bus. One rising edge on the NMI input is remembered for processing after the HOLD input is negated.

Table 4.3. Output Pin State during HOLD

Pin Value	Pin Names
1	HLDA
Float	$\overline{LOCK}$, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$, $\overline{ADS}$, A_{23}–A_1, $\overline{BHE}$, $\overline{BLE}$, D_{15}–D_0

Hold Latencies

The maximum possible HOLD latency depends on the software being executed. The actual HOLD latency at any time depends on the current bus activity, the state of the $\overline{LOCK}$ signal (internal to the CPU) activated by the $\overline{LOCK}$ prefix, and interrupts. The 80376 will not honor a HOLD request until the current bus operation is complete.

The 80376 breaks 32-bit data or I/O accesses into 2 internally locked 16-bit bus cycles; the $\overline{LOCK}$ signal is not asserted. The 80376 breaks unaligned 16-bit or 32-bit data or I/O accesses into 2 or 3 internally locked 16-bit bus cycles. Again the $\overline{LOCK}$ signal is not asserted but a HOLD request will not be recognized until the end of the entire transfer.

Wait states affect HOLD latency. The 80376 will not honor a HOLD request until the end of the current bus operation, no matter how many wait states are required. Systems with DMA where data transfer is critical must insure that $\overline{\text{READY}}$ returns sufficiently soon.

COPROCESSOR INTERFACE SIGNALS (PEREQ, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$)

In the following sections are descriptions of signals dedicated to the numeric coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, these following signals control communication between the 80376 and the 80387SX processor extension.

Coprocessor Request (PEREQ)

When asserted (HIGH), this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the 80376. In response, the 80376 transfers information between the coprocessor and memory. Because the 80376 has internally stored the coprocessor opcode being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is a level-sensitive active HIGH asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This signal is provided with a weak internal pull-down resistor of around 20 KΩ to ground so that it will not float active when left unconnected.

Coprocessor Busy ($\overline{\text{BUSY}}$)

When asserted (LOW), this input indicates the coprocessor is still executing an instruction, and is not yet able to accept another. When the 80376 encounters any coprocessor instruction which operates on the numerics stack (e.g. load, pop, or arithmetic operation), or the WAIT instruction, this input is first automatically sampled until it is seen to be inactive. This sampling of the $\overline{\text{BUSY}}$ input prevents overrunning the execution of a previous coprocessor instruction.

The F(N)INIT, F(N)CLEX coprocessor instructions are allowed to execute even if $\overline{\text{BUSY}}$ is active, since these instructions are used for coprocessor initialization and exception-clearing.

$\overline{\text{BUSY}}$ is an active LOW, level-sensitive asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 KΩ to V_{CC} so that it will not float active when left unconnected.

$\overline{\text{BUSY}}$ serves an additional function. If $\overline{\text{BUSY}}$ is sampled LOW at the falling edge of RESET, the 80376 processor performs an internal self-test (see **Bus Activity During and Following Reset.** If $\overline{\text{BUSY}}$ is sampled HIGH, no self-test is performed.

Coprocessor Error ($\overline{\text{ERROR}}$)

When asserted (LOW), this input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the 80376 when a coprocessor instruction is encountered, and if active, the 80376 generates exception 16 to access the error-handling software.

Several coprocessor instructions, generally those which clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the 80376 generating exception 16 even if $\overline{\text{ERROR}}$ is active. These instructions are FNINIT, FNCLEX, FNSTSW, FNSTSWAX, FNSTCW, FNSTENV and FNSAVE.

$\overline{\text{ERROR}}$ is an active LOW, level-sensitive asynchronous signal. Setup and hold times t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 KΩ to V_{CC} so that it will not float active when left unconnected.

INTERRUPT SIGNALS (INTR, NMI, RESET)

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

Maskable Interrupt Request (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the 80376 Flag Register IF bit. When the 80376 responds to the INTR input, it performs two interrupt acknowledge bus cycles and, at the end of the second, latches an 8-bit interrupt vector on D_7-D_0 to identify the source of the interrupt.

INTR is an active HIGH, level-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of an INTR request, INTR should remain active until the first interrupt acknowledge bus cycle begins. INTR is sampled at the beginning of every instruction. In order to be recognized at a particular instruction boundary, INTR must be active at least eight CLK2 clock periods before the beginning of the execution of the instruction. If recognized, the 80376 will begin execution of the interrupt.

Non-Maskable Interrupt Request (NMI)

This input indicates a request for interrupt service which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are performed when processing NMI.

NMI is an active HIGH, rising edge-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of NMI, it must be inactive for at least eight CLK2 periods, and then be active for at least eight CLK2 periods before the beginning of the execution of an instruction.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

Interrupt Latency

The time that elapses before an interrupt request is serviced (interrupt latency) varies according to several factors. This delay must be taken into account by the interrupt source. Any of the following factors can affect interrupt latency:

1. If interrupts are masked, and INTR request will not be recognized until interrupts are reenabled.
2. If an NMI is currently being serviced, an incoming NMI request will not be recognized until the 80376 encounters the IRET instruction.
3. An interrupt request is recognized only on an instruction boundary of the 80376 ***Execution Unit*** except for the following cases:
   - — Repeat string instructions can be interrupted after each iteration.
   - — If the instruction loads the Stack Segment register, an interrupt is not processed until after the following instruction, which should be an ESP load. This allows the entire stack pointer to be loaded without interruption.
   - — If an instruction sets the interrupt flag (enabling interrupts), an interrupt is not processed until after the next instruction.

   The longest latency occurs when the interrupt request arrives while the 80376 processor is executing a long instruction such as multiplication, division or a task-switch.
4. Saving the Flags register and CS:EIP registers.
5. If interrupt service routine requires a task switch, time must be allowed for the task switch.
6. If the interrupt service routine saves registers that are not automatically saved by the 80376.

RESET

This input signal suspends any operation in progress and places the 80376 in a known reset state. The 80376 is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self-test). When RESET is active, all other input pins except $\overline{FLT}$ are ignored, and all other bus pins are driven to an idle bus state as shown in Table 4.4. If RESET and HOLD are both active at a point in time, RESET takes priority even if the 80376 was in a Hold Acknowledge state prior to RESET active.

RESET is an active HIGH, level-sensitive synchronous signal. Setup and hold times, t_{25} and t_{26}, must be met in order to assure proper operation of the 80376.

Table 4.4. Pin State (Bus Idle) during RESET

Pin Name	Signal Level during RESET
$\overline{ADS}$	1
$D_{15}-D_0$	Float
$\overline{BHE}$, $\overline{BLE}$	0
$A_{23}-A_1$	1
W/$\overline{R}$	0
D/$\overline{C}$	1
M/$\overline{IO}$	0
$\overline{LOCK}$	1
HLDA	0

4.2 Bus Transfer Mechanism

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte and word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two physical bus cycles are performed as required for unaligned operand transfers.

The 80376 processor address signals are designed to simplify external system hardware. $\overline{BHE}$ and $\overline{BLE}$ provide linear selects for the two bytes of the 16-bit data bus.

Byte Enable outputs $\overline{BHE}$ and $\overline{BLE}$ are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 4.5.

Table 4.5. Byte Enables and Associated Data and Operand Bytes

Byte Enable	Associated Data Bus Signals
$\overline{BHE}$	$D_{15}-D_8$ (Byte 1—Most Significant)
$\overline{BLE}$	D_7-D_0 (Byte 0—Least Significant)

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See **Bus Functional Description** for additional information.

4.3 Memory and I/O Spaces

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. As shown in Figure 4.3, physical memory addresses range from 000000H to 0FFFFFFH (16 Mbytes) and I/O addresses from 000000H to 00FFFFH (64 Kbytes). Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 8000F8H to 8000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A_{23} and M/$\overline{IO}$ signals.

OPERAND ALIGNMENT

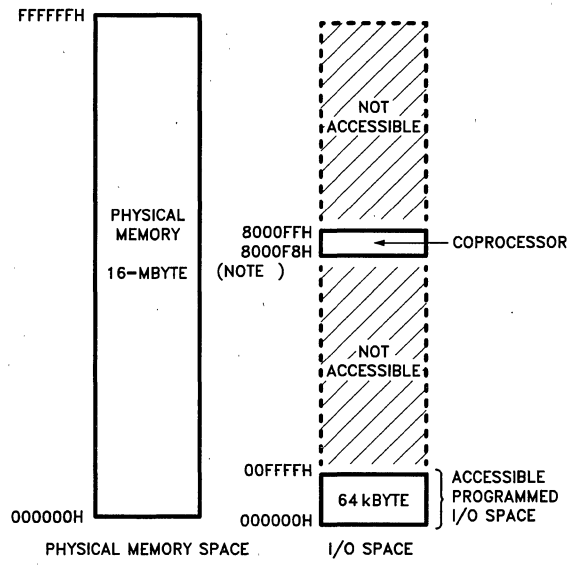
With the flexibility of memory addressing on the 80376, it is possible to transfer a logical operand that spans more than one physical Dword or word of memory or I/O. Examples are 32-bit Dword or 16-bit word operands beginning at addresses not evenly divisible by 2.

Operand alignment and size dictate when multiple bus cycles are required. Table 4.6 describes the transfer cycles generated for all combinations of logical operand lengths and alignment.

Table 4.6. Transfer Bus Cycles for Bytes, Words and Dwords

	Byte-Length of Logical Operand								
	1	2				4			
Physical Byte Address in Memory (Low-Order Bits)	xx	00	01	10	11	00	01	10	11
Transfer Cycles	b	w	lb, hb	w	hb, l,b	lw, hw	hb, lb, mw	hw, lw	mw, hb, lb

Key: b = byte transfer
w = word transfer
l = low-order portion
m = mid-order portion
x = don't care
h = high-order portion



NOTE:
Since A_{23} is HIGH during automatic communication with coprocessor, A_{23} HIGH and M/$\overline{IO}$ LOW can be used to easily generate a coprocessor select signal.

Figure 4.3. Physical Memory and I/O Spaces

4.4 Bus Functional Description

The 80376 has separate, parallel buses for data and address. The data bus is 16 bits in width, and bidirectional. The address bus provides a 24-bit value using 23 signals for the 23 upper-order address bits and 2 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled by several definition signals.

The definition of each bus cycle is given by three signals: M/$\overline{IO}$, W/$\overline{R}$ and D/$\overline{C}$. At the same time, a valid address is present on the byte enable signals, $\overline{BHE}$ and $\overline{BLE}$, and the other address signals A_{23}–A_1. A status signal, $\overline{ADS}$, indicates when the 80376 issues a new bus cycle definition and address.

Collectively, the address bus, data bus and all associated control signals are referred to simply as "the bus". When active, the bus performs one of the bus cycles below:

1. Read from memory space
2. Locked read from memory space
3. Write to memory space
4. Locked write to memory space
5. Read from I/O space (or coprocessor)
6. Write to I/O space (or coprocessor)
7. Interrupt acknowledge (always locked)
8. Indicate halt, or indicate shutdown

Table 4.2 shows the encoding of the bus cycle definition signals for each bus cycle. See **Bus Cycle Definition Signals** for additonal information.

When the 80376 bus is not performing one of the activities listed above, it is either Idle or in the Hold Acknowledge state, which may be detected by external circuitry. The idle state can be identified by the 80376 giving no further assertions on its address strobe output ($\overline{ADS}$) since the beginning of its most recent bus cycle, and the most recent bus cycle having been terminated. The hold acknowledge state is identified by the 80376 asserting its hold acknowledge (HLDA) output.

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

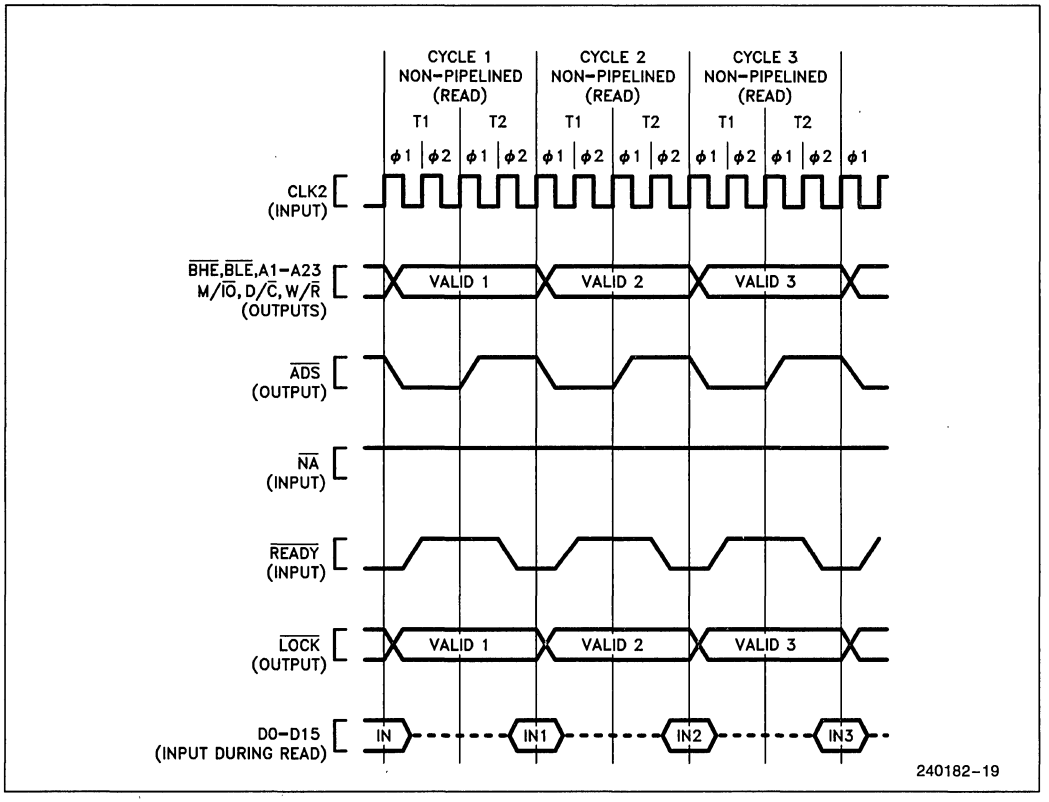


Figure 4.4. Fastest Read Cycles with Non-Pipelined Timing

The fastest 80376 bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown by Figure 4.4. The bus states in each cycle are named T1 and T2. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough.

Every bus cycle continues until it is acknowledged by the external system hardware, using the 80376 $\overline{\text{READY}}$ input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If $\overline{\text{READY}}$ is not immediately asserted however, T2 states are repeated indefinitely until the $\overline{\text{READY}}$ input is sampled active.

The pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined cycles are selectable on a cycle-by-cycle basis with the Next Address ($\overline{\text{NA}}$) input.

When pipelining is selected the address ($\overline{\text{BHE}}$, $\overline{\text{BLE}}$ and A_{23}–A_1) and definition (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$ and $\overline{\text{LOCK}}$) of the next cycle are available before the end of the current cycle. To signal their availability, the 80376 address status output ($\overline{\text{ADS}}$) is asserted. Figure 4.5 illustrates the fastest read cycles with pipelined timing.

Note from Figure 4.5 the fastest bus cycles using pipelining require only two bus states, named **T1P** and **T2P**. Therefore pipelined cycles allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased by one T-state time compared to that of a non-pipelined cycle.

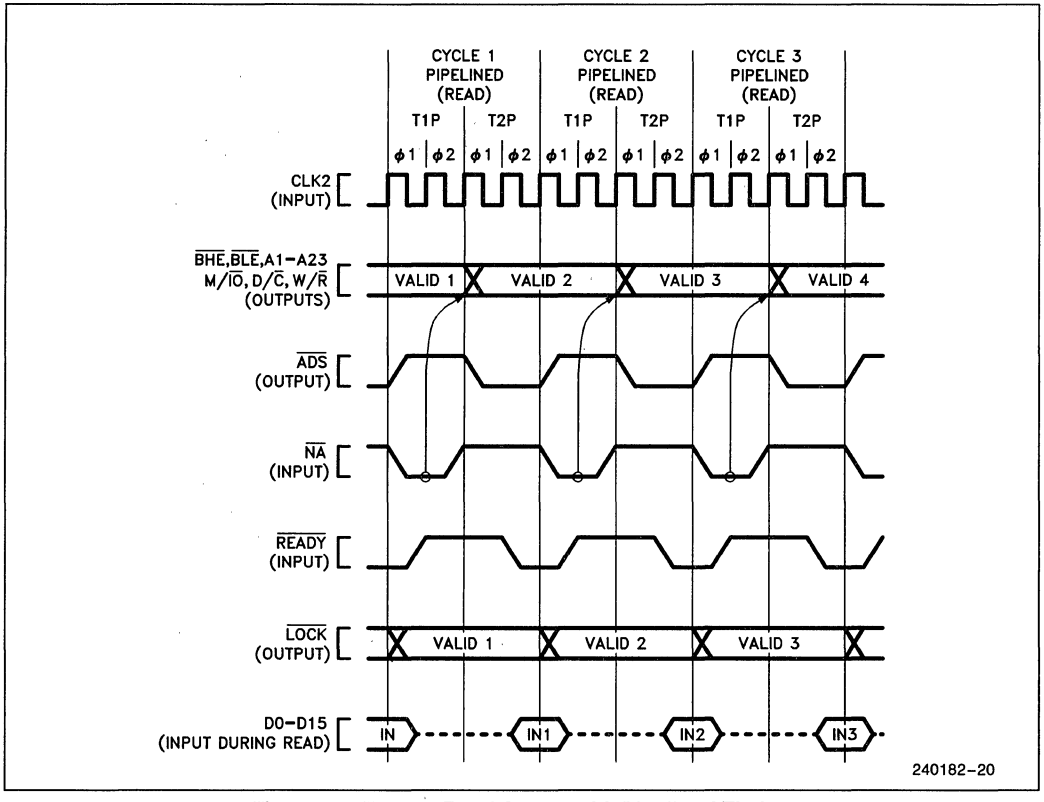


Figure 4.5. Fastest Read Cycles with Pipelined Timing

READ AND WRITE CYCLES

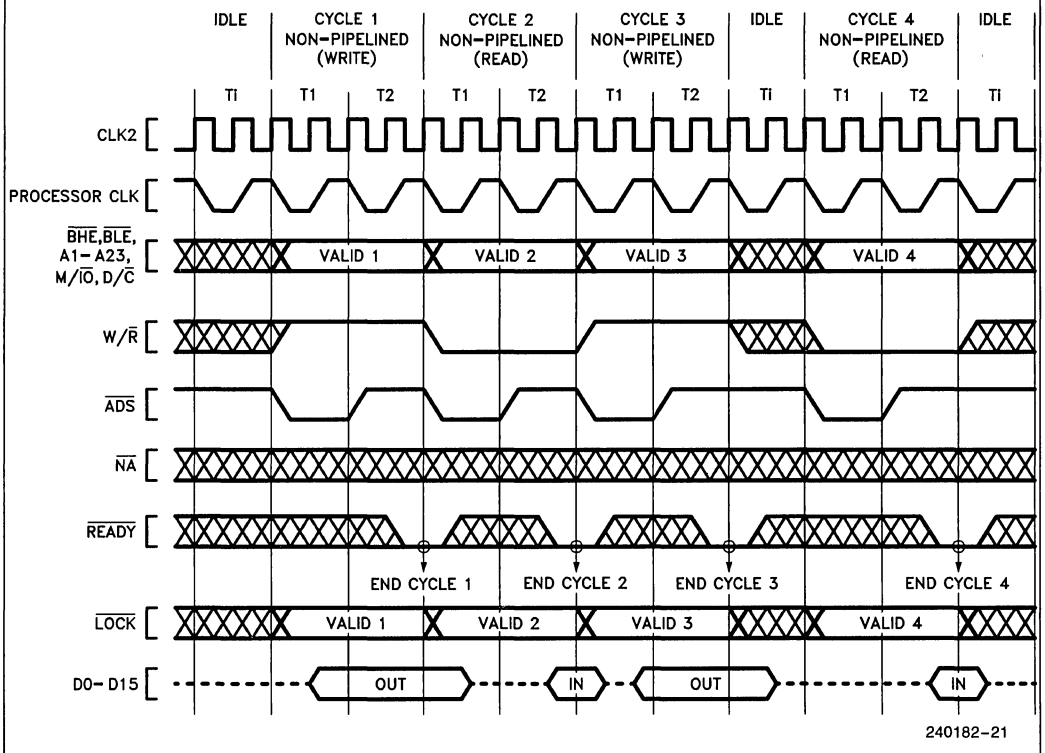
Data transfers occur as a result of bus cycles, classified as read or write cycles. During read cycles, data is transferred from an external device to the processor. During write cycles, data is transferred from the processor to an external device.

Two choices of bus cycle timing are dynamically selectable: non-pipelined or pipelined. After an idle bus state, the processor always uses non-pipelined timing. However the $\overline{NA}$ (Next Address) input may be asserted to select pipelined timing for the next bus cycle. When pipelining is selected and the 80376 has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by $\overline{READY}$.

Terminating a read or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the $\overline{READY}$ input. Until acknowledged, the processor inserts wait states into the bus cycle, to allow adjustment for the speed of any external device. External hardware, which has decoded the address and bus cycle type, asserts the $\overline{READY}$ input at the appropriate time.

At the end of the second bus state within the bus cycle, $\overline{READY}$ is sampled. At that time, if external hardware acknowledges the bus cycle by asserting $\overline{READY}$, the bus cycle terminates as shown in Figure 4.6. If $\overline{READY}$ is negated as in Figure 4.7, the 80376 executes another bus state (a wait state) and $\overline{READY}$ is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by $\overline{READY}$ asserted.

When the current cycle is acknowledged, the 80376 terminates it. When a read cycle is acknowledged, the 80376 latches the information present at its data pins. When a write cycle is acknowledged, the write data of the 80376 remains valid throughout phase one of the next bus state, to provide write data hold time.



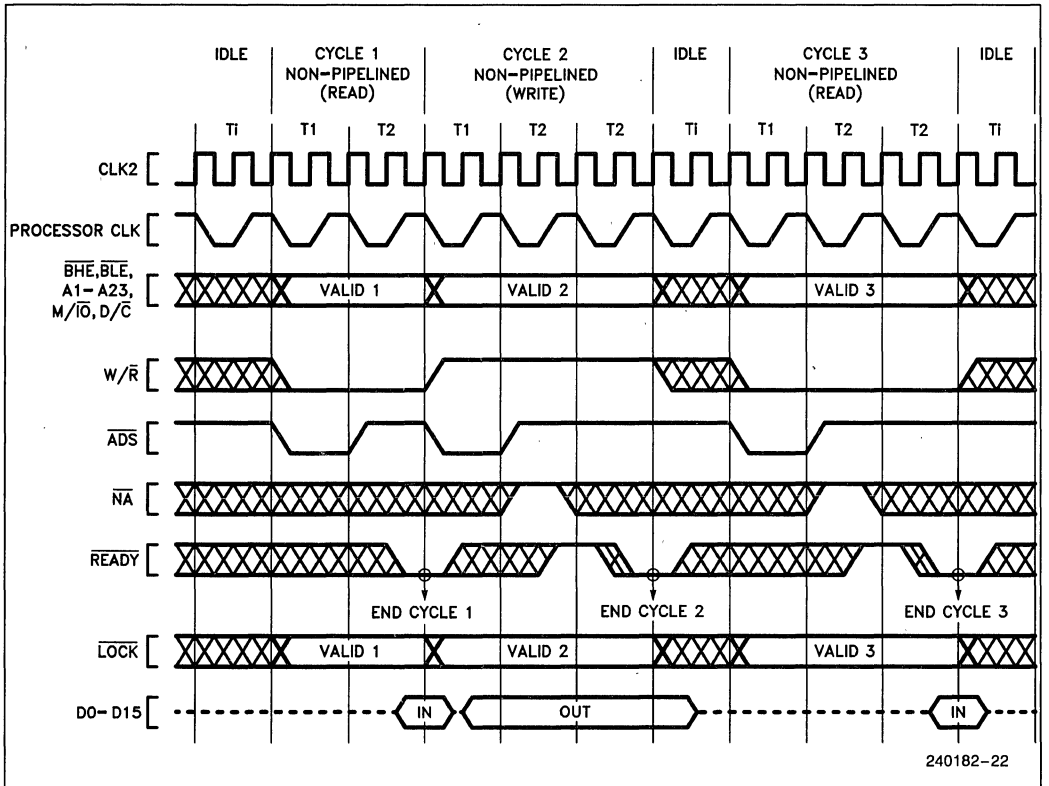
Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 4.6. Various Non-Pipelined Bus Cycles (Zero Wait States)

Non-Pipelined Bus Cycles

Any bus cycle may be performed with non-pipelined timing. For example, Figure 4.6 shows a mixture of non-pipelined read and write cycles. Figure 4.6 shows that the fastest possible non-pipelined cycles have two bus states per bus cycle. The states are named T1 and T2. In phase one of T1, the address signals and bus cycle definition signals are driven valid and, to signal their availability, address strobe ($\overline{ADS}$) is simultaneously asserted.

During read or write cycles, the data bus behaves as follows. If the cycle is a read, the 80376 floats its data signals to allow driving by the external device being addressed. **The 80376 requires that all data bus pins be at a valid logic state (HIGH or LOW) at the end of each read cycle, when $\overline{READY}$ is asserted. The system MUST be designed to meet this requirement.** If the cycle is a write, data signals are driven by the 80376 beginning in phase two of T1 until phase one of the bus state following cycle acknowledgement.



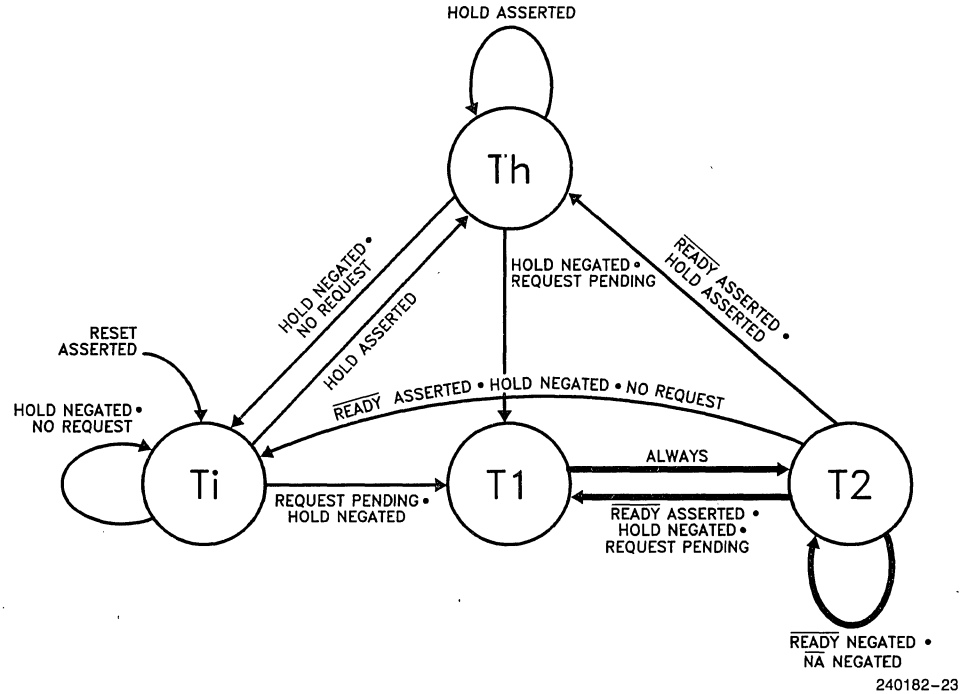
Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 4.7. Various Non-Pipelined Bus Cycles (Various Number of Wait States)

Figure 4.7 illustrates non-pipelined bus cycles with one wait state added to Cycles 2 and 3. $\overline{\text{READY}}$ is sampled inactive at the end of the first T2 in Cycles 2 and 3. Therefore Cycles 2 and 3 have T2 repeated again. At the end of the second T2, $\overline{\text{READY}}$ is sampled active.

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states. When wait states are added and it is desirable to maintain non-pipelined timing, it is necessary to negate $\overline{\text{NA}}$ during each T2 state except the last one, as shown in Figure 4.7, Cycles 2 and 3. If $\overline{\text{NA}}$ is sampled active during a T2 other than the last one, the next state would be T2I or T2P instead of another T2.

When address pipelining is not used, the bus states and transitions are completely illustrated by Figure 4.8. The bus transitions between four possible states, T1, T2, T_i, and T_h. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise the bus may be idle, T_i, or in the hold acknowledge state T_h.



Bus States:
T1—first clock of a non-pipelined bus cycle (80376 drives new address and asserts $\overline{ADS}$).
T2—subsequent clocks of a bus cycle when $\overline{NA}$ has not been sampled asserted in the current bus cycle.
Ti—idle state.
Th—hold acknowledge state (80376 asserts HLDA).
The fastest bus cycle consists of two states: T1 and T2.
Four basic bus states describe bus operation when not using pipelined address.

Figure 4.8. 80376 Bus States (Not Using Pipelined Address)

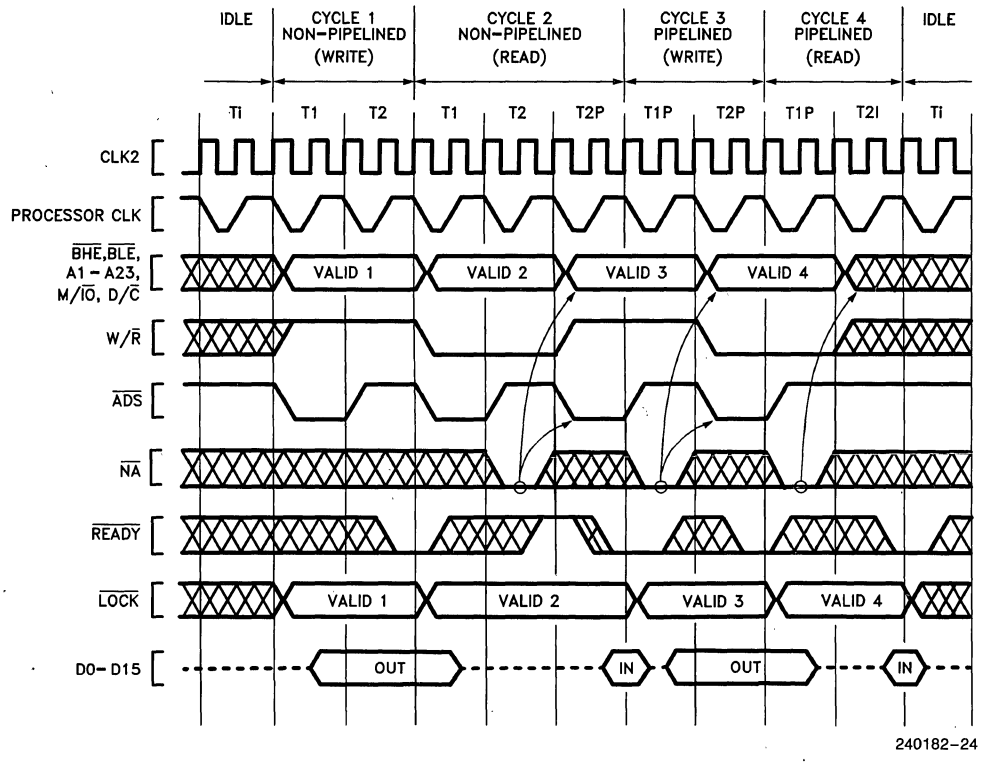
Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and $\overline{NA}$ is inactive, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle if a bus request is pending internally, or T_i if there is no bus request pending, or T_h if the HOLD input is being asserted.

Use of pipelining allows the 80376 to enter three additional bus states not shown in Figure 4.8. Figure 4.12 is the complete bus state diagram, including pipelined cycles.

Pipelined Bus Cycles

Pipelining is the option of requesting the address and the bus cycle definition of the next internally pending bus cycle before the current bus cycle is acknowledged with $\overline{READY}$ asserted. $\overline{ADS}$ is asserted by the 80376 when the next address is issued. The pipelining option is controlled on a cycle-by-cycle basis with the $\overline{NA}$ input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the $\overline{NA}$ input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles $\overline{NA}$ is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 4.9, during which $\overline{NA}$ is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).



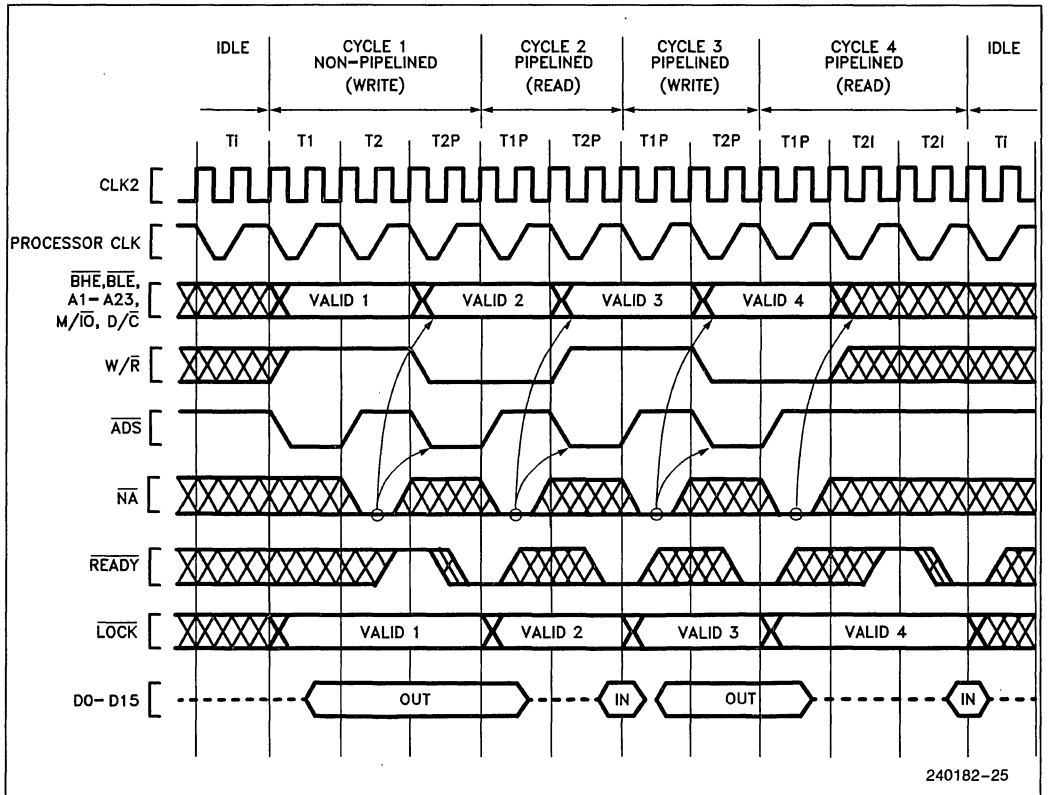
Following any idle bus state (Ti), bus cycles are non-pipelined. Within non-pipelined bus cycles, $\overline{NA}$ is only sampled during wait states. Therefore, to begin pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cylcle 2 above).

Figure 4.9. Transitioning to Pipelining during Burst of Bus Cycles

If $\overline{NA}$ is sampled active, the 80376 is free to drive the address and bus cycle definition of the next bus cycle, and assert $\overline{ADS}$, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of pipelining, the 80376 has the following characteristics:

1. The next address and status may appear as early as the bus state after $\overline{NA}$ was sampled active (see Figures 4.9 or 4.10). In that case, state T2P is entered immediately. However, when there is not an internal bus request already pending, the next address and status will not be available immediately after $\overline{NA}$ is asserted and T2I is entered instead of T2P (see Figure 4.11 Cycle 3). Provided the current bus cycle isn't yet acknowledged by $\overline{READY}$ asserted, T2P will be entered as soon as the 80376 does drive the next address and status. External hardware should therefore observe the $\overline{ADS}$ output as confirmation the next address and status are actually being driven on the bus.
2. Any address and status which are validated by a pulse on the 80376 $\overline{ADS}$ output will remain stable on the address pins for at least two processor clock periods. The 80376 cannot produce a new address and status more frequently than every two processor clock periods (see Figures 4.9, 4.10 and 4.11).
3. Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 4.11, Cycle 1).



Following any idle bus state (Ti) the bus cycle is always non-pipelined and $\overline{NA}$ is only sampled during wait states. To start, address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (cycle 1 above). The pipelined cycles (2, 3, 4 above) are shown with various numbers of wait states.

Figure 4.10. Fastest Transition to Pipelined Bus Cycle Following Idle Bus State

The complete bus state transition diagram, including pipelining is given by Figure 4.12. Note it is a superset of the diagram for non-pipelined only, and the three additional bus states for pipelining are drawn in bold.

The fastest bus cycle with pipelining consists of just two bus states, T1P and T2P (recall for non-pipelined it is T1 and T2). T1P is the first bus state of a pipelined cycle.

Initiating and Maintaining Pipelined Bus Cycles

Using the state diagram Figure 4.12, observe the transitions from an idle state, T_i, to the beginning of a pipelined bus cycle T1P. From an idle state, T_i, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided $\overline{NA}$ is asserted and the first bus cycle ends in a T2P state (the address and status for the next bus cycle is driven during T2P). The fastest path from an idle state to a pipelined bus cycle is shown in bold below:

T_i, T_i,	**T1-T2-T2P,**	**T1P-T2P,**
idle states	non-pipelined cycle	pipelined cycle

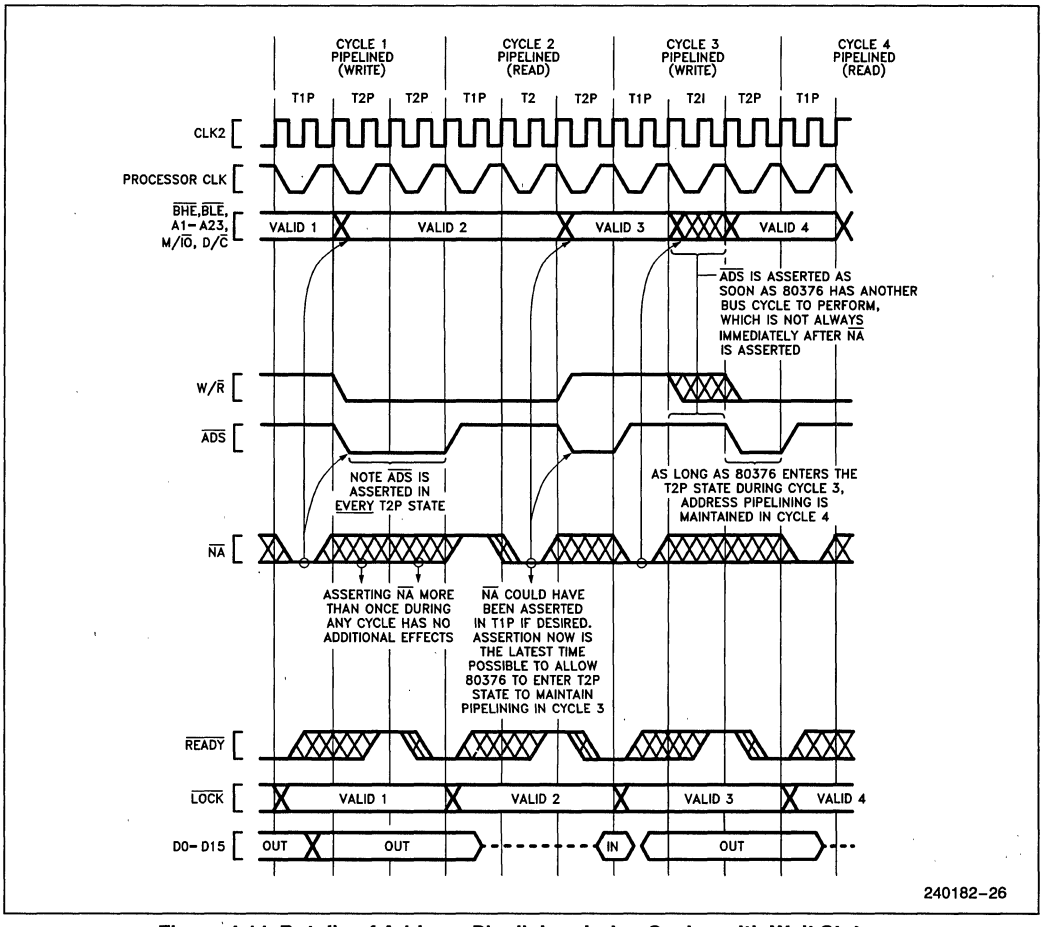


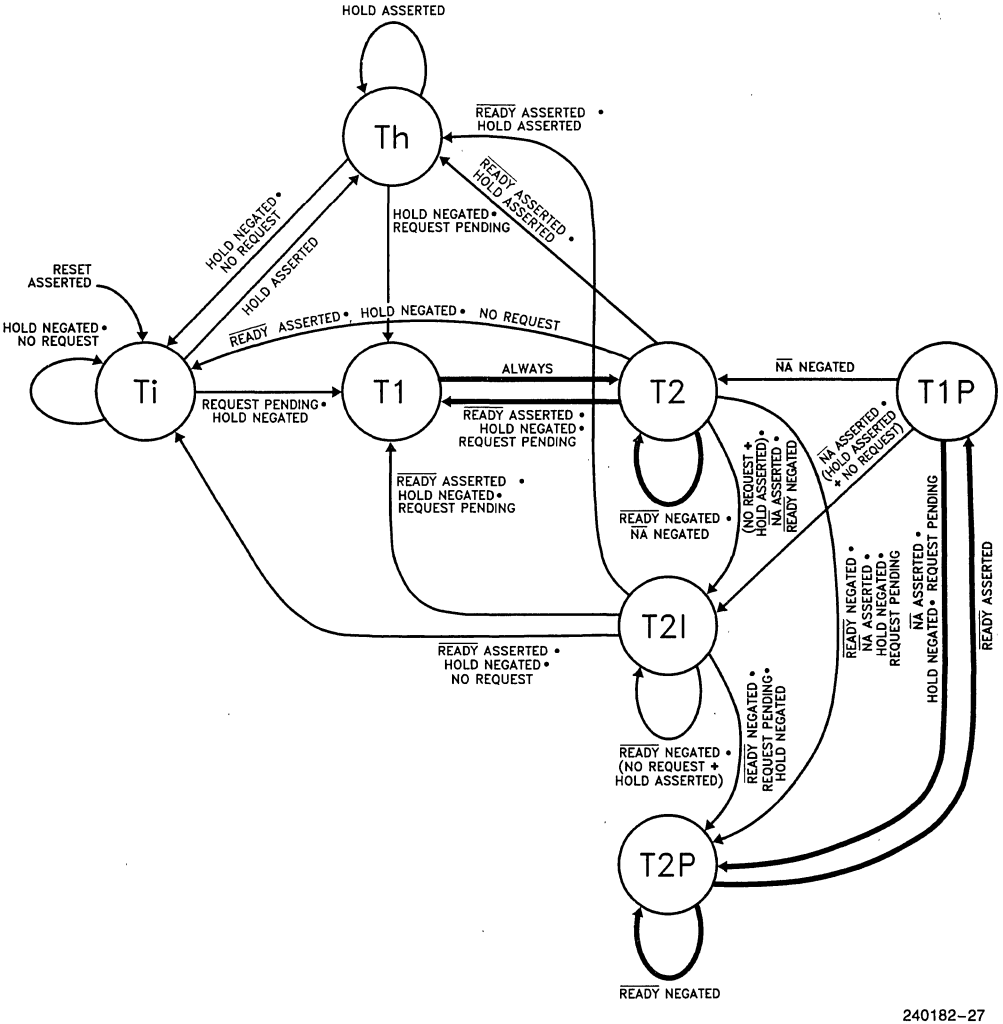
Figure 4.11. Details of Address Pipelining during Cycles with Wait States

T1-T2-T2P are the states of the bus cycle that establishes address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:

T_h, T_h, T_h,	T1-T2-T2P,	T1P-T2P,
hold aknowledge states	non-pipelined cycle	pipelined cycle

The transition to pipelined address is shown functionally by Figure 4.10, Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3 and 4, which are pipelined. The $\overline{NA}$ input is asserted at the appropriate time to select address pipelining for Cycles 2, 3 and 4.

Once a bus cycle is in progress and the current address and status has been valid for one entire bus state, the $\overline{NA}$ input is sampled at the end of every phase one until the bus cycle is acknowledged.



Bus States:
T1—first clock of a non-pipelined bus cycle (80376 drives new address, status and asserts $\overline{ADS}$).
T2—subsequent clocks of a bus cycle when $\overline{NA}$ has not been sampled asserted in the current bus cycle.
T2I—subsequent clocks of a bus cycle when $\overline{NA}$ has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (80376 will not drive new address, status or assert $\overline{ADS}$).
T2P—subsequent clocks of a bus cycle when $\overline{NA}$ has been sampled asserted in the current bus cycle and there is an internal bus request pending (80376 drives new address, status and asserts $\overline{ADS}$).
T1P—first clock of a pipelined bus cycle.
Ti—idle state.
Th—hold acknowledge state (80376 asserts HLDA).
Asserting $\overline{NA}$ for pipelined bus cycles gives access to three more bus states: T2I, T2P and T1P.
Using pipelining the fastest bus cycle consists of T1P and T2P.

Figure 4.12. 80376 Processor Complete Bus States (Including Pipelining)

Sampling begins in T2 during Cycle 1 in Figure 4.10. Once $\overline{NA}$ is sampled active during the current cycle, the 80376 is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 4.10, Cycle 1 for example, the next address and status is driven during state T2P. Thus Cycle 1 makes the transition to pipelined timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as $\overline{READY}$ asserted terminates Cycle 1.

Examples of transition bus cycles are Figure 4.10, Cycle 1 and Figure 4.9, Cycle 2. Figure 4.10 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 4.9, Cycle 2 shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 ($\overline{NA}$ is asserted at that time), and T2P (provided the 80376 has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note that only three states (T1, T2 and T2P) are required in a bus cycle performing a **transition** from non-pipelined into pipelined timing, for example Figure 4.10, Cycle 1. Figure 4.10, Cycles 2, 3 and 4 show that pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting $\overline{NA}$ and detecting that the 80376 enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of $\overline{ADS}$. Figures 4.9 and 4.10 however, each show pipelining ending after Cycle 4 because Cycle 4 ends in T2I. This indicates the 80376 didn't have an internal bus request prior to the acknowledgement of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

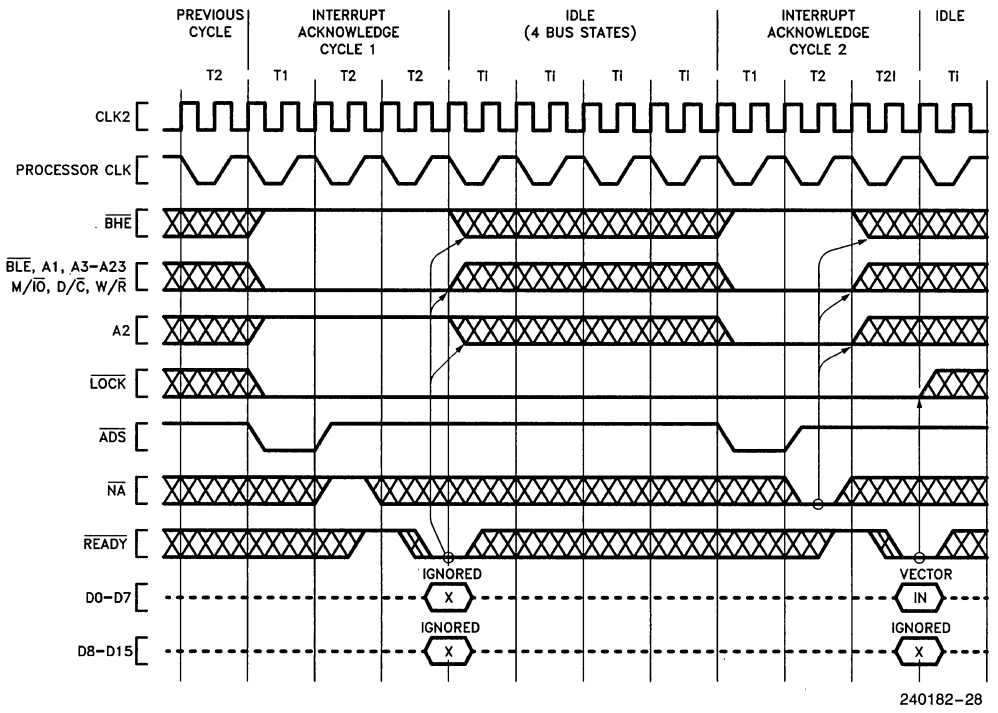
Realistically, pipelining is almost always maintained as long as $\overline{NA}$ is sampled asserted. This is so because in the absence of any other request, a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD inactive) and $\overline{NA}$ is sampled active in each of the bus cycles.

INTERRUPT ACKNOWLEDGE (INTA) CYCLES

In repsonse to an interrupt request on the INTR input when interrupts are enabled, the 80376 performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by $\overline{READY}$ sampled active.

The state of A_2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A_{23}–A_3, A_1, $\overline{BLE}$ LOW, A_2 and $\overline{BHE}$ HIGH). The byte address driven during the second interrupt acknowledge cycle is 0 (A_{23}–A_1, $\overline{BLE}$ LOW and $\overline{BHE}$ HIGH).

The $\overline{LOCK}$ output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states, T_i, are inserted by the 80376 between the two interrupt acknowledge cycles for compatibility with the interrupt specification T_{RHRL} of the 8259A Interrupt Controller and the 82370 Integrated Peripheral.



Interrupt Vector (0–255) is read on D0–D7 at end of second Interrupt Acknowledge bus cycle.
Because each Interrupt Acknowledge bus cycle is followed by idle bus states, asserting $\overline{NA}$ has no practical effect. Choose the approach which is simplest for your system hardware design.

Figure 4.13. Interrupt Acknowledge Cycles

During both interrupt acknowledge cycles, D_{15}–D_0 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the 80376 will read an external interrupt vector from D_7–D_0 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.

HALT INDICATION CYCLE

The 80376 execution unit halts as a result of executing a HLT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals and a byte address of 2. See the **Bus Cycle Definition Signals** section. The halt indication cycle must be acknowledged by $\overline{READY}$ asserted. A halted 80376 resumes execution when INTR (if interrupts are enabled), NMI or RESET is asserted.

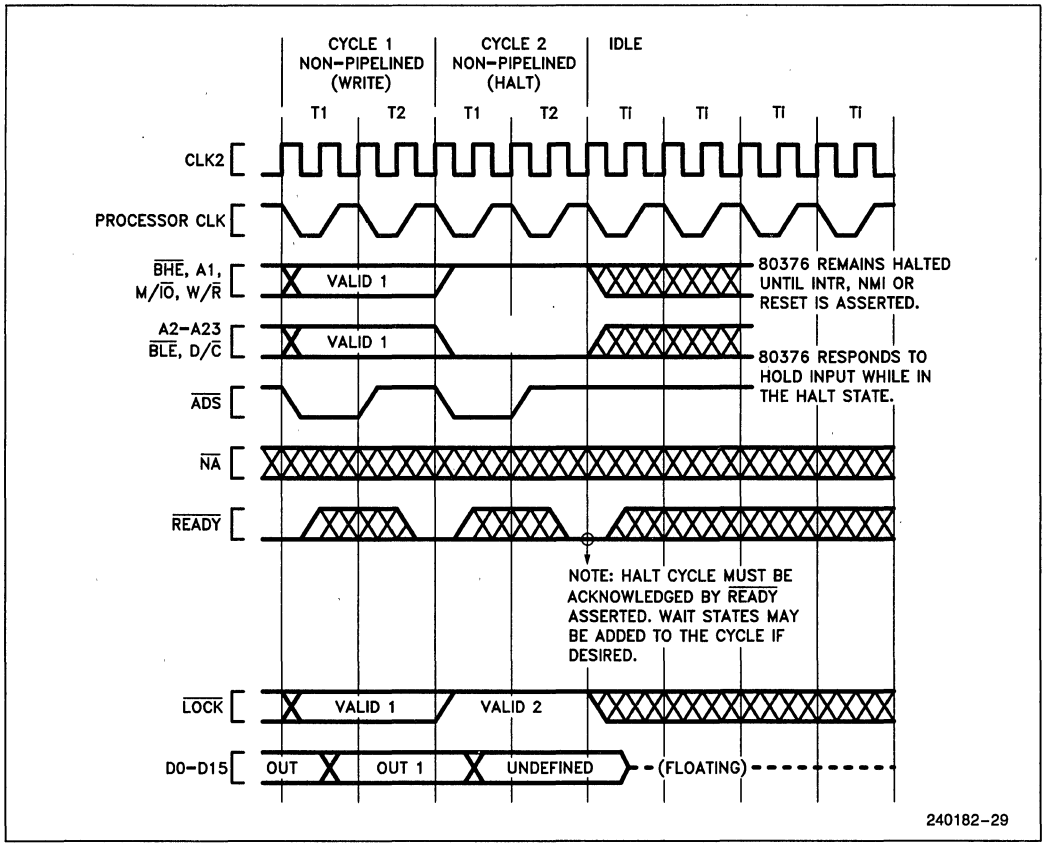


Figure 4.14. Example Halt Indication Cycle from Non-Pipelined Cycle

SHUTDOWN INDICATION CYCLE

The 80376 shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown in **Bus Cycle Definition Signals** and a byte address of 0. The shutdown indication cycle must be acknowledged by $\overline{\text{READY}}$ asserted. A shutdown 80376 resumes execution when NMI or RESET is asserted.

ENTERING AND EXITING HOLD ACKNOWLEDGE

The bus hold acknowledge state, T_h, is entered in response to the HOLD input being asserted. In the bus hold acknowledge state, the 80376 floats all outputs or bidirectional signals, except for HLDA. HLDA is asserted as long as the 80376 remains in the bus hold acknowledge state. In the bus hold acknowledge state, all inputs except HOLD and RESET are ignored.

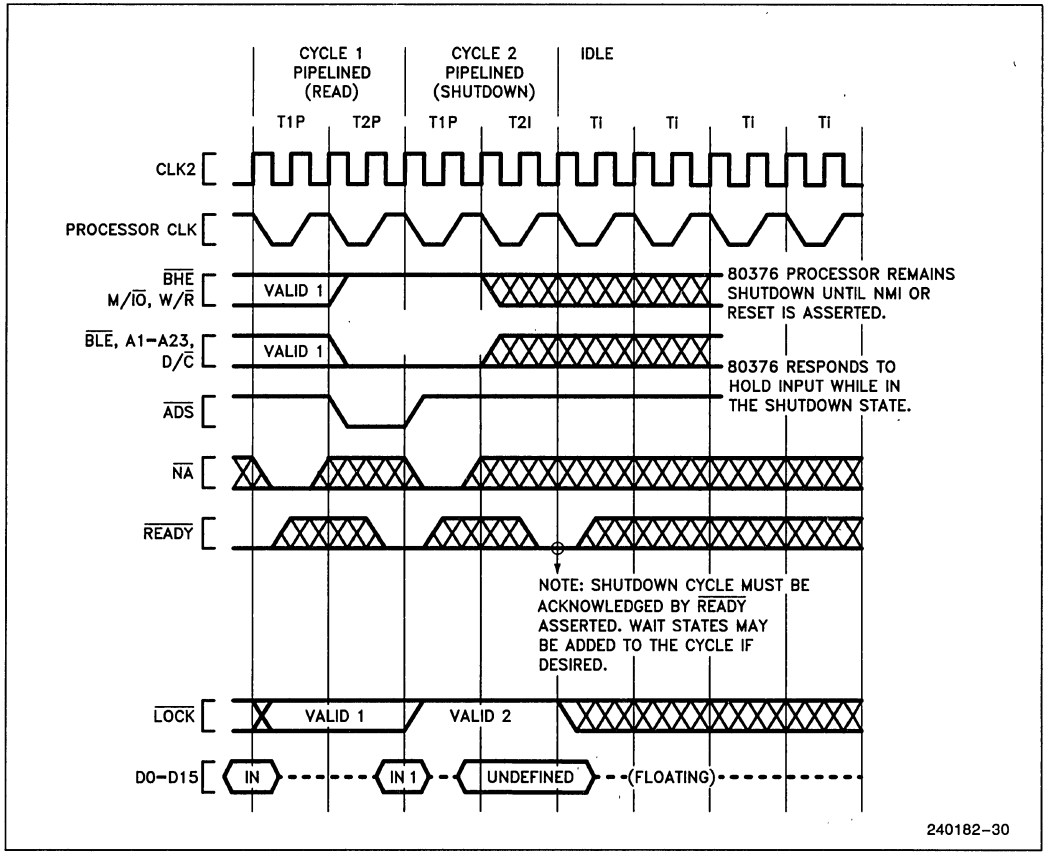
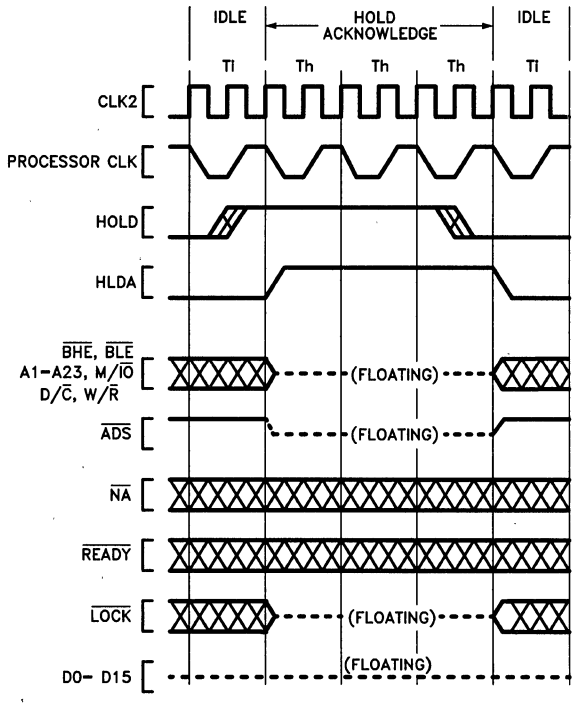


Figure 4.15. Example Shutdown Indication Cycle from Non-Pipelined Cycle

T_h may be entered from a bus idle state as in Figure 4.16 or after the acknowledgement of the current physical bus cycle if the LOCK signal is not asserted, as in Figures 4.17 and 4.18.

T_h is exited in response to the HOLD input being negated. The following state will be T_i as in Figure 4.16 if no bus request is pending. The following bus state will be T1 if a bus request is internally pending, as in Figures 4.17 and 4.18. T_h is exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in T_h, the event is remembered as a non-maskable interrupt 2 and is serviced when T_h is exited unless the 80376 is reset before T_h is exited.



NOTE:
For maximum design flexibility the 80376 has no internal pull-up resistors on its outputs. Your design may require an external pullup on $\overline{ADS}$ and other 80376 outputs to keep them negated during float periods.

Figure 4.16. Requesting Hold from Idle Bus

RESET DURING HOLD ACKNOWLEDGE

RESET being asserted takes priority over HOLD being asserted. If RESET is asserted while HOLD remains asserted, the 80376 drives its pins to defined states during reset, as in **Table 4.5, Pin State During Reset,** and performs internal reset activity as usual.

If HOLD remains asserted when RESET is inactive, the 80376 enters the hold acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the 80376 processor would otherwise perform its first bus cycle. If HOLD remains asserted when RESET is inactive, the $\overline{BUSY}$ input is still sampled as usual to determine whether a self test is being requested.

FLOAT

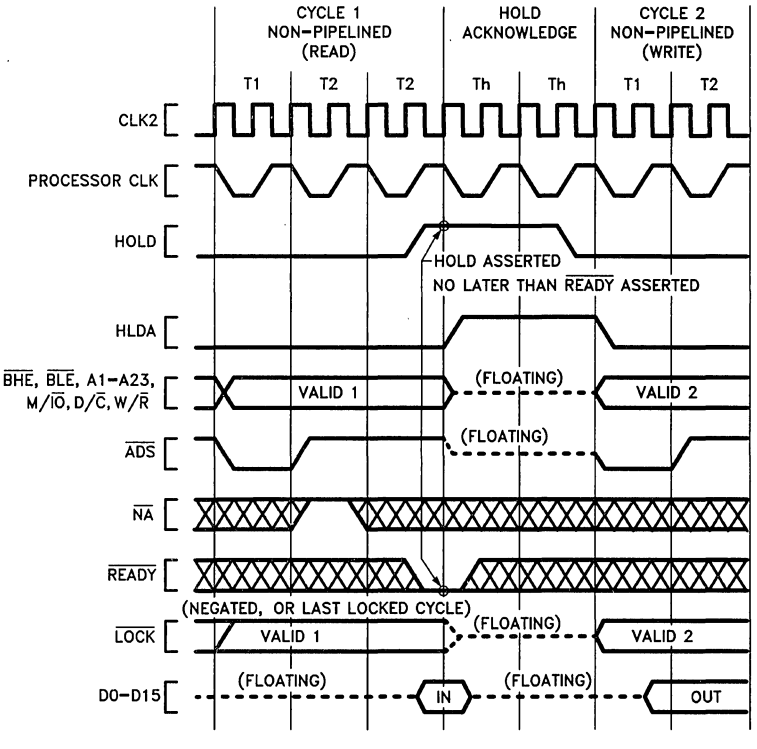
Activating the $\overline{FLT}$ input floats all 80376 bidirectional and output signals, including HLDA. Asserting $\overline{FLT}$ isolates the 80376 from the surrounding circuitry.

When an 80376 in a PQFP surface-mount package is used without a socket, it cannot be removed from the printed circuit board. The $\overline{FLT}$ input allows the 80376 to be electrically isolated to allow testing of external circuitry. This technique is known as ONCE for "ON-Circuit Emulation".

ENTERING AND EXITING FLOAT

$\overline{FLT}$ is an asynchronous, active-low input. It is recognized on the rising edge of CLK2. When recognized, it aborts the current bus cycle and floats the outputs of the 80376 (Figure 4.20). $\overline{FLT}$ must be held low for a minimum of 16 CLK2 cycles. Reset should be asserted and held asserted until after $\overline{FLT}$ is deasserted. This will ensure that the 80376 will exit float in a valid state.

Asserting the $\overline{FLT}$ input unconditionally aborts the current bus cycle and forces the 80376 into the FLOAT mode. Since activating $\overline{FLT}$ unconditionally forces the 80376 into FLOAT mode, the 80376 is not



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 4.17. Requesting Hold from Active Bus ($\overline{NA}$ Inactive)

guaranteed to enter FLOAT in a valid state. After deactivating $\overline{FLT}$, the 80376 is not guaranteed to exit FLOAT mode in a valid state. This is not a problem as the $\overline{FLT}$ pin is meant to be used only during ONCE. After exiting FLOAT, the 80376 must be reset to return it to a valid state. Reset should be asserted before $\overline{FLT}$ is deasserted. This will ensure that the 80376 will exit float in a valid state.

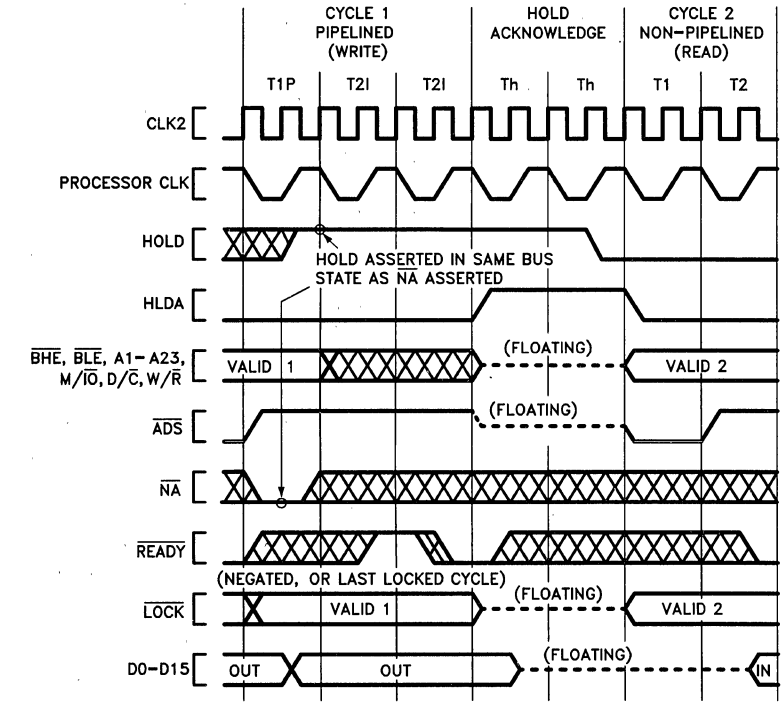
$\overline{FLT}$ has an internal pull-up resistor, and if it is not used it should be unconnected.

BUS ACTIVITY DURING AND FOLLOWING RESET

RESET is the highest priority input signal, capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage, or idle states or bus hold acknowledge states discontinued so that the reset state is established.

RESET should remain asserted for at least 15 CLK2 periods to ensure it is recognized throughout the 80376, and at least 80 CLK2 periods if a 80376 self-test is going to be requested at the falling edge. RESET asserted pulses less than 15 CLK2 periods may not be recognized. RESET pulses less than 80 CLK2 periods followed by a self-test may cause the self-test to report a failure when no true failure exists.

Provided the RESET falling edge meets setup and hold times t_{25} and t_{26}, the internal processor clock phase is defined at that time as illustrated by Figure 4.19 and Figure 6.7.

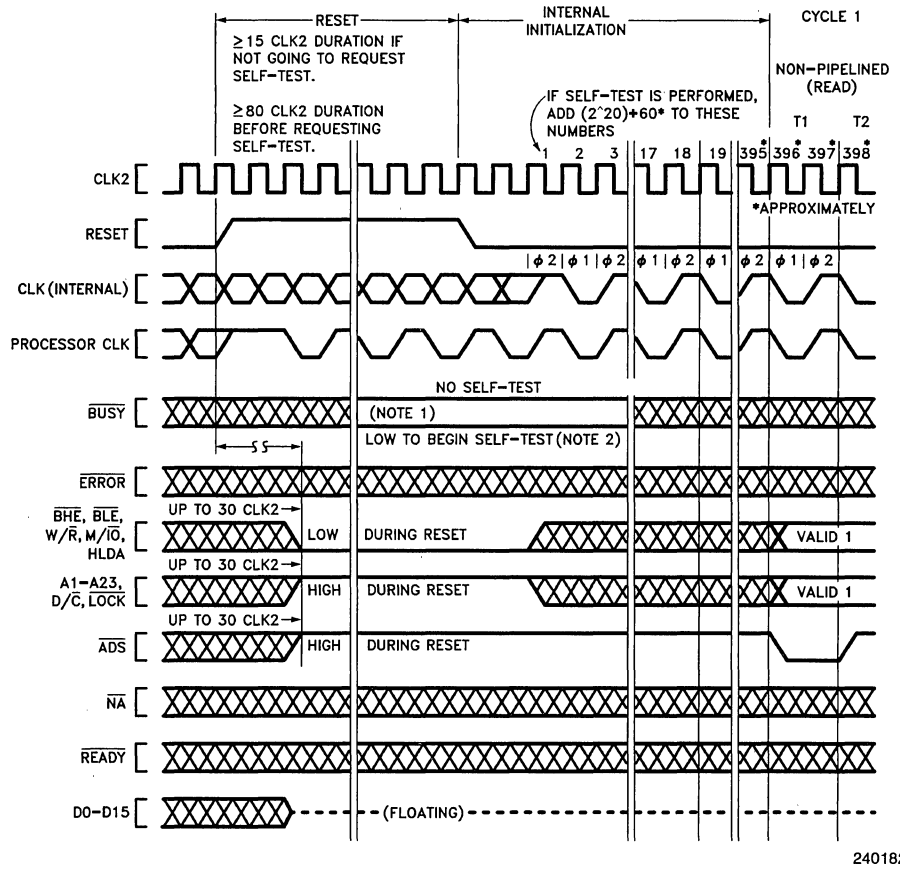


NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 4.18. Requesting Hold from Idle Bus (NA Active)

An 80376 self-test may be requested at the time RESET goes inactive by having the BUSY input at a LOW level as shown in Figure 4.19. The self-test requires (2^{20} + approximately 60) CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the 80376 attempts to proceed with the reset sequence afterwards.

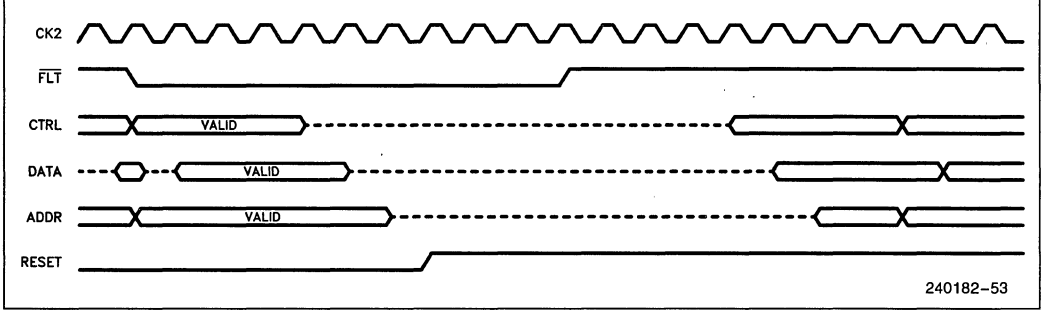
After the RESET falling edge (and after the self-test if it was requested) the 80376 performs an internal initialization sequence for approximately 350 to 450 CLK2 periods.



240182-34

NOTES:
1. BUSY should be held stable for 8 CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested, the 80376 outputs remain in their reset state as shown here.

Figure 4.19. Bus Activity from Reset until First Code Fetch



240182-53

Figure 4.20. Entering and Exiting FLOAT

4.5 Self-Test Signature

Upon completion of self-test (if self-test was requested by driving $\overline{BUSY}$ LOW at the falling edge of RESET) the EAX register will contain a signature of 00000000H indicating the 80376 passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000H, applies to all 80376 revision levels. Any non-zero signature indicates the 80376 unit is faulty.

4.6 Component and Revision Identifiers

To assist 80376 users, the 80376 after reset holds a component identifier and revision identifier in its DX register. The upper 8 bits of DX hold 33H as identification of the 80376 component. (The lower nibble, 03H, refers to the Intel386™ architecture. The upper nibble, 30H, refers to the third member of the Intel386 family). The lower 8 bits of DX hold an 8-bit unsigned binary number related to the component revision level. The revision identifier will, in general, chronologically track those component steppings which are intended to have certain improvements or distinction from previous steppings. The 80376 revision identifier will track that of the 80386 where possible.

The revision identifier is intended to assist 80376 users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision, or to follow a completely uniform numerical sequence, depending on the type or intention of revision, or manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component.

Table 4.7. Component and Revision Identifier History

80376 Stepping Name	Revision Identifier
A0	05H
B	08H

4.7 Coprocessor Interfacing

The 80376 provides an automatic interface for the Intel 80387SX numeric floating-point coprocessor. The 80387SX coprocessor uses an I/O mapped interface driven automatically by the 80376 and assisted by three dedicated signals: $\overline{BUSY}$, $\overline{ERROR}$ and PEREQ.

As the 80376 begins supporting a coprocessor instruction, it tests the $\overline{BUSY}$ and $\overline{ERROR}$ signals to determine if the coprocessor can accept its next instruction. Thus, the $\overline{BUSY}$ and $\overline{ERROR}$ inputs eliminate the need for any "preamble" bus cycles for communication between processor and coprocessor. The 80387SX can be given its command opcode immediately. The dedicated signals provide instruction synchronization, and eliminate the need of using the 80376 WAIT opcode (9BH) for 80387SX instruction synchronization (the WAIT opcode was required when the 8086 or 8088 was used with the 8087 coprocessor).

Custom coprocessors can be included in 80376 based systems by memory-mapped or I/O-mapped interfaces. Such coprocessor interfaces allow a completely custom protocol, and are not limited to a set of coprocessor protocol "primitives". Instead, memory-mapped or I/O-mapped interfaces may use all applicable 80376 instructions for high-speed coprocessor communication. The $\overline{BUSY}$ and $\overline{ERROR}$ inputs of the 80376 may also be used for the custom coprocessor interface, if such hardware assist is desired. These signals can be tested by the 80376 WAIT opcode (9BH). The WAIT instruction will wait until the $\overline{BUSY}$ input is inactive (interruptable by an NMI or enabled INTR input), but generates an exception 16 fault if the $\overline{ERROR}$ pin is active when the $\overline{BUSY}$ goes (or is) inactive. If the custom coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the segmentation mechanism of the 80376. If the custom interface is I/O-mapped, protection of the interface can be provided with the 80376 IOPL (I/O Privilege Level) mechanism.

The 80387SX numeric coprocessor interface is I/O mapped as shown in Table 4.8. Note that the 80387SX coprocessor interface addresses are beyond the 0H-0FFFFH range for programmed I/O. When the 80376 supports the 80387SX coprocessor, the 80376 automatically generates bus cycles to the coprocessor interface addresses.

Table 4.8 Numeric Coprocessor Port Addresses

Address in 80376 I/O Space	80387SX Coprocessor Register
8000F8H	Opcode Register
8000FCH	Operand Register
8000FEH	Operand Register

SOFTWARE TESTING FOR COPROCESSOR PRESENCE

When software is used to test coprocessor (80387SX) presence, it should use only the following coprocessor opcodes: FNINIT, FNSTCW and FNSTSW. To use other coprocessor opcodes when a coprocessor is known to be not present, first set EM = 1 in the 80376 CR0 register.

5.0 PACKAGE THERMAL SPECIFICATIONS

The Intel 80376 embedded processor is specified for operation when case temperature is within the range of 0°C–115°C for both the ceramic 88-pin PGA package and the plastic 100-pin PQFP package. The case temperature may be measured in any environment, to determine whether the 80376 is within specified operating range. The case temperature should be measured at the center of the top surface.

The ambient temperature is guaranteed as long as T_C is not violated. The ambient temperature can be calculated from the θ_{jc} and θ_{ja} from the following equations:

$$T_J = T_C + P^*\theta_{jc}$$

$$T_A = T_j - P^*\theta_{ja}$$

$$T_C = T_a + P^*[\theta_{ja} - \theta_{jc}]$$

Values for θ_{ja} and θ_{jc} are given in Table 5.1 for the 100-lead fine pitch. θ_{ja} is given at various airflows. Table 5.2 shows the maximum T_a allowable (without exceeding T_C) at various airflows. Note that T_a can be improved further by attaching "fins" or a "heat sink" to the package. P is calculated using the maximum ***cold*** I_{CC} of 305 mA and the maximum V_{CC} of 5.5V for both packages.

Table 5.1. 80376 Package Thermal Characteristics Thermal Resistances (°C/Watt) θ_{jc} and θ_{ja}

Package	θ_{jc}	θ_{ja} Versus Airflow-ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100-Lead Fine Pitch	7.5	34.5	29.5	25.5	22.5	21.5	21.0
88-Pin PGA	2.5	29.0	22.5	17.0	14.5	12.5	12.0

Table 5.2. 80376 Maximum Allowable Ambient Temperature at Various Airflows

Package	θ_{jc}	T_A(°C) vs Airflow-ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100-Lead Fine Pitch	7.5	70	78	85	90	92	93
88-Pin PGA	2.5	70	81	90	95	98	99

6.0 ELECTRICAL SPECIFICATIONS

The following sections describe recommended electrical connections for the 80376, and its electrical specifications.

2

6.1 Power and Grounding

The 80376 is implemented in CHMOS IV technology and has modest power requirements. However, its high clock frequency and 47 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 14 V_{CC} and 18 V_{SS} pins separately feed functional units of the 80376.

Power and ground connections must be made to all external V_{CC} and GND pins of the 80376. On the circuit board, all V_{CC} pins should be connected on a V_{CC} plane and all V_{SS} pins should be connected on a GND plane.

POWER DECOUPLING RECOMMENDATIONS

Liberal decoupling capacitors should be placed near the 80376. The 80376 driving its 24-bit address bus and 16-bit data bus at high frequencies can cause transient power surges, particularly when driving large capacitive loads. Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the 80376 and decoupling capacitors as much as possible.

RESISTOR RECOMMENDATIONS

The $\overline{\text{ERROR}}$, $\overline{\text{FLT}}$ and $\overline{\text{BUSY}}$ inputs have internal pull-up resistors of approximately 20 KΩ and the PEREQ input has an internal pull-down resistor of approximately 20 KΩ built into the 80376 to keep these signals inactive when the 80387SX is not present in the system (or temporarily removed from its socket).

In typical designs, the external pull-up resistors shown in Table 6.1 are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pull-up resistors in other ways.

Table 6.1. Recommended Resistor Pull-Ups to V_{CC}

Pin	Signal	Pull-Up Value	Purpose
16	$\overline{ADS}$	20 KΩ ± 10%	Lightly Pull $\overline{ADS}$ Inactive during 80376 Hold Acknowledge States
26	$\overline{LOCK}$	20 KΩ ± 10%	Lightly Pull $\overline{LOCK}$ Inactive during 80376 Hold Acknowledge States

OTHER CONNECTION RECOMMENDATIONS

For reliable operation, always connect unused inputs to an appropriate signal level. N/C pins should always remain **unconnected. Connection of N/C pins to V_{CC} or V_{SS} will result in incompatibility with future steppings of the 80376.**

Particularly when not using interrupts or bus hold (as when first prototyping), prevent any chance of spurious activity by connecting these associated inputs to GND:

—INTR
—NMI
—HOLD

If not using address pipelining connect the $\overline{NA}$ pin to a pull-up resistor in the range of 20 KΩ to V_{CC}.

6.2 Absolute Maximum Ratings

Table 6.2. Maximum Ratings

Parameter	Maximum Rating
Storage Temperature	−65°C to +150°C
Case Temperature under Bias	−65°C to +120°C
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V
Voltage on Other Pins	−0.5V to (V_{CC} + 0.5)V

Table 6.2 gives a stress ratings only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in **Section 6.3, D.C. Specifications,** and **Section 6.4, A.C. Specifications.**

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the 80376 contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

6.3 D.C. Specifications

ADVANCE INFORMATION SUBJECT TO CHANGE

Table 6.3: 80376 D.C. Characteristics

Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C for 88-pin PGA or 100-pin PQFP

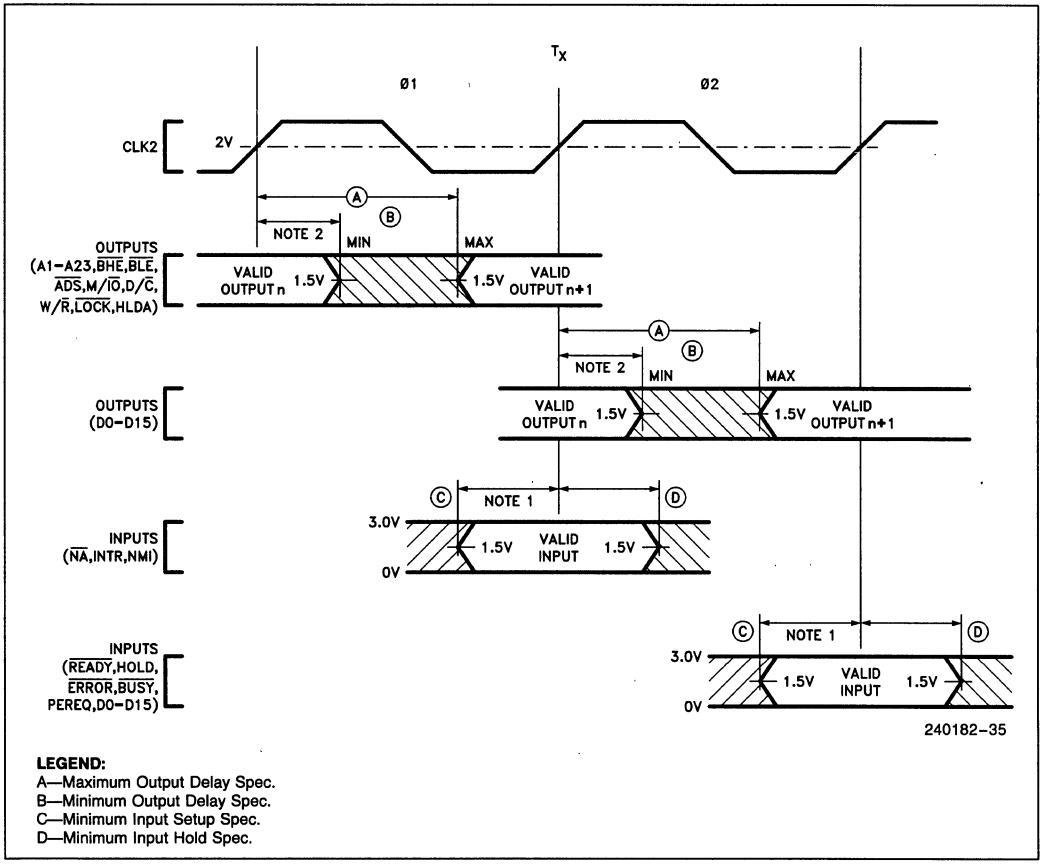
Symbol	Parameter	Min	Max	Unit
V_{IL}	Input LOW Voltage	−0.3	+0.8	V(1)
V_{IH}	Input HIGH Voltage	2.0	V_{CC} +0.3	V(1)
V_{ILC}	CLK2 Input LOW Voltage	−0.3	+0.8	V(1)
V_{IHC}	CLK2 Input HIGH Voltage	V_{CC} − 0.8	V_{CC} + 0.3	V(1)
V_{OL}	Output LOW Voltage			
I_{OL} = 4 mA:	A_{23}–A_1, D_{15}–D_0		0.45	V(1)
I_{OL} = 5 mA:	$\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA		0.45	V(1)
V_{OH}	Output High Voltage			
I_{OH} = −1 mA:	A_{23}–A_1, D_{15}–D_0	2.4		V(1)
I_{OH} = −0.2 mA:	A_{23}–A_1, D_{15}–D_0	V_{CC} − 0.5		V(1)
I_{OH} = −0.9 mA:	$\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA	2.4		V(1)
I_{OH} = −0.18 mA:	$\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$ $\overline{ADS}$, HLDA	V_{CC} − 0.5		V(1)
I_{LI}	Input Leakage Current (For All Pins except PEREQ, $\overline{BUSY}$, $\overline{FLT}$ and $\overline{ERROR}$)		±15	μA, 0V ≤ V_{IN} ≤ V_{CC}(1)
I_{IH}	Input Leakage Current (PEREQ Pin)		200	μA, V_{IH} = 2.4V(1, 2)
I_{IL}	Input Leakage Current ($\overline{BUSY}$ and $\overline{ERROR}$ Pins)		−400	μA, V_{IL} = 0.45V(3)
I_{LO}	Output Leakage Current		±15	μA, 0.45V ≤ V_{OUT} ≤ V_{CC}(1)
I_{CC}	Supply Current CLK2 = 32 MHz CLK2 = 40 MHz		 275 305	 mA, I_{CC} typ = 175 mA(4) mA, I_{CC} typ = 200 mA(4)
C_{IN}	Input Capacitance		10	pF, F_C = 1 MHz(5)
C_{OUT}	Output or I/O Capacitance		12	pF, F_C = 1 MHz(5)
C_{CLK}	CLK2 Capacitance		20	pF, F_C = 1 MHz(5)

NOTES:
1. Tested at the minimum operating frequency of the device.
2. PEREQ input has an internal pull-down resistor.
3. $\overline{BUSY}$, $\overline{FLT}$ and $\overline{ERROR}$ inputs each have an internal pull-up resistor.
4. I_{CC} max measurement at worse case load, V_{CC} and temperature (0°C).
5. Not 100% tested.

The A.C. specifications given in Table 6.4 consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the CLK2 rising edge crossing the 2.0V level.

A.C. specification measurement is defined by Figure 6.1. Inputs must be driven to the voltage levels indicated by Figure 6.1 when A.C. specifications are measured. 80376 output delays are specified with minimum and maximum limits measured as shown. The minimum 80376 delay times are hold times provided to external circuitry. 80376 input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct 80376 processor operation.

Outputs $\overline{NA}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{BHE}$, $\overline{BLE}$, A_{23}–A_1 and HLDA only change at the beginning of phase one. D_{15}–D_0 (write cycles) only change at the beginning of phase two. The $\overline{READY}$, HOLD, $\overline{BUSY}$, $\overline{ERROR}$, PEREQ and D_{15}–D_0 (read cycles) inputs are sampled at the beginning of phase one. The $\overline{NA}$, INTR and NMI inputs are sampled at the beginning of phase two.



LEGEND:
A—Maximum Output Delay Spec.
B—Minimum Output Delay Spec.
C—Minimum Input Setup Spec.
D—Minimum Input Hold Spec.

Figure 6.1. Drive Levels and Measurement Points for A.C. Specifications

6.4 A.C. Specifications

Table 6.4. 80376 A.C. Characteristics at 16 MHz

Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C for 88-pin PGA or 100-pin PQFP

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Operating Frequency	4	16	MHz		Half CLK2 Freq
t_1	CLK2 Period	31	125	ns	6.3	
t_{2a}	CLK2 HIGH Time	9		ns	6.3	At 2(3)
t_{2b}	CLK2 HIGH Time	5		ns	6.3	At (V_{CC} − 0.8)V(3)
t_{3a}	CLK2 LOW Time	9		ns	6.3	At 2V(3)
t_{3b}	CLK2 LOW Time	7		ns	6.3	At 0.8V(3)
t_4	CLK2 Fall Time		8	ns	6.3	(V_{CC}−0.8)V to 0.8V(3)
t_5	CLK2 Rise Time		8	ns	6.3	0.8V to (V_{CC}−0.8)(3)
t_6	A_{23}–A_1 Valid Delay	4	36	ns	6.5	C_L = 120 pF(4)
t_7	A_{23}–A_1 Float Delay	4	40	ns	6.6	(1)
t_8	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Valid Delay	4	36	ns	6.5	C_L = 75 pF(4)
t_9	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Float Delay	4	40	ns	6.6	(1)
t_{10}	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Valid Delay	6	33	ns	6.5	C_L = 75 pF(4)
t_{11}	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Float Delay	6	35	ns	6.6	(1)
t_{12}	D_{15}–D_0 Write Data Valid Delay	4	40	ns	6.5	C_L = 120 pF(4)
t_{13}	D_{15}–D_0 Write Data Float Delay	4	35	ns	6.6	(1)
t_{14}	HLDA Valid Delay	4	33	ns	6.6	C_L = 75 pF(4)
t_{15}	$\overline{NA}$ Setup Time	5		ns	6.4	
t_{16}	$\overline{NA}$ Hold Time	21		ns	6.6	
t_{19}	$\overline{READY}$ Setup Time	19		ns	6.4	
t_{20}	$\overline{READY}$ Hold Time	4		ns	6.4	
t_{21}	Setup Time D_{15}–D_0 Read Data	9		ns	6.4	
t_{22}	Hold Time D_{15}–D_0 Read Data	6		ns	6.4	
t_{23}	HOLD Setup Time	26		ns	6.4	
t_{24}	HOLD Hold Time	5		ns	6.4	
t_{25}	RESET Setup Time	13		ns	6.7	
t_{26}	RESET Hold Time	4		ns	6.7	

Table 6.4. 80376 A.C. Characteristics at 16 MHz (Continued)
Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C for 88-pin PGA or 100-pin PQFP

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{27}	NMI, INTR Setup Time	16		ns	6.4	(2)
t_{28}	NMI, INTR Hold Time	16		ns	6.4	(2)
t_{29}	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Setup Time	16		ns	6.4	(2)
t_{30}	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Hold Time	5		ns	6.4	(2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with C_L set to 50 pF and derated to support the indicated distributed capacitive load. See Figures 6.8 through 6.10 for capacitive derating curves.
5. The 80376 does not have t_{17} or t_{18} timing specifications.

Table 6.5. 80376 A.C. Characteristics at 20 MHz
Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C for 88-pin PGA or 100-pin PQFP

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Operating Frequency	4	20	MHz		Half CLK2 Frequency
t_1	CLK2 Period	25	125	ns	6.3	
t_{2a}	CLK2 HIGH Time	8		ns	6.3	At 2V(3)
t_{2b}	CLK2 HIGH Time	5		ns	6.3	At (V_{CC} − 0.8)V(3)
t_{3a}	CLK2 LOW Time	8		ns	6.3	At 2V(3)
t_{3b}	CLK2 LOW Time	6		ns	6.3	At 0.8V(3)
t_4	CLK2 Fall Time		8	ns	6.3	(V_{CC}−0.8V) to 0.8V(3)
t_5	CLK2 Rise Time		8	ns	6.3	0.8V to (V_{CC}−0.8)(3)
t_6	A_{23}–A_1 Valid Delay	4	30	ns	6.5	C_L = 120 pF(4)
t_7	A_{23}–A_1 Float Delay	4		ns	6.6	(1)
t_8	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Valid Delay	4	30	ns	6.5	C_L = 75 pF(4)
t_9	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Float Delay	4	32	ns	6.6	(1)
t_{10a}	M/$\overline{IO}$, D/$\overline{C}$ Valid Delay	6	28	ns	6.5	C_L = 75 pF(4)
t_{10b}	W/$\overline{R}$, $\overline{ADS}$ Valid Delay	6	26	ns	6.5	C_L = 75 pF(4)
t_{11}	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Float Delay	6	30	ns	6.6	(1)
t_{12}	D_{15}–D_0 Write Data Valid Delay	4	38	ns	6.5	C_L = 120 pF
t_{13}	D_{15}–D_0 Write Data Float Delay	4	27	ns	6.6	(1)

Table 6.5. 80376 A.C. Characteristics at 20 MHz (Continued)
Functional Operating Range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 115°C for 88-pin PGA or 100-pin PQFP

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{14}	HLDA Valid Delay	4	28	ns	6.5	C_L = 75 pF(4)
t_{15}	$\overline{NA}$ Setup Time	5		ns	6.4	
t_{16}	$\overline{NA}$ Hold Time	12		ns	6.4	
t_{19}	$\overline{READY}$ Setup Time	12		ns	6.4	
t_{20}	$\overline{READY}$ Hold Time	4		ns	6.4	
t_{21}	D_{15}–D_0 Read Data Setup Time	9		ns	6.4	
t_{22}	D_{15}–D_0 Read Data Hold Time	6		ns	6.4	
t_{23}	HOLD Setup Time	17		ns	6.4	
t_{24}	HOLD Hold Time	5		ns	6.4	
t_{25}	RESET Setup Time	12		ns	6.7	
t_{26}	RESET Hold Time	4		ns	6.7	
t_{27}	NMI, INTR Setup Time	16		ns	6.4	(2)
t_{28}	NMI, INTR Hold Time	16		ns	6.4	(2)
t_{29}	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Setup Time	14		ns	6.4	(2)
t_{30}	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Hold Time	5		ns	6.4	(2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with C_L set to 50 pF and derated to support the indicated distributed capacitive load. See Figures 6.8 through 6.10 for capacitive derating curves.
5. The 80376 does not have t_{17} or t_{18} timing specifications.

A.C. TEST LOADS

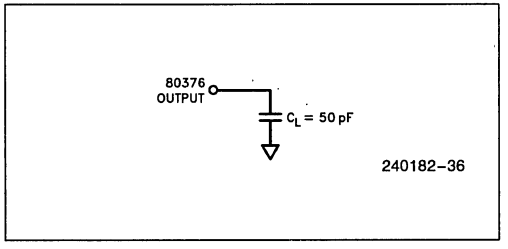


Figure 6.2. A.C. Test Loads

A.C. TIMING WAVEFORMS

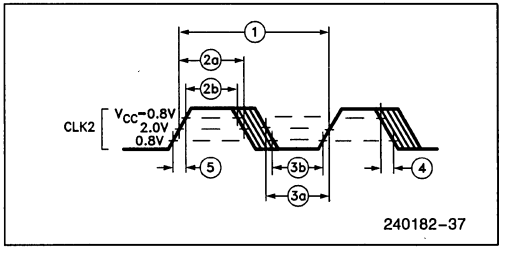


Figure 6.3. CLK2 Waveform

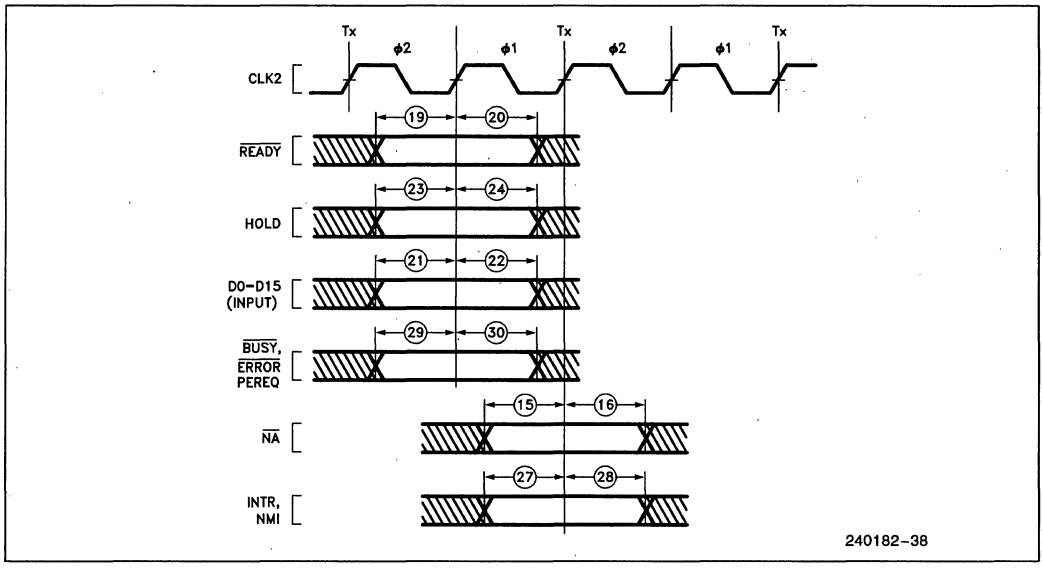


Figure 6.4. A.C. Timing Waveforms—Input Setup and Hold Timing

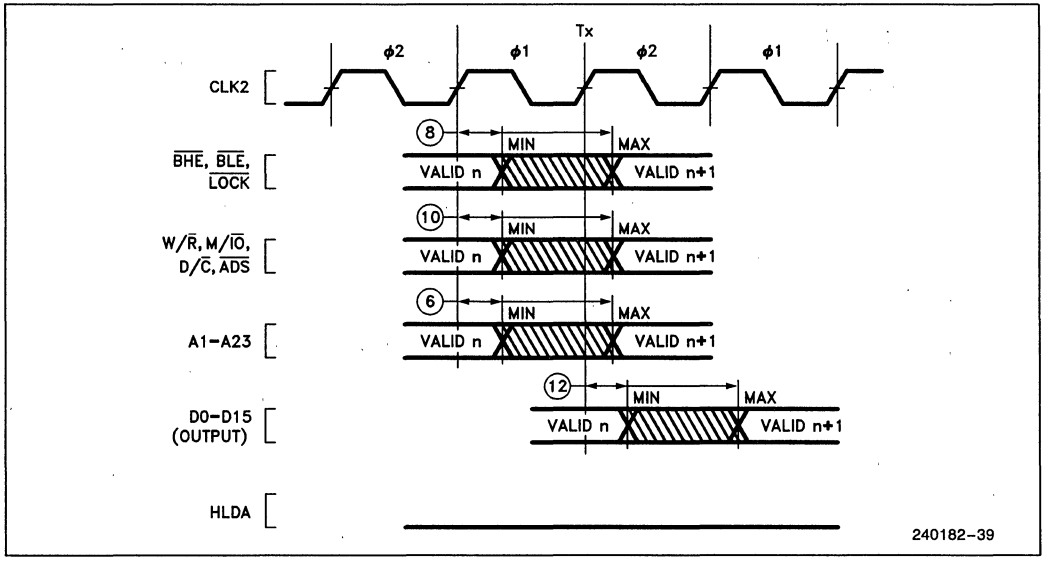


Figure 6.5. A.C. Timing Waveforms—Output Valid Delay Timing

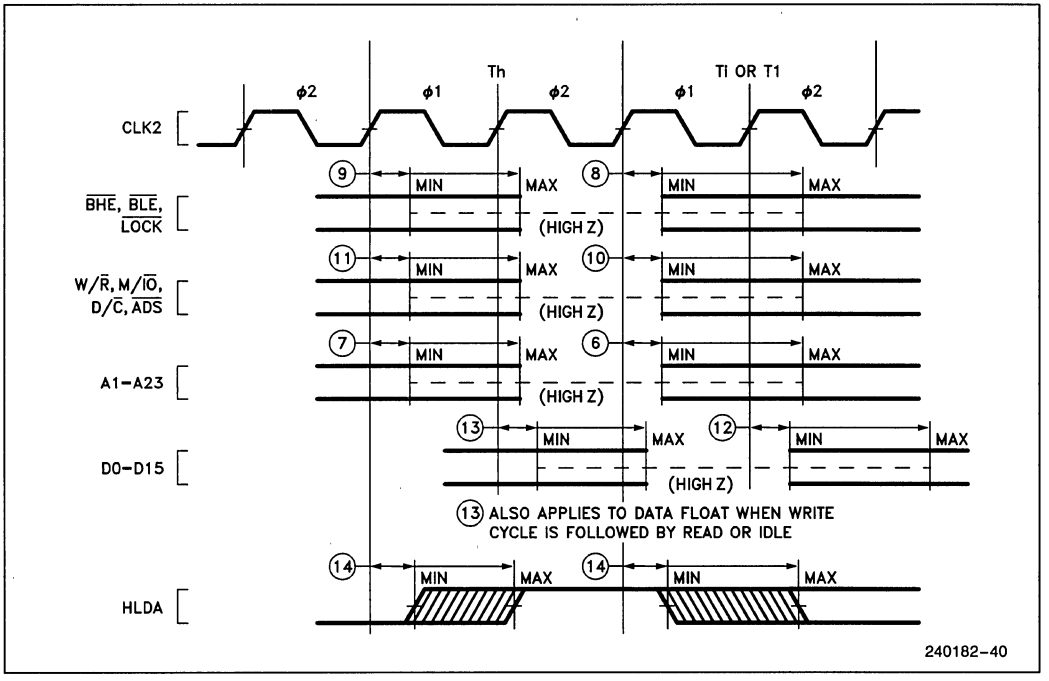
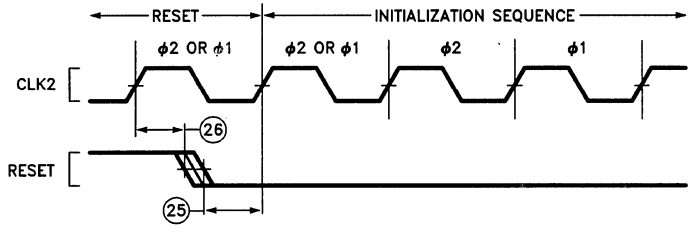


Figure 6.6. A.C. Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET high-to-low transition (provided t_{25} and t_{26} are met) is Φ2.

Figure 6.7. A.C. Timing Waveforms—RESET Setup and Hold Timing, and Internal Phase

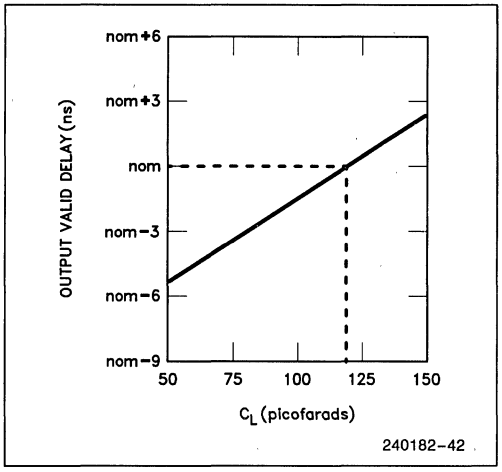


Figure 6.8. Typical Output Valid Delay versus Load Capacitance at Maximum Operating Temperature (C_L = 120 pF)

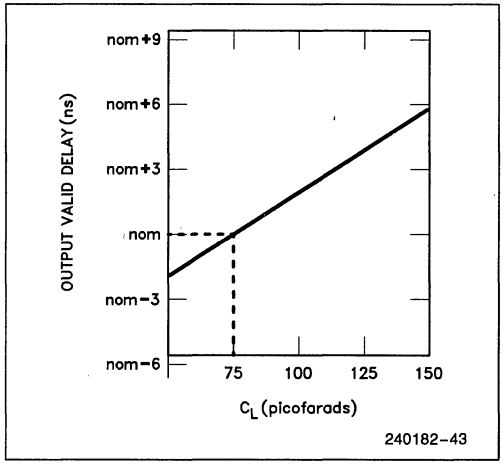


Figure 6.9. Typical Output Valid Delay versus Load Capacitance at Maximum Operating Temperature (C_L = 75 pF)

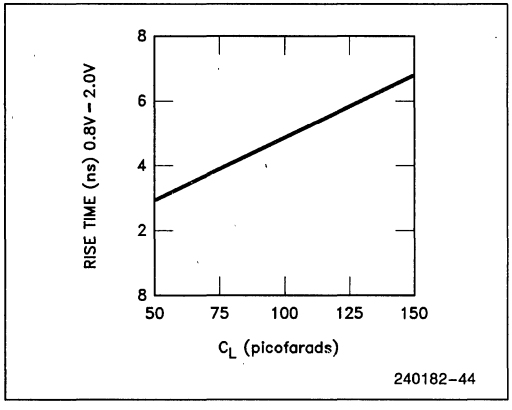


Figure 6.10. Typical Output Rise Time versus Load Capacitance at Maximum Operating Temperature

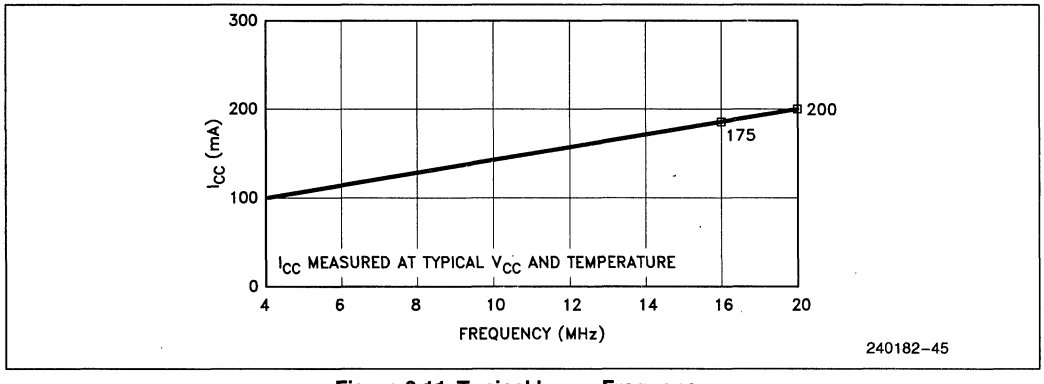


Figure 6.11. Typical I_{CC} vs Frequency

6.5 Designing for the ICE™-376 Emulator

The 376 embedded processor in-circuit emulator product is the ICE-376 emulator. Use of the emulator requires the target system to provide a socket that is compatible with the ICE-376 emulator. The 80376 offers two different probes for emulating user systems: an 88-pin PGA probe and a 100-pin fine pitch flat-pack probe. The 100-pin fine pitch flat-pack probe requires a socket, called the 100-pin PQFP, which is available from 3-M Textool (part number 2-0100-07243-000). The ICE-376 emulator probe attaches to the target system via an adapter which replaces the 80376 component in the target system. Because of the high operating frequency of 80376 systems and of the ICE-376 emulator, there is no buffering between the 80376 emulation processor in the ICE-376 emulator probe and the target system. A direct result of the non-buffered interconnect is that the ICE-376 emulator shares the address and data bus with the user's system, and the RESET signal is intercepted by the ICE emulator hardware. In order for the ICE-376 emulator to be functional in the user's system without the Optional Isolation Board (OIB) the designer must be aware of the following conditions:

1. The bus controller must only enable data transceivers onto the data bus during valid read cycles of the 80376, other local devices or other bus masters.
2. Before another bus master drives the local processor address bus, the other master must gain control of the address bus by asserting HOLD and receiving the HLDA response.
3. The emulation processor receives the RESET signal 2 or 4 CLK2 cycles later than an 80376 would, and responds to RESET later. Correct phase of the response is guaranteed.

In addition to the above considerations, the ICE-376 emulator processor module has several electrical and mechanical characteristics that should be taken into consideration when designing the 80376 system.

Capacitive Loading: ICE-376 adds up to 27 pF to each 80376 signal.

Drive Requirements: ICE-376 adds one FAST TTL load on the CLK2, control, address, and data lines. These loads are within the processor module and are driven by the 80376 emulation processor, which has standard drive and loading capability listed in Tables 6.3 and 6.4.

Power Requirements: For noise immunity and CMOS latch-up protection the ICE-376 emulator processor module is powered by the user system. The circuitry on the processor module draws up to 1.4A including the maximum 80376 I_{CC} from the user 80376 socket.

80376 Location and Orientation: The ICE-376 emulator processor module may require lateral clearance. Figure 6.12 shows the clearance requirements of the iMP adapter and Figure 6.13 shows the clearance requirements of the 88-pin PGA adapter. The

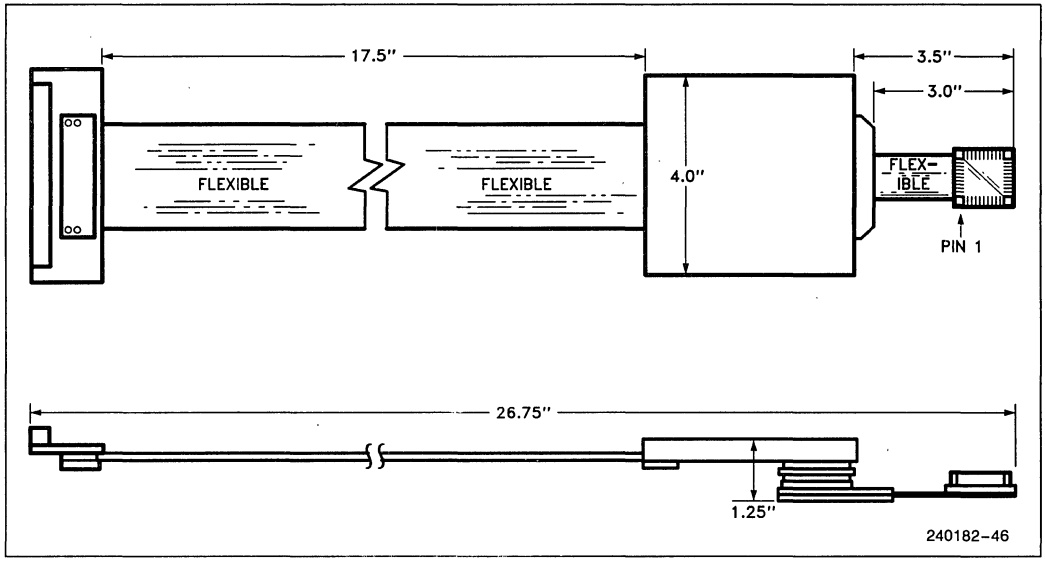


Figure 6.12. Preliminary ICE™-376 Emulator User Cable with PQFP Adapter

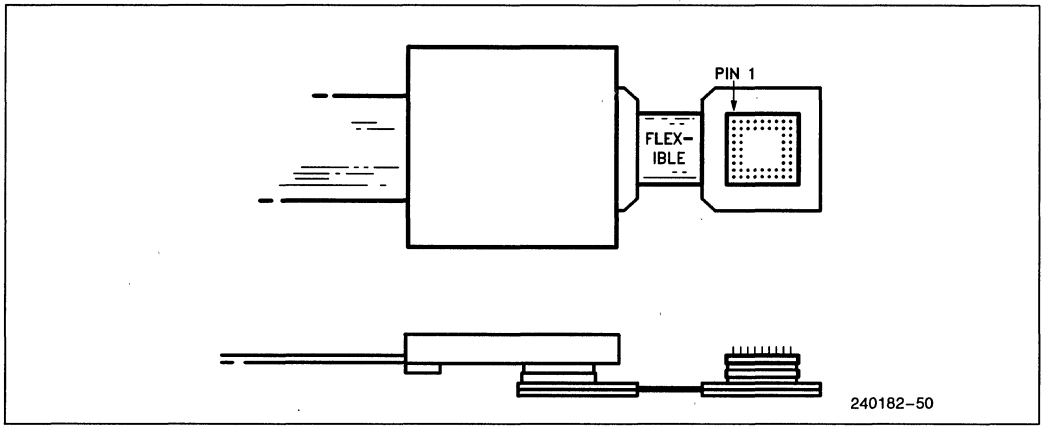


Figure 6.13. ICE™-376 Emulator User Cable with 88-Pin PGA Adapter

optional isolation board (OIB), which provides extra electrical buffering and has the same lateral clearance requirements as Figures 6.12 and 6.13, adds an additional 0.5 inches to the vertical clearance requirement. This is illustrated in Figure 6.14.

Optional Isolation Board (OIB) and the CLK2 speed reduction: Due to the unbuffered probe design, the ICE-376 emulator is susceptible to errors on the user's bus. The OIB allows the ICE-376 emulator to function in user systems with faults (shorted signals, etc.). After electrical verification the OIB may be removed. When the OIB is installed, the user system must have a maximum CLK2 frequency of 20 MHz.

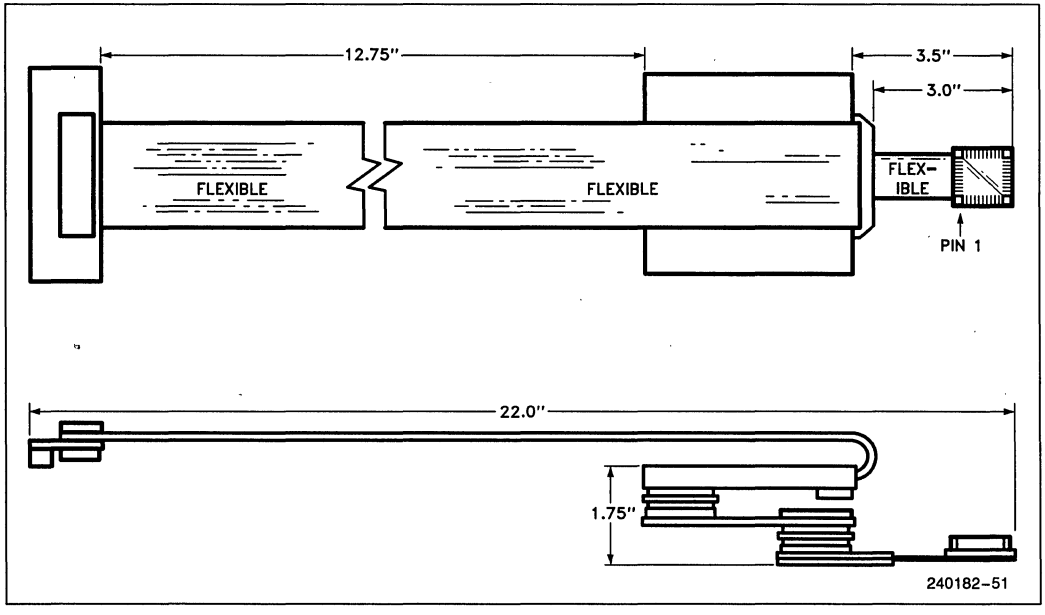


Figure 6.14. ICE™-376 Emulator User Cable with OIB and PQFP Adapter

7.0 DIFFERENCES BETWEEN THE 80376 AND THE 80386

The following are the major differences between the 80376 and the 80386.

1. The 80376 generates byte selects on $\overline{BHE}$ and $\overline{BLE}$ (like the 8086 and 80286 microprocessors) to distinguish the upper and lower bytes on its 16-bit data bus. The 80386 uses four-byte selects, $\overline{BE0}$–$\overline{BE3}$, to distinguish between the different bytes on its 32-bit bus.
2. The 80376 has no bus sizing option. The 80386 can select between either a 32-bit bus or a 16-bit bus by use of the $\overline{BS16}$ input. The 80376 has a 16-bit bus size.
3. The $\overline{NA}$ pin operation in the 80376 is identical to that of the $\overline{NA}$ pin on the 80386 with one exception: the $\overline{NA}$ pin of the 80386 cannot be activated on 16-bit bus cycles (where $\overline{BS16}$ is LOW in the 80386 case), whereas $\overline{NA}$ can be activated on any 80376 bus cycle.
4. The contents of all 80376 registers at reset are identical to the contents of the 80386 registers at reset, except the DX register. The DX register contains a component-stepping identifier at reset, i.e.

   in 80386, after reset DH = 03H indicates 80386
   DL = revision number;
   in 80376, after reset DH = 33H indicates 80376
   DL = revision number.

5. The 80386 uses A_{31} and M/$\overline{IO}$ as a select for numerics coprocessor. The 80376 uses the A_{23} and M/$\overline{IO}$ to select its numerics coprocessor.
6. The 80386 prefetch unit fetches code in four-byte units. The 80376 prefetch unit reads two bytes as one unit (like the 80286 microprocessor). In $\overline{BS16}$ mode, the 80386 takes two consecutive bus cycles to complete a prefetch request. If there is a data read or write request after the prefetch starts, the 80386 will fetch all four bytes before addressing the new request.
7. The 80376 has no paging mechanism.
8. The 80376 starts executing code in what corresponds to the 80386 protected mode. The 80386 starts execution in real mode, which is then used to enter protected mode.
9. The 80386 has a virtual-86 mode that allows the execution of a real mode 8086 program as a task in protected mode. The 80376 has no virtual-86 mode.
10. The 80386 maps a 48-bit logical address into a 32-bit physical address by segmentation and paging. The 80376 maps its 48-bit logical address into a 24-bit physical address by segmentation only.
11. The 80376 uses the 80387SX numerics coprocessor for floating point operations, while the 80386 uses the 80387 coprocessor.
12. The 80386 can execute from 16-bit code segments. The 80376 can **only** execute from 32-bit code Segments.
13. The 80376 has an input called $\overline{FLT}$ which three-states all bidirectional and output pins, including HLDA, when asserted. It is used with ON Circuit Emulation (ONCE).

8.0 INSTRUCTION SET

This section describes the 376 embedded processor instruction set. Table 8.1 lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within 80376 instructions.

8.1 80376 Instruction Encoding and Clock Count Summary

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 8.1 below, by the processor clock period (e.g. 50 ns for an 80376 operating at 20 MHz). The actual clock count of an 80376 program will average 10% more

than the calculated clock count due to instruction sequences which execute faster than they can be fetched from memory.

Instruction Clock Count Assumptions:

1. The instruction has been prefetched, decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor acess to the bus.
4. No exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts showns. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.
6. Memory reference instruction accesses byte or aligned 16-bit operands.

Instruction Clock Count Notation

— If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.

—n = number of times repeated.

—m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and all other bytes of the instruction and prefix(es) each count as one component.

Misaligned or 32-Bit Operand Accesses:

— If instructions accesses a misaligned 16-bit operand or 32-bit operand on even address add:

2* clocks for read or write.

4** clocks for read and write.

— If instructions accesses a 32-bit operand on odd address add:

4* clocks for read or write.

8** clocks for read and write.

Wait States:

Wait states add 1 clock per wait state to instruction execution for each data access.

Table 8.1. 80376 Instruction Set Clock Count Summary

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
GENERAL DATA TRANSFER				
MOV = Move:				
Register to Register/Memory	1000100w | mod reg r/m	2/2*	0/1*	a
Register/Memory to Register	1000101w | mod reg r/m	2/4*	0/1*	a
Immediate to Register/Memory	1100011w | mod000 r/m | immediate data	2/2*	0/1*	a
Immediate to Register (Short Form)	1011 w reg | immediate data	2	2	
Memory to Accumulator (Short Form)	1010000w | full displacement	4*	1*	a
Accumulator to Memory (Short Form)	1010001w | full displacement	2*	1*	a
Register/Memory to Segment Register	10001110 | mod sreg3 r/m	22/23*	0/6*	a,b,c
Segment Register to Register/Memory	10001100 | mod sreg3 r/m	2/2*	0/1*	a
MOVSX = Move with Sign Extension				
Register from Register/Memory	00001111 | 1011111w | mod reg r/m	3/6*	0/1*	a
MOVZX = Move with Zero Extension				
Register from Register/Memory	00001111 | 1011011w | mod reg r/m	3/6*	0/1*	a
PUSH = Push:				
Register/Memory	11111111 | mod110 r/m	7/9*	2/4*	a
Register (Short Form)	01010 reg	4	2	a
Segment Register (ES, CS, SS or DS)	000sreg2110	4	2	a
Segment Register (FS or GS)	00001111 | 10sreg3000	4	2	a
Immediate	011010s0 | immediate data	4	2	a
PUSHA = Push All	01100000	34	16	a
POP = Pop				
Register/Memory	10001111 | mod000 r/m	7/9*	2/4*	a
Register (Short Form)	01011 reg	6	2	a
Segment Register (ES, SS or DS)	000sreg2111	25	6	a, b, c
Segment Register (FS or GS)	00001111 | 10sreg3001	25	6	a, b, c
POPA = Pop All	01100001	40	16	a
XCHG = Exchange				
Register/Memory with Register	1000011w | mod reg r/m	3/5**	0/2**	a, m
Register with Accumulator (Short Form)	10010 reg	3	0	
IN = Input from:				
Fixed Port	1110010w | port number	6*	1*	f,k
		26*	1*	f,l
Variable Port	1110110w	7*	1*	f,k
		27*	1*	f,l
OUT = Output to:				
Fixed Port	1110011w | port number	4*	1*	f,k
		24*	1*	f,l
Variable Port	1110111w	5*	1*	f,k
		26*	1*	f,l
LEA = Load EA to Register	10001101 | mod reg r/m	2		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
SEGMENT CONTROL				
LDS = Load Pointer to DS	11000101 | mod reg r/m	26*	6*	a, b, c
LES = Load Pointer to ES	11000100 | mod reg r/m	26*	6*	a, b, c
LFS = Load Pointer to FS	00001111 | 10110100 | mod reg r/m	29*	6*	a, b, c
LGS = Load Pointer to GS	00001111 | 10110101 | mod reg r/m	29*	6*	a, b, c
LSS = Load Pointer to SS	00001111 | 10110010 | mod reg r/m	26*	6*	a, b, c
FLAG CONTROL				
CLC = Clear Carry Flag	11111000	2		
CLD = Clear Direction Flag	11111100	2		
CLI = Clear Interrupt Enable Flag	11111010	8		f
CLTS = Clear Task Switched Flag	00001111 | 00000110	5		e
CMC = Complement Carry Flag	11110101	2		
LAHF = Load AH into Flag	10011111	2		
POPF = Pop Flags	10011101	7		a, g
PUSHF = Push Flags	10011100	4		a
SAHF = Store AH into Flags	10011110	3		
STC = Set Carry Flag	11111001	2		
STD = Set Direction Flag	11111101	2		
STI = Set Interrupt Enable Flag	11111011	8		f
ARITHMETIC				
ADD = Add				
Register to Register	000000dw | mod reg r/m	2		
Register to Memory	0000000w | mod reg r/m	7**	2**	a
Memory to Register	0000001w | mod reg r/m	6*	1*	a
Immediate to Register/Memory	100000sw | mod 000 r/m | immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0000010w | immediate data	2		
ADC = Add with Carry				
Register to Register	000100dw | mod reg r/m	2		
Register to Memory	0001000w | mod reg r/m	7**	2**	a
Memory to Register	0001001w | mod reg r/m	6*	1*	a
Immediate to Register/Memory	100000sw | mod 010 r/m | immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0001010w | immediate data	2		
INC = Increment				
Register/Memory	1111111w | mod 000 r/m	2/6**	0/2**	a
Register (Short Form)	01000 reg	2		
SUB = Subtract				
Register from Register	001010dw | mod reg r/m	2		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number Of Data Cycles	Notes
ARITHMETIC (Continued)				
Register from Memory	0010100w mod reg r/m	7**	2**	a
Memory from Register	0010101w mod reg r/m	6*	1	a
Immediate from Register/Memory	100000sw mod101 r/m immediate data	2/7**	0/1**	a
Immediate from Accumulator (Short Form)	0010110w immediate data	2		
SBB = Subtract with Borrow				
Register from Register	000110dw mod reg r/m	2		
Register from Memory	0001100w mod reg r/m	7**	2**	a
Memory from Register	0001101w mod reg r/m	6*	1*	a
Immediate from Register/Memory	100000sw mod011 r/m immediate data	2/7**	0/2**	a
Immediate from Accumulator (Short Form)	0001110w immediate data	2		
DEC = Decrement				
Register/Memory	1111111w reg001 r/m	2/6**	0/2**	a
Register (Short Form)	01001 reg	2		
CMP = Compare				
Register with Register	001110dw mod reg r/m	2		
Memory with Register	0011100w mod reg r/m	5*	1*	a
Register with Memory	0011101w mod reg r/m	6**	2**	a
Immediate with Register/Memory	100000sw mod111 r/m immediate data	2/5*	0/1*	a
Immediate with Accumulator (Short Form)	0011110w immediate data	2		
NEG = Change Sign	1111011w mod011 r/m	2/6*	0/2*	a
AAA = ASCII Adjust for Add	00110111	4		
AAS = ASCII Adjust for Subtract	00111111	4		
DAA = Decimal Adjust for Add	00100111	4		
DAS = Decimal Adjust for Subtract	00101111	4		
MUL = Multiply (Unsigned)				
Accumulator with Register/Memory	1111011w mod100 r/m			
Multiplier—Byte		12–17/15–20	0/1	a,n
—Word		12–25/15–28*	0/1*	a,n
—Doubleword		12–41/17–46*	0/2*	a,n
IMUL = Integer Multiply (Signed)				
Accumulator with Register/Memory	1111011w mod101 r/m			
Multiplier—Byte		12–17/15–20	0/1	a,n
—Word		12–25/15–28*	0/1*	a,n
—Doubleword		12–41/17–46*	0/2*	a,n
Register with Register/Memory	00001111 10101111 mod reg r/m			
Multiplier—Byte		12–17/15–20	0/1	a,n
—Word		12–25/15–28*	0/1*	a,n
—Doubleword		12–41/17–46*	0/2*	a,n
Register/Memory with Immediate to Register	011010s1 mod reg r/m immediate data			
—Word		13–26/14–27*	0/1*	a,n
—Doubleword		13–42/16–45*	0/2*	a,n

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number Of Data Cycles	Notes
ARITHMETIC (Continued)				
DIV = Divide (Unsigned)				
Accumulator by Register/Memory	1 1 1 1 0 1 1 w | mod 1 1 0 r/m			
Divisor—Byte		14/17	0/1	a, o
—Word		22/25*	0/1*	a, o
—Doubleword		38/43*	0/2*	a, o
IDIV = Integer Divide (Signed)				
Accumulator by Register/Memory	1 1 1 1 0 1 1 w | mod 1 1 1 r/m			
Divisor—Byte		19/22	0/1	a, o
—Word		27/30*	0/1	a, o
—Doubleword		43/48*	0/2*	a, o
AAD = ASCII Adjust for Divide	1 1 0 1 0 1 0 1 | 0 0 0 0 1 0 1 0	19		
AAM = ASCII Adjust for Multiply	1 1 0 1 0 1 0 0 | 0 0 0 0 1 0 1 0	17		
CBW = Convert Byte to Word	1 0 0 1 1 0 0 0	3		
CWD = Convert Word to Double Word	1 0 0 1 1 0 0 1	2		
LOGIC				
Shift Rotate Instructions Not Through Carry (**ROL, ROR, SAL, SAR, SHL,** and **SHR**)				
Register/Memory by 1	1 1 0 1 0 0 0 w | mod TTT r/m	3/7**	0/2**	a
Register/Memory by CL	1 1 0 1 0 0 1 w | mod TTT r/m	3/7**	0/2**	a
Register/Memory by Immediate Count	1 1 0 0 0 0 0 w | mod TTT r/m | immed 8-bit data	3/7**	0/2**	a
Through Carry (**RCL** and **RCR**)				
Register/Memory by 1	1 1 0 1 0 0 0 w | mod TTT r/m	9/10**	0/2**	a
Register/Memory by CL	1 1 0 1 0 0 1 w | mod TTT r/m	9/10**	10/2**	a
Register/Memory by Immediate Count	1 1 0 0 0 0 0 w | mod TTT r/m | immed 8-bit data	9/10**	0/2**	a
	TTT Instruction 0 0 0 ROL 0 0 1 ROR 0 1 0 RCL 0 1 1 RCR 1 0 0 SHL/SAL 1 0 1 SHR 1 1 1 SAR			
SHLD = Shift Left Double				
Register/Memory by Immediate	0 0 0 0 1 1 1 1 | 1 0 1 0 0 1 0 0 | mod reg r/m | immed 8-bit data	3/7**	0/2**	
Register/Memory by CL	0 0 0 0 1 1 1 1 | 1 0 1 0 0 1 0 1 | mod reg r/m	3/7**	0/2**	
SHRD = Shift Right Double				
Register/Memory by Immediate	0 0 0 0 1 1 1 1 | 1 0 1 0 1 1 0 0 | mod reg r/m | immed 8-bit data	3/7**	0/2**	
Register/Memory by CL	0 0 0 0 1 1 1 1 | 1 0 1 0 1 1 0 1 | mod reg r/m	3/7**	0/2**	
AND = And				
Register to Register	0 0 1 0 0 0 d w | mod reg r/m	2		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
LOGIC (Continued)				
Register to Memory	0010000w mod reg r/m	7**	2**	a
Memory to Register	0010001w mod reg r/m	6*	1*	a
Immediate to Register/Memory	1000000w mod100 r/m immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0010010w immediate data	2		
TEST = And Function to Flags, No Result				
Register/Memory and Register	1000010w mod reg r/m	2/5*	0/1*	a
Immediate Data and Register/Memory	1111011w mod000 r/m immediate data	2/5*	0/1*	a
Immediate Data and Accumulator (Short Form)	1010100w immediate data	2		
OR = Or				
Register to Register	000010dw mod reg r/m	2		
Register to Memory	0000100w mod reg r/m	7**	2**	a
Memory to Register	0000101w mod reg r/m	6*	1*	a
Immediate to Register/Memory	1000000w mod001 r/m immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0000110w immediate data	2		
XOR = Exclusive Or				
Register to Register	001100dw mod reg r/m	2		
Register to Memory	0011000w mod reg r/m	7**	2**	a
Memory to Register	0011001w mod reg r/m	6*	1*	a
Immediate to Register/Memory	1000000w mod110 r/m immediate data	2/7**	0/2**	a
Immediate to Accumulator (Short Form)	0011010w immediate data	2		
NOT = Invert Register/Memory	1111011w mod010 r/m	2/6**	0/2**	a
STRING MANIPULATION				
CMPS = Compare Byte Word	1010011w	10*	2*	a
INS = Input Byte/Word from DX Port	0110110w	9** 29**	1** 1**	a,f,k a,f,l
LODS = Load Byte/Word to AL/AX/EAX	1010110w	5*	1*	a
MOVS = Move Byte Word	1010010w	7**	2**	a
OUTS = Output Byte/Word to DX Port	0110111w	8** 28**	1** 1**	a,f,k a,f,l
SCAS = Scan Byte Word	1010111w	7*	1*	a
STOS = Store Byte/Word from AL/AX/EX	1010101w	4*	1*	a
XLAT = Translate String	11010111	5*	1*	a
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX				
REPE CMPS = Compare String (Find Non-Match)	11110011 1010011w	5 + 9n**	2n**	a

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
REPEATED STRING MANIPULATION (Continued)				
REPNE CMPS = Compare String				
(Find Match)	11110010 | 1010011w	5 + 9n**	2n**	a
REP INS = Input String	11110011 | 0110110w	7 + 6n* 27 + 6n*	1n* 1n*	a,f,k a,f,l
REP LODS = Load String	11110011 | 1010110w	5 + 6n*	1n*	a
REP MOVS = Move String	11110011 | 1010010w	7 + 4n**	2n**	a
REP OUTS = Output String	11110011 | 0110111w	6 + 5n* 26 + 5n*	1n* 1n*	a,f,k a,f,l
REPE SCAS = Scan String				
(Find Non-AL/AX/EAX)	11110011 | 1010111w	5 + 8n*	1n*	a
REPNE SCAS = Scan String				
(Find AL/AX/EAX)	11110010 | 1010111w	5 + 8n*	1n*	a
REP STOS = Store String	11110011 | 1010101w	5 + 5n*	1n*	a
BIT MANIPULATION				
BSF = Scan Bit Forward	00001111 | 10111100 | mod reg r/m	10 + 3n**	2n**	a
BSR = Scan Bit Reverse	00001111 | 10111101 | mod reg r/m	10 + 3n**	2n**	a
BT = Test Bit				
Register/Memory, Immediate	00001111 | 10111010 | mod 100 r/m | immed 8-bit data	3/6*	0/1*	a
Register/Memory, Register	00001111 | 10100011 | mod reg r/m	3/12*	0/1*	a
BTC = Test Bit and Complement				
Register/Memory, Immediate	00001111 | 10111010 | mod 111 r/m | immed 8-bit data	6/8*	0/2*	a
Register/Memory, Register	00001111 | 10111011 | mod reg r/m	6/13*	0/2*	a
BTR = Test Bit and Reset				
Register/Memory, Immediate	00001111 | 10111010 | mod 110 r/m | immed 8-bit data	6/8*	0/2*	a
Register/Memory, Register	00001111 | 10110011 | mod reg r/m	6/13*	0/2*	a
BTS = Test Bit and Set				
Register/Memory, Immediate	00001111 | 10111010 | mod 101 r/m | immed 8-bit data	6/8*	0/2*	a
Register/Memory, Register	00001111 | 10101011 | mod reg r/m	6/13*	0/2*	a
CONTROL TRANSFER				
CALL = Call				
Direct within Segment	11101000 | full displacement	9 + m*	2	j
Register/Memory				
Indirect within Segment	11111111 | mod 010 r/m	9 + m/12 + m	2/3	a, j
Direct Intersegment	10011010 | unsigned full offset, selector	42 + m	9	c, d, j

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
CONTROL TRANSFER (Continued)				
(Direct Intersegment)				
Via Call Gate to Same Privilege Level		64 + m	13	a,c,d,j
Via Call Gate to Different Privilege Level, (No Parameters)		98 + m	13	a,c,d,j
Via Call Gate to Different Privilege Level, (x Parameters)		106 + 8x + m	13 + 4x	a,c,d,j
From 386 Task to 386 TSS		392	124	a,c,d,j
Indirect Intersegment	11111111 | mod 011 r/m	46 + m	10	a,c,d,j
Via Call Gate to Same Privilege Level		68 + m	14	a,c,d,j
Via Call Gate to Different Privilege Level, (No Parameters)		102 + m	14	a,c,d,j
Via Call Gate to Different Privilege Level, (x Parameters)		110 + 8x + m	14 + 4x	a,c,d,j
From 386 Task to 386 TSS		399	130	a,c,d,j
JMP = Unconditional Jump				
Short	11101011 | 8-bit displacement	7 + m		j
Direct within Segment	11101001 | full displacement	7 + m		j
Register/Memory Indirect within Segment	11111111 | mod 100 r/m	9 + m/14 + m	2/4	a,j
Direct Intersegment	11101010 | unsigned full offset, selector	37 + m	5	c,d,j
Via Call Gate to Same Privilege Level		53 + m	9	a,c,d,j
From 386 Task to 386 TSS		395	124	a,c,d,j
Indirect Intersegment	11111111 | mod 101 r/m	37 + m	9	a,c,d,j
Via Call Gate to Same Privilege Level		59 + m	13	a,c,d,j
From 386 Task to 386 TSS		401	124	a,c,d,j

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
CONTROL TRANSFER (Continued)				
RET = Return from CALL:				
Within Segment	11000011	12 + m	2	a,j,p
Within Segment Adding Immediate to SP	11000010 | 16-bit displ	12 + m	2	a,j,p
Intersegment	11001011	36 + m	4	a,c,d,j,p
Intersegment Adding Immediate to SP	11001010 | 16-bit displ	36 + m	4	a,c,d,j,p
to Different Privilege Level				
Intersegment		80	4	c,d,j,p
Intersegment Adding Immediate to SP		80	4	c,d,j,p
CONDITIONAL JUMPS				
NOTE: Times Are Jump "Taken or Not Taken"				
JO = Jump on Overflow				
8-Bit Displacement	01110000 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000000 | full displacement	7 + m or 3		j
JNO = Jump on Not Overflow				
8-Bit Displacement	01110001 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000001 | full displacement	7 + m or 3		j
JB/JNAE = Jump on Below/Not Above or Equal				
8-Bit Displacement	01110010 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000010 | full displacement	7 + m or 3		j
JNB/JAE = Jump on Not Below/Above or Equal				
8-Bit Displacement	01110011 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000011 | full displacement	7 + m or 3		j
JE/JZ = Jump on Equal/Zero				
8-Bit Displacement	01110100 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000100 | full displacement	7 + m or 3		j
JNE/JNZ = Jump on Not Equal/Not Zero				
8-Bit Displacement	01110101 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000101 | full displacement	7 + m or 3		j
JBE/JNA = Jump on Below or Equal/Not Above				
8-Bit Displacement	01110110 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000110 | full displacement	7 + m or 3		j
JNBE/JA = Jump on Not Below or Equal/Above				
8-Bit Displacement	01110111 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10000111 | full displacement	7 + m or 3		j
JS = Jump on Sign				
8-Bit Displacement	01111000 | 8-bit displ	7 + m or 3		j
Full Displacement	00001111 | 10001000 | full displacement	7 + m or 3		j

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format			Clock Counts	Number of Data Cycles	Notes
CONDITIONAL JUMPS (Continued)						
JNS = Jump on Not Sign						
8-Bit Displacement	01111001	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001001	full displacement	7 + m or 3		j
JP/JPE = Jump on Parity/Parity Even						
8-Bit Displacement	01111010	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001010	full displacement	7 + m or 3		j
JNP/JPO = Jump on Not Parity/Parity Odd						
8-Bit Displacement	01111011	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001011	full displacement	7 + m or 3		j
JL/JNGE = Jump on Less/Not Greater or Equal						
8-Bit Displacement	01111100	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001100	full displacement	7 + m or 3		j
JNL/JGE = Jump on Not Less/Greater or Equal						
8-Bit Displacement	01111101	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001101	full displacement	7 + m or 3		j
JLE/JNG = Jump on Less or Equal/Not Greater						
8-Bit Displacement	01111110	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001110	full displacement	7 + m or 3		j
JNLE/JG = Jump on Not Less or Equal/Greater						
8-Bit Displacement	01111111	8-bit displ		7 + m or 3		j
Full Displacement	00001111	10001111	full displacement	7 + m or 3		j
JECXZ = Jump on ECX Zero	11100011	8-bit displ		9 + m or 5		j
(Address Size Prefix Differentiates JCXZ from JECXZ)						
LOOP = Loop ECX Times	11100010	8-bit displ		11 + m		j
LOOPZ/LOOPE = Loop with Zero/Equal	11100001	8-bit displ		11 + m		j
LOOPNZ/LOOPNE = Loop While Not Zero	11100000	8-bit displ		11 + m		j
CONDITIONAL BYTE SET						
NOTE: Times Are Register/Memory						
SETO = Set Byte on Overflow						
To Register/Memory	00001111	10010000	mod 000 r/m	4/5*	0/1*	a
SETNO = Set Byte on Not Overflow						
To Register/Memory	00001111	10010001	mod 000 r/m	4/5*	0/1*	a
SETB/SETNAE = Set Byte on Below/Not Above or Equal						
To Register/Memory	00001111	10010010	mod 000 r/m	4/5*	0/1*	a

2

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format			Clock Counts	Number of Data Cycles	Notes
CONDITIONAL BYTE SET (Continued)						
SETNB = Set Byte on Not Below/Above or Equal						
To Register/Memory	00001111	10010011	mod000 r/m	4/5*	0/1*	a
SETE/SETZ = Set Byte on Equal/Zero						
To Register/Memory	00001111	10010100	mod000 r/m	4/5*	0/1*	a
SETNE/SETNZ = Set Byte on Not Equal/Not Zero						
To Register/Memory	00001111	10010101	mod000 r/m	4/5*	0/1*	a
SETBE/SETNA = Set Byte on Below or Equal/Not Above						
To Register/Memory	00001111	10010110	mod000 r/m	4/5*	0/1*	a
SETNBE/SETA = Set Byte on Not Below or Equal/Above						
To Register/Memory	00001111	10010111	mod000 r/m	4/5*	0/1*	a
SETS = Set Byte on Sign						
To Register/Memory	00001111	10011000	mod000 r/m	4/5*	0/1*	a
SETNS = Set Byte on Not Sign						
To Register/Memory	00001111	10011001	mod000 r/m	4/5*	0/1*	a
SETP/SETPE = Set Byte on Parity/Parity Even						
To Register/Memory	00001111	10011010	mod000 r/m	4/5*	0/1*	a
SETNP/SETPO = Set Byte on Not Parity/Parity Odd						
To Register/Memory	00001111	10011011	mod000 r/m	4/5*	0/1*	a
SETL/SETNGE = Set Byte on Less/Not Greater or Equal						
To Register/Memory	00001111	10011100	mod000 r/m	4/5*	0/1*	a
SETNL/SETGE = Set Byte on Not Less/Greater or Equal						
To Register/Memory	00001111	01111101	mod000 r/m	4/5*	0/1*	a
SETLE/SETNG = Set Byte on Less or Equal/Not Greater						
To Register/Memory	00001111	10011110	mod000 r/m	4/5*	0/1*	a
SETNLE/SETG = Set Byte on Not Less or Equal/Greater						
To Register/Memory	00001111	10011111	mod000 r/m	4/5*	0/1*	a
ENTER = Enter Procedure	11001000	16-bit displacement, 8-bit level				
L = 0				10		a
L = 1				14	1	a
L > 1				17 + 8(n − 1)	4(n − 1)	a
LEAVE = Leave Procedure	11001001			6		a

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
INTERRUPT INSTRUCTIONS				
INT = Interrupt:				
Type Specified	11001101 | type			
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		111	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		467	140	c,d,j,p
Type 3	11001100			
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		111	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		308	138	c,d,j,p
INTO = Interrupt 4 if Overflow Flag Set	11001110			
If OF = 1:		3		
If OF = 0				
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		111	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		413	138	c,d,j,p

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number Of Data Cycles	Notes
INTERRUPT INSTRUCTIONS (Continued)				
Bound = Out of Range Interrupt 5 if Detect Value	01100010 | mod reg r/m			
If In Range		10	0	a,c,d,j,o,p
If Out of Range:				
Via Interrupt or Trap Gate to Same Privilege Level		71	14	c,d,j,p
Via Interrupt or Trap Gate to Different Privilege Level		111	14	c,d,j,p
From 386 Task to 386 TSS via Task Gate		398	138	c,d,j,p
INTERRUPT RETURN				
IRET = Interrupt Return	11001111			
To the Same Privilege Level (within Task)		42	5	a,c,d,j,p
To Different Privilege Level (within Task)		86	5	a,c,d,j,p
From 386 Task to 386 TSS		328	138	c,d,j,p
PROCESSOR CONTROL				
HLT = HALT	11110100	5		b
MOV = Move to and from Control/Debug/Test Registers				
CR0 from register	00001111 | 00100010 | 11 eee reg	10		b
Register from CR0	00001111 | 00100000 | 11 eee reg	6		b
DR0–3 from Register	00001111 | 00100011 | 11 eee reg	22		b
DR6–7 from Register	00001111 | 00100011 | 11 eee reg	16		b
Register from DR6–7	00001111 | 00100001 | 11 eee reg	14		b
Register from DR0–3	00001111 | 00100001 | 11 eee reg	22		b
NOP = No Operation	10010000	3		
WAIT = Wait until $\overline{\text{BUSY}}$ Pin is Negated	10011011	6		

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
PROCESSOR EXTENSION INSTRUCTIONS				
Processor Extension Escape	11011TTT | mod LLL r/m TTT and LLL bits are opcode information for coprocessor.	See 80387SX Data Sheet		a
PREFIX BYTES				
Address Size Prefix	01100111	0		
LOCK = Bus Lock Prefix	11110000	0		f
Operand Size Prefix	01100110	0		
Segment Override Prefix				
CS:	00101110	0		
DS:	00111110	0		
ES:	00100110	0		
FS:	01100100	0		
GS:	01100101	0		
SS:	00110110	0		
PROTECTION CONTROL				
ARPL = Adjust Requested Privilege Level				
From Register/Memory	01100011 | mod reg r/m	20/21**	2**	a
LAR = Load Access Rights				
From Register/Memory	00001111 | 00000010 | mod reg r/m	17/18*	1*	a,c,i,p
LGDT = Load Global Descriptor				
Table Register	00001111 | 00000001 | mod 010 r/m	13**	3*	a,e
LIDT = Load Interrupt Descriptor				
Table Register	00001111 | 00000001 | mod 011 r/m	13**	3*	a,e
LLDT = Load Local Descriptor				
Table Register to Register/Memory	00001111 | 00000000 | mod 010 r/m	24/28*	5*	a,c,e,p
LMSW =Load Machine Status Word				
From Register/Memory	00001111 | 00000001 | mod 110 r/m	10/13*	1*	a,e
LSL = Load Segment Limit				
From Register/Memory	00001111 | 00000011 | mod reg r/m			
Byte-Granular Limit		24/27*	2*	a,c,i,p
Page-Granular Limit		29/32*	2*	a,c,i,p
LTR = Load Task Register				
From Register/Memory	00001111 | 00000000 | mod 001 r/m	27/31*	4*	a,c,e,p
SGDT = Store Global Descriptor				
Table Register	00001111 | 00000001 | mod 000 r/m	11*	3*	a
SIDT = Store Interrupt Descriptor				
Table Register	00001111 | 00000001 | mod 001 r/m	11*	3*	a
SLDT = Store Local Descriptor Table Register				
To Register/Memory	00001111 | 00000000 | mod 000 r/m	2/2*	4*	a

Table 8.1. 80376 Instruction Set Clock Count Summary (Continued)

Instruction	Format	Clock Counts	Number of Data Cycles	Notes
PROTECTION CONTROL (Continued)				
SMSW = Store Machine Status Word	00001111 | 00000001 | mod100 r/m	2/2*	1*	a, c
STR = Store Task Register To Register/Memory	00001111 | 00000000 | mod001 r/m	2/2*	1*	a
VERR = Verify Read Accesss Register/Memory	00001111 | 00000000 | mod100 r/m	10/11**	2**	a,c,i,p
VERW = Verify Write Accesss	00001111 | 00000000 | mod101 r/m	15/16**	2**	a,c,i,p

NOTES:
a. Exception 13 fault (general violation) will occur if the memory operand in CS, DS, ES, FS or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, and exception 12 (stack segment limit violation or not present) occurs.
b. For segment load operations, the CPL, RPL and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segments's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present occurs).
c. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert $\overline{LOCK}$ to maintain descriptor integrity in multiprocessor systems.
d. JMP, CALL, INT, RET and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is volated.
e. An exception 13 fault occurs if CPL is greater than 0.
f. An exception 13 fault occurs if CPL is greater than IOPL.
g. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL field of the flag register is updated only if CPL = 0.
h. Any violation of privelege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.
i. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or no present) will occur if the stack limit is violated by the operand's starting address.
j. The destination of a JMP, CALL, INT, RET or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.
k. If CPL $\leq$ IOPL
l. If CPL > IOPL
m. $\overline{LOCK}$ is automatically asserted, regardless of the presence or absence of the $\overline{LOCK}$ prefix.
n. The 80376 uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier). Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:

Actual Clock = if m < > 0 then max ([$\log_2$ |m|], 3) + 9 clocks:
if m = 0 then 12 clocks (where m is the multiplier)

o. An exception may occur, depending on the value of the operand.
p. $\overline{LOCK}$ is asserted during descriptor table accesses.

8.2 INSTRUCTION ENCODING

Overview

All instruction encodings are subsets of the general instruction format shown in Figure 8.1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the "mod r/m" byte and "scaled index" byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16 or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure 8.1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the Figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table 8.2 is a complete list of all fields appearing in the 80376 instruction set. Further ahead, following Table 8.2, are detailed tables for each field.

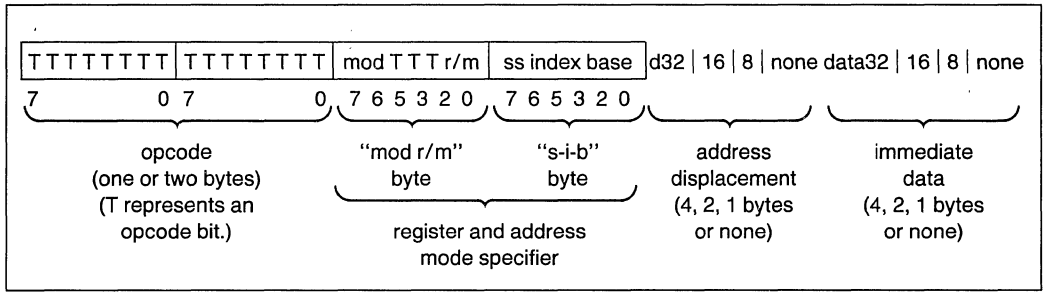


Figure 8.1. General Instruction Format

Table 8.2. Fields within 80376 Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bits	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

Note: Table 8.1 shows encoding of individual instructions.

16-Bit Extensions of the Instruction Set

Two prefixes, the operand size prefix (66H) and the effective address size prefix (67H), allow overriding individually the default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the opcode bytes. The presence of the operand size prefix (66H) and the effective address prefix will allow 16-bit data operation and 16-bit effective address calculations.

For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on.

ENCODING OF OPERAND LENGTH (w) FIELD

For any given instruction performing a data operation, the instruction will execute as a 32-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size with 66H Prefix	Normal Operand Size
0	8 Bits	8 Bits
1	16 Bits	32 Bits

ENCODING OF THE GENERAL REGISTER (reg) FIELD

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected with 66H Prefix	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field with 66H Prefix		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register Specified by reg Field without 66H Prefix		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

ENCODING OF THE SEGMENT REGISTER (sreg) FIELD

The sreg field in certain instructions is a 2-bit field allowing one of the CS, DS, ES or SS segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the FS and GS segment registers to be specified also.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

ENCODING OF ADDRESS MODE

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte, can be specified.

The s-i-b byte (scale-index-base byte) is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01 or 10. When the sib byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure 8.1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following three pages define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Encoding of Normal Address Mode with "mod r/m" byte (no "s-i-b" byte present):

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]

mod r/m	Effective Address
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by reg or r/m during Normal Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Register Specified by reg or r/m during 16-Bit Data Operations: (66H Prefix)		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Encoding of 16-bit Address Mode with "mod r/m" Byte Using 67H Prefix

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	SS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]

mod r/m	Effective Address
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP + DI + d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Encoding of 32-bit Address Mode ("mod r/m" byte and "s-i-b" byte present):

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

NOTE:
Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg**
101	EBP
110	ESI
111	EDI

****IMPORTANT NOTE:**
When index field is 100, indicating "no index register," then ss field MUST equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

ENCODING OF OPERATION DIRECTION (d) FIELD

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory <- - Register "reg" Field Indicates Source Operand; "mod r/m" or "mod ss index base" Indicates Destination Operand
1	Register <- - Register/Memory "reg" Field Indicates Destination Operand; "mod r/m" or "mod ss index base" Indicates Source Operand

ENCODING OF SIGN-EXTEND (s) FIELD

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

s	Effect on Immediate Data8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extend Data8 to Fill 16-Bit or 32-Bit Destination	None

ENCODING OF CONDITIONAL TEST (tttn) FIELD

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n=0) or its negation (n=1), and tt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111

ENCODING OF CONTROL OR DEBUG REGISTER (eee) FIELD

For the loading and storing of the Control and Debug registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	Reserved
011	Reserved
Do not use any other encoding	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding	

9.0 REVISION HISTORY

The sections significantly revised since version -003 are:

Section 1.0 Added $\overline{FLT}$ pin.

Section 4.4 Added description of FLOAT operation and ONCE Mode. Figure 4.20 is new.

Section 4.6 Added revision identifier information for change to CHMOS IV manufacturing process.

Section 5.0 Both packages now specified for 0°C–115°C case temperature operation. Thermal resistance values changed.

Section 6.3 I_{CC} Max. specifications changed from 400 mA (cold) and 360 mA (hot) to 275 mA (cold, 16 MHz) and 305 mA (cold, 20 MHz).

Section 6.4 HLDA Valid Delay, t_{14}, min. changed from 6 ns to 4 ns. Added 20 MHz A.C. specifications in Table 6.5. Replaced Capacitive Derating Curves in Figures 6.8–6.10 to reflect new manufacturing process. Replaced I_{CC} vs. Frequency data (Figure 6.11) to reflect new specifications.

The sections significantly revised since version -002 are:

Section 1.0 Modified table 1.1. to list pins in alphabetical order.

The sections significantly revised since version -001 are:

Section 2.0 Figure 2.0 was updated to show the 16-bit registers SI, DI, BP and SP.

Section 2.1 Figure 2.2 was updated to show the correct bit polarity for bit 4 in the CR0 register.

Section 2.1 Tables 2.1 and 2.2 were updated to include additional information on the EFLAGs and CR0 registers.

Section 2.3 Figure 2.3 was updated to more accurately reflect the addressing mechanism of the 80376.

Section 2.6 In the subsection **Maskable Interrupt** a paragraph was added to describe the effect of interrupt gates on the IF EFLAGs bit.

Section 2.8 Table 2.7 was updated to reflect the correct power up condition of the CR0 register.

Section 2.10 Figure 2.6 was updated to show the correct bit positions of the BT, BS and BD bits in the DR6 register.

Section 3.0 Figure 3.1 was updated to clearly show the address calculation process.

Section 3.2 The subsection **DESCRIPTORS** was elaborated upon to clearly define the relationship between the linear address space and physical address space of the 80376.

Section 3.2 Figures 3.3 and 3.4 were updated to show the AVL bit field.

Section 3.3 The last sentence in the first paragraph of subsection **PROTECTION AND I/O PERMISSION BIT MAP** was deleted. This was an incorrect statement.

Section 4.1 In the Subsection **ADDRESS BUS ($\overline{BHE}$, $\overline{BLE}$, A_{23}–A_1** last sentence in the first paragraph was updated to reflect the numerics operand addresses as 8000FCH and 8000FEH. Because the 80376 sometimes does a double word I/O access a second access to 8000FEH can be seen.

Section 4.1 The Subsection **Hold Lantencies** was updated to describe how 32-bit and unaligned accesses are internally locked but do not assert the $\overline{LOCK}$ signal.

Section 4.2 Table 4.6 was updated to show the correct active data bits during a $\overline{BLE}$ assertion.

9.0 REVISION HISTORY (Continued)

Section 4.4 This section was updated to correctly reflect the pipelining of the address and status of the 80376 as opposed to "Address Pipelining" which occurs on processors such as the 80286.

Section 4.6 Table 4.7 was updated to show the correct Revision number, 05H.

Section 4.7 Table 4.8 was updated to show the numerics operand register 8000FEH. This address is seen when the 80376 does a DWORD operation to the port address 8000FCH.

Section 5.0 In the first paragraph the case temperatures were updated to reflect the 0°C–115°C for the ceramic package and 0°C–110°C for the plastic package.

Section 6.2 Table 6.2 was updated to reflect the Case Temperature under Bias specification of −65°C–120°C.

Section 6.4 Figure 6.8 vertical axis was updated to reflect "Output Valid Delay (ns)".

Section 6.4 Figure 6.11 was updated to show typical I_{CC} vs Frequency for the 80376.

Section 8.1 The clock counts and opcodes for various instructions were updated to their correct values.

Section 8.2 The section **INSTRUCTION ENCODING** was appended to the data sheet.

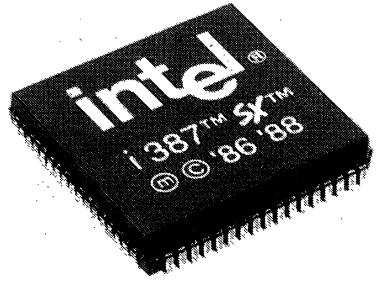
Intel387™ SX MATH COPROCESSOR

- **New Automatic Power Management**
  - **— Low Power Consumption**
  - **— Typically 100 mA in Dynamic Mode, and 4 mA in Idle Mode**
- **Socket Compatible with Intel387 Family of Math CoProcessors**
  - **— Hardware and Software Compatible**
  - **— Supported by Over 2100 Commercial Software Packages**
  - **— 10% to 15% Performance Increase on Whetstone and Livermore Benchmarks**
- **Compatible with the Intel386™ SX Microprocessor**
  - **— Extends CPU Instruction Set to Include Trigonometric, Logarithmic, and Exponential**
- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Available in a 68-Pin PLCC Package**

  See Intel Packaging Specification, Order #231369

The Intel387™ SX Math CoProcessor is an extension to the Intel386™ SX microprocessor architecture. The combination of the Intel387™ SX with the Intel386™ SX microprocessor dramatically increases the processing speed of computer application software that utilizes high performance floating-point operations. An internal Power Management Unit enables the Intel387™ SX to perform these floating-point operations while maintaining very low power consumption for portable and desktop applications. The internal Power Management Unit effectively reduces power consumption by 95% when the device is idle.

The Intel387™ SX Math CoProcessor is available in a 68-pin PLCC package, and is manufactured on Intel's advanced 1.0 micron CHMOS IV technology.



240225-22

Intel386 and Intel387 are trademarks of Intel Corporation.

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

January 1994
Order Number: 240225-009

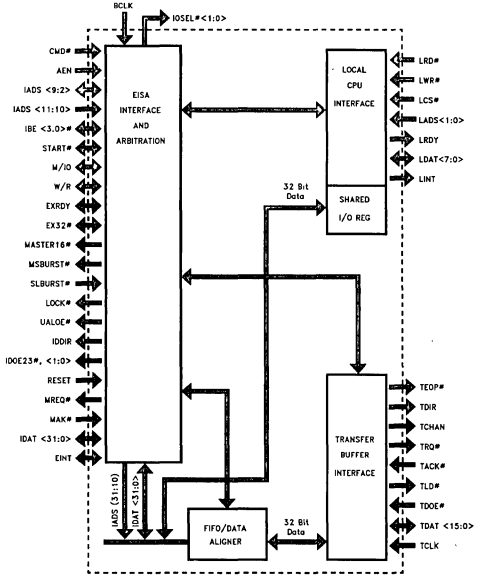
82355
BUS MASTER INTERFACE CONTROLLER (BMIC)

- Designed for use in 32-Bit EISA Bus Master Expansion Board Designs
  - Integrates Three Interfaces (EISA, Local CPU, and Transfer Buffer)
- Supports 16- and 32-Bit Burst Transfers
  - 33 Mbytes/Sec Maximum Data Transfers
- Supports 32-Bit Non-Burst and Mismatched Data Size Transfers
- Supports 32-Bit EISA Addressability (4 Gigabyte)
- Two independent Data Transfer Channels with 24-Byte FIFOs
  - Expansion Board Timing and EISA Timing Operate Asynchronously
- Supports Peek/Poke Operation with the Ability to Access Individual Locations in EISA Memory or I/O space
- Automatically Handles Misaligned Doubleword Data Transfers with No Performance Penalty
- Supports Automatic Handling of Complete EISA Bus Master Protocol
  - EISA Arbitration/Preemption
  - Cycle Timing and Execution
  - Byte Alignment
  - 1K Boundary Detection
- Supports Local Data Transfer Protocol Similar to Traditional DMA
- Supports a General Purpose Command and Status Interface
  - Local and EISA System Interrupt Support
  - General Purpose Information Transfers
  - Set-and-Test-Functions in I/O Space (Semaphore Function)
  - Supports the EISA Expansion Board ID Function
- Supports Decode of Slot Specific and General I/O Addresses
- 132-Pin JEDEC PQFP Package

  (See Packaging Specification Order #240800, Package Type NG)

2

82355 Internal Block Diagram



290255-1

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

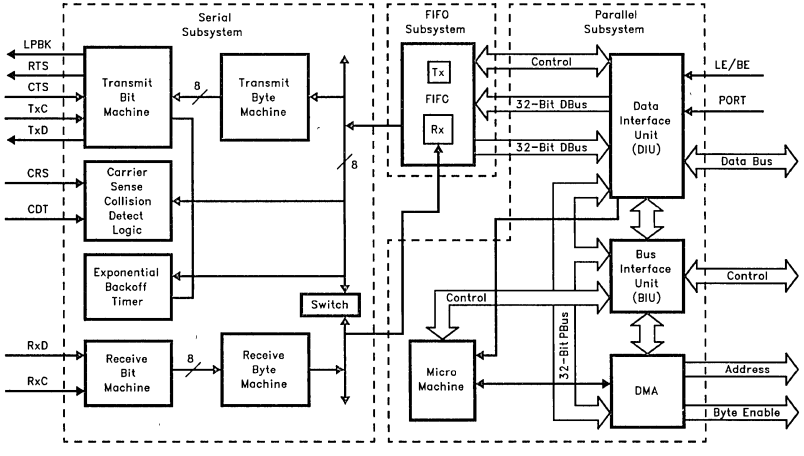
December 1995
Order Number: 290255-008

82596DX AND 82596SX HIGH-PERFORMANCE 32-BIT LOCAL AREA NETWORK COPROCESSOR

- **Performs Complete CSMA/CD Medium Access Control (MAC) Functions—Independently of CPU**
  - **— IEEE 802.3 (EOC) Frame Delimiting**
- **Supports Industry Standard LANs**
  - **— IEEE TYPE 10BASE-T (TPE), IEEE TYPE 10BASE5 (Ethernet*), IEEE TYPE 10BASE2 (Cheapernet), IEEE TYPE 1BASE5 (StarLAN), and the Proposed Standard TYPE 10BASE-F**
  - **— Proprietary CSMA/CD Networks Up to 20 Mb/s**
- **On-Chip Memory Management**
  - **— Automatic Buffer Chaining**
  - **— Buffer Reclamation after Receipt of Bad Frames; Optional Save Bad Frames**
  - **— 32-Bit Segmented or Linear (Flat) Memory Addressing Formats**
- **82586 Software Compatible**
- **Optimized CPU Interface**
  - **— 82596DX Bus Interface Optimized to Intel's 32-Bit i386™DX**
  - **— 82596SX Bus Interface Optimized to Intel's 16-Bit i386™SX**
  - **— Supports Big Endian and Little Endian Byte Ordering**
- **High-Performance 16-/32-Bit Bus Master Interface**
  - **— 66-MB/s Bus Bandwidth**
  - **— 33-MHz Clock, Two Clocks Per Transfer**
  - **— Bus Throttle Timers**
  - **— Transfers Data at 100% of Serial Bandwidth**
  - **— 128-Byte Receive FIFO, 64-Byte Transmit FIFO**
- **Network Management and Diagnostics**
  - **— Monitor Mode**
  - **— 32-Bit Statistical Counters**
- **Self-Test Diagnostics**
- **Configurable Initialization Root for Data Structures**
- **High-Speed, 5-V, CHMOS** IV Technology**
- **132-Pin Plastic Quad Flat Pack (PQFP) and PGA Package**

  (See Packaging Specifications Order Number: 240800-001, Package Type KU and A)

i386™ is a trademark of Intel Corporation
*Ethernet is a registered trademark of Xerox Corporation.
**CHMOS is a patented process of Intel Corporation.



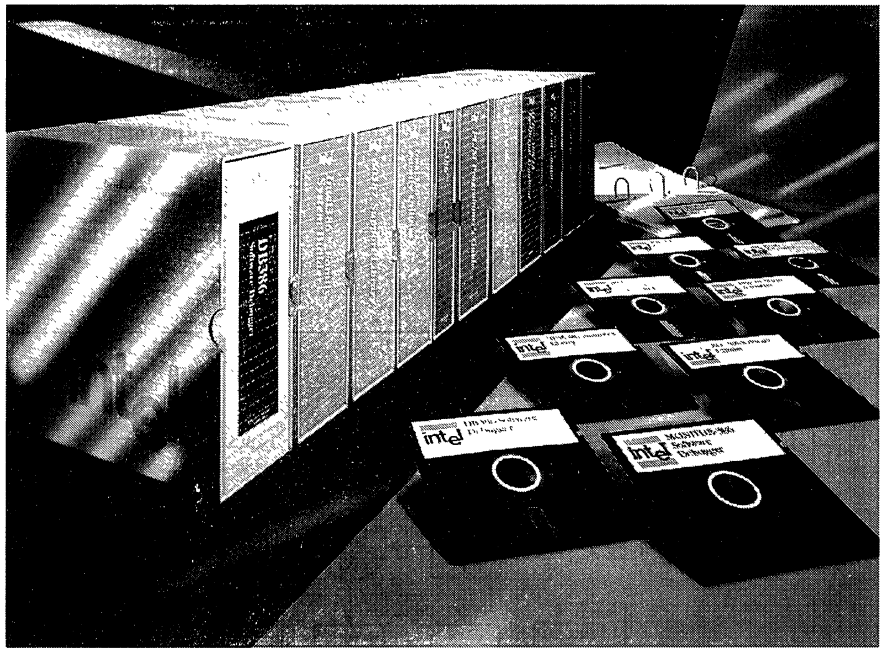
290219-1

Figure 1. 82596DX/SX Block Diagram

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

November 1995
Order Number: 290219-006

Intel386™ AND Intel486™ FAMILY DEVELOPMENT SUPPORT



280808-1

COMPREHENSIVE DEVELOPMENT SUPPORT FOR THE Intel386™ AND Intel486™ FAMILIES OF MICROPROCESSORS

The perfect complement to the Intel386™ and i486™ microprocessor family is a comprehensive development solution. Intel provides a complete, synergistic hardware and software development toolset, that delivers full access to the power of the Intel386 and i486 microprocessor family architectures.

Intel development tools are easy to use, yet powerful, with an up-date user interface and productivity boosting features such as symbolic debugging. Each tool is designed to help move your application from the lab to the market.

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 and Intel486 microprocessor architectures—includes protected mode, 4 gigabyte physical memory addressing, and Intel486 microprocessor on-chip cache and numerics
- In-circuit emulators provide a standard windowed interface that is common across Intel debug tools and architectures
- Emulators also feature a source line display and symbolics to allow debugging in the context of the original program
- Intel high-level languages provide architectural extensions for manipulating hardware directly without assembly language routines
- Languages provide a common object code format (Intel OMF386) that supports symbolic debug and permits 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
- Extensive support for the Intel family of math coprocessors
- Operation in DOS IBM PC AT and PS/2 Model 60 and 80, running DOS.

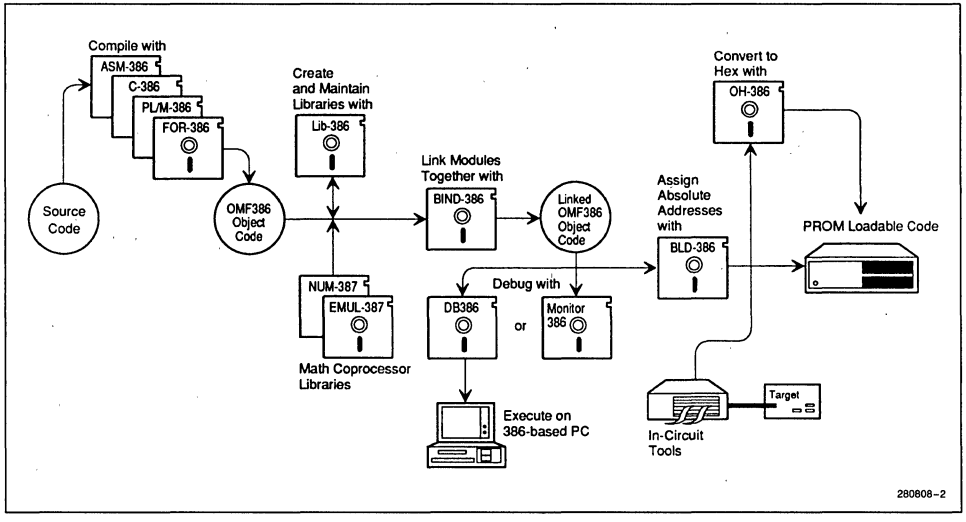


Figure 1: Intel Microprocessor Development Environment

FEATURES

ASM-386/486 MACRO ASSEMBLER

Intel's ASM 386 macro assembler for the Intel386 and Intel486 Families 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 Intel ASM 386 features include:

- "High-level" assembler mnemonics to simplify the language
- Structures and records for data representation
- Support for Intel's standard object code format for source-level symbolic debug, and for linking object modules from other Intel386 and Intel486 microprocessor languages
- Full support for processor and math coprocessor instruction sets
- A "MOD486" switch for support of the i486 microprocessor instructions
- 16 bit or 32 bit address overrides
- Supports development for Virtual 86, Real, 286 Protected, and 386 Protected modes

iC386/486 COMPILER

Intel's iC-386 compiler combines the power of C programming language with special features for architectural support and code efficiency. The compiler produces code for Intel386 and Intel486 processors from C source files, and conforms to the 1989 ANSI standard (ANS X3.159-1989) for the C programming language.

Key Intel iC-386 features include:

- Controls to tailor the compilation for each step of your application development process
- In-line versions of many ANSI-standard library functions
- Expanded memory support (LIM Version 3.0 and higher) for large applications
- Object code (including supplied run-time libraries) suitable for ROM
- Three different levels of optimization
- A choice of three segmentation memory models (small, compact, and flat) to create compact and efficient code
- In-line processor-specific functions and time-saving macros that provide access to the special features of the Intel386 and Intel486 processors
- In-line floating-point instructions for the Intel387™ numerics coprocessor and Intel486 processor floating-point unit
- Time-saving macros and functions to help assembly language routines interface with Intel's high-level programming languages
- The standard C run-time library plus libraries for floating-point support and the iRMX® III C interface library
- An easy interface to Intel's non-C programming languages
- Support for source-level debugging
- Programming with subsystems, allowing mixed segmentation memory models
- Extensions to the 1989 ANSI C standard for compatibility with previous versions Intel C
- Fast and efficient functions for common programming tasks

PL/M-386/486 COMPILER

Intel's PL/M-386 is a structured high-level system implementation language for the Intel386 and Intel486 Families. PL/M-386 supports the implementation of protected operating system software by providing built-in procedures and variables to access the Intel386 and Intel486 architectures. 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.

FEATURES

Other Intel 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 input and output from microprocessor ports
- Upward compatibility with Intel PL/M-86 and PL/M-286 source code
- A "MOD486" compiler switch for Intel486 microprocessor instruction generation

PL/M-386 combines the benefits of a high-level language with the ability to access the Intel386 and Intel486 architectures. For the development of systems software, PL/M-386 is a costeffective alternative to assembly language programming.

FORTRAN-386/486 COMPILER

Intel's FORTRAN-386 compiler is a cross-compiler that supports the entire Intel386 family of components and Intel486 microprocessors (when operating in the 386 chip mode) microprocessors.

FORTRAN-386 features high-level support for floating-point calculations, transcendentals, interrupt procedures, and run-time exception handling. FORTRAN-386 meets the ANSI FORTRAN-77 language subset specification and supports extensions endorsed by the Department of Defense (DOD), extensions that support programs written for the ANSI FORTRAN 66 standard, and extensions that support the Intel386 microprocessor and related numerics coprocessors.

To aid in the development and debugging process, the compiler generates warning and error messages and an optional listing file. The listing file can include symbol cross-reference tables and a listing of the generated Intel386 microprocessor assembly-language instructions. Library routines are reentrant and ROMable.

Other Intel FORTRAN-386 compiler features include:

- Object code can be configured to reside in either RAM or ROM
- The program code can be optimized for execution speed or memory size
- Source-level debugging is supported via the rich symbolics provided in the object module format (Intel OMF386)
- Support for the proposed REALMATH IEEE floating point standard

RLL-386/486 RELOCATION, LINKAGE, AND LIBRARY TOOLS

The RLL 386 relocation, linkage, and library tools feature comprehensive support of the full Intel386 and Intel486 architectures. The tools link separate modules, build object libraries, link in Intel387 support, build tasks to execute under protected mode, or multitasking, memory protected software. RLL-386 supports loadable, linkable, and bootloadable Intel object module formats; and supports all segmentation models. RLL-386 consists of the following:

Binder — for linking multiple object modules into a single program and resolving references between modules.

Builder — for producing absolute object modules, assigning addresses, and creating protected mode data structures.

Librarian — for creating and maintaining libraries of object modules.

EMUL-387, NUM-387 NUMERICS SUPPORT LIBRARIES

Intel's EMUL-387 and NUM-387 Numerics Libraries fully support the Intel387™, Intel 387 DX, Intel 387 SX math coprocessors and the Intel486 internal numerics unit—whether an actual math coprocessor is used in the final system or not.

For Intel386 microprocessor based applications without a math coprocessor, EMUL-387, a numerics software emulator, will execute instructions as though the coprocessor were present. Its functionality is identical to that of the math coprocessor. It is ideal for prototyping and debugging floating-point application software independent of hardware. Further, this permits portability of application code regardless of the presence of math coprocessor hardware in target systems.

For applications with a math coprocessor, NUM-387 numerics support library provides Intel's ASM 386, C-386, PL/M-386, and FORTRAN-386 language users with enhanced numeric data processing capability. With the library, it is easy for programs to do floating point arithmetic. Programmers can bind in library modules to do trigonometric, logarithmic and other numeric functions.

FEATURES

The user is guaranteed accurate, reliable results for all appropriate inputs.

Intel's NUM-387 support library is a collection of four functionally distinct libraries:

- Common elementary function library routines perform algebraic, logarithmic, exponential, trigonometric, and hyperbolic operations on real and complex numbers, as well as real-to-integer conversions; the routines extend the ranges of the coprocessor instructions
- Initialization library routines set up the numerics processing environment for the Intel386 family of processors with an Intel387, DX, or SX or true software emulator
- Decimal conversion library routines convert floating-point numbers from one Intel387, DX, or SX binary storage format to another, or from ASCII decimal strings to Intel387, DX, or SX binary floating-point format and vice versa
- Exception handling library routines make writing numerics exception handlers easier

All support library modules are in Intel386 microprocessor object module format (Intel OMF-386) so they can be linked with the object output of any Intel language. All routines are reentrant and ROMable.

By using Intel's NUM-387, the user is guaranteed that the numeric software meets industry standard (ANSI/IEEE standard for binary floating point arithmetic, 754-1985) and is portable, thus maintaining software investment.

DB-386 Software Debugger

Intel's DB-386 is a PC-based software development environment with source-level symbolic debug capabilities for object modules produced by Intel's assembler and high-level language compilers. This software debug environment allows Intel386 microprocessor code to be executed and debugged directly on a Intel386 DX or Intel386 SX microprocessor based PC, without any additional target hardware required. With Intel's standard windowed human interface, users can focus their efforts on finding bugs rather than spending time learning and manipulating the debug environment.

Other Intel DB-386 features include:

- A run-time interface allows protected-mode Intel386 microprocessor programs to be executed directly on a Intel386 DX or Intel386 SX microprocessor based PC
- 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
- Watch windows (which display user-specified variables), trace points, and breakpoints (including fixed, temporary, and conditional) can be set and modified as needed
- The user can browse source and callstacks, observe processor registers, and access watch window variables by either pull down menus or by a single keystroke, using function keys
- The user need not know whether a variable is an unsigned integer, a real, or a structure—the debugger uses the wealth of typing information available in Intel languages to display program variables in their respective type formats
- DB-386 supports the Intel486 microprocessor when operated in the Intel386 microprocessor mode

Intel386 and Intel486 Family In-Circuit Tools

Intel in-circuit emulators are used in many different debug environments including the design and test of: PC BIOS software and motherboard hardware, Intel386 and Intel486 based single board computers, and application and operating system software for DOS-based, ROM-based, and UNIX-based systems.

The Intel386 and Intel486 In-Circuit Emulators (ICE™) take advantage of exclusive Intel technology to provide accurate emulation for Intel's 80386 SX, 80386 DX, 80376, and 80486 microprocessors. Special access to internal processor states provides information not available to emulators which simply monitor the external buses. Emulators which do not have access to the internal processor conditions cannot guarantee accurate display of instructions executed by the microprocessor. With an Intel In-circuit Emulator you can be certain that the emulator is displaying accurate execution history, even when executing code from the on-chip cache memory of the Intel486.

The DOS hosted Intel386 DX and Intel386 SX emulators feature a windowed, menu-driven, human interface which provides easy access to the powerful features of these emulators. This makes it easy for novice or infrequent users to get the most out of every debug session. This interface features multiple windows which

allow you to simultaneously view source code, assembly code, memory, trace, variables, and registers. This interface is fully symbolic when used with Intel languages.

All of the emulators feature a combination of powerful and flexible breakpoints. The products use a combination of software breakpoints, hardware breakpoints, and on-chip debug registers to provide a rich set of recognition logic. Flexible breakpoints make it possible to set breakpoints on instruction execution and/or any possible bus event.

Trace filtering provides the ability to select the information captured in the trace buffer. ICE-386 SX allows capture of solely bus cycle information or both bus cycle and execution information. In addition, the ICE-386 DX can filter wait-state information from the trace buffer. ICE-486 provides the most flexible trace collection by allowing capture of information by any combination of bus cycle type including filtering of wait states, by instructions only, or by both bus cycles and instructions.

Other features of Intel emulators include:

- Unparalleled support of the Intel386 and Intel486 architectures, notably the native protected mode
- Emulation at clock speeds to 33 MHz, and full featured trigger and trace capabilities
- The Intel386 family emulators are convertible using removable probes to support the 80386 DX and 80386 SX microprocessors. The Intel486 processor is also supported via a product upgrade.

Relocatable Expanded Memory

Designed to enhance your existing ICE-486 and the ICD-486 debugger (REM486 is included with ICE-486 and an option for ICD-486). This optional relocatable expansion memory board adds 2 Mbyte of memory which the ICE or ICD can use in place of memory on the user target board.

ONCE-386 and Transmuter Adapters

If you have a surface mount Intel386 SX microprocessor design using 100-pin PQFP parts, Intel ICE emulators have on-circuit emulation (ONCE) capability. With surface mounted components, the ICE-386 SX emulator cabling clamps over the part, tri-stating the component, and allowing the emulator to operate. This allows you to debug manufactured boards without resoldering. For early target load development, a transmuter adapter can be used. The transmuter provides a better connection technique for debugging systems where the adapter cable will have to be attached and removed many times (like in prototype development).

ICD-486 In-Circuit Debugger

The ICD-486 In-circuit Debugger provides a low-cost alternative for full speed in-target Intel486 development. ICD-486 implements a subset of ICE functionality including: symbolic debugging, debug of high-speed cached applications, software and debug register breakpoints, and in-circuit operation.

Worldwide Service, Support, and Training

To augment its developing tools, Intel offers field application engineering expertise, hotline technical support, and on-site service.

Intel also offers Software Support which includes technical software information, automatic distributions of software and documentation updates, *iCOMMENTS* publication, remote diagnostic software, and development tools troubleshooting guide.

Intel's 90-day Hardware Support package includes technical hardware information, telephone support, warranty on parts, labor, material, and on-site hardware support.

Intel Development Tools also offers a 30-day, money-back guarantee to customers who are not satisfied after purchasing any Intel development tool.

FEATURES

PRODUCT SUPPORT MATRIX

Product	Component			Host
	i486	386 DX	386 SX	DOS 3.x and 5.0
ASM-386 Macro Assembler	✓	✓	✓	✓
iC-386 Compiler	✓	✓	✓	✓
PL/M-386 Compiler	✓	✓	✓	✓
FORTRAN-386 Compiler	✓	✓	✓	✓
RLL-386 Relocation, Linkage, Library, Support Tools	✓	✓	✓	✓
NUM-387 Libraries	✓	✓	✓	✓
EMUL-387 Libraries	NA	✓	✓	✓
In-Circuit Emulators	✓	✓	✓	✓
In-Circuit Debugger	✓			✓
DB-386 Software Debugger	✓	✓	✓	✓

ORDERING INFORMATION

386/i486™ FAMILY DOS HOSTED DEVELOPMENT KIT ORDER CODES

Software Order Codes

All software supports 386 and 486 microprocessor families except where indicated.

DKIT386C Compiler Software Development Kit (See following content list).

D86ASM386NL ASM macro assembler for PC-DOS systems.

D86C386NL DOS resident, ANSI standard (ANS X3.159-1989) C compiler.

D86PLM386NL DOS resident PL/M compiler.

D86FOR386NL DOS resident Fortan Compiler.

D86RLL386NL DOS resident software development package. Contains Binder (for linking separately compiled modules), a Builder (for configuring protected multi-tasking systems), a cross reference Mapper, and a Librarian. Use this tool in conjunction with Intel's 80386 compilers and macro assembler.

DB386 DOS S/W debugger.

The Intel Basic Software Development Kit for the DOS hosted environment includes:

iC386 compiler
ASM386 assembler
RLL386 relocation linker and locator
NUM387 numerics library
EMUL387 math coprocessor emulator library
DB386 software debugger
OMF386LOAD loader development object module format documentation

ORDERING INFORMATION

IN-CIRCUIT TOOL ORDER CODES

All In-circuit emulator codes include: control unit, power supply, processor module, Stand-Alone Self Test board, bus Isolation Board, and DOS host software and serial interface cable.

ICE386SX25V	ICE-386 SX In-circuit emulator for the Intel386 SX component to 25 MHz.
pICE386SX20D	ICE-386 SX In-circuit emulator for the 80386 SX component to 20 MHz.
pICE386DX25DZ	ICE-386 DX In-circuit emulator for the 80386 DX component to 25 MHz.
ICE386DX33D	ICE-386 DX In-circuit emulator for the 80386 DX component to 33 MHz.
ICD48650D	In-circuit debugger for the 80486 microprocessor to 50 MHz.
pICE48633DZ	ICE-486 In-circuit emulator for the 80486 component to 33 MHz.

ICE CONVERSION KITS

KBASECONC	Converts ICE-486 to ICE-376, ICE-386 SX, or ICE-386 DX.
KBASECONV	Converts ICE-386 SX or ICE-386 DX to ICE-486.
TOICE386SX20D	Converts ICE-386 DX to ICE-386 SX 20 MHz.
TOICE386DX25D	Converts ICE-386 SX to ICE-386 DX 20 MHz.
TOICE48633D	Converts ICE-386 SX or ICE-386 DX to ICE-486 33 MHz.

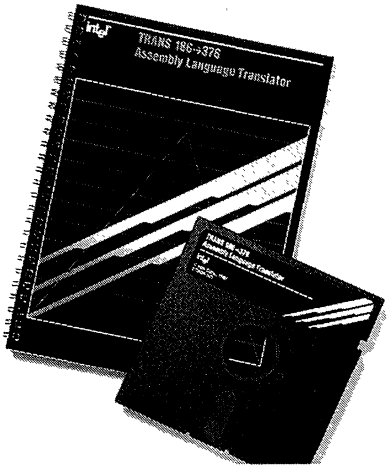
ADDITIONAL TOOL ORDER CODES

386SXONCE Kit	100 pin PQFP to 132 pin PGA adaptor kit.
REM486A	2 Mbyte relocatable expansion memory option for ICD-486 (included with ICE-486).

To order your Intel Development Tool product, for more information, or for the number of your nearest sales office or distributor, call 800-874-6835 (North America). For literature on other Intel products call 800-548-4725 (North America). Outside of North America, please contact your local Intel sales office or distributor for more information.

intel®

TRANS 186 → 376 ASSEMBLY CODE TRANSLATOR



270919-1

To Order TRANS 186 → 376 Software, contact your local Intel sales office

2

TRANS 186 → 376 PRESERVES YOUR PROGRAMMING INVESTMENT

When your embedded application outgrows the 80C186 family, TRANS 186 → 376 is ready to help you upgrade to the 376 Embedded Processor. TRANS 186 → 376 is a DOS-based tool to automate the translation of Intel ASM86 source code to ASM386 source code. This program can actually help protect the man-years of investment in your original 86 software.

TRANS 186 → 376 LOWERS THE 32-BIT BARRIER

TRANS 186 → 376 accepts 16-bit source code written for any member of the 8086/8088 and 80C186/80C188 families. The output source code, with its 32-bit offsets, is suitable for Protected Mode execution on the 376 Embedded Processor or any 386, 386SX, or 486 microprocessor. The time you save by recycling your software can be applied toward system enhancements.

You control TRANS 186 → 376 operation from either the DOS command line or a control file. Major control switches cover:

- Choice of FLAT model or LARGE16 memory environment
- Redefinition of segments
- Optional 32-bit data declaration

TRANS 186 → 376 translates your routines on a line-by-line basis, converting as much code as possible. Whenever the tool does not have enough information to make conversions, it highlights the code section with messages, alerting you to edit by hand. TRANS 186 → 376 can write the ASM86 source code as comments in the ASM386 source file for side-by-side comparison.

* PC AT and PC-DOS are trademarks of IBM.
**MS-DOS is a trademark of Microsoft Corporation.

November 1990
Order Number: 270919-001

intel

TRANS 186 → 376 ASSEMBLY CODE TRANSLATOR

TRANS 186 → 376 COMPLEMENTS OTHER DEVELOPMENT TOOLS

Upon request, TRANS 186 → 376 generates a build file for the Intel System Builder, BLD386. This allows you to get your software running with only minimal BLD386 experience. A 72-page manual accompanies the TRANS 186 → 376 tool. The manual coverage includes:

- Practical tips on the overall conversion process
- Initializing the CPU and generating Protected Mode data structures
- Producing code for emulators and debuggers

Your 80C186 experience can release the power of the 376 Embedded Processor with the TRANS 186 → 376 Assembly Language Translator as your partner.

System requirements: PC AT* or compatible computer with PC-DOS* or MS-DOS** operating system version 3.0 or later, hard disk, and 512K RAM.

intel®

3

Embedded Intel386™ Processors

Intel386™ CXSA EMBEDDED MICROPROCESSOR

- Static Intel386™ CPU Core
  - Low Power Consumption
  - Operating Power Supply
    4.5V to 5.5V—25 MHz and 33 MHz
    4.75V to 5.25V—40 MHz
  - Operating Frequency
    SA-40 = 40 MHz
    SA-33 = 33 MHz
    SA-25 = 25 MHz
- Transparent Power-Management System Architecture
  - Intel System Management Mode Architecture Extension for Truly Compatible Systems
  - Power Management Transparent to Operating Systems and Application Programs
  - Programmable Power-Management Modes
- Clock Freeze Mode Allows Clock Stopping at Any Time
- Full 32-bit Internal Architecture
  - 8-, 16-, 32-bit Data Types
  - 8 General Purpose 32-bit Registers
- Runs Intel386 Architecture Software in a Cost-Effective, 16-bit Hardware Environment
  - Runs Same Applications and Operating Systems as the Intel386 SX and Intel386 DX Processors
  - Object Code Compatible with 8086, 80186, 80286, and Intel386 Processors
- High-Performance 16-bit Data Bus
  - Two-Clock Bus Cycles
  - Address Pipelining Allows Use of Slower, Inexpensive Memories
- Integrated Memory Management Unit (MMU)
  - Virtual Memory Support
  - Optional On-Chip Paging
  - 4 Levels of Hardware-Enforced Protection
  - MMU Fully Compatible with Those of the 80286 and Intel386 DX Processors
- Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System
- Large, Uniform Address Space
  - 64 Megabyte Physical
  - 64 Terabyte Virtual
  - 4 Gigabyte Maximum Segment Size
- Numerics Support with Intel387™ SX and Intel387 SL Math Coprocessors
- On-Chip Debugging Support Including Breakpoint Registers
- Complete System Development Support
- High-Speed CHMOS Technology
- 100-pin Plastic Quad Flatpack Package

3

The Intel386™ CXSA embedded microprocessor is a 5-volt, 32-bit, fully static CPU with a 16-bit external data bus, a 26-bit external address bus, and Intel's System Management Mode (SMM). The Intel386 CXSA CPU brings the vast software library of the Intel386 architecture to embedded systems. It provides the performance benefits of 32-bit programming with the cost savings associated with 16-bit hardware systems.

The Intel386 CXSA microprocessor is manufactured on Intel's 0.8-micron CHMOS V process. This process provides high performance and low power consumption for power-sensitive applications. Figure 3 and Figure 4 illustrate the flexibility of low power devices with respect to temperature and frequency relationships.

December 1994
Order Number: 272418-003

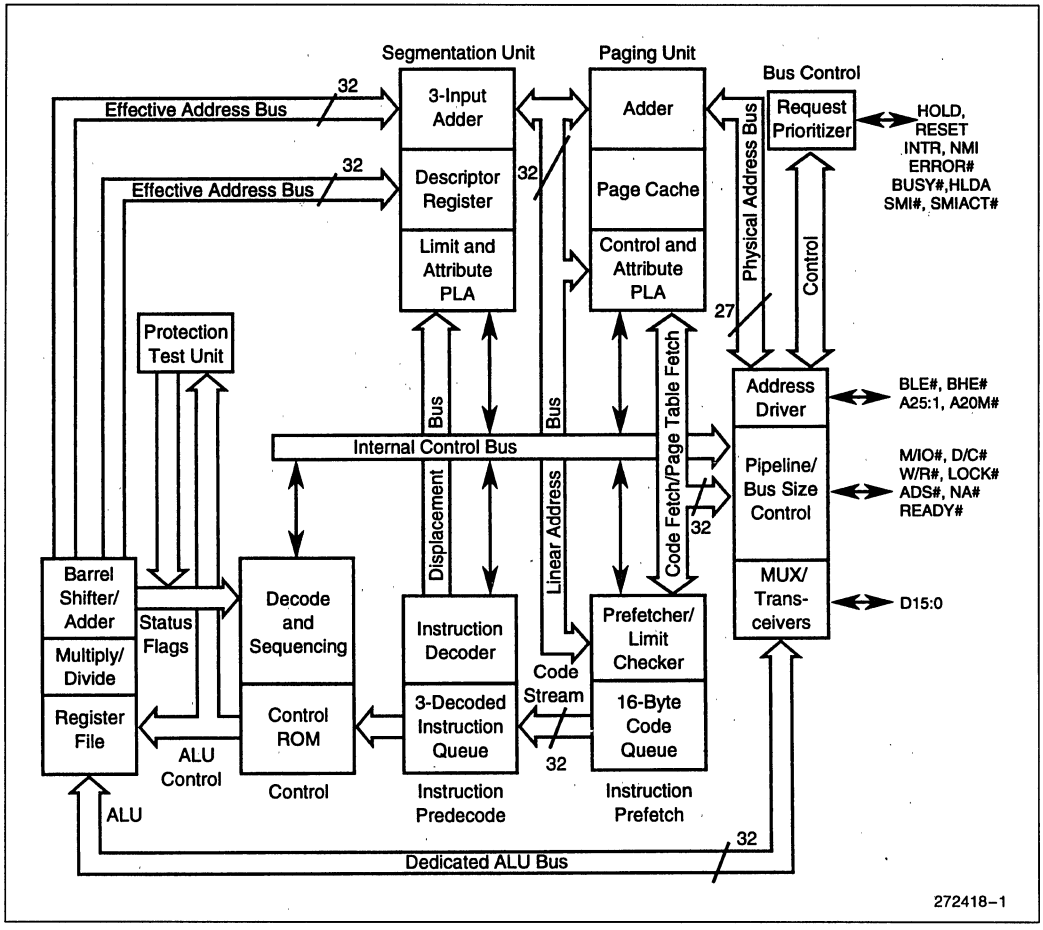
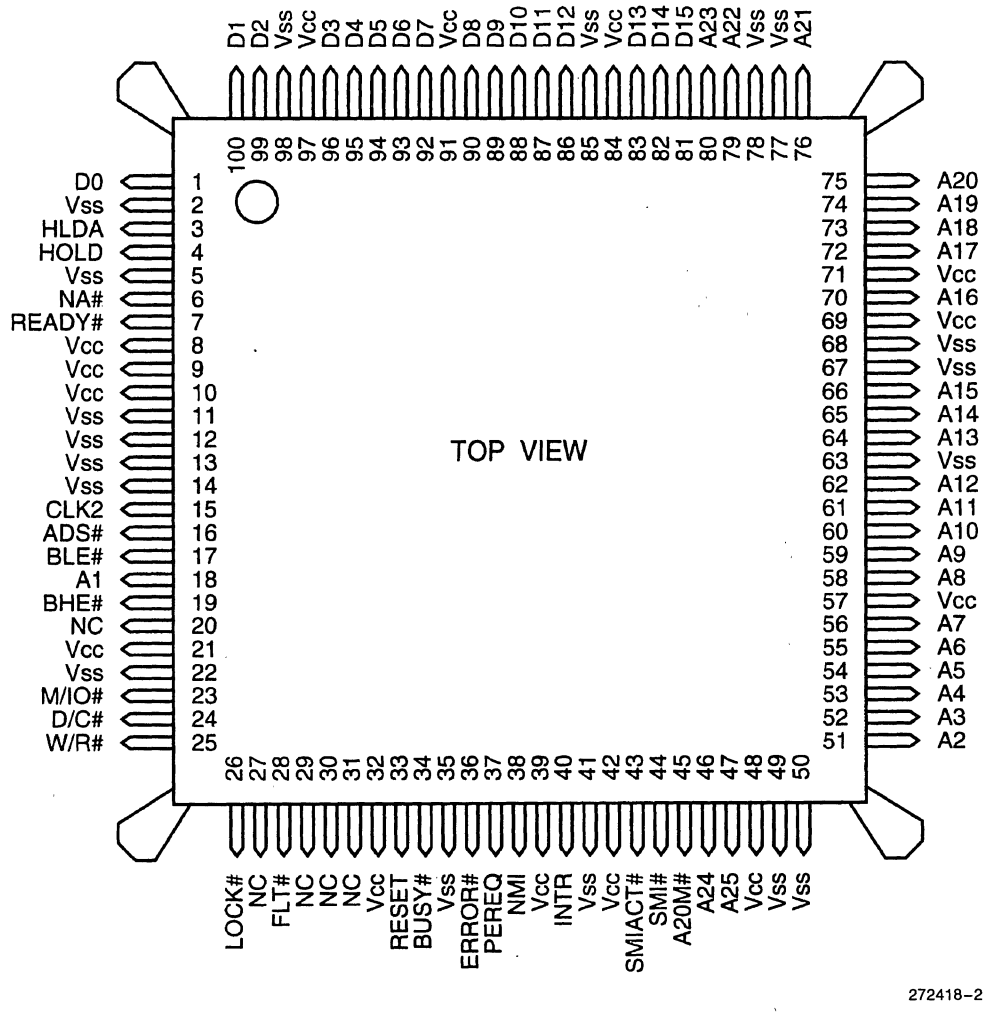


Figure 1. Intel386™ CXSA Microprocessor Block Diagram

1.0 PIN ASSIGNMENT



NOTE:
NC = No Connection

Figure 2. Intel386™ CXSA Microprocessor Pin Assignment (PQFP)

Table 1. Pin Assignment

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	D0	26	LOCK#	51	A2	76	A21
2	V_{SS}	27	NC	52	A3	77	V_{SS}
3	HLDA	28	FLT#	53	A4	78	V_{SS}
4	HOLD	29	NC	54	A5	79	A22
5	V_{SS}	30	NC	55	A6	80	A23
6	NA#	31	NC	56	A7	81	D15
7	READY#	32	V_{CC}	57	V_{CC}	82	D14
8	V_{CC}	33	RESET	58	A8	83	D13
9	V_{CC}	34	BUSY#	59	A9	84	V_{CC}
10	V_{CC}	35	V_{SS}	60	A10	85	V_{SS}
11	V_{SS}	36	ERROR#	61	A11	86	D12
12	V_{SS}	37	PEREQ	62	A12	87	D11
13	V_{SS}	38	NMI	63	V_{SS}	88	D10
14	V_{SS}	39	V_{CC}	64	A13	89	D9
15	CLK2	40	INTR	65	A14	90	D8
16	ADS#	41	V_{SS}	66	A15	91	V_{CC}
17	BLE#	42	V_{CC}	67	V_{SS}	92	D7
18	A1	43	SMIACT#	68	V_{SS}	93	D6
19	BHE#	44	SMI#	69	V_{CC}	94	D5
20	NC	45	A20M#	70	A16	95	D4
21	V_{CC}	46	A24	71	V_{CC}	96	D3
22	V_{SS}	47	A25	72	A17	97	V_{CC}
23	M/IO#	48	V_{CC}	73	A18	98	V_{SS}
24	D/C#	49	V_{SS}	74	A19	99	D2
25	W/R#	50	V_{SS}	75	A20	100	D1

2.0 PIN DESCRIPTIONS

Table 2 lists the Intel386 CXSA microprocessor pin descriptions. The following definitions are used in the pin descriptions:

- # The named signal is active low.
- I Input signal.
- O Output signal.
- I/O Input and output signal.
- P Power pin.
- G Ground pin.

Table 2. Pin Descriptions

Symbol	Type	Pin	Name and Function
A20M# (Note 1)	I	45	**ADDRESS 20 MASK** controls the A20 address signal. When A20M# is low, the CPU masks off (forces low) the internal A20 physical address signal. This enables the CPU to run software that was developed using the 8086 address "wraparound" techniques. When A20M# is high, A20 is available on the address bus. While the bus is floating, A20M# has no effect on the A20 address signal. A20M# should be deasserted during SMM if the SMM handler accesses more than 1 Mbyte of memory.
A25:1 (Note 2)	O	47–46, 80–79, 76–72, 70, 66–64, 62–58, 56–51, 18	**ADDRESS BUS** outputs physical memory or port I/O addresses.
ADS#	O	16	**ADDRESS STATUS** indicates that the processor is driving a valid bus-cycle definition and address onto its pins (W/R#, D/C#, M/IO#, BHE#, BLE#, and A25:1).
BHE#	O	19	**BYTE HIGH ENABLE** indicates that the processor is transferring a high data byte.
BLE#	O	17	**BYTE LOW ENABLE** indicates that the processor is transferring a low data byte.
BUSY#	I	34	**BUSY** indicates that the math coprocessor is busy.
CLK2	I	15	**CLK2** provides the fundamental timing for the device.
D/C#	O	24	**DATA/CONTROL** indicates whether the current bus cycle is a data cycle (memory or I/O) or a control cycle (interrupt acknowledge, halt, or code fetch). When D/C# is high, the bus cycle is a data cycle; when D/C# is low, the bus cycle is a control cycle.
D15:0	I/O	81–83, 86–90, 92–96, 99–100, 1	**DATA BUS** inputs data during memory read, I/O read, and interrupt acknowledge cycles and outputs data during memory and I/O write cycles.
ERROR#	I	36	**ERROR** indicates that the math coprocessor has an error condition.

NOTES:
1. This pin supports the additional features of the Intel386 CXSA microprocessor; it is not present on the Intel386 SXSA microprocessor.
2. The A25:24 pins support the additional features of the Intel386 CXSA microprocessor; they are not present on the Intel386 SXSA microprocessor.

Table 2. Pin Descriptions (Continued)

Symbol	Type	Pin	Name and Function
FLT#	I	28	**FLOAT** forces all bidirectional and output signals, including HLDA, to a high-impedance state.
HLDA	O	3	**BUS HOLD ACKNOWLEDGE** indicates that the CPU has surrendered control of its local bus to another bus master.
HOLD	I	4	**BUS HOLD REQUEST** allows another bus master to request control of the local bus.
INTR	I	40	**INTERRUPT REQUEST** is a maskable input that causes the CPU to suspend execution of the current program and then execute an interrupt acknowledge cycle.
LOCK#	O	26	**BUS LOCK** prevents other system bus masters from gaining control of the system bus while it is active (low).
M/IO#	O	23	**MEMORY/IO** indicates whether the current bus cycle is a memory cycle or an input/output cycle. When M/IO# is high, the bus cycle is a memory cycle; when M/IO# is low, the bus cycle is an I/O cycle.
NA#	I	6	**NEXT ADDRESS** requests address pipelining.
NC		20, 27, 29–31	**NO CONNECTION** should always be left unconnected. Connecting a NC pin may cause the processor to malfunction or cause your application to be incompatible with future steppings of the device.
NMI	I	38	**NONMASKABLE INTERRUPT REQUEST** is a nonmaskable input that causes the CPU to suspend execution of the current program and execute an interrupt acknowledge function.
PEREQ	I	37	**PROCESSOR EXTENSION REQUEST** indicates that the math coprocessor has data to transfer to the processor.
READY#	I	7	**BUS READY** indicates that the current bus cycle is finished and the external device is ready to accept more data from the processor.
RESET	I	33	**RESET** suspends any operation in progress and places the processor into a known reset state.
SMI# (Note 1)	I	44	**SYSTEM MANAGEMENT INTERRUPT** invokes System Management Mode (SMM). SMI# is the highest priority interrupt. It is latched on its falling edge and it forces the CPU into SMM upon completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# cannot interrupt LOCKed bus cycles or a currently executing SMM. If the processor receives a second SMI# while it is in SMM, it will latch the second SMI# on the SMI# falling edge. However, the processor must exit SMM by executing a Resume instruction (RSM) before it can service the second SMI#.

NOTES:
1. This pin supports the additional features of the Intel386 CXSA microprocessor; it is not present on the Intel386 SXSA microprocessor.
2. The A25:24 pins support the additional features of the Intel386 CXSA microprocessor; they are not present on the Intel386 SXSA microprocessor.

Table 2. Pin Descriptions (Continued)

Symbol	Type	Pin	Name and Function
SMIACT# (Note 1)	O	43	**SYSTEM MANAGEMENT INTERRUPT ACTIVE** indicates that the processor is operating in System Management Mode (SMM). It is asserted when the processor initiates an SMM sequence and remains asserted (low) until the processor executes the Resume instruction (RSM).
W/R#	O	25	**WRITE/READ** indicates whether the current bus cycle is a write cycle or a read cycle. When W/R# is high, the bus cycle is a write cycle; when W/R# is low, it is a read cycle.
V_{CC}	P	8–10, 21, 32, 39, 42, 48, 57, 69, 71, 84, 91, 97	**SYSTEM POWER** provides the nominal DC supply input.
V_{SS}	G	2, 5, 11–14, 22, 35, 41, 49–50, 63, 67–68, 77–78, 85, 98	**SYSTEM GROUND** provides the 0V connection from which all inputs and outputs are measured.

NOTES:
1. This pin supports the additional features of the Intel386 CXSA microprocessor; it is not present on the Intel386 SXSA microprocessor.
2. The A25:24 pins support the additional features of the Intel386 CXSA microprocessor; they are not present on the Intel386 SXSA microprocessor.

3.0 DESIGN CONSIDERATIONS

This section describes the Intel386 CXSA microprocessor instruction set, component and revision identifier, and package thermal specifications.

3.1 Instruction Set

The Intel386 CXSA microprocessor uses the same instruction set as the Intel386 SX microprocessor with the following exceptions.

The Intel386 CXSA microprocessor has one new instruction (RSM). This Resume instruction causes the processor to exit System Management Mode (SMM). RSM requires 338 clocks for execution.

The Intel386 CXSA microprocessor requires more clock cycles than the Intel386 SX microprocessor to execute some instructions. Table 4 lists these instructions and the Intel386 CXSA microprocessor execution times. For the equivalent Intel386 SX microprocessor execution times, refer to the "Instruction Set Clock Count Summary" table in the *Intel386™ SX Microprocessor* datasheet (order number 240187).

3.2 Component and Revision Identifier

To assist users, the microprocessor holds a component identifier and revision identifier in its DX register after reset. The upper 8 bits of DX hold the component identifier, 23H. (The lower nibble, 3H, identifies the Intel386 architecture, while the upper nibble, 2H, identifies the second member of the Intel386 microprocessor family.)

The lower 8 bits of DX hold the revision level identifier. The revision identifier will, in general, chronologically track those component steppings that are intended to have certain improvements or distinction from previous steppings. The revision identifier will track that of the Intel386 CPU whenever possible. However, the revision identifier value is not guaranteed to change with every stepping revision or to follow a completely uniform numerical sequence, depending on the type or intent of the revision or the manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component. The initial revision identifier for the Intel386 CXSA microprocessor is 09H.

3.3 Package Thermal Specifications

The Intel386 CXSA microprocessor is specified for operation with case temperature (T_{CASE}) as specified in the "DC SPECIFICATIONS" on page 10. The case temperature can be measured in any environment to determine whether the microprocessor is within the specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

An increase in the ambient temperature (T_A) causes a proportional increase in the case temperature (T_{CASE}) and the junction temperature (T_J). See Figure 3 and Figure 4 for case and ambient temperature relationships to frequency. A packaged device produces thermal resistance between junction and case temperatures (θ_{JC}) and between junction and ambient temperatures (θ_{JA}). The relationships between the temperature and thermal resistance parameters are expressed by these equations (P = power dissipated as heat = $V_{CC} \times I_{CC}$):

1. $T_J = T_{CASE} + P \times \theta_{JC}$
2. $T_A = T_J - P \times \theta_{JA}$
3. $T_{CASE} = T_A + P \times [\theta_{JA} - \theta_{JC}]$

A safe operating temperature can be calculated from equation 1 by using the maximum safe T_J of 115°C, the maximum power drawn by the chip in the specific design, and the θ_{JC} value from Table 3. The θ_{JA} value depends on the airflow (measured at the top of the chip) provided by the system ventilation. The θ_{JA} values are given for reference only and are not guaranteed.

Table 3. Thermal Resistances (0°C/W) θ_{JA}, θ_{JC}

Pkg	θ_{JC}	θ_{JA} vs Airflow (ft/min)		
		0	100	200
100 PQFP	5.1	46.0	44.8	41.2

Table 4. Intel386™ CXSA Microprocessor Instruction Execution Times (in Clock Counts)

Instruction	Clock Count		
	Virtual 8086 Mode (Note 1)	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode (Note 3)
POPA		28	35
IN:			
Fixed Port	27	14	7/29
Variable Port	28	15	8/29
OUT:			
Fixed Port	27	14	7/29
Variable Port	28	15	9/29
INS	30	17	9/32
OUTS	31	18	10/33
REP INS	31+6n (Note 2)	17+6n (Note 2)	10+6n/32+6n (Note 2)
REP OUTS	30+8n (Note 2)	16+8n (Note 2)	10+8n/31+8n (Note 2)
HLT		7	7
MOV C0, reg		10	10

NOTES:
1. The clock count values in this column apply if I/O permission allows I/O to the port in virtual 8086 mode. If the I/O bit map denies permission, exception fault 13 occurs; see clock counts for the INT 3 instruction in the "Instruction Set Clock Count Summary" table in the *Intel386™ SX Microprocessor* datasheet (order number 240187).
2. n = the number of times repeated.
3. When two clock counts are listed, the smaller value refers to a register operand and the larger value refers to a memory operand.

4.0 DC SPECIFICATIONS

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

ABSOLUTE MAXIMUM RATINGS*

Storage Temperature ..... −65°C to +150°C
Case Temperature under Bias ... −65°C to +112°C
Supply Voltage
with Respect to V_{SS} ..... −0.5V to +6.5V
Voltage on Other Pins ..... −0.5V to V_{CC} + 0.5V

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

OPERATING CONDITIONS*

V_{CC} (Digital Supply Voltage—
25 MHz and 33 MHz) .....4.5V to 5.5V

V_{CC} (Digital Supply Voltage—
40 MHz) .....4.75V to 5.25V

T_{CASE} Minimum (Case Temperature
under Bias) .....0°C

T_{CASE} Maximum .....see Figure 4

Operating Frequency .....0 MHz to 40 MHz

Table 5. DC Characteristics

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	+0.8	V	
V_{IHC}	CLK2 Input High Voltage	V_{CC}−0.8	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 5 mA
V_{OH}	Output High Voltage	2.4 V_{CC}−0.5		V V	I_{OH} = −1 mA I_{OH} = −0.2 mA
I_{LI}	Input Leakage Current (for all pins except PEREQ, BUSY#, FLT#, ERROR#, A20M#, SMI#)		±15	μA	0≤ V_{IN}≤ V_{CC}
I_{IH}	Input Leakage Current (PEREQ)		150	μA	V_{IH} = 2.4V (Note 1)
I_{IL}	Input Leakage Current (BUSY#, FLT#, ERROR#, A20M#, and SMI#)		−120	μA	V_{IL} = 0.45V (Note 2)
I_{LO}	Output Leakage Current		±15	μA	0.45V≤ V_{OUT}≤ V_{CC}

NOTES:
1. PEREQ input has an internal weak pull-down resistor.
2. BUSY#, FLT#, SMI#, A20M# and ERROR# inputs each have an internal weak pull-up resistor.
3. I_{CC} max measurement at worst-case frequency, V_{CC}, and temperature with reset active.
4. I_{CC} typical and I_{CCF} typical are measured at nominal V_{CC} and are not fully tested.
5. Not fully tested.

Table 5. DC Characteristics (Continued)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
I_{CC}	Supply Current				(Notes 3, 4)
	CLK2 = 80 MHz, CLK = 40 MHz		275	mA	Typical = 200 mA
	CLK2 = 66 MHz, CLK = 33 MHz		225	mA	Typical = 175 mA
	CLK2 = 50 MHz, CLK = 25 MHz		175	mA	Typical = 140 mA
I_{CCF}	Standby Current (Freeze Mode)		150	μA	Typical = 10μA (Notes 3, 4)
C_{IN}	Input Capacitance		10	pF	F_C = 1 MHz (Note 5)
C_{OUT}	Output or I/O Capacitance		12	pF	F_C = 1 MHz (Note 5)
C_{CLK}	CLK2 Capacitance		20	pF	F_C = 1 MHz (Note 5)

NOTES:
1. PEREQ input has an internal weak pull-down resistor.
2. BUSY#, FLT#, SMI#, A20M# and ERROR# inputs each have an internal weak pull-up resistor.
3. I_{CC} max measurement at worst-case frequency, V_{CC}, and temperature with reset active.
4. I_{CC} typical and I_{CCF} typical are measured at nominal V_{CC} and are not fully tested.
5. Not fully tested.

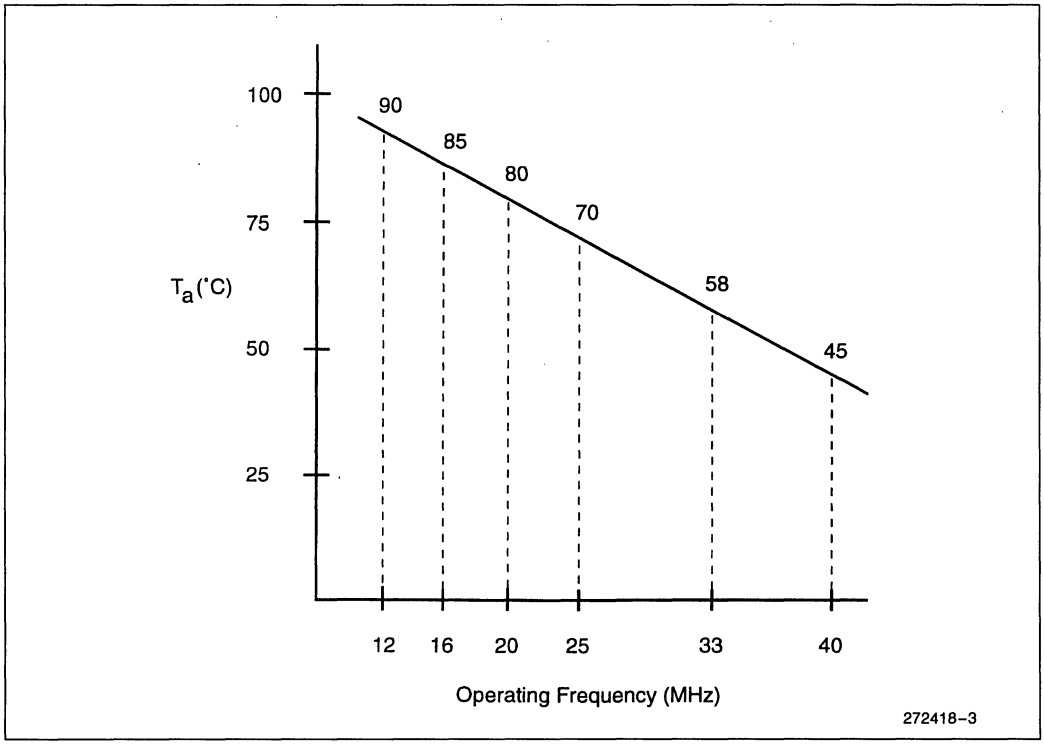


Figure 3. Ambient Temperature vs. Frequency at Zero Air Flow and T_J = 115°C

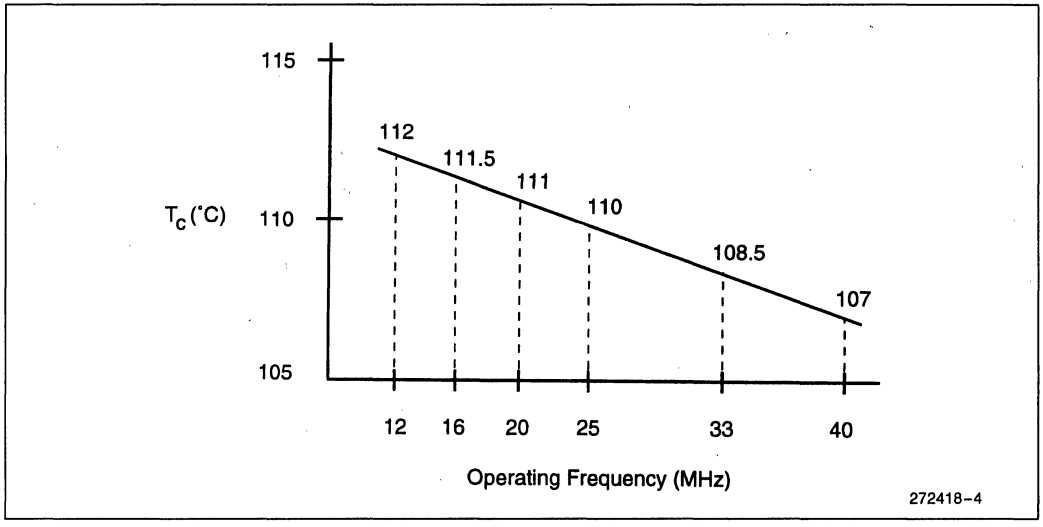


Figure 4. Case Temperature vs. Frequency at T_J = 115°C

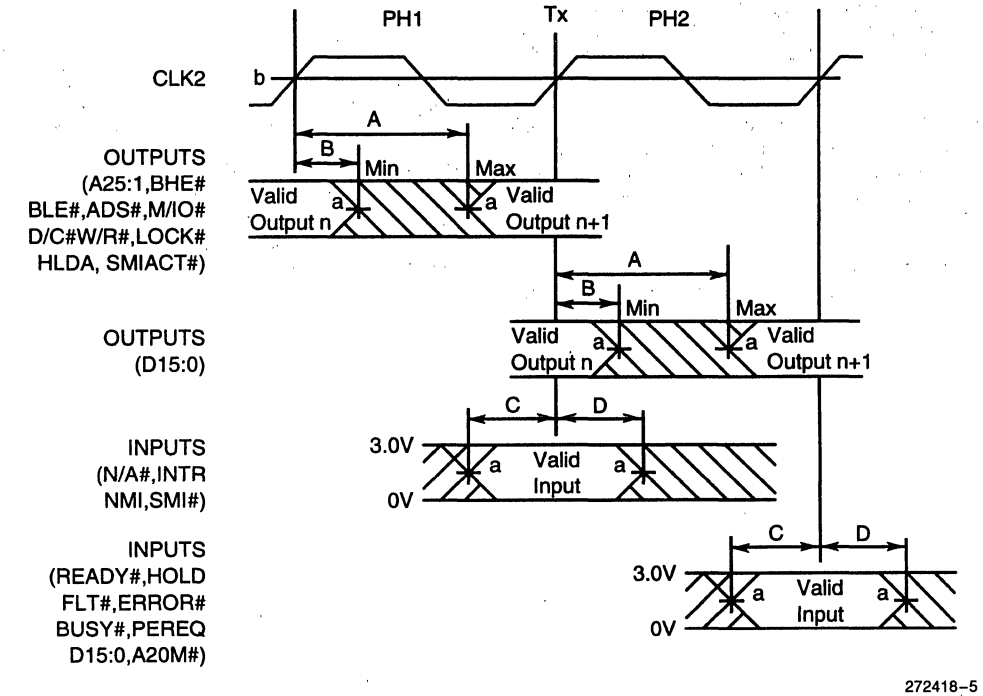
5.0 AC SPECIFICATIONS

Table 6 lists output delays, input setup requirements, and input hold requirements. All AC specifications are relative to the CLK2 rising edge crossing the 2.0V level.

Figure 5 shows the measurement points for AC specifications. Inputs must be driven to the indicated voltage levels when AC specifications are measured. Output delays are specified with minimum and maximum limits measured as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs ADS#, W/R#, D/C#, MI/O#, LOCK#, BHE#, BLE#, A25:1, SMIACT# and HLDA change only at the beginning of phase one. D15:0 (write cycles) change only at the beginning of phase two.

The READY#, HOLD, BUSY#, ERROR#, PEREQ, FLT#, A20M# and D15:0 (read cycles) inputs are sampled at the beginning of phase one. The NA#, INTR, SMI# and NMI inputs are sampled at the beginning of phase two.



LEGEND
a - 1.5V
b - 2.0V
A - Maximum Output Delay Spec
B - Minimum Output Delay Spec
C - Minimum Input Setup Spec
D - Minimum Input Hold Spec

Figure 5. Drive Levels and Measurement Points for AC Specifications

Table 6. AC Characteristics

Symbol	Parameter	40 MHz		33 MHz		25 MHz		Test Condition
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)	
	Operating Frequency	0	40	0	33	0	25	MHz (Note 1)
t1	CLK2 Period	12.5		15		20		
t2a	CLK2 High Time	4.5		6.25		7		(Note 2)
t2b	CLK2 High Time	3.5		4		4		(Note 2)
t3a	CLK2 Low Time	4.5		6.25		7		(Note 2)
t3b	CLK2 Low Time	3.5		4.5		5		(Note 2)
t4	CLK2 Fall Time		4		4		7	(Note 2)
t5	CLK2 Rise Time		4		4		7	(Note 2)
t6	A25:1 Valid Delay	4	13	4	15	4	17	C_L = 50 pF
t7	A25:1 Float Delay	4	20	4	20	4	30	(Note 3)
t8	BHE#, BLE#, LOCK# Valid Delay	4	13	4	15	4	17	C_L = 50 pF
t8a	SMIACT# Valid Delay	4	13	4	15	4	17	C_L = 50 pF
t9	BHE#, BLE#, LOCK# Float Delay	4	20	4	20	4	30	(Note 3)
t10	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	13	4	15	4	17	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	20	4	20	4	30	(Note 3)
t12	D15:0 Write Data Valid Delay	7	18	7	23	7	23	C_L = 50 pF
t12a	D15:0 Write Data Hold Time	2		2		2		C_L = 50 pF
t13	D15:0 Write Data Float Delay	4	17	4	17	4	22	(Note 3)
t14	HLDA Valid Delay	4	17	4	20	4	22	C_L = 50 pF
t15	NA# Setup Time	5		5		5		
t16	NA# Hold Time	2		2		3		
t19	READY#, A20M# Setup Time	7		7		9		

NOTES:
1. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
2. These are not tested. They are guaranteed by characterization.
3. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
4. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.

Table 6. AC Characteristics (Continued)

Symbol	Parameter	40 MHz		33 MHz		25 MHz		Test Condition
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)	
t20	READY#, A20M# Hold Time	4		4		4		
t21	D15:0 Read Setup Time	4		5		7		
t22	D15:0 Read Hold Time	3		3		5		
t23	HOLD Setup Time	4		9		9		
t24	HOLD Hold Time	2		2		3		
t25	RESET Setup Time	4		5		8		
t26	RESET Hold Time	2		2		3		
t27	NMI, INTR Setup Time	5		5		6		(Note 4)
t27a	SMI# Setup Time	5		5		6		(Note 4)
t28	NMI, INTR Hold Time	5		5		6		(Note 4)
t28a	SMI# Hold Time	5		5		6		(Note 4)
t29	PEREQ, ERROR#, BUSY#, FLT# Setup Time	5		5		6		(Note 4)
t30	PEREQ, ERROR#, BUSY#, FLT# Hold Time	4		4		5		(Note 4)

NOTES:
1. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
2. These are not tested. They are guaranteed by characterization.
3. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
4. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.

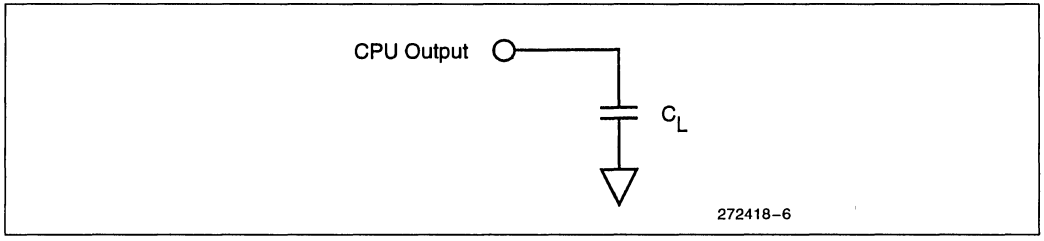


Figure 6. AC Test Loads

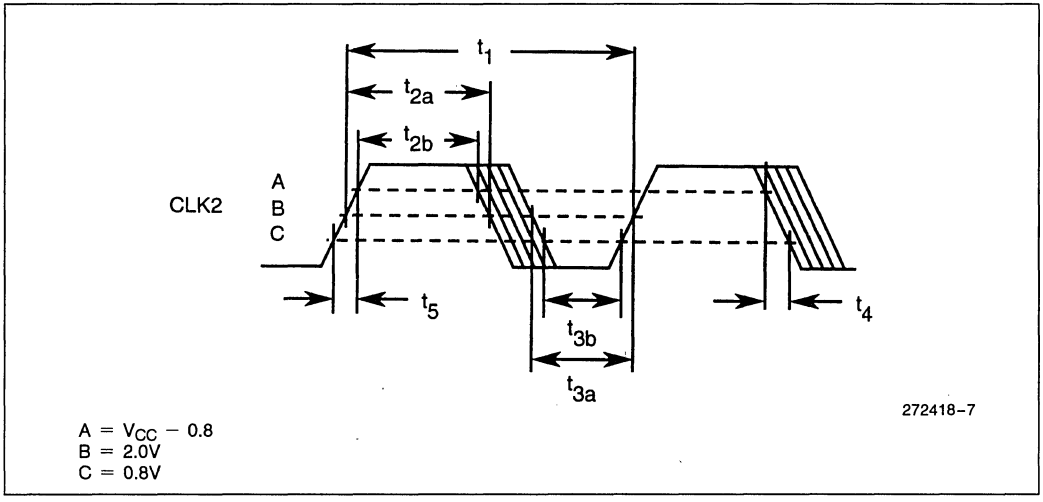


Figure 7. CLK2 Waveform

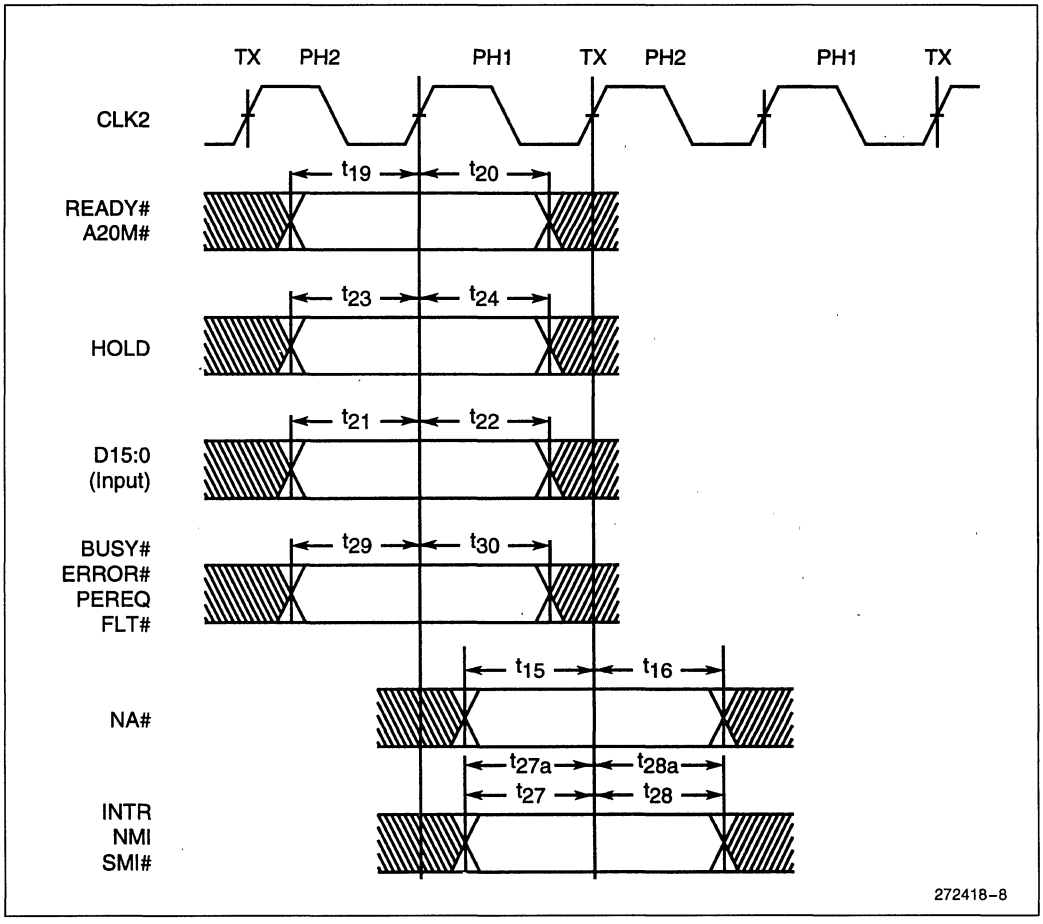


Figure 8. AC Timing Waveforms—Input Setup and Hold Timing

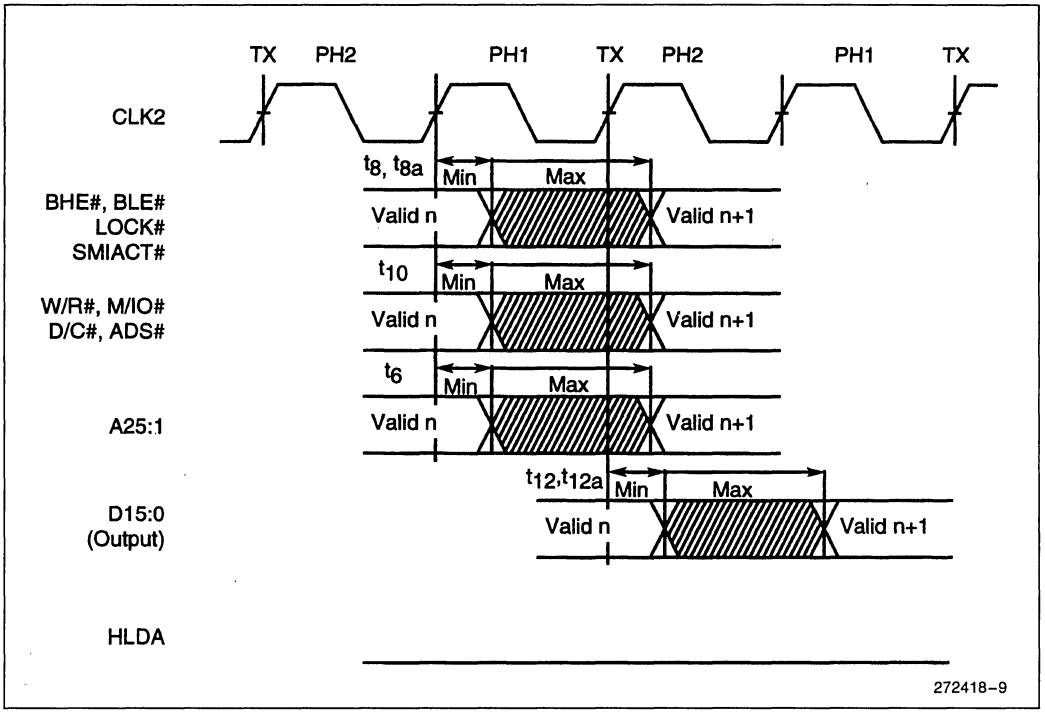


Figure 9. AC Timing Waveforms—Output Valid Delay Timing

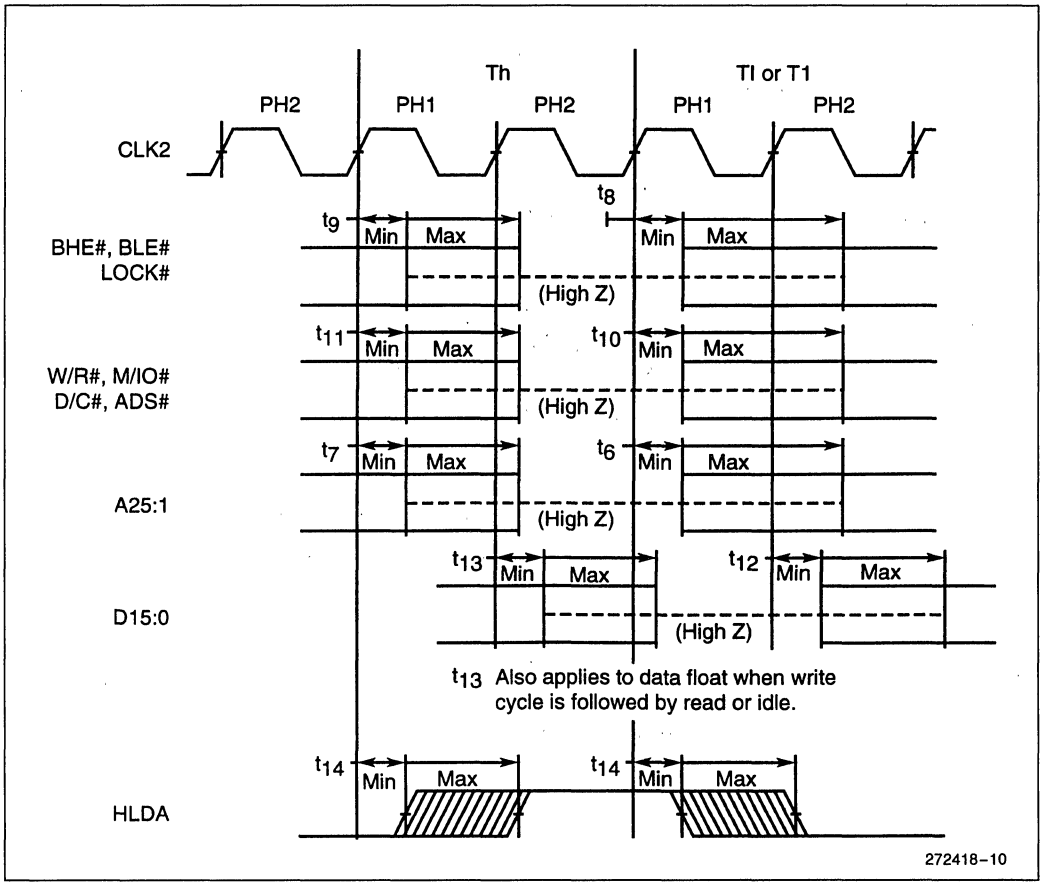


Figure 10. AC Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing

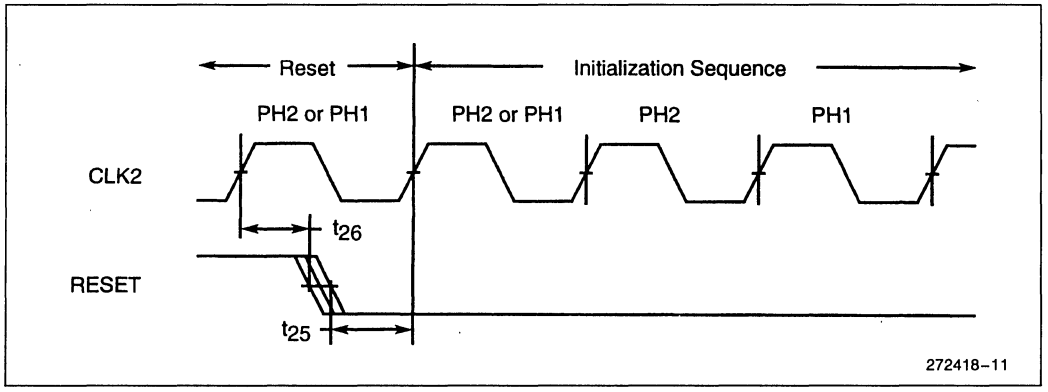


Figure 11. AC Timing Waveforms—RESET Setup and Hold Timing and Internal Phase

6.0 REVISION HISTORY

This -003 data sheet contains the following changes from the -002 version.

- Changed V_{CC} at 40 MHz to 4.75V to 5.25V (Pages 1 and 10)
- Renamed "Powerdown Mode" to "Clock Freeze Mode" on page 1.
- Added Clarifications to Figure 1.
- Corrected pin numbering for A25:1 in Table 2.
- Changed section 3.3 first sentence from "... on page 12" to "... on page 10".
- Changed note 2 on page 10 to reflect A20M#
- Changed first sentence on page 13 from "Table 7 lists ..." to "Table 6 lists ...".
- Added INTR and A20M# to Figure 8 and removed BS8#.

Intel386™ CXSA EMBEDDED MICROPROCESSOR

SmartDie Product Specification

- **Static Intel386™ CPU Core**
  - **Low Power Consumption**
  - **Operating Power Supply 4.5V to 5.5V**
  - **Operating Frequency 33 MHz**
- **Transparent Power-Management System Architecture**
  - **Intel System Management Mode (SMM) Architecture Extension for Truly Compatible Systems**
  - **Power Management Transparent to Operating System and Application Programs**
  - **Programmable Power Management Modes**
- **Clock Freeze Mode Allows Clock Stopping at Any Time**
- **Full 32-Bit Internal Architecture**
  - **8-, 16-, 32-Bit Data Types**
  - **8 General-Purpose 32-Bit Registers**
- **Runs Intel386 Architecture Software in a Cost Effective 16-Bit Hardware Environment**
  - **Runs Same Applications and Operating Systems as the Intel386 SX and Intel386 DX Processors**
  - **Object Code Compatible with 8086, 80186, 80286 and Intel386 Processors**
- **High Performance 16-Bit Data Bus**
  - **33 MHz Clock**
  - **Two-Clock Bus Cycles**
  - **Address Pipelining Allows Use of Slower, Inexpensive Memories**
- **Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System**
- **Large Uniform Address Space**
  - **64 Megabyte Physical**
  - **64 Terabyte Virtual**
  - **4 Gigabyte Maximum Segment Size**
- **On-Chip Debugging Support Including Breakpoint Registers**
- **Complete System Development Support**
- **High-Speed CHMOS Technology**
- **Integrated Memory Management Unit (MMU)**
  - **Virtual Memory Support**
  - **Optional On-Chip Paging**
  - **Four Levels of Hardware-Enforced Protection**
  - **MMU fully compatible with those of the 80286 and Intel386 DX Processors**
- **Intel SmartDie Product**
  - **Full AC/DC Testing at Die Level**
  - **+25°C–80°C (Junction) Temperature Range**

NOTICE: This document contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel sales office that you have the latest SmartDie Product Specification before finalizing a design.

REFERENCE INFORMATION: The information in this document is provided as a supplement to the Standard Package Data Sheet on a specific product. Please reference the Standard Package Data Sheet (Order No. 272418) for additional product information and specifications not found in this document.

---

*Other brands and names are the property of their respective owners.

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

December 1994
Order Number: 271290-001

Intel386™ CXSB EMBEDDED MICROPROCESSOR

- Static Intel386™ CPU Core
  - Low Power Consumption
  - Operating Power Supply 2.7V to 3.6V
  - Operating Frequency
    25 MHz at 3.3V ±0.3V
    16 MHz at 3.0V ±0.3V
- Transparent Power-Management System Architecture
  - Intel System Management Mode Architecture Extension for Truly Compatible Systems
  - Power Management Transparent to Operating Systems and Application Programs
  - Programmable Power-Management Modes
- Clock Freeze Mode Allows Clock Stopping at Any Time
- Full 32-bit Internal Architecture
  - 8-, 16-, 32-bit Data Types
  - 8 General Purpose 32-bit Registers
- Runs Intel386 Architecture Software in a Cost-Effective, 16-bit Hardware Environment
  - Runs Same Applications and Operating Systems as the Intel386 SX and Intel386 DX Processors
  - Object Code Compatible with 8086, 80186, 80286, and Intel386 Processors
- High-performance 16-bit Data Bus
  - Two-Clock Bus Cycles
  - Address Pipelining Allows Use of Slower, Inexpensive Memories
- Integrated Memory Management Unit (MMU)
  - Virtual Memory Support
  - Optional On-Chip Paging
  - 4 Levels of Hardware-Enforced Protection
  - MMU Fully Compatible with Those of the 80286 and Intel386 DX Processors
- Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System
- Large, Uniform Address Space
  - 64 Megabyte Physical
  - 64 Terabyte Virtual
  - 4 Gigabyte Maximum Segment Size
- Numerics Support with Intel387™ SX and Intel387™ SL Math Coprocessors
- On-Chip Debugging Support Including Breakpoint Registers
- Complete System Development Support
- High-Speed CHMOS Technology
- 100-pin Plastic Quad Flatpack Package

3

The Intel386™ CXSB embedded microprocessor is a low-voltage, 32-bit, fully static CPU with a 16-bit external data bus, a 26-bit external address bus, and Intel's System Management Mode (SMM). The Intel386 CXSB CPU brings the vast software library of the Intel386 architecture to embedded systems. It provides the performance benefits of 32-bit programming with the cost savings associated with 16-bit hardware systems.

The Intel386 CXSB microprocessor is manufactured on Intel's 0.8-micron CHMOS V process. This process provides high performance and low power consumption for power-sensitive applications.

February 1995
Order Number: 272552-003

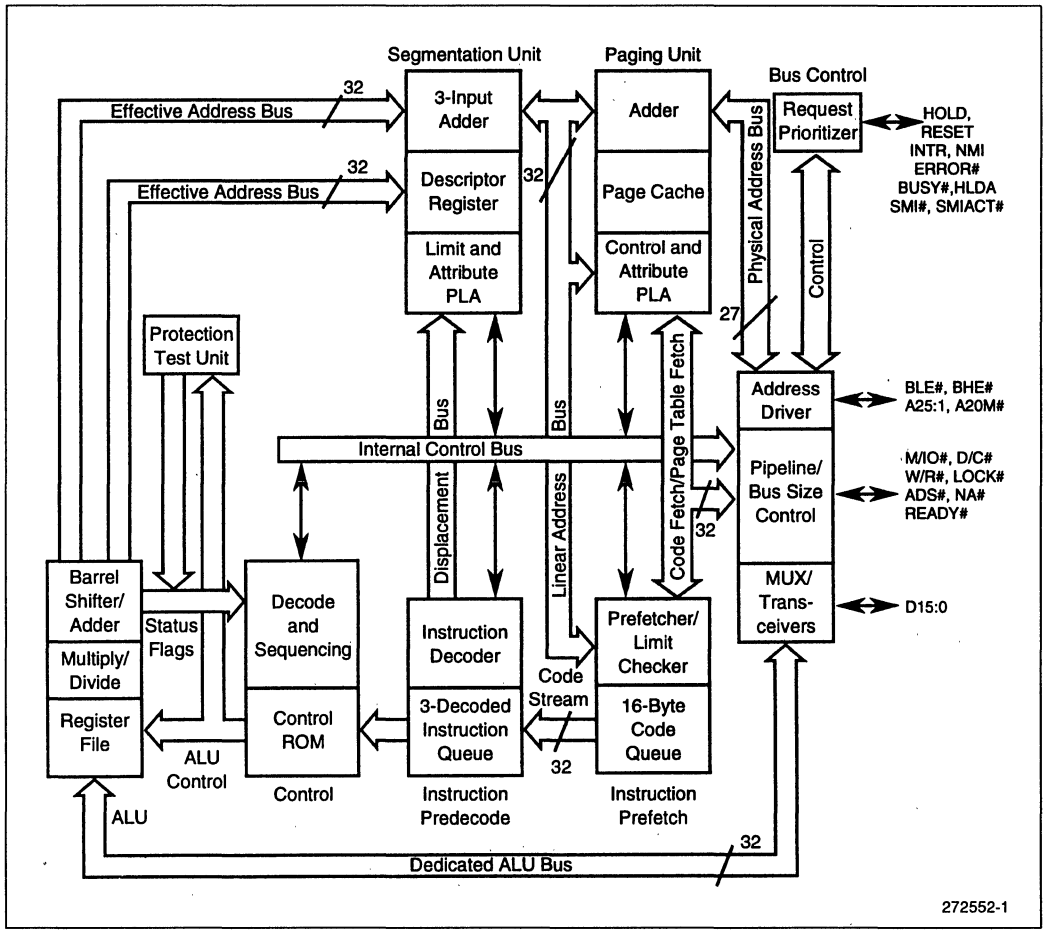
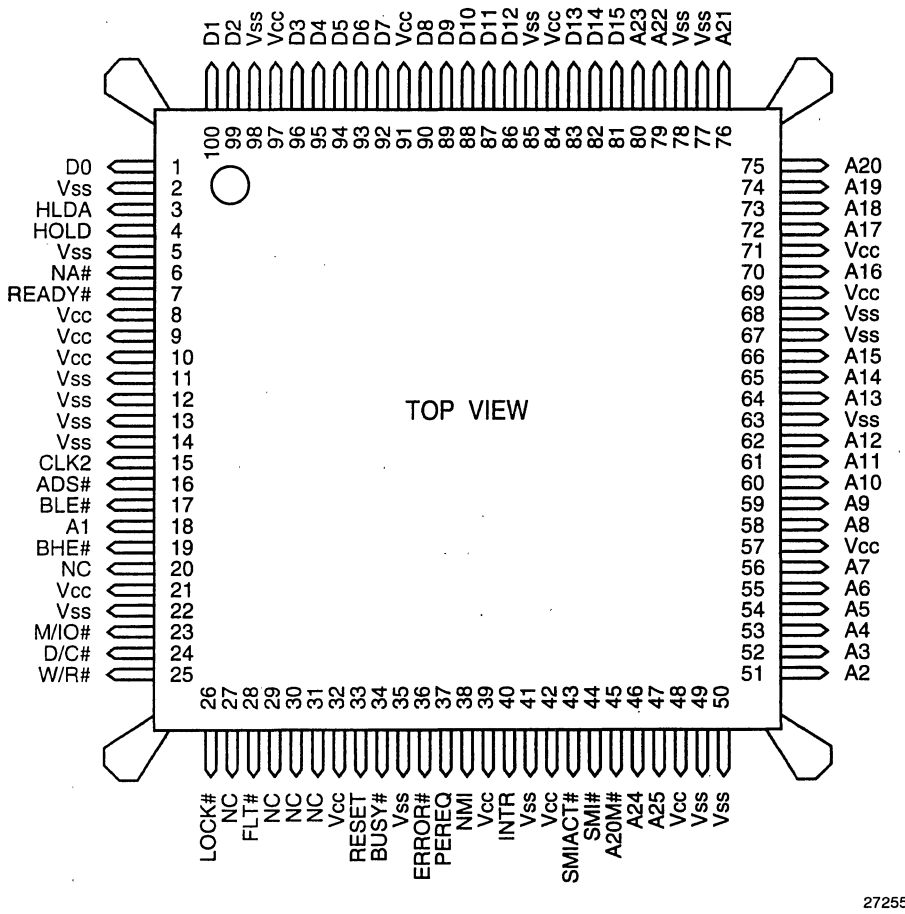


Figure 1. Intel386™ CXSB Microprocessor Block Diagram

1.0 PIN ASSIGNMENT



272552-2

NOTE:
NC = No Connection

Figure 2. Intel386™ CXSB Microprocessor Pin Assignment (PQFP)

Table 1. Pin Assignment

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	D0	26	LOCK#	51	A2	76	A21
2	V_{SS}	27	NC	52	A3	77	V_{SS}
3	HLDA	28	FLT#	53	A4	78	V_{SS}
4	HOLD	29	NC	54	A5	79	A22
5	V_{SS}	30	NC	55	A6	80	A23
6	NA#	31	NC	56	A7	81	D15
7	READY#	32	V_{CC}	57	V_{CC}	82	D14
8	V_{CC}	33	RESET	58	A8	83	D13
9	V_{CC}	34	BUSY#	59	A9	84	V_{CC}
10	V_{CC}	35	V_{SS}	60	A10	85	V_{SS}
11	V_{SS}	36	ERROR#	61	A11	86	D12
12	V_{SS}	37	PEREQ	62	A12	87	D11
13	V_{SS}	38	NMI	63	V_{SS}	88	D10
14	V_{SS}	39	V_{CC}	64	A13	89	D9
15	CLK2	40	INTR	65	A14	90	D8
16	ADS#	41	V_{SS}	66	A15	91	V_{CC}
17	BLE#	42	V_{CC}	67	V_{SS}	92	D7
18	A1	43	SMIACT#	68	V_{SS}	93	D6
19	BHE#	44	SMI#	69	V_{CC}	94	D5
20	NC	45	A20M#	70	A16	95	D4
21	V_{CC}	46	A24	71	V_{CC}	96	D3
22	V_{SS}	47	A25	72	A17	97	V_{CC}
23	M/IO#	48	V_{CC}	73	A18	98	V_{SS}
24	D/C#	49	V_{SS}	74	A19	99	D2
25	W/R#	50	V_{SS}	75	A20	100	D1

2.0 PIN DESCRIPTIONS

Table 2 lists the Intel386 CXSB microprocessor pin descriptions. The following definitions are used in the pin descriptions:

- # The named signal is active low.
- I Input signal.
- O Output signal.
- I/O Input and output signal.
- P Power pin.
- G Ground pin.

Table 2. Pin Descriptions

Symbol	Type	Pin	Name and Function
A20M# (Note 1)	I	45	**Address 20 Mask** controls the A20 address signal. When A20M# is low, the CPU masks off (forces low) the internal A20 physical address signal. This enables the CPU to run software that was developed using the 8086 address "wraparound" techniques. When A20M# is high, A20 is available on the address bus. While the bus is floating, A20M# has no effect on the A20 address signal. A20M# should be deasserted during SMM if the SMM handler accesses more than 1 Mbyte of memory.
A25:1 (Note 2)	O	47–46, 80–79, 76–72, 70, 66–64, 62–58, 56–51, 18	**Address Bus** outputs physical memory or port I/O addresses.
ADS#	O	16	**Address Status** indicates that the processor is driving a valid bus-cycle definition and address onto its pins (W/R#, D/C#, M/IO#, BHE#, BLE#, and A25:1).
BHE#	O	19	**Byte High Enable** indicates that the processor is transferring a high data byte.
BLE#	O	17	**Byte Low Enable** indicates that the processor is transferring a low data byte.
BUSY#	I	34	**Busy** indicates that the math coprocessor is busy.
CLK2	I	15	**CLK2** provides the fundamental timing for the device.
D/C#	O	24	**Data/Control** indicates whether the current bus cycle is a data cycle (memory or I/O) or a control cycle (interrupt acknowledge, halt, or code fetch). When D/C# is high, the bus cycle is a data cycle; when D/C# is low, the bus cycle is a control cycle.
D15:0	I/O	81–83, 86–90, 92–96, 99–100, 1	**Data Bus** inputs data during memory read, I/O read, and interrupt acknowledge cycles and outputs data during memory and I/O write cycles.
ERROR#	I	36	**Error** indicates that the math coprocessor has an error condition.

NOTES:
1. This pin supports the additional features of the Intel386 CXSB microprocessor; it is not present on the Intel386 SX microprocessor.
2. The A25:24 pins support the additional features of the Intel386 CXSB microprocessor; they are not present on the Intel386 SX microprocessor.

Table 2. Pin Descriptions (Continued)

Symbol	Type	Pin	Name and Function
FLT#	I	28	**Float** forces all bidirectional and output signals, including HLDA, to a high-impedance state.
HLDA	O	3	**Bus Hold Acknowledge** indicates that the CPU has surrendered control of its local bus to another bus master.
HOLD	I	4	**Bus Hold Request** allows another bus master to request control of the local bus.
INTR	I	40	**Interrupt Request** is a maskable input that causes the CPU to suspend execution of the current program and then execute an interrupt acknowledge cycle.
LOCK#	O	26	**Bus Lock** prevents other system bus masters from gaining control of the system bus while it is active (low).
M/IO#	O	23	**Memory/IO** indicates whether the current bus cycle is a memory cycle or an input/output cycle. When M/IO# is high, the bus cycle is a memory cycle; when M/IO# is low, the bus cycle is an I/O cycle.
NA#	I	6	**Next Address** requests address pipelining.
NC		20, 27, 29–31	**No Connection** should always be left unconnected. Connecting a NC pin may cause the processor to malfunction or cause your application to be incompatible with future steppings of the device.
NMI	I	38	**Nonmaskable Interrupt Request** is a nonmaskable input that causes the CPU to suspend execution of the current program and execute an interrupt acknowledge function.
PEREQ	I	37	**Processor Extension Request** indicates that the math coprocessor has data to transfer to the processor.
READY#	I	7	**Bus Ready** indicates that the current bus cycle is finished and the external device is ready to accept more data from the processor.
RESET	I	33	**Reset** suspends any operation in progress and places the processor into a known reset state.
SMI# (Note 1)	I	44	**System Management Interrupt** invokes System Management Mode (SMM). SMI# is the highest priority interrupt. It is latched on its falling edge and it forces the CPU into SMM upon completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# cannot interrupt LOCKed bus cycles or a currently executing SMM. If the processor receives a second SMI# while it is in SMM, it will latch the second SMI# on the SMI# falling edge. However, the processor must exit SMM by executing a Resume instruction (RSM) before it can service the second SMI#.

NOTES:
1. This pin supports the additional features of the Intel386 CXSB microprocessor; it is not present on the Intel386 SX microprocessor.
2. The A25:24 pins support the additional features of the Intel386 CXSB microprocessor; they are not present on the Intel386 SX microprocessor.

Table 2. Pin Descriptions (Continued)

Symbol	Type	Pin	Name and Function
SMIACT# (Note 1)	O	43	**System Management Interrupt Active** indicates that the processor is operating in System Management Mode (SMM). It is asserted when the processor initiates an SMM sequence and remains asserted (low) until the processor executes the Resume instruction (RSM).
W/R#	O	25	**Write/Read** indicates whether the current bus cycle is a write cycle or a read cycle. When W/R# is high, the bus cycle is a write cycle; when W/R# is low, it is a read cycle.
V_{CC}	P	8–10, 21, 32, 39, 42, 48, 57, 69, 71, 84, 91, 97	**System Power** provides the nominal DC supply input.
V_{SS}	G	2, 5, 11–14, 22 35, 41, 49–50, 63, 67–68, 77–78, 85, 98	**System Ground** provides the 0V connection from which all inputs and outputs are measured.

NOTES:
1. This pin supports the additional features of the Intel386 CXSB microprocessor; it is not present on the Intel386 SX microprocessor.
2. The A25:24 pins support the additional features of the Intel386 CXSB microprocessor; they are not present on the Intel386 SX microprocessor.

3.0 DESIGN CONSIDERATIONS

This section describes the Intel386 CXSB microprocessor instruction set, component and revision identifier, and package thermal specifications.

3.1 Instruction Set

The Intel386 CXSB microprocessor uses the same instruction set as the Intel386 SX microprocessor with the following exceptions.

The Intel386 CXSB microprocessor has one new instruction (RSM). This Resume instruction causes the processor to exit System Management Mode (SMM). RSM requires 338 clocks for execution.

The Intel386 CXSB microprocessor requires more clock cycles than the Intel386 SX microprocessor to execute some instructions. Table 4 lists these instructions and the Intel386 CXSB microprocessor execution times. For the equivalent Intel386 SX microprocessor execution times, refer to the "Instruction Set Clock Count Summary" table in the *Intel386 SX Microprocessor* data sheet (order number 240187).

3.2 Component and Revision Identifier

To assist users, the microprocessor holds a component identifier and revision identifier in its DX register after reset. The upper 8 bits of DX hold the component identifier, 23H. (The lower nibble, 03H, identifies the Intel386 architecture, while the upper nibble, 02H, identifies the second member of the Intel386 microprocessor family.)

The lower 8 bits of DX hold the revision level identifier. The revision identifier will, in general, chronologically track those component steppings that are intended to have certain improvements or distinction from previous steppings. The revision identifier will track that of the Intel386 CPU whenever possible. However, the revision identifier value is not guaranteed to change with every stepping revision or to follow a completely uniform numerical sequence, depending on the type or intent of the revision or the manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component. The initial revision identifier for the Intel386 CXSB microprocessor is 09H.

3.3 Package Thermal Specifications

The Intel386 CXSB microprocessor is specified for operation with case temperature (T_C) within the range of 0°C–100°C. The case temperature can be measured in any environment to determine whether the microprocessor is within the specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

An increase in the ambient temperature (T_A) causes a proportional increase in the case temperature (T_C) and the junction temperature (T_J). A packaged device produces thermal resistance between junction and case temperatures (θ_{JC}) and between junction and ambient temperatures (θ_{JA}). The relationships between the temperature and thermal resistance parameters are expressed by these equations (P = power dissipated as heat = $V_{CC} \times I_{CC}$):

1. $T_J = T_c + P\,\theta_{JC}$
2. $T_A = T_J - P\,\theta_{JA}$
3. $T_C = T_A + P \times [\theta_{JA} - \theta_{JC}]$

A safe operating temperature can be calculated from the above equations by using the maximum safe T_c (100°C), the maximum power drawn by the chip in the specific design, and the θ_{JA} and θ_{JC} values from Table 3. The θ_{JA} value depends on the airflow (measured at the top of the chip) provided by the system ventilation.

Table 3. Thermal Resistances (0°C/W) θ_{JA}, θ_{JC}

Pkg	θ_{JC}	θ_{JA} versus Airflow (ft/min)		
		0	100	200
100 PQFP	5.1	46.0	44.8	41.2
100 SQFP	14	63	N/A	N/A

Table 4. Intel386™ CXSB Microprocessor Instruction Execution Times (in Clock Counts)

Instruction	Clock Count		
	Virtual 8086 Mode (Note 1)	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode (Note 3)
POPA		28	35
IN: Fixed Port Variable Port	 27 28	 14 15	 7/29 8/29
OUT: Fixed Port Variable Port	 27 28	 14 15	 7/29 9/29
INS	30	17	9/32
OUTS	31	18	10/33
REP INS	31 + 6n (Note 2)	17 + 6n (Note 2)	10 + 6n/32 + 6n (Note 2)
REP OUTS	30 + 8n (Note 2)	16 + 8n (Note 2)	10 + 8n/31 + 8n (Note 2)
HLT		7	7
MOV C0, reg		10	10

NOTES:
1. The clock count values in this column apply if I/O permission allows I/O to the port in virtual 8086 mode. If the I/O bit map denies permission, exception fault 13 occurs; see clock counts for the INT 3 instruction in the "Instruction Set Clock Count Summary" table in the *Intel386 SX Microprocessor* data sheet (order number 240187).
2. n = the number of times repeated.
3. When two clock counts are listed, the smaller value refers to a register operand and the larger value refers to a memory operand.

4.0 DC SPECIFICATIONS

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

ABSOLUTE MAXIMUM RATINGS*

Storage Temperature ..... −65°C to +150°C
Case Temperature Under Bias... −65°C to +110°C
Supply Voltage with Respect to V_{SS} .. −0.5V to 6.5V
Voltage on Other Pins ..... −0.5V to V_{CC} + 0.5V

WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

OPERATING CONDITIONS*

V_{CC} (Digital Supply Voltage) .....2.7V to 3.6V
T_{CASE} (Case Temperature Under Bias) 0°C to 100°C
F_{OSC} (Operating Frequency) .....0 MHz to 25 MHz

Table 5. DC Characteristics

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.3	$0.3V_{CC}$	V	
V_{IH}	Input High Voltage	$0.7V_{CC}$	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	+0.8	V	
V_{IHC}	CLK2 Input High Voltage	V_{CC} −0.6	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage		0.4 0.2	V	I_{OL} = 2 mA I_{OL} = 0.5 mA
V_{OH}	Output High Voltage	V_{CC} − 0.4 V_{CC} − 0.2		V V	I_{OH} = −0.5 mA I_{OH} = −0.1 mA
I_{LI}	Input Leakage Current (For All Pins except PEREQ, BUSY#, FLT#, ERROR#, A20M#, SMI#)		±15	μA	$0 \le V_{IN} \le V_{CC}$
I_{IH}	Input Leakage Current (PEREQ)		150	μA	V_{IH} = 2.2V (Note 1)
I_{IL}	Input Leakage Current (BUSY#, FLT#, ERROR#, A20M#, and SMI#)		−120	μA	V_{IL} = 0.45V (Note 2)
I_{LO}	Output Leakage Current		±15	μA	$0.45V \le V_{OUT} \le V_{CC}$
I_{CC}	Supply Current CLK2 = 50 MHz, CLK = 25 MHz CLK2 = 32 MHz, CLK = 16 MHz		 115 85	 mA mA	(Notes 3, 4) Typical = 80 mA Typical = 50 mA
I_{CCF}	Standby Current (Freeze Mode)		150	μA	Typical = 10 μA (Notes 3, 4)
C_{IN}	Input Capacitance		10	pF	F_c = 1 MHz (Note 5)
C_{OUT}	Output or I/O Capacitance		12	pF	F_c = 1 MHz (Note 5)
C_{CLK}	CLK2 Capacitance		20	pF	F_c = 1 MHz (Note 5)

NOTES:
1. PEREQ input has an internal weak pull-down resistor.
2. BUSY#, FLT#, SMI#, A20M# and ERROR# inputs each have an internal weak pull-up resistor.
3. I_{CC} max measurement at worst-case frequency, V_{CC}, and temperature with reset active.
4. I_{CC} typical and I_{CCF} typical are measured at nominal V_{CC} and are not fully tested.
5. Not fully tested.

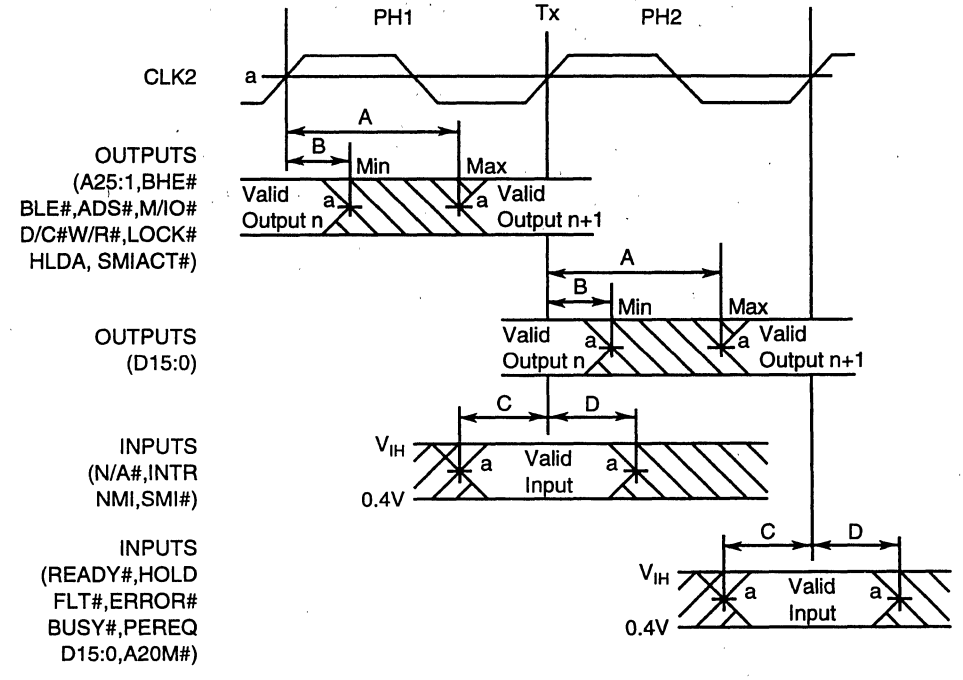
5.0 AC SPECIFICATIONS

Table 6 lists output delays, input setup requirements, and input hold requirements. All AC specifications are relative to the CLK2 rising edge crossing the $V_{CC}/2$ level.

Figure 3 shows the measurement points for AC specifications. Inputs must be driven to the indicated voltage levels when AC specifications are measured. Output delays are specified with minimum and maximum limits measured as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs ADS#, W/R#, D/C#, MI/O#, LOCK#, BHE#, BLE#, A25:1, SMIACT# and HLDA change only at the beginning of phase one. D15:0 (write cycles) change only at the beginning of phase two.

The READY#, HOLD, BUSY#, ERROR#, PEREQ, FLT#, A20M# and D15:0 (read cycles) inputs are sampled at the beginning of phase one. The NA#, INTR, SMI# and NMI inputs are sampled at the beginning of phase two.



LEGEND
a-$V_{CC}/2$
A-Maximum Output Delay Spec
B-Minimum Output Delay Spec
C-Minimum Input Setup Spec
D-Minimum Input Hold Spec

Figure 3. Drive Levels and Measurement Points for AC Specifications

Table 6. AC Characteristics

Symbol	Parameter	25 MHz 3.0V–3.6V		16 MHz 2.7V–3.3V		Test Condition
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
	Operating Frequency	0	25	0	16	MHz (Note 1)
t1	CLK2 Period	20		31		
t2a	CLK2 High Time	7		9		(Note 2)
t2b	CLK2 High Time	4		5		(Note 2)
t3a	CLK2 Low Time	7		9		(Note 2)
t3b	CLK2 Low Time	5		7		(Note 2)
t4	CLK2 Fall Time		7		8	(Note 2)
t5	CLK2 Rise Time		7		8	(Note 2)
t6	A25:1 Valid Delay	4	17	4	36	C_L = 50 pF
t7	A25:1 Float Delay	4	30	4	40	(Note 3)
t8	BHE#, BLE#, LOCK# Valid Delay	4	17	4	36	C_L = 50 pF
t8a	SMIACT# Valid Delay	4	17	4	36	C_L = 50 pF
t9	BHE#, BLE#, LOCK# Float Delay	4	30	4	40	(Note 3)
t10	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	17	4	33	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	30	4	35	(Note 3)
t12	D15:0 Write Data Valid Delay	7	23	4	40	C_L = 50 pF
t12a	D15:0 Write Data Hold Time	2		—		C_L = 50 pF
t13	D15:0 Write Data Float Delay	4	22	4	35	(Note 3)
t14	HLDA Valid Delay	4	22	4	33	C_L = 50 pF
t15	NA# Setup Time	5		5		
t16	NA# Hold Time	3		21		
t19	READY#, A20M# Setup Time	9		19		
t20	READY#, A20M# Hold Time	4		4		
t21	D15:0 Read Setup Time	7		9		
t22	D15:0 Read Hold Time	5		6		
t23	HOLD Setup Time	9		26		
t24	HOLD Hold Time	3		5		
t25	RESET Setup Time	8		13		
t26	RESET Hold Time	3		4		
t27	NMI, INTR Setup Time	6		16		(Note 4)
t27a	SMI# Setup Time	6		16		(Note 4)

NOTES:
1. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
2. These are not tested. They are guaranteed by characterization.
3. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
4. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.

Table 6. AC Characteristics (Continued)

Symbol	Parameter	25 MHz 3.0V–3.6V		16 MHz 2.7V–3.3V		Test Condition
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
t28	NMI, INTR Hold Time	6		16		(Note 4)
t28a	SMI# Hold Time	6		16		(Note 4)
t29	PEREQ, ERROR#, BUSY#, FLT# Setup Time	6		16		(Note 4)
t30	PEREQ, ERROR#, BUSY#, FLT# Hold Time	5		5		(Note 4)

NOTES:
1. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
2. These are not tested. They are guaranteed by characterization.
3. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
4. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.

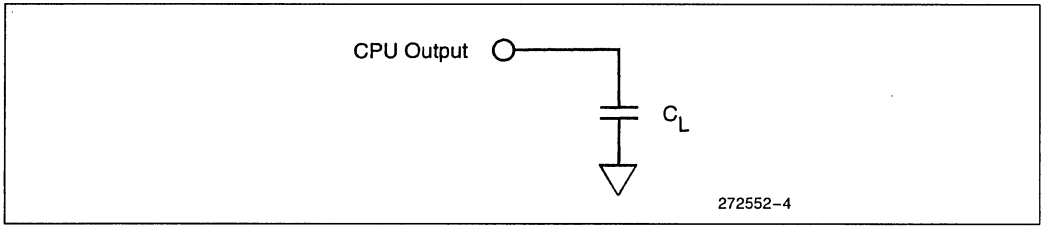


Figure 4. AC Test Loads

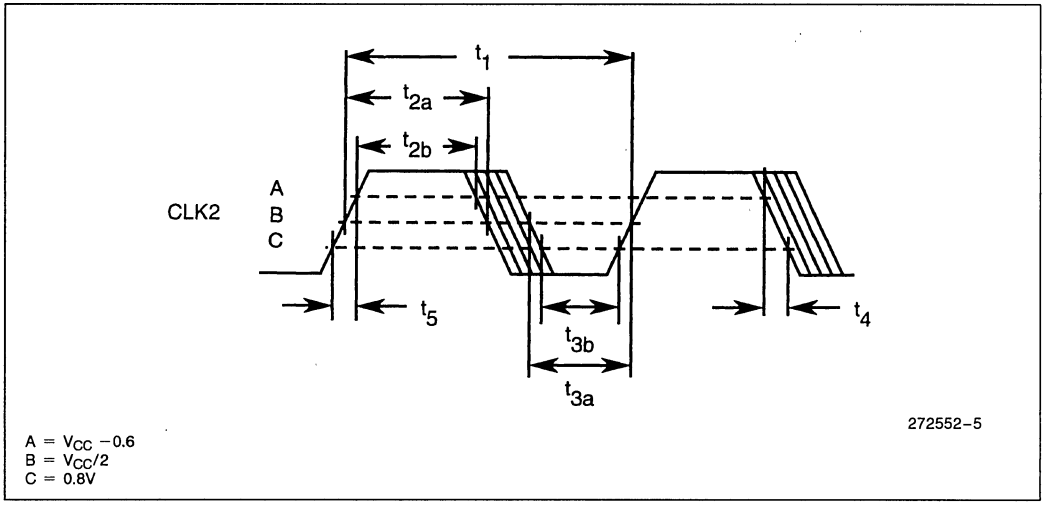


Figure 5. CLK2 Waveform

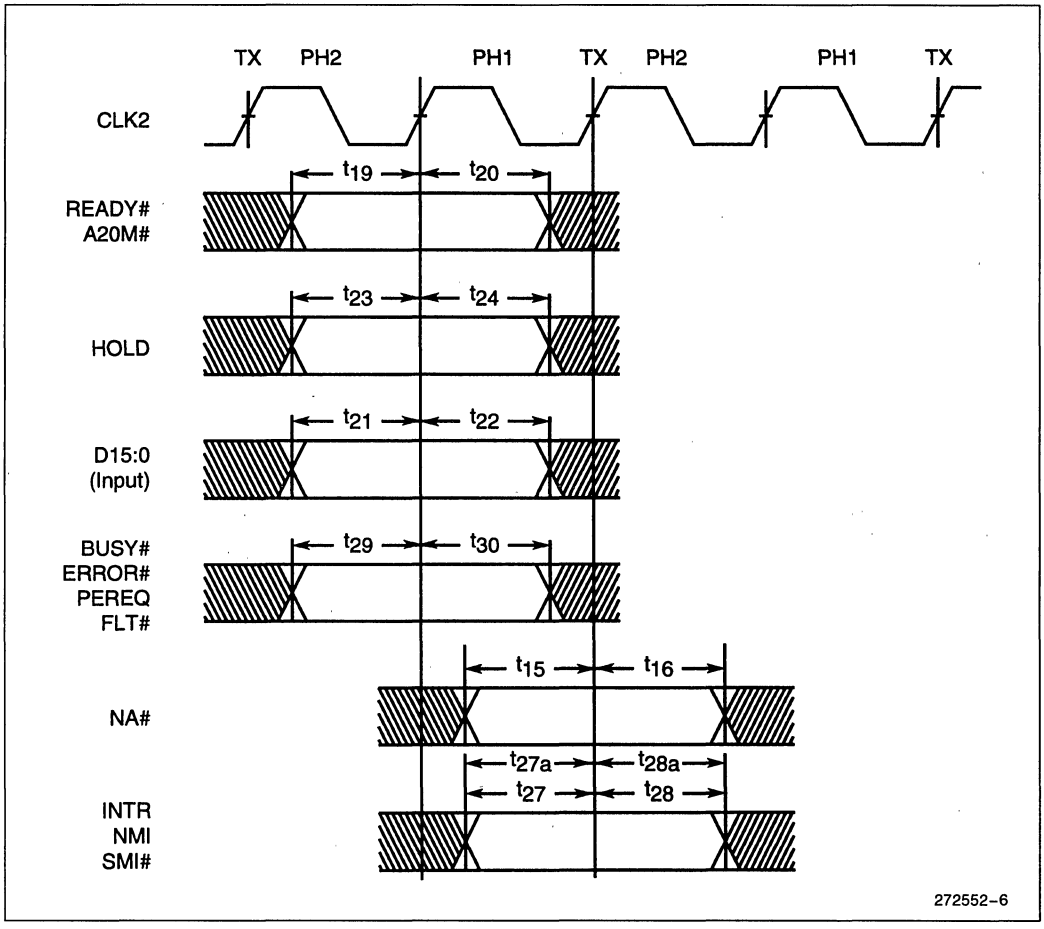


Figure 6. AC Timing Waveforms—Input Setup and Hold Timing

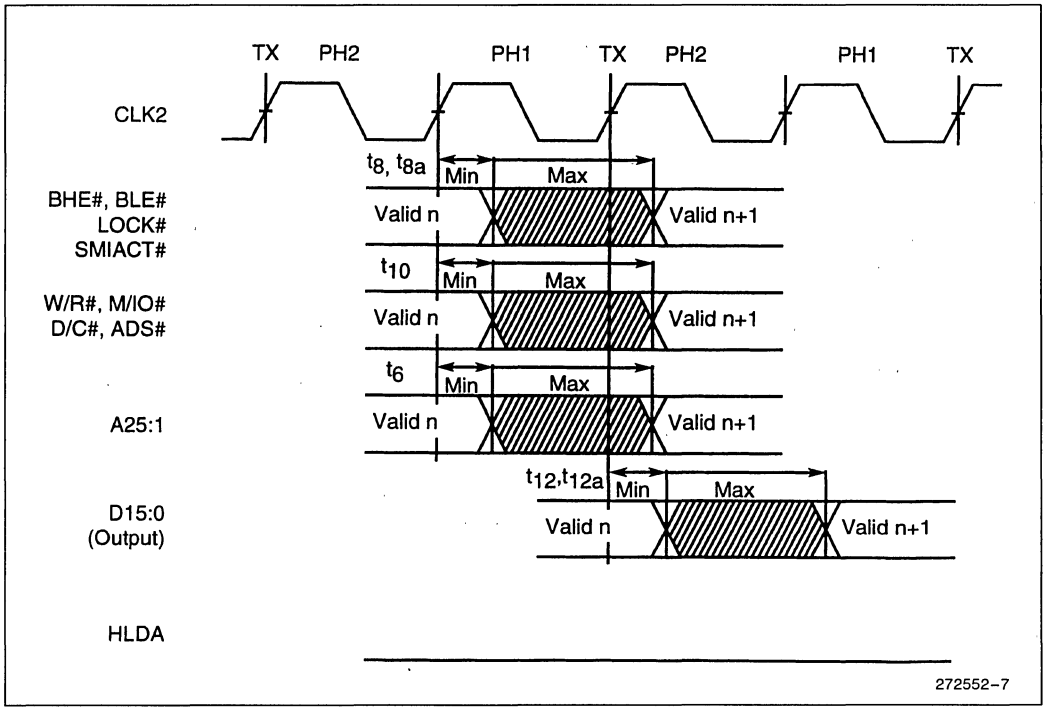


Figure 7. AC Timing Waveforms—Output Valid Delay Timing

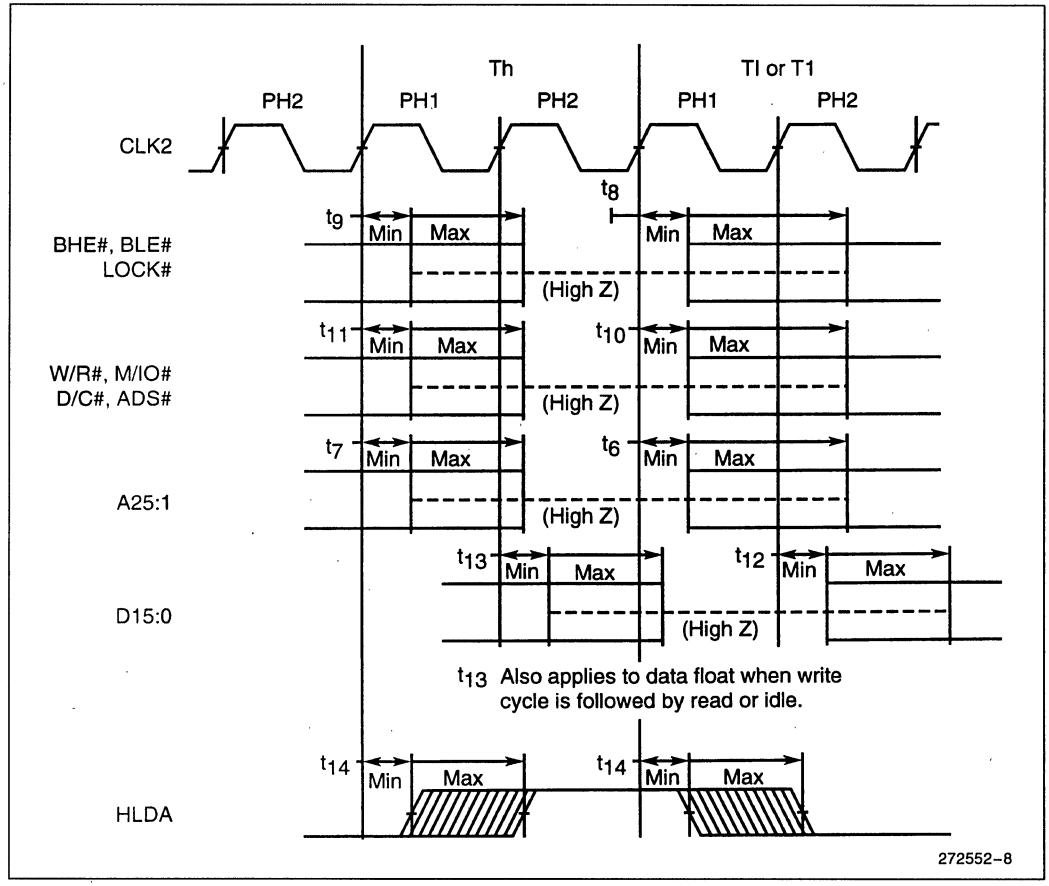


Figure 8. AC Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing

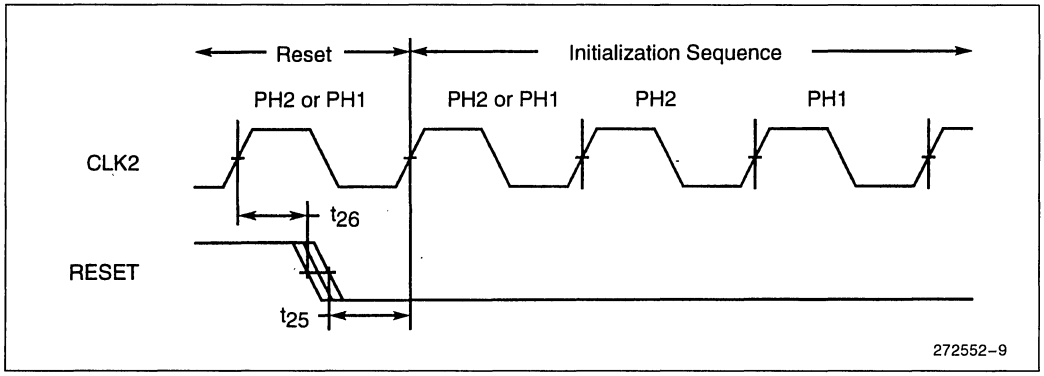


Figure 9. AC Timing Waveforms—RESET Setup and Hold Timing and Internal Phase

6.0 REVISION HISTORY

This -003 datasheet contains the following changes from the -002 version.

- Added thermal resistance data for 100 lead SQFP package.
- Updated dynamic I_{CC} values in DC Characteristics Table.

3

Intel386™ EX EMBEDDED MICROPROCESSOR

- Static Intel386™ CPU Core
  - Low Power Consumption
  - Operating Power Supply 2.7V to 5.5V
  - Operating Frequency 16 MHz at 2.7V to 3.3V; 20 MHz at 3.0V to 3.6V; 25 MHz at 4.5V to 5.5V
- Transparent Power-management System Architecture
  - Intel System Management Mode Architecture Extension for Truly Compatible Systems
  - Power Management Transparent to Operating Systems and Application Programs
  - Programmable Power-management Modes
- Powerdown Mode
  - Clock Stopping at Any Time
  - Only 10–20 μA Typical CPU Sink Current
- Full 32-bit Internal Architecture
  - 8-, 16-, 32-bit Data Types
  - 8 General Purpose 32-bit Registers
- Runs Intel386 Architecture Software in a Cost-effective 16-bit Hardware Environment
  - Runs Same Applications and Operating Systems as the Intel386 SX and Intel386 DX Processors
  - Object Code Compatible with 8086, 80186, 80286, and Intel386 Processors
- High-performance 16-bit Data Bus
  - Two-clock Bus Cycles
  - Address Pipelining Allows Use of Slower, Inexpensive Memories
- Integrated Memory Management Unit
  - Virtual Memory Support
  - Optional On-chip Paging
  - 4 Levels of Hardware-enforced Protection
  - MMU Fully Compatible with Those of the 80286 and Intel386 DX Processors
- Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System
- Large Uniform Address Space
  - 64 Megabyte Physical
  - 64 Terabyte Virtual
  - 4 Gigabyte Maximum Segment Size
- Numerics Support with Intel387™ SX and Intel387 SL Math Coprocessors
- On-chip Debugging Support Including Breakpoint Registers
- Complete System Development Support
- High Speed CHMOS Technology
- Two Package Types
  - 132-pin Plastic Quad Flatpack
  - 144-pin Thin Quad Flatpack
- Integrated Peripheral Functions
  - Clock and Power Management Unit
  - Chip-select Unit
  - Interrupt Control Unit
  - Timer Control Unit
  - Watchdog Timer Unit
  - Asynchronous Serial I/O Unit
  - Synchronous Serial I/O Unit
  - Parallel I/O Unit
  - DMA and Bus Arbiter Unit
  - Refresh Control Unit
  - JTAG-compliant Test-logic Unit

The Intel386™ EX Embedded Microprocessor is a highly integrated, 32-bit, fully static CPU optimized for embedded control applications. With a 16-bit external data bus, a 26-bit external address bus, and Intel's System Management Mode (SMM), the Intel386 EX microprocessor brings the vast software library of Intel386 architecture to embedded systems. It provides the performance benefits of 32-bit programming with the cost savings associated with 16-bit hardware systems.

September 1994
Order Number: 272420-004

Intel386™ EX EMBEDDED MICROPROCESSOR

CONTENTS

CONTENTS

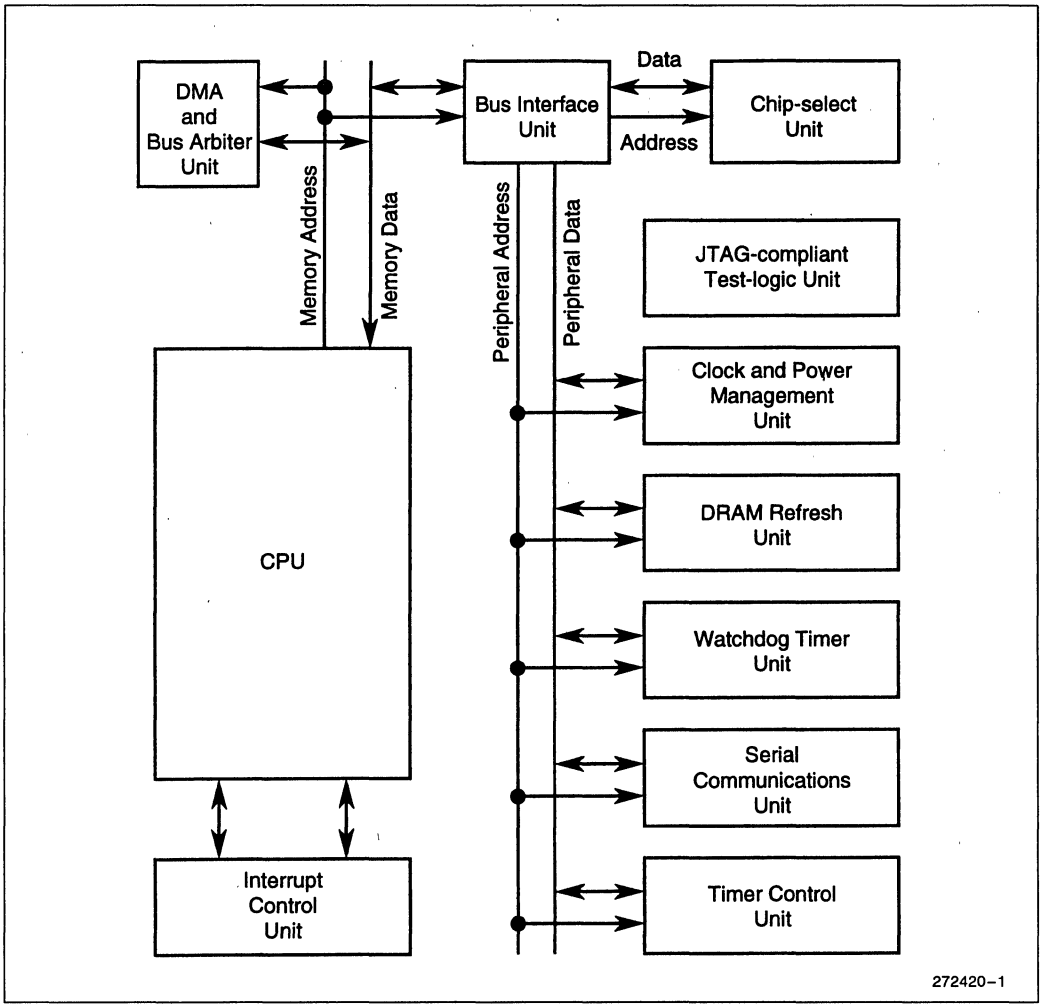


Figure 1. Intel386™ EX Microprocessor Block Diagram

1.0 PIN ASSIGNMENT

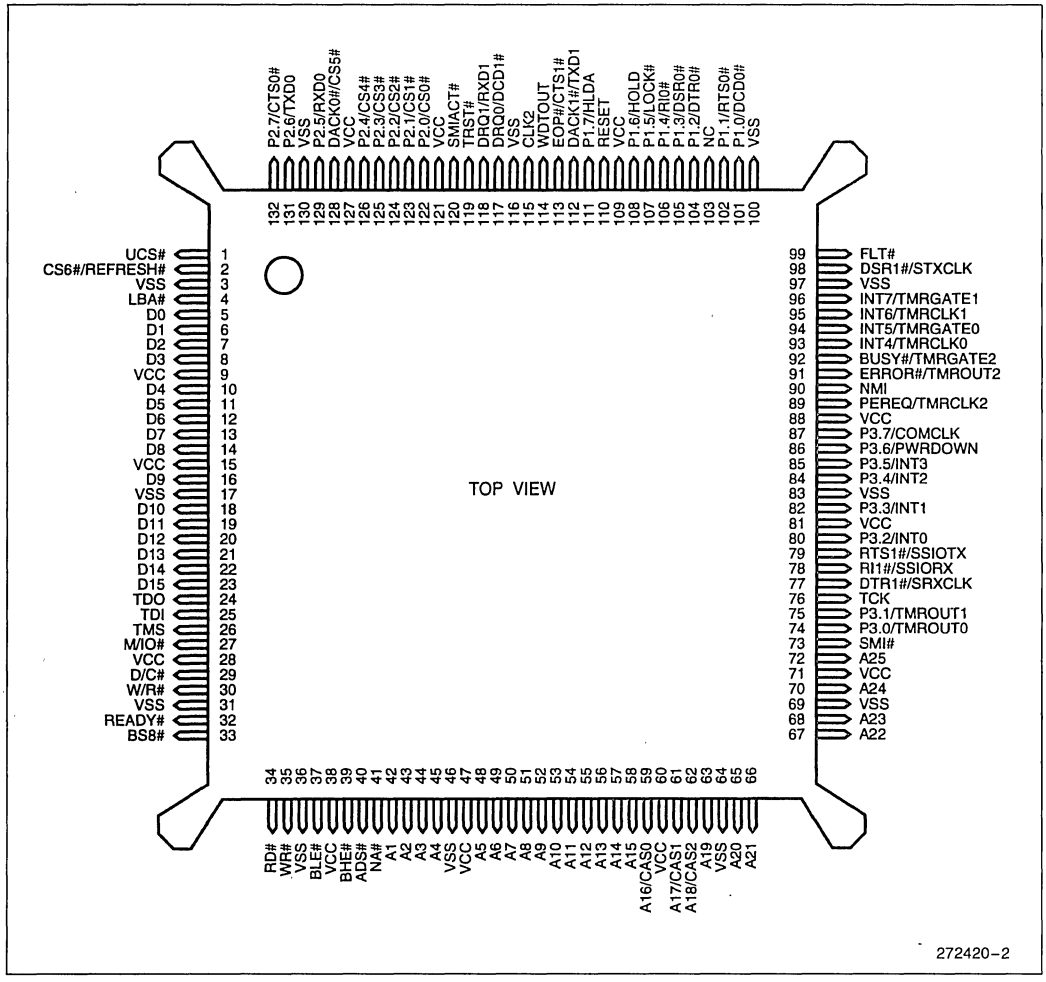


Figure 2. Intel386™ EX Microprocessor 132-Pin PQFP Pin Assignment

Table 1. 132-Pin PQFP Pin Assignment

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	UCS#	34	RD#	67	A22	100	V_{SS}
2	CS6#/REFRESH#	35	WR#	68	A23	101	P1.0/DCD0#
3	V_{SS}	36	V_{SS}	69	V_{SS}	102	P1.1/RTS0#
4	LBA#	37	BLE#	70	A24	103	NC
5	D0	38	V_{CC}	71	V_{CC}	104	P1.2/DTR0#
6	D1	39	BHE#	72	A25	105	P1.3/DSR0#
7	D2	40	ADS#	73	SMI#	106	P1.4/RI0#
8	D3	41	NA#	74	P3.0/TMROUT0	107	P1.5/LOCK#
9	V_{CC}	42	A1	75	P3.1/TMROUT1	108	P1.6/HOLD
10	D4	43	A2	76	TCK	109	V_{CC}
11	D5	44	A3	77	DTR1#/SRXCLK	110	RESET
12	D6	45	A4	78	RI1#/SSIORX	111	P1.7/HLDA
13	D7	46	V_{SS}	79	RTS1#/SSIOTX	112	DACK1#/TXD1
14	D8	47	V_{CC}	80	P3.2/INT0	113	EOP#/CTS1#
15	V_{CC}	48	A5	81	V_{CC}	114	WDTOUT
16	D9	49	A6	82	P3.3/INT1	115	CLK2
17	V_{SS}	50	A7	83	V_{SS}	116	V_{SS}
18	D10	51	A8	84	P3.4/INT2	117	DRQ0/DCD1#
19	D11	52	A9	85	P3.5/INT3	118	DRQ1/RXD1
20	D12	53	A10	86	P3.6/PWRDOWN	119	TRST#
21	D13	54	A11	87	P3.7/COMCLK	120	SMIACT#
22	D14	55	A12	88	V_{CC}	121	V_{CC}
23	D15	56	A13	89	PEREQ/TMRCLK2	122	P2.0/CS0#
24	TDO	57	A14	90	NMI	123	P2.1/CS1#
25	TDI	58	A15	91	ERROR#/TMROUT2	124	P2.2/CS2#
26	TMS	59	A16/CAS0	92	BUSY#/TMRGATE2	125	P2.3/CS3#
27	M/IO#	60	V_{CC}	93	INT4/TMRCLK0	126	P2.4/CS4#
28	V_{CC}	61	A17/CAS1	94	INT5/TMRGATE0	127	V_{CC}
29	D/C#	62	A18/CAS2	95	INT6/TMRCLK1	128	DACK0#/CS5#
30	W/R#	63	A19	96	INT7/TMRGATE1	129	P2.5/RXD0
31	V_{SS}	64	V_{SS}	97	V_{SS}	130	V_{SS}
32	READY#	65	A20	98	DSR1#/STXCLK	131	P2.6/TXD0
33	BS8#	66	A21	99	FLT#	132	P2.7/CTS0#

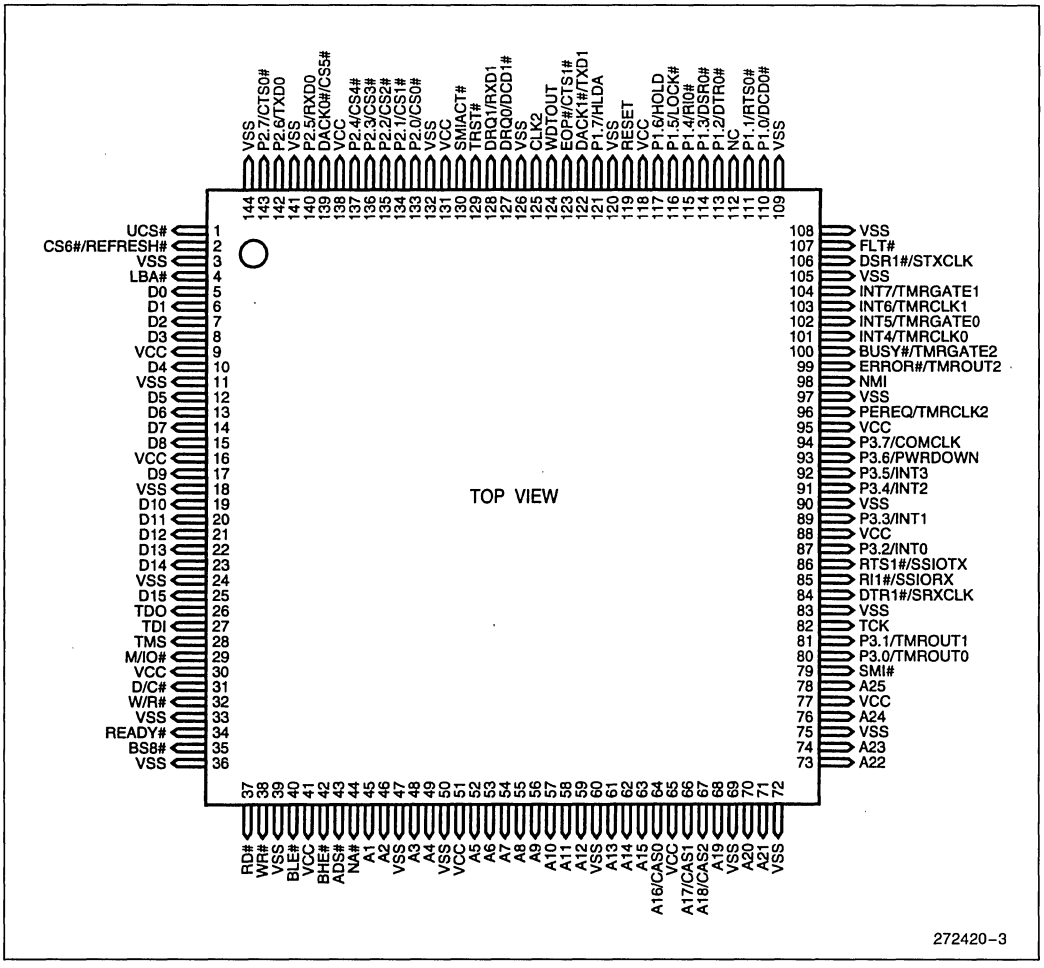


Figure 3. Intel386™ EX Microprocessor 144-Pin TQFP Pin Assignment

Table 2. 144 Pin TQFP Pin Assignment

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	UCS#	37	RD#	73	A22	109	V_{SS}
2	CS6#/REFRESH#	38	WR#	74	A23	110	P1.0/DCD0#
3	V_{SS}	39	V_{SS}	75	V_{SS}	111	P1.1/RTS0#
4	LBA#	40	BLE#	76	A24	112	NC
5	D0	41	V_{CC}	77	V_{CC}	113	P1.2/DTR0#
6	D1	42	BHE#	78	A25	114	P1.3/DSR0#
7	D2	43	ADS#	79	SMI#	115	P1.4/RI0#
8	D3	44	NA#	80	P3.0/TMROUT0	116	P1.5/LOCK#
9	V_{CC}	45	A1	81	P3.1/TMROUT1	117	P1.6/HOLD
10	D4	46	A2	82	TCK	118	V_{CC}
11	V_{SS}	47	V_{SS}	83	V_{SS}	119	RESET
12	D5	48	A3	84	DTR1#/SRXCLK	120	V_{SS}
13	D6	49	A4	85	RI1#/SSIORX	121	P1.7/HLDA
14	D7	50	V_{SS}	86	RTS1#/SSIOTX	122	DACK1#/TXD1
15	D8	51	V_{CC}	87	P3.2/INT0	123	EOP#/CTS1#
16	V_{CC}	52	A5	88	V_{CC}	124	WDTOUT
17	D9	53	A6	89	P3.3/INT1	125	CLK2
18	V_{SS}	54	A7	90	V_{SS}	126	V_{SS}
19	D10	55	A8	91	P3.4/INT2	127	DRQ0/DCD1#
20	D11	56	A9	92	P3.5/INT3	128	DRQ1/RXD1
21	D12	57	A10	93	P3.6/PWRDOWN	129	TRST#
22	D13	58	A11	94	P3.7/COMCLK	130	SMIACT#
23	D14	59	A12	95	V_{CC}	131	V_{CC}
24	V_{SS}	60	V_{SS}	96	PEREQ/TMRCLK2	132	V_{SS}
25	D15	61	A13	97	V_{SS}	133	P2.0/CS0#
26	TDO	62	A14	98	NMI	134	P2.1/CS1#
27	TDI	63	A15	99	ERROR#/TMROUT2	135	P2.2/CS2#
28	TMS	64	A16/CAS0	100	BUSY#/TMRGATE2	136	P2.3/CS3#
29	M/IO#	65	V_{CC}	101	INT4/TMRCLK0	137	P2.4/CS4#
30	V_{CC}	66	A17/CAS1	102	INT5/TMRGATE0	138	V_{CC}
31	D/C#	67	A18/CAS2	103	INT6/TMRCLK1	139	DACK0#/CS5#
32	W/R#	68	A19	104	INT7/TMRGATE1	140	P2.5/RXD0
33	V_{SS}	69	V_{SS}	105	V_{SS}	141	V_{SS}
34	READY#	70	A20	106	DSR1#/STXCLK	142	P2.6/TXD0
35	BS8#	71	A21	107	FLT#	143	P2.7/CTS0#
36	V_{SS}	72	V_{SS}	108	V_{SS}	144	V_{SS}

2.0 PIN DESCRIPTIONS

Table 3 lists the Intel386 EX microprocessor pin descriptions. The following definitions are used in the pin descriptions:

#	The named signal is active low.
I	Standard CMOS input signal.
O	Standard CMOS output signal.
I/O	Input and output signal.
I/OD	Input and open-drain output signal.
ST	Schmitt-triggered input signal.
P	Power pin.
G	Ground pin.

Table 3. Intel386™ EX Microprocessor Pin Descriptions

Symbol	Type	Name and Function
A25:1	O	**Address Bus** outputs physical memory or port I/O addresses. These signals are valid when ADS# is active and remain valid until the next T1, T2P, or Ti. During HOLD cycles they are driven to a high-impedance state. A18:16 are multiplexed with CAS2:0.
ADS#	O	**Address Status** indicates that the processor is driving a valid bus-cycle definition and address (W/R#, D/C#, M/IO#, A25:1, BHE#, BLE#) onto its pins.
BHE#	O	**Byte High Enable** indicates that the processor is transferring a high data byte.
BLE#	O	**Byte Low Enable** indicates that the processor is transferring a low data byte.
BS8#	I	**Bus Size** indicates that an 8-bit device is currently being addressed.
BUSY#	I	**Busy** indicates that the math coprocessor is busy. If BUSY# is sampled low at the falling edge of RESET, the processor performs an internal self test. BUSY# is multiplexed with TMRGATE2.
CAS2:0	O	**Cascade Address** carries the slave address information from the 8259A master interrupt module during interrupt acknowledge bus cycles. CAS2:0 are multiplexed with A18:16.
CLK2	ST	**Clock Input** is connected to an external clock that provides the fundamental timing for the device.
COMCLK	I	**Serial Communications Baud Clock** is an alternate clock source for the asynchronous serial ports. COMCLK is multiplexed with P3.7.
CS6:0#	O	**Chip-selects** (lower) are activated when the address of a memory or I/O bus cycle is within the address region programmed by the user. They are multiplexed as follows: CS6# with REFRESH#, CS5# with DACK0#, and CS4:0# with P2.4:0.
CTS1:0#	I	**Clear to Send SIO1 and SIO0** prevent the transmission of data to the asynchronous serial port's RXD1 and RXD0 pins, respectively. CTS1# is multiplexed with EOP#, and CTS0# is multiplexed with P2.7. CTS1# requires an external pull-up resistor.
D15:0	I/O	**Data Bus** inputs data during memory read, I/O read, and interrupt acknowledge cycles and outputs data during memory and I/O write cycles. During writes, this bus is driven during phase 2 of T1 and remains active until phase 2 of the next T1, T1P, or Ti. During reads, data is latched on the falling edge of phase 2.

Table 3. Intel386™ EX Microprocessor Pin Descriptions (Continued)

Symbol	Type	Name and Function
DACK1:0#	O	**DMA Acknowledge 1 and 0** signal to an external device that the processor has acknowledged the corresponding DMA request and is relinquishing the bus. DACK1# is multiplexed with TXD1, and DACK0# is multiplexed with CS5#.
D/C#	O	**Data/Control** indicates whether the current bus cycle is a data cycle (memory or I/O read or write) or a control cycle (interrupt acknowledge, halt, or code fetch).
DCD1:0#	I	**Data Carrier Detect SIO1 and SIO0** indicate that the modem or data set has detected the corresponding asynchronous serial channel's data carrier. DCD1# is multiplexed with DRQ0, and DCD0# is multiplexed with P1.0.
DRQ1:0	I	**DMA External Request 1 and 0** indicate that a peripheral requires DMA service. DRQ1 is multiplexed with RXD1, and DRQ0 is multiplexed with DCD1#.
DSR1:0#	I	**Data Set Ready SIO1 and SIO0** indicate that the modem or data set is ready to establish a communication link with the corresponding asynchronous serial channel. DSR1# is multiplexed with STXCLK, and DSR0# is multiplexed with P1.3.
DTR1:0#	O	**Data Terminal Ready SIO1 and SIO0** indicate that the corresponding asynchronous serial channel is ready to establish a communication link with the modem or data set. DTR1# is multiplexed with SRXCLK, and DTR0# is multiplexed with P1.2.
EOP#	I/OD	**End of Process** indicates that the processor has reached terminal count during a DMA transfer. An external device can also pull this pin low. EOP# is multiplexed with CTS1#.
ERROR#	I	**Error** indicates that the math coprocessor has an error condition. ERROR# is multiplexed with TMROUT2.
FLT#	I	**Float** forces all bidirectional and output signals except TDO to a high-impedance state.
HLDA	O	**Bus Hold Acknowledge** indicates that the processor has surrendered control of its local bus to another bus master. HLDA is multiplexed with P1.7.
HOLD	I	**Bus Hold Request** allows another bus master to request control of the local bus. HLDA active indicates that bus control has been granted. HOLD is multiplexed with P1.6.
INT7:0	I	**Interrupt Requests** are maskable inputs that cause the CPU to suspend execution of the current program and then execute an interrupt acknowledge cycle. They are multiplexed as follows: INT7 with TMRGATE1, INT6 with TMRCLK1, INT5 with TMRGATE0, INT4 with TMRCLK0, and INT3:0 with P3.5:2.
LBA#	O	**Local Bus Access** is asserted whenever the processor provides the READY# signal to terminate a bus transaction. This occurs when an internal peripheral address is accessed or when the chip-select unit provides the READY# signal. LOCK# O Bus Lock prevents other bus masters from gaining control of the system bus. LOCK# is multiplexed with P1.5.
M/IO#	O	**Memory/IO Indicates** whether the current bus cycle is a memory cycle or an I/O cycle. When M/IO# is high, the bus cycle is a memory cycle; when M/IO# is low, the bus cycle is an I/O cycle.
NA#	I	**Next Address** requests address pipelining.

Table 3. Intel386™ EX Microprocessor Pin Descriptions (Continued)

Symbol	Type	Name and Function
NMI	ST	**Nonmaskable Interrupt Request** is a non-maskable input that causes the CPU to suspend execution of the current program and execute an interrupt acknowledge cycle.
PEREQ	I	**Processor Extension Request** indicates that the math coprocessor has data to transfer to the processor. PEREQ is multiplexed with TMRCLK2.
P1.7:0	I/O	**Port 1, Pins 7:0** are multipurpose bidirectional port pins. They are multiplexed as follows: P1.7 with HLDA, P1.6 with HOLD, P1.5 with LOCK#, P1.4 with RI0#, P1.3 with DSR0#, P1.2 with DTR0#, P1.1 with RTS0#, and P1.0 with DCD0#.
P2.7:0	I/O	**Port 2, Pins 7:0** are multipurpose bidirectional port pins. They are multiplexed as follows: P2.7 with CTS0#, P2.6 with TXD0, P2.5 with RXD0, and P2.4:0 with CS4:0#.
P3.7:0	I/O	**Port 3, Pins 7:0** are multipurpose bidirectional port pins. They are multiplexed as follows: P3.7 with COMCLK, P3.6 with PWRDOWN, P3.5:2 with INT3:0, and P3.1:0 with TMROUT1:0.
PWRDOWN	O	**Powerdown** indicates that the processor is in powerdown mode. PWRDOWN is multiplexed with P3.6.
RD#	O	**Read Enable** indicates that the current bus cycle is a read cycle.
READY#	I/O	**Ready** indicates that the current bus transaction has completed. An external device or an internal signal can drive READY#. Internally, the chip-select wait-state logic can generate the ready signal and drive the READY# pin active.
RESET	ST	**Reset** suspends any operation in progress and places the processor into a known reset state.
REFRESH#	O	**Refresh** indicates that the current bus cycle is a refresh cycle. REFRESH# is multiplexed with CS6#.
RI1:0#	I	**Ring Indicator SIO1 and SIO0** indicate that the modem or data set has received a telephone ringing signal. RI1# is multiplexed with SSIORX, and RI0# is multiplexed with P1.4.
RTS1:0#	O	**Request-to-send SIO1 and SIO0** indicate that corresponding asynchronous serial channel is ready to exchange data with the modem or data set. RTS1# is multiplexed with SSIOTX, and RTS0# is multiplexed with P1.1.
RXD1:0	I	**Receive Data SIO1 and SIO0** accept serial data from the modem or data set to the corresponding asynchronous serial channel. RXD1 is multiplexed with DRQ1, and RXD0 is multiplexed with P2.5.
SMI#	ST	**System Management Interrupt** invokes System Management Mode (SMM). SMI# is the highest priority external interrupt. It is latched on its falling edge and it forces the CPU into SMM upon completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# cannot interrupt LOCKed bus cycles or a currently executing SMM. If the processor receives a second SMI# while it is in SMM, it will latch the second SMI# on the SMI# falling edge. However, the processor must exit SMM by executing a resume instruction (RSM) before it can service the second SMI#.
SMIACT#	O	**System Management Interrupt Active** indicates that the processor is operating in System Management Mode (SMM). It is asserted when the processor initiates an SMM sequence and remains asserted (low) until the processor executes the resume instruction (RSM).

Table 3. Intel386™ EX Microprocessor Pin Descriptions (Continued)

Symbol	Type	Name and Function
SRXCLK	I/O	**SSIO Receive Clock** synchronizes data being accepted by the synchronous serial port. SRXCLK is multiplexed with DTR1#.
SSIORX	I	**SSIO Receive Serial Data** accepts serial data (most-significant bit first) being sent to the synchronous serial port. SSIORX is multiplexed with RI1#.
SSIOTX	O	**SSIO Transmit Serial Data** sends serial data (most-significant bit first) from the synchronous serial port. SSIOTX is multiplexed with RTS1#.
STXCLK	I/O	**SSIO Transmit Clock** synchronizes data being sent by the synchronous serial port. STXCLK is multiplexed with DSR1.
TCK	I	**TAP (Test Access Port) Controller Clock** provides the clock input for the JTAG logic.
TDI	I	**TAP (Test Access Port) Controller Data Input** is the serial input for test instructions and data.
TDO	O	**TAP (Test Access Port) Controller Data Output** is the serial output for test instructions and data.
TMRCLK2:0	I	**Timer/Counter Clock Inputs** can serve as external clock inputs for the corresponding timer/counters. (The timer/counters can also be clocked internally.) They are multiplexed as follows: TMRCLK2 with PEREQ, TMRCLK1 with INT6, and TMRCLK0 with INT4.
TMRGATE2:0	I	**Timer/Counter Gate Inputs** can control the corresponding timer/counter's counting (enable, disable, or trigger, depending on the programmed mode). They are multiplexed as follows: TMRGATE2 with BUSY#, TMRGATE1 with INT7, and TMRGATE0 with INT5.
TMROUT2:0	O	**Timer/Counter Outputs** provide the output of the corresponding timer/counter. The form of the output depends on the programmed mode. They are multiplexed as follows: TMROUT2 with ERROR#, TMROUT1 with P3.1, and TMROUT0 with P3.0.
TMS	I	**TAP (Test Access Port) Controller Mode Select** controls the sequence of the TAP controller's states.
TRST#	ST	**TAP (Test Access Port) Controller Reset** resets the TAP controller at power-up and each time it is activated.
TXD1:0	O	**Transmit Data SIO1 and SIO0** transmit serial data from the individual serial channels. TXD1 is multiplexed with DACK1#, and TXD0 is multiplexed with P2.6.
UCS#	O	**Upper Chip-select** is activated when the address of a memory or I/O bus cycle is within the address region programmed by the user.
V_{CC}	P	**System Power** provides the nominal DC supply input. Connected externally to a V_{CC} board plane.
V_{SS}	G	**System Ground** provides the 0V connection from which all inputs and outputs are measured. Connected externally to a ground board plane.
WDTOUT	O	**Watchdog Timer Output** indicates that the watchdog timer has expired.
W/R#	O	**Write/Read** indicates whether the current bus cycle is a write cycle or a read cycle. When W/R# is high, the bus cycle is a write cycle; when W/R# is low, the bus cycle is a read cycle.
WR#	O	**Write Enable** indicates that the current bus cycle is a write cycle.

3.0 FUNCTIONAL DESCRIPTION

The Intel386 EX microprocessor is a fully static, 32-bit processor optimized for embedded applications. It features low power and low voltage capabilities, integration of many commonly used DOS-type peripherals, and a 32-bit programming architecture compatible with the large software base of Intel386 processors. The following sections provide an overview of the integrated peripherals.

3.1 Clock Generation and Power Management Unit

The clock generation circuit includes a divide-by-two counter, a programmable divider for generating a prescaled clock (PSCLK), a divide-by-two counter for generating baud-rate clock inputs, and Reset circuitry. The CLK2 input provides the fundamental timing for the chip. It is divided by two internally to generate a 50% duty cycle Phase1 (PH1) and Phase 2 (PH2) for the core and integrated peripherals. For power manage-ment, separate clocks are routed to the core (PH1C/PH2C) and the peripheral modules (PH1P/PH2P).

Two Power Management modes are provided for flexible power-saving options. During Idle mode, the clocks to the CPU core are frozen in a known state (PH1C low and PH2C high), while the clocks to the peripherals continue to toggle. In Powerdown mode, the clocks to both core and peripherals are frozen in a known state (PH1C low and PH2C high). The Bus Interface Unit will not honor any DMA, DRAM refresh, or HOLD requests in Powerdown mode because the clocks to the entire device are frozen.

3.2 Chip-select Unit

The Chip-select Unit (CSU) decodes bus cycle address and status information and enables the appropriate chip-selects. The individual chip-selects become valid in the same bus state as the address and become inactive when either a new address is selected or the current bus cycle is complete.

The CSU is divided into eight separate chip-select regions, each of which can enable one of the eight chip-select pins. Each chip-select region can be mapped into memory or I/O space. A memory-mapped chip-select region can start on any 2(n+1) Kbyte address location (where n = 0–15, depending upon the mask register). An I/O-mapped chip-select region can start on any 2(n+1) byte address location (where n = 0–15, depending upon the mask register). The size of the region is also dependent upon the mask used.

3.3 Interrupt Control Unit

The Intel386 EX microprocessor's Interrupt Control Unit (ICU) contains two 8259A modules connected in a cascade mode. The 8259A modules make up the heart of the ICU. These modules are similar to the industry-standard 8259A architecture.

The Interrupt Control Unit directly supports up to eight external (INT7:0) and up to eight internal interrupt request signals. Pending interrupt requests are posted in the Interrupt Request Register, which contains one bit for each interrupt request signal. When an interrupt request is asserted, the corresponding Interrupt Request Register bit is set. The 8259A module can be programmed to recognize either an active-high level or a positive transition on the interrupt request lines. An internal Priority Resolver decides which pending interrupt request (if more than one exists) is the highest priority, based on the programmed operating mode. The Priority Resolver controls the single interrupt request line to the CPU. The Priority Resolver's default priority scheme places the master interrupt controller's IR0 as the highest priority and the master's IR7 as the lowest. The priority can be modified through software.

Besides the eight interrupt request inputs available to the Intel386 EX microprocessor, additional interrupts can be supported by cascaded external 8259A modules. Up to four external 8259A units can be cascaded to the master through connections to the INT3:0 pins. In this configuration, the interrupt acknowledge (INTA#) signal can be decoded externally using the ADS#, D/C#, W/R#, and M/IO# signals.

3.4 Timer Control Unit

The Timer Control Unit (TCU) on the Intel386 EX microprocessor has the same basic functionality as the industry-standard 82C54 counter/timer. The

TCU provides three independent 16-bit counters, each capable of handling clock inputs up to 8 MHz. This maximum frequency must be considered when programming the input clocks for the counters. Six programmable timer modes allow the counters to be used as event counters, elapsed-time indicators, programmable one-shots, and in many other applications. All modes are software programmable.

3.5 Watchdog Timer Unit

The Watchdog Timer (WDT) unit consists of a 32-bit down-counter that decrements every PH1P cycle, allowing up to 4.3 billion count intervals. The WDTOUT pin is driven high for sixteen CLK2 cycles when the down-counter reaches zero (the WDT times out). The WDTOUT signal can be used to reset the chip, to request an interrupt, or to indicate to the user that a ready-hang situation has occurred. The down-counter can also be updated with a user-defined 32-bit reload value under certain conditions. Alternatively, the WDT unit can be used as a bus monitor or as a general-purpose timer.

3.6 Asynchronous Serial I/O Unit

The Intel386 EX microprocessor's asynchronous Serial I/O (SIO) unit is a Universal Asynchronous Receiver/ Transmitter (UART). Functionally, it is equivalent to the National Semiconductor NS16450 and INS8250. The Intel386 EX microprocessor contains two full-duplex, asynchronous serial channels.

The SIO unit converts serial data characters received from a peripheral device or modem to parallel data and converts parallel data characters received from the CPU to serial data. The CPU can read the status of the serial port at any time during its operation. The status information includes the type and condition of the transfer operations being performed and any errors (parity, framing, overrun, or break interrupt).

Each asynchronous serial channel includes full modem control support (CTS#, RTS#, DSR#, DTR#, RI#, and DCD#) and is completely programmable. The programmable options include character length (5, 6, 7, or 8 bits), stop bits (1, 1.5, or 2), and parity (even, odd, forced, or none). In addition, it contains a programmable baud-rate generator capable of clock rates from 0 to 512 Kbaud.

3.7 Synchronous Serial I/O Unit

The Synchronous Serial I/O (SSIO) unit provides for simultaneous, bidirectional communications. It consists of a transmit channel, a receive channel, and a dedicated baud-rate generator. The transmit and receive channels can be operated independently (with different clocks) to provide non-lockstep, full-duplex communications; either channel can originate the clocking signal (Master Mode) or receive an externally generated clocking signal (Slave Mode).

The SSIO provides numerous features for ease and flexibility of operation. With a maximum clock input of 12.5 MHz to the baud-rate generator, the SSIO can deliver a baud rate of 5 Mbits per second. Each channel is double buffered. The two channels share the baud-rate generator and a multiply-by-two transmit and receive clock. The SSIO supports 16-bit serial communications with independently enabled transmit and receive functions and gated interrupt outputs to the interrupt controller.

3.8 Parallel I/O Unit

The Intel386 EX microprocessor has three 8-bit, general-purpose I/O ports. All port pins are bidirectional, with CMOS-level input and outputs. All pins have both a standard operating mode and a peripheral mode (a multiplexed function), and all have similar sets of control registers located in I/O address space. Ports 1 and 2 provide 8 mA of drive capability, while port 3 provides 16 mA.

3.9 DMA and Bus Arbiter Unit

The Intel386 EX microprocessor's DMA controller is a two-channel DMA; each channel operates independently of the other. Within the operation of the individual channels, several different data transfer modes are available. These modes can be combined in various configurations to provide a very versatile DMA controller. Its feature set has enhancements beyond the 8237 DMA family; however, it can be configured such that it can be used in an 8237-

like mode. Each channel can transfer data between any combination of memory and I/O with any combination (8 or 16 bits) of data path widths. An internal temporary register that can disassemble or assemble data to or from either an aligned or a nonaligned destination or source optimizes bus bandwidth.

The bus arbiter, a part of the DMA controller, works much like the priority resolving circuitry of a DMA. It receives service requests from the two DMA channels, the external bus master, and the DRAM Refresh controller. The bus arbiter requests bus ownership from the core and resolves priority issues among all active requests when bus mastership is granted.

Each DMA channel consists of three major components: the Requestor, the Target, and the Byte Count. These components are identified by the contents of programmable registers that define the memory or I/O device being serviced by the DMA. The Requestor is the device that requires and requests service from the DMA controller. Only the Requestor is considered capable of initializing or terminating a DMA process. The Target is the device with which the Requestor wishes to communicate. The DMA process considers the Target a slave that is incapable of controlling the process. The Byte Count dictates the amount of data that must be transferred.

3.10 Refresh Control Unit

The Refresh Control Unit (RCU) simplifies dynamic memory controller design with its integrated address and clock counters. Integrating the RCU into the processor allows an external DRAM controller to use chip-selects, wait state logic, and status lines.

The Intel386 EX microprocessor's RCU consists of four basic functions. First, it provides a programmable-interval timer that keeps track of time. Second, it provides the bus arbitration logic to gain control of the bus to run refresh cycles. Third, it contains the logic to generate row addresses to refresh DRAM rows individually. And fourth, it contains the logic to signal the start of a refresh cycle.

Additionally, it contains a 13-bit address counter that forms the refresh address, supporting DRAMs with up to 13 rows of memory cells (13 refresh address bits). This includes all practical DRAM sizes for the Intel386 EX microprocessor's 64 Mbyte address space.

3.11 JTAG Test-logic Unit

The JTAG Test-logic Unit provides access to the device pins and to a number of other testable areas on the device. It is fully compliant with the IEEE 1149.1 standard and thus interfaces with five dedicated pins: TRST#, TCK, TMS, TDI, and TDO. It contains the Test Access Port (TAP) finite-state machine, a 4-bit instruction register, a 32-bit identification register, and a single-bit bypass register. The test-logic unit also contains the necessary logic to generate clock and control signals for the Boundary Scan chain.

Since the test-logic unit has its own clock and reset signals, it can operate autonomously. Thus, while the rest of the microprocessor is in Reset or Powerdown, the JTAG unit can read or write various register chains.

4.0 DESIGN CONSIDERATIONS

This section describes the Intel386 EX microprocessor's instruction set and its component and revision identifiers.

4.1 Instruction Set

The Intel386 EX microprocessor uses the same instruction set as the Intel386 SX microprocessor with the following exceptions.

The Intel386 EX microprocessor has one new instruction (RSM). This Resume instruction causes the processor to exit System Management Mode (SMM). RSM requires 338 clocks per instruction (CPI).

The Intel386 EX microprocessor requires more clock cycles than the Intel386 SX microprocessor to execute some instructions. Table 5 lists these instructions and the Intel386 EX microprocessor CPI. For

the equivalent Intel386 SX microprocessor CPI, refer to the "Instruction Set Clock Count Summary" table in the Intel386™ SX Microprocessor data sheet (order number 240187).

4.2 Component and Revision Identifiers

To assist users, the microprocessor holds a component identifier and revision identifier in its DX register after reset. The upper 8 bits of DX hold the component identifier, 23H. (The lower nibble, 3H, identifies the Intel386 architecture, while the upper nibble, 2H, identifies the second member of the Intel386 microprocessor family.)

The lower 8 bits of DX hold the revision level identifier. The revision identifier will, in general, chronologically track those component steppings that are intended to have certain improvements or distinction from previous steppings. The revision identifier will track that of the Intel386 CPU whenever possible. However, the revision identifier value is not guaranteed to change with every stepping revision or to follow a completely uniform numerical sequence, depending on the type or intent of the revision or the manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component. The initial revision identifier for the Intel386 EX microprocessor is 09H.

4.3 Package Thermal Specifications

The Intel386 EX microprocessor is specified for operation with case temperature (T_{CASE}) within the range of 0°C to 100°C. The case temperature can be measured in any environment to determine whether the microprocessor is within the specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

An increase in the ambient temperature (T_A) causes a proportional increase in the case temperature (T_{CASE}) and the junction temperature (T_J). A packaged device produces thermal resistance between junction and case temperatures (θ_{JC}) and between junction and ambient temperatures (θ_{JA}). The relationships between the temperature and thermal resistance parameters are expressed by these equations (P = power dissipated as heat = $V_{CC} \times I_{CC}$):

1. $T_J = T_{CASE} + P \times \theta_{JC}$
2. $T_A = T_J - P \times \theta_{JA}$
3. $T_{CASE} = T_A + P \times [\theta_{JA} - \theta_{JC}]$

A safe operating temperature can be calculated from equation 1 by using the maximum safe T_C of 100°C, the maximum power drawn by the chip in the specific design, and the θ_{JC} value from Table 4. The θ_{JA} value depends on the airflow (measured at the top of the chip) provided by the system ventilation. The θ_{JA} values are given for reference only and are not guaranteed.

Table 4. Thermal Resistances (0°C/W) θ_{JA}, θ_{JC}

Pkg	θ_{JC}	θ_{JA} versus Airflow (ft/min)		
		0	100	200
132 PQFP	6	41	36	32
144 TQFP	5	36	31	27

Figure 4 and Figure 5 provide examples of ambient temperature performance as a function of frequency. Each graph indicates a low, typical, and high power case for each package type. The high power case reflects high current consumption - i.e. many active peripherals, heavily loaded outputs, etc. The low power case reflects a condition similar to what would be seen with the device held in reset.

It should be noted that these graphs are only examples of thermal performance and not guaranteed values. Actual thermal performance may vary based on system design and should be calculated using actual power dissipation and the thermal resistance values given in Table 4.

Table 5. Intel386™ EX Microprocessor Clocks Per Instruction

Instruction	Clock Count		
	Virtual 8086 Mode (Note 1)	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode (Note 3)
POPA		28	35
IN:			
Fixed Port	27	14	7/29
Variable Port	28	15	8/29
OUT:			
Fixed Port	27	14	7/29
Variable Port	28	15	9/29
INS	30	17	9/32
OUTS	31	18	10/33
REP INS	31 + 6n (Note 2)	17 + 6n (Note 2)	10 + 6n\32 + 6n (Note 2)
REP OUTS	30 + 8n (Note 2)	16 + 8n (Note 2)	10 + 8n\31 + 8n (Note 2)
HLT		7	7
MOV C0, reg		10	10

NOTES:
1. The clock count values in this column apply if I/O permission allows I/O to the port in virtual 8086 mode. If the I/O bit map denies permission, exception fault 13 occurs; see clock counts for the INT 3 instruction in the "Instruction Set Clock Count Summary" table in the Intel386™ SX Microprocessor data sheet (order number 240187).
2. n = the number of times repeated.
3. When two clock counts are listed, the smaller value refers to a register operand and the larger value refers to a memory operand.

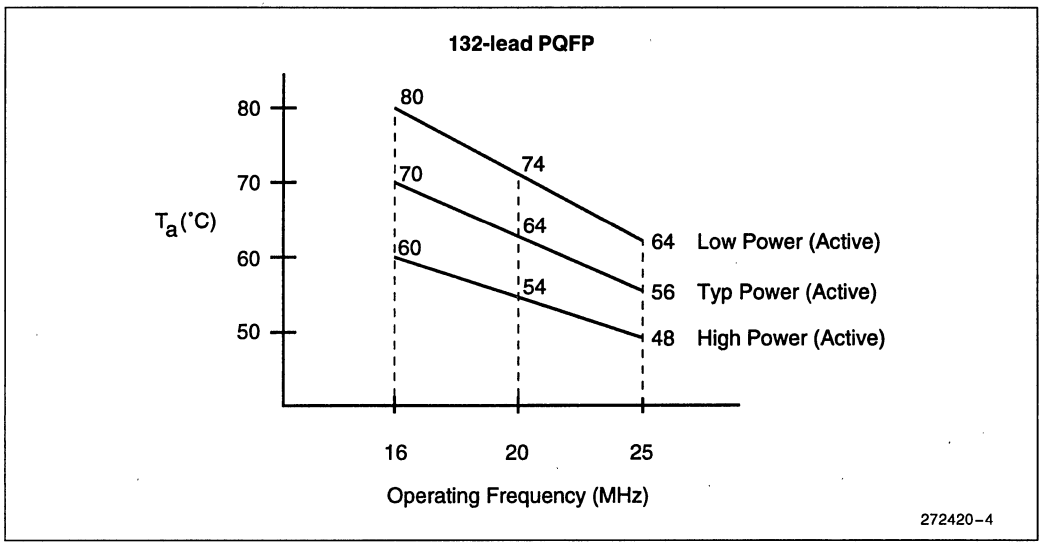


Figure 4. Ambient Temperature vs. Frequency for High, Low, and Typical Power Values (132-lead PQFP, V_{CC} = 5.0V nominal)

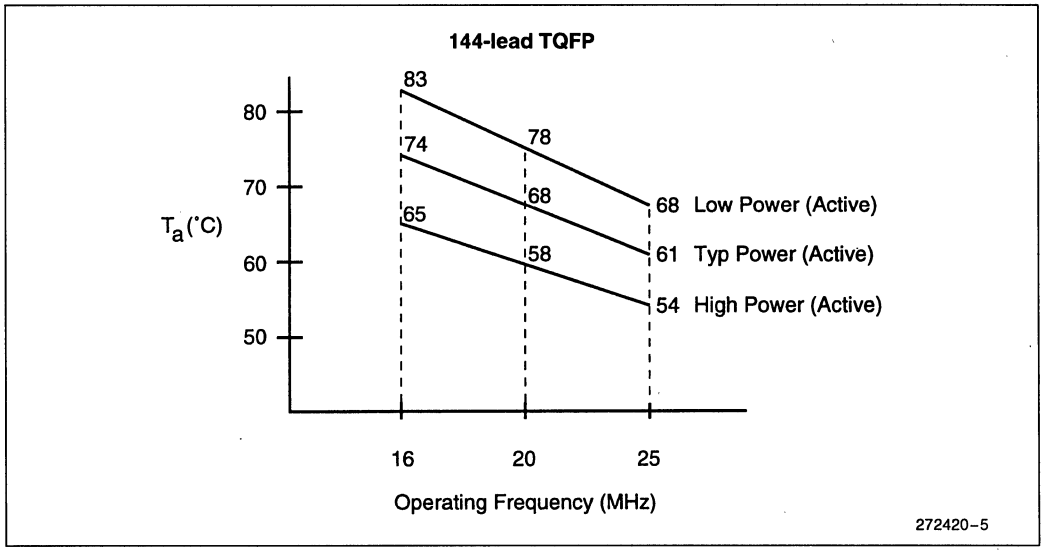


Figure 5. Ambient Temperature vs. Frequency for High, Low, and Typical Power Values (144-lead TQFP, V_{CC} = 5.0V nominal)

5.0 DC SPECIFICATIONS

NOTICE: This document contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

ABSOLUTE MAXIMUM RATINGS*

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

Case Temperature Under Bias... −65°C to +110°C

Supply Voltage with Respect to V_{SS} ..... −0.5V to 6.5V

Voltage on Other Pins ..... −0.5V to V_{CC} + 0.5V

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

OPERATING CONDITIONS*

V_{CC} (Digital Supply Voltage) .....2.7V to 5.5 V

T_{CASE} (Case Temperature Under Bias) .....0°C to 100°C

F_{OSC} (Operating Frequency) .....0 MHz to 25 MHz

Table 6. DC Characteristics

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.5	0.3 V_{CC}	V	
V_{IH}	Input High Voltage	0.7 V_{CC}	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage				V_{CC} = 4.5V to 5.5V
	All pins except Port 3		0.40	V	I_{OL} = 8 mA
	Port 3		0.40	V	I_{OL} = 16 mA
V_{OL1}	Output Low Voltage				V_{CC} = 2.7V to 3.6V
	All pins except Port 3		0.40	V	I_{OL} = 4 mA
	Port 3		0.40	V	I_{OL} = 8 mA
V_{OH}	Output High Voltage				V_{CC} = 4.5V to 5.5V
	All pins except Port 3	V_{CC} − 0.8		V	I_{OH} = −8 mA
	Port 3	V_{CC} − 0.8		V	I_{OH} = −16 mA
V_{OH1}	Output High Voltage				V_{CC} = 2.7V to 3.6V
	All pins except Port 3	V_{CC} − 0.6		V	I_{OH} = −4 mA
	Port 3	V_{CC} − 0.6		V	I_{OH} = −8 mA
I_{LI}	Input Leakage Current		±15	μA	$0 \le V_{IN} \le V_{CC}$
I_{LO}	Output Leakage Current		±15	μA	$0.45V \le V_{OUT} \le V_{CC}$
I_{CC}	Supply Current		100	mA	16 MHz, 3.3V
			130	mA	20 MHz, 3.6V
			250	mA	25 MHz, 5.5V
I_{IDLE}	Idle Mode Current		35	mA	16 MHz, 3.3V
			45	mA	20 MHz, 3.6V
			85	mA	25 MHz, 5.5V
I_{PD}	Powerdown Current		100	μA	
C_S	Pin Capacitance (any pin to V_{SS})		10	pF	

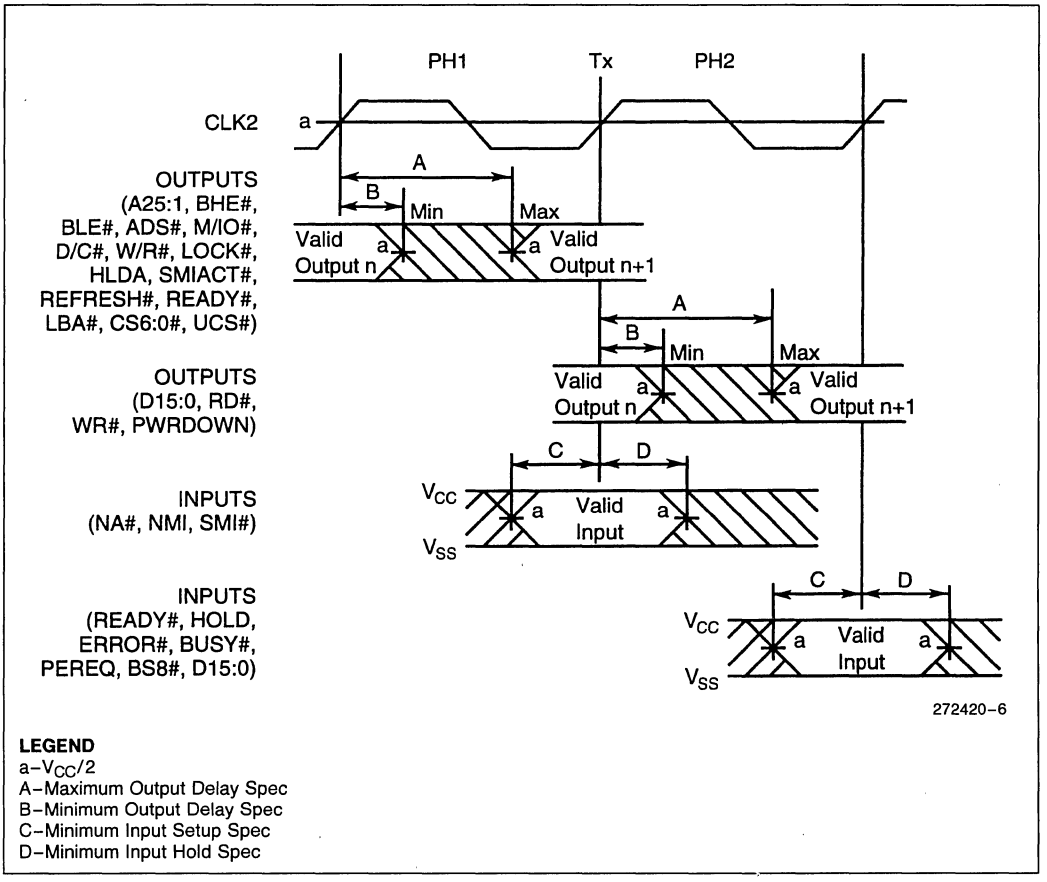
6.0 AC SPECIFICATIONS

Table 7 lists output delays, input setup requirements, and input hold requirements. All AC specifications are relative to the CLK2 rising edge crossing the $V_{CC}/2$ level.

Figure 6 shows the measurement points for AC specifications. Inputs must be driven to the indicated voltage levels when AC specifications are measured. Output delays are specified with minimum and maximum limits measured as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs ADS#, W/R#, CS5:0#, UCS#, D/C#, M/IO#, LOCK#, BHE#, BLE#, REFRESH#/CS6#, READY#, LBA#, A25:1, HLDA and SMIACT# change only at the beginning of phase one. D15:0 (write cycles) and PWRDOWN change only at the beginning of phase two. RD# and WR# change to their active states at the beginning of phase two, and to their inactive states (end of cycle) at the beginning of phase one.

The READY#, HOLD, BUSY#, ERROR#, PEREQ, BS8#, and D15:0 (read cycles) inputs are sampled at the beginning of phase one. The NA#, SMI#, and NMI inputs are sampled at the beginning of phase two.



LEGEND
a–$V_{CC}/2$
A–Maximum Output Delay Spec
B–Minimum Output Delay Spec
C–Minimum Input Setup Spec
D–Minimum Input Hold Spec

Figure 6. Drive Levels and Measurement Points for AC Specifications

3

Table 7. AC Characteristics

Symbol	Parameter	25 MHz 4.5V to 5.5V		20 MHz 3.0V to 3.6V		16 MHz 2.7V to 3.3V		Test Condition (Note 1)
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
	Operating Frequency	0	25	0	20	0	16	one-half CLK2 frequency in MHz (Note 2)
t1	CLK2 Period	20		25	31			
t2a	CLK2 High Time	7		8		9		at $V_{CC}/2$ (Note 3)
t2b	CLK2 High Time	4		5		5		at V_{CC} − 0.8V for HV, at V_{CC} − 0.6V for LV (Note 3)
t3a	CLK2 Low Time	7		8		9		at $V_{CC}/2$ (Note 3)
t3b	CLK2 Low Time	5		6		7		at 0.8V (Note 3)
t4	CLK2 Fall Time		7		8		8	V_{CC} − 0.8V to 0.8V for HV, V_{CC} − 0.6V to 0.8V for LV (Note 3)
t5	CLK2 Rise Time		7		8		8	0.8V to V_{CC} − 0.8V for HV, 0.8V to V_{CC} − 0.6V for LV (Note 3)
t6	A25:1 Valid Delay	4	29	4	38	4	42	C_L = 50 pF (Note 4)
t7	A25:1 Float Delay	4	36	4	38	4	46	(Note 5)
t8	BHE#, BLE#, LOCK# Valid Delay	4	29	4	36	4	42	C_L = 50 pF (Note 4)

NOTES:
1. Throughout this table, HV refers to devices operating with V_{CC} = 4.5V to 5.5V. LV refers to devices operating with V_{CC} = 2.7V to 3.6V.
2. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
3. These are not tested. They are guaranteed by characterization.
4. Tested with C_L set at 50 pF. For LV devices, the t6 and t12 timings are guaranteed by design characterization with C_L set at 120 pF and all other Note 4 timings are guaranteed with C_L set at 75 pF.
5. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
6. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.
7. These specifications are for information only and are not tested. They are intended to assist the designer in selecting memory speeds. For each wait state in the design add two CLK2 cycles to the specification.

Table 7. AC Characteristics (Continued)

Symbol	Parameter	25 MHz 4.5V to 5.5V		20 MHz 3.0V to 3.6V		16 MHz 2.7V to 3.3V		Test Condition (Note 1)
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
t8a	SMIACT# Valid Delay	4	29	4	36	4	44	C_L = 50 pF (Note 4)
t9	BHE#, BLE#, LOCK# Float Delay	4	0	4	32	4	40	(Note 5)
t10	M/IO#, D/C#, W/R#, ADS#, REFRESH# Valid Delay	4	29	4	36	4	42	C_L = 50 pF (Note 4)
t10a	RD#, WR# Valid Delay	4	29	4	36	4	42	
t11	M/IO#, D/C#, W/R#, REFRESH#, ADS# Float Delay	4	39	4	39	4	44	(Note 5)
t12	D15:0 Write Data Valid Delay	4	31	4	40	4	44	C_L = 50 pF (Note 4)
t13	D15:0 Write Data Float delay	4	24	4	29	4	37	(Note 5)
t14	HLDA Valid Delay	4	27	4	36	4	41	C_L = 50 pF (Note 4)
t15	NA# Setup Time	5		7		9		
t16	NA# Hold Time	10		13		15		
t19	READY# Setup Time	9		12		19		
t19a	BS8# Setup Time	11		17		19		
t20	READY#, BS8# Hold Time	4		4		4		
t21	D15:0 Read Setup Time	7		9		9		
t22	D15:0 Read Hold Time	5		6		6		

NOTES:
1. Throughout this table, HV refers to devices operating with V_{CC} = 4.5V to 5.5V. LV refers to devices operating with V_{CC} = 2.7V to 3.6V.
2. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
3. These are not tested. They are guaranteed by characterization.
4. Tested with C_L set at 50 pF. For LV devices, the t6 and t12 timings are guaranteed by design characterization with C_L set at 120 pF and all other Note 4 timings are guaranteed with C_L set at 75 pF.
5. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
6. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.
7. These specifications are for information only and are not tested. They are intended to assist the designer in selecting memory speeds. For each wait state in the design add two CLK2 cycles to the specification.

Table 7. AC Characteristics (Continued)

Symbol	Parameter	25 MHz 4.5V to 5.5V		20 MHz 3.0V to 3.6V		16 MHz 2.7V to 3.3V		Test Condition (Note 1)
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
t23	HOLD Setup Time	9		17		26		
t24	HOLD Hold Time	3		5		5		
t25	RESET Setup Time	8		12		13		
t26	RESET Hold Time	4		4		4		
t27	NMI Setup Time	12		16		18		(Note 6)
t27a	SMI# Setup Time	12		16		18		(Note 6)
t28	NMI Hold Time	6		16		16		(Note 6)
t28a	SMI# Hold Time	6		16		16		(Note 6)
t29	PEREQ, ERROR#, BUSY# Setup Time	6		14		16		(Note 6)
t30	PEREQ, ERROR#, BUSY# Hold Time	5		5		5		(Note 6)
t31	READY# Valid Delay	4	36	4	44	4	52	
t32	READY# Float Delay	4	34	4	42	4	50	
t33	LBA# Valid Delay	4	32	4	40	4	48	
t34	CS6:0#, UCS# Valid Delay	4	39	4	48	4	54	
t41	A25:1, BHE#, BLE# Valid to WR# Low	0		0		0		
t41a	UCS#, CS6:0# Valid to WR# Low	0		0		0		
t42	A25:1, BHE#, BLE# Hold After WR# High	5		5		5		

NOTES:
1. Throughout this table, HV refers to devices operating with V_{CC} = 4.5V to 5.5V. LV refers to devices operating with V_{CC} = 2.7V to 3.6V.
2. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
3. These are not tested. They are guaranteed by characterization.
4. Tested with C_L set at 50 pF. For LV devices, the t6 and t12 timings are guaranteed by design characterization with C_L set at 120 pF and all other Note 4 timings are guaranteed with C_L set at 75 pF.
5. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
6. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.
7. These specifications are for information only and are not tested. They are intended to assist the designer in selecting memory speeds. For each wait state in the design add two CLK2 cycles to the specification.

Table 7. AC Characteristics (Continued)

Symbol	Parameter	25 MHz 4.5V to 5.5V		20 MHz 3.0V to 3.6V		16 MHz 2.7V to 3.3V		Test Condition (Note 1)
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
t42a	UCS#, CS6:0# Hold after WR# High	5		5		5		
t43	D15:0 Output Valid to WR# High	3CLK2 −27		3CLK2 −36		3CLK2 −40		(Note 7)
t44	D15:0 Output Hold After WR# High	CLK2 −10		CLK2 −10		CLK2 −10		
t45	WR# High to D15:0 Float		CLK2 +10		CLK2 +10		CLK2 +10	(Note 5)
t46	WR# Pulse Width	3CLK2 −15		3CLK2 −15		3CLK2 −15		
t47	A25:1, BHE#, BLE# Valid to D15:0 Valid		4CLK2 −36		4CLK2 −47		4CLK2 −51	(Note 7)
t47a	UCS#, CS6:0# Valid to D15-D0 Valid		4CLK2 −46		4CLK2 −57		4CLK2 −61	(Note 7)
t48	RD# Low to D15:0 Input Valid		3CLK2 −36		3CLK2 −45		3CLK2 −51	(Note 7)
t49	D15:0 Hold After RD# High	2		2		2		
t50	RD# High to D15:0 Float		10		15		18	(Note 5)
t51	A25:1, BHE#, BLE# Hold After RD# High	0		0		0		
t51a	UCS#, CS6:0# Hold after RD# High	0		0		0		
t52	RD# Pulse Width	3CLK2 −15		3CLK2 −15		3CLK2 −15		

NOTES:
1. Throughout this table, HV refers to devices operating with V_{CC} = 4.5V to 5.5V. LV refers to devices operating with V_{CC} = 2.7V to 3.6V.
2. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
3. These are not tested. They are guaranteed by characterization.
4. Tested with C_L set at 50 pF. For LV devices, the t6 and t12 timings are guaranteed by design characterization with C_L set at 120 pF and all other Note 4 timings are guaranteed with C_L set at 75 pF.
5. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
6. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.
7. These specifications are for information only and are not tested. They are intended to assist the designer in selecting memory speeds. For each wait state in the design add two CLK2 cycles to the specification.

Table 7. AC Characteristics (Continued)

Symbol	Parameter	25 MHz 4.5V to 5.5V		20 MHz 3.0V to 3.6V		16 MHz 2.7V to 3.3V		Test Condition (Note 1)
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
Synchronous Serial I/O (SSIO) Unit								
t100	STXCLK, SRXCLK Frequency (Master Mode)		CLK2/8		CLK2/8		CLK2/8	(Unit is MHz)
t101	STXCLK, SRXCLK Frequency (Slave Mode)		CLK2/4		CLK2/4		CLK2/4	(Unit is MHz; CLK2/4 or 6.25 MHZ, whichever is lesser)
t102	STXCLK, SRXCLK Low Time	3CLK2/2		3CLK2/2		3CLK2/2		
t103	STXCLK, SRXCLK High Time	3CLK2/2		3CLK2/2		3CLK2/2		
t104	STXCLK Low to SSIOTX Delay		10		15		18	
t105	SSIORX to SRXCLK High Setup Time	10		15		18		
t106	SSIORX from SRXCLK Hold Time	10		15		18		
Timer Control Unit (TCU) Inputs								
t107	TMRCLKn Frequency		8		8		8	(Unit is MHz)
t108	TMRCLKn Low	60		60		60		
t109	TMRCLKn High	60		60		60		
t110	TMRGATEn High Width	50		50		60		
t111	TMRGATEn Low Width	50		50		60		

NOTES:
1. Throughout this table, HV refers to devices operating with V_{CC} = 4.5V to 5.5V. LV refers to devices operating with V_{CC} = 2.7V to 3.6V.
2. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
3. These are not tested. They are guaranteed by characterization.
4. Tested with C_L set at 50 pF. For LV devices, the t6 and t12 timings are guaranteed by design characterization with C_L set at 120 pF and all other Note 4 timings are guaranteed with C_L set at 75 pF.
5. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
6. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.
7. These specifications are for information only and are not tested. They are intended to assist the designer in selecting memory speeds. For each wait state in the design add two CLK2 cycles to the specification.

Table 7. AC Characteristics (Continued)

Symbol	Parameter	25 MHz 4.5V to 5.5V		20 MHz 3.0V to 3.6V		16 MHz 2.7V to 3.3V		Test Condition (Note 1)
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
t112	TMRGATEn to TMRCLK Setup Time (external TMRCLK only)	10		15		18		
Timer Control Unit (TCU) Outputs								
t113	TMRGATEn Low to TMROUT Valid		36		48		52	
t114	TMRCLKn Low to TMROUT Valid		36		48		52	
Interrupt Control Unit (ICU) Inputs								
t115	D7:0 Setup Time (INTA# Cycle 2)	7		9		9		
t116	D7:0 Hold Time (INTA# Cycle 2)	5		6		6		
Interrupt Control Unit (ICU) Outputs								
t117	CLK2 High to CAS2:0 Valid		34		48		54	
DMA Unit Inputs								
t118	DREQ Setup Time (Sync Mode)	17		19		21		
t119	DREQ Hold Time (Sync Mode)	4		4		4		
t120	DREQ Setup Time (Async Mode)	10		11		11		
t121	DREQ Hold Time (Async Mode)	10		11		11		

NOTES:
1. Throughout this table, HV refers to devices operating with V_{CC} = 4.5V to 5.5V. LV refers to devices operating with V_{CC} = 2.7V to 3.6V.
2. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
3. These are not tested. They are guaranteed by characterization.
4. Tested with C_L set at 50 pF. For LV devices, the t6 and t12 timings are guaranteed by design characterization with C_L set at 120 pF and all other Note 4 timings are guaranteed with C_L set at 75 pF.
5. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
6. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.
7. These specifications are for information only and are not tested. They are intended to assist the designer in selecting memory speeds. For each wait state in the design add two CLK2 cycles to the specification.

Table 7. AC Characteristics (Continued)

Symbol	Parameter	25 MHz 4.5V to 5.5V		20 MHz 3.0V to 3.6V		16 MHz 2.7V to 3.3V		Test Condition (Note 1)
		Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	
t122	EOP# Setup Time (Sync Mode)	13		17		21		
t123	EOP# Hold Time (Sync Mode)	4		4		4		
t124	EOP# Setup Time (Async Mode)	10		11		11		
t125	EOP# Hold Time (Async Mode)	11		11		11		
DMA Unit Outputs								
t126	DACK# Output Valid Delay	4	29	4	36	4	42	
t127	EOP# Active Delay	4	30	4	40	4	46	
t128	EOP# Float Delay	4	33	4	41	4	49	(Note 5)
JTAG Test-logic Unit								
t129	TCK Frequency		10		10		10	(Unit is MHz)

NOTES:
1. Throughout this table, HV refers to devices operating with V_{CC} = 4.5V to 5.5V. LV refers to devices operating with V_{CC} = 2.7V to 3.6V.
2. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
3. These are not tested. They are guaranteed by characterization.
4. Tested with C_L set at 50 pF. For LV devices, the t6 and t12 timings are guaranteed by design characterization with C_L set at 120 pF and all other Note 4 timings are guaranteed with C_L set at 75 pF.
5. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
6. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.
7. These specifications are for information only and are not tested. They are intended to assist the designer in selecting memory speeds. For each wait state in the design add two CLK2 cycles to the specification.

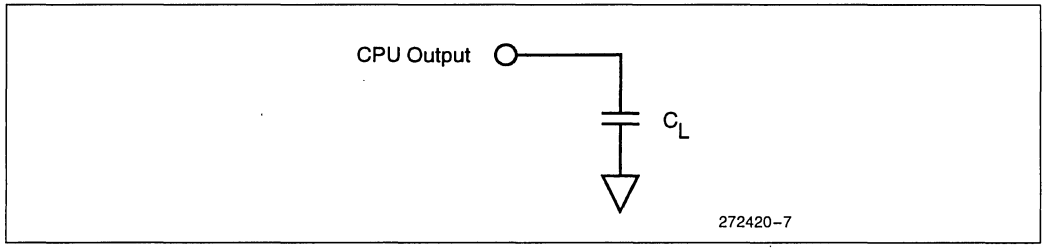


Figure 7. AC Test Loads

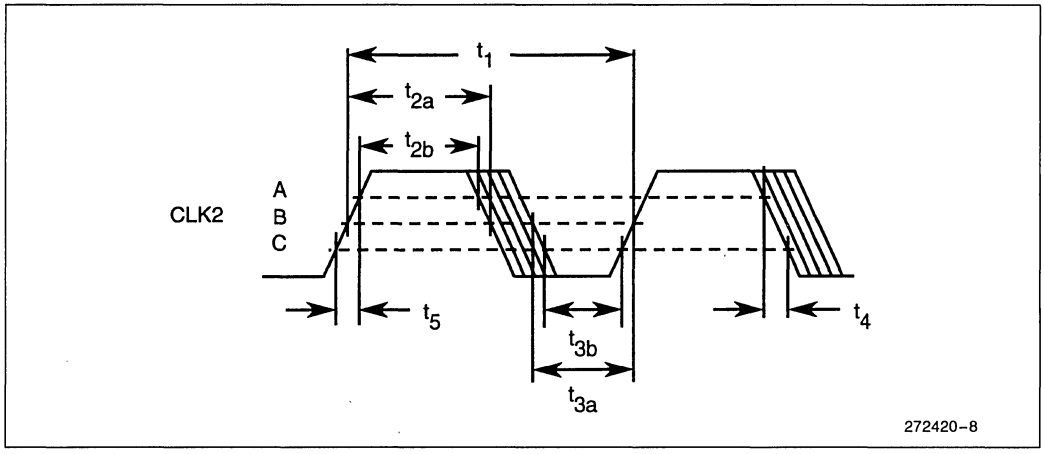


Figure 8. CLK2 Waveform

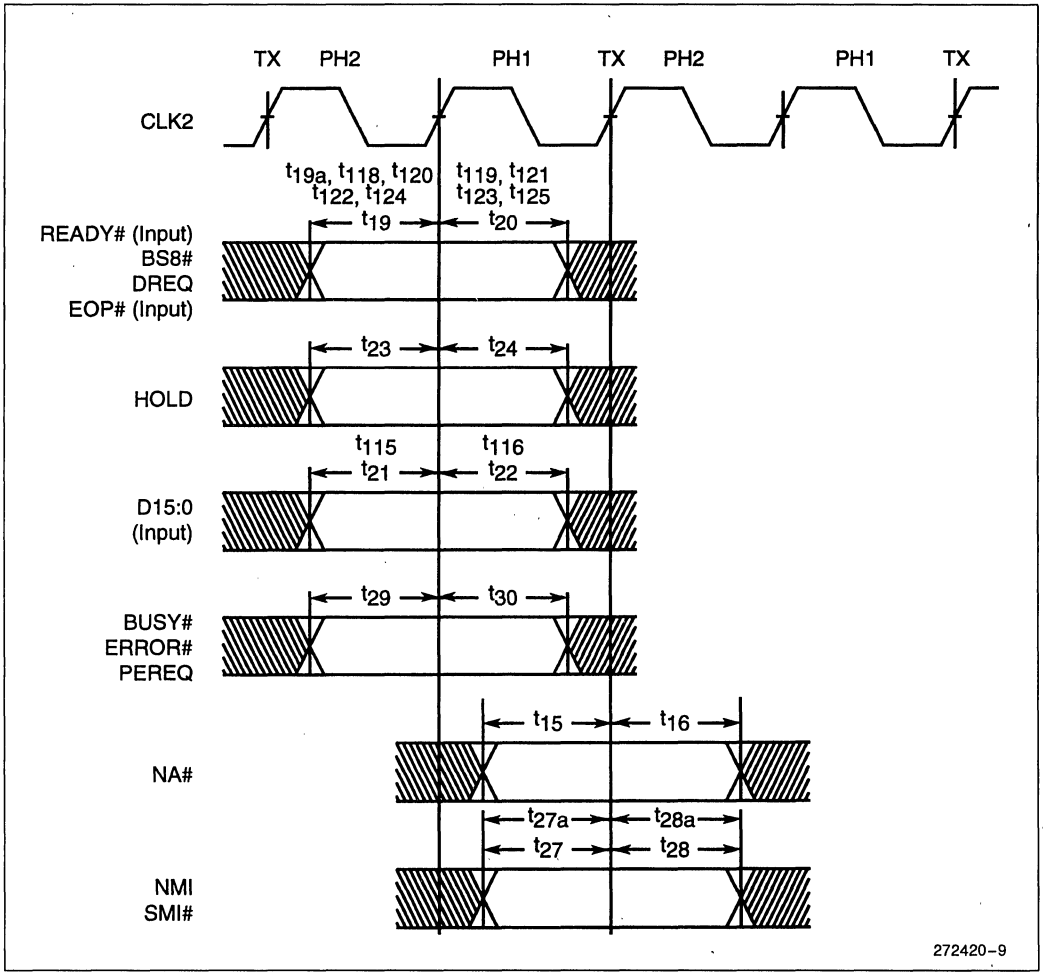


Figure 9. AC Timing Waveforms—Input Setup and Hold Timing

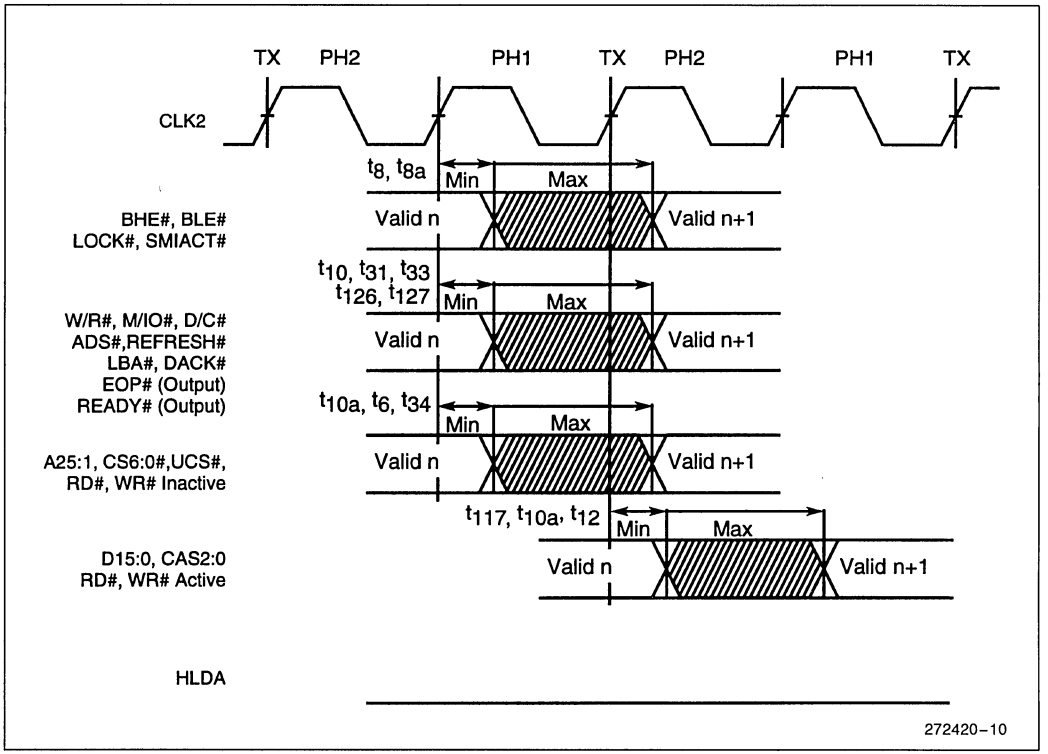


Figure 10. AC Timing Waveforms—Output Valid Delay Timing

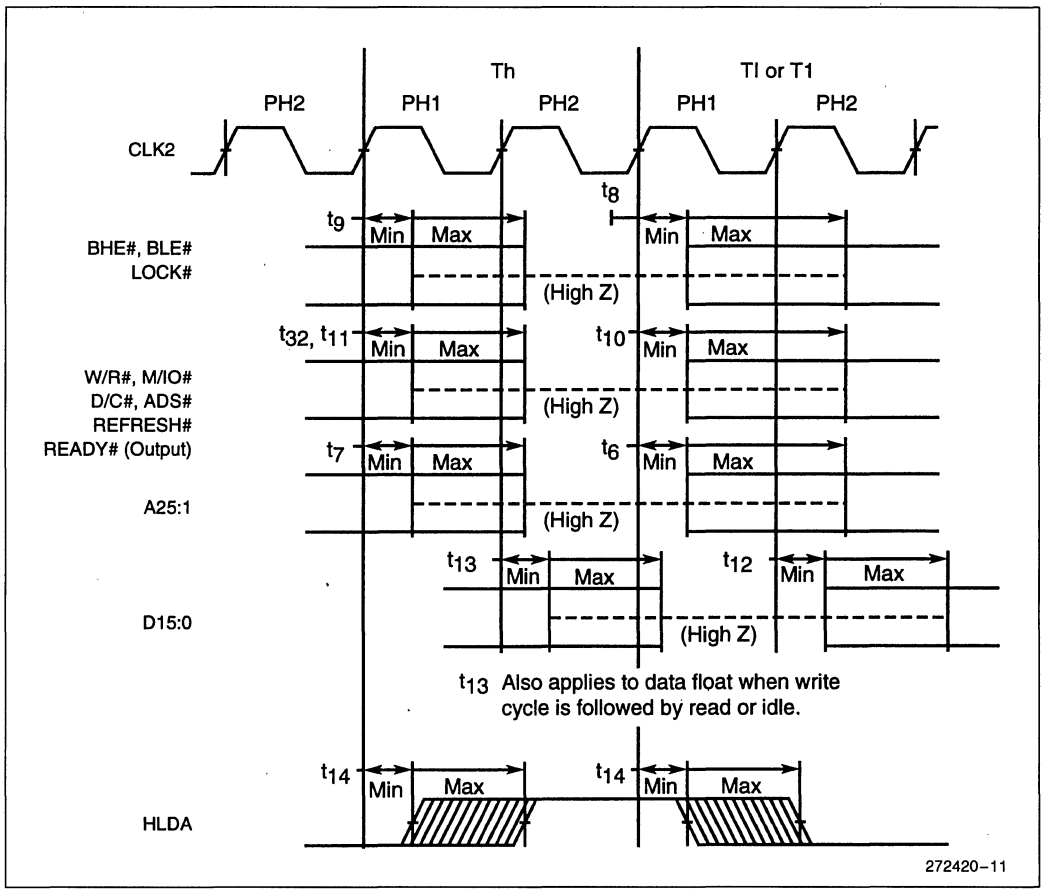


Figure 11. AC Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing

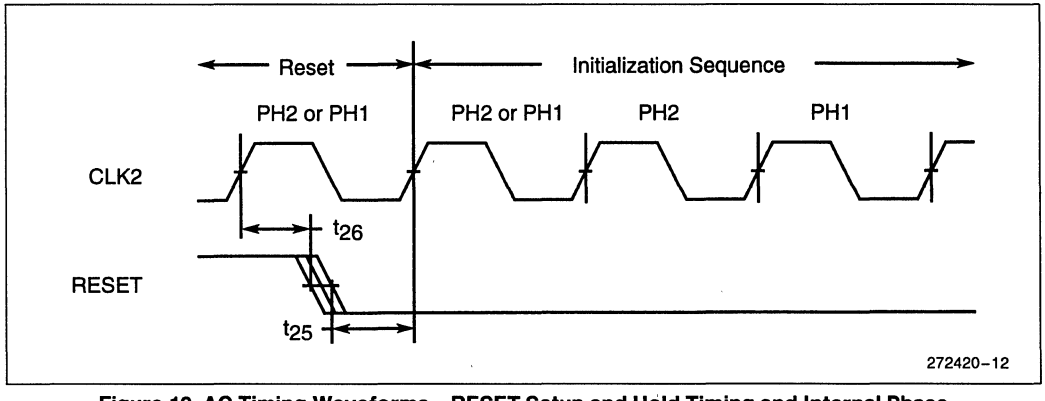


Figure 12. AC Timing Waveforms—RESET Setup and Hold Timing and Internal Phase

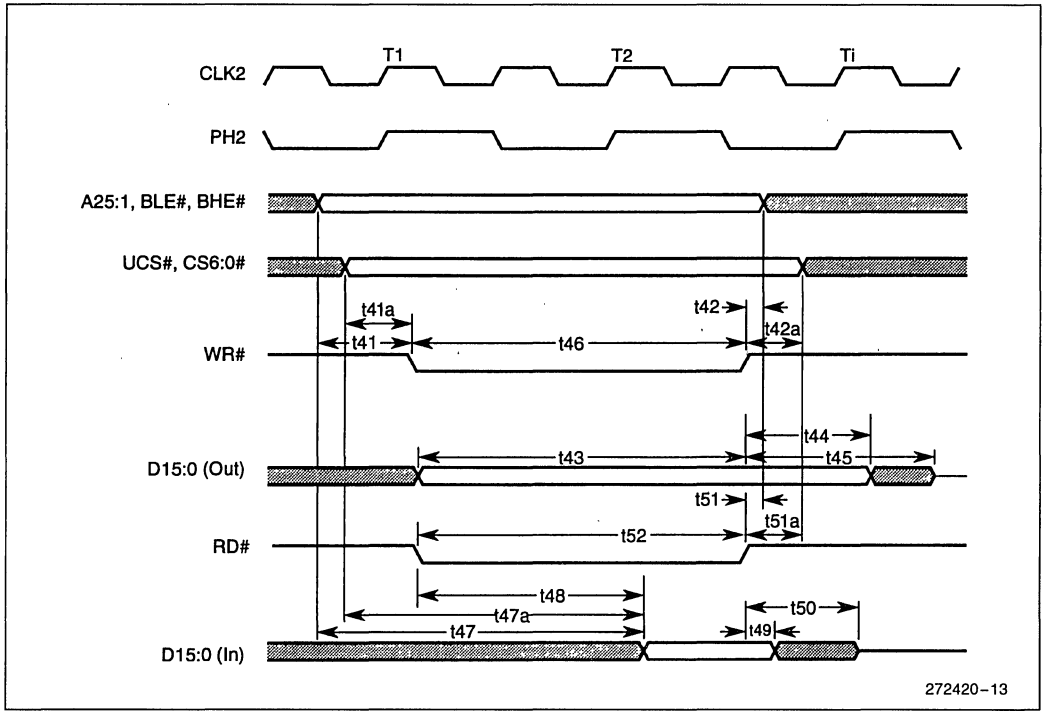


Figure 13. AC Timing Waveforms—Relative Signal Timing

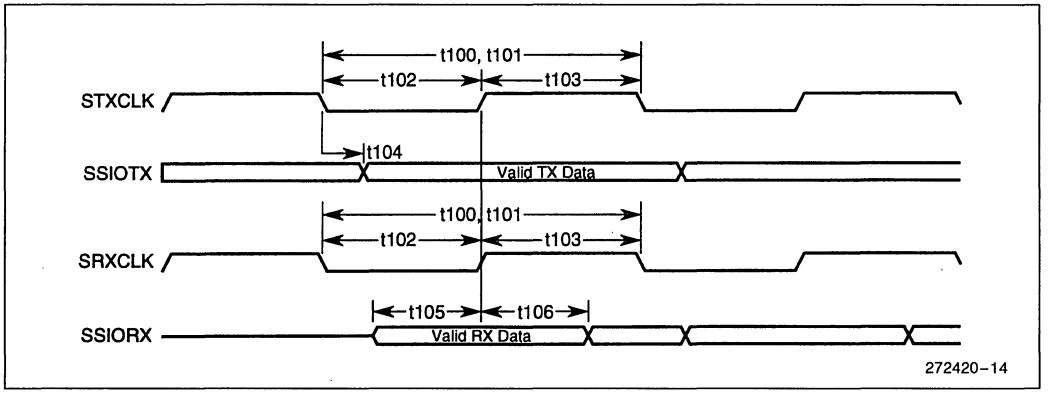


Figure 14. AC Timing Waveforms—SSIO Timing

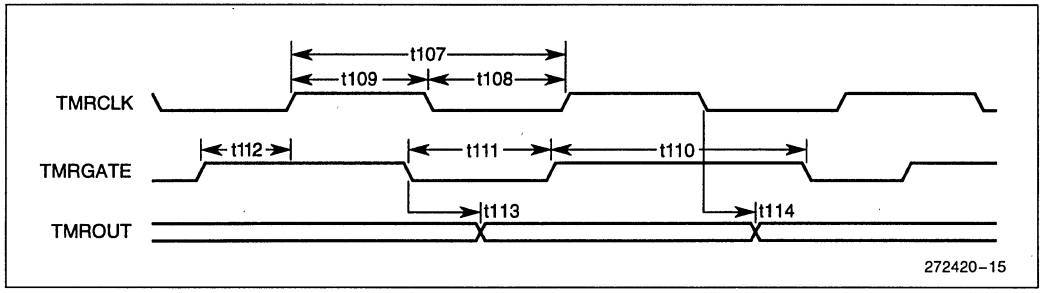


Figure 15. AC Timing Waveforms—Timer/Counter Timing

7.0 BUS CYCLE WAVEFORMS

Figures 16 through 24 present various bus cycles that are generated by the processor. What is shown in the figure is the relationship of the various bus signals to CLK2. These figures along with the information present in AC Specifications allow the user to determine critical timing analysis for a given application

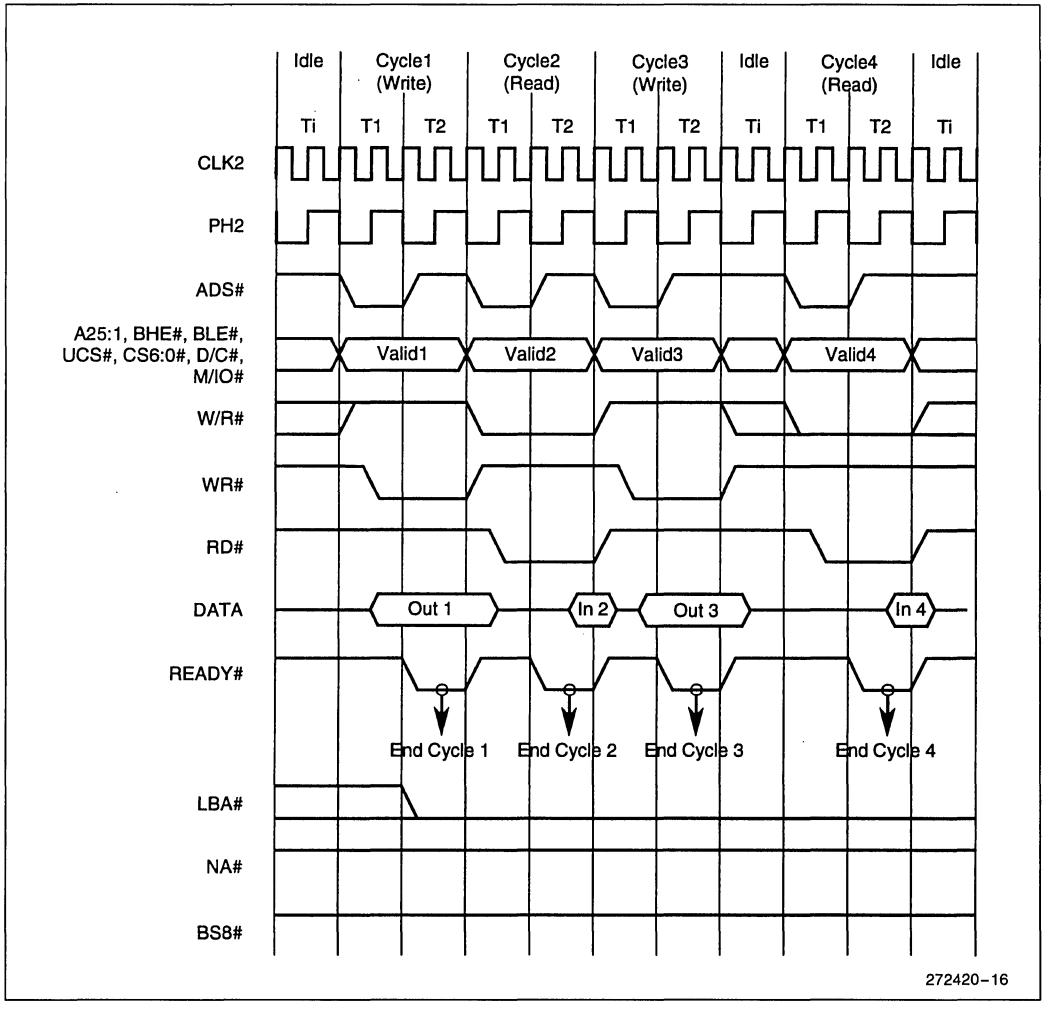


Figure 16. Local Bus Read and Write Cycles (Zero Wait States)

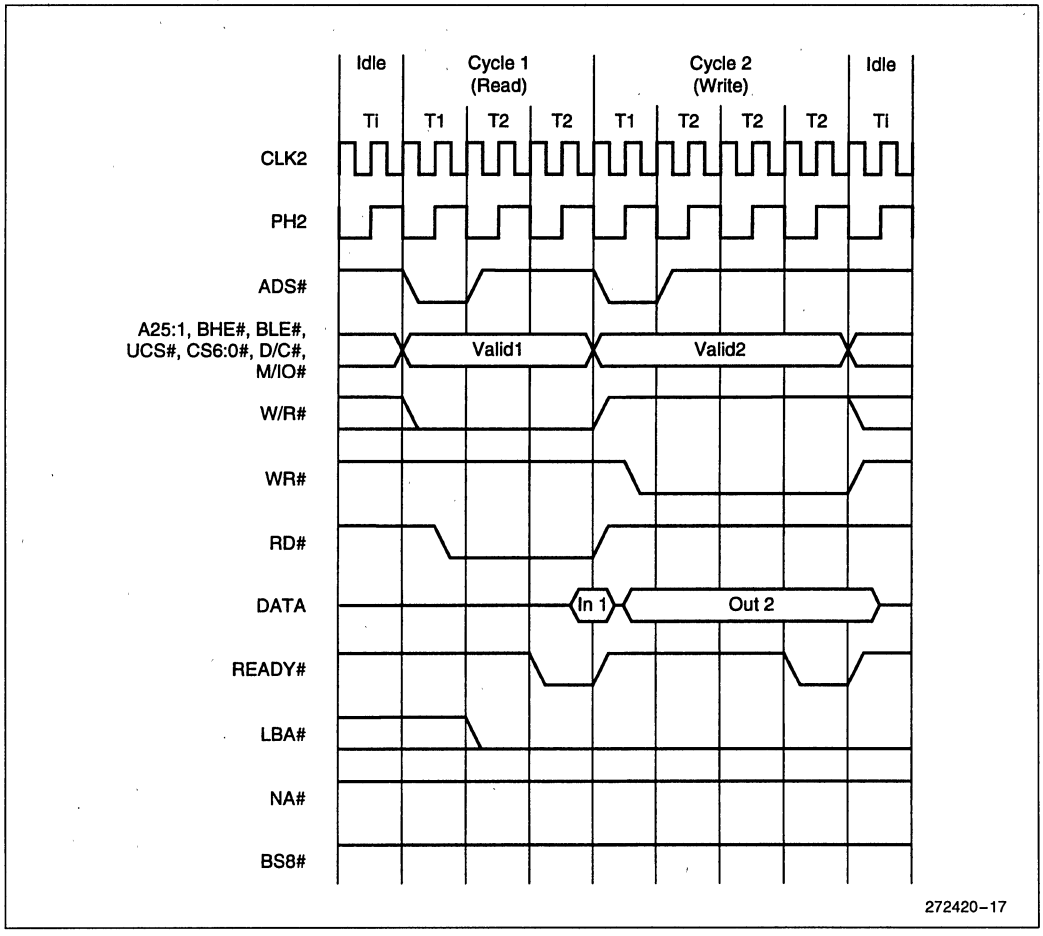


Figure 17. Local Bus Read and Write Cycles (With Wait States)

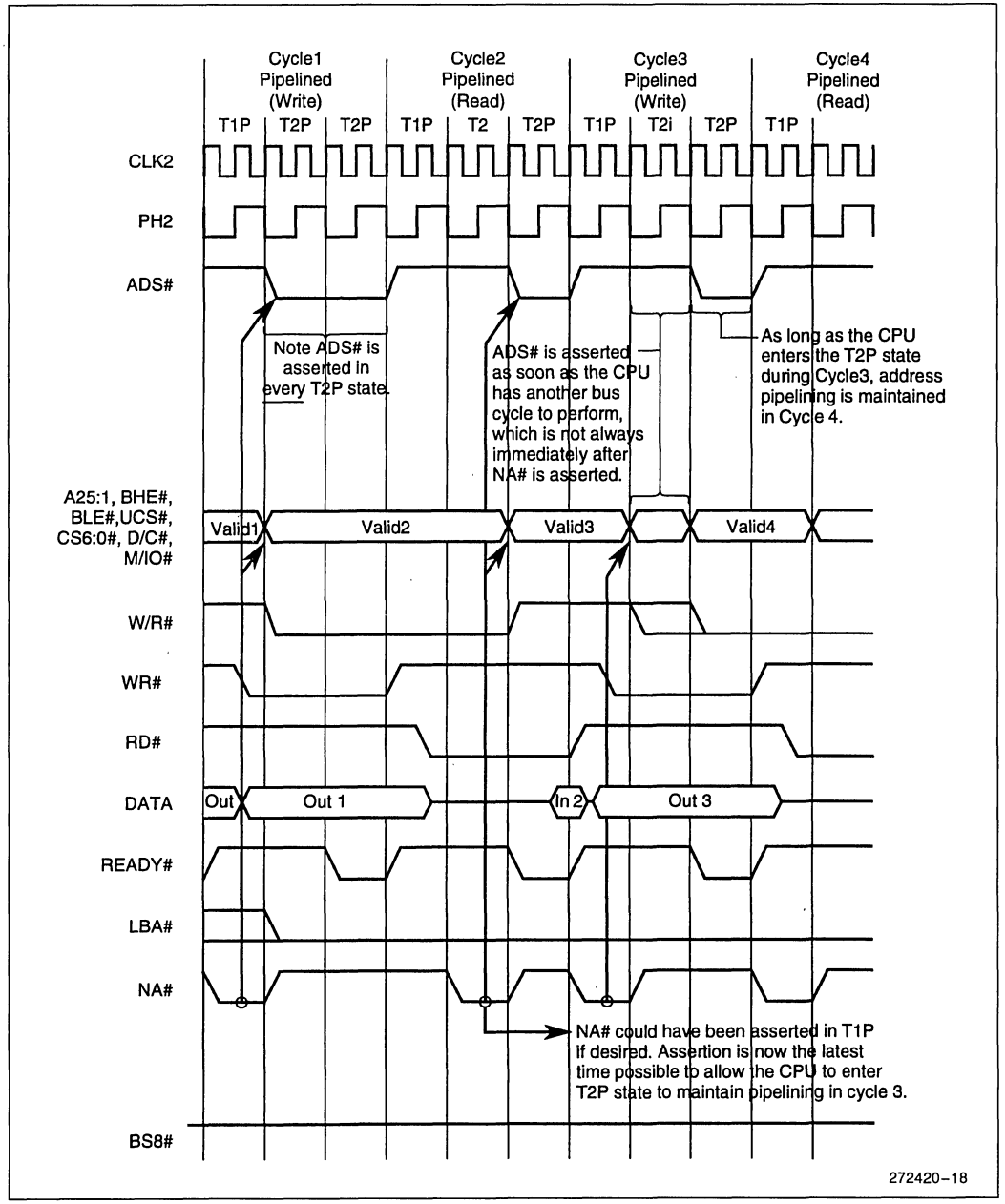


Figure 18. Pipelined Local Bus Read and Write Cycles (With Wait States)

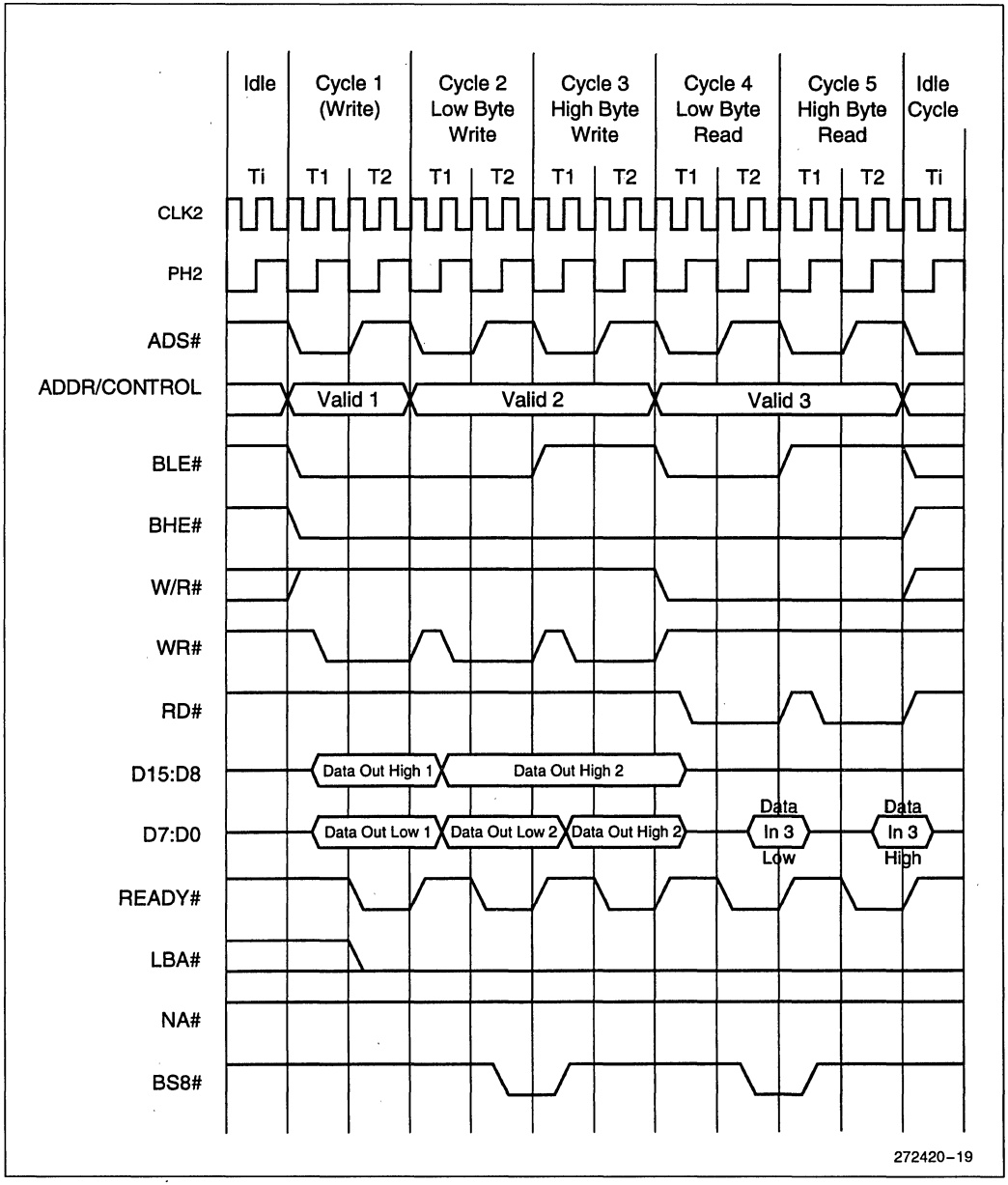


Figure 19. Local Bus Read and Write BS8# Cycles

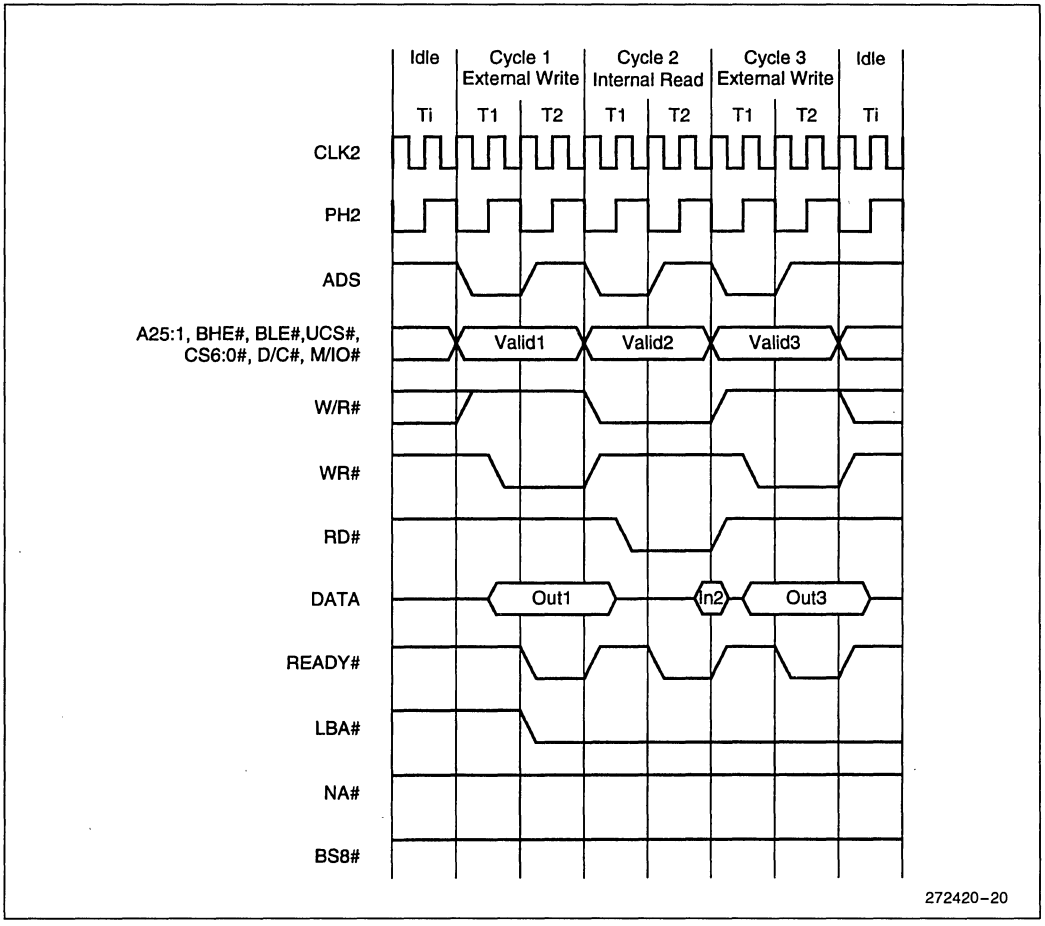


Figure 20. Local Bus Read and Write Cycles (Internal and External)

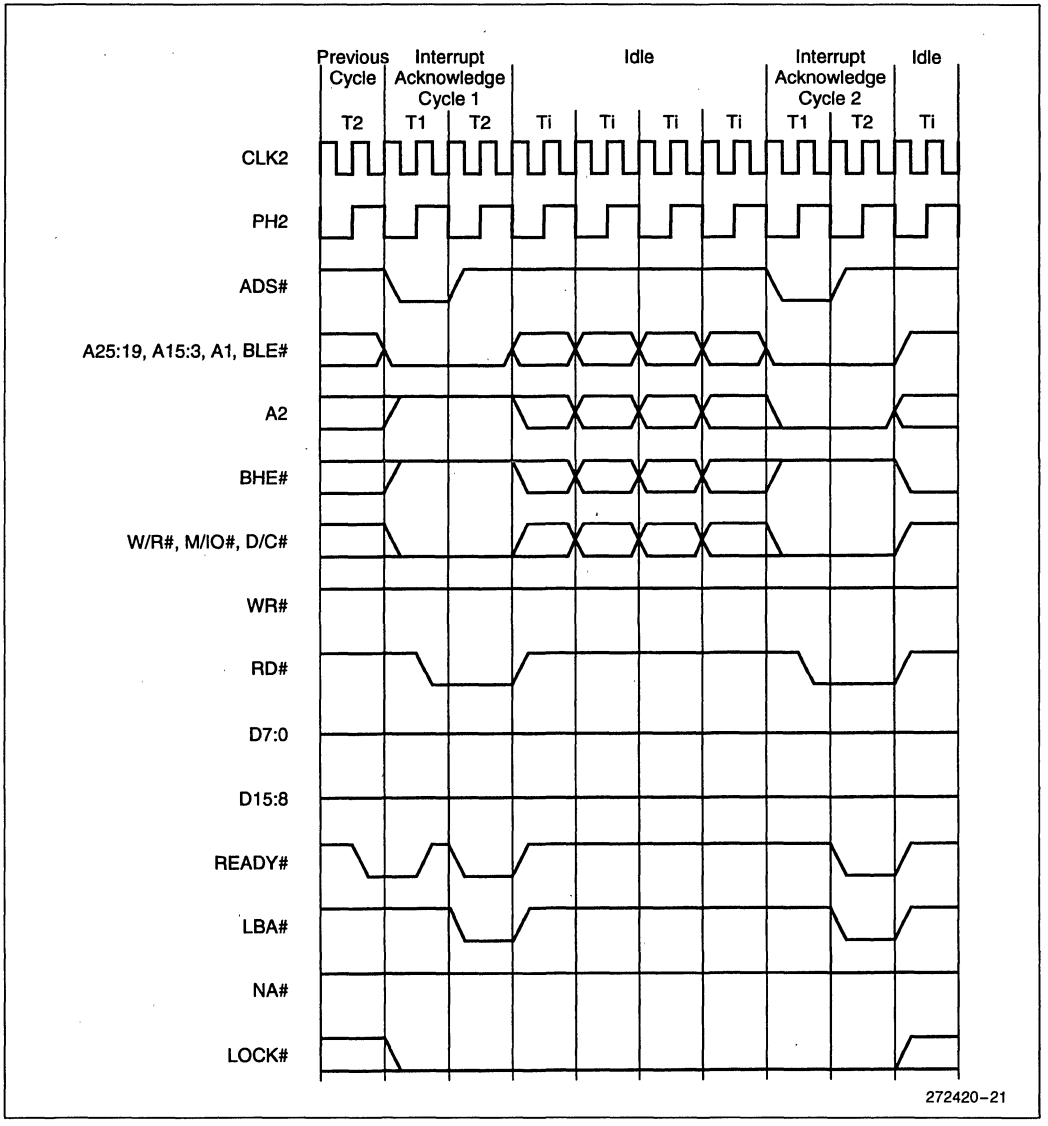


Figure 21. Local Bus Interrupt Acknowledge Cycle (Internal Cascade)

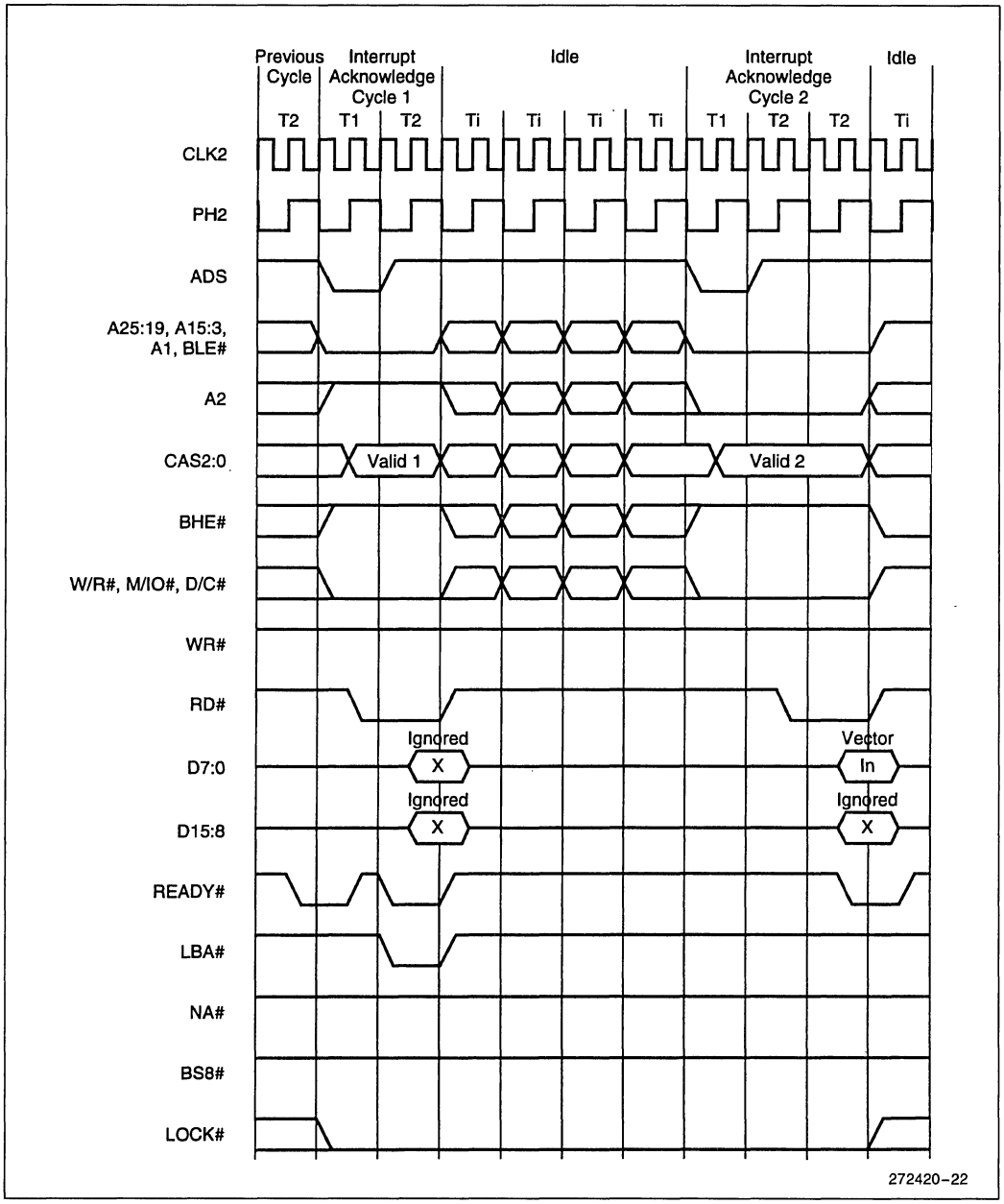


Figure 22. Local Bus Interrupt Acknowledge Cycle (External Cascade)

3

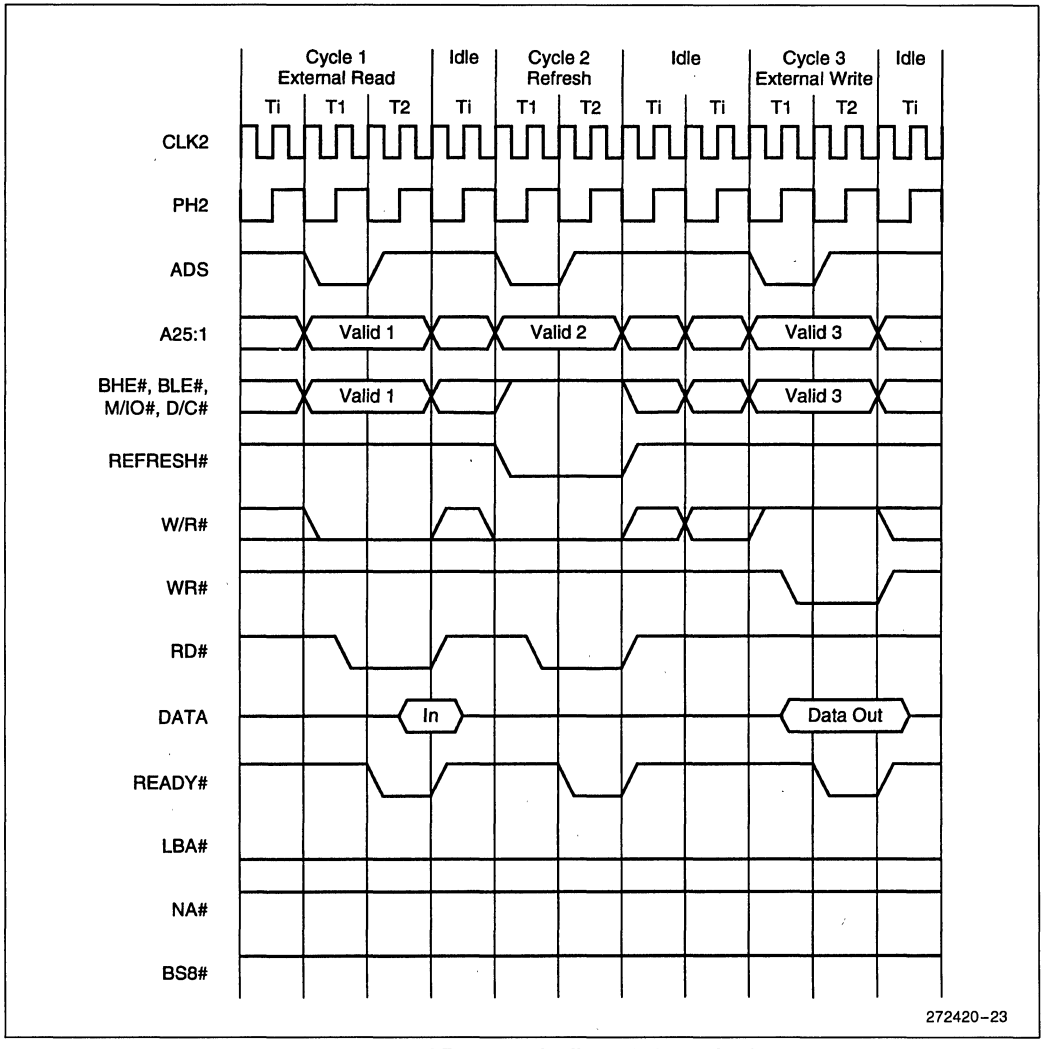


Figure 23. Local Bus Nonpipelined Refresh Cycle

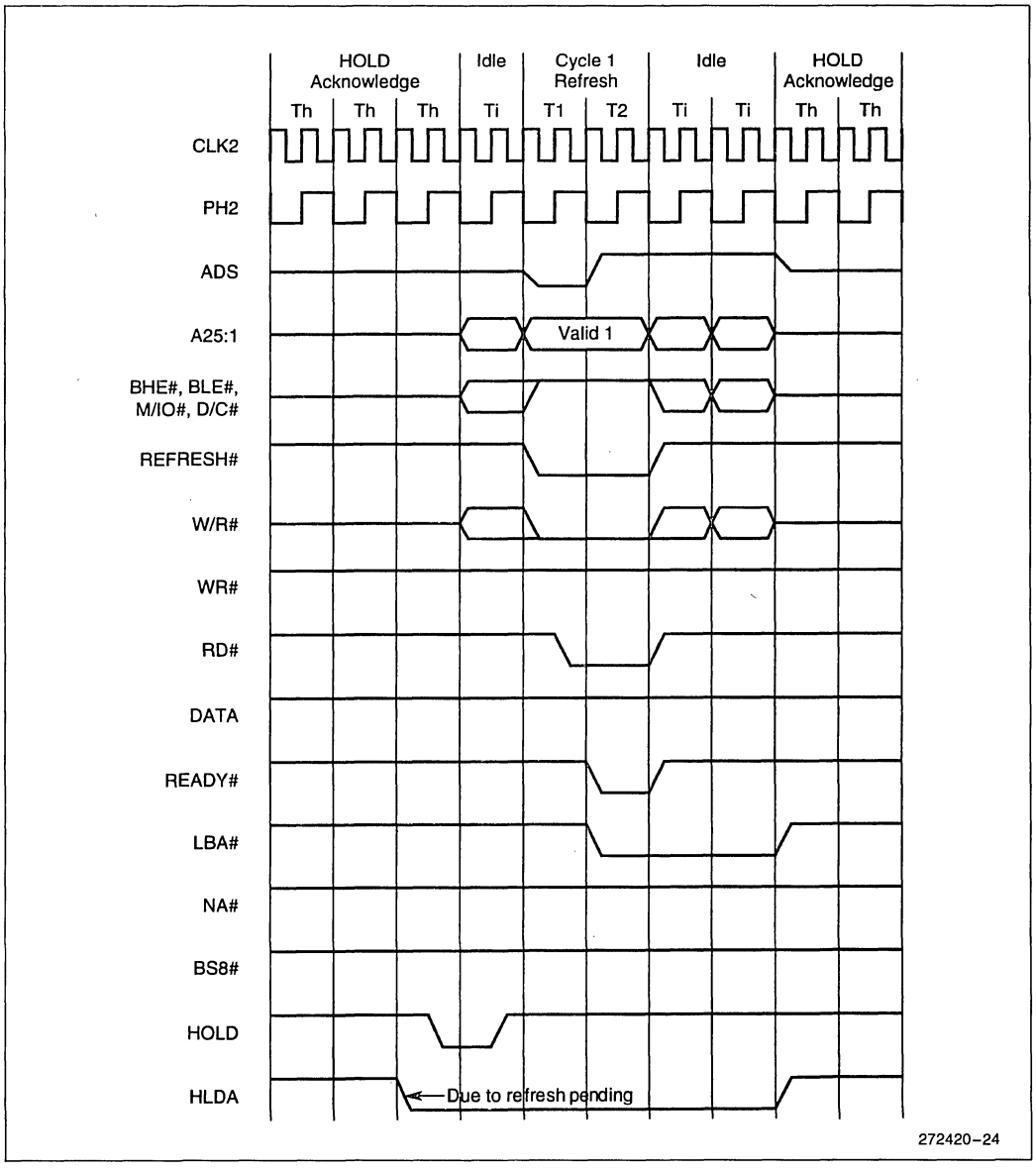


Figure 24. Local Bus Refresh Cycle During HOLD/HLDA

8.0 REVISION HISTORY

This data sheet (272420-004) is valid for devices marked with a "B" at the end of the top side tracking number. Data sheets are changed as new device information becomes available. Verify with your local Intel sales office that you have the latest version before finalizing a design or ordering devices.

This -004 data sheet contains the following changes from the -003 version.

- The -003 data sheet was valid for devices marked with an "A" at the end of the top side tracking number
- A significant number of pinout assignments have changed in the 144 pin TQFP package (Figure 3 and Table 2)
- Section 4.3 "Package Thermal Characteristics" has been added
- The separate V_{il1} and V_{il1} specs were removed
- Some corrections were added to the text describing RD#, WR#, READY# (out), and BS8# phase relationships in section 6.0
- Correction made to BS8# input in the "Drive Levels and Measurement Points for AC Specifications" figure. Also, drive levels for inputs were changed to V_{CC} and V_{SS} in this figure
- The following AC Timings were updated in Table 7:
  t6, t8a, t12, t14, t15, t34, t43, t44, t45, t46, t47, t48, t52, t115, t116, t127, t128
- The following new AC Timings were added to Table 7:
  t10a, t19a, t41a, t42a, t47a, t51a, t112, t113, t114, t118, t119, t120, t121, t122, t123, t124, t125, t126
- The following AC Timing Waveforms were updated:
  Input Setup and Hold Timing
  Output Valid Delay Timing
  Output Float Delay and HLDA Valid Delay Timing
- The following new AC Timing Waveforms were added:
  Relative Signal Timing
  SSIO Timing
  Timer/Counter Timing
- Section 7 - Bus Cycle Waveforms has been added
- This revision history has been added

Intel386™ SXSA EMBEDDED MICROPROCESSOR

- Static Intel386™ CPU Core
  - Low Power Consumption
  - Operating Power Supply
    4.5V to 5.5V—25 MHz and 33 MHz
    4.75V to 5.25V—40 MHz
  - Operating Frequency
    SA-40 = 40 MHz
    SA-33 = 33 MHz
    SA-25 = 25 MHz
- Clock Freeze Mode Allows Clock Stopping at Any Time
- Full 32-bit Internal Architecture
  - 8-, 16-, 32-bit Data Types
  - 8 General Purpose 32-bit Registers
- Runs Intel386 Architecture Software in a Cost-Effective, 16-bit Hardware Environment
  - Runs Same Applications and Operating Systems as the Intel386 SX and Intel386 DX Processors
  - Object Code Compatible with 8086, 80186, 80286, and Intel386 Processors
- TTL-Compatible Inputs
- High-Performance 16-bit Data Bus
  - Two-Clock Bus Cycles
  - Address Pipelining Allows Use of Slower, Inexpensive Memories
- Integrated Memory Management Unit (MMU)
  - Virtual Memory Support
  - Optional On-Chip Paging
  - 4 Levels of Hardware-Enforced Protection
  - MMU Fully Compatible with Those of the 80286 and Intel386 DX Processors
- Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System
- Large, Uniform Address Space
  - 16 Megabyte Physical
  - 64 Terabyte Virtual
  - 4 Gigabyte Maximum Segment Size
- Numerics Support with Intel387™ SX and Intel387 SL Math Coprocessors
- On-Chip Debugging Support Including Breakpoint Registers
- Complete System Development Support
- High-Speed CHMOS Technology
- 100-pin Plastic Quad Flatpack Package

3

The Intel386™ SXSA embedded microprocessor is a 5-volt, 32-bit, fully static CPU with a 16-bit external data bus and a 24-bit external address bus. The Intel386 SXSA CPU brings the vast software library of the Intel386 architecture to embedded systems. It provides the performance benefits of 32-bit programming with the cost savings associated with 16-bit hardware systems.

The Intel386 SXSA microprocessor is manufactured on Intel's 0.8-micron CHMOS V process. This process provides high performance and low power consumption for power-sensitive applications. Figure 3 and Figure 4 illustrate the flexibility of low power devices with respect to temperature and frequency relationships.

December 1994
Order Number: 272419-003

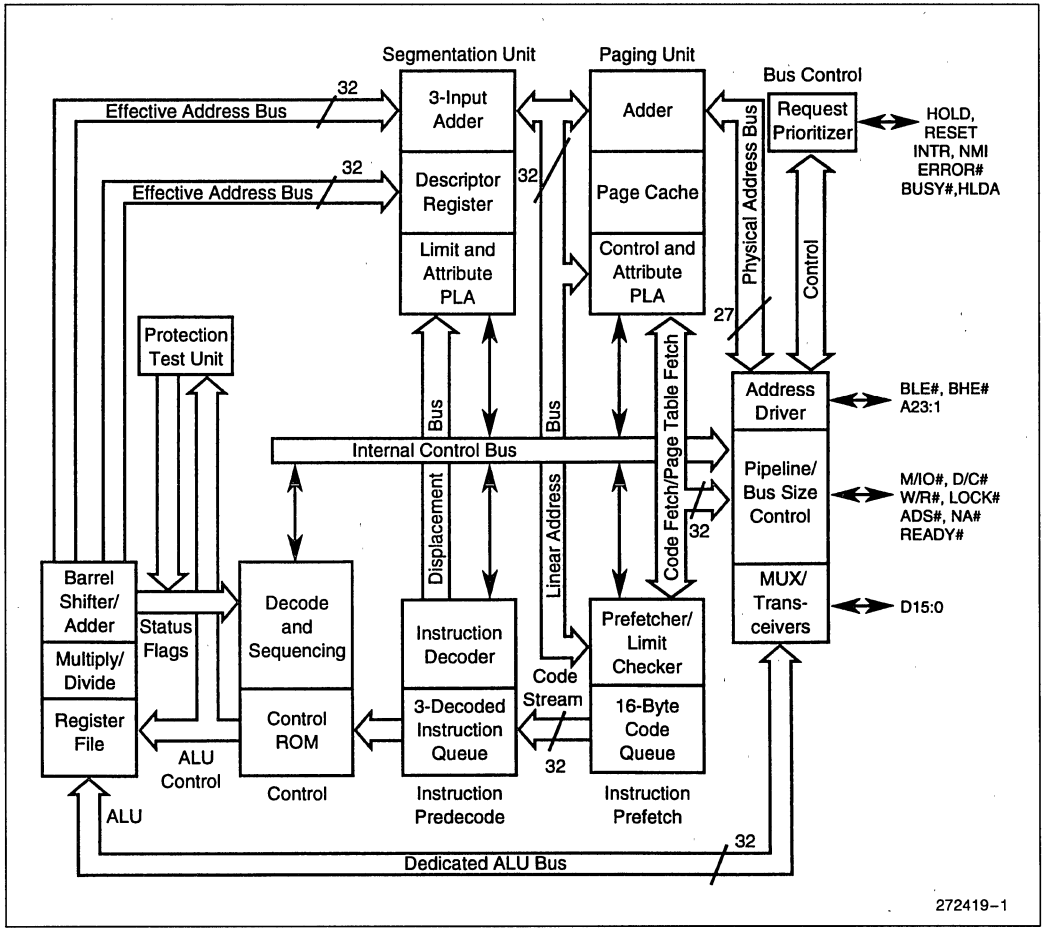
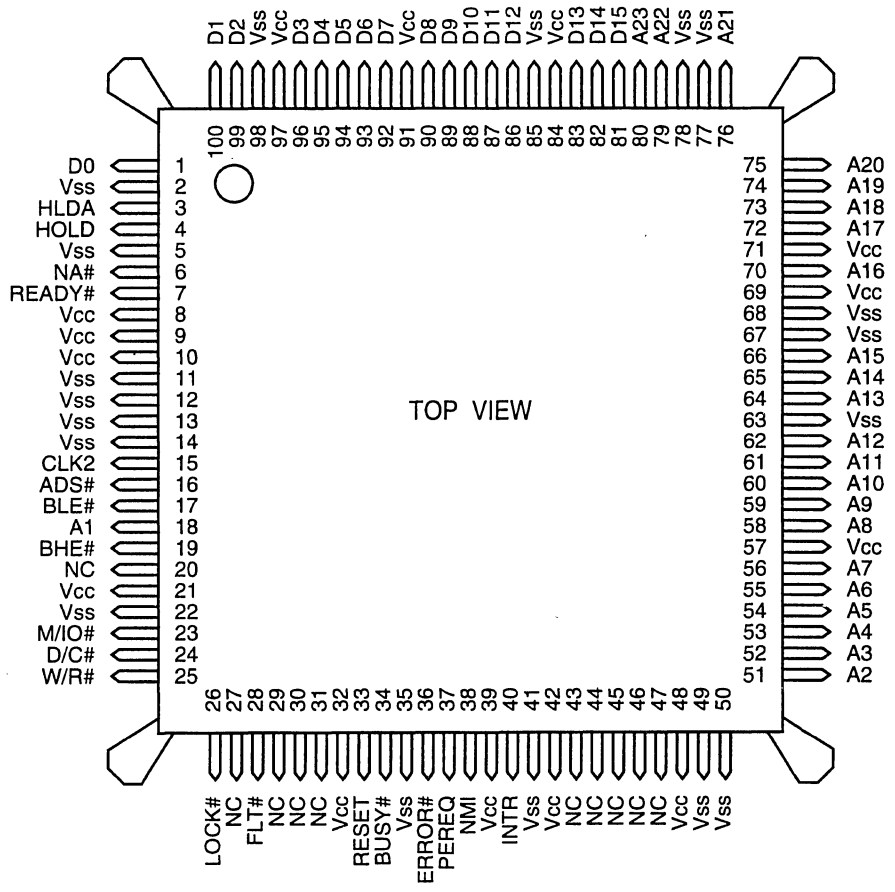


Figure 1. Intel386™ SXSA Microprocessor Block Diagram

1.0 PIN ASSIGNMENT



NOTE:
NC = No Connection

Figure 2. Intel386™ SXSA Microprocessor Pin Assignment (PQFP)

Table 1. Pin Assignment

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	D0	26	LOCK#	51	A2	76	A21
2	V_{SS}	27	NC	52	A3	77	V_{SS}
3	HLDA	28	FLT#	53	A4	78	V_{SS}
4	HOLD	29	NC	54	A5	79	A22
5	V_{SS}	30	NC	55	A6	80	A23
6	NA#	31	NC	56	A7	81	D15
7	READY#	32	V_{CC}	57	V_{CC}	82	D14
8	V_{CC}	33	RESET	58	A8	83	D13
9	V_{CC}	34	BUSY#	59	A9	84	V_{CC}
10	V_{CC}	35	V_{SS}	60	A10	85	V_{SS}
11	V_{SS}	36	ERROR#	61	A11	86	D12
12	V_{SS}	37	PEREQ	62	A12	87	D11
13	V_{SS}	38	NMI	63	V_{SS}	88	D10
14	V_{SS}	39	V_{CC}	64	A13	89	D9
15	CLK2	40	INTR	65	A14	90	D8
16	ADS#	41	V_{SS}	66	A15	91	V_{CC}
17	BLE#	42	V_{CC}	67	V_{SS}	92	D7
18	A1	43	NC	68	V_{SS}	93	D6
19	BHE#	44	NC	69	V_{CC}	94	D5
20	NC	45	NC	70	A16	95	D4
21	V_{CC}	46	NC	71	V_{CC}	96	D3
22	V_{SS}	47	NC	72	A17	97	V_{CC}
23	M/IO#	48	V_{CC}	73	A18	98	V_{SS}
24	D/C#	49	V_{SS}	74	A19	99	D2
25	W/R#	50	V_{SS}	75	A20	100	D1

2.0 PIN DESCRIPTIONS

Table 2 lists the Intel386 SXSA microprocessor pin descriptions. The following definitions are used in the pin descriptions:

The named signal is active low.
I Input signal.
O Output signal.
I/O Input and output signal.
P Power pin.
G Ground pin.

Table 2. Pin Descriptions

Symbol	Type	Pin	Name and Function
A23:1	O	80–79, 76–72, 70, 66–64, 62–58, 56–51, 18	**ADDRESS BUS** outputs physical memory or port I/O addresses.
ADS#	O	16	**ADDRESS STATUS** indicates that the processor is driving a valid bus-cycle definition and address onto its pins (W/R#, D/C#, M/IO#, BHE#, BLE#, and A23:1).
BHE#	O	19	**BYTE HIGH ENABLE** indicates that the processor is transferring a high data byte.
BLE#	O	17	**BYTE LOW ENABLE** indicates that the processor is transferring a low data byte.
BUSY#	I	34	**BUSY** indicates that the math coprocessor is busy.
CLK2	I	15	**CLK2** provides the fundamental timing for the device.
D/C#	O	24	**DATA/CONTROL** indicates whether the current bus cycle is a data cycle (memory or I/O) or a control cycle (interrupt acknowledge, halt, or code fetch). When D/C# is high, the bus cycle is a data cycle; when D/C# is low, the bus cycle is a control cycle.
D15:0	I/O	81–83, 86–90, 92–96, 99–100, 1	**DATA BUS** inputs data during memory read, I/O read, and interrupt acknowledge cycles and outputs data during memory and I/O write cycles.
ERROR#	I	36	**ERROR** indicates that the math coprocessor has an error condition.
FLT#	I	28	**FLOAT** forces all bidirectional and output signals, including HLDA, to a high-impedance state.
HLDA	O	3	**BUS HOLD ACKNOWLEDGE** indicates that the CPU has surrendered control of its local bus to another bus master.
HOLD	I	4	**BUS HOLD REQUEST** allows another bus master to request control of the local bus.
INTR	I	40	**INTERRUPT REQUEST** is a maskable input that causes the CPU to suspend execution of the current program and then execute an interrupt acknowledge cycle.
LOCK#	O	26	**BUS LOCK** prevents other system bus masters from gaining control of the system bus while it is active (low).

Table 2. Pin Descriptions (Continued)

Symbol	Type	Pin	Name and Function
M/IO#	O	23	**MEMORY/IO** indicates whether the current bus cycle is a memory cycle or an input/output cycle. When M/IO# is high, the bus cycle is a memory cycle; when M/IO# is low, the bus cycle is an I/O cycle.
NA#	I	6	**NEXT ADDRESS** requests address pipelining.
NC		20, 27, 29–31, 43–47	**NO CONNECTION** should always be left unconnected. Connecting a NC pin may cause the processor to malfunction or cause your application to be incompatible with future steppings of the device.
NMI	I	38	**NONMASKABLE INTERRUPT REQUEST** is a nonmaskable input that causes the CPU to suspend execution of the current program and execute an interrupt acknowledge function.
PEREQ	I	37	**PROCESSOR EXTENSION REQUEST** indicates that the math coprocessor has data to transfer to the processor.
READY#	I	7	**BUS READY** indicates that the current bus cycle is finished and the external device is ready to accept more data from the processor.
RESET	I	33	**RESET** suspends any operation in progress and places the processor into a known reset state.
W/R#	O	25	**WRITE/READ** indicates whether the current bus cycle is a write cycle or a read cycle. When W/R# is high, the bus cycle is a write cycle; when W/R# is low, it is a read cycle.
V_{CC}	P	8–10, 21, 32, 39, 42, 48, 57, 69, 71, 84, 91, 97	**SYSTEM POWER** provides the nominal DC supply input.
V_{SS}	G	2, 5, 11–14, 22, 35, 41, 49–50, 63, 67–68, 77–78, 85, 98	**SYSTEM GROUND** provides the 0V connection from which all inputs and outputs are measured.

3.0 DESIGN CONSIDERATIONS

This section describes the Static Intel386 SXSA microprocessor instruction set, component and revision identifier, and package thermal specifications.

3.1 Instruction Set

The Static Intel386 SXSA microprocessor uses the same instruction set as the dynamic Intel386 SX microprocessor. However, the Static Intel386 SXSA microprocessor requires more clock cycles than the dynamic Intel386 SX microprocessor to execute some instructions. Table 4 lists these instructions and the Static Intel386 SXSA microprocessor execution times. For the equivalent dynamic Intel386 SX microprocessor execution times, refer to the "Instruction Set Clock Count Summary" table in the *Intel386™ SX Microprocessor* data sheet (order number 240187).

3.2 Component and Revision Identifier

To assist users, the microprocessor holds a component identifier and revision identifier in its DX register after reset. The upper 8 bits of DX hold the component identifier, 23H. (The lower nibble, 3H, identifies the Intel386 architecture, while the upper nibble, 2H, identifies the second member of the Intel386 microprocessor family.)

The lower 8 bits of DX hold the revision level identifier. The revision identifier will, in general, chronologically track those component steppings that are intended to have certain improvements or distinction from previous steppings. The revision identifier will track that of the Intel386 CPU whenever possible. However, the revision identifier value is not guaranteed to change with every stepping revision or to follow a completely uniform numerical sequence, depending on the type or intent of the revision or the manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component. The initial revision identifier for the Static Intel386 SXSA microprocessor is 09H.

3.3 Package Thermal Specifications

Static Intel386 SXSA microprocessor is specified for operation with case temperature (T_{CASE}) as specified in the "DC SPECIFICATIONS" on page 9. The case temperature can be measured in any environment to determine whether the microprocessor is within the specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

An increase in the ambient temperature (T_A) causes a proportional increase in the case temperature (T_{CASE}) and the junction temperature (T_J). See Figure 3 and Figure 4 for case and ambient temperature relationships to frequency. A packaged device produces thermal resistance between junction and case temperatures (θ_{JC}) and between junction and ambient temperatures (θ_{JA}). The relationships between the temperature and thermal resistance parameters are expressed by these equations (P = power dissipated as heat = $V_{CC} \times I_{CC}$):

1. $T_J = T_{CASE} + P \times \theta_{JC}$
2. $T_A = T_J - P \times \theta_{JA}$
3. $T_{CASE} = T_A + P \times [\theta_{JA} - \theta_{JC}]$

A safe operating temperature can be calculated from equation 1 by using the maximum safe T_J of 115°C, the maximum power drawn by the chip in the specific design, and the θ_{JC} value from Table 3. The θ_{JA} value depends on the airflow (measured at the top of the chip) provided by the system ventilation. The θ_{JA} values are given for reference only and are not guaranteed.

Table 3. Thermal Resistances (0°C/W) θ_{JA}, θ_{JC}

Pkg	θ_{JC}	θ_{JA} vs Airflow (ft/min)		
		0	100	200
100 PQFP	5.1	46.0	44.8	41.2

Table 4. Intel386™ SXSA Microprocessor Instruction Execution Times (in Clock Counts)

Instruction	Clock Count		
	Virtual 8086 Mode (Note 1)	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode (Note 3)
POPA		28	35
IN:			
Fixed Port	27	14	7/29
Variable Port	28	15	8/29
OUT:			
Fixed Port	27	14	7/29
Variable Port	28	15	9/29
INS	30	17	9/32
OUTS	31	18	10/33
REP INS	31 + 6n (Note 2)	17 + 6n (Note 2)	10 + 6n/32 + 6n (Note 2)
REP OUTS	30 + 8n (Note 2)	16 + 8n (Note 2)	10 + 8n/31 + 8n (Note 2)
HLT		7	7
MOV C0, reg		10	10

NOTES:
1. The clock count values in this column apply if I/O permission allows I/O to the port in virtual 8086 mode. If the I/O bit map denies permission, exception fault 13 occurs; see clock counts for the INT 3 instruction in the "Instruction Set Clock Count Summary" table in the *Intel386™ SX Microprocessor* data sheet (order number 240187).
2. n = the number of times repeated.
3. When two clock counts are listed, the smaller value refers to a register operand and the larger value refers to a memory operand.

4.0 DC SPECIFICATIONS

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

ABSOLUTE MAXIMUM RATINGS*

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

Case Temperature under Bias ... −65°C to +112°C

Supply Voltage with Respect to V_{SS} ..... −0.5V to +6.5V

Voltage on Other Pins ..... −0.5V to V_{CC} + 0.5V

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

OPERATING CONDITIONS*

V_{CC} (Digital Supply Voltage— 25 MHz and 33 MHz) .....4.5V to 5.5V

V_{CC} (Digital Supply Voltage— 40 MHz) .....4.75V to 5.25V

T_{CASE} Minimum (Case Temperature under Bias) .....0°C

T_{CASE} Maximum .....see Figure 4

Operating Frequency .....0 MHz to 40 MHz

3

Table 5. DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	+0.8	V	
V_{IHC}	CLK2 Input High Voltage	V_{CC} − 0.8	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 5 mA
V_{OH}	Output High Voltage	2.4 V_{CC} − 0.5		V V	I_{OH} = −1 mA I_{OH} = −0.2 mA
I_{LI}	Input Leakage Current (for all pins except PEREQ, BUSY#, FLT#, ERROR#)		±15	μA	$0 \le V_{IN} \le V_{CC}$
I_{IH}	Input Leakage Current (PEREQ)		150	μA	V_{IH} = 2.4V (Note 1)
I_{IL}	Input Leakage Current (BUSY#, FLT#, ERROR#)		−120	μA	V_{IL} = 0.45V (Note 2)
I_{LO}	Output Leakage Current		±15	μA	$0.45V \le V_{OUT} \le V_{CC}$

NOTES:
1. PEREQ input has an internal weak pull-down resistor.
2. BUSY#, FLT# and ERROR# inputs each have an internal weak pull-up resistor.
3. I_{CC} max measurement at worst-case frequency, V_{CC}, and temperature with reset active.
4. I_{CC} typical and I_{CCF} typical are measured at nominal V_{CC} and are not fully tested.
5. Not fully tested.

Table 5. DC Characteristics (Continued)

Symbol	Parameter	Min	Max	Unit	Test Condition
I_{CC}	Supply Current				(Notes 3, 4)
	CLK2 = 80 MHz, CLK = 40 MHz		275	mA	Typical = 200 mA
	CLK2 = 66 MHz, CLK = 33 MHz		225	mA	Typical = 175 mA
	CLK2 = 50 MHz, CLK = 25 MHz		175	mA	Typical = 140 mA
I_{CCF}	Standby Current (Freeze Mode)		150	μA	Typical = 10 μA (Notes 3, 4)
C_{IN}	Input Capacitance		10	pF	F_C = 1 MHz (Note 5)
C_{OUT}	Output or I/O Capacitance		12	pF	F_C = 1 MHz (Note 5)
C_{CLK}	CLK2 Capacitance		20	pF	F_C = 1 MHz (Note 5)

NOTES:
1. PEREQ input has an internal weak pull-down resistor.
2. BUSY#, FLT# and ERROR# inputs each have an internal weak pull-up resistor.
3. I_{CC} max measurement at worst-case frequency, V_{CC}, and temperature with reset active.
4. I_{CC} typical and I_{CCF} typical are measured at nominal V_{CC} and are not fully tested.
5. Not fully tested.

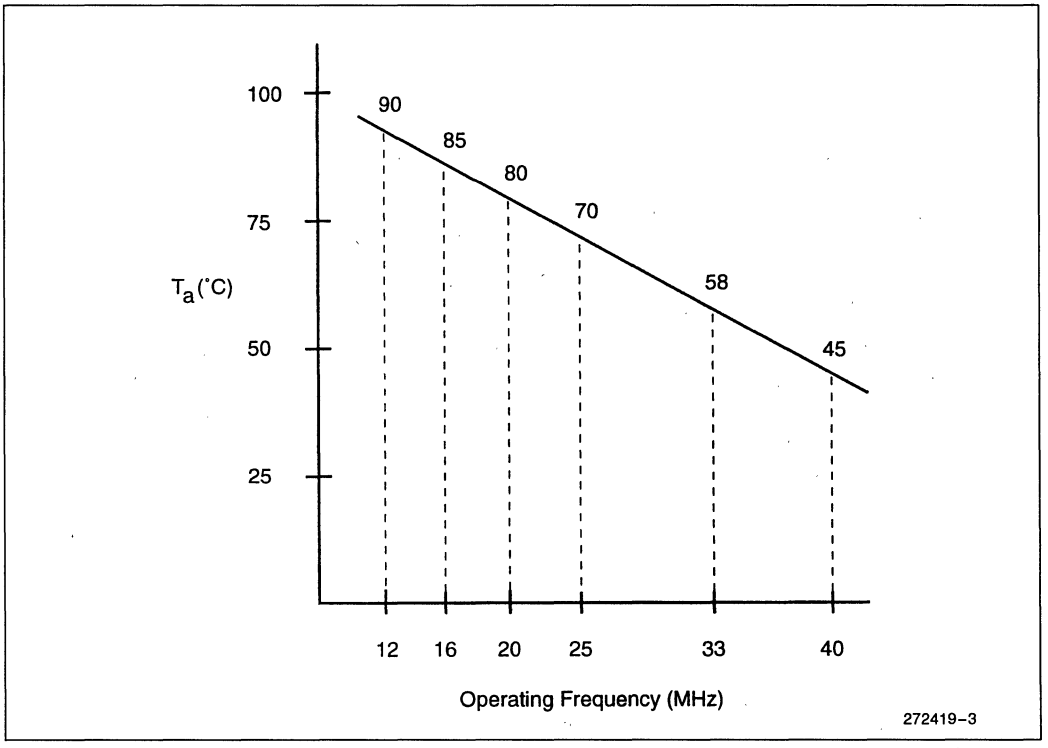


Figure 3. Ambient Temperature vs Frequency at Zero Air Flow and T_J = 115°C

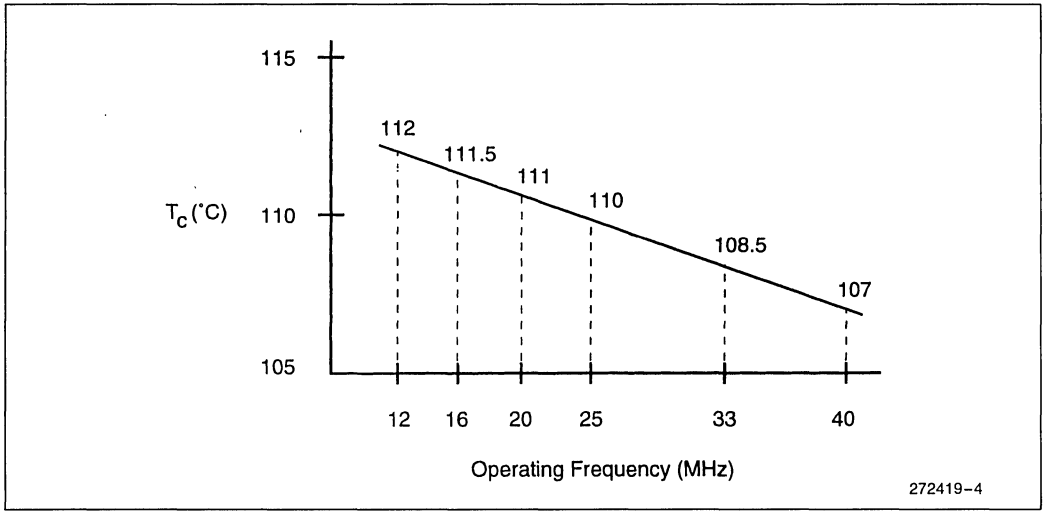


Figure 4. Case Temperature vs Frequency at T_J = 115°C

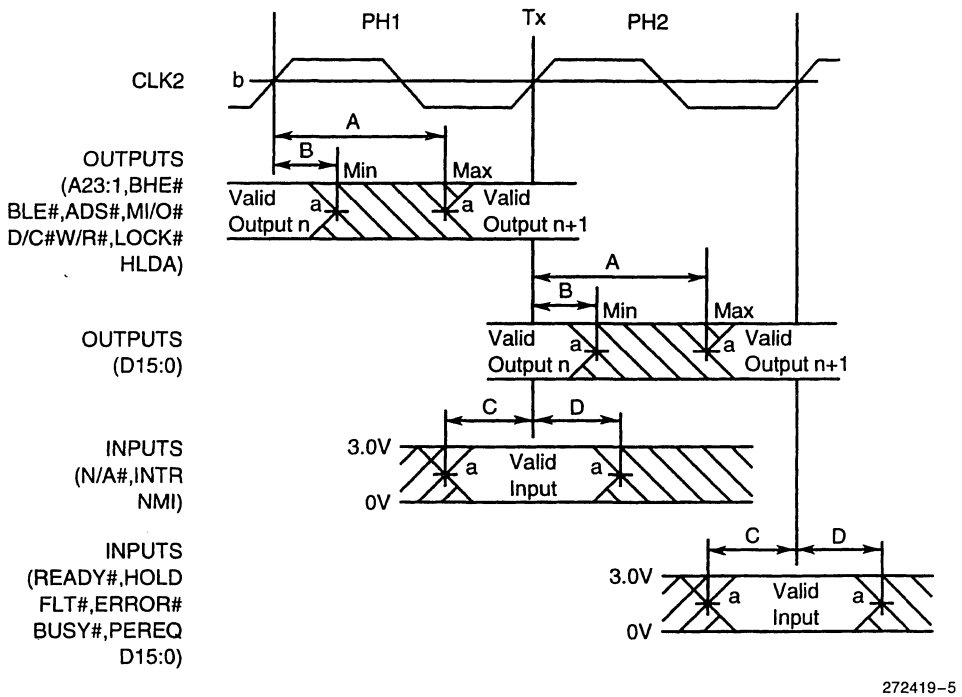
5.0 AC SPECIFICATIONS

Table 6 lists output delays, input setup requirements, and input hold requirements. All AC specifications are relative to the CLK2 rising edge crossing the 2.0V level.

Figure 5 shows the measurement points for AC specifications. Inputs must be driven to the indicated voltage levels when AC specifications are measured. Output delays are specified with minimum and maximum limits measured as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs ADS#, W/R#, D/C#, MI/O#, LOCK#, BHE#, BLE#, A23:1 and HLDA change only at the beginning of phase one. D15:0 (write cycles) change only at the beginning of phase two.

The READY#, HOLD, BUSY#, ERROR#, PEREQ, FLT# and D15:0 (read cycles) inputs are sampled at the beginning of phase one. The NA#, INTR and NMI inputs are sampled at the beginning of phase two.



LEGEND
a - 1.5V
b - 2.0V
A - Maximum Output Delay Spec
B - Minimum Output Delay Spec
C - Minimum Input Setup Spec
D - Minimum Input Hold Spec

Figure 5. Drive Levels and Measurement Points for AC Specifications

Table 6. AC Characteristics

Symbol	Parameter	40 MHz		33 MHz		25 MHz		Test Condition
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)	
	Operating Frequency	0	40	0	33	0	25	MHz (Note 1)
t_1	CLK2 Period	12.5		15		20		
t_{2a}	CLK2 High Time	4.5		6.25		7		(Note 2)
t_{2b}	CLK2 High Time	3.5		4		4		(Note 2)
t_{3a}	CLK2 Low Time	4.5		6.25		7		(Note 2)
t_{3b}	CLK2 Low Time	3.5		4.5		5		(Note 2)
t_4	CLK2 Fall Time		4		4		7	(Note 2)
t_5	CLK2 Rise Time		4		4		7	(Note 2)
t_6	A23:1 Valid Delay	4	13	4	15	4	17	C_L = 50 pF
t_7	A23:1 Float Delay	4	20	4	20	4	30	(Note 3)
t_8	BHE#, BLE#, LOCK# Valid Delay	4	13	4	15	4	17	C_L = 50 pF
t_9	BHE#, BLE#, LOCK# Float Delay	4	20	4	20	4	30	(Note 3)
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	13	4	15	4	17	C_L = 50 pF
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	4	20	4	20	4	30	(Note 3)
t_{12}	D15:0 Write Data Valid Delay	7	18	7	23	7	23	C_L = 50 pF
t_{12a}	D15:0 Write Data Hold Time	2		2		2		C_L = 50 pF
t_{13}	D15:0 Write Data Float Delay	4	17	4	17	4	22	(Note 3)
t_{14}	HLDA Valid Delay	4	17	4	20	4	22	C_L = 50 pF
t_{15}	NA# Setup Time	5		5		5		
t_{16}	NA# Hold Time	2		2		3		
t_{19}	READY# Setup Time	7		7		9		
t_{20}	READY# Hold Time	4		4		4		
t_{21}	D15:0 Read Setup Time	4		5		7		
t_{22}	D15:0 Read Hold Time	3		3		5		

NOTES:
1. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
2. These are not tested. They are guaranteed by characterization.
3. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
4. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.

Table 6. AC Characteristics (Continued)

Symbol	Parameter	40 MHz		33 MHz		25 MHz		Test Condition
		Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)	
t_{23}	HOLD Setup Time	4		9		9		
t_{24}	HOLD Hold Time	2		2		3		
t_{25}	RESET Setup Time	4		5		8		
t_{26}	RESET Hold Time	2		2		3		
t_{27}	NMI, INTR Setup Time	5		5		6		(Note 4)
t_{28}	NMI, INTR Hold Time	5		5		6		(Note 4)
t_{29}	PEREQ, ERROR#, BUSY#, FLT# Setup Time	5		5		6		(Note 4)
t_{30}	PEREQ, ERROR#, BUSY#, FLT# Hold Time	4		4		5		(Note 4)

NOTES:
1. Tested at maximum operating frequency and guaranteed by design characterization at lower operating frequencies.
2. These are not tested. They are guaranteed by characterization.
3. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not fully tested.
4. These inputs may be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to ensure recognition within a specific CLK2 period.

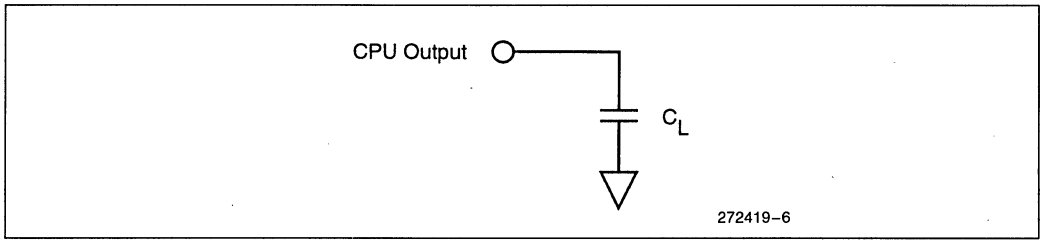


Figure 6. AC Test Loads

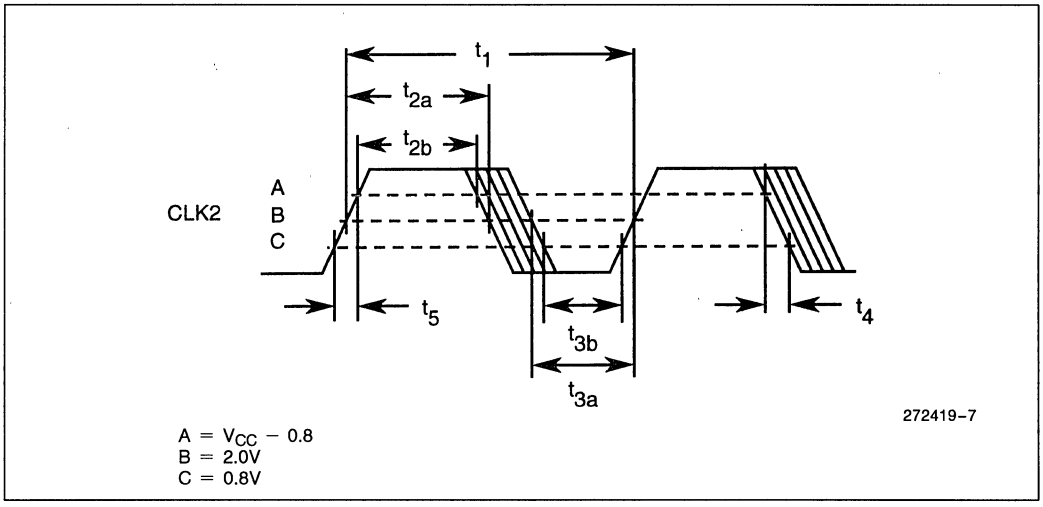


Figure 7. CLK2 Waveform

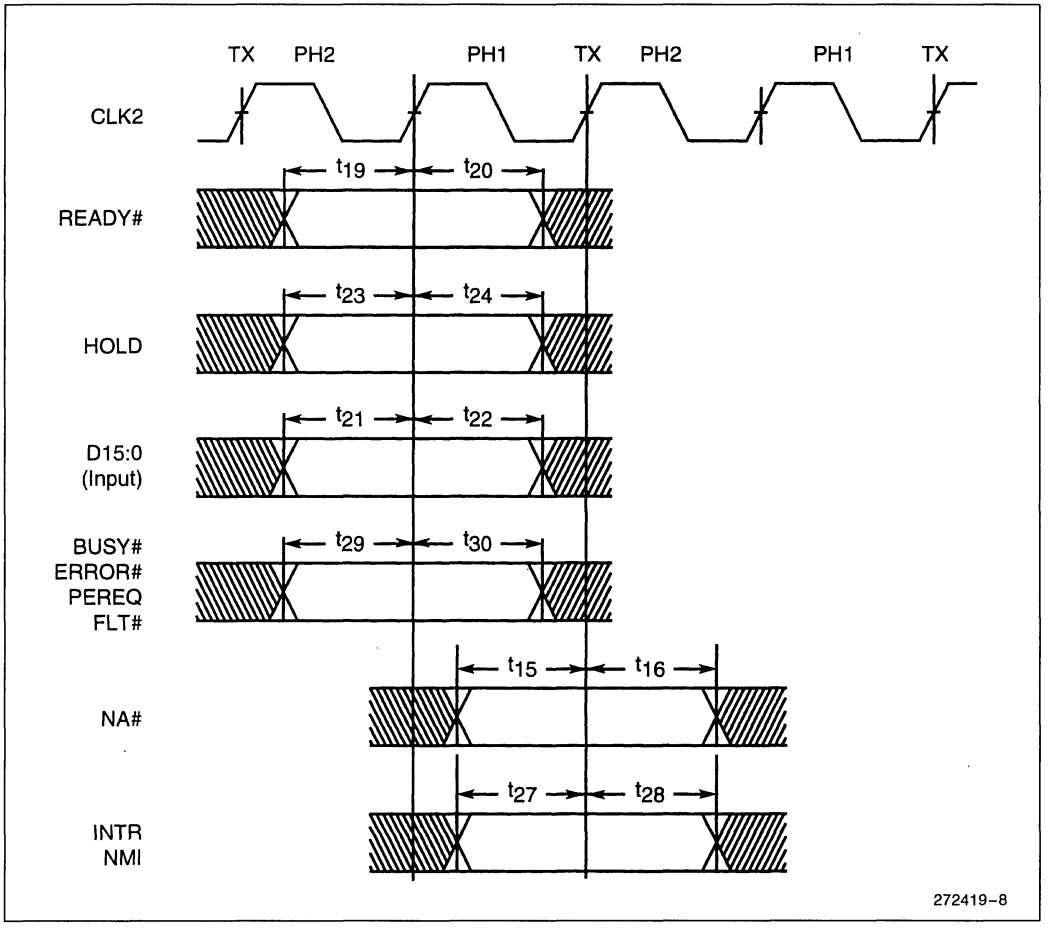


Figure 8. AC Timing Waveforms—Input Setup and Hold Timing

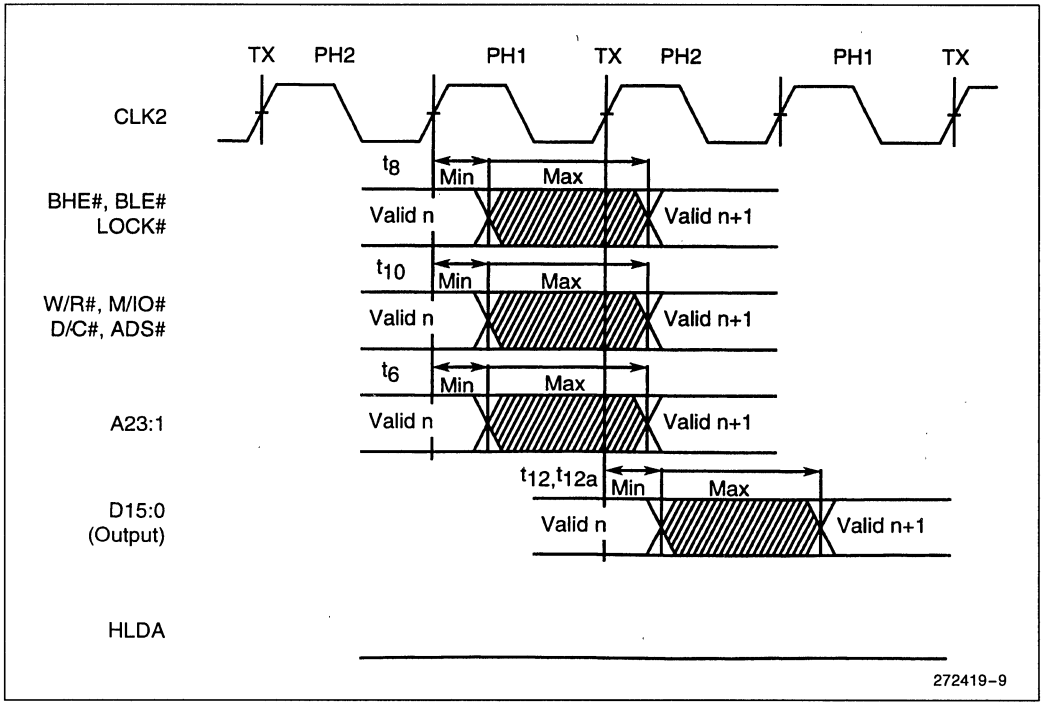


Figure 9. AC Timing Waveforms—Output Valid Delay Timing

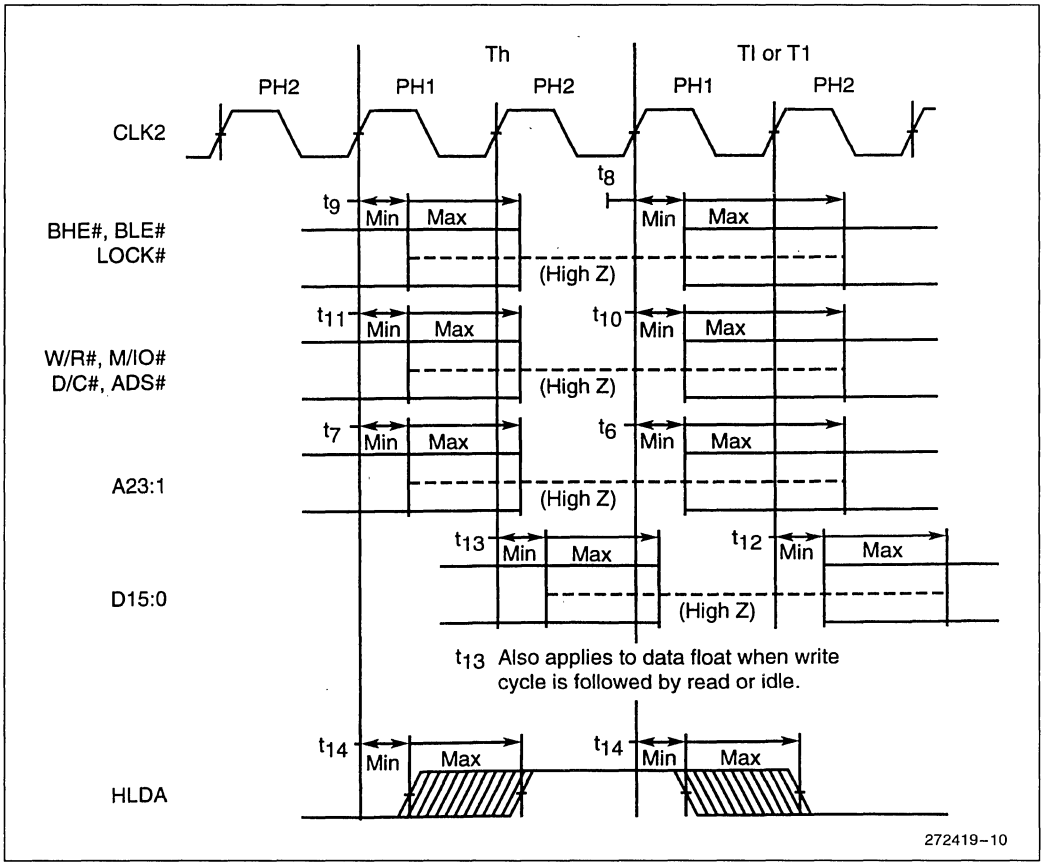


Figure 10. AC Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing

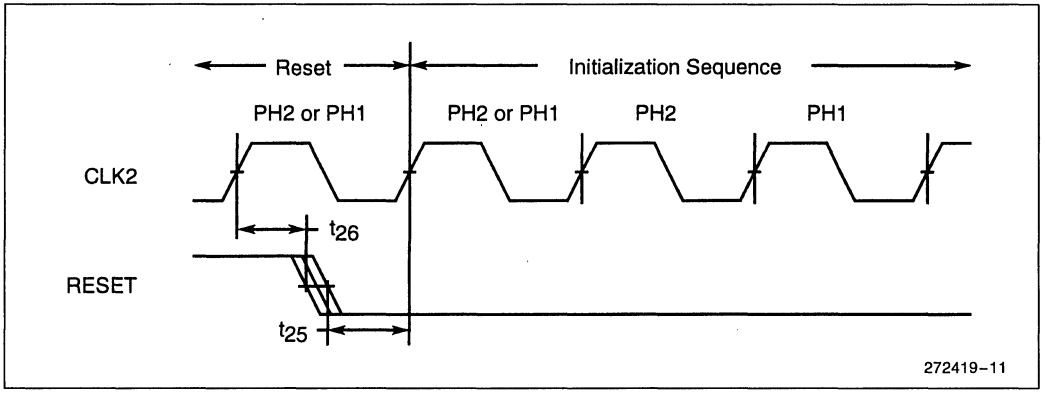


Figure 11. AC Timing Waveforms—RESET Setup and Hold Timing and Internal Phase

6.0 REVISION HISTORY

This -003 data sheet contains the following changes from the -002 version.

- Changed V_{CC} at 40 MHz to 4.75V to 5.25V (Pages 1 and 9)
- Renamed "Powerdown Mode" to "Clock Freeze Mode" on page 1.
- Added Clarifications to Figure 1.
- Corrected pin numbering for A23:1 in Table 2.
- Changed Section 3.3 first sentence from "... on page 12" to "... on page 9".
- Changed first sentence on page 12 from "Table 7 lists ..." to "Table 6 lists ...". Also changed fourth paragraph, first sentence on page 12 from "... A25:1" to "... A23:1".

Intel386™ DX MICROPROCESSOR 32-BIT CHMOS MICROPROCESSOR WITH INTEGRATED MEMORY MANAGEMENT

- **Flexible 32-Bit Microprocessor**
  - **— 8, 16, 32-Bit Data Types**
  - **— 8 General Purpose 32-Bit Registers**
- **Very Large Address Space**
  - **— 4 Gigabyte Physical**
  - **— 64 Terabyte Virtual**
  - **— 4 Gigabyte Maximum Segment Size**
- **Integrated Memory Management Unit**
  - **— Virtual Memory Support**
  - **— Optional On-Chip Paging**
  - **— 4 Levels of Protection**
  - **— Fully Compatible with 80286**
- **Object Code Compatible with All 8086 Family Microprocessors**
- **Virtual 8086 Mode Allows Running of 8086 Software in a Protected and Paged System**
- **Hardware Debugging Support**
- **Optimized for System Performance**
  - **— Pipelined Instruction Execution**
  - **— On-Chip Address Translation Caches**
  - **— 20, 25 and 33 MHz Clock**
  - **— 40, 50 and 66 Megabytes/Sec Bus Bandwidth**
- **Numerics Support via Intel387™ DX Math Coprocessor**
- **Complete System Development Support**
  - **— Software: C, PL/M, Assembler System Generation Tools**
  - **— Debuggers: PSCOPE, ICE™-386**
- **High Speed CHMOS IV Technology**
- **132 Pin Grid Array Package**
- **132 Pin Plastic Quad Flat Package**

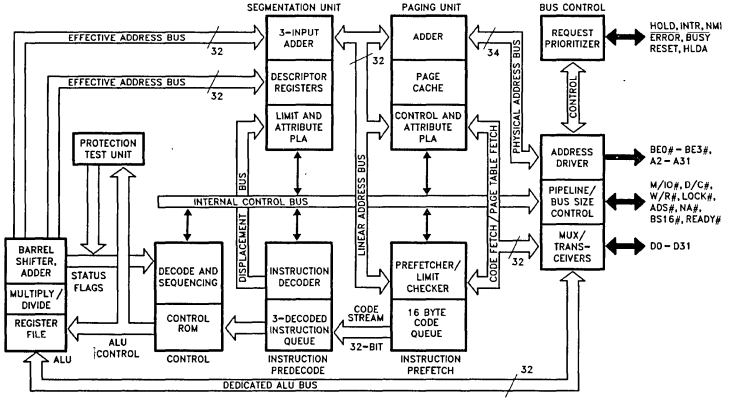
(See Packaging Specification, Order #231369)

3

The Intel386 DX Microprocessor is an entry-level 32-bit microprocessor designed for single-user applications and operating systems such as MS-DOS and Windows. The 32-bit registers and data paths support 32-bit addresses and data types. The processor addresses up to four gigabytes of physical memory and 64 terabytes (2**46) of virtual memory. The integrated memory management and protection architecture includes address translation registers, multitasking hardware and a protection mechanism to support operating systems. Instruction pipelining, on-chip address translation, ensure short average instruction execution times and maximum system throughput.

The Intel386 DX CPU offers new testability and debugging features. Testability features include a self-test and direct access to the page translation cache. Four new breakpoint registers provide breakpoint traps on code execution or data accesses, for powerful debugging of even ROM-based systems.

Object-code compatibility with all 8086 family members (8086, 8088, 80186, 80188, 80286) means the Intel386 DX offers immediate access to the world's largest microprocessor software base.



231630-49

Intel386™ DX Pipelined 32-Bit Microarchitecture

Intel386™ DX and Intel387™ DX are Trademarks of Intel Corporation.
MS-DOS and Windows are Trademarks of MICROSOFT Corporation.

December 1994
Order Number: 231630-011

Intel386™ DX MICROPROCESSOR
32-BIT CHMOS MICROPROCESSOR
WITH INTEGRATED MEMORY MANAGEMENT

CONTENTS

CONTENTS

PAGE

NOTE:
This is revision 011; This supercedes all previous revisions.

1. PIN ASSIGNMENT

The Intel386 DX pinout as viewed from the top side of the component is shown by Figure 1-1. Its pinout as viewed from the Pin side of the component is Figure 1-2.

V_{CC} and GND connections must be made to multiple V_{CC} and V_{SS} (GND) pins. Each V_{CC} and V_{SS} must be connected to the appropriate voltage level. The circuit board should include V_{CC} and GND planes for power distribution and all V_{CC} and V_{SS} pins must be connected to the appropriate plane.

NOTE:
Pins identified as "N.C." should remain completely unconnected.

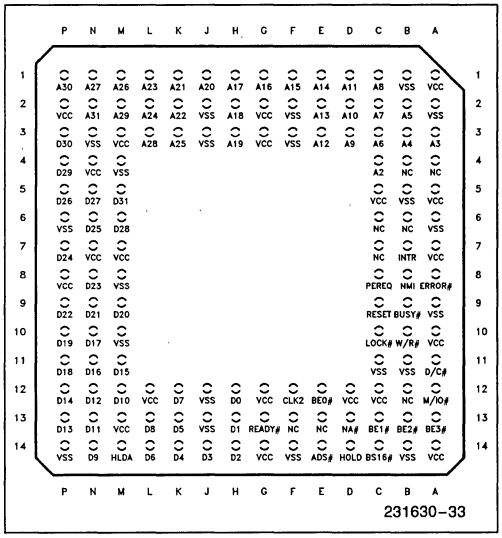


Figure 1-1. Intel386™ DX PGA Pinout—View from Top Side

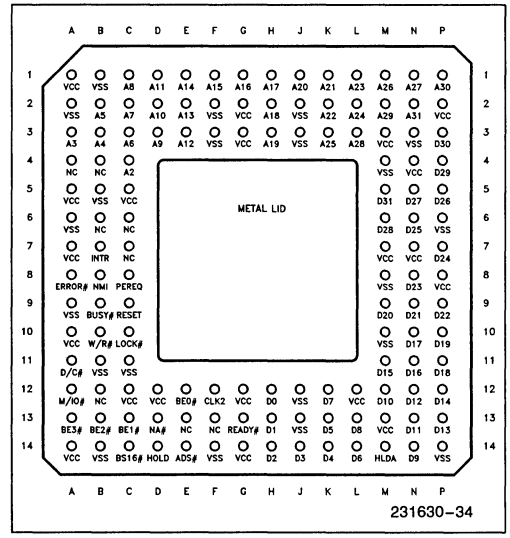


Figure 1-2. Intel386™ DX PGA Pinout—View from Pin Side

Table 1-1. Intel386™ DX PGA Pinout—Functional Grouping

Signal/Pin		Signal/Pin		Signal/Pin		Signal/Pin		Signal/Pin		Signal/Pin	
A2	C4	A24	L2	D6	L14	D28	M6	V_{CC}	C12	V_{SS}	F2
A3	A3	A25	K3	D7	K12	D29	P4		D12		F3
A4	B3	A26	M1	D8	L13	D30	P3		G2		F14
A5	B2	A27	N1	D9	N14	D31	M5		G3		J2
A6	C3	A28	L3	D10	M12	D/C#	A11		G12		J3
A7	C2	A29	M2	D11	N13	ERROR#	A8		G14		J12
A8	C1	A30	P1	D12	N12	HLDA	M14		L12		J13
A9	D3	A31	N2	D13	P13	HOLD	D14		M3		M4
A10	D2	ADS#	E14	D14	P12	INTR	B7		M7		M8
A11	D1	BE0#	E12	D15	M11	LOCK#	C10		M13		M10
A12	E3	BE1#	C13	D16	N11	M/IO#	A12		N4		N3
A13	E2	BE2#	B13	D17	N10	NA#	D13		N7		P6
A14	E1	BE3#	A13	D18	P11	NMI	B8		P2		P14
A15	F1	BS16#	C14	D19	P10	PEREQ	C8		P8	W/R#	B10
A16	G1	BUSY#	B9	D20	M9	READY#	G13	V_{SS}	A2	N.C.	A4
A17	H1	CLK2	F12	D21	N9	RESET	C9		A6		B4
A18	H2	D0	H12	D22	P9	V_{CC}	A1		A9		B6
A19	H3	D1	H13	D23	N8		A5		B1		B12
A20	J1	D2	H14	D24	P7		A7		B5		C6
A21	K1	D3	J14	D25	N6		A10		B11		C7
A22	K2	D4	K14	D26	P5		A14		B14		E13
A23	L1	D5	K13	D27	N5		C5		C11		F13

1.1 PIN DESCRIPTION TABLE

The following table lists a brief description of each pin on the Intel386 DX. The following definitions are used in these descriptions:

The named signal is active LOW.
I Input signal.
O Output signal.
I/O Input and Output signal.
— No electrical connection.

For a more complete description refer to Section 5.2 Signal Description.

Symbol	Type	Name and Function
CLK2	I	**CLK2** provides the fundamental timing for the Intel386 DX.
D_{31}–D_0	I/O	**DATA BUS** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.
A_{31}–A_2	O	**ADDRESS BUS** outputs physical memory or port I/O addresses.
BE0#–BE3#	O	**BYTE ENABLES** indicate which data bytes of the data bus take part in a bus cycle.
W/R#	O	**WRITE/READ** is a bus cycle definition pin that distinguishes write cycles from read cycles.
D/C#	O	**DATA/CONTROL** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and instruction fetching.
M/IO#	O	**MEMORY I/O** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles.
LOCK#	O	**BUS LOCK** is a bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active.
ADS#	O	**ADDRESS STATUS** indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BE0#, BE1#, BE2#, BE3# and A_{31}–A_2) are being driven at the Intel386 DX pins.
NA#	I	**NEXT ADDRESS** is used to request address pipelining.
READY#	I	**BUS READY** terminates the bus cycle.
BS16#	I	**BUS SIZE 16** input allows direct connection of 32-bit and 16-bit data buses.
HOLD	I	**BUS HOLD REQUEST** input allows another bus master to request control of the local bus.

1.1 PIN DESCRIPTION TABLE (Continued)

Symbol	Type	Name and Function
HLDA	O	**BUS HOLD ACKNOWLEDGE** output indicates that the Intel386 DX has surrendered control of its local bus to another bus master.
BUSY#	I	**BUSY** signals a busy condition from a processor extension.
ERROR#	I	**ERROR** signals an error condition from a processor extension.
PEREQ	I	**PROCESSOR EXTENSION REQUEST** indicates that the processor extension has data to be transferred by the Intel386 DX.
INTR	I	**INTERRUPT REQUEST** is a maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
NMI	I	**NON-MASKABLE INTERRUPT REQUEST** is a non-maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
RESET	I	**RESET** suspends any operation in progress and places the Intel386 DX in a known reset state. See **Interrupt Signals** for additional information.
N/C	—	**NO CONNECT** should always remain unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the Intel386 DX.
V_{CC}	I	**SYSTEM POWER** provides the +5V nominal D.C. supply input.
V_{SS}	I	**SYSTEM GROUND** provides 0V connection from which all inputs and outputs are measured.

2. BASE ARCHITECTURE

2.1 INTRODUCTION

The Intel386 DX consists of a central processing unit, a memory management unit and a bus interface.

The central processing unit consists of the execution unit and instruction unit. The execution unit contains the eight 32-bit general purpose registers which are used for both address calculation, data operations and a 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations. The multiply and divide logic uses a 1-bit per cycle algorithm. The multiply algorithm stops the iteration when the most significant bits of the multiplier are all zero. This allows typical 32-bit multiplies to be executed in under one microsecond. The instruction unit decodes the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

The memory management unit (MMU) consists of a segmentation unit and a paging unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability, and efficient sharing. The paging mechanism operates beneath and is transparent to the segmentation process, to allow management of the physical address space. Each segment is divided into one or more 4K byte pages. To implement a virtual memory system, the Intel386 DX supports full restartability for all page and segment faults.

Memory is organized into one or more variable length segments, each up to four gigabytes in size. A given region of the linear address space, a segment, can have attributes associated with it. These attributes include its location, size, type (i.e. stack, code or data), and protection characteristics. Each task on an Intel386 DX can have a maximum of 16,381 segments of up to four gigabytes each, thus providing 64 terabytes (trillion bytes) of virtual memory to each task.

The segmentation unit provides four-levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of integrity.

The Intel386 DX has two modes of operation: Real Address Mode (Real Mode), and Protected Virtual Address Mode (Protected Mode). In Real Mode the Intel386 DX operates as a very fast 8086, but with 32-bit extensions if desired. Real Mode is required primarily to setup the processor for Protected Mode operation. Protected Mode provides access to the sophisticated memory management, paging and privilege capabilities of the processor.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each such task behaves with 8086 semantics, thus allowing 8086 software (an application program, or an entire operating system) to execute. The Virtual 8086 tasks can be isolated and protected from one another and the host Intel386 DX operating system, by the use of paging, and the I/O Permission Bitmap.

Finally, to facilitate high performance system hardware designs, the Intel386 DX bus interface offers address pipelining, dynamic data bus sizing, and direct Byte Enable signals for each byte of the data bus. These hardware features are described fully beginning in Section 5.

2.2 REGISTER OVERVIEW

The Intel386 DX has 32 register resources in the following categories:

- General Purpose Registers
- Segment Registers
- Instruction Pointer and Flags
- Control Registers
- System Address Registers
- Debug Registers
- Test Registers.

The registers are a superset of the 8086, 80186 and 80286 registers, so all 16-bit 8086, 80186 and 80286 registers are contained within the 32-bit Intel386 DX.

Figure 2-1 shows all of Intel386 DX base architecture registers, which include the general address and data registers, the instruction pointer, and the flags register. The contents of these registers are task-specific, so these registers are automatically loaded with a new context upon a task switch operation.

The base architecture also includes six directly accessible segments, each up to 4 Gbytes in size. The segments are indicated by the selector values placed in Intel386 DX segment registers of Figure 2-1. Various selector values can be loaded as a program executes, if desired.

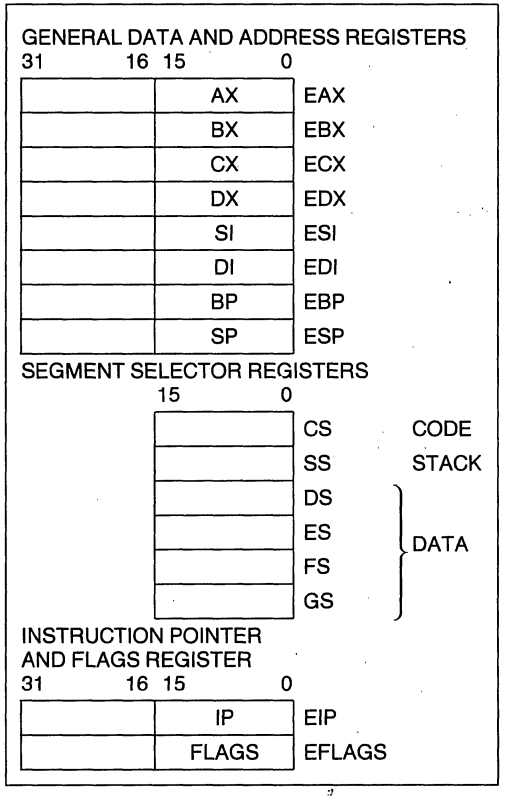


Figure 2-1. Intel386™ DX Base Architecture Registers

The selectors are also task-specific, so the segment registers are automatically loaded with new context upon a task switch operation.

The other types of registers, Control, System Address, Debug, and Test, are primarily used by system software.

2.3 REGISTER DESCRIPTIONS

2.3.1 General Purpose Registers

General Purpose Registers: The eight general purpose registers of 32 bits hold data or address quantities. The general registers, Figure 2-2, support data operands of 1, 8, 16, 32 and 64 bits, and bit fields of 1 to 32 bits. They support address operands of 16 and 32 bits. The 32-bit registers are named EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP.

The least significant 16 bits of the registers can be accessed separately. This is done by using the 16-bit names of the registers AX, BX, CX, DX, SI, DI, BP, and SP. When accessed as a 16-bit operand, the upper 16 bits of the register are neither used nor changed.

Finally 8-bit operations can individually access the lowest byte (bits 0–7) and the higher byte (bits 8–15) of general purpose registers AX, BX, CX and DX. The lowest bytes are named AL, BL, CL and DL, respectively. The higher bytes are named AH, BH, CH and DH, respectively. The individual byte accessibility offers additional flexibility for data operations, but is not used for effective address calculation.

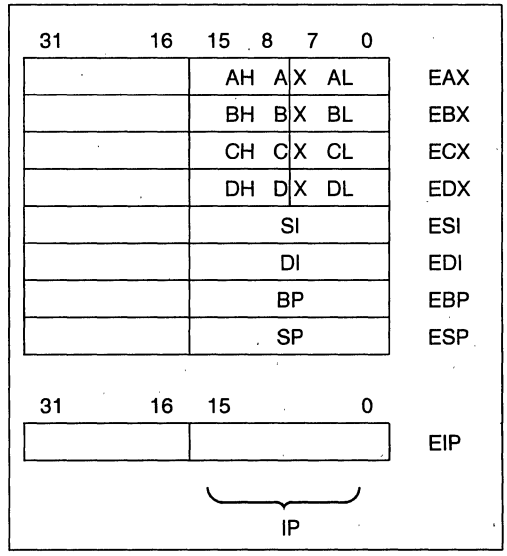


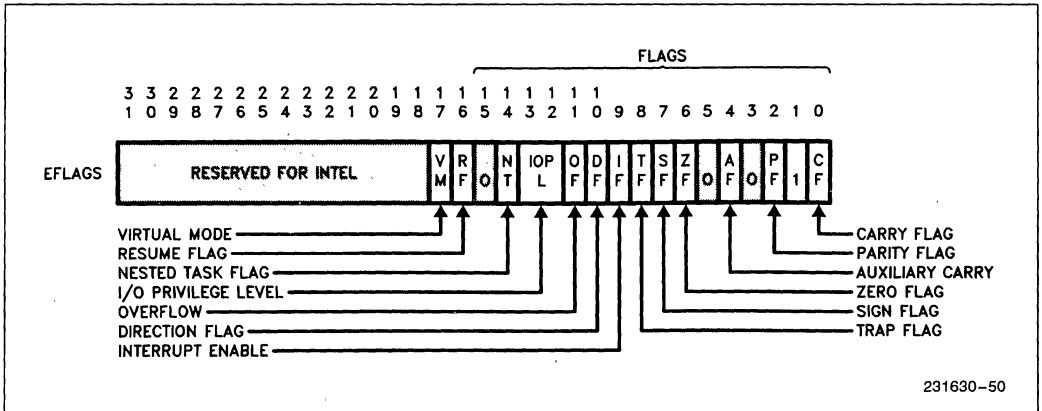
Figure 2-2. General Registers and Instruction Pointer

2.3.2 Instruction Pointer

The instruction pointer, Figure 2-2, is a 32-bit register named EIP. EIP holds the offset of the next instruction to be executed. The offset is always relative to the base of the code segment (CS). The lower 16 bits (bits 0–15) of EIP contain the 16-bit instruction pointer named IP, which is used by 16-bit addressing.

2.3.3 Flags Register

The Flags Register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS, shown in Figure 2-3, control certain operations and indicate status of the Intel386 DX. The lower 16 bits (bit 0–15) of EFLAGS contain the 16-bit flag register named FLAGS, which is most useful when executing 8086 and 80286 code.



NOTE:
0 indicates Intel reserved: do not define; see section 2.3.10.

Figure 2-3. Flags Register

VM (Virtual 8086 Mode, bit 17)

The VM bit provides Virtual 8086 Mode within Protected Mode. If set while the Intel386 DX is in Protected Mode, the Intel386 DX will switch to Virtual 8086 operation, handling segment loads as the 8086 does, but generating exception 13 faults on privileged opcodes. The VM bit can be set only in Protected Mode, by the IRET instruction (if current privilege level = 0) and by task switches at any privilege level. The VM bit is unaffected by POPF. PUSHF always pushes a 0 in this bit, even if executing in virtual 8086 Mode. The EFLAGS image pushed during interrupt processing or saved during task switches will contain a 1 in this bit if the interrupted code was executing as a Virtual 8086 Task.

RF (Resume Flag, bit 16)

The RF flag is used in conjunction with the debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. When RF is set, it causes any debug fault to be ignored on the next instruction. RF is then automatically reset at the successful completion of every instruction (no faults are signalled) except the IRET instruction, the POPF instruction, (and JMP, CALL, and INT instructions causing a task switch). These instructions set RF to the value specified by the memory image. For example, at the end of the breakpoint service routine, the IRET instruction can pop an EFLAG image having the RF bit set and resume the program's execution at the breakpoint address without generating another breakpoint fault on the same location.

NT (Nested Task, bit 14)

This flag applies to Protected Mode. NT is set to indicate that the execution of this task is nested within another task. If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS. This bit is set or reset by control transfers to other tasks. The value of NT in EFLAGS is tested by the IRET instruction to determine whether to do an inter-task return or an intra-task return. A POPF or an IRET instruction **will** affect the setting of this bit according to the image popped, at any privilege level.

IOPL (Input/Output Privilege Level, bits 12-13)

This two-bit field applies to Protected Mode. IOPL indicates the numerically maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O Permission Bitmap. It also indicates the maximum CPL value allowing alteration of the IF (INTR Enable Flag) bit when new values are popped into the EFLAG register. POPF and IRET instruction can alter the IOPL field when executed at CPL = 0. Task switches can always alter the IOPL field, when the new flag image is loaded from the incoming task's TSS.

OF (Overflow Flag, bit 11)

OF is set if the operation resulted in a signed overflow. Signed overflow occurs when the operation resulted in carry/borrow **into** the sign bit (high-order bit) of the result but did not result in a carry/borrow **out of** the high-order bit, or vice-versa. For 8/16/32 bit operations, OF is set according to overflow at bit 7/15/31, respectively.

DF (Direction Flag, bit 10)

DF defines whether ESI and/or EDI registers postdecrement or postincrement during the string instructions. Postincrement occurs if DF is reset. Postdecrement occurs if DF is set.

IF (INTR Enable Flag, bit 9)

The IF flag, when set, allows recognition of external interrupts signalled on the INTR pin. When IF is reset, external interrupts signalled on the INTR are not recognized. IOPL indicates the maximum CPL value allowing alteration of the IF bit when new values are popped into EFLAGS or FLAGS.

TF (Trap Enable Flag, bit 8)

TF controls the generation of exception 1 trap when single-stepping through code. When TF is set, the Intel386 DX generates an exception 1 trap after the next instruction is executed. When TF is reset, exception 1 traps occur only as a function of the breakpoint addresses loaded into debug registers DR0–DR3.

SF (Sign Flag, bit 7)

SF is set if the high-order bit of the result is set, it is reset otherwise. For 8-, 16-, 32-bit operations, SF reflects the state of bit 7, 15, 31 respectively.

ZF (Zero Flag, bit 6)

ZF is set if all bits of the result are 0. Otherwise it is reset.

AF (Auxiliary Carry Flag, bit 4)

The Auxiliary Flag is used to simplify the addition and subtraction of packed BCD quantities. AF is set if the operation resulted in a carry out of bit 3 (addition) or a borrow into bit 3 (subtraction). Otherwise AF is reset. AF is affected by carry out of, or borrow into bit 3 only, regardless of overall operand length: 8, 16 or 32 bits.

PF (Parity Flags, bit 2)

PF is set if the low-order eight bits of the operation contains an even number of "1's" (even parity). PF is reset if the low-order eight bits have odd parity. PF is a function of only the low-order eight bits, regardless of operand size.

CF (Carry Flag, bit 0)

CF is set if the operation resulted in a carry out of (addition), or a borrow into (subtraction) the high-order bit. Otherwise CF is reset. For 8-, 16- or 32-bit operations, CF is set according to carry/borrow at bit 7, 15 or 31, respectively.

Note in these descriptions, "set" means "set to 1," and "reset" means "reset to 0."

2.3.4 Segment Registers

Six 16-bit segment registers hold segment selector values identifying the currently addressable memory segments. Segment registers are shown in Figure 2-4. In Protected Mode, each segment may range in size from one byte up to the entire linear and physi-

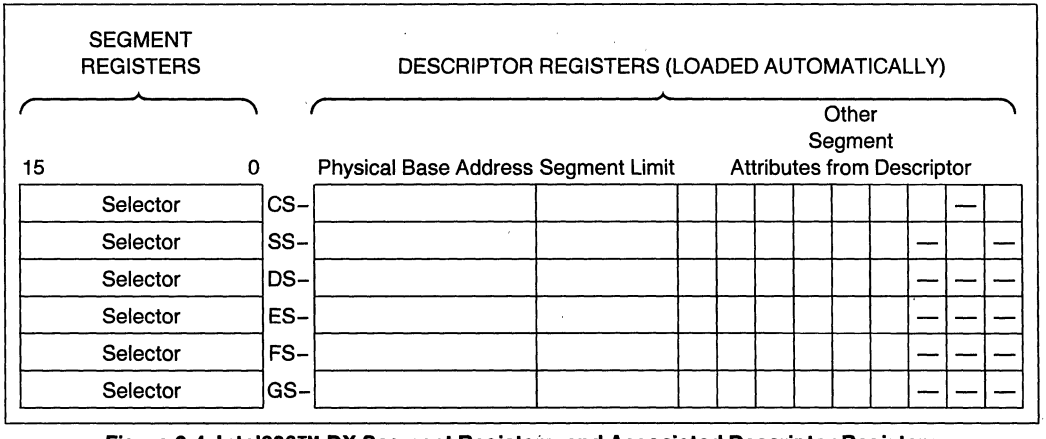


Figure 2-4. Intel386™ DX Segment Registers, and Associated Descriptor Registers

cal space of the machine, 4 Gbytes (2^{32} bytes). If a maximum sized segment is used (limit = FFFFFFFFH) it should be Dword aligned (i.e., the least two significant bits of the segment base should be zero). This will avoid a segment limit violation (exception 13) caused by the wrap around. In Real Address Mode, the maximum segment size is fixed at 64 Kbytes (2^{16} bytes).

The six segments addressable at any given moment are defined by the segment registers CS, SS, DS, ES, FS and GS. The selector in CS indicates the current code segment; the selector in SS indicates the current stack segment; the selectors in DS, ES, FS and GS indicate the current data segments.

2.3.5 Segment Descriptor Registers

The segment descriptor registers are not programmer visible, yet it is very useful to understand their content. Inside the Intel386 DX, a descriptor register (programmer invisible) is associated with each programmer-visible segment register, as shown by Figure 2-4. Each descriptor register holds a 32-bit segment base address, a 32-bit segment limit, and the other necessary segment attributes.

When a selector value is loaded into a segment register, the associated descriptor register is automatically updated with the correct information. In Real Address Mode, only the base address is updated directly (by shifting the selector value four bits to the left), since the segment maximum limit and attributes are fixed in Real Mode. In Protected Mode, the base address, the limit, and the attributes are all updated per the contents of the segment descriptor indexed by the selector.

Whenever a memory reference occurs, the segment descriptor register associated with the segment being used is automatically involved with the memory reference. The 32-bit segment base address becomes a component of the linear address calculation, the 32-bit limit is used for the limit-check operation, and the attributes are checked against the type of memory reference requested.

2.3.6 Control Registers

The Intel386 DX has three control registers of 32 bits, CR0, CR2 and CR3, to hold machine state of a global nature (not specific to an individual task). These registers, along with System Address Registers described in the next section, hold machine state that affects all tasks in the system. To access the Control Registers, load and store instructions are defined.

CR0: Machine Control Register (includes 80286 Machine Status Word)

CR0, shown in Figure 2-5, contains 6 defined bits for control and status purposes. The low-order 16 bits of CR0 are also known as the Machine Status Word, MSW, for compatibility with 80286 Protected Mode. LMSW and SMSW instructions are taken as special aliases of the load and store CR0 operations, where only the low-order 16 bits of CR0 are involved. For compatibility with 80286 operating systems the Intel386 DX LMSW instructions work in an identical fashion to the LMSW instruction on the 80286. (i.e. It only operates on the low-order 16-bits of CR0 and it ignores the new bits in CR0.) New Intel386 DX operating systems should use the MOV CR0, Reg instruction.

The defined CR0 bits are described below.

PG (Paging Enable, bit 31)

the PG bit is set to enable the on-chip paging unit. It is reset to disable the on-chip paging unit.

R (reserved, bit 4)

This bit is reserved by Intel. When loading CR0 care should be taken to not alter the value of this bit.

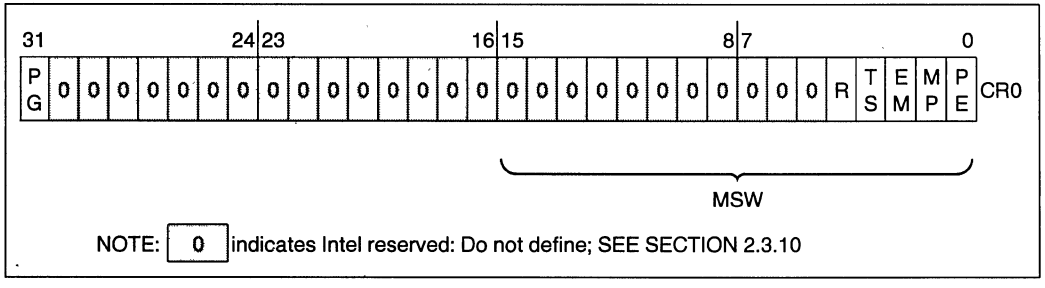


Figure 2-5. Control Register 0

TS (Task Switched, bit 3)

TS is automatically set whenever a task switch operation is performed. If TS is set, a coprocessor ESCape opcode will cause a Coprocessor Not Available trap (exception 7). The trap handler typically saves the Intel387 DX coprocessor context belonging to a previous task, loads the Intel387 DX coprocessor state belonging to the current task, and clears the TS bit before returning to the faulting coprocessor opcode.

EM (Emulate Coprocessor, bit 2)

The EMulate coprocessor bit is set to cause all coprocessor opcodes to generate a Coprocessor Not Available fault (exception 7). It is reset to allow coprocessor opcodes to be executed on an actual Intel387 DX coprocessor (this is the default case after reset). Note that the WAIT opcode is not affected by the EM bit setting.

MP (Monitor Coprocessor, bit 1)

The MP bit is used in conjunction with the TS bit to determine if the WAIT opcode will generate a Coprocessor Not Available fault (exception 7) when TS = 1. When both MP = 1 and TS = 1, the WAIT opcode generates a trap. Otherwise, the WAIT opcode does not generate a trap. Note that TS is automatically set whenever a task switch operation is performed.

PE (Protection Enable, bit 0)

The PE bit is set to enable the Protected Mode. If PE is reset, the processor operates again in Real Mode. PE may be set by loading MSW or CR0. PE can be reset only by a load into CR0. Resetting the PE bit is typically part of a longer instruction sequence needed for proper transition from Protected Mode to Real Mode. Note that for strict 80286 compatibility, PE cannot be reset by the LMSW instruction.

CR1: reserved

CR1 is reserved for use in future Intel processors.

CR2: Page Fault Linear Address

CR2, shown in Figure 2-6, holds the 32-bit linear address that caused the last page fault detected. The error code pushed onto the page fault handler's stack when it is invoked provides additional status information on this page fault.

CR3: Page Directory Base Address

CR3, shown in Figure 2-6, contains the physical base address of the page directory table. The Intel386 DX page directory table is always page-aligned (4 Kbyte-aligned). Therefore the lowest twelve bits of CR3 are ignored when written and they store as undefined.

A task switch through a TSS which **changes** the value in CR3, or an explicit load into CR3 with any value, will invalidate all cached page table entries in the paging unit cache. Note that if the value in CR3 does not change during the task switch, the cached page table entries are not flushed.

2.3.7 System Address Registers

Four special registers are defined to reference the tables or segments supported by the 80286 CPU and Intel386 DX protection model. These tables or segments are:

GDT (Global Descriptor Table),

IDT (Interrupt Descriptor Table),

LDT (Local Descriptor Table),

TSS (Task State Segment).

The addresses of these tables and segments are stored in special registers, the System Address and System Segment Registers illustrated in Figure 2-7. These registers are named GDTR, IDTR, LDTR and TR, respectively. Section 4 **Protected Mode Architecture** describes the use of these registers.

GDTR and IDTR

These registers hold the 32-bit linear base address and 16-bit limit of the GDT and IDT, respectively.

The GDT and IDT segments, since they are global to all tasks in the system, are defined by 32-bit linear addresses (subject to page translation if paging is enabled) and 16-bit limit values.

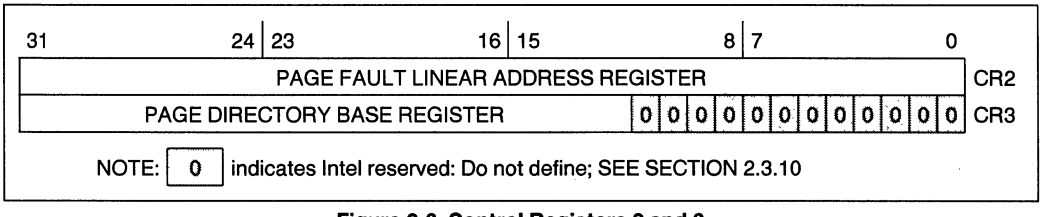


Figure 2-6. Control Registers 2 and 3

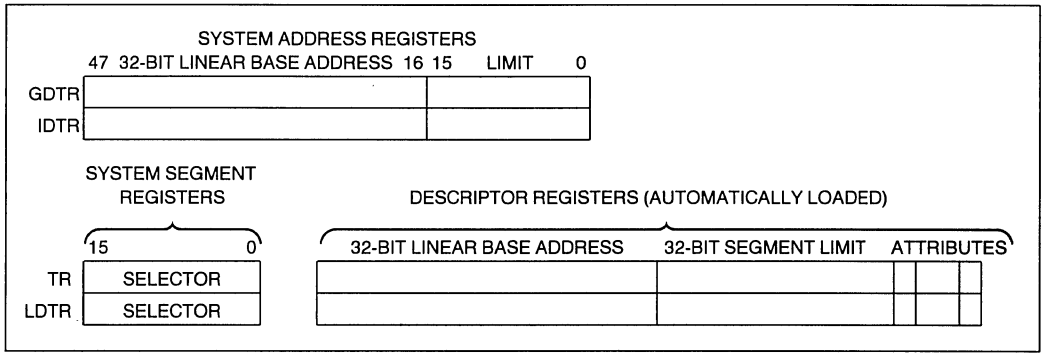


Figure 2-7. System Address and System Segment Registers

LDTR and TR

These registers hold the 16-bit selector for the LDT descriptor and the TSS descriptor, respectively.

The LDT and TSS segments, since they are task-specific segments, are defined by selector values stored in the system segment registers. Note that a segment descriptor register (programmer-invisible) is associated with each system segment register.

2.3.8 Debug and Test Registers

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. Debug Registers DR0–3 specify the four linear breakpoints. The Debug Control Register DR7 is used to set the breakpoints and the Debug Status Register DR6, displays the current state of the breakpoints. The use of the debug registers is described in section 2.12 **Debugging support.**

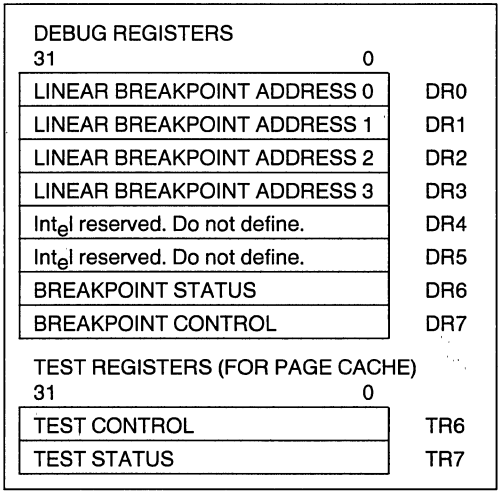


Figure 2-8. Debug and Test Registers

Test Registers: Two registers are used to control the testing of the RAM/CAM (Content Addressable Memories) in the Translation Lookaside Buffer portion of the Intel386 DX. TR6 is the command test register, and TR7 is the data register which contains the data of the Translation Lookaside buffer test. Their use is discussed in section 2.11 **Testability.**

Figure 2-8 shows the Debug and Test registers.

2.3.9 Register Accessibility

There are a few differences regarding the accessibility of the registers in Real and Protected Mode. Table 2-1 summarizes these differences. See Section 4 **Protected Mode Architecture** for further details.

2.3.10 Compatibility

VERY IMPORTANT NOTE:
COMPATIBILITY WITH FUTURE PROCESSORS

In the preceding register descriptions, note certain Intel386 DX register bits are Intel reserved. When reserved bits are called out, treat them as fully undefined. This is essential for your software compatibility with future processors! Follow the guidelines below:

1) **Do not depend on the states of any undefined bits when testing the values of defined register bits. Mask them out when testing.**
2) **Do not depend on the states of any undefined bits when storing them to memory or another register.**
3) **Do not depend on the ability to retain information written into any undefined bits.**
4) **When loading registers always load the undefined bits as zeros.**

Table 2-1. Register Usage

Register	Use in Real Mode		Use in Protected Mode		Use in Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
General Registers	Yes	Yes	Yes	Yes	Yes	Yes
Segment Registers	Yes	Yes	Yes	Yes	Yes	Yes
Flag Register	Yes	Yes	Yes	Yes	IOPL*	IOPL*
Control Registers	Yes	Yes	PL = 0	PL = 0	No	Yes
GDTR	Yes	Yes	PL = 0	Yes	No	Yes
IDTR	Yes	Yes	PL = 0	Yes	No	Yes
LDTR	No	No	PL = 0	Yes	No	No
TR	No	No	PL = 0	Yes	No	No
Debug Control	Yes	Yes	PL = 0	PL = 0	No	No
Test Registers	Yes	Yes	PL = 0	PL = 0	No	No

NOTES:
PL = 0: The registers can be accessed only when the current privilege level is zero.
*IOPL: The PUSHF and POPF instructions are made I/O Privilege Level sensitive in Virtual 8086 Mode.

5) However, registers which have been previously stored may be reloaded without masking.

Depending upon the values of undefined register bits will make your software dependent upon the unspecified Intel386 DX handling of these bits. Depending on undefined values risks making your software incompatible with future processors that define usages for the Intel386 DX-undefined bits. AVOID ANY SOFTWARE DEPENDENCE UPON THE STATE OF UNDEFINED Intel386 DX REGISTER BITS.

2.4 INSTRUCTION SET

2.4.1 Instruction Set Overview

The instruction set is divided into nine categories of operations:

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High Level Language Support
Operating System Support
Processor Control

These Intel386 DX instructions are listed in Table 2-2.

All Intel386 DX instructions operate on either 0, 1, 2, or 3 operands; where an operand resides in a register, in the instruction itself, or in memory. Most zero operand instructions (e.g. CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2 bytes long. Since the Intel386 DX has a 16-byte instruction queue, an average of 5 instructions will be prefetched. The use of two operands permits the following types of common instructions:

Register to Register
Memory to Register
Immediate to Register
Register to Memory
Immediate to Memory.

The operands can be either 8, 16, or 32 bits long. As a general rule, when executing code written for the Intel386 DX (32-bit code), operands are 8 or 32 bits; when executing existing 80286 or 8086 code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to all instructions which override the default length of the operands, (i.e. use 32-bit operands for 16-bit code, or 16-bit operands for 32-bit code).

For a more elaborate description of the instruction set, refer to the *Intel386 DX Programmer's Reference Manual.*

2.4.2 Intel386™ DX Instructions

Table 2-2a. Data Transfer

GENERAL PURPOSE	
MOV	Move operand
PUSH	Push operand onto stack
POP	Pop operand off stack
PUSHA	Push all registers on stack
POPA	Pop all registers off stack
XCHG	Exchange Operand, Register
XLAT	Translate
CONVERSION	
MOVZX	Move byte or Word, Dword, with zero extension
MOVSX	Move byte or Word, Dword, sign extended
CBW	Convert byte to Word, or Word to Dword
CWD	Convert Word to DWORD
CWDE	Convert Word to DWORD extended
CDQ	Convert DWORD to QWORD
INPUT/OUTPUT	
IN	Input operand from I/O space
OUT	Output operand to I/O space
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer into D segment register
LES	Load pointer into E segment register
LFS	Load pointer into F segment register
LGS	Load pointer into G segment register
LSS	Load pointer into S (Stack) segment register
FLAG MANIPULATION	
LAHF	Load A register from Flags
SAHF	Store A register in Flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack
PUSHFD	Push EFlags onto stack
POPFD	Pop EFlags off stack
CLC	Clear Carry Flag
CLD	Clear Direction Flag
CMC	Complement Carry Flag
STC	Set Carry Flag
STD	Set Direction Flag

Table 2-2b. Arithmetic Instructions

ADDITION	
ADD	Add operands
ADC	Add with carry
INC	Increment operand by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract operands
SBB	Subtract with borrow
DEC	Decrement operand by 1
NEG	Negate operand
CMP	Compare operands
DAS	Decimal adjust for subtraction
AAS	ASCII Adjust for subtraction
MULTIPLICATION	
MUL	Multiply Double/Single Precision
IMUL	Integer multiply
AAM	ASCII adjust after multiply
DIVISION	
DIV	Divide unsigned
IDIV	Integer Divide
AAD	ASCII adjust before division

Table 2-2c. String Instructions

MOVS	Move byte or Word, Dword string
INS	Input string from I/O space
OUTS	Output string to I/O space
CMPS	Compare byte or Word, Dword string
SCAS	Scan Byte or Word, Dword string
LODS	Load byte or Word, Dword string
STOS	Store byte or Word, Dword string
REP	Repeat
REPE/ REPZ	Repeat while equal/zero
RENE/ REPNZ	Repeat while not equal/not zero

Table 2-2d. Logical Instructions

LOGICALS	
NOT	"NOT" operands
AND	"AND" operands
OR	"Inclusive OR" operands
XOR	"Exclusive OR" operands
TEST	"Test" operands

Table 2-2d. Logical Instructions (Continued)

SHIFTS	
SHL/SHR	Shift logical left or right
SAL/SAR	Shift arithmetic left or right
SHLD/ SHRD	Double shift left or right
ROTATES	
ROL/ROR	Rotate left/right
RCL/RCR	Rotate through carry left/right

Table 2-2e. Bit Manipulation Instructions

SINGLE BIT INSTRUCTIONS	
BT	Bit Test
BTS	Bit Test and Set
BTR	Bit Test and Reset
BTC	Bit Test and Complement
BSF	Bit Scan Forward
BSR	Bit Scan Reverse

Table 2-2f. Program Control Instructions

CONDITIONAL TRANSFERS	
SETCC	Set byte equal to condition code
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if Sign

Table 2-2f. Program Control Instructions (Continued)

UNCONDITIONAL TRANSFERS	
CALL	Call procedure/task
RET	Return from procedure
JMP	Jump
ITERATION CONTROLS	
LOOP	Loop
LOOPE/ LOOPZ	Loop if equal/zero
LOOPNE/ LOOPNZ	Loop if not equal/not zero
JCXZ	JUMP if register CX = 0
INTERRUPTS	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Return from Interrupt/Task
CLI	Clear interrupt Enable
STI	Set Interrupt Enable

Table 2-2g. High Level Language Instructions

BOUND	Check Array Bounds
ENTER	Setup Parameter Block for Entering Procedure
LEAVE	Leave Procedure

Table 2-2h. Protection Model

SGDT	Store Global Descriptor Table
SIDT	Store Interrupt Descriptor Table
STR	Store Task Register
SLDT	Store Local Descriptor Table
LGDT	Load Global Descriptor Table
LIDT	Load Interrupt Descriptor Table
LTR	Load Task Register
LLDT	Load Local Descriptor Table
ARPL	Adjust Requested Privilege Level
LAR	Load Access Rights
LSL	Load Segment Limit
VERR/ VERW	Verify Segment for Reading or Writing
LMSW	Load Machine Status Word (lower 16 bits of CR0)
SMSW	Store Machine Status Word

Table 2-2i. Processor Control Instructions

HLT	Halt
WAIT	Wait until BUSY# negated
ESC	Escape
LOCK	Lock Bus

2.5 ADDRESSING MODES

2.5.1 Addressing Modes Overview

The Intel386 DX provides a total of 11 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

2.5.2 Register and Immediate Modes

Two of the addressing modes provide for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8-, 16- or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the opcode.

2.5.3 32-Bit Memory Addressing Modes

The remaining 9 modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by using combinations of the following four address elements:

DISPLACEMENT: An 8-, or 32-bit immediate value, following the instruction.

BASE: The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area.

INDEX: The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters.

SCALE: The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. Scaled index mode is especially useful for accessing arrays or structures.

Combinations of these 4 components make up the 9 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of Base and Index components which requires one additional clock.

As shown in Figure 2-9, the effective address (EA) of an operand is calculated according to the following formula.

EA = Base Reg + (Index Reg * Scaling) + Displacement

Direct Mode: The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit displacement.
EXAMPLE: INC Word PTR [500]

Register Indirect Mode: A BASE register contains the address of the operand.
EXAMPLE: MOV [ECX], EDX

Based Mode: A BASE register's contents is added to a DISPLACEMENT to form the operands offset.
EXAMPLE: MOV ECX, [EAX + 24]

Index Mode: An INDEX register's contents is added to a DISPLACEMENT to form the operands offset.
EXAMPLE: ADD EAX, TABLE[ESI]

Scaled Index Mode: An INDEX register's contents is multiplied by a scaling factor which is added to a DISPLACEMENT to form the operands offset.
EXAMPLE: IMUL EBX, TABLE[ESI*4],7

Based Index Mode: The contents of a BASE register is added to the contents of an INDEX register to form the effective address of an operand.
EXAMPLE: MOV EAX, [ESI] [EBX]

Based Scaled Index Mode: The contents of an INDEX register is multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operands offset.
EXAMPLE: MOV ECX, [EDX*8] [EAX]

Based Index Mode with Displacement: The contents of an INDEX Register and a BASE register's contents and a DISPLACEMENT are all summed together to form the operand offset.
EXAMPLE: ADD EDX, [ESI] [EBP + 00FFFFF0H]

Based Scaled Index Mode with Displacement: The contents of an INDEX register are multiplied by a SCALING factor, the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.
EXAMPLE: MOV EAX, LOCALTABLE[EDI*4] [EBP + 80]

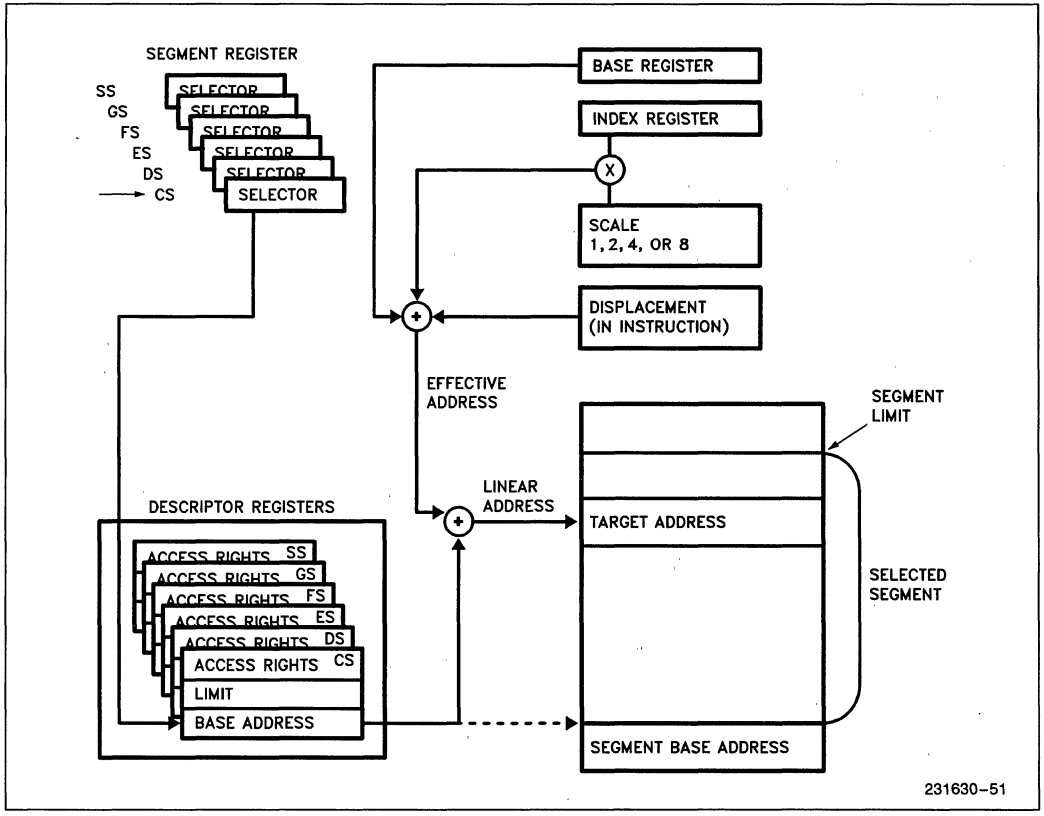


Figure 2-9. Addressing Mode Calculations

2.5.4 Differences Between 16 and 32 Bit Addresses

In order to provide software compatibility with the 80286 and the 8086, the Intel386 DX can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in the CS segment Descriptor. If the D bit is 0 then all operand lengths and effective addresses are assumed to be 16 bits long. If the D bit is 1 then the default length for operands and addresses is 32 bits. In Real Mode the default size for operands and addresses is 16-bits.

Regardless of the default precision of the operands or addresses, the Intel386 DX is able to execute either 16 or 32-bit instructions. This is specified via the use of override prefixes. Two prefixes, the **Operand Size Prefix** and the **Address Length Prefix**, override the value of the D bit on an individual instruction basis. These prefixes are automatically added by Intel assemblers.

Example: The processor is executing in Real Mode and the programmer needs to access the EAX registers. The assembler code for this might be MOV EAX, 32-bit MEMORYOP, ASM386 Macro Assembler automatically determines that an Operand Size Prefix is needed and generates it.

Example: The D bit is 0, and the programmer wishes to use Scaled Index addressing mode to access an array. The Address Length Prefix allows the use of MOV DX, TABLE[ESI*2]. The assembler uses an Address Length Prefix since, with D=0, the default addressing mode is 16-bits.

Example: The D bit is 1, and the program wants to store a 16-bit quantity. The Operand Length Prefix is used to specify only a 16-bit value; MOV MEM16, DX.

Table 2-3. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX,BP	Any 32-bit GP Register
INDEX REGISTER	SI,DI	Any 32-bit GP Register Except ESP
SCALE FACTOR	none	1, 2, 4, 8
DISPLACEMENT	0, 8, 16 bits	0, 8, 32 bits

The OPERAND LENGTH and Address Length Prefixes can be applied separately or in combination to any instruction. The Address Length Prefix does not allow addresses over 64K bytes to be accessed in Real Mode. A memory address which exceeds FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional Intel386 DX addressing modes.

When executing 32-bit code, the Intel386 DX uses either 8-, or 32-bit displacements, and any register can be used as base or index registers. When executing 16-bit code, the displacements are either 8, or 16 bits, and the base and index register conform to the 80286 model. Table 2-3 illustrates the differences.

2.6 DATA TYPES

The Intel386 DX supports all of the data types commonly used in high level languages:

Bit: A single bit quantity.

Bit Field: A group of up to 32 contiguous bits, which spans a maximum of four bytes.

Bit String: A set of contiguous bits, on the Intel386 DX bit strings can be up to 4 gigabits long.

Byte: A signed 8-bit quantity.

Unsigned Byte: An unsigned 8-bit quantity.

Integer (Word): A signed 16-bit quantity.

Long Integer (Double Word): A signed 32-bit quantity. All operations assume a 2's complement representation.

Unsigned Integer (Word): An unsigned 16-bit quantity.

Unsigned Long Integer (Double Word): An unsigned 32-bit quantity.

Signed Quad Word: A signed 64-bit quantity.

Unsigned Quad Word: An unsigned 64-bit quantity.

Offset: A 16- or 32-bit offset only quantity which indirectly references another memory location.

Pointer: A full pointer which consists of a 16-bit segment selector and either a 16- or 32-bit offset.

Char: A byte representation of an ASCII Alphanumeric or control character.

String: A contiguous sequence of bytes, words or dwords. A string may contain between 1 byte and 4 Gbytes.

BCD: A byte (unpacked) representation of decimal digits 0–9.

Packed BCD: A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

When the Intel386 DX is coupled with an Intel387 DX Numerics Coprocessor then the following common Floating Point types are supported.

Floating Point: A signed 32-, 64-, or 80-bit real number representation. Floating point numbers are supported by the Intel387 DX numerics coprocessor.

Figure 2-10 illustrates the data types supported by the Intel386 DX and the Intel387 DX numerics coprocessor.

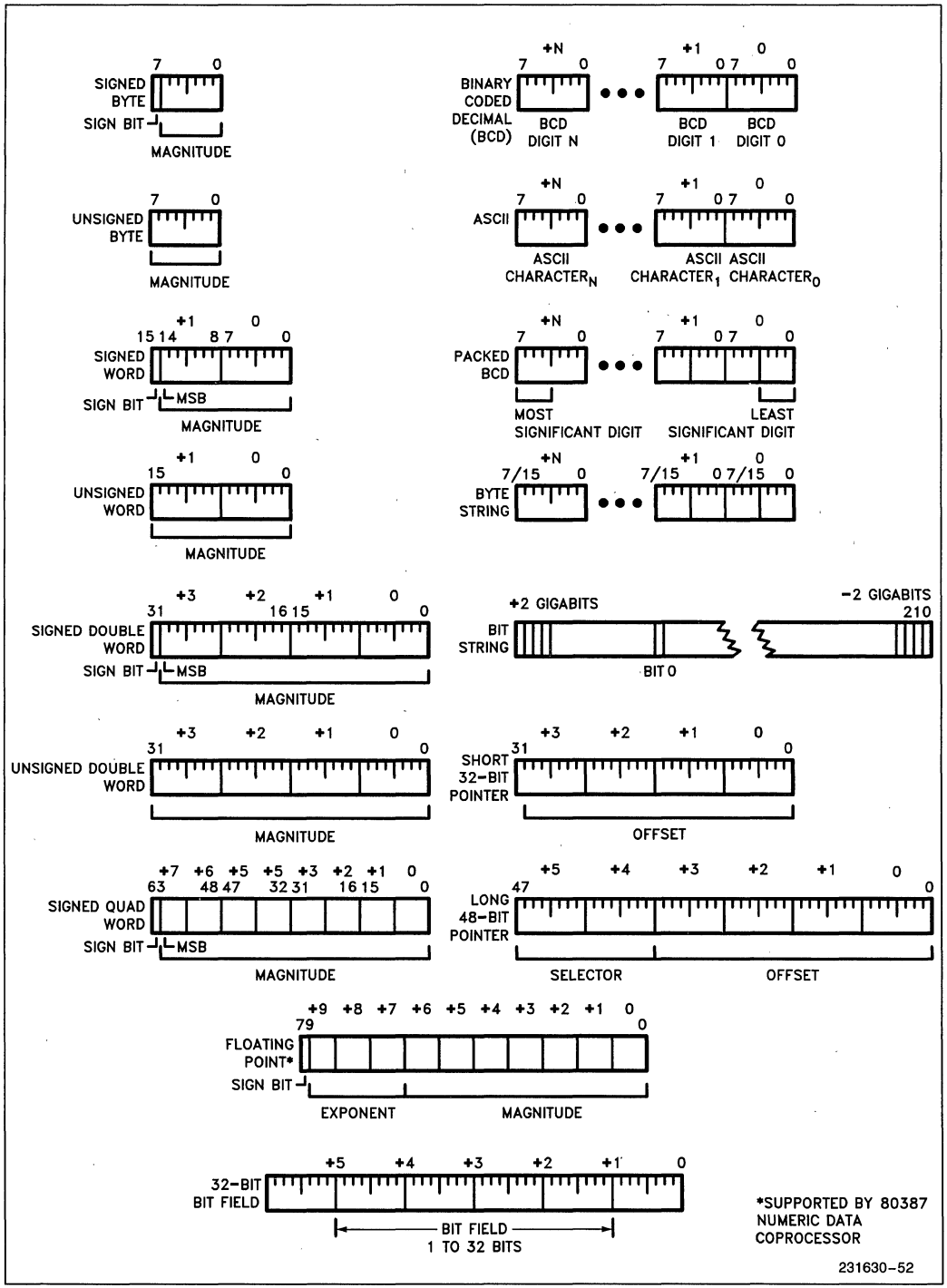


Figure 2-10. Intel386™ DX Supported Data Types

2.7 MEMORY ORGANIZATION

2.7.1 Introduction

Memory on the Intel386 DX is divided up into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address, the high order byte at the high address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address, the high-order byte at the highest address. The address of a word or dword is the byte address of the low-order byte.

In addition to these basic data types, the Intel386 DX supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4K byte pages. Finally, both segmentation and paging can be combined, gaining the advantages of both systems. The Intel386 DX supports both pages and segments in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and as such is a tool for the application programmer, while pages are useful for the system programmer for managing the physical memory of a system.

2.7.2 Address Spaces

The Intel386 DX has three distinct address spaces: **logical, linear,** and **physical**. A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, DISPLACEMENT) discussed in section 2.5.3 **Memory Addressing Modes** into an effective address. Since each task on Intel386 DX has a maximum of 16K ($2^{14} - 1$) selectors, and offsets can be 4 gigabytes, (2^{32} bits) this gives a total of 2^{46} bits or 64 terabytes of **logical** address space per task. The programmer sees this virtual address space.

The segmentation unit translates the **logical** address space into a 32-bit **linear** address space. If the paging unit is not enabled then the 32-bit **linear** address corresponds to the **physical** address. The paging unit translates the **linear** address space into the **physical** address space. The **physical address** is what appears on the address pins.

The primary difference between Real Mode and Protected Mode is how the segmentation unit performs the translation of the **logical** address into the **linear** address. In Real Mode, the segmentation unit shifts the selector left four bits and adds the result to the offset to form the **linear** address. While in Protected Mode every selector has a **linear** base address associated with it. The **linear base** address is stored in one of two operating system tables (i.e. the Local Descriptor Table or Global Descriptor Table). The selector's **linear base** address is added to the offset to form the final **linear** address.

Figure 2-11 shows the relationship between the various address spaces.

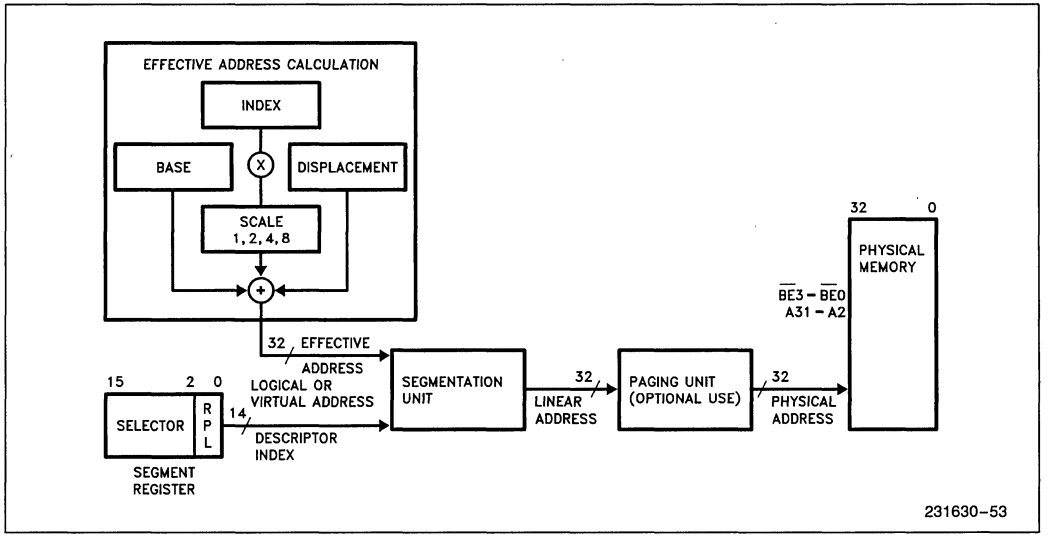


Figure 2-11. Address Translation

2.7.3 Segment Register Usage

The main data structure used to organize memory is the segment. On the Intel386 DX, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments: code and data, the segments are of variable size and can be as small as 1 byte or as large as 4 gigabytes (2^{32} bytes).

In order to provide compact instruction encoding, and increase processor performance, instructions do not need to explicitly specify which segment register is used. A default segment register is automatically chosen according to the rules of Table 2-4 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register; Stack references use the SS register and Instruction fetches use the CS register. The contents of the Instruction Pointer provides the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 2-4. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero and create a system with a four gigabyte linear address space. This creates a system where the virtual address space is the same as the linear address space. Further details of segmentation are discussed in section 4.1.

2.8 I/O SPACE

The Intel386 DX has two distinct physical address spaces: Memory and I/O. Generally, peripherals are placed in I/O space although the Intel386 DX also supports memory-mapped peripherals. The I/O space consists of 64K bytes, it can be divided into 64K 8-bit ports, 32K 16-bit ports, or 16K 32-bit ports, or any combination of ports which add up to less than 64K bytes. The 64K I/O address space refers to physical memory rather than linear address since I/O instructions do not go through the segmentation or paging hardware. The M/IO# pin acts as an additional address line thus allowing the system designer to easily determine which address space the processor is accessing.

Table 2-4. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References, with Effective Address Using Base Register of:		
[EAX]	DS	DS,CS,SS,ES,FS,GS
[EBX]	DS	DS,CS,SS,ES,FS,GS
[ECX]	DS	DS,CS,SS,ES,FS,GS
[EDX]	DS	DS,CS,SS,ES,FS,GS
[ESI]	DS	DS,CS,SS,ES,FS,GS
[EDI]	DS	DS,CS,SS,ES,FS,GS
[EBP]	SS	DS,CS,SS,ES,FS,GS
[ESP]	SS	DS,CS,SS,ES,FS,GS

The I/O ports are accessed via the IN and OUT I/O instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8- and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/IO# pin to be driven low.

I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

2.9 INTERRUPTS

2.9.1 Interrupts and Exceptions

Interrupts and exceptions alter the normal program flow, in order to handle external events, to report errors or exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction. Sections 2.9.3 and 2.9.4 discuss the differences between Maskable and Non-Maskable interrupts.

Exceptions are classified as faults, traps, or aborts depending on the way they are reported, and whether or not restart of the instruction causing the exception is supported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. A fault would occur in a virtual memory system, when the processor referenced a page or a segment which was not present. The operating system would fetch the page or segment from disk, and then the Intel386 DX would restart the instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction which caused the problem. User defined interrupts are examples of traps. **Aborts** are exceptions which do not permit the precise location of the instruction causing the exception to be determined. Aborts are used to report severe errors, such as a hardware error, or illegal values in system tables.

Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point at the instruction causing the exception and include any leading instruction prefixes. Table 2-5 summarizes the possible interrupts for the Intel386 DX and shows where the return address points.

The Intel386 DX has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode (see section 3.1), the vectors are 4 byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8 byte quantities, which are put in an Interrupt Descriptor Table (see section 4.1). Of the 256 possible interrupts, 32 are reserved for use by Intel, the remaining 224 are free to be used by the system designer.

2.9.2 Interrupt Processing

When an interrupt occurs the following actions happen. First, the current program address and the Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the Intel386 DX which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the Intel386 DX in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

2.9.3 Maskable Interrupt

Maskable interrupts are the most common way used by the Intel386 DX to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled high and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions, (REPeat String instructions, have an "interrupt window", between memory moves, which allows interrupts during long

Table 2-5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	any instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any Illegal Instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any Instruction That Can Generate an Exception		ABORT
Coprocessor Segment Overrun	9	ESC	NO	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	11	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Intel Reserved	15			
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Coprocessor Error	16	ESC, WAIT	YES	FAULT
Intel Reserved	17–31			
Two Byte Interrupt	0–255	INT n	NO	TRAP

* Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

string moves). When an interrupt occurs the processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt, (one of 224 user defined interrupts). The exact nature of the interrupt sequence is discussed in section 5.

The IF bit in the EFLAG registers is reset when an interrupt is being serviced. This effectively disables servicing additional interrupts during an interrupt service routine. However, the IF may be set explicitly by the interrupt handler, to allow the nesting of interrupts. When an IRET instruction is executed the original state of the IF is restored.

2.9.4 Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. A common example of the use of a non-maskable interrupt (NMI) would be to activate a power failure routine. When the NMI input is pulled high it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for an NMI.

While executing the NMI servicing procedure, the Intel386 DX will not service further NMI requests, until an interrupt return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

2.9.5 Software Interrupts

A third type of interrupt/exception for the Intel386 DX is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the nth vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt, is the single step interrupt. It is discussed in section 2.12.

2.9.6 Interrupt and Exception Priorities

Interrupts are externally-generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are **both** recognized at the **same** instruction boundary, the Intel386 DX invokes the NMI service routine first. If, after the NMI service routine has been invoked, maskable interrupts are still enabled, then the Intel386 DX will invoke the appropriate interrupt service routine.

Table 2-6a. Intel386™ DX Priority for Invoking Service Routines in Case of Simultaneous External Interrupts

1. NMI
2. INTR

Exceptions are internally-generated events. Exceptions are detected by the Intel386 DX if, in the course of executing an instruction, the Intel386 DX detects a problematic condition. The Intel386 DX then immediately invokes the appropriate exception service routine. The state of the Intel386 DX is such that the instruction causing the exception can be restarted. If the exception service routine has taken care of the problematic condition, the instruction will execute without causing the same exception.

It is possible for a single instruction to generate several exceptions (for example, transferring a single operand could generate two page faults if the operand location spans two "not present" pages). However, only one exception is generated upon each attempt to execute the instruction. Each exception service routine should correct its corresponding exception, and restart the instruction. In this manner, exceptions are serviced until the instruction executes successfully.

As the Intel386 DX executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 2-6b. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution.

Table 2-6b. Sequence of Exception Checking

Consider the case of the Intel386 DX having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
3. Check for external NMI and INTR.
4. Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5. Check for Page Faults that prevented fetching the entire next instruction (exception 14).
6. Check for Faults decoding the next instruction (exception 6 if illegal opcode; exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only (see 4.6.4); or exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e. not at IOPL or at CPL = 0).
7. If WAIT opcode, check if TS = 1 and MP = 1 (exception 7 if both are 1).
8. If ESCAPE opcode for numeric coprocessor, check if EM = 1 or TS = 1 (exception 7 if either are 1).
9. If WAIT opcode or ESCAPE opcode for numeric coprocessor, check ERROR# input signal (exception 16 if ERROR# input is asserted).
10. Check in the following order for each memory reference required by the instruction:
    a. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13).
    b. Check for Page Faults that prevent transferring the entire memory quantity (exception 14).

Note that the order stated supports the concept of the paging mechanism being "underneath" the segmentation mechanism. Therefore, for any given code or data reference in memory, segmentation exceptions are generated before paging exceptions are generated.

2.9.7 Instruction Restart

The Intel386 DX fully supports restarting all instructions after faults. If an exception is detected in the instruction to be executed (exception categories 4 through 10 in Table 2-6b), the Intel386 DX invokes the appropriate exception service routine. The Intel386 DX is in a state that permits restart of the instruction, for all cases but those in Table 2-6c. Note that all such cases are easily avoided by proper design of the operating system.

Table 2-6c. Conditions Preventing Instruction Restart

A. An instruction causes a task switch to a task whose Task State Segment is **partially** "not present". (An entirely "not present" TSS is restartable.) Partially present TSS's can be avoided either by keeping the TSS's of such tasks present in memory, or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4K bytes or less).

B. A coprocessor operand wraps around the top of a 64K-byte segment or a 4G-byte segment, and spans three pages, and the page holding the middle portion of the operand is "not present." This condition can be avoided by starting **at a page boundary** any segments containing coprocessor operands if the segments are approximately 64K-200 bytes or larger (i.e. large enough for wraparound of the coprocessor operand to possibly occur).

Note that these conditions are avoided by using the operating system designs mentioned in this table.

2.9.8 Double Fault

A Double Fault (exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13), but in the process of doing so, detects an exception **other than** a Page Fault (exception 14).

A Double Fault (exception 8) will also be generated when the processor attempts to invoke the Page Fault (exception 14) service routine, and detects an exception other than a second Page Fault. In any functional system, the entire Page Fault service routine must remain "present" in memory.

Double page faults however do not raise the double fault exception. If a second page fault occurs while the processor is attempting to enter the service routine for the first time, then the processor will invoke the page fault (exception 14) handler a second time, rather than the double fault (exception 8) handler. A subsequent fault, though, will lead to shutdown.

When a Double Fault occurs, the Intel386 DX invokes the exception service routine for exception 8.

2.10 RESET AND INITIALIZATION

When the processor is initialized or Reset the registers have the values shown in Table 2-7. The Intel386 DX will then start executing instructions near the top of physical memory, at location FFFFFFF0H. When the first InterSegment Jump or Call is executed, address lines A20-31 will drop low for CS-relative memory cycles, and the Intel386 DX will only execute instructions in the lower one megabyte of physical memory. This allows the system designer to use a ROM at the top of physical memory to initialize the system and take care of Resets.

RESET forces the Intel386 DX to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350 and 450 CLK2 periods after Reset becomes inactive the Intel386 DX will start executing instructions at the top of physical memory.

Table 2-7. Register Values after Reset

Flag Word	UUUU0002H	Note 1
Machine Status Word (CR0)	UUUUUUU0H	Note 2
Instruction Pointer	0000FFF0H	
Code Segment	F000H	Note 3
Data Segment	0000H	
Stack Segment	0000H	
Extra Segment (ES)	0000H	
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
DX register	component and stepping ID	Note 5
All other registers	undefined	Note 4

NOTES:
1. EFLAG Register. The upper 14 bits of the EFLAGS register are undefined, VM (Bit 17) and RF (BIT) 16 are 0 as are all other defined flag bits.
2. CR0: (Machine Status Word). All of the defined fields in the CR0 are 0 (PG Bit 31, TS Bit 3, EM Bit 2, MP Bit 1, and PE Bit 0).
3. The Code Segment Register (CS) will have its Base Address set to FFFF0000H and Limit set to 0FFFFH.
4. All undefined bits are Intel Reserved and should not be used.
5. DX register always holds component and stepping identifier (see 5.7). EAX register holds self-test signature if self-test was requested (see 5.6).

2.11 TESTABILITY

2.11.1 Self-Test

The Intel386 DX has the capability to perform a self-test. The self-test checks the function of all of the Control ROM and most of the non-random logic of the part. Approximately one-half of the Intel386 DX can be tested during self-test.

Self-Test is initiated on the Intel386 DX when the RESET pin transitions from HIGH to LOW, and the BUSY# pin is low. The self-test takes about 2**19 clocks, or approximately 26 milliseconds with a 20 MHz Intel386 DX. At the completion of self-test the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX register are zero (0). If the results of EAX are not zero then the self-test has detected a flaw in the part.

2.11.2 TLB Testing

The Intel386 DX provides a mechanism for testing the Translation Lookaside Buffer (TLB) if desired. This particular mechanism is unique to the Intel386 DX and may not be continued in the same way in future processors. When testing the TLB paging must be turned off (PG = 0 in CR0) to enable the TLB testing hardware and avoid interference with the test data being written to the TLB.

There are two TLB testing operations: 1) write entries into the TLB, and, 2) perform TLB lookups. Two Test Registers, shown in Figure 2-12, are provided for the purpose of testing. TR6 is the "test command register", and TR7 is the "test data register". The fields within these registers are defined below.

C: This is the command bit. For a write into TR6 to cause an immediate write into the TLB entry, write a 0 to this bit. For a write into TR6 to cause an immediate TLB lookup, write a 1 to this bit.

Linear Address: This is the tag field of the TLB. On a TLB write, a TLB entry is allocated to this linear address and the rest of that TLB entry is set per the value of TR7 and the value just written into TR6. On a TLB lookup, the TLB is interrogated per this value and if one and only one TLB entry matches, the rest of the fields of TR6 and TR7 are set from the matching TLB entry.

Physical Address: This is the data field of the TLB. On a write to the TLB, the TLB entry allocated to the linear address in TR6 is set to this value. On a TLB lookup, the data field (physical address) from the TLB is read out to here.

PL: On a TLB write, PL=1 causes the REP field of TR7 to select which of four associative blocks of the TLB is to be written, but PL=0 allows the internal pointer in the paging unit to select which TLB block is written. On a TLB lookup, the PL bit indicates whether the lookup was a hit (PL gets set to 1) or a miss (PL gets reset to 0).

V: The valid bit for this TLB entry. All valid bits can also be cleared by writing to CR3.

D, D#: The dirty bit for/from the TLB entry.

U, U#: The user bit for/from the TLB entry.

W, W#: The writable bit for/from the TLB entry.

For D, U and W, both the attribute and its complement are provided as tag bits, to permit the option of a "don't care" on TLB lookups. The meaning of these pairs of bits is given in the following table:

X	X#	Effect During TLB Lookup	Value of Bit X after TLB Write
0	0	Miss All	Bit X Becomes Undefined
0	1	Match if X = 0	Bit X Becomes 0
1	0	Match if X = 1	Bit X Becomes 1
1	1	Match all	Bit X Becomes Undefined

For writing a TLB entry:

1. Write TR7 for the desired physical address, PL and REP values.
2. Write TR6 with the appropriate linear address, etc. (be sure to write C = 0 for "write" command).

For looking up (reading) a TLB entry:

1. Write TR6 with the appropriate linear address (be sure to write C=1 for "lookup" command).
2. Read TR7 and TR6. If the PL bit in TR7 indicates a hit, then the other values reveal the TLB contents. If PL indicates a miss, then the other values in TR7 and TR6 are indeterminate.

2.12 DEBUGGING SUPPORT

The Intel386 DX provides several features which simplify the debugging process. The three categories of on-chip debugging aids are:

1) the code execution breakpoint opcode (0CCH),
2) the single-step capability provided by the TF bit in the flag register, and
3) the code and data breakpoint capability provided by the Debug Registers DR0-3, DR6, and DR7.

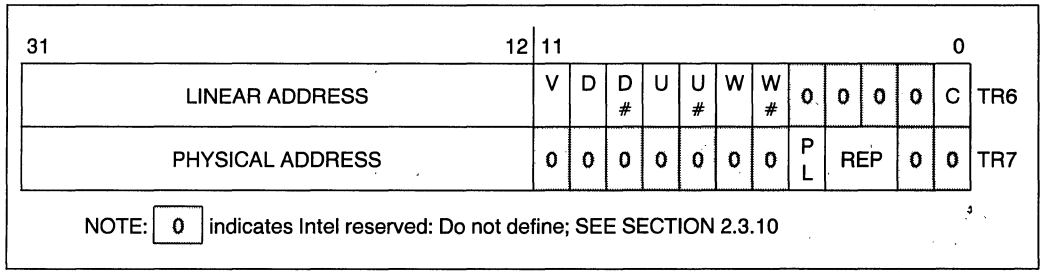


Figure 2-12. Test Registers

2.12.1 Breakpoint Instruction

A single-byte-opcode breakpoint instruction is available for use by software debuggers. The breakpoint opcode is 0CCh, and generates an exception 3 trap when executed. In typical use, a debugger program can "plant" the breakpoint instruction at all desired code execution breakpoints. The single-byte breakpoint opcode is an alias for the two-byte general software interrupt instruction, INT n, where n=3. The only difference between INT 3 (0CCh) and INT n is that INT 3 is never IOPL-sensitive but INT n is IOPL-sensitive in Protected Mode and Virtual 8086 Mode.

2.12.2 Single-Step Trap

If the single-step flag (TF, bit 8) in the EFLAG register is found to be set at the end of an instruction, a single-step exception occurs. The single-step exception is auto vectored to exception number 1. Precisely, exception 1 occurs as a trap after the instruction following the instruction which set TF. In typical practice, a debugger sets the TF bit of a flag register image on the debugger's stack. It then typically transfers control to the user program and loads the flag image with a signal instruction, the IRET instruction. The single-step trap occurs after executing one instruction of the user program.

Since the exception 1 occurs as a trap (that is, it occurs after the instruction has already executed), the CS:EIP pushed onto the debugger's stack points to the next unexecuted instruction of the program being debugged. An exception 1 handler, merely by ending with an IRET instruction, can therefore efficiently support single-stepping through a user program.

2.12.3 Debug Registers

The Debug Registers are an advanced debugging feature of the Intel386 DX. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT3 breakpoint opcode.

The Intel386 DX contains six Debug Registers, providing the ability to specify up to four distinct breakpoints addresses, breakpoint control options, and read breakpoint status. Initially after reset, breakpoints are in the disabled state. Therefore, no breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are autovectored to exception number 1.

2.12.3.1 LINEAR ADDRESS BREAKPOINT REGISTERS (DR0–DR3)

Up to four breakpoint addresses can be specified by writing into Debug Registers DR0–DR3, shown in Figure 2-13. The breakpoint addresses specified are 32-bit linear addresses. Intel386 DX hardware continuously compares the linear breakpoint addresses in DR0–DR3 with the linear addresses generated by executing software (a linear address is the result of computing the effective address and adding the 32-bit segment base address). Note that if paging is not enabled the linear address equals the physical address. If paging is enabled, the linear address is translated to a physical 32-bit address by the on-chip paging unit. Regardless of whether paging is enabled or not, however, the breakpoint registers hold linear addresses.

2.12.3.2 DEBUG CONTROL REGISTER (DR7)

A Debug Control Register, DR7 shown in Figure 2-13, allows several debug control functions such as enabling the breakpoints and setting up other control options for the breakpoints. The fields within the Debug Control Register, DR7, are as follows:

LENi (breakpoint length specification bits)

A 2-bit LEN field exists for each of the four breakpoints. LEN specifies the length of the associated breakpoint field. The choices for data breakpoints are: 1 byte, 2 bytes, and 4 bytes. Instruction execu-

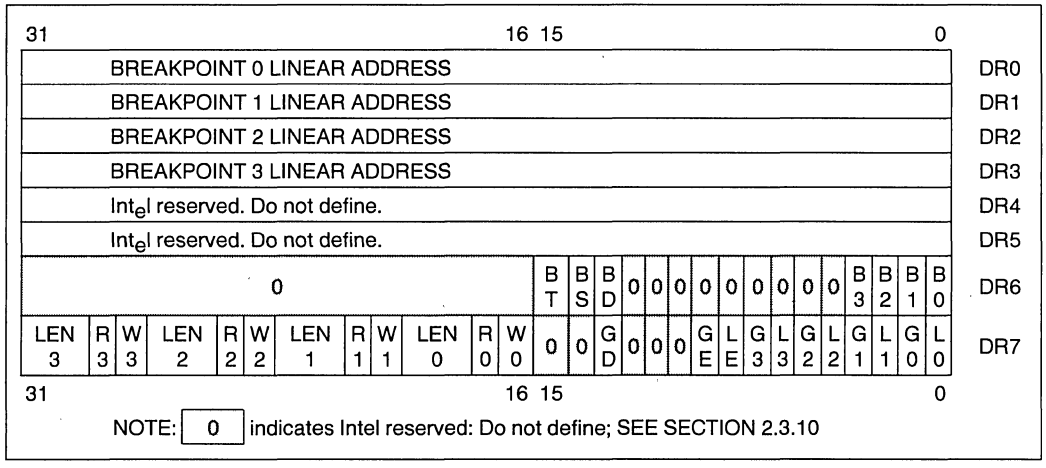


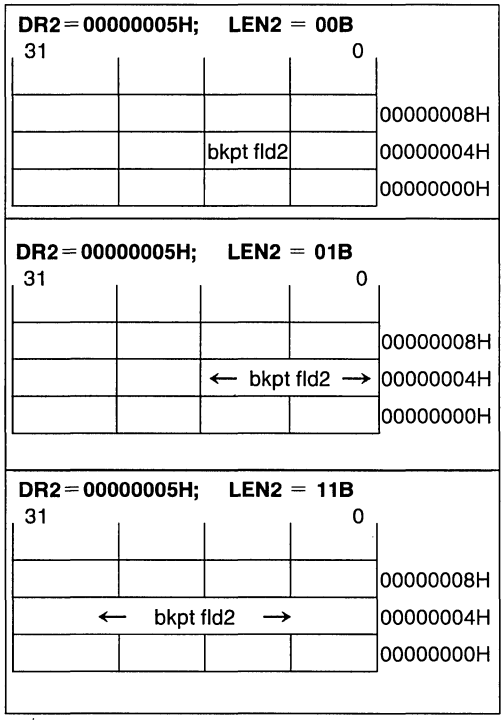
Figure 2-13. Debug Registers

tion breakpoints must have a length of 1 (LENi = 00). Encoding of the LENi field is as follows:

LENi Encoding	Breakpoint Field Width	Usage of Least Significant Bits in Breakpoint Address Register i, (i = 0 – 3)
00	1 byte	All 32-bits used to specify a single-byte breakpoint field.
01	2 bytes	A1–A31 used to specify a two-byte, word-aligned breakpoint field. A0 in Breakpoint Address Register is not used.
10	Undefined—do not use this encoding	
11	4 bytes	A2–A31 used to specify a four-byte, dword-aligned breakpoint field. A0 and A1 in Breakpoint Address Register are not used.

The LENi field controls the size of breakpoint field i by controlling whether all low-order linear address bits in the breakpoint address register are used to detect the breakpoint event. Therefore, all breakpoint fields are aligned; 2-byte breakpoint fields begin on Word boundaries, and 4-byte breakpoint fields begin on Dword boundaries.

The following is an example of various size breakpoint fields. Assume the breakpoint linear address in DR2 is 00000005H. In that situation, the following illustration indicates the region of the breakpoint field for lengths of 1, 2, or 4 bytes.



RWi (memory access qualifier bits)

A 2-bit RW field exists for each of the four breakpoints. The 2-bit RW field specifies the type of usage which must occur in order to activate the associated breakpoint.

RW Encoding	Usage Causing Breakpoint
00	Instruction execution only
01	Data writes only
10	Undefined—do not use this encoding
11	Data reads and writes only

RW encoding 00 is used to set up an instruction execution breakpoint. RW encodings 01 or 11 are used to set up write-only or read/write data breakpoints.

Note that **instruction execution breakpoints are taken as faults** (i.e. before the instruction executes), but **data breakpoints are taken as traps** (i.e. after the data transfer takes place).

Using LENi and RWi to Set Data Breakpoint i

A data breakpoint can be set up by writing the linear address into DRi (i = 0–3). For data breakpoints, RWi can = 01 (write-only) or 11 (write/read). LEN can = 00, 01, or 11.

If a data access entirely or partly falls within the data breakpoint field, the data breakpoint condition has occurred, and if the breakpoint is enabled, an exception 1 trap will occur.

Using LENi and RWi to Set Instruction Execution Breakpoint i

An instruction execution breakpoint can be set up by writing address of the beginning of the instruction (including prefixes if any) into DRi (i = 0–3). RWi must = 00 and LEN must = 00 for instruction execution breakpoints.

If the instruction beginning at the breakpoint address is about to be executed, the instruction execution breakpoint condition has occurred, and if the breakpoint is enabled, an exception 1 fault will occur before the instruction is executed.

Note that an instruction execution breakpoint address must be equal to the **beginning** byte address of an instruction (including prefixes) in order for the instruction execution breakpoint to occur.

GD (Global Debug Register access detect)

The Debug Registers can only be accessed in Real Mode or at privilege level 0 in Protected Mode. The GD bit, when set, provides extra protection against **any** Debug Register access even in Real Mode or at privilege level 0 in Protected Mode. This additional protection feature is provided to guarantee that a software debugger (or ICE™-386) can have full control over the Debug Register resources when required. The GD bit, when set, causes an exception 1 fault if an instruction attempts to read or write any Debug Register. The GD bit is then automatically cleared when the exception 1 handler is invoked, allowing the exception 1 handler free access to the debug registers.

GE and LE (Exact data breakpoint match, global and local)

If either GE or LE is set, any data breakpoint trap will be reported exactly after completion of the instruction that caused the operand transfer. Exact reporting is provided by forcing the Intel386 DX execution unit to wait for completion of data operand transfers before beginning execution of the next instruction.

If exact data breakpoint match is not selected, data breakpoints may not be reported until several instructions later or may not be reported at all. When enabling a data breakpoint, it is therefore recommended to enable the exact data breakpoint match.

When the Intel386 DX performs a task switch, the LE bit is cleared. Thus, the LE bit supports fast task switching out of tasks, that have enabled the exact data breakpoint match for their task-local breakpoints. The LE bit is cleared by the processor during a task switch, to avoid having exact data breakpoint match enabled in the new task. Note that exact data breakpoint match must be re-enabled under software control.

The Intel386 DX GE bit is unaffected during a task switch. The GE bit supports exact data breakpoint match that is to remain enabled during all tasks executing in the system.

Note that **instruction execution** breakpoints are always reported exactly, whether or not exact data breakpoint match is selected.

Gi and Li (breakpoint enable, global and local)

If either Gi or Li is set then the associated breakpoint (as defined by the linear address in DRi, the length in LENi and the usage criteria in RWi) is enabled. If either Gi or Li is set, and the Intel386 DX detects the ith breakpoint condition, then the exception 1 handler is invoked.

When the Intel386 DX performs a task switch to a new Task State Segment (TSS), all Li bits are cleared. Thus, the Li bits support fast task switching out of tasks that use some task-local breakpoint

registers. The Li bits are cleared by the processor during a task switch, to avoid spurious exceptions in the new task. Note that the breakpoints must be re-enabled under software control.

All Intel386 DX Gi bits are unaffected during a task switch. The Gi bits support breakpoints that are active in all tasks executing in the system.

2.12.3.3 DEBUG STATUS REGISTER (DR6)

A Debug Status Register, DR6 shown in Figure 2-13, allows the exception 1 handler to easily determine why it was invoked. Note the exception 1 handler can be invoked as a result of one of several events:

1) DR0 Breakpoint fault/trap.
2) DR1 Breakpoint fault/trap.
3) DR2 Breakpoint fault/trap.
4) DR3 Breakpoint fault/trap.
5) Single-step (TF) trap.
6) Task switch trap.
7) Fault due to attempted debug register access when GD=1.

The Debug Status Register contains single-bit flags for each of the possible events invoking exception 1. Note below that some of these events are faults (exception taken before the instruction is executed), while other events are traps (exception taken after the debug events occurred).

The flags in DR6 are set by the hardware but never cleared by hardware. Exception 1 handler software should clear DR6 before returning to the user program to avoid future confusion in identifying the source of exception 1.

The fields within the Debug Status Register, DR6, are as follows:

Bi (debug fault/trap due to breakpoint 0–3)

Four breakpoint indicator flags, B0–B3, correspond one-to-one with the breakpoint registers in DR0–DR3. A flag Bi is set when the condition described by DRi, LENi, and RWi occurs.

If Gi or Li is set, and if the ith breakpoint is detected, the processor will invoke the exception 1 handler. The exception is handled as a fault if an instruction execution breakpoint occurred, or as a trap if a data breakpoint occurred.

IMPORTANT NOTE: A flag Bi is set whenever the hardware detects a match condition on **enabled** breakpoint i. Whenever a match is detected on at least one **enabled** breakpoint i, the hardware immediately sets all Bi bits corresponding to breakpoint conditions matching at that instant, whether enabled **or not.** Therefore, the exception 1 handler may see that multiple Bi bits are set, but only set Bi bits corresponding to **enabled** breakpoints (Li or Gi set) are **true** indications of why the exception 1 handler was invoked.

BD (debug fault due to attempted register access when GD bit set)

This bit is set if the exception 1 handler was invoked due to an instruction attempting to read or write to the debug registers when GD bit was set. If such an event occurs, then the GD bit is automatically cleared when the exception 1 handler is invoked, allowing handler access to the debug registers.

BS (debug trap due to single-step)

This bit is set if the exception 1 handler was invoked due to the TF bit in the flag register being set (for single-stepping). See section 2.12.2.

BT (debug trap due to task switch)

This bit is set if the exception 1 handler was invoked due to a task switch occurring to a task having an Intel386 DX TSS with the T bit set. (See Figure 4-15a). Note the task switch into the new task occurs normally, but before the first instruction of the task is executed, the exception 1 handler is invoked. With respect to the task switch operation, the operation is considered to be a trap.

2.12.3.4 USE OF RESUME FLAG (RF) IN FLAG REGISTER

The Resume Flag (RF) in the flag word can suppress an instruction execution breakpoint when the exception 1 handler returns to a user program at a user address which is also an instruction execution breakpoint. See section 2.3.3.

3. REAL MODE ARCHITECTURE

3.1 REAL MODE INTRODUCTION

When the processor is reset or powered up it is initialized in Real Mode. Real Mode has the same base architecture as the 8086, but allows access to the 32-bit register set of the Intel386 DX. The addressing mechanism, memory size, interrupt handling, are all identical to the Real Mode on the 80286.

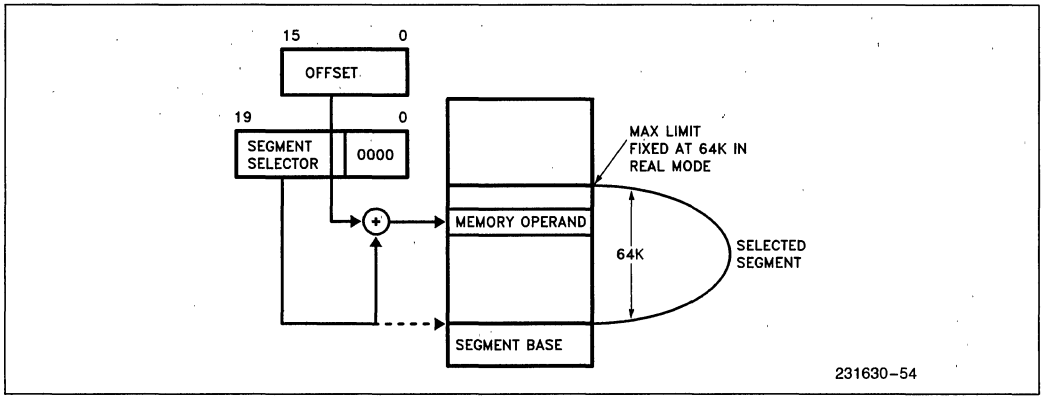


Figure 3-1. Real Address Mode Addressing

All of the Intel386 DX instructions are available in Real Mode (except those instructions listed in 4.6.4). The default operand size in Real Mode is 16-bits, just like the 8086. In order to use the 32-bit registers and addressing modes, override prefixes must be used. In addition, the segment size on the Intel386 DX in Real Mode is 64K bytes so 32-bit effective addresses must have a value less the 0000FFFFH. The primary purpose of Real Mode is to set up the processor for Protected Mode Operation.

The LOCK prefix on the Intel386 DX, even in Real Mode, is more restrictive than on the 80286. This is due to the addition of paging on the Intel386 DX in Protected Mode and Virtual 8086 Mode. Paging makes it impossible to guarantee that repeated string instructions can be LOCKed. The Intel386 DX can't require that all pages holding the string be physically present in memory. Hence, a Page Fault (exception 14) might have to be taken during the repeated string instruction. Therefore the LOCK prefix can't be supported during repeated string instructions.

These are the only instruction forms where the LOCK prefix is legal on the Intel386 DX:

Opcode	Operands (Dest, Source)
BIT Test and SET/RESET/COMPLEMENT	Mem, Reg/immed
XCHG	Reg, Mem
XCHG	Mem, Reg
ADD, OR, ADC, SBB, AND, SUB, XOR	Mem, Reg/immed
NOT, NEG, INC, DEC	Mem

An exception 6 will be generated if a LOCK prefix is placed before any instruction form or opcode not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above. For example, even the ADD Reg, Mem is not LOCKable, because the Mem operand is not the destination (and therefore no memory read/modify/operation is being performed).

Since, on the Intel386 DX, repeated string instructions are not LOCKable, it is not possible to LOCK the bus for a long period of time. Therefore, the LOCK prefix is not IOPL-sensitive on the Intel386 DX. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed above.

3.2 MEMORY ADDRESSING

In Real Mode the maximum memory size is limited to 1 megabyte. Thus, only address lines A2–A19 are active. (Exception, the high address lines A20–A31 are high during CS-relative memory cycles until an intersegment jump or call is executed (see section 2.10)).

Since paging is not allowed in Real Mode the linear addresses are the same as physical addresses. Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register which is shifted left by four bits to an effective address. This addition results in a physical address from 00000000H to 0010FFEFH. This is compatible with 80286 Real Mode. Since segment registers are shifted left by 4 bits this implies that Real Mode segments always start on 16 byte boundaries.

All segments in Real Mode are exactly 64K bytes long, and may be read, written, or executed. The Intel386 DX will generate an exception 13 if a data operand or instruction fetch occurs past the end of a segment. (i.e. if an operand has an offset greater than FFFFH, for example a word with a low byte at FFFFH and the high byte at 0000H.)

Segments may be overlapped in Real Mode. Thus, if a particular segment does not use all 64K bytes another segment can be overlayed on top of the unused portion of the previous segment. This allows the programmer to minimize the amount of physical memory needed for a program.

3.3 RESERVED LOCATIONS

There are two fixed areas in memory which are reserved in Real address mode: system initialization area and the interrupt table area. Locations 00000H through 003FFH are reserved for interrupt vectors. Each one of the 256 possible interrupts has a 4-byte jump vector reserved for it. Locations FFFFFFF0H through FFFFFFFFH are reserved for system initialization.

3.4 INTERRUPTS

Many of the exceptions shown in Table 2-5 and discussed in section 2.9 are not applicable to Real Mode operation, in particular exceptions 10, 11, 14, will not happen in Real Mode. Other exceptions have slightly different meanings in Real Mode; Table 3-1 identifies these exceptions.

3.5 SHUTDOWN AND HALT

The HLT instruction stops program execution and prevents the processor from using the local bus until restarted. Either NMI, INTR with interrupts enabled (IF = 1), or RESET will force the Intel386 DX out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

Shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions:

An interrupt or an exception occur (Exceptions 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table (i.e. There is not an interrupt handler for the interrupt).

A CALL, INT or PUSH instruction attempts to wrap around the stack segment when SP is not even. (e.g. pushing a value on the stack when SP = 0001 resulting in a stack segment greater than FFFFH)

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 0017H) and the stack has enough room to contain the vector and flag information (i.e. SP is greater than 0005H). Otherwise shutdown can only be exited via the RESET input.

4. PROTECTED MODE ARCHITECTURE

4.1 INTRODUCTION

The complete capabilities of the Intel386 DX are unlocked when the processor operates in Protected Virtual Address Mode (Protected Mode). Protected Mode vastly increases the linear address space to four gigabytes (2^{32} bytes) and allows the running of virtual memory programs of almost unlimited size (64 terabytes or 2^{46} bytes). In addition Protected Mode allows the Intel386 DX to run all of the existing 8086 and 80286 software, while providing a sophisticated memory management and a hardware-assisted protection mechanism. Protected Mode allows the use of additional instructions especially optimized for supporting multitasking operating systems. The base architecture of the Intel386 DX remains the same, the registers, instructions, and addressing modes described in the previous sections are retained. The main difference between Protected Mode, and Real Mode from a programmer's view is the increased address space, and a different addressing mechanism.

Table 3-1

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT Vector is not within table limit	Before Instruction
CS, DS, ES, FS, GS Segment overrun exception	13	Word memory reference beyond offset = FFFFH. An attempt to execute past the end of CS segment.	Before Instruction
SS Segment overrun exception	12	Stack Reference beyond offset = FFFFH	Before Instruction

4.2 ADDRESSING MECHANISM

Like Real Mode, Protected Mode uses two components to form the logical address, a 16-bit selector is used to determine the linear base address of a segment, the base address is added to a 32-bit effective address to form a 32-bit linear address. The linear address is then either used as the 32-bit physical address, or if paging is enabled the paging mechanism maps the 32-bit linear address into a 32-bit physical address.

The difference between the two modes lies in calculating the base address. In Protected Mode the selector is used to specify an index into an operating system defined table (see Figure 4-1). The table contains the 32-bit base address of a given segment. The physical address is formed by adding the base address obtained from the table to the offset.

Paging provides an additional memory management mechanism which operates only in Protected Mode. Paging provides a means of managing the very large segments of the Intel386 DX. As such, paging operates beneath segmentation. The paging mechanism translates the protected linear address which comes from the segmentation unit into a physical address. Figure 4-2 shows the complete Intel386 DX addressing mechanism with paging enabled.

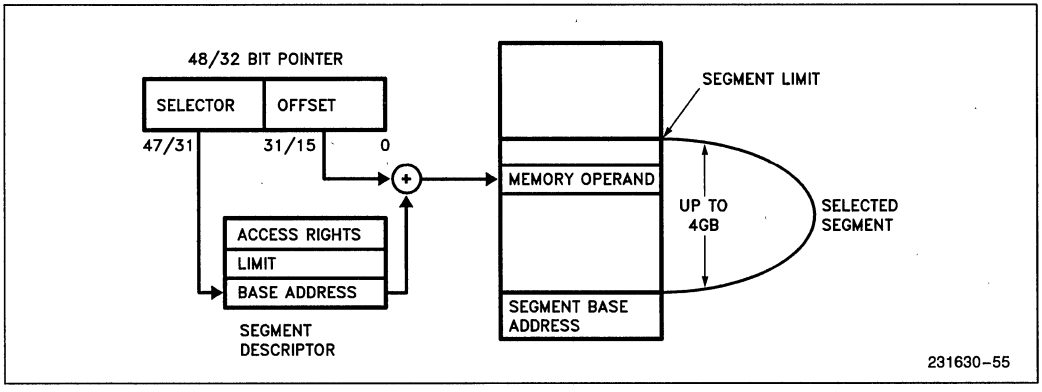


Figure 4-1. Protected Mode Addressing

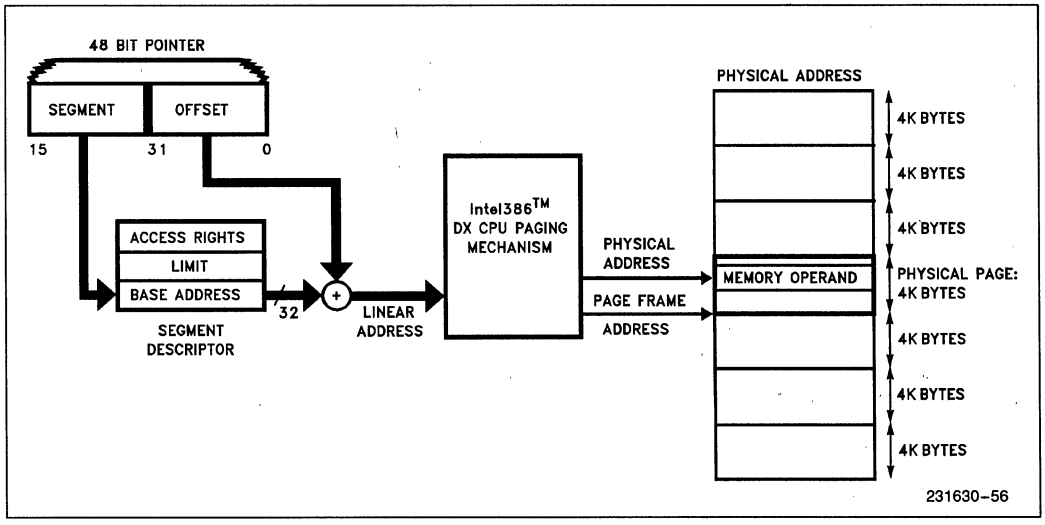


Figure 4-2. Paging and Segmentation

4.3 SEGMENTATION

4.3.1 Segmentation Introduction

Segmentation is one method of memory management. Segmentation provides the basis for protection. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about a segment is stored in an 8 byte data structure called a descriptor. All of the descriptors in a system are contained in tables recognized by hardware.

4.3.2 Terminology

The following terms are used throughout the discussion of descriptors, privilege levels and protection:

PL: Privilege Level—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged. More privileged levels are numerically smaller than less privileged levels.

RPL: Requestor Privilege Level—The privilege level of the original supplier of the selector. RPL is determined by the **least two** significant bits of a selector.

DPL: Descriptor Privilege Level—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

CPL: Current Privilege Level—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: Effective Privilege Level—The effective privilege level is the least privileged of the RPL and DPL. Since smaller privilege level **values** indicate greater privilege, EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

4.3.3 Descriptor Tables

4.3.3.1 DESCRIPTOR TABLES INTRODUCTION

The descriptor tables define all of the segments which are used in an Intel386 DX system. There are three types of tables on the Intel386 DX which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Descriptor Table. All of the tables are variable length memory arrays. They can range in size between 8 bytes and 64K bytes. Each table can hold up to 8192 8 byte descriptors. The upper 13 bits of a selector are used as an index into the descriptor table. The tables have registers associated with them which hold the 32-bit linear base address, and the 16-bit limit of each table.

Each of the tables has a register associated with it the GDTR, LDTR, and the IDTR (see Figure 4-3). The LGDT, LLDT, and LIDT instructions, load the base and limit of the Global, Local, and Interrupt Descriptor Tables, respectively, into the appropriate register. The SGDT, SLDT, and SIDT instructions store the base and limit values. These tables are manipulated by the operating system. Therefore, the load descriptor table instructions are privileged instructions.

4.3.3.2 GLOBAL DESCRIPTOR TABLE

The Global Descriptor Table (GDT) contains descriptors which are possibly available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for descriptors which are used for servicing interrupts (i.e. interrupt and trap descriptors). Every Intel386 DX system contains a

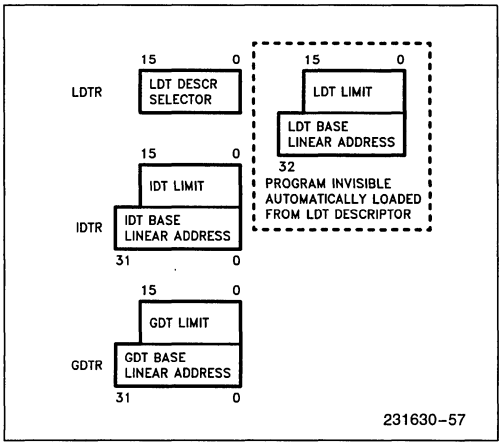


Figure 4-3. Descriptor Table Registers

GDT. Generally the GDT contains code and data segments used by the operating systems and task state segments, and descriptors for the LDTs in a system.

The first slot of the Global Descriptor Table corresponds to the null selector and is not used. The null selector defines a null pointer value.

4.3.3.3 LOCAL DESCRIPTOR TABLE

LDTs contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This provides both isolation and protection for a task's segments, while still allowing global data to be shared among tasks.

Unlike the 6 byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a Local Descriptor Table descriptor in the GDT.

4.3.3.4 INTERRUPT DESCRIPTOR TABLE

The third table needed for Intel386 DX systems is the Interrupt Descriptor Table. (See Figure 4-4.) The IDT contains the descriptors which point to the location of up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates, and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 Intel Reserved Interrupts. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced via INT instructions, external interrupt vectors, and exceptions. (See 2.9 **Interrupts).**

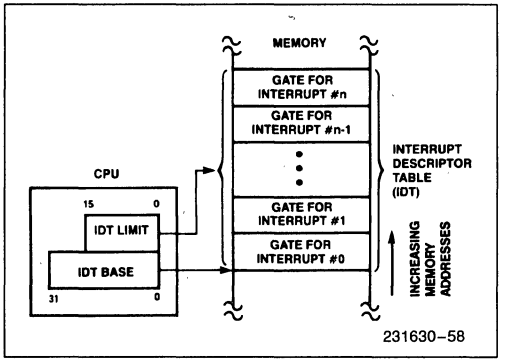
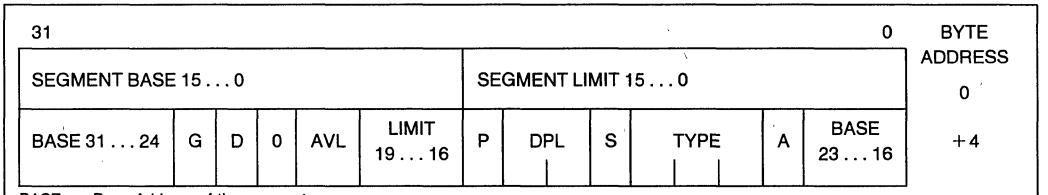


Figure 4-4. Interrupt Descriptor Table Register Use

4.3.4 Descriptors

4.3.4.1 DESCRIPTOR ATTRIBUTE BITS

The object to which the segment selector points to is called a descriptor. Descriptors are eight byte quantities which contain attributes about a given region of linear address space (i.e. a segment). These attributes include the 32-bit base linear address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write or execute privileges, the default size of the operands (16-bit or



BASE	Base Address of the segment
LIMIT	The length of the segment
P	Present Bit 1 = Present 0 = Not Present
DPL	Descriptor Privilege Level 0–3
S	Segment Descriptor 0 = System Descriptor 1 = Code or Data Segment Descriptor
TYPE	Type of Segment
A	Accessed Bit
G	Granularity Bit 1 = Segment length is page granular 0 = Segment length is byte granular
D	Default Operation Size (recognized in code segment descriptors only) 1 = 32-bit segment 0 = 16-bit segment
0	Bit must be zero (0) for compatibility with future processors
AVL	Available field for user or OS

NOTE:
In a maximum-size segment (ie. a segment with G = 1 and segment limit 19...0 = FFFFFH), the lowest 12 bits of the segment base should be zero (ie. segment base 11...000 = 000H).

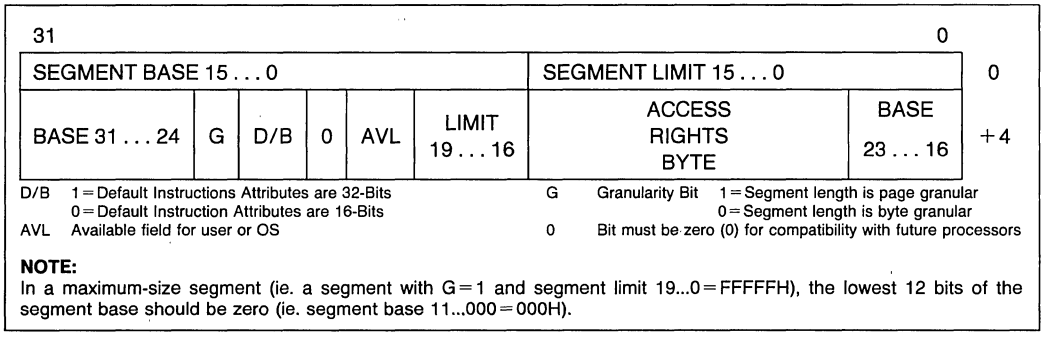
Figure 4-5. Segment Descriptors

32-bit), and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. Figure 4-5 shows the general format of a descriptor. All segments on the Intel386 DX have three attribute fields in common: the **P** bit, the **DPL** bit, and the **S** bit. The Present **P** bit is 1 if the segment is loaded in physical memory, if P = 0 then any attempt to access this segment causes a not present exception (exception 11). The Descriptor Privilege Level **DPL** is a two-bit field which specifies the protection level 0–3 associated with a segment.

The Intel386 DX has two main categories of segments system segments and non-system segments (for code and data). The segment **S** bit in the segment descriptor determines if a given segment is a system segment or a code or data segment. If the S bit is 1 then the segment is either a code or data segment, if it is 0 then the segment is a system segment.

4.3.4.2 Intel386™ DX CODE, DATA DESCRIPTORS (S = 1)

Figure 4-6 shows the general format of a code and data descriptor and Table 4-1 illustrates how the bits in the Access Rights Byte are interpreted.



NOTE:
In a maximum-size segment (ie. a segment with G = 1 and segment limit 19...0 = FFFFFH), the lowest 12 bits of the segment base should be zero (ie. segment base 11...000 = 000H).

Figure 4-6. Segment Descriptors

Table 4-1. Access Rights Byte Definition for Code and Data Descriptions

	Bit Position	Name	Function		
Type Field Definition	7	Present (P)	P = 1	Segment is mapped into physical memory.	
			P = 0	No mapping to physical memory exits, base and limit are not used.	
	6–5	Descriptor Privilege Level (DPL)		Segment privilege attribute used in privilege tests.	
	4	Segment Descriptor (S)	S = 1	Code or Data (includes stacks) segment descriptor	
			S = 0	System Segment Descriptor or Gate Descriptor	
	3	Executable (E)	E = 0	Descriptor type is data segment:	If Data Segment (S = 1, E = 0)
	2	Expansion Direction (ED)	ED 0	Expand up segment, offsets must be ≤ limit.	
			ED = 1	Expand down segment, offsets must be > limit.	
	1	Writeable (W)	W = 0	Data segment may not be written into.	
			W = 1	Data segment may be written into.	
	3	Executable (E)	E = 1	Descriptor type is code segment:	If Code Segment (S = 1, E = 1)
	2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
	1	Readable (R)	R = 0	Code segment may not be read.	
			R = 1	Code segment may be read.	
	0	Accessed (A)	A = 0	Segment has not been accessed.	
			A = 1	Segment selector has been loaded into segment register or used by selector test instructions.	

Code and data segments have several descriptor fields in common. The accessed **A** bit is set whenever the processor accesses a descriptor. The **A** bit is used by operating systems to keep usage statistics on a given segment. The **G** bit, or granularity bit, specifies if a segment length is byte-granular or page-granular. Intel386 DX segments can be one megabyte long with byte granularity (G=0) or four gigabytes with page granularity (G=1), (i.e., 2^{20} pages each page is 4K bytes in length). The granularity is totally unrelated to paging. An Intel386 DX system can consist of segments with byte granularity, and page granularity, whether or not paging is enabled.

The executable **E** bit tells if a segment is a code or data segment. A code segment (E=1, S=1) may be execute-only or execute/read as determined by the Read **R** bit. Code segments are execute only if R=0, and execute/read if R=1. Code segments may never be written into.

NOTE:
Code segments may be modified via aliases. Aliases are writeable data segments which occupy the same range of linear address space as the code segment.

The **D** bit indicates the default length for operands and effective addresses. If D=1 then 32-bit operands and 32-bit addressing modes are assumed. If D=0 then 16-bit operands and 16-bit addressing modes are assumed. Therefore all existing 80286 code segments will execute on the Intel386 DX assuming the D bit is set 0.

Another attribute of code segments is determined by the conforming **C** bit. Conforming segments, C=1, can be executed and shared by programs at different privilege levels. (See section 4.4 **Protection**.)

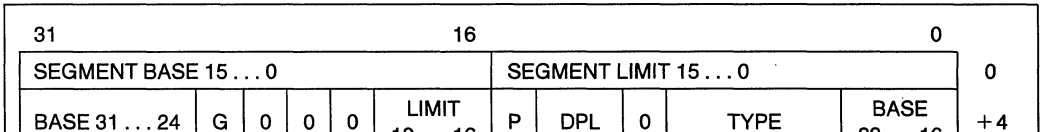
Segments identified as data segments (E=0, S=1) are used for two types of Intel386 DX segments: stack and data segments. The expansion direction **(ED)** bit specifies if a segment expands downward (stack) or upward (data). If a segment is a stack segment all offsets must be greater than the segment limit. On a data segment all offsets must be less than or equal to the limit. In other words, stack segments start at the base linear address plus the maximum segment limit and grow down to the base linear address plus the limit. On the other hand, data segments start at the base linear address and expand to the base linear address plus limit.

The write **W** bit controls the ability to write into a segment. Data segments are read-only if W=0. The stack segment must have W=1.

The **B** bit controls the size of the stack pointer register. If B=1, then PUSHes, POPs, and CALLs all use the 32-bit ESP register for stack references and assume an upper limit of FFFFFFFFH. If B=0, stack instructions all use the 16-bit SP register and assume an upper limit of FFFFH.

4.3.4.3 SYSTEM DESCRIPTOR FORMATS

System segments describe information about operating system tables, tasks, and gates. Figure 4-7 shows the general format of system segment descriptors, and the various types of system segments. Intel386 DX system descriptors contain a 32-bit base linear address and a 20-bit segment limit. 80286 system descriptors have a 24-bit base address and a 16-bit segment limit. 80286 system descriptors are identified by the upper 16 bits being all zero.



Type	Defines	Type	Defines
0	Invalid	8	Invalid
1	Available 80286 TSS	9	Available Intel386™ DX TSS
2	LDT	A	Undefined (Intel Reserved)
3	Busy 80286 TSS	B	Busy Intel386™ DX TSS
4	80286 Call Gate	C	Intel386™ DX Call Gate
5	Task Gate (for 80286 or Intel386™ DX Task)	D	Undefined (Intel Reserved)
6	80286 Interrupt Gate	E	Intel386™ DX Interrupt Gate
7	80286 Trap Gate	F	Intel386™ DX Trap Gate

NOTE:
In a maximum-size segment (ie. a segment with G=1 and segment limit 19...0=FFFFFH), the lowest 12 bits of the segment base should be zero (ie. segment base 11...000=000H).

Figure 4-7. System Segments Descriptors

4.3.4.4 LDT DESCRIPTORS (S=0, TYPE=2)

LDT descriptors (S=0 TYPE=2) contain information about Local Descriptor Tables. LDTs contain a table of segment descriptors, unique to a particular task. Since the instruction to load the LDTR is only available at privilege level 0, the DPL field is ignored. LDT descriptors are only allowed in the Global Descriptor Table (GDT).

4.3.4.5 TSS DESCRIPTORS (S=0, TYPE=1, 3, 9, B)

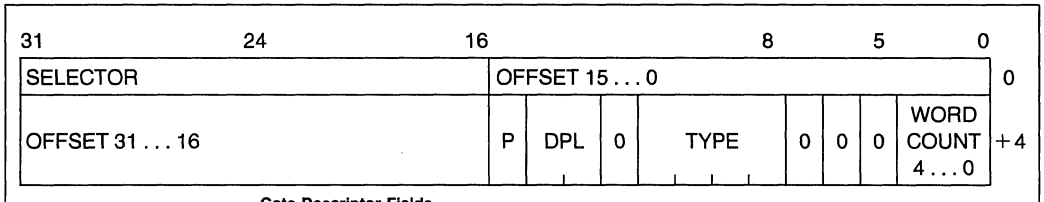
A Task State Segment (TSS) descriptor contains information about the location, size, and privilege level of a Task State Segment (TSS). A TSS in turn is a special fixed format segment which contains all the state information for a task and a linkage field to permit nesting tasks. The TYPE field is used to indicate whether the task is currently BUSY (i.e. on a chain of active tasks) or the TSS is available. The TYPE field also indicates if the segment contains a 80286 or an Intel386 DX TSS. The Task Register (TR) contains the selector which points to the current Task State Segment.

4.3.4.6 GATE DESCRIPTORS (S=0, TYPE=4-7, C, F)

Gates are used to control access to entry points within the target code segment. The various types of gate descriptors are **call** gates, **task** gates, **interrupt** gates, and **trap** gates. Gates provide a level of indirection between the source and destination of the control transfer. This indirection allows the processor to automatically perform protection checks. It also allows system designers to control entry points to the operating system. Call gates are used to change privilege levels (see section 4.4 **Protection**), task gates are used to perform a task switch, and interrupt and trap gates are used to specify interrupt service routines.

Figure 4-8 shows the format of the four types of gate descriptors. Call gates are primarily used to transfer program control to a more privileged level. The call gate descriptor consists of three fields: the access byte, a long pointer (selector and offset) which points to the start of a routine and a word count which specifies how many parameters are to be copied from the caller's stack to the stack of the called routine. The word count field is only used by call gates when there is a change in the privilege level, other types of gates ignore the word count field.

Interrupt and trap gates use the destination selector and destination offset fields of the gate descriptor as a pointer to the start of the interrupt or trap handler routines. The difference between interrupt gates and trap gates is that the interrupt gate disables interrupts (resets the IF bit) while the trap gate does not.



Gate Descriptor Fields

Name	Value	Description
Type	4	80286 call gate
	5	Task gate (for 80286 or Intel386™ DX task)
	6	80286 interrupt gate
	7	80286 trap gate
	C	Intel386™ DX call gate
	E	Intel386™ DX interrupt gate
	F	Intel386™ DX trap gate
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid

DPL—least privileged level at which a task may access the gate. WORD COUNT 0-31—the number of parameters to copy from caller's stack to the called procedure's stack. The parameters are 32-bit quantities for Intel386™ DX gates, and 16-bit quantities for 80286 gates.

DESTINATION SELECTOR	16-bit selector	Selector to the target code segment or Selector to the target task state segment for task gate
DESTINATION OFFSET	offset 16-bit 80286 32-bit Intel386™ DX	Entry point within the target code segment

Figure 4-8. Gate Descriptor Formats

Task gates are used to switch tasks. Task gates may only refer to a task state segment (see section 4.4.6 **Task Switching**) therefore only the destination selector portion of a task gate descriptor is used, and the destination offset is ignored.

Exception 13 is generated when a destination selector does not refer to a correct descriptor type, i.e., a code segment for an interrupt, trap or call gate, a TSS for a task gate.

The access byte format is the same for all gate descriptors. P=1 indicates that the gate contents are valid. P=0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the descriptor privilege level and specifies when this descriptor may be used by a task (see section 4.4 **Protection**). The S field, bit 4 of the access rights byte, must be 0 to indicate a system control descriptor. The type field specifies the descriptor type as indicated in Figure 4-8.

4.3.4.7 DIFFERENCES BETWEEN Intel386™ DX AND 80286 DESCRIPTORS

In order to provide operating system compatibility between the 80286 and Intel386 DX, the Intel386 DX supports all of the 80286 segment descriptors. Figure 4-9 shows the general format of an 80286 system segment descriptor. The only differences between 80286 and Intel386 DX descriptor formats are that the values of the type fields, and the limit and base address fields have been expanded for the Intel386 DX. The 80286 system segment descriptors contained a 24-bit base address and 16-bit limit, while the Intel386 DX system segment descriptors have a 32-bit base address, a 20-bit limit field, and a granularity bit.

By supporting 80286 system segments the Intel386 DX is able to execute 80286 application programs on an Intel386 DX operating system. This is possible because the processor automatically understands which descriptors are 80286-style descriptors and which descriptors are Intel386 DX-style descriptors. In particular, if the upper word of a descriptor is zero, then that descriptor is a 80286-style descriptor.

The only other differences between 80286-style descriptors and Intel386 DX descriptors is the interpretation of the word count field of call gates and the B bit. The word count field specifies the number of 16-bit quantities to copy for 80286 call gates and 32-bit quantities for Intel386 DX call gates. The B bit controls the size of PUSHes when using a call gate; if B=0 PUSHes are 16 bits, if B=1 PUSHes are 32 bits.

4.3.4.8 SELECTOR FIELDS

A selector in Protected Mode has three fields: Local or Global Descriptor Table Indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL) as shown in Figure 4-10. The TI bits select one of two memory-based tables of descriptors (the Global Descriptor Table or the Local Descriptor Table). The Index selects one of 8K descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

4.3.4.9 SEGMENT DESCRIPTOR CACHE

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs which modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.

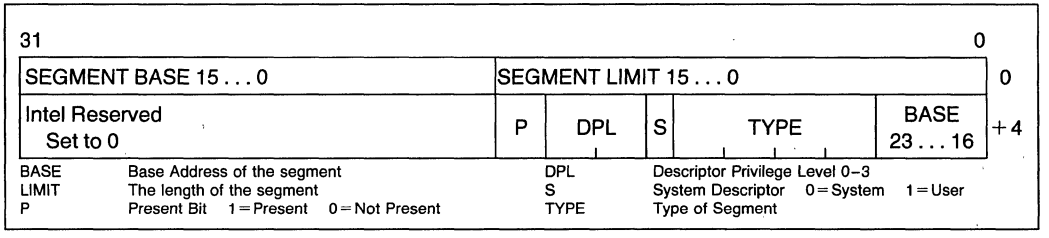


Figure 4-9. 80286 Code and Data Segment Descriptors

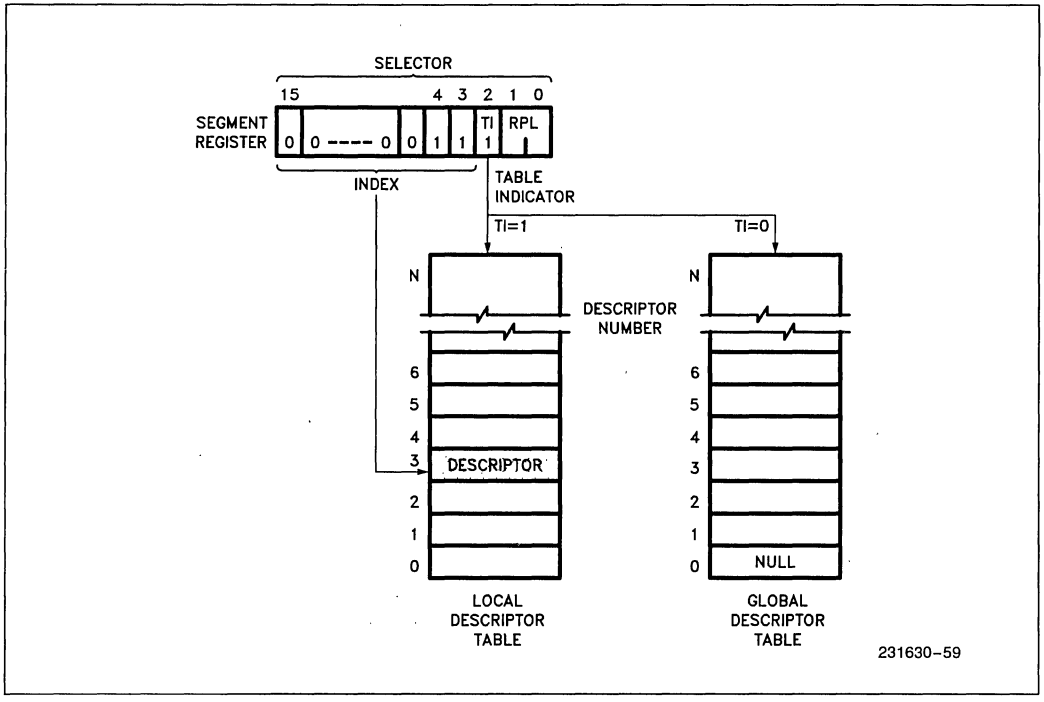
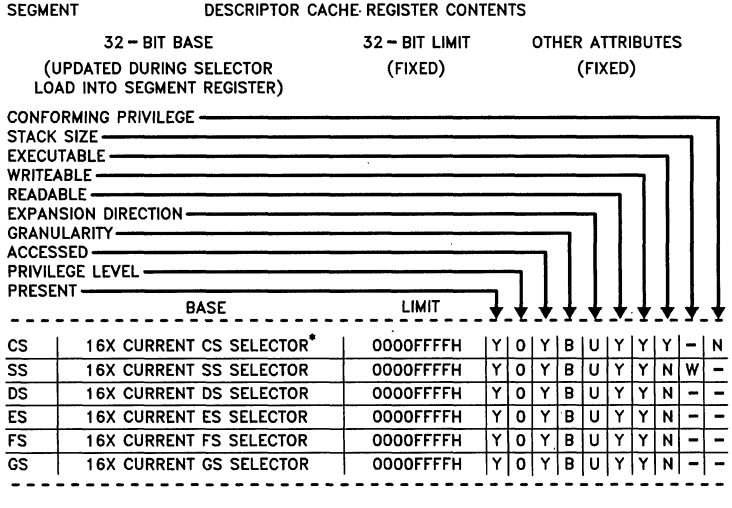


Figure 4-10. Example Descriptor Selection

4.3.4.10 SEGMENT DESCRIPTOR REGISTER SETTINGS

The contents of the segment descriptor cache vary depending on the mode the Intel386 DX is operating in. When operating in Real Address Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 4-11.

For compatiblity with the 8086 architecture, the base is set to sixteen times the current selector value, the limit is fixed at 0000FFFFH, and the attributes are fixed so as to indicate the segment is present and fully usable. In Real Address Mode, the internal "privilege level" is always fixed to the highest level, level 0, so I/O and other privileged opcodes may be executed.

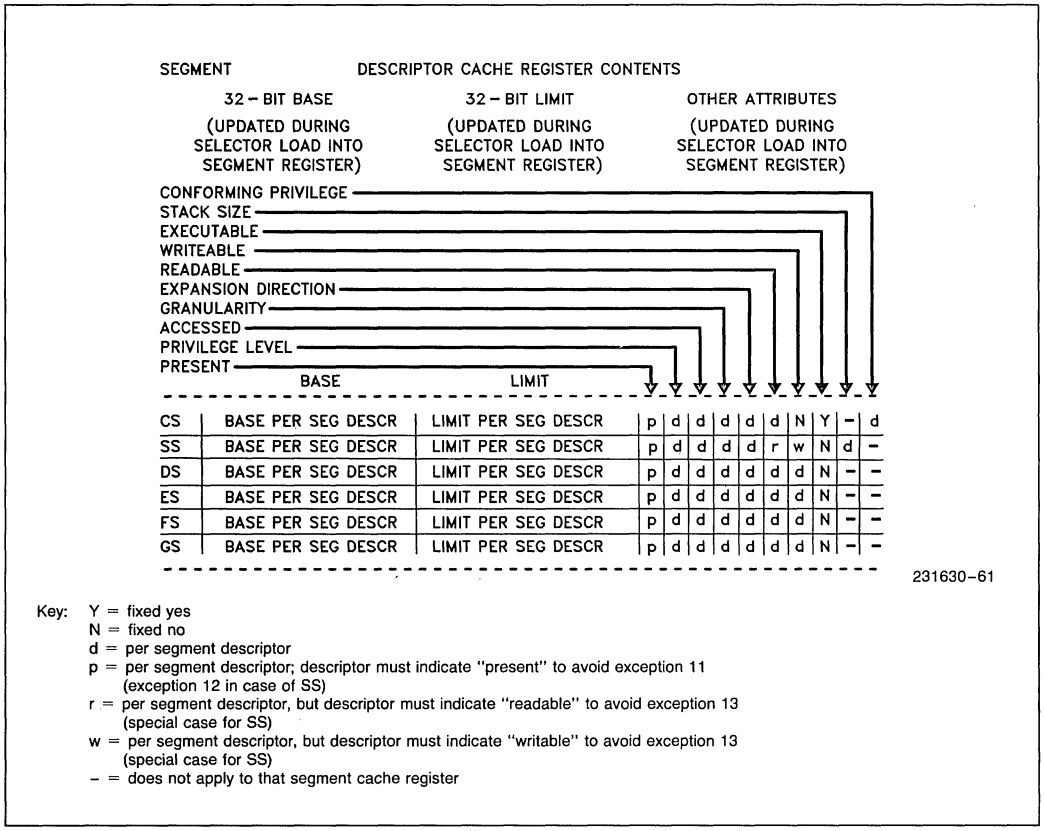


*Except the 32-bit CS base is initialized to FFFFF000H after reset until first intersegment control transfer (e.g. intersegment CALL, or intersegment JMP, or INT). (See Figure 4-13 Example.)

Key:
- Y = yes
- N = no
- 0 = privilege level 0
- 1 = privilege level 1
- 2 = privilege level 2
- 3 = privilege level 3
- U = expand up
- D = expand down
- B = byte granularity
- P = page granularity
- W = push/pop 16-bit words
- F = push/pop 32-bit dwords
- – = does not apply to that segment cache register

Figure 4-11. Segment Descriptor Caches for Real Address Mode (Segment Limit and Attributes are Fixed)

When operating in Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 4-12. In Protected Mode, each of these fields are defined according to the contents of the segment descriptor indexed by the selector value loaded into the segment register.



Key: Y = fixed yes
N = fixed no
d = per segment descriptor
p = per segment descriptor; descriptor must indicate "present" to avoid exception 11 (exception 12 in case of SS)
r = per segment descriptor, but descriptor must indicate "readable" to avoid exception 13 (special case for SS)
w = per segment descriptor, but descriptor must indicate "writable" to avoid exception 13 (special case for SS)
– = does not apply to that segment cache register

Figure 4-12. Segment Descriptor Caches for Protected Mode (Loaded per Descriptor)

When operating in a Virtual 8086 Mode within the Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 4-13. For compatibility with the 8086 architecture, the base is set to sixteen times the current selector value, the limit is fixed at 0000FFFFH, and the attributes are fixed so as to indicate the segment is present and fully usable. The virtual program executes at lowest privilege level, level 3, to allow trapping of all IOPL-sensitive instructions and level-0-only instructions.

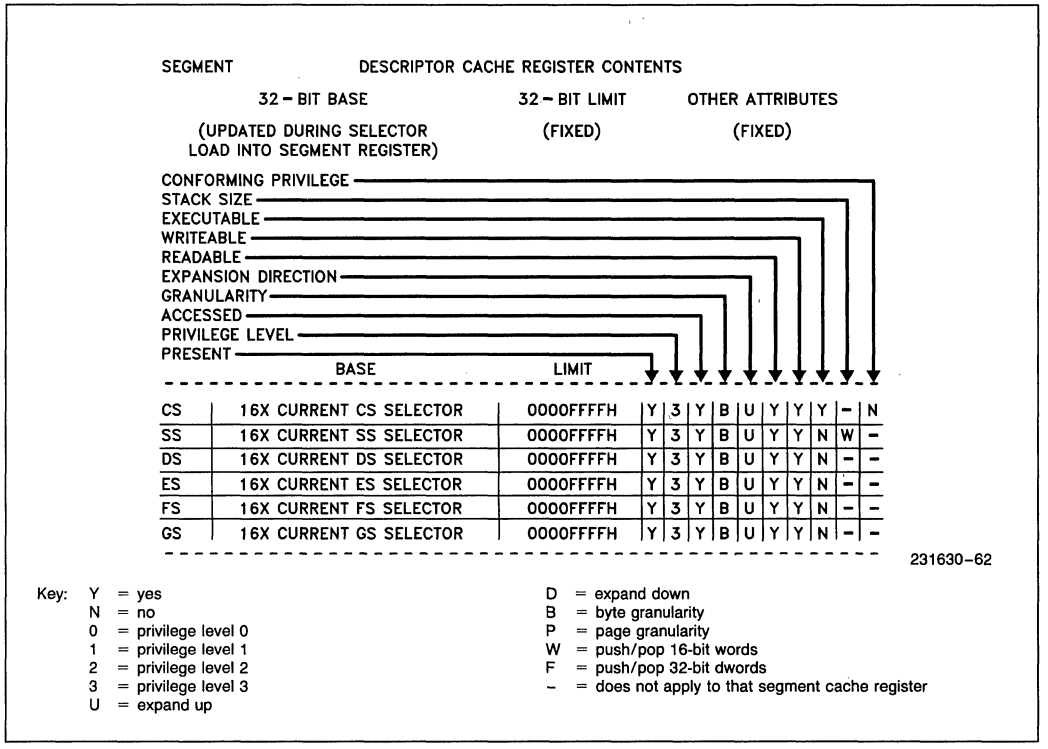


Figure 4-13. Segment Descriptor Caches for Virtual 8086 Mode within Protected Mode (Segment Limit and Attributes are Fixed)

4.4 PROTECTION

4.4.1 Protection Concepts

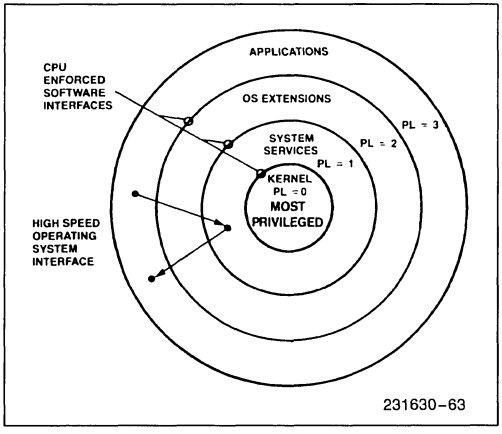


Figure 4-14. Four-Level Hierachical Protection

The Intel386 DX has four levels of protection which are optimized to support the needs of a multi-tasking operating system to isolate and protect user programs from each other and the operating system. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. Unlike traditional microprocessor-based systems where this protection is achieved only through the use of complex external hardware and software the Intel386 DX provides the protection as part of its integrated Memory Management Unit. The Intel386 DX offers an additional type of protection on a page basis, when paging is enabled (See section 4.5.3 **Page Level Protection**).

The four-level hierarchical privilege system is illustrated in Figure 4-14. It is an extension of the user/supervisor privilege mode commonly used by minicomputers and, in fact, the user/supervisor mode is fully supported by the Intel386 DX paging mechanism. The privilege levels (PL) are numbered 0 through 3. Level 0 is the most privileged or trusted level.

4.4.2 Rules of Privilege

The Intel386 DX controls access to both data and procedures between levels of a task, according to the following rules.

- Data stored in a segment with privilege level **p** can be accessed only by code executing at a privilege level at least as privileged as **p**.
- A code segment/procedure with privilege level **p** can only be called by a task executing at the same or a lesser privilege level than **p**.

4.4.3 Privilege Levels

4.4.3.1 TASK PRIVILEGE

At any point in time, a task on the Intel386 DX always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies the task's privilege level. A task's CPL may only be changed by control transfers through gate descriptors to a code segment with a different privilege level. (See section 4.4.4 **Privilege Level Transfers**) Thus, an application program running at PL = 3 may call an operating system routine at PL = 1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

4.4.3.2 SELECTOR PRIVILEGE (RPL)

The privilege level of a selector is specified by the RPL field. The RPL is the two least significant bits of the selector. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (i.e. numerically larger) level of a task's CPL and a selector's RPL. Thus, if selector's RPL = 0 then the CPL always specifies the privilege level for making an access using the selector. On the other hand if RPL = 3 then a selector can only access segments at level 3 regardless of the task's CPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

4.4.3.3 I/O PRIVILEGE AND I/O PERMISSION BITMAP

The I/O privilege level (IOPL, a 2-bit field in the EFLAG register) defines the least privileged level at which I/O instructions can be unconditionally performed. I/O instructions can be unconditionally performed when CPL ≤ IOPL. (The I/O instructions are IN, OUT, INS, OUTS, REP INS, and REP OUTS.) When CPL > IOPL, and the current task is associated with a 286 TSS, attempted I/O instructions cause an exception 13 fault. When CPL > IOPL, and the current task is associated with an Intel386 DX TSS, the I/O Permission Bitmap (part of an Intel386 DX TSS) is consulted on whether I/O to the port is allowed, or an exception 13 fault is to be generated

instead. For diagrams of the I/O Permission Bitmap, refer to Figures 4-15a and 4-15b. For further information on how the I/O Permission Bitmap is used in Protected Mode or in Virtual 8086 Mode, refer to section 4.6.4 Protection and I/O Permission Bitmap.

The I/O privilege level (IOPL) also affects whether several other instructions can be executed or cause an exception 13 fault instead. These instructions are called "IOPL-sensitive" instructions and they are CLI and STI. (Note that the LOCK prefix is *not* IOPL-sensitive on the Intel386 DX.)

The IOPL also affects whether the IF (interrupts enable flag) bit can be changed by loading a value into the EFLAGS register. When CPL ≤ IOPL, then the IF bit can be changed by loading a new value into the EFLAGS register. When CPL > IOPL, the IF bit cannot be changed by a new value POP'ed into (or otherwise loaded into) the EFLAGS register; the IF bit merely remains unchanged and no exception is generated.

Table 4-2. Pointer Test Instructions

Instruction	Operands	Function
ARPL	Selector, Register	Adjust Requested Privilege Level: adjusts the RPL of the selector to the numeric maximum of current selector RPL value and the RPL value in the register. Set zero flag if selector RPL was changed.
VERR	Selector	VERify for Read: sets the zero flag if the segment referred to by the selector can be read.
VERW	Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL	Register, Selector	Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

4.4.3.4 PRIVILEGE VALIDATION

The Intel386 DX provides several instructions to speed pointer testing and help maintain system integrity by verifying that the selector value refers to an appropriate segment. Table 4-2 summarizes the selector validation procedures available for the Intel386 DX.

This pointer verification prevents the common problem of an application at PL = 3 calling a operating systems routine at PL = 0 and passing the operating system routine a "bad" pointer which corrupts a data structure belonging to the operating system. If the operating system routine uses the ARPL instruction to ensure that the RPL of the selector has no greater privilege than that of the caller, then this problem can be avoided.

4.4.3.5 DESCRIPTOR ACCESS

There are basically two types of segment accesses: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL as described above.

Any time an instruction loads data segment registers (DS, ES, FS, GS) the Intel386 DX makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segments or readable code segments. The data access rules are specified in section 4.2.2 **Rules of Privilege**. The only exception to those rules is readable conforming code segments which can be accessed at any privilege level.

Finally the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL an exception 13 (general protection fault) is generated.

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL. All other descriptor types or a privilege level violation will cause exception 13. A stack not present fault causes exception 12. Note that an exception 11 is used for a not-present code or data segment.

4.4.4 Privilege Level Transfers

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call

Table 4-3. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL	CALL	Call Gate	GDT/LDT
	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0
**NT (Nested Task bit of flag register) = 1

or a jump to another routine. There are five types of control transfers which are summarized in Table 4-3. Many of these transfers result in a privilege level transfer. Changing privilege levels is done only via control transfers, by using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an exception 13 (e.g. JMP through a call gate, or IRET from a normal subroutine call).

In order to provide further system security, all control transfers are also subject to the privilege rules.

The privilege rules require that:

— Privilege level transitions can only occur via gates.

— JMPs can be made to a non-conforming code segment with the same privilege or to a conforming code segment with greater or equal privilege.

— CALLs can be made to a non-conforming code segment with the same privilege or via a gate to a more privileged level.

— Interrupts handled within the task obey the same privilege rules as CALLs.

— Conforming Code segments are accessible by privilege levels which are the same or less privileged than the conforming-code segment's DPL.

— Both the requested privilege level (RPL) in the selector pointing to the gate and the task's CPL must be of equal or greater privilege than the gate's DPL.

— The code segment selected in the gate must be the same or more privileged than the task's CPL.

— Return instructions that do not switch tasks can only return control to a code segment with same or less privilege.

— Task switches can be performed by a CALL, JMP, or INT which references either a task gate or task state segment who's DPL is less privileged or the same privilege as the old task's CPL.

Any control transfer that changes CPL within a task causes a change of stacks as a result of the privilege level change. The initial values of SS:ESP for privilege levels 0, 1, and 2 are retained in the task state segment (see section 4.4.6 **Task Switching**). During a JMP or CALL control transfer, the new stack pointer is loaded into the SS and ESP registers and the previous stack pointer is pushed onto the new stack.

When RETurning to the original privilege level, use of the lower-privileged stack is restored as part of the RET or IRET instruction operation. For subroutine calls that pass parameters on the stack and cross privilege levels, a fixed number of words (as specified in the gate's word count field) are copied from the previous stack to the current stack. The inter-segment RET instruction with a stack adjustment value will correctly restore the previous stack pointer upon return.

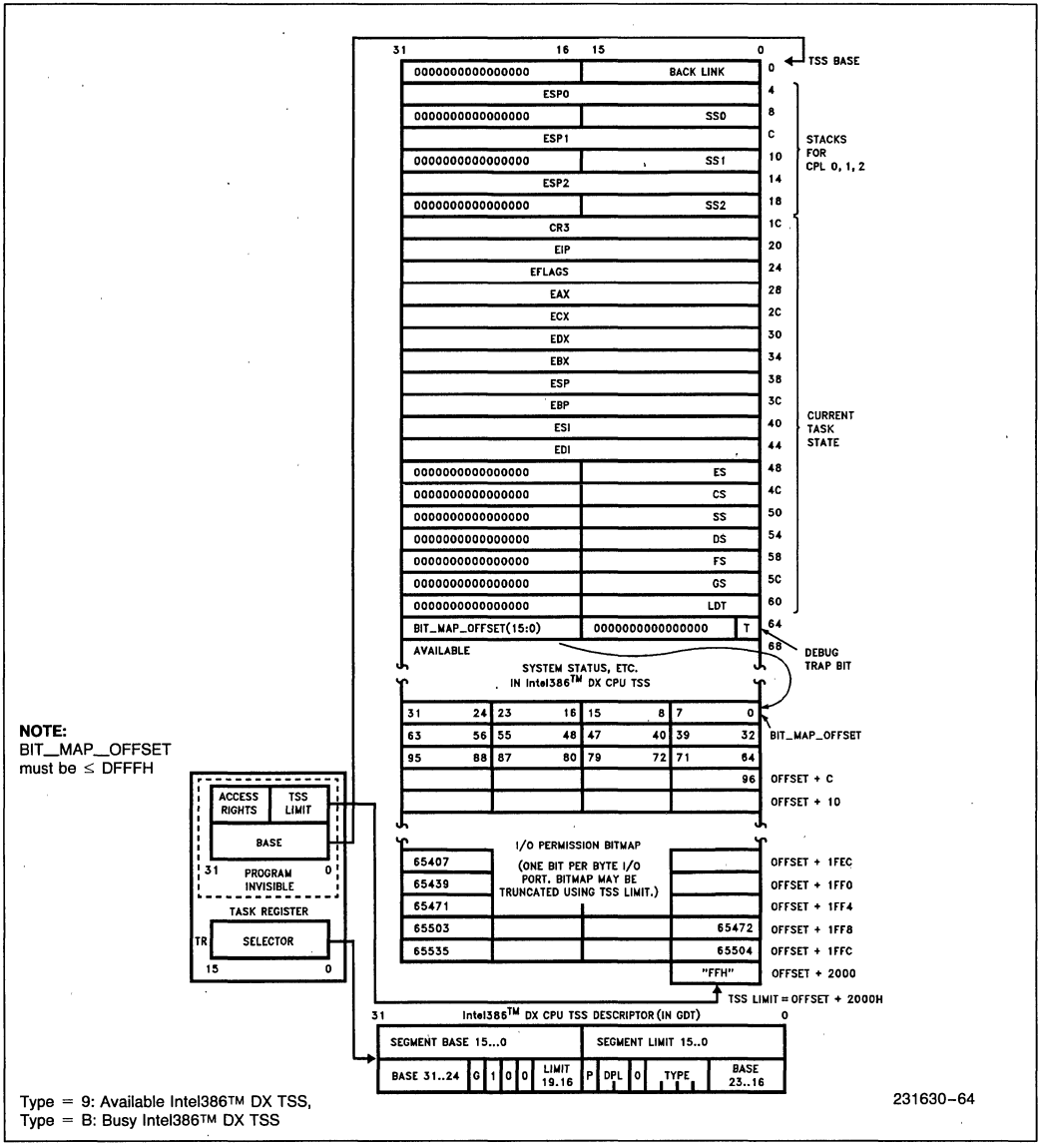


Figure 4-15a. Intel386™ DX TSS and TSS Registers

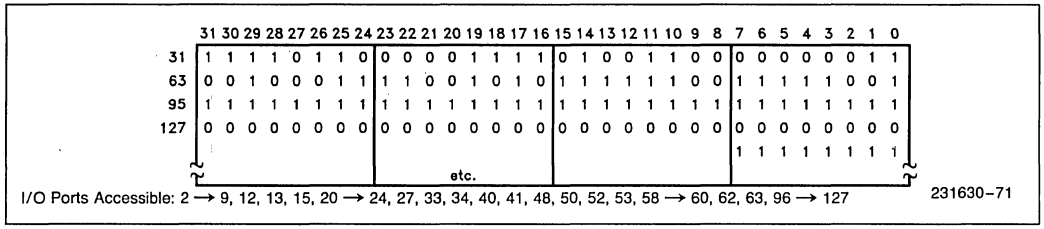


Figure 4-15b. Sample I/O Permission Bit Map

4.4.5 Call Gates

Gates provide protected, indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures (such as those which allocate memory, or perform I/O).

Gate descriptors follow the data access rules of privilege; that is, gates can be accessed by a task if the EPL, is equal to or more privileged than the gate descriptor's DPL. Gates follow the control transfer rules of privilege and therefore may only transfer control to a more privileged level.

Call Gates are accessed via a CALL instruction and are syntactically identical to calling a normal subroutine. When an inter-level Intel386 DX call gate is activated, the following actions occur.

1. Load CS:EIP from gate check for validity
2. SS is pushed zero-extended to 32 bits
3. ESP is pushed
4. Copy Word Count 32-bit parameters from the old stack to the new stack
5. Push Return address on stack

The procedure is identical for 80286 Call gates, except that 16-bit parameters are copied and 16-bit registers are pushed.

Interrupt Gates and Trap gates work in a similar fashion as the call gates, except there is no copying of parameters. The only difference between Trap and Interrupt gates is that control transfers through an Interrupt gate disable further interrupts (i.e. the IF bit is set to 0), and Trap gates leave the interrupt status unchanged.

4.4.6 Task Switching

A very important attribute of any multi-tasking/multi-user operating systems is its ability to rapidly switch between tasks or processes. The Intel386 DX directly supports this operation by providing a task switch instruction in hardware. The Intel386 DX task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task, in about 17 microseconds. Like transfer of control via gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap, or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot.

The TSS descriptor points to a segment (see Figure 4-15) containing the entire Intel386 DX execution state while a task gate descriptor contains a TSS selector. The Intel386 DX supports both 80286 and Intel386 DX style TSSs. Figure 4-16 shows a 80286 TSS. The limit of an Intel386 DX TSS must be greater than 0064H (002BH for a 80286 TSS), and can be as large as 4 Gigabytes. In the additional TSS space, the operating system is free to store additional information such as the reason the task is inactive, time the task has spent running, and open files belong to the task.

Each task must have a TSS associated with it. The current TSS is identified by a special register in the Intel386 DX called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TR are loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The current executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and machine status word (CR0) give information about the state of a task which are useful to the operating system. The Nested Task (NT) (bit 14 in EFLAGS) controls the function of the IRET instruction. If NT = 0, the IRET instruction performs the regular return; when NT = 1, IRET performs a task switch operation back to the previous task. The NT bit is set or reset in the following fashion:

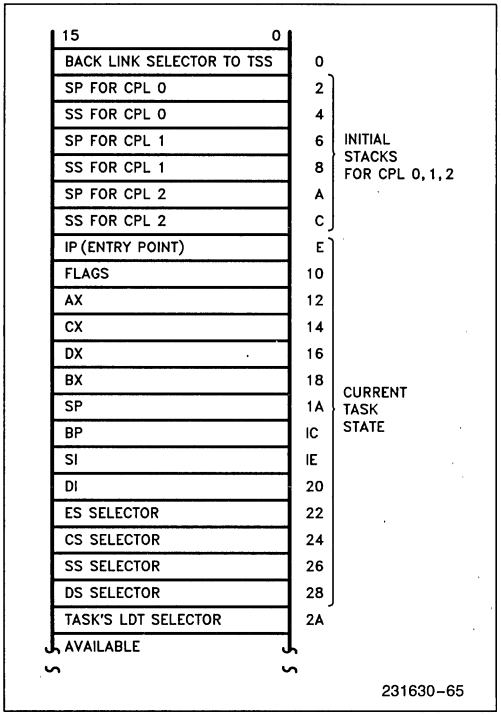


Figure 4-16. 80286 TSS

When a CALL or INT instruction initiates a task switch, the new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT. (The NT bit will be restored after execution of the interrupt handler) NT may also be set or cleared by POPF or IRET instructions.

The Intel386 DX task state segment is marked busy by changing the descriptor type field from TYPE 9H to TYPE BH. An 80286 TSS is marked busy by changing the descriptor type field from TYPE 1 to TYPE 3. Use of a selector that references a busy task state segment causes an exception 13.

The Virtual Mode (VM) bit 17 is used to indicate if a task, is a virtual 8086 task. If VM = 1, then the tasks will use the Real Mode addressing mechanism. The virtual 8086 environment is only entered and exited via a task switch (see section 4.6 **Virtual Mode**).

The coprocessor's state is not automatically saved when a task switch occurs, because the incoming task may not use the coprocessor. The Task Switched (TS) Bit (bit 3 in the CR0) helps deal with the coprocessor's state in a multi-tasking environment. Whenever the Intel386 DX switches tasks, it sets the TS bit. The Intel386 DX detects the first use of a processor extension instruction after a task switch and causes the processor extension not available exception 7. The exception handler for exception 7 may then decide whether to save the state of the coprocessor. A processor extension not present exception (7) will occur when attempting to execute an ESC or WAIT instruction if the Task Switched and Monitor coprocessor extension bits are both set (i.e. TS = 1 and MP = 1).

The **T** bit in the Intel386 DX TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1 then upon entry to a new task a debug exception 1 will be generated.

4.4.7 Initialization and Transition to Protected Mode

Since the Intel386 DX begins executing in Real Mode immediately after RESET it is necessary to initialize the system tables and registers with the appropriate values.

The GDT and IDT registers must refer to a valid GDT and IDT. The IDT should be at least 256 bytes long, and GDT must contain descriptors for the initial code, and data segments. Figure 4-17 shows the tables and Figure 4-18 the descriptors needed for a simple Protected Mode Intel386 DX system. It has a single code and single data/stack segment each four gigabytes long and a single privilege level PL = 0.

The actual method of enabling Protected Mode is to load CR0 with the PE bit set, via the MOV CR0, R/M instruction. This puts the Intel386 DX in Protected Mode.

After enabling Protected Mode, the next instruction should execute an intersegment JMP to load the CS register and flush the instruction decode queue. The final step is to load all of the data segment registers with the initial selector values.

An alternate approach to entering Protected Mode which is especially appropriate for multi-tasking operating systems, is to use the built in task-switch to load all of the registers. In this case the GDT would contain two TSS descriptors in addition to the code and data descriptors needed for the first task. The first JMP instruction in Protected Mode would jump to the TSS causing a task switch and loading all of the registers with the values stored in the TSS. The Task State Segment Register should be initialized to point to a valid TSS descriptor since a task switch saves the state of the current task in a task state segment.

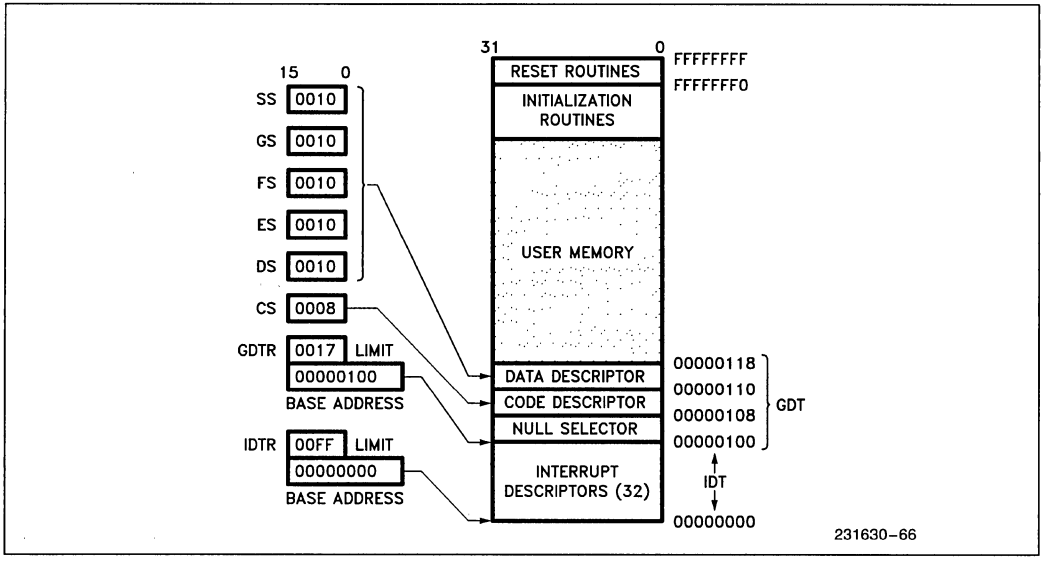


Figure 4-17. Simple Protected System

DATA DESCRIPTOR	SEGMENT BASE 15 . . . 0 0118 (H)								SEGMENT LIMIT 15 . . . 0 FFFF (H)								
	BASE 31 . . . 24 00 (H)	G 1	D 1	0	0	LIMIT 19.16 F (H)			1	0	0	1	0	0	1	0	BASE 23 . . . 16 00 (H)
CODE DESCRIPTOR	SEGMENT BASE 15 . . . 0 0118 (H)								SEGMENT LIMIT 15 . . . 0 FFFF (H)								
	BASE 31 . . . 24 00 (H)	G 1	D 1	0	0	LIMIT 19.16 F (H)			1	0	0	1	1	0	1	0	BASE 23 . . . 16 00 (H)
	NULL DESCRIPTOR																
	31					24		16	15					8			0

Figure 4-18. GDT Descriptors for Simple System

4.4.8 Tools for Building Protected Systems

In order to simplify the design of a protected multitasking system, Intel provides a tool which allows the system designer an easy method of constructing the data structures needed for a Protected Mode Intel386 DX system. This tool is the builder BLD-386. BLD-386 lets the operating system writer specify all of the segment descriptors discussed in the previous sections (LDTs, IDTs, GDTs, Gates, and TSSs) in a high-level language.

4.5 PAGING

4.5.1 Paging Concepts

Paging is another type of memory management useful for virtual memory multitasking operating systems. Unlike segmentation which modularizes programs and data into variable length segments, paging divides programs into multiple uniform size pages. Pages bear no direct relation to the logical

structure of a program. While segment selectors can be considered the logical "name" of a program module or data structure, a page most likely corresponds to only a portion of a module or data structure.

By taking advantage of the locality of reference displayed by most programs, only a small number of pages from each active task need be in memory at any one moment.

4.5.2 Paging Organization

4.5.2.1 PAGE MECHANISM

The Intel386 DX uses two levels of tables to translate the linear address (from the segmentation unit) into a physical address. There are three components to the paging mechanism of the Intel386 DX: the page directory, the page tables, and the page itself (page frame). All memory-resident elements of the Intel386 DX paging mechanism are the same size, namely, 4K bytes. A uniform size for all of the elements simplifies memory allocation and reallocation schemes, since there is no problem with memory fragmentation. Figure 4-19 shows how the paging mechanism works.

4.5.2.2 PAGE DESCRIPTOR BASE REGISTER

CR2 is the Page Fault Linear Address register. It holds the 32-bit linear address which caused the last page fault detected.

CR3 is the Page Directory Physical Base Address Register. It contains the physical starting address of the Page Directory. The lower 12 bits of CR3 are always zero to ensure that the Page Directory is always page aligned. Loading it via a MOV CR3, reg instruction causes the Page Table Entry cache to be flushed, as will a task switch through a TSS which **changes** the value of CR0. (See 4.5.4 **Translation Lookaside Buffer**).

4.5.2.3 PAGE DIRECTORY

The Page Directory is 4K bytes long and allows up to 1024 Page Directory Entries. Each Page Directory Entry contains the address of the next level of tables, the Page Tables and information about the page table. The contents of a Page Directory Entry are shown in Figure 4-20. The upper 10 bits of the linear address (A22–A31) are used as an index to select the correct Page Directory Entry.

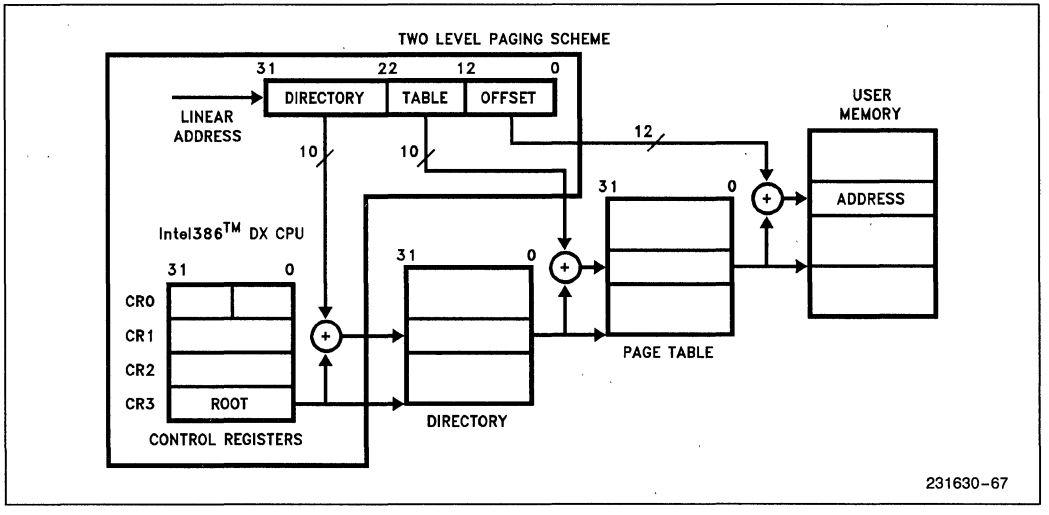


Figure 4-19. Paging Mechanism

31 — 12	11 — 9	8	7	6	5	4	3	2	1	0
PAGE TABLE ADDRESS 31..12	OS RESERVED	0	0	D	A	0	0	U/S	R/W	P

Figure 4-20. Page Directory Entry (Points to Page Table)

31 12	11 10	9	8	7	6	5	4	3	2	1	0
PAGE FRAME ADDRESS 31..12	OS RESERVED		0	0	D	A	0	0	U/S	R/W	P

Figure 4-21. Page Table Entry (Points to Page)

4.5.2.4 PAGE TABLES

Each Page Table is 4K bytes and holds up to 1024 Page Table Entries. Page Table Entries contain the starting address of the page frame and statistical information about the page (see Figure 4-21). Address bits A12–A21 are used as an index to select one of the 1024 Page Table Entries. The 20 upper-bit page frame address is concatenated with the lower 12 bits of the linear address to form the physical address. Page tables can be shared between tasks and swapped to disks.

4.5.2.5 PAGE DIRECTORY/TABLE ENTRIES

The lower 12 bits of the Page Table Entries and Page Directory Entries contain statistical information about pages and page tables respectively. The **P** (Present) bit 0 indicates if a Page Directory or Page Table entry can be used in address translation. If P = 1 the entry can be used for address translation; if P = 0 the entry can not be used for translation. Note that the present bit of the page table entry that points to the page where code is currently being executed should always be set. Code that marks its own page not present should not be written. All of the other bits are available for use by the software. For example the remaining 31 bits could be used to indicate where on the disk the page is stored.

The **A** (Accessed) bit 5, is set by the Intel386 DX for both types of entries before a read or write access occurs to an address covered by the entry. The **D** (Dirty) bit 6 is set to 1 before a write to an address covered by that page table entry occurs. The D bit is undefined for Page Directory Entries. When the P, A and D bits are updated by the Intel386 DX, the processor generates a Read-Modify-Write cycle which locks the bus and prevents conflicts with other processors or perpherials. Software which modifies these bits should use the LOCK prefix to ensure the integrity of the page tables in multi-master systems.

The 3 bits marked **OS Reserved** in Figure 4-20 and Figure 4-21 (bits 9–11) are software definable. OSs are free to use these bits for whatever purpose they wish. An example use of the **OS Reserved** bits would be to store information about page aging. By keeping track of how long a page has been in memory since being accessed, an operating system can implement a page replacement algorithm like Least Recently Used.

The (User/Supervisor) U/S bit 2 and the (Read/Write) R/W bit 1 are used to provide protection attributes for individual pages.

4.5.3 Page Level Protection (R/W, U/S Bits)

The Intel386 DX provides a set of protection attributes for paging systems. The paging mechanism distinguishes between two levels of protection: User which corresponds to level 3 of the segmentation based protection, and supervisor which encompasses all of the other protection levels (0, 1, 2). Programs executing at Level 0, 1 or 2 bypass the page protection, although segmentation based protection is still enforced by the hardware.

The U/S and R/W bits are used to provide User/Supervisor and Read/Write protection for individual pages or for all pages covered by a Page Table Directory Entry. The U/S and R/W bits in the first level Page Directory Table apply to all pages described by the page table pointed to by that directory entry. The U/S and R/W bits in the second level Page Table Entry apply only to the page described by that entry. The U/S and R/W bits for a given page are obtained by taking the most restrictive of the U/S and R/W from the Page Directory Table Entries and the Page Table Entries and using these bits to address the page.

Example: If the U/S and R/W bits for the Page Directory entry were 10 and the U/S and R/W bits for the Page Table Entry were 01, the access rights for the page would be 01, the numerically smaller of the two. Table 4-4 shows the affect of the U/S and R/W bits on accessing memory.

Table 4-4. Protection Provided by R/W and U/S

U/S	R/W	Permitted Level 3	Permitted Access Levels 0, 1, or 2
0	0	None	Read/Write
0	1	None	Read/Write
1	0	Read-Only	Read/Write
1	1	Read/Write	Read/Write

However a given segment can be easily made read-only for level 0, 1, or 2 via the use of segmented protection mechanisms. (Section 4.4 **Protection**).

4.5.4 Translation Lookaside Buffer

The Intel386 DX paging hardware is designed to support demand paged virtual memory systems. However, performance would degrade substantially if the processor was required to access two levels of tables for every memory reference. To solve this problem, the Intel386 DX keeps a cache of the most recently accessed pages, this cache is called the Translation Lookaside Buffer (TLB). The TLB is a four-way set associative 32-entry page table cache. It automatically keeps the most commonly used Page Table Entries in the processor. The 32-entry TLB coupled with a 4K page size, results in coverage of 128K bytes of memory addresses. For many common multi-tasking systems, the TLB will have a hit rate of about 98%. This means that the processor will only have to access the two-level page structure on 2% of all memory references. Figure 4-22 illustrates how the TLB complements the Intel386 DX's paging mechanism.

4.5.5 Paging Operation

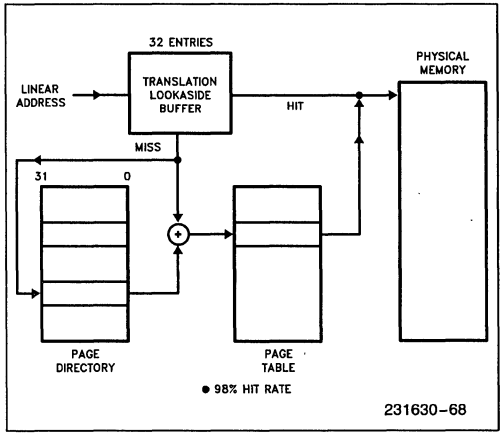


Figure 4-22. Translation Lookaside Buffer

The paging hardware operates in the following fashion. The paging unit hardware receives a 32-bit linear address from the segmentation unit. The upper 20 linear address bits are compared with all 32 entries in the TLB to determine if there is a match. If there is a match (i.e. a TLB hit), then the 32-bit physical address is calculated and will be placed on the address bus.

However, if the page table entry is not in the TLB, the Intel386 DX will read the appropriate Page Directory Entry. If P = 1 on the Page Directory Entry indicating that the page table is in memory, then the Intel386 DX will read the appropriate Page Table Entry and set the Access bit. If P = 1 on the Page Table Entry indicating that the page is in memory, the Intel386 DX will update the Access and Dirty bits as needed and fetch the operand. The upper 20 bits of the linear address, read from the page table, will be stored in the TLB for future accesses. However, if P = 0 for either the Page Directory Entry or the Page Table Entry, then the processor will generate a page fault, an Exception 14.

The processor will also generate an exception 14, page fault, if the memory reference violated the page protection attributes (i.e. U/S or R/W) (e.g. trying to write to a read-only page). CR2 will hold the linear address which caused the page fault. If a second page fault occurs, while the processor is attempting to enter the service routine for the first, then the processor will invoke the page fault (exception 14) handler a second time, rather than the double fault (exception 8) handler. Since Exception 14 is classified as a fault, CS: EIP will point to the instruction causing the page fault. The 16-bit error code pushed as part of the page fault handler will contain status bits which indicate the cause of the page fault.

The 16-bit error code is used by the operating system to determine how to handle the page fault Figure 4-23A shows the format of the page-fault error code and the interpretation of the bits.

NOTE:
Even though the bits in the error code (U/S, W/R, and P) have similar names as the bits in the Page Directory/Table Entries, the interpretation of the error code bits is different. Figure 4-23B indicates what type of access caused the page fault.

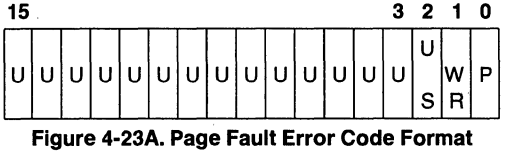


Figure 4-23A. Page Fault Error Code Format

U/S: The U/S bit indicates whether the access causing the fault occurred when the processor was executing in User Mode (U/S = 1) or in Supervisor mode (U/S = 0)

W/R: The W/R bit indicates whether the access causing the fault was a Read (W/R = 0) or a Write (W/R = 1)

P: The P bit indicates whether a page fault was caused by a not-present page (P = 0), or by a page level protection violation (P = 1)

U: UNDEFINED

U/S	W/R	Access Type
0	0	Supervisor* Read
0	1	Supervisor Write
1	0	User Read
1	1	User Write

*Descriptor table access will fault with U/S = 0, even if the program is executing at level 3.

Figure 4-23B. Type of Access Causing Page Fault

4.5.6 Operating System Responsibilities

The Intel386 DX takes care of the page address translation process, relieving the burden from an operating system in a demand-paged system. The operating system is responsible for setting up the initial page tables, and handling any page faults. The operating system also is required to invalidate (i.e. flush) the TLB when any changes are made to any of the page table entries. The operating system must reload CR3 to cause the TLB to be flushed.

Setting up the tables is simply a matter of loading CR3 with the address of the Page Directory, and allocating space for the Page Directory and the Page Tables. The primary responsibility of the operating system is to implement a swapping policy and handle all of the page faults.

A final concern of the operating system is to ensure that the TLB cache matches the information in the paging tables. In particular, any time the operating system sets the P present bit of page table entry to zero, the TLB must be flushed. Operating systems may want to take advantage of the fact that CR3 is stored as part of a TSS, to give every task or group of tasks its own set of page tables.

4.6 VIRTUAL 8086 ENVIRONMENT

4.6.1 Executing 8086 Programs

The Intel386 DX allows the execution of 8086 application programs in both Real Mode and in the Virtual 8086 Mode (Virtual Mode). Of the two methods, Virtual 8086 Mode offers the system designer the most flexibility. The Virtual 8086 Mode allows the execution of 8086 applications, while still allowing the system designer to take full advantage of the Intel386 DX protection mechanism. In particular, the Intel386 DX allows the simultaneous execution of 8086 operating systems and its applications, and an Intel386 DX operating system and both 80286 and Intel386 DX applications. Thus, in a multi-user Intel386 DX computer, one person could be running an MS-DOS spreadsheet, another person using MS-DOS, and a third person could be running multiple Unix utilities and applications. Each person in this scenario would believe that he had the computer completely to himself. Figure 4-24 illustrates this concept.

4.6.2 Virtual 8086 Mode Addressing Mechanism

One of the major differences between Intel386 DX Real and Protected modes is how the segment selectors are interpreted. When the processor is executing in Virtual 8086 Mode the segment registers are used in an identical fashion to Real Mode. The contents of the segment register is shifted left 4 bits and added to the offset to form the segment base linear address.

The Intel386 DX allows the operating system to specify which programs use the 8086 style address mechanism, and which programs use Protected Mode addressing, on a per task basis. Through the use of paging, the one megabyte address space of the Virtual Mode task can be mapped to anywhere in the 4 gigabyte linear address space of the Intel386 DX. Like Real Mode, Virtual Mode effective addresses (i.e., segment offsets) that exceed 64K byte will cause an exception 13. However, these restrictions should not prove to be important, because most tasks running in Virtual 8086 Mode will simply be existing 8086 application programs.

4.6.3 Paging In Virtual Mode

The paging hardware allows the concurrent running of multiple Virtual Mode tasks, and provides protection and operating system isolation. Although it is not strictly necessary to have the paging hardware enabled to run Virtual Mode tasks, it is needed in order to run multiple Virtual Mode tasks or to relocate the address space of a Virtual Mode task to physical address space greater than one megabyte.

The paging hardware allows the 20-bit linear address produced by a Virtual Mode program to be divided into up to 256 pages. Each one of the pages can be located anywhere within the maximum 4 gigabyte physical address space of the Intel386 DX. In addition, since CR3 (the Page Directory Base Register) is loaded by a task switch, each Virtual Mode task can use a different mapping scheme to map pages to different physical locations. Finally, the paging hardware allows the sharing of the 8086 op-

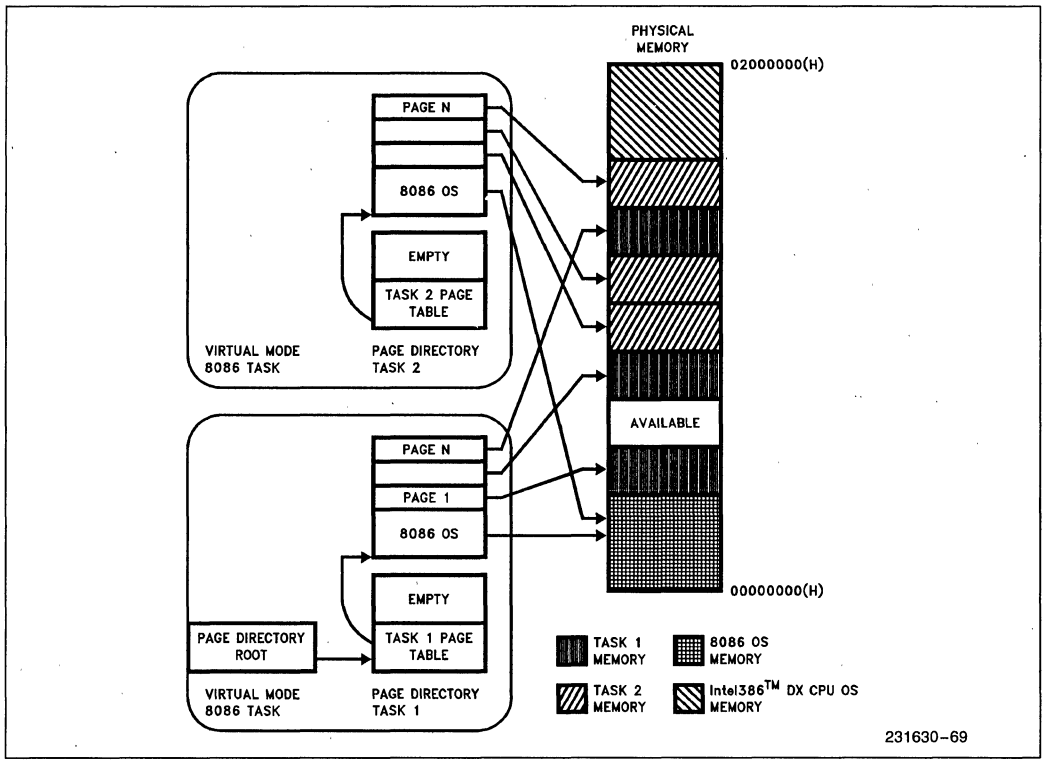


Figure 4-24. Virtual 8086 Environment Memory Management

erating system code between multiple 8086 applications. Figure 4-24 shows how the Intel386 DX paging hardware enables multiple 8086 programs to run under a virtual memory demand paged system.

4.6.4 Protection and I/O Permission Bitmap

All Virtual 8086 Mode programs execute at privilege level 3, the level of least privilege. As such, Virtual 8086 Mode programs are subject to all of the protection checks defined in Protected Mode. (This is different from Real Mode which implicitly is executing at privilege level 0, the level of greatest privilege.) Thus, an attempt to execute a privileged instruction when in Virtual 8086 Mode will cause an exception 13 fault.

The following are privileged instructions, which may be executed only at Privilege Level 0. Therefore, attempting to execute these instructions in Virtual 8086 Mode (or anytime CPL > 0) causes an exception 13 fault:

```
LIDT;    MOV DRn,reg;    MOV reg,DRn;
LGDT;    MOV TRn,reg;    MOV reg,TRn;
LMSW;    MOV CRn,reg;    MOV reg,CRn.
CLTS;
HLT;
```

Several instructions, particularly those applying to the multitasking model and protection model, are available only in Protected Mode. Therefore, attempting to execute the following instructions in Real Mode or in Virtual 8086 Mode generates an exception 6 fault:

```
LTR;     STR;
LLDT;    SLDT;
LAR;     VERR;
LSL;     VERW;
ARPL.
```

The instructions which are IOPL-sensitive in Protected Mode are:

```
IN;          STI;
OUT;         CLI
INS;
OUTS;
REP INS;
REP OUTS;
```

In Virtual 8086 Mode, a slightly different set of instructions are made IOPL-sensitive. The following instructions are IOPL-sensitive in Virtual 8086 Mode:

```
INT n;    STI;
PUSHF;    CLI;
POPF;     IRET
```

The PUSHF, POPF, and IRET instructions are IOPL-sensitive in Virtual 8086 Mode only. This provision allows the IF flag (interrupt enable flag) to be virtualized to the Virtual 8086 Mode program. The INT n software interrupt instruction is also IOPL-sensitive in Virtual 8086 Mode. Note, however, that the INT 3 (opcode 0CCH), INTO, and BOUND instructions are not IOPL-sensitive in Virtual 8086 mode (they aren't IOPL sensitive in Protected Mode either).

Note that the I/O instructions (IN, OUT, INS, OUTS, REP INS, and REP OUTS) are **not** IOPL-sensitive in Virtual 8086 mode. Rather, the I/O instructions become automatically sensitive to the **I/O Permission Bitmap** contained in the **Intel386 DX Task State Segment**. The I/O Permission Bitmap, automatically used by the Intel386 DX in Virtual 8086 Mode, is illustrated by Figures 4.15a and 4-15b.

The I/O Permission Bitmap can be viewed as a 0–64 Kbit bit string, which begins in memory at offset Bit_Map_Offset in the current TSS. Bit_Map_Offset must be ≤ DFFFH so the entire bit map and the byte FFH which follows the bit map are all at offsets ≤ FFFFH from the TSS base. The 16-bit pointer Bit_Map_Offset (15:0) is found in the word beginning at offset 66H (102 decimal) from the TSS base, as shown in Figure 4-15a.

Each bit in the I/O Permission Bitmap corresponds to a single byte-wide I/O port, as illustrated in Figure 4-15a. If a bit is 0, I/O to the corresponding byte-wide port can occur without generating an exception. Otherwise the I/O instruction causes an exception 13 fault. Since every byte-wide I/O port must be protectable, all bits corresponding to a word-wide or dword-wide port must be 0 for the word-wide or dword-wide I/O to be permitted. If all the referenced bits are 0, the I/O will be allowed. If any referenced bits are 1, the attempted I/O will cause an exception 13 fault.

Due to the use of a pointer to the base of the I/O Permission Bitmap, the bitmap may be located anywhere within the TSS, or may be ignored completely by pointing the Bit_Map_Offset (15:0) beyond the limit of the TSS segment. In the same manner, only a small portion of the 64K I/O space need have an associated map bit, by adjusting the TSS limit to truncate the bitmap. This eliminates the commitment of 8K of memory when a complete bitmap is not required, while allowing the fully general case if desired.

EXAMPLE OF BITMAP FOR I/O PORTS 0–255: Setting the TSS limit to {bit_Map_Offset + 31 +1**} [** see note below] will allow a 32-byte bitmap for the I/O ports #0–255, plus a terminator byte of all 1's [** see note below]. This allows the I/O bitmap to control I/O Permission to I/O port 0–255 while causing an exception 13 fault on attempted I/O to any I/O port 80256 through 65,565.

****IMPORTANT IMPLEMENTATION NOTE:** Beyond the last byte of I/O mapping information in the I/O Permission Bitmap **must** be a byte containing all 1's. The byte of all 1's must be within the limit of the Intel386 DX TSS segment (see Figure 4-15a).

4.6.5 Interrupt Handling

In order to fully support the emulation of an 8086 machine, interrupts in Virtual 8086 Mode are handled in a unique fashion. When running in Virtual Mode all interrupts and exceptions involve a privilege change back to the host Intel386 DX operating system. The Intel386 DX operating system determines if the interrupt comes from a Protected Mode application or from a Virtual Mode program by examining the VM bit in the EFLAGS image stored on the stack.

When a Virtual Mode program is interrupted and execution passes to the interrupt routine at level 0, the VM bit is cleared. However, the VM bit is still set in the EFLAG image on the stack.

The Intel386 DX operating system in turn handles the exception or interrupt and then returns control to the 8086 program. The Intel386 DX operating system may choose to let the 8086 operating system handle the interrupt or it may emulate the function of the interrupt handler. For example, many 8086 operating system calls are accessed by PUSHing parameters on the stack, and then executing an INT n instruction. If the IOPL is set to 0 then all INT n instructions will be intercepted by the Intel386 DX Microprocessor operating system. The Intel386 DX operating system could emulate the 8086 operating system's call. Figure 4-25 shows how the Intel386 DX operating system could intercept an 8086 operating system's call to "Open a File".

An Intel386 DX operating system can provide a Virtual 8086 Environment which is totally transparent to the application software via intercepting and then emulating 8086 operating system's calls, and intercepting IN and OUT instructions.

4.6.6 Entering and Leaving Virtual 8086 Mode

Virtual 8086 mode is entered by executing an IRET instruction (at CPL = 0), or Task Switch (at any CPL) to an Intel386 DX task whose Intel386 DX TSS has a FLAGS image containing a 1 in the VM bit position while the processor is executing in Protected Mode. That is, one way to enter Virtual 8086 mode is to switch to a task with an Intel386 DX TSS that has a 1 in the VM bit in the EFLAGS image. The other way is to execute a 32-bit IRET instruction at privilege level 0, where the stack has a 1 in the VM bit in the EFLAGS image. POPF does not affect the VM bit, even if the processor is in Protected Mode or level 0, and so cannot be used to enter Virtual 8086 Mode. PUSHF always pushes a 0 in the VM bit, even if the processor is in Virtual 8086 Mode, so that a program cannot tell if it is executing in REAL mode, or in Virtual 8086 mode.

The VM bit can be set by executing an IRET instruction only at privilege level 0, or by any instruction or Interrupt which causes a task switch in Protected Mode (with VM = 1 in the new FLAGS image), and can be cleared only by an interrupt or exception in Virtual 8086 Mode. IRET and POPF instructions executed in REAL mode or Virtual 8086 mode will not change the value in the VM bit.

The transition out of virtual 8086 mode to Intel386 DX protected mode occurs only on receipt of an interrupt or exception (such as due to a sensitive instruction). In Virtual 8086 mode, all interrupts and exceptions vector through the protected mode IDT, and enter an interrupt handler in protected Intel386 DX mode. That is, as part of interrupt processing, the VM bit is cleared.

Because the matching IRET must occur from level 0, if an Interrupt or Trap Gate is used to field an interrupt or exception out of Virtual 8086 mode, the Gate must perform an inter-level interrupt only to level 0. Interrupt or Trap Gates through conforming segments, or through segments with DPL > 0, will raise a GP fault with the CS selector as the error code.

4.6.6.1 TASK SWITCHES TO/FROM VIRTUAL 8086 MODE

Tasks which can execute in virtual 8086 mode must be described by a TSS with the new Intel386 DX format (TYPE 9 or 11 descriptor).

A task switch out of virtual 8086 mode will operate exactly the same as any other task switch out of a task with an Intel386 DX TSS. All of the programmer visible state, including the FLAGS register with the VM bit set to 1, is stored in the TSS. The segment registers in the TSS will contain 8086 segment base values rather than selectors.

A task switch into a task described by an Intel386 DX TSS will have an additional check to determine if the incoming task should be resumed in virtual 8086 mode. Tasks described by 80286 format TSSs cannot be resumed in virtual 8086 mode, so no check is required there (the FLAGS image in 80286 format TSS has only the low order 16 FLAGS bits). Before loading the segment register images from an Intel386 DX TSS, the FLAGS image is loaded, so that the segment registers are loaded from the TSS image as 8086 segment base values. The task is now ready to resume in virtual 8086 execution mode.

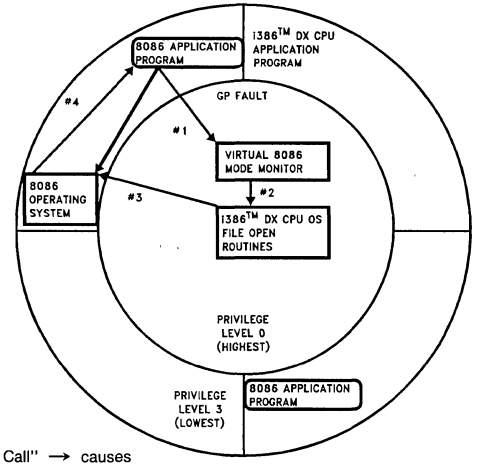
4.6.6.2 TRANSITIONS THROUGH TRAP AND INTERRUPT GATES, AND IRET

A task switch is one way to enter or exit virtual 8086 mode. The other method is to exit through a Trap or Interrupt gate, as part of handling an interrupt, and to enter as part of executing an IRET instruction. The transition out must use an Intel386 DX Trap Gate (Type 14), or Intel386 DX Interrupt Gate (Type 15), which must point to a non-conforming level 0 segment (DPL = 0) in order to permit the trap handler to IRET back to the Virtual 8086 program. The Gate must point to a non-conforming level 0 segment to perform a level switch to level 0 so that the matching IRET can change the VM bit. Intel386 DX gates must be used, since 80286 gates save only the low 16 bits of the FLAGS register, so that the VM bit will not be saved on transitions through the 80286 gates. Also, the 16-bit IRET (presumably) used to terminate the 80286 interrupt handler will pop only the lower 16 bits from FLAGS, and will not affect the VM bit. The action taken for an Intel386 DX Trap or Interrupt gate if an interrupt occurs while the task is executing in virtual 8086 mode is given by the following sequence.

(1) Save the FLAGS register in a temp to push later. Turn off the VM and TF bits, and if the interrupt is serviced by an Interrupt Gate, turn off IF also.

(2) Interrupt and Trap gates must perform a level switch from 3 (where the VM86 program executes) to level 0 (so IRET can return). This process involves a stack switch to the stack given in the TSS for privilege level 0. Save the Virtual 8086 Mode SS and ESP registers to push in a later step. The segment register load of SS will be done as a Protected Mode segment load, since the VM bit was turned off above.

(3) Push the 8086 segment register values onto the new stack, in the order: GS, FS, DS, ES. These are pushed as 32-bit quantities, with undefined values in the upper 16 bits. Then load these 4 registers with null selectors (0).



8086 Application makes "Open File Call" → causes General Protection Fault (Arrow #1)
Virtual 8086 Monitor intercepts call. Calls Intel386™ DX OS (Arrow #2)
Intel386™ DX OS opens file returns control to 8086 OS (Arrow #3)
8086 OS returns control to application. (Arrow #4)
Transparent to Application

231630-70

Figure 4-25. Virtual 8086 Environment Interrupt and Call Handling

(4) Push the old 8086 stack pointer onto the new stack by pushing the SS register (as 32-bits, high bits undefined), then pushing the 32-bit ESP register saved above.

(5) Push the 32-bit FLAGS register saved in step 1.

(6) Push the old 8086 instruction pointer onto the new stack by pushing the CS register (as 32-bits, high bits undefined), then pushing the 32-bit EIP register.

(7) Load up the new CS:EIP value from the interrupt gate, and begin execution of the interrupt routine in protected Intel386 DX mode.

The transition out of virtual 8086 mode performs a level change and stack switch, in addition to changing back to protected mode. In addition, all of the 8086 segment register images are stored on the stack (behind the SS:ESP image), and then loaded with null (0) selectors before entering the interrupt handler. This will permit the handler to safely save and restore the DS, ES, FS, and GS registers as 80286 selectors. This is needed so that interrupt handlers which don't care about the mode of the interrupted program can use the same prolog and epilog code for state saving (i.e. push all registers in prolog, pop all in epilog) regardless of whether or not a "native" mode or Virtual 8086 mode program was interrupted. Restoring null selectors to these registers before executing the IRET will not cause a trap in the interrupt handler. Interrupt routines which expect values in the segment registers, or return values in segment registers will have to obtain/return values from the 8086 register images pushed onto the new stack. They will need to know the mode of the interrupted program in order to know where to find/return segment registers, and also to know how to interpret segment register values.

The IRET instruction will perform the inverse of the above sequence. Only the extended Intel386 DXs IRET instruction (operand size=32) can be used, and must be executed at level 0 to change the VM bit to 1.

(1) If the NT bit in the FLAGs register is on, an inter-task return is performed. The current state is stored in the current TSS, and the link field in the current TSS is used to locate the TSS for the interrupted task which is to be resumed.

Otherwise, continue with the following sequence.

(2) Read the FLAGS image from SS:8[ESP] into the FLAGS register. This will set VM to the value active in the interrupted routine.

(3) Pop off the instruction pointer CS:EIP. EIP is popped first, then a 32-bit word is popped which contains the CS value in the lower 16 bits. If VM=0, this CS load is done as a protected mode segment load. If VM=1, this will be done as an 8086 segment load.

(4) Increment the ESP register by 4 to bypass the FLAGS image which was "popped" in step 1.

(5) If VM=1, load segment registers ES, DS, FS, and GS from memory locations SS:[ESP+8], SS:[ESP+12], SS:[ESP+16], and SS:[ESP+20], respectively, where the new val-

ue of ESP stored in step 4 is used. Since VM = 1, these are done as 8086 segment register loads.

Else if VM = 0, check that the selectors in ES, DS, FS, and GS are valid in the interrupted routine. Null out invalid selectors to trap if an attempt is made to access through them.

(6) If (RPL(CS) > CPL), pop the stack pointer SS:ESP from the stack. The ESP register is popped first, followed by 32-bits containing SS in the lower 16 bits. If VM = 0, SS is loaded as a protected mode segment register load. If VM = 1, an 8086 segment register load is used.

(7) Resume execution of the interrupted routine. The VM bit in the FLAGS register (restored from the interrupt routine's stack image in step 1) determines whether the processor resumes the interrupted routine in Protected mode of Virtual 8086 mode.

5. FUNCTIONAL DATA

5.1 INTRODUCTION

The Intel386 DX features a straightforward functional interface to the external hardware. The Intel386 DX has separate, parallel buses for data and address. The data bus is 32-bits in width, and bidirectional. The address bus outputs 32-bit address values in the most directly usable form for the high-speed local bus: 4 individual byte enable signals, and the 30 upper-order bits as a binary value. The data and address buses are interpreted and controlled with their associated control signals.

A **dynamic data bus sizing** feature allows the processor to handle a mix of 32- and 16-bit external buses on a cycle-by-cycle basis (see **5.3.4 Data Bus Sizing**). If 16-bit bus size is selected, the Intel386 DX automatically makes any adjustment needed, even performing another 16-bit bus cycle to complete the transfer if that is necessary. 8-bit peripheral devices may be connected to 32-bit or 16-bit buses with no loss of performance. A **new address pipelining option** is provided and applies to 32-bit and 16-bit buses for substantially improved memory utilization, especially for the most heavily used memory resources.

The **address pipelining option**, when selected, typically allows a given memory interface to operate with one less wait state than would otherwise be required (see **5.4.2 Address Pipelining**). The pipelined bus is also well suited to interleaved memory designs. When address pipelining is requested by the external hardware, the Intel386 DX will output the address and bus cycle definition of the next bus cycle (if it is internally available) even while waiting for the current cycle to be acknowledged.

Non-pipelined address timing, however, is ideal for external cache designs, since the cache memory will typically be fast enough to allow non-pipelined cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor, or from processor to system. Intel386 DX bus cycles perform data transfer in a minimum of only two clock periods. On a 32-bit data bus, the maximum Intel386 DX transfer bandwidth at 20 MHz is therefore 40 MBytes/sec, at 25 MHz bandwidth, is 50 Mbytes/sec, and at 33 MHz bandwidth, is 66 Mbytes/sec. Any bus cycle will be extended for more than two clock periods, however, if external hardware withholds acknowledgement of the cycle. At the appropriate time, acknowledgement is signalled by asserting the Intel386 DX READY# input.

The Intel386 DX can relinquish control of its local buses to allow mastership by other devices, such as direct memory access channels. When relinquished, HLDA is the only output pin driven by the Intel386 DX providing near-complete isolation of the processor from its system. The near-complete isolation characteristic is ideal when driving the system from test equipment, and in fault-tolerant applications.

Functional data covered in this chapter describes the processor's hardware interface. First, the set of signals available at the processor pins is described (see **5.2 Signal Description**). Following that are the signal waveforms occurring during bus cycles (see **5.3 Bus Transfer Mechanism**, **5.4 Bus Functional Description** and **5.5 Other Functional Descriptions**).

5.2 SIGNAL DESCRIPTION

5.2.1 Introduction

Ahead is a brief description of the Intel386 DX input and output signals arranged by functional groups. Note the # symbol at the end of a signal name indicates the active, or asserted, state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

Example signal: M/IO# — High voltage indicates Memory selected

— Low voltage indicates I/O selected

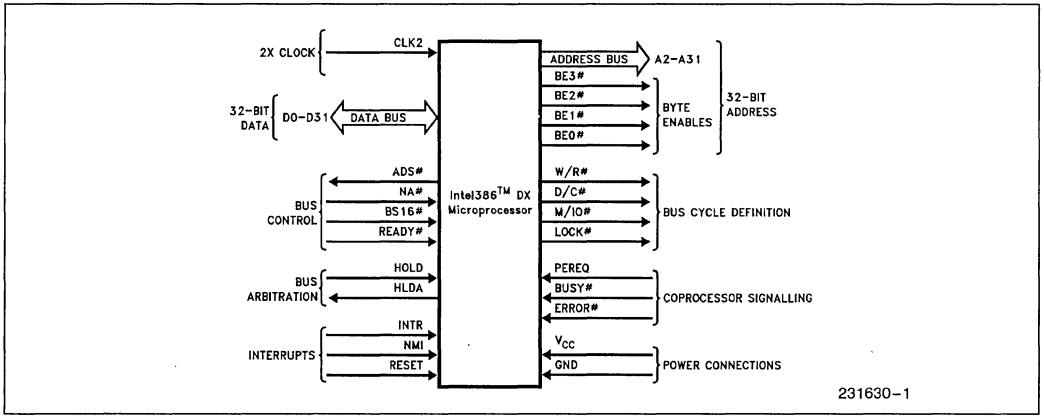


Figure 5-1. Functional Signal Groups

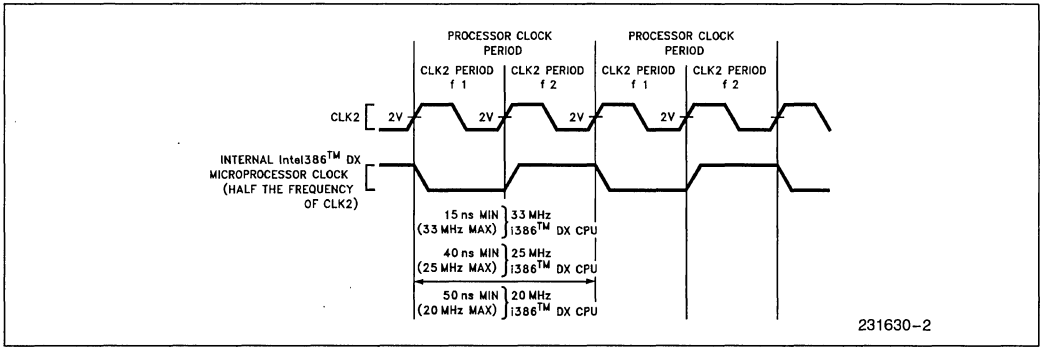


Figure 5-2. CLK2 Signal and Internal Processor Clock

The signal descriptions sometimes refer to AC timing parameters, such as "t_{25} Reset Setup Time" and "t_{26} Reset Hold Time." The values of these parameters can be found in Tables 7-4 and 7-5.

5.2.2 Clock (CLK2)

CLK2 provides the fundamental timing for the Intel386 DX. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two phases, "phase one" and "phase two." Each CLK2 period is a phase of the internal clock. Figure 5-2 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the RESET signal falling edge meets its applicable setup and hold times, t_{25} and t_{26}.

5.2.3 Data Bus (D0 through D31)

These three-state bidirectional signals provide the general purpose data path between the Intel386 DX and other devices. Data bus inputs and outputs indicate "1" when HIGH. The data bus can transfer data on 32- and 16-bit buses using a data bus sizing feature controlled by the BS16# input. See section **5.2.6 Bus Contol**. Data bus reads require that read data setup and hold times t_{21} and t_{22} be met for correct operation. In addition, the Intel386 DX requires that all data bus pins be at a valid logic state (high or low) at the end of each read cycle, when READY# is asserted. During any write operation (and during halt cycles and shutdown cycles), the Intel386 DX always drives all 32 signals of the data bus even if the current bus size is 16-bits.

5.2.4 Address Bus (BE0# through BE3#, A2 through A31)

These three-state outputs provide physical memory addresses or I/O port addresses. The address bus is capable of addressing 4 gigabytes of physical memory space (00000000H through FFFFFFFFH), and 64 kilobytes of I/O address space (00000000H through 0000FFFFH) for programmed I/O. I/O

transfers automatically generated for Intel386 DX-to-coprocessor communication use I/O addresses 800000F8H through 800000FFH, so A31 HIGH in conjunction with M/IO# LOW allows simple generation of the coprocessor select signal.

The Byte Enable outputs, BE0#–BE3#, directly indicate which bytes of the 32-bit data bus are involved with the current transfer. This is most convenient for external hardware.

BE0# applies to D0–D7
BE1# applies to D8–D15
BE2# applies to D16–D23
BE3# applies to D24–D31

The number of Byte Enables asserted indicates the physical size of the operand being transferred (1, 2, 3, or 4 bytes). Refer to section **5.3.6 Operand Alignment**.

When a memory write cycle or I/O write cycle is in progress, and the operand being transferred occupies **only** the upper 16 bits of the data bus (D16–D31), duplicate data is simultaneously presented on the corresponding lower 16-bits of the data bus (D0–D15). This duplication is performed for optimum write performance on 16-bit buses. The pattern of write data duplication is a function of the Byte Enables asserted during the write cycle. Table 5-1 lists the write data present on D0–D31, as a function of the asserted Byte Enable outputs BE0#–BE3#.

5.2.5 Bus Cycle Definition Signals (W/R#, D/C#, M/IO#, LOCK#)

These three-state outputs define the type of bus cycle being performed. W/R# distinguishes between write and read cycles. D/C# distinguishes between data and control cycles. M/IO# distinguishes between memory and I/O cycles. LOCK# distinguishes between locked and unlocked bus cycles.

The primary bus cycle definition signals are W/R#, D/C# and M/IO#, since these are the signals driven valid as the ADS# (Address Status output) is driven asserted. The LOCK# is driven valid at the same time as the first locked bus cycle begins, which due to address pipelining, could be later than ADS# is driven asserted. See **5.4.3.4 Pipelined Address.** The LOCK# is negated when the READY# input terminates the last bus cycle which was locked.

Exact bus cycle definitions, as a function of W/R#, D/C#, and M/IO#, are given in Table 5-2. Note one combination of W/R#, D/C# and M/IO# is never given when ADS# is asserted (however, that combination, which is listed as "does not occur," may occur during **idle** bus states when ADS# is **not** asserted). If M/IO#, D/C#, and W/R# are qualified by ADS# asserted, then a decoding scheme may be simplified by using this definition of the "does not occur" combination.

Table 5-1. Write Data Duplication as a Function of BE0#–BE3#

Intel386™ DX Byte Enables				Intel386™ DX Write Data				Automatic Duplication?
BE3#	BE2#	BE1#	BE0#	D24–D31	D16–D23	D8–D15	D0–D7	
High	High	High	Low	undef	undef	undef	A	No
High	High	Low	High	undef	undef	B	undef	No
High	Low	High	High	undef	C	undef	C	Yes
Low	High	High	High	D	undef	D	undef	Yes
High	High	Low	Low	undef	undef	B	A	No
High	Low	Low	High	undef	C	B	undef	No
Low	Low	High	High	D	C	D	C	Yes
High	Low	Low	Low	undef	C	B	A	No
Low	Low	Low	High	D	C	B	undef	No
Low	Low	Low	Low	D	C	B	A	No

Key:
D = logical write data d24–d31
C = logical write data d16–d23
B = logical write data d8–d15
A = logical write data d0–d7.

Table 5-2. Bus Cycle Definition

M/IO#	D/C#	W/R#	Bus Cycle Type	Locked?
Low	Low	Low	INTERRUPT ACKNOWLEDGE	Yes
Low	Low	High	does not occur	—
Low	High	Low	I/O DATA READ	No
Low	High	High	I/O DATA WRITE	No
High	Low	Low	MEMORY CODE READ	No
High	Low	High	HALT: Address = 2 (BE0# High, BE1# High, BE2# Low, BE3# High, A2–A31 Low) SHUTDOWN: Address = 0 (BE0# Low, BE1# High, BE2# High, BE3# High, A2–A31 Low)	No
High	High	Low	MEMORY DATA READ	Some Cycles
High	High	High	MEMORY DATA WRITE	Some Cycles

5.2.6 Bus Control Signals (ADS#, READY#, NA#, BS16#)

5.2.6.1 INTRODUCTION

The following signals allow the processor to indicate when a bus cycle has begun, and allow other system hardware to control address pipelining, data bus width and bus cycle termination.

5.2.6.2 ADDRESS STATUS (ADS#)

This three-state output indicates that a valid bus cycle definition, and address (W/R#, D/C#, M/IO#, BE0#–BE3#, and A2–A31) is being driven at the Intel386 DX pins. It is asserted during T1 and T2P bus states (see **5.4.3.2 Non-pipelined Address** and **5.4.3.4 Pipelined Address** for additional information on bus states).

5.2.6.3 TRANSFER ACKNOWLEDGE (READY#)

This input indicates the current bus cycle is complete, and the active bytes indicated by BE0#–BE3# and BS16# are accepted or provided. When READY# is sampled asserted during a read cycle or interrupt acknowledge cycle, the Intel386 DX latches the input data and terminates the cycle. When READY# is sampled asserted during a write cycle, the processor terminates the bus cycle.

READY# is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. READY# must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, READY must always meet setup and hold times t_{19} and t_{20} for correct operation. See all sections of **5.4 Bus Functional Description**.

5.2.6.4 NEXT ADDRESS REQUEST (NA#)

This is used to request address pipelining. This input indicates the system is prepared to accept new values of BE0#–BE3#, A2–A31, W/R#, D/C# and M/IO# from the Intel386 DX even if the end of the current cycle is not being acknowledged on READY#. If this input is asserted when sampled, the next address is driven onto the bus, provided the next bus request is already pending internally. See **5.4.2 Address Pipelining** and **5.4.3 Read and Write Cycles**. NA# must always meet setup and hold times, t_{15} and t_{16}, for correct operation.

5.2.6.5 BUS SIZE 16 (BS16#)

The BS16# feature allows the Intel386 DX to directly connect to 32-bit and 16-bit data buses. Asserting this input constrains the current bus cycle to use only the lower-order half (D0–D15) of the data bus, corresponding to BE0# and BE1#. Asserting BS16# has no additional effect if only BE0# and/or BE1# are asserted in the current cycle. However, during bus cycles asserting BE2# or BE3#, asserting BS16# will automatically cause the Intel386 DX to make adjustments for correct transfer of the upper bytes(s) using only physical data signals D0–D15.

If the operand spans both halves of the data bus and BS16# is asserted, the Intel386 DX will automatically perform another 16-bit bus cycle. BS16# must always meet setup and hold times t_{17} and t_{18} for correct operation.

Intel386 DX I/O cycles are automatically generated for coprocessor communication. Since the Intel386 DX must transfer 32-bit quantities between itself and the Intel387 DX, BS16# *must not* be asserted during Intel387 DX communication cycles.

5.2.7 Bus Arbitration Signals (HOLD, HLDA)

5.2.7.1 INTRODUCTION

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See **5.5.1 Entering and Exiting Hold Acknowledge** for additional information.

5.2.7.2 BUS HOLD REQUEST (HOLD)

This input indicates some device other than the Intel386 DX requires bus mastership.

HOLD must remain asserted as long as any other device is a local bus master. HOLD is not recognized while RESET is asserted. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the hold acknowledge (high impedance) state.

HOLD is level-sensitive and is a synchronous input. HOLD signals must always meet setup and hold times t_{23} and t_{24} for correct operation.

5.2.7.3 BUS HOLD ACKNOWLEDGE (HLDA)

Assertion of this output indicates the Intel386 DX has relinquished control of its local bus in response to HOLD asserted, and is in the bus Hold Acknowledge state.

The Hold Acknowledge state offers near-complete signal isolation. In the Hold Acknowledge state, HLDA is the only signal being driven by the Intel386 DX. The other output signals or bidirectional signals (D0–D31, BE0#–BE3#, A2–A31, W/R#, D/C#, M/IO#, LOCK# and ADS#) are in a high-impedance state so the requesting bus master may control them. Pullup resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See **7.2.3 Resistor Recommendations**. Also, one rising edge occuring on the NMI input during Hold Acknowledge is remembered, for processing after the HOLD input is negated.

In addition to the normal usage of Hold Acknowledge with DMA controllers or master peripherals, the near-complete isolation has particular attractiveness during system test when test equipment drives the system, and in hardware-fault-tolerant applications.

5.2.8 Coprocessor Interface Signals (PEREQ, BUSY#, ERROR#)

5.2.8.1 INTRODUCTION

In the following sections are descriptions of signals dedicated to the numeric coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, these following signals control communication between the Intel386 DX and its Intel387 DX processor extension.

5.2.8.2 COPROCESSOR REQUEST (PEREQ)

When asserted, this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the Intel386 DX. In response, the Intel386 DX transfers information between the coprocessor and memory. Because the Intel386 DX has internally stored the coprocessor opcode being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

5.2.8.3 COPROCESSOR BUSY (BUSY#)

When asserted, this input indicates the coprocessor is still executing an instruction, and is not yet able to accept another. When the Intel386 DX encounters any coprocessor instruction which operates on the numeric stack (e.g. load, pop, or arithmetic operation), or the WAIT instruction, this input is first automatically sampled until it is seen to be negated. This sampling of the BUSY# input prevents overrunning the execution of a previous coprocessor instruction.

The FNINIT and FNCLEX coprocessor instructions are allowed to execute even if BUSY# is asserted, since these instructions are used for coprocessor initialization and exception-clearing.

BUSY# is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

BUSY# serves an additional function. If BUSY# is sampled LOW at the falling edge of RESET, the Intel386 DX performs an internal self-test (see **5.5.3 Bus Activity During and Following Reset**). If BUSY# is sampled HIGH, no self-test is performed.

5.2.8.4 COPROCESSOR ERROR (ERROR#)

This input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the Intel386 DX when a coprocessor instruction is encountered, and if asserted, the Intel386 DX generates exception 16 to access the error-handling software.

Several coprocessor instructions, generally those which clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the Intel386 DX generating exception 16 even if ERROR# is asserted. These instructions are FNINIT, FNCLEX, FSTSW, FSTSWAX, FSTCW, FSTENV, FSAVE, FESTENV and FESAVE.

ERROR# is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

5.2.9 Interrupt Signals (INTR, NMI, RESET)

5.2.9.1 INTRODUCTION

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

5.2.9.2 MASKABLE INTERRUPT REQUEST (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the Intel386 DX Flag Register IF bit. When the Intel386 DX responds to the INTR input, it performs two interrupt acknowledge bus cycles, and at the end of the second, latches an 8-bit interrupt vector on D0–D7 to identify the source of the interrupt.

INTR is level-sensitive and is allowed to be asynchronous to the CLK2 signal. To assure recognition of an INTR request, INTR should remain asserted until the first interrupt acknowledge bus cycle begins.

5.2.9.3 NON-MASKABLE INTERRUPT REQUEST (NMI)

This input indicates a request for interrupt service, which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are perfomed when processing NMI.

NMI is rising edge-sensitive and is allowed to be asynchronous to the CLK2 signal. To assure recognition of NMI, it must be negated for at least eight CLK2 periods, and then be asserted for at least eight CLK2 periods.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

5.2.9.4 RESET (RESET)

This input signal suspends any operation in progress and places the Intel386 DX in a known reset state. The Intel386 DX is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self test). When RESET is asserted, all other input pins are ignored, and all other bus pins are driven to an idle bus state as shown in Table 5-3. If RESET and HOLD are both asserted at a point in time, RESET takes priority even if the Intel386 DX was in a Hold Acknowledge state prior to RESET asserted.

RESET is level-sensitive and must be synchronous to the CLK2 signal. If desired, the phase of the internal processor clock, and the entire Intel386 DX state can be completely synchronized to external circuitry by ensuring the RESET signal falling edge meets its applicable setup and hold times, t_{25} and t_{26}.

Table 5-3. Pin State (Bus Idle) During Reset

Pin Name	Signal Level During Reset
ADS#	High
D0–D31	High Impedance
BE0#–BE3#	Low
A2–A31	High
W/R#	Low
D/C#	High
M/IO#	Low
LOCK#	High
HLDA	Low

5.2.10 Signal Summary

Table 5-4 summarizes the characteristics of all Intel386 DX signals.

Table 5-4. Intel386™ DX Signal Summary

Signal Name	Signal Function	Active State	Input/ Output	Input Synch or Asynch to CLK2	Output High Impedance During HLDA?
CLK2	Clock	—	I	—	—
D0–D31	Data Bus	High	I/O	S	Yes
BE0#–BE3#	Byte Enables	Low	O	—	Yes
A2–A31	Address Bus	High	O	—	Yes
W/R#	Write-Read Indication	High	O	—	Yes
D/C#	Data-Control Indication	High	O	—	Yes
M/IO#	Memory-I/O Indication	High	O	—	Yes
LOCK#	Bus Lock Indication	Low	O	—	Yes
ADS#	Address Status	Low	O	—	Yes
NA#	Next Address Request	Low	I	S	—
BS16#	Bus Size 16	Low	I	S	—
READY#	Transfer Acknowledge	Low	I	S	—
HOLD	Bus Hold Request	High	I	S	—
HLDA	Bus Hold Acknowledge	High	O	—	No
PEREQ	Coprocessor Request	High	I	A	—
BUSY#	Coprocessor Busy	Low	I	A	—
ERROR#	Coprocessor Error	Low	I	A	—
INTR	Maskable Interrupt Request	High	I	A	—
NMI	Non-Maskable Intrpt Request	High	I	A	—
RESET	Reset	High	I	S	—

5.3 BUS TRANSFER MECHANISM

5.3.1 Introduction

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte, word and double-word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two or even three physical bus cycles are performed as required for unaligned operand transfers. See **5.3.4 Dynamic Data Bus Sizing** and **5.3.6 Operand Alignment**.

The Intel386 DX address signals are designed to simplify external system hardware. Higher-order address bits are provided by A2–A31. Lower-order address in the form of BE0#–BE3# directly provides linear selects for the four bytes of the 32-bit data bus. Physical operand size information is thereby implicitly provided each bus cycle in the most usable form.

Byte Enable outputs BE0#–BE3# are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 5-5. During a bus cycle, any possible pattern of contiguous, asserted Byte Enable outputs can occur, but never patterns having a negated Byte Enable separating two or three asserted Enables.

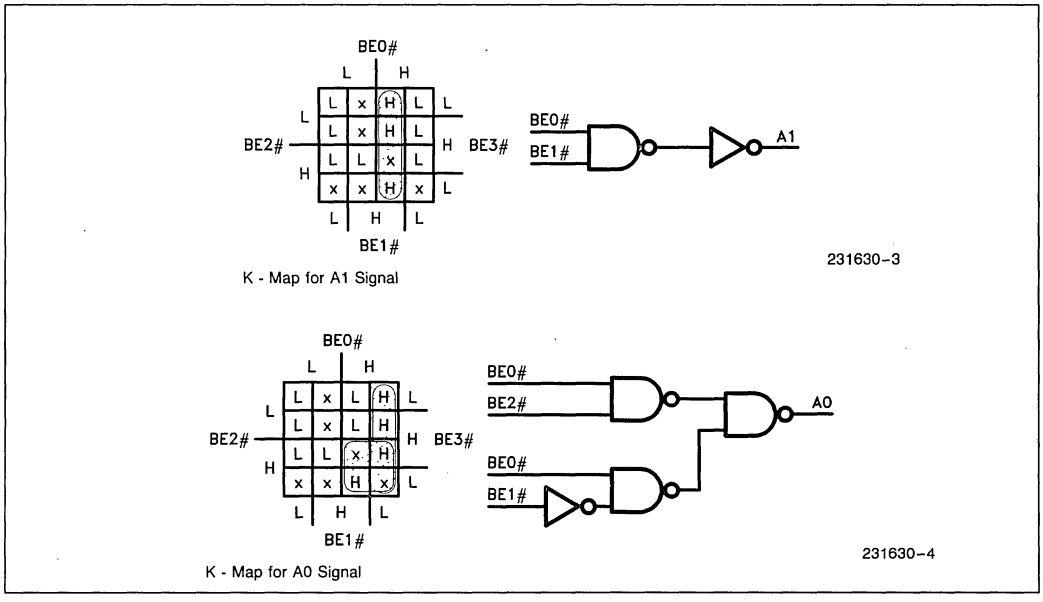
Address bits A0 and A1 of the physical operand's base address can be created when necessary (for instance, for MULTIBUS I or MULTIBUS II interface), as a function of the lowest-order asserted Byte Enable. This is shown by Table 5-6. Logic to generate A0 and A1 is given by Figure 5-3.

Table 5-5. Byte Enables and Associated Data and Operand Bytes

Byte Enable Signal	Associated Data Bus Signals
BE0#	D0–D7 (byte 0—least significant)
BE1#	D8–D15 (byte 1)
BE2#	D16–D23 (byte 2)
BE3#	D24–D31 (byte 3—most significant)

Table 5-6. Generating A0–A31 from BE0#–BE3# and A2–A31

Intel386™ DX Address Signals								
A31 ..... A2					BE3#	BE2#	BE1#	BE0#
	Physical Base Address							
A31	.....	A2	A1	A0				
A31	.....	A2	0	0	X	X	X	Low
A31	.....	A2	0	1	X	X	Low	High
A31	.....	A2	1	0	X	Low	High	High
A31	.....	A2	1	1	Low	High	High	High



K - Map for A1 Signal

K - Map for A0 Signal

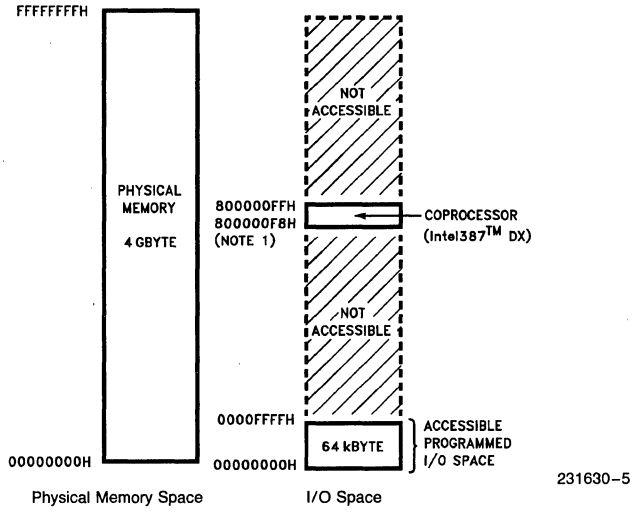
Figure 5-3. Logic to Generate A0, A1 from BE0#–BE3#

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See **5.4 Bus Functional Description**.

Since a bus cycle requires a minimum of two bus states (equal to two processor clock periods), data can be transferred between external devices and the Intel386 DX at a maximum rate of one 4-byte Dword every two processor clock periods, for a maximum bus bandwidth of 66 megabytes/second (Intel386 DX operating at 33 MHz processor clock rate).

5.3.2 Memory and I/O Spaces

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. As shown in Figure 5-4, physical memory addresses range from 00000000H to FFFFFFFFH (4 gigabytes) and I/O addresses from 00000000H to 0000FFFFH (64 kilobytes) for programmed I/O. Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 800000F8H to 800000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A31 and M/IO# signals.



NOTE:
Since A31 is HIGH during automatic communication with coprocessor, A31 HIGH and M/IO# LOW can be used to easily generate a coprocessor select signal.

Figure 5-4. Physical Memory and I/O Spaces

5.3.3 Memory and I/O Organization

The Intel386 DX datapath to memory and I/O spaces can be 32 bits wide or 16 bits wide. When 32-bits wide, memory and I/O spaces are organized naturally as arrays of physical 32-bit Dwords. Each memory or I/O Dword has four individually addressable bytes at consecutive byte addresses. The lowest-addressed byte is associated with data signals D0–D7; the highest-addressed byte with D24–D31.

The Intel386 DX includes a bus control input, BS16#, that also allows direct connection to 16-bit memory or I/O spaces organized as a sequence of 16-bit words. Cycles to 32-bit and 16-bit memory or I/O devices may occur in any sequence, since the BS16# control is sampled during each bus cycle. See **5.3.4 Dynamic Data Bus Sizing**. The Byte Enable signals, BE0#–BE3#, allow byte granularity when addressing any memory or I/O structure, whether 32 or 16 bits wide.

5.3.4 Dynamic Data Bus Sizing

Dynamic data bus sizing is a feature allowing direct processor connection to 32-bit or 16-bit data buses for memory or I/O. A single processor may connect to both size buses. Transfers to or from 32- or 16-bit ports are supported by dynamically determining the bus width during each bus cycle. During each bus cycle an address decoding circuit or the slave device itself may assert BS16# for 16-bit ports, or negate BS16# for 32-bit ports.

With BS16# asserted, the processor automatically converts operand transfers larger than 16 bits, or misaligned 16-bit transfers, into two or three transfers as required. All operand transfers physically occur on D0–D15 when BS16# is asserted. Therefore, 16-bit memories or I/O devices only connect on data signals D0–D15. No extra transceivers are required.

Asserting BS16# only affects the processor when BE2# and/or BE3# are asserted during the current cycle. If only D0–D15 are involved with the transfer, asserting BS16# has no affect since the transfer can proceed normally over a 16-bit bus whether BS16# is asserted or not. In other words, asserting BS16# has no effect when only the lower half of the bus is involved with the current cycle.

There are two types of situations where the processor is affected by asserting BS16#, depending on which Byte Enables are asserted during the current bus cycle:

Upper Half Only:
Only BE2# and/or BE3# asserted.

Upper and Lower Half:
At least BE1#, BE2# asserted (and perhaps also BE0# and/or BE3#).

Effect of asserting BS16# during "upper half only" read cycles:

Asserting BS16# during "upper half only" reads causes the Intel386 DX to read data on the lower 16 bits of the data bus and ignore data on the upper 16 bits of the data bus. Data that would have been read from D16–D31 (as indicated by BE2# and BE3#) will instead be read from D0–D15 respectively.

Effect of asserting BS16# during "upper half only" write cycles:

Asserting BS16# during "upper half only" writes does not affect the Intel386 DX. When only BE2# and/or BE3# are asserted during a write cycle the Intel386 DX always duplicates data signals D16–D31 onto D0–D15 (see Table 5-1). Therefore, no further Intel386 DX action is required to perform these writes on 32-bit or 16-bit buses.

Effect of asserting BS16# during "upper and lower half" read cycles:

Asserting BS16# during "upper and lower half" reads causes the processor to perform two 16-bit read cycles for complete physical operand transfer. Bytes 0 and 1 (as indicated by BE0# and BE1#) are read on the first cycle using D0–D15. Bytes 2 and 3 (as indicated by BE2# and BE3#) are read during the second cycle, again using D0–D15. D16–D31 are ignored during both 16-bit cycles. BE0# and BE1# are always negated during the second 16-bit cycle (See **Figure 5-14, cycles 2 and 2a**).

Effect of asserting BS16# during "upper and lower half" write cycles:

Asserting BS16# during "upper and lower half" writes causes the Intel386 DX to perform two 16-bit write cycles for complete physical operand transfer. All bytes are available the first write cycle allowing external hardware to receive Bytes 0 and 1 (as indicated by BE0# and BE1#) using D0–D15. On the second cycle the Intel386 DX duplicates Bytes 2 and 3 on D0–D15 and Bytes 2 and 3 (as indicated by BE2# and BE3#) are written using D0–D15. BE0# and BE1# are always negated during the second 16-bit cycle. BS16# must be asserted during the second 16-bit cycle. See **Figure 5-14, cycles 1 and 1a**.

5.3.5 Interfacing with 32- and 16-Bit Memories

In 32-bit-wide physical memories such as Figure 5-5, each physical Dword begins at a byte address that is a multiple of 4. A2–A31 are directly used as a Dword select and BE0#–BE3# as byte selects. BS16# is negated for all bus cycles involving the 32-bit array.

When 16-bit-wide physical arrays are included in the system, as in Figure 5-6, each 16-bit physical word begins at a address that is a multiple of 2. Note the address is decoded, to assert BS16# only during bus cycles involving the 16-bit array. (If desiring to

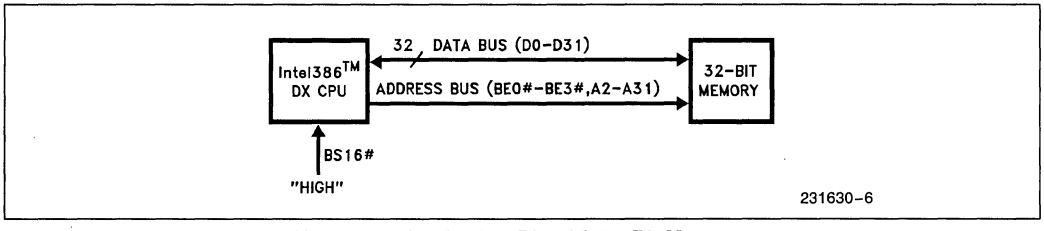


Figure 5-5. Intel386™ DX with 32-Bit Memory

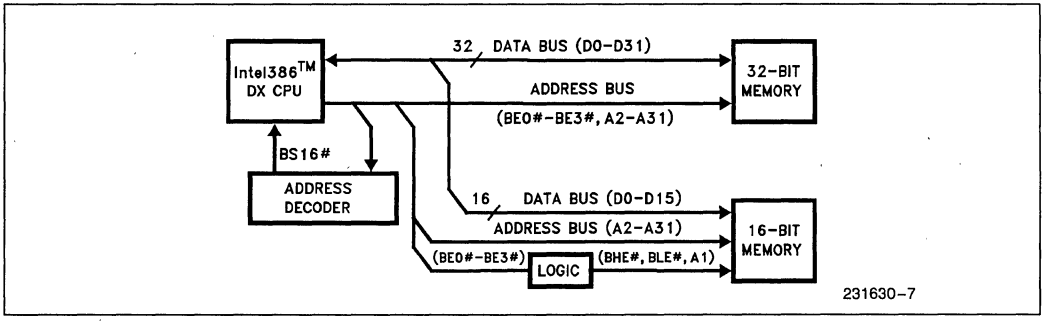


Figure 5-6. Intel386™ DX with 32-Bit and 16-Bit Memory

use pipelined address with 16-bit memories then BE0#–BE3# and W/R# are also decoded to determine when BS16# should be asserted. See **5.4.3.6 Pipelined Address with Dynamic Data Bus Sizing.**)

A2–A31 are directly usable for addressing 32-bit and 16-bit devices. To address 16-bit devices, A1 and two byte enable signals are also needed.

To generate an A1 signal and two Byte Enable signals for 16-bit access, BE0#–BE3# should be decoded as in Table 5-7. Note certain combinations of BE0#–BE3# are never generated by the Intel386 DX, leading to "don't care" conditions in the decoder. Any BE0#–BE3# decoder, such as Figure 5-7, may use the non-occurring BE0#–BE3# combinations to its best advantage.

5.3.6 Operand Alignment

With the flexibility of memory addressing on the Intel386 DX, it is possible to transfer a logical operand that spans more than one physical Dword or word of memory or I/O. Examples are 32-bit Dword operands beginning at addresses not evenly divisible by 4, or a 16-bit word operand split between two physical Dwords of the memory array.

Operand alignment and data bus size dictate when multiple bus cycles are required. Table 5-8 describes the transfer cycles generated for all combinations of logical operand lengths, alignment, and data bus sizing. When multiple bus cycles are required to transfer a multi-byte logical operand, the highest-order bytes are transferred first (but if BS16# asserted requires two 16-bit cycles be performed, that part of the transfer is low-order first).

5.4 BUS FUNCTIONAL DESCRIPTION

5.4.1 Introduction

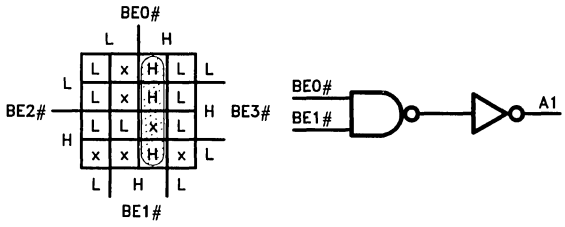
The Intel386 DX has separate, parallel buses for data and address. The data bus is 32-bits in width, and bidirectional. The address bus provides a 32-bit value using 30 signals for the 30 upper-order address bits and 4 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled via several associated definition or control signals.

Table 5-7. Generating A1, BHE# and BLE# for Addressing 16-Bit Devices

Intel386™ DX Signals				16-Bit Bus Signals			Comments
BE3#	BE2#	BE1#	BE0#	A1	BHE#	BLE# (A0)	
H*	H*	H*	H*	x	x	x	x—no active bytes
H	H	H	L	L	H	L	
H	H	L	H	L	L	H	
H	H	L	L	L	L	L	
H	L	H	H	H	H	L	
H*	L*	H*	L*	x	x	x	x—not contiguous bytes
H	L	L	H	L	L	H	
H	L	L	L	L	L	L	
L	H	H	H	H	L	H	
L*	H*	H*	L*	x	x	x	x—not contiguous bytes
L*	H*	L*	H*	x	x	x	x—not contiguous bytes
L*	H*	L*	L*	x	x	x	x—not contiguous bytes
L	L	H	H	H	L	L	
L*	L*	H*	L*	x	x	x	x—not continguous bytes
L	L	L	H	L	L	H	
L	L	L	L	L	L	L	

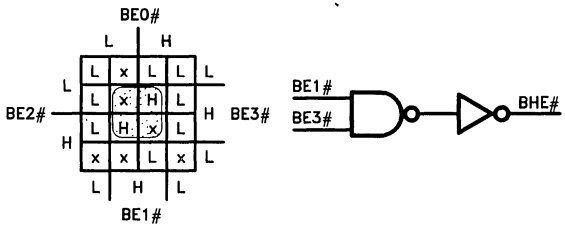
BLE# asserted when D0–D7 of 16-bit bus is active.
BHE# asserted when D8–D15 of 16-bit bus is active.
A1 low for all even words; A1 high for all odd words.

Key:
x = don't care
H = high voltage level
L = low voltage level
* = a non-occurring pattern of Byte Enables; either none are asserted, or the pattern has Byte Enables asserted for non-contiguous bytes



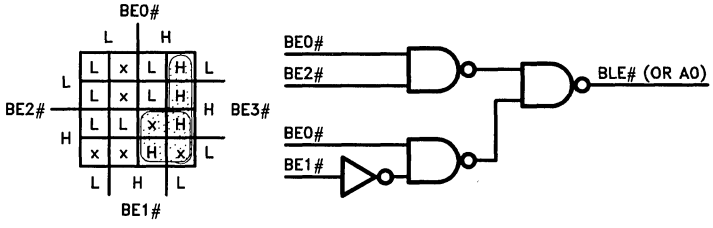
231630-8

K-map for A1 signal (same as Figure 5-3)



231630-9

K-map for 16-bit BHE# signal



231630-10

K-map for 16-bit BLE# signal (same as A0 signal in Figure 5-3)

Figure 5-7. Logic to Generate A1, BHE# and BLE# for 16-Bit Buses

Table 5-8. Transfer Bus Cycles for Bytes, Words and Dwords

	Byte-Length of Logical Operand								
	1	2				4			
Physical Byte Address in Memory (low-order bits)	xx	00	01	10	11	00	01	10	11
Transfer Cycles over 32-Bit Data Bus	b	w	w	w	hb,* lb	d	hb l3	hw, lw	h3, lb
Transfer Cycles over 16-Bit Data Bus	b	w	lb, hb	w	hb, lb	lw, hw	hb, lb, mw	hw, lw	mw, hb, lb

Key: b = byte transfer
w = word transfer
l = low-order portion
m = mid-order portion
x = don't care
= BS16# asserted causes second bus cycle
3 = 3-byte transfer
d = Dword transfer
h = high-order portion

*For this case, 8086, 8088, 80186, 80188, 80286 transfer lb first, then hb.

The definition of each bus cycle is given by three definition signals: M/IO#, W/R# and D/C#. At the same time, a valid address is present on the byte enable signals BE0#–BE3# and other address signals A2–A31. A status signal, ADS#, indicates when the Intel386 DX issues a new bus cycle definition and address.

Collectively, the address bus, data bus and all associated control signals are referred to simply as "the bus".

When active, the bus performs one of the bus cycles below:

1) read from memory space
2) locked read from memory space
3) write to memory space
4) locked write to memory space
5) read from I/O space (or coprocessor)
6) write to I/O space (or coprocessor)
7) interrupt acknowledge
8) indicate halt, or indicate shutdown

Table 5-2 shows the encoding of the bus cycle definition signals for each bus cycle. See section **5.2.5 Bus Cycle Definition**.

The data bus has a dynamic sizing feature supporting 32- and 16-bit bus size. Data bus size is indicated to the Intel386 DX using its Bus Size 16 (BS16#) input. All bus functions can be performed with either data bus size.

When the Intel386 DX bus is not performing one of the activities listed above, it is either Idle or in the Hold Acknowledge state, which may be detected by external circuitry. The idle state can be identified by the Intel386 DX giving no further assertions on its address strobe output (ADS#) since the beginning of its most recent bus cycle, and the most recent bus cycle has been terminated. The hold acknowledge state is identified by the Intel386 DX asserting its hold acknowledge (HLDA) output.

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

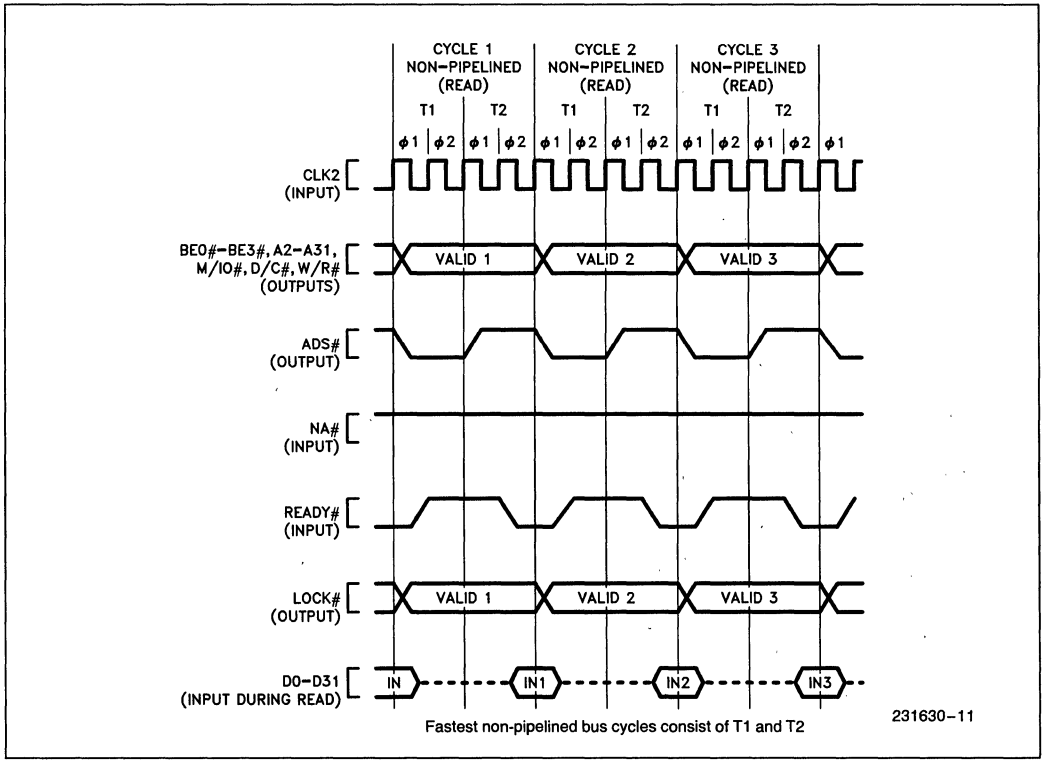


Figure 5-8. Fastest Read Cycles with Non-Pipelined Address Timing

The fastest Intel386 DX bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown by Figure 5-8. The bus states in each cycle are named **T1** and **T2**. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough. The high-bandwidth, two-clock bus cycle realizes the full potential of fast main memory, or cache memory.

Every bus cycle continues until it is acknowledged by the external system hardware, using the Intel386 DX READY# input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If READY# is not immediately asserted, however, T2 states are repeated indefinitely until the READY# input is sampled asserted.

5.4.2 Address Pipelining

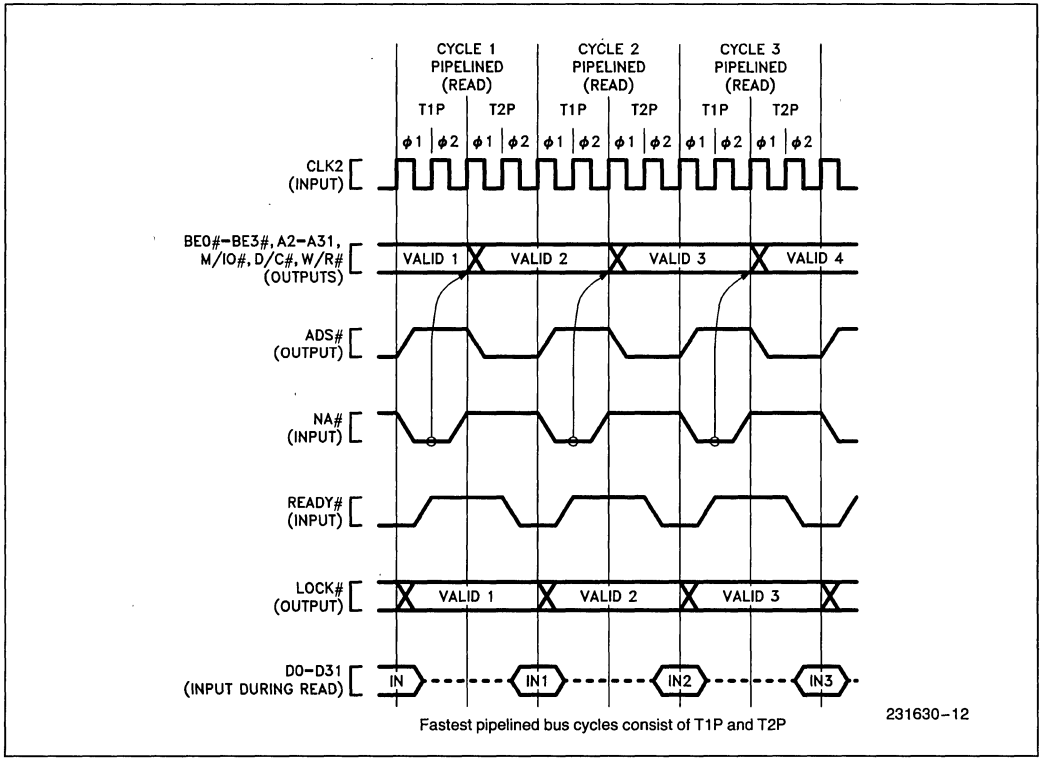
The address pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined address timing is selectable on a cycle-by-cycle basis with the Next Address (NA#) input.

When address pipelining is not selected, the current address and bus cycle definition remain stable throughout the bus cycle.

When address pipelining is selected, the address (BE0#–BE3#, A2–A31) and definition (W/R#, D/C# and M/IO#) of the next cycle are available before the end of the current cycle. To signal their availability, the Intel386 DX address status output (ADS#) is also asserted. Figure 5-9 illustrates the fastest read cycles with pipelined address timing.

Note from Figure 5-9 the fastest bus cycles using pipelined address require only two bus states, named **T1P** and **T2P**. Therefore cycles with pipelined address timing allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased compared to that of a non-pipelined cycle.

By increasing the address-to-data access time, pipelined address timing reduces wait state requirements. For example, if one wait state is required with non-pipelined address timing, no wait states would be required with pipelined address.



Fastest pipelined bus cycles consist of T1P and T2P

Figure 5-9. Fastest Read Cycles with Pipelined Address Timing

Pipelined address timing is useful in typical systems having address latches. In those systems, once an address has been latched, pipelined availability of the next address allows decoding circuitry to generate chip selects (and other necessary select signals) in advance, so selected devices are accessed immediately when the next cycle begins. In other words, the decode time for the next cycle can be overlapped with the end of the current cycle.

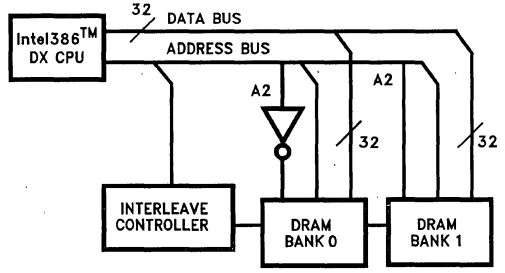
If a system contains a memory structure of two or more interleaved memory banks, pipelined address timing potentially allows even more overlap of activity. This is true when the interleaved memory controller is designed to allow the next memory operation to begin in one memory bank while the current bus cycle is still activating another memory bank. Figure 5-10 shows the general structure of the Intel386 DX with 2-bank and 4-bank interleaved memory. Note each memory bank of the interleaved memory has full data bus width (32-bit data width typically, unless 16-bit bus size is selected).

Further details of pipelined address timing are given in **5.4.3.4 Pipelined Address, 5.4.3.5 Initiating and Maintaining Pipelined Address, 5.4.3.6 Pipelined Address with Dynamic Bus Sizing,** and **5.4.3.7 Maximum Pipelined Address Usage with 16-Bit Bus Size.**

TWO-BANK INTERLEAVED MEMORY

a) Address signal A2 selects bank

b) 32-bit datapath to each bank

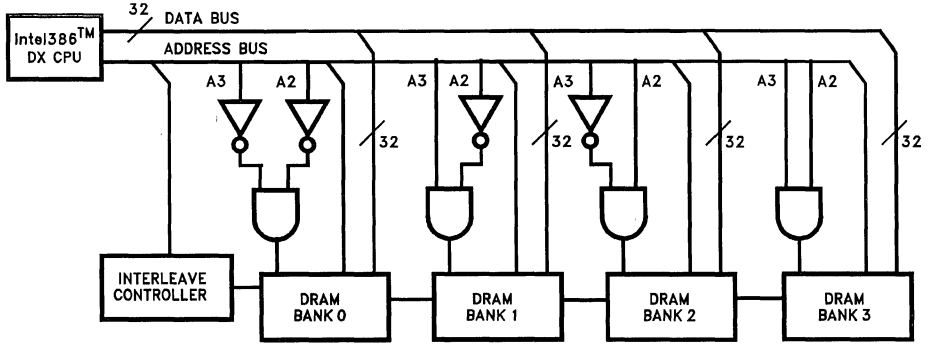


231630-13

FOUR-BANK INTERLEAVED MEMORY

a) Address signals A3 and A2 select bank

b) 32-bit datapath to each bank



231630-14

Figure 5-10. 2-Bank and 4-Bank Interleaved Memory Structure

5.4.3 Read and Write Cycles

5.4.3.1 INTRODUCTION

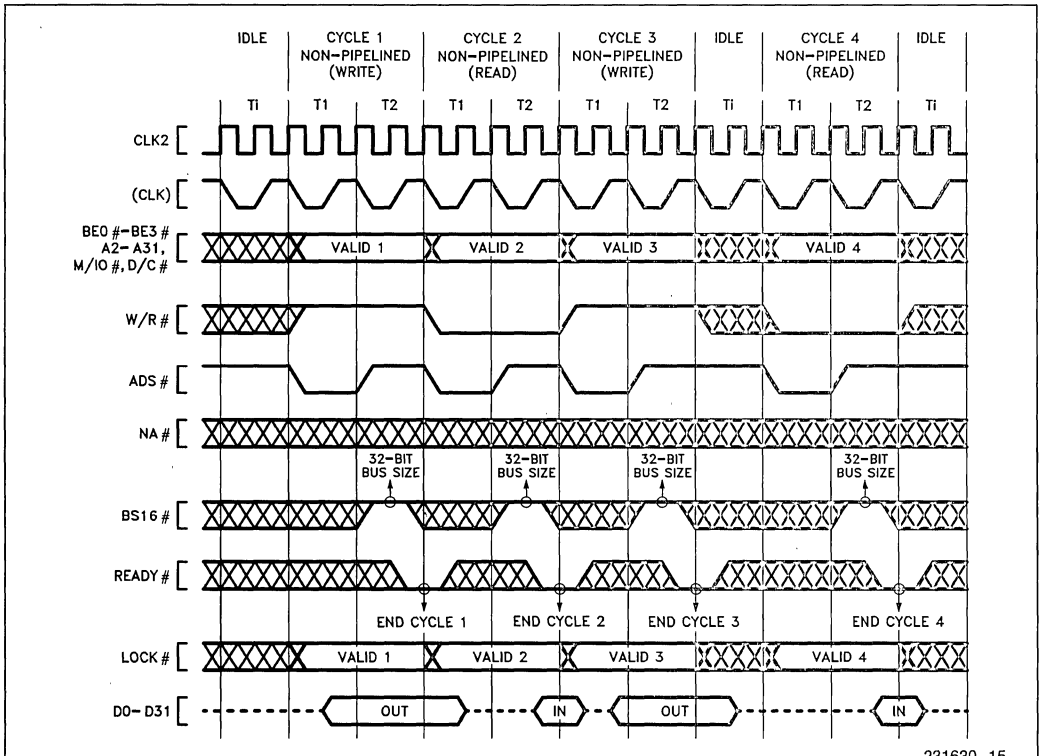
Data transfers occur as a result of bus cycles, classified as read or write cycles. During read cycles, data is transferred from an external device to the processor. During write cycles data is transferred in the other direction, from the processor to an external device.

Two choices of address timing are dynamically selectable: non-pipelined, or pipelined. After a bus idle state, the processor always uses non-pipelined address timing. However, the NA# (Next Address) input may be asserted to select pipelined address timing for the next bus cycle. When pipelining is selected and the Intel386 DX has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by READY#. Generally, the NA# input is sampled each bus cycle to select the desired address timing for the next bus cycle.

Two choices of physical data bus width are dynamically selectable: 32 bits, or 16 bits. Generally, the BS16# (Bus Size 16) input is sampled near the end of the bus cycle to confirm the physical data bus size applicable to the current cycle. Negation of BS16# indicates a 32-bit size, and assertion indicates a 16-bit bus size.

If 16-bit bus size is indicated, the Intel386 DX automatically responds as required to complete the transfer on a 16-bit data bus. Depending on the size and alignment of the operand, another 16-bit bus cycle may be required. Table 5-7 provides all details. When necessary, the Intel386 DX performs an additional 16-bit bus cycle, using D0–D15 in place of D16–D31.

Terminating a read cycle or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the READY# input. Until acknowledged, the processor inserts wait states into the bus cycle, to allow adjustment for the speed of any external device. External hardware, which has decoded the address and bus cycle type asserts the READY# input at the appropriate time.



Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 5-11. Various Bus Cycles and Idle States with Non-Pipelined Address (zero wait states)

At the end of the second bus state within the bus cycle, READY# is sampled. At that time, if external hardware acknowledges the bus cycle by asserting READY#, the bus cycle terminates as shown in Figure 5-11. If READY# is negated as in Figure 5-12, the cycle continues another bus state (a wait state) and READY# is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by READY# asserted.

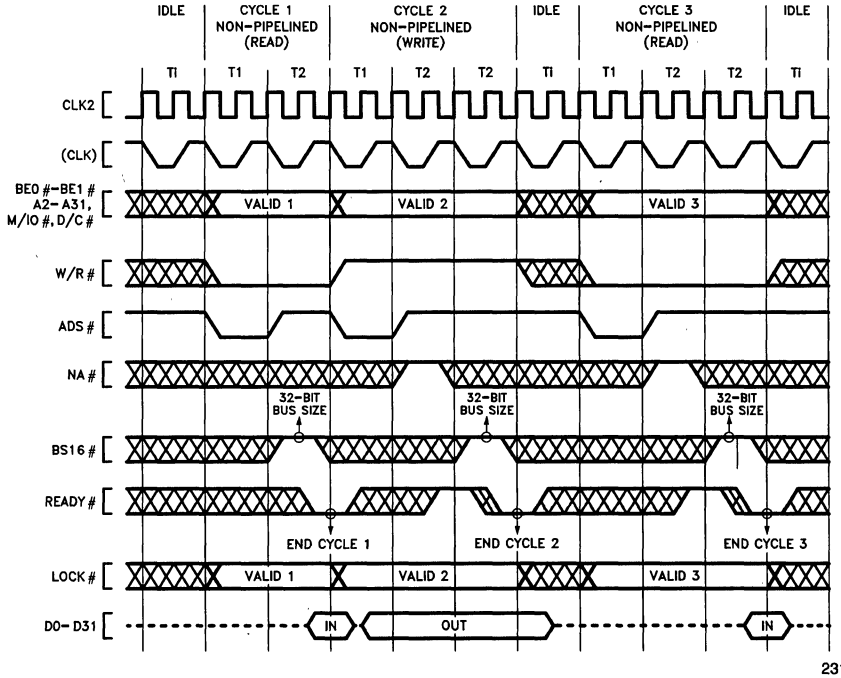
When the current cycle is acknowledged, the Intel386 DX terminates it. When a read cycle is acknowledged, the Intel386 DX latches the information present at its data pins. When a write cycle is acknowledged, the Intel386 DX write data remains valid throughout phase one of the next bus state, to provide write data hold time.

5.4.3.2 NON-PIPELINED ADDRESS

Any bus cycle may be performed with non-pipelined address timing. For example, Figure 5-11 shows a mixture of read and write cycles with non-pipelined address timing. Figure 5-11 shows the fastest possible cycles with non-pipelined address have two bus states per bus cycle. The states are named T1 and T2. In phase one of the T1, the address signals and bus cycle definition signals are driven valid, and to signal their availability, address status (ADS#) is simultaneously asserted.

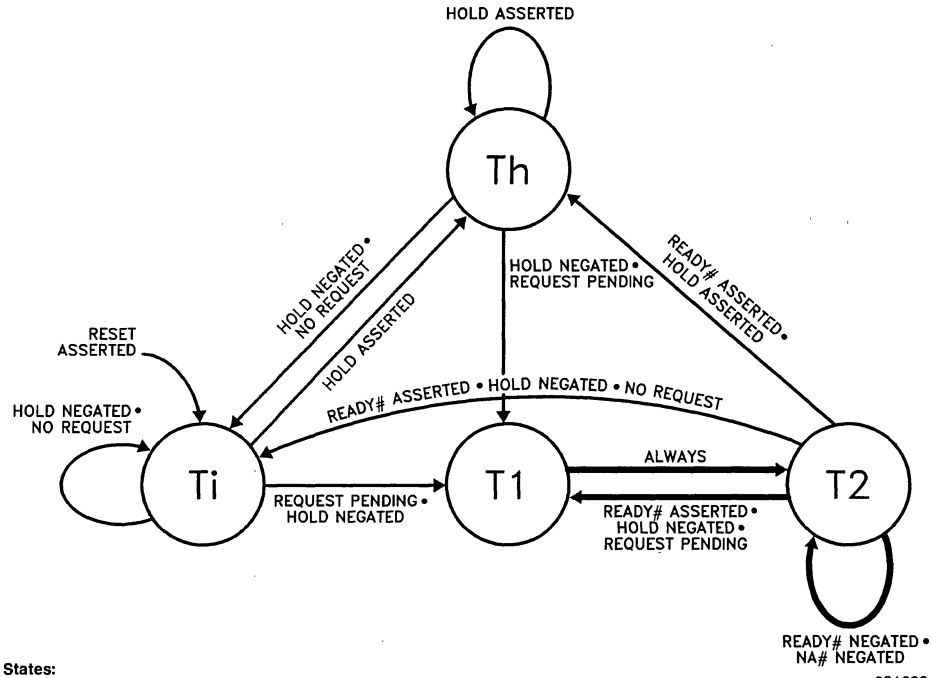
During read or write cycles, the data bus behaves as follows. If the cycle is a read, the Intel386 DX floats its data signals to allow driving by the external device being addressed. **The Intel386 DX requires that all data bus pins be at a valid logic state (high or low) at the end of each read cycle, when READY# is asserted, even if all byte enables are not asserted. The system MUST be designed to meet this requirement.** If the cycle is a write, data signals are driven by the Intel386 DX beginning in phase two of T1 until phase one of the bus state following cycle acknowledgment.

Figure 5-12 illustrates non-pipelined bus cycles with one wait added to cycles 2 and 3. READY# is sampled negated at the end of the first T2 in cycles 2 and 3. Therefore cycles 2 and 3 have T2 repeated. At the end of the second T2, READY# is sampled asserted.



Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 5-12. Various Bus Cycles and Idle States with Non-Pipelined Address (various number of wait states)



Bus States:
T1—first clock of a non-pipelined bus cycle (Intel386 DX drives new address and asserts ADS#)
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle
Ti— idle state
Th—hold acknowledge state (Intel386 DX asserts HLDA)
The fastest bus cycle consists of two states: T1 and T2.
Four basic bus states describe bus operation when not using pipelined address. These states do include BS16# usage for 32-bit and 16-bit bus size. If asserting BS16# requires a second 16-bit bus cycle to be performed, it is performed before HOLD asserted is acknowledged.

Figure 5-13. Intel386™ DX Bus States (not using pipelined address)

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states. When wait states are added and you desire to maintain non-pipelined address timing, it is necessary to negate NA# during each T2 state except the last one, as shown in Figure 5-12 cycles 2 and 3. If NA# is sampled asserted during a T2 other than the last one, the next state would be T2I (for pipelined address) or T2P (for pipelined address) instead of another T2 (for non-pipelined address).

When address pipelining is not used, the bus states and transitions are completely illustrated by Figure 5-13. The bus transitions between four possible states: T1, T2, Ti, and Th. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise, the bus may be idle, in the Ti state, or in hold acknowledge, the Th state.

When address pipelining is not used, the bus state diagram is as shown in Figure 5-13. When the bus is idle it is in state Ti. Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and NA# is negated, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle if a bus request is pending internally, or Ti if there is no bus request pending, or Th if the HOLD input is being asserted.

The bus state diagram in Figure 5-13 also applies to the use of BS16#. If the Intel386 DX makes internal adjustments for 16-bit bus size, the adjustments do not affect the external bus states. If an additional 16-bit bus cycle is required to complete a transfer on a 16-bit bus, it also follows the state transitions shown in Figure 5-13.

Use of pipelined address allows the Intel386 DX to enter three additional bus states not shown in Figure 5-13. Figure 5-20 in **5.4.3.4 Pipelined Address** is the complete bus state diagram, including pipelined address cycles.

5.4.3.3 NON-PIPELINED ADDRESS WITH DYNAMIC DATA BUS SIZING

The physical data bus width for any non-pipelined bus cycle can be either 32-bits or 16-bits. At the beginning of the bus cycle, the processor behaves as if the data bus is 32-bits wide. When the bus cycle is acknowledged, by asserting READY# at the end of a T2 state, the most recent sampling of BS16# determines the data bus size for the cycle being acknowledged. If BS16# was most recently negated, the physical data bus size is defined as 32 bits. If BS16# was most recently asserted, the size is defined as 16 bits.

When BS16# is asserted and two 16-bit bus cycles are required to complete the transfer, BS16# must be asserted during the second cycle; 16-bit bus size is not assumed. Like any bus cycle, the second 16-bit cycle must be acknowledged by asserting READY#.

When a second 16-bit bus cycle is required to complete the transfer over a 16-bit bus, the addresses

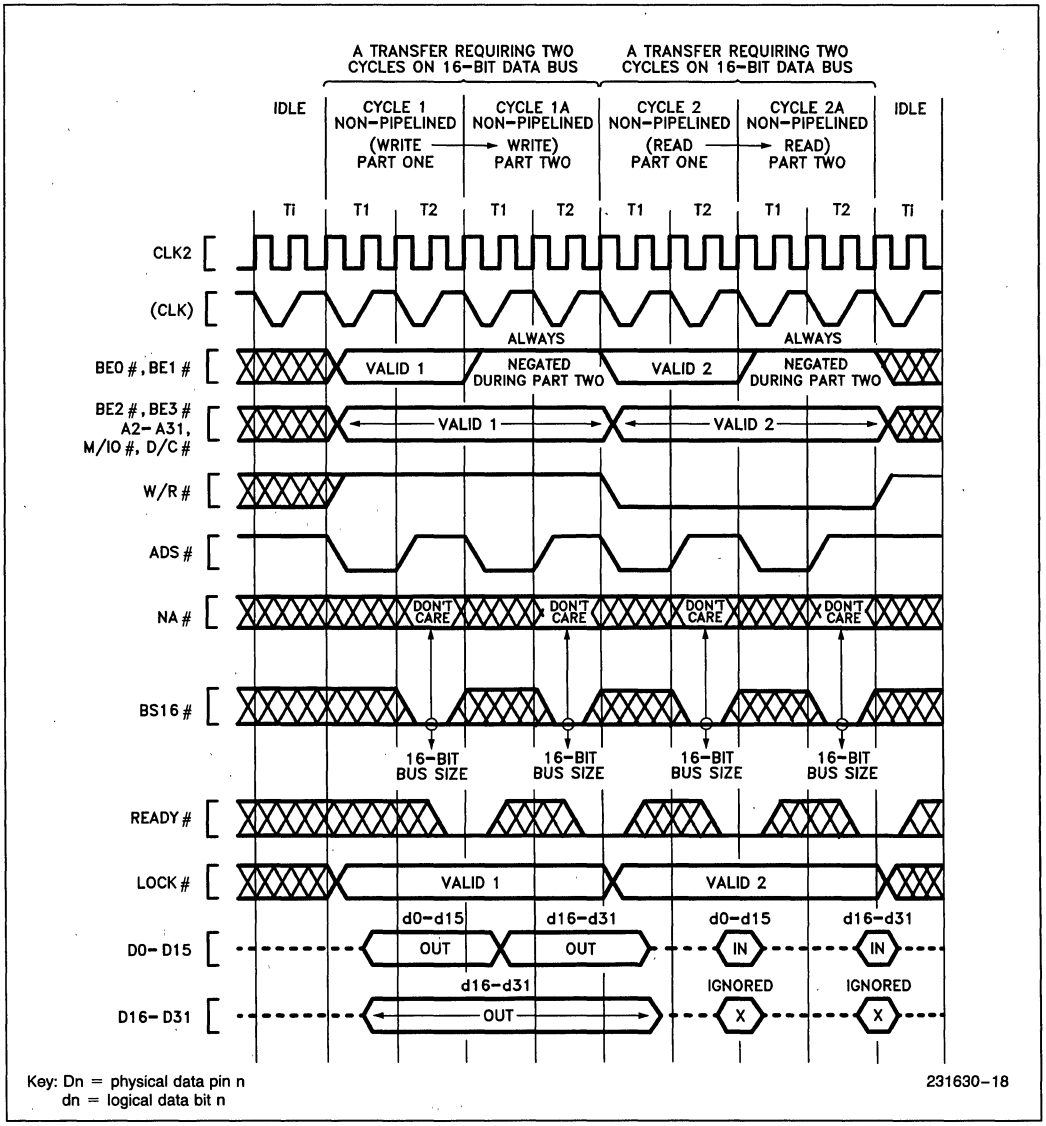


Figure 5-14. Asserting BS16# (zero wait states, non-pipelined address)

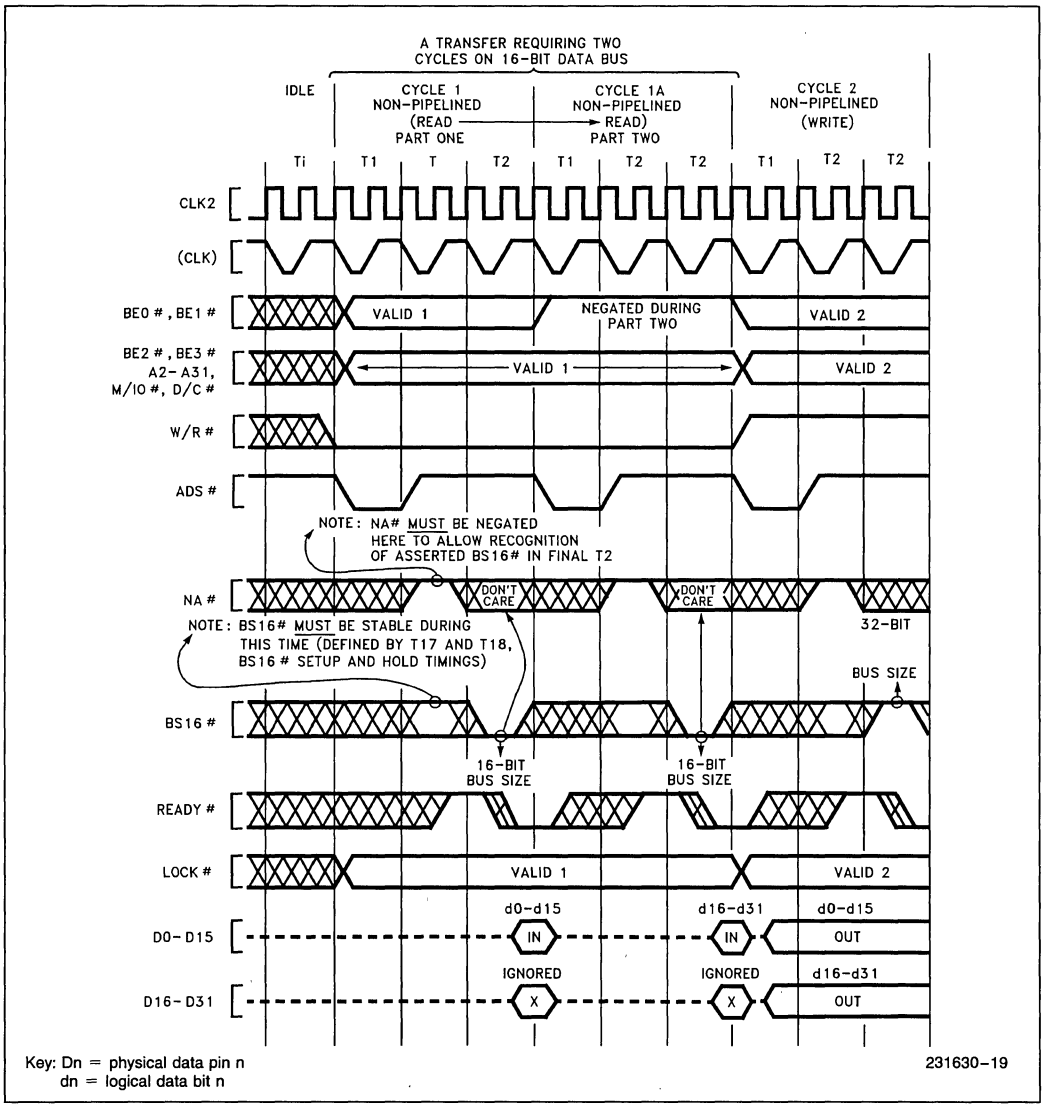


Figure 5-15. Asserting BS16# (one wait state, non-pipelined address)

generated for the two 16-bit bus cycles are closely related to each other. The addresses are the same except BE0# and BE1# are always negated for the second cycle. This is because data on D0–D15 was already transferred during the first 16-bit cycle.

Figures 5-14 and 5-15 show cases where assertion of BS16# requires a second 16-bit cycle for complete operand transfer. Figure 5-14 illustrates cycles without wait states. Figure 5-15 illustrates cycles with one wait state. In Figure 5-15 cycle 1, the bus cycle during which BS16# is asserted, note that NA# must be negated in the T2 state(s) prior to the last T2 state. This is to allow the recognition of BS16# asserted in the final T2 state. Also note that during this state BS16# must be stable (defined by t17 and t18, BS16# setup and hold timings), in order to prevent potential data corruption during split cycle reads. The logic state of BS16# during this time is not important. The relation of NA# and BS16# is given fully in **5.4.3.4 Pipelined Address**, but Figure 5-15 illustrates these precautions you need to know when using BS16# with non-pipelined address.

5.4.3.4 PIPELINED ADDRESS

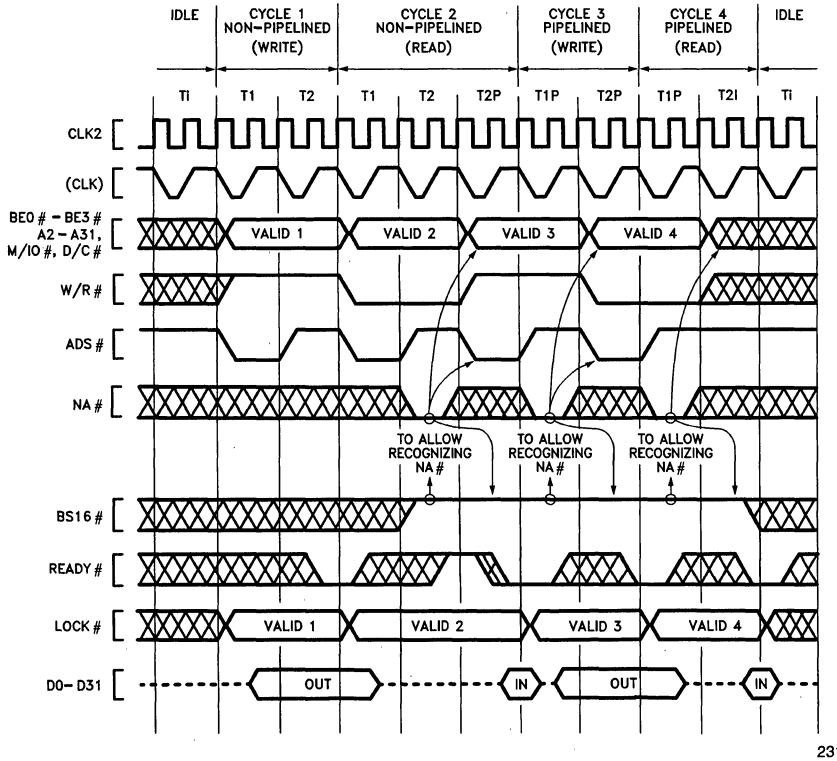
Address pipelining is the option of requesting the address and the bus cycle definition of the next, internally pending bus cycle before the current bus cycle is acknowledged with READY# asserted. ADS# is asserted by the Intel386 DX when the next address is issued. The address pipelining option is controlled on a cycle-by-cycle basis with the NA# input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles, therefore, NA# is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 5-16, during which NA# is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).

If NA# is sampled asserted, the Intel386 DX is free to drive the address and bus cycle definition of the next bus cycle, and assert ADS#, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of address pipelining, the Intel386 DX has the following characteristics:

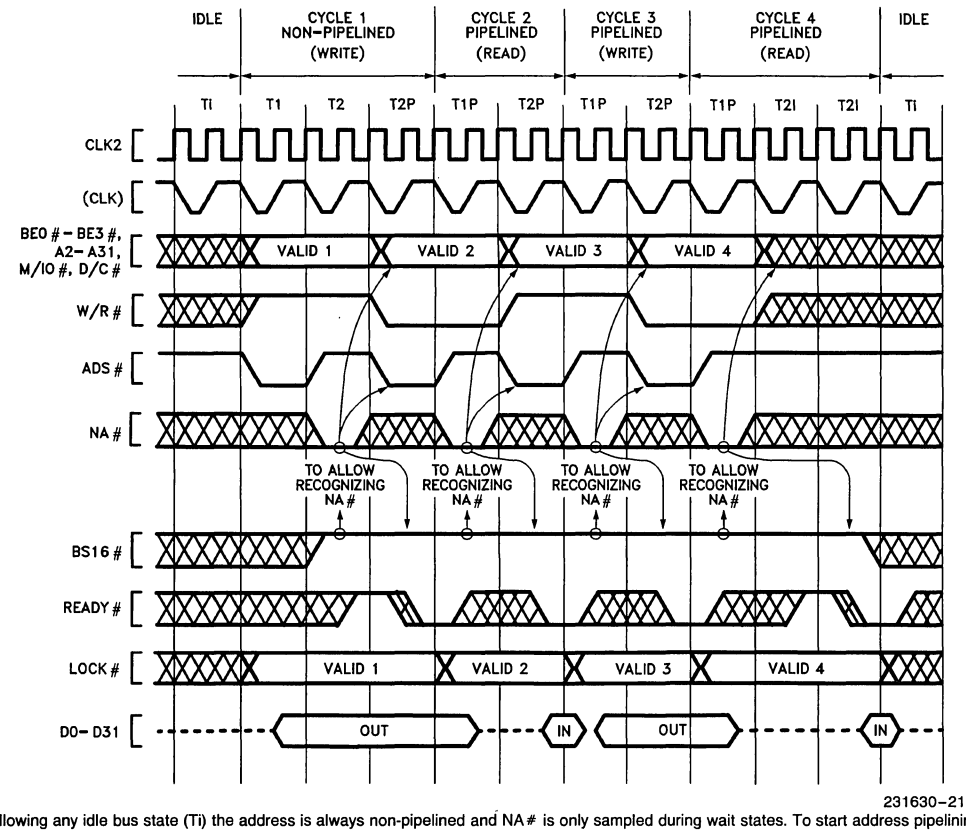
1) For NA# to be sampled asserted, BS16# must be negated at that sampling window (see Figure 5-16 Cycles 2 through 4, and Figure 5-17 Cycles 1 through 4). If NA# and BS16# are both sampled asserted during the last T2 period of a bus cycle, BS16# asserted has priority. Therefore, if both are asserted, the current bus size is taken to be 16 bits and the next address is not pipelined.



231630-20

Following any idle bus state (Ti), addresses are non-pipelined. Within non-pipelined bus cycles, NA# is only sampled during wait states. Therefore, to begin address pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cycle 2 above).

Figure 5-16. Transitioning to Pipelined Address During Burst of Bus Cycles



Following any idle bus state (Ti) the address is always non-pipelined and NA# is only sampled during wait states. To start address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (cycle 1 above).
The pipelined cycles (2, 3, 4 above) are shown with various numbers of wait states.

Figure 5-17. Fastest Transition to Pipelined Address Following Idle Bus State

2) The next address may appear as early as the bus state after NA# was sampled asserted (see Figures 5-16 or 5-17). In that case, state T2P is entered immediately. However, when there is not an internal bus request already pending, the next address will not be available immediately after NA# is asserted and T2I is entered instead of T2P (see Figure 5-19 Cycle 3). Provided the current bus cycle isn't yet acknowledged by READY# asserted, T2P will be entered as soon as the Intel386 DX does drive the next address. External hardware should therefore observe the ADS# output as confirmation the next address is actually being driven on the bus.

3) Once NA# is sampled asserted, the Intel386 DX commits itself to the highest priority bus request that is pending internally. It can no longer perform another 16-bit transfer to the same address should BS16# be asserted externally, so thereafter must assume the current bus size is 32 bits. Therefore if NA# is sampled asserted within a bus cycle, BS16# must be negated thereafter in that bus cycle (see Figures 5-16, 5-17, 5-19). Consequently, do not assert NA# during bus cycles which must have BS16# driven asserted. See **5.4.3.6 Dynamic Bus Sizing with Pipelined Address.**

4) Any address which is validated by a pulse on the Intel386 DX ADS# output will remain stable on the address pins for at least two processor clock periods. The Intel386 DX cannot produce a new address more frequently than every two processor clock periods (see Figures 5-16, 5-17, 5-19).

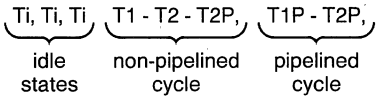
5) Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 5-19 Cycle 1).

The complete bus state transition diagram, including operation with pipelined address is given by 5-20. Note it is a superset of the diagram for non-pipelined address only, and the three additional bus states for pipelined address are drawn in bold.

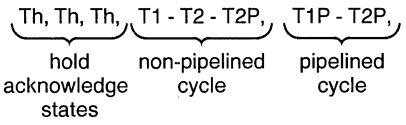
The fastest bus cycle with pipelined address consists of just two bus states, T1P and T2P (recall for non-pipelined address it is T1 and T2). T1P is the first bus state of a pipelined cycle.

5.4.3.5 INITIATING AND MAINTAINING PIPELINED ADDRESS

Using the state diagram Figure 5-20, observe the transitions from an idle state, Ti, to the beginning of a pipelined bus cycle, T1P. From an idle state Ti, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided NA# is asserted and the first bus cycle ends in a T2P state (the address for the next bus cycle is driven during T2P). The fastest path from an idle state to a bus cycle with pipelined address is shown in bold below:



T1-T2-T2P are the states of the bus cycle that establishes address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:



The transition to pipelined address is shown functionally by Figure 5-17 Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3 and 4, which are pipelined. The NA# input is asserted at the appropriate time to select address pipelining for Cycles 2, 3 and 4.

Once a bus cycle is in progress and the current address has become valid, the NA# input is sampled at the end of every phase one, beginning with the next bus state, until the bus cycle is acknowledged. During Figure 5-17 Cycle 1 therefore, sampling begins in T2. Once NA# is sampled asserted during the current cycle, the Intel386 DX is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 5-16 Cycle 1 for example, the next address is driven during state T2P. Thus Cycle 1 makes the transition to pipelined address timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as READY# asserted terminates Cycle 1.

Example transition bus cycles are Figure 5-17 Cycle 1 and Figure 5-16 Cycle 2. Figure 5-17 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 5-16 Cycle 2 shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 (you assert NA# at that time), and T2P (provided the Intel386 DX has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note three states (T1, T2 and T2P) are only required in a bus cycle performing a **transition** from non-pipelined address into pipelined address timing, for example Figure 5-17 Cycle 1. Figure 5-17 Cycles 2, 3 and 4 show that address pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting NA# and detecting that the Intel386 DX enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of ADS#. Figures 5-16 and 5-17 however, each show pipelining ending after Cycle 4 because Cycle 4 ends in T2I. This indicates the Intel386 DX didn't have an internal bus request prior to the acknowledgement of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

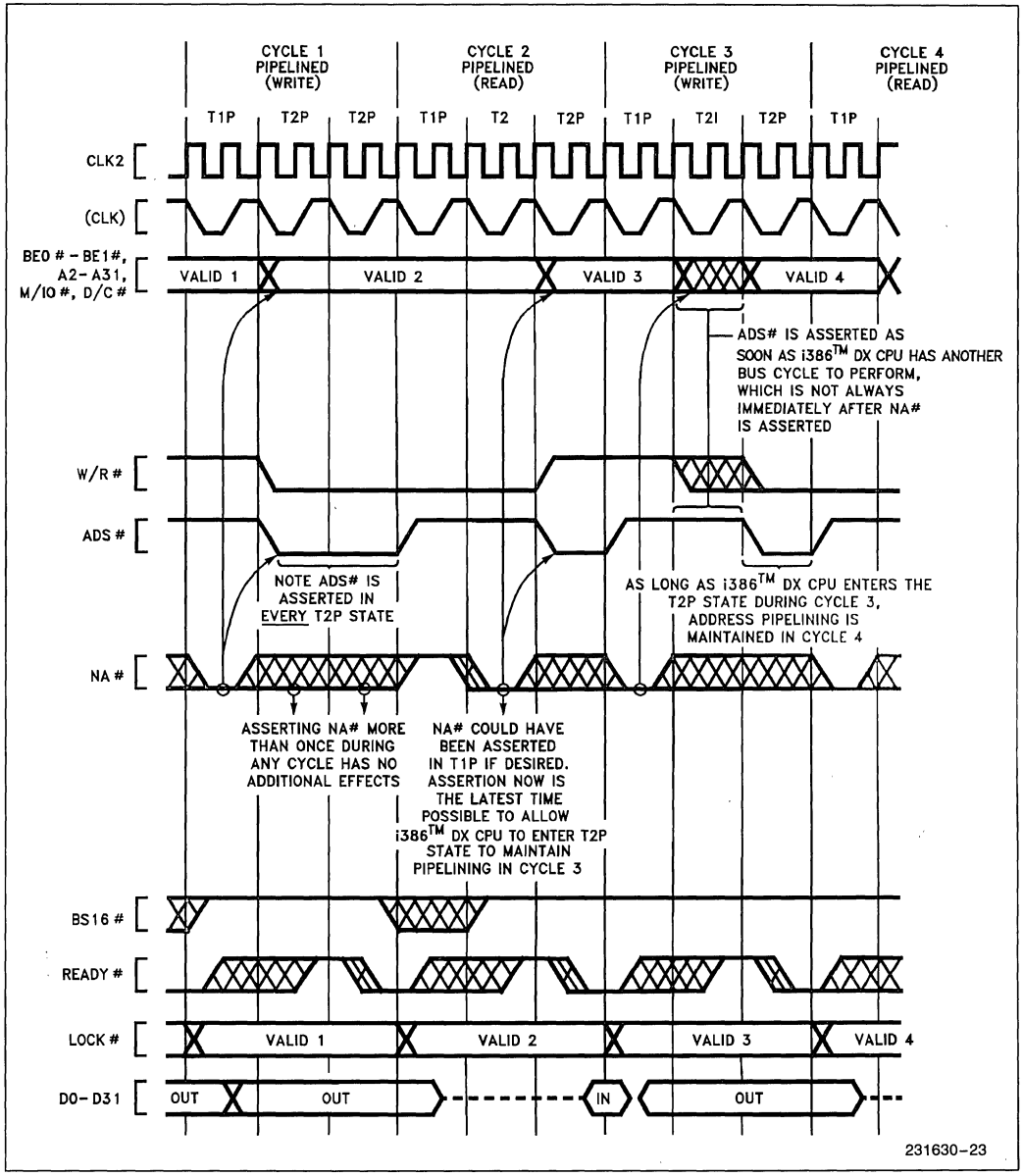


Figure 5-19. Details of Address Pipelining During Cycles with Wait States

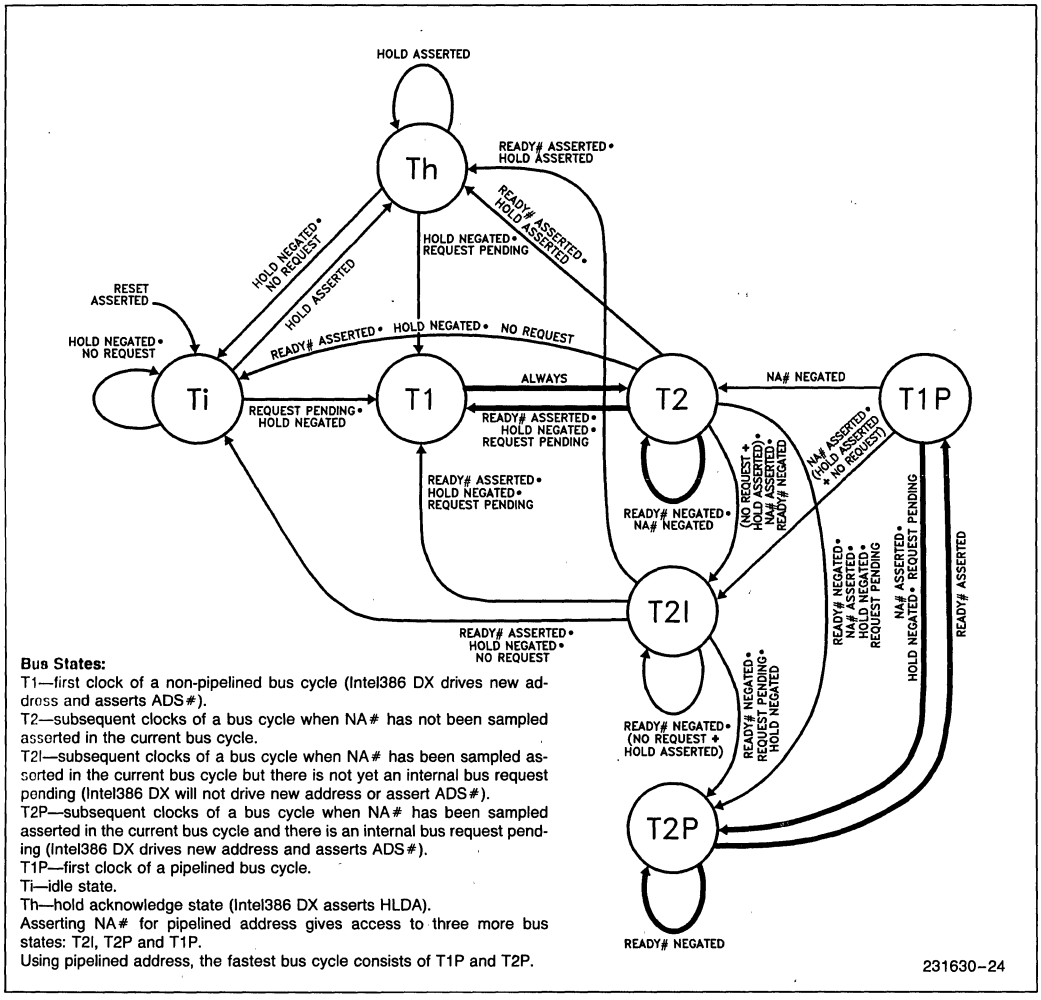


Figure 5-20. Intel386™ DX Complete Bus States (including pipelined address)

Realistically, address pipelining is almost always maintained as long as NA# is sampled asserted. This is so because in the absence of any other request, a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore address pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD negated) and NA# is sampled asserted in each of the bus cycles.

5.4.3.6 PIPELINED ADDRESS WITH DYNAMIC DATA BUS SIZING

The BS16# feature allows easy interface to 16-bit data buses. When asserted, the Intel386 DX bus interface hardware performs appropriate action to make the transfer using a 16-bit data bus connected on D0–D15.

There is a degree of interaction, however, between the use of Address Pipelining and the use of Bus Size 16. The interaction results from the multiple bus cycles required when transferring 32-bit operands over a 16-bit bus. If the operand requires both 16-bit halves of the 32-bit bus, the appropriate Intel386 DX action is a second bus cycle to complete the operand's transfer. It is this necessity that conflicts with NA# usage.

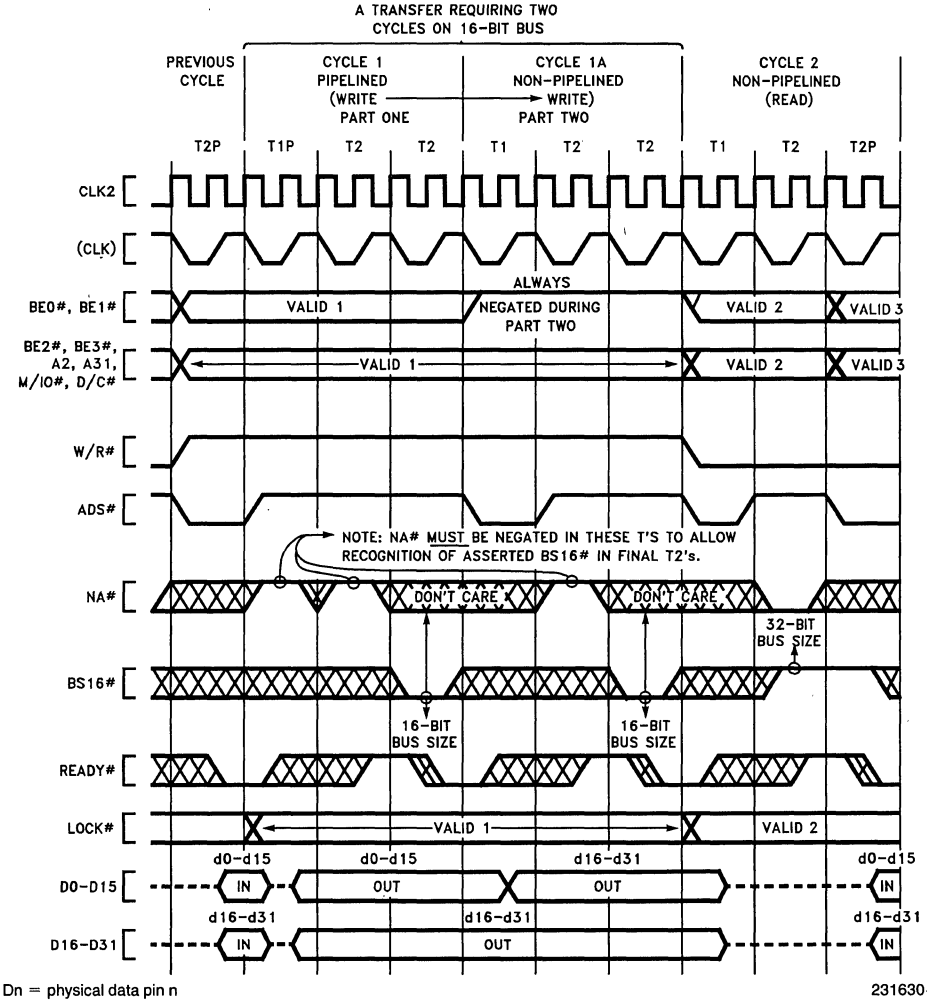
When NA# is sampled asserted, the Intel386 DX commits itself to perform the next inter-

nally pending bus request, and is allowed to drive the next internally pending address onto the bus. Asserting NA# therefore makes it impossible for the next bus cycle to again access the current address on A2–A31, such as may be required when BS16# is asserted by the external hardware.

To avoid conflict, the Intel386 DX is designed with following two provisions:

1) To avoid conflict, BS16# must be negated in the current bus cycle if NA# has already been sampled asserted in the current cycle. If NA# is sampled asserted, the current data bus size is assumed to be 32 bits.

2) To also avoid conflict, if NA# and BS16# are both asserted during the same sampling window, BS16# asserted has priority and the Intel386 DX acts as if NA# was negated at that time. Internal Intel386 DX circuitry, shown conceptually in Figure 5-18, assures that BS16# is sampled asserted and NA# is sampled negated if both inputs are externally asserted at the same sampling window.



Key: Dn = physical data pin n
dn = logical data bit n

231630-25

Cycle 1 is pipelined. Cycle 1a cannot be pipelined, but its address can be inferred from that of Cycle 1, to externally simulate address pipelining during Cycle 1a.

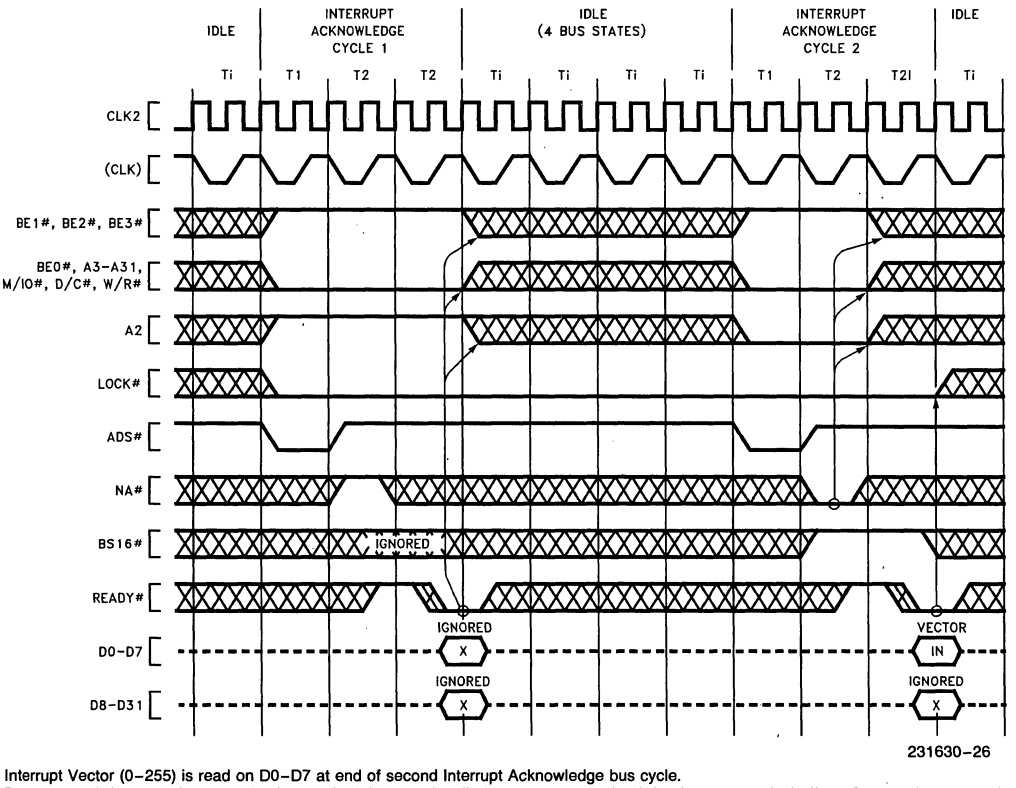
Figure 5-21. Using NA# and BS16#

Certain types of 16-bit or 8-bit operands require no adjustment for correct transfer on a 16-bit bus. Those are read or write operands using only the lower half of the data bus, and write operands using only the upper half of the bus since the Intel386 DX simultaneously duplicates the write data on the lower half of the data bus. For these patterns of Byte Enables and the R/W# signals, BS16# need not be asserted at the Intel386 DX allowing NA# to be asserted during the bus cycle if desired.

5.4.4 Interrupt Acknowledge (INTA) Cycles

In response to an interrupt request on the INTR input when interrupts are enabled, the Intel386 DX performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by READY# sampled asserted.

The state of A2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A31–A3 low, A2 high, BE3#–BE1# high, and BE0# low). The address driven during the second interrupt acknowledge cycle is 0 (A31–A2 low, BE3#–BE1# high, BE0# low).



Interrupt Vector (0–255) is read on D0–D7 at end of second Interrupt Acknowledge bus cycle.
Because each Interrupt Acknowledge bus cycle is followed by idle bus states, asserting NA# has no practical effect. Choose the approach which is simplest for your system hardware design.

Figure 5-22. Interrupt Acknowledge Cycles

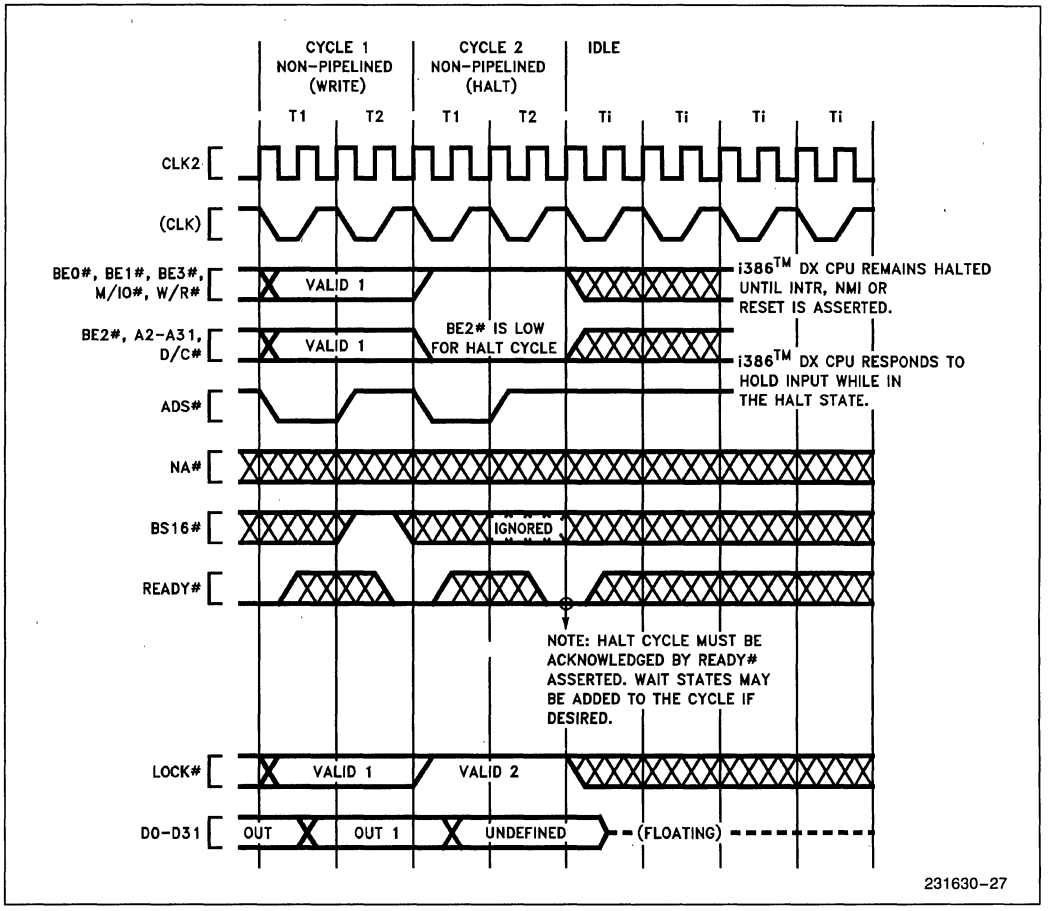


Figure 5-23. Halt Indication Cycle

The LOCK# output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states, Ti, are inserted by the Intel386 DX between the two interrupt acknowledge cycles, allowing for compatibility with spec TRHRL of the 8259A Interrupt Controller.

During both interrupt acknowledge cycles, D0–D31 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the Intel386 DX will read an external interrupt vector from D0–D7 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.

5.4.5 Halt Indication Cycle

The Intel386 DX halts as a result of executing a HALT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals shown in **5.2.5 Bus Cycle Definition** and a byte address of 2. BE0# and BE2# are the only signals distinguishing halt indication from shutdown indication, which drives an address of 0. During the halt cycle undefined data is driven on D0–D31. The halt indication cycle must be acknowledged by READY# asserted.

A halted Intel386 DX resumes execution when INTR (if interrupts are enabled) or NMI or RESET is asserted.

5.4.6 Shutdown Indication Cycle

The Intel386 DX shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown in **5.2.5 Bus Cycle Definition** and a byte address of 0. BE0# and BE2# are the only signals distinguishing shutdown indication from halt indication, which drives an address of 2. During the shutdown cycle undefined data is driven on D0–D31. The shutdown indication cycle must be acknowledged by READY# asserted.

A shutdown Intel386 DX resumes execution when NMI or RESET is asserted.

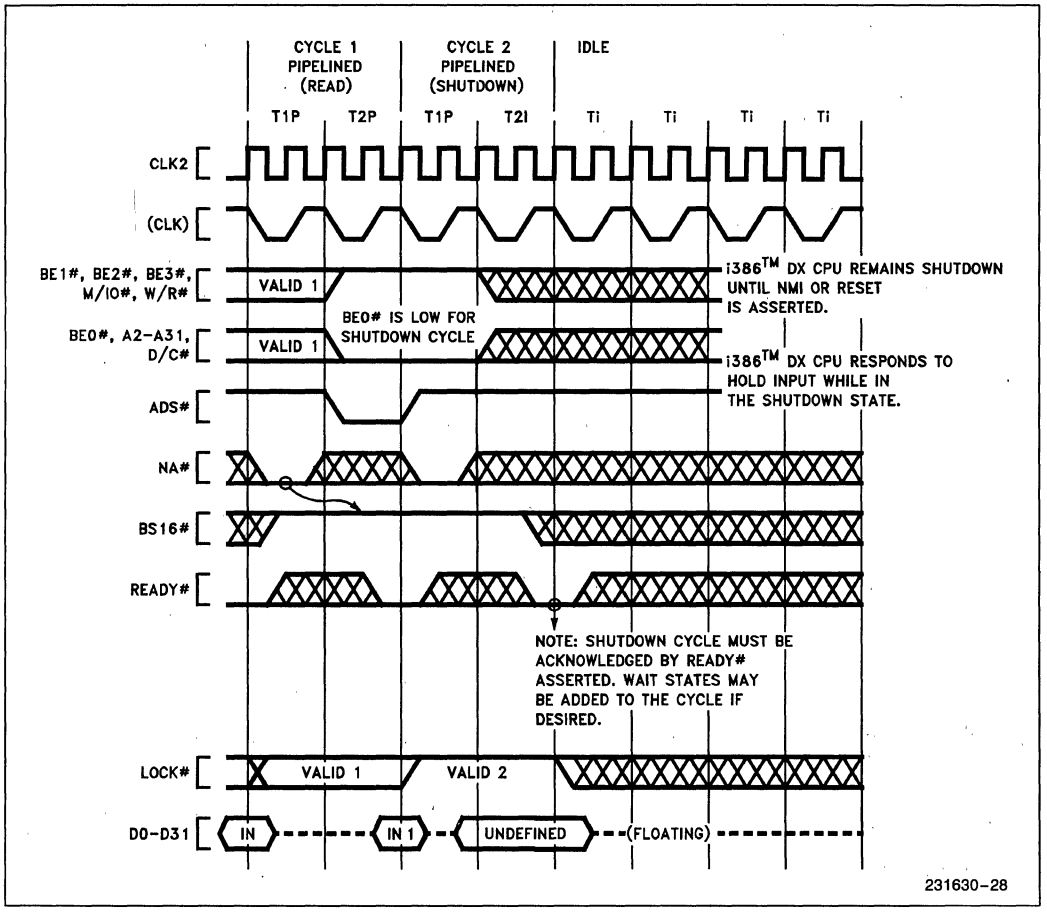
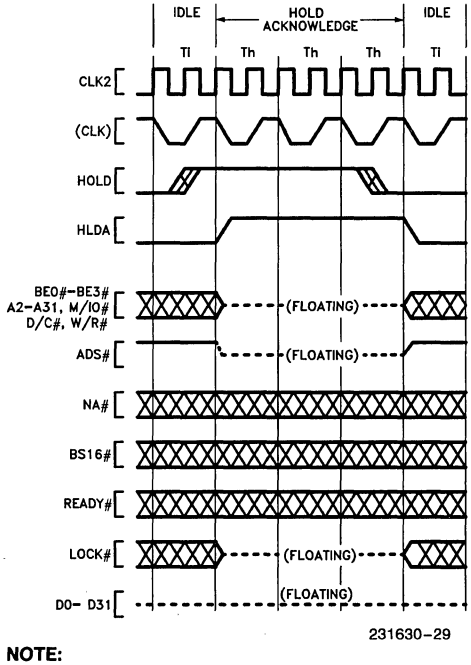


Figure 5-24. Shutdown Indication Cycle

5.5 OTHER FUNCTIONAL DESCRIPTIONS

5.5.1 Entering and Exiting Hold Acknowledge

The bus hold acknowledge state, Th, is entered in response to the HOLD input being asserted. In the bus hold acknowledge state, the Intel386 DX floats all output or bidirectional signals, except for HLDA. HLDA is asserted as long as the Intel386 DX remains in the bus hold acknowledge state. In the bus hold acknowledge state, all inputs except HOLD, RESET, BUSY#, ERROR#, and PEREQ are ignored (also up to one rising edge on NMI is remembered for processing when HOLD is no longer asserted).



NOTE:
For maximum design flexibility the Intel386 DX has no internal pullup resistors on its outputs. Your design may require an external pullup on ADS# and other Intel386 DX outputs to keep them negated during float periods.

Figure 5-25. Requesting Hold from Idle Bus

Th may be entered from a bus idle state as in Figure 5-25 or after the acknowledgement of the current physical bus cycle if the LOCK# signal is not asserted, as in Figures 5-26 and 5-27. If HOLD is asserted during a locked bus cycle, the Intel386 DX may execute one unlocked bus cycle before acknowledging HOLD. If asserting BS16# requires a second 16-bit bus cycle to complete a physical operand transfer, it is performed before HOLD is acknowledged, although the bus state diagrams in Figures 5-13 and 5-20 do not indicate that detail.

Th is exited in response to the HOLD input being negated. The following state will be Ti as in Figure 5-25 if no bus request is pending. The following bus state will be T1 if a bus request is internally pending, as in Figures 5-26 and 5-27.

Th is also exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in Th, the event is remembered as a nonmaskable interrupt 2 and is serviced when Th is exited, unless of course, the Intel386 DX is reset before Th is exited.

5.5.2 Reset During Hold Acknowledge

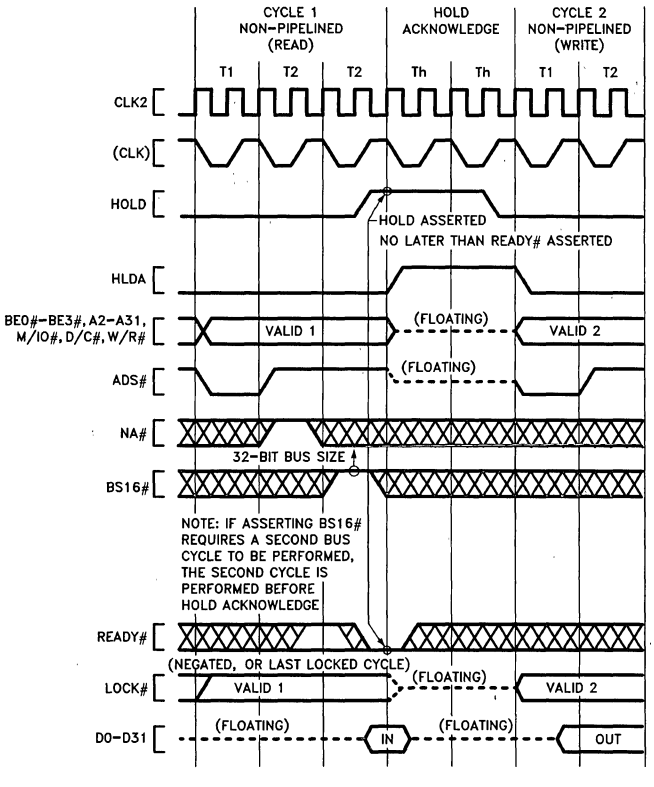
RESET being asserted takes priority over HOLD being asserted. Therefore, Th is exited in reponse to the RESET input being asserted. If RESET is asserted while HOLD remains asserted, the Intel386 DX drives its pins to defined states during reset, as in **Table 5-3 Pin State During Reset**, and performs internal reset activity as usual.

If HOLD remains asserted when RESET is negated, the Intel386 DX enters the hold acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the Intel386 DX would otherwise perform its first bus cycle. If HOLD remains asserted when RESET is negated, the BUSY# input is still sampled as usual to determine whether a self test is being requested, and ERROR# is still sampled as usual to determine whether a Intel387 DX coprocessor vs. an 80287 (or none) is present.

5.5.3 Bus Activity During and Following Reset

RESET is the highest priority input signal, capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage, or idle states or bus hold acknowledge states discontinued so that the reset state is established.

RESET should remain asserted for at least 15 CLK2 periods to ensure it is recognized throughout the Intel386 DX, and at least 80 CLK2 periods if Intel386 DX self-test is going to be requested at the falling edge. RESET asserted pulses less than 15 CLK2 periods may not be recognized. RESET pulses less than 80 CLK2 periods followed by a self-test may



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 5-26. Requesting Hold from Active Bus (NA# negated)

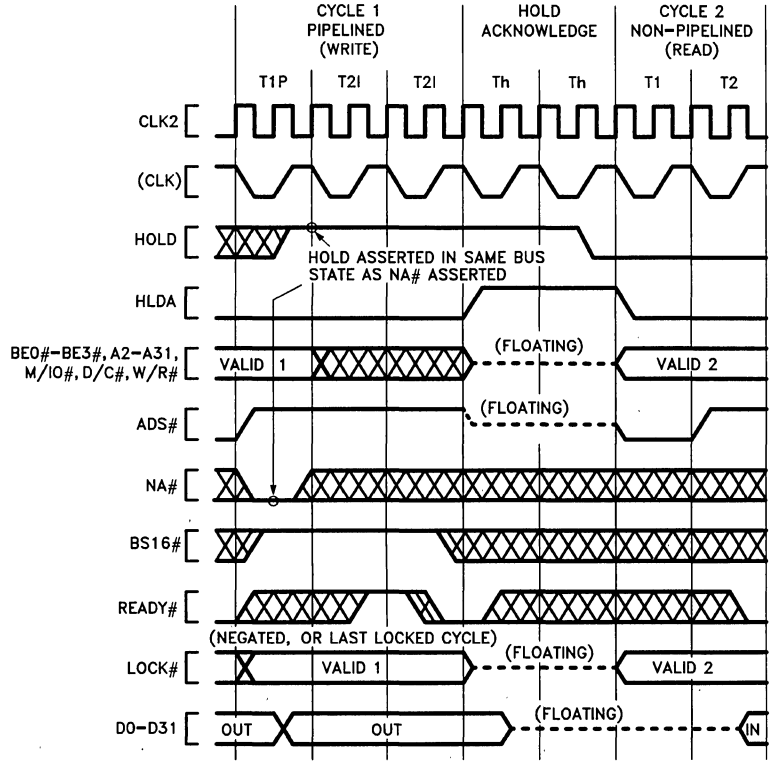
cause the self-test to report a failure when no true failure exists. The additional RESET pulse width is required to clear additional state prior to a valid self-test.

Provided the RESET falling edge meets setup and hold times t_{25} and t_{26}, the internal processor clock phase is defined at that time, as illustrated by Figure 5-28 and Figure 7-7.

A Intel386 DX self-test may be requested at the time RESET is negated by having the BUSY# input at a LOW level, as shown in Figure 5-28. The self-test requires (2^{20}) + approximately 60 CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the Intel386 DX attempts to proceed with the reset sequence afterwards.

After the RESET falling edge (and after the self-test if it was requested) the Intel386 DX performs an internal initialization sequence for approximately 350 to 450 CLK2 periods.

The Intel386 DX samples its ERROR# input some time after the falling edge of RESET and before executing the first ESC instruction. During this sampling period BUSY# must be HIGH. If ERROR# was sampled active, the Intel386 DX employs the 32-bit protocol of the Intel387 DX. Even though this protocol was selected, it is still necessary to use a software recognition test to determine the presence or identity of the coprocessor and to assure compatibility with future processors. (See Chapter 11 of the Intel386 DX Programmer's Reference Manual, Order #230985-002).



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 5-27. Requesting Hold from Active Bus (NA# asserted)

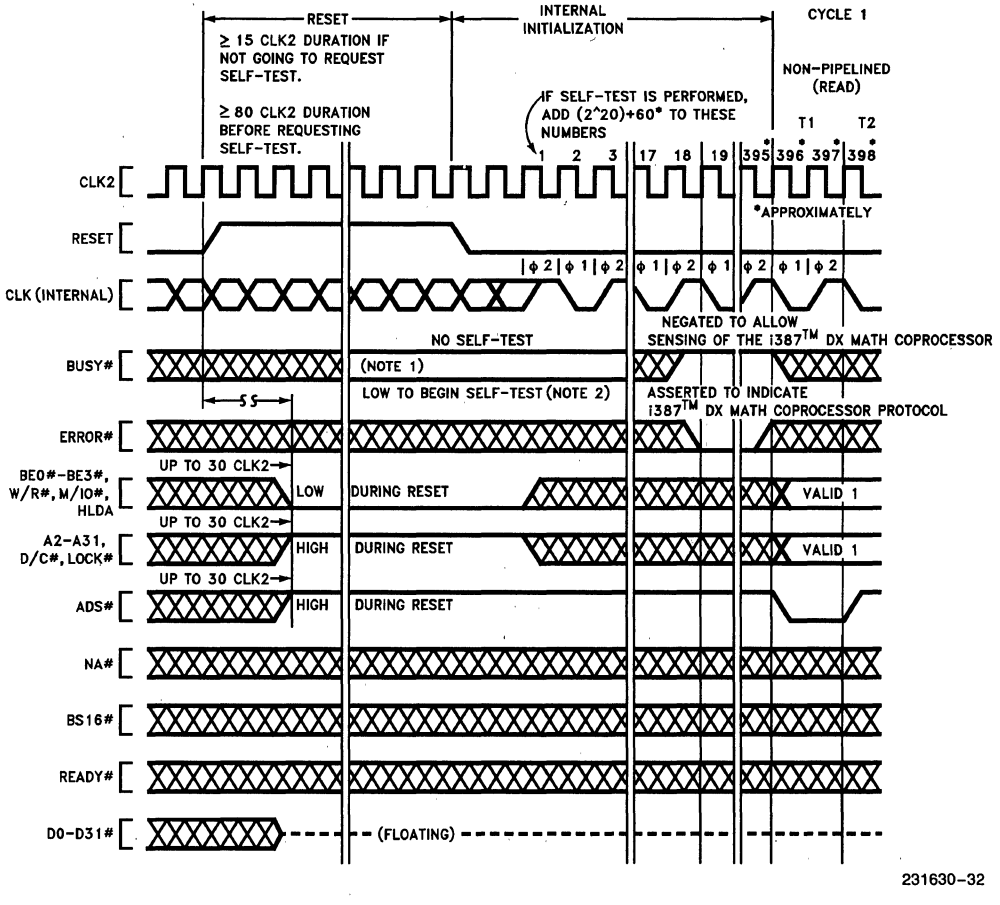
5.6 SELF-TEST SIGNATURE

Upon completion of self-test, (if self-test was requested by holding BUSY# LOW at least eight CLK2 periods before and after the falling edge of RESET), the EAX register will contain a signature of 00000000h indicating the Intel386 DX passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000h, applies to all Intel386 DX revision levels. Any non-zero signature indicates the Intel386 DX unit is faulty.

5.7 COMPONENT AND REVISION IDENTIFIERS

To assist Intel386 DX users, the Intel386 DX after reset holds a component identifier and a revision identifier in its DX register. The upper 8 bits of DX hold 03h as identification of the Intel386 DX component. The lower 8 bits of DX hold an 8-bit unsigned binary number related to the component revision level. The revision identifier begins chronologically with a value zero and is subject to change (typically it will be incremented) with component steppings intended to have certain improvements or distinctions from previous steppings.

These features are intended to assist Intel386 DX users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision, or to follow a completely uniform numerical sequence, depending on the type or intention of revision, or manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component.



NOTES:
1. BUSY# should be held stable for 8 CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested, the Intel386 DX outputs remain in their reset state as shown here and in Table 5-3.

Figure 5-28. Bus Activity from Reset Until First Code Fetch

Table 5-10. Component and Revision Identifier History

Intel386™ DX Stepping Name	Component Identifier	Revision Identifier	Intel386 DX Stepping Name	Component Identifier	Revision Identifier
B0	03	03	D0	03	05
B1	03	03	D1	03	08

5.8 COPROCESSOR INTERFACING

The Intel386 DX provides an automatic interface for the Intel Intel387 DX numeric floating-point coprocessor. The Intel387 DX coprocessor uses an I/O-mapped interface driven automatically by the Intel386 DX and assisted by three dedicated signals: BUSY#, ERROR#, and PEREQ.

As the Intel386 DX begins supporting a coprocessor instruction, it tests the BUSY# and ERROR# signals to determine if the coprocessor can accept its next instruction. Thus, the BUSY# and ERROR# inputs eliminate the need for any "preamble" bus cycles for communication between processor and coprocessor. The Intel387 DX can be given its command opcode immediately. The dedicated signals provide instruction synchronization, and eliminate the need of using the Intel386 DX WAIT opcode (9Bh) for Intel387 DX coprocessor instruction synchronization (the WAIT opcode was required when 8086 or 8088 was used with the 8087 coprocessor).

Custom coprocessors can be included in Intel386 DX-based systems, via memory-mapped or I/O-mapped interfaces. Such coprocessor interfaces allow a completely custom protocol, and are not limited to a set of coprocessor protocol "primitives". Instead, memory-mapped or I/O-mapped interfaces may use all applicable Intel386 DX instructions for high-speed coprocessor communication. The BUSY# and ERROR# inputs of the Intel386 DX may also be used for the custom coprocessor interface, if such hardware assist is desired. These signals can be tested by the Intel386 DX WAIT opcode (9Bh). The WAIT instruction will wait until the BUSY# input is negated (interruptable by an NMI or enabled INTR input), but generates an exception 16 fault if the ERROR# pin is in the asserted state when the BUSY# goes (or is) negated. If the custom coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the Intel386 DX on-chip paging or segmentation mechanisms. If the custom interface is I/O-mapped, protection of the interface can be provided with the Intel386 DX IOPL (I/O Privilege Level) mechanism.

The Intel387 DX numeric coprocessor interface is I/O mapped as shown in Table 5-11. Note that the Intel387 DX coprocessor interface addresses are beyond the 0h-FFFFh range for programmed I/O. When the Intel386 DX supports the Intel387 DX coprocessor, the Intel386 DX automatically generates bus cycles to the coprocessor interface addresses.

Table 5-11. Numeric Coprocessor Port Addresses

Address in Intel386™ DX I/O Space	Intel387™ DX Coprocessor Register
800000F8h	Opcode Register (32-bit port)
800000FCh	Operand Register (32-bit port)

To correctly map the Intel387 DX coprocessor registers to the appropriate I/O addresses, connect the Intel387 DX coprocessor CMD0# pin directly to the A2 output of the Intel386 DX.

5.8.1 Software Testing for Coprocessor Presence

When software is used to test for coprocessor (Intel387 DX) presence, it should use only the following coprocessor opcodes: FINIT, FNINIT, FSTCW mem, FSTSW mem, FSTSW AX. To use other coprocessor opcodes when a coprocessor is known to be not present, first set EM = 1 in Intel386 DX CR0.

6. INSTRUCTION SET

This section describes the Intel386 DX instruction set. A table lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within Intel386 DX instructions.

6.1 Intel386™ DX INSTRUCTION ENCODING AND CLOCK COUNT SUMMARY

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 6-1 below, by the processor clock period (e.g. 50 ns for a 20 MHz Intel386 DX, 40 ns for a 25 MHz Intel386 DX, and 30 ns for a 33 MHz Intel386 DX).

For more detailed information on the encodings of instructions refer to section 6.2 Instruction Encodings. Section 6.2 explains the general structure of instruction encodings, and defines exactly the encodings of all fields contained within the instruction.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor access to the bus.
4. No exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts shown. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.

Instruction Clock Count Notation

1. If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.
2. n = number of times repeated.
3. m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and each of the **other** bytes of the instruction and prefix(es) each count as one component.

Wait States

Add 1 clock per wait state to instruction execution for each data access.

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
GENERAL DATA TRANSFER					
MOV = Move:					
Register to Register/Memory	1000100w mod reg r/m	2/2	2/2	b	h
Register/Memory to Register	1000101w mod reg r/m	2/4	2/4	b	h
Immediate to Register/Memory	1100011w mod000 r/m immediate data	2/2	2/2	b	h
Immediate to Register (short form)	1011 w reg immediate data	2	2		
Memory to Accumulator (short form)	1010000w full displacement	4	4	b	h
Accumulator to Memory (short form)	1010001w full displacement	2	2	b	h
Register Memory to Segment Register	10001110 mod sreg3 r/m	2/5	18/19	b	h, i, j
Segment Register to Register/Memory	10001100 mod sreg3 r/m	2/2	2/2	b	h
MOVSX = Move With Sign Extension					
Register From Register/Memory	00001111 1011111w mod reg r/m	3/6	3/6	b	h
MOVZX = Move With Zero Extension					
Register From Register/Memory	00001111 1011011w mod reg r/m	3/6	3/6	b	h
PUSH = Push:					
Register/Memory	11111111 mod110 r/m	5	5	b	h
Register (short form)	01010 reg	2	2	b	h
Segment Register (ES, CS, SS or DS)	000sreg2110	2	2	b	h
Segment Register (FS or GS)	00001111 10sreg3000	2	2	b	h
Immediate	011010s0 immediate data	2	2	b	h
PUSHA = Push All	01100000	18	18	b	h
POP = Pop					
Register/Memory	10001111 mod000 r/m	5	5	b	h
Register (short form)	01011 reg	4	4	b	h
Segment Register (ES, SS or DS)	000sreg2111	7	21	b	h, i, j
Segment Register (FS or GS)	00001111 10sreg3001	7	21	b	h, i, j
POPA = Pop All	01100001	24	24	b	h
XCHG = Exchange					
Register/Memory With Register	1000011w mod reg r/m	3/5	3/5	b, f	f, h
Register With Accumulator (short form)	10010 reg	3	3		
IN = Input from:	**Clk Count Virtual 8086 Mode**				
Fixed Port	1110010w port number †26	12	6*/26**		m
Variable Port	1110110w †27	13	7*/27**		m
OUT = Output to:					
Fixed Port	1110011w port number †24	10	4*/24**		m
Variable Port	1110111w †25	11	5*/25**		m
LEA = Load EA to Register	10001101 mod reg r/m	2	2		

* If CPL ≤ IOPL ** If CPL > IOPL

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
SEGMENT CONTROL					
LDS = Load Pointer to DS	11000101 | mod reg r/m	7	22	b	h, i, j
LES = Load Pointer to ES	11000100 | mod reg r/m	7	22	b	h, i, j
LFS = Load Pointer to FS	00001111 | 10110100 | mod reg r/m	7	25	b	h, i, j
LGS = Load Pointer to GS	00001111 | 10110101 | mod reg r/m	7	25	b	h, i, j
LSS = Load Pointer to SS	00001111 | 10110010 | mod reg r/m	7	22	b	h, i, j
FLAG CONTROL					
CLC = Clear Carry Flag	11111000	2	2		
CLD = Clear Direction Flag	11111100	2	2		
CLI = Clear Interrupt Enable Flag	11111010	8	8		m
CLTS = Clear Task Switched Flag	00001111 | 00000110	6	6	c	l
CMC = Complement Carry Flag	11110101	2	2		
LAHF = Load AH into Flag	10011111	2	2		
POPF = Pop Flags	10011101	5	5	b	h, n
PUSHF = Push Flags	10011100	4	4	b	h
SAHF = Store AH into Flags	10011110	3	3		
STC = Set Carry Flag	11111001	2	2		
STD = Set Direction Flag	11111101	2	2		
STI = Set Interrupt Enable Flag	11111011	8	8		m
ARITHMETIC					
ADD = Add					
Register to Register	000000dw | mod reg r/m	2	2		
Register to Memory	0000000w | mod reg r/m	7	7	b	h
Memory to Register	0000001w | mod reg r/m	6	6	b	h
Immediate to Register/Memory	100000sw | mod 000 r/m | immediate data	2/7	2/7	b	h
Immediate to Accumulator (short form)	0000010w | immediate data	2	2		
ADC = Add With Carry					
Register to Register	000100dw | mod reg r/m	2	2		
Register to Memory	0001000w | mod reg r/m	7	7	b	h
Memory to Register	0001001w | mod reg r/m	6	6	b	h
Immediate to Register/Memory	100000sw | mod 010 r/m | immediate data	2/7	2/7	b	h
Immediate to Accumulator (short form)	0001010w | immediate data	2	2		
INC = Increment					
Register/Memory	1111111w | mod 000 r/m	2/6	2/6	b	h
Register (short form)	01000 reg	2	2		
SUB = Subtract					
Register from Register	001010dw | mod reg r/m	2	2		

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
ARITHMETIC (Continued)					
Register from Memory	0010100w | mod reg r/m	7	7	b	h
Memory from Register	0010101w | mod reg r/m	6	6	b	h
Immediate from Register/Memory	100000sw | mod101 r/m | immediate data	2/7	2/7	b	h
Immediate from Accumulator (short form)	0010110w | immediate data	2	2		
SBB = Subtract with Borrow					
Register from Register	000110dw | mod reg r/m	2	2		
Register from Memory	0001100w | mod reg r/m	7	7	b	h
Memory from Register	0001101w | mod reg r/m	6	6	b	h
Immediate from Register/Memory	100000sw | mod011 r/m | immediate data	2/7	2/7	b	h
Immediate from Accumulator (short form)	0001110w | immediate data	2	2		
DEC = Decrement					
Register/Memory	1111111w | reg001 r/m	2/6	2/6	b	h
Register (short form)	01001 reg	2	2		
CMP = Compare					
Register with Register	001110dw | mod reg r/m	2	2		
Memory with Register	0011100w | mod reg r/m	5	5	b	h
Register with Memory	0011101w | mod reg r/m	6	6	b	h
Immediate with Register/Memory	100000sw | mod111 r/m | immediate data	2/5	2/5	b	h
Immediate with Accumulator (short form)	0011110w | immediate data	2	2		
NEG = Change Sign	1111011w | mod011 r/m	2/6	2/6	b	h
AAA = ASCII Adjust for Add	00110111	4	4		
AAS = ASCII Adjust for Subtract	00111111	4	4		
DAA = Decimal Adjust for Add	00100111	4	4		
DAS = Decimal Adjust for Subtract	00101111	4	4		
MUL = Multiply (unsigned)					
Accumulator with Register/Memory	1111011w | mod100 r/m				
Multiplier-Byte		12–17/15-20	12–17/15–20	b, d	d, h
-Word		12–25/15-28	12–25/15–28	b, d	d, h
-Doubleword		12–41/15-44	12–41/15–44	b, d	d, h
IMUL = Integer Multiply (signed)					
Accumulator with Register/Memory	1111011w | mod101 r/m				
Multiplier-Byte		12–17/15-20	12–17/15–20	b, d	d, h
-Word		12–25/15-28	12–25/15–28	b, d	d, h
-Doubleword		12–41/15-44	12–41/15–44	b, d	d, h
Register with Register/Memory	00001111 | 10101111 | mod reg r/m				
Multiplier-Byte		12–17/15-20	12–17/15–20	b, d	d, h
-Word		12–25/15-28	12–25/15–28	b, d	d, h
-Doubleword		12–41/15-44	12–41/15–44	b, d	d, h
Register/Memory with Immediate to Register	011010s1 | mod reg r/m | immediate data				
-Word		13–26/14-27	13–26/14–27	b, d	d, h
-Doubleword		13–42/14-43	13–42/14–43	b, d	d, h

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
DIV = Divide (Unsigned)					
Accumulator by Register/Memory	1111011w | mod110 r/m				
Divisor—Byte		14/17	14/17	b,e	e,h
—Word		22/25	22/25	b,e	e,h
—Doubleword		38/41	38/41	b,e	e,h
IDIV = Integer Divide (Signed)					
Accumulator By Register/Memory	1111011w | mod111 r/m				
Divisor—Byte		19/22	19/22	b,e	e,h
—Word		27/30	27/30	b,e	e,h
—Doubleword		43/46	43/46	b,e	e,h
AAD = ASCII Adjust for Divide	11010101 | 00001010	19	19		
AAM = ASCII Adjust for Multiply	11010100 | 00001010	17	17		
CBW = Convert Byte to Word	10011000	3	3		
CWD = Convert Word to Double Word	10011001	2	2		
LOGIC					
Shift Rotate Instructions					
Not Through Carry (**ROL, ROR, SAL, SAR, SHL,** and **SHR**)					
Register/Memory by 1	1101000w | mod TTT r/m	3/7	3/7	b	h
Register/Memory by CL	1101001w | mod TTT r/m	3/7	3/7	b	h
Register/Memory by Immediate Count	1100000w | mod TTT r/m | immed 8-bit data	3/7	3/7	b	h
Through Carry (**RCL** and **RCR**)					
Register/Memory by 1	1101000w | mod TTT r/m	9/10	9/10	b	h
Register/Memory by CL	1101001w | mod TTT r/m	9/10	9/10	b	h
Register/Memory by Immediate Count	1100000w | mod TTT r/m | immed 8-bit data	9/10	9/10	b	h
	TTT Instruction 000 ROL 001 ROR 010 RCL 011 RCR 100 SHL/SAL 101 SHR 111 SAR				
SHLD = Shift Left Double					
Register/Memory by Immediate	00001111 | 10100100 | mod reg r/m | immed 8-bit data	3/7	3/7		
Register/Memory by CL	00001111 | 10100101 | mod reg r/m	3/7	3/7		
SHRD = Shift Right Double					
Register/Memory by Immediate	00001111 | 10101100 | mod reg r/m | immed 8-bit data	3/7	3/7		
Register/Memory by CL	00001111 | 10101101 | mod reg r/m	3/7	3/7		
AND = And					
Register to Register	001000dw | mod reg r/m	2	2		

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT			CLOCK COUNT		NOTES	
				Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
LOGIC (Continued)							
Register to Memory	0010000w	mod reg r/m		7	7	b	h
Memory to Register	0010001w	mod reg r/m		6	6	b	h
Immediate to Register/Memory	100000sw	mod 100 r/m	immediate data	2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0010010w	immediate data		2	2		
TEST = And Function to Flags, No Result							
Register/Memory and Register	1000010w	mod reg r/m		2/5	2/5	b	h
Immediate Data and Register/Memory	1111011w	mod 000 r/m	immediate data	2/5	2/5	b	h
Immediate Data and Accumulator (Short Form)	1010100w	immediate data		2	2		
OR = Or							
Register to Register	000010dw	mod reg r/m		2	2		
Register to Memory	0000100w	mod reg r/m		7	7	b	h
Memory to Register	0000101w	mod reg r/m		6	6	b	h
Immediate to Register/Memory	100000sw	mod 001 r/m	immediate data	2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0000110w	immediate data		2	2		
XOR = Exclusive Or							
Register to Register	001100dw	mod reg r/m		2	2		
Register to Memory	0011000w	mod reg r/m		7	7	b	h
Memory to Register	0011001w	mod reg r/m		6	6	b	h
Immediate to Register/Memory	100000sw	mod 110 r/m	immediate data	2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0011010w	immediate data		2	2		
NOT = Invert Register/Memory	1111011w	mod 010 r/m		2/6	2/6	b	h
STRING MANIPULATION			**Clk Count Virtual 8086 Mode**				
CMPS = Compare Byte Word	1010011w			10	10	b	h
INS = Input Byte/Word from DX Port	0110110w		†29	15	9*/29**	b	h, m
LODS = Load Byte/Word to AL/AX/EAX	1010110w			5	5	b	h
MOVS = Move Byte Word	1010010w			8	8	b	h
OUTS = Output Byte/Word to DX Port	0110111w		†28	14	8*/28**	b	h, m
SCAS = Scan Byte Word	1010111w			8	8	b	h
STOS = Store Byte/Word from AL/AX/EX	1010101w			5	5	b	h
XLAT = Translate String	11010111			5	5		h
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX							
REPE CMPS = Compare String (Find Non-Match)	11110011	1010011w		5+9n	5+9n	b	h

* If CPL ≤ IOPL ** If CPL > IOPL

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
REPEATED STRING MANIPULATION (Continued)						
REPNE CMPS = Compare String						
(Find Match)	11110010 1010011w		5+9n	5+9n	b	h
REP INS = Input String	11110010 0110110w	†28+6n	14+6n	8+6n*/28+6n**	b	h, m
REP LODS = Load String	11110010 1010110w		5+6n	5+6n	b	h
REP MOVS = Move String	11110010 1010010w		8+4n	8+4n	b	h
REP OUTS = Output String	11110010 0110111w	†26+5n	12+5n	6+5n*/26+5n**	b	h, m
REPE SCAS = Scan String						
(Find Non-AL/AX/EAX)	11110011 1010111w		5+8n	5+8n	b	h
REPNE SCAS = Scan String						
(Find AL/AX/EAX)	11110010 1010111w		5+8n	5+8n	b	h
REP STOS = Store String	11110010 1010101w		5+5n	5+5n	b	h
BIT MANIPULATION						
BSF = Scan Bit Forward	00001111 10111100 mod reg r/m		11+3n	11+3n	b	h
BSR = Scan Bit Reverse	00001111 10111101 mod reg r/m		9+3n	9+3n	b	h
BT = Test Bit						
Register/Memory, Immediate	00001111 10111010 mod 100 r/m immed 8-bit data		3/6	3/6	b	h
Register/Memory, Register	00001111 10100011 mod reg r/m		3/12	3/12	b	h
BTC = Test Bit and Complement						
Register/Memory, Immediate	00001111 10111010 mod 111 r/m immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 10111011 mod reg r/m		6/13	6/13	b	h
BTR = Test Bit and Reset						
Register/Memory, Immediate	00001111 10111010 mod 110 r/m immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 10110011 mod reg r/m		6/13	6/13	b	h
BTS = Test Bit and Set						
Register/Memory, Immediate	00001111 10111010 mod 101 r/m immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 10101011 mod reg r/m		6/13	6/13	b	h
CONTROL TRANSFER						
CALL = Call						
Direct Within Segment	11101000 full displacement		7+m	7+m	b	r
Register/Memory						
Indirect Within Segment	11111111 mod 010 r/m		7+m/ 10+m	7+m/ 10+m	b	h, r
Direct Intersegment	10011010 unsigned full offset, selector		17+m	34+m	b	j,k,r

NOTES:
† Clock count shown applies if I/O permission allows I/O to the port in virtual 8086 mode. If I/O bit map denies permission exception 13 fault occurs; refer to clock counts for INT 3 instruction.
* If CPL ≤ IOPL ** If CPL > IOPL

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			52+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			86+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			94+4x+m		h,j,k,r
From 80286 Task to 80286 TSS			273		h,j,k,r
From 80286 Task to Intel386 DX TSS			298		h,j,k,r
From 80286 Task to Virtual 8086 Task (Intel386 DX TSS)			218		h,j,k,r
From Intel386 DX Task to 80286 TSS			273		h,j,k,r
From Intel386 DX Task to Intel386 DX TSS			300		h,j,k,r
From Intel386 DX Task to Virtual 8086 Task (Intel386 DX TSS)			218		h,j,k,r
Indirect Intersegment	11111111 mod011 r/m	22+m	38+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			56+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			90+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			98+4x+m		h,j,k,r
From 80286 Task to 80286 TSS			278		h,j,k,r
From 80286 Task to Intel386 DX TSS			303		h,j,k,r
From 80286 Task to Virtual 8086 Task (Intel386 DX TSS)			222		h,j,k,r
From Intel386 DX Task to 80286 TSS			278		h,j,k,r
From Intel386 DX Task to Intel386 DX TSS			305		h,j,k,r
From Intel386 DX Task to Virtual 8086 Task (Intel386 DX TSS)			222		h,j,k,r
JMP = Unconditional Jump					
Short	11101011 8-bit displacement	7+m	7+m		r
Direct within Segment	11101001 full displacement	7+m	7+m		r
Register/Memory Indirect within Segment	11111111 mod100 r/m	7+m/ 10+m	7+m/ 10+m	b	h,r
Direct Intersegment	11101010 unsigned full offset, selector	12+m	27+m		j,k,r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			45+m		h,j,k,r
From 80286 Task to 80286 TSS			274		h,j,k,r
From 80286 Task to Intel386 DX TSS			301		h,j,k,r
From 80286 Task to Virtual 8086 Task (Intel386 DX TSS)			219		h,j,k,r
From Intel386 DX Task to 80286 TSS			270		h,j,k,r
From Intel386 DX Task to Intel386 DX TSS			303		h,j,k,r
From Intel386 DX Task to Virtual 8086 Task (Intel386 DX TSS)			221		h,j,k,r
Indirect Intersegment	11111111 mod101 r/m	17+m	31+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			49+m		h,j,k,r
From 80286 Task to 80286 TSS			279		h,j,k,r
From 80286 Task to Intel386 DX TSS			306		h,j,k,r
From 80286 Task to Virtual 8086 Task (Intel386 DX TSS)			223		h,j,k,r
From Intel386 DX Task to 80286 TSS			275		h,j,k,r
From Intel386 DX Task to Intel386 DX TSS			308		h,j,k,r
From Intel386 DX Task to Virtual 8086 Task (Intel386 DX TSS)			225		h,j,k,r

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
RET = Return from CALL:					
Within Segment	11000011	10 + m	10 + m	b	g, h, r
Within Segment Adding Immediate to SP	11000010 | 16-bit displ	10 + m	10 + m	b	g, h, r
Intersegment	11001011	18 + m	32+m	b	g, h, j, k, r
Intersegment Adding Immediate to SP	11001010 | 16-bit displ	18 + m	32+m	b	g, h, j, k, r
Protected Mode Only (RET):					
to Different Privilege Level					
Intersegment			69		h, j, k, r
Intersegment Adding Immediate to SP			69		h, j, k, r
CONDITIONAL JUMPS					
NOTE: Times Are Jump "Taken or Not Taken"					
JO = Jump on Overflow					
8-Bit Displacement	01110000 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000000 | full displacement	7 + m or 3	7 + m or 3		r
JNO = Jump on Not Overflow					
8-Bit Displacement	01110001 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000001 | full displacement	7 + m or 3	7 + m or 3		r
JB/JNAE = Jump on Below/Not Above or Equal					
8-Bit Displacement	01110010 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000010 | full displacement	7 + m or 3	7 + m or 3		r
JNB/JAE = Jump on Not Below/Above or Equal					
8-Bit Displacement	01110011 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000011 | full displacement	7 + m or 3	7 + m or 3		r
JE/JZ = Jump on Equal/Zero					
8-Bit Displacement	01110100 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000100 | full displacement	7 + m or 3	7 + m or 3		r
JNE/JNZ = Jump on Not Equal/Not Zero					
8-Bit Displacement	01110101 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000101 | full displacement	7 + m or 3	7 + m or 3		r
JBE/JNA = Jump on Below or Equal/Not Above					
8-Bit Displacement	01110110 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000110 | full displacement	7 + m or 3	7 + m or 3		r
JNBE/JA = Jump on Not Below or Equal/Above					
8-Bit Displacement	01110111 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10000111 | full displacement	7 + m or 3	7 + m or 3		r
JS = Jump on Sign					
8-Bit Displacement	01111000 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001000 | full displacement	7 + m or 3	7 + m or 3		r

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL JUMPS (Continued)					
JNS = Jump on Not Sign					
8-Bit Displacement	01111001 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001001 | full displacement	7 + m or 3	7 + m or 3		r
JP/JPE = Jump on Parity/Parity Even					
8-Bit Displacement	01111010 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001010 | full displacement	7 + m or 3	7 + m or 3		r
JNP/JPO = Jump on Not Parity/Parity Odd					
8-Bit Displacement	01111011 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001011 | full displacement	7 + m or 3	7 + m or 3		r
JL/JNGE = Jump on Less/Not Greater or Equal					
8-Bit Displacement	01111100 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001100 | full displacement	7 + m or 3	7 + m or 3		r
JNL/JGE = Jump on Not Less/Greater or Equal					
8-Bit Displacement	01111101 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001101 | full displacement	7 + m or 3	7 + m or 3		r
JLE/JNG = Jump on Less or Equal/Not Greater					
8-Bit Displacement	01111110 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001110 | full displacement	7 + m or 3	7 + m or 3		r
JNLE/JG = Jump on Not Less or Equal/Greater					
8-Bit Displacement	01111111 | 8-bit displ	7 + m or 3	7 + m or 3		r
Full Displacement	00001111 | 10001111 | full displacement	7 + m or 3	7 + m or 3		r
JCXZ = Jump on CX Zero	11100011 | 8-bit displ	9 + m or 5	9 + m or 5		r
JECXZ = Jump on ECX Zero	11100011 | 8-bit displ	9 + m or 5	9 + m or 5		r
(Address Size Prefix Differentiates JCXZ from JECXZ)					
LOOP = Loop CX Times	11100010 | 8-bit displ	11 + m	11 + m		r
LOOPZ/LOOPE = Loop with Zero/Equal	11100001 | 8-bit displ	11 + m	11 + m		r
LOOPNZ/LOOPNE = Loop While Not Zero	11100000 | 8-bit displ	11 + m	11 + m		r
CONDITIONAL BYTE SET NOTE: Times Are Register/Memory					
SETO = Set Byte on Overflow					
To Register/Memory	00001111 | 10010000 | mod000 r/m	4/5	4/5		h
SETNO = Set Byte on Not Overflow					
To Register/Memory	00001111 | 10010001 | mod000 r/m	4/5	4/5		h
SETB/SETNAE = Set Byte on Below/Not Above or Equal					
To Register/Memory	00001111 | 10010010 | mod000 r/m	4/5	4/5		h

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL BYTE SET (Continued)					
SETNB = Set Byte on Not Below/Above or Equal					
To Register/Memory	00001111 10010011 mod000 r/m	4/5	4/5		h
SETE/SETZ = Set Byte on Equal/Zero					
To Register/Memory	00001111 10010100 mod000 r/m	4/5	4/5		h
SETNE/SETNZ = Set Byte on Not Equal/Not Zero					
To Register/Memory	00001111 10010101 mod000 r/m	4/5	4/5		h
SETBE/SETNA = Set Byte on Below or Equal/Not Above					
To Register/Memory	00001111 10010110 mod000 r/m	4/5	4/5		h
SETNBE/SETA = Set Byte on Not Below or Equal/Above					
To Register/Memory	00001111 10010111 mod000 r/m	4/5	4/5		h
SETS = Set Byte on Sign					
To Register/Memory	00001111 10011000 mod000 r/m	4/5	4/5		h
SETNS = Set Byte on Not Sign					
To Register/Memory	00001111 10011001 mod000 r/m	4/5	4/5		h
SETP/SETPE = Set Byte on Parity/Parity Even					
To Register/Memory	00001111 10011010 mod000 r/m	4/5	4/5		h
SETNP/SETPO = Set Byte on Not Parity/Parity Odd					
To Register/Memory	00001111 10011011 mod000 r/m	4/5	4/5		h
SETL/SETNGE = Set Byte on Less/Not Greater or Equal					
To Register/Memory	00001111 10011100 mod000 r/m	4/5	4/5		h
SETNL/SETGE = Set Byte on Not Less/Greater or Equal					
To Register/Memory	00001111 01111101 mod000 r/m	4/5	4/5		h
SETLE/SETNG = Set Byte on Less or Equal/Not Greater					
To Register/Memory	00001111 10011110 mod000 r/m	4/5	4/5		h
SETNLE/SETG = Set Byte on Not Less or Equal/Greater					
To Register/Memory	00001111 10011111 mod000 r/m	4/5	4/5		h
ENTER = Enter Procedure	11001000 16-bit displacement, 8-bit level				
L = 0		10	10	b	h
L = 1		12	12	b	h
L > 1		15 + 4(n − 1)	15 + 4(n − 1)	b	h
LEAVE = Leave Procedure	11001001	4	4	b	h

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS					
INT = Interrupt:					
Type Specified	11001101 | type	37		b	
Type 3	11001100	33		b	
INTO = Interrupt 4 if Overflow Flag Set	11001110				
If OF = 1		35		b, e	
If OF = 0		3	3	b, e	
Bound = Interrupt 5 if Detect Value Out of Range	01100010 | mod reg r/m				
If Out of Range		44		b, e	e, g, h, j, k, r
If In Range		10	10	b, e	e, g, h, j, k, r
Protected Mode Only (INT)					
INT: Type Specified					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			282		g, j, k, r
From 80286 Task to Intel386 DX TSS via Task Gate			309		g, j, k, r
From 80286 Task to virt 8086 md via Task Gate			226		g, j, k, r
From Intel386 DX Task to 80286 TSS via Task Gate			284		g, j, k, r
From Intel386 DX Task to Intel386 DX TSS via Task Gate			311		g, j, k, r
From Intel386 DX Task to virt 8086 md via Task Gate			228		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			289		g, j, k, r
From virt 8086 md to Intel386 DX TSS via Task Gate			316		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INT: TYPE 3					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			278		g, j, k, r
From 80286 Task to Intel386 DX TSS via Task Gate			305		g, j, k, r
From 80286 Task to Virt 8086 md via Task Gate			222		g, j, k, r
From Intel386 DX Task to 80286 TSS via Task Gate			280		g, j, k, r
From Intel386 DX Task to Intel386 DX TSS via Task Gate			307		g, j, k, r
From Intel386 DX Task to Virt 8086 md via Task Gate			224		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			285		g, j, k, r
From virt 8086 md to Intel386 DX TSS via Task Gate			312		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INTO:					
Via Interrupt or Trap Grate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			280		g, j, k, r
From 80286 Task to Intel386 DX TSS via Task Gate			307		g, j, k, r
From 80286 Task to virt 8086 md via Task Gate			224		g, j, k, r
From Intel386 DX Task to 80286 TSS via Task Gate			282		g, j, k, r
From Intel386 DX Task to Intel386 DX TSS via Task Gate			309		g, j, k, r
From Intel386 DX Gate			225		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			287		g, j, k, r
From virt 8086 md to Intel386 DX TSS via Task Gate			314		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		

3

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS (Continued)					
BOUND:					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			254		g, j, k, r
From 80286 Task to Intel386 DX TSS via Task Gate			284		g, j, k, r
From 80268 Task to virt 8086 Mode via Task Gate			231		g, j, k, r
From Intel386 DX Task to 80286 TSS via Task Gate			264		g, j, k, r
From Intel386 DX Task to Intel386 DX TSS via Task Gate			294		g, j, k, r
From 80368 Task to virt 8086 Mode via Task Gate			243		g, j, k, r,
From virt 8086 Mode to 80286 TSS via Task Gate			264		g, j, k, r
From virt 8086 Mode to Intel386 DX TSS via Task Gate			294		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INTERRUPT RETURN					
IRET = Interrupt Return	11001111	22			g, h, j, k, r
Protected Mode Only (IRET)					
To the Same Privilege Level (within task)			38		g, h, j, k, r
To Different Privilege Level (within task)			82		g, h, j, k, r
From 80286 Task to 80286 TSS			232		h, j, k, r
From 80286 Task to Intel386 DX TSS			265		h, j, k, r
From 80286 Task to Virtual 8086 Task			213		h, j, k, r
From 80286 Task to Virtual 8086 Mode (within task)			60		
From Intel386 DX Task to 80286 TSS			271		h, j, k, r
From Intel386 DX Task to Intel386 DX TSS			275		h, j, k, r
From Intel386 DX Task to Virtual 8086 Task			223		h, j, k, r
From Intel386 DX Task to Virtual 8086 Mode (within task)			60		
PROCESSOR CONTROL					
HLT = HALT	11110100	5	5		I
MOV = Move to and From Control/Debug/Test Registers					
CR0/CR2/CR3 from register	00001111 00100010 11 eee reg	11/4/5	11/4/5		I
Register From CR0–3	00001111 00100000 11 eee reg	6	6		I
DR0–3 From Register	00001111 00100011 11 eee reg	22	22		I
DR6–7 From Register	00001111 00100011 11 eee reg	16	16		I
Register from DR6–7	00001111 00100001 11 eee reg	14	14		I
Register from DR0–3	00001111 00100001 11 eee reg	22	22		I
TR6–7 from Register	00001111 00100110 11 eee reg	12	12		I
Register from TR6–7	00001111 00100100 11 eee reg	12	12		I
NOP = No Operation	10010000	3	3		
WAIT = Wait until BUSY# pin is negated	10011011	7	7		

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
PROCESSOR EXTENSION INSTRUCTIONS					
Processor Extension Escape	11011TTT | mod LLL r/m TTT and LLL bits are opcode information for coprocessor.	See 80287/80Intel387 data sheets for clock counts			h
PREFIX BYTES					
Address Size Prefix	01100111	0	0		
LOCK = Bus Lock Prefix	11110000	0	0		m
Operand Size Prefix	01100110	0	0		
Segment Override Prefix					
CS:	00101110	0	0		
DS:	00111110	0	0		
ES:	00100110	0	0		
FS:	01100100	0	0		
GS:	01100101	0	0		
SS:	00110110	0	0		
PROTECTION CONTROL					
ARPL = Adjust Requested Privilege Level					
From Register/Memory	01100011 | mod reg r/m	N/A	20/21	a	h
LAR = Load Access Rights					
From Register/Memory	00001111 | 00000010 | mod reg r/m	N/A	15/16	a	g, h, j, p
LGDT = Load Global Descriptor					
Table Register	00001111 | 00000001 | mod 010 r/m	11	11	b, c	h, l
LIDT = Load Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod 011 r/m	11	11	b, c	h, l
LLDT = Load Local Descriptor					
Table Register to Register/Memory	00001111 | 00000000 | mod 010 r/m	N/A	20/24	a	g, h, j, l
LMSW = Load Machine Status Word					
From Register/Memory	00001111 | 00000001 | mod 110 r/m	11/14	11/14	b, c	h, l
LSL = Load Segment Limit					
From Register/Memory	00001111 | 00000011 | mod reg r/m				
Byte-Granular Limit		N/A	21/22	a	g, h, j, p
Page-Granular Limit		N/A	25/26	a	g, h, j, p
LTR = Load Task Register					
From Register/Memory	00001111 | 00000000 | mod 011 r/m	N/A	23/27	a	g, h, j, l
SGDT = Store Global Descriptor					
Table Register	00001111 | 00000001 | mod 000 r/m	9	9	b, c	h
SIDT = Store Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod 001 r/m	9	9	b, c	h
SLDT = Store Local Descriptor Table Register					
To Register/Memory	00001111 | 00000000 | mod 000 r/m	N/A	2/2	a	h

3

Table 6-1. Intel386™ DX Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
SMSW = Store Machine Status Word	00001111 | 00000001 | mod 100 r/m	2/2	2/2	b, c	h, l
STR = Store Task Register					
To Register/Memory	00001111 | 00000000 | mod 001 r/m	N/A	2/2	a	h
VERR = Verify Read Accesss					
Register/Memory	00001111 | 00000000 | mod 100 r/m	N/A	10/11	a	g, h, j, p
VERW = Verify Write Accesss	00001111 | 00000000 | mod 101 r/m	N/A	15/16	a	g, h, j, p

INSTRUCTION NOTES FOR TABLE 6-1

Notes a through c apply to Intel386 DX Real Address Mode only:

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid opcode).

b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS or GS limit, FFFFH. Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Notes d through g apply to Intel386 DX Real Address Mode and Intel386 DX Protected Virtual Address Mode:

d. The Intel386 DX uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).

Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:

Actual Clock = if m < > 0 then max ([$\log_2$ |m|], 3) + b clocks:

if m = 0 then 3+b clocks

In this formula, m is the multiplier, and

b = 9 for register to register,

b = 12 for memory to register,

b = 10 for register with immediate to register,

b = 11 for memory with immediate to register.

e. An exception may occur, depending on the value of the operand.

f. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK# prefix.

g. LOCK# is asserted during descriptor table accesses.

Notes h through r apply to Intel386 DX Protected Virtual Address Mode only:

h. Exception 13 fault (general protection violation) will occur if the memory operand in CS, DS, ES, FS or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, an exception 12 (stack segment limit violation or not present) occurs.

i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present) occurs.

j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.

k. JMP, CALL, INT, RET and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is violated.

l. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).

m. An exception 13 fault occurs if CPL is greater than IOPL.

n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.

o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 if desiring to reset the PE bit.

p. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.

q. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or not present) will occur if the stack limit is violated by the operand's starting address.

r. The destination of a JMP, CALL, INT, RET or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.

6.2 INSTRUCTION ENCODING

6.2.1 Overview

All instruction encodings are subsets of the general instruction format shown in Figure 6-1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the "mod r/m" byte and "scaled index" byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16 or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure 6-1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the Figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table 6-2 is a complete list of all fields appearing in the Intel386 DX instruction set. Further ahead, following Table 6-2, are detailed tables for each field.

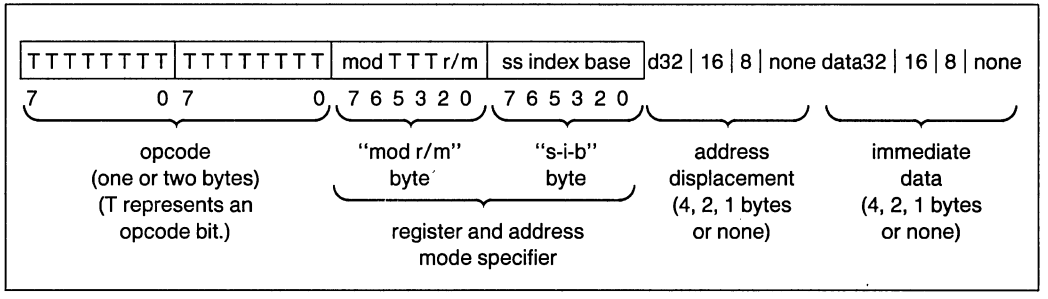


Figure 6-1. General Instruction Format

Table 6-2. Fields within Intel386™ DX Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bits	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

Note: Table 6-1 shows encoding of individual instructions.

6.2.2 32-Bit Extensions of the Instruction Set

With the Intel386 DX, the 8086/80186/80286 instruction set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types, and 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

Whether the instruction defaults to operations of 16 bits or 32 bits depends on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 bits or 16 bits) for both operands and effective addresses when executing that code segment. In the Real Address Mode or Virtual 8086 Mode, no code segment descriptors are used, but a D value of 0 is assumed internally by the Intel386 DX when operating in those modes (for 16-bit default sizes compatible with the 8086/80186/80286).

Two prefixes, the Operand Size Prefix and the Effective Address Size Prefix, allow overriding individually the Default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the opcode bytes. The presence of the Operand Size Prefix and the Effective Address Prefix will toggle the operand size or the effective address size, respectively, to the value "opposite" from the Default setting. For example, if the default operand size is for 32-bit data operations, then presence of the Operand Size Prefix toggles the instruction to 16-bit data operation. As another example, if the default effective address size is 16 bits, presence of the Effective Address Size prefix toggles the instruction to use 32-bit effective address computations.

These 32-bit extensions are available in all Intel386 DX modes, including the Real Address Mode or the Virtual 8086 Mode. In these modes the default is always 16 bits, so prefixes are needed to specify 32-bit operands or addresses. For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

6.2.3 Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on. The exact encodings of these fields are defined immediately ahead.

6.2.3.1 ENCODING OF OPERAND LENGTH (w) FIELD

For any given instruction performing a data operation, the instruction is executing as a 32-bit operation or a 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

6.2.3.2 ENCODING OF THE GENERAL REGISTER (reg) FIELD

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations:		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register Specified by reg Field During 32-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

6.2.3.3 ENCODING OF THE SEGMENT REGISTER (sreg) FIELD

The sreg field in certain instructions is a 2-bit field allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the Intel386 DX FS and GS segment registers to be specified.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

6.2.3.4 ENCODING OF ADDRESS MODE

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte, can be specified.

The s-i-b byte (scale-index-base byte) is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01 or 10. When the sib byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure 6-1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following three pages define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Encoding of 16-bit Address Mode with "mod r/m" Byte

mod r/m	Effective Address
00 000	DS:[BX+SI]
00 001	DS:[BX+DI]
00 010	SS:[BP+SI]
00 011	SS:[BP+DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX+SI+d8]
01 001	DS:[BX+DI+d8]
01 010	SS:[BP+SI+d8]
01 011	SS:[BP+DI+d8]
01 100	DS:[SI+d8]
01 101	DS:[DI+d8]
01 110	SS:[BP+d8]
01 111	DS:[BX+d8]

mod r/m	Effective Address
10 000	DS:[BX+SI+d16]
10 001	DS:[BX+DI+d16]
10 010	SS:[BP+SI+d16]
10 011	SS:[BP+DI+d16]
10 100	DS:[SI+d16]
10 101	DS:[DI+d16]
10 110	SS:[BP+d16]
10 111	DS:[BX+d16]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by r/m During 16-Bit Data Operations		
mod r/m	Function of w Field	
	(when w=0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-Bit Data Operations		
mod r/m	Function of w Field	
	(when w=0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode with "mod r/m" byte (no "s-i-b" byte present):

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]

mod r/m	Effective Address
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by reg or r/m during 16-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m during 32-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode ("mod r/m" byte and "s-i-b" byte present):

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

NOTE:
Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg**
101	EBP
110	ESI
111	EDI

****IMPORTANT NOTE:**
When index field is 100, indicating "no index register," then ss field MUST equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

6.2.3.5 ENCODING OF OPERATION DIRECTION (d) FIELD

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory <- - Register "reg" Field Indicates Source Operand; "mod r/m" or "mod ss index base" Indicates Destination Operand
1	Register <- - Register/Memory "reg" Field Indicates Destination Operand; "mod r/m" or "mod ss index base" Indicates Source Operand

6.2.3.6 ENCODING OF SIGN-EXTEND (s) FIELD

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

s	Effect on Immediate Data8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extend Data8 to Fill 16-Bit or 32-Bit Destination	None

6.2.3.7 ENCODING OF CONDITIONAL TEST (tttn) FIELD

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n = 0) or its negation (n = 1), and ttt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111

6.2.3.8 ENCODING OF CONTROL OR DEBUG OR TEST REGISTER (eee) FIELD

For the loading and storing of the Control, Debug and Test registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	CR2
011	CR3
Do not use any other encoding	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding	

When Interpreted as Test Register Field

eee Code	Reg Name
110	TR6
111	TR7
Do not use any other encoding	

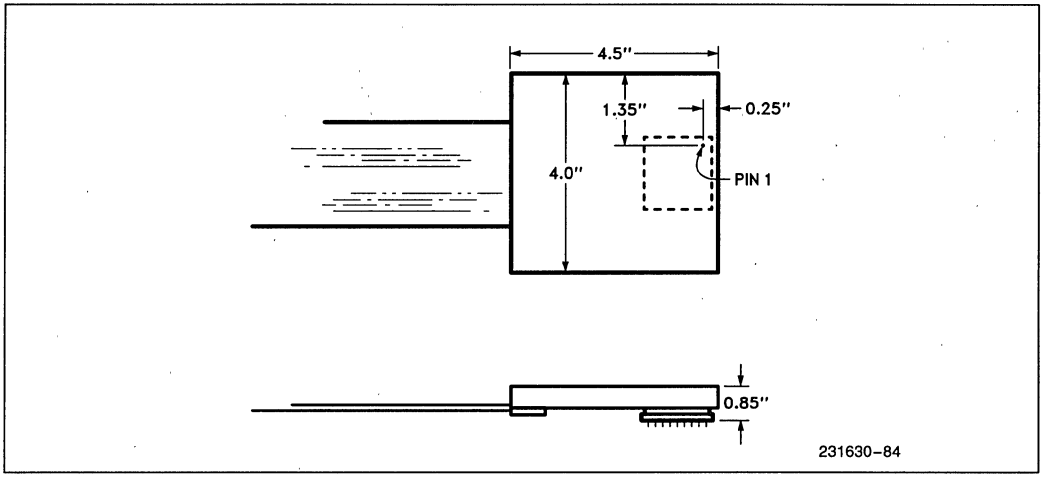


Figure 7-1. Processor Module Dimensions

7. DESIGNING FOR ICE™-Intel386 DX EMULATOR USE

The Intel386 DX in-circuit emulator products are ICE-Intel386 DX 25 MHz or 33 MHz (both referred to as ICE-Intel386 DX emulator). The ICE-Intel386 DX emulator probe module has several electrical and mechanical characteristics that should be taken into consideration when designing the hardware.

Capacitive loading: The ICE-Intel386 DX emulator adds up to 25 pF to each line.

Drive requirement: The ICE-Intel386 DX emulator adds one standard TTL load on the CLK2 line, up to one advanced low-power Schottky TTL load per control signal line, and one advanced low-power Schottky TTL load per address, byte enable, and data line. These loads are within the probe module and are driven by the probe's Intel386 DX component, which has standard drive and loading capability listed in the A.C. and D.C. Specification Tables in Sections 9.4 and 9.5.

Power requirement: For noise immunity the ICE-Intel386 DX emulator probe is powered by the user system. This high-speed probe circuitry draws up to 1.5A plus the maximum I_{CC} from the user Intel386 DX component socket.

Intel386 DX location and orientation: The ICE-Intel386 DX processor module, target-adaptor cable (which does not exist for the ICE-Intel386 DX 33 MHz emulator), and the isolation board used for extra electrical buffering of the emulator initially, require clearance as illustrated in Figures 7-1 and 7-2.

Interface Board and CLK2 speed reduction: When the ICE-Intel386 DX emulator probe is first attached to an unverified user system, the interface board helps the ICE-Intel386 DX emulator function in user systems with bus faults (shorted signals, etc.). After electrical verification it may be removed. Only when the interface board is installed, the user system must have a reduced CLK2 frequency of 25 MHz maximum.

Cache coherence: The ICE-Intel386 DX emulator loads user memory by performing Intel386 DX component write cycles. Note that if the user system is not designed to update or invalidate its cache (if it has a cache) upon processor writes to memory, the cache could contain stale instruction code and/or data. For best use of the ICE-Intel386 DX emulator, the user should consider designing the cache (if any) to update itself automatically when processor writes occur, or find another method of maintaining cache data coherence with main user memory.

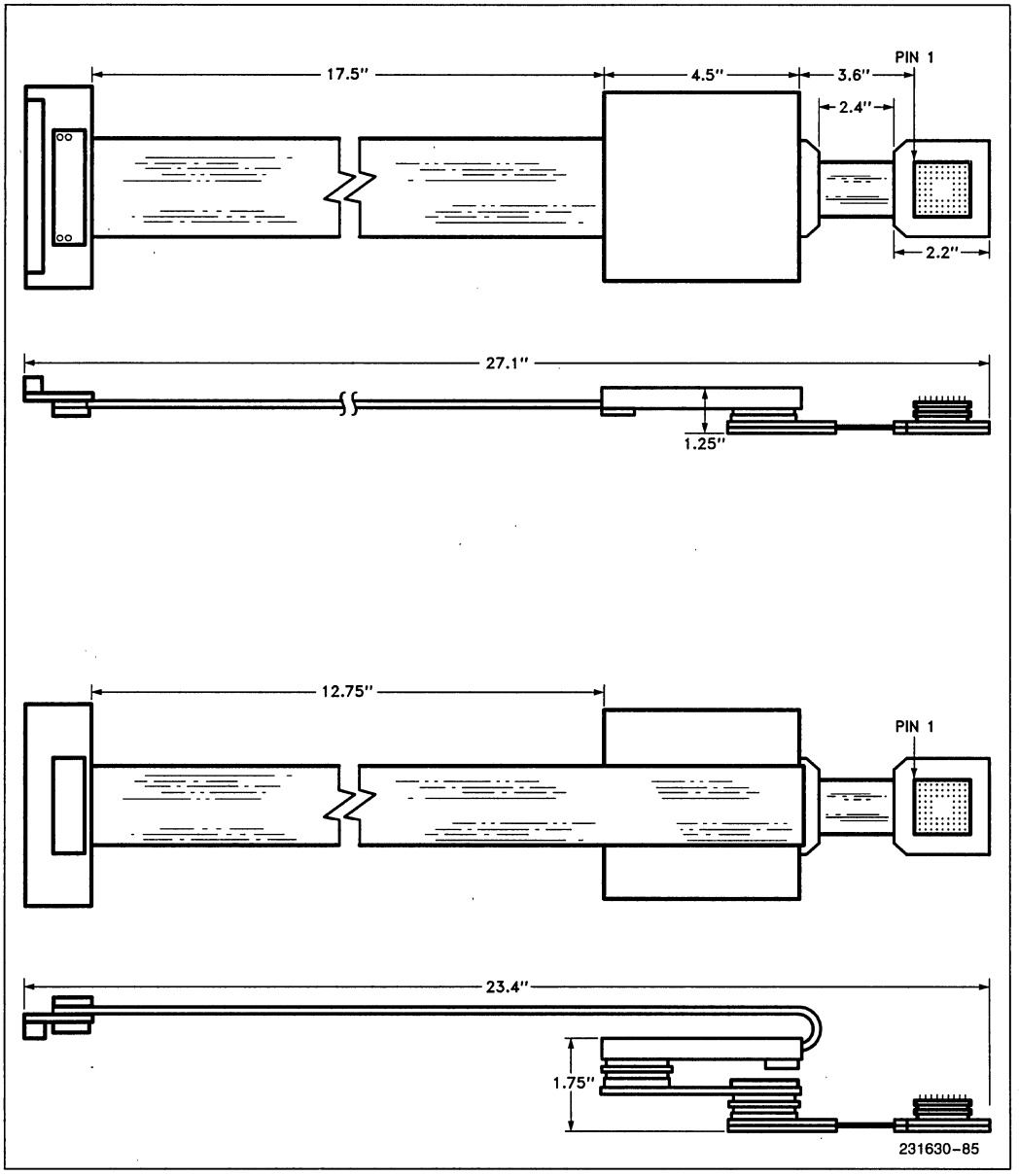


Figure 7-2. Processor Module, Target-Adapter Cable, and Isolation Board Dimensions

8. MECHANICAL DATA

8.1 INTRODUCTION

In this section, the physical packaging and its connections are described in detail.

8.2 PACKAGE DIMENSIONS AND MOUNTING

The initial Intel386 DX package is a 132-pin ceramic pin grid array (PGA). Pins of this package are arranged 0.100 inch (2.54mm) center-to-center, in a 14 x 14 matrix, three rows around.

A wide variety of available sockets allow low insertion force or zero insertion force mountings, and a choice of terminals such as soldertail, surface mount, or wire wrap. Several applicable sockets are listed in Table 8.1.

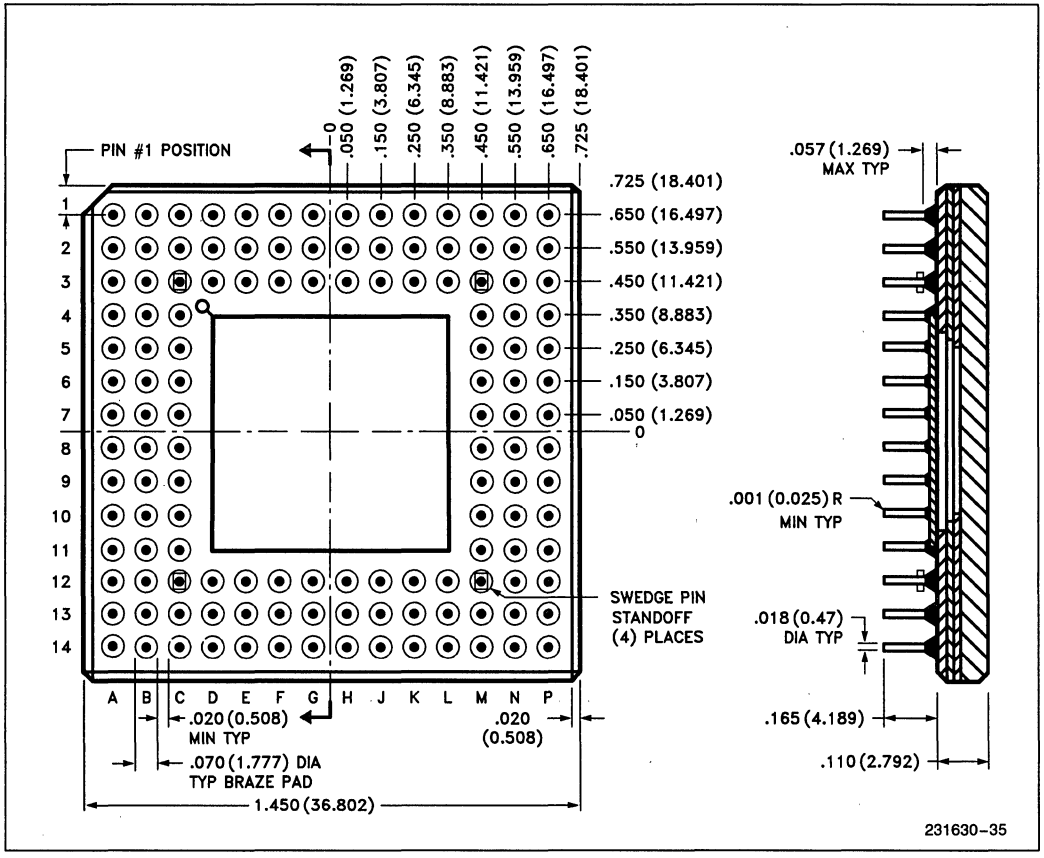


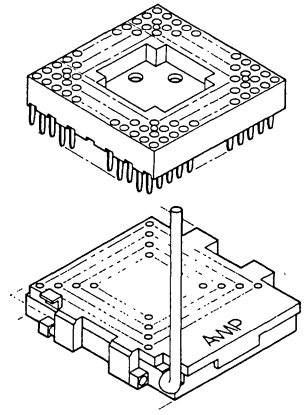
Figure 8.1. 132-Pin Ceramic PGA Package Dimensions

Table 8.1. Several Socket Options for 132-Pin PGA

* Low insertion force (LIF) soldertail 55274-1
* Amp tests indicate 50% reduction in insertion force compared to machined sockets

Other socket options
* Zero insertion force (ZIF) soldertail 55583-1
* Zero insertion force (ZIF) Burn-in version 55573-2

Amp Incorporated
(Harrisburg, PA 17105 U.S.A.
Phone 717-564-0100)



231630-45

Cam handle locks in low profile position when substrate is installed (handle UP for open and DOWN for closed positions)

courtesy Amp Incorporated

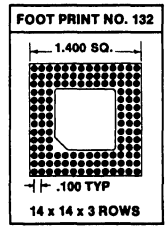
3

Peel-A-Way Mylar and Kapton Socket Terminal Carriers
* Low insertion force surface mount CS132-37TG
* Low insertion force soldertail CS132-01TG
* Low insertion force wire-wrap CS132-02TG (two level) CS132-03TG (three-level)
* Low insertion force press-fit CS132-05TG

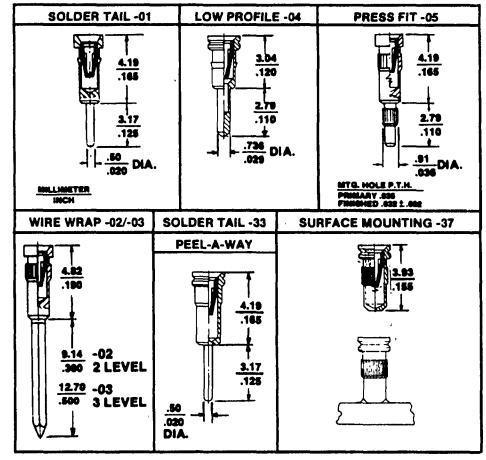
Advanced Interconnections
(5 Division Street
Warwick, RI 02818 U.S.A.
Phone 401-885-0485)

Peel-A-Way Carrier No. 132:
Kapton Carrier is KS132
Mylar Carrier is MS132

Molded Plastic Body KS132 is shown below:



231630-46



231630-47

courtesy Advanced Interconnections
(Peel-A-Way Terminal Carriers
U.S. Patent No. 4442938)

Table 8.1. Several Socket Options for 132-Pin PGA (Continued)

PIN GRID ARRAY
DECOUPLING SOCKETS

* Low insertion force soldertail 0.125 length PGD-005-1A1 Finish: Term/Contact Tin-Lead/Gold
* Low insertion force soldertail 0.180 length PGD-005-1B1 Finish: Term/Contact: Tin-Lead/Gold
* Low insertion 3 level Wire/Wrap PGD-005-1C1 Finish: Term/Contact Tin-Lead/Gold

Includes 0.10 μF & 1.0 μF Decoupling Capacitors

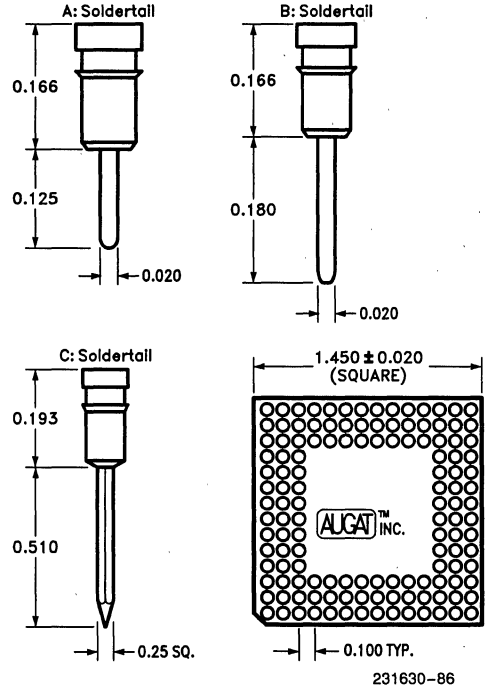
VisinPak Kapton Carrier

PKC Series

Pin Grid Array

PGM (Plastic) or PPS (Glass Epoxy) Series

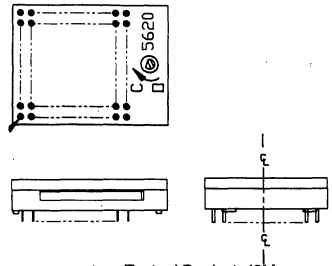
AUGAT INC.
33 Perry Ave., P.O. Box 779 Attleboro, MA 02703
TECHNICAL INFORMATION: (508) 222-2202
CUSTOMER SERVICE: (508) 699-9800



231630-86

* Low insertion force socket soldertail (for production use)
  2XX-6576-00-3308 (new style)
  2XX-6003-00-3302 (older style)
* Zero insertion force soldertail (for test and burn-in use)
  2XX-6568-00-3302

Textool Products
Electronic Products Division/3M
(1410 West Pioneer Drive
Irving, Texas 75601 U.S.A.
Phone 214-259-2676)



courtesy Textool Products/3M

231630-48

8.3 PACKAGE THERMAL SPECIFICATION

The Intel386 DX is specified for operation when case temperature is within the range of 0°C–85°C. The case temperature may be measured in any environment, to determine whether the Intel386 DX is within specified operating range.

The PGA case temperature should be measured at the center of the top surface opposite the pins, as in Figure 8.2.

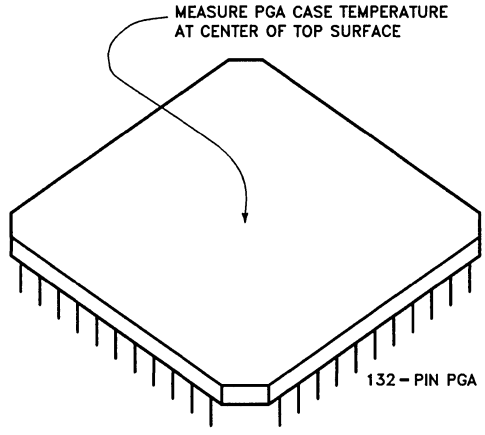
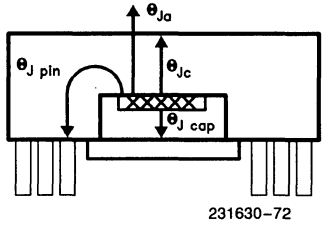


Figure 8.2. Measuring Intel386™ DX PGA Case Temperature

Table 8.2. Intel386™ DX PGA Package Thermal Characteristics

Parameter	Thermal Resistance — °C/Watt: Airflow — ft./min (m/sec) 0 (0)	50 (0.25)	100 (0.50)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)
θ Junction-to-Case (case measured as Fig. 8-2)	2	2	2	2	2	2	2
θ Case-to-Ambient (no heatsink)	19	18	17	15	12	10	9
θ Case-to-Ambient (with omnidirectional heatsink)	16	15	14	12	9	7	6
θ Case-to-Ambient (with unidirectional heatsink)	15	14	13	11	8	6	5



NOTES:
1. Table 8.2 applies to Intel386™ DX PGA plugged into socket or soldered directly into board.
2. $\theta_{JA} = \theta_{JC} + \theta_{CA}$.
3. $\theta_{J\text{-}CAP}$ = 4°C/w (approx.)
   $\theta_{J\text{-}PIN}$ = 4°C/w (inner pins) (approx.)
   $\theta_{J\text{-}PIN}$ = 8°C/w (outer pins) (approx.)
4. $T_A = T_C - P * \theta_{CA}$ (ambient temperature)

9. ELECTRICAL DATA

9.1 INTRODUCTION

The following sections describe recommended electrical connections for the Intel386 DX, and its electrical specifications.

9.2 POWER AND GROUNDING

9.2.1 Power Connections

The Intel386 DX is implemented in CHMOS III and CHMOS IV technology and has modest power requirements. However, its high clock frequency and 72 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 20 V_{CC} and 21 V_{SS} pins separately feed functional units of the Intel386 DX.

Power and ground connections must be made to all external V_{CC} and GND pins of the Intel386 DX. On the circuit board, all V_{CC} pins must be connected on a V_{CC} plane. All V_{SS} pins must be likewise connected on a GND plane.

9.2.2 Power Decoupling Recommendations

Liberal decoupling capacitance should be placed near the Intel386 DX. The Intel386 DX driving its 32-bit parallel address and data buses at high frequencies can cause transient power surges, particularly when driving large capacitive loads.

Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the Intel386 DX and decoupling capacitors as much as possible. Capacitors specifically for PGA packages are also commercially available, for the lowest possible inductance.

9.2.3 Resistor Recommendations

The ERROR# and BUSY# inputs have resistor pullups of approximately 20 KΩ built-in to the Intel386 DX to keep these signals negated when no Intel387 DX coprocessor is present in the system (or temporarily removed from its socket). The BS16# input also has an internal pullup resistor of approximately 20 KΩ, and the PEREQ input has an internal pulldown resistor of approximately 20 KΩ.

In typical designs, the external pullup resistors shown in Table 9-1 are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pullup resistors in other ways.

9.2.4 Other Connection Recommendations

For reliable operation, always connect unused inputs to an appropriate signal level. N.C. pins should always remain unconnected.

Particularly when not using interrupts or bus hold, (as when first prototyping, perhaps) prevent any chance of spurious activity by connecting these associated inputs to GND:

Pin	Signal
B7	INTR
B8	NMI
D14	HOLD

If not using address pipelining, pullup D13 NA# to V_{CC}.

If not using 16-bit bus size, pullup C14 BS16# to V_{CC}.

Pullups in the range of 20 KΩ are recommended.

Table 9-1. Recommended Resistor Pullups to V_{CC}

Pin and Signal	Pullup Value	Purpose
E14 ADS#	20 KΩ ±10%	Lightly Pull ADS# Negated During Intel386 DX Hold Acknowledge States
C10 LOCK#	20 KΩ ±10%	Lightly Pull LOCK# Negated During Intel386 DX Hold Acknowledge States

9.3 MAXIMUM RATINGS

Table 9-2. Maximum Ratings

Parameter	Intel386™ DX 20, 25, 33 MHz Maximum Rating
Storage Temperature	−65°C to +150°C
Case Temperature Under Bias	−65°C to +110°C
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V
Voltage on Other Pins	−0.5V to V_{CC} + 0.5V

Table 9-2 is a stress rating only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in **9.4 D.C. Specifications** and **9.5 A.C. Specifications**.

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the Intel386 DX contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

9.4 D.C. SPECIFICATIONS

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to 85°C

Table 9-3. Intel386™ DX D.C. Characteristics

Symbol	Parameter	Intel386™ DX 20 MHz, 25 MHz, 33 MHz		Unit	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	CLK2 Input High Voltage 20 MHz 25 MHz and 33 MHz	 V_{CC} − 0.8 3.7	 V_{CC} + 0.3 V_{CC} + 0.3	 V V	
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A2–A31, D0–D31 I_{OL} = 5 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	 	 0.45 0.45	 V V	
V_{OH}	Output High Voltage I_{OH} = 1 mA: A2–A31, D0–D31 I_{OH} = 0.9 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	 2.4 2.4	 	 V V	
I_{LI}	Input Leakage Current (For All Pins except BS16#, PEREQ, BUSY#, and ERROR#)		±15	µA	0V ≤ V_{IN} ≤ V_{CC}
I_{IH}	Input Leakage Current (PEREQ Pin)		200	µA	V_{IH} = 2.4V (Note 2)
I_{IL}	Input Leakage Current (BS16#, BUSY#, and ERROR# Pins)		−400	µA	V_{IL} = 0.45 (Note 3)
I_{LO}	Output Leakage Current		±15	µA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	Supply Current CLK2 = 40 MHz: with 20 MHz Intel386™ DX CLK2 = 50 MHz: with 25 MHz Intel386™ DX CLK2 = 66 MHz: with 33 MHz Intel386™ DX	 	 260 320 390	 mA mA mA	(Note 4) I_{CC} Typ. = 200 mA I_{CC} Typ. = 240 mA I_{CC} Typ. = 300 mA
C_{IN}	Input or I/O Capacitance		10	pF	F_C = 1 MHz
C_{OUT}	Output Capacitance		12	pF	F_C = 1 MHz
C_{CLK}	CLK2 Capacitance		20	pF	F_C = 1 MHz

NOTES:
1. The min value, −0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. BS16#, BUSY# and ERROR# inputs each have an internal pullup resistor.
4. CHMOS IV Technology (CHMOS III Max I_{CC} at 20 MHz, 25 MHz = 500 mA, 550 mA).

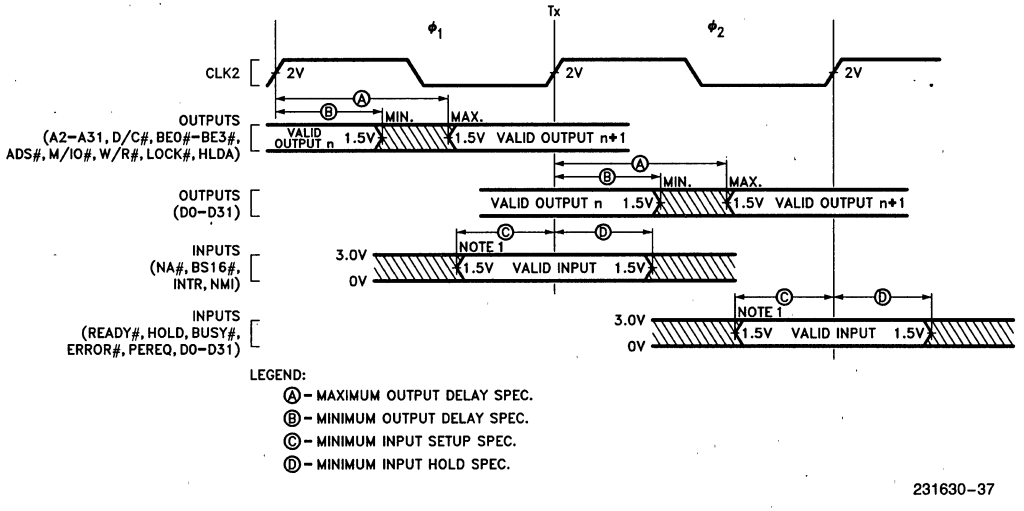
9.5 A.C. SPECIFICATIONS

9.5.1 A.C. Spec Definitions

The A.C. specifications, given in Tables 9-4, 9-5, and 9-6, consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the CLK2 rising edge crossing the 2.0V level.

A.C. spec measurement is defined by Figure 9-1. Inputs must be driven to the voltage levels indicated by Figure 9-1 when A.C. specifications are measured. Intel386 DX output delays are specified with minimum and maximum limits, measured as shown. The minimum Intel386 DX delay times are hold times provided to external circuitry. Intel386 DX input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct Intel386 DX operation.

Outputs NA#, W/R#, D/C#, M/IO#, LOCK#, BE0#–BE3#, A2–A31 and HLDA only change at the beginning of phase one. D0–D31 (write cycles) only change at the beginning of phase two. The READY#, HOLD, BUSY#, ERROR#, PEREQ and D0–D31 (read cycles) inputs are sampled at the beginning of phase one. The NA#, BS16#, INTR and NMI inputs are sampled at the beginning of phase two.



NOTES:
1. Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.
2. See section 9.5.8 for typical output rise time versus load capacitance.

Figure 9-1. Drive Levels and Measurement Points for A.C. Specifications

9.5.2 A.C. Specification Tables

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-4. 33 MHz Intel386™ DX A.C. Characteristics

Symbol	Parameter	33 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	8	33.3	MHz		Half of CLK2 Frequency
t1	CLK2 Period	15.0	62.5	ns	9-3	
t2a	CLK2 High Time	6.25		ns	9-3	at 2V
t2b	CLK2 High Time	4.5		ns	9-3	at 3.7V
t3a	CLK2 Low Time	6.25		ns	9-3	at 2V
t3b	CLK2 Low Time	4.5		ns	9-3	at 0.8V
t4	CLK2 Fall Time		4	ns	9-3	3.7V to 0.8V (Note 3)
t5	CLK2 Rise Time		4	ns	9-3	0.8V to 3.7V (Note 3)
t6	A2–A31 Valid Delay	4	15	ns	9-5	C_L = 50 pF
t7	A2–A31 Float Delay	4	20	ns	9-6	(Note 1)
t8	BE0#–BE3#, LOCK# Valid Delay	4	15	ns	9-5	C_L = 50 pF
t9	BE0#–BE3#, LOCK# Float Delay	4	20	ns	9-6	(Note 1)
t10	W/R#, M/IO#, D/C#, Valid Delay	4	15	ns	9-5	C_L = 50 pF
t10a	ADS# Valid Delay	4	14.5	ns	9-5	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	20	ns	9-6	(Note 1)
t12	D0–D31 Write Data Valid Delay	7	24	ns	9-5a	C_L = 50 pF, (Note 4)
t12a	D0–D31 Write Data Hold Time	2			9-5b	C_L = 50 pF
t13	D0–D31 Float Delay	4	17	ns	9-6	(Note 1)
t14	HLDA Valid Delay	4	20	ns	9-6	C_L = 50 pF
t15	NA# Setup Time	5		ns	9-4	
t16	NA# Hold Time	2		ns	9-4	
t17	BS16# Setup Time	5		ns	9-4	
t18	BS16# Hold Time	2		ns	9-4	
t19	READY# Setup Time	7		ns	9-4	
t20	READY# Hold Time	4		ns	9-4	

9.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-4. 33 MHz Intel386™ DX A.C. Characteristics (Continued)

Symbol	Parameter	33 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
t21	D0–D31 Read Setup Time	5		ns	9-4	
t22	D0–D31 Read Hold Time	3		ns	9-4	
t23	HOLD Setup Time	11		ns	9-4	
t24	HOLD Hold Time	2		ns	9-4	
t25	RESET Setup Time	5		ns	9-7	
t26	RESET Hold Time	2		ns	9-7	
t27	NMI, INTR Setup Time	5		ns	9-4	(Note 2)
t28	NMI, INTR Hold Time	5		ns	9-4	(Note 2)
t29	PEREQ, ERROR#, BUSY# Setup Time	5		ns	9-4	(Note 2)
t30	PEREQ, ERROR#, BUSY# Hold Time	4		ns	9-4	(Note 2)

NOTES:

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.
4. Min. time not 100% tested.

9.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-5. 25 MHz Intel386™ DX A.C. Characteristics

Symbol	Parameter	25 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	4	25	MHz		Half of CLK2 Frequency
t1	CLK2 Period	20	125	ns	9-3	
t2a	CLK2 High Time	7		ns	9-3	at 2V
t2b	CLK2 High Time	4		ns	9-3	at 3.7V
t3a	CLK2 Low Time	7		ns	9-3	at 2V
t3b	CLK2 Low Time	5		ns	9-3	at 0.8V
t4	CLK2 Fall Time		7	ns	9-3	3.7V to 0.8V
t5	CLK2 Rise Time		7	ns	9-3	0.8V to 3.7V
t6	A2–A31 Valid Delay	4	21	ns	9-5	C_L = 50 pF
t7	A2–A31 Float Delay	4	30	ns	9-6	(Note 1)
t8	BE0#–BE3# Valid Delay	4	24	ns	9-5	C_L = 50 pF
t8a	LOCK# Valid Delay	4	21	ns	9-5	C_L = 50 pF
t9	BE0#–BE3#, LOCK# Float Delay	4	30	ns	9-6	(Note 1)
t10	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	21	ns	9-5	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	30	ns	9-6	(Note 1)
t12	D0–D31 Write Data Valid Delay	7	27	ns	9-5a	C_L = 50 pF
t12a	D0–D31 Write Data Hold Time	2			9-5b	C_L = 50 pF
t13	D0–D31 Float Delay	4	22	ns	9-6	(Note 1)
t14	HLDA Valid Delay	4	22	ns	9-6	C_L = 50 pF
t15	NA# Setup Time	7		ns	9-4	
t16	NA# Hold Time	3		ns	9-4	
t17	BS16# Setup Time	7		ns	9-4	
t18	BS16# Hold Time	3		ns	9-4	
t19	READY# Setup Time	9		ns	9-4	
t20	READY# Hold Time	4		ns	9-4	

9.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-5. 25 MHz Intel386™ DX A.C. Characteristics (Continued)

Symbol	Parameter	25 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
t21	D0–D31 Read Setup Time	7		ns	9-4	
t22	D0–D31 Read Hold Time	5		ns	9-4	
t23	HOLD Setup Time	15		ns	9-4	
t24	HOLD Hold Time	3		ns	9-4	
t25	RESET Setup Time	10		ns	9-7	
t26	RESET Hold Time	3		ns	9-7	
t27	NMI, INTR Setup Time	6		ns	9-4	(Note 2)
t28	NMI, INTR Hold Time	6		ns	9-4	(Note 2)
t29	PEREQ, ERROR#, BUSY# Setup Time	6		ns	9-4	(Note 2)
t30	PEREQ, ERROR#, BUSY# Hold Time	5		ns	9-4	(Notes 2, 3)

NOTES:

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3.

	Symbol	Parameter	Min
T_C = 0°C	t30	PEREQ, ERROR#, BUSY# Hold Time	4
T_C = +85°C	t30	PEREQ, ERROR#, BUSY# Hold Time	5

9.5.2 A.C. Specification Tables (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9.6. 20 MHz Intel386™ DX A.C. Characteristics

Symbol	Parameter	20 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	4	20	MHz		Half of CLK2 Frequency
t_1	CLK2 Period	25	125	ns	9-3	
t_{2a}	CLK2 High Time	8		ns	9-3	at 2V
t_{2b}	CLK2 High Time	5		ns	9-3	at (V_{CC} − 0.8V)
t_{3a}	CLK2 Low Time	8		ns	9-3	at 2V
t_{3b}	CLK2 Low Time	6		ns	9-3	at 0.8V
t_4	CLK2 Fall Time		8	ns	9-3	(V_{CC} − 0.8V) to 0.8V
t_5	CLK2 Rise Time		8	ns	9-3	0.8V to (V_{CC} − 0.8V)
t_6	A2–A31 Valid Delay	4	30	ns	9-5	C_L = 120 pF
t_7	A2–A31 Float Delay	4	32	ns	9-6	(Note 1)
t_8	BE0#–BE3#, LOCK# Valid Delay	4	30	ns	9-5	C_L = 75 pF
t_9	BE0#–BE3#, LOCK# Float Delay	4	32	ns	9-6	(Note 1)
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	6	28	ns	9-5	C_L = 75 pF
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	6	30	ns	9-6	(Note 1)
t_{12}	D0–D31 Write Data Valid Delay	4	38	ns	9-5c	C_L = 120 pF
t_{13}	D0–D31 Float Delay	4	27	ns	9-6	(Note 1)
t_{14}	HLDA Valid Delay	6	28	ns	9-6	C_L = 75 pF
t_{15}	NA# Setup Time	9		ns	9-4	
t_{16}	NA# Hold Time	14		ns	9-4	
t_{17}	BS16# Setup Time	13		ns	9-4	
t_{18}	BS16# Hold Time	21		ns	9-4	
t_{19}	READY# Setup Time	12		ns	9-4	
t_{20}	READY# Hold Time	4		ns	9-4	
t_{21}	D0–D31 Read Setup Time	11		ns	9-4	
t_{22}	D0–D31 Read Hold Time	6		ns	9-4	
t_{23}	HOLD Setup Time	17		ns	9-4	
t_{24}	HOLD Hold Time	5		ns	9-4	
t_{25}	RESET Setup Time	12		ns	9-7	

9.5.2 A.C. Specification Tables (Continued)

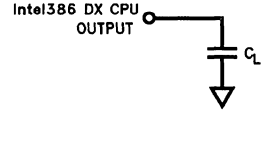
Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +85°C

Table 9-6. 20 MHz Intel386™ DX A.C. Characteristics (Continued)

Symbol	Parameter	20 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
t_{26}	RESET Hold Time	4		ns	9-7	
t_{27}	NMI, INTR Setup Time	16		ns	9-4	(Note 2)
t_{28}	NMI, INTR Hold Time	16		ns	9-4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY# Setup Time	14		ns	9-4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY# Hold Time	5		ns	9-4	(Note 2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

9.5.3 A.C. Test Loads



C_L = 120 pF on A2–A31, D0–D31
C_L = 75 pF on BE0#–BE3#, W/R#, M/IO#, D/C#, ADS#, LOCK#, HLDA
C_L includes all parasitic capacitances.

Figure 9-2. A.C. Test Load

9.5.4 A.C. Timing Waveforms

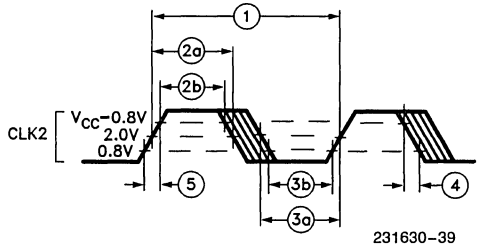


Figure 9-3. CLK2 Timing

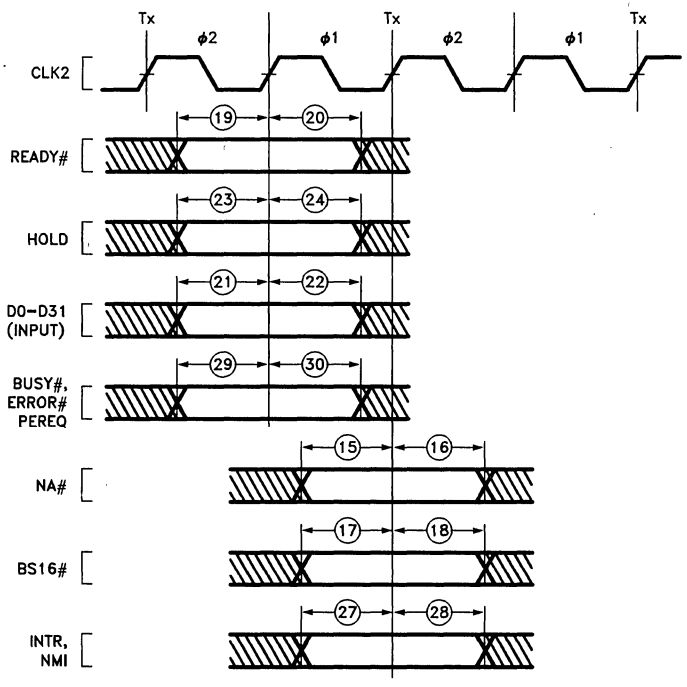


Figure 9-4. Input Setup and Hold Timing

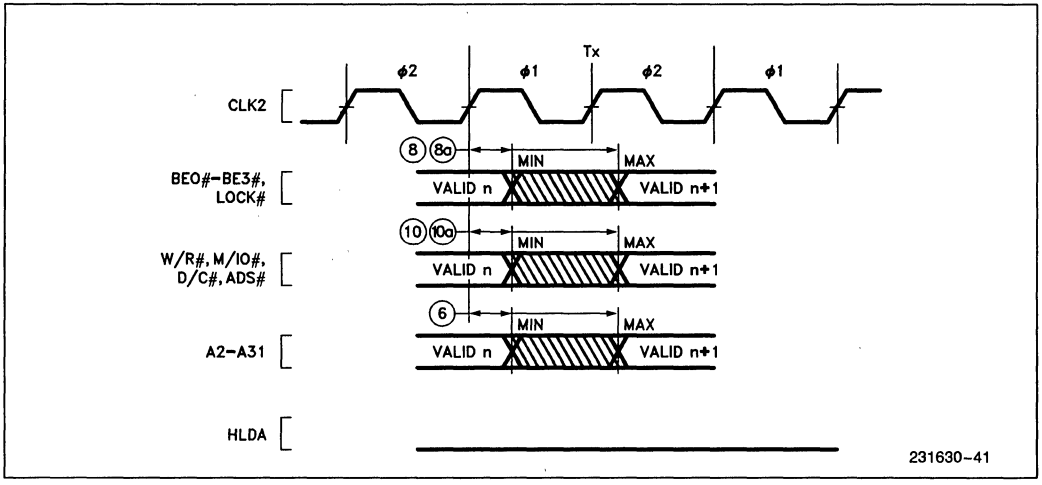


Figure 9-5. Output Valid Delay Timing

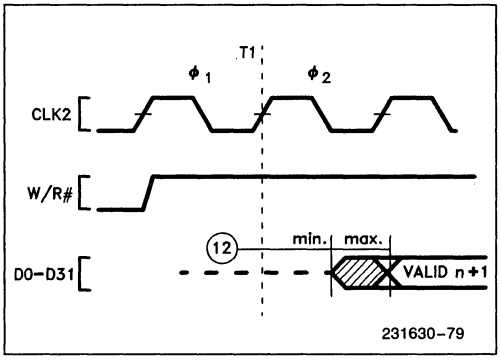


Figure 9-5a. Write Data Valid Delay Timing (25 MHz, 33 MHz)

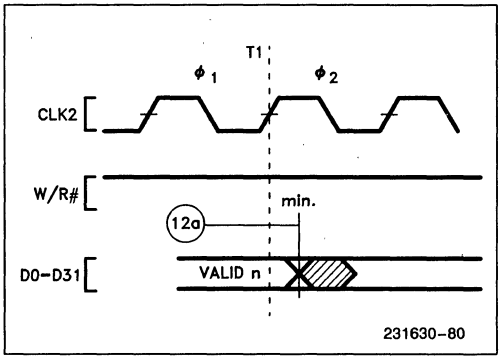


Figure 9-5b. Write Data Hold Timing (25 MHz, 33 MHz)

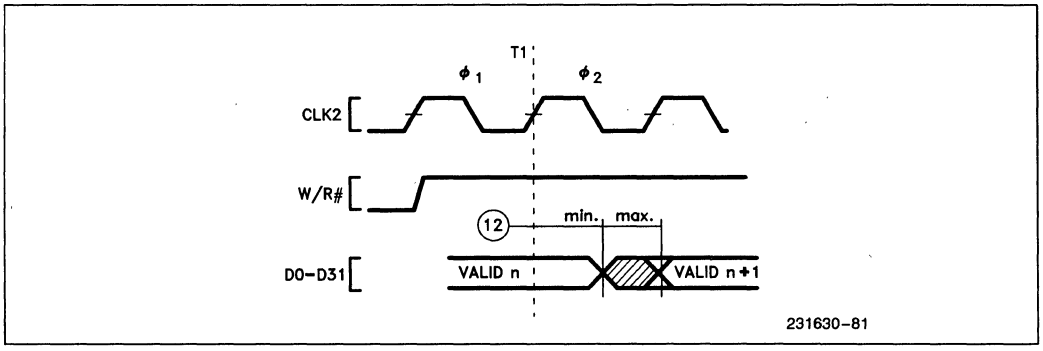
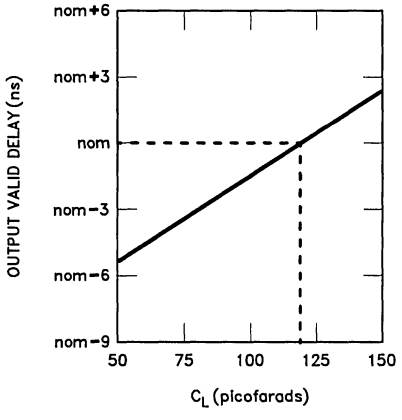


Figure 9-5c. Write Data Valid Delay Timing (20 MHz)

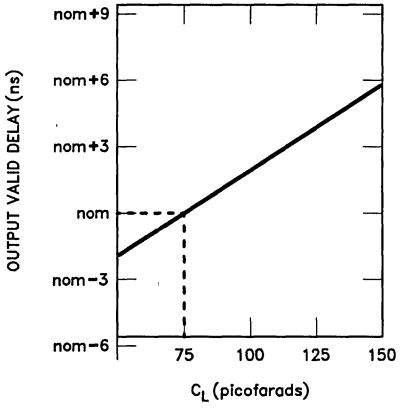
9.5.5 Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 120 pF)



231630-77

NOTE:
This graph will not be linear outside of the C_L range shown.

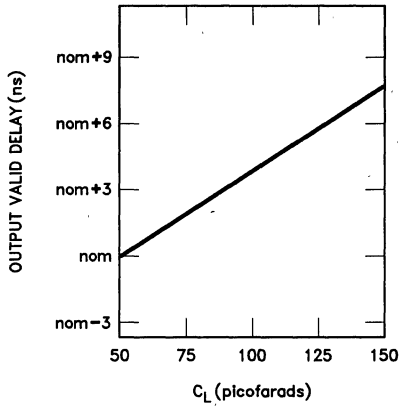
9.5.6 Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 75 pF)



231630-82

NOTE:
This graph will not be linear outside of the C_L range shown.

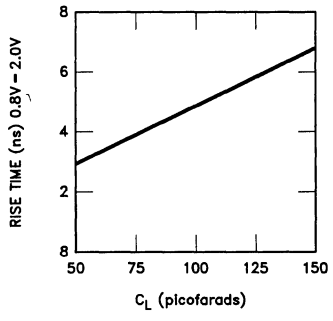
9.5.7 Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 50 pF)



231630-83

NOTE:
This graph will not be linear outside of the C_L range shown.

9.5.8 Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature



231630-78

NOTE:
This graph will not be linear outside of the C_L range shown.

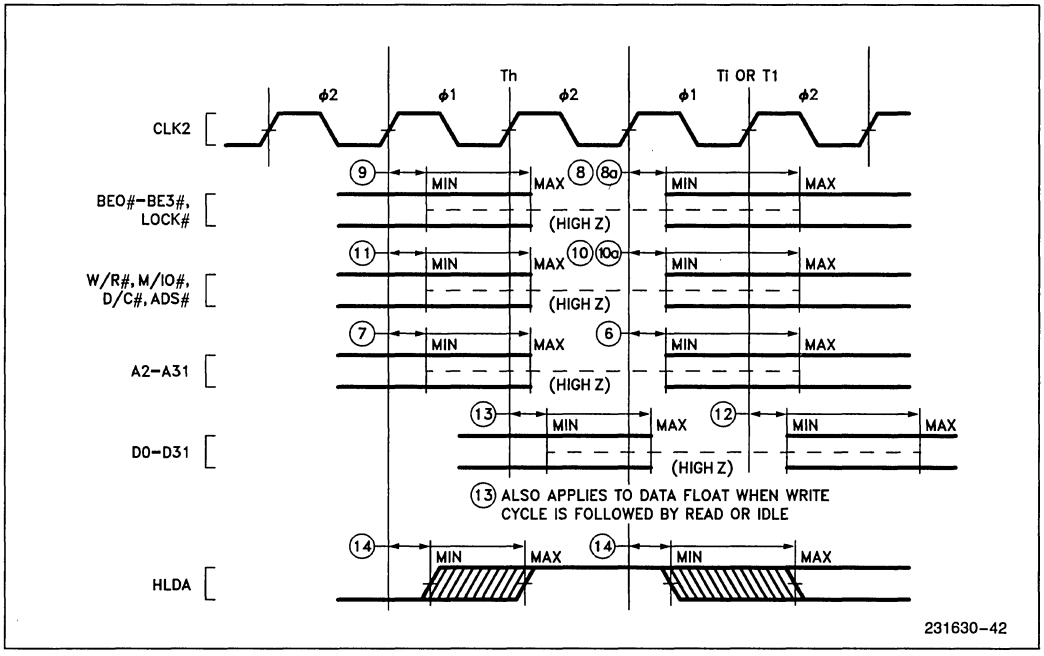
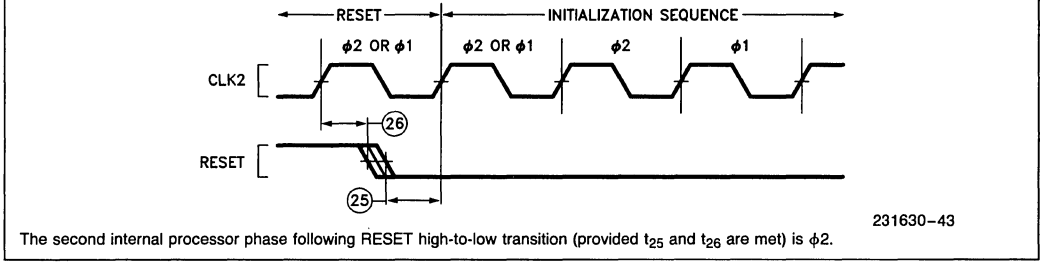


Figure 9-6. Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET high-to-low transition (provided t_{25} and t_{26} are met) is $\phi 2$.

Figure 9-7. RESET Setup and Hold Timing, and Internal Phase

10. REVISION HISTORY

This Intel386 DX data sheet, version -005, contains updates and improvements to previous versions. A revision summary is listed here for your convenience.

The sections significantly revised since version -001 are:

2.9.6	Sequence of exception checking table added.
2.9.7	Instruction restart revised.
2.11.2	TLB testing revised.
2.12	Debugging support revised.
3.1	LOCK prefix restricted to certain instructions.
4.4.3.3	I/O privilege level and I/O permission bitmap added.
Figures 4-15a, 4-15b	I/O permission bitmap added.
4.6.4	Protection and I/O permission bitmap revised.
4.6.6	Entering and leaving virtual 8086 mode through task switches, trap and interrupt gates, and IRET explained.
5.6	Self-test signature stored in EAX.
5.8	Coprocessor interface description added.
5.8.1	Software testing for coprocessor presence added.
Table 6-3	PGA package thermal characteristics added.
7.	Designing for ICE-Intel386 revised.
Figures 7-8, 7-9, 7-10	ICE-Intel386 clearance requirements added.
6.2.3.4	Encoding of 32-bit address mode with no "sib" byte corrected.

The sections significantly revised since version -002 are:

Table 2-5	Interrupt vector assignments updated.
Figure 4-15a	Bit_map_offset must be less than or equal to DFFFH.
Figure 5-28	Intel386 DX outputs remain in their reset state during self-test.
5.7	Component and revision identifier history updated.
9.4	20 MHz D.C. specifications added.
9.5	16 MHz A.C. specifications updated. 20 MHz A.C. specifications added.
Table 6-1	Clock counts updated.

The sections significantly revised since version -003 are:

Table 2-6b	Interrupt priorities 2 and 3 interchanged.
2.9.8	Double page faults do not raise double fault exception.
Figure 4-5	Maximum-sized segments must have segments $Base_{11..0} = 0$.
5.4.3.4	BS16# timing corrected.
Figures 5-16, 5-17, 5-19, 5-22	BS16# timing corrected. BS16# must not be asserted once NA# has been sampled asserted in the current bus cycle.
9.5	16 MHz and 20 MHz A.C. specifications revised. All timing parameters are now guaranteed at 1.5V test levels. The timing parameters have been adjusted to remain compatible with previous 0.8V/2.0V specifications.

The sections significantly revised since version -004 are:

Chapter 4	25 MHz Clock data included.
Table 2-4	Segment Register Selection Rules updated.
5.4.4	Interrupt Acknowledge Cycles discussion corrected.
Table 5-10	Additional Stepping Information added.
Table 9-3	I_{CC} values updated.
9.5.2	Table for 25 MHz A.C. Characteristics added. A.C. Characteristics tables reordered.
Figure 9-5	Output Valid Delay Timing Figure reconfigured. Partial data now provided in additional Figures 9-5a and 9-5b.
Table 6-1	Clock counts updated and formats corrected.

The sections significantly revised since version -005 are:

Table of Contents	Simplified.
Chapter 1	Pin Assignment.
2.3.6	Control Register 0.
Table 2-4	Segment override prefixes possible.
Figure 4-6	Note added.
Figure 4-7	Note added.
5.2.3	Data bus state at end of cycle.
5.2.8.4	Coprocessor error.
5.5.3	Bus activity during and following reset.
Figure 5-28	ERROR#.
Chapter 6	Moved forward in datasheet.
Chapter 7	Moved forward in datasheet.
Chapter 8	Upgraded to chapter.
Table 9-3	25 MHz I_{CC} Typ. value corrected.
Table 9-3	33 MHz D.C. Specifications added.
Table 9-4	33 MHz A.C. Specifications added.
Figure 9-5	t8a and t10a added.
Figure 9-5c	Added.
9.5.6	Added derating for $C_L = 75$ pF.
9.5.7	Added derating for $C_L = 50$ pF.
Figure 9.6	t8a and t10a added.

The sections significantly revised since version -006 are:

2.3.4	Alignment of maximum sized segments.
2.9.8	Double page faults do not raise double fault exception.
5.5.3	ERROR# and BUSY# sampling after RESET.
Figure 5-21	BS16# timing altered.
Figure 5-26	READY# timing altered.
Figure 5-28	ERROR# timing corrected.
6.2.3.1	Corrected Encoding of Register Field Chart.
Chapter 7	Updated ICE-Intel386 DX information.
9.5.2	Remove preliminary stamp on 25 MHz A.C. Specifications.
9.5.2	Remove preliminary stamp on 33 MHz A.C. Specifications.

The sections significantly revised since version -007 are:

Table of Contents	Page numbers revised.
Figure 5-15	BS16# timing altered.
Figure 5-22	Previous cycle, T2 changed to Idle cycle, Ti.
6.1	Note about wait states added.
Table 6-1	Opcodes for AND, OR, and XOR instructions corrected.
Table 6-1	Bits 3, 4, and 5 of the "mod r/m" byte corrected for the LTR instruction.
Table 8-2	Reference to Figure 6-4 should be reference Figure 8-2.
Table 8-2	Note #4 added.

The sections significantly revised since version -008 are:

Table 9-3	20, 25, 33 MHz I_{CC} specifications updated.

Intel386™ DX MICROPROCESSOR 32-BIT CHMOS MICROPROCESSOR WITH INTEGRATED MEMORY MANAGEMENT (PQFP SUPPLEMENT)

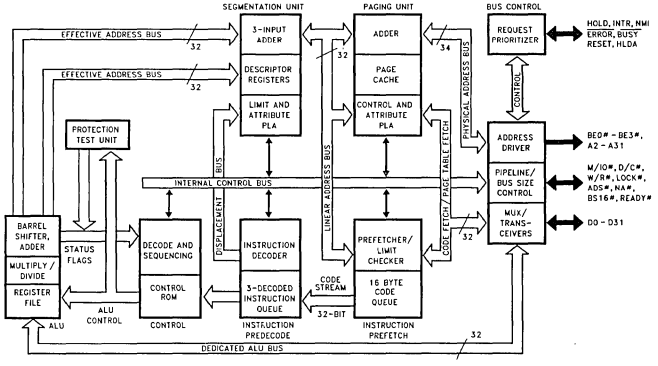
- **Flexible 32-Bit Microprocessor**
  - 8, 16, 32-Bit Data Types
  - 8 General Purpose 32-Bit Registers
- **Very Large Address Space**
  - 4 Gigabyte Physical
  - 64 Terabyte Virtual
  - 4 Gigabyte Maximum Segment Size
- **Integrated Memory Management Unit**
  - Virtual Memory Support
  - Optional On-Chip Paging
  - 4 Levels of Protection
  - Fully Compatible with 80286
- **Object Code Compatible with All 8086 Family Microprocessors**
- **Virtual 8086 Mode Allows Running of 8086 Software in a Protected and Paged System**
- **Hardware Debugging Support**
- **Optimized for System Performance**
  - Pipelined Instruction Execution
  - On-Chip Address Translation Caches
  - 20, 25 and 33 MHz Clock
  - 40, 50 and 66 Megabytes/Sec Bus Bandwidth
- **Numerics Support via Intel387™ DX Math Coprocessor**
- **Complete System Development Support**
  - Software: C, PL/M, Assembler System Generation Tools
  - Debuggers: PSCOPE, ICE™-386
- **High Speed CHMOS IV Technology**
- **132 Pin PQFP Package**
  (See Packaging Specification, Order #231369)

3

The Intel386 DX Microprocessor is an entry-level 32-bit microprocessor designed for single-user applications and operating systems such as MS-DOS and Windows. The 32-bit registers and data paths support 32-bit addresses and data types. The processor addresses up to four gigabytes of physical memory and 64 terabytes (2**46) of virtual memory. The integrated memory management and protection architecture includes address translation registers, multitasking hardware and a protection mechanism to support operating systems. Instruction pipelining, on-chip address translation, ensure short average instruction execution times and maximum system throughput.

The Intel386 DX CPU offers new testability and debugging features. Testability features include a self-test and direct access to the page translation cache. Four new breakpoint registers provide breakpoint traps on code execution or data accesses, for powerful debugging of even ROM-based systems.

Object-code compatibility with all 8086 family members (8086, 8088, 80186, 80188, 80286) means the Intel386 DX offers immediate access to the world's largest microprocessor software base.



241267-1

Intel386™ DX Pipelined 32-Bit Microarchitecture

Intel386™ DX and Intel387™ DX are Trademarks of Intel Corporation.
MS-DOS and Windows are Trademarks of MICROSOFT Corporation.

Intel386™ DX Microprocessor High-Performance 32-Bit CHMOS Microprocessor with Integrated Memory Management (PQFP Supplement)

CONTENTS

CONTENTS

This document should be used in conjunction with the Intel386™ DX Microprocessor data sheet (order number 231630-011, October 1993).

1.0 PIN ASSIGNMENT

The Intel386 DX pinout as viewed from the top side of the component is shown by Figure 1-1.

V_{CC} and GND connections must be made to multiple V_{CC} and V_{SS} (GND) pins. Each V_{CC} and V_{SS} must be connected to the appropriate voltage level.

The circuit board should include V_{CC} and GND planes for power distribution and all V_{CC} and V_{SS} pins must be connected to the appropriate plane.

NOTE:
Pins identified as "N.C." should remain completely unconnected.

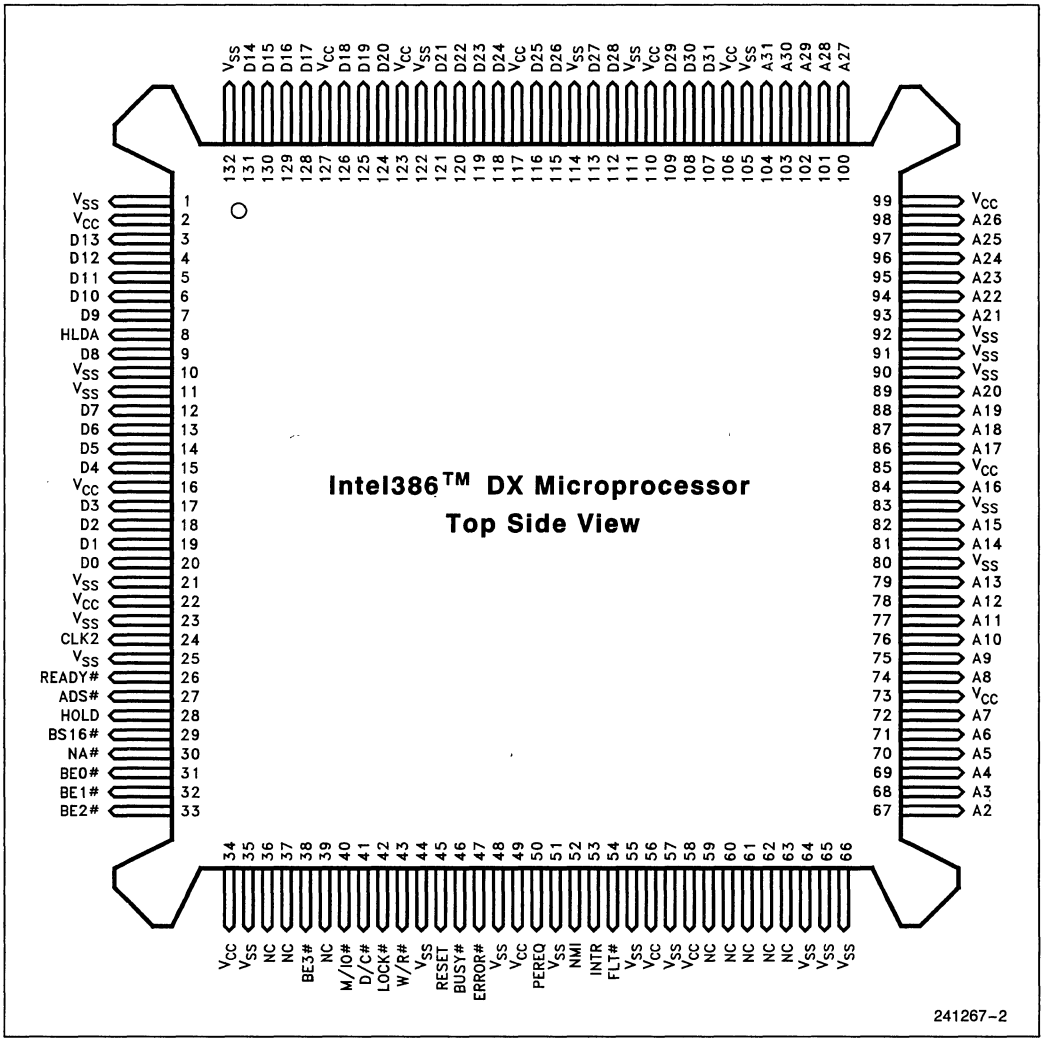


Figure 1-1. Intel386™ DX PQFP Pinout—View from Top Side

Table 1-1. Intel386™ DX PQFP Pinout—Functional Grouping

Address		Data		Control		N/C	V_{SS}	V_{CC}
A2	67	D0	20	ADS#	27	36	1	2
A3	68	D1	19	BE0#	31	37	10	16
A4	69	D2	18	BE1#	32	39	11	22
A5	70	D3	17	BE2#	33	59	21	34
A6	71	D4	15	BE3#	38	60	23	49
A7	72	D5	14	BS16#	29	61	25	56
A8	74	D6	13	BUSY#	46	62	35	58
A9	75	D7	12	CLK2	24	63	44	73
A10	76	D8	9	D/C#	41		48	85
A11	77	D9	7	ERROR#	47		51	99
A12	78	D10	6	FLT#	54		55	106
A13	79	D11	5	HLDA	8		57	110
A14	81	D12	4	HOLD	28		64	117
A15	82	D13	3	INTR	53		65	123
A16	84	D14	131	LOCK#	42		66	127
A17	86	D15	130	M/IO#	40		80	
A18	87	D16	129	NA#	30		83	
A19	88	D17	128	NMI	52		90	
A20	89	D18	126	PEREQ	50		91	
A21	93	D19	125	READY#	26		92	
A22	94	D20	124	RESET	45		105	
A23	95	D21	121	W/R#	43		111	
A24	96	D22	120				114	
A25	97	D23	119				122	
A26	98	D24	118				132	
A27	100	D25	116					
A28	101	D26	115					
A29	102	D27	113					
A30	103	D28	112					
A31	104	D29	109					
		D30	108					
		D31	107					

1.1 Pin Description Table

The following table lists a brief description of each pin on the Intel386 DX. The following definitions are used in these descriptions:

\# The named signal is active LOW.

I Input signal.

O Output signal.

I/O Input and Output signal.

— No electrical connection.

Symbol	Type	Name and Function
CLK2	I	**CLK2** provides the fundamental timing for the Intel386 DX.
$D_{31}-D_0$	I/O	**DATA BUS** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.
$A_{31}-A_2$	O	**ADDRESS BUS** outputs physical memory or port I/O addresses.
BE0#-BE3#	O	**BYTE ENABLES** indicate which data bytes of the data bus take part in a bus cycle.
W/R#	O	**WRITE/READ** is a bus cycle definition pin that distinguishes write cycles from read cycles.
D/C#	O	**DATA/CONTROL** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and instruction fetching.
M/IO#	O	**MEMORY I/O** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles.
LOCK#	O	**BUS LOCK** is a bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active.
ADS#	O	**ADDRESS STATUS** indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BE0#, BE1#, BE2#, BE3# and $A_{31}-A_2$) are being driven at the Intel386 DX pins.
NA#	I	**NEXT ADDRESS** is used to request address pipelining.
READY#	I	**BUS READY** terminates the bus cycle.
BS16#	I	**BUS SIZE 16** input allows direct connection of 32-bit and 16-bit data buses.
HOLD	I	**BUS HOLD REQUEST** input allows another bus master to request control of the local bus.

1.1 Pin Description Table (Continued)

Symbol	Type	Name and Function
HLDA	O	**BUS HOLD ACKNOWLEDGE** output indicates that the Intel386 DX has surrendered control of its local bus to another bus master.
BUSY#	I	**BUSY** signals a busy condition from a processor extension.
ERROR#	I	**ERROR** signals an error condition from a processor extension.
PEREQ	I	**PROCESSOR EXTENSION REQUEST** indicates that the processor extension has data to be transferred by the Intel386 DX.
FLT#	I	**FLOAT** is an input which forces all bidirectional and output signals, including HLDA, to the tri-state condition. This allows the electrically isolated 386 DX PQFP to use On-Circuit Emulation (ONCE) directly on the motherboard. The FLT# pin has an internal pull-up resistor; if the FLT# pin is not used, it should not be connected.
INTR	I	**INTERRUPT REQUEST** is a maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
NMI	I	**NON-MASKABLE INTERRUPT REQUEST** is a non-maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
RESET	I	**RESET** suspends any operation in progress and places the Intel386 DX in a known reset state. See **Interrupt Signals** for additional information.
N/C	—	**NO CONNECT** should always remain unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the Intel386 DX.
V_{CC}	I	**SYSTEM POWER** provides the +5V nominal D.C. supply input.
V_{SS}	I	**SYSTEM GROUND** provides 0V connection from which all inputs and outputs are measured.

1.2 Float Pin Description

Activating the FLT# input floats all Intel386 DX bidirectional and output signals, including HLDA. Asserting FLT# isolates the Intel386 DX microprocessor from the surrounding circuitry.

Packaged in a surface mount PQFP, it cannot be removed from the motherboard when In-Circuit Emulation (ICE) is needed. The FLT# input allows the Intel386 CPU to be electrically isolated from the surrounding circuitry. This allows connection of an emulator to the processor without removing it from the PCB. This method of emulation is referred to as ON-Circuit Emulation (ONCE).

ENTERING AND EXITING FLOAT

FLT# is an asynchronous, active-low input. It is recognized on the rising edge of CLK2. When recognized, it aborts the current bus cycle and floats the outputs of the processor (Figure 1-2). FLT# must be held low for a minimum of 16 CLK2 cycles. Reset should be asserted and held asserted until after FLT# is deasserted. This will ensure that the Intel386 DX CPU will exit float in a valid state.

Asserting the FLT# input unconditionally aborts the current bus cycle and forces the processor into the FLOAT mode, and is therefore not guaranteed to enter FLOAT in a valid state. After deactivating FLT#, the processor is not guaranteed to exit FLOAT mode in a valid state. This is not a problem as the FLT# pin is meant to be used only during ONCE. After exiting FLOAT, the processor must be reset to return it to a valid state. Reset should be asserted before FLT# is deasserted. This will ensure that the processor will exit float in a valid state.

FLT# has an internal pull-up resistor, and if it is not used it should be unconnected.

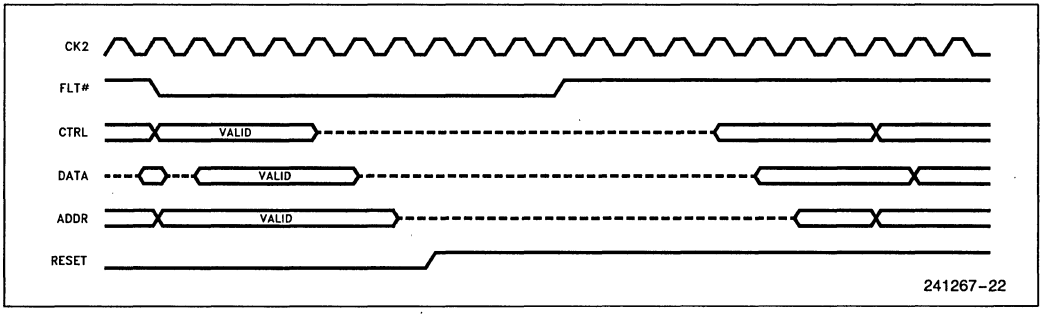


Figure 1-2. Entering and Exiting, FLT#

2.0 MECHANICAL DATA

2.1 Package Dimensions

The Intel386 DX is available in a 132 lead plastic quad flat pack (PQFP) package. Table 2.1 and Figures 2.1–2.5 show the physical dimensions of this package.

Table 2.1. Intel Case Outline Dimensions for 132 Lead Plastic Quad Flat Pack 0.025 Inch Pitch

Symbol	Description	Inch		mm	
		Min	Max	Min	Max
A	Package Height	0.160	0.170	4.06	4.32
A1	Standoff	0.020	0.030	0.51	0.76
D, E	Terminal Dimension	1.075	1.085	27.31	27.56
D1, E1	Package Body	0.947	0.953	24.05	24.21
D2, E2	Bumper Distance	1.097	1.103	27.86	28.02
D3, E3	Lead Dimension	0.800 REF		20.32 REF	
L1	Foot Length	0.020	0.030	0.51	0.76
Issue	IWS Preliminary 1/15/87				

Symbol List

Letter or Symbol	Description of Dimensions
A	Package Height: Distance from Seating Plane to Highest Point of Body
A1	Standoff: Distance from Seating Plane to Base Plane
D/E	Overall Package Dimension: Lead Tip to Lead Tip
D1/E1	Plastic Body Dimension
D2/E2	Bumper Distance
D3/E3	Footprint
L1	Foot Length

NOTES:
1. All dimensions and tolerances conform to ANSI Y14.5M-1982.
2. Datum plane H located at the mold parting line and coincident with the bottom of the lead where lead exits plastic body.
3. Datums A B and D to be determined where center leads exit plastic body at datum plane H.
4. Controlling Dimension, Inch.
5. Dimensions D1, D2, E1, and E2 are measured at the mold parting line and do not include mold protrusion. Allowable mold protrusion is 0.18 mm (0.007 in) per side.
6. Pin 1 identifier is located within one of the two zones indicated.

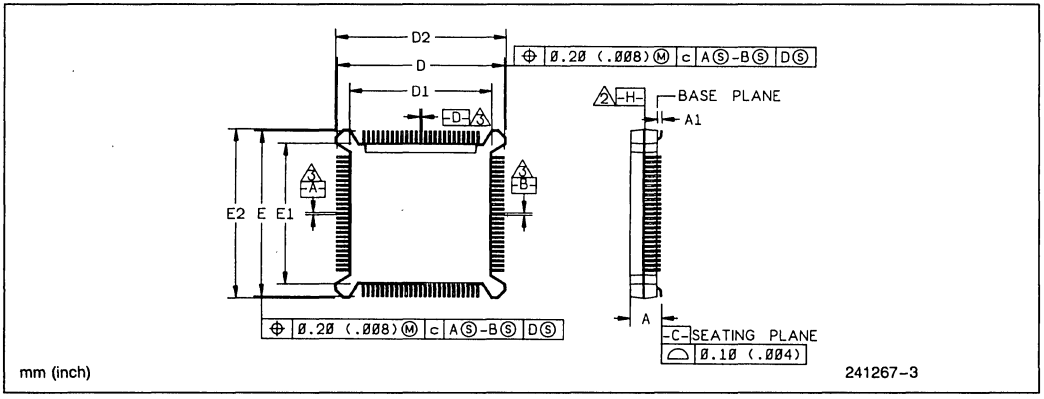


Figure 2.1. Principal Dimensions and Datums

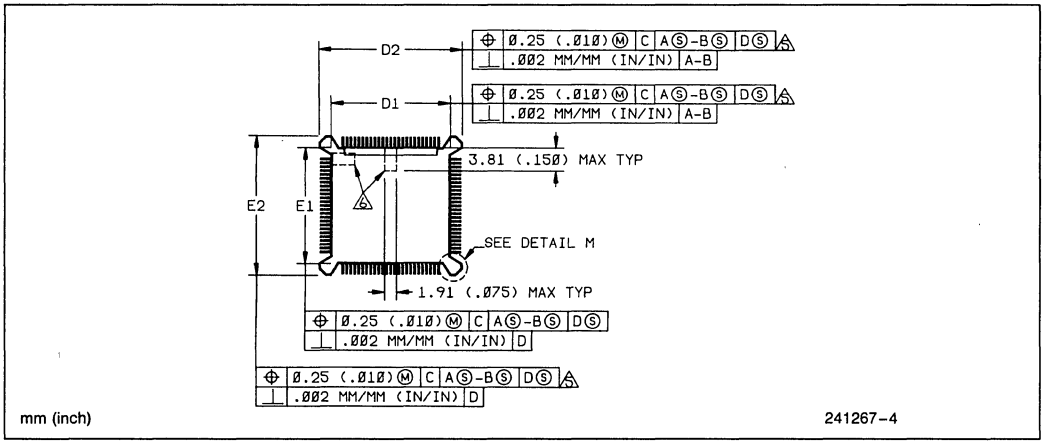


Figure 2.2. Molded Details

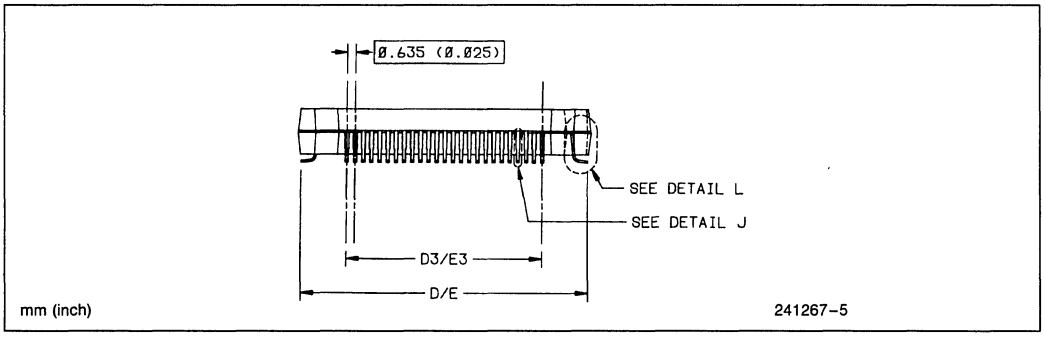


Figure 2.3. Terminal Details

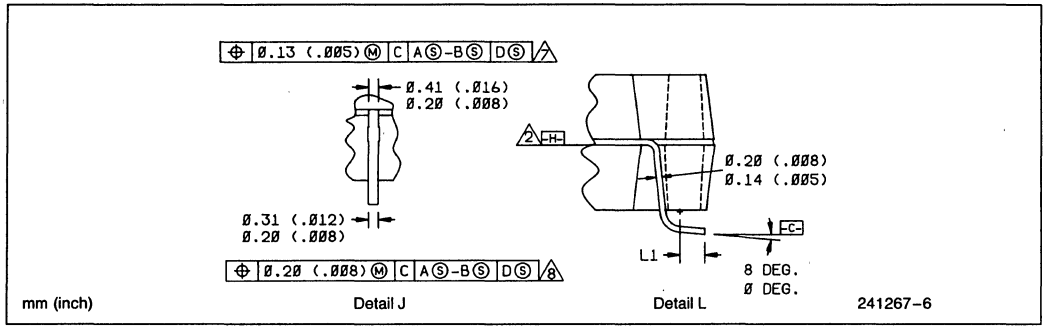


Figure 2.4. Typical Lead

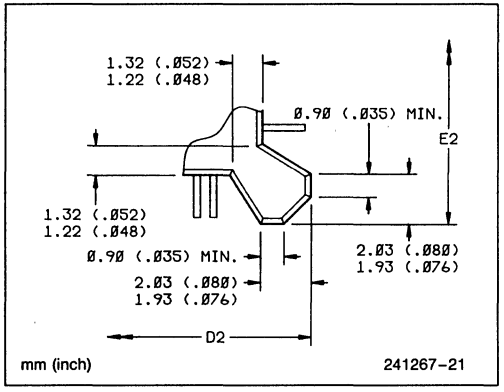


Figure 2.5. Detail M

2.2 Package Thermal Specifications

The Intel386 DX Microprocessor is specified for operation when case temperature is within the range of 0°C–100°C. The case temperature may be measured in any environment, to determine whether the Intel386 DX Microprocessor is within specified operating range. The case temperature should be measured at the center of the top surface.

The ambient temperature is guaranteed as long as T_C is not violated. The ambient temperature can be calculated from the θ_{jc} and θ_{ja} from the following equations:

$$T_j = T_c + P^*\theta_{jc}$$

$$T_a = T_j - P^*\theta_{ja}$$

$$T_c = T_a + P^*[\theta_{ja} - \theta_{jc}]$$

Values for θ_{ja} and θ_{jc} are given in Table 2.2 for the 100 lead fine pitch. θ_{ja} is given at various airflows. Table 2.3 shows the maximum T_a allowable (without exceeding T_c) at various airflows.

Table 2.2. Thermal Resistances (°C/Watt) θ_{jc} and θ_{ja}

Package	θjc	θ_{ja} versus Airflow - ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
132 Lead PQFP	9.0	31.0	24.5	21.5	19.0	17.0	16.0

Table 2.3. Maximum T_a at various airflows

Package	Frequency	T_A(°C) versus Airflow - ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
132 Lead PQFP	20 MHz	71	79	83	87	89	90
	25 MHz	65	75	80	84	87	89
	33 MHz	60	72	78	82	85	87

NOTE:
The numbers in Table 2.3 were calculated using worst case I_{CC} at T_C = 100°C with the outputs unloaded.

3

3.0 D.C./A.C. SPECIFICATIONS

Table 3-1. Maximum Ratings

Parameter	Intel386™ DX 20, 25, 33 MHz Maximum Rating
Storage Temperature	−65°C to +150°C
Case Temperature Under Bias	−65°C to +110°C
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V
Voltage on Other Pins	−0.5V to V_{CC} + 0.5V

Table 3-1 is a stress rating only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in **3.1 D.C. Specifications** and **3.2 A.C. Specifications**.

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the Intel386 DX contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

3.1 D.C. Specifications

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +100°C

Table 3-2. Intel386™ DX D.C. Characteristics

Symbol	Parameter	Intel386™ DX 20 MHz, 25 MHz, 33 MHz		Unit	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	CLK2 Input High Voltage 20 MHz 25 MHz and 33 MHz	 V_{CC} − 0.8 3.7	 V_{CC} + 0.3 V_{CC} + 0.3	 V V	
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A2–A31, D0–D31 I_{OL} = 5 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA		 0.45 0.45	 V V	
V_{OH}	Output High Voltage I_{OH} = 1 mA: A2–A31, D0–D31 I_{OH} = 0.9 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	 2.4 2.4		 V V	
I_{LI}	Input Leakage Current (For All Pins except BS16#, PEREQ, BUSY#, and ERROR#)		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{IH}	Input Leakage Current (PEREQ Pin)		200	μA	V_{IH} = 2.4V (Note 2)
I_{IL}	Input Leakage Current (BS16#, BUSY#, and ERROR# Pins)		−400	μA	V_{IL} = 0.45 (Note 3)
I_{LO}	Output Leakage Current		±15	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	Supply Current CLK2 = 40 MHz: with 20 MHz Intel386™ DX CLK2 = 50 MHz: with 25 MHz Intel386™ DX CLK2 = 66 MHz: with 33 MHz Intel386™ DX		 260 320 390	 mA mA mA	(Note 4) I_{CC} Typ. = 200 mA I_{CC} Typ. = 240 mA I_{CC} Typ. = 300 mA
C_{IN}	Input or I/O Capacitance		10	pF	F_C = 1 MHz
C_{OUT}	Output Capacitance		12	pF	F_C = 1 MHz
C_{CLK}	CLK2 Capacitance		20	pF	F_C = 1 MHz

NOTES:
1. The min value, −0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. BS16#, BUSY# and ERROR# inputs each have an internal pullup resistor.
4. CHMOS IV Technology.

3.2 A.C. SPECIFICATIONS

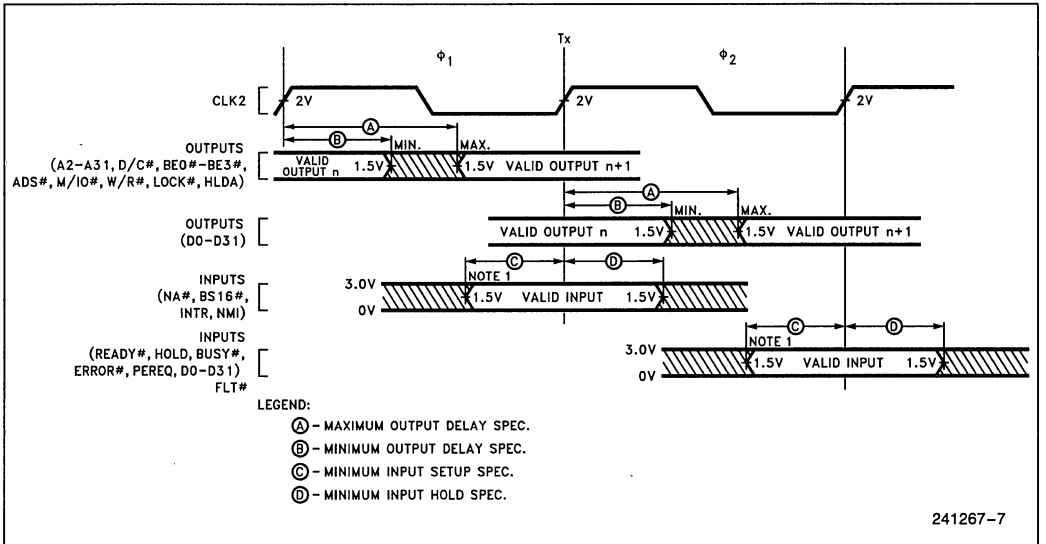
3.2.1 A.C. Spec Definitions

The A.C. specifications, given in Tables 3-3, 3-4, and 3-5, consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the CLK2 rising edge crossing the 2.0V level.

A.C. spec measurement is defined by Figure 3-1. Inputs must be driven to the voltage levels indicated by Figure 3-1 when A.C. specifications are measured. Intel386 DX output delays are specified with minimum and maximum limits, measured as shown.

The minimum Intel386 DX delay times are hold times provided to external circuitry. Intel386 DX input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct Intel386 DX operation.

Outputs NA#, W/R#, D/C#, M/IO#, LOCK#, BE0#–BE3#, A2–A31 and HLDA only change at the beginning of phase one. D0–D31 (write cycles) only change at the beginning of phase two. The READY#, HOLD, BUSY#, ERROR#, PEREQ and D0–D31 (read cycles) inputs are sampled at the beginning of phase one. The NA#, BS16#, INTR and NMI inputs are sampled at the beginning of phase two.



NOTES:
1. Input waveforms have tr ≤ 2.0 ns from 0.8V to 2.0V.
2. See section 9.5.8 for typical output rise time versus load capacitance.

Figure 3-1. Drive Levels and Measurement Points for A.C. Specifications

3.2.2 A.C. SPECIFICATION TABLES

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +100°C

Table 3-3. 33 MHz Intel386™ DX A.C. Characteristics

Symbol	Parameter	33 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	8	33.3	MHz		Half of CLK2 Frequency
t1	CLK2 Period	15.0	62.5	ns	3-3	
t2a	CLK2 High Time	6.25		ns	3-3	at 2V
t2b	CLK2 High Time	4.5		ns	3-3	at 3.7V
t3a	CLK2 Low Time	6.25		ns	3-3	at 2V
t3b	CLK2 Low Time	4.5		ns	3-3	at 0.8V
t4	CLK2 Fall Time		4	ns	3-3	3.7V to 0.8V (Note 3)
t5	CLK2 Rise Time		4	ns	3-3	0.8V to 3.7V (Note 3)
t6	A2–A31 Valid Delay	4	15	ns	3-5	C_L = 50 pF
t7	A2–A31 Float Delay	4	20	ns	3-6	(Note 1)
t8	BE0#–BE3#, LOCK# Valid Delay	4	15	ns	3-5	C_L = 50 pF
t9	BE0#–BE3#, LOCK# Float Delay	4	20	ns	3-6	(Note 1)
t10	W/R#, M/IO#, D/C#, Valid Delay	4	15	ns	3-5	C_L = 50 pF
t10a	ADS# Valid Delay	4	14.5	ns	3-5	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	20	ns	3-6	(Note 1)
t12	D0–D31 Write Data Valid Delay	7	24	ns	3-5a	C_L = 50 pF, (Note 4)
t12a	D0–D31 Write Data Hold Time	2			3-5b	C_L = 50 pF
t13	D0–D31 Float Delay	4	17	ns	3-6	(Note 1)
t14	HLDA Valid Delay	4	20	ns	3-6	C_L = 50 pF
t15	NA# Setup Time	5		ns	3-4	
t16	NA# Hold Time	2		ns	3-4	
t17	BS16# Setup Time	5		ns	3-4	
t18	BS16# Hold Time	2		ns	3-4	
t19	READY# Setup Time	7		ns	3-4	
t20	READY# Hold Time	4		ns	3-4	

3.2.2 A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +100°C

Table 3-3. 33 MHz Intel386™ DX A.C. Characteristics (Continued)

Symbol	Parameter	33 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
t21	D0–D31 Read Setup Time	5		ns	3-4	
t22	D0–D31 Read Hold Time	3		ns	3-4	
t23	HOLD Setup Time	11		ns	3-4	
t24	HOLD Hold Time	2		ns	3-4	
t25	RESET Setup Time	5		ns	3-7	
t26	RESET Hold Time	2		ns	3-7	
t27	NMI, INTR Setup Time	5		ns	3-4	(Note 2)
t28	NMI, INTR Hold Time	5		ns	3-4	(Note 2)
t29	PEREQ, ERROR#, FLT#, BUSY# Setup Time	5		ns	3-4	(Note 2)
t30	PEREQ, ERROR#, FLT#, BUSY# Hold Time	4		ns	3-4	(Note 2)

NOTES:

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.
4. Min. time not 100% tested.

3.2.2 A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +100°C

Table 3-4. 25 MHz Intel386™ DX A.C. Characteristics

Symbol	Parameter	25 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	4	25	MHz		Half of CLK2 Frequency
t1	CLK2 Period	20	125	ns	3-3	
t2a	CLK2 High Time	7		ns	3-3	at 2V
t2b	CLK2 High Time	4		ns	3-3	at 3.7V
t3a	CLK2 Low Time	7		ns	3-3	at 2V
t3b	CLK2 Low Time	5		ns	3-3	at 0.8V
t4	CLK2 Fall Time		7	ns	3-3	3.7V to 0.8V
t5	CLK2 Rise Time		7	ns	3-3	0.8V to 3.7V
t6	A2–A31 Valid Delay	4	21	ns	3-5	C_L = 50 pF
t7	A2–A31 Float Delay	4	30	ns	3-6	(Note 1)
t8	BE0#–BE3# Valid Delay	4	24	ns	3-5	C_L = 50 pF
t8a	LOCK# Valid Delay	4	21	ns	3-5	C_L = 50 pF
t9	BE0#–BE3#, LOCK# Float Delay	4	30	ns	3-6	(Note 1)
t10	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	21	ns	3-5	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	30	ns	3-6	(Note 1)
t12	D0–D31 Write Data Valid Delay	7	27	ns	3-5a	C_L = 50 pF
t12a	D0–D31 Write Data Hold Time	2			3-5b	C_L = 50 pF
t13	D0–D31 Float Delay	4	22	ns	3-6	(Note 1)
t14	HLDA Valid Delay	4	22	ns	3-6	C_L = 50 pF
t15	NA# Setup Time	7		ns	3-4	
t16	NA# Hold Time	3		ns	3-4	
t17	BS16# Setup Time	7		ns	3-4	
t18	BS16# Hold Time	3		ns	3-4	
t19	READY# Setup Time	9		ns	3-4	
t20	READY# Hold Time	4		ns	3-4	

3.2.2 A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +100°C

Table 3-4. 25 MHz Intel386™ DX A.C. Characteristics (Continued)

Symbol	Parameter	25 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
t21	D0–D31 Read Setup Time	7		ns	3-4	
t22	D0–D31 Read Hold Time	5		ns	3-4	
t23	HOLD Setup Time	15		ns	3-4	
t24	HOLD Hold Time	3		ns	3-4	
t25	RESET Setup Time	10		ns	3-7	
t26	RESET Hold Time	3		ns	3-7	
t27	NMI, INTR Setup Time	6		ns	3-4	(Note 2)
t28	NMI, INTR Hold Time	6		ns	3-4	(Note 2)
t29	PEREQ, ERROR#, FLT#, BUSY# Setup Time	6		ns	3-4	(Note 2)
t30	PEREQ, ERROR#, FLT#, BUSY# Hold Time	5		ns	3-4	(Notes 2, 3)

3

NOTES:

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3.

	Symbol	Parameter	Min
T_C = 0°C	t30	PEREQ, ERROR#, FLT#, BUSY# Hold Time	4
T_C = +100°C	t30	PEREQ, ERROR#, FLT#, BUSY# Hold Time	5

3.2.2 A.C. SPECIFICATION TABLES (Continued)

Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +100°C

Table 3-5. 20 MHz Intel386™ DX A.C. Characteristics

Symbol	Parameter	20 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
	Operating Frequency	4	20	MHz		Half of CLK2 Frequency
t_1	CLK2 Period	25	125	ns	3-3	
t_{2a}	CLK2 High Time	8		ns	3-3	at 2V
t_{2b}	CLK2 High Time	5		ns	3-3	at (V_{CC} − 0.8V)
t_{3a}	CLK2 Low Time	8		ns	3-3	at 2V
t_{3b}	CLK2 Low Time	6		ns	3-3	at 0.8V
t_4	CLK2 Fall Time		8	ns	3-3	(V_{CC} − 0.8V) to 0.8V
t_5	CLK2 Rise Time		8	ns	3-3	0.8V to (V_{CC} − 0.8V)
t_6	A2–A31 Valid Delay	4	30	ns	3-5	C_L = 120 pF
t_7	A2–A31 Float Delay	4	32	ns	3-6	(Note 1)
t_8	BE0#–BE3#, LOCK# Valid Delay	4	30	ns	3-5	C_L = 75 pF
t_9	BE0#–BE3#, LOCK# Float Delay	4	32	ns	3-6	(Note 1)
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	6	28	ns	3-5	C_L = 75 pF
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	6	30	ns	3-6	(Note 1)
t_{12}	D0–D31 Write Data Valid Delay	4	38	ns	3-5c	C_L = 120 pF
t_{13}	D0–D31 Float Delay	4	27	ns	3-6	(Note 1)
t_{14}	HLDA Valid Delay	6	28	ns	3-6	C_L = 75 pF
t_{15}	NA# Setup Time	9		ns	3-4	
t_{16}	NA# Hold Time	14		ns	3-4	
t_{17}	BS16# Setup Time	13		ns	3-4	
t_{18}	BS16# Hold Time	21		ns	3-4	
t_{19}	READY# Setup Time	12		ns	3-4	
t_{20}	READY# Hold Time	4		ns	3-4	
t_{21}	D0–D31 Read Setup Time	11		ns	3-4	
t_{22}	D0–D31 Read Hold Time	6		ns	3-4	
t_{23}	HOLD Setup Time	17		ns	3-4	
t_{24}	HOLD Hold Time	5		ns	3-4	
t_{25}	RESET Setup Time	12		ns	3-7	

3.2.2 A.C. SPECIFICATION TABLES (Continued)

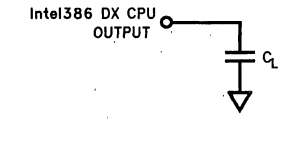
Functional Operating Range: V_{CC} = 5V ±5%; T_{CASE} = 0°C to +100°C

Table 3-5. 20 MHz Intel386™ DX A.C. Characteristics (Continued)

Symbol	Parameter	20 MHz Intel386™ DX		Unit	Ref. Fig.	Notes
		Min	Max			
t_{26}	RESET Hold Time	4		ns	3-7	
t_{27}	NMI, INTR Setup Time	16		ns	3-4	(Note 2)
t_{28}	NMI, INTR Hold Time	16		ns	3-4	(Note 2)
t_{29}	PEREQ, ERROR#, FLT#, BUSY# Setup Time	14		ns	3-4	(Note 2)
t_{30}	PEREQ, ERROR#, FLT#, BUSY# Hold Time	5		ns	3-4	(Note 2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

3.2.3 A.C. TEST LOADS



C_L = 120 pF on A2–A31, D0–D31
C_L = 75 pF on BE0#–BE3#, W/R#, M/IO#, D/C#, ADS#, LOCK#, HLDA
C_L includes all parasitic capacitances.

Figure 3-2. A.C. Test Load

3.2.4 A.C. TIMING WAVEFORMS

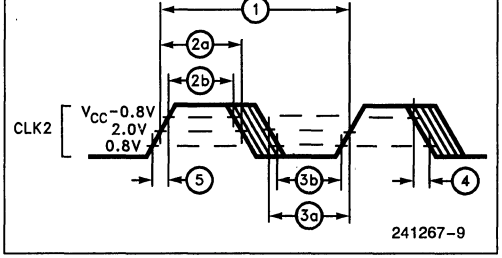


Figure 3-3. CLK2 Timing

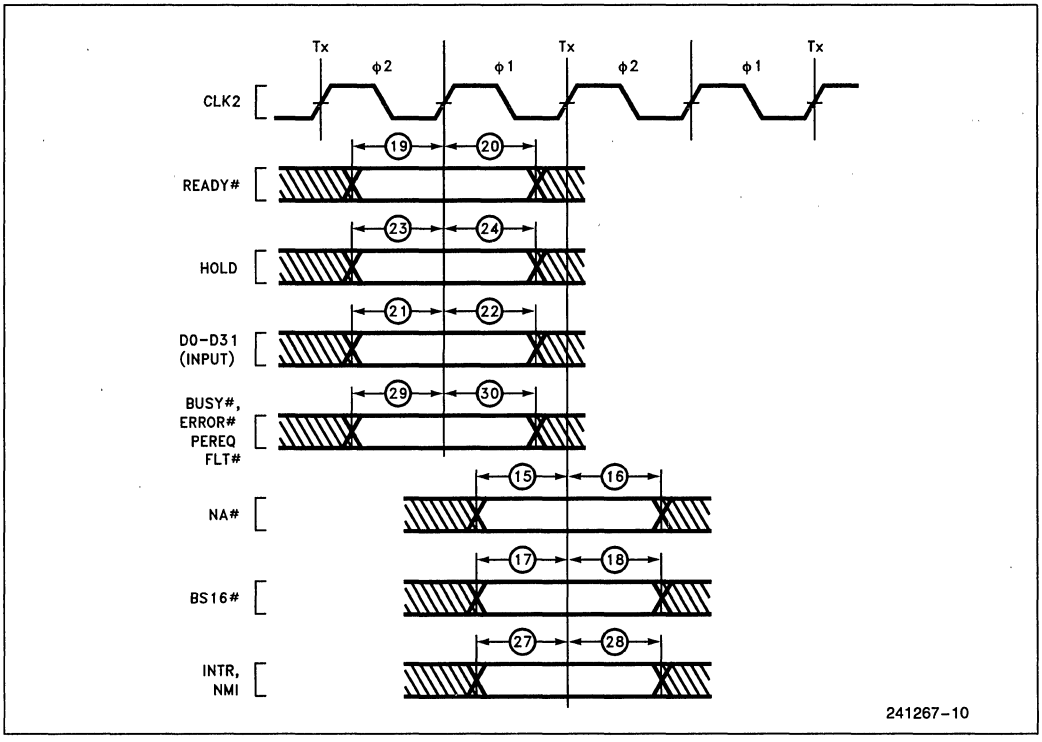


Figure 3-4. Input Setup and Hold Timing

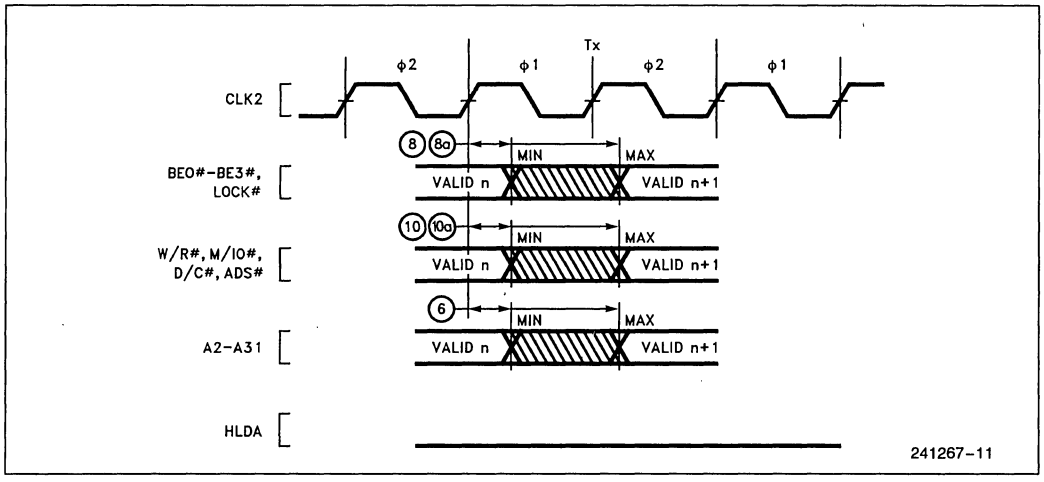


Figure 3-5. Output Valid Delay Timing

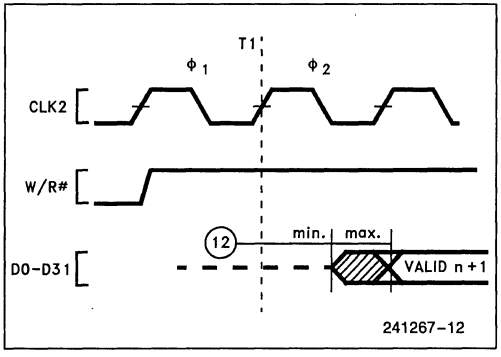


Figure 3-5a. Write Data Valid Delay Timing (25 MHz, 33 MHz)

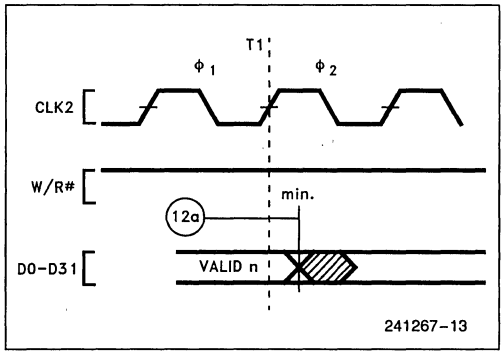


Figure 3-5b. Write Data Hold Timing (25 MHz, 33 MHz)

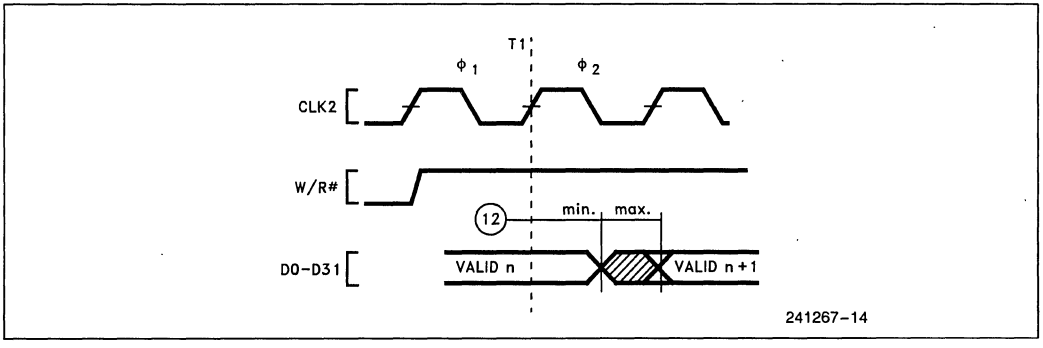
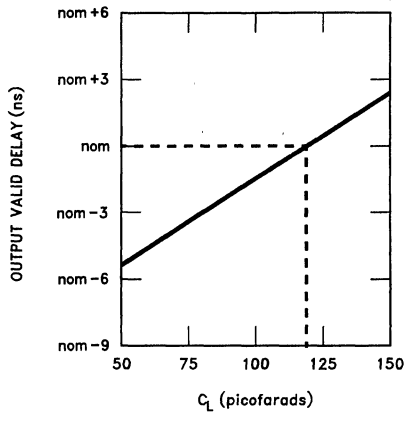


Figure 3-5c. Write Data Valid Delay Timing (20 MHz)

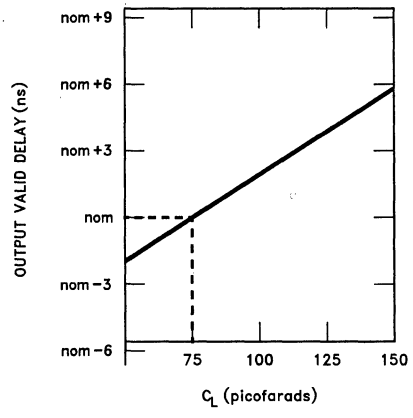
3.2.5 TYPICAL OUTPUT VALID DELAY VERSUS LOAD CAPACITANCE AT MAXIMUM OPERATING TEMPERATURE (C_L = 120 pF)



241267-15

NOTE:
This graph will not be linear outside of the C_L range shown.

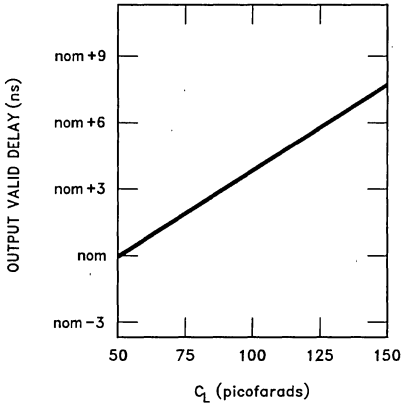
3.2.6 TYPICAL OUTPUT VALID DELAY VERSUS LOAD CAPACITANCE AT MAXIMUM OPERATING TEMPERATURE (C_L = 75 pF)



241267-16

NOTE:
This graph will not be linear outside of the C_L range shown.

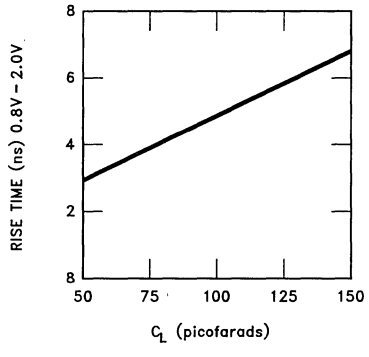
3.2.7 TYPICAL OUTPUT VALID DELAY VERSUS LOAD CAPACITANCE AT MAXIMUM OPERATING TEMPERATURE (C_L = 50 pF)



241267-17

NOTE:
This graph will not be linear outside of the C_L range shown.

3.2.8 TYPICAL OUTPUT RISE TIME VERSUS LOAD CAPACITANCE AT MAXIMUM OPERATING TEMPERATURE



241267-18

NOTE:
This graph will not be linear outside of the C_L range shown.

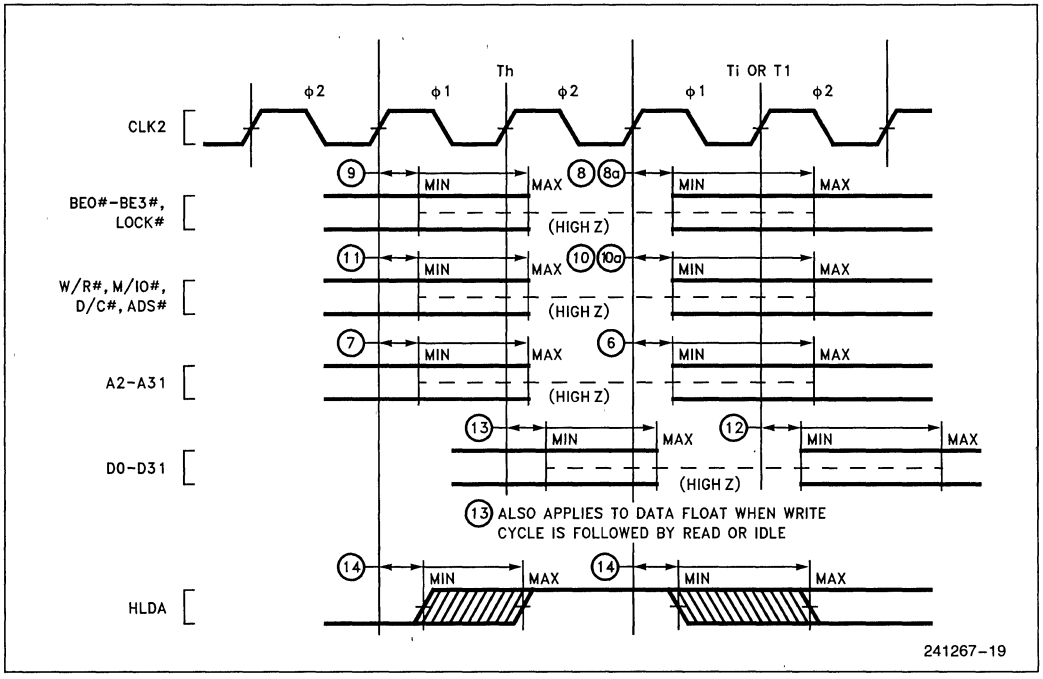
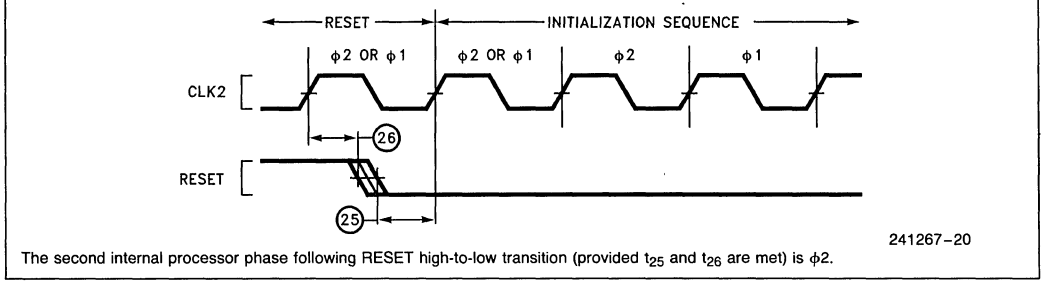


Figure 3-6. Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET high-to-low transition (provided t_{25} and t_{26} are met) is φ2.

Figure 3-7. RESET Setup and Hold Timing, and Internal Phase

Intel387™ DX MATH COPROCESSOR

- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Expands Intel386™ DX CPU Data Types to Include 32-, 64-, 80-Bit Floating Point, 32-, 64-Bit Integers and 18-Digit BCD Operands**
- **Directly Extends Intel386™ DX CPU Instruction Set to Include Trigonometric, Logarithmic, Exponential and Arithmetic Instructions for All Data Types**
- **Upward Object-Code Compatible from 8087 and 80287**
- **Full-Range Transcendental Operations for SINE, COSINE, TANGENT, ARCTANGENT and LOGARITHM**
- **Built-In Exception Handling**
- **Operates Independently of Real, Protected and Virtual-8086 Modes of the Intel386™ DX Microprocessor**
- **Eight 80-Bit Numeric Registers, Usable as Individually Addressable General Registers or as a Register Stack**
- **Available in 68-Pin PGA Package**
- **One Version Supports 16 MHz–33 MHz Speeds**

(See Packaging Spec: Order #231369)

3

The Intel387™ DX Math CoProcessor (MCP) is an extension of the Intel386™ microprocessor architecture. The combination of the Intel387 DX MCP with the Intel386™ DX Microprocessor dramatically increases the processing speed of computer application software which utilize mathematical operations. This makes an ideal computer workstation platform for applications such as financial modeling and spreadsheets, CAD/CAM, or graphics.

The Intel387 DX Math CoProcessor adds over seventy mnemonics to the Intel386 DX Microprocessor instruction set. Specific Intel387 DX MCP math operations include logarithmic, arithmetic, exponential, and trigonometric functions. The Intel387 DX MCP supports integer, extended integer, floating point and BCD data formats, and fully conforms to the ANSI/IEEE floating point standard.

The Intel387 DX Math CoProcessor is object code compatible with the Intel387 SX MCP, and upward object code compatible from the 80287 and 8087 math coprocessors. Object code for Intel386 DX/Intel387 DX is also compatible with the Intel486™ microprocessor. The Intel387 DX MCP is manufactured on 1 micron, CHMOS IV technology and packaged in a 68-pin PGA package.

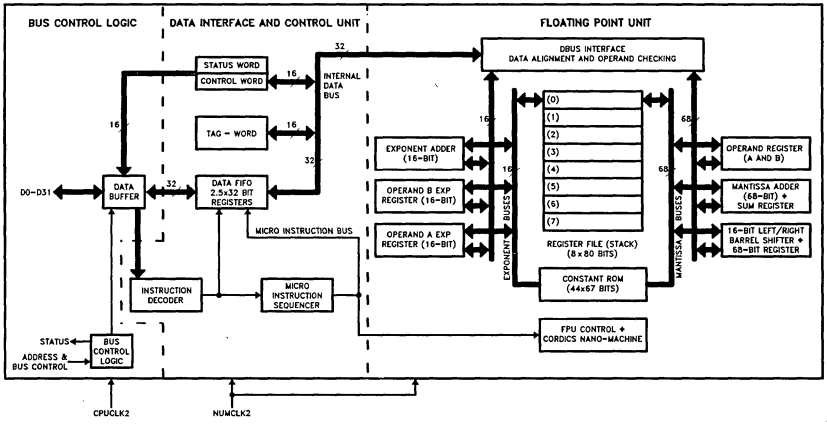


Figure 0.1. Intel387™ DX Math CoProcessor Block Diagram

Intel387™ DX Math CoProcessor

CONTENTS

CONTENTS

FIGURES

CONTENTS

PAGE

FIGURES (Continued)

TABLES

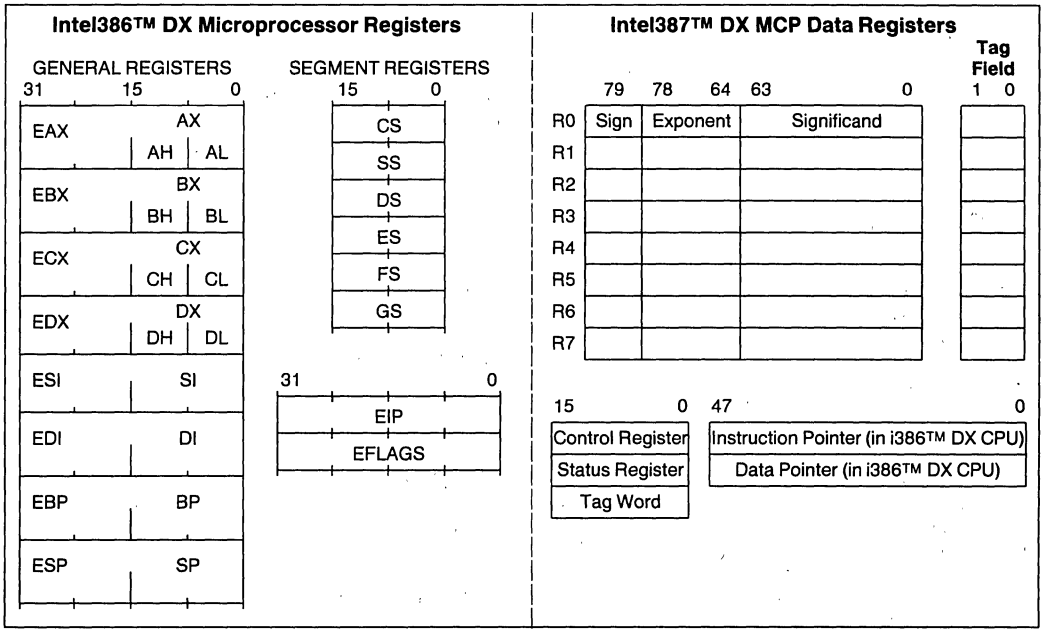


Figure 1.1. Intel386™ DX Microprocessor and Intel387™ DX Math Coprocessor Register Set

1.0 FUNCTIONAL DESCRIPTION

The Intel387™ DX Math Coprocessor provides arithmetic instructions for a variety of numeric data types in Intel386™ DX Microprocessor systems. It also executes numerous built-in transcendental functions (e.g. tangent, sine, cosine, and log functions). The Intel387 DX MCP effectively extends the register and instruction set of a Intel386 DX Microprocessor system for existing data types and adds several new data types as well. Figure 1.1 shows the model of registers visible to programs. Essentially, the Intel387 DX MCP can be treated as an additional resource or an extension to the Intel386 DX Microprocessor. The Intel386 DX Microprocessor together with a Intel387 DX MCP can be used as a single unified system.

The Intel387 DX MCP works the same whether the Intel386 DX Microprocessor is executing in real-address mode, protected mode, or virtual-8086 mode. All memory access is handled by the Intel386 DX Microprocessor; the Intel387 DX MCP merely operates on instructions and values passed to it by the Intel386 DX Microprocessor. Therefore, the Intel387 DX MCP is not sensitive to the processing mode of the Intel386 DX Microprocessor.

In real-address mode and virtual-8086 mode, the Intel386 DX Microprocessor and Intel387 DX MCP are completely upward compatible with software for 8086/8087, 80286/80287 real-address mode, and Intel386 DX Microprocessor and 80287 Coprocessor real-address mode systems.

In protected mode, the Intel386 DX Microprocessor and Intel387 DX MCP are completely upward compatible with software for 80286/80287 protected mode, and Intel386 DX Microprocessor and 80287 Coprocessor protected mode systems.

The only differences of operation that may appear when 8086/8087 programs are ported to a protected-mode Intel386 DX Microprocessor and Intel387 DX MCP system (*not* using virtual-8086 mode), is in the format of operands for the administrative instructions FLDENV, FSTENV, FRSTOR and FSAVE. These instructions are normally used only by exception handlers and operating systems, not by applications programs.

The Intel387 DX MCP contains three functional units that can operate in parallel to increase system performance. The Intel386 DX Microprocessor can be transferring commands and data to the MCP *bus control logic* for the next instruction while the MCP *floating-point unit* is performing the current numeric instruction.

2.0 PROGRAMMING INTERFACE

The MCP adds to the Intel386 DX Microprocessor system additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. To use the MCP requires no special programming tools, because all new instructions and data types are directly supported by the Intel386 DX CPU assembler and compilers for high-level languages. All 8086/8088 development tools that support the 8087 can also be used to develop software for the Intel386 DX Microprocessor and Intel387 DX Math Coprocessor in real-address mode or virtual-8086 mode. All 80286 development tools that support the 80287 can also be used to develop software for the Intel386 DX Microprocessor and Intel387 DX Math Coprocessor.

All communication between the Intel386 DX Microprocessor and the MCP is transparent to applications software. The CPU automatically controls the MCP whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the MCP. All memory addressing modes, including use of displacement, base register, index register, and scaling, are available for addressing numerics operands.

Section 6 at the end of this data sheet lists by class the instructions that the MCP adds to the instruction set of the Intel386 DX Microprocessor system.

2.1 Data Types

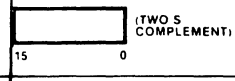
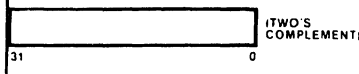
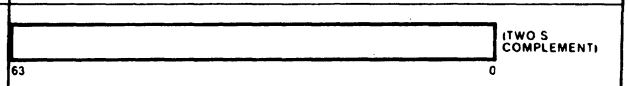
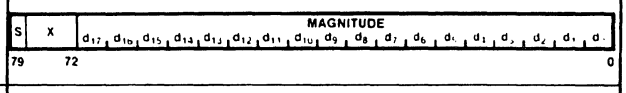
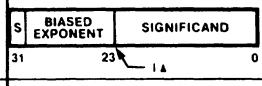
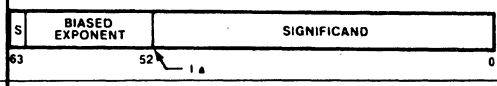
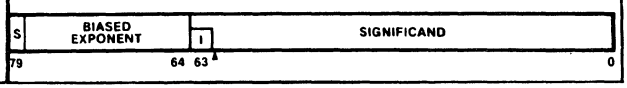
Table 2.1 lists the seven data types that the Intel387 DX MCP supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at physical-memory addresses evenly divisible by four (doubleword boundaries); operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

Internally, the Intel387 DX MCP holds all numbers in the extended-precision real format. Instructions that load operands from memory automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating-point numbers, or 18-digit packed BCD numbers into extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

2.2 Numeric Operands

A typical MCP instruction accepts one or two operands and produces a single result. In two-operand instructions, one operand is the contents of an MCP register, while the other may be a memory location. The operands of some instructions are predefined; for example FSQRT always takes the square root of the number in the top stack element.

Table 2.1. Intel387™ DX MCP Data Type Representation in Memory

Data Formats	Range	Precision	Most Significant Byte = Highest Addressed Byte
Word Integer	$\pm 10^4$	16 Bits	
Short Integer	$\pm 10^9$	32 Bits	
Long Integer	$\pm 10^{18}$	64 Bits	
Packed BCD	$\pm 10^{\pm 18}$	18 Digits	
Single Precision	$\pm 10^{\pm 38}$	24 Bits	
Double Precision	$\pm 10^{\pm 308}$	53 Bits	
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits	

240448-2

NOTES:
(1) S = Sign bit (0 = positive, 1 = negative)
(2) d_n = Decimal digit (two per byte)
(3) X = Bits have no significance; Intel387™ DX MCP ignores when loading, zeros when storing
(4)▲ = Position of implicit binary point
(5) I = Integer bit of significand; stored in temporary real, implicit in single and double precision
(6) Exponent Bias (normalized values):
Single: 127 (7FH)
Double: 1023 (3FFH)
Extended Real: 16383 (3FFFH)
(7) Packed BCD: $(-1)^S (D_{17}...D_0)$
(8) Real: $(-1)^S (2^{E-BIAS}) (F_0 F_1...)$

3

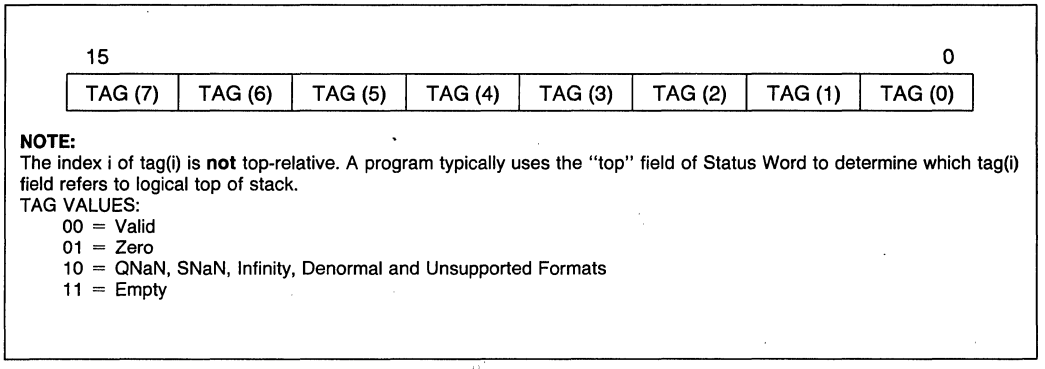


Figure 2.1. Intel387™ DX MCP Tag Word

2.3 Register Set

Figure 1.1 shows the Intel387 DX MCP register set. When an MCP is present in a system, programmers may use these registers in addition to the registers normally available on the Intel386 DX CPU.

2.3.1 DATA REGISTERS

Intel387 DX MCP computations use the MCP's data registers. These eight 80-bit registers provide the equivalent capacity of twenty 32-bit registers. Each of the eight data registers in the MCP is 80 bits wide and is divided into "fields" corresponding to the MCPs extended-precision real data type.

The Intel387 DX MCP register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as a fixed register set, with instructions operating on explicitly designated registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by one. Like the Intel386 DX Microprocessor stacks in memory, the MCP register stack grows "down" toward lower-addressed registers.

Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to user. This explicit register addressing is also relative to TOP.

2.3.2 TAG WORD

The tag word marks the content of each numeric data register, as Figure 2.1 shows. Each two-bit tag represents one of the eight numerics registers. The principal function of the tag word is to optimize the MCPs performance and stack handling by making it possible to distinguish between empty and nonempty register locations. It also enables exception handlers to check the contents of a stack location without the need to perform complex decoding of the actual data.

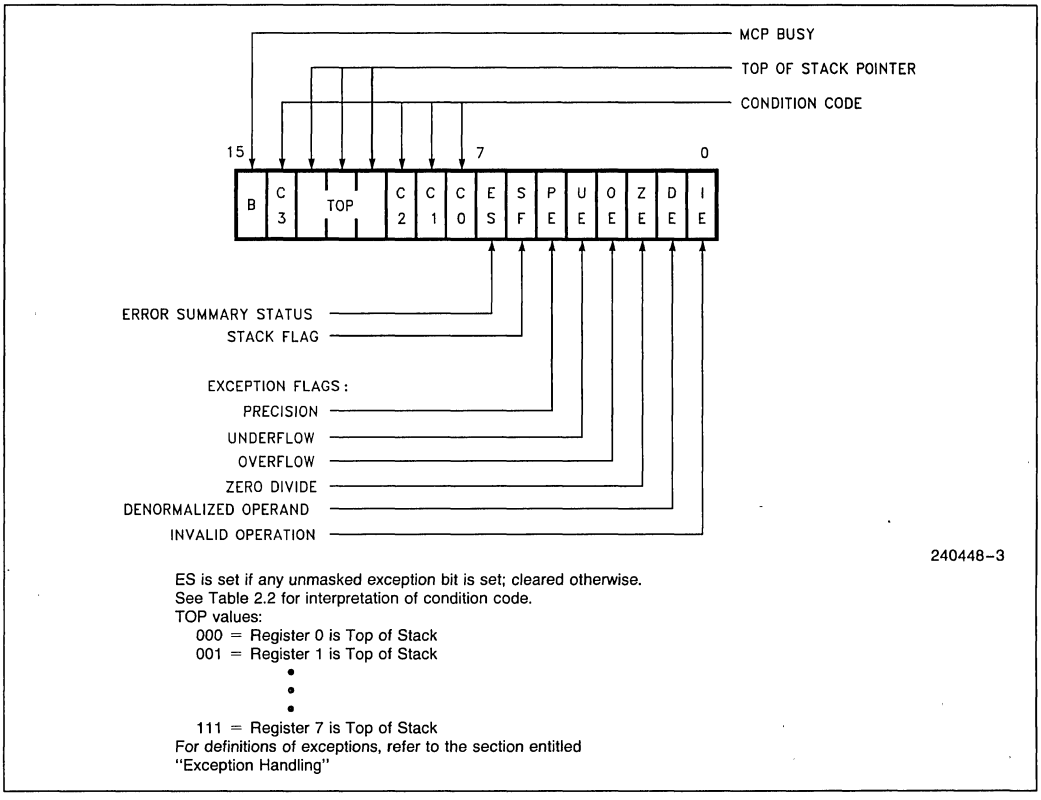


Figure 2.2. MCP Status Word

2.3.3 STATUS WORD

The 16-bit status word (in the status register) shown in Figure 2.2 reflects the overall state of the MCP. It may be read and inspected by CPU code.

Bit 15, the B-bit (busy bit) is included for 8087 compatibility only. It reflects the contents of the ES bit (bit 7 of the status word), not the status of the BUSY# output of the Intel387 DX MCP.

Bits 13–11 (TOP) point to the Intel387 DX MCP register that is the current top-of-stack.

The four numeric condition code bits (C_3–C_0) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of these instructions on the condition code are summarized in Tables 2.2 through 2.5.

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the ERROR# signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C_1) distinguishes between stack overflow (C_1 = 1) and underflow (C_1 = 0).

Figure 2.2 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the MCP has detected an exception while executing an instruction. A later section entitled "Exception Handling" explains how they are set and used.

Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the value of ES (bit 7) and its reflection in the B-bit (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the ERROR# output of the MCP is activated immediately.

Table 2.2. Condition Code Interpretation

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1 (see Table 2.3)	Three least significant bits of quotient Q2 Q0		Q1 or O/U#	Reduction 0 = complete 1 = incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 2.4)		Zero or O/U#	Operand is not comparable (Table 2.4)
FXAM	Operand class (see Table 2.5)		Sign or O/U#	Operand class (Table 2.5)
FCHS, FABS, FXCH, FINCSTP, FDECSTP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or O/U#	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U#	UNDEFINED
FPTAN, FSIN FCOS, FSINCOS	UNDEFINED		Roundup or O/U#, undefined if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

O/U#	When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).
Reduction	If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.
Roundup	When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.
UNDEFINED	Do not rely on finding any specific value in these bits.

Table 2.3. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interation required for complete reduction	
	Q1	Q0	Q2	Q MOD8	
0	0	0	0	0	Complete Reduction: C0, C3, C1 contain three least significant bits of quotient
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 2.4. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 2.5. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NāN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

2.3.4 INSTRUCTION AND DATA POINTERS

Because the MCP operates in parallel with the CPU, any errors detected by the MCP may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the failing numeric instruction, the Intel386 DX Microprocessor and Intel387 DX Math CoProcessor contains two pointer registers that supply the address of the failing numeric instruction and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written error handlers. These registers are actually located in the Intel386 DX CPU, but appear to be located in the MCP because they are accessed by the ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR. (In the 8086/8087 and 80286/80287, these registers are located in the MCP.) Whenever the Intel386 DX CPU decodes a new ESC instruction, it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if present), and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the Intel386 DX Microprocessor (protected mode or real-address mode) and depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). When the Intel386 DX Microprocessor is in virtual-8086 mode, the real-address mode formats are used. (See Figures 2.3 through 2.6.) The ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR are used to transfer these values between the Intel386 DX Microprocessor registers and memory. Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

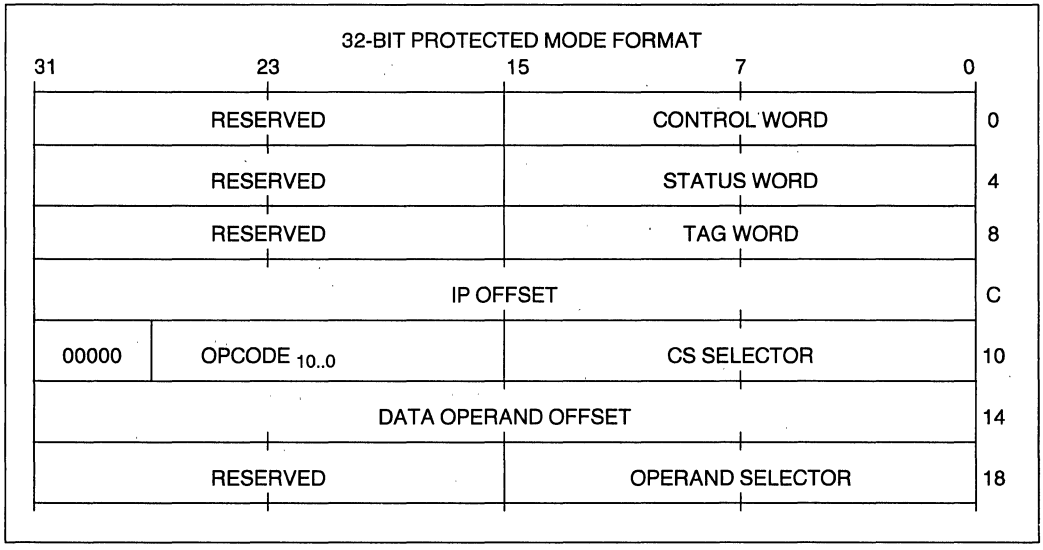


Figure 2.3. Protected Mode Intel387™ DX MCP Instruction and Data Pointer Image in Memory, 32-Bit Format

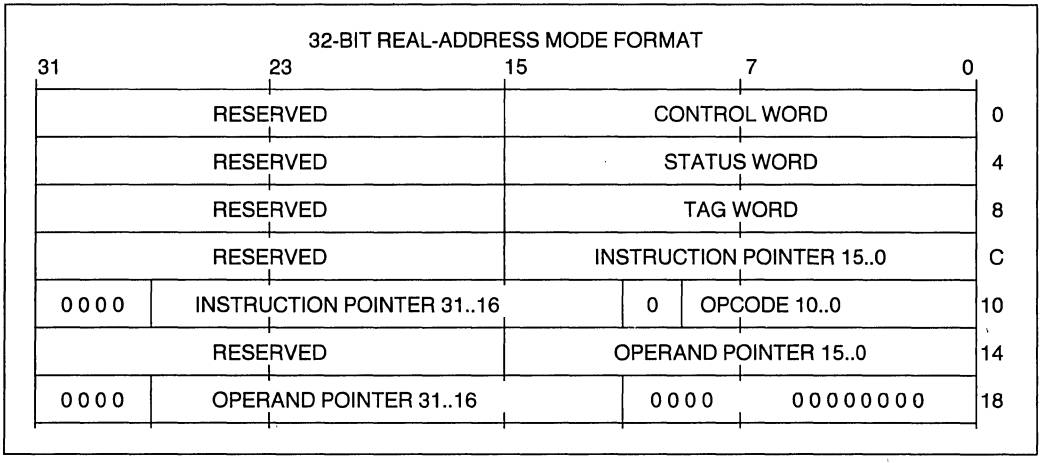


Figure 2.4. Real Mode Intel387™ DX MCP Instruction and Data Pointer Image in Memory, 32-Bit Format

3

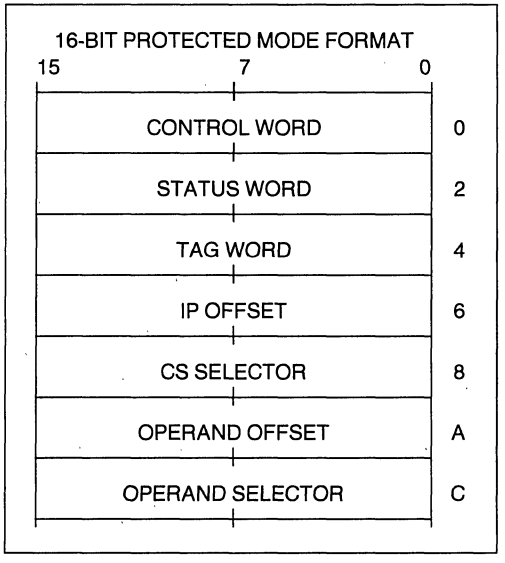


Figure 2.5. Protected Mode Intel387™ DX MCP Instruction and Data Pointer Image in Memory, 16-Bit Format

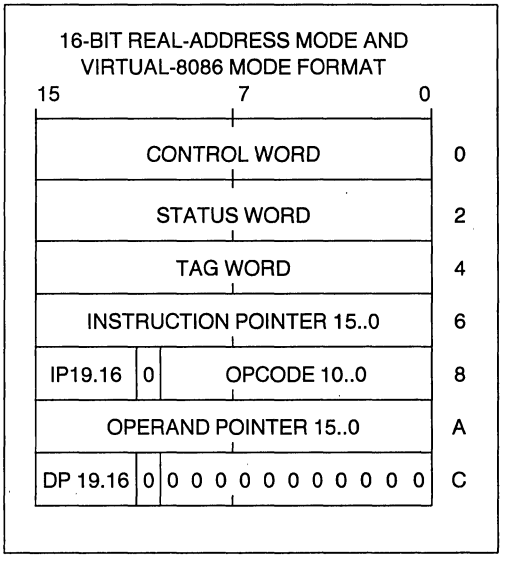


Figure 2.6. Real Mode Intel387™ DX MCP Instruction and Data Pointer Image in Memory, 16-Bit Format

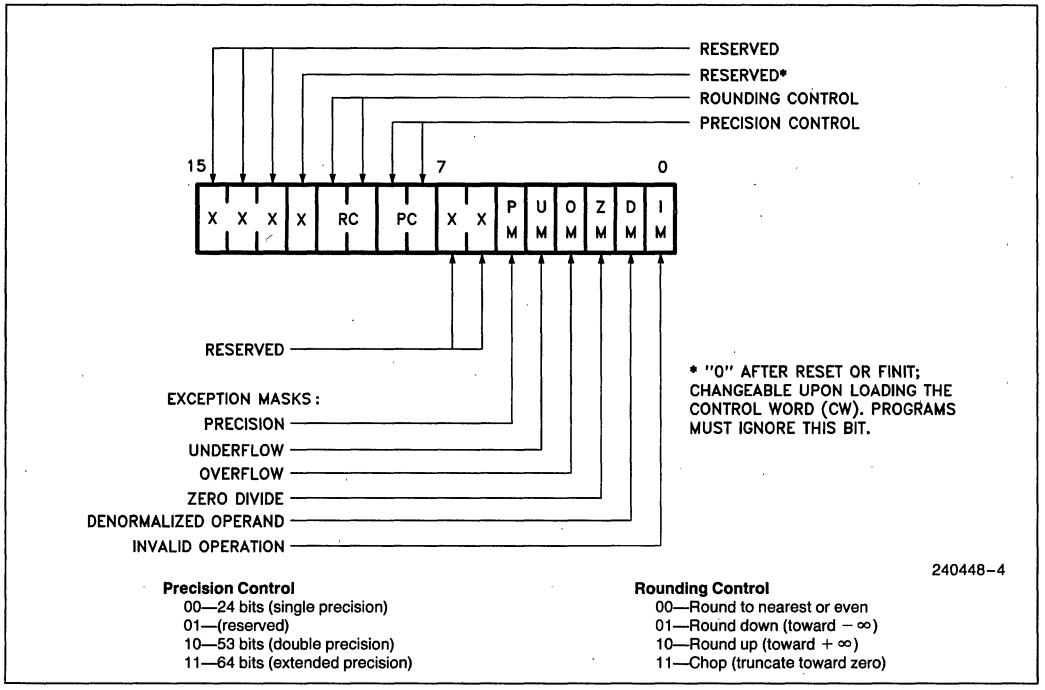


Figure 2.7. Intel387™ DX MCP Control Word

2.3.5 CONTROL WORD

The MCP provides several processing options that are selected by loading a control word from memory into the control register. Figure 2.7 shows the format and encoding of fields in the control word.

The low-order byte of this control word configures the MCP error and exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the MCP recognizes.

The high-order byte of the control word configures the MCP operating mode, including precision and rounding.

- Bit 12 no longer defines infinity control and is a reserved bit. Only affine closure is supported for infinity arithmetic. The bit is initialized to zero after RESET or FINIT and is changeable upon loading the CW. Programs must ignore this bit.
- The rounding control (RC) bits (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS, and FCHS), and all transcendental instructions.
- The precision control (PC) bits (bits 9–8) can be used to set the MCP internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

2.4 Interrupt Description

Several interrupts of the Intel386 DX CPU are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 2.6 shows these interrupts and their causes.

Table 2.6. Intel386™ DX Microprocessor Interrupt Vectors Reserved for MCP

Interrupt Number	Cause of Interrupt
7	An ESC instruction was encountered when EM or TS of the Intel386™ DX CPU control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction causes interrupt 7. This indicates that the current MCP context may not belong to the current task.
9	An operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for small segments, 0FFFFFFFFH for big segments, zero for expand-down segments) and spanned inaccessible addresses(1). The failing numerics instruction is not restartable. The address of the failing numerics instruction and data operand may be lost; an FSTENV does not return reliable addresses. As with the 80286/80287, the segment overrun exception should be handled by executing an FNINIT instruction (i.e. an FINIT without a preceding WAIT). The return address on the stack does not necessarily point to the failing instruction nor to the following instruction. The interrupt can be avoided by never allowing numeric data to start within 108 bytes of the end of a segment.
13	The first word or doubleword of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the ESC instruction that caused the exception, including any prefixes. The Intel387™ DX MCP has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC and WAIT instructions can cause this interrupt. The Intel386™ DX CPU return address pushed onto the stack of the exception handler points to a WAIT or ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the MCP. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.

1. An operand may wrap around an addressing limit when the segment limit is near an addressing limit and the operand is near the largest valid address in the segment. Because of the wrap-around, the beginning and ending addresses of such an operand will be at opposite ends of the segment. There are two ways that such an operand may also span inaccessible addresses: 1) if the segment limit is not equal to the addressing limit (e.g. addressing limit is FFFFH and segment limit is FFFDH) the operand will span addresses that are not within the segment (e.g. an 8-byte operand that starts at valid offset FFFC will span addresses FFFC–FFFF and 0000-0003; however addresses FFFE and FFFF are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible pages but intermediate bytes of the operand fall in a not-present page or a page to which the procedure does not have access rights.

2.5 Exception Handling

The Intel387 DX MCP detects six different exception conditions that can occur during instruction execution. Table 2.7 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the MCP if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the ERROR# signal. When the CPU attempts to execute another ESC instruction or WAIT, exception 7 occurs. The exception condition must be resolved via an interrupt service routine. The Intel386 DX Microprocessor saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

2.6 Initialization

Intel387 DX MCP initialization software must execute an FNINIT instruction (i.e. an FINIT without a preceding WAIT) to clear ERROR#. After a hardware RESET, the ERROR# output is asserted to indicate that a Intel387 DX MCP is present. To accomplish this, the IE and ES bits of the status word are set, and the IM bit in the control word is reset. After FNINIT, the status word and the control word have the same values as in an 80287 after RESET.

2.7 8087 and 80287 Compatibility

This section summarizes the differences between the Intel387 DX MCP and the 80287. Any migration from the 8087 directly to the Intel387 DX MCP must also take into account the differences between the 8087 and the 80287 as listed in Appendix A.

Many changes have been designed into the Intel387 DX MCP to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

2.7.1 GENERAL DIFFERENCES

The Intel387 DX MCP supports only affine closure for infinity arithmetic, not projective closure. Bit 12 of the Control Word (CW) no longer defines infinity control. It is a reserved bit; but it is initialized to zero after RESET or FINIT and is changeable upon loading the CW. Programs must ignore this bit.

Operands for FSCALE and FPATAN are no longer restricted in range (except for $\pm\infty$); F2XM1 and FPTAN accept a wider range of operands.

The results of transcendental operations may be slightly different from those computed by 80287.

In the case of FPTAN, the Intel387 DX MCP supplies a true tangent result in ST(1), and (always) a floating point 1 in ST.

Rounding control is in effect for FLD *constant*.

Software cannot change entries of the tag word to values (other than empty) that do not reflect the actual register contents.

After reset, FINIT, and incomplete FPREM, the Intel387 DX MCP resets to zero the condition code bits C_3–C_0 of the status word.

In conformance with the IEEE standard, the Intel387 DX MCP does not support the special data formats: pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

Table 2.7. Exceptions

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signaling NaN, unsupported format, indeterminate form ($0*\infty$, 0/0, $(+\infty)+(-\infty)$, etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinite
Denormalized Operand	At least one of the operands is denormalized, i.e. it has the smallest exponent but a nonzero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number.	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g. 1/3); the result is rounded according to the rounding mode.	Normal processing continues

2.7.2 EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the Intel387 DX MCP:

1. When the overflow or underflow exception is masked, the Intel387 DX MCP differs from the 80287 in rounding when overflow or underflow occurs. The Intel387 DX MCP produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the Intel387 DX MCP sets its underflow flag only if there is also a loss of accuracy during denormalization.
3. Fewer invalid-operation exceptions due to denormal operands, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
4. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
5. The denormal exception can occur during the transcendental instructions and the FXTRACT instruction.
6. The denormal exception no longer takes precedence over all other exceptions.
7. When the denormal exception is masked, the Intel387 DX MCP automatically normalizes denormal operands. The 8087/80287 performs unnormal arithmetic, which might produce an unnormal result.
8. When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
9. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
10. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
11. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized operand exception. When loading a signaling NaN, FLD *single/double precision* signals an invalid-operand exception.
12. The Intel387 DX MCP only generates quiet NaNs (as on the 80287); however, the Intel387 DX MCP distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
13. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 80287/8087 leaves the original operand in ST(1) intact.
14. When the scaling factor is $\pm\infty$, the FSCALE (ST(0), ST(1)) instruction behaves as follows (ST(0) and ST(1) contain the scaled and scaling operands respectively):
    - FSCALE(0,∞) generates the invalid operation exception.
    - FSCALE(finite, $-\infty$) generates zero with the same sign as the scaled operand.
    - FSCALE(finite, $+\infty$) generates ∞ with the same sign as the scaled operand.

    The 8087/80287 returns zero in the first case and raises the invalid-operation exception in the other cases.
15. The Intel387 DX MCP returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 and 80287 support a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

3.0 HARDWARE INTERFACE

3

In the following description of hardware interface, the # symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

3.1 Signal Description

In the following signal descriptions, the Intel387 DX Math Coprocessor pins are grouped by function as follows:

1. Execution control—CPUCLK2, NUMCLK2, CKM, RESETIN
2. MCP handshake—PEREQ, BUSY#, ERROR#
3. Bus interface pins—D31–D0, W/R#, ADS#, READY#, READYO#
4. Chip/Port Select—STEN, NPS1#, NPS2, CMD0#
5. Power supplies—V_{CC}, V_{SS}

Table 3.1 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics. All output signals are tristate; they leave floating state only when STEN is active. The output buffers of the bidirectional data pins D31–D0 are also tristate; they leave floating state only in read cycles when the MCP is selected (i.e. when STEN, NPS1#, and NPS2 are all active).

Figure 3.1 and Table 3.2 together show the location of every pin in the pin grid array.

Table 3.1. Intel387™ DX MCP Pin Summary

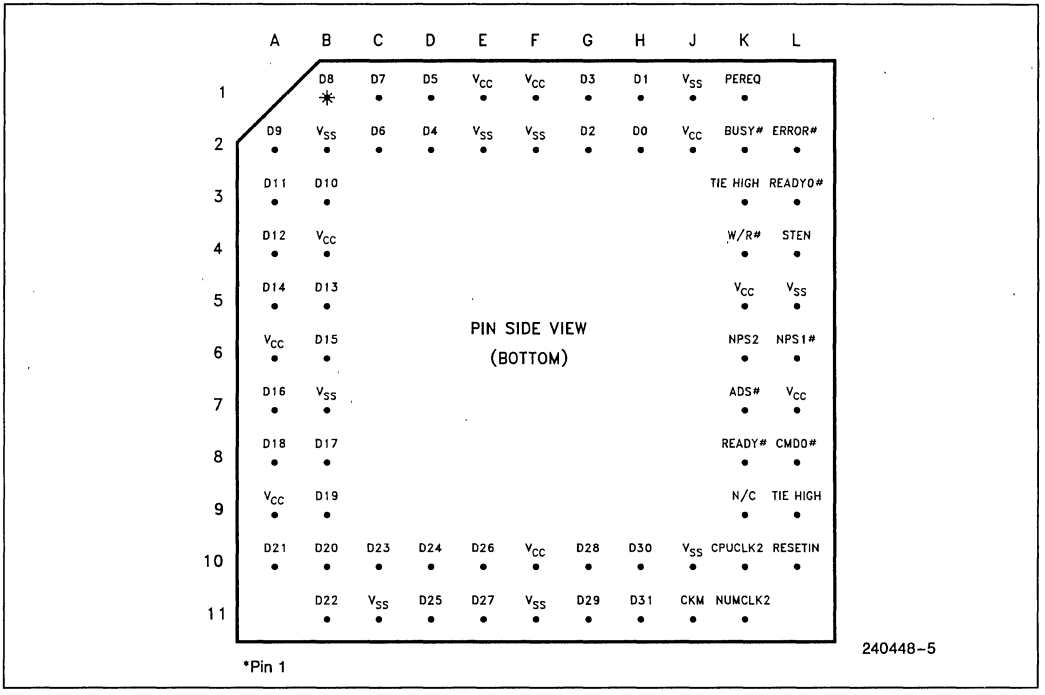
Pin Name	Function	Active State	Input/ Output	Referenced To
CPUCLK2	Intel386™ DX CPU CLocK 2		I	
NUMCLK2	Intel387™ DX MCP CLocK 2		I	
CKM	Intel387™ DX MCP CLocKing Mode		I	
RESETIN	System reset	High	I	CPUCLK2
PEREQ	Processor Extension REQuest	High	O	CPUCLK2/STEN
BUSY#	Busy status	Low	O	CPUCLK2/STEN
ERROR#	Error status	Low	O	NUMCLK2/STEN
D31–D0	Data pins	High	I/O	CPUCLK2
W/R#	Write/Read bus cycle	Hi/Lo	I	CPUCLK2
ADS#	ADdress Strobe	Low	I	CPUCLK2
READY#	Bus ready input	Low	I	CPUCLK2
READYO#	Ready output	Low	O	CPUCLK2/STEN
STEN	STatus ENable	High	I	CPUCLK2
NPS1#	MCP select #1	Low	I	CPUCLK2
NPS2	MCP select #2	High	I	CPUCLK2
CMD0#	CoMmanD	Low	I	CPUCLK2
V_{CC}			I	
V_{SS}			I	

NOTE:
STEN is referenced to only when getting the output pins into or out of tristate mode.

Table 3.2. Intel387™ DX MCP Pin Cross-Reference

ADS#	—	K7	D18	—	A8	STEN	—	L4
BUSY#	—	K2	D19	—	B9	W/R#	—	K4
CKM	—	J11	D20	—	B10			
CPUCLK24	—	K10	D21	—	A10	V_{CC}	—	A6, A9, B4,
CMD0#	—	L8	D22	—	B11			E1, F1, F10,
D0	—	H2	D23	—	C10			J2, K5,
D1	—	H1	D24	—	D10			L7
D2	—	G2	D25	—	D11			
D3	—	G1	D26	—	E10	V_{SS}	—	B2, B7, C11,
D4	—	D2	D27	—	E11			E2, F2, F11,
D5	—	D1	D28	—	G10			J1, J10, L5
D6	—	C2	D29	—	G11			
D7	—	C1	D30	—	H10	NO CONNECT	—	K9
D8	—	B1	D31	—	H11	TIE HIGH	—	K3, L9*
D9	—	A2	ERROR#	—	L2			
D10	—	B3	NPS1#	—	L6			
D11	—	A3	NPS2	—	K6			
D12	—	A4	NUMCLK2	—	K11			
D13	—	B5	PEREQ	—	K1			
D14	—	A5	READY#	—	K8			
D15	—	B6	READYO#	—	L3			
D16	—	A7	RESETIN	—	L10			
D17	—	B8						

*Tie high pins may either be tied high with a pullup resistor or connected to V_{CC}.



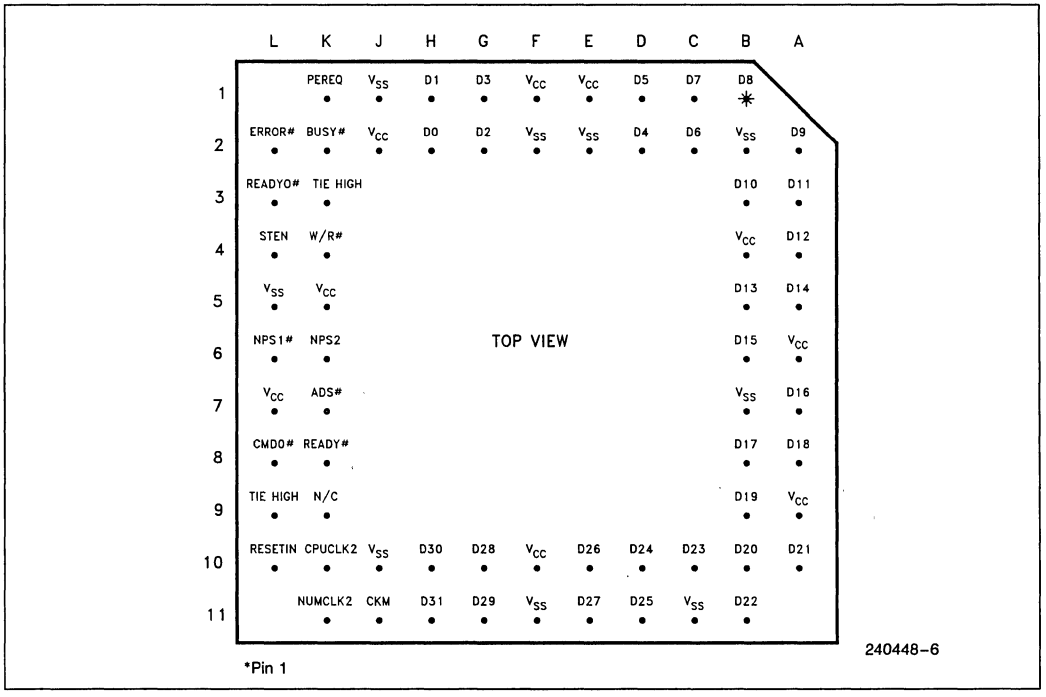


Figure 3.1. Intel387™ DX MCP Pin Configuration

3.1.1 Intel386™ DX CPU CLOCK 2 (CPUCLK2)

This input uses the Intel386 DX CPU CLK2 signal to time the bus control logic. Several other MCP signals are referenced to the rising edge of this signal. When CKM = 1 (synchronous mode) this pin also clocks the data interface and control unit and the floating-point unit of the MCP. This pin requires MOS-level input. The signal on this pin is divided by two to produce the internal clock signal CLK.

3.1.2 Intel387™ DX MCP CLOCK 2 (NUMCLK2)

When CKM = 0 (asynchronous mode) this pin provides the clock for the data interface and control unit and the floating-point unit of the MCP. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. When CKM = 1 (synchronous mode) this pin is ignored; CPUCLK2 is used instead for the data interface and control unit and the floating-point unit. This pin requires TTL-level input.

3.1.3 Intel387™ DX MCP CLOCKING MODE (CKM)

This pin is a strapping option. When it is strapped to V_{CC}, the MCP operates in synchronous mode; when strapped to V_{SS}, the MCP operates in asynchronous mode. These modes relate to clocking of the data interface and control unit and the floating-point unit only; the bus control logic always operates synchronously with respect to the Intel386 DX Microprocessor.

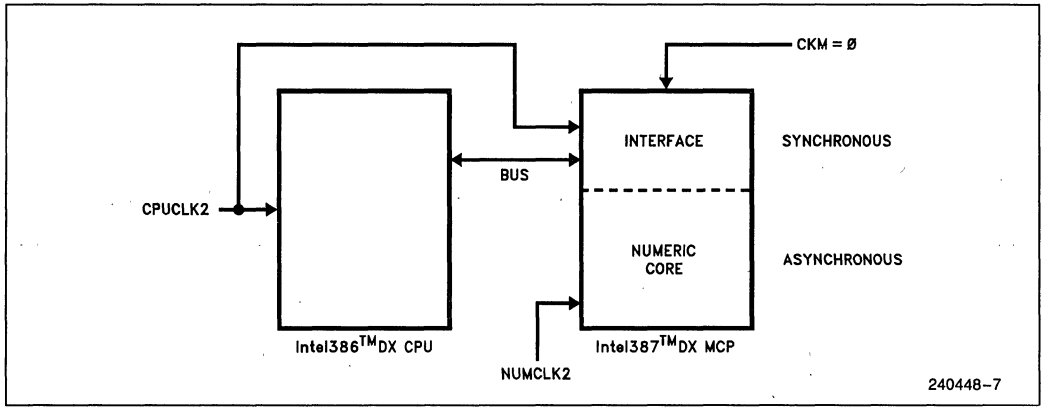


Figure 3.2. Asynchronous Operation

3.1.4 SYSTEM RESET (RESETIN)

A LOW to HIGH transition on this pin causes the MCP to terminate its present activity and to enter a dormant state. RESETIN must remain HIGH for at least 40 NUMCLK2 periods. The HIGH to LOW transitions of RESETIN must be synchronous with CPUCLK2, so that the phase of the internal clock of the bus control logic (which is the CPUCLK2 divided by 2) is the same as the phase of the internal clock of the Intel386 DX CPU. After RESETIN goes LOW, at least 50 NUMCLK2 periods must pass before the first MCP instruction is written into the Intel387 DX MCP. This pin should be connected to the Intel386 DX CPU RESET pin. Table 3.3 shows the status of other pins after a reset.

Table 3.3. Output Pin Status During Reset

Pin Value	Pin Name
HIGH	READYO#, BUSY#
LOW	PEREQ, ERROR#
Tri-State OFF	D31–D0

3.1.5 PROCESSOR EXTENSION REQUEST (PEREQ)

When active, this pin signals to the Intel386 DX CPU that the MCP is ready for data transfer to/from its data FIFO. When all data is written to or read from the data FIFO, PEREQ is deactivated. This signal always goes inactive before BUSY# goes inactive. This signal is referenced to CPUCLK2. It should be connected to the Intel386 DX CPU PEREQ input.

3.1.6 BUSY STATUS (BUSY#)

When active, this pin signals to the Intel386 DX CPU that the MCP is currently executing an instruction. This signal is referenced to CPUCLK2. It should be connected to the Intel386 DX CPU BUSY# pin.

3.1.7 ERROR STATUS (ERROR#)

This pin reflects the ES bits of the status register. When active, it indicates that an unmasked exception has occurred (except that, immediately after a reset, it indicates to the Intel386 DX Microprocessor that a Intel387 DX MCP is present in the system). This signal can be changed to inactive state only by the following instructions (without a preceding WAIT): FNINIT, FNCLEX, FNSTENV, and FNSAVE. This signal is referenced to NUMCLK2. It should be connected to the Intel386 DX CPU ERROR# pin.

3.1.8 DATA PINS (D31–D0)

These bidirectional pins are used to transfer data and opcodes between the Intel386 DX CPU and Intel387 DX MCP. They are normally connected directly to the corresponding Intel386 DX CPU data pins. HIGH state indicates a value of one. D0 is the least significant data bit. Timings are referenced to CPUCLK2.

3.1.9 WRITE/READ BUS CYCLE (W/R#)

This signal indicates to the MCP whether the Intel386 DX CPU bus cycle in progress is a read or a write cycle. This pin should be connected directly to the Intel386 DX CPU W/R# pin. HIGH indicates a write cycle; LOW, a read cycle. This input is ignored if any of the signals STEN, NPS1#, or NPS2 is inactive. Setup and hold times are referenced to CPUCLK2.

3.1.10 ADDRESS STROBE (ADS#)

This input, in conjunction with the READY# input indicates when the MCP bus-control logic may sample W/R# and the chip-select signals. Setup and hold times are referenced to CPUCLK2. This pin should be connected to the Intel386 DX CPU ADS# pin.

3.1.11 BUS READY INPUT (READY#)

This input indicates to the MCP when a Intel386 DX CPU bus cycle is to be terminated. It is used by the bus-control logic to trace bus activities. Bus cycles can be extended indefinitely until terminated by READY#. This input should be connected to the same signal that drives the Intel386 DX CPU READY# input. Setup and hold times are referenced to CPUCLK2.

3.1.12 READY OUTPUT (READYO#)

This pin is activated at such a time that write cycles are terminated after two clocks (except FLDENV and FRSTOR) and read cycles after three clocks. In configurations where no extra wait states are required, this pin must directly or indirectly drive the Intel386 DX CPU READY# input. Refer to section 3.4 "Bus Operation" for details. This pin is activated only during bus cycles that select the MCP. This signal is referenced to CPUCLK2.

3.1.13 STATUS ENABLE (STEN)

This pin serves as a chip select for the MCP. When inactive, this pin forces BUSY#, PEREQ, ERROR#, and READYO# outputs into floating state. D31–D0 are normally floating and leave floating state only if STEN is active and additional conditions are met. STEN also causes the chip to recognize its other chip-select inputs. STEN makes it easier to do on-board testing (using the overdrive method) of other chips in systems containing the MCP. STEN should be pulled up with a resistor so that it can be pulled down when testing. In boards that do not use on-board testing, STEN should be connected to V_{CC}. Setup and hold times are relative to CPUCLK2. Note that STEN must maintain the same setup and hold times as NPS1#, NPS2, and CMD0# (i.e. if STEN changes state during a Intel387 DX MCP bus cycle, it should change state during the same CLK period as the NPS1#, NPS2, and CMD0# signals).

3.1.14 MCP Select #1 (NPS1#)

When active (along with STEN and NPS2) in the first period of a Intel386 DX CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the MCP. This pin should be connected directly to the Intel386 DX CPU M/IO# pin, so that the MCP is selected only when the Intel386 DX CPU performs I/O cycles. Setup and hold times are referenced to CPUCLK2.

3.1.15 MCP SELECT #2 (NPS2)

When active (along with STEN and NPS1#) in the first period of an Intel386 DX CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the MCP. This pin should be connected directly to the Intel386 DX CPU A31 pin, so that the MCP is selected only when the Intel386 DX CPU uses one of the I/O addresses reserved for the MCP (800000F8 or 800000FC). Setup and hold times are referenced to CPUCLK2.

3.1.16 COMMAND (CMD0#)

During a write cycle, this signal indicates whether an opcode (CMD0# active) or data (CMD0# inactive) is being sent to the MCP. During a read cycle, it indicates whether the control or status register (CMD0# active) or a data register (CMD0# inactive) is being read. CMD0# should be connected directly to the A2 output of the Intel386 DX Microprocessor. Setup and hold times are referenced to CPUCLK2.

3.2 Processor Architecture

As shown by the block diagram on the front page, the MCP is internally divided into three sections: the bus control logic (BCL), the data interface and control unit, and the floating point unit (FPU). The FPU (with the support of the control unit which contains the sequencer and other support units) executes all numerics instructions. The data interface and control unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, and sequencing the microinstructions, and for handling some of the administrative instructions. The BCL is responsible for the Intel386 DX CPU bus tracking and interface. The BCL is the only unit in the Intel387 DX MCP that must run synchronously with the Intel386 DX CPU; the rest of the MCP can run asynchronously with respect to the Intel386 DX Microprocessor.

3.2.1 BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from memory to the MCP and transferring outputs from the MCP to memory.

3.2.2 DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the FIFO or the instruction decoder. The instruction decoder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, or FSTCW, the control executes it independently of the FPU and the sequencer. The data interface and control unit is the one that generates the BUSY#, PEREQ and ERROR# signals that synchronize Intel387 DX MCP activities with the Intel386 DX CPU. It also supports the FPU in all operations that it cannot perform alone (e.g. exceptions handling, transcendental operations, etc.).

3.2.3 FLOATING POINT UNIT

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

3.3 System Configuration

As an extension to the Intel386 DX Microprocessor, the Intel387 DX Math Coprocessor can be connected to the CPU as shown by Figure 3.3. A dedicated

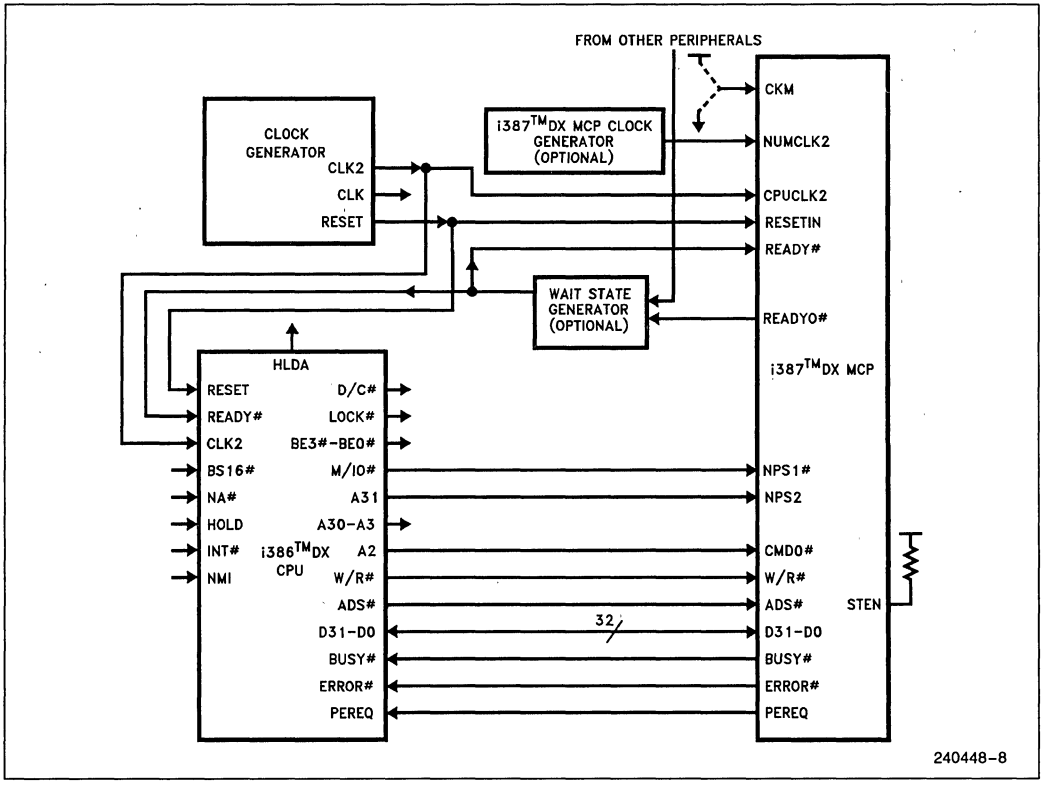


Figure 3.3. Intel386™ DX Microprocessor and Intel387™ DX Math Coprocessor System Configuration

Table 3.4. Bus Cycles Definition

STEN	NPS1#	NPS2	CMD0#	W/R#	Bus Cycle Type
0	x	x	x	x	MCP not selected and all outputs in floating state
1	1	x	x	x	MCP not selected
1	x	0	x	x	MCP not selected
1	0	1	0	0	CW or SW read from MCP
1	0	1	0	1	Opcode write to MCP
1	0	1	1	0	Data read from MCP
1	0	1	1	1	Data write to MCP

communication protocol makes possible high-speed transfer of opcodes and operands between the Intel386 DX CPU and Intel387 DX MCP. The Intel387 DX MCP is designed so that no additional components are required for interface with the Intel386 DX CPU. The Intel387 DX MCP shares the 32-bit wide local bus of the Intel386 DX CPU and most control pins of the Intel387 DX MCP are connected directly to pins of the Intel386 DX Microprocessor.

3.3.1 BUS CYCLE TRACKING

The ADS# and READY# signals allow the MCP to track the beginning and end of the Intel386 DX CPU bus cycles, respectively. When ADS# is asserted at the same time as the MCP chip-select inputs, the bus cycle is intended for the MCP. To signal the end of a bus cycle for the MCP, READY# may be asserted directly or indirectly by the MCP or by other bus-control logic. Refer to Table 3.4 for definition of the types of MCP bus cycles.

3.3.2 MCP ADDRESSING

The NPS1#, NPS2 and STEN signals allow the MCP to identify which bus cycles are intended for the MCP. The MCP responds only to I/O cycles when bit 31 of the I/O address is set. In other words, the MCP acts as an I/O device in a reserved I/O address space.

Because A_{31} is used to select the MCP for data transfers, it is not possible for a program running on the Intel386 DX CPU to address the MCP with an I/O instruction. Only ESC instructions cause the Intel386 DX Microprocessor to communicate with the MCP. The Intel386 DX CPU BS16# input must be inactive during I/O cycles when A_{31} is active.

3.3.3 FUNCTION SELECT

The CMD0# and W/R# signals identify the four kinds of bus cycle: control or status register read, data read, opcode write, data write.

3.3.4 CPU/MCP Synchronization

The pin pairs BUSY#, PEREQ, and ERROR# are used for various aspects of synchronization between the CPU and the MCP.

BUSY# is used to synchronize instruction transfer from the Intel386 DX CPU to the MCP. When the MCP recognizes an ESC instruction, it asserts BUSY#. For most ESC instructions, the Intel386 DX CPU waits for the MCP to deassert BUSY# before sending the new opcode.

The MCP uses the PEREQ pin of the Intel386 DX CPU to signal that the MCP is ready for data transfer to or from its data FIFO. The MCP does not directly access memory; rather, the Intel386 DX Microprocessor provides memory access services for the MCP. Thus, memory access on behalf of the MCP always obeys the rules applicable to the mode of the Intel386 DX CPU, whether the Intel386 DX CPU be in real-address mode or protected mode.

Once the Intel386 DX CPU initiates an MCP instruction that has operands, the Intel386 DX CPU waits for PEREQ signals that indicate when the MCP is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the Intel386 DX CPU continues program execution while the MCP executes the ESC instruction.

In 8086/8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In 80286/80287, Intel386 DX Microprocessor and Intel387 DX Math Coprocessor systems, WAIT instructions are required only for operand synchronization; namely, after MCP stores to memory (except FSTSW and FSTCW) or loads from memory. Used this way, WAIT ensures that the value has already been written or read by the MCP before the CPU reads or changes the value.

Once it has started to execute a numerics instruction and has transferred the operands from the Intel386 DX CPU, the MCP can process the instruction in parallel with and independent of the host CPU. When the MCP detects an exception, it asserts the ERROR# signal, which causes a Intel386 DX CPU interrupt.

3.3.5 SYNCHRONOUS OR ASYNCHRONOUS MODES

The internal logic of the Intel387 DX MCP (the FPU) can either operate directly from the CPU clock (synchronous mode) or from a separate clock (asynchronous mode). The two configurations are distinguished by the CKM pin. In either case, the bus control logic (BCL) of the MCP is synchronized with the CPU clock. Use of asynchronous mode allows the Intel386 DX CPU and the FPU section of the MCP to run at different speeds. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. Use of synchronous mode eliminates one clock generator from the board design.

3.3.6 AUTOMATIC BUS CYCLE TERMINATION

In configurations where no extra wait states are required, READYO# can be used to drive the Intel386 DX CPU READY# input. If this pin is used, it should be connected to the logic that ORs all READY outputs from peripherals on the Intel386 DX CPU bus. READYO# is asserted by the MCP only during I/O cycles that select the MCP. Refer to section 3.4 "Bus Operation" for details.

3.4 Bus Operation

With respect to the bus interface, the Intel387 DX MCP is fully synchronous with the Intel386 DX Microprocessor. Both operate at the same rate, because each generates its internal CLK signal by dividing CPUCLK2 by two.

The Intel386 DX CPU initiates a new bus cycle by activating ADS#. The MCP recognizes a bus cycle, if, during the cycle in which ADS# is activated, STEN, NPS1#, and NPS2 are all activated. Proper operation is achieved if NPS1# is connected to the M/IO# output of the Intel386 DX CPU, and NPS2 to the A31 output. The Intel386 DX CPU's A31 output is guaranteed to be inactive in all bus cycles that do not address the MCP (i.e. I/O cycles to other devices, interrupt acknowledge, and reserved types of bus cycles). System logic must not signal a 16-bit bus cycle via the Intel386 DX CPU BS16# input during I/O cycles when A31 is active.

During the CLK period in which ADS# is activated, the MCP also examines the W/R# input signal to determine whether the cycle is a read or a write cycle and examines the CMD0# input to determine whether an opcode, operand, or control/status register transfer is to occur.

The Intel387 DX MCP supports both pipelined and nonpipelined bus cycles. A nonpipelined cycle is one for which the Intel386 DX CPU asserts ADS# when no other MCP bus cycle is in progress. A pipelined bus cycle is one for which the Intel386 DX CPU asserts ADS# and provides valid next-address and control signals as soon as in the second CLK period after the ADS# assertion for the previous Intel386 DX CPU bus cycle. Pipelining increases the availability of the bus by at least one CLK period. The MCP supports pipelined bus cycles in order to optimize address pipelining by the Intel386 DX CPU for memory cycles.

Bus operation is described in terms of an abstract *state machine*. Figure 3.4 illustrates the states and state transitions for MCP bus cycles:

- T_I is the idle state. This is the state of the bus logic after RESET, the state to which bus logic returns after evey nonpipelined bus cycle, and the state to which bus logic returns after a series of pipelined cycles.
- T_{RS} is the READY# sensitive state. Different types of bus cycle may require a minimum of one or two successive T_{RS} states. The bus logic remains in T_{RS} state until READY# is sensed, at which point the bus cycle terminates. Any number of wait states may be implemented by delaying READY#, thereby causing additional successive T_{RS} states.
- T_P is the first state for every pipelined bus cycle.

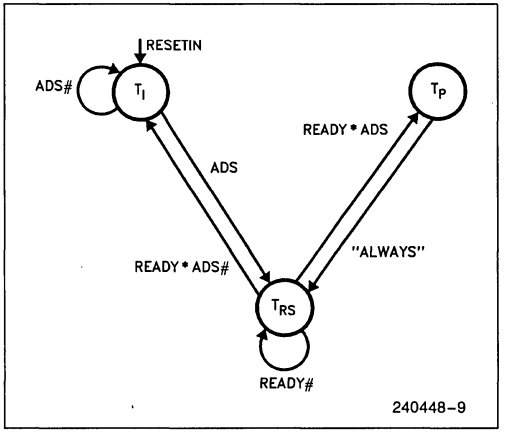


Figure 3.4. Bus State Diagram

The READYO# output of the Intel387 DX MCP indicates when a bus cycle for the MCP may be terminated if no extra wait states are required. For all write cycles (except those for the instructions FLDENV and FRSTOR), READYO# is always asserted in the first T_{RS} state, regardless of the number of wait states. For all read cycles and write cycles for FLDENV and FRSTOR, READYO# is always asserted in the second T_{RS} state, regardless of the number of wait states. These rules apply to both pipelined and nonpipelined cycles. Systems designers must use READYO# in one of the following ways:

1. Connect it (directly or through logic that ORs READY signals from other devices) to the READY# inputs of the Intel386 DX CPU and Intel387 DX MCP.
2. Use it as one input to a wait-state generator.

The following sections illustrate different types of MCP bus cycles.

Because different instructions have different amounts of overhead before, between, and after operand transfer cycles, it is not possible to represent in a few diagrams all of the combinations of successive operand transfer cycles. The following bus-cycle diagrams show memory cycles between MCP operand-transfer cycles. Note however that, during the instructions FLDENV, FSTENV, FSAVE, and FRSTOR, some consecutive accesses to the MCP do not have intervening memory accesses. For the timing relationship between operand transfer cycles and opcode write or other overhead activities, see Figure 3.8.

3.4.1 NONPIPELINED BUS CYCLES

Figure 3.5 illustrates bus activity for consecutive nonpipelined bus cycles.

3.4.1.1 Write Cycle

At the second clock of the bus cycle, the Intel387 DX MCP enters the T_{RS} (READY#-sensitive) state. During this state, the Intel387 DX MCP samples the READY# input and stays in this state as long as READY# is inactive.

In write cycles, the MCP drives the READYO# signal for one CLK period beginning with the second CLK of the bus cycle; therefore, the fastest write cycle takes two CLK cycles (see cycle 2 of Figure 3.5). For the instructions FLDENV and FRSTOR, however, the MCP forces a wait state by delaying the activation of READYO# to the second T_{RS} cycle (not shown in Figure 3.5).

When READY# is asserted the MCP returns to the idle state, in which ADS# could be asserted again by the Intel386 DX Microprocessor for the next cycle.

3.4.1.2 Read Cycle

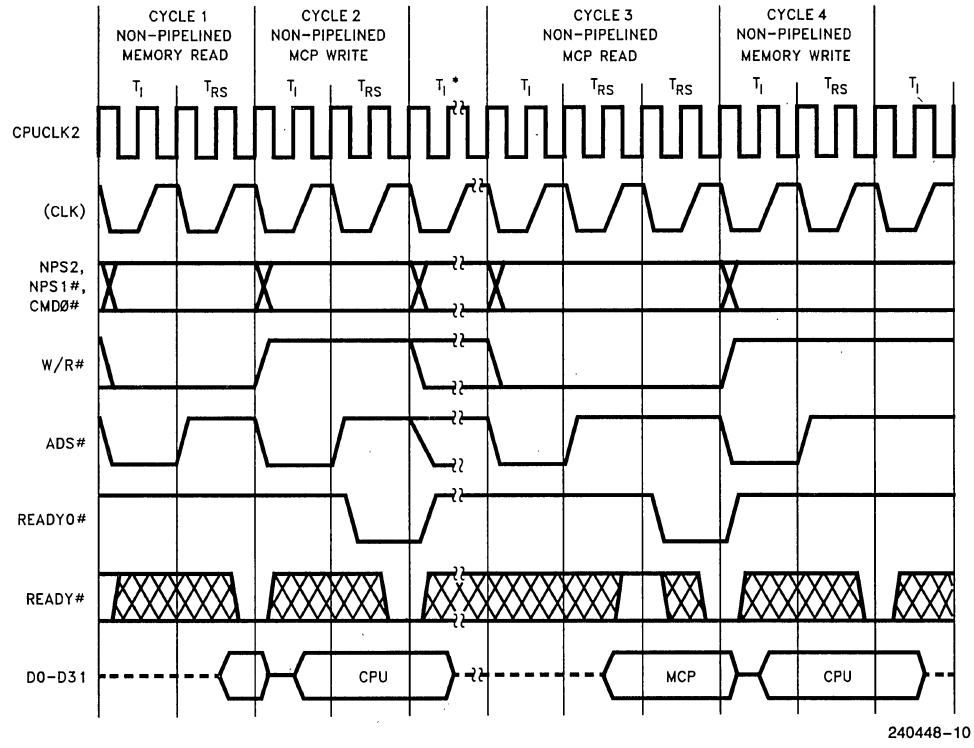
At the second clock of the bus cycle, the MCP enters the T_{RS} state. See Figure 3.5. In this state, the MCP samples the READY# input and stays in this state as long as READY# is inactive.

At the rising edge of CLK in the second clock period of the cycle, the MCP starts to drive the D31–D0 outputs and continues to drive them as long as it stays in T_{RS} state.

In read cycles that address the MCP, at least one wait state must be inserted to insure that the Intel386 DX CPU latches the correct data. Since the MCP starts driving the system data bus only at the rising edge of CLK in the second clock period of the bus cycle, not enough time is left for the data signals to propagate and be latched by the Intel386 DX CPU at the falling edge of the same clock period. The MCP drives the READYO# signal for one CLK period in the third CLK of the bus cycle. Therefore, if the READYO# output is used to drive the Intel386 DX CPU READY# input, one wait state is inserted automatically.

Because one wait state is required for MCP reads, the minimum is three CLK cycles per read, as cycle 3 of Figure 3.5 shows.

When READY# is asserted the MCP returns to the idle state, in which ADS# could be asserted again by the Intel386 DX CPU for the next cycle. The transition from T_{RS} state to idle state causes the MCP to put the tristate D31–D0 outputs into the floating state, allowing another device to drive the system data bus.



Cycles 1 & 2 represent part of the operand transfer cycle for instructions involving either 4-byte or 8-byte operand loads.
Cycles 3 & 4 represent part of the operand transfer cycle for a store operation.
*Cycles 1 & 2 could repeat here or T_I states for various non-operand transfer cycles and overhead.

Figure 3.5. Nonpipelined Read and Write Cycles

3.4.2 PIPELINED BUS CYCLES

Because all the activities of the Intel387 DX MCP bus interface occur either during the T_{RS} state or during the transitions to or from that state, the only difference between a pipelined and a nonpipelined cycle is the manner of changing from one state to another. The exact activities in each state are detailed in the previous section "Nonpipelined Bus Cycles".

When the Intel386 DX CPU asserts ADS# before the end of a bus cycle, both ADS# and READY# are active during a T_{RS} state. This condition causes the MCP to change to a different state named T_P. The MCP activities in the transition from a T_{RS} state to a T_P state are exactly the same as those in the transition from a T_{RS} state to a T_I state in nonpipelined cycles.

T_P state is metastable; therefore, one clock period later the MCP returns to T_{RS} state. In consecutive pipelined cycles, the MCP bus logic uses only T_{RS} and T_P states.

Figure 3.6 shows the fastest transition into and out of the pipelined bus cycles. Cycle 1 in this figure represents a nonpipelined cycle. (Nonpipelined write cycles with only one T_{RS} state (i.e. no wait states) are always followed by another nonpipelined cycle, because READY# is asserted before the earliest possible assertion of ADS# for the next cycle.)

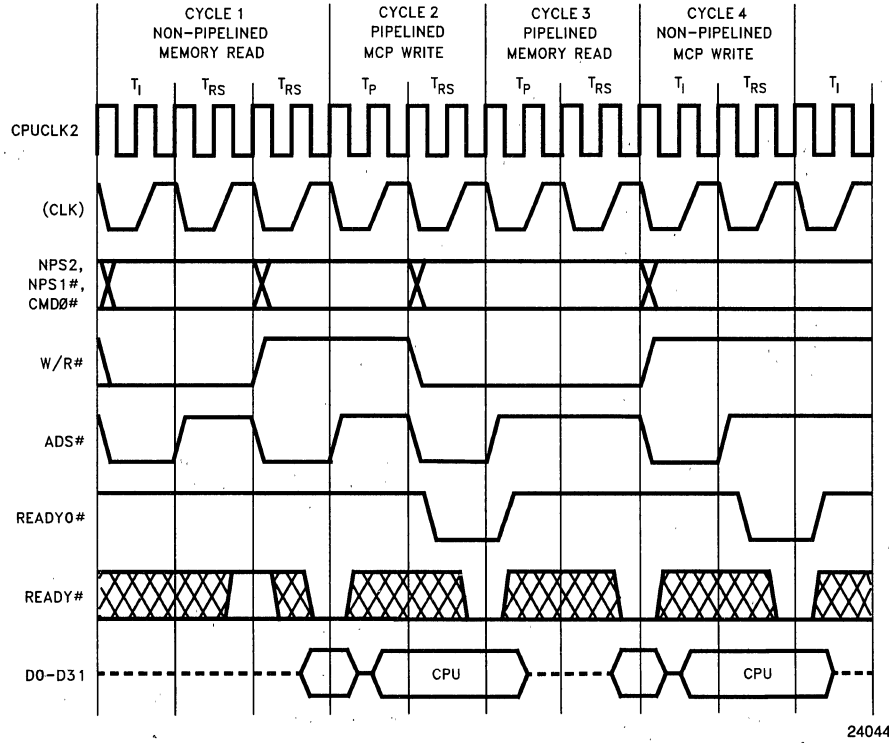
Figure 3.7 shows the pipelined write and read cycles with one additional T_{RS} states beyond the minimum required. To delay the assertion of READY# requires external logic.

3.4.3 BUS CYCLES OF MIXED TYPE

When the Intel387 DX MCP bus logic is in the T_{RS} state, it distinguishes between nonpipelined and pipelined cycles according to the behavior of ADS# and READY#. In a nonpipelined cycle, only READY# is activated, and the transition is from T_{RS} to idle state. In a pipelined cycle, both READY# and ADS# are active and the transition is first from T_{RS} state to T_P state then, after one clock period, back to T_{RS} state.

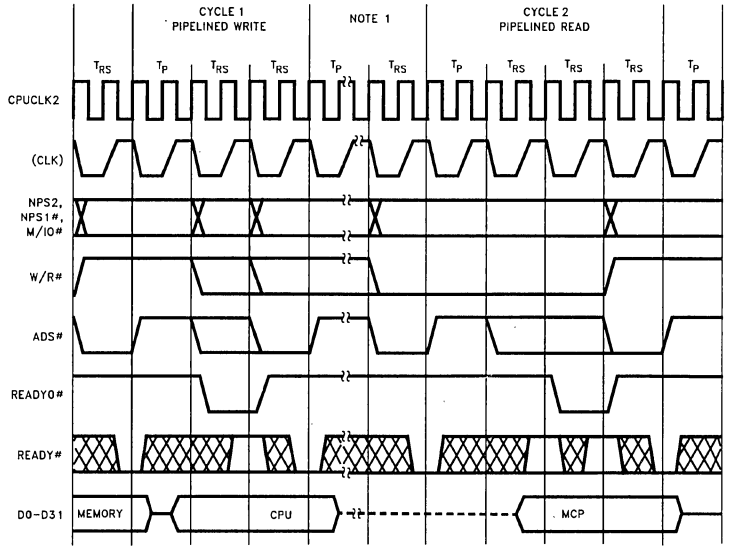
3.4.4 BUSY# AND PEREQ TIMING RELATIONSHIP

Figure 3.8 shows the activation of BUSY# at the beginning of instruction execution and its deactivation after execution of the instruction is complete. When possible, the Intel387 DX MCP may deactivate BUSY# prior to the completion of the current instruction allowing the CPU to transfer the next instruction's opcode and operands. PEREQ is activated in this interval. If ERROR# (not shown in the diagram) is ever asserted, it would occur at least six CPUCLK2 periods after the deactivation of PEREQ and at least six CPUCLK2 periods before the deactivation of BUSY#. Figure 3.8 shows also that STEN is activated at the beginning of a bus cycle.



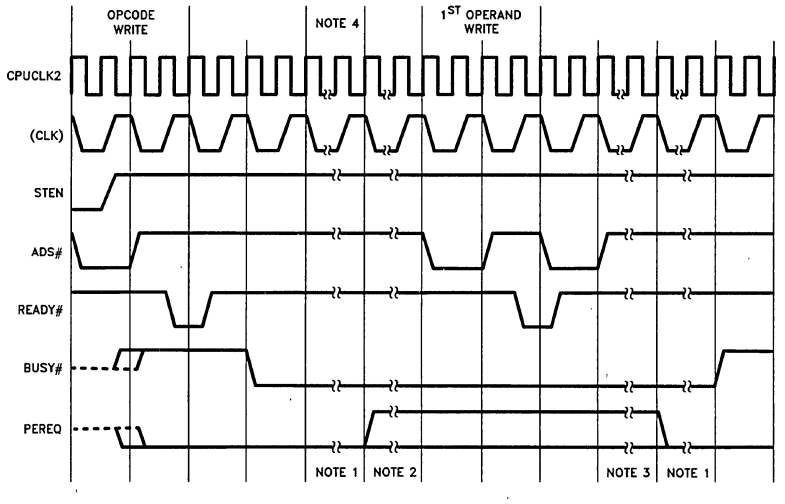
Cycle 1–Cycle 4 represent the operand transfer cycle for an instruction involving a transfer of two 32-bit loads in total. The opcode write cycles and other overhead are not shown.
Note that the next cycle will be a pipelined cycle if both READY# and ADS# are sampled active at the end of a T_{RS} state of the current cycle.

Figure 3.6. Fastest Transitions to and from Pipelined Cycles



NOTE:
1. Cycles between operand write to the MCP and storing result.

Figure 3.7. Pipelined Cycles with Wait States



NOTES:
1. Instruction dependent.
2. PEREQ is an asynchronous input to the Intel386™ DX Microprocessor; it may not be asserted (instruction dependent).
3. More operand transfers.
4. Memory read (operand) cycle is not shown.

Figure 3.8. STEN, BUSY# and PEREQ Timing Relationship

4.0 ELECTRICAL DATA

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

4.1 Absolute Maximum Ratings*

Case Temperature T_C
Under Bias ..... −65°C to +110°C
Storage Temperature ..... −65°C to +150°C
Voltage on Any Pin with
Respect to Ground ..... −0.5 to V_{CC} +0.5V
Power Dissipation..... 1.5W

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

4.2 DC Characteristics

Table 4.1. DC Specifications T_C = 0° to 85°C, V_{CC} = 5V ±5%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LO Voltage	−0.3	+0.8	V	(Note 1)
V_{IH}	Input HI Voltage	2.0	V_{CC} + 0.3	V	(Note 1)
V_{CL}	CPUCLK2 Input LO Voltage	−0.3	+0.8	V	
V_{CH}	CPUCLK2 Input HI Voltage	3.7	V_{CC} +0.3	V	
V_{OL}	Output LO Voltage		0.45	V	(Note 2)
V_{OH}	Output HI Voltage	2.4		V	(Note 3)
I_{CC}	Supply Current				
	NUMCLK2 = 32 MHz(4)		160	mA	I_{CC} typ. = 95 mA
	NUMCLK2 = 40 MHz(4)		180	mA	I_{CC} typ. = 105 mA
	NUMCLK2 = 50 MHz(4)		210	mA	I_{CC} typ. = 125 mA
	NUMCLK2 = 66.6 MHz(4)		250	mA	I_{CC} typ. = 150 mA
I_{LI}	Input Leakage Current		±15	μA	$0V \le V_{IN} \le V_{CC}$
I_{LO}	I/O Leakage Current		±15	μA	$0.45V \le V_O \le V_{CC}$
C_{IN}	Input Capacitance		10	pF	fc = 1 MHz
C_O	I/O or Output Capacitance		12	pF	fc = 1 MHz
C_{CLK}	Clock Capacitance		15	pF	fc = 1 MHz

NOTES:
1. This parameter is for all inputs, including NUMCLK2 but excluding CPUCLK2.
2. This parameter is measured at I_{OL} as follows:
   data = 4.0 mA
   READYO# = 2.5 mA
   ERROR#, BUSY#, PEREQ = 2.5 mA
3. This parameter is measured at I_{OH} as follows:
   data = 1.0 mA
   READYO# = 0.6 mA
   ERROR#, BUSY#, PEREQ = 0.6 mA
4. I_{CC} is measured at steady state, maximum capacitive loading on the outputs, CPUCLK2 at the same frequency as NUMCLK2.

4.3 AC Characteristics

Table 4.2a. i387 DX/i386 DX Interface and Execution Frequencies

i386 DX System Frequency (MHz)	i387 DX 16-33 Execution Frequency (MHz)	
	Min	Max
16 MHz	10.0 MHz	22.4 MHz
20 MHz	12.5 MHz	28.0 MHz
25 MHz	15.6 MHz	33.0 MHz
33 MHz	20.6 MHz	33.0 MHz

NOTE:
The external clock frequencies for the i387 DX and i386 DX are equal to twice the interface and execution frequencies show above.

Table 4.2b. Timing Requirements of the Execution Unit
T_C = 0°C to +85°C, V_{CC} = 5V ±5%

Pin	Symbol	Parameter	16 MHz–33 MHz		Test Conditions	Figure Reference
			Min (ns)	Max (ns)		
NUMCLK2	t1	Period	15	125	2.0V	4.1
NUMCLK2	t2a	High Time	6.25		2.0V	
NUMCLK2	t2b	High Time	4.5		3.7V	
NUMCLK2	t3a	Low Time	6.25		2.0V	
NUMCLK2	t3b	Low Time	4.5		0.8V	
NUMCLK2	t4	Fall Time		6	3.7V to 0.8V	
NUMCLK2	t5	Rise Time		6	0.8V to 2.7V	

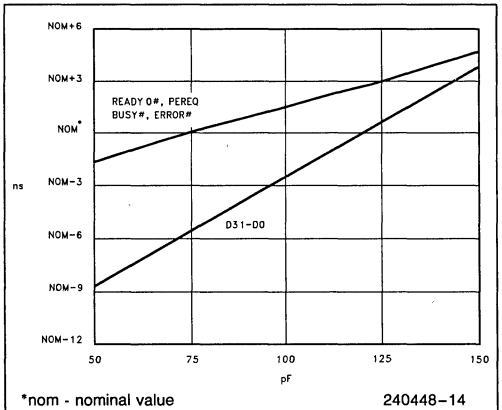
Table 4.2c. Timing Requirements of the Bus Interface Unit
T_C = 0°C to +85°C, V_{CC} = 5V ±5%
(All measurements made at 1.5V and 50 pF unless otherwise specified)

Pin	Symbol	Parameter	16 MHz–33 MHz		Test Conditions	Figure Reference
			Min (ns)	Max (ns)		
CPUCLK2	t1	Period	15	125	2.0V	4.1
CPUCLK2	t2a	High Time	6.25		2.0V	
CPUCLK2	t2b	High Time	4.5		3.7V	
CPUCLK2	t3a	Low Time	6.25		2.0V	
CPUCLK2	t3b	Low Time	4.5		0.8V	
CPUCLK2	t4	Fall Time		6	3.7V to 0.8V	
CPUCLK2	t5	Rise Time		6	0.8V to 3.7V	
NUMCLK2/ CPUCLK2		Ratio	10/16	14/10		
READYO#	t7	Out Delay	4	17		4.2
READYO#(1)	t7	Out Delay	4	15	C_L = 25 pF	
PEREQ	t7	Out Delay	4	25		
BUSY#	t7	Out Delay	4	21		
BUSY#(1)	t7	Out Delay	4	19	C_L = 25 pF	
ERROR#	t7	Out Delay	4	25		
D31–D0	t8	Out Delay	0	37		4.3
D31–D0	t10	Setup Time	8			
D31–D0	t11	Hold Time	8			
D31–D0(2)	t12	Float Time	3	19		

Table 4.2c. Timing Requirements of the Bus Interface Unit (Continued)
T_C = 0°C to +85°C, V_{CC} = 5V ±5%
(All measurements made at 1.5V and 50 pF unless otherwise specified)

Pin	Symbol	Parameter	16 MHz–33 MHz		Test Conditions	Figure Reference
			Min (ns)	Max (ns)		
PEREQ(2)	t13	Float Time	1	30		4.5
BUSY#(2)	t13	Float Time	1	30		
ERROR#(2)	t13	Float Time	1	30		
READYO#(2)	t13	Float Time	1	30		
ADS#	t14	Setup Time	13			4.3
ADS#	t15	Hold Time	4			
W/R#	t14	Setup Time	13			
W/R#	t15	Hold Time	4			
READY#	t16	Setup Time	7			
READY#	t17	Hold Time	4			
CMDO#	t16	Setup Time	13			
CMDO#	t17	Hold Time	2			
NPS1# NPS2	t16	Setup Time	13			
NPS1# NPS2	t17	Hold Time	2			
STEN	t16	Setup Time	13			
STEN	t17	Hold Time	2			
RESETIN	t18	Setup Time	5			4.4
RESETIN	t19	Hold Time	3			

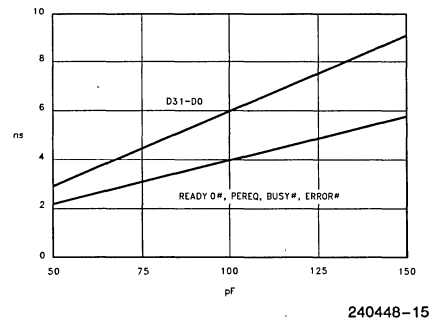
NOTES:
1. Not tested at 25 pF.
2. Float delay is not tested. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude.



*nom - nominal value 240448-14

NOTE:
This graph will not be linear outside of the C_L range shown.

Figure 4.0a. Typical Output Valid Delay vs Load Capacitance at Max Operating Temperature



240448-15

NOTE:
This graph will not be linear outside of the C_L range shown.

Figure 4.0b. Typical Output Rise Time vs Load Capacitance at Max Operating Temperature

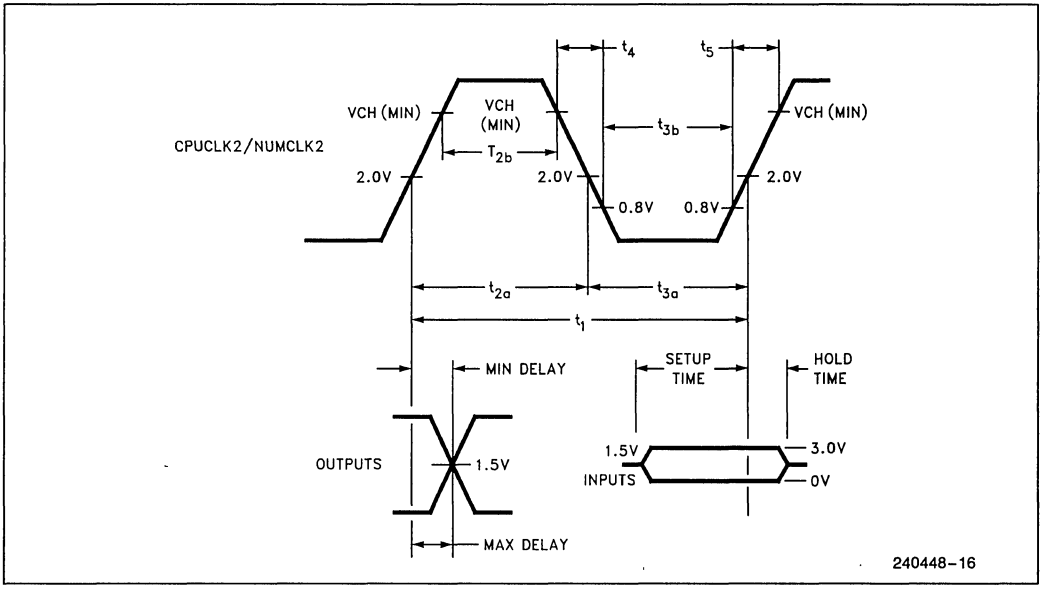


Figure 4.1. CPUCLK2/NUMCLK2 Waveform and Measurement Points for Input/Output A.C. Specifications

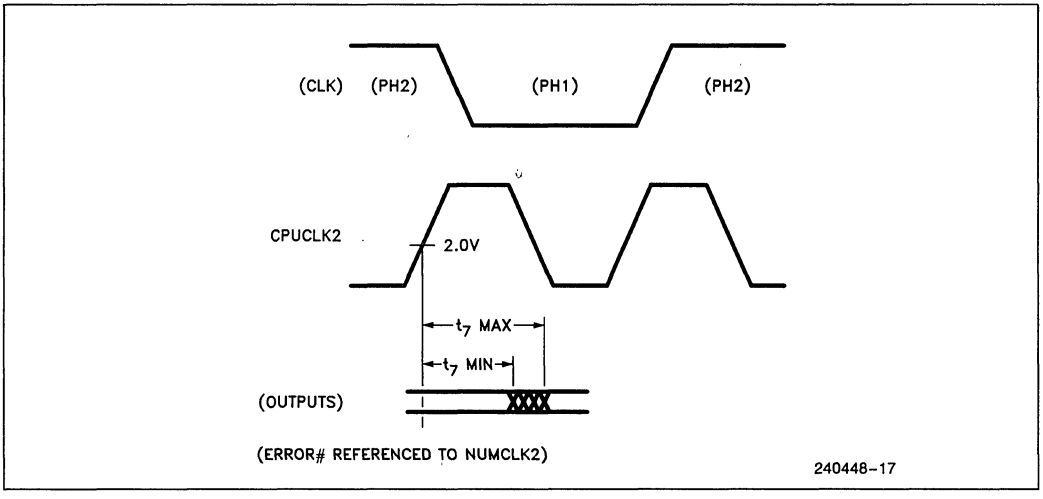


Figure 4.2. Output Signals

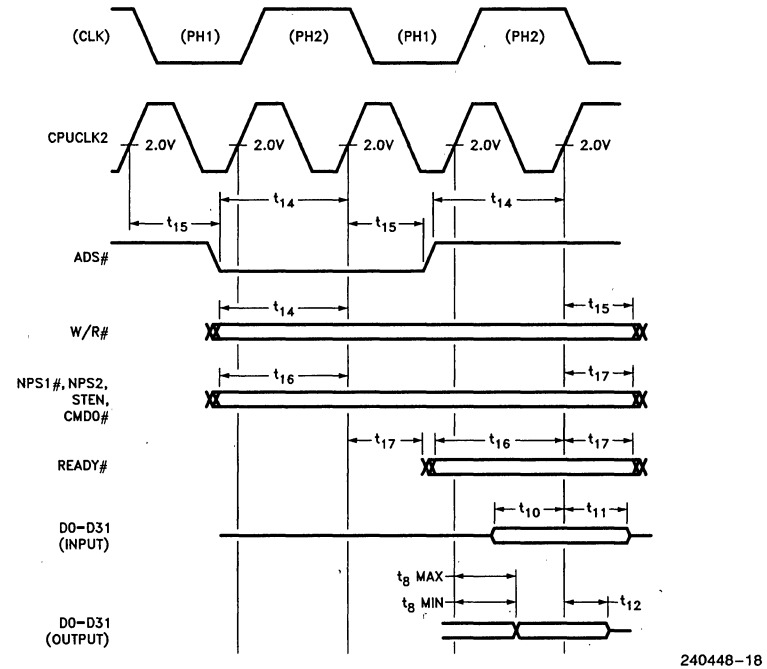
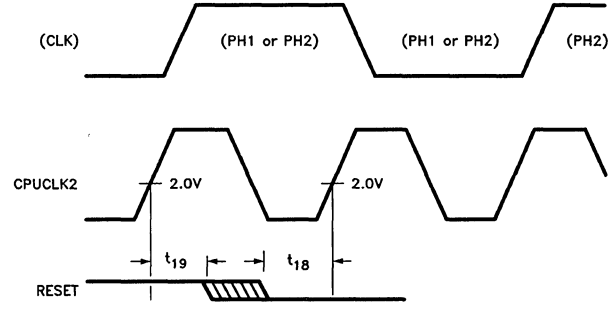


Figure 4.3. Input and I/O Signals



NOTE:
The second internal processor phase following RESET high to low transition is PH2.

Figure 4.4. RESET Signal

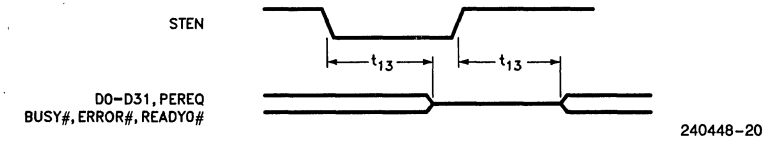
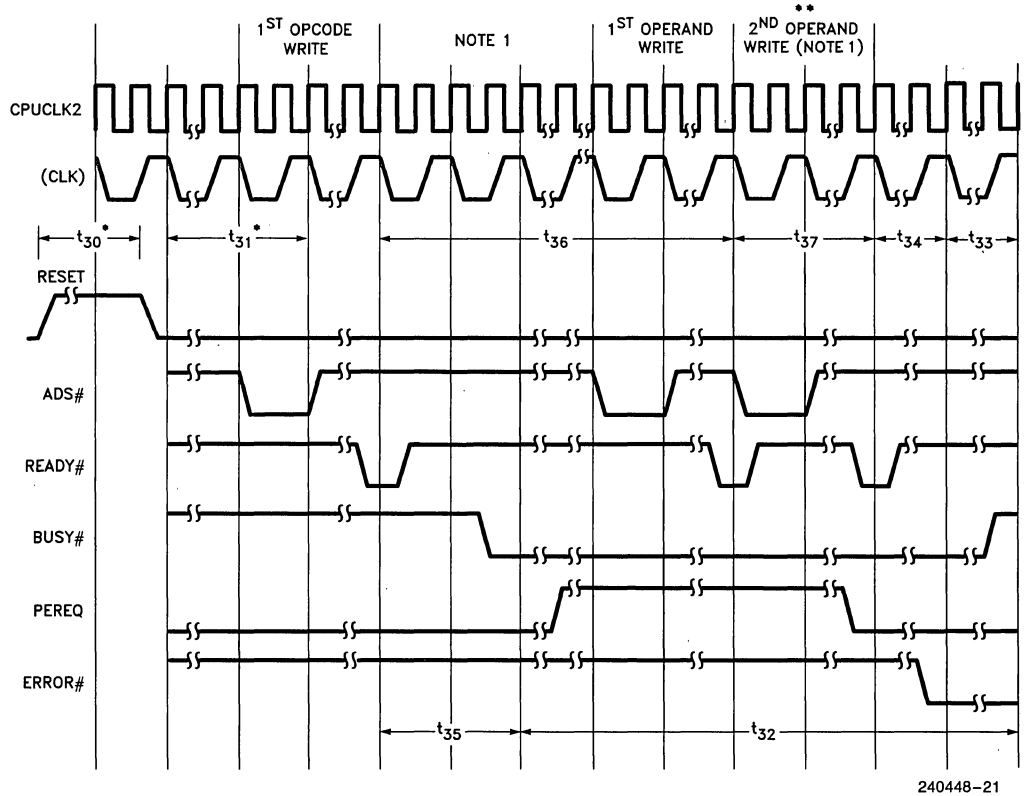


Figure 4.5. Float from STEN

Table 4.3. Other Parameters

Pin	Symbol	Parameter	Min	Max	Units
RESETIN	t30	Duration	40		NUMCLK2
RESETIN	t31	RESETIN Inactive to 1st Opcode Write	50		NUMCLK2
BUSY#	t32	Duration	6		CPUCLK2
BUSY#, ERROR#	t33	ERROR# (In) Active to BUSY# Inactive	6		CPUCLK2
PEREQ, ERROR#	t34	PEREQ Inactive to ERROR# Active	6		CPUCLK2
READY#, BUSY#	t35	READY# Active to BUSY# Active	4	4	CPUCLK2
READY#	t36	Minimum Time from Opcode Write to Opcode/Operand Write	6		CPUCLK2
READY#	t37	Minimum Time from Operand Write to Operand Write	8		CPUCLK2



* In NUMCLK2's
** or last operand

NOTE:
1. Memory read (operand) cycle is not shown.

Figure 4.6. Other Parameters

	Instruction								Optional Fields	
	First Byte				Second Byte					
1	11011	OPA		1	MOD		1 OPB	R/M	SIB	DISP
2	11011	MF		OPA	MOD		OPB	R/M	SIB	DISP
3	11011	d	P	OPA	1	1	OPB	ST(i)		
4	11011	0	0	1	1	1	1 OP			
5	11011	0	1	1	1	1	1 OP			
	15–11	10	9	8	7	6	5	4 3 2 1 0		

5.0 Intel387™ DX MCP EXTENSIONS TO THE Intel386™ DX CPU INSTRUCTION SET

Instructions for the Intel387 DX MCP assume one of the five forms shown in the following table. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the Intel386 DX CPU addressing modes.

OP = Instruction opcode, possible split into two fields OPA and OPB

MF = Memory Format
- 00—32-bit real
- 01—32-bit integer
- 10—64-bit real
- 11—16-bit integer

P = Pop
- 0—Do not pop stack
- 1—Pop stack after operation

ESC = 11011

d = Destination
- 0—Destination is ST(0)
- 1—Destination is ST(i)

R XOR d = 0—Destination (op) Source
R XOR d = 1—Source (op) Destination

ST(i) = Register stack element *i*
- 000 = Stack top
- 001 = Second stack element
- •
- •
- •
- 111 = Eighth stack element

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of the Intel386 DX Microprocessor instructions (refer to *Intel386™ DX Microprocessor Programmer's Reference Manual*).

SIB (Scale Index Base) byte and DISP (displacement) are optionally present in instructions that have MOD and R/M fields. Their presence depends on the values of MOD and R/M, as for Intel386 DX Microprocessor instructions.

The instruction summaries that follow assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD request delaying processor access to the bus; and that no exceptions are detected during instruction execution. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

Intel387™ DX MCP Extensions to the Intel386™ DX CPU Instruction Set

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
DATA TRANSFER							
FLD = Load[a]							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	SIB/DISP	9–18	26–42	16–23	42–53
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	SIB/DISP		26–54		
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	SIB/DISP		12–43		
BCD memory to ST(0)	ESC 111	MOD 100 R/M	SIB/DISP		45–97		
ST(i) to ST(0)	ESC 001	11000 ST(i)			7–12		
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	SIB/DISP	25–43	57–76	32–44	58–76
ST(0) to ST(i)	ESC 101	11010 ST(i)			7–11		
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	SIB/DISP	25–43	57–76	32–44	58–76
ST(0) to long integer memory	ESC 111	MOD 111 R/M	SIB/DISP		60–82		
ST(0) to extended real	ESC 011	MOD 111 R/M	SIB/DISP		46–52		
ST(0) to BCD memory	ESC 111	MOD 110 R/M	SIB/DISP		112–190		
ST(0) to ST(i)	ESC 101	11011 ST (i)			7–11		
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)			10–17		
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	SIB/DISP	13–25	34–52	14–27	39–62
ST(i) to ST(0)	ESC 000	11010 ST(i)			13–21		
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	SIB/DISP	13–25	34–52	14–27	39–62
ST(i) to ST(0)	ESC 000	11011 ST(i)			13–21		
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001			13–21		
FTST = Test ST(0)	ESC 001	1110 0100			17–25		
FUCOM = Unordered compare	ESC 101	11100 ST(i)			13–21		
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)			13–21		
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001			13–21		
FXAM = Examine ST(0)	ESC 001	11100101			24–37		
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110			10–17		
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000			15–22		
FLDPI = Load pi into ST(0)	ESC 001	1110 1011			26–36		
FLDL2T = Load $\log_2(10)$ into ST(0)	ESC 001	1110 1001			26–36		

Shaded areas indicate instructions not available in 8087/80287.

NOTE:
a. When loading single- or double-precision zero from memory, add 5 clocks.

Intel387™ DX MCP Extensions to the Intel386™ DX CPU Instruction Set (Continued)

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2-6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
CONSTANTS (Continued)							
FLDL2E = Load $\log_2(e)$ into ST(0)	ESC 001	1110 1010		26-36			
FLDLG2 = Load $\log_{10}(2)$ into ST(0)	ESC 001	1110 1100		25-35			
FLDLN2 = Load $\log_e(2)$ into ST(0)	ESC 001	1110 1101		26-38			
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	SIB/DISP	12-29	34-56	15-34	38-64
ST(i) and ST(0)	ESC d P 0	11000 ST(i)		12-26[b]			
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	SIB/DISP	12-29	34-56	15-34	38-64[c]
ST(i) and ST(0)	ESC d P 0	1110 R R/M		12-26[d]			
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	SIB/DISP	19-32	43-71	23-53	46-74
ST(i) and ST(0)	ESC d P 0	1100 1 R/M		17-50[e]			
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	SIB/DISP	77-85	101-114[f]	81-91	105-1249
ST(i) and ST(0)	ESC d P 0	1111 R R/M		77-80[h]			
FSQRT[i] = Square root	ESC 001	1111 1010		97-111			
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101		44-82			
FPREM = Partial remainder	ESC 001	1111 1000		56-140			
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101		81-168			
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100		41-62			
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100		42-63			
FABS = Absolute value of ST(0)	ESC 001	1110 0001		14-21			
FCHS = Change sign of ST(0)	ESC 001	1110 0000		17-24			

Shaded areas indicate instructions not available in 8087/80287.

NOTES:
b. Add 3 clocks to the range when d = 1.
c. Add 1 clock to **each** range when R = 1.
d. Add 3 clocks to the range when d = 0.
e. typical = 52 (When d = 0, 46-54, typical = 49).
f. Add 1 clock to the range when R = 1.
g. 135-141 when R = 1.
h. Add 3 clocks to the range when d = 1.
i. $-0 \leq ST(0) \leq +\infty$.

Intel387™ DX MCP Extensions to the Intel386™ DX CPU Instruction Set (Continued)

Instruction	Encoding			Clock Count Range
	Byte 0	Byte 1	Optional Bytes 2–6	
TRANSCENDENTAL				
FCOS[k] = Cosine of ST(0)	ESC 001	1111 1111		122–680
FPTAN[k] = Partial tangent of ST(0)	ESC 001	1111 0010		162–430[j]
FPATAN = Partial arctangent	ESC 001	1111 0011		250–420
FSIN[k] = Sine of ST(0)	ESC 001	1111 1110		121–680
FSINCOS[k] = Sine and cosine of ST(0)	ESC 001	1111 1011		150–650
F2XM1[l] = $2^{ST(0)} - 1$	ESC 001	1111 0000		167–410
FYL2X[m] = ST(1) * $\log_2(ST(0))$	ESC 001	1111 0001		99–436
FYL2XP1[n] = ST(1) * $\log_2(ST(0) + 1.0)$	ESC 001	1111 1001		210–447
PROCESSOR CONTROL				
FINIT = Initialize MCP	ESC 011	1110 0011		33
FSTSW AX = Store status word	ESC 111	1110 0000		13
FLDCW = Load control word	ESC 001	MOD 101 R/M	SIB/DISP	19
FSTCW = Store control word	ESC 101	MOD 111 R/M	SIB/DISP	15
FSTSW = Store status word	ESC 101	MOD 111 R/M	SIB/DISP	15
FCLEX = Clear exceptions	ESC 011	1110 0010		11
FSTENV = Store environment	ESC 001	MOD 110 R/M	SIB/DISP	103–104
FLDENV = Load environment	ESC 001	MOD 100 R/M	SIB/DISP	71
FSAVE = Save state	ESC 101	MOD 110 R/M	SIB/DISP	375–376
FRSTOR = Restore state	ESC 101	MOD 100 R/M	SIB/DISP	308
FINCSTP = Increment stack pointer	ESC 001	1111 0111		21
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		22
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)		18
FNOP = No operations	ESC 001	1101 0000		12

Shaded areas indicate instructions not available in 8087/80287.

NOTES:
j. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 76 additional clocks may be needed to reduce the operand.
k. $0 \leq |ST(0)| < 2^{63}$.
l. $-1.0 \leq ST(0) \leq 1.0$.
m. $0 \leq ST(0) < \infty$, $-\infty < ST(1) < +\infty$.
n. $0 \leq |ST(0)| < (2 - SQRT(2))/2$, $-\infty < ST(1) < +\infty$.

APPENDIX A
COMPATIBILITY BETWEEN THE 80287 AND THE 8087

The 80286/80287 operating in Real-Address mode will execute 8086/8087 programs without major modification. However, because of differences in the handling of numeric exceptions by the 80287 MCP and the 8087 MCP, exception-handling routines *may* need to be changed.

This appendix summarizes the differences between the 80287 MCP and the 8087 MCP, and provides details showing how 8086/8087 programs can be ported to the 80286/80287.

1. The MCP signals exceptions through a dedicated ERROR# line to the 80286. The MCP error signal does not pass through an interrupt controller (the 8087 INT signal does). Therefore, any interrupt-controller-oriented instructions in numeric exception handlers for the 8086/8087 should be deleted.
2. The 8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the 80287. If the 80287 encounters one of these opcodes in its instruction stream, the instruction will effectively be ignored—none of the 80287 internal states will be updated. While 8086/8087 containing these instructions may be executed on the 80286/80287, it is unlikely that the exception-handling routines containing these instructions will be completely portable to the 80287.
3. Interrupt vector 16 must point to the numeric exception handling routine.
4. The ESC instruction address saved in the 80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
5. In Protected-Address mode, the format of the 80287's saved instruction and address pointers is different than for the 8087. The instruction opcode is not saved in Protected mode—exception handlers will have to retrieve the opcode from memory if needed.
6. Interrupt 7 will occur in the 80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the 80286 MSW set (TS = 1 or EM = 1). If TS is set, then a WAIT instruction will also cause interrupt 7. An exception handler should be included in 80286/80287 code to handle these situations.
7. Interrupt 9 will occur if the second or subsequent words of a floating-point operand fall outside a segment's size. Interrupt 13 will occur if the starting address of a numeric operand falls outside a segment's size. An exception handler should be included in 80286/80287 code to report these programming errors.
8. Except for the processor control instructions, all of the 80287 numeric instructions are automatically synchronized by the 80286 CPU—the 80286 automatically tests the BUSY# line from the 80287 to ensure that the 80287 has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the 8087 used with 8086 and 8088 processors, explicit WAITs are required before each numeric instruction to ensure synchronization. Although 8086/8087 programs having explicit WAIT instructions will execute perfectly on the 80286/80287 without reassembly, these WAIT instructions are unnecessary.
9. Since the 80287 does not require WAIT instructions before each numeric instruction, the ASM286 assembler does not automatically generate these WAIT instructions. The ASM86 assembler, however, automatically precedes every ESC instruction with a WAIT instruction. Although numeric routines generated using the ASM86 assembler will generally execute correctly on the 80286/80287, reassembly using ASM286 may result in a more compact code image.

   The processor control instructions for the 80287 may be coded using either a WAIT or No-WAIT form of mnemonic. The WAIT forms of these instructions cause ASM286 to precede the ESC instruction with a CPU WAIT instruction, in the identical manner as does ASM86.

DATA SHEET REVISION REVIEW

The following list represents the key differences between this and the -003 versions of the Intel387™ Math Coprocessor Data Sheet. Please review this summary carefully.

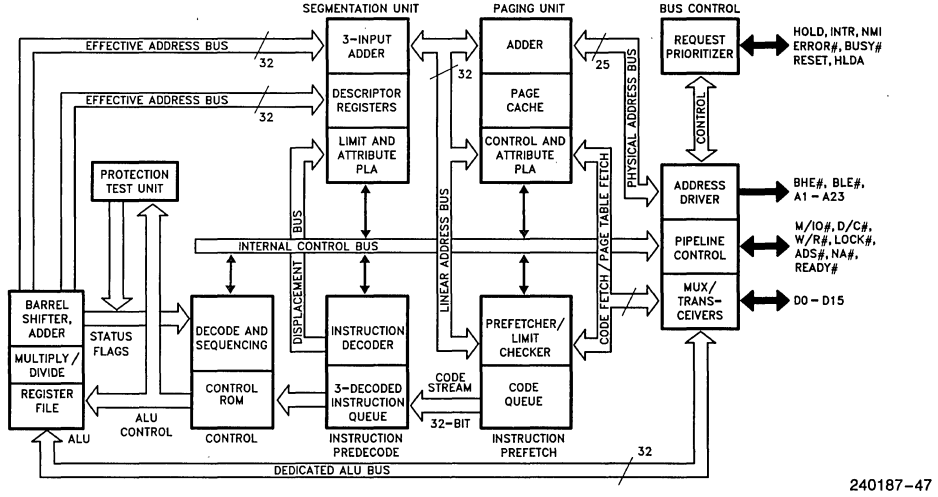
1. Corrected typographical errors.
2. Corrected clock ratio "PIN" name on Table 4.2c to NUMCLK/CPUCLK.

Intel386™ SX MICROPROCESSOR

- **Full 32-Bit Internal Architecture**
  - **— 8-, 16-, 32-Bit Data Types**
  - **— 8 General Purpose 32-Bit Registers**
- **Runs Intel386™ Software in a Cost Effective 16-Bit Hardware Environment**
  - **— Runs Same Applications and O.S.'s as the Intel386™ DX Processor**
  - **— Object Code Compatible with 8086, 80186, 80286, and Intel386™ Processors**
- **High Performance 16-Bit Data Bus**
  - **— 16, 20, 25 and 33 MHz Clock**
  - **— Two-Clock Bus Cycles**
  - **— Address Pipelining Allows Use of Slower/Cheaper Memories**
- **Integrated Memory Management Unit**
  - **— Virtual Memory Support**
  - **— Optional On-Chip Paging**
  - **— 4 Levels of Hardware Enforced Protection**
  - **— MMU Fully Compatible with Those of the 80286 and Intel386 DX CPUs**
- **Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System**
- **Large Uniform Address Space**
  - **— 16 Megabyte Physical**
  - **— 64 Terabyte Virtual**
  - **— 4 Gigabyte Maximum Segment Size**
- **Numerics Support with the Intel387™ SX Math CoProcessor**
- **On-Chip Debugging Support Including Breakpoint Registers**
- **Complete System Development Support**
  - **— Software: C, PL/M, Assembler**
  - **— Debuggers: PMON-386 DX, ICE™-386 SX**
- **High Speed CHMOS IV Technology**
- **Operating Frequency:**
  - **— Standard (Intel386 SX -33, -25, -20, -16) Min/Max Frequency (4/33, 4/25, 4/20, 4/16) MHz**
  - **— Low Power (Intel386 SX -33, -25, -20, -16, -12) Min/Max Frequency (2/33, 2/25, 2/20, 2/16, 2/12) MHz**
- **100-Pin Plastic Quad Flatpack Package**
  (See Packaging Outlines and Dimensions #231369)

3

The Intel386™ SX Microprocessor is an entry-level 32-bit CPU with a 16-bit external data bus and a 24-bit external address bus. The Intel386 SX CPU brings the vast software library of the Intel386™ Architecture to entry-level systems. It provides the performance benefits of a 32-bit programming architecture with the cost savings associated with 16-bit hardware systems.



240187-47

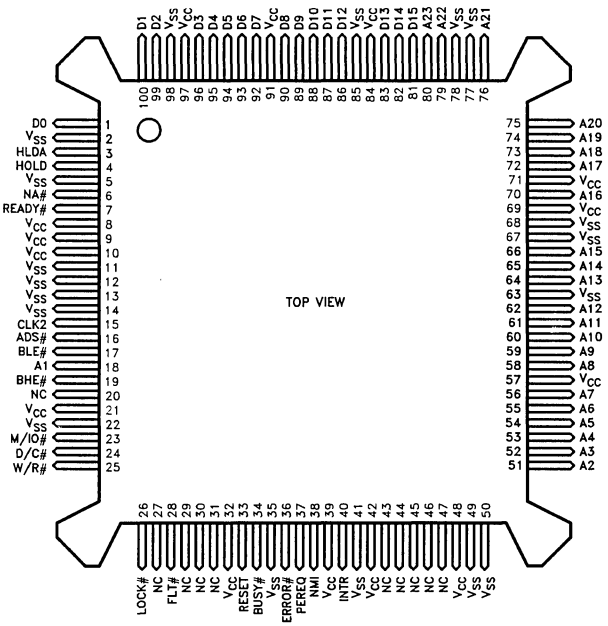
Intel386™ SX Pipelined 32-Bit Microarchitecture

Intel386™ SX MicroProcessor

CONTENTS

CONTENTS

1.0 PIN DESCRIPTION



240187-1

NOTE:
NC = No Connect

Figure 1.1. Intel386™ SX Microprocessor Pin out Top View

Table 1.1. Alphabetical Pin Assignments

Address		Data		Control		N/C	V_{CC}	V_{SS}
A_1	18	D_0	1	ADS#	16	20	8	2
A_2	51	D_1	100	BHE#	19	27	9	5
A_3	52	D_2	99	BLE#	17	29	10	11
A_4	53	D_3	96	BUSY#	34	30	21	12
A_5	54	D_4	95	CLK2	15	31	32	13
A_6	55	D_5	94	D/C#	24	43	39	14
A_7	56	D_6	93	ERROR#	36	44	42	22
A_8	58	D_7	92	FLT#	28	45	48	35
A_9	59	D_8	90	HLDA	3	46	57	41
A_{10}	60	D_9	89	HOLD	4	47	69	49
A_{11}	61	D_{10}	88	INTR	40		71	50
A_{12}	62	D_{11}	87	LOCK#	26		84	63
A_{13}	64	D_{12}	86	M/IO#	23		91	67
A_{14}	65	D_{13}	83	NA#	6		97	68
A_{15}	66	D_{14}	82	NMI	38			77
A_{16}	70	D_{15}	81	PEREQ	37			78
A_{17}	72			READY#	7			85
A_{18}	73			RESET	33			98
A_{19}	74			W/R#	25			
A_{20}	75							
A_{21}	76							
A_{22}	79							
A_{23}	80							

1.0 PIN DESCRIPTION (Continued)

The following are the Intel386™ SX Microprocessor pin descriptions. The following definitions are used in the pin descriptions:

#	The named signal is active LOW.
I	Input signal.
O	Output signal.
I/O	Input and Output signal.
-	No electrical connection.

Symbol	Type	Pin	Name and Function
CLK2	I	15	**CLK2** provides the fundamental timing for the Intel386 SX Microprocessor. For additional information see **Clock.**
RESET	I	33	**RESET** suspends any operation in progress and places the Intel386 SX Microprocessor in a known reset state. See **Interrupt Signals** for additional information.
D_{15}–D_0	I/O	81-83,86-90, 92-96,99-100,1	**Data Bus** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles. See **Data Bus** for additional information.
A_{23}–A_1	O	80-79,76-72,70, 66-64,62-58, 56-51,18	**Address Bus** outputs physical memory or port I/O addresses. See **Address Bus** for additional information.
W/R#	O	25	**Write/Read** is a bus cycle definition pin that distinguishes write cycles from read cycles. See **Bus Cycle Definition Signals** for additional information.
D/C#	O	24	**Data/Control** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and code fetch. See **Bus Cycle Definition Signals** for additional information.
M/IO#	O	23	**Memory/IO** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles. See **Bus Cycle Definition Signals** for additional information.
LOCK#	O	26	**Bus Lock** is a bus cycle definition pin that indicates that other system bus masters are not to gain control of the system bus while it is active. See **Bus Cycle Definition Signals** for additional information.
ADS#	O	16	**Address Status** indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BHE#, BLE# and A_{23}–A_1 are being driven at the Intel386 SX Microprocessor pins. See **Bus Control Signals** for additional information.
NA#	I	6	**Next Address** is used to request address pipelining. See **Bus Control Signals** for additional information.
READY#	I	7	**Bus Ready** terminates the bus cycle. See **Bus Control Signals** for additional information.
BHE#, BLE#	O	19,17	**Byte Enables** indicate which data bytes of the data bus take part in a bus cycle. See **Address Bus** for additional information.

1.0 PIN DESCRIPTION (Continued)

Symbol	Type	Pin	Name and Function
HOLD	I	4	**Bus Hold Request** input allows another bus master to request control of the local bus. See **Bus Arbitration Signals** for additional information.
HLDA	O	3	**Bus Hold Acknowledge** output indicates that the Intel386 SX Microprocessor has surrendered control of its local bus to another bus master. See **Bus Arbitration Signals** for additional information.
INTR	I	40	**Interrupt Request** is a maskable input that signals the Intel386 SX Microprocessor to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** for additional information.
NMI	I	38	**Non-Maskable Interrupt Request** is a non-maskable input that signals the Intel386 SX Microprocessor to suspend execution of the current program and execute an interrupt acknowledge function. See **Interrupt Signals** for additional information.
BUSY#	I	34	**Busy** signals a busy condition from a processor extension. See **Coprocessor Interface Signals** for additional information.
ERROR#	I	36	**Error** signals an error condition from a processor extension. See **Coprocessor Interface Signals** for additional information.
PEREQ	I	37	**Processor Extension Request** indicates that the processor has data to be transferred by the Intel386 SX Microprocessor. See **Coprocessor Interface Signals** for additional information.
FLT#	I	28	**Float** is an input which forces all bidirectional and output signals, including HLDA, to the tri-state condition. This allows the electrically isolated Intel386SX PQFP to use ONCE (On-Circuit Emulation) method without removing it from the PCB. See **Float** for additional information.
N/C	-	20, 27, 29-31, 43-47	**No Connects** should always be left unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the Intel386 SX Microprocessor.
V_{CC}	I	8-10,21,32,39 42,48,57,69, 71,84,91,97	**System Power** provides the +5V nominal DC supply input.
V_{SS}	I	2,5,11-14,22 35,41,49-50, 63,67-68, 77-78,85,98	**System Ground** provides the 0V connection from which all inputs and outputs are measured.

INTRODUCTION

The Intel386 SX Microprocessor is 100% object code compatible with the Intel386 DX, 286 and 8086 microprocessors. Systems based on the Intel386 SX CPU can access the world's largest existing microcomputer software base, including the growing 32-bit software base.

Instruction pipelining and a high performance ALU ensure short average instruction execution times and high system throughput.

The integrated memory management unit (MMU) includes an address translation cache, multi-tasking hardware, and a four-level hardware-enforced protection mechanism to support operating systems. The virtual machine capability of the Intel386 SX CPU allows simultaneous execution of applications from multiple operating systems.

The Intel386 SX CPU offers on-chip testability and debugging features. Four breakpoint registers allow conditional or unconditional breakpoint traps on code execution or data accesses for powerful debugging of even ROM-based systems. Other testability features include self-test, tri-state of output buffers, and direct access to the page translation cache.

The Low Power Intel386 SX CPU brings the benefits of the Intel386 Microprocessor 32-bit architecture to Laptop and Notebook personal computer applications. With its power saving 2 MHz sleep-mode and extended functional temperature range of 0°C to 100°C T_{CASE}, the Lower Power Intel386 SX CPU specifically satisfies the power consumption and heat dissipation requirements of today's small form factor computers.

2.0 BASE ARCHITECTURE

The Intel386 SX Microprocessor consists of a central processing unit, a memory management unit and a bus interface.

The central processing unit consists of the execution unit and the instruction unit. The execution unit contains the eight 32-bit general purpose registers which are used for both address calculation and data operations and a 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations. The instruction unit decodes the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

The memory management unit (MMU) consists of a segmentation unit and a paging unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability, and efficient sharing. The paging mechanism operates beneath and is transparent to the segmentation process, to allow management of the physical address space.

The segmentation unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of integrity.

The Intel386 SX Microprocessor has two modes of operation: Real Address Mode (Real Mode), and Protected Virtual Address Mode (Protected Mode). In Real Mode the Intel386 SX Microprocessor operates as a very fast 8086, but with 32-bit extensions if desired. Real Mode is required primarily to set up the processor for Protected Mode operation.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each such task behaves with 8086 semantics, thus allowing 8086 software (an application program or an entire operating system) to execute. The Virtual 8086 tasks can be isolated and protected from one another and the host Intel386 SX Microprocessor operating system by use of paging.

Finally, to facilitate system hardware designs, the Intel386 SX Microprocessor bus interface offers address pipelining and direct Byte Enable signals for each byte of the data bus.

2.1 Register Set

The Intel386 SX Microprocessor has thirty-four registers as shown in Figure 2-1. These registers are grouped into the following seven categories:

General Purpose Registers: The eight 32-bit general purpose registers are used to contain arithmetic and logical operands. Four of these (EAX, EBX, ECX, and EDX) can be used either in their entirety as 32-bit registers, as 16-bit registers, or split into pairs of separate 8-bit registers.

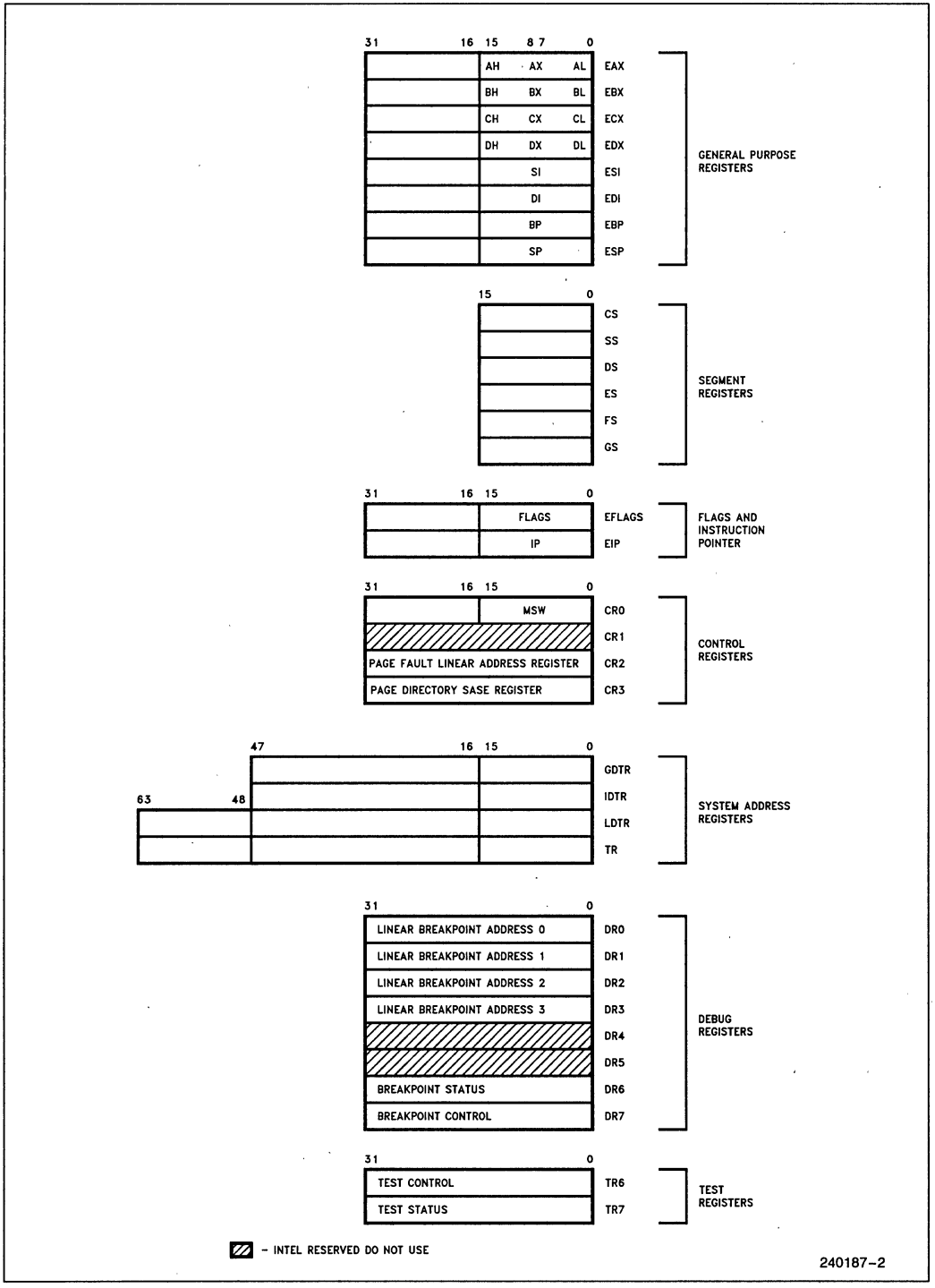


Figure 2.1. Intel386™ SX Microprocessor Registers

Segment Registers: Six 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data.

Flags and Instruction Pointer Registers: The two 32-bit special purpose registers in figure 2.1 record or control certain aspects of the Intel386 SX Microprocessor state. The EFLAGS register includes status and control bits that are used to reflect the outcome of many instructions and modify the semantics of some instructions. The Instruction Pointer, called EIP, is 32 bits wide. The Instruction Pointer controls instruction fetching and the processor automatically increments it after executing an instruction.

Control Registers: The four 32-bit control register are used to control the global nature of the Intel386 SX Microprocessor. The CR0 register contains bits that set the different processor modes (Protected, Real, Paging and Coprocessor Emulation). CR2 and CR3 registers are used in the paging operation.

System Address Registers: These four special registers reference the tables or segments supported by the 80286/Intel386 SX/Intel386 DX CPU's protection model. These tables or segments are:

GDTR (Global Descriptor Table Register),
IDTR (Interrupt Descriptor Table Register),
LDTR (Local Descriptor Table Register),
TR (Task State Segment Register).

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. The use of the debug registers is described in Section 2.10 **Debugging Support**.

Test Registers: Two registers are used to control the testing of the RAM/CAM (Content Addressable Memories) in the Translation Lookaside Buffer portion of the Intel386 SX Microprocessor. Their use is discussed in **Testability**.

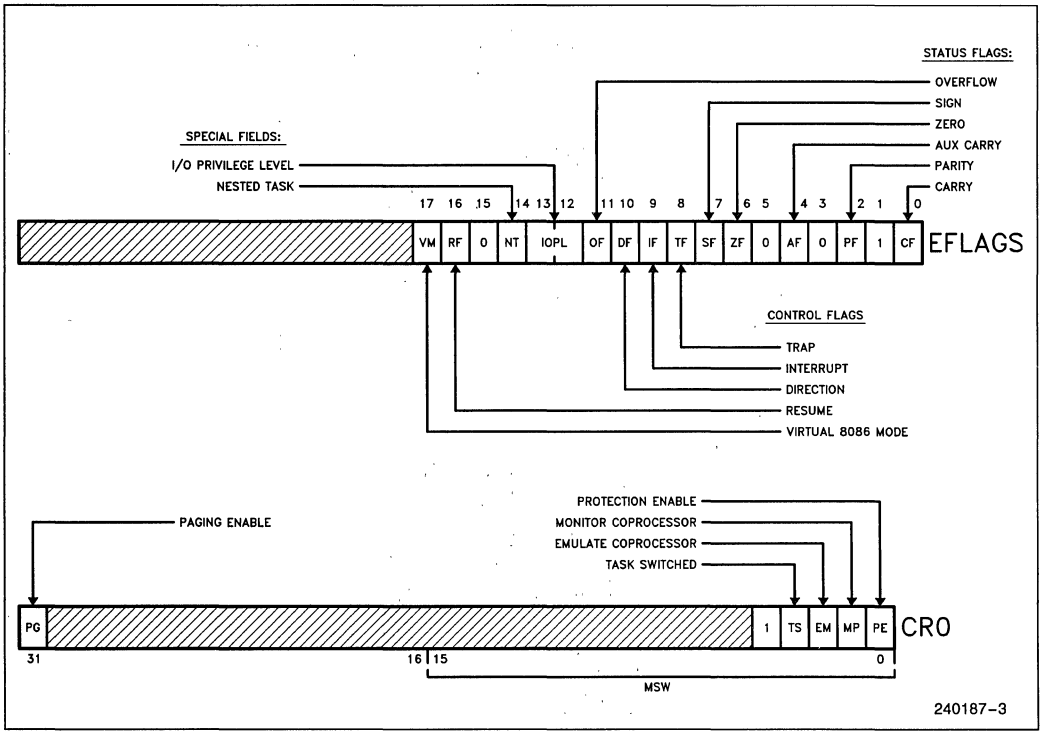


Figure 2.2. Status and Control Register Bit Functions

EFLAGS REGISTER

The flag register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS, shown in Figure 2.2, control certain operations and indicate the status of the Intel386 SX Microprocessor. The lower 16 bits (bits 0–15) of EFLAGS contain the 16-bit flag register named FLAGS. This is the default flag register used when executing 8086, 80286, or real mode code. The functions of the flag bits are given in Table 2.1.

CONTROL REGISTERS

The Intel386 SX Microprocessor has three control registers of 32 bits, CR0, CR2 and CR3, to hold the machine state of a global nature. These registers are shown in Figures 2.1 and 2.2. The defined CR0 bits are described in Table 2.2.

Table 2.1. Flag Definitions

Bit Position	Name	Function
0	CF	Carry Flag—Set on high-order bit carry or borrow; cleared otherwise.
2	PF	Parity Flag—Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise.
4	AF	Auxiliary Carry Flag—Set on carry from or borrow to the low order four bits of AL; cleared otherwise.
6	ZF	Zero Flag—Set if result is zero; cleared otherwise.
7	SF	Sign Flag—Set equal to high-order bit of result (0 if positive, 1 if negative).
8	TF	Single Step Flag—Once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt.
9	IF	Interrupt-Enable Flag—When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	Direction Flag—Causes string instructions to auto-increment (default) the appropriate index registers when cleared. Setting DF causes auto-decrement.
11	OF	Overflow Flag—Set if the operation resulted in a carry/borrow into the sign bit (high-order bit) of the result but did not result in a carry/borrow out of the high-order bit or vice-versa.
12,13	IOPL	I/O Privilege Level—Indicates the maximum Current Privilege Level (CPL) permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O permission bit map while executing in protected mode. For virtual 86 mode it indicates the maximum CPL allowing alteration of the IF bit. See Section 4.2 for a further discussion and definitions on various privilege levels.
14	NT	Nested Task—Set if the execution of the current task is nested within another task. Cleared otherwise.
16	RF	Resume Flag—Used in conjunction with debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. If set, any debug fault is ignored on the next instruction.
17	VM	Virtual 8086 Mode—If set while in protected mode, the Intel386 SX Microprocessor will switch to virtual 8086 operation, handling segment loads as the 8086 does, but generating exception 13 faults on privileged opcodes.

Table 2.2. CR0 Definitions

Bit Position	Name	Function
0	PE	Protection mode enable—places the Intel386 SX Microprocessor into protected mode. If PE is reset, the processor operates again in Real Mode. PE may be set by loading MSW or CR0. PE can be reset only by loading CR0, it cannot be reset by the LMSW instruction.
1	MP	Monitor coprocessor extension—allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	Emulate processor extension—causes a processor extension not present exception (number 7) on ESC instructions to allow emulating a processor extension.
3	TS	Task switched—indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task.
31	PG	Paging enable bit—is set to enable the on-chip paging unit. It is reset to disable the on-chip paging unit.

2.2 Instruction Set

The instruction set is divided into nine categories of operations:

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High Level Language Support
Operating System Support
Processor Control

These instructions are listed in Table 9.1 **Instruction Set Clock Count Summary**.

All Intel386 SX Microprocessor instructions operate on either 0, 1, 2 or 3 operands; an operand resides in a register, in the instruction itself, or in memory. Most zero operand instructions (e.g CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2 bytes long. Since the Intel386 SX Microprocessor has a 16 byte prefetch instruction queue, an average of 5 instructions will be prefetched. The use of two operands permits the following types of common instructions:

Register to Register
Memory to Register
Immediate to Register
Memory to Memory
Register to Memory
Immediate to Memory.

The operands can be either 8, 16, or 32 bits long. As a general rule, when executing code written for the Intel386 SX Microprocessor (32-bit code), operands are 8 or 32 bits; when executing existing 8086 or 80286 code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to all instructions which override the default length of the operands (i.e. use 32-bit operands for 16-bit code, or 16-bit operands for 32-bit code).

2.3 Memory Organization

Memory on the Intel386 SX Microprocessor is divided into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address. The address of a word or dword is the byte address of the low-order byte.

In addition to these basic data types, the Intel386 SX Microprocessor supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4K byte pages. Finally, both segmentation and paging can be combined, gaining the advantages of both systems. The Intel386 SX Microprocessor supports both pages and segmentation in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and as such is a tool for the application programmer, while pages are useful to the system programmer for managing the physical memory of a system.

ADDRESS SPACES

The Intel386 SX Microprocessor has three types of address spaces: **logical**, **linear**, and **physical**. A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, DISPLACEMENT), discussed in section 2.4 **Addressing Modes**, into an effective address. This effective address along with the selector is known as the logical address. Since each task on the Intel386 SX Microprocessor has a maximum of 16K ($2^{14}-1$) selectors, and offsets can be 4 gigabytes (with paging enabled) this gives a total of 2^{46} bits, or 64 terabytes, of **logical** address space per task. The programmer sees the logical address space.

The segmentation unit translates the **logical** address space into a 32-bit **linear** address space. If the paging unit is not enabled then the 32-bit **linear** address is truncated into a 24-bit **physical** address. The **physical address** is what appears on the address pins.

The primary differences between Real Mode and Protected Mode are how the segmentation unit performs the translation of the **logical** address into the **linear** address, size of the address space, and paging capability. In Real Mode, the segmentation unit shifts the selector left four bits and adds the result to the effective address to form the **linear** address. This **linear** address is limited to 1 megabyte. In addition, real mode has no paging capability.

Protected Mode will see one of two different address spaces, depending on whether or not paging is enabled. Every selector has a **logical base** address associated with it that can be up to 32 bits in length. This 32-bit **logical base** address is added to the effective address to form a final 32-bit **linear** address. If paging is disabled this final **linear** address reflects physical memory and is truncated so that only the lower 24 bits of this address are used to address the 16 megabyte memory address space. If paging is enabled this final **linear** address reflects a 32-bit address that is translated through the paging unit to form a 16-megabyte physical address. The **logical base** address is stored in one of two operating system tables (i.e. the Local Descriptor Table or Global Descriptor Table).

Figure 2.3 shows the relationship between the various address spaces.

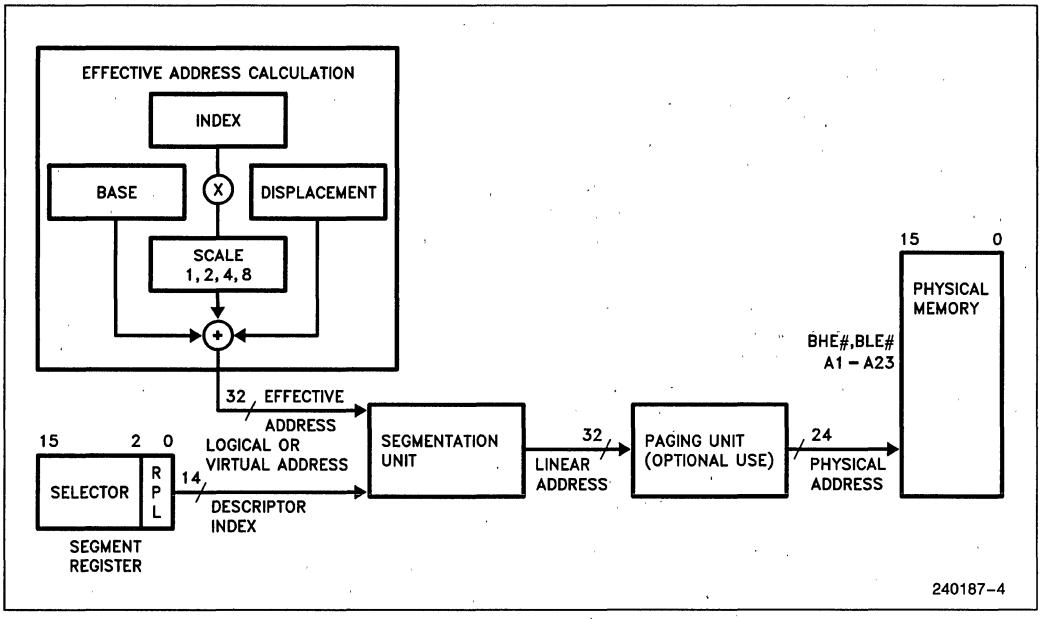


Figure 2.3. Address Translation

SEGMENT REGISTER USAGE

The main data structure used to organize memory is the segment. On the Intel386 SX Microprocessor, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments, code and data. The segments are of variable size and can be as small as 1 byte or as large as 4 gigabytes (2^{32} bits).

In order to provide compact instruction encoding and increase processor performance, instructions do not need to explicitly specify which segment register is used. The segment register is automatically chosen according to the rules of Table 2.3 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register, stack references use the SS register and instruction fetches use the CS register. The contents of the Instruction Pointer provide the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 2.3. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero and create a system with a four gigabyte linear address space. This creates a system where the virtual address space is the same as the linear address space. Further details of segmentation are discussed in chapter 4 **PROTECTED MODE ARCHITECTURE**.

2.4 Addressing Modes

The Intel386 SX Microprocessor provides a total of 8 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

REGISTER AND IMMEDIATE MODES

Two of the addressing modes provide for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8, 16 or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the opcode.

Table 2.3. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructons	SS	None
Source of POP, POPA, POPF, IRET, RET Instructions	SS	None
Destination of STOS, MOVE, REP STOS, and REP MOVS instructions	ES	None
Other data references, with effective address using base register of:		
[EAX]	DS	CS,SS,ES,FS,GS
[EBX]	DS	CS,SS,ES,FS,GS
[ECX]	DS	CS,SS,ES,FS,GS
[EDX]	DS	CS,SS,ES,FS,GS
[ESI]	DS	CS,SS,ES,FS,GS
[EDI]	DS	CS,SS,ES,FS,GS
[EBP]	SS	CS,DS,ES,FS,GS
[ESP]	SS	CS,DS,ES,FS,GS

32-BIT MEMORY ADDRESSING MODES

The remaining 6 modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by summing any combination of the following three address elements (see Figure 2.3):

DISPLACEMENT: an 8, 16 or 32-bit immediate value, following the instruction.

BASE: The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area.

INDEX: The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters. The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. The scaled index is especially useful for accessing arrays or structures.

Combinations of these 3 components make up the 6 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of Base and Index components which requires one additional clock.

As shown in Figure 2.4, the effective address (EA) of an operand is calculated according to the following formula:

$$EA = Base_{Register} + (Index_{Register} * scaling) + Displacement$$

1. **Direct Mode:** The operand's offset is contained as part of the instruction as an 8, 16 or 32-bit displacement.
2. **Register Indirect Mode:** A BASE register contains the address of the operand.
3. **Based Mode:** A BASE register's contents are added to a DISPLACEMENT to form the operand's offset.
4. **Scaled Index Mode:** An INDEX register's contents are multiplied by a SCALING factor, and the result is added to a DISPLACEMENT to form the operand's offset.
5. **Based Scaled Index Mode:** The contents of an INDEX register are multiplied by a SCALING factor, and the result is added to the contents of a BASE register to obtain the operand's offset.
6. **Based Scaled Index Mode with Displacement:** The contents of an INDEX register are multiplied by a SCALING factor, and the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.

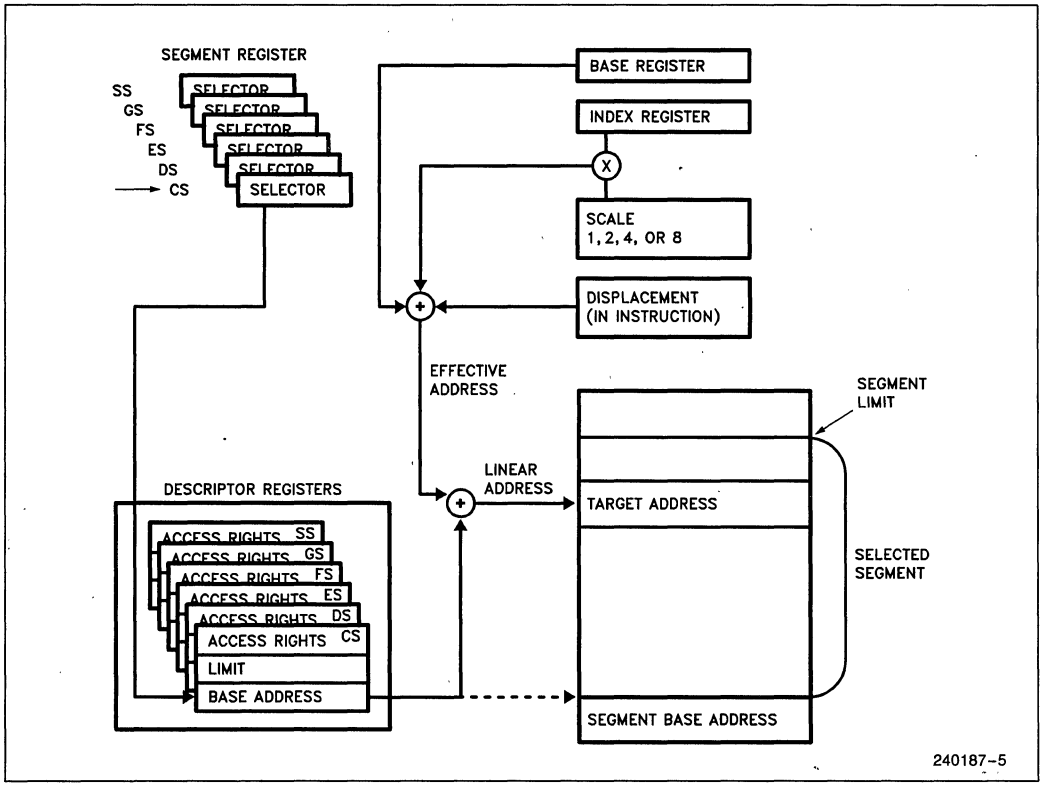


Figure 2.4. Addressing Mode Calculations

DIFFERENCES BETWEEN 16 AND 32 BIT ADDRESSES

In order to provide software compatibility with the 8086 and the 80286, the Intel386 SX Microprocessor can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in a Segment Descriptor. If the D bit is 0 then all operand lengths and effective addresses are assumed to be 16 bits long. If the D bit is 1 then the default length for operands and addresses is 32 bits. In Real Mode the default size for operands and addresses is 16 bits.

Regardless of the default precision of the operands or addresses, the Intel386 SX Microprocessor is able to execute either 16 or 32-bit instructions. This is specified through the use of override prefixes. Two prefixes, the **Operand Length Prefix** and the **Address Length Prefix**, override the value of the D bit on an individual instruction basis. These prefixes are automatically added by assemblers.

The Operand Length and Address Length Prefixes can be applied separately or in combination to any instruction. The Address Length Prefix does not allow addresses over 64K bytes to be accessed in Real Mode. A memory address which exceeds 0FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional Intel386 SX Microprocessor addressing modes.

When executing 32-bit code, the Intel386 SX Microprocessor uses either 8 or 32-bit displacements, and any register can be used as base or index registers. When executing 16-bit code, the displacements are either 8 or 16-bits, and the base and index register conform to the 80286 model. Table 2.4 illustrates the differences.

Table 2.4. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX,BP	Any 32-bit GP Register
INDEX REGISTER	SI,DI	Any 32-bit GP Register Except ESP
SCALE FACTOR	None	1, 2, 4, 8
DISPLACEMENT	0, 8, 16-bits	0, 8, 32-bits

2.5 Data Types

The Intel386 SX Microprocessor supports all of the data types commonly used in high level languages:

Bit: A single bit quantity.

Bit Field: A group of up to 32 contiguous bits, which spans a maximum of four bytes.

Bit String: A set of contiguous bits; on the Intel386 SX Microprocessor, bit strings can be up to 4 gigabits long.

Byte: A signed 8-bit quantity.

Unsigned Byte: An unsigned 8-bit quantity.

Integer (Word): A signed 16-bit quantity.

Long Integer (Double Word): A signed 32-bit quantity. All operations assume a 2's complement representation.

Unsigned Integer (Word): An unsigned 16-bit quantity.

Unsigned Long Integer (Double Word): An unsigned 32-bit quantity.

Signed Quad Word: A signed 64-bit quantity.

Unsigned Quad Word: An unsigned 64-bit quantity.

Pointer: A 16 or 32-bit offset-only quantity which indirectly references another memory location.

Long Pointer: A full pointer which consists of a 16-bit segment selector and either a 16 or 32-bit offset.

Char: A byte representation of an ASCII Alphanumeric or control character.

String: A contiguous sequence of bytes, words or dwords. A string may contain between 1 byte and 4 gigabytes.

BCD: A byte (unpacked) representation of decimal digits 0–9.

Packed BCD: A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

When the Intel386 SX Microprocessor is coupled with its numerics coprocessor, the Intel387 SX, then the following common floating point types are supported:

Floating Point: A signed 32, 64, or 80-bit real number representation. Floating point numbers are supported by the Intel387 SX numerics coprocessor.

Figure 2.5 illustrates the data types supported by the Intel386 SX Microprocessor and the Intel387 SX.

2.6 I/O Space

The Intel386 SX Microprocessor has two distinct physical address spaces: physical memory and I/O. Generally, peripherals are placed in I/O space although the Intel386 SX Microprocessor also supports memory-mapped peripherals. The I/O space consists of 64K bytes which can be divided into 64K 8-bit ports or 32K 16-bit ports, or any combination of ports which add up to no more than 64K bytes. The 64K I/O address space refers to physical addresses rather than linear addresses since I/O instructions do not go through the segmentation or paging hardware. The M/IO# pin acts as an additional address line, thus allowing the system designer to easily determine which address space the processor is accessing.

The I/O ports are accessed by the IN and OUT instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8-bit and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/IO# pin to be driven LOW. I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

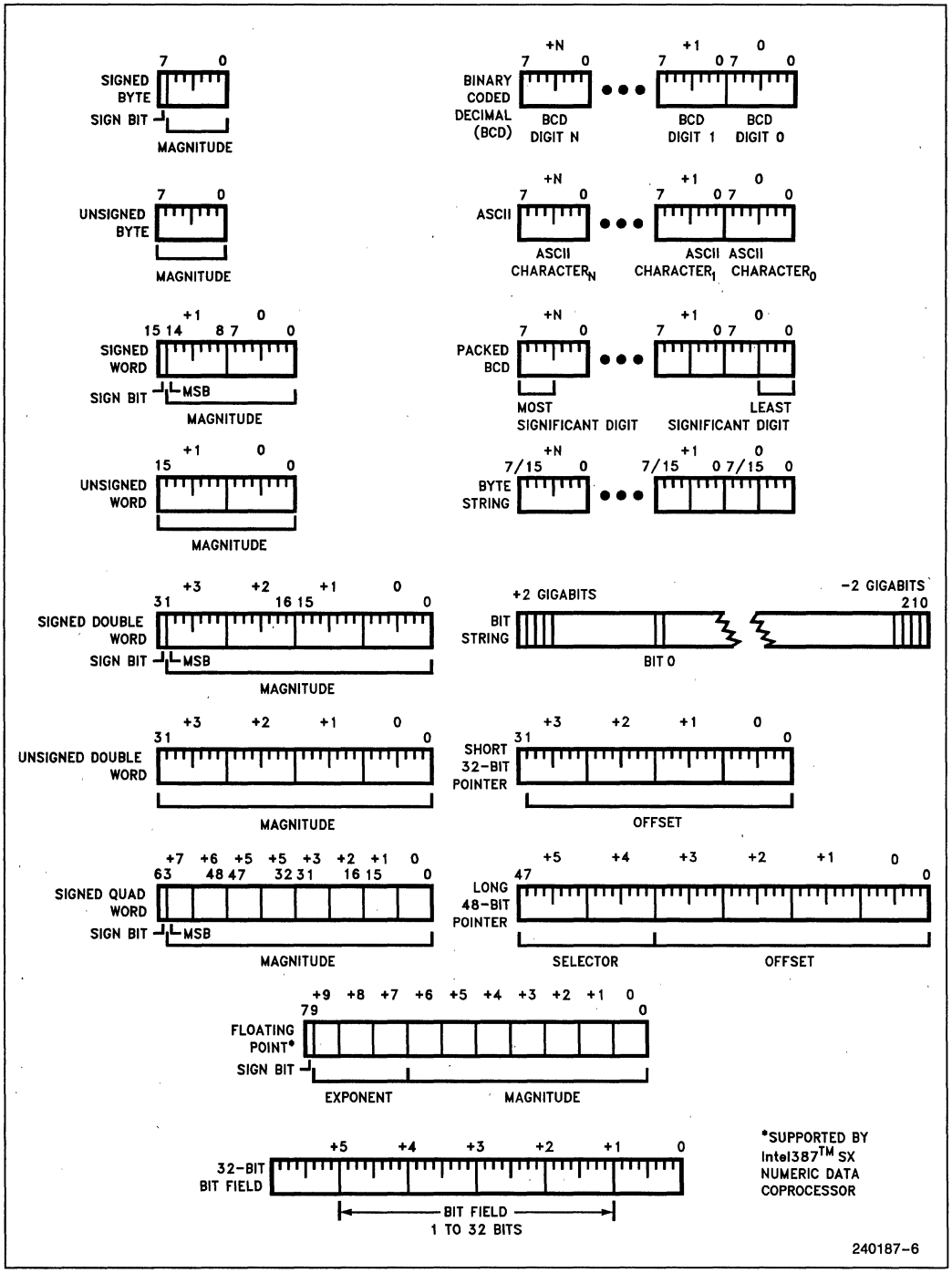


Figure 2.5. Intel386™ SX Microprocessor Supported Data Types

2.7 Interrupts and Exceptions

Interrupts and exceptions alter the normal program flow in order to handle external events, report errors or exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction.

Exceptions are classified as faults, traps, or aborts, depending on the way they are reported and whether or not restart of the instruction causing the exception is supported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction which caused the problem. **Aborts** are exceptions which do not permit the precise location of the instruction causing the exception to be determined.

Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point to the instruction causing the exception and will include any leading instruction prefixes. Table 2.5 summarizes the possible interrupts for the Intel386 SX Microprocessor and shows where the return address points to.

3

Table 2.5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	any instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any illegal instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any instruction that can generate an exception		ABORT
Coprocessor Segment Overrun	9	ESC	NO	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	11	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Coprocessor Error	16	ESC, WAIT	YES	FAULT
Intel Reserved	17–32			
Two Byte Interrupt	33–255	INT n	NO	TRAP

*Some debug exceptions may report both traps on the previous instruction and faults on the next instruction.

The Intel386 SX Microprocessor has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode, the vectors are 4-byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8 byte quantities, which are put in an Interrupt Descriptor Table. Of the 256 possible interrupts, 32 are reserved for use by Intel and the remaining 224 are free to be used by the system designer.

INTERRUPT PROCESSING

When an interrupt occurs, the following actions happen. First, the current program address and Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the Intel386 SX Microprocessor which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the Intel386 SX Microprocessor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

Maskable Interrupt

Maskable interrupts are the most common way to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled HIGH and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions (string instructions have an 'interrupt window' between memory moves which allows interrupts during long string moves). When an interrupt occurs the processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt (one of 224 user defined interrupts).

Interrupts through interrupt gates automatically reset IF, disabling INTR requests. Interrupts through Trap Gates leave the state of the IF bit unchanged. Interrupts through a Task Gate change the IF bit according to the image of the EFLAGs register in the task's Task State Segment (TSS). When an IRET instruction is executed, the original state of the IF bit is restored.

Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. When the NMI input is pulled HIGH it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for an NMI.

While executing the NMI servicing procedure, the Intel386 SX Microprocessor will not service any further NMI request or INT requests until an interrupt return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

Software Interrupts

A third type of interrupt/exception for the Intel386 SX Microprocessor is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the n^{th} vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt is the single step interrupt. It is discussed in **Single Step Trap**.

INTERRUPT AND EXCEPTION PRIORITIES

Interrupts are externally generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are **both** recognized at the **same** instruction boundary, the Intel386 SX Microprocessor invokes the NMI service routine first. If maskable interrupts are still enabled after the NMI service routine has been invoked, then the Intel386 SX Microprocessor will invoke the appropriate interrupt service routine.

As the Intel386 SX Microprocessor executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 2.6. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution.

INSTRUCTION RESTART

The Intel386 SX Microprocessor fully supports restarting all instructions after Faults. If an exception is detected in the instruction to be executed (exception categories 4 through 10 in Table 2.6), the Intel386 SX Microprocessor invokes the appropriate exception service routine. The Intel386 SX Microprocessor is in a state that permits restart of the instruction, for all cases but those given in Table 2.7. Note that all such cases will be avoided by a properly designed operating system.

Table 2.6. Sequence of Exception Checking

Consider the case of the Intel386 SX Microprocessor having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2. Check for external NMI and INTR.
3. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
4. Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5. Check for Page Faults that prevented fetching the entire next instruction (exception 14).
6. Check for Faults decoding the next instruction (exception 6 if illegal opcode; exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only; or exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e. not at IOPL or at CPL = 0).
7. If WAIT opcode, check if TS = 1 and MP = 1 (exception 7 if both are 1).
8. If ESCape opcode for numeric coprocessor, check if EM = 1 or TS = 1 (exception 7 if either are 1).
9. If WAIT opcode or ESCape opcode for numeric coprocessor, check ERROR# input signal (exception 16 if ERROR# input is asserted).
10. Check in the following order for each memory reference required by the instruction:
    a. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13).
    b. Check for Page Faults that prevent transferring the entire memory quantity (exception 14).

NOTE:
Segmentation exceptions are generated before paging exceptions.

Table 2.7. Conditions Preventing Instruction Restart

1. An instruction causes a task switch to a task whose Task State Segment is **partially** 'not present' (An entirely 'not present' TSS is restartable). Partially present TSS's can be avoided either by keeping the TSS's of such tasks present in memory, or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4K bytes or less).
2. A coprocessor operand wraps around the top of a 64K-byte segment or a 4G-byte segment, and spans three pages, and the page holding the middle portion of the operand is 'not present'. This condition can be avoided by starting **at a page boundary** any segments containing coprocessor operands if the segments are approximately 64K-200 bytes or larger (i.e. large enough for wraparound of the coprocessor operand to possibly occur).

Note that these conditions are avoided by using the operating system designs mentioned in this table.

Table 2.8. Register Values after Reset

Flag Word (EFLAGS)	uuuu0002H	Note 1
Machine Status Word (CR0)	uuuuuu10H	
Instruction Pointer (EIP)	0000FFF0H	
Code Segment (CS)	F000H	Note 2
Data Segment (DS)	0000H	Note 3
Stack Segment (SS)	0000H	
Extra Segment (ES)	0000H	Note 3
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
EAX register	0000H	Note 4
EDX register	component and stepping ID	Note 5
All other registers	undefined	Note 6

NOTES:
1. EFLAG Register. The upper 14 bits of the EFLAGS register are undefined, all defined flag bits are zero.
2. The Code Segment Register (CS) will have its Base Address set to 0FFFF0000H and Limit set to 0FFFFH.
3. The Data and Extra Segment Registers (DS, ES) will have their Base Address set to 000000000H and Limit set to 0FFFFH.
4. If self-test is selected, the EAX register should contain a 0 value. If a value of 0 is not found then the self-test has detected a flaw in the part.
5. EDX register always holds component and stepping identifier.
6. All undefined bits are Intel Reserved and should not be used.

DOUBLE FAULT

A Double Fault (exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13), but in the process of doing so detects an exception **other than** a Page Fault (exception 14).

One other cause of generating a Double Fault is the Intel386 SX Microprocessor detecting any other exception when it is attempting to invoke the Page Fault (exception 14) service routine (for example, if a Page Fault is detected when the Intel386 SX Microprocessor attempts to invoke the Page Fault service routine). Of course, in any functional system, not only in Intel386 SX Microprocessor-based systems, the entire page fault service routine must remain 'present' in memory.

2.8 Reset and Initialization

When the processor is initialized or Reset the registers have the values shown in Table 2.8. The Intel386 SX Microprocessor will then start executing instructions near the top of physical memory, at location 0FFFFFF0H. When the first Intersegment Jump or Call is executed, address lines A_{20}–A_{23} will drop LOW for CS-relative memory cycles, and the Intel386 SX Microprocessor will only execute instructions in the lower one megabyte of physical memory. This allows the system designer to use a shadow ROM at the top of physical memory to initialize the system and take care of Resets.

RESET forces the Intel386 SX Microprocessor to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350 and 450 CLK2 periods after Reset becomes inactive, the Intel386 SX Microprocessor will start executing instructions at the top of physical memory.

2.9 Testability

The Intel386 SX Microprocessor, like the Intel386 Microprocessor, offers testability features which include a self-test and direct access to the page translation cache.

SELF-TEST

The Intel386 SX Microprocessor has the capability to perform a self-test. The self-test checks the function of all of the Control ROM and most of the non-random logic of the part. Approximately one-half of the Intel386 SX Microprocessor can be tested during self-test.

Self-Test is initiated on the Intel386 SX Microprocessor when the RESET pin transitions from HIGH to LOW, and the BUSY# pin is LOW. The self-test takes about 2^{20} clocks, or approximately 33 milliseconds with a 16 MHz Intel386 SX CPU. At the completion of self-test the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX are zero. If the results of the EAX are not zero then the self-test has detected a flaw in the part.

TLB TESTING

The Intel386 SX Microprocessor also provides a mechanism for testing the Translation Lookaside Buffer (TLB) if desired. This particular mechanism may not be continued in the same way in future processors.

There are two TLB testing operations: 1) writing entries into the TLB, and, 2) performing TLB lookups. Two Test Registers, shown in Figure 2.6, are provided for the purpose of testing. TR6 is the "test command register", and TR7 is the "test data register". For a more detailed explanation of testing the TLB, see the Intel386™ SX Microprocessor Programmer's Reference Manual.

2.10 Debugging Support

The Intel386 SX Microprocessor provides several features which simplify the debugging process. The three categories of on-chip debugging aids are:

1. The code execution breakpoint opcode (0CCH).
2. The single-step capability provided by the TF bit in the flag register.
3. The code and data breakpoint capability provided by the Debug Registers DR0–3, DR6, and DR7.

BREAKPOINT INSTRUCTION

A single-byte software interrupt (Int 3) breakpoint instruction is available for use by software debuggers. The breakpoint opcode is 0CCh, and generates an exception 3 trap when executed.

SINGLE-STEP TRAP

If the single-step flag (TF, bit 8) in the EFLAG register is found to be set at the end of an instruction, a single-step exception occurs. The single-step exception is auto vectored to exception number 1.

DEBUG REGISTERS

The Debug Registers are an advanced debugging feature of the Intel386 SX Microprocessor. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT 3 breakpoint opcode.

The Intel386 SX Microprocessor contains six Debug Registers, consisting of four breakpoint address registers and two breakpoint control registers. Initially after reset, breakpoints are in the disabled state; therefore, no breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are auto-vectored to exception 1. Figure 2.7 shows the breakpoint status and control registers.

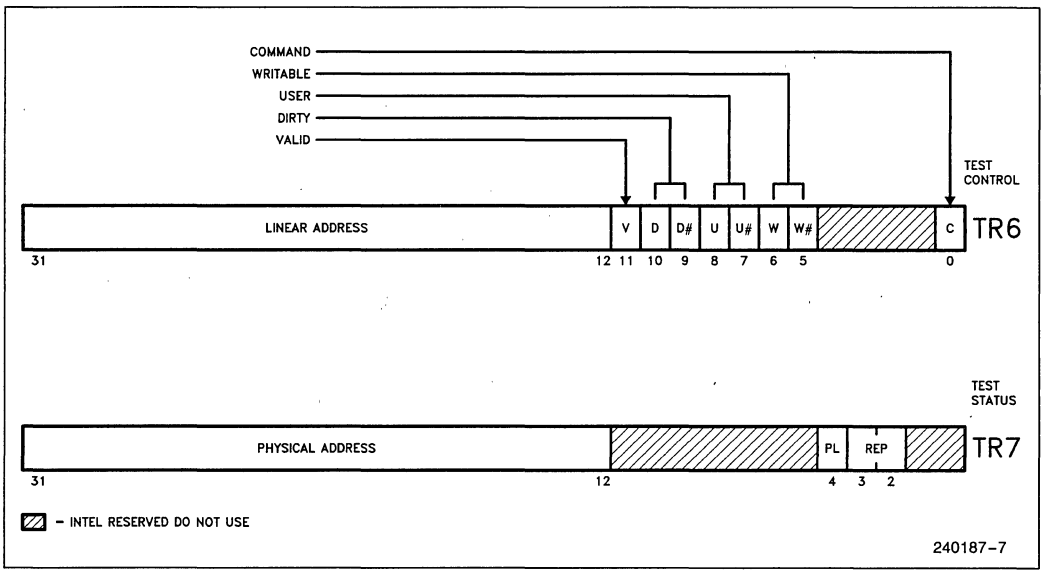


Figure 2.6. Test Registers

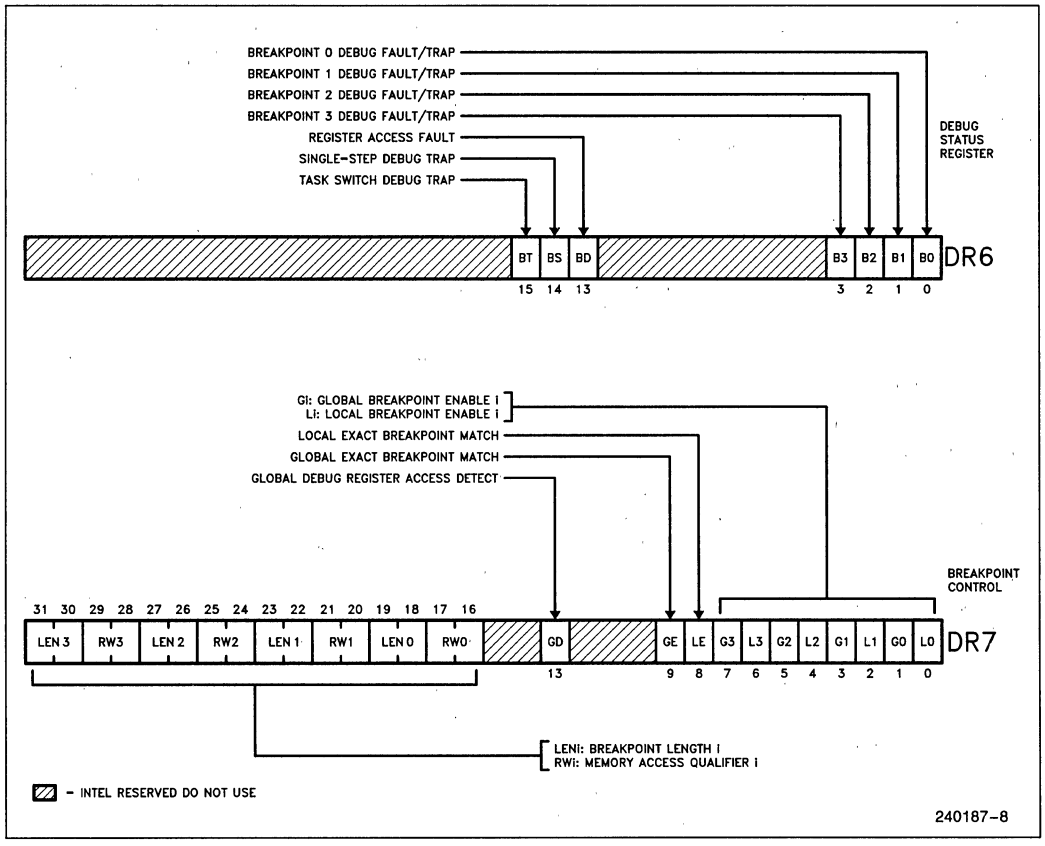


Figure 2.7. Debug Registers

3.0 REAL MODE ARCHITECTURE

When the processor is reset or powered up it is initialized in Real Mode. Real Mode has the same base architecture as the 8086, but allows access to the 32-bit register set of the Intel386 SX Microprocessor. The addressing mechanism, memory size, and interrupt handling are all identical to the Real Mode on the 80286.

The default operand size in Real Mode is 16 bits, as in the 8086. In order to use the 32-bit registers and addressing modes, override prefixes must be used. In addition, the segment size on the Intel386 SX Microprocessor in Real Mode is 64K bytes so 32-bit addresses must have a value less then 0000FFFFH. The primary purpose of Real Mode is to set up the processor for Protected Mode operation.

3.1 Memory Addressing

In Real Mode the linear addresses are the same as physical addresses (paging is not allowed). Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register which is shifted left by four bits to an effective address. This addition results in a 20-bit physical address or a 1 megabyte address space. Since segment registers are shifted left by 4 bits, Real Mode segments always start on 16-byte boundaries.

All segments in Real Mode are exactly 64K bytes long, and may be read, written, or executed. The Intel386 SX Microprocessor will generate an exception 13 if a data operand or instruction fetch occurs past the end of a segment.

Table 3.1. Exceptions in Real Mode

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT vector is not within table limit	Before Instruction
CS, DS, ES, FS, GS Segment overrun exception	13	Word memory reference with offset = 0FFFFH. an attempt to execute past the end of CS segment.	Before Instruction
SS Segment overrun exception	12	Stack Reference beyond offset = 0FFFFH	Before Instruction

3.2 Reserved Locations

There are two fixed areas in memory which are reserved in Real address mode: the system initialization area and the interrupt table area. Locations 00000H through 003FFH are reserved for interrupt vectors. Each one of the 256 possible interrupts has a 4-byte jump vector reserved for it. Locations 0FFFFF0H through 0FFFFFFH are reserved for system initialization.

3.3 Interrupts

Many of the exceptions discussed in section 2.7 are not applicable to Real Mode operation; in particular, exceptions 10, 11 and 14 do not occur in Real Mode. Other exceptions have slightly different meanings in Real Mode; Table 3.1 identifies these exceptions.

3.4 Shutdown and Halt

The HLT instruction stops program execution and prevents the processor from using the local bus until restarted. Either NMI, FLT#, INTR with interrupts enabled (IF = 1), or RESET will force the Intel386 SX Microprocessor out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

Shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions:

1. An interrupt or an exception occurs (Exceptions 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table.
2. A CALL, INT or PUSH instruction attempts to wrap around the stack segment when SP is not even.

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 000FH) and the stack has enough room to contain the vector and flag information (i.e. SP is greater that 0005H). Otherwise, shutdown can only be exited by a processor reset.

3.5 LOCK Operation

The LOCK prefix on the Intel386 SX Microprocessor, even in Real Mode, is more restrictive than on the 80286. This is due to the addition of paging on the Intel386 SX Microprocessor in Protected Mode and Virtual 8086 Mode. The LOCK prefix is not supported during repeat string instructions.

The only instruction forms where the LOCK prefix is legal on the Intel386 SX Microprocessor are shown in Table 3.2.

Table 3.2. Legal Instructions for the LOCK Prefix

Opcode	Operands (Dest, Source)
BIT Test and SET/RESET /COMPLEMENT	Mem, Reg/Immediate
XCHG	Reg, Mem
XCHG	Mem, Reg
ADD, OR, ADC, SBB, AND, SUB, XOR	Mem, Reg/Immediate
NOT, NEG, INC, DEC	Mem

An exception 6 will be generated if a LOCK prefix is placed before any instruction form or opcode not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above.

The LOCK prefix is not IOPL-sensitive on the Intel386 SX Microprocessor. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed in Table 3.2.

4.0 PROTECTED MODE ARCHITECTURE

The complete capabilities of the Intel386 SX Microprocessor are unlocked when the processor operates in Protected Virtual Address Mode (Protected Mode). Protected Mode vastly increases the linear address space to four gigabytes (2^{32} bytes) and allows the running of virtual memory programs of almost unlimited size (64 terabytes (2^{46} bytes)). In addition, Protected Mode allows the Intel386 SX Microprocessor to run all of the existing Intel386 DX CPU (using only 16 megabytes of physical memory), 80286 and 8086 CPU's software, while providing a sophisticated memory management and a hardware-assisted protection mechanism. Protected Mode allows the use of additional instructions specially optimized for supporting multitasking operating systems. The base architecture of the Intel386 SX Microprocessor remains the same; the registers, instructions, and addressing modes described in the previous sections are retained. The main difference between Protected Mode and Real Mode from a programmer's viewpoint is the increased address space and a different addressing mechanism.

4.1 Addressing Mechanism

Like Real Mode, Protected Mode uses two components to form the logical address; a 16-bit selector is used to determine the linear base address of a segment, the base address is added to a 32-bit effective address to form a 32-bit linear address. The linear address is then either used as a 24-bit physical address, or if paging is enabled the paging mechanism maps the 32-bit linear address into a 24-bit physical address.

The difference between the two modes lies in calculating the base address. In Protected Mode, the selector is used to specify an index into an operating system defined table (see Figure 4.1). The table contains the 32-bit base address of a given segment. The physical address is formed by adding the base address obtained from the table to the offset.

Paging provides an additional memory management mechanism which operates only in Protected Mode. Paging provides a means of managing the very large segments of the Intel386 SX Microprocessor, as paging operates beneath segmentation. The page mechanism translates the protected linear address which comes from the segmentation unit into a physical address. Figure 4.2 shows the complete Intel386 SX Microprocessor addressing mechanism with paging enabled.

4.2 Segmentation

Segmentation is one method of memory management. Segmentation provides the basis for protection. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about each segment is stored in an 8 byte data structure called a descriptor. All of the descriptors in a system are contained in descriptor tables which are recognized by hardware.

TERMINOLOGY

The following terms are used throughout the discussion of descriptors, privilege levels and protection:

PL: Privilege Level—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged.

RPL: Requestor Privilege Level—The privilege level of the original supplier of the selector. RPL is determined by the least two significant bits of a selector.

DPL: Descriptor Privilege Level—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

CPL: Current Privilege Level—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: Effective Privilege Level—The effective privilege level is the least privileged of the RPL and the DPL. EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

DESCRIPTOR TABLES

The descriptor tables define all of the segments which are used in a Intel386 SX Microprocessor system. There are three types of tables which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Descriptor Table. All of the tables are variable length memory arrays and can vary in size from 8 bytes to 64K bytes. Each table can hold up to 8192 8-byte descriptors. The upper 13 bits of a selector are used as an index into the descriptor table. The tables have registers associated with them which hold the 32-bit linear base address and the 16-bit limit of each table.

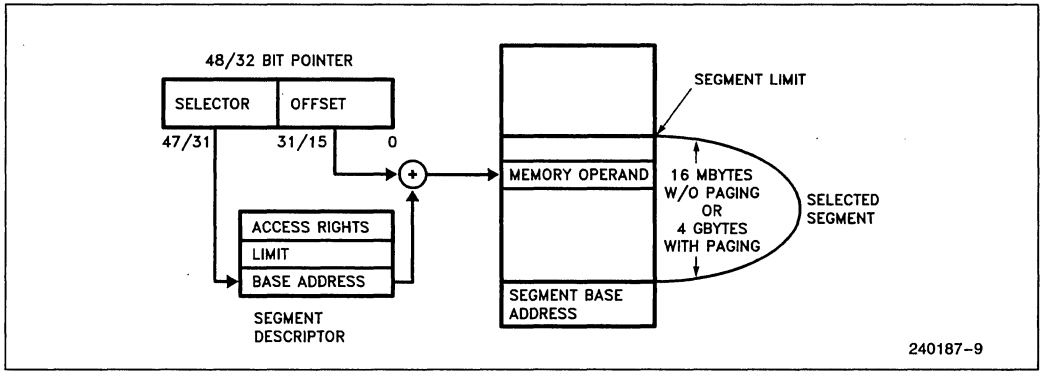


Figure 4.1. Protected Mode Addressing

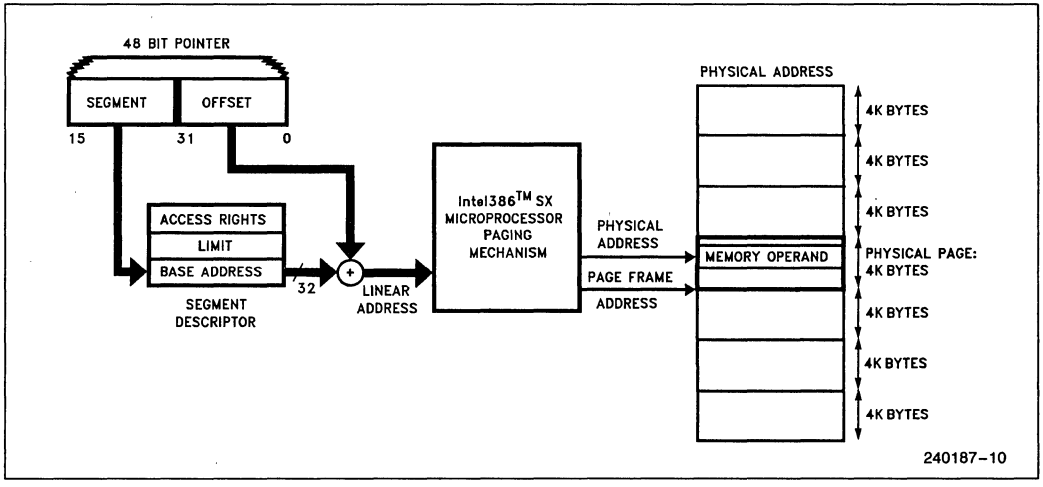


Figure 4.2. Paging and Segmentation

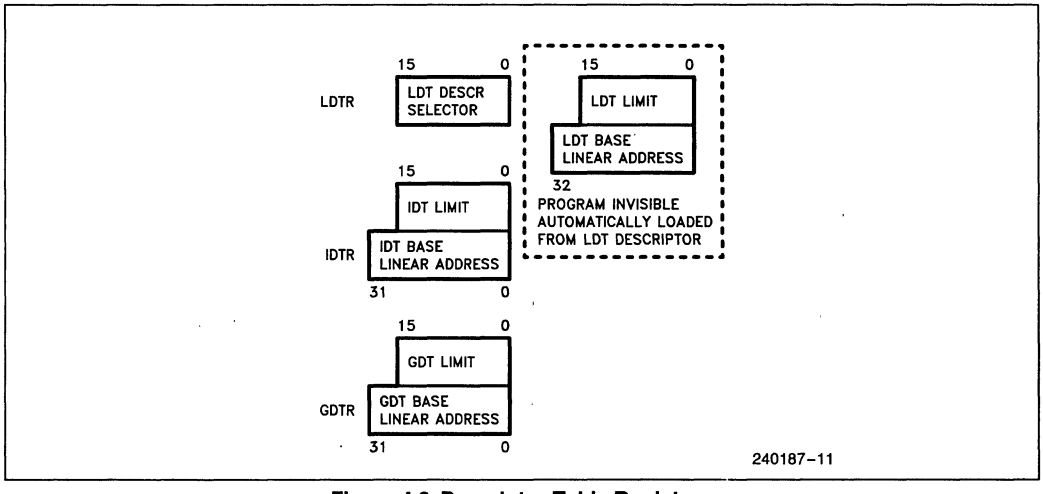


Figure 4.3. Descriptor Table Registers

Each of the tables has a register associated with it: GDTR, LDTR, and IDTR; see Figure 2.1. The LGDT, LLDT, and LIDT instructions load the base and limit of the Global, Local, and Interrupt Descriptor Tables into the appropriate register. The SGDT, SLDT, and SIDT store the base and limit values. These are privileged instructions.

Global Descriptor Table

The Global Descriptor Table (GDT) contains descriptors which are available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for interrupt and trap descriptors. Every Intel386 SX CPU system contains a GDT.

The first slot of the Global Descriptor Table corresponds to the null selector and is not used. The null selector defines a null pointer value.

Local Descriptor Table

LDTs contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This provides both isolation and protection for a task's segments while still allowing global data to be shared among tasks.

Unlike the 6-byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a Local Descriptor Table descriptor in the GDT (see figure 2.1).

Interrupt Descriptor Table

The third table needed for Intel386 SX Microprocessor systems is the Interrupt Descriptor Table. The IDT contains the descriptors which point to the location of the up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates, and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 Intel Reserved Interrupts. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced by INT instructions, external interrupt vectors, and exceptions.

DESCRIPTORS

The object to which the segment selector points to is called a descriptor. Descriptors are eight byte quantities which contain attributes about a given region of linear address space. These attributes include the 32-bit base linear address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write or execute privileges, the default size of the operands (16-bit or 32-bit), and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. Figure 4.4 shows the general format of a descriptor. All segments on the Intel386 SX Microprocessor have three attribute fields in common: the P bit, the DPL bit, and the S bit. The P

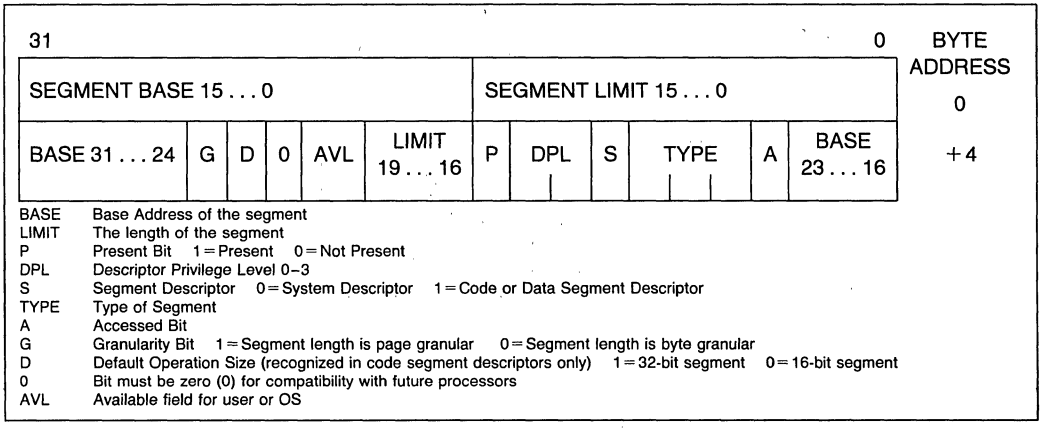


Figure 4.4. Segment Descriptors

(Present) Bit is 1 if the segment is loaded in physical memory. If P=0 then any attempt to access this segment causes a not present exception (exception 11). The Descriptor Privilege Level, DPL, is a two bit field which specifies the protection level, 0–3, associated with a segment.

The Intel386 SX Microprocessor has two main categories of segments: system segments and non-system segments (for code and data). The segment bit, S, determines if a given segment is a system segment or a code or data segment. If the S bit is 1 then the segment is either a code or data segment; if it is 0 then the segment is a system segment.

Code and Data Descriptors (S=1)

Figure 4.5 shows the general format of a code and data descriptor and Table 4.1 illustrates how the bits in the Access Right Byte are interpreted.

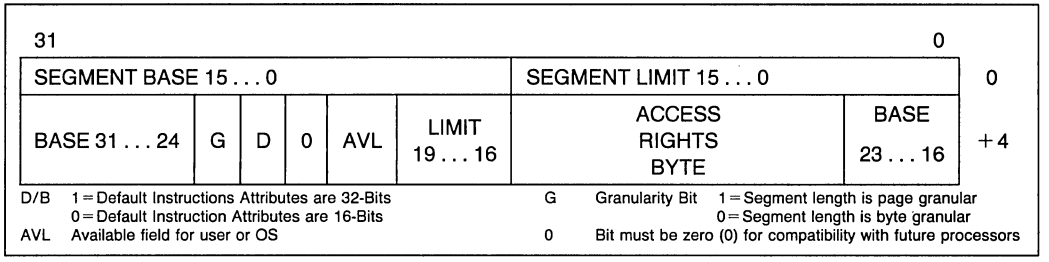
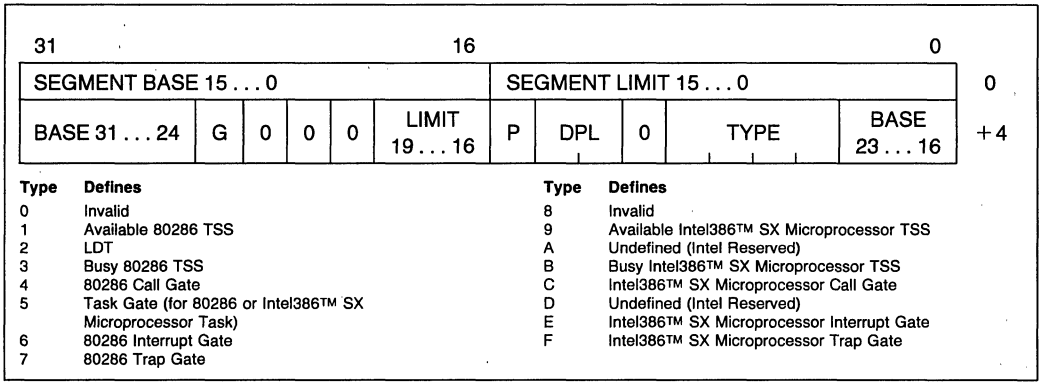


Figure 4.5. Code and Data Descriptors

Table 4.1. Access Rights Byte Definition for Code and Data Descriptors

Bit Position	Name	Function
7	Present (P)	P = 1 Segment is mapped into physical memory. P = 0 No mapping to physical memory exists, base and limt are not used.
6–5	Descriptor Privilege Level (DPL)	Segment privilege attribute used in privilege tests.
4	Segment Descriptor (S)	S = 1 Code or Data (includes stacks) segment descriptor S = 0 System Segment Descriptor or Gate Descriptor
3	Executable (E)	E = 0 Descriptor type is data segment: (If Data Segment (S = 1, E = 0))
2	Expansion Direction (ED)	ED = 0 Expand up segment, offsets must be ≤ limit. ED = 1 Expand down segment, offsets must be > limit.
1	Writeable (W)	W = 0 Data segment may not be written into. W = 1 Data segment may be written into.
3	Executable (E)	E = 1 Descriptor type is code segment: (If Code Segment (S = 1, E = 1))
2	Conforming (C)	C = 1 Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.
1	Readable (R)	R = 0 Code segment may not be read. R = 1 Code segment may be read.
0	Accessed (A)	A = 0 Segment has not been accessed. A = 1 Segment selector has been loaded into segment register or used by selector test instructions.



Type	Defines
0	Invalid
1	Available 80286 TSS
2	LDT
3	Busy 80286 TSS
4	80286 Call Gate
5	Task Gate (for 80286 or Intel386™ SX Microprocessor Task)
6	80286 Interrupt Gate
7	80286 Trap Gate

Type	Defines
8	Invalid
9	Available Intel386™ SX Microprocessor TSS
A	Undefined (Intel Reserved)
B	Busy Intel386™ SX Microprocessor TSS
C	Intel386™ SX Microprocessor Call Gate
D	Undefined (Intel Reserved)
E	Intel386™ SX Microprocessor Interrupt Gate
F	Intel386™ SX Microprocessor Trap Gate

Figure 4.6. System Descriptors

Code and data segments have several descriptor fields in common. The accessed bit, A, is set whenever the processor accesses a descriptor. The granularity bit, G, specifies if a segment length is byte-granular or page-granular.

System Descriptor Formats (S = 0)

System segments describe information about operating system tables, tasks, and gates. Figure 4.6 shows the general format of system segment descriptors, and the various types of system segments. Intel386 SX system descriptors (which are the same as Intel386 DX CPU system descriptors) contain a 32-bit base linear address and a 20-bit segment limit. 80286 system descriptors have a 24-bit base address and a 16-bit segment limit. 80286 system descriptors are identified by the upper 16 bits being all zero.

Differences Between Intel386™ SX Microprocessor and 80286 Descriptors

In order to provide operating system compatibility with the 80286 the Intel386 SX CPU supports all of the 80286 segment descriptors. The 80286 system segment descriptors contain a 24-bit base address and 16-bit limit, while the Intel386 SX CPU system segment descriptors have a 32-bit base address, a 20-bit limit field, and a granularity bit. The word count field specifies the number of 16-bit quantities to copy for 80286 call gates and 32-bit quantities for Intel386 SX CPU call gates.

Selector Fields

A selector in Protected Mode has three fields: Local or Global Descriptor Table indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL) as shown in Figure 4.7. The TI bit selects either the Global Descriptor Table or the Local Descriptor Table. The Index selects one of 8k descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

Segment Descriptor Cache

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs which modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.

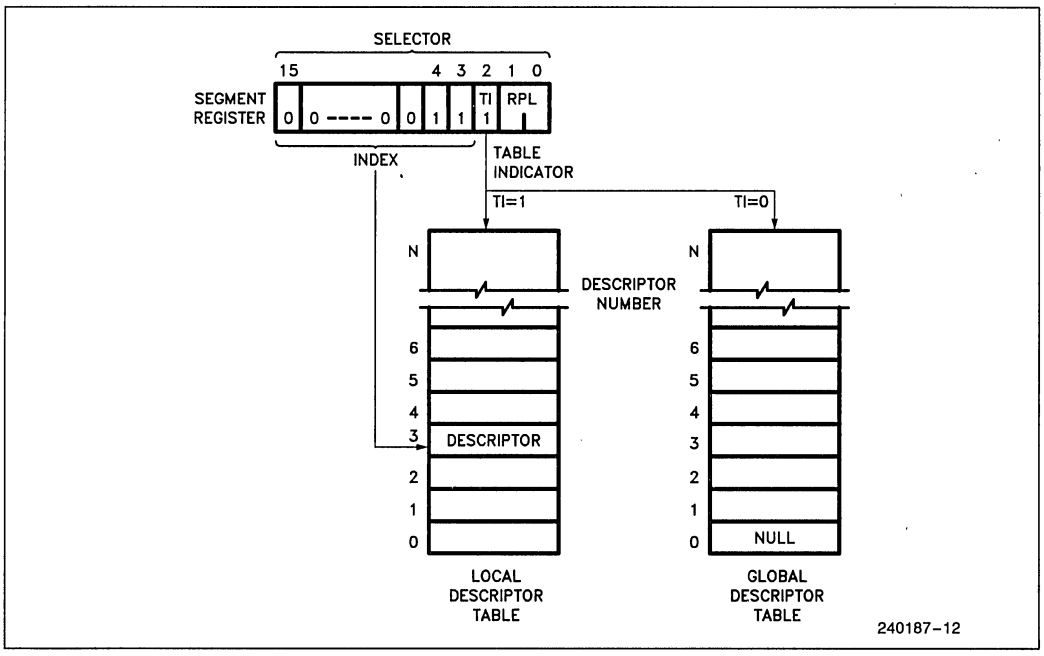


Figure 4.7. Example Descriptor Selection

4.3 Protection

The Intel386 SX Microprocessor has four levels of protection which are optimized to support a multitasking operating system and to isolate and protect user programs from each other and the operating system. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. The Intel386 SX Microprocessor also offers an additional type of protection on a page basis when paging is enabled.

The four-level hierarchical privilege system is an extension of the user/supervisor privilege mode commonly used by minicomputers. The user/supervisor mode is fully supported by the Intel386 SX Microprocessor paging mechanism. The privilege levels (PL) are numbered 0 through 3. Level 0 is the most privileged level.

RULES OF PRIVILEGE

The Intel386 SX Microprocessor controls access to both data and procedures between levels of a task, according to the following rules.

— Data stored in a segment with privilege level **p** can be accessed only by code executing at a privilege level at least as privileged as **p**.

— A code segment/procedure with privilege level **p** can only be called by a task executing at the same or a lesser privilege level than **p**.

PRIVILEGE LEVELS

At any point in time, a task on the Intel386 SX Microprocessor always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies what the task's privilege level is. A task's CPL may only be changed by control transfers through gate descriptors to a code segment with a different privilege level. Thus, an application program running at PL=3 may call an operating system routine at PL=1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

Selector Privilege (RPL)

The privilege level of a selector is specified by the RPL field. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level of the task for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (numerically larger) level of a task's CPL and a selector's RPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

Table 4.2. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL	CALL	Call Gate	GDT/LDT
	Interrupt instruction Exception External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0
**NT (Nested Task bit of flag register) = 1

I/O Privilege

The I/O privilege level (IOPL) lets the operating system code executing at CPL = 0 define the least privileged level at which I/O instructions can be used. An exception 13 (General Protection Violation) is generated if an I/O instruction is attempted when the CPL of the task is less privileged then the IOPL. The IOPL is stored in bits 13 and 14 of the EFLAGS register. The following instructions cause an exception 13 if the CPL is greater than IOPL: IN, INS, OUT, OUTS, STI, CLI, LOCK prefix.

Descriptor Access

There are basically two types of segment accesses: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL as described above.

Any time an instruction loads a data segment register (DS, ES, FS, GS) the Intel386 SX Microprocessor makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segment or readable code segments.

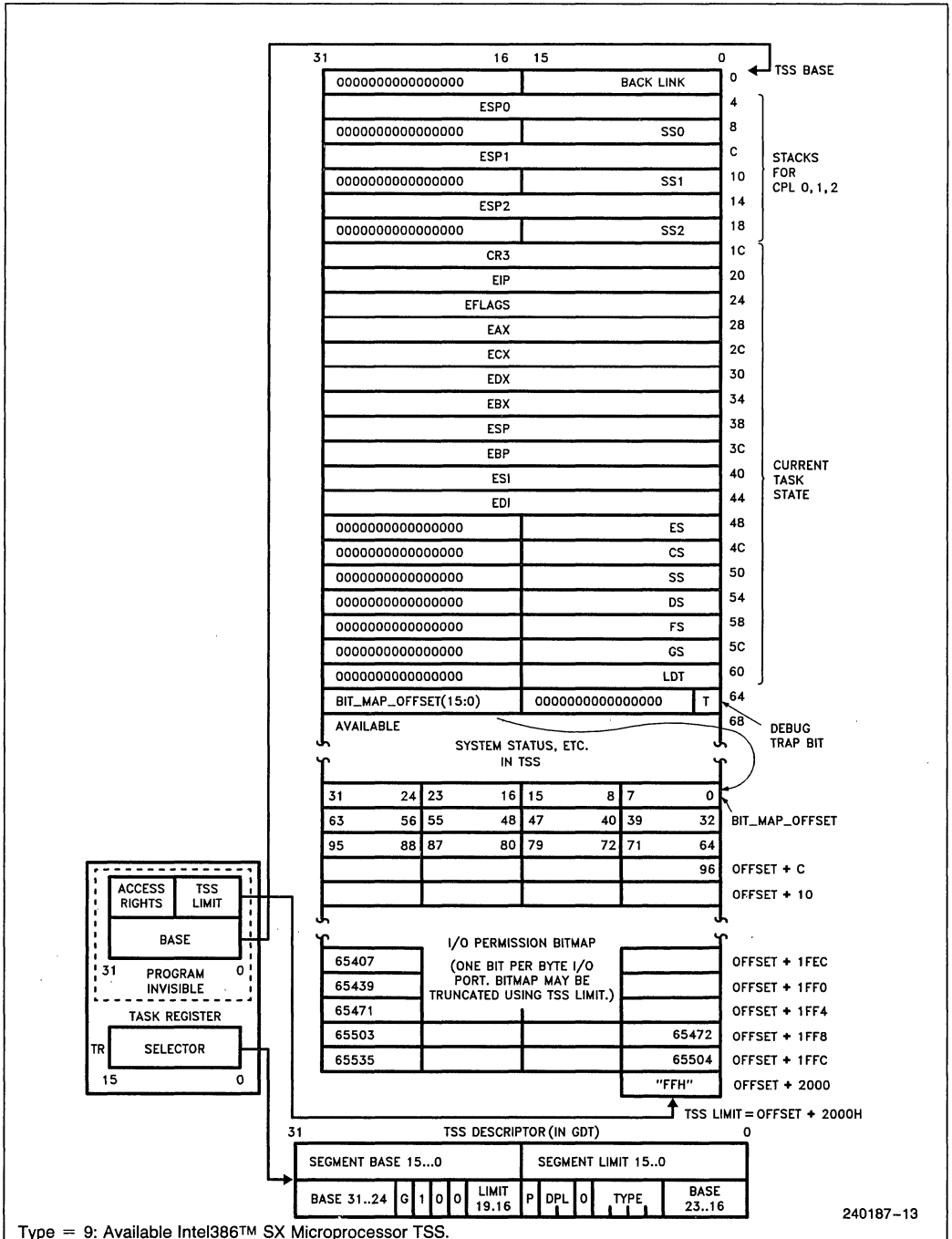
Finally the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL, an exception 13 (general protection fault) is generated.

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL of all other descriptor types or a privilege level violation will cause an exception 13. A stack not present fault causes an exception 12.

PRIVILEGE LEVEL TRANSFERS

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call or a jump to another routine. There are five types of control transfers which are summarized in Table 4.2. Many of these transfers result in a privilege level transfer. Changing privilege levels is done only by control transfers, using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an exception 13.



Type = 9: Available Intel386™ SX Microprocessor TSS.
Type = B: Busy Intel386 SX Microprocessor TSS.

Figure 4.8. Intel386™ SX Microprocessor TSS and TSS Registers

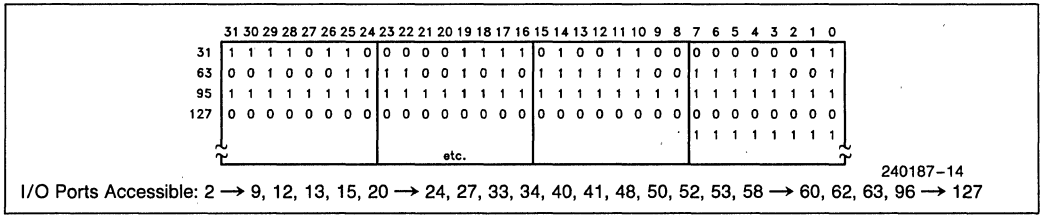


Figure 4.9. Sample I/O Permission Bit Map

CALL GATES

Gates provide protected indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures.

TASK SWITCHING

A very important attribute of any multi-tasking/multi-user operating system is its ability to rapidly switch between tasks or processes. The Intel386 SX Microprocessor directly supports this operation by providing a task switch instruction in hardware. The task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task. Like transfer of control by gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap, or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot.

The TSS descriptor points to a segment (see Figure 4.8) containing the entire execution state. A task gate descriptor contains a TSS selector. The Intel386 SX Microprocessor supports both the 80286 and Intel386 SX CPU TSSs. The limit of a Intel386 SX Microprocessor TSS must be greater than 64H (2BH for an 80286 TSS), and can be as large as 16 megabytes. In the additional TSS space, the operating system is free to store additional information such as the reason the task is inactive, time the task has spent running, or open files belonging to the task.

Each task must have a TSS associated with it. The current TSS is identified by a special register in the Intel386 SX Microprocessor called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TSS descriptor are loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The currently executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and machine status word (CR0) give information about the state of a task which is useful to the operating system. The Nested Task bit, NT, controls the function of the IRET instruction. If NT = 0 the IRET instruction performs the regular return. If NT = 1 IRET performs a task switch operation back to the previous task. The NT bit is set or reset in the following fashion:

> When a CALL or INT instruction initiates a task switch, the new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT (The NT bit will be restored after execution of the interrupt handler). NT may also be set or cleared by POPF or IRET instructions.

The Intel386 SX Microprocessor task state segment is marked busy by changing the descriptor type field from TYPE 9 to TYPE 0BH. An 80286 TSS is marked busy by changing the descriptor type field from TYPE 1 to TYPE 3. Use of a selector that references a busy task state segment causes an exception 13.

The VM (Virtual Mode) bit is used to indicate if a task is a Virtual 8086 task. If VM = 1 then the tasks will use the Real Mode addressing mechanism. The virtual 8086 environment is only entered and exited by a task switch.

The coprocessor's state is not automatically saved when a task switch occurs. The Task Switched Bit, TS, in the CR0 register helps deal with the coprocessor's state in a multi-tasking environment. Whenever the Intel386 SX Microprocessor switches task, it sets the TS bit. The Intel386 SX Microprocessor detects the first use of a processor extension instruction after a task switch and causes the processor extension not available exception 7. The exception handler for exception 7 may then decide whether to save the state of the coprocessor.

The T bit in the Intel386 SX Microprocessor TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1 then upon entry to a new task a debug exception 1 will be generated.

INITIALIZATION AND TRANSITION TO PROTECTED MODE

Since the Intel386 SX Microprocessor begins executing in Real Mode immediately after RESET it is necessary to initialize the system tables and registers with the appropriate values. The GDT and IDT registers must refer to a valid GDT and IDT. The IDT should be at least 256 bytes long, and the GDT must contain descriptors for the initial code and data segments.

Protected Mode is enabled by loading CR0 with PE bit set. This can be accomplished by using the **MOV CR0, R/M** instruction. After enabling Protected Mode, the next instruction should execute an intersegment JMP to load the CS register and flush the instruction decode queue. The final step is to load all of the data segment registers with the initial selector values.

An alternate approach to entering Protected Mode is to use the built in task-switch to load all of the registers. In this case the GDT would contain two TSS descriptors in addition to the code and data descriptors needed for the first task. The first JMP instruction in Protected Mode would jump to the TSS causing a task switch and loading all of the registers with the values stored in the TSS. The Task State Segment Register should be initialized to point to a valid TSS descriptor.

4.4 Paging

Paging is another type of memory management useful for virtual memory multi-tasking operating systems. Unlike segmentation, which modularizes programs and data into variable length segments, paging divides programs into multiple uniform size pages. Pages bear no direct relation to the logical structure of a program. While segment selectors can be considered the logical 'name' of a program module or data structure, a page most likely corresponds to only a portion of a module or data structure.

PAGE ORGANIZATION

The Intel386 SX Microprocessor uses two levels of tables to translate the linear address (from the segmentation unit) into a physical address. There are three components to the paging mechanism of the Intel386 SX Microprocessor: the page directory, the page tables, and the page itself (page frame). All memory-resident elements of the Intel386 SX Microprocessor paging mechanism are the same size, namely 4K bytes. A uniform size for all of the elements simplifies memory allocation and reallocation schemes, since there is no problem with memory fragmentation. Figure 4.10 shows how the paging mechanism works.

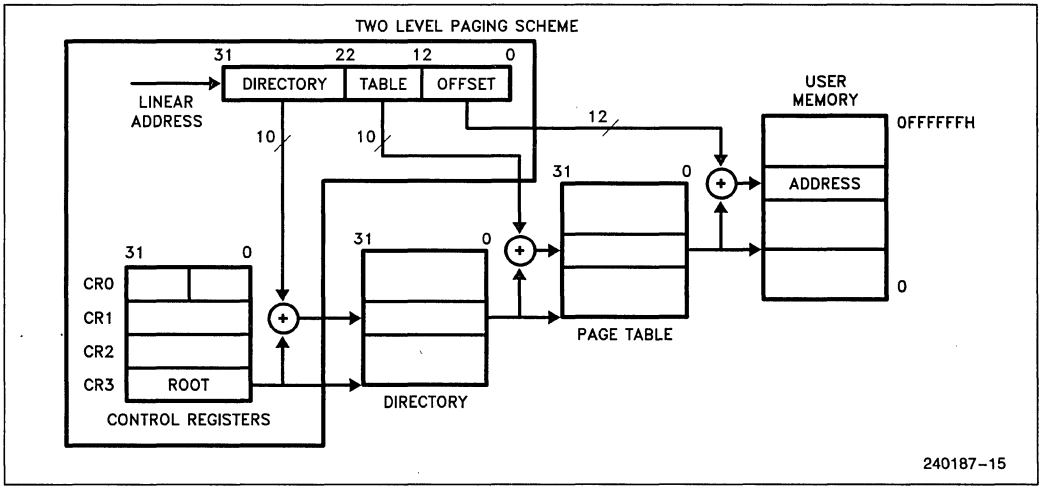


Figure 4.10. Paging Mechanism

31 12	11 10	9	8	7	6	5	4	3	2	1	0
PAGE TABLE ADDRESS 31..12	System Software Defineable		0	0	D	A	0	0	U / — / S	R / — / W	P

Figure 4.11. Page Directory Entry (Points to Page Table)

31 12	11 10	9	8	7	6	5	4	3	2	1	0
PAGE FRAME ADDRESS 31..12	System Software Defineable		0	0	D	A	0	0	U — S	R — W	P

Figure 4.12. Page Table Entry (Points to Page)

Page Fault Register

CR2 is the Page Fault Linear Address register. It holds the 32-bit linear address which caused the last Page Fault detected.

Page Descriptor Base Register

CR3 is the Page Directory Physical Base Address Register. It contains the physical starting address of the Page Directory (this value is truncated to a 24-bit value associated with the Intel386 SX CPU's 16 megabyte physical memory limitation). The lower 12 bits of CR3 are always zero to ensure that the Page Directory is always page aligned. Loading it with a **MOV CR3, reg** instruction causes the page table entry cache to be flushed, as will a task switch through a TSS which changes the value of CR0.

Page Directory

The Page Directory is 4k bytes long and allows up to 1024 page directory entries. Each page directory entry contains information about the page table and the address of the next level of tables, the Page Tables. The contents of a Page Directory Entry are shown in figure 4.11. The upper 10 bits of the linear address (A_{31}–A_{22}) are used as an index to select the correct Page Directory Entry.

The page table address contains the upper 20 bits of a 32-bit physical address that is used as the base address for the next set of tables, the page tables. The lower 12 bits of the page table address are zero so that the page table addresses appear on 4 kbyte boundaries. For a Intel386 DX CPU system the upper 20 bits will select one of 2^{20} page tables, but for a Intel386 SX Microprocessor system the upper 20 bits only select one of 2^{12} page tables. Again, this is because the Intel386 SX Microprocessor is limited to a 24-bit physical address and the upper 8 bits (A_{24}–A_{31}) are truncated when the address is output on its 24 address pins.

Page Tables

Each Page Table is 4K bytes long and allows up to 1024 Page table Entries. Each page table entry contains information about the Page Frame and its address. The contents of a Page Table Entry are shown in figure 4.12. The middle 10 bits of the linear address (A_{21}–A_{12}) are used as an index to select the correct Page Table Entry.

The Page Frame Address contains the upper 20 bits of a 32-bit physical address that is used as the base address for the Page Frame. The lower 12 bits of the Page Frame Address are zero so that the Page Frame addresses appear on 4 kbyte boundaries. For an Intel386 DX CPU system the upper 20 bits will select one of 2^{20} Page Frames, but for an Intel386 SX Microprocessor system the upper 20 bits only select one of 2^{12} Page Frames. Again, this is because the Intel386 SX Microprocessor is limited to a 24-bit physical address space and the upper 8 bits (A_{24}–A_{31}) are truncated when the address is output on its 24 address pins.

Page Directory/Table Entries

The lower 12 bits of the Page Table Entries and Page Directory Entries contain statistical information about pages and page tables respectively. The P (Present) bit indicates if a Page Directory or Page Table entry can be used in address translation. If P = 1, the entry can be used for address translation. If P = 0, the entry cannot be used for translation. All of the other bits are available for use by the software. For example, the remaining 31 bits could be used to indicate where on disk the page is stored.

The A (Accessed) bit is set by the Intel386 SX CPU for both types of entries before a read or write access occurs to an address covered by the entry. The D (Dirty) bit is set to 1 before a write to an address covered by that page table entry occurs. The D bit is undefined for Page Directory Entries. When the P, A and D bits are updated by the Intel386 SX CPU, the processor generates a Read- Modify-Write cycle which locks the bus and prevents conflicts with other processors or peripherals. Software which modifies these bits should use the LOCK prefix to ensure the integrity of the page tables in multi-master systems.

The 3 bits marked system software definable in Figures 4.11 and Figure 4.12 are software definable. System software writers are free to use these bits for whatever purpose they wish.

PAGE LEVEL PROTECTION (R/W, U/S BITS)

The Intel386 SX Microprocessor provides a set of protection attributes for paging systems. The paging mechanism distinguishes between two levels of protection: User, which corresponds to level 3 of the segmentation based protection, and supervisor which encompasses all of the other protection levels (0, 1, 2). Programs executing at Level 0, 1 or 2 bypass the page protection, although segmentation-based protection is still enforced by the hardware.

The U/S and R/W bits are used to provide User/Supervisor and Read/Write protection for individual pages or for all pages covered by a Page Table Directory Entry. The U/S and R/W bits in the second level Page Table Entry apply only to the page described by that entry. While the U/S and R/W bits in the first level Page Directory Table apply to all pages described by the page table pointed to by that directory entry. The U/S and R/W bits for a given page are obtained by taking the most restrictive of the U/S and R/W from the Page Directory Table Entries and using these bits to address the page.

TRANSLATION LOOKASIDE BUFFER

The Intel386 SX Microprocessor paging hardware is designed to support demand paged virtual memory systems. However, performance would degrade substantially if the processor was required to access two levels of tables for every memory reference. To solve this problem, the Intel386 SX Microprocessor keeps a cache of the most recently accessed pages, this cache is called the Translation Lookaside Buffer (TLB). The TLB is a four-way set associative 32-entry page table cache. It automatically keeps the most commonly used page table entries in the processor. The 32-entry TLB coupled with a 4K page size results in coverage of 128K bytes of memory addresses. For many common multi-tasking systems, the TLB will have a hit rate of greater than 98%. This means that the processor will only have to access the two-level page structure for less than 2% of all memory references.

PAGING OPERATION

The paging hardware operates in the following fashion. The paging unit hardware receives a 32-bit linear address from the segmentation unit. The upper 20 linear address bits are compared with all 32 entries in the TLB to determine if there is a match. If there is a match (i.e. a TLB hit), then the 24-bit physical address is calculated and is placed on the address bus.

If the page table entry is not in the TLB, the Intel386 SX Microprocessor will read the appropriate Page Directory Entry. If P=1 on the Page Directory Entry, indicating that the page table is in memory, then the Intel386 SX Microprocessor will read the appropriate Page Table Entry and set the Access bit. If P=1 on the Page Table Entry, indicating that the page is in memory, the Intel386 SX Microprocessor will update the Access and Dirty bits as needed and fetch the operand. The upper 20 bits of the linear address, read from the page table, will be stored in the TLB for future accesses. If P=0 for either the Page Directory Entry or the Page Table Entry, then the processor will generate a page fault Exception 14.

The processor will also generate a Page Fault (Exception 14) if the memory reference violated the page protection attributes. CR2 will hold the linear address which caused the page fault. Since Exception 14 is classified as a fault, CS:EIP will point to the instruction causing the page-fault. The 16-bit error code pushed as part of the page fault handler will contain status bits which indicate the cause of the page fault.

The 16-bit error code is used by the operating system to determine how to handle the Page Fault. Figure 4.13 shows the format of the Page Fault error code and the interpretation of the bits. Even though the bits in the error code (U/S, W/R, and P) have similar names as the bits in the Page Directory/Table Entries, the interpretation of the error code bits is different. Figure 4.14 indicates what type of access caused the page fault.

15												3	2	1	0
U	U	U	U	U	U	U	U	U	U	U	U	U	U/S	W/R	P

Figure 4.13. Page Fault Error Code Format

U/S: The U/S bit indicates whether the access causing the fault occurred when the processor was executing in User Mode (U/S = 1) or in Supervisor mode (U/S = 0)

W/R: The W/R bit indicates whether the access causing the fault was a Read (W/R = 0) or a Write (W/R = 1)

P: The P bit indicates whether a page fault was caused by a not-present page (P = 0), or by a page level protection violation (P = 1)

U = Undefined

U/S	W/R	Access Type
0	0	Supervisor* Read
0	1	Supervisor Write
1	0	User Read
1	1	User Write

*Descriptor table access will fault with U/S = 0, even if the program is executing at level 3.

Figure 4.14. Type of Access Causing Page Fault

3

OPERATING SYSTEM RESPONSIBILITIES

When the operating system enters or exits paging mode (by setting or resetting bit 31 in the CR0 register) a short JMP must be executed to flush the Intel386 SX Microprocessor's prefetch queue. This ensures that all instructions executed after the address mode change will generate correct addresses.

The Intel386 SX Microprocessor takes care of the page address translation process, relieving the burden from an operating system in a demand-paged system. The operating system is responsible for setting up the initial page tables and handling any page faults. The operating system also is required to invalidate (i.e. flush) the TLB when any changes are made to any of the page table entries. The operating system must reload CR3 to cause the TLB to be flushed.

Setting up the tables is simply a matter of loading CR3 with the address of the Page Directory, and allocating space for the Page Directory and the Page Tables. The primary responsibility of the operating system is to implement a swapping policy and handle all of the page faults.

A final concern of the operating system is to ensure that the TLB cache matches the information in the paging tables. In particular, any time the operating systems sets the P (Present) bit of page table entry to zero. The TLB must be flushed by reloading CR3. Operating systems may want to take advantage of the fact that CR3 is stored as part of a TSS, to give every task or group of tasks its own set of page tables.

4.5 Virtual 8086 Environment

The Intel386 SX Microprocessor allows the execution of 8086 application programs in both Real Mode and in the Virtual 8086 Mode. The Virtual 8086 Mode allows the execution of 8086 applications, while still allowing the system designer to take full advantage of the Intel386 SX CPU's protection mechanism.

VIRTUAL 8086 ADDRESSING MECHANISM

One of the major differences between Intel386 SX CPU Real and Protected modes is how the segment selectors are interpreted. When the processor is executing in Virtual 8086 Mode, the segment registers are used in a fashion identical to Real Mode. The contents of the segment register are shifted left 4 bits and added to the offset to form the segment base linear address.

The Intel386 SX Microprocessor allows the operating system to specify which programs use the 8086 address mechanism and which programs use Protected Mode addressing on a per task basis. Through the use of paging, the one megabyte address space of the Virtual Mode task can be mapped to anywhere in the 4 gigabyte linear address space of the Intel386 SX Microprocessor. Like Real Mode, Virtual Mode addresses that exceed one megabyte will cause an exception 13. However, these restrictions should not prove to be important, because most tasks running in Virtual 8086 Mode will simply be existing 8086 application programs.

PAGING IN VIRTUAL MODE

The paging hardware allows the concurrent running of multiple Virtual Mode tasks, and provides protection and operating system isolation. Although it is not strictly necessary to have the paging hardware enabled to run Virtual Mode tasks, it is needed in order to run multiple Virtual Mode tasks or to relocate the address space of a Virtual Mode task to physical address space greater than one megabyte.

The paging hardware allows the 20-bit linear address produced by a Virtual Mode program to be divided into as many as 256 pages. Each one of the pages can be located anywhere within the maximum 16 megabyte physical address space of the Intel386 SX Microprocessor. In addition, since CR3 (the Page Directory Base Register) is loaded by a task switch, each Virtual Mode task can use a different mapping scheme to map pages to different physical locations. Finally, the paging hardware allows the sharing of the 8086 operating system code between multiple 8086 applications.

PROTECTION AND I/O PERMISSION BIT MAP

All Virtual Mode programs execute at privilege level 3. As such, Virtual Mode programs are subject to all of the protection checks defined in Protected Mode. This is different than Real Mode, which implicitly is executing at privilege level 0. Thus, an attempt to execute a privileged instruction in Virtual Mode will cause an exception 13 fault.

The following are privileged instructions, which may be executed only at Privilege Level 0. Attempting to execute these instructions in Virtual 8086 Mode (or anytime CPL≥0) causes an exception 13 fault:

```
LIDT;   MOV DRn,REG;   MOV reg,DRn;
LGDT;   MOV TRn,reg;   MOV reg,TRn;
LMSW;   MOV CRn,reg;   MOV reg,CRn;

CLTS;
HLT;
```

Several instructions, particularly those applying to the multitasking and the protection model, are available only in Protected Mode. Therefore, attempting to execute the following instructions in Real Mode or in Virtual 8086 Mode generates an exception 6 fault:

```
LTR;    STR;
LLDT;   SLDT;
LAR;    VERR;
LSL;    VERW;
ARPL;
```

The instructions which are IOPL sensitive in Protected Mode are:

```
IN;     STI;
OUT;    CLI
INS;
OUTS;
REP INS;
REP OUTS;
```

In Virtual 8086 Mode the following instructions are IOPL-sensitive:

```
INT n;  STI;
PUSHF;  CLI;
POPF;   IRET;
```

The PUSHF, POPF, and IRET instructions are IOPL-sensitive in Virtual 8086 Mode only. This provision allows the IF flag to be virtualized to the virtual 8086 Mode program. The INT n software interrupt instruction is also IOPL-sensitive in Virtual 8086 mode. Note that the INT 3, INTO, and BOUND instructions are not IOPL-sensitive in Virtual 8086 Mode.

The I/O instructions that directly refer to addresses in the processor's I/O space are IN, INS, OUT, and OUTS. The Intel386 SX Microprocessor has the ability to selectively trap references to specific I/O addresses. The structure that enables selective trapping is the *I/O Permission Bit Map* in the TSS segment (see Figures 4.8 and 4.9). The I/O permission map is a bit vector. The size of the map and its location in the TSS segment are variable. The processor locates the I/O permission map by means of the **I/O map base** field in the fixed portion of the TSS. The **I/O map base** field is 16 bits wide and contains the offset of the beginning of the I/O permission map.

In protected mode when an I/O instruction (IN, INS, OUT or OUTS) is encountered, the processor first checks whether CPL ≤ IOPL. If this condition is true, the I/O operation may proceed. If not true, the processor checks the I/O permission map (in Virtual 8086 Mode, the processor consults the map without regard for the IOPL).

Each bit in the map corresponds to an I/O port byte address; for example, the bit for port 41 is found at **I/O map base** + 5, bit offset 1. The processor tests all the bits that correspond to the I/O addresses spanned by an I/O operation; for example, a double word operation tests four bits corresponding to four adjacent byte addresses. If any tested bit is set, the processor signals a general protection exception. If all the tested bits are zero, the I/O operations may proceed.

It is not necessary for the I/O permission map to represent all the I/O addresses. I/O addresses not spanned by the map are treated as if they had one-bits in the map. The **I/O map base** should be at least one byte less than the TSS limit, the last byte beyond the I/O mapping information must contain all 1's.

Because the I/O permission map is in the TSS segment, different tasks can have different maps. Thus, the operating system can allocate ports to a task by changing the I/O permission map in the task's TSS.

IMPORTANT IMPLEMENTATION NOTE: Beyond the last byte of I/O mapping information in the I/O permission bit map **must** be a byte containing all 1's. The byte of all 1's must be within the limit of the Intel386 SX CPU TSS segment (see Figure 4.8).

Interrupt Handling

In order to fully support the emulation of an 8086 machine, interrupts in Virtual 8086 Mode are handled in a unique fashion. When running in Virtual Mode all interrupts and exceptions involve a privilege change back to the host Intel386 SX Microprocessor operating system. The Intel386 SX Microprocessor operating system determines if the interrupt comes from a Protected Mode application or from a Virtual Mode program by examining the VM bit in the EFLAGS image stored on the stack.

When a Virtual Mode program is interrupted and execution passes to the interrupt routine at level 0, the VM bit is cleared. However, the VM bit is still set in the EFLAG image on the stack.

The Intel386 SX Microprocessor operating system in turn handles the exception or interrupt and then returns control to the 8086 program. The Intel386 SX Microprocessor operating system may choose to let the 8086 operating system handle the interrupt or it may emulate the function of the interrupt handler. For example, many 8086 operating system calls are accessed by PUSHing parameters on the stack, and then executing an INT n instruction. If the IOPL is set to 0 then all INT n instructions will be intercepted by the Intel386 SX Microprocessor operating system.

An Intel386 SX Microprocessor operating system can provide a Virtual 8086 Environment which is totally transparent to the application software by intercepting and then emulating 8086 operating system's calls, and intercepting IN and OUT instructions.

Entering and Leaving Virtual 8086 Mode

Virtual 8086 mode is entered by executing a 32-bit IRET instruction at CPL=0 where the stack has a 1 in the VM bit of its EFLAGS image, or a Task Switch (at any CPL) to a Intel386 SX Microprocessor task whose Intel386 SX CPU TSS has a EFLAGS image containing a 1 in the VM bit position while the processor is executing in the Protected Mode. POPF does not affect the VM bit but a PUSHF always pushes a 0 in the VM bit.

The transition out of Virtual 8086 mode to protected mode occurs only on receipt of an interrupt or exception. In Virtual 8086 mode, all interrupts and exceptions vector through the protected mode IDT, and enter an interrupt handler in protected mode. As part of the interrupt processing the VM bit is cleared.

Because the matching IRET must occur from level 0, Interrupt or Trap Gates used to field an interrupt or exception out of Virtual 8086 mode must perform an inter-level interrupt only to level 0. Interrupt or Trap Gates through conforming segments, or through segments with DPL>0, will raise a GP fault with the CS selector as the error code.

Task Switches To/From Virtual 8086 Mode

Tasks which can execute in Virtual 8086 mode must be described by a TSS with the Intel386 SX CPU format (type 9 or 11 descriptor). A task switch out of virtual 8086 mode will operate exactly the same as any other task switch out of a task with a Intel386 SX CPU TSS. All of the programmer visible state, including the EFLAGS register with the VM bit set to 1, is stored in the TSS. The segment registers in the TSS will contain 8086 segment base values rather than selectors.

A task switch into a task described by a Intel386 SX CPU TSS will have an additional check to determine if the incoming task should be resumed in Virtual 8086 mode. Tasks described by 286 format TSSs cannot be resumed in Virtual 8086 mode, so no check is required there (the FLAGS image in 286 format TSS has only the low order 16 FLAGS bits). Before loading the segment register images from a Intel386 SX CPU TSS, the FLAGS image is loaded, so that the segment registers are loaded from the TSS image as 8086 segment base values. The task is now ready to resume in Virtual 8086 mode.

Transitions Through Trap and Interrupt Gates, and IRET

A task switch is one way to enter or exit Virtual 8086 mode. The other method is to exit through a Trap or Interrupt gate, as part of handling an interrupt, and to enter as part of executing an IRET instruction. The transition out must use a Intel386 SX CPU Trap Gate (Type 14), or Intel386 SX CPU Interrupt Gate (Type 15), which must point to a non-conforming level 0 segment (DPL=0) in order to permit the trap handler to IRET back to the Virtual 8086 program. The Gate must point to a non-conforming level 0 segment to perform a level switch to level 0 so that the matching IRET can change the VM bit. Intel386 SX CPU gates must be used since 286 gates save only the low 16 bits of the EFLAGS register (the VM bit will not be saved). Also, the 16-bit IRET used to terminate the 286 interrupt handler will pop only the lower 16 bits from FLAGS, and will not affect the VM bit. The action taken for a Intel386 SX CPU Trap or Interrupt gate if an interrupt occurs while the task is executing in virtual 8086 mode is given by the following sequence:

1. Save the FLAGS register in a temp to push later. Turn off the VM, TF, and IF bits.
2. Interrupt and Trap gates must perform a level switch from 3 (where the Virtual 8086 Mode program executes) to level 0 (so IRET can return).
3. Push the 8086 segment register values onto the new stack, in this order: GS, FS, DS, ES. These are pushed as 32-bit quantities. Then load these 4 registers with null selectors (0).
4. Push the old 8086 stack pointer onto the new stack by pushing the SS register (as 32-bits), then pushing the 32-bit ESP register saved above.
5. Push the 32-bit EFLAGS register saved in step 1.
6. Push the old 8086 instruction onto the new stack by pushing the CS register (as 32-bits), then pushing the 32-bit EIP register.
7. Load up the new CS:EIP value from the interrupt gate, and begin execution of the interrupt routine in protected mode.

The transition out of V86 mode performs a level change and stack switch, in addition to changing back to protected mode. Also all of the 8086 segment register images are stored on the stack (behind the SS:ESP image), and then loaded with null (0) selectors before entering the interrupt handler. This will permit the handler to safely save and restore the DS, ES, FS, and GS registers as 286 selectors. This is needed so that interrupt handlers which don't care about the mode of the interrupted program can use the same prologue and epilogue code for state saving regardless of whether or not a 'native' mode or Virtual 8086 Mode program was inter-

rupted. Restoring null selectors to these registers before executing the IRET will cause a trap in the interrupt handler. Interrupt routines which expect or return values in the segment registers will have to obtain/return values from the 8086 register images pushed onto the new stack. They will need to know the mode of the interrupted program in order to know where to find/return segment registers, and also to know how to interpret segment register values.

The IRET instruction will perform the inverse of the above sequence. Only the extended IRET instruction (operand size=32) can be used and must be executed at level 0 to change the VM bit to 1.

1. If the NT bit in the FLAGS register is on, an inter-task return is performed. The current state is stored in the current TSS, and the link field in the current TSS is used to locate the TSS for the interrupted task which is to be resumed. Otherwise, continue with the following sequence:
2. Read the FLAGS image from SS:8[ESP] into the FLAGS register. This will set VM to the value active in the interrupted routine.
3. Pop off the instruction pointer CS:EIP. EIP is popped first, then a 32-bit word is popped which contains the CS value in the lower 16 bits. If VM=0, this CS load is done as a protected mode segment load. If VM=1, this will be done as an 8086 segment load.
4. Increment the ESP register by 4 to bypass the FLAGS image which was 'popped' in step 1.
5. If VM=1, load segment registers ES, DS, FS, and GS from memory locations SS:[ESP+8], SS:[ESP+12], SS:[ESP+16], and SS:[ESP=20], respectively, where the new value of ESP stored in step 4 is used. Since VM=1, these are done as 8086 segment register loads.

   Else if VM=0, check that the selectors in ES, DS, FS, and GS are valid in the interrupted routine. Null out invalid selectors to trap if an attempt is made to access through them.
6. If RPL(CS)>CPL, pop the stack pointer SS:ESP from the stack. The ESP register is popped first, followed by 32-bits containing SS in the lower 16 bits. If VM=0, SS is loaded as a protected mode segment register load. If VM=1, an 8086 segment register load is used.
7. Resume execution of the interrupted routine. The VM bit in the FLAGS register (restored from the interrupt routine's stack image in step 1) determines whether the processor resumes the interrupted routine in Protected mode or Virtual 8086 Mode.

5.0 FUNCTIONAL DATA

The Intel386 SX Microprocessor features a straightforward functional interface to the external hardware. The Intel386 SX Microprocessor has separate parallel buses for data and address. The data bus is 16-bits in width, and bi-directional. The address bus outputs 24-bit address values using 23 address lines and two byte enable signals.

The Intel386 SX Microprocessor has two selectable address bus cycles: address pipelined and non-address pipelined. The address pipelining option allows as much time as possible for data access by starting the pending bus cycle before the present bus cycle is finished. A non-pipelined bus cycle gives the highest bus performance by executing every bus cycle in two processor CLK cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor, or from processor to system. Intel386 SX Microprocessor bus cycles perform data transfer in a minimum of only two clock periods. The maximum transfer bandwidth at 16 MHz is therefore 16 Mbytes/sec. However, any bus cycle will be extended for more than two clock periods if external hardware withholds acknowledgement of the cycle.

The Intel386 SX Microprocessor can relinquish control of its local buses to allow mastership by other devices, such as direct memory access (DMA) channels. When relinquished, HLDA is the only output pin driven by the Intel386 SX Microprocessor, providing near-complete isolation of the processor from its system (all other output pins are in a float condition).

5.1 Signal Description Overview

Ahead is a brief description of the Intel386 SX Microprocessor input and output signals arranged by functional groups. Note the # symbol at the end of a signal name indicates the active, or asserted, state occurs when the signal is at a LOW voltage. When no # is present after the signal name, the signal is asserted when at the HIGH voltage level.

Example signal: M/IO# — HIGH voltage indicates Memory selected

— LOW voltage indicates I/O selected

The signal descriptions sometimes refer to AC timing parameters, such as 't_{25} Reset Setup Time' and 't_{26} Reset Hold Time.' The values of these parameters can be found in Table 7.4.

CLOCK (CLK2)

CLK2 provides the fundamental timing for the Intel386 SX Microprocessor. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two phases, 'phase one' and 'phase two'. Each CLK2 period is a phase of the internal clock. Figure 5.2 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the falling edge of the RESET signal meets the applicable setup and hold times t_{25} and t_{26}.

DATA BUS (D_{15}–D_0)

These three-state bidirectional signals provide the general purpose data path between the Intel386 SX Microprocessor and other devices. The data bus outputs are active HIGH and will float during bus hold acknowledge. Data bus reads require that read-data setup and hold times t_{21} and t_{22} be met relative to CLK2 for correct operation.

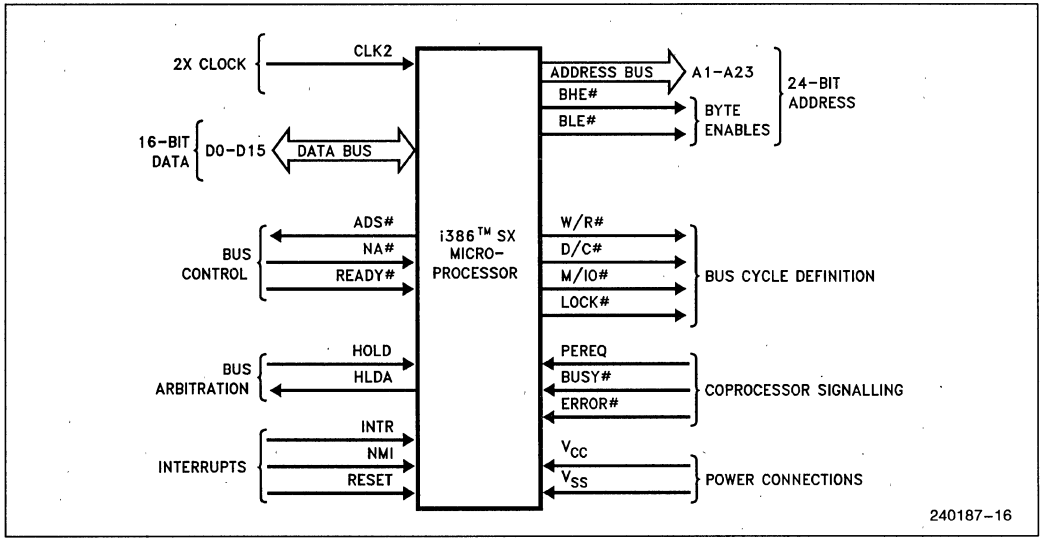


Figure 5.1. Functional Signal Groups

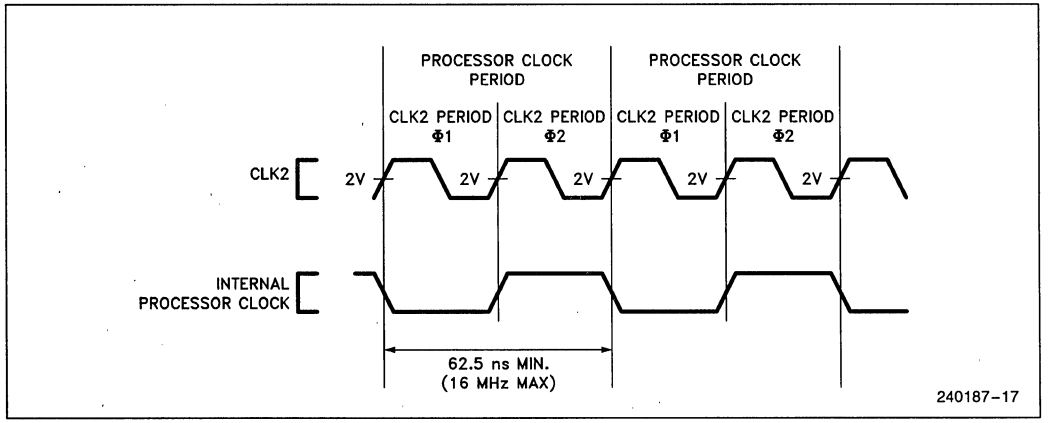


Figure 5.2. CLK2 Signal and Internal Processor Clock

ADDRESS BUS (A_{23}–A_1, BHE#, BLE#)

These three-state outputs provide physical memory addresses or I/O port addresses. A_{23}–A_{16} are LOW during I/O transfers except for I/O transfers automatically generated by coprocessor instructions. During coprocessor I/O transfers, A_{22}–A_{16} are driven LOW, and A_{23} is driven HIGH so that this address line can be used by external logic to generate the coprocessor select signal. Thus, the I/O address driven by the Intel386 SX Microprocessor for coprocessor commands is 8000F8H, the I/O addresses driven by the Intel386 SX Microprocessor for coprocessor data are 8000FCH or 8000FEH for cycles to the Intel387™ SX.

The address bus is capable of addressing 16 megabytes of physical memory space (000000H through FFFFFFH), and 64 kilobytes of I/O address space (000000H through 00FFFFH) for programmed I/O. The address bus is active HIGH and will float during bus hold acknowledge.

The Byte Enable outputs, BHE# and BLE#, directly indicate which bytes of the 16-bit data bus are involved with the current transfer. BHE# applies to D_{15}–D_8 and BLE# applies to D_7–D_0. If both BHE# and BLE# are asserted, then 16 bits of data are being transferred. See Table 5.1 for a complete decoding of these signals. The byte enables are active LOW and will float during bus hold acknowledge.

BUS CYCLE DEFINITION SIGNALS (W/R#, D/C#, M/IO#, LOCK#)

These three-state outputs define the type of bus cycle being performed: W/R# distinguishes between write and read cycles, D/C# distinguishes between data and control cycles, M/IO# distinguishes between memory and I/O cycles, and LOCK# distinguishes between locked and unlocked bus cycles. All of these signals are active LOW and will float during bus acknowledge.

The primary bus cycle definition signals are W/R#, D/C# and M/IO#, since these are the signals driven valid as ADS# (Address Status output) becomes active. The LOCK# is driven valid at the same time the bus cycle begins, which due to address pipelining, could be after ADS# becomes active. Exact bus cycle definitions, as a function of W/R#, D/C#, and M/IO# are given in Table 5.2.

LOCK# indicates that other system bus masters are not to gain control of the system bus while it is active. LOCK# is activated on the CLK2 edge that begins the first locked bus cycle (i.e., it is not active at the same time as the other bus cycle definition pins) and is deactivated when ready is returned at the end of the last bus cycle which is to be locked. The beginning of a bus cycle is determined when READY# is returned in a previous bus cycle and another is pending (ADS# is active) or by the clock edge in which ADS# is driven active if the bus was idle. This means that it follows more closely with the write data rules when it is valid, but may cause the bus to be locked longer than desired. The LOCK# signal may be explicitly activated by the LOCK prefix on certain instructions. LOCK# is always asserted when executing the XCHG instruction, during descriptor updates, and during the interrupt acknowledge sequence.

Table 5.1. Byte Enable Definitions

BHE#	BLE#	Function
0	0	Word Transfer
0	1	Byte transfer on upper byte of the data bus, D_{15}–D_8
1	0	Byte transfer on lower byte of the data bus, D_7–D_0
1	1	Never occurs

Table 5.2. Bus Cycle Definition

M/IO#	D/C#	W/R#	Bus Cycle Type	Locked?
0	0	0	Interrupt Acknowledge	Yes
0	0	1	does not occur	—
0	1	0	I/O Data Read	No
0	1	1	I/O Data Write	No
1	0	0	Memory Code Read	No
1	0	1	Halt: Shutdown: Address = 2 Address = 0 BHE# = 1 BHE# = 1 BLE# = 0 BLE# = 0	No
1	1	0	Memory Data Read	Some Cycles
1	1	1	Memory Data Write	Some Cycles

BUS CONTROL SIGNALS (ADS#, READY#, NA#)

The following signals allow the processor to indicate when a bus cycle has begun, and allow other system hardware to control address pipelining and bus cycle termination.

Address Status (ADS#)

This three-state output indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BHE#, BLE# and A_{23}–A_1) are being driven at the Intel386 SX Microprocessor pins. ADS# is an active LOW output. Once ADS# is driven active, valid address, byte enables, and definition signals will not change. In addition, ADS# will remain active until its associated bus cycle begins (when READY# is returned for the previous bus cycle when running pipelined bus cycles). When address pipelining is utilized, maximum throughput is achieved by initiating bus cycles when ADS# and READY# are active in the same clock cycle. ADS# will float during bus hold acknowledge. See sections **Non-Pipelined Address** and **Pipelined Address** for additional information on how ADS# is asserted for different bus states.

Transfer Acknowledge (READY#)

This input indicates the current bus cycle is complete, and the active bytes indicated by BHE# and BLE# are accepted or provided. When READY# is sampled active during a read cycle or interrupt acknowledge cycle, the Intel386 SX Microprocessor latches the input data and terminates the cycle. When READY# is sampled active during a write cycle, the processor terminates the bus cycle.

READY# is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. READY# must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, READY# must always meet setup and hold times t_{19} and t_{20} for correct operation.

Next Address Request (NA#)

This is used to request address pipelining. This input indicates the system is prepared to accept new values of BHE#, BLE#, A_{23}–A_1, W/R#, D/C# and M/IO# from the Intel386 SX Microprocessor even if the end of the current cycle is not being acknowledged on READY#. If this input is active when sampled, the next address is driven onto the bus, provided the next bus request is already pending internally. NA# is ignored in CLK cycles in which ADS# or READY# is activated. This signal is active LOW and must satisfy setup and hold times t_{15} and t_{16} for correct operation. See **Pipelined Address** and **Read and Write Cycles** for additional information.

BUS ARBITRATION SIGNALS (HOLD, HLDA)

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See **Entering and Exiting Hold Acknowledge** for additional information.

Bus Hold Request (HOLD)

This input indicates some device other than the Intel386 SX Microprocessor requires bus mastership. When control is granted, the Intel386 SX Microprocessor floats A_{23}–A_1, BHE#, BLE#, D_{15}–D_0, LOCK#, M/IO#, D/C#, W/R# and ADS#, and then activates HLDA, thus entering the bus hold acknowledge state. The local bus will remain granted to the requesting master until HOLD becomes inactive. When HOLD becomes inactive, the Intel386 SX Microprocessor will deactivate HLDA and drive the local bus (at the same time), thus terminating the hold acknowledge condition.

HOLD must remain asserted as long as any other device is a local bus master. External pull-up resistors may be required when in the hold acknowledge state since none of the Intel386 SX Microprocessor floated outputs have internal pull-up resistors. See **Resistor Recommendations** for additional information. HOLD is not recognized while RESET is active. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the hold acknowledge (high-impedance) state.

HOLD is a level-sensitive, active HIGH, synchronous input. HOLD signals must always meet setup and hold times t_{23} and t_{24} for correct operation.

Bus Hold Acknowledge (HLDA)

When active (HIGH), this output indicates the Intel386 SX Microprocessor has relinquished control of its local bus in response to an asserted HOLD signal, and is in the bus Hold Acknowledge state.

The Bus Hold Acknowledge state offers near-complete signal isolation. In the Hold Acknowledge state, HLDA is the only signal being driven by the Intel386 SX Microprocessor. The other output signals or bidirectional signals (D_{15}–D_0, BHE#, BLE#, A_{23}–A_1, W/R#, D/C#, M/IO#, LOCK# and ADS#) are in a high-impedance state so the re-

questing bus master may control them. These pins remain OFF throughout the time that HLDA remains active (see Table 5.3)). Pull-up resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See **Resistor Recommendations** for additional information.

When the HOLD signal is made inactive, the Intel386 SX Microprocessor will deactivate HLDA and drive the bus. One rising edge on the NMI input is remembered for processing after the HOLD input is negated.

Table 5.3. Output pin State During HOLD

Pin Value	Pin Names
1	HLDA
Float	LOCK#, M/IO#, D/C#, W/R#, ADS#, A_{23}–A_1, BHE#, BLE#, D_{15}–D_0

In addition to the normal usage of Hold Acknowledge with DMA controllers or master peripherals, the near-complete isolation has particular attractiveness during system test when test equipment drives the system, and in hardware fault-tolerant applications.

HOLD Latencies

The maximum possible HOLD latency depends on the software being executed. The actual HOLD latency at any time depends on the current bus activity, the state of the LOCK# signal (internal to the CPU) activated by the LOCK# prefix, and interrupts. The Intel386 SX Microprocessor will not honor a HOLD request until the current bus operation is complete.

The Intel386 SX Microprocessor breaks 32-bit data or I/O accesses into 2 internally locked 16-bit bus cycles; the LOCK# signal is not asserted. The Intel386 SX Microprocessor breaks unaligned 16-bit or 32-bit data or I/O accesses into 2 or 3 internally locked 16-bit bus cycles. Again, the LOCK# signal is not asserted but a HOLD request will not be recognized until the end of the entire transfer.

Wait states affect HOLD latency. The Intel386 SX Microprocessor will not honor a HOLD request until the end of the current bus operation, no matter how many wait states are required. Systems with DMA where data transfer is critical must insure that READY# returns sufficiently soon.

COPROCESSOR INTERFACE SIGNALS (PEREQ, BUSY#, ERROR#)

In the following sections are descriptions of signals dedicated to the numeric coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, these following signals control communication between the Intel386 SX Microprocessor and its Intel387™ SX processor extension.

Coprocessor Request (PEREQ)

When asserted (HIGH), this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the Intel386 SX Microprocessor. In response, the Intel386 SX Microprocessor transfers information between the coprocessor and memory. Because the Intel386 SX Microprocessor has internally stored the coprocessor opcode being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is a level-sensitive active HIGH asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This signal is provided with a weak internal pull-down resistor of around 20 K-ohms to ground so that it will not float active when left unconnected.

Coprocessor Busy (BUSY#)

When asserted (LOW), this input indicates the coprocessor is still executing an instruction, and is not yet able to accept another. When the Intel386 SX Microprocessor encounters any coprocessor instruction which operates on the numerics stack (e.g. load, pop, or arithmetic operation), or the WAIT instruction, this input is first automatically sampled until it is seen to be inactive. This sampling of the BUSY# input prevents overrunning the execution of a previous coprocessor instruction.

The FNINIT, FNSTENV, FNSAVE, FNSTSW, FNSTCW and FNCLEX coprocessor instructions are allowed to execute even if BUSY# is active, since these instructions are used for coprocessor initialization and exception-clearing.

BUSY# is an active LOW, level-sensitive asynchronous signal. Setup and hold times, t_{29} and t_{30}, rela-

tive to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 K-ohms to Vcc so that it will not float active when left unconnected.

BUSY# serves an additional function. If BUSY# is sampled LOW at the falling edge of RESET, the Intel386 SX Microprocessor performs an internal self-test (see **Bus Activity During and Following Reset**. If BUSY# is sampled HIGH, no self-test is performed.

Coprocessor Error (ERROR#)

When asserted (LOW), this input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the Intel386 SX Microprocessor when a coprocessor instruction is encountered, and if active, the Intel386 SX Microprocessor generates exception 16 to access the error-handling software.

Several coprocessor instructions, generally those which clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the Intel386 SX Microprocessor generating exception 16 even if ERROR# is active. These instructions are FNINIT, FNCLEX, FNSTSW, FNSTSWAX, FNSTCW, FNSTENV and FNSAVE.

ERROR# is an active LOW, level-sensitive asynchronous signal. Setup and hold times, t_{29} and t_{30}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 K-ohms to Vcc so that it will not float active when left unconnected.

INTERRUPT SIGNALS (INTR, NMI, RESET)

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

Maskable Interrupt Request (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the Intel386 SX CPU Flag Register IF bit. When the Intel386 SX Microprocessor responds to the INTR input, it performs two interrupt acknowledge bus cycles and, at the end of the second, latches an 8-bit interrupt vector on D_7–D_0 to identify the source of the interrupt.

INTR is an active HIGH, level-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of an INTR request, INTR should remain active until the first interrupt acknowledge bus cycle begins. INTR is sampled at the beginning of every instruction in the Intel386 SX Microprocessor's Execution Unit. In order to be recognized at a particular instruction boundary, INTR must be active at least eight CLK2 clock periods before the beginning of the instruction. If recognized, the Intel386 SX Microprocessor will begin execution of the interrupt.

Non-Maskable Interrupt Request (NMI))

This input indicates a request for interrupt service which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are performed when processing NMI.

NMI is an active HIGH, rising edge-sensitive asynchronous signal. Setup and hold times, t_{27} and t_{28}, relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. To assure recognition of NMI, it must be inactive for at least eight CLK2 periods, and then be active for at least eight CLK2 periods before the beginning of the instruction boundary in the Intel386 SX Microprocessor's Execution Unit.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

Interrupt Latency

The time that elapses before an interrupt request is serviced (interrupt latency) varies according to several factors. This delay must be taken into account by the interrupt source. Any of the following factors can affect interrupt latency:

1. If interrupts are masked, an INTR request will not be recognized until interrupts are reenabled.
2. If an NMI is currently being serviced, an incoming NMI request will not be recognized until the Intel386 SX Microprocessor encounters the IRET instruction.
3. An interrupt request is recognized only on an instruction boundary of the Intel386 SX Microprocessor's Execution Unit except for the following cases:
   — Repeat string instructions can be interrupted after each iteration.

— If the instruction loads the Stack Segment register, an interrupt is not processed until after the following instruction, which should be an ESP. This allows the entire stack pointer to be loaded without interruption.

— If an instruction sets the interrupt flag (enabling interrupts), an interrupt is not processed until after the next instruction.

The longest latency occurs when the interrupt request arrives while the Intel386 SX Microprocessor is executing a long instruction such as multiplication, division, or a task-switch in the protected mode.

4. Saving the Flags register and CS:EIP registers.
5. If interrupt service routine requires a task switch, time must be allowed for the task switch.
6. If the interrupt service routine saves registers that are not automatically saved by the Intel386 SX Microprocessor.

RESET

This input signal suspends any operation in progress and places the Intel386 SX Microprocessor in a known reset state. The Intel386 SX Microprocessor is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self-test). When RESET is active, all other input pins, except FLT#, are ignored, and all other bus pins are driven to an idle bus state as shown in Table 5.5. If RESET and HOLD are both active at a point in time, RESET takes priority even if the Intel386 SX Microprocessor was in a Hold Acknowledge state prior to RESET active.

RESET is an active HIGH, level-sensitive synchronous signal. Setup and hold times, t_{25} and t_{26}, must be met in order to assure proper operation of the Intel386 SX Microprocessor.

Table 5.5. Pin State (Bus Idle) During Reset

Pin Name	Signal Level During Reset
ADS#	1
D_{15}–D_0	Float
BHE#, BLE#	0
A_{23}–A_1	1
W/R#	0
D/C#	1
M/IO#	0
LOCK#	1
HLDA	0

5.2 Bus Transfer Mechanism

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte and word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two physical bus cycles are performed as required for unaligned operand transfers.

The Intel386 SX Microprocessor address signals are designed to simplify external system hardware. Higher-order address bits are provided by A_{23}–A_1. BHE# and BLE# provide linear selects for the two bytes of the 16-bit data bus.

Byte Enable outputs BHE# and BLE# are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 5.6.

Table 5.6. Byte Enables and Associated Data and Operand Bytes

Byte Enable Signal	Associated Data Bus Signals	
BLE#	D_7–D_0	(byte 0 — least significant)
BHE#	D_{15}–D_8	(byte 1 — most significant)

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See section **5.4 Bus Functional Description**.

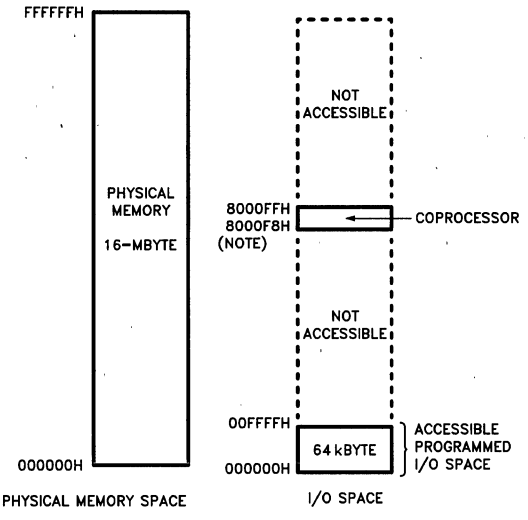
5.3 Memory and I/O Spaces

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. As shown in Figure 5.3, physical memory addresses range from 000000H to 0FFFFFFH (16 megabytes) and I/O addresses from 000000H to 00FFFFH (64 kilobytes). Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 8000F8H to 8000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A_{23} and M/IO# signals.

5.4 Bus Functional Description

The Intel386 SX Microprocessor has separate, parallel buses for data and address. The data bus is 16-bits in width, and bidirectional. The address bus provides a 24-bit value using 23 signals for the 23 upper-order address bits and 2 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled by several definition signals.

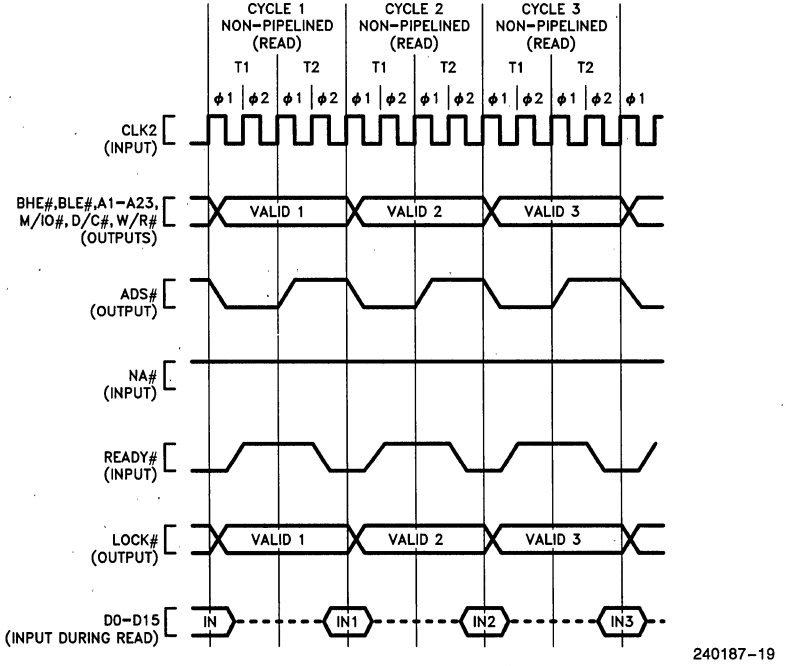
The definition of each bus cycle is given by three signals: M/IO#, W/R# and D/C#. At the same time, a valid address is present on the byte enable signals, BHE# and BLE#, and the other address signals A_{23}–A_1. A status signal, ADS#, indicates



NOTE:
Since A23 is HIGH during automatic communication with coprocessor, A23 HIGH and M/IO# LOW can be used to easily generate a coprocessor select signal.

240187-18

Figure 5.3. Physical Memory and I/O Spaces



240187-19

Fastest non-pipelined bus cycles consist of T1 and T2

Figure 5.4. Fastest Read Cycles with Non-pipelined Address Timing

when the Intel386 SX Microprocessor issues a new bus cycle definition and address.

Collectively, the address bus, data bus and all associated control signals are referred to simply as 'the bus'. When active, the bus performs one of the bus cycles below:

1. Read from memory space
2. Locked read from memory space
3. Write to memory space
4. Locked write to memory space
5. Read from I/O space (or coprocessor)
6. Write to I/O space (or coprocessor)
7. Interrupt acknowledge (always locked)
8. Indicate halt, or indicate shutdown

Table 5.2 shows the encoding of the bus cycle definition signals for each bus cycle. See **Bus Cycle Definition Signals** for additional information.

When the Intel386 SX Microprocessor bus is not performing one of the activities listed above, it is either Idle or in the Hold Acknowledge state, which may be detected externally. The idle state can be identified by the Intel386 SX Microprocessor giving no further assertions on its address strobe output (ADS#) since the beginning of its most recent bus cycle, and the most recent bus cycle having been terminated. The hold acknowledge state is identified by the Intel386 SX Microprocessor asserting its hold acknowledge (HLDA) output.

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

The fastest Intel386 SX Microprocessor bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown by Figure 5.4. The bus states in each cycle are named T1 and T2. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough.

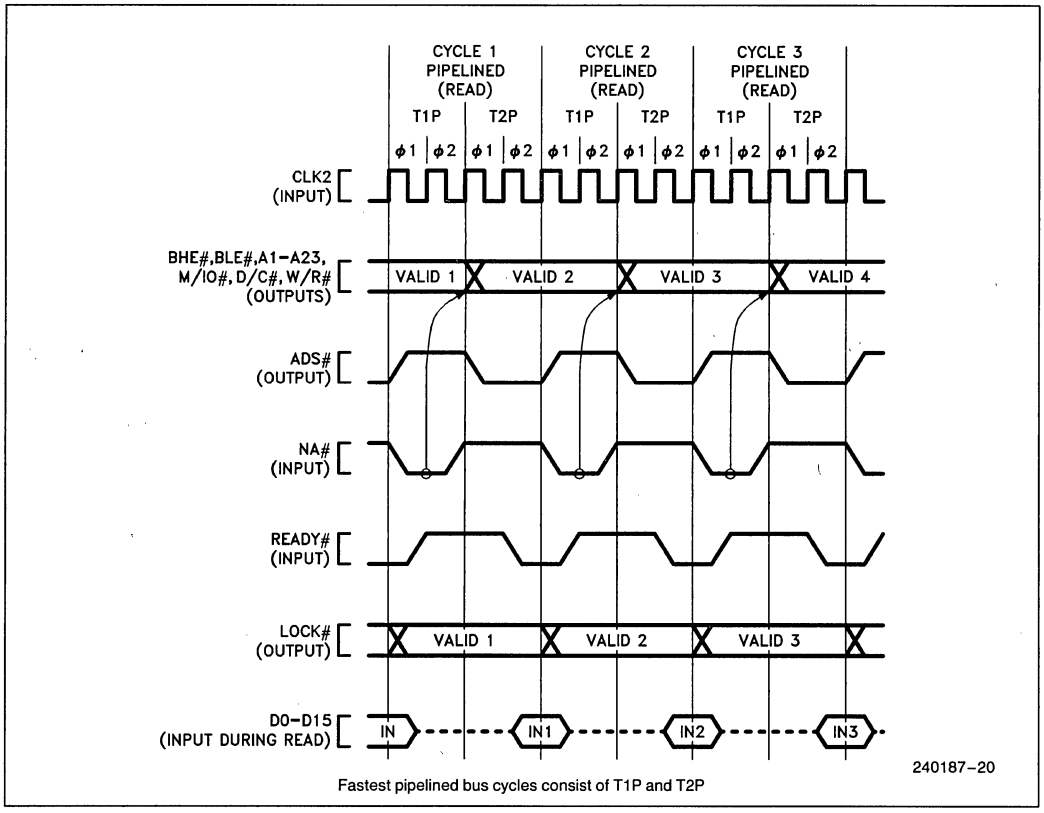


Figure 5.5. Fastest Read Cycles with Pipelined Address Timing

Every bus cycle continues until it is acknowledged by the external system hardware, using the Intel386 SX Microprocessor READY# input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If READY# is not immediately asserted however, T2 states are repeated indefinitely until the READY# input is sampled active.

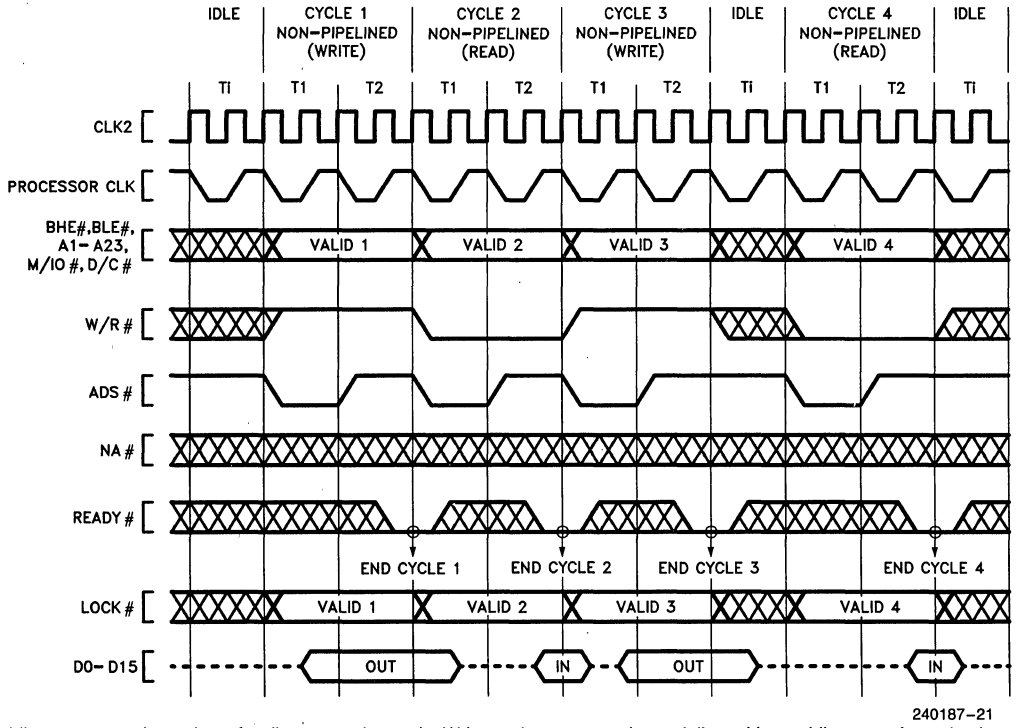
The address pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined address timing is selectable on a cycle-by-cycle basis with the Next Address (NA#) input.

When address pipelining is selected the address (BHE#, BLE# and A_{23}–A_1) and definition (W/R#, D/C#, M/IO# and LOCK#) of the next cycle are available before the end of the current cycle. To signal their availability, the Intel386 SX Microprocessor address status output (ADS#) is asserted. Figure 5.5 illustrates the fastest read cycles with pipelined address timing.

Note from Figure 5.5 the fastest bus cycles using pipelined address require only two bus states, named **T1P** and **T2P**. Therefore cycles with pipelined address timing allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased by one T-state time compared to that of a non-pipelined cycle.

READ AND WRITE CYCLES

Data transfers occur as a result of bus cycles, classified as read or write cycles. During read cycles, data is transferred from an external device to the processor. During write cycles, data is transferred from the processor to an external device.



Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

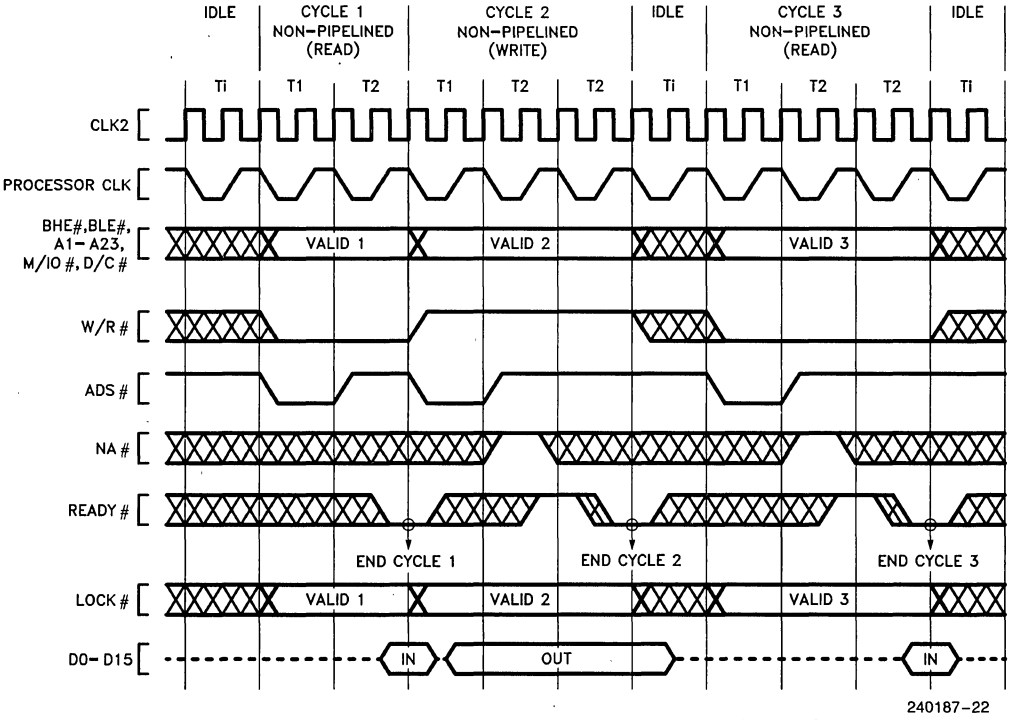
Figure 5.6. Various Bus Cycles with Non-Pipelined Address (zero wait states)

Two choices of address timing are dynamically selectable: non-pipelined or pipelined. After an idle bus state, the processor always uses non-pipelined address timing. However the NA# (Next Address) input may be asserted to select pipelined address timing for the next bus cycle. When pipelining is selected and the Intel386 SX Microprocessor has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by READY#.

Terminating a read or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the READY# input. Until acknowledged, the processor inserts wait states into the bus cycle, to allow adjustment for the speed of any external device. External hardware, which has decoded the address and bus cycle type, asserts the READY# input at the appropriate time.

At the end of the second bus state within the bus cycle, READY# is sampled. At that time, if external hardware acknowledges the bus cycle by asserting READY#, the bus cycle terminates as shown in Figure 5.6. If READY# is negated as in Figure 5.7, the Intel386 SX Microprocessor executes another bus state (a wait state) and READY# is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by READY# asserted.

When the current cycle is acknowledged, the Intel386 SX Microprocessor terminates it. When a read cycle is acknowledged, the Intel386 SX Microprocessor latches the information present at its data pins. When a write cycle is acknowledged, the Intel386 SX CPU's write data remains valid throughout phase one of the next bus state, to provide write data hold time.



Idle states are shown here for diagram variety only. Write cycles are **not** always followed by an idle state. An active bus cycle can immediately follow the write cycle.

Figure 5.7. Various Bus Cycles with Non-Pipelined Address (various number of wait states)

Non-Pipelined Address

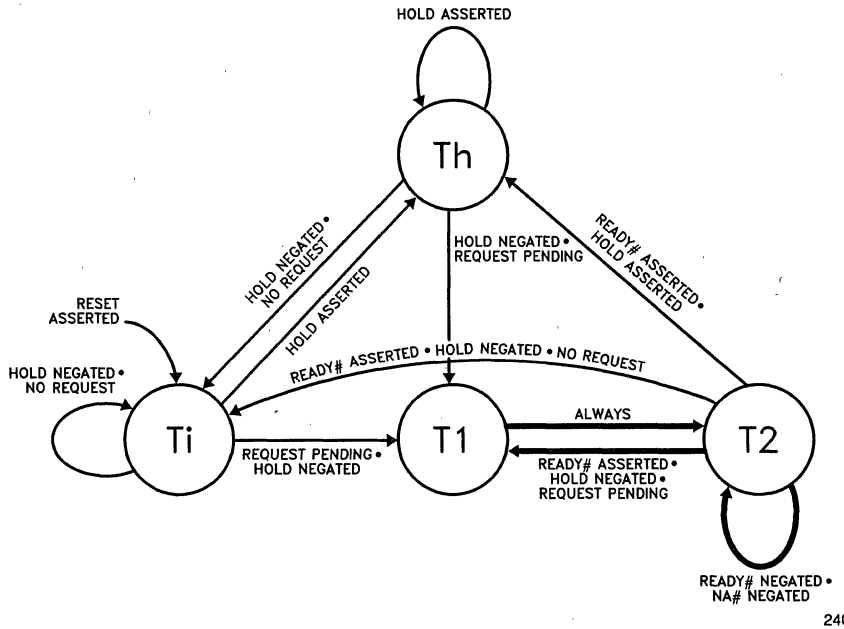
Any bus cycle may be performed with non-pipelined address timing. For example, Figure 5.6 shows a mixture of read and write cycles with non-pipelined address timing. Figure 5.6 shows that the fastest possible cycles with non-pipelined address have two bus states per bus cycle. The states are named T1 and T2. In phase one of T1, the address signals and bus cycle definition signals are driven valid and, to signal their availability, address strobe (ADS#) is simultaneously asserted.

During read or write cycles, the data bus behaves as follows. If the cycle is a read, the Intel386 SX Microprocessor floats its data signals to allow driving by the external device being addressed. **The Intel386 SX Microprocessor requires that all data bus pins be at a valid logic state (HIGH or LOW) at the end of each read cycle, when READY# is asserted. The system MUST be designed to meet this requirement.** If the cycle is a write, data signals are driven by the Intel386 SX Microprocessor beginning in phase two of T1 until phase one of the bus state following cycle acknowledgment.

Figure 5.7 illustrates non-pipelined bus cycles with one wait state added to Cycles 2 and 3. READY# is sampled inactive at the end of the first T2 in Cycles 2 and 3. Therefore Cycles 2 and 3 have T2 repeated again. At the end of the second T2, READY# is sampled active.

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states. When wait states are added and it is desirable to maintain non-pipelined address timing, it is necessary to negate NA# during each T2 state except the last one, as shown in Figure 5.7 Cycles 2 and 3. If NA# is sampled active during a T2 other than the last one, the next state would be T2I or T2P instead of another T2.

When address pipelining is not used, the bus states and transitions are completely illustrated by Figure 5.8. The bus transitions between four possible states, T1, T2, T_i, and T_h. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise the bus may be idle, T_i, or in the hold acknowledge state T_h.



Bus States:
T1—first clock of a non-pipelined bus cycle (Intel386™ SX CPU drives new address and asserts ADS#).
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.
Ti—idle state.
Th—hold acknowledge state (Intel386 SX CPU asserts HLDA).
The fastest bus cycle consists of two states T1 and T2.
Four basic bus states describe bus operation when not using pipelined address.

Figure 5.8. Bus States (not using pipelined address)

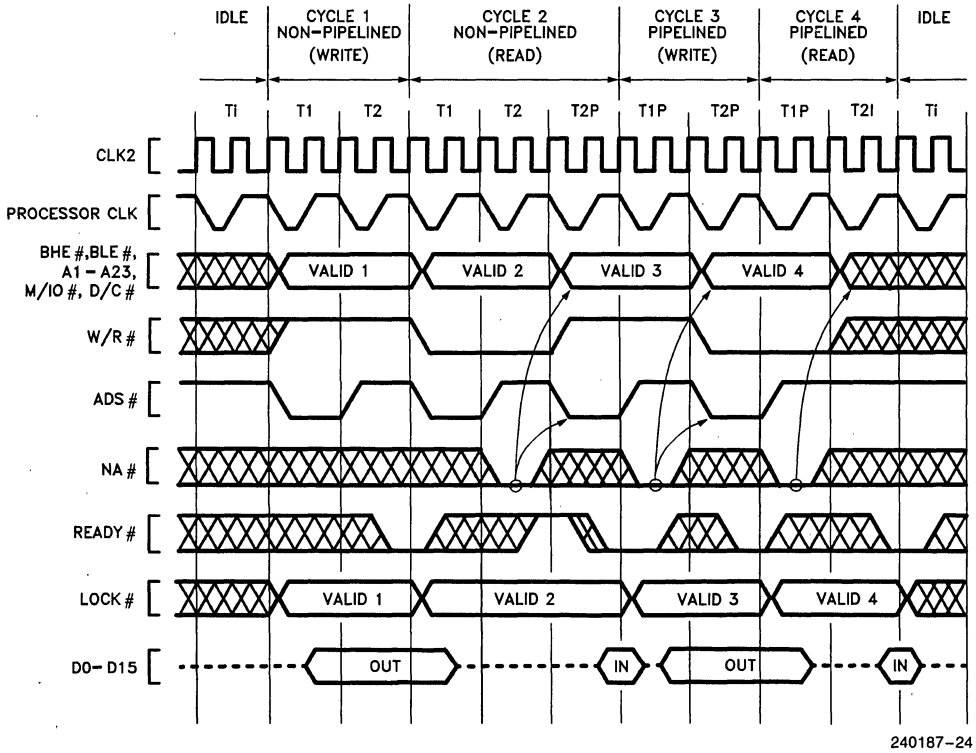
Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and NA# is inactive, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle if a bus request is pending internally, or T_i if there is no bus request pending, or T_h if the HOLD input is being asserted.

Use of pipelined address allows the Intel386 SX Microprocessor to enter three additional bus states not shown in Figure 5.8. Figure 5.12 is the complete bus state diagram, including pipelined address cycles.

Pipelined Address

Address pipelining is the option of requesting the address and the bus cycle definition of the next internally pending bus cycle before the current bus cycle is acknowledged with READY# asserted. ADS# is asserted by the Intel386 SX Microprocessor when the next address is issued. The address pipelining option is controlled on a cycle-by-cycle basis with the NA# input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles NA# is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 5.9, during which NA# is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).



240187-24

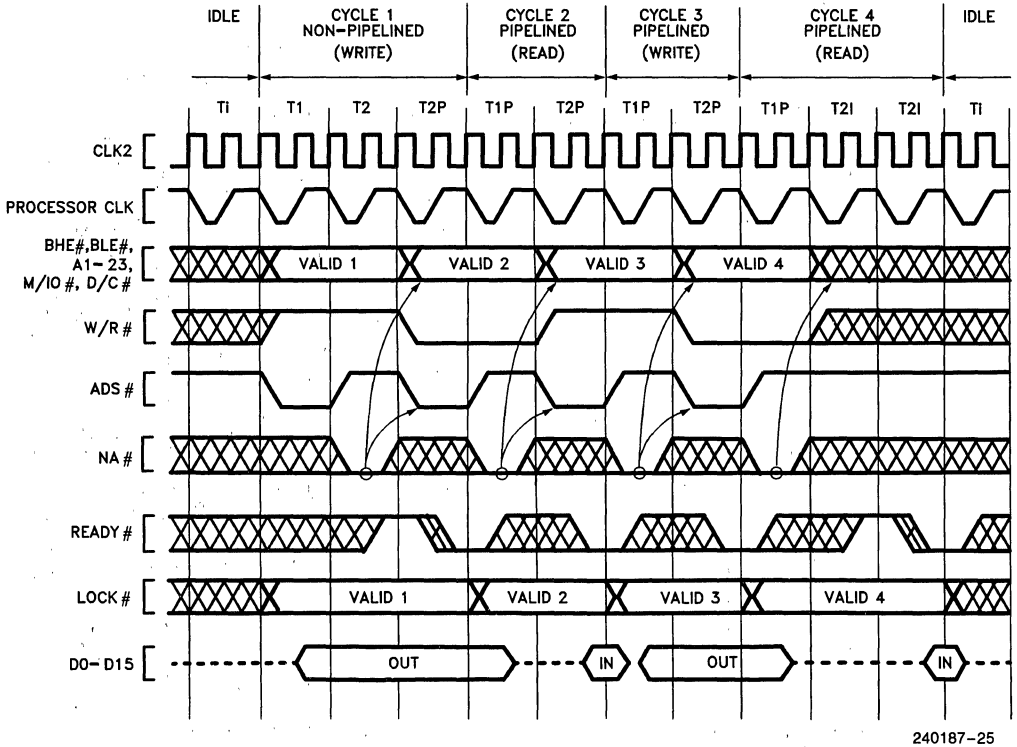
Following any idle bus state (Ti), addresses are non-pipelined. Within non-pipelined bus cycles, NA# is only sampled during wait states. Therefore, to begin address pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cycle 2 above).

Figure 5.9. Transitioning to Pipelined Address During Burst of Bus Cycles

If NA# is sampled active, the Intel386 SX Microprocessor is free to drive the address and bus cycle definition of the next bus cycle, and assert ADS#, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of address pipelining, the Intel386 SX Microprocessor has the following characteristics:

1. The next address may appear as early as the bus state after NA# was sampled active (see Figures 5.9 or 5.10). In that case, state T2P is entered immediately. However, when there is not an internal bus request already pending, the next address will not be available immediately after NA# is asserted and T2I is entered instead of T2P (see Figure 5.11 Cycle 3). Provided the current bus cycle isn't yet acknowledged by READY# asserted, T2P will be entered as soon as the Intel386 SX Microprocessor does drive the next address. External hardware should therefore observe the ADS# output as confirmation the next address is actually being driven on the bus.
2. Any address which is validated by a pulse on the ADS# output will remain stable on the address pins for at least two processor clock periods. The Intel386 SX Microprocessor cannot produce a new address more frequently than every two processor clock periods (see Figures 5.9, 5.10, and 5.11).
3. Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 5.11 Cycle 1).



Following any bus state (Ti) the address is always non-pipelined and NA# is only sampled during wait states. To start address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (cycle 1 above) The pipelined cycles (2, 3, 4 above) are shown with various numbers of wait states.

Figure 5.10. Fastest Transition to Pipelined Address Following Idle Bus State

The complete bus state transition diagram, including operation with pipelined address is given by Figure 5.12. Note it is a superset of the diagram for non-pipelined address only, and the three additional bus states for pipelined address are drawn in bold.

The fastest bus cycle with pipelined address consists of just two bus states, T1P and T2P (recall for non-pipelined address it is T1 and T2). T1P is the first bus state of a pipelined cycle.

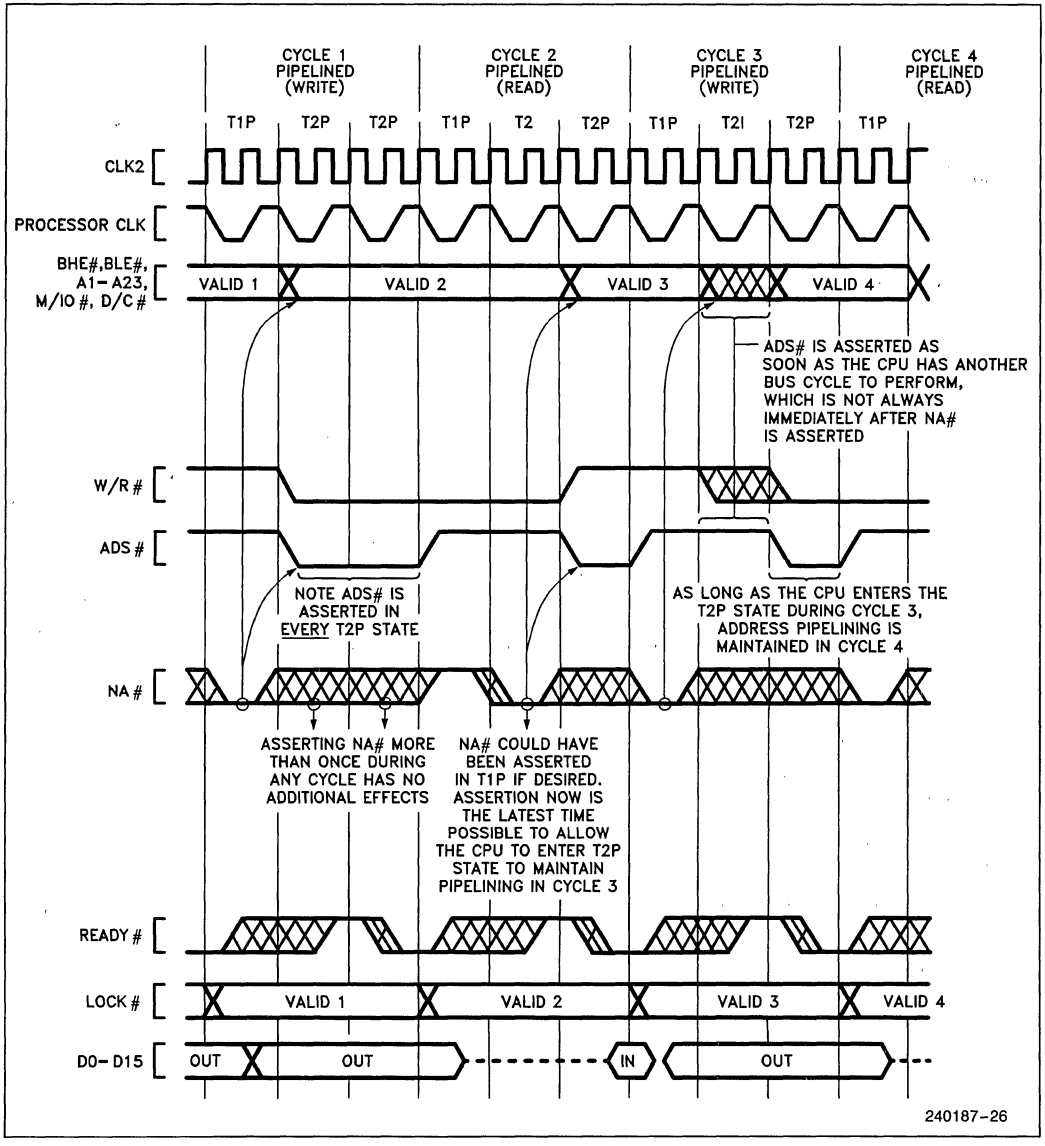
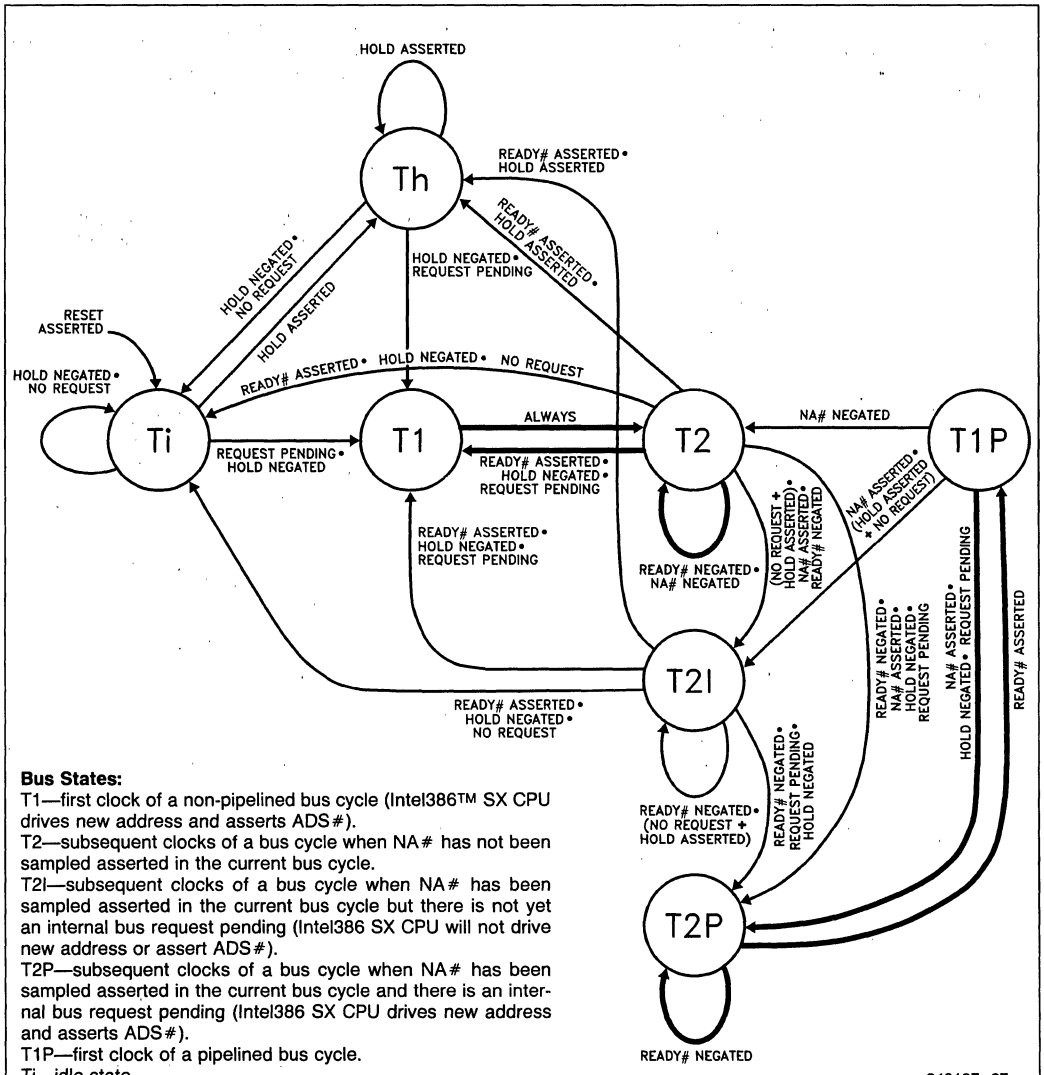


Figure 5.11. Details of Address Pipelining During Cycles with Wait States



Bus States:
T1—first clock of a non-pipelined bus cycle (Intel386™ SX CPU drives new address and asserts ADS#).
T2—subsequent clocks of a bus cycle when NA# has not been sampled asserted in the current bus cycle.
T2I—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (Intel386 SX CPU will not drive new address or assert ADS#).
T2P—subsequent clocks of a bus cycle when NA# has been sampled asserted in the current bus cycle and there is an internal bus request pending (Intel386 SX CPU drives new address and asserts ADS#).
T1P—first clock of a pipelined bus cycle.
Ti—idle state.
Th—hold acknowledge state (Intel386 SX CPU asserts HLDA).
Asserting NA# for pipelined address gives access to three more bus states: T2I, T2P and T1P.
Using pipelined address, the fastest bus cycle consists of T1P and T2P.

Figure 5.12. Complete Bus States (including pipelined address)

Initiating and Maintaining Pipelined Address

Using the state diagram Figure 5.12, observe the transitions from an idle state, T_i, to the beginning of a pipelined bus cycle T1P. From an idle state, T_i, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided NA# is asserted and the first bus cycle ends in a T2P state (the address for the next bus cycle is driven during T2P). The fastest path from an idle state to a bus cycle with pipelined address is shown in bold below:

T_i, T_i, T_i, T1 - T2 - T2P, T1P - T2P,

idle states	non-pipelined cycle	pipelined cycle

T1-T2-T2P are the states of the bus cycle that establish address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:

T_h, T_h, T_h, T1 - T2 - T2P, T1P - T2P,

hold acknowledge states	non-pipelined cycle	pipelined cycle

The transition to pipelined address is shown functionally by Figure 5.10 Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3 and 4, which are pipelined. The NA# input is asserted at the appropriate time to select address pipelining for Cycles 2, 3 and 4.

Once a bus cycle is in progress and the current address has been valid for one entire bus state, the NA# input is sampled at the end of every phase one until the bus cycle is acknowledged. Sampling begins in T2 during Cycle 1 in Figure 5.10. Once NA# is sampled active during the current cycle, the Intel386 SX Microprocessor is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 5.10 Cycle 1 for example, the next address is driven during state T2P. Thus Cycle 1 makes the transition to pipelined address timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as READY# asserted terminates Cycle 1.

Examples of transition bus cycles are Figure 5.10 Cycle 1 and Figure 5.9 Cycle 2. Figure 5.10 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 5.9 Cycle 2 shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 (NA# is asserted at that time), and T2P (provided the Intel386 SX Microprocessor has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note that only three states (T1, T2 and T2P) are required in a bus cycle performing a **transition** from non-pipelined address into pipelined address timing, for example Figure 5.10 Cycle 1. Figure 5.10 Cycles 2, 3 and 4 show that address pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting NA# and detecting that the Intel386 SX Microprocessor enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of ADS#. Figures 5.9 and 5.10 however, each show pipelining ending after Cycle 4 because Cycle 4 ends in T2I. This indicates the Intel386 SX Microprocessor didn't have an internal bus request prior to the acknowledgement of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

Realistically, address pipelining is almost always maintained as long as NA# is sampled asserted. This is so because in the absence of any other request, a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore, address pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD inactive) and NA# is sampled active in each of the bus cycles.

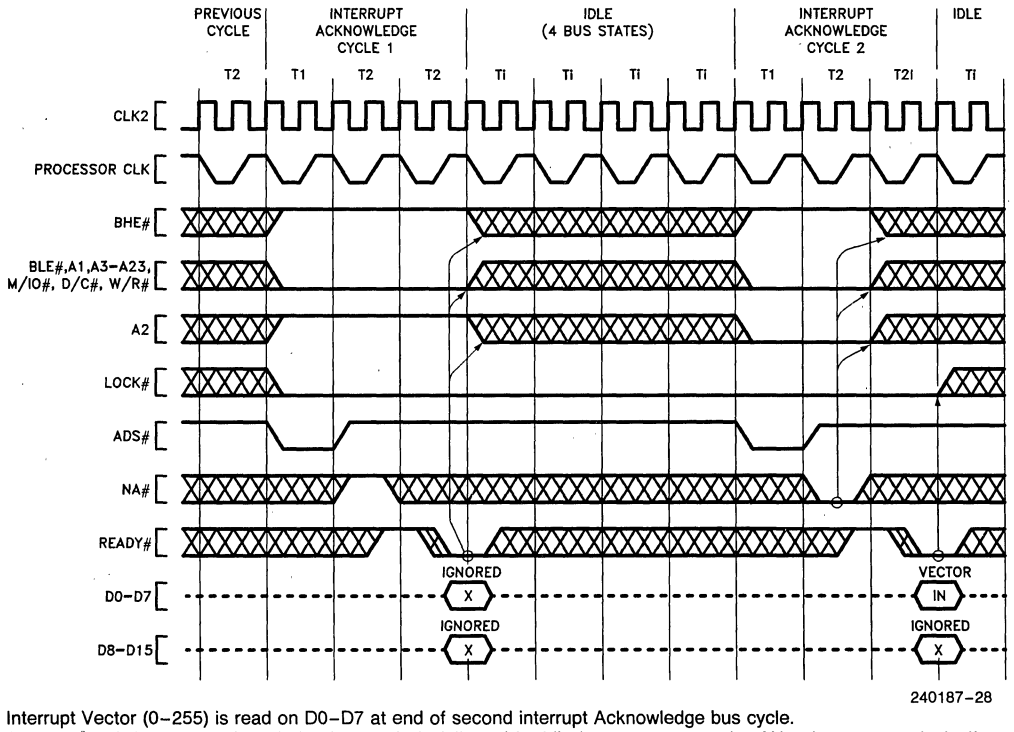
INTERRUPT ACKNOWLEDGE (INTA) CYCLES

In response to an interrupt request on the INTR input when interrupts are enabled, the Intel386 SX Microprocessor performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by READY# sampled active.

The state of A_2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A_{23}–A_3, A_1, BLE# LOW, A_2 and BHE# HIGH). The byte address driven during the second interrupt acknowledge cycle is 0 (A_{23}–A_1, BLE# LOW, and BHE# HIGH).

The LOCK# output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states, T_i, are inserted by the Intel386 SX Microprocessor between the two interrupt acknowledge cycles for compatibility with spec TRHRL of the 8259A Interrupt Controller.

During both interrupt acknowledge cycles, D_{15}–D_0 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the Intel386 SX Microprocessor will read an external interrupt vector from D_7–D_0 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.



Interrupt Vector (0–255) is read on D0–D7 at end of second interrupt Acknowledge bus cycle.
Because each Interrupt Acknowledge bus cycle is followed by idle bus states. asserting NA# has no practical effect. Choose the approach which is simplest for your system hardware design.

Figure 5.13. Interrupt Acknowledge Cycles

HALT INDICATION CYCLE

The execution unit halts as a result of executing a HLT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals shown on page 40, **Bus Cycle Definition Signals**, and an address of 2. The halt indication cycle must be acknowledged by READY# asserted. A halted Intel386 SX Microprocessor resumes execution when INTR (if interrupts are enabled), NMI or RESET is asserted.

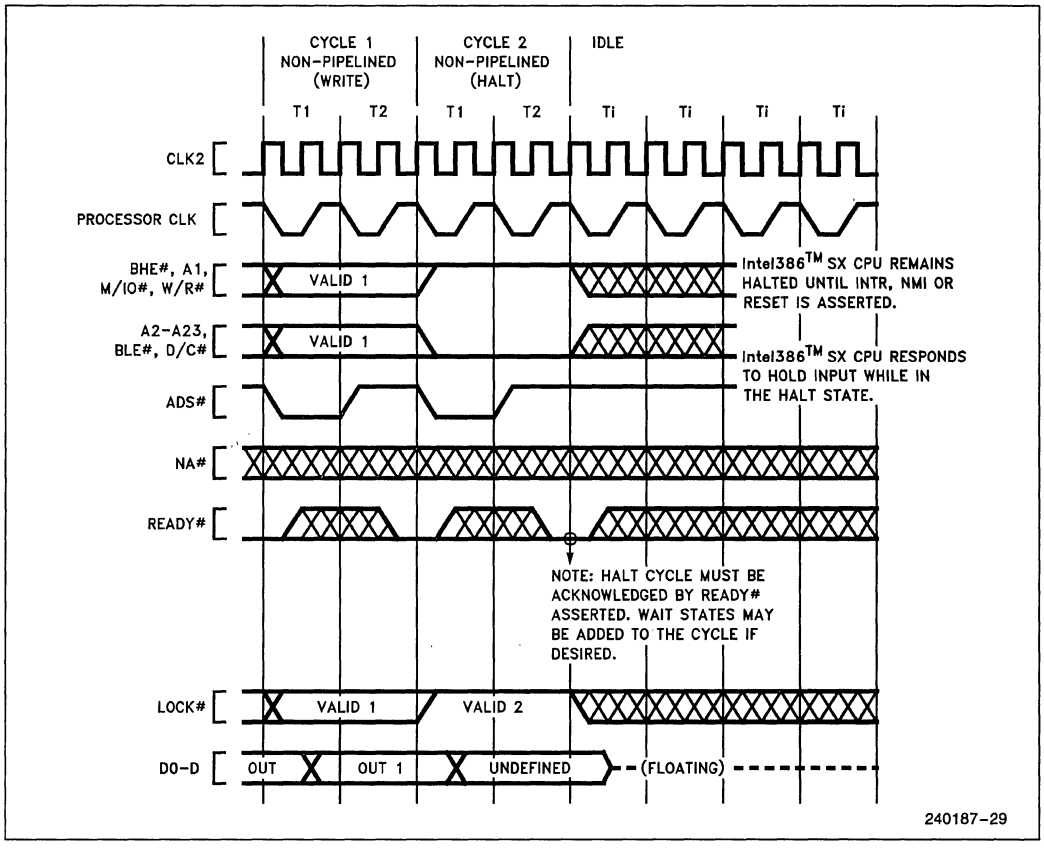


Figure 5.14. Example Halt Indication Cycle from Non-Pipelined Cycle

SHUTDOWN INDICATION CYCLE

The Intel386 SX Microprocessor shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown in **Bus Cycle Definition Signals** and an address of 0. The shutdown indication cycle must be acknowledged by READY# asserted. A shutdown Intel386 SX Microprocessor resumes execution when NMI or RESET is asserted.

ENTERING AND EXITING HOLD ACKNOWLEDGE

The bus hold acknowledge state, T_h, is entered in response to the HOLD input being asserted. In the bus hold acknowledge state, the Intel386 SX Microprocessor floats all outputs or bidirectional signals, except for HLDA. HLDA is asserted as long as the Intel386 SX Microprocessor remains in the bus hold acknowledge state. In the bus hold acknowledge state, all inputs except HOLD, FLT# and RESET are ignored.

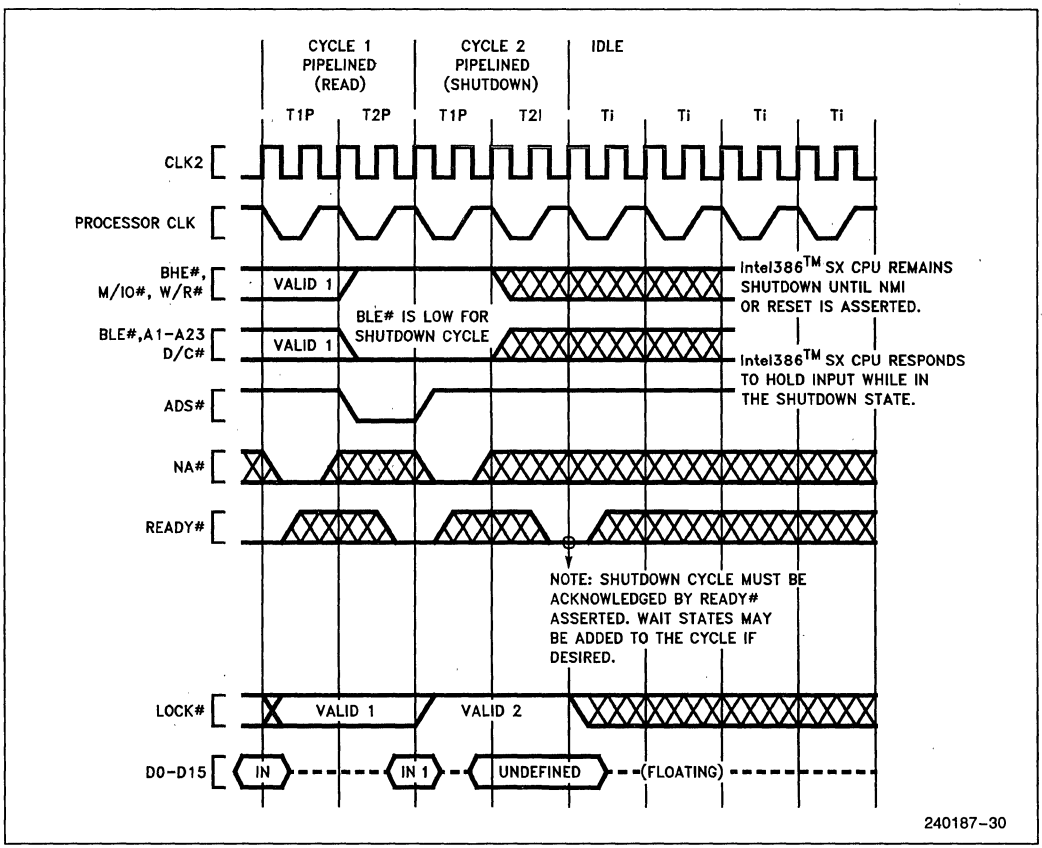


Figure 5.15. Example Shutdown Indication Cycle from Non-Pipelined Cycle

T_h may be entered from a bus idle state as in Figure 5.16 or after the acknowledgement of the current physical bus cycle if the LOCK# signal is not asserted, as in Figures 5.17 and 5.18.

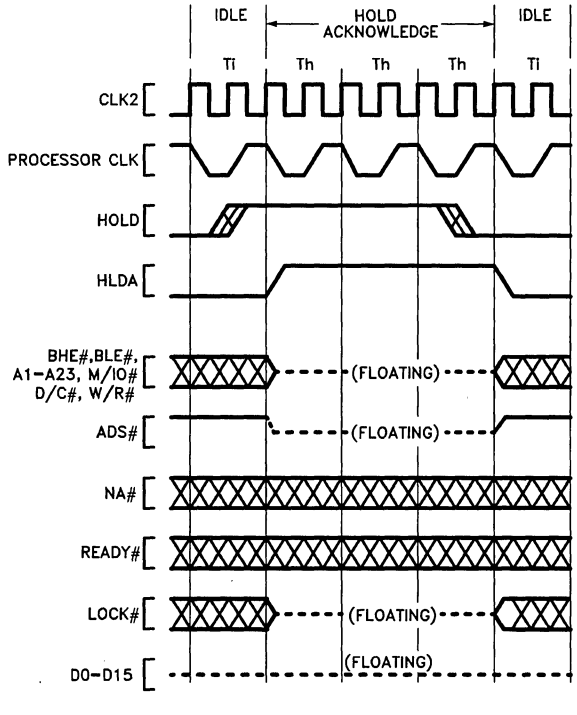
T_h is exited in response to the HOLD input being negated. The following state will be T_i as in Figure 5.16 if no bus request is pending. The following bus state will be T1 if a bus request is internally pending, as in Figures 5.17 and 5.18. T_h is exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in T_h, the event is remembered as a nonmaskable interrupt 2 and is serviced when T_h is exited unless the Intel386 SX Microprocessor is reset before T_h is exited.

RESET DURING HOLD ACKNOWLEDGE

RESET being asserted takes priority over HOLD being asserted. If RESET is asserted while HOLD remains asserted, the Intel386 SX Microprocessor drives its pins to defined states during reset, as in **Table 5.5 Pin State During Reset**, and performs internal reset activity as usual.

If HOLD remains asserted when RESET is inactive, the Intel386 SX Microprocessor enters the hold acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the Intel386 SX Microprocessor would otherwise perform its first bus cycle.



NOTE:
For maximum design flexibility the Intel386™ SX CPU has no internal pullup resistors on its outputs. Your design may require an external pullup on ADS# and other outputs to keep them negated during float periods.

Figure 5.16. Requesting Hold from Idle Bus

FLOAT

Activating the FLT# input floats all Intel386 SX bidirectional and output signals, including HLDA. Asserting FLT# isolates the Intel386 SX from the surrounding circuitry.

As the Intel386 SX is packaged in a surface mount PQFP, it cannot be removed from the motherboard when In-Circuit Emulation (ICE) is needed. The FLT# input allows the Intel386 SX to be electrically isolated from the surrounding circuitry. This allows connection of an emulator to the Intel386 SX PQFP without removing it from the PCB. This method of emulation is referred to as ON-Circuit Emulation (ONCE).

ENTERING AND EXITING FLOAT

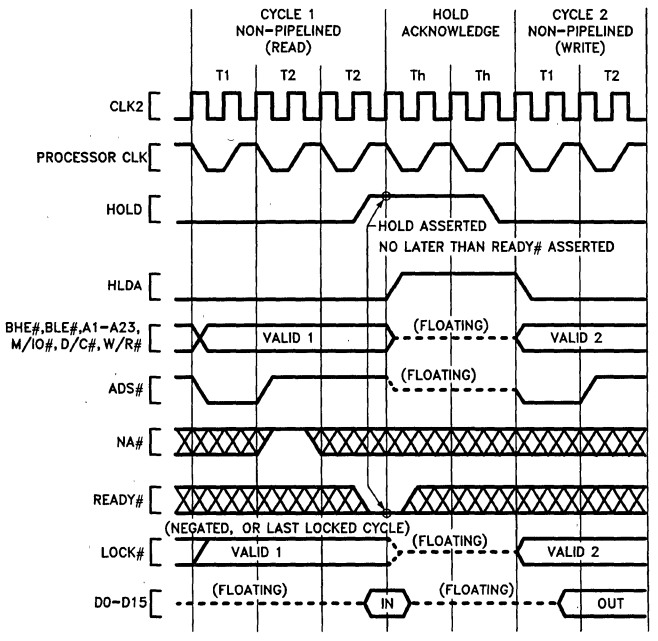
FLT# is an asynchronous, active-low input. It is recognized on the rising edge of CLK2. When recognized, it aborts the current bus cycle and floats the outputs of the Intel386 SX (Figure 5.20). FLT# must be held low for a minimum of 16 CLK2 cycles. Reset should be asserted and held asserted until after FLT# is deasserted. This will ensure that the Intel386 SX will exit float in a valid state.

Asserting the FLT# input unconditionally aborts the current bus cycle and forces the Intel386 SX into the FLOAT mode. Since activating FLT# unconditionally forces the Intel386 SX into FLOAT mode, the Intel386 SX is not guaranteed to enter FLOAT in a valid state. After deactivating FLT#, the Intel386 SX is not guaranteed to exit FLOAT mode in a valid state. This is not a problem as the FLT# pin is meant to be used only during ONCE. After exiting FLOAT, the Intel386 SX must be reset to return it to a valid state. Reset should be asserted before FLT# is deasserted. This will ensure that the Intel386 SX will exit float in a valid state.

FLT# has an internal pull-up resistor, and if it is not used it should be unconnected.

BUS ACTIVITY DURING AND FOLLOWING RESET

RESET is the highest priority input signal, capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage, or idle states or bus hold acknowledge states discontinued so that the reset state is established.



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t_{23} and t_{24}) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

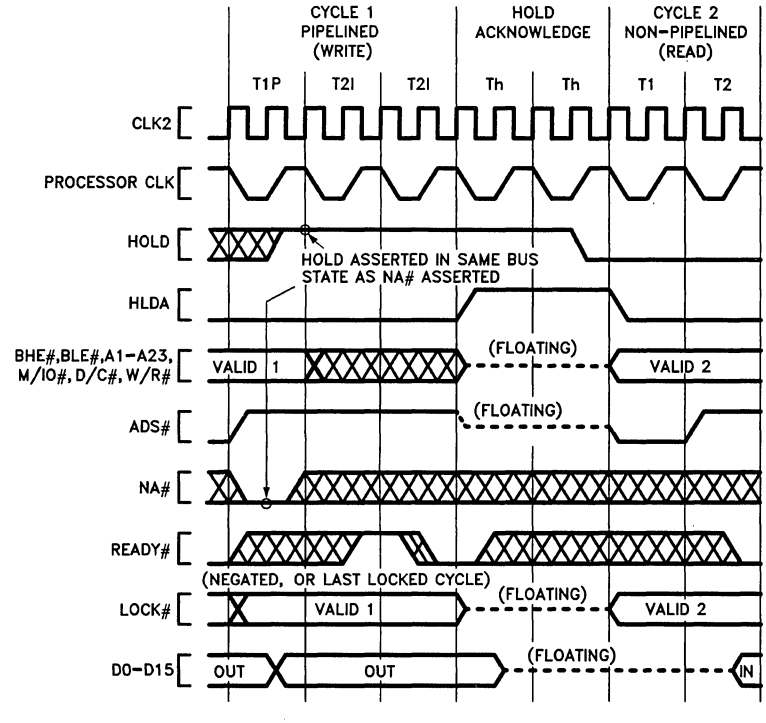
Figure 5.17. Requesting Hold from Active Bus (NA# inactive)

RESET should remain asserted for at least 15 CLK2 periods to ensure it is recognized throughout the Intel386 SX Microprocessor, and at least 80 CLK2 periods if self-test is going to be requested at the falling edge. RESET asserted pulses less than 15 CLK2 periods may not be recognized. RESET pulses less than 80 CLK2 periods followed by a self-test may cause the self-test to report a failure when no true failure exists.

Provided the RESET falling edge meets setup and hold times t_{25} and t_{26}, the internal processor clock phase is defined at that time as illustrated by Figure 5.19 and Figure 7.7.

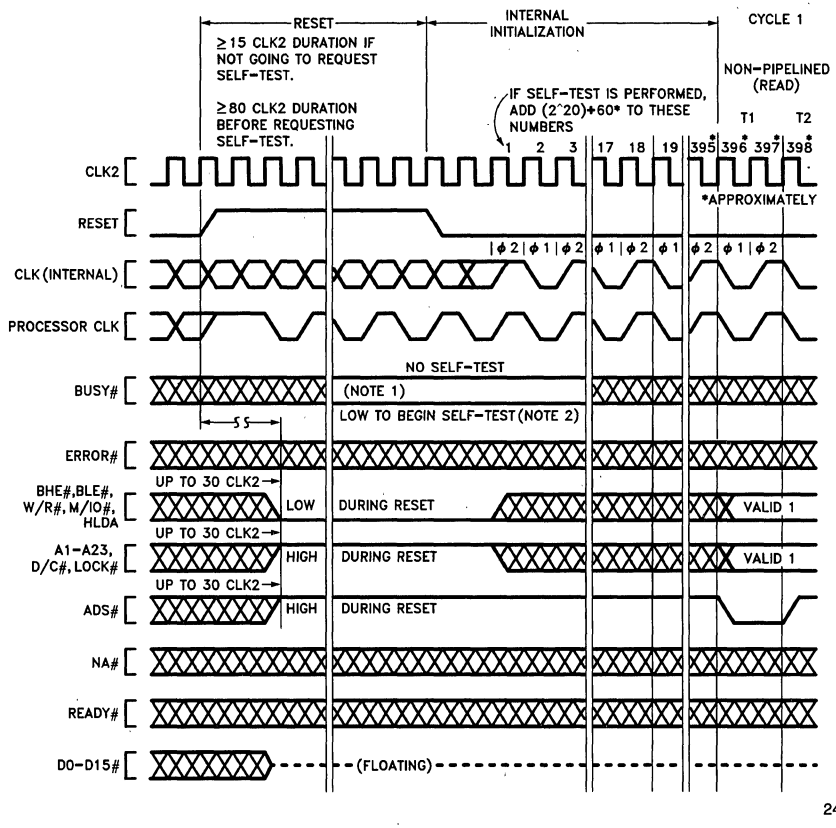
A self-test may be requested at the time RESET goes inactive by having the BUSY# input at a LOW level as shown in Figure 5.19. The self-test requires approximately (2^{20} + 60) CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the Intel386 SX Microprocessor attempts to proceed with the reset sequence afterwards.

After the RESET falling edge (and after the self-test if it was requested) the Intel386 SX Microprocessor performs an internal initialization sequence for approximately 350 to 450 CLK2 periods.



NOTE:
HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t23 and t24) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

Figure 5.18. Requesting Hold from Idle Bus (NA# active)



NOTES:
1. BUSY# should be held stable for 8 CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested the outputs remain in their reset state as shown here.

Figure 5.19. Bus Activity from Reset Until First Code Fetch

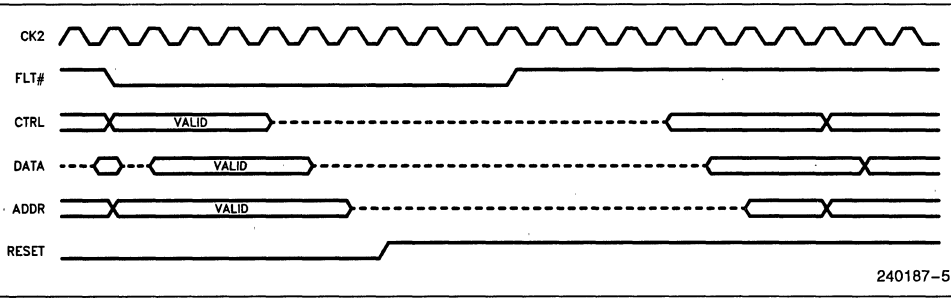


Figure 5.20. Entering and Exiting, FLT#

5.5 Self-test Signature

Upon completion of self-test (if self-test was requested by driving BUSY# LOW at the falling edge of RESET) the EAX register will contain a signature of 00000000H indicating the Intel386 SX Microprocessor passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000H, applies to all revision levels. Any non-zero signature indicates the unit is faulty.

5.6 Component and Revision Identifiers

To assist users, the Intel386 SX Microprocessor after reset holds a component identifier and revision identifier in its DX register. The upper 8 bits of DX hold 23H as identification of the Intel386 SX Microprocessor (the lower nibble, 03H, refers to the Intel386 DX Architecture. The upper nibble, 02H, refers to the second member of the Intel386 DX Family). The lower 8 bits of DX hold an 8-bit unsigned binary number related to the component revision level. The revision identifier will, in general, chronologically track those component steppings which are intended to have certain improvements or distinction from previous steppings. The Intel386 SX Microprocessor revision identifier will track that of the Intel386 DX CPU where possible.

The revision identifier is intended to assist users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision, or to follow a completely uniform numerical sequence, depending on the type or intention of revision, or manufacturing materials required to be changed. Intel has sole discretion over these characteristics of the component.

Table 5.7. Component and Revision Identifier History

Stepping	Revision Identifier
A0	04H
B	05H
C	08H
D	08H
E	08H

5.7 Coprocessor Interfacing

The Intel386 SX Microprocessor provides an automatic interface for the Intel Intel387 SX numeric floating-point coprocessor. The Intel387 SX coprocessor uses an I/O mapped interface driven automatically by the Intel386 SX Microprocessor and assisted by three dedicated signals: BUSY#, ERROR# and PEREQ.

As the Intel386 SX Microprocessor begins supporting a coprocessor instruction, it tests the BUSY# and ERROR# signals to determine if the coprocessor can accept its next instruction. Thus, the BUSY# and ERROR# inputs eliminate the need for any 'preamble' bus cycles for communication between processor and coprocessor. The Intel387 SX can be given its command opcode immediately. The dedicated signals provide instruction synchronization, and eliminate the need of using the WAIT opcode (9BH) for Intel387 SX instruction synchronization (the WAIT opcode was required when the 8086 or 8088 was used with the 8087 coprocessor).

Custom coprocessors can be included in Intel386 SX Microprocessor based systems by memory-mapped or I/O-mapped interfaces. Such coprocessor interfaces allow a completely custom protocol, and are not limited to a set of coprocessor protocol 'primitives'. Instead, memory-mapped or I/O-mapped interfaces may use all applicable instructions for high-speed coprocessor communication. The BUSY# and ERROR# inputs of the Intel386 SX Microprocessor may also be used for the custom coprocessor interface, if such hardware assist is desired. These signals can be tested by the WAIT opcode (9BH). The WAIT instruction will wait until the BUSY# input is inactive (interruptable by an NMI or enabled INTR input), but generates an exception 16 fault if the ERROR# pin is active when the BUSY# goes (or is) inactive. If the custom coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the Intel386 SX CPU's on-chip paging or segmentation mechanisms. If the custom interface is I/O-mapped, protection of the interface can be provided with the IOPL (I/O Privilege Level) mechanism.

The Intel387 SX numeric coprocessor interface is I/O mapped as shown in Table 5.8. Note that the Intel387 SX coprocessor interface addresses are beyond the 0H-0FFFFH range for programmed I/O. When the Intel386 SX Microprocessor supports the Intel387 SX coprocessor, the Intel386 SX Microprocessor automatically generates bus cycles to the coprocessor interface addresses.

Table 5.8. Numeric Coprocessor Port Addresses

Address in Intel386 SX CPU I/O Space	Intel387 SX Coprocessor Register
8000F8H	Opcode Register
8000FCH/8000FEH*	Operand Register

*Generated as 2nd bus cycle during Dword transfer.

To correctly map the Intel387 SX registers to the appropriate I/O addresses, connect the CMD0 and CMD1 lines of the Intel387 SX as listed in Table 5.9.

Table 5.9. Connections for CMD0 and CMD1 Inputs for the Intel387 SX

Signal	Connection
CMD0	Connect directly to Intel386 SX CPU A2 signal
CMD1	Connect to ground.

Software Testing for Coprocessor Presence

When software is used to test for coprocessor (Intel387 SX) presence, it should use only the following coprocessor opcodes: FINIT, FNINIT, FSTCW mem, FSTSW mem and FSTSW AX. To use other coprocessor opcodes when a coprocessor is known to be not present, first set EM = 1 in the Intel386 SX CPU's CR0 register.

6.0 PACKAGE THERMAL SPECIFICATIONS

The Intel386 SX Microprocessor is specified for operation when case temperature (T_C) is within the range of 0°C–100°C. The case temperature may be measured in any environment, to determine whether the Intel386 SX Microprocessor is within specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

The ambient temperature (T_a) is related to T_C and the thermal conductivity parameters θ_{ja} and θ_{jc} from the following equations (eqn. 3 is derived by eliminating the junction temperature (T_j) between eqns. 1 and 2):

1) $T_j = T_c + P*\theta_{jc}$
2) $T_a = T_j - P*\theta_{ja}$
3) $T_c = T_a + P*[\theta_{ja} - \theta_{jc}]$

Values for θ_{ja} and θ_{jc} are given in Table 6.1 for the 100 lead fine pitch. θ_{ja} is given at various airflows. The power (P) dissipated by the chip as heat is Vcc*Icc. A guaranteed maximum safe T_a can be calculated from eqn. 3 by using the maximum safe T_C of 100°C, along with the maximum power drawn by the chip in the given design, and θ_{jc} and θ_{ja} values from Table 6.1. (The θ_{ja} value depends on the airflow, measured at the top of the chip, provided by the system ventilation.)

7.0 ELECTRICAL SPECIFICATIONS

The following sections describe recommended electrical connections for the Intel386 SX Microprocessor, and its electrical specifications.

7.1 Power and Grounding

The Intel386 SX Microprocessor is implemented in CHMOS IV technology and has modest power requirements. However, its high clock frequency and 47 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 14 Vcc and 18 Vss pins separately feed functional units of the Intel386 SX Microprocessor.

Power and ground connections must be made to all external Vcc and Vss pins of the Intel386 SX Microprocessor. On the circuit board, all Vcc pins should be connected on a Vcc plane and all Vss pins should be connected on a GND plane.

POWER DECOUPLING RECOMMENDATIONS

Liberal decoupling capacitors should be placed near the Intel386 SX Microprocessor. The Intel386 SX Microprocessor driving its 24-bit address bus and 16-bit data bus at high frequencies can cause transient power surges, particularly when driving large capacitive loads. Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the Intel386 SX Microprocessor and decoupling capacitors as much as possible.

Table 6.1. Thermal Resistances (°C/Watt) θ_{jc} and θ_{ja}.

Package	θ_{jc}	θ_{ja} versus Airflow - ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
100 Lead Fine Pitch	7.5	34.5	29.5	25.5	22.5	21.5	21

Table 7.1. Recommended Resistor Pull-ups to Vcc

Pin	Signal	Pull-up Value	Purpose
16	ADS#	20 KΩ ±10%	Lightly pull ADS# inactive during Intel386™ SX CPU hold acknowledge states
26	LOCK#	20 KΩ ±10%	Lightly pull LOCK# inactive during Intel386™ SX CPU hold acknowledge states

RESISTOR RECOMMENDATIONS

The ERROR#, FLT# and BUSY# inputs have internal pull-up resistors of approximately 20 KΩ and the PEREQ input has an internal pull-down resistor of approximately 20 KΩ built into the Intel386 SX Microprocessor to keep these signals inactive when the Intel387 SX is not present in the system (or temporarily removed from its socket).

In typical designs, the external pull-up resistors shown in Table 7.1 are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pull-up resistors in other ways.

OTHER CONNECTION RECOMMENDATIONS

For reliable operation, always connect unused inputs to an appropriate signal level. N/C pins should always remain **unconnected. Connection of N/C pins to Vcc or Vss will result in component malfunction or incompatibility with future steppings of the Intel386 SX Microprocessor.**

Particularly when not using interrupts or bus hold (as when first prototyping), prevent any chance of spurious activity by connecting these associated inputs to GND:

Pin	Signal
40	INTR
38	NMI
4	HOLD

If not using address pipelining, connect pin 6, NA#, through a pull-up in the range of 20 KΩ to Vcc.

7.2 Maximum Ratings

Table 7.2. Maximum Ratings

Parameter	Maximum Rating
Storage temperature	−65 °C to 150 °C
Case temperature under bias	−65 °C to 110 °C
Supply voltage with respect to Vss	−.5V to 6.5V
Voltage on other pins	−.5V to (Vcc+.5)V

Table 7.2 gives stress ratings only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in section **7.3, D.C. Specifications**, and section **7.4, A.C. Specifications**.

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the Intel386 SX Microprocessor contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

7.3 D.C. Specifications

Functional operating range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 100°C

Table 7.3. Intel386™ SX Microprocessor D.C. Characteristics—33 MHz, 25 MHz, 20 MHz and 16 MHz

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC}+0.3	V	
V_{ILC}	CLK2 Input LOW Voltage	−0.3	+0.8	V	
V_{IHC}	CLK2 Input HIGH Voltage	V_{CC}−0.8	V_{CC}+0.3	V	
V_{OL}	Output LOW Voltage I_{OL}=4 mA: A_{23}–A_1,D_{15}–D_0 I_{OL}=5 mA: BHE#,BLE#,W/R#,D/C#, M/IO#,LOCK#,ADS#,HLDA		 0.45 0.45	 V V	
V_{OH}	Output HIGH Voltage I_{OH}=−1 mA: A_{23}–A_1,D_{15}–D_0 I_{OH}=−0.2 mA: A_{23}–A_1,D_{15}–D_0 I_{OH}=−0.9 mA: BHE#,BLE#,W/R#, D/C#,M/IO#,LOCK#, ADS#,HLDA I_{OH}=−0.18 mA: BHE#,BLE#,W/R#, D/C#,M/IO#,LOCK#, ADS#,HLDA	 2.4 V_{CC}−0.5 2.4 V_{CC}−0.5	 	 V V V V	
I_{LI}	Input Leakage Current (for all pins except PEREQ, BUSY#, FLT# and ERROR#)		±15	μA	0V≤V_{IN}≤V_{CC}
I_{IH}	Input Leakage Current (PEREQ pin)		200	μA	V_{IH}=2.4V, Note 1
I_{IL}	Input Leakage Current (BUSY#, ERROR# and FLT# pins)		−400	μA	V_{IL}=0.45V, Note 2
I_{LO}	Output Leakage Current		±15	μA	0.45V≤V_{OUT}≤V_{CC}
I_{CC}	Supply Current CLK2=32 MHz: with 16 MHz Intel386 SX CPU CLK2=40 MHz: with 20 MHz Intel386 SX CPU CLK2=50 MHz: with 25 MHz Intel386 SX CPU CLK2=66 MHz: with 33 MHz Intel386 SX CPU		 220 250 280 380	 mA mA mA mA	(See Note 3) I_{CC} typ=150 mA I_{CC} typ=180 mA I_{CC} typ=210 mA I_{CC} typ=290 mA
C_{IN}	Input Capacitance		10	pF	F_C=1 MHz, Note 4
C_{OUT}	Output or I/O Capacitance		12	pF	F_C=1 MHz, Note 4
C_{CLK}	CLK2 Capacitance		20	pF	F_C=1 MHz, Note 4

All values except I_{CC} tested at the minimum operating frequency of the part (CLK2 = 8 MHz).

NOTES:
1. PEREQ input has an internal pull-down resistor.
2. BUSY#, FLT# and ERROR# inputs each have an internal pull-up resistor.
3. I_{CC} max measurement at worst case frequency, V_{CC} and temperature, with 50 pF output load.
4. Not 100% tested.

Functional operating range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 100°C

Table 7.4. Low Power (LP) Intel386™ SX Microprocessor D.C. Characteristics—33 MHz, 25 MHz, 20 MHz, 16 MHz, and 12 MHz

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC}+0.3	V	
V_{ILC}	CLK2 Input LOW Voltage	−0.3	+0.8	V	
V_{IHC}	CLK2 Input HIGH Voltage	V_{CC}−0.8	V_{CC}+0.3	V	
V_{OL}	Output LOW Voltage I_{OL}=4 mA: A_{23}–A_1,D_{15}–D_0 I_{OL}=5 mA: BHE#,BLE#,W/R#,D/C#, M/IO#,LOCK#,ADS#,HLDA		 0.45 0.45	 V V	
V_{OH}	Output HIGH Voltage I_{OH}=−1 mA: A_{23}–A_1,D_{15}–D_0 I_{OH}=−0.2 mA: A_{23}–A_1,D_{15}–D_0 I_{OH}=−0.9 mA: BHE#,BLE#,W/R#, D/C#,M/IO#,LOCK#, ADS#,HLDA I_{OH}=−0.18 mA: BHE#,BLE#,W/R#, D/C#,M/IO#,LOCK#, ADS#,HLDA	 2.4 V_{CC}−0.5 2.4 V_{CC}−0.5	 	 V V V V	
I_{LI}	Input Leakage Current (for all pins except PEREQ, BUSY#, FLT# and ERROR#)		±15	μA	0V≤V_{IN}≤V_{CC}
I_{IH}	Input Leakage Current (PEREQ pin)		200	μA	V_{IH}=2.4V, Note 1
I_{IL}	Input Leakage Current (BUSY#, ERROR# and FLT# pins)		−400	μA	V_{IL}=0.45V, Note 2
I_{LO}	Output Leakage Current		±15	μA	0.45V≤V_{OUT}≤V_{CC}
I_{CC}	Supply Current CLK2=4 MHz CLK2=24 MHz: with 12 MHz Intel386 SX CPU CLK2=32 MHz: with 16 MHz Intel386 SX CPU CLK2=40 MHz: with 20 MHz Intel386 SX CPU CLK2=50 MHz: with 25 MHz Intel386 SX CPU CLK2=66 MHz: with 33 MHz Intel386 SX CPU		 100 190 220 250 280 380	 mA mA mA mA mA mA	(See Note 3) I_{CC} typ=50 mA I_{CC} typ=120 mA I_{CC} typ=150 mA I_{CC} typ=180 mA I_{CC} typ=210 mA I_{CC} typ=290 mA
C_{IN}	Input Capacitance		10	pF	F_C=1 MHz, Note 4
C_{OUT}	Output or I/O Capacitance		12	pF	F_C=1 MHz, Note 4
C_{CLK}	CLK2 Capacitance		20	pF	F_C=1 MHz, Note 4

All values except I_{CC} tested at the minimum operating frequency of the part (CLK2 = 4 MHz).

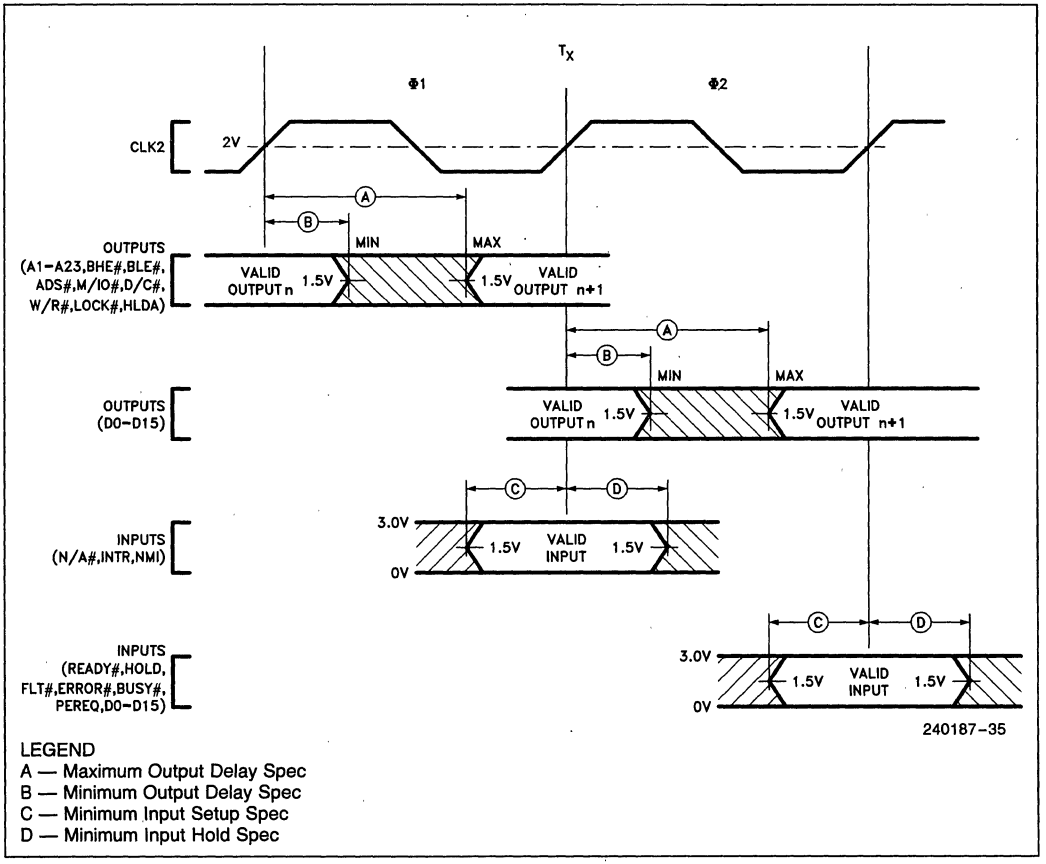
NOTES:
1. PEREQ input has an internal pull-down resistor.
2. BUSY#, FLT# and ERROR# inputs each have an internal pull-up resistor.
3. I_{CC} max measurement at worst case frequency, V_{CC} and temperature, with 50 pF output load.
4. Not 100% tested.

7.4 A.C. Specifications

The A.C. specifications given in Tables 7.5 through 7.8 consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the CLK2 rising edge crossing the 2.0V level.

A.C. spec measurement is defined by Figure 7.1. Inputs must be driven to the voltage levels indicated by Figure 7.1 when A.C. specifications are measured. Output delays are specified with minimum and maximum limits measured as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs ADS#, W/R#, D/C#, M/IO#, LOCK#, BHE#, BLE#, A_{23}–A_1 and HLDA only change at the beginning of phase one. D_{15}–D_0 (write cycles) only change at the beginning of phase two. The READY#, HOLD, BUSY#, ERROR#, PEREQ, FLT# and D_{15}–D_0 (read cycles) inputs are sampled at the beginning of phase one. The NA#, INTR and NMI inputs are sampled at the beginning of phase two.



LEGEND
A — Maximum Output Delay Spec
B — Minimum Output Delay Spec
C — Minimum Input Setup Spec
D — Minimum Input Hold Spec

Figure 7.1. Drive Levels and Measurement Points for A.C. Specifications

A.C. SPECIFICATIONS

Functional operating range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 100°C

Table 7.5. Intel386™ SX Microprocessor A.C. Characteristics—33 MHz and 25 MHz

Symbol	Parameter	33 MHz Intel386 SX		25 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max			
	Operating Frequency	4	33	4	25	MHz		Half CLK2 Frequency
t_1	CLK2 Period	15	125	20	125	ns	7.3	
t_{2a}	CLK2 HIGH Time	6.25		7		ns	7.3	at 2V(3)
t_{2b}	CLK2 HIGH Time	4.0		4		ns	7.3	at (V_{CC}−0.8)V(3); Note 3
t_{3a}	CLK2 LOW Time	6.25		7		ns	7.3	at 2V(3)
t_{3b}	CLK2 LOW Time	4.5		5		ns	7.3	at 0.8V(3)
t_4	CLK2 Fall Time		4		7	ns	7.3	(V_{CC}−0.8)V to 0.8V(3)
t_5	CLK2 Rise Time		4		7	ns	7.3	0.8V to (V_{CC}−0.8)V(3)
t_6	A_{23}–A_1 Valid Delay	4	15	4	17	ns	7.5	C_L = 50 pF(4)
t_7	A_{23}–A_1 Float Delay	4	20	4	30	ns	7.6	(Note 1)
t_8	BHE#, BLE#, LOCK# Valid Delay	4	15	4	17	ns	7.5	C_L = 50 pF(4)
t_9	BHE#, BLE#, LOCK# Float Delay	4	20	4	30	ns	7.6	(Note 1)
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	15	4	17	ns	7.5	C_L = 50 pF(4)
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	4	20	4	30	ns	7.6	(Note 1)
t_{12}	D_{15}–D_0 Write Data Valid Delay	7	23	7	23	ns	7.5	C_L = 50 pF(4, 5)
t_{12a}	D_{15}–D_0 Write Data Hold Time	2		2		ns		C_L = 50 pF(4)
t_{13}	D_{15}–D_0 Write Data Float Delay	4	17	4	22	ns	7.6	(Note 1)
t_{14}	HLDA Valid Delay	4	20	4	22	ns	7.6	C_L = 75 pF(4)
t_{15}	NA# Setup Time	5		5		ns	7.4	
t_{16}	NA# Hold Time	2		3		ns	7.4	
t_{19}	READY# Setup Time	7		9		ns	7.4	
t_{20}	READY# Hold Time	4		4		ns	7.4	
t_{21}	D_{15}–D_0 Read Data Setup Time	5		7		ns	7.4	
t_{22}	D_{15}–D_0 Read Data Hold Time	3		5		ns	7.4	

Functional operating range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 100°C

Table 7.5. Intel386™ SX Microprocessor A.C. Characteristics—33 MHz and 25 MHz (Continued)

Symbol	Parameter	33 MHz Intel386 SX		25 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max			
t_{23}	HOLD Setup Time	9		9		ns	7.4	
t_{24}	HOLD Hold Time	2		3		ns	7.4	
t_{25}	RESET Setup Time	5		8		ns	7.7	
t_{26}	RESET Hold Time	2		3		ns	7.7	
t_{27}	NMI, INTR Setup Time	5		6		ns	7.4	(Note 2)
t_{28}	NMI, INTR Hold Time	5		6		ns	7.4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY#, FLT# Setup Time	5		6		ns	7.4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY#, FLT# Hold Time	4		5		ns	7.4	(Note 2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with CL set at 50 pF. See Figures 7 and 8 for load capacitance derating curve.
5. Minimum time not 100% tested.

Table 7.6. Low Power (LP) Intel386™ SX Microprocessor A.C. Characteristics—33 MHz and 25 MHz

Symbol	Parameter	33 MHz Intel386 SX		25 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max			
	Operating Frequency	2	33	2	25	MHz		Half CLK2 Frequency
t_1	CLK2 Period	15	250	20	250	ns	7.3	
t_{2a}	CLK2 HIGH Time	6.25		7		ns	7.3	at 2V(3)
t_{2b}	CLK2 HIGH Time	4.0		4		ns	7.3	at (V_{CC}−0.8)V(3); Note 3
t_{3a}	CLK2 LOW Time	6.25		7		ns	7.3	at 2V(3)
t_{3b}	CLK2 LOW Time	4.5		5		ns	7.3	at 0.8V(3)
t_4	CLK2 Fall Time		4		7	ns	7.3	(V_{CC}−0.8)V to 0.8V(3)
t_5	CLK2 Rise Time		4		7	ns	7.3	0.8V to (V_{CC}−0.8)V(3)
t_6	A_{23}–A_1 Valid Delay	4	15	4	17	ns	7.5	C_L = 50 pF(4)
t_7	A_{23}–A_1 Float Delay	4	20	4	30	ns	7.6	(Note 1)
t_8	BHE#, BLE#, LOCK# Valid Delay	4	15	4	17	ns	7.5	C_L = 50 pF(4)
t_9	BHE#, BLE#, LOCK# Float Delay	4	20	4	30	ns	7.6	(Note 1)

Functional operating range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 100°C

Table 7.6. Low Power (LP) Intel386™ SX Microprocessor A.C. Characteristics—33 MHz and 25 MHz (Continued)

Symbol	Parameter	33 MHz Intel386 SX		25 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max			
t_{10}	W/R#, M/IO#, D/C#, ADS# Valid Delay	4	15	4	17	ns	7.5	C_L = 50 pF(4)
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	4	20	4	30	ns	7.6	(Note 1)
t_{12}	D_{15}–D_0 Write Data Valid Delay	7	23	7	23	ns	7.5	C_L = 50 pF(4, 5)
t_{12a}	D_{15}–D_0 Write Data Hold Time	2		2		ns		C_L = 50 pF(4)
t_{13}	D_{15}–D_0 Write Data Float Delay	4	17	4	22	ns	7.6	(Note 1)
t_{14}	HLDA Valid Delay	4	20	4	22	ns	7.6	C_L = 50 pF(4)
t_{15}	NA# Setup Time	5		5		ns	7.4	
t_{16}	NA# Hold Time	2		3		ns	7.4	
t_{19}	READY# Setup Time	7		9		ns	7.4	
t_{20}	READY# Hold Time	4		4		ns	7.4	
t_{21}	D_{15}–D_0 Read Data Setup Time	5		7		ns	7.4	
t_{22}	D_{15}–D_0 Read Data Hold Time	3		5		ns	7.4	
t_{23}	HOLD Setup Time	9		9		ns	7.4	
t_{24}	HOLD Hold Time	2		3		ns	7.4	
t_{25}	RESET Setup Time	5		8		ns	7.7	
t_{26}	RESET Hold Time	2		3		ns	7.7	
t_{27}	NMI, INTR Setup Time	5		6		ns	7.4	(Note 2)
t_{28}	NMI, INTR Hold Time	5		6		ns	7.4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY#, FLT# Setup Time	5		6		ns	7.4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY#, FLT# Hold Time	4		5		ns	7.4	(Note 2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with CL set at 50 pF. See Figures 7 and 8 for load capacitance derating curve.
5. Minimum time not 100% tested.

Functional operating range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 100°C

Table 7.7. Intel386™ SX A.C. Characteristics—20 MHz and 16 MHz

Symbol	Parameter	20 MHz Intel386 SX		16 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max			
	Operating Frequency	4	20	4	16	MHz		Half CLK2 Frequency
t_1	CLK2 Period	25	125	31	125	ns	7.3	
t_{2a}	CLK2 HIGH Time	8		9		ns	7.3	at 2V(3)
t_{2b}	CLK2 HIGH Time	5		5		ns	7.3	at (V_{CC}−0.8)V(3)
t_{3a}	CLK2 LOW Time	8		9		ns	7.3	at 2V(3)
t_{3b}	CLK2 LOW Time	6		7		ns	7.3	at 0.8V(3)
t_4	CLK2 Fall Time		8		8	ns	7.3	(V_{CC}−0.8)V to 0.8V(3)
t_5	CLK2 Rise Time		8		8	ns	7.3	0.8V to (V_{CC}−0.8)V(3)
t_6	A_{23}–A_1 Valid Delay	4	30	4	36	ns	7.5	C_L = 120 pF(4)
t_7	A_{23}–A_1 Float Delay	4	32	4	40	ns	7.6	(Note 1)
t_8	BHE#, BLE#, LOCK# Valid Delay	4	30	4	36	ns	7.5	C_L = 75 pF(4)
t_9	BHE#, BLE#, LOCK# Float Delay	4	32	4	40	ns	7.6	(Note 1)
t_{10a}	M/IO# D/C# Valid Delay	6	28	6	33	ns	7.5	C_L = 75 pF(4)
t_{10b}	W/R#, ADS# Valid Delay		26					
t_{11}	W/R#, M/IO#, D/C#, ADS# Float Delay	6	30	6	35	ns	7.6	(Note 1)
t_{12}	D_{15}–D_0 Write Data Valid Delay	4	38	4	40	ns	7.5	C_L = 120 pF(4)
t_{13}	D_{15}–D_0 Write Data Float Delay	4	27	4	35	ns	7.6	(Note 1)
t_{14}	HLDA Valid Delay	4	28	4	33	ns	7.5	C_L = 75 pF(4)
t_{15}	NA# Setup Time	5		5		ns	7.4	
t_{16}	NA# Hold Time	12		21		ns	7.4	
t_{19}	READY# Setup Time	12		19		ns	7.4	
t_{20}	READY# Hold Time	4		4		ns	7.4	
t_{21}	D_{15}–D_0 Read Data Setup Time	9		9		ns	7.4	
t_{22}	D_{15}–D_0 Read Data Hold Time	6		6		ns	7.4	
t_{23}	HOLD Setup Time	17		26		ns	7.4	
t_{24}	HOLD Hold Time	5		5		ns	7.4	
t_{25}	RESET Setup Time	12		13		ns	7.7	
t_{26}	RESET Hold Time	4		4		ns	7.7	

Functional operating range: V_{CC} = 5V ±10%; T_{CASE} = 0°C to 100°C

Table 7.7. Intel386™ SX A.C. Characteristics—20 MHz and 16 MHz (Continued)

Symbol	Parameter	20 MHz Intel386 SX		16 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max			
t_{27}	NMI, INTR Setup Time	16		16		ns	7.4	(Note 2)
t_{28}	NMI, INTR Hold Time	16		16		ns	7.4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY#, FLT# Setup Time	14		16		ns	7.4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY#, FLT# Hold Time	5		5		ns	7.4	(Note 2)

Table 7.8. Low Power (LP) Intel386™ SX A.C. Characteristics—20 MHz, 16 MHz and 12 MHz

Symbol	Parameter	20 MHz Intel386 SX		16 MHz Intel386 SX		12 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max	Min	Max			
	Operating Frequency	2	20	2	16	2	12.5	MHz		Half CLK2 Frequency
t_1	CLK2 Period	25	250	31	250	40	250	ns	7.3	
t_{2a}	CLK2 HIGH Time	8		9		11		ns	7.3	at 2V (Note 3)
t_{2b}	CLK2 HIGH Time	5		5		7		ns	7.3	at (V_{CC} − 0.8V)(3)
t_{3a}	CLK2 LOW Time	8		9		11		ns	7.3	at 2V(3)
t_{3b}	CLK2 LOW Time	6		7		9		ns	7.3	at 0.8V(3)
t_4	CLK2 Fall Time		8		8		8	ns	7.3	(V_{CC} − 0.8V) to 0.8V(3)
t_5	CLK2 Rise Time		8		8		8	ns	7.3	0.8V to (V_{CC} − 0.8V)(3)
t_6	A_{23}–A_1 Valid Delay	4	30	4	36	4	42	ns	7.5	C_L = 120 pF(4)
t_7	A_{23}–A_1 Float Delay	4	32	4	40	4	45	ns	7.6	(Note 1)
t_8	BHE#, BLE#, LOCK# Valid Delay	4	30	4	36	4	36	ns	7.5	C_L = 75 pF
t_9	BHE#, BLE#, LOCK# Float Delay	4	32	4	40	4	40	ns	7.6	(Note 1)
t_{10}	M/IO#, D/C#, W/R#, ADS# Valid Delay	6	28	6	33	4	33	ns	7.5	C_L = 75 pF
t_{11}	M/IO#, D/C#, W/R#, ADS# Float Delay	6	30	6	35	4	35	ns	7.6	(Note 1)
t_{12}	D15–D0 Write Data Valid Delay	4	38	4	40	4	50	ns	7.5	C_L = 120 pF(4)
t_{13}	D15–D0 Write Data Float Delay	4	27	4	35	4	40	ns	7.6	(Note 1)
t_{14}	HLDA Valid Delay	4	28	6	33	4	33	ns	7.5	C_L = 75 pF(4)
t_{15}	NA# Setup Time	5		5		7		ns	7.4	
t_{16}	NA# Hold Time	12		21		21		ns	7.4	

Functional operating range: V_{CC} = 5V ± 10%; T_{CASE} = 0°C to 100°C

Table 7.8. Low Power (LP) Intel386™ SX A.C. Characteristics—20 MHz, 16 MHz and 12 MHz (Continued)

Symbol	Parameter	20 MHz Intel386 SX		16 MHz Intel386 SX		12 MHz Intel386 SX		Unit	Figure	Notes
		Min	Max	Min	Max	Min	Max			
t_{19}	READY# Setup Time	12		19		19		ns	7.4	
t_{20}	READY# Hold Time	4		4		4		ns	7.4	
t_{21}	D15–D0 Read Data Setup Time	9		9		9		ns	7.4	
t_{22}	D15–D0 Read Data Hold Time	6		6		6		ns	7.4	
t_{23}	HOLD Setup Time	17		26		26		ns	7.4	
t_{24}	HOLD Hold Time	5		5		7		ns	7.4	
t_{25}	RESET Setup Time	12		13		15		ns	7.7	
t_{26}	RESET Hold Time	4		4		6		ns	7.7	
t_{27}	NMI, INTR Setup Time	16		16		16		ns	7.4	(Note 2)
t_{28}	NMI, INTR Hold Time	16		16		16		ns	7.4	(Note 2)
t_{29}	PEREQ, ERROR#, BUSY#, FLT# Setup Time	14		16		16		ns	7.4	(Note 2)
t_{30}	PEREQ, ERROR#, BUSY#, FLT# Hold Time	5		5		5		ns	7.4	(Note 2)

NOTES:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2: These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3: These are not tested. They are guaranteed by design characterization.
4: Tested with C_L set at 50 pf and derated to support the indicated distributed capacitive load. See Figures 7.8 though 7.10 for the capacitive derating curve.

A.C. TEST LOADS

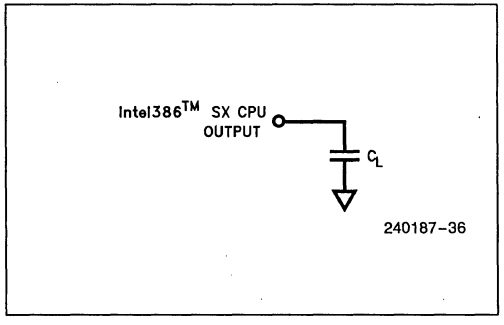


Figure 7.2. A.C. Test Loads

A.C. TIMING WAVEFORMS

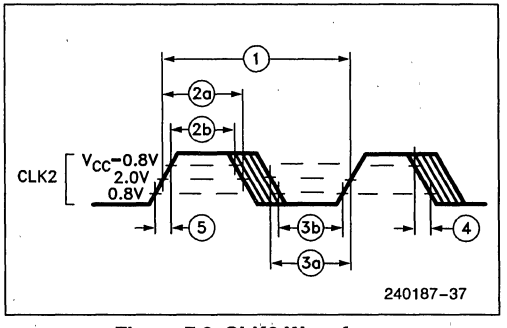


Figure 7.3. CLK2 Waveform

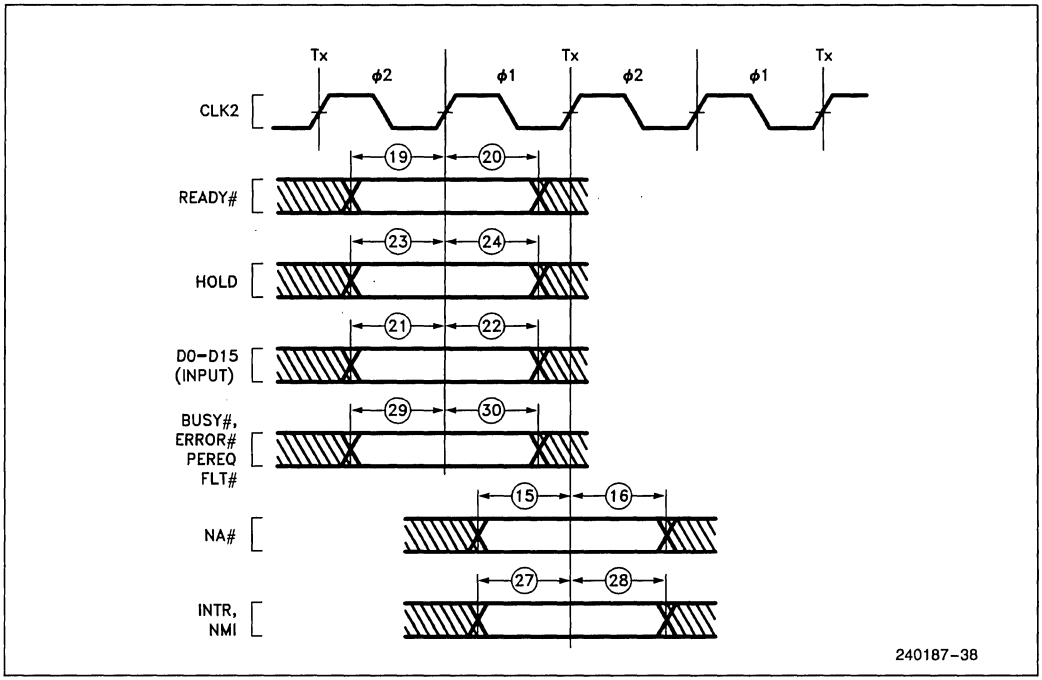


Figure 7.4. A.C. Timing Waveforms—Input Setup and Hold Timing

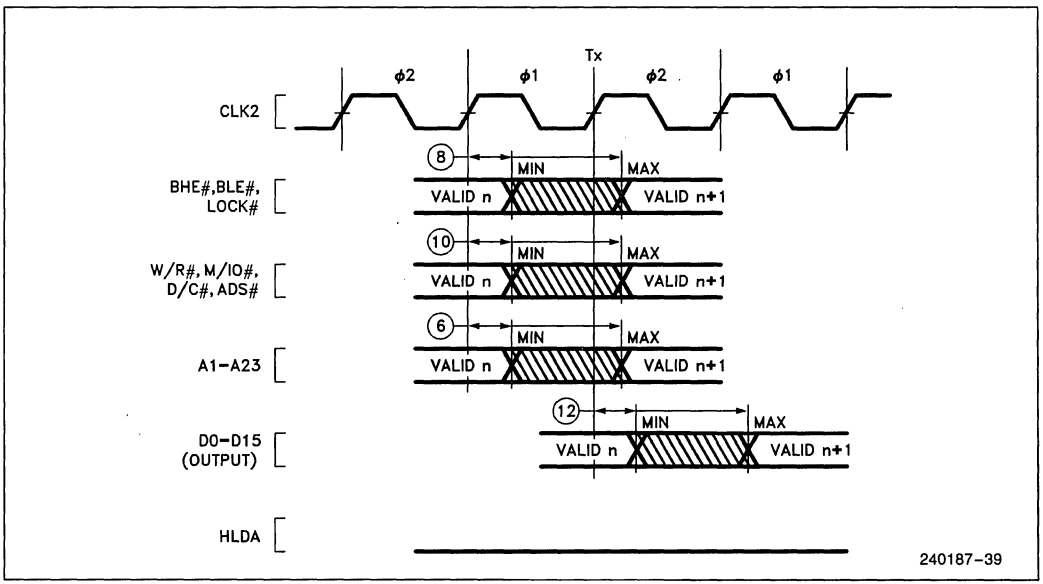


Figure 7.5. A.C. Timing Waveforms—Output Valid Delay Timing

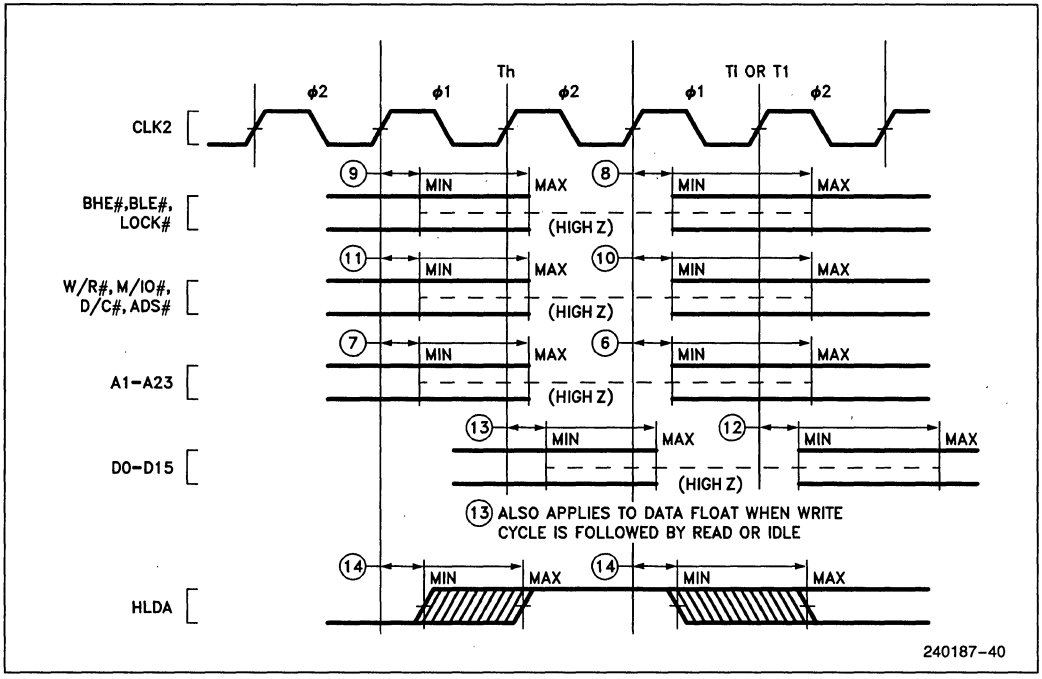


Figure 7.6. A.C. Timing Waveforms—Output Float Delay and HLDA Valid Delay Timing

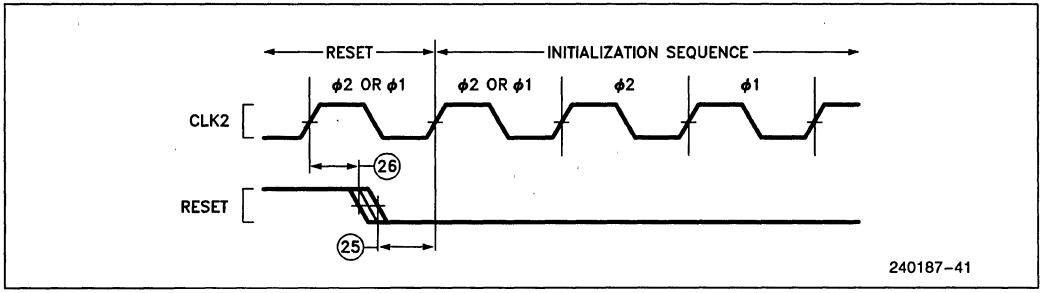


Figure 7.7. A.C. Timing Waveforms—RESET Setup and Hold Timing and Internal Phase

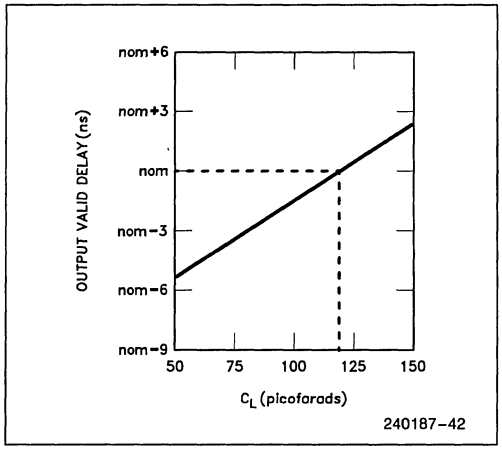


Figure 7.8. Typical Output Valid Delay versus Load Capacitance at Maximum Operating Temperature (C_L = 120 pF)

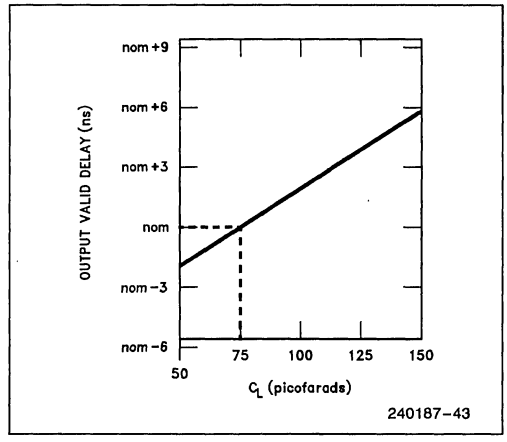


Figure 7.9. Typical Output Valid Delay versus Load Capacitance at Maximum Operating Temperature (C_L = 75 pF)

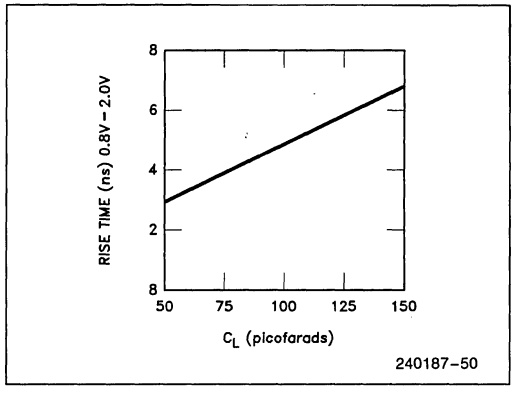


Figure 7.10. Typical Output Rise Time versus Load Capacitance at Maximum Operating Temperature

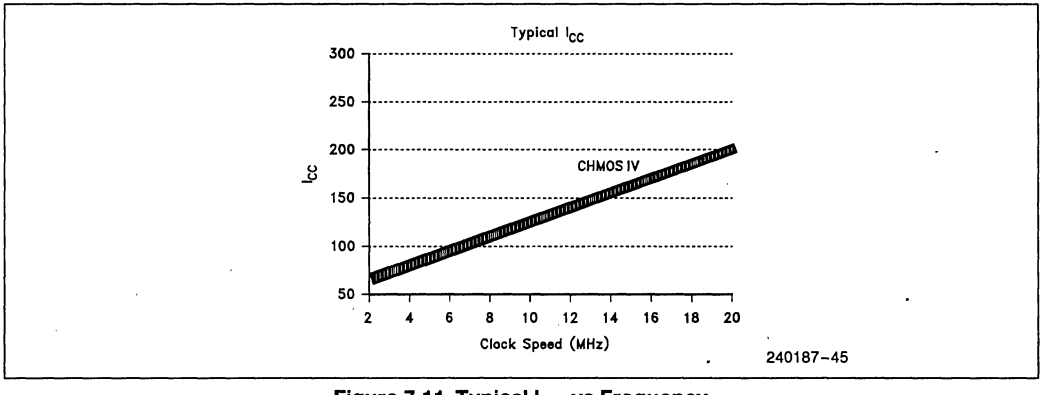


Figure 7.11. Typical I_{CC} vs Frequency

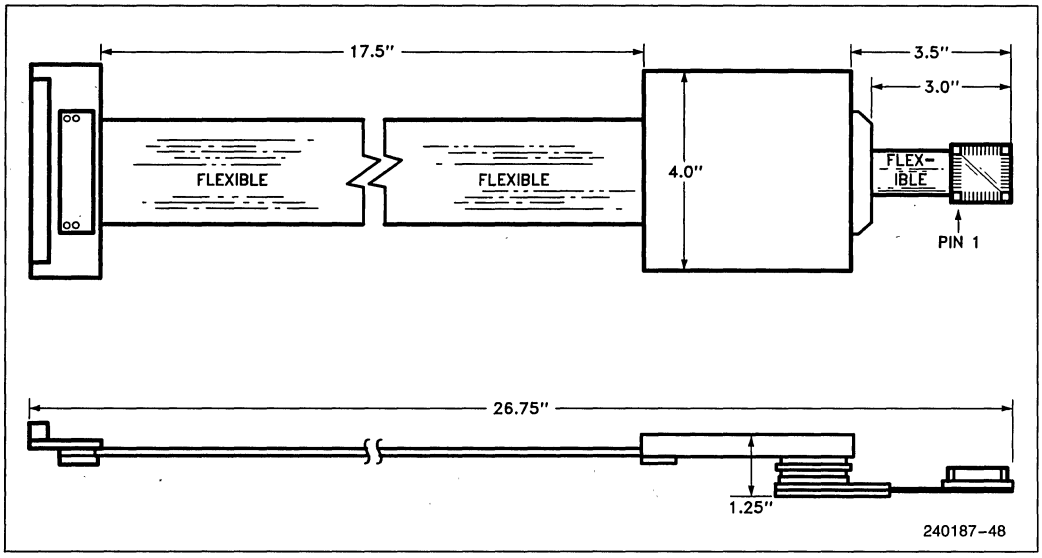


Figure 7.12. Preliminary ICE™-Intel386™ SX Emulator User Cable with PQFP Adapter

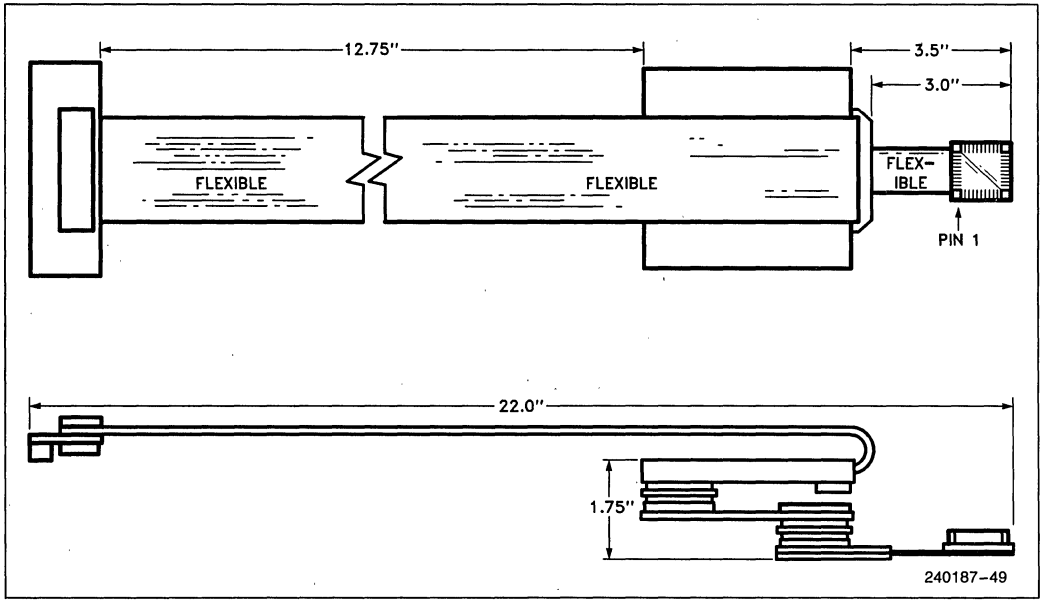


Figure 7.13. Preliminary ICE™-Intel386™ SX Emulator User Cable with OIB and PQFP Adapter

7.5 Designing for the ICE™-Intel386™ SX Emulator

ICE-Intel386 SX is the in-circuit emulator for the Intel386™ SX CPU. The ICE-386 SX emulator provides a 100-pin fine pitch flat-pack probe for connection to a socket located on the target system. Sockets that accept this probe are available from 3M (part #2-0100-07243-000) or from AMP (part #821959-1 and part #821949-4). The ICE-386 SX emulator probe attaches to the target system via an adapter that replaces the Intel386 SX component in the target system.

Due to the high operating frequency of Intel386 SX CPU based systems, there is no buffering between the Intel386 SX emulation processor (on the emulator probe) and the target system. A direct result of the non-buffered interconnect is that the ICE-Intel386 SX emulator shares the address and data busses with the target system.

In order to avoid problems with the shared bus and maintain signal integrity, the system designer must adhere to the following guidelines:

1. The bus controller must only enable data transceivers onto the data bus during valid read cycles (initiated by assertion of ADS#) of the Intel386 SX CPU, other local devices or other bus masters.
2. Before another bus master drives the local processor address bus, the other master must gain control of the address bus by asserting HOLD and receiving the HLDA response.
3. The emulation processor receives the RESET signal 2 or 4 CLK2 cycles later than an Intel386 SX CPU would, and responds to RESET later. Correct phase of the response is guaranteed.

In order to avoid problems that might arise due to the shared busses, an Optional use Isolation Board (OIB) is included with the emulator hardware. The OIB may be used to provide buffering between the emulation processor and the target system, but inserts a delay of approximately 10 ns in signal path.

In addition to the above considerations, the ICE-386 SX emulator processor module has several electrical and mechanical characteristics that should be taken into consideration when designing the Intel386 SX CPU system.

Capacitive Loading: ICE-Intel386 SX adds up to 27 pF to each Intel386 SX CPU signal.

Drive Requirements: ICE-Intel386 SX adds one FAST TTL load on the CLK2, control, address, and data lines. These loads are within the processor module and are driven by the Intel386 SX CPU emulation processor, which has standard drive and loading capability listed in Tables 7.3 and 7.4.

Power Requirements: For noise immunity and CMOS latch-up protection the ICE-Intel386 SX emulator processor module is powered by the user system.

The circuitry on the processor module draws up to 1.4A including the maximum Intel386 SX CPU I_{CC} from the user Intel386 SX CPU socket.

Intel386 SX CPU Location and Orientation: The ICE-Intel386 SX emulator processor module may require lateral clearance. Figure 7.12 shows the clearance requirements of the iMP adapter. The optional isolation board (OIB), which provides extra electrical buffering and has the same lateral clearance requirements as Figure 7.12, adds an additional 0.5 inches to the vertical clearance requirement. This is illustrated in Figure 7.13.

Optional Isolation Board (OIB) and the CLK2 speed reduction: Due to the unbuffered probe design, the ICE-Intel386 SX emulator is susceptible to errors on the user's bus. The OIB allows the ICE-Intel386 SX emulator to function in user systems with faults (shorted signals, etc.). After electrical verification the OIB may be removed. When the OIB is installed, the user system must have a maximum CLK2 frequency of 20 MHz.

8.0 DIFFERENCES BETWEEN THE Intel386™ SX CPU AND THE Intel386™ DX CPU

3

The following are the major differences between the Intel386 SX CPU and the Intel386 DX CPU:

1. The Intel386 SX CPU generates byte selects on BHE# and BLE# (like the 8086 and 80286) to distinguish the upper and lower bytes on its 16-bit data bus. The Intel386 DX CPU uses four byte selects, BE0#-BE3#, to distinguish between the different bytes on its 32-bit bus.
2. The Intel386 SX CPU has no bus sizing option. The Intel386 DX CPU can select between either a 32-bit bus or a 16-bit bus by use of the BS16# input. The Intel386 SX CPU has a 16-bit bus size.
3. The NA# pin operation in the Intel386 SX CPU is identical to that of the NA# pin on the Intel386 DX CPU with one exception: the Intel386 DX CPU NA# pin cannot be activated on 16-bit bus cycles (where BS16# is LOW in the Intel386 DX CPU case), whereas NA# can be activated on any Intel386 SX CPU bus cycle.
4. The contents of all Intel386 SX CPU registers at reset are identical to the contents of the Intel386 DX CPU registers at reset, except the DX register. The DX register contains a component-stepping identifier at reset, i.e.

   in Intel386 DX CPU, DH = 3 indicates Intel386 DX CPU after reset
   DL = revision number;

   in Intel386 SX CPU, DH = 23H indicates Intel386 SX CPU after reset
   DL = revision number.

5. The Intel386 DX CPU uses A_{31} and M/IO# as selects for the numerics coprocessor. The Intel386 SX CPU uses A_{23} and M/IO# as selects.
6. The Intel386 DX CPU prefetch unit fetches code in four-byte units. The Intel386 SX CPU prefetch unit reads two bytes as one unit (like the 80286). In BS16 mode, the Intel386 DX CPU takes two consecutive bus cycles to complete a prefetch request. If there is a data read or write request after the prefetch starts, the Intel386 DX CPU will fetch all four bytes before addressing the new request.
7. Both Intel386 DX CPU and Intel386 SX CPU have the same logical address space. The only difference is that the Intel386 DX CPU has a 32-bit physical address space and the Intel386 SX CPU has a 24-bit physical address space. The Intel386 SX CPU has a physical memory address space of up to 16 megabytes instead of the 4 gigabytes available to the Intel386 DX CPU. Therefore, in Intel386 SX CPU systems, the operating system must be aware of this physical memory limit and should allocate memory for applications programs within this limit. If a Intel386 DX CPU system uses only the lower 16 megabytes of physical address, then there will be no extra effort required to migrate Intel386 DX CPU software to the Intel386 SX CPU. Any application which uses more than 16 megabytes of memory can run on the Intel386 SX CPU if the operating system utilizes the Intel386 SX CPU's paging mechanism. In spite of this difference in physical address space, the Intel386 SX CPU and Intel386 DX CPU can run the same operating systems and applications within their respective physical memory constraints.
8. The Intel386 SX has an input called FLT# which tri-states all bidirectional and output pins, including HLDA#, when asserted. It is used with ON Circuit Emulation (ONCE). In the Intel386 DX CPU, FLT# is found only on the plastic quad flat package version and not on the ceramic pin grid array version. For a more detailed explanation of FLT# and testability, please refer to section 5.4.

9.0 INSTRUCTION SET

This section describes the instruction set. Table 9.1 lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within instructions.

9.1 Intel386™ SX CPU Instruction Encoding and Clock Count Summary

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 9.1 below, by the processor clock period (e.g. 62.5 ns for an Intel386 SX Microprocessor operating at 16 MHz). The actual clock count of an Intel386 SX Microprocessor program will average 5% more than the calculated clock count due to instruction sequences which execute faster than they can be fetched from memory.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor access to the bus.
4. No exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts shown. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.

Instruction Clock Count Notation

1. If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.
2. n = number of times repeated.
3. m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and all other bytes of the instruction and prefix(es) each count as one component.

Misaligned or 32-Bit Operand Accesses

— If instructions accesses a misaligned 16-bit operand or 32-bit operand on even address add:
  2* clocks for read or write
  4** clocks for read and write

— If instructions accesses a 32-bit operand on odd address add:
  4* clocks for read or write
  8** clocks for read and write

Wait States

Wait states add 1 clock per wait state to instruction execution for each data access.

Table 9-1. Instruction Set Clock Count Summary

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
GENERAL DATA TRANSFER					
MOV = Move:					
Register to Register/Memory	1000100w | mod reg r/m	2/2	2/2*	b	h
Register/Memory to Register	1000101w | mod reg r/m	2/4	2/4*	b	h
Immediate to Register/Memory	1100011w | mod 000 r/m | immediate data	2/2	2/2*	b	h
Immediate to Register (short form)	1011 w reg | immediate data	2	2		
Memory to Accumulator (short form)	1010000w | full displacement	4*	4*	b	h
Accumulator to Memory (short form)	1010001w | full displacement	2*	2*	b	h
Register Memory to Segment Register	10001110 | mod sreg3 r/m	2/5	22/23	b	h, i, j
Segment Register to Register/Memory	10001100 | mod sreg3 r/m	2/2	2/2	b	h
MOVSX = Move With Sign Extension					
Register From Register/Memory	00001111 | 1011111w | mod reg r/m	3/6*	3/6*	b	h
MOVZX = Move With Zero Extension					
Register From Register/Memory	00001111 | 1011011w | mod reg r/m	3/6*	3/6*	b	h
PUSH = Push:					
Register/Memory	11111111 | mod 110 r/m	5/7*	7/9*	b	h
Register (short form)	01010 reg	2	4	b	h
Segment Register (ES, CS, SS or DS) (short form)	000 sreg2 110	2	4	b	h
Segment Register (ES, CS, SS, DS, FS or GS)	00001111 | 10 sreg3 000	2	4	b	h
Immediate	011010s0 | immediate data	2	4	b	h
PUSHA = Push All	01100000	18	34	b	h
POP = Pop					
Register/Memory	10001111 | mod 000 r/m	5/7	7/9	b	h
Register (short form)	01011 reg	6	6	b	h
Segment Register (ES, CS, SS or DS) (short form)	000 sreg 2 111	7	25	b	h, i, j
Segment Register (ES, CS, SS or DS), FS or GS	00001111 | 10 sreg 3 001	7	25	b	h, i, j
POPA = Pop All	01100001	24	40	b	h
XCHG = Exchange					
Register/Memory With Register	1000011w | mod reg r/m	3/5**	3/5**	b, f	f, h
Register With Accumulator (short form)	10010 reg	3	3		
IN = Input from:		**Clk Count Virtual 8086 Mode**			
Fixed Port	1110010w | port number	†26	12*	6*/26*	s/t,m
Variable Port	1110110w	†27	13*	7*/27*	s/t,m
OUT = Output to:					
Fixed Port	1110011w | port number	†24	10*	4*/24*	s/t,m
Variable Port	1110111w	†25	11*	5*/25*	s/t,m
LEA = Load EA to Register	10001101 | mod reg r/m		2	2		

3

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
SEGMENT CONTROL					
LDS = Load Pointer to DS	11000101 | mod reg r/m	7*	26*/28*	b	h, i, j
LES = Load Pointer to ES	11000100 | mod reg r/m	7*	26*/28*	b	h, i, j
LFS = Load Pointer to FS	00001111 | 10110100 | mod reg r/m	7*	29*/31*	b	h, i, j
LGS = Load Pointer to GS	00001111 | 10110101 | mod reg r/m	7*	26*/28*	b	h, i, j
LSS = Load Pointer to SS	00001111 | 10110010 | mod reg r/m	7*	26*/28*	b	h, i, j
FLAG CONTROL					
CLC = Clear Carry Flag	11111000	2	2		
CLD = Clear Direction Flag	11111100	2	2		
CLI = Clear Interrupt Enable Flag	11111010	8	8		m
CLTS = Clear Task Switched Flag	00001111 | 00000110	5	5	c	l
CMC = Complement Carry Flag	11110101	2	2		
LAHF = Load AH into Flag	10011111	2	2		
POPF = Pop Flags	10011101	5	5	b	h, n
PUSHF = Push Flags	10011100	4	4	b	h
SAHF = Store AH into Flags	10011110	3	3		
STC = Set Carry Flag	11111001	2	2		
STD = Set Direction Flag	11111101				
STI = Set Interrupt Enable Flag	11111011	8	8		m
ARITHMETIC					
ADD = Add					
Register to Register	000000dw | mod reg r/m	2	2		
Register to Memory	0000000w | mod reg r/m	7**	7**	b	h
Memory to Register	0000001w | mod reg r/m	6*	6*	b	h
Immediate to Register/Memory	100000sw | mod000 r/m | immediate data	2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0000010w | immediate data	2	2		
ADC = Add With Carry					
Register to Register	000100dw | mod reg r/m	2	2		
Register to Memory	0001000w | mod reg r/m	7**	7**	b	h
Memory to Register	0001001w | mod reg r/m	6*	6*	b	h
Immediate to Register/Memory	100000sw | mod010 r/m | immediate data	2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0001010w | immediate data	2	2		
INC = Increment					
Register/Memory	1111111w | mod000 r/m	2/6**	2/6**	b	h
Register (short form)	01000 reg	2	2		
SUB = Subtract					
Register from Register	001010dw | mod reg r/m	2	2		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
Register from Memory	0010100w | mod reg r/m	7**	7**	b	h
Memory from Register	0010101w | mod reg r/m	6*	6*	b	h
Immediate from Register/Memory	100000sw | mod101 r/m | immediate data	2/7**	2/7**	b	h
Immediate from Accumulator (short form)	0010110w | immediate data	2	2		
SBB = Subtract with Borrow					
Register from Register	000110dw | mod reg r/m	2	2		
Register from Memory	0001100w | mod reg r/m	7**	7**	b	h
Memory from Register	0001101w | mod reg r/m	6*	6*	b	h
Immediate from Register/Memory	100000sw | mod011 r/m | immediate data	2/7**	2/7**	b	h
Immediate from Accumulator (short form)	0001110w | immediate data	2	2		
DEC = Decrement					
Register/Memory	1111111w | reg001 r/m	2/6	2/6	b	h
Register (short form)	01001 reg	2	2		
CMP = Compare					
Register with Register	001110dw | mod reg r/m	2	2		
Memory with Register	0011100w | mod reg r/m	5*	5*	b	h
Register with Memory	0011101w | mod reg r/m	6*	6*	b	h
Immediate with Register/Memory	100000sw | mod111 r/m | immediate data	2/5*	2/5*	b	h
Immediate with Accumulator (short form)	0011110w | immediate data	2	2		
NEG = Change Sign	1111011w | mod011 r/m	2/6*	2/6*	b	h
AAA = ASCII Adjust for Add	00110111	4	4		
AAS = ASCII Adjust for Subtract	00111111	4	4		
DAA = Decimal Adjust for Add	00100111	4	4		
DAS = Decimal Adjust for Subtract	00101111	4	4		
MUL = Multiply (unsigned)					
Accumulator with Register/Memory	1111011w | mod100 r/m				
Multiplier-Byte		12-17/15-20*	12-17/15-20*	b, d	d, h
-Word		12-25/15-28*	12-25/15-28*	b, d	d, h
-Doubleword		12-41/17-46*	12-41/17-46*	b, d	d, h
IMUL = Integer Multiply (signed)					
Accumulator with Register/Memory	1111011w | mod101 r/m				
Multiplier-Byte		12-17/15-20*	12-17/15-20*	b, d	d, h
-Word		12-25/15-28*	12-25/15-28*	b, d	d, h
-Doubleword		12-41/17-46*	12-41/17-46*	b, d	d, h
Register with Register/Memory	00001111 | 10101111 | mod reg r/m				
Multiplier-Byte		12-17/15-20*	12-17/15-20*	b, d	d, h
-Word		12-25/15-28*	12-25/15-28*	b, d	d, h
-Doubleword		12-41/17-46*	12-41/17-46*	b, d	d, h
Register/Memory with Immediate to Register	011010s1 | mod reg r/m | immediate data				
-Word		13-26	13-26/14-27	b, d	d, h
-Doubleword		13-42	13-42/16-45	b, d	d, h

3

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
DIV = Divide (Unsigned)					
Accumulator by Register/Memory	1111011w mod110 r/m				
Divisor—Byte		14/17	14/17	b,e	e,h
—Word		22/25	22/25	b,e	e,h
—Doubleword		38/43	38/43	b,e	e,h
IDIV = Integer Divide (Signed)					
Accumulator By Register/Memory	1111011w mod111 r/m				
Divisor—Byte		19/22	19/22	b,e	e,h
—Word		27/30	27/30	b,e	e,h
—Doubleword		43/48	43/48	b,e	e,h
AAD = ASCII Adjust for Divide	11010101 00001010	19	19		
AAM = ASCII Adjust for Multiply	11010100 00001010	17	17		
CBW = Convert Byte to Word	10011000	3	3		
CWD = Convert Word to Double Word	10011001	2	2		
LOGIC					
Shift Rotate Instructions Not Through Carry **(ROL, ROR, SAL, SAR, SHL,** and **SHR)**					
Register/Memory by 1	1101000w mod TTT r/m	3/7**	3/7**	b	h
Register/Memory by CL	1101001w mod TTT r/m	3/7*	3/7*	b	h
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	3/7*	3/7*	b	h
Through Carry **(RCL** and **RCR)**					
Register/Memory by 1	1101000w mod TTT r/m	9/10*	9/10*	b	h
Register/Memory by CL	1101001w mod TTT r/m	9/10*	9/10*	b	h
Register/Memory by Immediate Count	1100000w mod TTT r/m immed 8-bit data	9/10*	9/10*	b	h
	TTT Instruction 000 ROL 001 ROR 010 RCL 011 RCR 100 SHL/SAL 101 SHR 111 SAR				
SHLD = Shift Left Double					
Register/Memory by Immediate	00001111 10100100 mod reg r/m immed 8-bit data	3/7**	3/7**		
Register/Memory by CL	00001111 10100101 mod reg r/m	3/7**	3/7**		
SHRD = Shift Right Double					
Register/Memory by Immediate	00001111 10101100 mod reg r/m immed 8-bit data	3/7**	3/7**		
Register/Memory by CL	00001111 10101101 mod reg r/m	3/7**	3/7**		
AND = And					
Register to Register	001000dw mod reg r/m	2	2		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
LOGIC (Continued)						
Register to Memory	0010000w mod reg r/m		7**	7**	b	h
Memory to Register	0010001w mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1000000w mod 100 r/m immediate data		2/7*	2/7**	b	h
Immediate to Accumulator (Short Form)	0010010w immediate data		2	2		
TEST = And Function to Flags, No Result						
Register/Memory and Register	1000010w mod reg r/m		2/5*	2/5*	b	h
Immediate Data and Register/Memory	1111011w mod 000 r/m immediate data		2/5*	2/5*	b	h
Immediate Data and Accumulator (Short Form)	1010100w immediate data		2	2		
OR = Or						
Register to Register	000010dw mod reg r/m		2	2		
Register to Memory	0000100w mod reg r/m		7**	7**	b	h
Memory to Register	0000101w mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1000000w mod 001 r/m immediate data		2/7**	2/7**	b	h
Immediate to Accumulator (Short Form)	0000110w immediate data		2	2		
XOR = Exclusive Or						
Register to Register	001100dw mod reg r/m		2	2		
Register to Memory	0011000w mod reg r/m		7**	7**	b	h
Memory to Register	0011001w mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1000000w mod 110 r/m immediate data		2/7**	2/7**	b	h
Immediate to Accumulator (Short Form)	0011010w immediate data		2	2		
NOT = Invert Register/Memory	1111011w mod 010 r/m		2/6**	2/6**	b	h
STRING MANIPULATION						
CMPS = Compare Byte Word	1010011w		10*	10*	b	h
INS = Input Byte/Word from DX Port	0110110w	†29	15	9*/29**	b	s/t, h, m
LODS = Load Byte/Word to AL/AX/EAX	1010110w		5	5*	b	h
MOVS = Move Byte Word	1010010w		7	7**	b	h
OUTS = Output Byte/Word to DX Port	0110111w	†28	14	8*/28*	b	s/t, h, m
SCAS = Scan Byte Word	1010111w		7*	7*	b	h
STOS = Store Byte/Word from AL/AX/EX	1010101w		4*	4*	b	h
XLAT = Translate String	11010111		5*	5*		h
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX						
REPE CMPS = Compare String (Find Non-Match)	11110011 1010011w		5 + 9n**	5 + 9n**	b	h

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
REPEATED STRING MANIPULATION (Continued)					
REPNE CMPS = Compare String					
(Find Match)	11110010 1010011w — Clk Count Virtual 8086 Mode	5+9n**	5+9n**	b	h
REP INS = Input String	11110010 0110110w — †	13+6n*	7+6n*/ 27+6n*	b	s/t, h, m
REP LODS = Load String	11110010 1010110w	5+6n*	5+6n*	b	h
REP MOVS = Move String	11110010 1010010w	7+4n*	7+4n**	b	h
REP OUTS = Output String	11110010 0110111w — †	12+5n*	6+5n*/ 26+5n*	b	s/t, h, m
REPE SCAS = Scan String					
(Find Non-AL/AX/EAX)	11110011 1010111w	5+8n*	5+8n*	b	h
REPNE SCAS = Scan String					
(Find AL/AX/EAX)	11110010 1010111w	5+8n*	5+8n*	b	h
REP STOS = Store String	11110010 1010101w	5+5n*	5+5n*	b	h
BIT MANIPULATION					
BSF = Scan Bit Forward	00001111 10111100 mod reg r/m	10+3n*	10+3n**	b	h
BSR = Scan Bit Reverse	00001111 10111101 mod reg r/m	10+3n*	10+3n**	b	h
BT = Test Bit					
Register/Memory, Immediate	00001111 10111010 mod 100 r/m immed 8-bit data	3/6*	3/6*	b	h
Register/Memory, Register	00001111 10100011 mod reg r/m	3/12*	3/12*	b	h
BTC = Test Bit and Complement					
Register/Memory, Immediate	00001111 10111010 mod 111 r/m immed 8-bit data	6/8*	6/8*	b	h
Register/Memory, Register	00001111 10111011 mod reg r/m	6/13*	6/13*	b	h
BTR = Test Bit and Reset					
Register/Memory, Immediate	00001111 10111010 mod 110 r/m immed 8-bit data	6/8*	6/8*	b	h
Register/Memory, Register	00001111 10110011 mod reg r/m	6/13*	6/13*	b	h
BTS = Test Bit and Set					
Register/Memory, Immediate	00001111 10111010 mod 101 r/m immed 8-bit data	6/8*	6/8*	b	h
Register/Memory, Register	00001111 10101011 mod reg r/m	6/13*	6/13*	b	h
CONTROL TRANSFER					
CALL = Call					
Direct Within Segment	11101000 full displacement	7+m*	9+m*	b	r
Register/Memory					
Indirect Within Segment	11111111 mod 010 r/m	7+m*/10+m*	9+m/ 12+m*	b	h, r
Direct Intersegment	10011010 unsigned full offset, selector	17+m*	42+m*	b	j,k,r

NOTE:
† Clock count shown applies if I/O permission allows I/O to the port in virtual 8086 mode. If I/O bit map denies permission exception 13 fault occurs; refer to clock counts for INT 3 instruction.

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			64+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			98+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			106+8x+m		h,j,k,r
From 286 Task to 286 TSS			285		h,j,k,r
From 286 Task to Intel386™ SX CPU TSS			310		h,j,k,r
From 286 Task to Virtual 8086 Task (Intel386 SX CPU TSS)			229		h,j,k,r
From Intel386 SX CPU Task to 286 TSS			285		h,j,k,r
From Intel386 SX CPU Task to Intel386 SX CPU TSS			392		h,j,k,r
From Intel386 SX CPU Task to Virtual 8086 Task (Intel386 SX CPU TSS)			309		h,j,k,r
Indirect Intersegment	11111111 mod011 r/m	30+m	46+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			68+m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			102+m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			110+8x+m		h,j,k,r
From 286 Task to 286 TSS					h,j,k,r
From 286 Task to Intel386 SX CPU TSS					h,j,k,r
From 286 Task to Virtual 8086 Task (Intel386 SX CPU TSS)					h,j,k,r
From Intel386 SX CPU Task to 286 TSS					h,j,k,r
From Intel386 SX CPU Task to Intel386 SX CPU TSS			399		h,j,k,r
From Intel386 SX CPU Task to Virtual 8086 Task (Intel386 SX CPU TSS)					h,j,k,r
JMP = Unconditional Jump					
Short	11101011 8-bit displacement	7+m	7+m		r
Direct within Segment	11101001 full displacement	7+m	7+m		r
Register/Memory Indirect within Segment	11111111 mod100 r/m	9+m/14+m	9+m/14+m	b	h,r
Direct Intersegment	11101010 unsigned full offset, selector	16+m	31+m		j,k,r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			53+m		h,j,k,r
From 286 Task to 286 TSS					h,j,k,r
From 286 Task to Intel386 SX CPU TSS					h,j,k,r
From 286 Task to Virtual 8086 Task (Intel386 SX CPU TSS)					h,j,k,r
From Intel386 SX CPU Task to 286 TSS					h,j,k,r
From Intel386 SX CPU Task to Intel386 SX CPU TSS					h,j,k,r
From Intel386 SX CPU Task to Virtual 8086 Task (Intel386 SX CPU TSS)			395		h,j,k,r
Indirect Intersegment	11111111 mod101 r/m	17+m	31+m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			49+m		h,j,k,r
From 286 Task to 286 TSS					h,j,k,r
From 286 Task to Intel386 SX CPU TSS					h,j,k,r
From 286 Task to Virtual 8086 Task (Intel386 SX CPU TSS)					h,j,k,r
From Intel386 SX CPU Task to 286 TSS					h,j,k,r
From Intel386 SX CPU Task to Intel386 SX CPU TSS			328		h,j,k,r
From Intel386 SX CPU Task to Virtual 8086 Task (Intel386 SX CPU TSS)					h,j,k,r

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
RET = Return from CALL:					
Within Segment	11000011		12+m	b	g, h, r
Within Segment Adding Immediate to SP	11000010 | 16-bit displ		12+m	b	g, h, r
Intersegment	11001011		36+m	b	g, h, j, k, r
Intersegment Adding Immediate to SP	11001010 | 16-bit displ		36+m	b	g, h, j, k, r
Protected Mode Only (RET):					
to Different Privilege Level					
Intersegment			72		h, j, k, r
Intersegment Adding Immediate to SP			72		h, j, k, r
CONDITIONAL JUMPS					
NOTE: Times Are Jump "Taken or Not Taken"					
JO = Jump on Overflow					
8-Bit Displacement	01110000 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000000 | full displacement	7+m or 3	7+m or 3		r
JNO = Jump on Not Overflow					
8-Bit Displacement	01110001 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000001 | full displacement	7+m or 3	7+m or 3		r
JB/JNAE = Jump on Below/Not Above or Equal					
8-Bit Displacement	01110010 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000010 | full displacement	7+m or 3	7+m or 3		r
JNB/JAE = Jump on Not Below/Above or Equal					
8-Bit Displacement	01110011 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000011 | full displacement	7+m or 3	7+m or 3		r
JE/JZ = Jump on Equal/Zero					
8-Bit Displacement	01110100 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000100 | full displacement	7+m or 3	7+m or 3		r
JNE/JNZ = Jump on Not Equal/Not Zero					
8-Bit Displacement	01110101 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000101 | full displacement	7+m or 3	7+m or 3		r
JBE/JNA = Jump on Below or Equal/Not Above					
8-Bit Displacement	01110110 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000110 | full displacement	7+m or 3	7+m or 3		r
JNBE/JA = Jump on Not Below or Equal/Above					
8-Bit Displacement	01110111 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10000111 | full displacement	7+m or 3	7+m or 3		r
JS = Jump on Sign					
8-Bit Displacement	01111000 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001000 | full displacement	7+m or 3	7+m or 3		r

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL JUMPS (Continued)					
JNS = Jump on Not Sign					
8-Bit Displacement	01111001 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001001 | full displacement	7+m or 3	7+m or 3		r
JP/JPE = Jump on Parity/Parity Even					
8-Bit Displacement	01111010 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001010 | full displacement	7+m or 3	7+m or 3		r
JNP/JPO = Jump on Not Parity/Parity Odd					
8-Bit Displacement	01111011 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001011 | full displacement	7+m or 3	7+m or 3		r
JL/JNGE = Jump on Less/Not Greater or Equal					
8-Bit Displacement	01111100 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001100 | full displacement	7+m or 3	7+m or 3		r
JNL/JGE = Jump on Not Less/Greater or Equal					
8-Bit Displacement	01111101 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001101 | full displacement	7+m or 3	7+m or 3		r
JLE/JNG = Jump on Less or Equal/Not Greater					
8-Bit Displacement	01111110 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001110 | full displacement	7+m or 3	7+m or 3		r
JNLE/JG = Jump on Not Less or Equal/Greater					
8-Bit Displacement	01111111 | 8-bit displ	7+m or 3	7+m or 3		r
Full Displacement	00001111 | 10001111 | full displacement	7+m or 3	7+m or 3		r
JCXZ = Jump on CX Zero	11100011 | 8-bit displ	9+m or 5	9+m or 5		r
JECXZ = Jump on ECX Zero	11100011 | 8-bit displ	9+m or 5	9+m or 5		r
(Address Size Prefix Differentiates JCXZ from JECXZ)					
LOOP = Loop CX Times	11100010 | 8-bit displ	11+m	11+m		r
LOOPZ/LOOPE = Loop with Zero/Equal	11100001 | 8-bit displ	11+m	11+m		r
LOOPNZ/LOOPNE = Loop While Not Zero	11100000 | 8-bit displ	11+m	11+m		r
CONDITIONAL BYTE SET NOTE: Times Are Register/Memory					
SETO = Set Byte on Overflow					
To Register/Memory	00001111 | 10010000 | mod 000 r/m	4/5*	4/5*		h
SETNO = Set Byte on Not Overflow					
To Register/Memory	00001111 | 10010001 | mod 000 r/m	4/5*	4/5*		h
SETB/SETNAE = Set Byte on Below/Not Above or Equal					
To Register/Memory	00001111 | 10010010 | mod 000 r/m	4/5*	4/5*		h

3

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL BYTE SET (Continued)					
SETNB = Set Byte on Not Below/Above or Equal					
To Register/Memory	00001111 | 10010011 | mod000 r/m	4/5*	4/5*		h
SETE/SETZ = Set Byte on Equal/Zero					
To Register/Memory	00001111 | 10010100 | mod000 r/m	4/5*	4/5*		h
SETNE/SETNZ = Set Byte on Not Equal/Not Zero					
To Register/Memory	00001111 | 10010101 | mod000 r/m	4/5*	4/5*		h
SETBE/SETNA = Set Byte on Below or Equal/Not Above					
To Register/Memory	00001111 | 10010110 | mod000 r/m	4/5*	4/5*		h
SETNBE/SETA = Set Byte on Not Below or Equal/Above					
To Register/Memory	00001111 | 10010111 | mod000 r/m	4/5*	4/5*		h
SETS = Set Byte on Sign					
To Register/Memory	00001111 | 10011000 | mod000 r/m	4/5*	4/5*		h
SETNS = Set Byte on Not Sign					
To Register/Memory	00001111 | 10011001 | mod000 r/m	4/5*	4/5*		h
SETP/SETPE = Set Byte on Parity/Parity Even					
To Register/Memory	00001111 | 10011010 | mod000 r/m	4/5*	4/5*		h
SETNP/SETPO = Set Byte on Not Parity/Parity Odd					
To Register/Memory	00001111 | 10011011 | mod000 r/m	4/5*	4/5*		h
SETL/SETNGE = Set Byte on Less/Not Greater or Equal					
To Register/Memory	00001111 | 10011100 | mod000 r/m	4/5*	4/5*		h
SETNL/SETGE = Set Byte on Not Less/Greater or Equal					
To Register/Memory	00001111 | 01111101 | mod000 r/m	4/5*	4/5*		h
SETLE/SETNG = Set Byte on Less or Equal/Not Greater					
To Register/Memory	00001111 | 10011110 | mod000 r/m	4/5*	4/5*		h
SETNLE/SETG = Set Byte on Not Less or Equal/Greater					
To Register/Memory	00001111 | 10011111 | mod000 r/m	4/5*	4/5*		h
ENTER = Enter Procedure	11001000 | 16-bit displacement, 8-bit level				
L = 0		10	10	b	h
L = 1		14	14	b	h
L > 1		17 + 8(n − 1)	17 + 8(n − 1)	b	h
LEAVE = Leave Procedure	11001001	4	4	b	h

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS					
INT = Interrupt:					
Type Specified	11001101 | type	37		b	
Type 3	11001100	33		b	
INTO = Interrupt 4 if Overflow Flag Set	11001110				
If OF = 1		35		b, e	
If OF = 0		3	3	b, e	
Bound = Interrupt 5 if Detect Value Out of Range	01100010 | mod reg r/m				
If Out of Range		44		b, e	e, g, h, j, k, r
If In Range		10	10	b, e	e, g, h, j, k, r
Protected Mode Only (INT)					
INT: Type Specified					
Via Interrupt or Trap Gate					
Via Interrupt or Trap Gate					
to Same Privilege Level			71		g, j, k, r
to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			438		g, j, k, r
From 286 Task to Intel386™ SX CPU TSS via Task Gate			465		g, j, k, r
From 286 Task to virt 8086 md via Task Gate			382		g, j, k, r
From Intel386™ SX CPU Task to 286 TSS via Task Gate			440		g, j, k, r
From Intel386™ SX CPU Task to Intel386™ SX CPU TSS via Task Gate			467		g, j, k, r
From Intel386™ SX CPU Task to virt 8086 md via Task Gate			384		g, j, k, r
From virt 8086 md to 286 TSS via Task Gate			445		g, j, k, r
From virt 8086 md to Intel386™ SX CPU TSS via Task Gate			472		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			275		
INT: TYPE 3					
Via Interrupt or Trap Gate					
to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate					
to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			382		g, j, k, r
From 286 Task to Intel386™ SX CPU TSS via Task Gate			409		g, j, k, r
From 286 Task to Virt 8086 md via Task Gate			326		g, j, k, r
From Intel386™ SX CPU Task to 286 TSS via Task Gate			384		g, j, k, r
From Intel386™ SX CPU Task to Intel386™ SX CPU TSS via Task Gate			411		g, j, k, r
From Intel386™ SX CPU Task to Virt 8086 md via Task Gate			328		g, j, k, r
From virt 8086 md to 286 TSS via Task Gate			389		g, j, k, r
From virt 8086 md to Intel386™ SX CPU TSS via Task Gate			416		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			223		
INTO:					
Via Interrupt or Trap Grate					
to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate					
to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			384		g, j, k, r
From 286 Task to Intel386™ SX CPU TSS via Task Gate			411		g, j, k, r
From 286 Task to virt 8086 md via Task Gate			328		g, j, k, r
From Intel386™ SX CPU Task to 286 TSS via Task Gate			Intel386 DX		g, j, k, r
From Intel386™ SX CPU Task to Intel386™ SX CPU TSS via Task Gate			413		g, j, k, r
From Intel386™ SX CPU Task to virt 8086 md via Task Gate			329		g, j, k, r
From virt 8086 md to 286 TSS via Task Gate			391		g, j, k, r
From virt 8086 md to Intel386™ SX CPU TSS via Task Gate			418		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			223		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS (Continued)					
BOUND:					
Via Interrupt or Trap Gate to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			111		g, j, k, r
From 286 Task to 286 TSS via Task Gate			358		g, j, k, r
From 286 Task to Intel386™ SX CPU TSS via Task Gate			388		g, j, k, r
From 268 Task to virt 8086 Mode via Task Gate			335		g, j, k, r
From Intel386 SX CPU Task to 286 TSS via Task Gate			368		g, j, k, r
From Intel386 SX CPU Task to Intel386 SX CPU TSS via Task Gate			398		g, j, k, r
From Intel386 SX CPU Task to virt 8086 Mode via Task Gate			347		g, j, k, r,
From virt 8086 Mode to 286 TSS via Task Gate			368		g, j, k, r
From virt 8086 Mode to Intel386 SX CPU TSS via Task Gate			398		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			223		
INTERRUPT RETURN					
IRET = Interrupt Return	11001111	24			g, h, j, k, r
Protected Mode Only (IRET)					
To the Same Privilege Level (within task)			42		g, h, j, k, r
To Different Privilege Level (within task)			86		g, h, j, k, r
From 286 Task to 286 TSS			285		h, j, k, r
From 286 Task to Intel386 SX CPU TSS			318		h, j, k, r
From 286 Task to Virtual 8086 Task			267		h, j, k, r
From 286 Task to Virtual 8086 Mode (within task)			113		
From Intel386 SX CPU Task to 286 TSS			324		h, j, k, r
From Intel386 SX CPU Task to Intel386 SX CPU TSS			328		h, j, k, r
From Intel386 SX CPU Task to Virtual 8086 Task			377		h, j, k, r
From Intel386 SX CPU Task to Virtual 8086 Mode (within task)			113		
PROCESSOR CONTROL					
HLT = HALT	11110100	5	5		I
MOV = Move to and From Control/Debug/Test Registers					
CR0/CR2/CR3 from register	00001111 00100010 11 eee reg	10/4/5	10/4/5		I
Register From CR0–3	00001111 00100000 11 eee reg	6	6		I
DR0–3 From Register	00001111 00100011 11 eee reg	22	22		I
DR6–7 From Register	00001111 00100011 11 eee reg	16	16		I
Register from DR6–7	00001111 00100001 11 eee reg	14	14		I
Register from DR0–3	00001111 00100001 11 eee reg	22	22		I
TR6–7 from Register	00001111 00100110 11 eee reg	12	12		I
Register from TR6–7	00001111 00100100 11 eee reg	12	12		I
NOP = No Operation	10010000	3	3		
WAIT = Wait until BUSY# pin is negated	10011011	6	6		

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
PROCESSOR EXTENSION INSTRUCTIONS					
Processor Extension Escape	11011TTT | modLLL r/m TTT and LLL bits are opcode information for coprocessor.	See Intel387SX data sheet for clock counts			h
PREFIX BYTES					
Address Size Prefix	01100111	0	0		
LOCK = Bus Lock Prefix	11110000	0	0		m
Operand Size Prefix	01100110	0	0		
Segment Override Prefix					
CS:	00101110	0	0		
DS:	00111110	0	0		
ES:	00100110	0	0		
FS:	01100100	0	0		
GS:	01100101	0	0		
SS:	00110110	0	0		
PROTECTION CONTROL					
ARPL = Adjust Requested Privilege Level					
From Register/Memory	01100011 | mod reg r/m	N/A	20/21**	a	h
LAR = Load Access Rights					
From Register/Memory	00001111 | 00000010 | mod reg r/m	N/A	15/16*	a	g, h, j, p
LGDT = Load Global Descriptor					
Table Register	00001111 | 00000001 | mod010 r/m	11*	11*	b, c	h, l
LIDT = Load Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod011 r/m	11*	11*	b, c	h, l
LLDT = Load Local Descriptor					
Table Register to Register/Memory	00001111 | 00000000 | mod010 r/m	N/A	20/24*	a	g, h, j, l
LMSW = Load Machine Status Word					
From Register/Memory	00001111 | 00000001 | mod110 r/m	10/13	10/13*	b, c	h, l
LSL = Load Segment Limit					
From Register/Memory	00001111 | 00000011 | mod reg r/m				
Byte-Granular Limit		N/A	20/21*	a	g, h, j, p
Page-Granular Limit		N/A	25/26*	a	g, h, j, p
LTR = Load Task Register					
From Register/Memory	00001111 | 00000000 | mod001 r/m	N/A	23/27*	a	g, h, j, l
SGDT = Store Global Descriptor					
Table Register	00001111 | 00000001 | mod000 r/m	9*	9*	b, c	h
SIDT = Store Interrupt Descriptor					
Table Register	00001111 | 00000001 | mod001 r/m	9*	9*	b, c	h
SLDT = Store Local Descriptor Table Register					
To Register/Memory	00001111 | 00000000 | mod000 r/m	N/A	2/2*	a	h

3

Table 9-1. Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
PROTECTION CONTROL (Continued)					
SMSW = **Store Machine Status Word**	00001111 | 00000001 | mod100 r/m	2/2*	2/2*	b, c	h, l
STR = **Store Task Register**					
To Register/Memory	00001111 | 00000000 | mod001 r/m	N/A	2/2*	a	h
VERR = **Verify Read Access**					
Register/Memory	00001111 | 00000000 | mod100 r/m	N/A	10/11*	a	g, h, j, p
VERW = **Verify Write Access**	00001111 | 00000000 | mod101 r/m	N/A	15/16*	a	g, h, j, p

INSTRUCTION NOTES FOR TABLE 9-1

Notes a through c apply to Real Address Mode only:

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid opcode).

b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS or GS limit, FFFFH. Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Notes d through g apply to Real Address Mode and Protected Virtual Address Mode:

d. The Intel386 SX CPU uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).

Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:

Actual Clock = if $m < > 0$ then max ($[\log_2 |m|]$, 3) + b clocks:
if $m = 0$ then 3+b clocks

In this formula, m is the multiplier, and

b = 9 for register to register,
b = 12 for memory to register,
b = 10 for register with immediate to register,
b = 11 for memory with immediate to register.

e. An exception may occur, depending on the value of the operand.

f. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK# prefix.

g. LOCK# is asserted during descriptor table accesses.

Notes h through s/t apply to Protected Virtual Address Mode only:

h. Exception 13 fault (general protection violation) will occur if the memory operand in CS, DS, ES, FS or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, an exception 12 (stack segment limit violation or not present) occurs.

i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present) occurs.

j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.

k. JMP, CALL, INT, RET and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is violated.

l. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).

m. An exception 13 fault occurs if CPL is greater than IOPL.

n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.

o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 if desiring to reset the PE bit.

p. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.

q. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or not present) will occur if the stack limit is violated by the operand's starting address.

r. The destination of a JMP, CALL, INT, RET or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.

s/t. The instruction will execute in s clocks if CPL ≤ IOPL. If CPL > IOPL, the instruction will take t clocks.

9.2 INSTRUCTION ENCODING

9.2.1 Overview

All instruction encodings are subsets of the general instruction format shown in Figure 8-1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the "mod r/m" byte and "scaled index" byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16 or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure 9-1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the Figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table 9-2 is a complete list of all fields appearing in the instruction set. Further ahead, following Table 9-2, are detailed tables for each field.

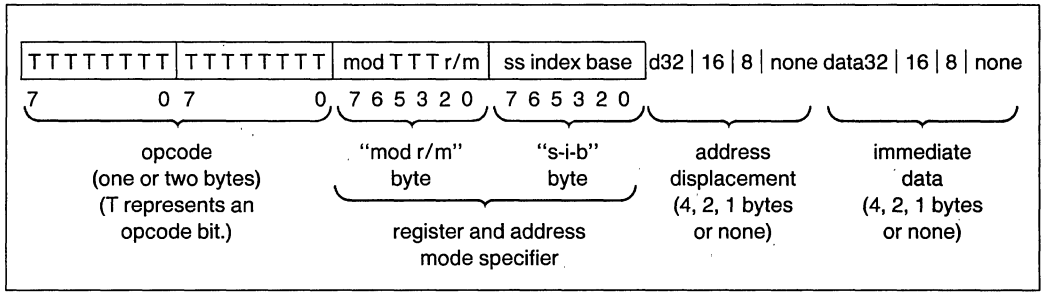


Figure 9-1. General Instruction Format

Table 9-2. Fields within Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bits	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

Note: Table 9-1 shows encoding of individual instructions.

9.2.2 32-Bit Extensions of the Instruction Set

With the Intel386 SX CPU, the 8086/80186/80286 instruction set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types, and 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

Whether the instruction defaults to operations of 16 bits or 32 bits depends on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 bits or 16 bits) for both operands and effective addresses when executing that code segment. In the Real Address Mode or Virtual 8086 Mode, no code segment descriptors are used, but a D value of 0 is assumed internally by the Intel386 SX CPU when operating in those modes (for 16-bit default sizes compatible with the 8086/80186/80286).

Two prefixes, the Operand Size Prefix and the Effective Address Size Prefix, allow overriding individually the Default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the opcode bytes. The presence of the Operand Size Prefix and the Effective Address Prefix will toggle the operand size or the effective address size, respectively, to the value "opposite" from the Default setting. For example, if the default operand size is for 32-bit data operations, then presence of the Operand Size Prefix toggles the instruction to 16-bit data operation. As another example, if the default effective address size is 16 bits, presence of the Effective Address Size prefix toggles the instruction to use 32-bit effective address computations.

These 32-bit extensions are available in all modes, including the Real Address Mode or the Virtual 8086 Mode. In these modes the default is always 16 bits, so prefixes are needed to specify 32-bit operands or addresses. For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

9.2.3 Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on. The exact encodings of these fields are defined immediately ahead.

9.2.3.1 ENCODING OF OPERAND LENGTH (w) FIELD

For any given instruction performing a data operation, the instruction is executing as a 32-bit operation or a 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

9.2.3.2 ENCODING OF THE GENERAL REGISTER (reg) FIELD

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
101	SI	ESI
101	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations:		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register Specified by reg Field During 32-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

9.2.3.3 ENCODING OF THE SEGMENT REGISTER (sreg) FIELD

The sreg field in certain instructions is a 2-bit field allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the Intel386 SX CPU FS and GS segment registers to be specified.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

9.2.3.4 ENCODING OF ADDRESS MODE

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte, can be specified.

The s-i-b byte (scale-index-base byte) is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01 or 10. When the sib byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure 8-1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following three pages define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Encoding of 16-bit Address Mode with "mod r/m" Byte

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	SS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]

mod r/m	Effective Address
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP + DI + d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by r/m During 16-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode with "mod r/m" byte (no "s-i-b" byte present):

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]

mod r/m	Effective Address
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by reg or r/m during 16-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m during 32-Bit Data Operations:		
mod r/m	function of w field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-bit Address Mode ("mod r/m" byte and "s-i-b" byte present):

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

NOTE:
Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg**
101	EBP
110	ESI
111	EDI

****IMPORTANT NOTE:**
When index field is 100, indicating "no index register," then ss field MUST equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

9.2.3.5 ENCODING OF OPERATION DIRECTION (d) FIELD

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory <- - Register "reg" Field Indicates Source Operand; "mod r/m" or "mod ss index base" Indicates Destination Operand
1	Register <- - Register/Memory "reg" Field Indicates Destination Operand; "mod r/m" or "mod ss index base" Indicates Source Operand

9.2.3.6 ENCODING OF SIGN-EXTEND (s) FIELD

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

s	Effect on Immediate Data8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extend Data8 to Fill 16-Bit or 32-Bit Destination	None

9.2.3.7 ENCODING OF CONDITIONAL TEST (tttn) FIELD

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n = 0) or its negation (n = 1), and tt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111

9.2.3.8 ENCODING OF CONTROL OR DEBUG OR TEST REGISTER (eee) FIELD

For the loading and storing of the Control, Debug and Test registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	CR2
011	CR3
Do not use any other encoding	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding	

When Interpreted as Test Register Field

eee Code	Reg Name
110	TR6
111	TR7
Do not use any other encoding	

DATA SHEET REVISION REVIEW

The following list represents key differences between this data sheet and the -007 version of the Intel386™ SX microprocessor data sheet. Please review the summary carefully.

1. Table 5.7, E-Step revision identifier is added.
2. Table 7.3, I_{CC} supply current for CLK2 = 40 MHz with 20 MHz Intel386 SX has a typical I_{CC} of 180 mA.
3. Table 7.5, t_4 CLK2 fall time and t_5 CLK2 rise time have no minimum time for all speeds but maximum time for all speeds is 8 ns.
4. Figure 7.11, CHMOS III characteristics for typical I_{CC} has been taken out.

Intel387™ SX MATH COPROCESSOR

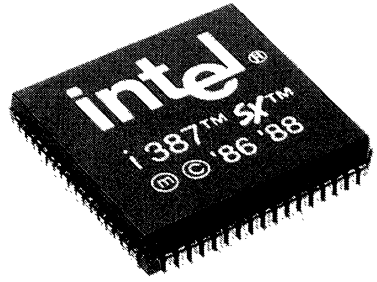
- **New Automatic Power Management**
  - **— Low Power Consumption**
  - **— Typically 100 mA in Dynamic Mode, and 4 mA in Idle Mode**
- **Socket Compatible with Intel387 Family of Math CoProcessors**
  - **— Hardware and Software Compatible**
  - **— Supported by Over 2100 Commercial Software Packages**
  - **— 10% to 15% Performance Increase on Whetstone and Livermore Benchmarks**
- **Compatible with the Intel386™ SX Microprocessor**
  - **— Extends CPU Instruction Set to Include Trigonometric, Logarithmic, and Exponential**
- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Available in a 68-Pin PLCC Package**

  See Intel Packaging Specification, Order #231369

The Intel387™ SX Math CoProcessor is an extension to the Intel386™ SX microprocessor architecture. The combination of the Intel387™ SX with the Intel386™ SX microprocessor dramatically increases the processing speed of computer application software that utilizes high performance floating-point operations. An internal Power Management Unit enables the Intel387™ SX to perform these floating-point operations while maintaining very low power consumption for portable and desktop applications. The internal Power Management Unit effectively reduces power consumption by 95% when the device is idle.

3

The Intel387™ SX Math CoProcessor is available in a 68-pin PLCC package, and is manufactured on Intel's advanced 1.0 micron CHMOS IV technology.



240225-22

Intel386 and Intel387 are trademarks of Intel Corporation.

Intel387™ SX Math CoProcessor

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

PAGE

FIGURES

CONTENTS

PAGE

TABLES

1.0 PIN ASSIGNMENT

The Intel387 SX Math CoProcessor pinout as viewed from the top side of the component is shown in Figure 1-1. V_{CC} and V_{SS} (GND) connections must be made to multiple pins. The circuit board should include V_{CC} and V_{SS} planes for power distribution and all V_{CC} and V_{SS} pins must be connected to the appropriate plane.

NOTE:
Pins identified as N.C. should remain completely unconnected.

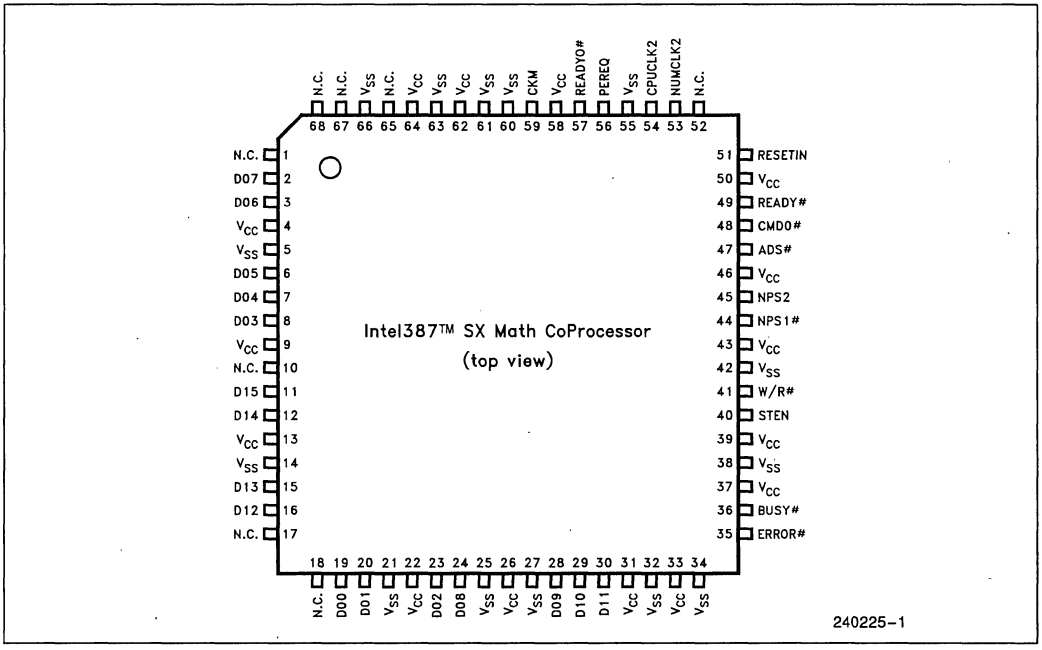


Figure 1-1. Intel387™ SX Math CoProcessor Pinout

Table 1-1. Pin Cross Reference—Functional Grouping

BUSY#	36	D00	19	V_{CC}	4	V_{SS}	5	N.C.	1
PEREQ	56	D01	20		9		14		10
ERROR#	35	D02	23		13		21		17
ADS#	47	D03	8		22		25		18
CMD0#	48	D04	7		26		27		52
NPS1#	44	D05	6		31		32		65
NPS2	45	D06	3		33		34		67
STEN	40	D07	2		37		38		68
W/R#	41	D08	24		39		42		
READY#	49	D09	28		43		55		
READYO#	57	D10	29		46		60		
CKM	59	D11	30		50		61		
CPUCLK2	54	D12	16		58		63		
NUMCLK2	53	D13	15		62		66		
RESETIN	51	D14	12		64				
		D15	11						

1.1 Pin Description Table

The following table lists a brief description of each pin on the Intel387 SX Math CoProcessor. For a more complete description refer to Section 4.1 Signal Description. The following definitions are used in these descriptions:

\# The signal is active LOW.

I Input Signal

O Output Signal

I/O Input and Output Signal

Symbol	Type	Name and Function
ADS#	I	**ADDRESS STROBE** indicates that the address and bus cycle definition is valid.
BUSY#	O	**BUSY** indicates that the Math CoProcessor is currently executing an instruction.
CKM	I	**CLOCKING MODE** is used to select synchronous or asynchronous clock modes.
CMD0	I	**COMMAND** determines whether an opcode or operand are being sent to the Math CoProcessor. During a read cycle it indicates which register group is being read.
CPUCLK2	I	**CPU CLOCK** input provides the timing for the bus interface unit and the execution unit in synchronous mode.
D15–D0	I/O	**DATA BUS** is used to transfer instructions and data between the Math CoProcessor and CPU.
ERROR#	O	**ERROR** signals that an unmasked exception has occurred.
NC	—	**NO CONNECT** should always remain unconnected. Connection of a N.C. pin may cause the Math CoProcessor to malfunction or be incompatible with future steppings.
NPS1#	I	**NPX SELECT 1** is used to select the Math CoProcessor.
NPS2	I	**NPX SELECT 2** is used to select the Math CoProcessor.
NUMCLK2	I	**NUMERICS CLOCK** is used in asynchronous mode to drive the Floating Point Execution Unit.
PEREQ	O	**PROCESSOR EXTENSION REQUEST** signals the CPU that the Math CoProcessor is ready for data transfer to/from its FIFO.
READY#	I	**READY** indicates that the bus cycle is being terminated.
READYO#	O	**READY OUT** signals the CPU that the Math CoProcessor is terminating the bus cycle.
RESETIN	I	**SYSTEM RESET** terminates any operation in progress and forces the Math CoProcessor to enter a dormant state.
STEN	I	**STATUS ENABLE** serves as a master chip select for the Math CoProcessor. When inactive, this pin forces all outputs and bi-directional pins into a floating state.
W/R#	I	**WRITE/READ** indicates whether the CPU bus cycle in progress is a read or a write cycle.
V_{CC}	I	**SYSTEM POWER** provides the +5V nominal D.C. supply input.
V_{SS}	I	**SYSTEM GROUND** provides the 0V connection from which all inputs and outputs are measured.

2.0 FUNCTIONAL DESCRIPTION

The Intel387 SX Math CoProcessor is designed to support the Intel386 SX Microprocessor and effectively extend the CPU architecture by providing fast execution of arithmetic instructions and transcendental functions. This component contains internal power management circuitry for reduced active power dissipation and an automatic idle mode.

2.1 Feature List

- New power saving design provides low power dissipation in active and idle modes.
- Higher Performance, 10%–25% higher benchmark performance than the original Intel387 SX Math CoProcessor.
- High Performance 84-bit Internal Architecture
- Eight 80-bit Numeric Registers, usable as individually addressable general registers or as a register stack.
- Full-range transcendental operations for SINE, COSINE, TANGENT, ARCTANGENT, and LOGARITHM.
- Programmable rounding modes and notification of rounding effects.
- Exception reporting either by software polling or hardware interrupts.
- Fully compatible with the SX Microprocessors.
- Expands Intel386 SX CPU data types to include 32-bit, 64-bit, and 80-bit Floating Point; 32-bit and 64-bit Integers; and 18 Digit BCD Operands.
- Directly extends the Intel386 SX CPU Instruction Set to trigonometric, logarithmic, exponential, and arithmetic functions for all data types.
- Operates independently of Real, Protected, and Virtual-86 Modes of the Intel386 SX Microprocessors.
- Fully compatible with the Intel387 SL Mobile and DX Math CoProcessors. Implements all Intel387 Math CoProcessor architectural enhancements over 8087 and 80287.
- Implements ANSI/IEEE Standard 754-1985 for binary floating point arithmetic.
- Upward Object Code compatible from 8087 and 80287.

2.2 Math CoProcessor Architecture

As shown in Figure 2-1, the Intel387 SX Math CoProcessor is internally divided into four sections; the Bus Control Logic, the Data Interface and Control Logic, the Floating Point Unit, and the Power Management Unit. The Bus Control Logic is responsible for the CPU bus tracking and interface. The Data Interface and Control Unit latches data and decodes instructions. The Floating Point Unit executes the mathematical instructions. The Power Management Unit is new to the Intel387 family and is the nucleus

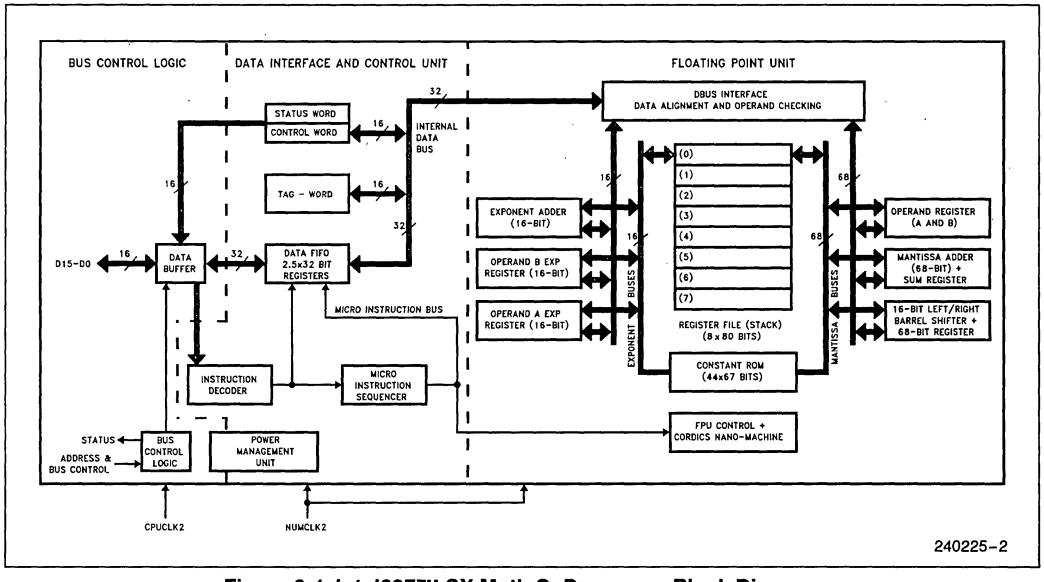


Figure 2-1. Intel387™ SX Math CoProcessor Block Diagram

of the static architecture. It is responsible for shutting down idle sections of the device to save power.

Microprocessor/Math CoProcessor Interface
The Intel386 CPU interprets the pattern 11011B in most significant five bits of an instruction as an opcode intended for a math coprocessor. Instructions thus marked are called ESCAPE or ESC instructions. Upon decoding the instruction as an ESC instruction, the Intel386 CPU transfers the opcode to the math coprocessor through an I/O write cycle at a dedicated address (8000F8H) outside the normal programmed I/O address range. The math coprocessor has dedicated output signals for controlling the data transfer and notifying the CPU if the Math CoProcessor is busy or that a floating point error has occurred.

2.3 Power Management

The Intel387 SX Math CoProcessor offers two modes of power management; dynamic and idle.

2.3.1 DYNAMIC MODE

Dynamic Mode is when the device is executing an instruction. Using Intel's CHMOS IV technology, the Intel387 SX Math CoProcessor draws considerably less power than its predecessor. The active power supply current is reduced to approximately 100 mA at 20 MHz and provides low case temperatures.

2.3.2 IDLE MODE

When an instruction is not being executed, the Intel387 SX Math CoProcessor will automatically change to ***Idle Mode***. Three clocks after completion of the previous instruction, the internal power manager shuts down the floating point execution unit and all non-essential circuitry. Only portions of the Bus Interface Unit remain active to monitor the CPU bus activity and to accept the next instruction when it is transferred. When the CPU transfers the next instruction to the Math CoProcessor, the Intel387 SX Math CoProcessor accepts the instruction and ramps the internal core within one clock so there is no impact to performance or throughput. In idle mode, the Intel387 SX Math CoProcessor draws typically 4 mA of current and reduces case temperature to near ambient.

NOTE:
In asynchronous clock mode (CKM = 0), the internal idle mode is disabled.

2.4 Compatibility

The Intel387 SX Math CoProcessor is compatible with the Intel387 SL Mobile Math CoProcessor. Due to the increased performance and internal pipelining effects, diagnostic programs should never use instruction execution time for test purposes.

2.5 Performance

The increased performance of floating point calculations can be attributed to the 84-bit architecture and floating point processor. For the CPU to execute floating point calculations requires very long software emulation methods with reduced resolution and accuracy. The performance of the Intel387 SX Math CoProcessor has been further enhanced through improvements in the internal microcode and through internal architectural changes. These refinements will increase Whetstone benchmarks by approximately 10% to 25% over the original Intel387 SX Math CoProcessor.

Real performance, however, should be measured with application software. Depending upon software coding, system overhead, and percentage of floating point instructions, performance can vary significantly.

3.0 PROGRAMMING INTERFACE

The Intel387 SX Math CoProcessor effectively extends to an Intel386 Microprocessor system additional instructions, registers, data types, and interrupts specifically designed to facilitate high-speed floating point processing. All communication between the CPU and the Math CoProcessor is transparent to applications software. The CPU automatically controls the Math CoProcessor whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the Math CoProcessor. All memory addressing modes, including use of displacement, base register, index register, and scaling are available for addressing numerical operands.

The Intel387 SX Math CoProcessor is software compatible with the Intel387 DX Math CoProcessors and supports all applications written for the Intel386 CPU and Intel387 Math CoProcessors.

3.1 Instruction Set

The Intel386 CPU interprets the pattern 11011B in most significant five bits of an instruction as an opcode intended for a math coprocessor. Instructions thus marked are called ESCAPE or ESC instruction.

The typical Math CoProcessor instruction accepts one or two operands and produces one or sometimes two results. In two-operand instructions, one operand is the contents of the Math CoProcessor register, while the other may be a memory location. The operands of some instructions are predefined; for example, FSQRT always takes the square root of the number in the top stack element.

The Intel387 SX Math CoProcessor instruction set can be divided into six groups. The following sections gives a brief description of each instruction. Section 8.0 defines the instruction format and byte fields. Further details can be obtained from the Intel387 User's Manual, Programmer's Reference, Order #231917.

3.1.1 DATA TRANSFER INSTRUCTIONS

The class includes the operations that load, store, and convert operands of any support data types.

Real Transfers

FLD	Load Real (single, double, extended)
FST	Store Real (single, double)
FSTP	Store Real and pop (single, double, extended)
FXCH	Exchange registers

Integer Transfers

FILD	Load (convert from) Integer (word, short, long)
FIST	Store (convert to) Integer (word, short)
FISTP	Store (convert to) Integer and pop (word, short, long)

Packed Decimal Transfers

FBLD	Load (convert from) packed decimal
FBSTP	Store packed decimal and pop

3.1.2 ARITHMETIC INSTRUCTIONS

This class of instructions provide variations on the basic add, subtract, multiply, and divide operations and a number of other basic arithmetic operations. Operands may reside in registers or one operand may reside in memory.

Addition

FADD	Add Real
FADDP	Add Real and pop
FIADD	Add Integer

Subtraction

FSUB	Subtract Real
FSUBP	Subtract Real and pop
FISUB	Subtract Integer
FSUBR	Subtract Real reversed
FSUBRP	Subtract Real reversed and pop
FISUBR	Subtract Integer reversed

Multiplication

FMUL	Multiply Real
FMULP	Multiply Real and pop
FIMUL	Multiply Integer

Division

FDIV	Divide Real
FDIVP	Divide Real and pop
FIDIV	Divide Integer
FDIVR	Divide Real reversed
FDIVRP	Divide Real reversed and pop
FIDIVR	Divide Integer reversed

Other Operations

FSQRT	Square Root
FSCALE	Scale
FPREM	Partial Remainder
FPREM1	IEEE standard partial remainder
FRNDINT	Round to Integer
FXTRACT	Extract Exponent and Significand
FABS	Absolute Value
FCHS	Change sign

3.1.3 COMPARISON INSTRUCTION

Instructions of this class allow comparison of numbers of all supported real and integer data types. Each of these instructions analyzes the top stack element often in relationship to another operand and reports the result as a condition code in the status word.

FCOM	Compare Real
FCOMP	Compare Real and pop
FCOMPP	Compare Real and pop twice
FUCOM	Unordered compare Real
FUCOMP	Unordered compare Real and pop
FUCOMPP	Unordered compare Real and pop twice
FICOM	Compare Integer
FICOMP	Compare Integer and pop
FTST	Test
FXAM	Examine

3.1.4 TRANSCENDENTAL INSTRUCTIONS

This group of the Intel387 operations includes trigonometric, inverse trigonometric, logarithmic and exponential functions. The transcendental operate on the top one or two stack elements, and they return their results to the stack. The trigonometric operations assume their arguments are expressed in radians. The logarithmic and exponential operations work in base 2.

FSIN	Sine
FCOS	Cosine
FSINCOS	Sine and cosine
FPTAN	Tangent
FPATAN	Arctangent of ST(1)/ST
F2XM1	2^X-1
FYL2X	$Y * \log_2 X$
FYL2XP1	$Y * \log_2(X + 1)$

3.1.5 LOAD CONSTANT INSTRUCTIONS

Each of these instructions loads (pushes) a commonly used constant onto the stack. The constants have extended real values nearest to the infinitely precise numbers. The only error that can be generated is an Invalid Exception if a stack overflow occurs.

FLDZ	Load +0.0
FLD1	Load +1.0
FLDPI	Load π
FLDL2T	Load $\log_2 10$
FLDL2E	Load $\log_2 e$
FLDLG2	Load $\log_{10} 2$
FLDLN2	Load $\log_e 2$

3.1.6 PROCESSOR INSTRUCTIONS (ADMINISTRATIVE)

FINIT	Initialize Math CoProcessor
FLDCW	Load Control Word
FSTCW	Store Control Word
FLDCW	Load Status Word
FSTSW	Store Status Word
FSTSW AX	Store Status Word to AX register
FCLEX	Clear Exceptions
FSTENV	Store Environment
FLDENV	Load Environment
FSAVE	Save State
FRSTOR	Restore State
FINCSTP	Increment Stack pointer
FDECSTP	Decrement Stack pointer
FFREE	Free Register
FNOP	No Operation
FWAIT	Report Math CoProcessor Error

3.2 Register Set

Figure 3-1 shows the Intel387 SX Math CoProcessor register set. When a Math CoProcessor is present in a system, programmers may use these registers in addition to the registers normally available on the CPU.

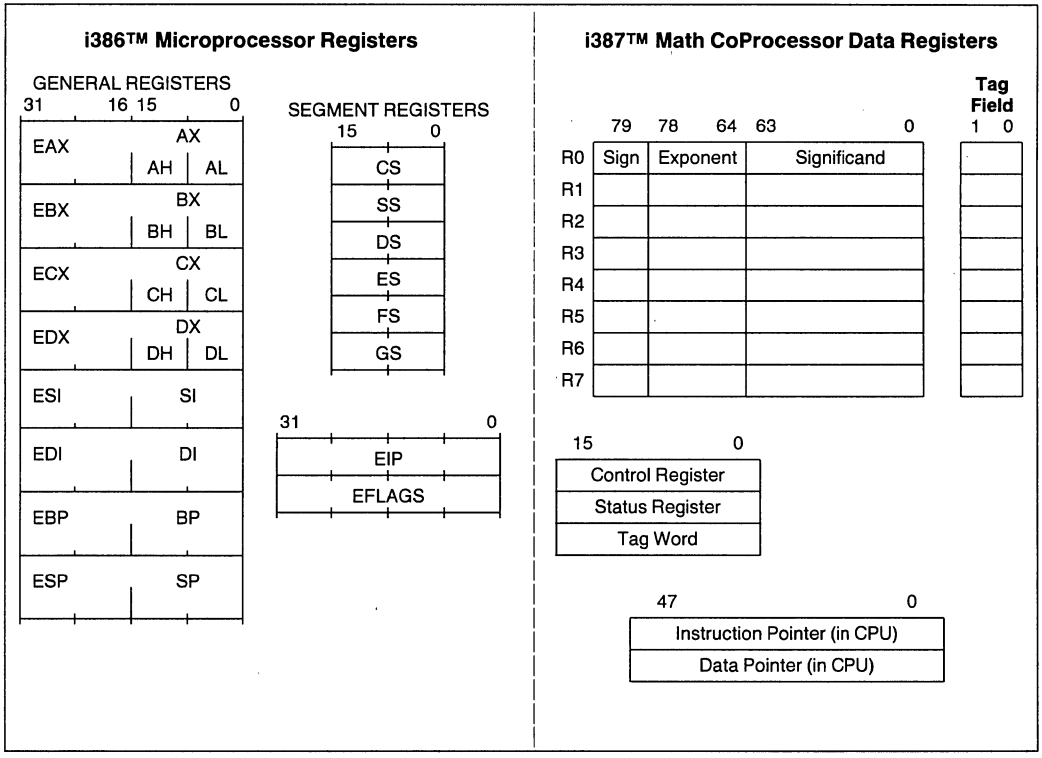


Figure 3-1. Intel386™ CPU and Intel387™ Math CoProcessor Register Set

3.2.1 STATUS WORD (SW) REGISTER

The 16-bit status word (in the status register) shown in Figure 3-2 reflects the overall state of the Math CoProcessor. It can be read and inspected by programs using the FSTSW memory or FSTSW AX instructions.

Bit 15, the Busy bit (B) is included for 8087 compatibility only. It always has the same value as the Error Summary bit (ES, bit 7 of status word); it does not indicate the status of the BUSY# output of the Math CoProcessor.

Bits 13–11 (TOP) serves as the pointer to the Math CoProcessor data register that is the current Top-Of-Stack. The significance of the stack top is described in Section 3.2.5 Data Registers.

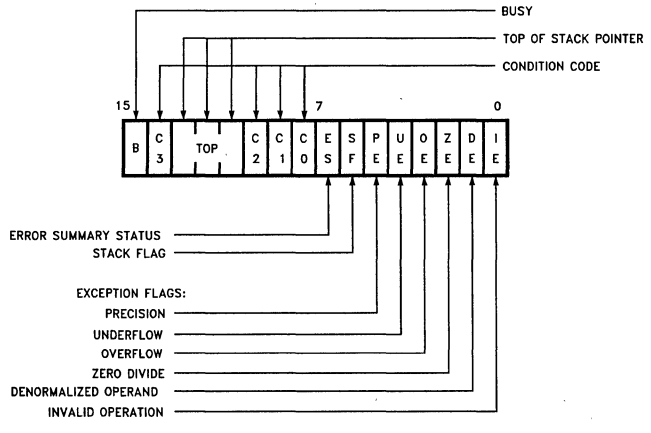
The four numeric condition code bits (C_3–C_0, Bit 14, 10–8) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of the instructions on the condition code are summarized in Tables 3-1 through 3-4. These condition code bits are used principally for conditional branching. The FSTSW AX instructions stores the Math CoProcessor status word directly to the CPU AX register, allowing the condition codes to be inspected efficiently by Intel386 CPU code. The Intel386 CPU SAHF instruction can copy C_3–C_0 directly to the flag bits to simplify conditional branching. Table 3-5 shows the mapping of these bits to the Intel386 CPU flag bits.

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the ERROR# signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C_1) distinguishes between stack overflow (C_1 = 1) or underflow (C_1 = 0).

Bit 5–0 are the six exception flags of the status word and are set to indicate that during an instruction execution the Math CoProcessor has detected one of six possible exception conditions since these status bits were last cleared or reset. Section 3.5 entitled Exception Handling explains how they are set and used.

The exception flags are "sticky" bits and can only be cleared by the instructions FINIT, FCLEX, FLDENV, FSAVE, and FRSTOR. Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the value of ES (bit 7) and B (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the ERROR# output of the Math CoProcessor is activated immediately.



ES is set if any unmasked exception bit is set; cleared otherwise. See Table 2-2 for interpretation of condition code.
TOP values:
000 = Register 0 is Top of Stack
001 = Register 1 is Top of Stack

111 = Register 7 is Top of Stack
For definitions of exceptions, refer to the section entitled "Exception Handling"

Figure 3-2. Status Word

Table 3-1. Condition Code Interpretation

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1 (see Table 3-2)	Three least significant bits of quotient Q2 Q0		Q1 or O/U#	Reduction 0 = complete 1 = incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 3-3)		Zero or O/U#	Operand is not comparable (Table 3-3)
FXAM	Operand class (see Table 3-4)		Sign or O/U#	Operand class (Table 3-4)
FCHS, FABS, FXCH, FINCSTP, FDECSTP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or O/U#	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U#	UNDEFINED
FPTAN, FSIN FCOS, FSINCOS	UNDEFINED		Roundup or O/U#, undefined if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

O/U#	When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).
Reduction	If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.
Roundup	When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.
UNDEFINED	Do not rely on finding any specific value in these bits.

Table 3-2. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interation required for complete reduction	
	Q1	Q0	Q2	Q MOD8	Complete Reduction: C0, C3, C1 contain three least significant bits of quotient
0	0	0	0	0	
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

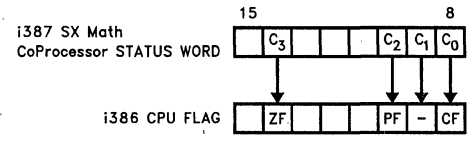
Table 3-3. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 3-4. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

Table 3-5 Mapping Condition Codes to Intel386™ CPU Flag Bits



240225-4

3.2.2 CONTROL WORD (CW) REGISTER

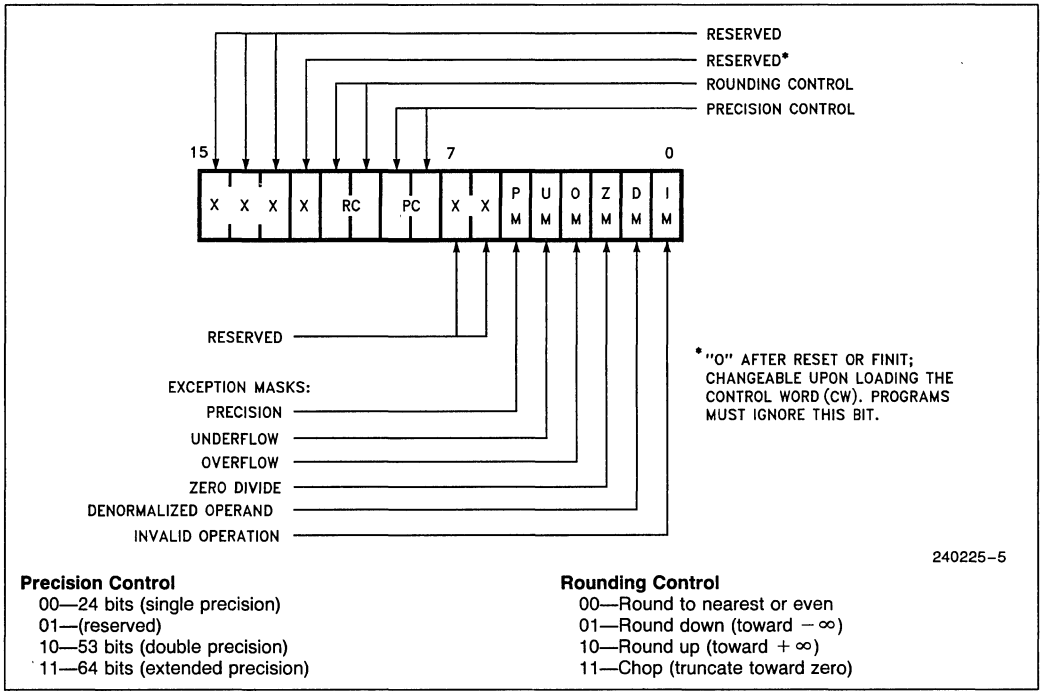
The Math CoProcessor provides the programmer with several processing options that are selected by loading a control word from memory into the control register. Figure 3-3 show the format and encoding of fields in the control word.

The low-order byte of the control word register is used to configure the exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the Math CoProcessor recognizes. See Section 3.5, Exception Handling, for further explanation on the exception control and definition.

The high-order byte of the control word is used to configure the Math CoProcessor operating mode, including precision, rounding and infinity control.

- The rounding control (RC) field (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS, and FCHS) and all transcendental instructions.
- The precision control (PC) field (bits 9–8) can be used to set the Math CoProcessor internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions FADD, FSUB(R), FMUL, FDIV(R), and FSQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.
- The "infinity control bit" (bit 12) is not meaningful to the Intel387 SX Math CoProcessor and programs must ignore its value. To maintain compatibility with the 8087 and 80287 (non-387 core), this bit can be programmed, however, regardless of its value the Intel387 SX Math CoProcessor always treats infinity in the affine sense ($-\infty < +\infty$). This bit is initialized to zero both after a hardware reset and after FINIT instruction.

All other bits are reserved and should not be programmed, to assure compatibility with future processors.



Precision Control
00—24 bits (single precision)
01—(reserved)
10—53 bits (double precision)
11—64 bits (extended precision)

Rounding Control
00—Round to nearest or even
01—Round down (toward $-\infty$)
10—Round up (toward $+\infty$)
11—Chop (truncate toward zero)

Figure 3-3. Control Word

3.2.3 DATA REGISTER

Intel387 SX Math CoProcessor data register set consists of eight registers (R0–R7) which are treated as both a stack and a general register file. Each of these data registers in the Math CoProcessor is 80 bits wide and is divided into fields corresponding to the Math CoProcessor's extended-precision real data type, which is used for internal calculations.

The Math CoProcessor register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as individually addressable registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "store and pop" operation stores the value from the current top register into memory and then increments TOP by one. The Math CoProcessor register stack grows "down" toward lower-addressed registers.

Most of the Intel387 SX Math CoProcessor operations use the register stack as the operand(s) and/or as a place to store the result. Instructions may address the data register either implicitly or explicitly. Many instructions operate on the register at the top of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to use. Explicit register addressing is also relative to TOP (where ST denotes the current stack top and ST(i) refers to the i'th register from the ST in the stack so the real register address in computed as ST+i).

3.2.4 TAG WORD (TW) REGISTER

The tag word marks the content of each numeric data register, as Figure 3-4 shows. Each two-bit tag represents one of the eight data register. The principal function of the tag word is to optimize the Math CoProcessor's performance and stack handling by making it possible to distinguish between empty and non-empty register locations. It also enables exception handlers to identify special values (e.g. NaNs or denormals) in the contents of a stack location without the need to perform complex decoding of the actual data.

3.2.5 INSTRUCTION AND DATA POINTERS

Because the Math CoProcessor operates in parallel with the CPU, any exceptions detected by the Math CoProcessor may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the numeric instruction which caused the exception, the Intel386 Microprocessor contains registers that aid in diagnosis. These registers supply the address of the failing instruction and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written exception handlers. These registers are located in the CPU, but appear to be located in the Math CoProcessor because they are accessed by the ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR; which transfer the values between the registers and memory. Whenever the CPU executes a new ESC instruction (except administrative instructions), it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if present) and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the CPU (protected mode or real-address mode) and depending on the operand size attribute in effect (32-bit operand or 16-bit operand). (See Figures 3-5, 3-6, 3-7, and 3-8.) Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

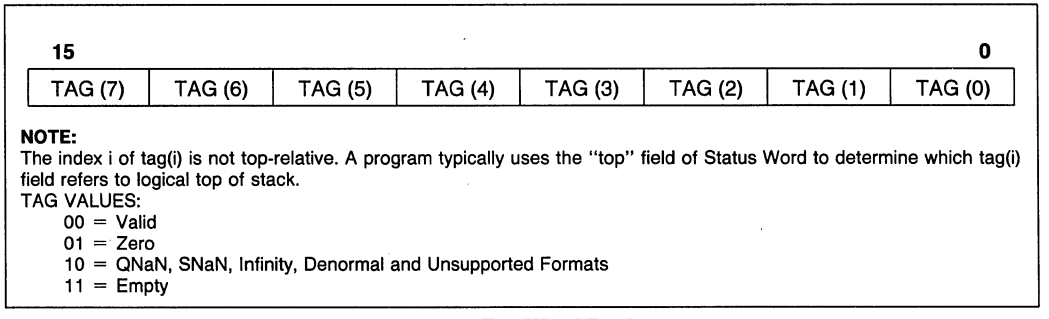


Figure 3-4. Tag Word Register

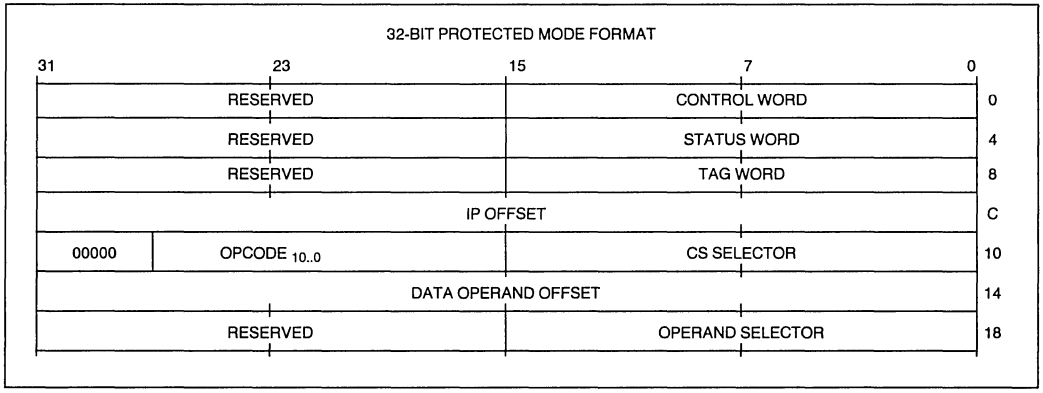


Figure 3-5. Instruction and Data Pointer Image in Memory, 32-Bit Protected-Mode Format

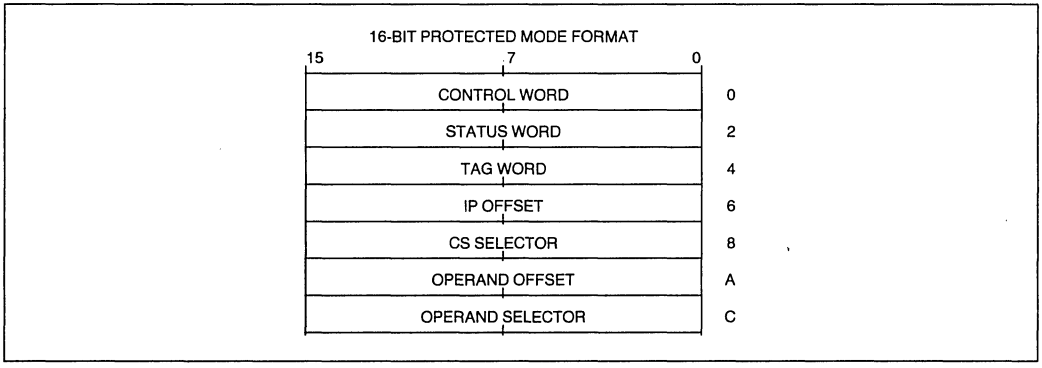


Figure 3-6. Instruction and Data Pointer Image in Memory, 16-Bit Protected-Mode Format

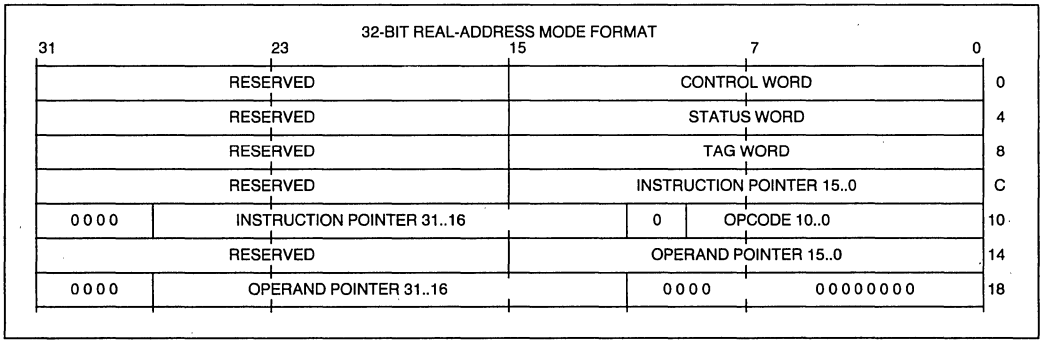


Figure 3-7. Instruction and Data Pointer Image in Memory, 32-Bit Real-Mode Format

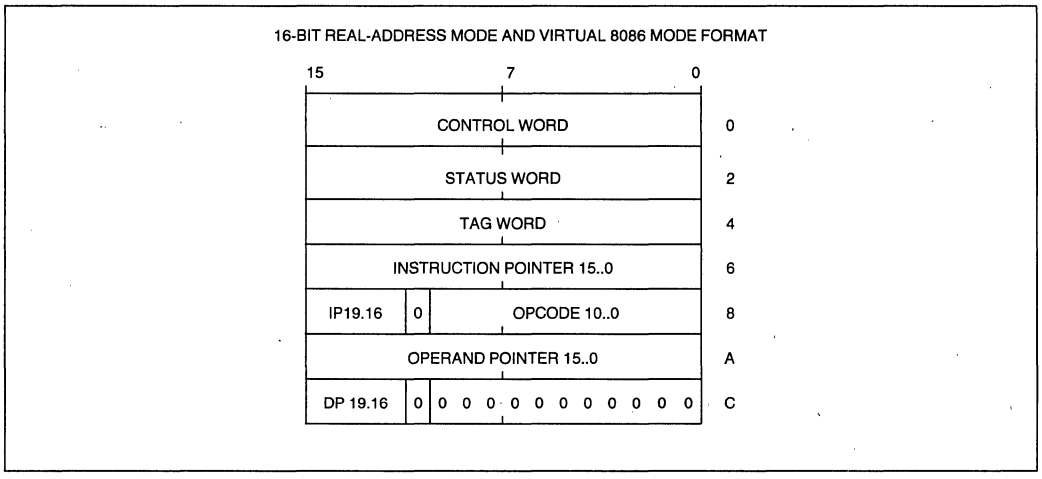


Figure 3-8. Instruction and Data Pointer Image in Memory, 16-Bit Real-Mode Format

3.3 Data Types

Table 3-6 lists the seven data types that the Math CoProcessor supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at physical-memory addresses that correspond to the word size of the CPU; operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

The data type formats can be divided into three classes: binary integer, decimal integer, and binary real. These formats, however, exist in memory only. Internally, the Math CoProcessor holds all numbers in the extended-precision real format. Instructions that load operands from memory automatically convert operands represented in memory as 16, 32, or 64-bit integers, 32 or 64-bit floating point numbers, or 18 digit packed BCD numbers into extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

In addition to the typical real and integer data values, the Intel387 SX Math CoProcessor data formats encompass encodings for a variety of special values. These special values have significance and can express relevant information about the computations or operations that produced them. The various types of special values are denormal real numbers, zeros, positive and negative infinity, NaNs (Not-a-Number), Indefinite, and unsupported formats. For further information on data types and formats, see the Intel387 Programmer's Reference Manual.

3.4 Interrupt Description

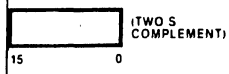
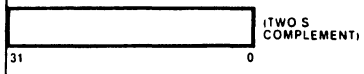
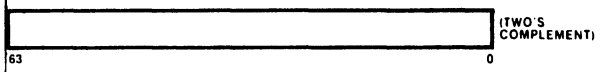
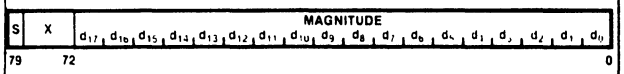
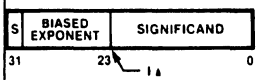
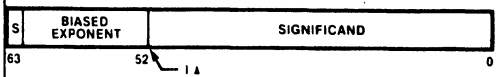
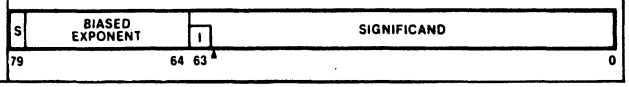
CPU interrupts are used to report errors or exceptional conditions while executing numeric programs in either real or protected mode. Table 3-7 shows these interrupts and their functions.

3.5 Exception Handling

The Math CoProcessor detects six different exception conditions that occur during instruction execution. Table 3-8 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the Math CoProcessor if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the ERROR# signal. When the CPU attempts to execute another ESC instruction or WAIT, exception 16 occurs. The exception condition must be resolved via an interrupt service routine. The return address pushed onto the CPU stack upon entry to the service routine does not necessarily point to the failing instruction nor to the following instruction. The CPU saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

Table 3-6. Intel387™ SX Math CoProcessor Data Type Representation In Memory

Data Formats	Range	Precision	Most Significant Byte = HIGHEST ADDRESSED BYTE
Word Integer	$\pm 10^4$	16 Bits	
Short Integer	$\pm 10^9$	32 Bits	
Long Integer	$\pm 10^{18}$	64 Bits	
Packed BCD	$\pm 10^{18}$	18 Digits	
Single Precision	$\pm 10^{\pm 38}$	24 Bits	
Double Precision	$\pm 10^{\pm 308}$	53 Bits	
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits	

240225-23

NOTES:
1. S = Sign bit (0 = positive, 1 = negative)
2. d_n = Decimal digit (two per byte)
3. X = Bits have no significance; Math CoProcessor ignores when loading, zeros when storing
4. ▲ = Position of implicit binary point
5. I = Integer bit of significand; stored in temporary real, implicit in single and double precision
6. Exponent Bias (normalized values):
   Single: 127 (7FH)
   Double: 1023 (3FFH)
   Extended REal: 16383 (3FFFH)
7. Packed BCD: $(-1)^S (D_{17}..D_0)$
8. Real: $(-1)^S (2^{E-BIAS}) (F_0 F_1...)$

Table 3-7. CPU Interrupt Vectors Reserved for Math CoProcessor

Interrupt Number	Cause of Interrupt
7	An ESC instruction was encountered when EM or TS of CPU control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction causes interrupt 7. This indicates that the current Math CoProcessor context may not belong to the current task.
9	In a protected-mode system, an operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for expand-up segments, zero for expand-down segments) and spanned inaccessible addresses(1). The failing numerics instruction is not restartable. The address of the failing numerics instruction and data operand may be lost; an FSTENV does not return reliable addresses. The segment overrun exception should be handled by executing an FNINIT instruction (i.e., an FINIT without a preceding WAIT). The exception can be avoided by never allowing numerics operands to cross the end of a segment.
13	In a protected-mode system, the first word of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the ESC instruction that caused the exception, including any prefixes. The Math CoProcessor has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC and WAIT instructions can cause this interrupt. The CPU return address pushed onto the stack of the exception handler points to a WAIT or ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the Math CoProcessor. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.

NOTE:
1. An operand may wrap around an addressing limit when the segment limit is near an addressing limit and the operand is near the largest valid address in the segment. Because of the wrap-around, the beginning and ending addresses of such an operand will be at opposite ends of the segment. There are two ways that such an operand may also span inaccessible addresses: 1) if the segment limit is not equal to the addressing limit (e.g. addressing limit is FFFFH and segment limit is FFFDH) the operand will span addresses that are not within the segment (e.g. an 8-byte operand that starts at valid offset FFFCH will span addresses FFFC–FFFFH and 0000-0003H; however addresses FFFEH and FFFFH are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible segments but intermediate bytes of the operand fall in a not-present page or in a segment or page to which the procedure does not have access rights.

Table 3-8. Intel387™ SX Math CoProcessor Exceptions

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signalling NaN, unsupported format, indeterminate for (0-∞, 0/0, (+∞) + (−∞), etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinte
Denormalized Operand	At least one of the operands is denormalized, i.e., it has the smallest exponent but a nonzero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number.	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes the loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g. 1/3); the result is rounded according to the rounding mode.	Normal processing continues

3.6 Initialization

After FNINIT or RESET, the control word contains the value 037FH (all exceptions masked, precision control 64 bits, rounding to nearest) the same values as in an Intel287 after RESET. For compatibility with the 8087 and Intel287, the bit that used to indicate infinity control (bit 12) is set to zero; however, regardless of its setting, infinity is treated in the affine sense. After FNINIT or RESET, the status word is initialized as follows:

- All exceptions are set to zero.
- Stack TOP is zero, so that after the first push the stack top will be register seven (111B).
- The condition code C_3–C_0 is undefined.
- The B-bit is zero.

The tag word contains FFFFH (all stack locations are empty).

The Intel386 Microprocessor and Intel387 Math CoProcessor initialization software must execute a FNINIT instruction (i.e., FINIT without a preceding WAIT) after RESET. The FNINIT is not strictly required for the Intel386 software, but Intel recommends its use to help ensure upware compatibility with other processors. After a hardware RESET, the ERROR# output is asserted to indicate that an Intel387 Math CoProcessor is present. To accomplish this, the IE (Invalid Exception) and ES (Error Summary) bits of the status word are set, and the IM bit (Invalid Exception Mask) in the control word is cleared. After FNINIT, the status word and the control word have the same values as in an Intel287 Math CoProcessor after RESET.

3.7 Processing Modes

The Intel387 SX Math CoProcessor works the same whether the CPU is executing in real-addressing mode, protected mode, or virtual-8086 mode. All references to memory for numerics data or status information are performed by the CPU, and therefore obey the memory-management and protection rules of the CPU mode currently in effect. The Intel387 SX Math CoProcessor merely operates on instructions and values passed to it by the CPU and therefore is not sensitive to the processing mode of the CPU.

The real-address mode and virtual-8086 mode, the Intel387 SX Math CoProcessor is completely upward compatible with software for the 8086/8087 and 80286/80287 real-address mode systems.

In protected mode, the Intel387 SX Math CoProcessor is completely upward compatible with software for the 80286/80287 protected mode system.

The only differences of operation that may appear when 8086/8087 programs are ported to the protected mode (not using virtual-8086 mode) is in the format of operands for the administrative instructions FLDENV, FSTENV, FRSTOR, and FSAVE.

3.8 Programming Support

Using the Intel387 SX Math CoProcessor requires no special programming tools, because all new instructions and data types are directly supported by the assembler and compilers for high-level languages. All Intel386 Microprocessor development tools that support Intel387 Math CoProcessor programs can also be used to develop software for the Intel386 SX Microprocessors and Intel387 SX Math CoProcessors. All 8086/8088 development tools that support the 8087 can also be used to develop software for the CPU and Math CoProcessor in real-address mode or virtual-8086 mode. All 80286 development tools that support the Intel287 Math CoProcessor can also be used to develop software for the Intel386 CPU and Intel387 Math CoProcessor.

4.0 HARDWARE SYSTEM INTERFACE

In the following description of hardware interface, the # symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

4.1 Signal Description

In the following signal descriptions, the Intel387 SX Math CoProcessor pins are grouped by function as shown by Table 4-1. Table 4-1 lists every pin by its identifier, gives a brief description and lists some of its characteristics (Refer to Figure 1-1 and Table 1-1 for pin configuration).

All output signals can be tri-stated by driving STEN inactive. The output buffers of the bi-directional data pins D15–D0 are also tri-state; they only leave the floating state during read cycles when the Math CoProcessor is selected.

4.1.1 Intel386 CPU CLOCK 2 (CPUCLK2)

This input uses the CLK2 signal of the CPU to time the bus control logic. Several other Math CoProcessor signals are referenced to the rising edge of this signal. When CKM = 1 (synchronous mode) this pin also clocks the data interface and control unit and the floating point unit of the Math CoProcessor. This pin requires CMOS-level input. The signal on this pin is divided by two to produce the internal clock signal CLK.

4.1.2 Intel387 MATH COPROCESSOR CLOCK 2 (NUMCLK2)

When CKM = 0 (asynchronous mode), this pin provides the clock for the data interface and control unit and the floating point unit of the Math CoProcessor. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10 and the maximum frequency must not exceed the device specifications. When CKM = 1 (synchronous mode), signals on this pin are ignored: CPUCLK2 is used instead for the data interface and control unit and the floating point unit. This pin requires CMOS level input and should be tied low if not used.

Table 4-1. Pin Summary

Pin Name	Function	Active State	Input/ Output	Referenced To . . .
Execution Control				
CPUCLK2	Microprocessor Clock2		I	
NUMCLK2	Math CoProcessor Clock2		I	
CKM	Math CoProcessor Clock Mode		I	
RESETIN	System Reset	High	I	CPUCLK2
Math CoProcessor Handshake				
PEREQ	Processor Request	High	O	CPUCLK2
BUSY#	Busy Status	Low	O	CPUCLK2
ERROR#	Error Status	Low	O	NUMCLK2
Bus Interface				
D15–D0	Data Pins		I/O	CPUCLK2
W/R#	Write/Read Bus Cycle	High/Low	I	CPUCLK2
ADS#	Address Strobe	Low	I	CPUCLK2
READY#	Bus Ready Input	Low	I	CPUCLK2
READYO#	Ready Output	Low	O	CPUCLK2
Chip/Port Select				
STEN	Status Enable	High	I	CPUCLK2
NPS1#	Numerics Select #1	Low	I	CPUCLK2
NPS2	Numerics Select #2	High	I	CPUCLK2
CMD0#	Command	Low	I	CPUCLK2
Power and Ground				
V_{CC}	System Power			
V_{SS}	System Ground			

4.1.3 CLOCKING MODE (CKM)

This pin is strapping option. When it is strapped to V_{CC} (HIGH), the Math CoProcessor operates in synchronous mode; when strapped to V_{SS} (LOW), the Math CoProcessor operates in asynchronous mode. These modes relate to clocking of the internal data interface and control unit and the floating point unit only; the bus control logic always operates synchronously with respect to the CPU.

Synchronous mode requires the use of only one clock, the CPU's CLK2. Use of synchronous mode eliminates one clock generator from the board design and is recommended for all designs. Synchronous mode also allows the internal Power Management Unit to enable the idle and standby power saving modes.

Asynchronous mode can provide higher performance of the floating point unit by running a faster clock on NUMCLK2. (The CPU's CLK2 must still be connected to CPUCLK2 input.) This allows the floating point unit to run up to 40% faster than in synchronous mode. Internal power management is disabled in asynchronous mode.

4.1.4 SYSTEM RESET (RESETIN)

A LOW to HIGH transition on this pin causes the Math CoProcessor to terminate its present activity and to enter a dormant state. RESETIN must remain active (HIGH) for at least 40 CPUCLK2 (NUMCLK2 if CKM = 0) periods.

The HIGH to LOW transitions of RESETIN must be synchronous with CPUCLK2, so that the phase of the internal clock of the bus control logic (which is the CPUCLK2 divided by two) is the same as the phase of the internal clock of the CPU. After RESETIN goes LOW, at least 50 CPUCLK2 (NUMCLK2 if CKM = 0) periods must pass before the first Math CoProcessor instruction is written into the Math CoProcessor. This pin should be connected to the CPU RESET pin. Table 4-2 shows the status of the output pins during the reset sequence. After a reset, all output pins return to their inactive state except for ERROR# which remains active (for CPU recognition) until cleared.

Table 4-2. Output Pin Status during Reset

Pin Value	Pin Name
HIGH	READYO#, BUSY#
LOW	PEREQ, ERROR#
Tri-State OFF	D15–D0

4.1.5 PROCESSOR REQUEST (PEREQ)

When active, this pin signals to the CPU that the Math CoProcessor is ready for data transfer to/from its data FIFO. When all data is written to or read from the data FIFO, PEREQ is deactivated. This signal always goes inactive before BUSY# goes inactive. This signal is reference to CPUCLK2. It should be connected to the CPU PEREQ input pin.

4.1.6 BUSY STATUS (BUSY#)

When active, this pin signals to the CPU that the Math CoProcessor is currently executing an instruction. This signal is referenced to CPUCLK2. It should be connected to the CPU BUSY# input pin.

4.1.7 ERROR STATUS (ERROR#)

This pin reflects the ES bit of the status register. When active, it indicates that an unmasked exception has occurred. This signal can be changed to the inactive state only by the following instructions (without a preceding WAIT); FNINIT, FNCLEX, FNSTENV, FNSAVE, FLDCW, FLDENV, and FRSTOR. ERROR# is driven active during RESET to indicate to the CPU that the Math CoProcessor is present. This pin is referenced to NUMCLK2 (or CPUCLK2 if CKM = 1). It should be connected to the ERROR# pin of the CPU.

4.1.8 DATA PINS (D15–D0)

These bi-directional pins are used to transfer data and opcodes between the CPU and Math CoProcessor. They are normally connected directly to the corresponding CPU data pins. HIGH state indicates a value of one. D0 is the least significant data bit. Timings are referenced to rising edge of CPUCLK2.

4.1.9 WRITE/READ BUS CYCLE (W/R#)

This signal indicates to the Math CoProcessor whether the CPU bus cycle in progress is a read or a write cycle. This pin should be connected directly to the CPU's W/R# pin. HIGH indicates a write cycle to the Math CoProcessor; LOW a read cycle from the Math CoProcessor. This input is ignored if any of the signals STEN, NPS1#, or NPS2 are inactive. Setup and hold times are referenced to CPUCLK2.

4.1.10 ADDRESS STROBE (ADS#)

This input, in conjunction with the READY# input, indicates when the Math CoProcessor bus control logic may sample W/R# and the chip select signals. Setup and hold times are referenced to CPUCLK2. This pin should be connected to the ADS# pin of the CPU.

4.1.11 BUS READY INPUT (READY#)

This input indicates to the Math CoProcessor when a CPU bus cycle is to be terminated. It is used by the bus control logic to trace bus activities. Bus cycles can be extended indefinitely until terminated by READY#. This input should be connected to the same signal that drives the CPU's READY# input. Setup and hold times are referenced to CPUCLK2.

4.1.12 READY OUTPUT (READYO#)

This pin is activated at such a time that write cycles are terminated after two clocks (except FLDENV and FRSTOR) and read cycles after three clocks. In configurations where no extra wait states are required, this pin must directly or indirectly drive the READY# input of the CPU. Refer to the section entitled "BUS OPERATION" for details. This pin is activated only during bus cycles that select the Math CoProcessor. This signal is referenced to CPUCLK2.

(FLDENV and FRSTOR require data transfers larger than the FIFO. Therefore, PEREQ is activated for the duration of transferring 2 words of 32 bits and then deactivated until the FIFO is ready to accept two additional words. The length of the write cycles of the last operand word in each transfer as well as the first operand word transfer of the entire instruction is 3 clocks instead of 2 clocks. This is done to give the Intel386 CPU enough time to sample PEREQ and to notice that the Intel387 is **not** ready for additional transfers.)

4.1.13 STATUS ENABLE (STEN)

This pin serves as a chip select for the Math CoProcessor. When inactive, this pin forces BUSY#, PEREQ, ERROR# and READYO# outputs into a floating state. D15–D0 are normally floating and will leave the floating state only if STEN is active and additional conditions are met (read cycle). STEN also causes the chip to recognize its other chip select inputs. STEN makes it easier to do on-board testing (using the overdrive method) of other chips in systems containing the Math CoProcessor. STEN should be pulled up with a resistor so that it can be pulled down when testing. In boards that do not use on-board testing STEN should be connected to V_{CC}. Setup and hold times are relative to CPUCLK2. Note that STEN must maintain the same setup and hold times as NPS1#, NPS2, and CMD0# (i.e., if STEN changes state during a Math CoProcessor bus cycle, it must change state during the same CLK period as the NPS1#, NPS2, and CMD0# signals).

4.1.14 MATH COPROCESSOR SELECT 1 (NPS1#)

When active (along with STEN and NPS2) in the first period of a CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the Math CoProcessor. This pin should be connected directly to the M/IO# pin of the CPU, so that the Math CoProcessor is selected only when the CPU performs I/O cycles. Setup and hold times are referenced to the rising edge of CPUCLK2.

4.1.15 MATH COPROCESSOR SELECT 2 (NPS2)

When active (along with STEN and NPS1#) in the first period of a CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the Math CoProcessor. This pin should be connected directly to the A23 pin of the CPU, so that the Math CoProcessor is selected only when the CPU issues one of the I/O addresses reserved for the Math CoProcessor (8000F8h, 8000FCh, or 8000FEh which is treated as 8000FCh by the Math CoProcessor). Setup and hold times are referenced to the rising edge of CPUCLK2.

4.1.16 COMMAND (CMD0#)

During a write cycle, this signal indicates whether an opcode (CMD0# active low) or data (CMD0# inactive high) is being sent to the Math CoProcessor. During a read cycle, it indicates whether the control or status register (CMD0# active) or a data register (CMD0#) is being read. CMD0# should be connected directly to the A2 output of the CPU. Setup and hold times are referenced to the rising edge of CPUCLK2 at the end of PH2.

4.1.17 SYSTEM POWER (V_{CC})

System power provides the +5V DC supply input. All V_{CC} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

4.1.18 SYSTEM GROUND (V_{SS})

System ground provides the 0V connection from which all inputs and outputs are measured. All V_{SS} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

4.2 System Configuration

The Intel387 SX Math CoProcessor is designed to interface with the Intel386 SX Microprocessor as shown by Figure 4-1. A dedicated communication protocol makes possible high-speed transfer of op-codes and operands between the CPU and Math CoProcessor. The Intel387 SX Math CoProcessor is designed so that no additional components are required for interface with the CPU. Most control pins of the Math CoProcessor are connected directly to pins of the CPU.

The interface between the Math CoProcessor and the CPU has these characteristics:

- The Math CoProcessor shares the local bus of the Intel386 SX Microprocessor.
- The CPU and Math CoProcessor share the same reset signals. They may also share the same clock input; however, for greatest performance, an external oscillator may be needed.
- The corresponding Busy#, ERROR#, and PEREQ pins are connected together.
- The Math CoProcessor NPS1# and NPS2 inputs are connected to the latched CPU M/IO# and A23 outputs respectively. For Math CoProcessor cycles, M/IO# is always LOW and A23 always HIGH.
- The Math CoProcessor input CMD0 is connected to the latched A_2 output. The Intel386 SX Microprocessor generates address 8000F8H when writing a command and address 8000FCH or 8000FEH (treated as 8000FCH by the Intel387 SX Math CoProcessor) when writing or reading data. It does not generate any other addresses during Math CoProcessor bus cycles.

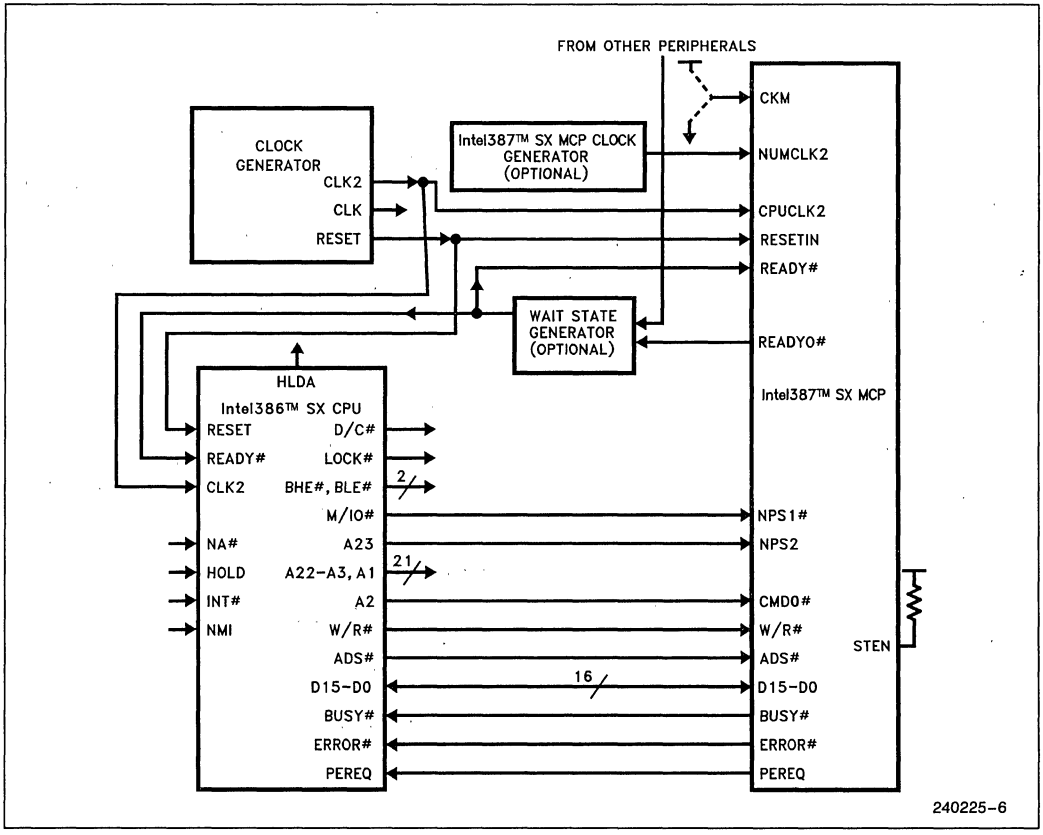


Figure 4-1. Intel386™ SX CPU and Intel387™ SX Math CoProcessor System Configuration

4.3 Math CoProcessor Architecture

As shown in Figure 2-1 Block Diagram, the Intel387 SX Math CoProcessor is internally divided into four sections; the Bus Control Logic (BCL), the Data Interface and Control Logic, the Floating Point Unit (FPU), and the Power Management Unit (PMU). The Bus Control Logic is responsible for the CPU bus tracking and interface. The BCL is the only unit in the Math CoProcessor that must run synchronously with the CPU; the rest of the Math CoProcessor can run asynchronously with respect to the CPU. The Data Interface and Control Unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, sequencing the microinstructions, and for handling some of the administrative instructions. The Floating Point Unit (with the support of the control unit which contains the sequencer and other support units) executes the mathematical instructions. The Power Manager is new to the Intel387 family. It is responsible for shutting down idle sections of the device to save power.

4.3.1 BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from the memory to the Math CoProcessor and transferring outputs from the Math CoProcessor to memory.

4.3.2 DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the FIFO or the instruction decoder. The instruction decoder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, FSTCW, FSETPM, or FRSTPM, the control unit executes it independently of the FPU and the sequencer. The data interface and control unit is the unit that generates the BUSY#, PEREQ, and ERROR# signals that synchronize the Math CoProcessor activities with the CPU.

4.3.3 FLOATING POINT UNIT

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

4.3.4 POWER MANAGEMENT UNIT

The Power Management Unit (PMU) controls all internal power savings circuits. When the Math CoProcessor is not executing an instruction, the PMU disables the internal clock to the FPU, Control Unit, and Data Interface within three clocks. The Bus Control Logic remains enabled to accept the next instruction. Upon decode of a valid Math CoProcessor bus cycle, the PMU enables the internal clock to all circuits. No loss in performance occurs.

4.4 Bus Cycles

All bus cycles are initiated by the CPU. The pins STEN, NPS1#, NPS2, CMD0, and W/R# identify bus cycles for the Math CoProcessor. Table 4-3 defines the types of Math CoProcessor bus cycles.

Table 4-3. Bus Cycle Definition

STEN	NPS1#	NPS2	CMD0#	W/R#	Bus Cycle Type
0	X	X	X	X	Math CoProcessor not selected and all outputs in floating state
1	1	X	X	X	Math CoProcessor not selected
1	X	0	X	X	Math CoProcessor not selected
1	0	1	0	0	CW or SW read from Math CoProcessor
1	0	1	0	1	Opcode write to Math CoProcessor
1	0	1	1	0	Data read from Math CoProcessor
1	0	1	1	1	Data write to Math CoProcessor

4.4.1 INTEL387 SX MATH COPROCESSOR ADDRESSING

The NPS1#, NPS2, and CMD0 signals allow the Math CoProcessor to identify which bus cycles are intended for the Math CoProcessor. The Math CoProcessor responds to I/O cycles when the I/O address is 8000F8h, 8000FCh, and 8000FEh (treated as 8000FCh). The Math CoProcessor responds to I/O cycles when bit 23 of the I/O address is set. In other words, the Math CoProcessor acts as an I/O device in a reserved I/O address space.

Because A23 is used to select the Intel387 SX Math CoProcessor for data transfers, it is not possible for a program running on the CPU to address the Math CoProcessor with an I/O instruction. Only ESC instructions cause the CPU to communicate with the Math CoProcessor.

4.4.2 CPU/MATH COPROCESSOR SYNCHRONIZATION

The pins BUSY#, PEREQ, and ERROR# are used for various aspects of synchronization between the CPU and the Math CoProcessor.

BUSY# is used to synchronize instruction transfer from the CPU to the Math CoProcessor. When the Math CoProcessor recognizes an ESC instruction it asserts BUSY#. For most ESC instructions, the CPU waits for the Math CoProcessor to deassert BUSY# before sending the new opcode.

The Math CoProcessor uses the PEREQ pin of the CPU to signal that the Math CoProcessor is ready for data transfer to or from its data FIFO. The Math CoProcessor does not directly access memory; rather, the CPU provides memory access services for the Math CoProcessor. (For this reason, memory access on behalf of the Math CoProcessor always obeys the protection rules applicable to the current CPU mode.) Once the CPU initiates an Math CoProcessor instruction that has operands, the CPU waits for PEREQ signals that indicate when the Math CoProcessor is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the CPU continues program execution while the Math CoProcessor executes the ESC instruction.

In 8087/8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In the Intel386 Microprocessor and Intel387 Math CoProcessor systems, however, WAIT instructions are required only for operand synchronization; namely, after Math CoProcessor stores to memory (except FSTSW and FSTCW) or load from memory. (In 80286/80287 systems, WAIT is required before FLDENV and FRSTOR.) Used this way, WAIT ensures that the value has already been written or read by the Math CoProcessor before the CPU reads or changes the value.

Once it has started to execute a numerics instruction and has transferred and operands from the CPU, the Math CoProcessor can process the instruction in parallel with and independent of the host CPU. When the Math CoProcessor detects an exception, it asserts the ERROR# signal, which causes a CPU interrupt.

4.4.3 SYNCHRONOUS/ASYNCHRONOUS MODES

The internal logic of the Math CoProcessor can operate either directly from the CPU clock (synchronous mode) or from a separate clock (asynchronous mode). The two configurations are distinguished by the CKM pin. In either case, the bus control logic (BCL) of the Math CoProcessor is synchronized with the CPU clock. Use of asynchronous mode allows the BCL and the FPU section of the Math CoProcessor to run at different speeds. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. Use of synchronous mode eliminates one clock generator from the board design. The internal Power Management Unit of the Intel387 SX Math CoProcessor is disabled in asynchronous mode.

4.4.4 AUTOMATIC BUS CYCLE TERMINATION

In configurations where no extra wait states are required, READYO# can drive the CPU's READY# input and the Math CoProcessors READY# input. If wait states are required, this pin should be connected to the logic that ORs all READY outputs from peripheral devices on the CPU bus. READYO# is asserted by the Math CoProcessor only during I/O cycles that select the Math CoProcessor. Refer to Section 5.0 Bus Operation for details.

5.0 BUS OPERATION

With respect to bus interface, the Intel387 SX Math CoProcessor is fully synchronous with the CPU. Both operate at the same rate because each generates its internal CLK signal by dividing CPUCLK2 by two. Furthermore, both internal CLK signals are in phase, because they are synchronized by the same RESETIN signal.

A bus cycle for the Math CoProcessor starts when the CPU activates ADS# and drives new values on the address and cycle definition lines (W/R#, M/IO#, etc.). The Math CoProcessor examines the address and cycle definition lines in the same CLK period during which ADS# is activated. This CLK period is considered the first CLK of the bus cycle.

During this first CLK period, the Math CoProcessor also examines the W/R# input signal to determine whether the cycle is a read or a write cycle and examines the CMD0# input to determine whether an opcode, operand, or control/status register transfer is to occur.

The Intel387 SX Math CoProcessor supports both pipelined (i.e., overlapped) and non-pipelined bus cycles. A non-pipelined cycle is one for which the CPU asserts ADS# when no other bus cycle is in progress. A pipelined bus cycle is one for which the CPU asserts ADS# and provides valid next address and control signals before the prior Math CoProcessor cycle terminates. The CPU may do this as early as the second CLK period after asserting ADS# for the prior cycle. Pipelining increases the availability of the bus by at least one CLK period. The Intel387 SX Math CoProcessor supports pipelined bus cycles in order to optimize address pipelining by the CPU for memory cycles.

Bus operation is described in terms of an abstract state machine. Figure 5-1 illustrates the states and state transitions for Math CoProcessor bus cycles:

- T_I is the idle state. This is the state of the bus logic after RESET, the state to which bus logic returns after every non-pipelined bus cycle, and the state to which bus logic returns after a series of pipelined cycles.
- T_{RS} is the READY# sensitive state. Different types of bus cycles may require a minimum of one or two successive T_{RS} states. The bus logic remains in T_{RS} state until READY# is sensed, at which point the bus cycle terminates. Any number of wait states may be implemented by delaying READY#, thereby causing additional successive T_{RS} states.
- T_P is the first state for every pipelined bus cycle. This state is not used by non-pipelined cycles.

Note that the bus logic tracks bus state regardless of the values on the chip/port select pins. The

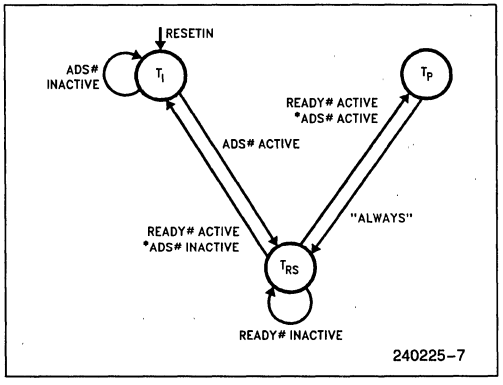


Figure 5-1. Bus State Diagram

READYO# output of the Math CoProcessor indicates when a Math CoProcessor bus cycle may be terminated if no extra wait states are required. For all write cycles (except those for the instructions FLDENV and FRSTOR), READYO# is always asserted during the first T_{RS} state, regardless of the number of wait states. For all read cycles (and write cycles for FLDENV and FRSTOR), READY# is always asserted in the second T_{RS} state, regardless of the number of wait states. These rules apply to both pipelined and non-pipelined cycles. Systems designers may use READYO# in one of the following ways:

1. Connect it (directly or through logic that ORs READY# signals from other devices) to the READY# inputs of the CPU and Math CoProcessor.
2. Use it as one input to a wait-state generator.

The following sections illustrate different types of Intel387 SX Math CoProcessor bus cycles. Because different instructions have different amounts of overhead before, between, and after operand transfer cycles, it is not possible to represent in a few diagrams all of the combinations of successive operand transfer cycles. The following bus cycle diagrams show memory cycles between Math CoProcessor operand transfer cycles. Note however that, during FRSTOR, some consecutive accesses to the Math CoProcessor do not have intervening memory accesses. For the timing relationship between operand transfer cycles and opcode write or other overhead activities, see Figure 7-7 "Other Parameters".

5.1 Non-Pipelined Bus Cycles

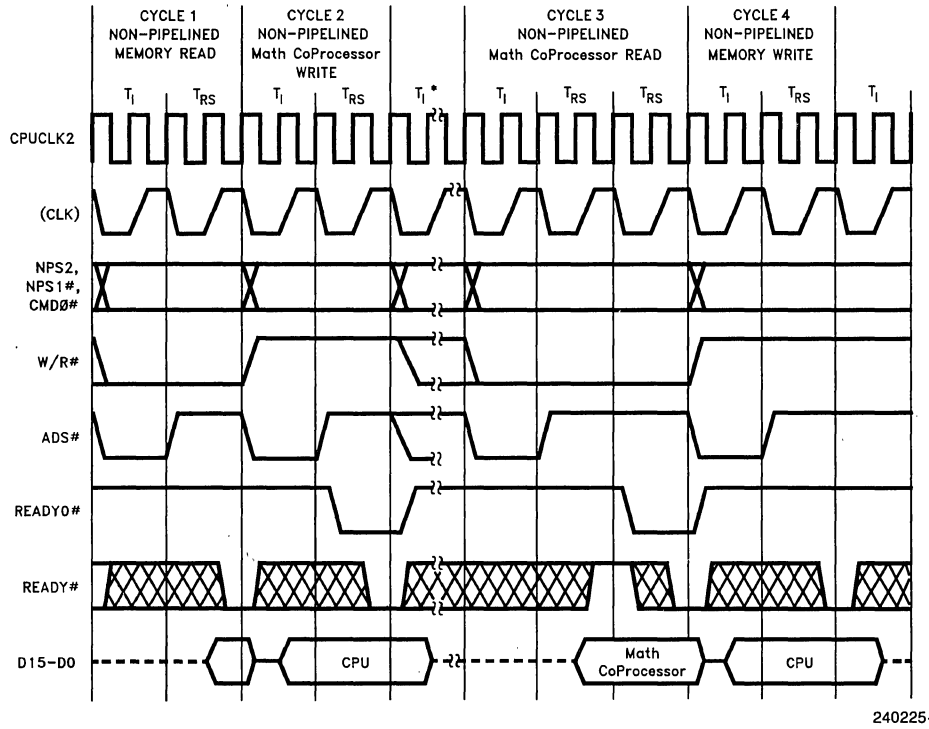
Figure 5-2 illustrates bus activity for consecutive non-pipelined bus cycles.

At the second clock of the bus cycle, the Math CoProcessor enters the T_{RS} state. During this state, it samples the READY# input and stays in this state as long as READY# is inactive.

5.1.1 WRITE CYCLE

In write cycles, the Math CoProcessor drives the READYO# signal for one CLK period during the second CLK period of the cycle (i.e., the first T_{RS} state); therefore, the fastest write cycle takes two CLK periods (see cycle 2 of Figure 5-2). For the instructions FLDENV and FRSTOR, however, the Math CoProcessor forces wait state by delaying the activation of READYO# to the second T_{RS} state (not shown in Figure 5-2).

The Math CoProcessor samples the D15–D0 inputs into data latches at the falling edge of CLK as long as it stays in T_{RS} state.



Cycles 1 & 2 represent part of the operand transfer cycle for instructions involving either 4-byte or 8-byte operand loads.
Cycles 3 & 4 represent part of the operand transfer cycle for a store operation.
*Cycles 1 & 2 could repeat here or T_I states for various non-operand transfer cycles and overhead.

Figure 5-2. Non-Pipelined Read and Write Cycles

When READY# is asserted, the Math CoProcessor returns to the idle state. Simultaneously with the Math CoProcessor entering the idle state, the CPU may assert ADS# again, signaling the beginning of yet another cycle.

5.1.2 READ CYCLE

At the rising edge of CLK in the second CLK period of the cycle (i.e., the first T_{RS} state), the Math CoProcessor starts to drive the D15–D0 outputs and continues to drive them as long as it stays in T_{RS} state.

At least one wait state must be inserted to ensure that the CPU latches the correct data. Because the Math CoProcessor starts driving the data bus only at the rising edge of CLK in the second clock period of the bus cycle, not enough time is left for the data signals to propagate and be latched by the CPU before the next falling edge of CLK. Therefore, the Math CoProcessor does not drive the READYO# signal until the third CLK period of the cycle. Thus, if the READYO# output drives the CPU's READY# input, one wait state is automatically inserted.

Because one wait state is required for Math CoProcessor reads, the minimum length of a Math CoProcessor read cycle is three CLK periods, as cycle 3 of Figure 5-2 shows.

When READY# is asserted, the Math CoProcessor returns to the idle state. Simultaneously with the Math CoProcessor's entering the idle state, the CPU may assert ADS# again, signaling the beginning of yet another cycle. The transition from T_{RS} state to idle state causes the Math CoProcessor to put the D15–D0 outputs into the floating state, allowing another device to drive the data bus.

5.2 Pipelined Bus Cycles

Because all the activities of the Math CoProcessor bus interface occur either during the T_{RS} state or

during the transitions to or from that state, the only difference between a pipelined and a non-pipelined cycle is the manner of changing from one state to another. The exact activities during each state are detailed in the previous section "Non-pipelined Bus Cycles".

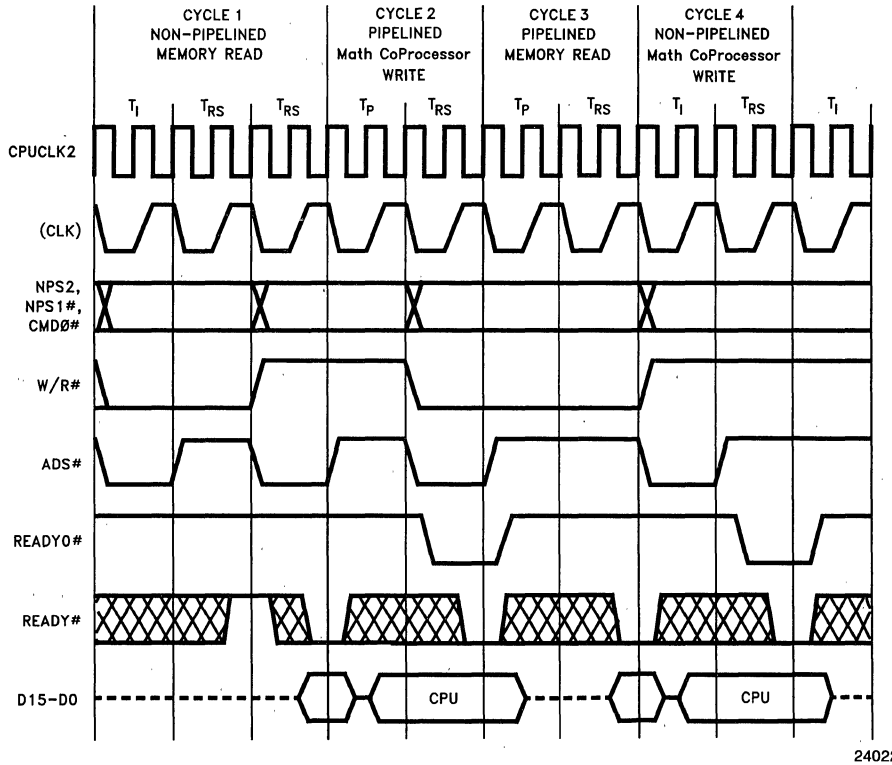
When the CPU asserts ADS# before the end of a bus cycle, both ADS# and READY# are active during a T_{RS} state. This condition causes the Math CoProcessor to change to a different state named T_P. One clock period after a T_P state, the Math CoProcessor always returns to the T_{RS} state. In consecutive pipelined cycles, the Math CoProcessor bus logic uses only the T_{RS} and T_P states.

Figure 5-3 shows the fastest transitions into and out of the pipelined bus cycles. Cycle 1 in the figure represents a non-pipelined cycle. (Non-pipelined write are always followed by another non-pipelined cycle, because READY# is asserted before the earliest possible assertion of ADS# for the next cycle.)

Figure 5-4 shows pipelined write and read cycles with one additional T_{RS} state beyond the minimum required. To delay the assertion of READY# requires external logic.

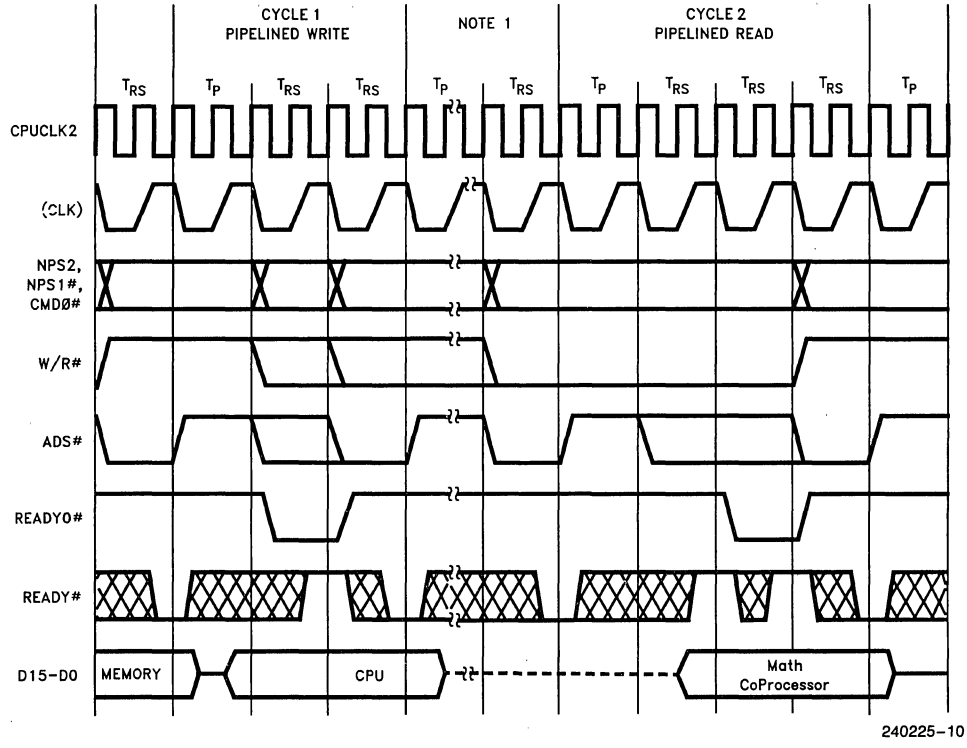
5.3 Mixed Bus Cycles

When the Math CoProcessor bus logic is in the T_{RS} state, it distinguishes between non-pipelined and pipelined cycles according to the behavior of ADS# and READY#. In a non-pipelined cycle, only READY# is activated, and the transition is from the T_{RS} state to the idle state. In a pipelined cycle, both READY# and ADS# are active, and the transition is first from T_{RS} state to T_P state, then, after one clock period, back to T_{RS} state.



Cycle 1–Cycle 4 represent the operand transfer cycle for an instruction involving a transfer of two 32-bit loads in total. The opcode write cycles and other overhead are not shown.
Note that the next cycle will be a pipelined cycle if both READY# and ADS# are sampled active at the end of a T_{RS} state of the current cycle.

Figure 5-3. Fastest Transitions to and from Pipelined Cycles

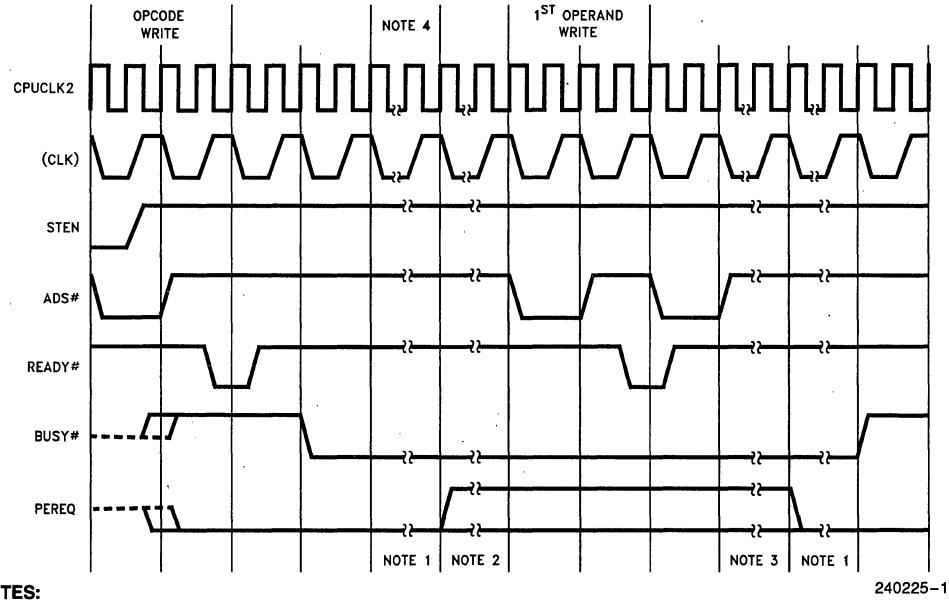


NOTE:
1. Cycles between operand write to the Math CoProcessor and storing result.

Figure 5-4. Pipelined Cycles with Wait States

5.4 BUSY# and PEREQ Timing Relationship

Figure 5-5 shows the activation of BUSY# at the beginning of instruction execution and its deactivation upon completion of the instruction. PEREQ is activated within this interval. If ERROR# is ever asserted, it would be asserted at least six CPUCLK2 periods after the deactivation of PEREQ and would be deasserted at least six CPUCLK2 periods before the deactivation of BUSY#.



NOTES:
1. Instruction dependent.
2. PEREQ is an asynchronous input to the Intel386™ Microprocessor; it may not be asserted (instruction dependent).
3. More operand transfers.
4. Memory read (operand) cycle is not shown.

Figure 5-5. STEN, BUSY#, and PEREQ Timing Relationships

6.0 PACKAGE SPECIFICATIONS

6.1 Mechanical Specifications

The Intel387 SX Math CoProcessor is packaged in a 68-pin PLCC package. Detailed mechanical specifications can be found in the Intel Packaging Specification, Order Number 231369.

6.2 Thermal Specifications

The Intel387 SX Math CoProcessor is specified for operation when the case temperature is within the range of 0°C to 100°C. The case temperature (T_C) may be measured in any environment to determine whether the Intel387 SX Math CoProcessor is within the specified operating range. The case temperature should be measured at the center of the top surface.

The ambient temperature (T_A) is guaranteed as long as T_C is not violated. The ambient temperature can be calculated from the θ_{JC} (thermal resistance constant from the transistor junction to the case) and θ_{JA} (thermal resistance from junction to ambient) from the following calculations:

Junction Temperature $T_J = T_C + P^*\theta_{JC}$

Ambient Temperature $T_A = T_J - P^*\theta_{JA}$

Case Temperature $T_C = T_A + P^* (\theta_{JA} - \theta_{JC})$

Values for θ_{JA} and θ_{JC} are given in Table 6-1 for the 68 pin PLCC package. θ_{JC} is given at various airflows. Table 6-2 shows the maximum T_A allowable without exceeding T_C at various airflows. Note that T_A can be improved further by attaching a heat sink to the package. P is calculated by using the maximum hot I_{CC} and maximum V_{CC}.

Table 6-1. Thermal Resistances (°C/Watt) θ_{JC} and θ_{JA}

Package	θ_{JC}	θ_{JA} versus Airflow - ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
68-Pin PLCC	8	30	25	20	15.5	13	12

Table 6-2. Maximum T_A at Various Airflows

Package		T_A (°C) versus Airflow - ft/min (m/sec)				
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
68-Pin PLCC	84.9	88.3	91.8	94.8	96.6	97.2

Maximum T_A is calculated at maximum V_{CC} and maximum I_{CC}.

7.0 ELECTRICAL CHARACTERISTICS

The following specifications represent the targets of the design effort. They are subject to change without notice. Contact your Intel representative to get the most up-to-date values.

7.1 Absolute Maximum Ratings*

Case Temperature T_C Under Bias . . . 0°C to +100°C

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

Voltage on Any Pin with Respect to Ground . . . . . −0.5 to V_{CC}+0.5

Power Dissipation 0.8W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

7.2 D.C. Characteristics

Table 7-1. D.C. Specifications T_C = 0°C to +100°C, V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LO Voltage	−0.3	+0.8	V	(Note 1)
V_{IH}	Input HI Voltage	2.0	V_{CC}+0.3	V	(Note 1)
V_{CL}	CPUCLK2 and NUMCLK2 Input LO Voltage	−0.3	+0.8	V	
V_{CH}	CPUCLK2 and NUMCLK2 Input HI Voltage	V_{CC}−0.8	V_{CC}+0.8	V	
V_{OL}	Output LO Voltage		0.45	V	(Note 2)
V_{OH}	Output HI Voltage	2.4		V	(Note 3)
V_{OH}	Output HI Voltage	V_{CC}−0.8		V	(Note 4)
I_{CC}	Power Supply Current Dynamic Mode				
	Freq. = 33 MHz(5)		150	mA	I_{CC} typ. = 135 mA
	Freq. = 25 MHz(5)		150	mA	I_{CC} typ. = 130 mA
	Freq. = 20 MHz(5)		125	mA	I_{CC} typ. = 110 mA
	Freq. = 16 MHz(5)		100	mA	I_{CC} typ. = 90 mA
	Freq. = 1 MHz(5)		20	mA	I_{CC} typ. = 5 mA
	Idle Mode(6)		7	mA	I_{CC} typ. = 4 mA
I_{LI}	Input Leakage Current		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{LO}	I/O Leakage Current		±15	μA	0.45V ≤ V_O ≤ V_{CC}
C_{IN}	Input Capacitance	7	10	pF	f_c = 1 MHz
C_O	I/O Capacitance	7	12	pF	f_c = 1 MHz
C_{CLK}	Clock Capacitance	7	20	pF	f_c = 1 MHz

NOTES:
1. This parameter is for all inputs, excluding the clock inputs.
2. This parameter is measured at I_{OL} as follows:
   Data = 4.0 mA
   READYO#, ERROR#, BUSY#, PEREQ = 25 mA
3. This parameter is measured at I_{OH} as follows:
   Data = 1.0 mA
   READYO#, ERROR#, BUSY#, PEREQ = 0.6 mA
4. This parameter is measured at I_{OH} as follows:
   Data = 0.2 mA
   READYO#, ERROR#, BUSY# PEREQ = 0.12 mA
5. Synchronous Clock Mode (CKM = 1). I_{CC} is measured at steady state, maximum capacitive loading on the outputs, and worst-case D.C. level at the inputs.
6. Intel387 SX Math CoProcessor Internal Idle Mode. Synchronous clock mode, clock and control inputs are active but the Math CoProcessor is not executing an instruction. Outputs driving CMOS inputs.

7.3 A.C. Characteristics

Table 7-2a. Timing Requirements of the Bus Interface Unit
T_C = 0°C to +100°C, V_{CC} = 5V ±10% (All measurements made at 1.5V unless otherwise specified)

Pin	Symbol	Parameter	16 MHz–25 MHz		33 MHz		Test Conditions	Refer to Figure
			Min (ns)	Max (ns)	Min (ns)	Max (ns)		
CPUCLK2	t1	Period	20	DC	15	DC	2.0V	7.2
CPUCLK2	t2a	High Time	6		6.25		2.0V	
CPUCLK2	t2b	High Time	3		4.5		V_{CC}–0.8V	
CPUCLK2	t3a	Low Time	6		6.25		2.0V	
CPUCLK2	t3b	Low Time	4		4.5		0.8V	
CPUCLK2	t4	Fall Time		7		4	From V_{CC}–0.8V to 0.8V	
CPUCLK2	t5	Rise Time		7		4	From 0.8V to V_{CC}–0.8V	
READYO#	t7a	Out Delay	4	25	4	17	C_L = 50 pF	7.3
PEREQ	t7b	Out Delay	4	23	4	21	C_L = 50 pF	
BUSY#	t7c	Out Delay	4	23	4	21	C_L = 50 pF	
ERROR#	t7d	Out Delay	4	23	4	23	C_L = 50 pF	
D15–D0	t8	Out Delay	1	45	0	37	C_L = 50 pF	7.4
D15–D0	t10	Setup Time	11		8			
D15–D0	t11	Hold Time	11		8			
D15–D0	t12*	Float Time	6	24	6	19		
READYO#	t13a*	Float Time	1	40	1	30		7.6
PEREQ	t13b*	Float Time	1	40	1	30		
BUSY#	t13c*	Float Time	1	40	1	30		
ERROR#	t13d*	Float Time	1	40	1	30		
ADS#	t14a	Setup Time	15		13			7.4
ADS#	t15a	Hold Time	4		4			
W/R#	t14b	Setup Time	15		13			
W/R#	t15b	Hold Time	4		4			
READY#	t16a	Setup Time	9		7			7.4
READY#	t17a	Hold Time	4		4			
CMD0#	t16b	Setup Time	16		13			
CMD0#	t17b	Hold Time	2		2			
NPS1#, NPS2	t16c	Setup Time	16		13			
NPS1#, NPS2	t17c	Hold Time	2		2			
STEN	t16d	Setup Time	15		13			
STEN	t17d	Hold Time	2		2			
RESETIN	t18	Setup Time	8		5			7.5
RESETIN	t19	Hold Time	3		2			

NOTE:
*Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not tested.

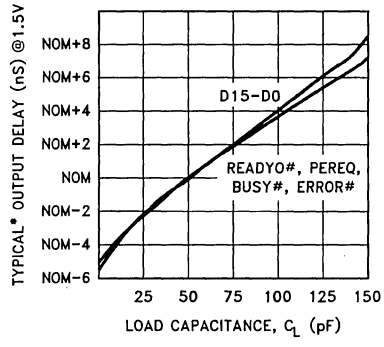
Table 7-2b. Timing Requirements of the Execution Unit (Asynchronous Mode CKM = 0)

Pin	Symbol	Parameter	16 MHz–25 MHz		33 MHz		Test Conditions	Refer to Figure
			Min (ns)	Max (ns)	Min (ns)	Max (ns)		
NUMCLK2	t1	Period	20	500	15	500	2.0V	7.2
NUMCLK2	t2a	High Time	6		6.25		2.0V	
NUMCLK2	t2b	High Time	3		4.5		$V_{CC}-0.8V$	
NUMCLK2	t3a	Low Time	6		6.25		2.0V	
NUMCLK2	t3b	Low Time	4		4.5		0.8V	
NUMCLK2	t4	Fall Time		7		6	From $V_{CC}-0.8V$ to 0.8V	
NUMCLK2	t5	Rise Time		7		6	From 0.8V to $V_{CC}-0.8V$	
NUMCLK2/ CPUCLK2		Ratio	10/16	14/10	10/16	14/10		

NOTE:
If not used (CKM = 1) tie NUMCLK2 low.

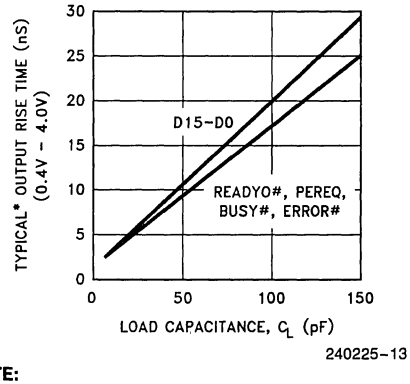
Table 7-2c. Other A.C. Parameters

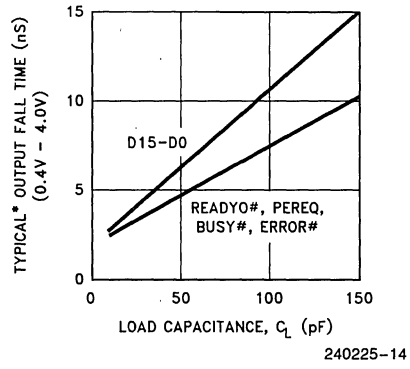
Pin	Symbol	Parameter	Min	Max	Units
RESETIN	t30	Duration	40		NUMCLK2
RESETIN	t31	RESETIN Inactive to 1st Opcode Write	50		NUMCLK2
BUSY#	t32	Duration	6		CPUCLK2
BUSY#, ERROR#	t33	ERROR# (In)Active to BUSY# Inactive	6		CPUCLK2
PEREQ, ERROR#	t34	PEREQ Inactive to ERROR# Active	6		CPUCLK2
READY#, BUSY#	t35	READY# Active to BUSY# Active	0	4	CPUCLK2
READY#	t36	Minimum Time from Opcode Write to Opcode/Operand Write	4		CPUCLK2
READY#	t37	Minimum Time from Operand Write to Operand Write	4		CPUCLK2



NOTE:
*Typical part under worst-case conditions.

Figure 7-1a. Typical Output Valid Delay vs Load Capacitance at Max Operating Temperature





NOTE:
*Typical part under worst-case conditions.

Figure 7-1b. Typical Output Slew Time vs Load Capacitance at Max Operating Temperature

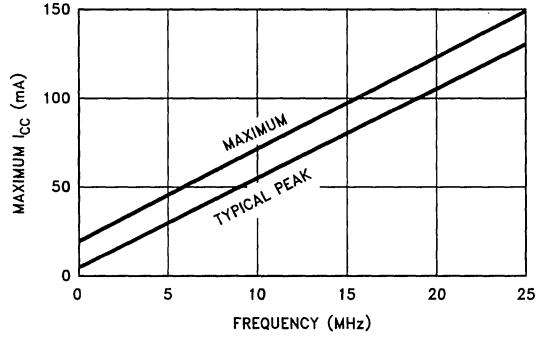


Figure 7-1c. Maximum I_{CC} vs Frequency

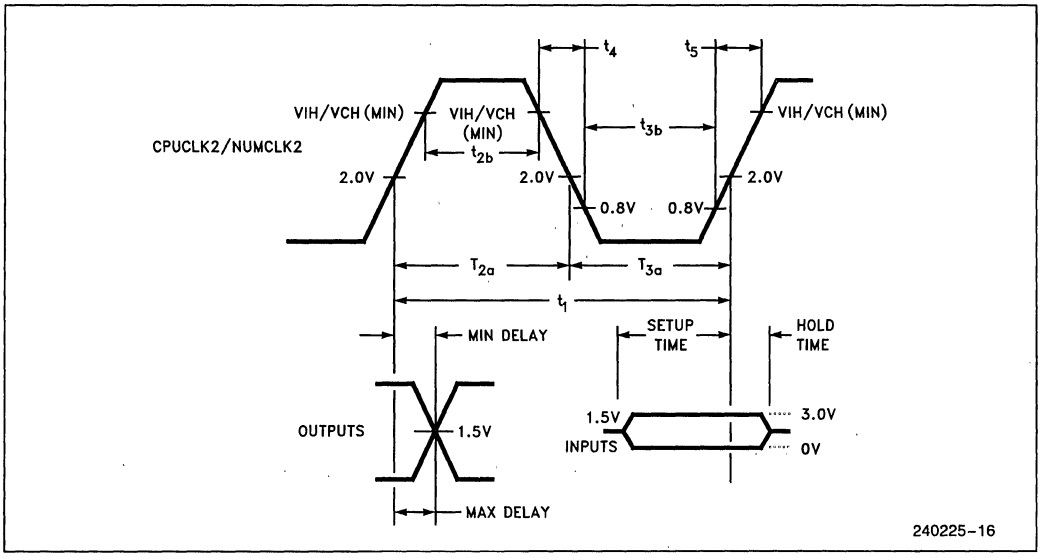


Figure 7-2. CPUCLK2/NUMCLK2 Waveform and Measurement Points for Input/Output

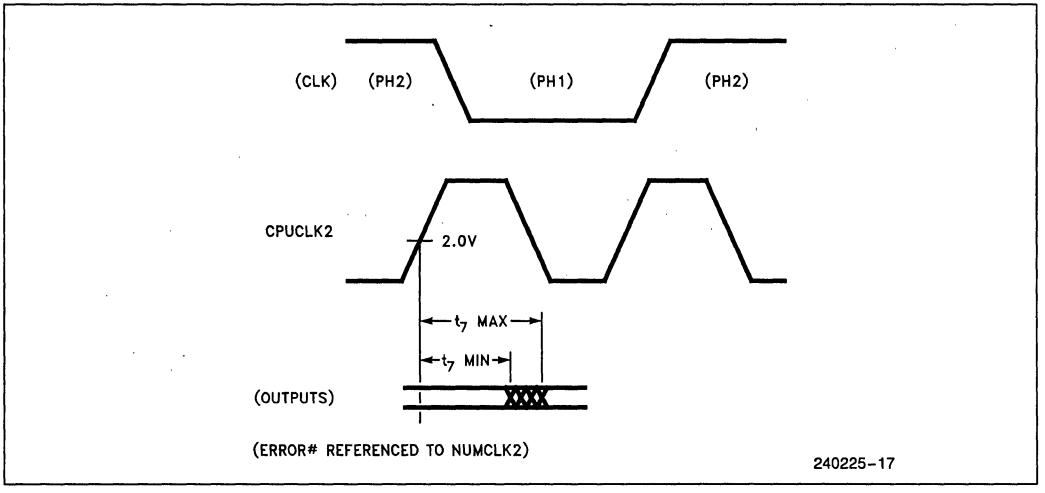


Figure 7-3. Output Signals

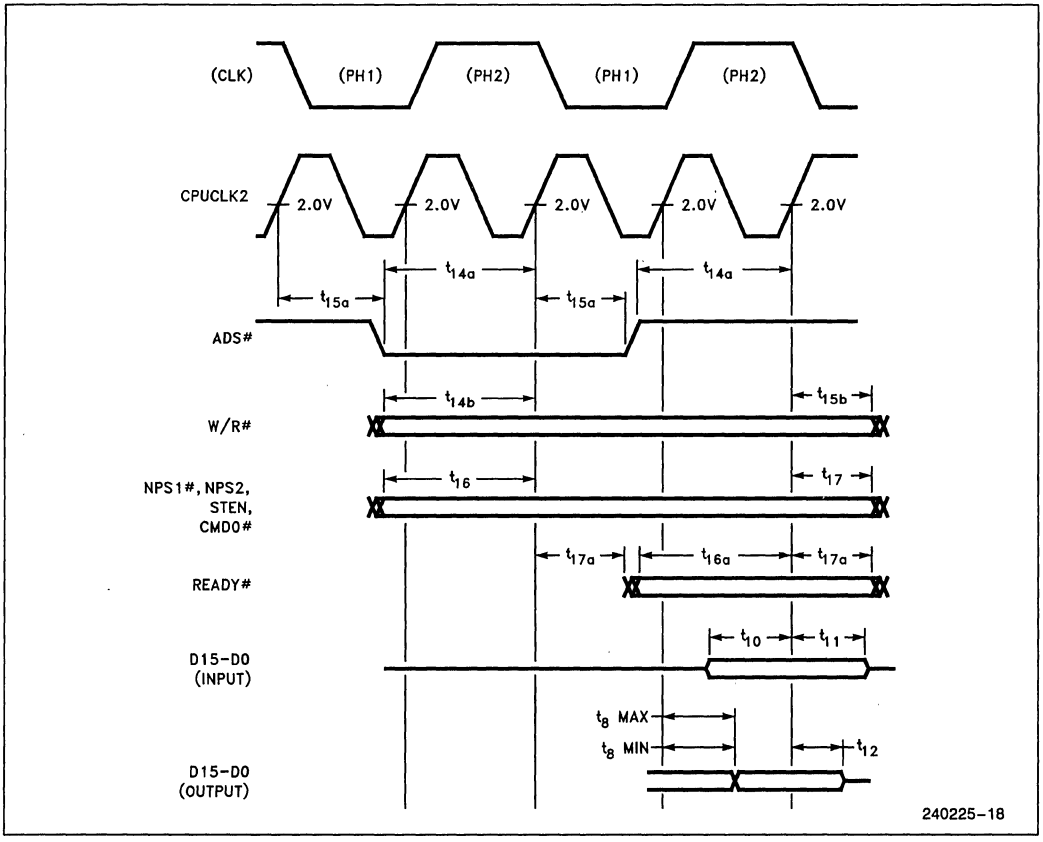
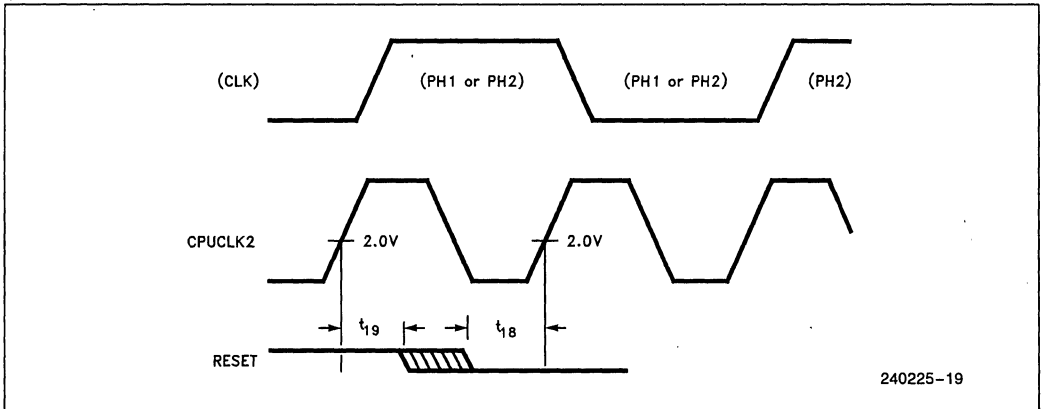


Figure 7-4. Input and I/O Signals



NOTE:
The second internal processor phase following RESET high to low transition is PH2.

Figure 7-5. RESET Signal

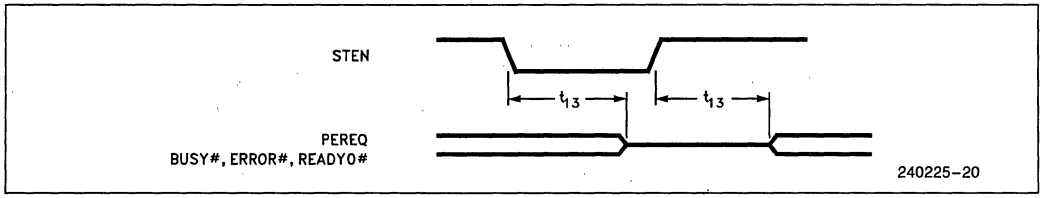
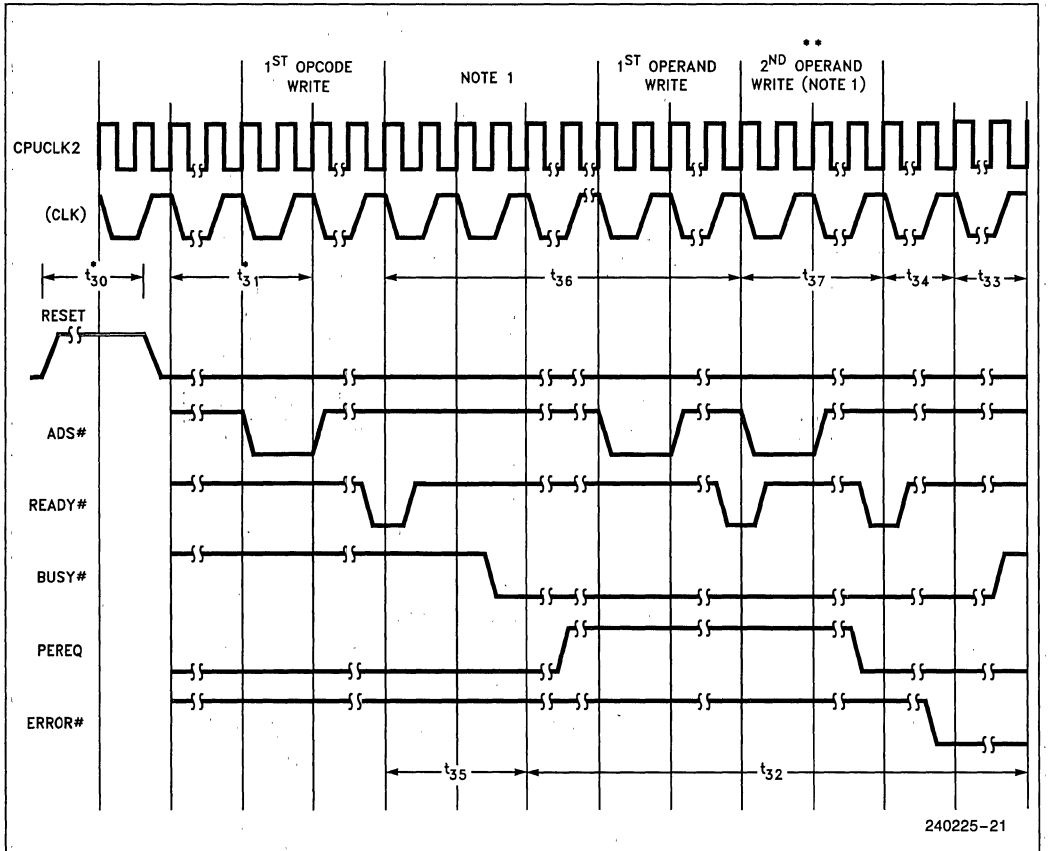


Figure 7-6. Float from STEN



*In NUMCLK2's
**or last operand

NOTE:
1. Memory read (operand) cycle is not shown.

Figure 7-7. Other Parameters

8.0 INTEL387 SX MATH COPROCESSOR INSTRUCTION SET

Instructions for the Intel387 SX Math CoProcessor assume one of the five forms shown in Table 8-1. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the CPU's addressing modes.

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of CPU instructions (refer to Programmer's Reference Manual for the CPU). SIB (Scale Index Base) byte and DISP (displacement) are optionally present in instructions that have MOD and R/M fields. Their presence depends on the values of MOD and R/M, as for instructions of the CPU.

The instruction summaries that follow in Table 8-2 assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD requests delaying processor access to the bus; and that no exceptions are detected during instruction execution. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

Table 8-1. Instruction Formats

	Instruction								Optional Fields	
	First Byte				Second Byte					
1	11011	OPA		1	MOD		1	OPB R/M	SIB	DISP
2	11011	MF		OPA	MOD		OPB*	R/M	SIB	DISP
3	11011	d	P	OPA	1	1	OPB*	ST(i)		
4	11011	0	0	1	1	1	1	OP		
5	11011	0	1	1	1	1	1	OP		
	15–11	10	9	8	7	6	5	4 3 2 1 0		

OP = Instruction opcode, possibly split into two fields OPA and OPB
MF = Memory Format
- 00 - 32-bit real
- 01 - 32-bit integer
- 10 - 64-bit real
- 11 - 16-bit integer

d = Destination
- 0 - Destination is ST(0)
- 1 - Destination is ST(i)

R XOR d = 0 - Destination (op) Source
R XOR d = 1 - Source (op) Destination
*In FSUB and FDIV, the low-order bit of OPB is the R (reversed) bit
P = POP
- 0 - Do not pop stack
- 1 - Pop stack after operation

ESC = 11011
ST(i) = Register stack element i
- 000 = Stack top
- 001 = Second stack element
- •
- •
- •
- 111 = Eighth stack element

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2-6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
DATA TRANSFER							
FLD = Load[a]							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	SIB/DISP	11-20	28-44	20-27	42-53
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	SIB/DISP		30-58		
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	SIB/DISP		16-47		
BCD memory to ST(0)	ESC 111	MOD 100 R/M	SIB/DISP		49-101		
ST(i) to ST(0)	ESC 001	11000 ST(i)			7-12		
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	SIB/DISP	27-45	59-78	59	58-76
ST(0) to ST(i)	ESC 101	11010 ST(i)			7-11		
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	SIB/DISP	27-45	59-78	59	58-76
ST(0) to long integer memory	ESC 111	MOD 111 R/M	SIB/DISP		64-86		
ST(0) to extended real memory	ESC 011	MOD 111 R/M	SIB/DISP		50-56		
ST(0) to BCD memory	ESC 111	MOD 110 R/M	SIB/DISP		116-194		
ST(0) to ST(i)	ESC 101	11011 ST (i)			7-11		
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)			10-17		
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	SIB/DISP	15-27	36-54	18-31	39-62
ST(i) to ST(0)	ESC 000	11010 ST(i)			13-21		
FCOMP = Compare and pop							
Integer/real memory to ST(0)	ESC MF 0	MOD 011 R/M	SIB/DISP	15-27	36-54	18-31	39-62
ST(i) to ST(0)	ESC 000	11011 ST(i)			13-21		
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001			13-21		
FTST = Test ST(0)	ESC 001	1110 0100			17-25		
FUCOM = Unordered compare	ESC 101	11100 ST(i)			13-21		
FUCOMP = Unordered compare and pop	ESC 101	11101 ST(i)			13-21		
FUCOMPP = Unordered compare and pop twice	ESC 010	1110 1001			13-21		
FXAM = Examine ST(0)	ESC 001	1110 0101			24-37		

Shaded areas indicate instructions not available in 8087/80287.

NOTE:
a. When loading single or double precision zero from memory, add 5 clocks.

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2–6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
ARITHMETIC							
FADD = Add							
Integer/real memory to ST(0)	ESC MF 0	MOD 000 R/M	SIB/DISP	14–31	36–58	19–38	38–64
ST(i) and ST(0)	ESC d P 0	11000 ST(i)	SIB/DISP	12–26[b]			
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	SIB/DISP	14–31	36–58	19–38	38–64[c]
ST(i) to ST(0)	ESC d P 0	1110 R R/M		12–26[d]			
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	SIB/DISP	21–33	45–73	27–57	46–74
ST(i) and ST(0)	ESC d P 0	1100 1 R/M		17–50[e]			
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	SIB/DISP	79–87	103–116[f]	85–95	105–124[g]
ST(i) and ST(0)	ESC d P 0	1111 R R/M		77–80[h]			
FSQRT[i] = Square root	ESC 001	1111 1010		97–111			
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101		44–82			
FPREM = Partial remainder	ESC 001	1111 1000		56–140			
FPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101		81–168			
FRNDINT = Round ST(0) to integer	ESC 001	1111 1100		41–62			
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100		42–63			
FABS = Absolute value of ST(0)	ESC 001	1110 0001		14–21			
FCHS = Change sign of ST(0)	ESC 001	1110 0000		17–24			
TRANSCENDENTAL							
FCOS[k] = Cosine of ST(0)	ESC 001	1111 1111		122–680			
FPTAN[k] = Partial tangent of ST(0)	ESC 001	1111 0010		162–430[j]			
FPATAN = Partial arctangent of ST(0)	ESC 001	1111 0011		250–420			
FSIN[k] = Sine of ST(0)	ESC 001	1111 1110		121–680			
FSINCOS[k] = Sine and cosine of ST(0)	ESC 001	1111 1011		150–650			
F2XM1[l] = $2^{ST(0)} - 1$	ESC 001	1111 0000		167–410			
FYL2X[m] = ST(1) * $\log_2$ST(0)	ESC 001	1111 0001		99–436			
FYL2XP1[n] = ST(1) * $\log_2$[ST(0) + 1.0]	ESC 001	1111 1001		210–447			

Shaded areas indicate instructions not available in 8087/80287.

NOTES:
b. Add 3 clocks to the range when d = 1.
c. Add 1 clock to each range when R = 1.
d. Add 3 clocks to the range when d = 0.
e. typical = 52 (When d = 0, 46–54, typical = 49).
f. Add 1 clock to the range when R = 1.
g. 135–141 when R = 1.
h. Add 3 clocks to the range when d = 1.
i. $-0 \leq ST(0) \leq +\infty$.
j. These timings hold for operands in the range $|x| < \pi$. For operands not in this range, up to 76 additional clocks may be needed to reduce the operand.
k. $0 \leq ST(0) < 2^{63}$.
l. $-1.0 \leq ST(0) \leq 1.0$.
m. $0 \leq ST(0) < \infty$, $-\infty < ST(1) < +\infty$.
n. $0 \leq |ST(0)| < [2\text{-SQRT}(2)]/2$, $-\infty < ST(1) < +\infty$.

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2-6	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
CONSTANTS							
FLDZ = Load +0.0 to ST(0)	ESC 001	1110 1110			10-17		
FLD1 = Load +1.0 to ST(0)	ESC 001	1110 1000			15-22		
FLDPI = Load π to ST(0)	ESC 001	1110 1011			26-36		
FLDL2T = Load $\log_2(10)$ to ST(0)	ESC 001	1110 1001			26-36		
FLDL2E = Load $\log_2(e)$ to ST(0)	ESC 001	1110 1010			26-36		
FLDLG2 = Load $\log_{10}(2)$ to ST(0)	ESC 001	1110 1100			25-35		
FLDLN2 = Load $\log_e(2)$ to ST(0)	ESC 001	1110 1101			26-38		
PROCESSOR CONTROL							
FINIT = Initialize Math CoProcessor	ESC 011	1110 0011			33		
FLDCW = Load control word from memory	ESC 001	MOD 101 R/M	SIB/DISP		19		
FSTCW = Store control word to memory	ESC 001	MOD 111 R/M	SIB/DISP		15		
FSTSW = Store status word to memory	ESC 101	MOD 111 R/M	SIB/DISP		15		
FSTSW AX = Store status word to AX	ESC 111	1110 0000			13		
FCLEX = Clear exceptions	ESC 011	1110 0010			11		
FSTENV = Store environment	ESC 001	MOD 110 R/M	SIB/DISP		117-118		
FLDENV = Load environment	ESC 001	MOD 100 R/M	SIB/DISP		85		
FSAVE = Save state	ESC 101	MOD 110 R/M	SIB/DISP		402-403		
FRSTOR = Restore state	ESC 101	MOD 100 R/M	SIB/DISP		415		
FINCSTP = Increment stack pointer	ESC 001	1111 0111			21		
FDECSTP = Decrement stack pointer	ESC 001	1111 0110			22		
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)			18		
FNOP = No operations	ESC 001	1101 0000			12		

APPENDIX A
INTEL387 SX MATH COPROCESSOR COMPATIBILITY

A.1 8087/80287 Compatibility

This section summarizes the differences between the Intel387 SX Math CoProcessor and the 80287 Math CoProcessor. Any migration from the 8087 directly to the Intel387 SX Math CoProcessor must also take into account the differences between the 8087 and the 80287 Math CoProcessor as listed in Appendix B.

Many changes have been designed into the Intel387 SX Math CoProcessor to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

A.1.1 GENERAL DIFFERENCES

The Intel387 SX Math CoProcessor supports only affine closure for infinity arithmetic, not projective closure.

Operands for FSCALE and FPATAN are no longer restricted in range (except for $\pm\infty$); F2XM1 and FPTAN accept a wider range of operands.

Rounding control is in effect for FLD constant.

Software cannot change entries of the tag word to values (other than empty) that differ from actual register contents.

After reset, FINIT, and incomplete FPREM, the Intel387 SX Math CoProcessor resets to zero the condition code bits C_3–C_0 of the status word.

In conformance with the IEEE standard, the Intel387 SX Math CoProcessor does not support the special data formats pseudo-zero, pseudo-NaN, pseudo-infinity, and unnormal.

The denormal exception has a different purpose on the Intel387 SX Math CoProcessor. A system that uses the denormal exception handler solely to normalize the denormal operands, would better mask the denormal exception on the Intel387 SX Math CoProcessor. The Intel387 SX Math CoProcessor automatically normalizes denormal operands when the denormal exception is masked.

A.1.2 EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the Intel387 SX Math CoProcessor:

1. When the overflow or underflow exception is masked, the Intel387 SX Math CoProcessor differs from the 80287 in rounding when overflow or underflow occurs. The Intel387 SX Math CoProcessor produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the Intel387 SX Math CoProcessor sets its underflow flag only if there is also a loss of accuracy during denormalization.
3. Fewer invalid-operations exceptions due to denormal operand, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
4. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
5. The denormal exception can occur during the transcendental instruction and the FXTRACT instruction.
6. The denormal exception no longer takes precedence over all other exceptions.
7. When the denormal exception is masked, the Intel387 SX Math CoProcessor automatically normalizes denormal operands. The 8087/80287 performs unnormal arithmetic, which might produce an unnormal result.
8. When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
9. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
10. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
11. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormal operand exception. When loading a signaling NaN, FLD *single/double precision* signals an invalid-operation exception.
12. The Intel387 SX Math CoProcessor only generates quiet NaNs (as on the 80287); however, the Intel387 SX Math CoProcessor distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
13. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 80287/8087 leaves the original operand in ST(1) intact.
14. When the scaling factor is $\pm\infty$, the FSCALE instruction behaves as follows:
    - FSCALE (0, ∞) generates the invalid operation exception.
    - FSCALE (finite, $-\infty$) generates zero with the same sign as the scaled operand.
    - FSCALE (finite, $+\infty$) generates ∞ with the same sign as the scaled operand.

    The 8087/80287 returns zero in the first case and raises the invalid-operation exception in the other cases.
15. The Intel387 SX Math CoProcessor returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 and 80287 support a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

APPENDIX B
COMPATIBILITY BETWEEN THE 80287 AND 8087 MATH COPROCESSOR

The 80286/80287 operating in Real Address mode will execute 8086/8087 programs without major modification. However, because of differences in the handling of numeric exceptions by the 80287 Math CoProcessor and the 8087 Math CoProcessor, exception handling routines *may* need to be changed. This appendix summarizes the differences between the 80287 Math CoProcessor and the 8087 Math CoProcessor, and provides details showing how 8087/8087 programs can be ported to the 80286/80287.

1. The Math CoProcessor signals exceptions through a dedicated ERROR# line to the 80286. The Math CoProcessor error signal does not pass through an interrupt controller (the 8087 INT signal does). Therefore, any interrupt controller oriented instructions in numeric exception handlers for the 8086/8087 should be deleted.
2. The 8087 instructions FENI and FDISI perform no useful function in the 80287. If the 80287 encounters one of these opcodes in its instruction stream, the instruction will effectively be ignored; none of the 80287 internal states will be updated. While 8086/8087 programs containing the instruction may be executed on the 80286/80287, it is unlikely that the exception handling routines containing these instructions will be completely portable to the 80287.
3. Interrupt vector 16 must point to the numeric exception handling routine.
4. The ESC instruction address saved in the 80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
5. In Protected Address mode, the format of the 80287's saved instruction and address pointers is different than for the 8087. The instruction opcode is not saved in Protected mode; exception handlers will have to retrieve the opcode from memory if needed.
6. Interrupt 7 will occur in the 80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the 80286 MSW set (TS = 1 or EM = 1). It TS is set, then a WAIT instruction will also cause interrupt 7. An exception handler should be included in 80286/80287 code to handle these situations.
7. Interrupt 9 will occur if the second or subsequent words of a floating point operand fall outside a segment's size. Interrupt 13 will occur if the starting address of a numeric operand falls outside a segment's size. An exception handler should be included in 80286/80287 code to report these programming errors.
8. Except for the processor control instructions, all of the 80287 numeric instructions are automatically synchronized by the 80286 CPU; the 80286 CPU automatically tests the BUSY# line from the 80287 to ensure that the 80287 has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the 8087 used witth 8086 and 8088 processors, explicit WAITs are required before each numeric instruction to ensure synchronization. Although 8086/8087 programs having explicit WAIT instructions will execute perfectly on the 80286/80287 without reassembly, these WAIT instructions are unnecessary.
9. Since the 80287 does not require WAIT instructions before each numeric instruction, the ASM286 assembler does not automatically generate these WAIT instuctions. The ASM86 assembler, however, automatically precedes every ESC instruction with a WAIT instruction. Although numeric routines generated using the ASM86 assembler will generally execute correctly on the 80286/80287, reassembly using ASM286 may result in a more compact code image.

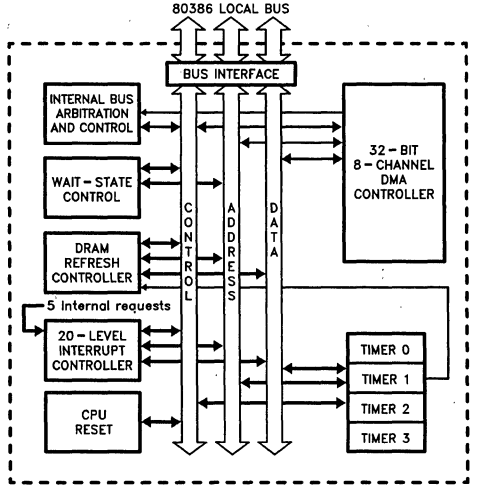
The processor control instructions for the 80287 may be coded using either a WAIT or No-WAIT form of mnemonic. The WAIT forms of these instructions cause ASM286 to precede the ESC instructions with a CPU WAIT instruction, in the identical manner as does ASM86.

82380
HIGH PERFORMANCE 32-BIT DMA CONTROLLER WITH INTEGRATED SYSTEM SUPPORT PERIPHERALS

- **High Performance 32-Bit DMA Controller**
  - **— 50 MBytes/sec Maximum Data Transfer Rate at 25 MHz**
  - **— 8 Independently Programmable Channels**
- **20-Source Interrupt Controller**
  - **— Individually Programmable Interrupt Vectors**
  - **— 15 External, 5 Internal Interrupts**
  - **— 82C59A Superset**
- **Four 16-Bit Programmable Interval Timers**
  - **— 82C54 Compatible**
- **Programmable Wait State Generator**
  - **— 0 to 15 Wait States Pipelined**
  - **— 1 to 16 Wait States Non-Pipelined**
- **DRAM Refresh Controller**
- **80386 Shutdown Detect and Reset Control**
  - **— Software/Hardware Reset**
- **High Speed CHMOS III Technology**
- **132-Pin PGA Package**
  (See Packaging Handbook Order 240800-001, Package Type A)
- **Optimized for use with the 80386 Microprocessor**
  - **— Resides on Local Bus for Maximum Bus Bandwidth**
  - **— 16, 20, and 25 MHz Clock**

The 82380 is a multi-function support peripheral that integrates system functions necessary in an 80386 environment. It has eight channels of high performance 32-bit DMA with the most efficient transfer rates possible on the 80386 bus. System support peripherals integrated into the 82380 provide Interrupt Control, Timers, Wait State generation, DRAM Refresh Control, and System Reset logic.

The 82380's DMA Controller can transfer data between devices of different data path widths using a single channel. Each DMA channel operates independently in any of several modes. Each channel has a temporary data storage register for handling non-aligned data without the need for external alignment logic.



82380 Internal Block Diagram

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

October 1994
Order Number: 290128-006

82370
INTEGRATED SYSTEM PERIPHERAL

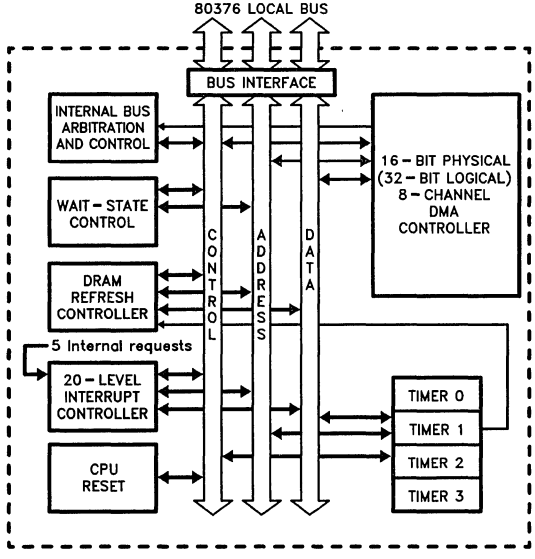
- **High Performance 32-Bit DMA Controller for 16-Bit Bus**
  - **— 16 MBytes/Sec Maximum Data Transfer Rate at 16 MHz**
  - **— 8 Independently Programmable Channels**
- **20-Source Interrupt Controller**
  - **— Individually Programmable Interrupt Vectors**
  - **— 15 External, 5 Internal Interrupts**
  - **— 82C59A Superset**
- **Four 16-Bit Programmable Interval Timers**
  - **— 82C54 Compatible**
- **Software Compatible to 82380**
- **Programmable Wait State Generator**
  - **— 0 to 15 Wait States Pipelined**
  - **— 1 to 16 Wait States Non-Pipelined**
- **DRAM Refresh Controller**
- **80376 Shutdown Detect and Reset Control**
  - **— Software/Hardware Reset**
- **High Speed CHMOS III Technology**
- **100-Pin Plastic Quad Flat-Pack Package and 132-Pin Pin Grid Array Package**

  (See Packaging Handbook Order #240800-001, Package Type NG or Package Type A)
- **Optimized for Use with the 80376 Microprocessor**
  - **— Resides on Local Bus for Maximum Bus Bandwidth**
  - **— 16 MHz Clock**

3

The 82370 is a multi-function support peripheral that integrates system functions necessary in an 80376 environment. It has eight channels of high performance 32-bit DMA (32-bit internal, 16-bit external) with the most efficient transfer rates possible on the 80376 bus. System support peripherals integrated into the 82370 provide Interrupt Control, Timers, Wait State generation, DRAM Refresh Control, and System Reset logic.

The 82370's DMA Controller can transfer data between devices of different data path widths using a single channel. Each DMA channel operates independently in any of several modes. Each channel has a temporary data storage register for handling non-aligned data without the need for external alignment logic.



290164-1

Internal Block Diagram

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

November 1994
Order Number: 290164-005

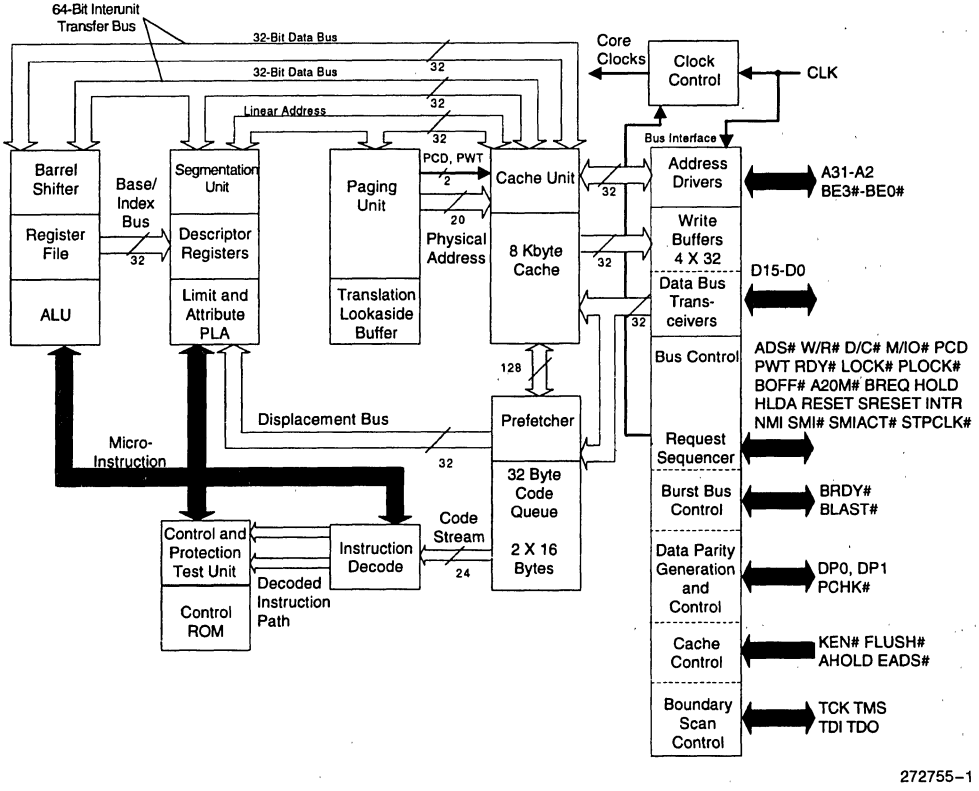
intel®

4

Embedded Intel486™ Processors

EMBEDDED ULTRA-LOW POWER Intel486™ GX PROCESSOR

- Ultra-Low Power Member of the Intel486 Processor Family
  - 32-Bit RISC Technology Core
  - 8-Kbyte Write-Through Cache
  - Four Internal Write Buffers
  - Burst Bus Cycles
  - Data Bus Parity Generation and Checking
  - Intel System Management Mode (SMM)
  - Boundary Scan (JTAG)
- 16-Bit External Data Bus
- 176-Lead Thin Quad Flat Pack (TQFP)
- Separate Voltage Supply for Core Circuitry
- Fast Core-Clock Restart
- Auto Clock Freeze
- Ideal for Embedded Battery-Operated and Hand-Held Applications



272755-1

Figure 1. Embedded ULP Intel486™ GX Processor Block Diagram

Embedded Ultra-Low Power Intel486™ GX Processor

CONTENTS

PAGE

CONTENTS

PAGE

FIGURES

CONTENTS

PAGE

TABLES

1.0 INTRODUCTION

This data sheet describes the embedded Ultra-Low Power (ULP) Intel486™ GX processor. It is intended for embedded battery-operated and hand-held applications. The embedded ULP Intel486 GX processor provides all of the features of the Intel486 SX processor except for the 8-bit bus sizing logic and the processor-upgrade pin. The processor typically uses 20% to 50% less power than the Intel486 SX processor. Additionally, the embedded ULP Intel486 GX processor external data bus and parity signals have level-keeper circuitry and a fast-recovery core clock which are vital for ultra-low-power system designs. The processor is available in a Thin Quad Flat Package (TQFP) enabling low-profile component implementation.

The embedded ULP Intel486 GX processor consists of a 32-bit integer processing unit, an on-chip cache, and a memory management unit. The design ensures full instruction-set compatibility with the 8086, 8088, 80186, 80286, Intel386™ SX, Intel386 DX, and all versions of Intel486 processors.

1.1 Features

The embedded ULP Intel486 GX processor offers these features of the Intel486 SX processor:

- **32-bit RISC-Technology Core**—The embedded ULP Intel486 GX processor performs a complete set of arithmetic and logical operations on 8-, 16-, and 32-bit data types using a full-width ALU and eight general purpose registers.
- **Single Cycle Execution**—Many instructions execute in a single clock cycle.
- **Instruction Pipelining**—Overlapped instruction fetching, decoding, address translation and execution.
- **On-Chip Cache with Cache Consistency Support**—An 8-Kbyte, write-through, internal cache is used for both data and instructions. Cache hits provide zero wait-state access times for data within the cache. Bus activity is tracked to detect alterations in the memory represented by the internal cache. The internal cache can be invalidated or flushed so that an external cache controller can maintain cache consistency.
- **External Cache Control**—Write-back and flush controls for an external cache are provided so the processor can maintain cache consistency.
- **On-Chip Memory Management Unit**—Address management and memory space protection mechanisms maintain the integrity of memory in a multitasking and virtual memory environment. Both segmentation and paging are supported.
- **Burst Cycles**—Burst transfers allow a new 16-bit data word to be read from memory on each bus clock cycle. This capability is especially useful for instruction prefetch and for filling the internal cache. Burst transfers also occur on some memory write and some I/O data transfers.
- **Write Buffers**—The processor contains four write buffers to enhance the performance of consecutive writes to memory. The processor can continue internal operations after a write to these buffers, without waiting for the write to be completed on the external bus.
- **Bus Backoff**—When another bus master needs control of the bus during a processor initiated bus cycle, the embedded ULP Intel486 GX processor floats its bus signals, then restarts the cycle when the bus becomes available again.
- **Instruction Restart**—Programs can continue execution following an exception generated by an unsuccessful attempt to access memory. This feature is important for supporting demand-paged virtual memory applications.
- **Boundary Scan (JTAG)**—Boundary Scan provides in-circuit testing of components on printed circuit boards. The Intel Boundary Scan implementation conforms with the IEEE Standard Test Access Port and Boundary Scan Architecture.

- **Intel System Management Mode (SMM)**—A unique Intel architecture operating mode provides a dedicated special purpose interrupt and address space that can be used to implement intelligent power management and other enhanced functions in a manner that is completely transparent to the operating system and applications software.
- **I/O Restart**—An I/O instruction interrupted by a System Management Interrupt (SMI#) can automatically be restarted following the execution of the RSM instruction.
- **Stop Clock**—The embedded ULP Intel486 GX processor has a stop clock control mechanism that provides two low-power states: a Stop Grant state (40–85 mW typical, depending on input clock frequency) and a Stop Clock state (~60 μW typical, with input clock frequency of 0 MHz).
- **Auto HALT Power Down**—After the execution of a HALT instruction, the embedded ULP Intel486 GX processor issues a normal Halt bus cycle and the clock input to the processor core is automatically stopped, causing the processor to enter the Auto HALT Power Down state (40–85 mW typical, depending on input clock frequency).

The embedded ULP Intel486 GX processor differs from the Intel486 SX processor in the following areas:

- **16-Bit External Data Bus**—The embedded ULP Intel486 GX processor is designed for 16-bit embedded systems, yet internally provides the 32-bit architecture of the Intel486 processor family. Two data parity bits are provided.
- **Processor Upgrade Removed**—The UP# signal is not provided.
- **Dynamic Bus-Sizing Removed**—The BS8# signal is not provided.
- **Separate Processor-Core Power**—While the embedded ULP Intel486 GX processor requires a supply voltage of 3.3V, the processor core has dedicated V_{CC} pins and operates with a supply voltage as low as 2.0V.
- **Small, Low-Profile Package**—The 176-Lead Thin Quad Flat Pack (TQFP) package is approximately 26mm square and only 1.5mm in height. This is approximately the diameter and thickness of a U.S. quarter. The embedded ULP Intel486 GX processor is ideal for embedded hand-held and battery-powered applications.
- **Level Keeper Circuits**—The embedded ULP Intel486 GX processor has level-keeper circuits for its 16-bit external data bus and parity signals. They retain valid high and low logic voltage levels when the processor is in the Stop Grant and Stop Clock states. The level-keeper circuits for the parity signals are always enabled. This is a power-saving improvement from the floating data bus of the Intel486 SX processor.
- **Auto Clock Freeze**—The embedded ULP Intel486 GX processor monitors bus events and internal activity. The Auto Clock Freeze feature automatically controls internal clock distribution, turning off clocks to internal units when they are idle. This power-saving function is transparent to the embedded system.
- **Fast Clock Restart**—The embedded ULP Intel486 GX processor requires only eight clock periods to synchronize its internal clock with the CLK input signal. This provides for faster transition from the Stop Clock State to the Normal State. For 33-MHz operation, this synchronization time is only 240 ns compared with 1 ms (PLL startup latency) for the Intel486 processor.

1.2 Family Members

Table 1 shows the embedded ULP Intel486 GX processor and briefly describes its characteristics.

Table 1. The Embedded Ultra-Low Power Intel486™ GX Processor

Product	Supply Voltage (V_{CCP})	Processor Core Supply Voltage (V_{CC})	Processor Frequency (MHz)	Package
FA80486GXSF-33	3.3V	2.0V to 3.3V	16	176-Lead TQFP
		2.2V to 3.3V	20	
		2.4V to 3.3V	25	
		2.7V to 3.3V	33	

2.0 HOW TO USE THIS DOCUMENT

Even though it has a 16-bit external data bus, the embedded ULP Intel486 GX processor has characteristics similar to the 32-bit Intel486 SX processor. This document describes the new features of the embedded ULP Intel486 GX processor. Some Intel486 SX processor information is also included to minimize the dependence on the reference documents.

For a complete set of documentation related to the embedded ULP Intel486 GX processor, use this document in conjunction with the following reference documents:

- *Intel486™ Processor Family* datasheet—Order No. 242202
- *Intel486 Microprocessor Family Programmer's Reference Manual*—Order No. 240486
- Intel Application Note AP-485—*Intel Processor Identification with the CPUID Instruction*—Order No. 241618

3.0 PIN DESCRIPTIONS

3.1 Pin Assignments

The following figures and tables show the pin assignments for the 176-pin Thin Quad Flat Pack (TQFP) package of the embedded ULP Intel486 GX processor. Included are:

- Figure 2, Package Diagram for 176-Lead TQFP Package Embedded ULP Intel486™ GX Processor
- Table 2, Pin Assignment for 176-Lead TQFP Package Embedded ULP Intel486™ GX Processor
- Table 3, Pin Cross Reference for 176-Lead TQFP Package Embedded ULP Intel486™ GX Processor
- Table 4, Embedded ULP Intel486™ GX Processor Pin Descriptions
- Table 5, Output Pins
- Table 6, Input/Output Pins
- Table 7, Test Pins
- Table 8, Input Pins

The tables and figures show "no-connects" as "N/C." These pins should always remain unconnected. Connecting N/C pins to V_{CC}, V_{CCP}, V_{SS}, or any other signal pin can result in component malfunction or incompatibility with future steppings of the embedded ULP Intel486 GX processor.

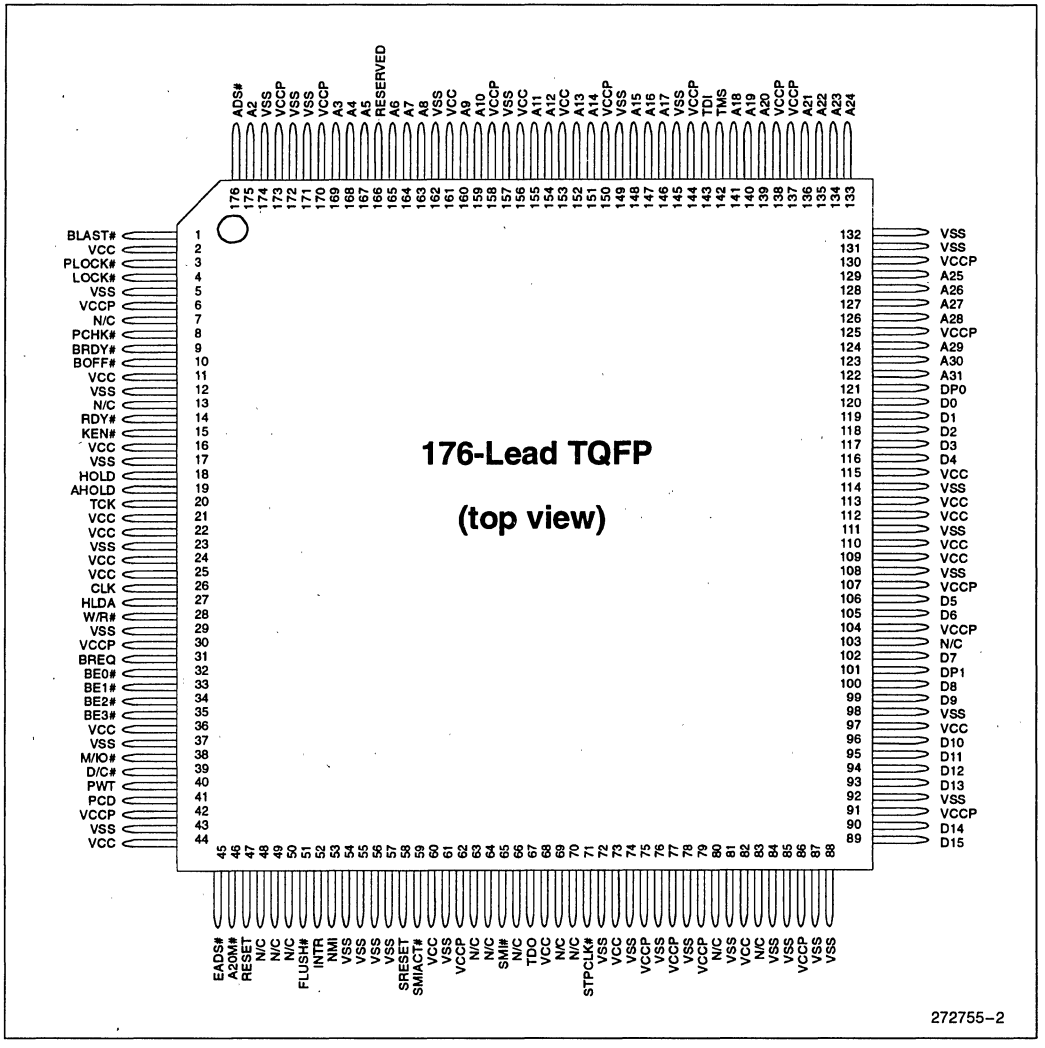


Figure 2. Package Diagram for 176-Lead TQFP Package Embedded ULP Intel486™ GX Processor

Table 2. Pin Assignment for 176-Lead TQFP Package
Embedded ULP Intel486™ GX Processor

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
1	BLAST#	31	BREQ	61	V_{SS}	91	V_{CCP}
2	V_{CC}	32	BE0#	62	V_{CCP}	92	V_{SS}
3	PLOCK#	33	BE1#	63	N/C	93	D13
4	LOCK#	34	BE2#	64	N/C	94	D12
5	V_{SS}	35	BE3#	65	SMI#	95	D11
6	V_{CCP}	36	V_{CC}	66	N/C	96	D10
7	N/C	37	V_{SS}	67	TDO	97	V_{CC}
8	PCHK#	38	M/IO#	68	V_{CC}	98	V_{SS}
9	BRDY#	39	D/C#	69	N/C	99	D9
10	BOFF#	40	PWT	70	N/C	100	D8
11	V_{CC}	41	PCD	71	STPCLK#	101	DP1
12	V_{SS}	42	V_{CCP}	72	V_{SS}	102	D7
13	N/C	43	V_{SS}	73	V_{CC}	103	N/C
14	RDY#	44	V_{CC}	74	V_{SS}	104	V_{CCP}
15	KEN#	45	EADS#	75	V_{CCP}	105	D6
16	V_{CC}	46	A20M#	76	V_{SS}	106	D5
17	V_{SS}	47	RESET	77	V_{CCP}	107	V_{CCP}
18	HOLD	48	N/C	78	V_{SS}	108	V_{SS}
19	AHOLD	49	N/C	79	V_{CCP}	109	V_{CC}
20	TCK	50	N/C	80	N/C	110	V_{CC}
21	V_{CC}	51	FLUSH#	81	V_{SS}	111	V_{SS}
22	V_{CC}	52	INTR	82	V_{CC}	112	V_{CC}
23	V_{SS}	53	NMI	83	N/C	113	V_{CC}
24	V_{CC}	54	V_{SS}	84	V_{SS}	114	V_{SS}
25	V_{CC}	55	V_{SS}	85	V_{SS}	115	V_{CC}
26	CLK	56	V_{SS}	86	V_{CCP}	116	D4
27	HLDA	57	V_{SS}	87	V_{SS}	117	D3
28	W/R#	58	SRESET	88	V_{SS}	118	D2
29	V_{SS}	59	SMIACT#	89	D15	119	D1
30	V_{CCP}	60	V_{CC}	90	D14	120	D0

Table 2. Pin Assignment for 176-Lead TQFP Package Embedded ULP Intel486™ GX Processor (Continued)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
121	DP0	135	A22	149	V_{SS}	163	A8
122	A31	136	A21	150	V_{CCP}	164	A7
123	A30	137	V_{CCP}	151	A14	165	A6
124	A29	138	V_{CCP}	152	A13	166	RESERVED
125	V_{CCP}	139	A20	153	V_{CC}	167	A5
126	A28	140	A19	154	A12	168	A4
127	A27	141	A18	155	A11	169	A3
128	A26	142	TMS	156	V_{CC}	170	V_{CCP}
129	A25	143	TDI	157	V_{SS}	171	V_{SS}
130	V_{CCP}	144	V_{CCP}	158	V_{CCP}	172	V_{SS}
131	V_{SS}	145	V_{SS}	159	A10	173	V_{CCP}
132	V_{SS}	146	A17	160	A9	174	V_{SS}
133	A24	147	A16	161	V_{CC}	175	A2
134	A23	148	A15	162	V_{SS}	176	ADS#

Table 3. Pin Cross Reference for 176-Lead TQFP Package Embedded ULP Intel486™ GX Processor

Address	Pin #	Data	Pin #	Control	Pin #	N/C	V_{CCP}	V_{CC}	V_{SS}
A2	175	D0	120	A20M#	46	7	6	2	5
A3	169	D1	119	ADS#	176	13	30	11	12
A4	168	D2	118	AHOLD	19	48	42	16	17
A5	167	D3	117	BE0#	32	49	62	21	23
A6	165	D4	116	BE1#	33	50	75	22	29
A7	164	D5	106	BE2#	34	63	77	24	37
A8	163	D6	105	BE3#	35	64	79	25	43
A9	160	D7	102	BLAST#	1	66	86	36	54
A10	159	D8	100	BOFF#	10	69	91	44	55
A11	155	D9	99	BRDY#	9	70	104	60	56
A12	154	D10	96	BREQ	31	80	107	68	57
A13	152	D11	95	CLK	26	83	125	73	61

Table 3. Pin Cross Reference for 176-Lead TQFP Package
Embedded ULP Intel486™ GX Processor (Continued)

Address	Pin #	Data	Pin #	Control	Pin #	N/C	V_{CCP}	V_{CC}	V_{SS}
A14	151	D12	94	D/C#	39	103	130	82	72
A15	148	D13	93	DP0	121		137	97	74
A16	147	D14	90	DP1	101		138	109	76
A17	146	D15	89	EADS#	45		144	110	78
A18	141			FLUSH#	51		150	112	81
A19	140			HLDA	27		158	113	84
A20	139			HOLD	18		170	115	85
A21	136			INTR	52		173	153	87
A22	135			KEN#	15			156	88
A23	134			LOCK#	4			161	92
A24	133			M/IO#	38				98
A25	129			NMI	53				108
A26	128			PCD	41				111
A27	127			PCHK#	8				114
A28	126			PLOCK#	3				131
A29	124			PWT	40				132
A30	123			RDY#	14				145
A31	122			RESERVED	166				149
				RESET	47				157
				SMI#	65				162
				SMIACT#	59				171
				SRESET	58				172
				STPCLK#	71				174
				TCK	20				
				TDI	143				
				TDO	67				
				TMS	142				
				W/R#	28				

3.2 Pin Quick Reference

The following is a brief pin description. For detailed signal descriptions refer to "Signal Description" in section 9 of the *Intel486™ Processor Family* datasheet.

Table 4. Embedded ULP Intel486™ GX Processor Pin Descriptions (Sheet 1 of 6)

Symbol	Type	Name and Function
CLK	I	**Clock** provides the fundamental timing and internal operating frequency for the embedded ULP Intel486 GX processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 **A3–A2**	I/O O	**Address Lines** A31–A2, together with the byte enable signals, BE3#–BE0#, define the physical area of memory or input/output space accessed. Address lines A31–A4 are used to drive addresses into the embedded ULP Intel486 GX processor to perform cache line invalidation. Input signals must meet setup and hold times t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.
BE3# **BE2#** **BE1#** **BE0#**	O O O O	**Byte Enable** signals indicate active bytes during read and write cycles. During the first cycle of a cache fill, the external system should assume that all byte enables are active. BE3#–BE0# are active LOW and are not driven during bus hold. BE3# applies to processor data bits D31–D24 BE2# applies to processor data bits D23–D16 BE1# applies to processor data bits D15–D8 BE0# applies to processor data bits D7–D0 The byte enables can be used by the external system to generate address bits A1 and A0, as well as byte-high (D15–D8) and byte-low (D7–D0) enables. These are needed to interpret the 16-bit external data bus.
DATA BUS		
D15–D0	I/O	**Data Lines.** D7–D0 define the least significant byte of the data bus; D15–D8 define the most significant byte of the data bus. These signals must meet setup and hold times t_{22} and t_{23} for proper operation on reads. These pins are driven during the second and subsequent clocks of write cycles.
DP1 **DP0**	I/O	There is one **Data Parity** pin for each byte of the data bus. Data parity is generated on all write data cycles with the same timing as the data driven by the embedded ULP Intel486 GX processor. Even parity information must be driven back into the processor on the data parity pins with the same timing as read information to ensure that the correct parity check status is indicated by the processor. The signals read on these pins do not affect program execution. Input signals must meet setup and hold times t_{22} and t_{23}. DP1 and DP0 must be connected to V_{CCP} through a pull-up resistor in systems that do not use parity. DP1 and DP0 are active HIGH and are driven during the second and subsequent clocks of write cycles.

Table 4. Embedded ULP Intel486™ GX Processor Pin Descriptions (Sheet 2 of 6)

Symbol	Type	Name and Function
PCHK#	O	**Parity Status** is driven on the PCHK# pin the clock after ready for read operations. The parity status is for data sampled at the end of the previous clock. A parity error is indicated by PCHK# being LOW. Parity status is only checked for enabled bytes as indicated by the byte enable signals. PCHK# is valid only in the clock immediately after read data is returned to the processor. At all other times PCHK# is inactive (HIGH). PCHK# is never floated.
BUS CYCLE DEFINITION		
M/IO# **D/C#** **W/R#**	O O O	**Memory/Input-Output, Data/Control** and **Write/Read** lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted. **M/IO# D/C# W/R# — Bus Cycle Initiated** 0 0 0 — Interrupt Acknowledge 0 0 1 — HALT/Special Cycle (see details below) 0 1 0 — I/O Read 0 1 1 — I/O Write 1 0 0 — Code Read 1 0 1 — Reserved 1 1 0 — Memory Read 1 1 1 — Memory Write **HALT/Special Cycle** **Cycle Name — BE3#–BE0# — A4–A2** Shutdown — 1110 — 000 HALT — 1011 — 000 Stop Grant bus cycle — 1011 — 100
LOCK#	O	**Bus Lock** indicates that the current bus cycle is locked. The embedded ULP Intel486 GX processor does not allow a bus hold when LOCK# is asserted (address holds are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the last clock of the last locked bus cycle. The last locked cycle ends when Ready is returned. LOCK# is active LOW and not driven during bus hold. Locked read cycles are not transformed into cache fill cycles when KEN# is returned active.
PLOCK#	O	**Pseudo-Lock** indicates that the current bus transaction requires more than one bus cycle to complete. For the embedded ULP Intel486 GX processor, examples of such operations are segment table descriptor reads (64 bits) and cache line fills (128 bits). The embedded ULP Intel486 GX processor drives PLOCK# active until the addresses for the last bus cycle of the transaction are driven, regardless of whether RDY# or BRDY# have been returned. PLOCK# should be sampled only in the clock in which Ready is returned. PLOCK# is active LOW and is not driven during bus hold.
BUS CONTROL		
ADS#	O	**Address Status** output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus. ADS# is driven active in the same clock in which the addresses are driven. ADS# is active LOW and not driven during bus hold.

4

Table 4. Embedded ULP Intel486™ GX Processor Pin Descriptions (Sheet 3 of 6)

Symbol	Type	Name and Function
RDY#	I	**Non-burst Ready** input indicates that the current bus cycle is complete. RDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted data from the embedded ULP Intel486 GX processor in response to a write. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. RDY# is active during address hold. Data can be returned to the embedded ULP Intel486 GX processor while AHOLD is active. RDY# is active LOW and is not provided with an internal pull-up resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.
BURST CONTROL		
BRDY#	I	**Burst Ready** input performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data in response to a read or that the external system has accepted data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle. BRDY# is sampled in the second and subsequent clocks of a burst cycle. Data presented on the data bus is strobed into the embedded ULP Intel486 GX processor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. BRDY# is active LOW and is provided with a small pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.
BLAST#	O	**Burst Last** signal indicates that the next time BRDY# is returned, the burst bus cycle is complete. BLAST# is active for both burst and non-burst bus cycles. BLAST# is active LOW and is not driven during bus hold.
INTERRUPTS		
RESET	I	**Reset** input forces the embedded ULP Intel486 GX processor to begin execution at a known state. The processor cannot begin executing instructions until at least 1ms after V_{CC}, V_{CCP}, and CLK have reached their proper DC and AC specifications. The RESET pin must remain active during this time to ensure proper processor operation. However, for warm resets, RESET should remain active for at least 15 CLK periods. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
INTR	I	**Maskable Interrupt** indicates that an external interrupt has been generated. When the internal interrupt flag is set in EFLAGS, active interrupt processing is initiated. The embedded ULP Intel486 GX processor generates two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the interrupt acknowledges have been performed to ensure processor recognition of the interrupt. INTR is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

Table 4. Embedded ULP Intel486™ GX Processor Pin Descriptions (Sheet 4 of 6)

Symbol	Type	Name and Function
NMI	I	**Non-Maskable Interrupt** request signal indicates that an external non-maskable interrupt has been generated. NMI is rising-edge sensitive and must be held LOW for at least four CLK periods before this rising edge. NMI is not provided with an internal pull-down resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SRESET	I	**Soft Reset** pin duplicates all functionality of the RESET pin except that the SMBASE register retains its previous value. For soft resets, SRESET must remain active for at least 15 CLK periods. SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SMI#	I	**System Management Interrupt** input invokes System Management Mode (SMM). SMI# is a falling-edge triggered signal which forces the embedded ULP Intel486 GX processor into SMM at the completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# does not break LOCKed bus cycles and cannot interrupt a currently executing SMM. The embedded ULP Intel486 GX processor latches the falling edge of one pending SMI# signal while it is executing an existing SMI#. The nested SMI# is not recognized until after the execution of a Resume (RSM) instruction.
SMIACT#	O	**System Management Interrupt Active**, an active LOW output, indicates that the embedded ULP Intel486 GX processor is operating in SMM. It is asserted when the processor begins to execute the SMI# state save sequence and remains active LOW until the processor executes the last state restore cycle out of SMRAM.
STPCLK#	I	**Stop Clock Request** input signal indicates a request was made to turn off or change the CLK input frequency. When the embedded ULP Intel486 GX processor recognizes a STPCLK#, it stops execution on the next instruction boundary (unless superseded by a higher priority interrupt), empties all internal pipelines and write buffers, and generates a Stop Grant bus cycle. STPCLK# is active LOW. **STPCLK# is an asynchronous signal, but must remain active until the embedded ULP Intel486 GX processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle.**
BUS ARBITRATION		
BREQ	O	**Bus Request** signal indicates that the embedded ULP Intel486 GX processor has internally generated a bus request. BREQ is generated whether or not the processor is driving the bus. BREQ is active HIGH and is never floated.
HOLD	I	**Bus Hold Request** allows another bus master complete control of the embedded ULP Intel486 GX processor bus. In response to HOLD going active, the processor floats most of its output and input/output pins. HLDA is asserted after completing the current bus cycle, burst cycle or sequence of locked cycles. The embedded ULP Intel486 GX processor remains in this state until HOLD is de-asserted. HOLD is active HIGH and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper operation.

Table 4. Embedded ULP Intel486™ GX Processor Pin Descriptions (Sheet 5 of 6)

Symbol	Type	Name and Function
HLDA	O	**Hold Acknowledge** goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the embedded ULP Intel486 GX processor has given the bus to another local bus master. HLDA is driven active in the same clock that the processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.
BOFF#	I	**Backoff** input forces the embedded ULP Intel486 GX processor to float its bus in the next clock. The processor floats all pins normally floated during bus hold but HLDA is not asserted in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#; if both are returned in the same clock, BOFF# takes effect. The embedded ULP Intel486 GX processor remains in bus hold until BOFF# is negated. If a bus cycle is in progress when BOFF# is asserted the cycle is restarted. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper operation.
CACHE INVALIDATION		
AHOLD	I	**Address Hold** request allows another bus master access to the embedded ULP Intel486 GX processor's address bus for a cache invalidation cycle. The processor stops driving its address bus in the clock following AHOLD going active. Only the address bus is floated during address hold, the remainder of the bus remains active. AHOLD is active HIGH and is provided with a small internal pull-down resistor. For proper operation, AHOLD must meet setup and hold times t_{18} and t_{19}.
EADS#	I	**External Address** - This signal indicates that a *valid* external address has been driven onto the embedded ULP Intel486 GX processor address pins. This address is used to perform an internal cache invalidation cycle. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy setup and hold times t_{12} and t_{13} for proper operation.
CACHE CONTROL		
KEN#	I	**Cache Enable** pin is used to determine whether the current cycle is cacheable. When the embedded ULP Intel486 GX processor generates a cycle that can be cached and KEN# is active one clock before RDY# or BRDY# during the first transfer of the cycle, the cycle becomes a cache line fill cycle. Returning KEN# active one clock before RDY# during the last read in the cache line fill causes the line to be placed in the on-chip cache. KEN# is active LOW and is provided with a small internal pull-up resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	**Cache Flush** input forces the embedded ULP Intel486 GX processor to flush its entire internal cache. FLUSH# is active LOW and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition in any specific clock.

Table 4. Embedded ULP Intel486™ GX Processor Pin Descriptions (Sheet 6 of 6)

Symbol	Type	Name and Function
PAGE CACHEABILITY		
PWT **PCD**	O O	**Page Write-Through** and **Page Cache Disable** pins reflect the state of the page attribute bits, PWT and PCD, in the page table entry, page directory entry or control register 3 (CR3) when paging is enabled. When paging is disabled, the embedded ULP Intel486 GX processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT pins. PWT and PCD have the same timing as the cycle definition pins (M/IO#, D/C#, and W/R#). PWT and PCD are active HIGH and are not driven during bus hold. PCD is masked by the cache disable bit (CD) in Control Register 0.
ADDRESS MASK		
A20M#	I	**Address Bit 20 Mask** pin, when asserted, causes the embedded ULP Intel486 GX processor to mask physical address bit 20 (A20) before performing a lookup to the internal cache or driving a memory cycle on the bus. A20M# emulates the address wraparound at 1Mbyte, which occurs on the 8086 processor. A20M# is active LOW and should be asserted only when the embedded ULP Intel486 GX processor is in real mode. This pin is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For proper operation, A20M# should be sampled HIGH at the falling edge of RESET.
TEST ACCESS PORT		
TCK	I	**Test Clock,** an input to the embedded ULP Intel486 GX processor, provides the clocking function required by the JTAG Boundary scan feature. TCK is used to clock state information (via TMS) and data (via TDI) into the component on the rising edge of TCK. Data is clocked out of the component (via TDO) on the falling edge of TCK. TCK is provided with an internal pull-up resistor.
TDI	I	**Test Data Input** is the serial input used to shift JTAG instructions and data into the processor. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and SHIFT-DR TAP controller states. During all other Test Access Port (TAP) controller states, TDI is a "don't care." TDI is provided with an internal pull-up resistor.
TDO	O	**Test Data Output** is the serial output used to shift JTAG instructions and data out of the component. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state.
TMS	I	**Test Mode Select** is decoded by the JTAG TAP to select test logic operation. TMS is sampled on the rising edge of TCK. To guarantee deterministic behavior of the TAP controller, TMS is provided with an internal pull-up resistor.
RESERVED PINS		
RESERVED	I	**Reserved** is reserved for future use. This pin MUST be connected to an external pull-up resistor circuit. The recommended resistor value is 10 KΩ.

Table 5. Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States(1)
BREQ	HIGH			Previous State
HLDA	HIGH			As per HOLD
BE3#-BE0#	LOW		•	Previous State
PWT, PCD	HIGH		•	Previous State
W/R#, M/IO#, D/C#	HIGH/LOW		•	Previous State
LOCK#	LOW		•	HIGH (inactive)
PLOCK#	LOW		•	HIGH (inactive)
ADS#	LOW		•	HIGH (inactive)
BLAST#	LOW		•	Previous State
PCHK#	LOW			Previous State
A3-A2	HIGH	•	•	Previous State
SMIACT#	LOW			Previous State

NOTE:
1. The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

Table 6. Input/Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States(1,2)
D15-D0	HIGH		•	Level-Keeper
DP1, DP0	HIGH		•	Level-Keeper
A31-A4	HIGH	•	•	Previous State

NOTE:
1. The term "Level-Keeper" means that the processor maintains the most recent logic level applied to the signal pin. This conserves power by preventing the signal pin from floating. If a system component, other than the processor, temporarily drives these signal pins and then floats them, the processor forces and maintains the most recent logic levels that were applied by the system component. The level keepers for DP1 and DP0 are always enabled.
2. The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

Table 7. Test Pins

Name	Input or Output	Sampled/ Driven On
TCK	Input	N/A
TDI	Input	Rising Edge of TCK
TDO	Output	Failing Edge of TCK
TMS	Input	Rising Edge of TCK

Table 8. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Pull-Up/ Pull-Down
CLK			
RESET	HIGH	Asynchronous	
SRESET	HIGH	Asynchronous	Pull-Down
HOLD	HIGH	Synchronous	
AHOLD	HIGH	Synchronous	Pull-Down
EADS#	LOW	Synchronous	Pull-Up
BOFF#	LOW	Synchronous	Pull-Up
FLUSH#	LOW	Asynchronous	Pull-Up
A20M#	LOW	Asynchronous	Pull-Up
KEN#	LOW	Synchronous	Pull-Up
RDY#	LOW	Synchronous	
BRDY#	LOW	Synchronous	Pull-Up
INTR	HIGH	Asynchronous	
NMI	HIGH	Asynchronous	
RESERVED			
SMI#	LOW	Asynchronous	Pull-Up
STPCLK#	LOW	Asynchronous	Pull-Up
TCK	HIGH		Pull-Up
TDI	HIGH		Pull-Up
TMS	HIGH		Pull-Up

4.0 ARCHITECTURAL AND FUNCTIONAL OVERVIEW

The embedded ULP Intel486 GX processor architecture is essentially the same as the 3.3V Intel486 SX processor with a 1X clock (CLK) input. Refer to the *Intel486™ Processor Family* datasheet (242202) for a description of the Intel486 SX processor. With some minor exceptions, the following datasheet sections apply to the embedded ULP Intel486 GX processor:

- Architectural Overview
- Real Mode Architecture
- Protected Mode Architecture
- On-Chip Cache
- System Management Mode (SMM) Architectures
- Hardware Interface
- Bus Operation
- Testability
- Debugging Support
- Instruction Set Summary
- Differences Between Intel486 Processors and Intel386™ Processors

Exceptions to these sections of the datasheet are:

- The embedded ULP Intel486 GX processor has a 16-bit external data bus and two data parity signals. While it has four byte-enable signals (BE3#–BE0#), the external system must generate address bits A1, A0 as well as enables for each byte of the 16-bit external data bus. More information about byte enables is provided in this datasheet.
- The information pertaining to dynamic bus sizing of the external data bus does not apply. The embedded ULP Intel486 GX processor does not have the BS8# signal pin.

- The embedded ULP Intel486 GX processor bursts data cycles similar to an Intel486 SX processor with bus-sizing BS16# active.
- References to the Upgrade Power Down Mode do not apply. The embedded ULP Intel486 GX processor does not have the UP# signal pin and does not support the Intel OverDrive® processor.
- References to "V_{CC}" are called "V_{CCP}" by the embedded ULP Intel486 GX processor when the supply voltage pertains to the processor's external interface drivers and receivers. The term "V_{CC}" pertains only to the processor core supply voltage of the embedded ULP Intel486 GX processor. Information about the split-supply voltage is provided in this datasheet.
- The Phase-Locked Loop (PLL) circuit of the 1X clock (CLK) input has been replaced by a proprietary Differential Delay Line (DDL) circuit that has a faster recovery time. Datasheet references to the PLL and its 1 ms recovery time are replaced with the DDL circuit and its eight-CLK recovery time. Information about the DDL circuit and recovery time is provided in this datasheet.
- The embedded ULP Intel486 GX processor has level-keeper circuits for its external 16-bit data bus (D15–D0) and data parity (DP1, DP0) signals. The Intel486 SX processor floats these signals instead. More information about the level-keeper circuitry is provided in this datasheet.
- The datasheet describes the processor supply-current consumption for the Auto HALT Power Down, Stop Grant, and Stop Clock states. This supply-current consumption for the embedded ULP Intel486 GX processor is much less than that of the Intel486 SX processor. Information about power consumption and these states is provided in this datasheet.
- The CPU ID, Boundary-Scan (JTAG) ID, and boundary-scan register bits for the embedded ULP Intel486 GX processor are in this datasheet.
- The embedded ULP Intel486 GX processor has one pin reserved for possible future use. This pin is an input signal, pin 166. It is called RESERVED and must be connected to a 10-KΩ pull-up resistor.

4.1 Separate Supply Voltages

The embedded ULP Intel486 GX processor has separate voltage-supply planes for its internal core-processor circuits and its external driver/receiver circuits. The supply voltage for the internal core processor is named V_{CC} and the supply voltage for the external circuits is named V_{CCP}.

For a single-supply voltage design, the embedded ULP Intel486 GX processor is functional at 3.3V ± 0.3V. In this type of system design, the processor's V_{CC} and V_{CCP} pins must be tied to the same power plane.

Even though V_{CCP} must be 3.3V ± 0.3V, the processor's external-output circuits can drive TTL-compatible components. However, the processor's external-input circuits do not allow connection to TTL-compatible components. **Section 5.2, DC Specifications** contains the DC specifications for the processor's input and output signals.

For lower-power operation, a separate, lower voltage for V_{CC} can be chosen, but V_{CCP} must be 3.3V ± 0.3V. Any voltage value between 2.0V and 3.3V can be chosen for V_{CC} for guaranteed processor operation up to 16 MHz. The embedded ULP Intel486 GX processor can also operate at 33 MHz, provided the V_{CC} value chosen is between 2.7V and 3.3V. **Section 5.2, DC Specifications** defines supply voltage specifications.

In systems with separate V_{CC} and V_{CCP} power planes, the processor-core voltage supply must always be less than or equal to the processor's external-interface voltage supply; e.g., the system design must guarantee:

$$V_{CC} \leq V_{CCP}$$

Violating this relationship causes excessive power consumption. Limited testing has shown no component damage when this relationship is violated. However, prolonged violation is not recommended and component integrity is not guaranteed.

The $V_{CC} \leq V_{CCP}$ relationship must also be guaranteed by the system design during power-up and power-down sequences. Refer to Figure 3.

Even though V_{CC} must be less than or equal to V_{CCP}, it is recommended that the system's power-on sequence allows V_{CC} to quickly achieve its op-

erational level once V_{CCP} achieves its operational level. Similarly, the power-down sequence should allow V_{CCP} to power down quickly after V_{CC} is below the operational voltage level. These recommendations are given to keep power consumption at a minimum. Deviating from the recommendations does not create a component reliability problem, but power consumption of the processor's external interface circuits increases beyond normal operating values.

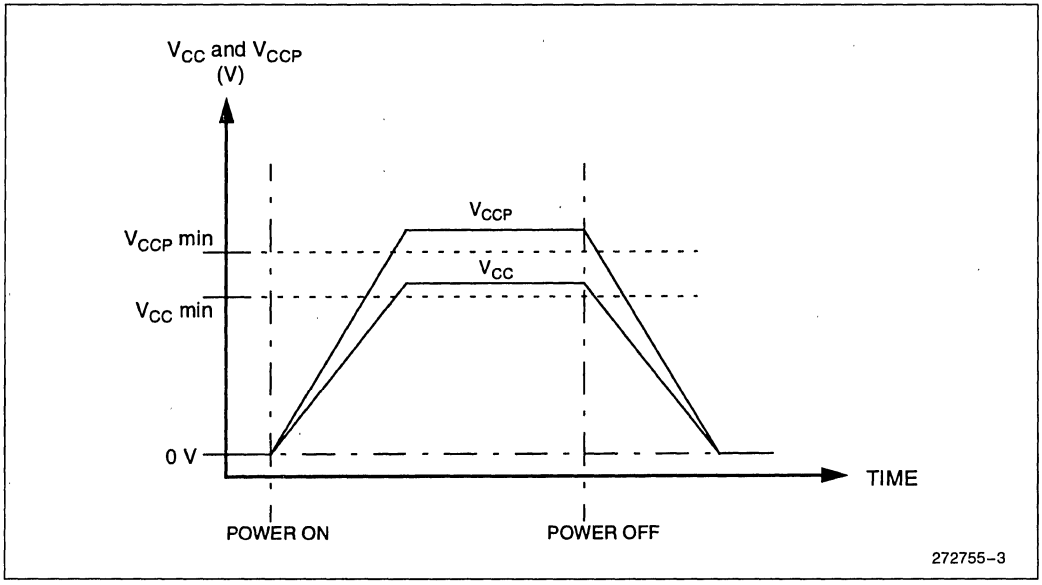


Figure 3. Example of Supply Voltage Power Sequence

4.2 Fast Clock Restart

The embedded ULP Intel486 GX processor has an integrated proprietary differential delay line (DDL) circuit for internal clock generation. The DDL is driven by the CLK input signal provided by the external system. During normal operation, the external system must guarantee that the CLK signal maintains its frequency so that the clock period deviates no more than 250 ps/CLK. This state, called the Normal State, is shown in Figure 4.

To increase or decrease the CLK frequency more quickly than this, the system must interrupt the processor with the STPCLK# signal. Once the processor indicates that it is in the Stop Grant State, the system can adjust the CLK signal to the new frequency, wait a minimum of eight CLK periods, then force the processor to return to the normal operational state by deactivating the STPCLK# interrupt.

This wait of eight CLK periods is much faster than the 1ms wait required by earlier Intel486 SX processor products.

While in the Stop Grant State, the external system may deactivate the CLK signal to the processor. This forces the processor to the Stop Clock State—the state in which the processor consumes the least power. Once the system reactivates the CLK signal, the processor transitions to the Stop Grant State within eight CLK periods.

Normal operation can be resumed by deactivating the STPCLK# interrupt signal. Here again, the embedded ULP Intel486 GX processor recovers from the Stop Clock State much faster than the 1ms PLL recovery of earlier Intel486 SX processors.

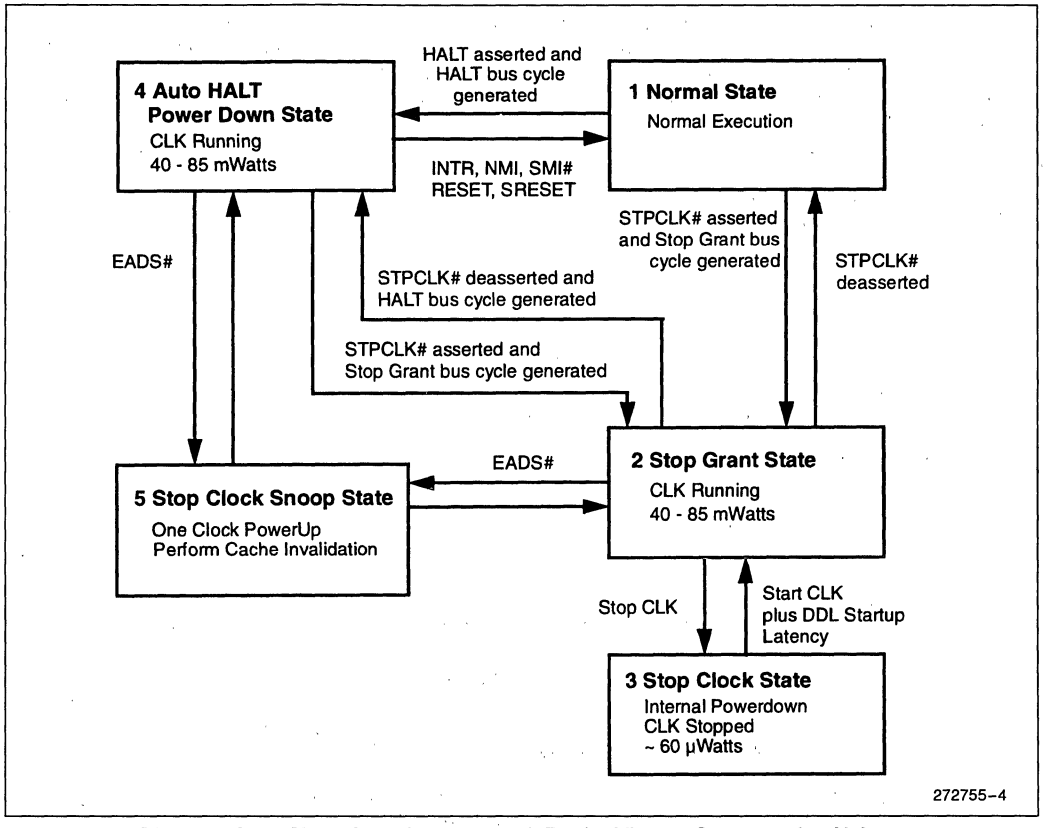


Figure 4. Stop Clock State Diagram with Typical Power Consumption Values

4.3 Level-Keeper Circuits

To obtain the lowest possible power consumption during the Stop Grant and Stop Clock states, system designers must ensure that:

- input signals with pull-up resistors are not driven LOW
- input signals with pull-down resistors are not driven HIGH

See Table 8, Input Pins for the list of signals with internal pull-up and pull-down resistors.

All other input pins except A31–A4, D15–D0, DP1, and DP0 must be driven to the power supply rails to ensure lowest possible current consumption.

During the Stop Grant and Stop Clock states, most processor output signals maintain their previous condition, which is the level they held when entering the Stop Grant state. In response to HOLD driven active during the Stop Grant state when the CLK input is running, the embedded ULP Intel486 GX processor generates HLDA and floats all output and input/output signals which are floated during the HOLD/HLDA state. When HOLD is deasserted, processor signals which maintain their previous state return to the state they were in prior to the HOLD/HLDA sequence.

The data bus (D15–D0) and parity bits also maintain their previous states during the Stop Grant and Stop Clock states, but do so differently, as described in the following paragraphs.

The embedded ULP Intel486 GX processor's data bus pins (D15–D0) and data parity pins have level keepers which maintain their previous states while in the Stop Grant and Stop Clock states. In response to HOLD driven active during the Stop Grant state when the CLK input is running, the embedded ULP Intel486 GX processor generates HLDA and floats D15–D0, DP1 and DP0 throughout the HOLD/HLDA cycles. When HOLD is deasserted, the processor's D15–D0, DP1 and DP0 signals return to the states they were in prior to the HOLD/HLDA sequence.

At all other times during the Stop Grant and Stop Clock states, the processor maintains the logic levels of D15–D0, DP1 and DP0. When the external system circuitry drives D15–D0, DP1 and DP0 to different logic levels, the processor flips its D15–D0, DP1 and DP0 logic levels to match the ones driven by the external system. The processor maintains (keeps) these new levels even after the external circuitry stops driving D15–D0, DP1 and DP0.

For some system designs, external resistors may not be required on D15–D0, DP1 and DP0 (they are required on previous Intel486 SX processor designs). System designs that never request Bus Hold during the Stop Grant and Stop Clock states might not require external resistors. If the system design uses Bus Hold during these states, the processor disables the level-keepers and floats the data bus. This type of design would require some kind of data bus termination to minimize power consumption. It is strongly recommended that the D15–D0, DP1 and DP0 pins do not have network resistors connected. External resistors used in the system design must be of a sufficient resistance value to "flip" the level-keeper circuitry and eliminate potential DC paths.

The level-keeper circuits for DP1 and DP0 are always enabled, while the level-keeper circuits for D15–D0 are enabled only during the Stop Grant and Stop Clock states.

The level-keeper circuit is designed to allow an external 27-KΩ pull-up resistor to switch the D15–D0, DP1 and DP0 circuits to a logic-HIGH level even though the level-keeper attempts to keep a logic-LOW level. In general, pull-up resistors smaller than 27 KΩ can be used as well as those greater than or equal to 1 MΩ. Pull-down resistors, when connected to D15–D0, DP1 and DP0, should be least 800 KΩ.

4.4 Low-Power Features

As with other Intel486 processors, the embedded ULP Intel486 GX processor minimizes power consumption by providing the Auto HALT Power Down, Stop Grant, and Stop Clock states (see Figure 4). The embedded ULP Intel486 GX processor has an Auto Clock Freeze feature that further conserves power by judiciously deactivating its internal clocks while in the Normal Execution Mode. The power-conserving mechanism is designed such that it does not degrade processor performance or require changes to AC timing specifications.

4.4.1 Auto Clock Freeze

To reduce power consumption, during the following bus cycles—under certain conditions—the processor slows-up or freezes some internal clocks:

- Data-Read Wait Cycles (Memory, I/O and Interrupt Acknowledge)
- Data-Write Wait Cycles (Memory, I/O)
- HOLD/HLDA Cycles
- AHOLD Cycles
- BOFF Cycles

Power is conserved during the wait periods in these cycles until the appropriate external-system signals are sent to the processor. These signals include:

- READY
- NMI, SMI#, INTR, and RESET
- BOFF#
- FLUSH#
- EADS#
- KEN# transitions

The embedded ULP Intel486 GX processor also reduces power consumption by temporarily freezing the clocks of its internal logic blocks. When a logic block is idle or in a wait state, its clock is frozen.

4.5 Bus Interface and Operation

4.5.1 16-Bit Data Bus

The bi-directional lines, D15–D0, form the data bus for the embedded ULP Intel486 GX processor. D7–D0 define the least-significant byte and D15–D8 the most-significant byte. Data transfers are possible only to 16-bit devices. Bus-sizing for 8-bit devices (BS8# signal pin) is not supported by the processor. In some cases, external circuitry is needed for the processor to interface with 8-bit devices. An example of when external circuitry is not needed is an 8-bit I/O port that is mapped to a byte address. Here only part of the 16-bit data word is meant for the device and BS8# is not needed.

D15–D0 are active HIGH. For reads, D15–D0 must meet the setup and hold times, t_{22} and t_{23}. D15–D0 are not driven during read cycles and bus hold.

4.5.2 Parity

Parity operation is the same as it is for the rest of the Intel486 processor family, with these exceptions:

- DP0 and DP1 are the data parity pins for the processor. There is one parity signal for each byte of the external data bus. Input signals on DP0 and DP1 must meet the setup and hold times, t_{22} and t_{23}. In systems not using parity, DP0 and DP1 must be connected to V_{CCP} through a pull-up resistor.
- The data parity pins have level-keeper circuits which are described later.

4.5.3 Data Transfer Mechanism

Data transfers operate in a manner similar to data transfers on the 32-bit data bus members of the Intel486 processor family with the BS16# pin driven active. For 32-bit data-bus family members, such 16-bit data transfers involve all 32 bits of their external data busses and all four parity bits. Since the embedded ULP Intel486 GX processor has a 16-bit external data bus, all data transfers occur on the low order data bits, D0 through D15. Parity is generated and checked on DP0 and DP1. Dynamic Data Bus Sizing (BS16#, and BS8#) is not supported. All address bits (A31–A2) and byte enables (BE0#, BE1#, BE2#, and BE3#) are supported. Address bits A1 and A0 can be generated from the byte-enable signals in the same manner as the other Intel486 processors. Typically in 16-bit data bus designs, A1, byte-low enable (BLE), and byte-high enable (BHE) are needed and can be generated from the four byte-enable signals. Figure 5 shows the logic that can be used to generate A1, BHE#, and BLE#.

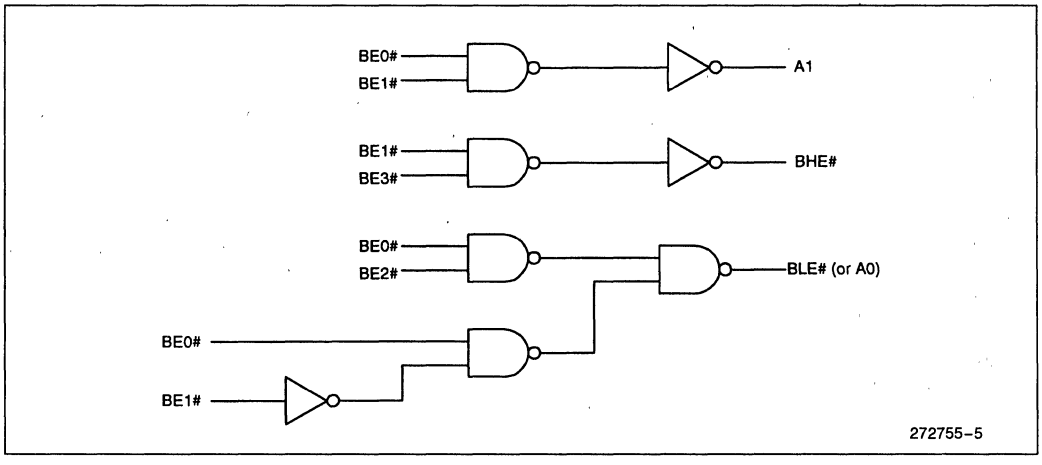


Figure 5. Logic to Generate A1, BHE# and BLE#

Table 9 contains the list of valid byte-enable combinations and how the 16-bit external data bus is interpreted.

Table 9. Valid Byte-Enable Cycles

Case	Byte Enables				From External Circuitry (Note 1)				External Data Bus	
	BE3#	BE2#	BE1#	BE0#	A1	A0	BHE#	BLE# (A0)	D15–D8, DP1	D7–D0 DP0
1	1	1	1	0	0	0	1	0	-	valid
2	1	1	0	0	0	0	0	0	valid	valid
3	1	0	0	0	0	0	0	0	valid	valid
4	0	0	0	0	0	0	0	0	valid	valid
5	1	1	0	1	0	1	0	1	valid	-
6	1	0	0	1	0	1	0	1	valid	-
7	0	0	0	1	0	1	0	1	valid	-
8	1	0	1	1	1	0	1	0	-	valid
9	0	0	1	1	1	0	0	0	valid	valid
10	0	1	1	1	1	1	0	1	valid	-

NOTE:
1. If the external system indicates to the processor that a read is cacheable, the processor initiates a cache-line fill. In this case, the external system should ignore BE3#, BE2#, BE1#, and BE0# and force A1, A0, and BHE# to a low logic level (0) for the first cycle of the transfer. This forces a memory read to start from a data address having its least significant digit 0, 4, 8, or C (hex). The byte-enable decodes for subsequent cycles of the line fill follow the table information as listed.

Except for the initial transfer of a cache-line fill, the Byte Enables BE3#, BE2#, BE1#, and BE0# for cases 1, 2, 5, 8, 9, and 10 indicate either a one-, or two-byte data transfer that can be accomplished in one 16-bit data cycle.

Except for the initial transfer of a cache-line fill, the Byte Enables BE3#, BE2#, BE1#, and BE0# for cases 3, 4, 6, and 7 indicate the transfer of two, three, or four data bytes that cannot be accomplished in one 16-bit data cycle. In these cases, the processor attempts to complete the partial transfer using an additional data cycle. The additional cycle could be burst by the processor (processor could respond with BLAST# unasserted for case 3, 4, 6, or 7). This is true for both memory and I/O reads and writes. There is more information about bursting in later sections.

During write cycles, valid data is only driven onto the external data bus pins corresponding to active byte enables. Other pins of the data bus are driven but do not contain valid data.

NOTE:
Unlike the Intel386™ processor, the embedded ULP Intel486 GX processor does not duplicate write data onto the parts of the data bus for which the corresponding byte enable is inactive.

4.5.3.1 Multiple and Burst Cycle Bus Transfers

The embedded ULP Intel486 GX processor, like all other Intel486 processors, requires more than one data cycle to read or write data having bit widths greater than 32. Examples of this data are cache lines (128 bits) and instruction prefetches (128 bits). In addition, the embedded ULP Intel486 GX processor requires multiple data cycles to transfer data having bit widths greater than 16. An example is a doubleword operand (32 bits). Transferring misaligned 16-bit words also requires multiple data cycles.

If a multiple data cycle is a memory-read or I/O-read data transfer, the processor could use burst cycles to perform the transfer. The processor could also burst misaligned 16-bit and 32-bit memory-write or I/O-write data transfers.

In designing a memory and I/O port controller for the embedded ULP Intel486 GX processor, knowledge of the address sequence for burst cycles can be used to provide high-speed data access (minimal number of wait states). The following sections describe this sequence.

4.5.3.2 Cacheable Cycles

The embedded ULP Intel486 GX processor uses burst cycles to perform a cache line fill. Because of its 16-bit external data bus, the processor bursts eight data cycles to read a 128-bit (16-byte) cache line from system memory. During the first cycle of the cache line transfer, the external system must ignore BE3#, BE2#, BE1#, and BE0# presented by the processor and proceed as if A1, A0, and BHE# were logic-low levels (0). This forces the memory read to start from a data address having its least significant hexadecimal digit 0, 4, 8, or C. The byte enables presented by the processor for subsequent cycles are decoded in the usual way by the external system. The sequences of data addresses are shown in Table 10. Like the rest of the Intel486 processor family, the initial value of A31–A4, M/IO#, W/R#, and D/C# are presented by the processor throughout the cache line fill. Also, the burst sequence can be terminated by the processor at any time by with an active BLAST# signal.

Table 10. Address Sequence for Cache Line Transfers and Instruction Prefetches

Starting Address (Least significant hexadecimal digit)	Data Cycle	Signals from the Processor				Address of Expected Read Data	
		A3 A2	Byte Enables BE3#–BE0#	A3–A0 (Hex)	BLAST#	D15–D8, DP1	D7–D0, DP0
0, 1, 2, 3	1	0 0	0 0 0 0	0	1	1	0
	2	0 0	0 0 1 1	2	1	3	2
	3	0 1	0 0 0 0	4	1	5	4
	4	0 1	0 0 1 1	6	1	7	6
	5	1 0	0 0 0 0	8	1	9	8
	6	1 0	0 0 1 1	A	1	B	A
	7	1 1	0 0 0 0	C	1	D	C
	8	1 1	0 0 1 1	E	0	F	E
4, 5, 6, 7	1	0 1	0 0 0 0	4	1	5	4
	2	0 1	0 0 1 1	6	1	7	6
	3	0 1	0 0 0 0	0	1	1	0
	4	0 0	0 0 1 1	2	1	3	2
	5	1 1	0 0 0 0	C	1	D	C
	6	1 1	0 0 1 1	E	1	F	E
	7	1 0	0 0 0 0	8	1	9	8
	8	1 0	0 0 1 1	A	0	B	A
8, 9, A, B	1	1 0	0 0 0 0	8	1	9	8
	2	1 0	0 0 1 1	A	1	B	A
	3	1 1	0 0 0 0	C	1	D	C
	4	1 1	0 0 1 1	E	1	F	E
	5	0 0	0 0 0 0	0	1	1	0
	6	0 0	0 0 1 1	2	1	3	2
	7	0 1	0 0 0 0	4	1	5	4
	8	0 1	0 0 1 1	6	0	7	6
C, D, E, F	1	1 1	0 0 0 0	C	1	D	C
	2	1 1	0 0 1 1	E	1	F	E
	3	1 0	0 0 0 0	8	1	9	8
	4	1 0	0 0 1 1	A	1	B	A
	5	0 1	0 0 0 0	4	1	5	4
	6	0 1	0 0 1 1	6	1	7	6
	7	0 0	0 0 0 0	0	1	1	0
	8	0 0	0 0 1 1	2	0	3	2

Whenever its cache circuitry is not busy, the processor uses this same bursting mechanism for prefetching instructions (128 bits, 16 bytes), even if the instructions are not indicated as cacheable by the external system. Instruction prefetches can occur that use the address sequencing shown in Table 10. The initial value of A31–A4, M/IO#, W/R#, and D/C# are presented by the processor throughout the 128-bit prefetch burst. It is possible for the processor to prefetch instructions not needed. The burst sequence can be terminated by the processor at any time with an active BLAST# signal.

4.5.3.3 Non-Cacheable Cycles

For memory and I/O data transfers, the embedded ULP Intel486 GX processor determines how many data cycles are required for the transfer based on its internal information. This information includes the byte-length of the data, the transfer's starting data address, and data alignment. For memory reads, the processor resorts to the 128-bit cache-line address sequence described above if the external system indicates the data is cacheable. Otherwise, the processor uses its internal information to determine whether to burst the data cycles of a multiple-cycle transfer. In some cases, the transfer can be performed entirely by burst cycles. In other cases, a combination of burst cycles and single cycles are required to perform the data transfer. There are also cases for which burst cycles cannot be used and the transfer consists of multiple cycles, each beginning with the ADS# signal.

I/O Writes, I/O Reads, and Memory Writes

If the processor initiates bursting (BLAST# inactive) during an I/O Write, I/O Read or Memory Write, the duration of the burst is a maximum of four bytes (32 bits). All of the possible burst situations are listed in Table 11. In all cases, the burst is two data cycles. The control signals M/IO#, D/C#, W/R#, address bits A31–A4 as well as A3 and A2 remain constant throughout each two-cycle burst.

Table 11. Valid Burst Cycle Sequences—I/O Reads and All Writes

Starting Address (Least significant hexadecimal digit)	Data Cycle	Signals from the Processor				Address of Expected Read Data	
		A3 A2	Byte Enables BE3#–BE0#	A3–A0 (Hex)	BLAST#	D15–D8, DP1	D7–D0, DP0
0, 4, 8, C	1	A3 A2	0 0 0 0	0, 4, 8, C	1	2nd	1st
	2	A3 A2	0 0 1 1	2, 6, A, E	0	4th	3rd
1, 5, 9, D	1	A3 A2	0 0 0 1	1, 5, 9, D	1	1st	-
	2	A3 A2	0 0 1 1	2, 6, A, E	0	3rd	2nd
1, 5, 9, D	1	A3 A2	1 0 0 1	1, 5, 9, D	1	1st	-
	2	A3 A2	1 0 1 1	2, 6, A, E	0	-	2nd
2, 6, A, E	1	A3 A2	1 0 0 0	0, 4, 8, C	1	2nd	1st
	2	A3 A2	1 0 1 1	2, 6, A, E	0	-	3rd

Non-Cacheable Memory Reads

When the processor initiates bursting, the duration of the burst is a maximum of 16 bytes (128 bits).

Non-cacheable instruction prefetches can be 16 bytes in duration. The possible burst sequences are the same as for cache-line transfers listed in Table 11. The burst sequence can be terminated at any time with an active BLAST# signal.

The length of a burst transfer can be 16, eight, or fewer than eight bytes.

For burst lengths of eight or less, the entire burst transfer is confined to a quad-word (eight-byte) data boundary of system memory. A31–A3 remain constant throughout this type of burst transfer.

4.5.3.4 Burst Transfer Address Prediction

The processor provides the data address (A31–A2) and byte enables (BE3#–BE0#) for the first data cycle while ADS# is inactive. The initial values for A1, BHE# and BLE# (A0) can be derived from the byte enables. If bursting is anticipated, the next data address can be predicted at this time and can be used by the memory controller to perform burst data transfers with minimal wait states. Rather than list all of the burst mode address combinations, a general algorithm is provided in Figure 6. This algorithm holds true for all burst transfers including cache-line fills, instruction prefetches, I/O and memory-write data transfers described in earlier sections.

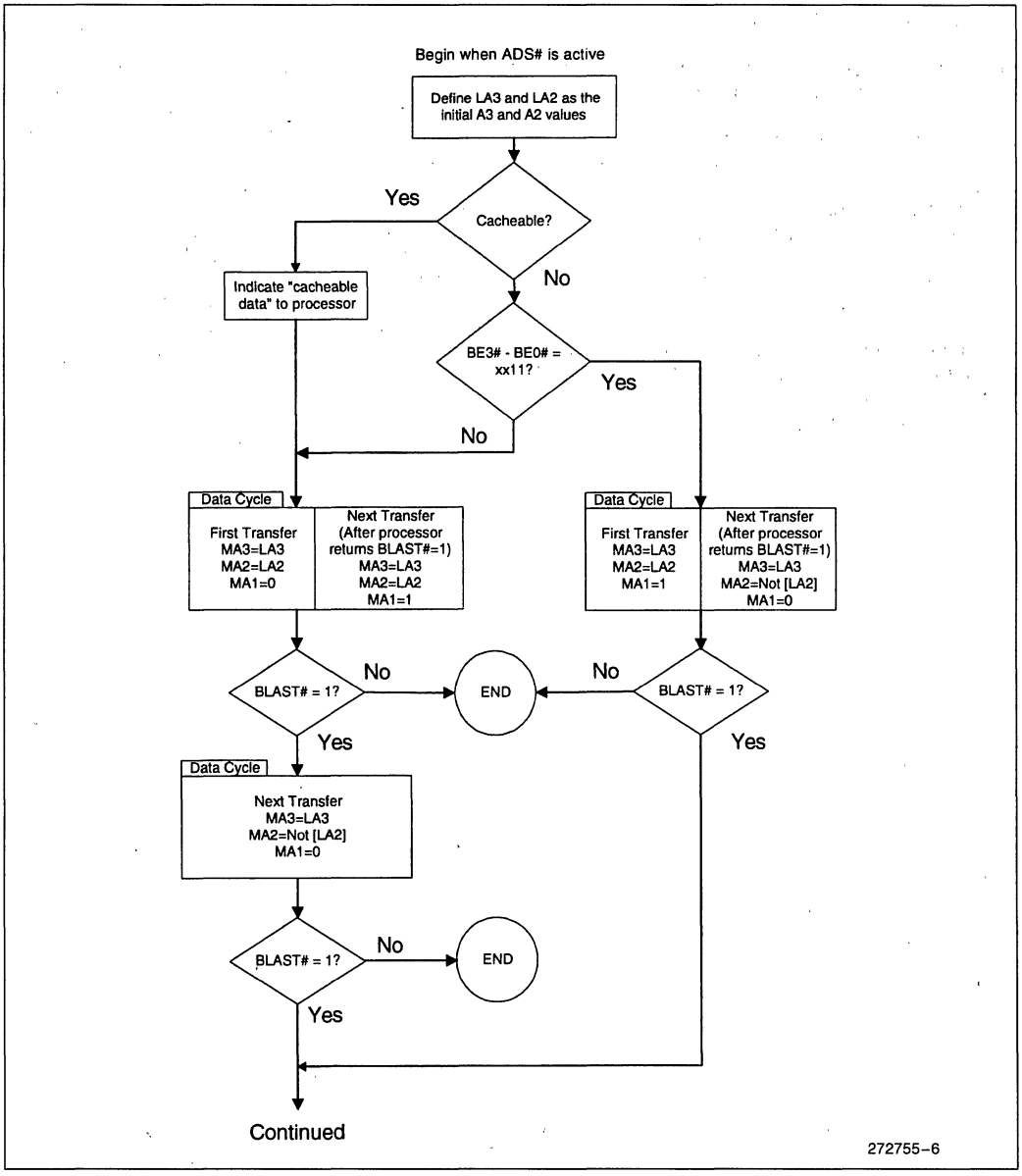


Figure 6. Address Prediction for Burst Transfers (1 of 3)

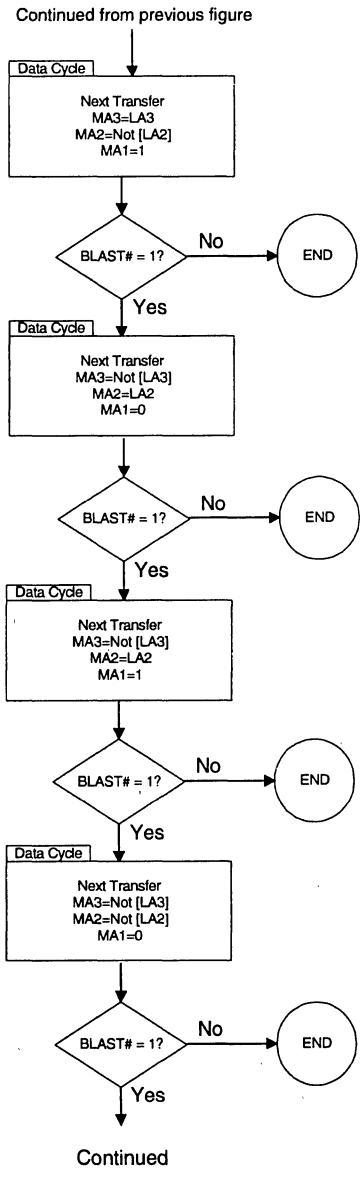


Figure 7. Address Prediction for Burst Transfers (2 of 3)

4

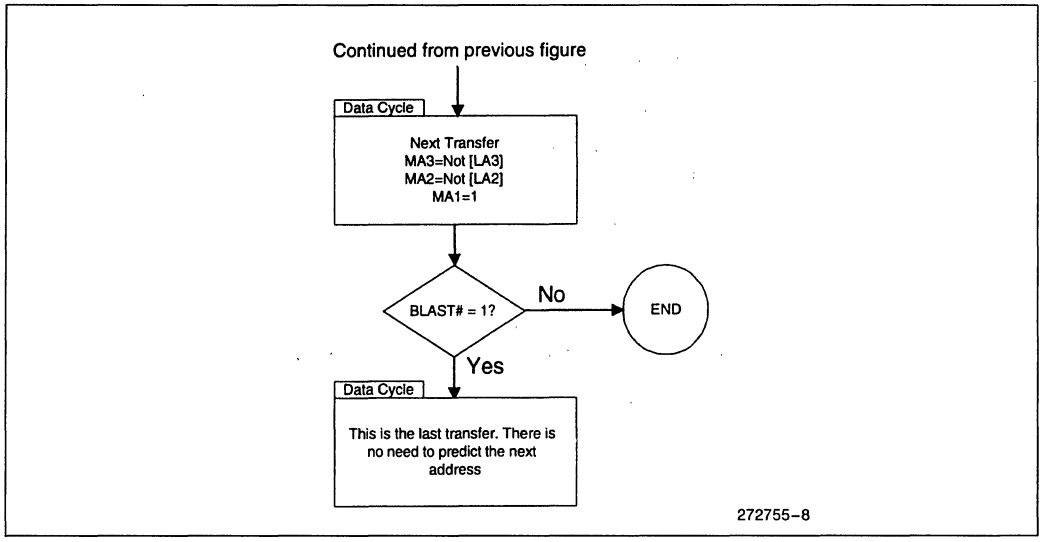


Figure 8. Address Prediction for Burst Transfers (3 of 3)

In the figure, MA3, MA2, and MA1 are memory address bits. LA3 and LA2 are the saved, initial values of A3 and A2 respectively. The term "MA2 = NOT [LA2]" means that MA2 is the opposite logic state as the saved initial A2 value. MA31–MA4 are derived directly from A31–A4, which remain constant throughout the burst transfer. M/IO#, W/R#, and D/C# also remain constant. BLE# (A0) is not shown, but is always active (LOW) throughout the transfer. BHE#, also not shown, cannot be predicted for the last data cycle of a burst transfer and must be decoded from the byte enable bits for the last burst cycle (follows BLAST# = 0). Otherwise BHE# is always active (LOW) throughout the burst. The processor defines "cacheable data" as the case where PCD is inactive (LOW) and LOCK# is inactive (HIGH) and KEN# is active (LOW).

4.6 CPUID Instruction

The embedded ULP Intel486 GX processor supports the CPUID instruction (see Table 12). Because not all Intel processors support the CPUID instruction, a simple test can determine if the instruction is supported. The test involves the processor's ID Flag, which is bit 21 of the EFLAGS register. If software can change the value of this flag, the CPUID instruction is available. The actual state of the ID Flag bit is irrelevant and provides no significance to the hardware. This bit is cleared (reset to zero) upon device reset (RESET or SRESET) for compatibility with Intel486 processor designs that do not support the CPUID instruction.

CPUID-instruction details are provided here for the embedded ULP Intel486 GX processor. Refer to Intel Application Note AP-485 *Intel Processor Identification with the CPUID Instruction* (Order No. 241618) for a description that covers all aspects of the CPUID instruction and how it pertains to other Intel processors.

4.6.1 Operation of the CPUID Instruction

The CPUID instruction requires the software developer to pass an input parameter to the processor in the EAX register. The processor response is returned in registers EAX, EBX, EDX, and ECX.

Table 12. CPUID Instruction Description

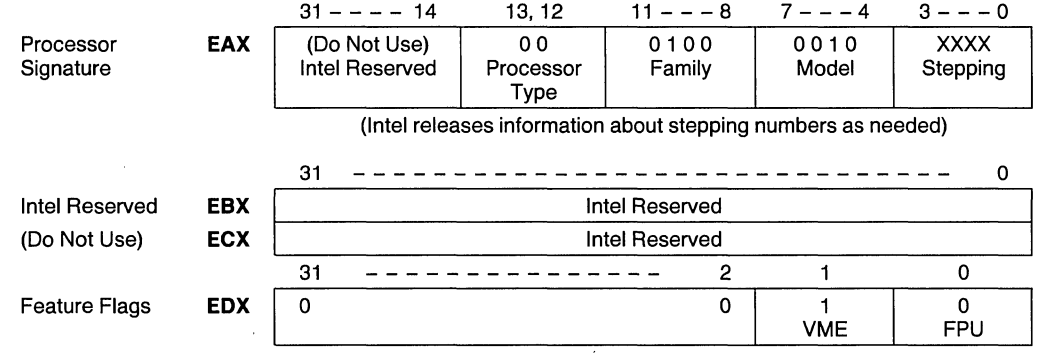
OP CODE	Instruction	Processor Core Clocks	Parameter passed in EAX (Input Value)	Description
0F A2	CPUID	9	0	Vendor (Intel) ID String
		14	1	Processor Identification
		9	> 1	Undefined (Do Not Use)

Vendor ID String—When the parameter passed in EAX is 0 (zero), the register values returned upon instruction execution are shown in the following table.

		31 – – – – – – 24	23 – – – – – – 16	15 – – – – – – 8	7 – – – – – – 0
High Value (= 1)	**EAX**	0000	0000	0000	0001
Vendor ID String (ASCII Characters)	**EBX**	u (75)	n (6E)	e (65)	G (47)
	EDX	I (49)	e (65)	n (6E)	i (69)
	ECX	l (6C)	e (65)	t (74)	n (6E)

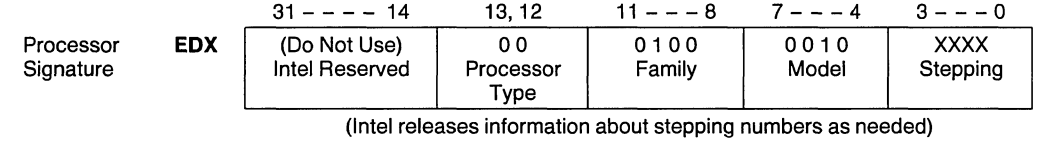
The values in EBX, EDX and ECX indicate an Intel processor. When taken in the proper order, they decode to the string "GenuineIntel."

Processor Identification—When the parameter passed to EAX is 1 (one), the register values returned upon instruction execution are:



4.7 Identification After Reset

Processor Identification—Upon reset, the EDX register contains the processor signature:



4.8 Boundary Scan (JTAG)

4.8.1 Device Identification

Table 13 shows the 32-bit code for the embedded ULP Intel486 GX processor which is loaded into the Device Identification Register.

Table 13. Boundary Scan Component Identification Code

Version	Part Number				Mfg ID	1
	V_{CC} 0=5V 1=3.3V	Intel Architecture Type	Family 0100 = Intel486 CPU Family	Model 00100 = embedded ULP Intel486 SX processor	009H = Intel	
31 - - - 28	27	26 - - - - - 21	20 - - - - 17	16 - - - - - 12	11 - - - - - 1	0
XXXX	1	000001	0100	00010	00000001001	1

(Intel releases information about version numbers as needed)
Boundary Scan Component Identification Code = x828 4013 (Hex)

4.8.2 Boundary Scan Register Bits and Bit Order

The boundary scan register contains a cell for each pin as well as cells for control of bidirectional and three-state pins. There are "Reserved" bits which correspond to no-connect (N/C) signals of the embedded ULP Intel486 GX processor. Control registers WRCTL, ABUSCTL, BUSCTL, and MISCCTL are used to select the direction of bidirectional or three-state output signal pins. A "1" in these cells designates that the associated bus or bits are floated if the pins are three-state, or selected as input if they are bidirectional.

- WRCTL controls D15–D0, DP1 and DP0
- ABUSCTL controls A31–A2
- BUSCTL controls ADS#, BLAST#, PLOCK#, LOCK#, W/R#, BE0#, BE1#, BE2#, BE3#, M/IO#, D/C#, PWT, and PCD
- MISCCTL controls PCHK#, HLDA, and BREQ

The following is the bit order of the embedded ULP Intel486 GX processor boundary scan register:

TDO ← A2, A3, A4, A5, RESERVED, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, STPCLK#, Reserved, Reserved, SMI#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, Reserved, Reserved, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRCTL **← TDI**

5.0 ELECTRICAL SPECIFICATIONS

5.1 Maximum Ratings

Table 14 is a stress rating only. Extended exposure to the Maximum Ratings may affect device reliability.

Furthermore, although the embedded ULP Intel486 GX processor contains protective circuitry to resist damage from electrostatic discharge, always take precautions to avoid high static voltages or electric fields.

Functional operating conditions are given in **Section 5.2, DC Specifications** and **Section 5.3, AC Specifications.**

Table 14. Absolute Maximum Ratings

Case Temperature under Bias	−65°C to +110°C
Storage Temperature	−65°C to +150°C
DC Voltage on Any Pin with Respect to Ground	−0.5V to V_{CCP} + 0.5V
Supply Voltage V_{CC} with Respect to V_{SS}	−0.5V to +4.6V
Supply Voltage V_{CCP} with Respect to V_{SS}	−0.5V to +4.6V

5.2 DC Specifications

The following tables show the operating supply voltages, DC I/O specifications, and component power consumption for the embedded ULP Intel486 GX processor.

Table 15. Operating Supply Voltages

Product	V_{CCP} Range(1)	Max. CLK Frequency	V_{CC} Range(2)	V_{CC} Fluctuation
FA80486GXSF-33	3.3V ± 0.3V	16	2.0V min 3.3V max	±0.2V at 2.0V ≤ V_{CC} ≤ 2.7V +0.3V/−0.2V at 2.7V < V_{CC} < 3.0V ±0.3V at 3.0V ≤ V_{CC} ≤ 3.3V
		20	2.2V min 3.3V max	
		25	2.4V min 3.3V max	
		33	2.7V min 3.3V max	

NOTES:
1. In all cases, V_{CCP} must be ≥ VCC.
2. V_{CC} may be set to any voltage within the V_{CC} Range. The setting determines the allowed V_{CC} Fluctuation.

Table 16. DC Specifications

T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min.	Max.	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CCP} +0.3	V	(Note 1)
V_{IHC}	Input HIGH Voltage of CLK	V_{CCP} −0.6	V_{CCP} +0.3	V	
V_{OL}	Output LOW Voltage I_{OL} = 2.0 mA I_{OL} = 100 μA		 0.4 0.2	 V V	
V_{OH}	Output HIGH Voltage I_{OH} = −2.0 mA I_{OH} = −100 μA	 2.4 V_{CCP} −0.2		 V V	
I_{LI}	Input Leakage Current		±15	μA	(Note 2)
I_{IH}	Input Leakage Current		200	μA	(Note 3)
I_{IL}	Input Leakage Current		−400	μA	(Note 4)
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		10	pF	(Note 5)
C_{OUT}	I/O or Output Capacitance		10	pF	(Note 5)
C_{CLK}	CLK Capacitance		6	pF	(Note 5)

NOTES:
1. All inputs except CLK.
2. This parameter is for inputs without pull-up or pull-down resistors and $0V \leq V_{IN} \leq V_{CCP}$.
3. This parameter is for inputs with pull-down resistors and V_{IH} = 2.4V, and for level-keeper pins at V = 0.4V.
4. This parameter is for inputs with pull-up resistors and V_{IL} = 0.4V, and for level-keeper pins at V = 2.4V.
5. F_C=1 MHz. Not 100% tested.

Table 17. Active I_{CC} Values

T_{CASE} = 0°C to +85°C

Symbol	Parameter	Frequency	Supply Voltage	Typical I_{CC}	Max. I_{CC}	Notes
I_{CC1}	I_{CC} Active (V_{CC} pins)	16 MHz	V_{CC} = 2.0 ± 0.2V	65 mA	105 mA	
			V_{CC} = 3.3 ± 0.3V	105 mA	170 mA	
		20 MHz	V_{CC} = 2.2 ± 0.2V	85 mA	140 mA	
			V_{CC} = 3.3±0.3V	130 mA	210 mA	
		25 MHz	V_{CC} = 2.4 ± 0.2V	120 mA	195 mA	
			V_{CC} = 3.3 ± 0.3V	165 mA	260 mA	
		33 MHz	V_{CC} = 2.7 ± 0.2V	180 mA	280 mA	
			V_{CC} = 3.3 ± 0.3V	220 mA	345 mA	
I_{CC2}	I_{CC} Active (V_{CCP} pins)	16 MHz	V_{CCP} = 3.3 ± 0.3V	5 mA	16 mA	1
		20 MHz	V_{CCP} = 3.3 ± 0.3V	6 mA	20 mA	1
		25 MHz	V_{CCP} = 3.3 ± 0.3V	7 mA	25 mA	1
		33 MHz	V_{CCP} = 3.3 ± 0.3V	9 mA	32 mA	1

NOTE:
1. These parameters are for C_L = 50 pF

4

Table 18. Clock Stop, Stop Grant, and Auto HALT Power Down I_{CC} Values

T_{CASE} = 0°C to +85°C

Symbol	Parameter	Frequency	Supply Voltage	Typical I_{CC}	Max. I_{CC}	Notes
I_{CCS0}	I_{CC} Stop Clock (V_{CC} pins)	0 MHz	V_{CC} = 2.0 ± 0.2V	3 μA	105 μA	Note 1
			V_{CC} = 2.2 ± 0.2V	3 μA	110 μA	
			V_{CC} = 2.4 ± 0.2V	4 μA	120 μA	
			V_{CC} = 2.7 ± 0.2V	4 μA	130 μA	
			V_{CC} = 3.3 ± 0.3V	5 μA	150 μA	
I_{CCS2}	I_{CC} Stop Clock (V_{CCP} pins)	0 MHz	V_{CCP} = 3.3 ± 0.3V	3 μA	80 μA	
I_{CCS1}	I_{CC} Stop Grant, Auto HALT Power Down (V_{CC} pins)	16 MHz	V_{CC} = 2.0 ± 0.2V	8 mA	15 mA	
			V_{CC} = 3.3 ± 0.3V	12 mA	20 mA	
		20 MHz	V_{CC} = 2.2 ± 0.2V	10 mA	20 mA	
			V_{CC} = 3.3 ± 0.3V	15 mA	25 mA	
		25 MHz	V_{CC} = 2.4 ± 0.2V	14 mA	25 mA	
			V_{CC} = 3.3 ± 0.3V	20 mA	30 mA	
		33 MHz	V_{CC} = 2.7 ± 0.2V	20 mA	30 mA	
			V_{CC} = 3.3 ± 0.3V	25 mA	35 mA	
I_{CCS3}	I_{CC} Stop Grant, Auto HALT Power Down (V_{CCP} pins)	16 MHz	V_{CCP} = 3.3 ± 0.3V	270 μA	1.0 mA	
		20 MHz	V_{CCP} = 3.3 ± 0.3V	340 μA	1.2 mA	
		25 MHz	V_{CCP} = 3.3 ± 0.3V	425 μA	1.5 mA	
		33 MHz	V_{CCP} = 3.3 ± 0.3V	610 μA	2.0 mA	

NOTE:
1. The I_{CC} Stop Clock specification refers to the I_{CC} value once the processor enters the Stop Clock state. For all input signals, the V_{IH} and V_{IL} levels must be equal to V_{CCP} and 0V, respectively, to meet the I_{CC} Stop Clock specifications.

5.3 AC Specifications

The AC specifications for the embedded ULP Intel486 GX processor are given in this section.

Table 19. AC Characteristics (Sheet 1 of 2)

valid over the operating supply voltages listed in Table 15, Operating Supply Voltages.
T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	2.0V ≤ V_{CC} < 2.2V		2.2V ≤ V_{CC} < 2.4V		2.4V ≤ V_{CC} < 2.7V		2.7V ≤ V_{CC} ≤ 3.3V		Unit	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
	Frequency	0	16	0	20	0	25	0	33	MHz	(Note 1)
t_1	CLK Period	62.5		50		40		30		ns	(Note 1)
t_{1a}	CLK Period Stability		250		250		250		250	ps/CLK	(Note 2)
t_2	CLK High Time	23		18		14		11		ns	at 2V
t_3	CLK Low Time	23		18		14		11		ns	at 0.8V
t_4	CLK Fall Time		4		4		4		3	ns	2V to 0.8V (Note 3)
t_5	CLK Rise Time		4		4		4		3	ns	0.8V to 2V (Note 3)
t_6	A2–A31, PWT, PCD, BE0#–BE3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT# Valid Delay	3	30	3	24	3	19	3	16	ns	
t_7	A2–A31, PWT, PCD, BE0#–BE3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, Float Delay		36		30		28		20	ns	(Note 3)
t_8	PCHK# Valid Delay	3	34	3	29	3	24	3	22	ns	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	34	3	29	3	24	3	20	ns	
t_9	BLAST#, PLOCK# Float Delay		36		30		28		20	ns	(Note 3)
t_{10}	D0–D15, DP0, DP1 Write Delay	3	31	3	26	3	20	3	19	ns	
t_{11}	D0–D15, DP0, DP1 Float Delay		36		30		28		20	ns	(Note 3)
t_{12}	EADS# Setup Time	13		11		8		6		ns	
t_{13}	EADS# Hold Time	4		4		3		3		ns	
t_{14}	KEN# Setup Time	13		11		8		6		ns	
t_{15}	KEN# Hold Time	4		4		3		3		ns	
t_{16}	RDY#, BRDY# Setup Time	13		11		8		6		ns	
t_{17}	RDY#, BRDY# Hold Time	4		4		3		3		ns	
t_{18}	HOLD, AHOLD Setup Time	15		13		10		6		ns	

Table 19. AC Characteristics (Sheet 2 of 2)

valid over the operating supply voltages listed in Table 15, Operating Supply Voltages.
T_{CASE} = 0°C to +85°C; C_L = 50 pF (Continued)

Symbol	Parameter	2.0V ≤ V_{CC} < 2.2V		2.2V ≤ V_{CC} < 2.4V		2.4V ≤ V_{CC} < 2.7V		2.7V ≤ V_{CC} ≤ 3.3V		Unit	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
t_{18a}	BOFF# Setup Time	15		13		10		9		ns	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	4		4		3		3		ns	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET Setup Time	15		13		10		6		ns	
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET Hold Time	4		4		3		3		ns	
t_{22}	D0–D15, DP0, DP1, A4–A31 Read Setup Time	11		8		6		6		ns	
t_{23}	D0–D15, DP0, DP1, A4–A31 Read Hold Time	4		4		3		3		ns	

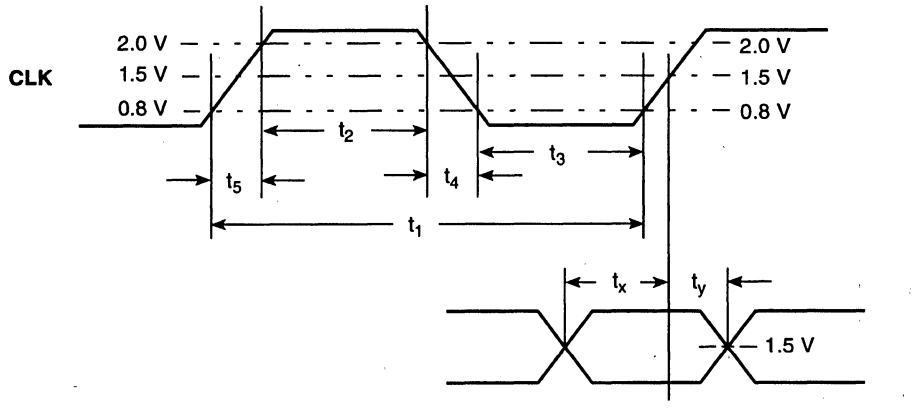
NOTES:
1. 0 Hz operation is tested and guaranteed by the STPCLK# and Stop Grant bus cycle protocol. 0 Hz < CLK < 8 MHz operation is confirmed by design characterization, but not 100% tested in production.
2. Specification t_{1a} is available only when CLK frequency is changed without STPCLK#/STOP GRANT bus cycle protocol.
3. Not 100% tested, guaranteed by design characterization.
4. CLK reference voltage for timing measurement is 1.5V except t_2 through t_5. Other signals are measured at 1.5V.

Table 20. AC Specifications for the Test Access Port

Symbol	Parameter	1.8V ≤ V_{CC} < 3.0V		V_{CC} = 3.3 ± 0.3V		Unit	Figure	Notes
		Min	Max	Min	Max			
t_{24}	TCK Frequency		5		8	MHz	15	
t_{25}	TCK Period	200		125		ns	15	Note 1
t_{26}	TCK High Time	65		40		ns	15	@ 2.0V
t_{27}	TCK Low Time	65		40		ns	15	@ 0.8V
t_{28}	TCK Rise Time		15		8	ns	15	Note 2
t_{29}	TCK Fall Time		15		8	ns	15	Note 2
t_{30}	TDI, TMS Setup Time	16		8		ns	16	Note 3
t_{31}	TDI, TMS Hold Time	20		10		ns	16	Note 3
t_{32}	TDO Valid Delay	3	46	3	30	ns	16	Note 3
t_{33}	TDO Float Delay		52		36	ns	16	Notes 3, 4
t_{34}	All Outputs (except TDO) Valid Delay	3	80	3	30	ns	16	Note 3
t_{35}	All Outputs (except TDO) Float Delay		88		36	ns	16	Notes 3, 4
t_{36}	All Inputs (except TDI, TMS, TCK) Setup Time	16		8		ns	16	Note 3
t_{37}	All Inputs (except TDI, TMS, TCK) Hold Time	35		15		ns	16	Note 3

NOTES:
1. TCK period ≥ CLK period.
2. Rise/Fall Times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10 ns increase in TCK period.
3. Parameter measured from TCK.
4. Not 100% tested, guaranteed by design characterization.

4



t_x = input setup times
t_y = input hold times, output float, valid and hold times

Figure 9. CLK Waveform

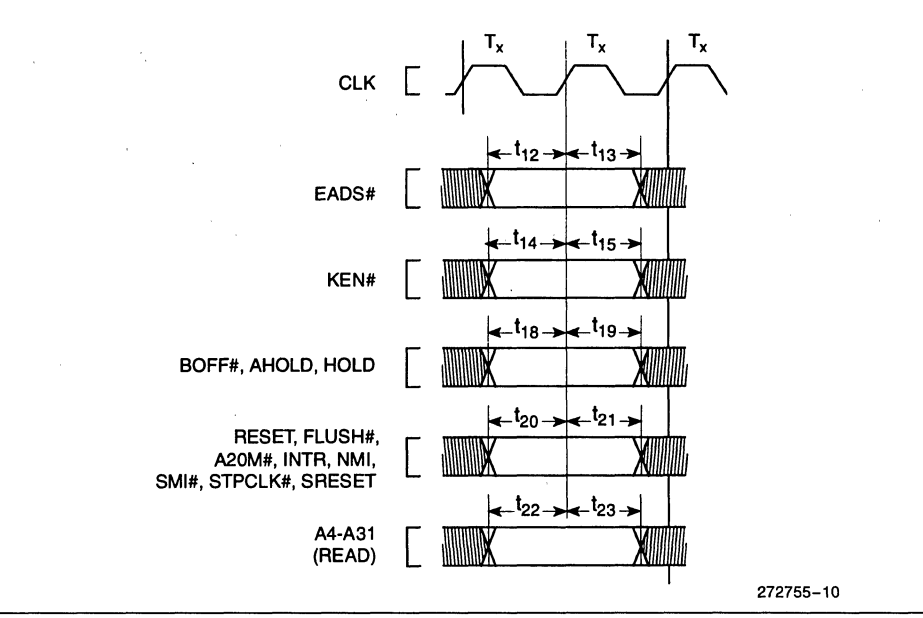


Figure 10. Input Setup and Hold Timing

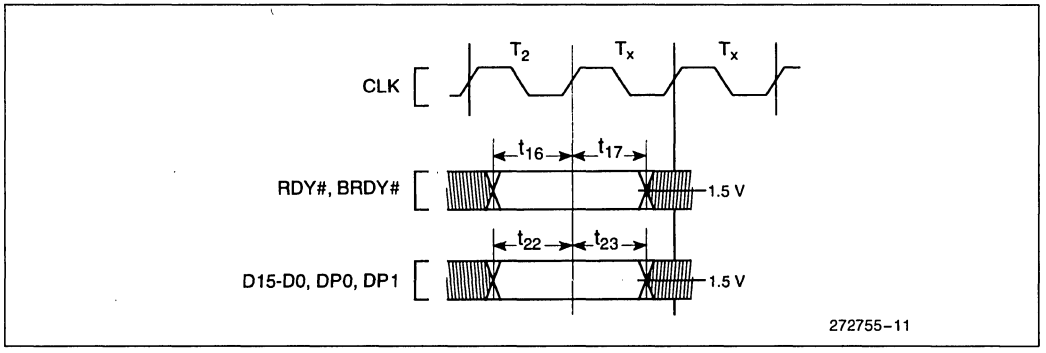


Figure 11. Input Setup and Hold Timing

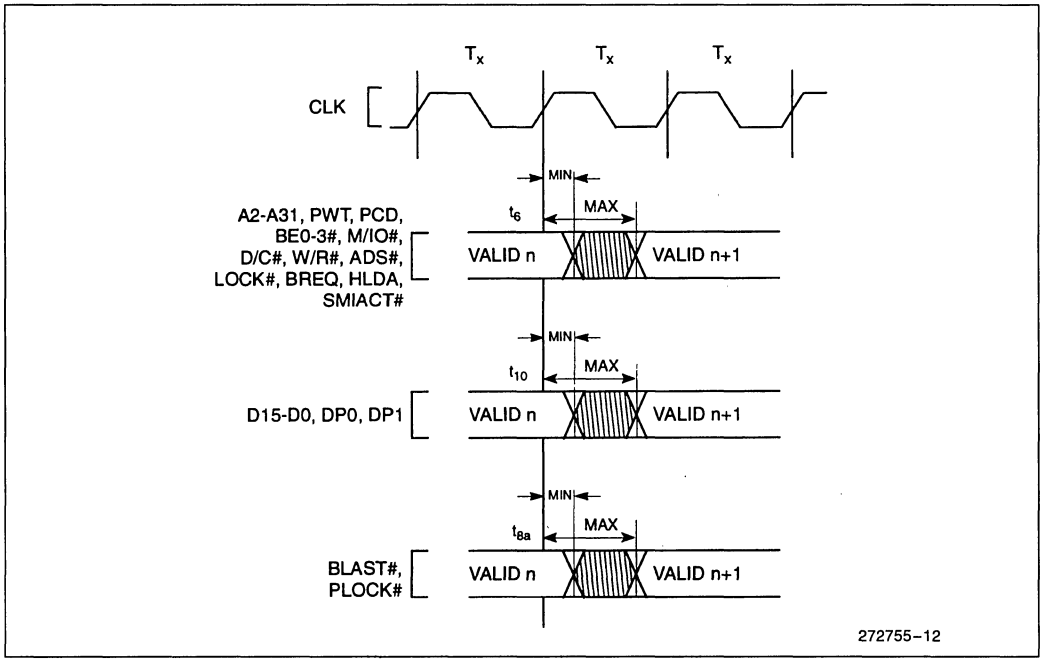


Figure 12. Output Valid Delay Timing

4

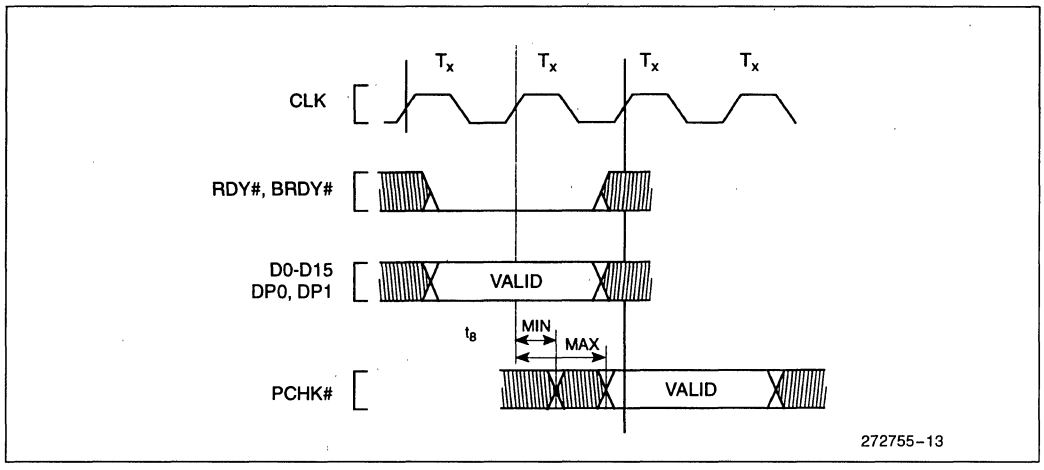


Figure 13. PCHK# Valid Delay Timing

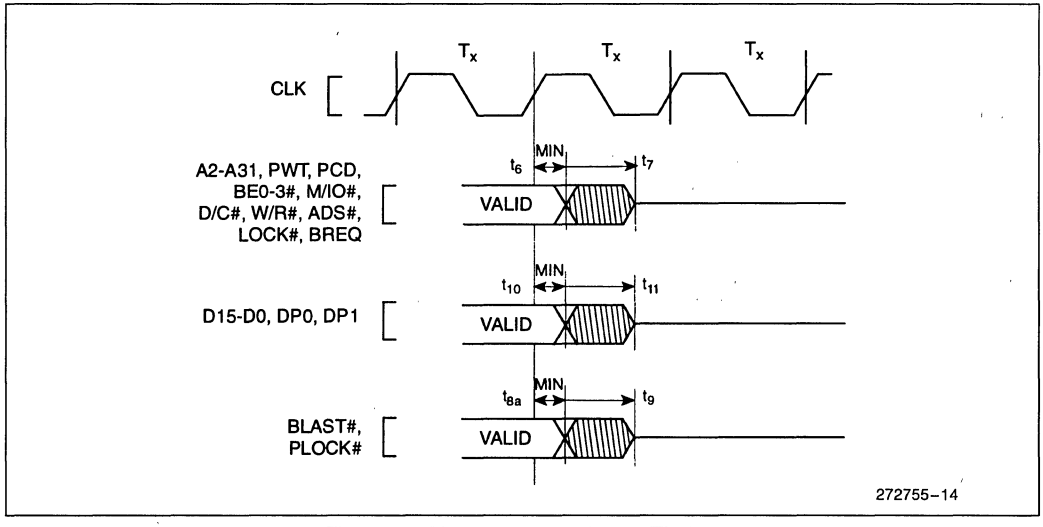


Figure 14. Maximum Float Delay Timing

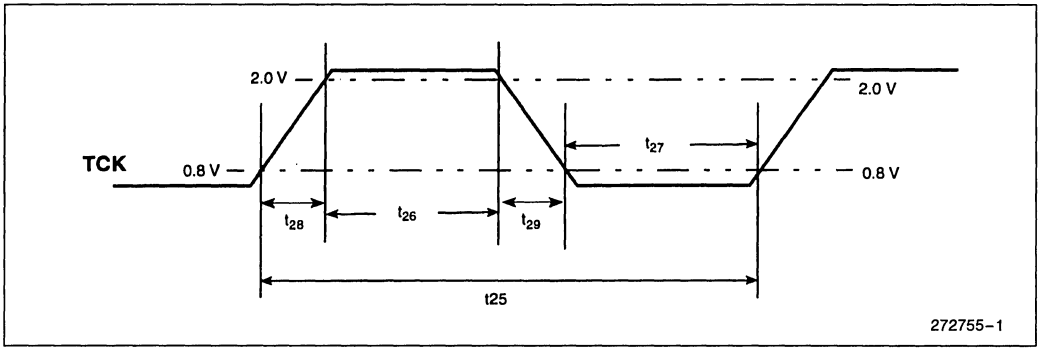


Figure 15. TCK Waveform

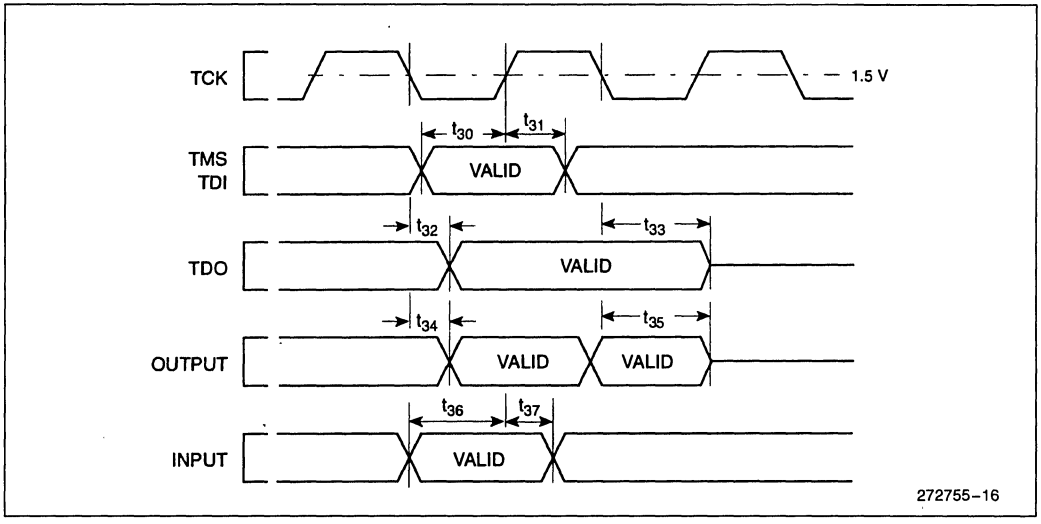
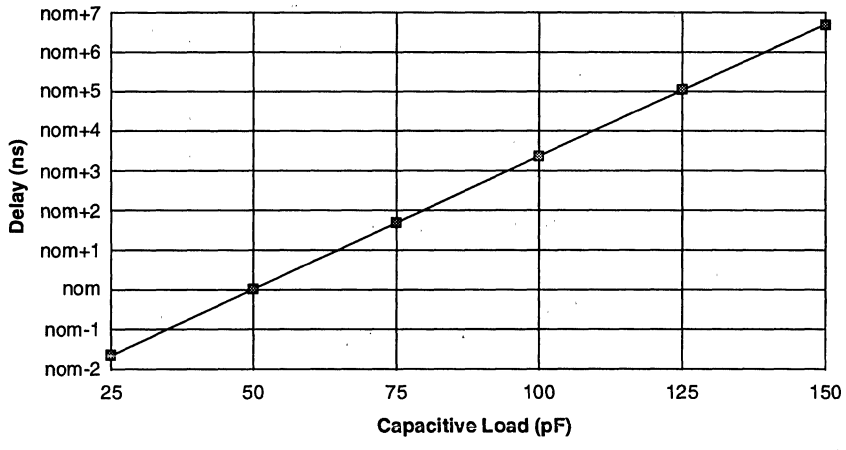


Figure 16. Test Signal Timing Diagram

4

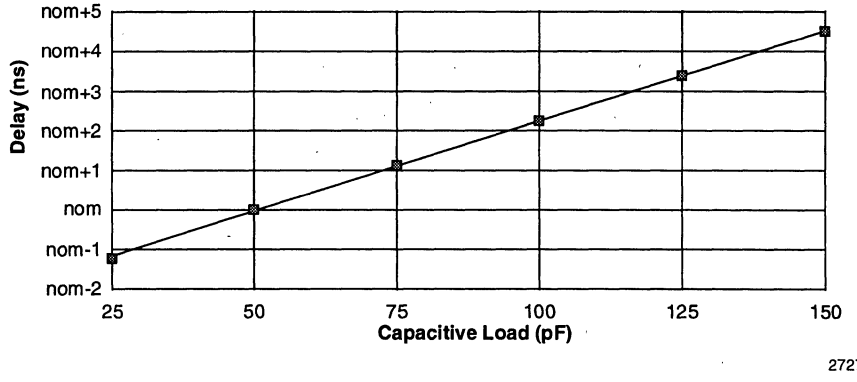
5.4 Capacitive Derating Curves

The following graphs are the capacitive derating curves for the embedded ULP Intel486 GX processor.



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



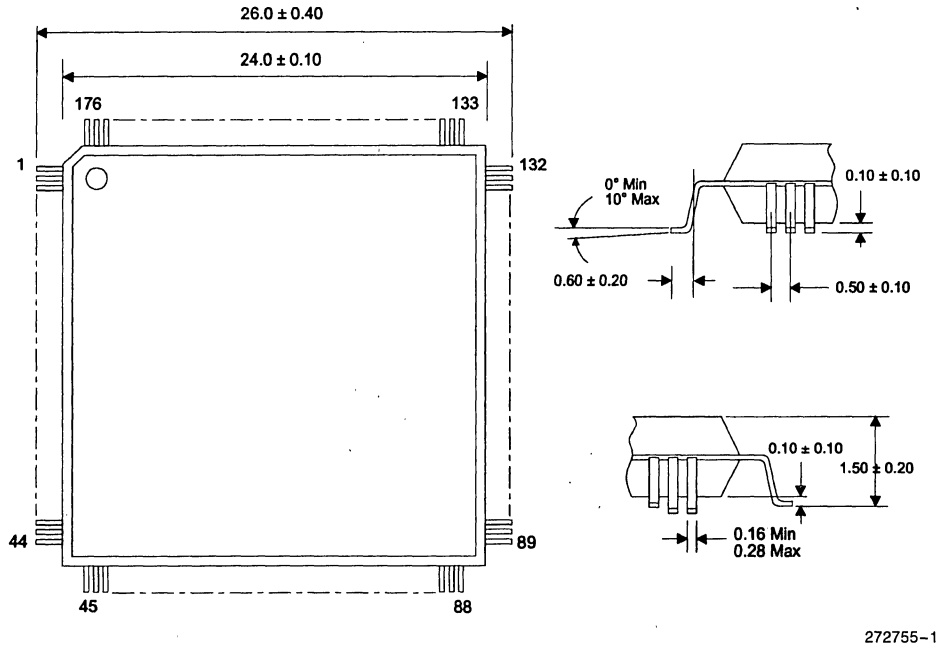
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 18. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition

6.0 MECHANICAL DATA

This section describes the packaging dimensions and thermal specifications for the embedded ULP Intel486 GX processor.

6.1 Package Dimensions



NOTE:
Height Measurements same as width Measurements

Units:mm

Figure 19. Package Mechanical Specifications for the 176 Lead TQFP Package

4

6.2 Package Thermal Specifications

The embedded ULP Intel486 GX processor is specified for operation when the case temperature (T_C) is within the range of 0°C to 85°C. T_C may be measured in any environment to determine whether the processor is within the specified operating range.

The ambient temperature (T_A) can be calculated from θ_{JC} and θ_{JA} from the following equations:

$$T_J = T_C + P * \theta_{JC}$$
$$T_A = T_J - P * \theta_{JA}$$
$$T_C = T_A + P * [\theta_{JA} - \theta_{JC}]$$
$$T_A = T_C - P * [\theta_{JA} - \theta_{JC}]$$

Where T_J, T_A, T_C equals Junction, Ambient and Case Temperature respectively. θ_{JC}, θ_{JA} equals Junction-to-Case and Junction-to-Ambient thermal Resistance, respectively. Maximum Power Consumption (P) is defined as

$$P = V\ (typ) * I_{CC}\ (max)$$
$$P = [V_{CC}\ (typ) * I_{CC1}\ (max)] + [V_{CCP}\ (typ) * I_{CC2}\ (max)]$$

where: I_{CC1} is the V_{CC} supply current
I_{CC2} is the V_{CCP} supply current

Values for θ_{JA} and θ_{JC} are given in the following tables for each product at its maximum operating frequencies.

Table 21. Thermal Resistance

(°C/W) θ_{JC} and θ_{JA} for the 176-Lead TQFP Package

θ_{JC} (°C/W)	θ_{JA} (°C/W) with no airflow
4.3	33.6

The following table shows maximum ambient temperatures of the embedded ULP Intel486 GX processor for each product and maximum operating frequencies. These temperatures are calculated using I_{CC1} and I_{CC1} values measured during component-validation testing using V_{CCP} = 3.6V and worst-case V_{CC} values.

Table 22. Maximum Ambient Temperature (T_A)

176-Lead TQFP Package

Frequency	V_{CC}	T_A (°C) with no airflow
16 MHz	2.0V	83
	3.3V	73
20 MHz	2.2V	80
	3.3V	70
25 MHz	2.4V	77
	3.3V	66
33 MHz	2.7V	70
	3.3V	60

EMBEDDED ULTRA-LOW POWER Intel486™ SX PROCESSOR

- **Ultra-Low Power Version of the Intel486 SX Processor**
  - **— 32-Bit RISC Technology Core**
  - **— 8-Kbyte Write-Through Cache**
  - **— Four Internal Write Buffers**
  - **— Burst Bus Cycles**
  - **— Dynamic Bus Sizing for 8- and 16-bit Data Bus Devices**
  - **— Intel System Management Mode (SMM)**
  - **— Boundary Scan (JTAG)**
- **176-Lead Thin Quad Flat Pack (TQFP)**
- **Separate Voltage Supply for Core Circuitry**
- **Fast Core-Clock Restart**
- **Auto Clock Freeze**
- **Ideal for Embedded Battery-Operated and Hand-Held Applications**

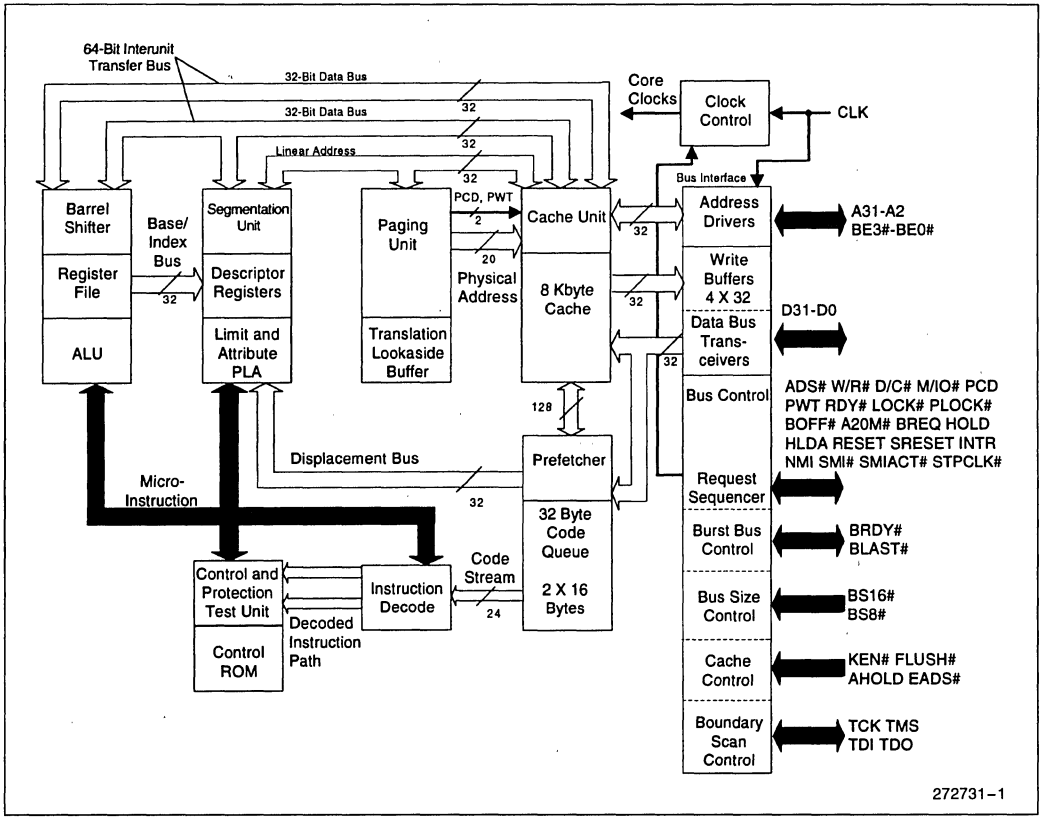


Figure 1. Embedded ULP Intel486™ SX Processor Block Diagram

EMBEDDED ULTRA-LOW POWER Intel486™ SX PROCESSOR

CONTENTS

CONTENTS

PAGE

FIGURES

CONTENTS

PAGE

TABLES

1.0 INTRODUCTION

This data sheet describes the embedded Ultra-Low Power (ULP) Intel486™ SX processor. It is intended for embedded battery-operated and hand-held applications. The embedded ULP Intel486 SX processor provides all of the features of the Intel486 SX processor except for the external data-bus parity logic and the processor-upgrade pin. The processor typically uses 20% to 50% less power than the Intel486 SX processor. Additionally, the embedded ULP Intel486 SX processor external data bus has level-keeper circuitry and a fast-recovery core clock which are vital for ultra-low-power system designs. The processor is available in a Thin Quad Flat Package (TQFP) enabling low-profile component implementation.

The embedded ULP Intel486 SX processor consists of a 32-bit integer processing unit, an on-chip cache, and a memory management unit. The design ensures full instruction-set compatibility with the 8086, 8088, 80186, 80286, Intel386™ SX, Intel386 DX, and all versions of Intel486 processors.

1.1 Features

The embedded ULP Intel486 SX processor offers these features of the Intel486 SX processor:

- **32-bit RISC-Technology Core**—The embedded ULP Intel486 SX processor performs a complete set of arithmetic and logical operations on 8-, 16-, and 32-bit data types using a full-width ALU and eight general purpose registers.
- **Single Cycle Execution**—Many instructions execute in a single clock cycle.
- **Instruction Pipelining**—Overlapped instruction fetching, decoding, address translation and execution.
- **On-Chip Cache with Cache Consistency Support**—An 8-Kbyte, write-through, internal cache is used for both data and instructions. Cache hits provide zero wait-state access times for data within the cache. Bus activity is tracked to detect alterations in the memory represented by the internal cache. The internal cache can be invalidated or flushed so that an external cache controller can maintain cache consistency.
- **External Cache Control**—Write-back and flush controls for an external cache are provided so the processor can maintain cache consistency.
- **On-Chip Memory Management Unit**—Address management and memory space protection mechanisms maintain the integrity of memory in a multitasking and virtual memory environment. Both segmentation and paging are supported.
- **Burst Cycles**—Burst transfers allow a new double word to be read from memory on each bus clock cycle. This capability is especially useful for instruction prefetch and for filling the internal cache.
- **Write Buffers**—The processor contains four write buffers to enhance the performance of consecutive writes to memory. The processor can continue internal operations after a write to these buffers, without waiting for the write to be completed on the external bus.
- **Bus Backoff**—When another bus master needs control of the bus during a processor initiated bus cycle, the embedded ULP Intel486 SX processor floats its bus signals, then restarts the cycle when the bus becomes available again.
- **Instruction Restart**—Programs can continue execution following an exception generated by an unsuccessful attempt to access memory. This feature is important for supporting demand-paged virtual memory applications.
- **Dynamic Bus Sizing**—External controllers can dynamically alter the effective width of the data bus. Bus widths of 8, 16, or 32 bits can be used.
- **Boundary Scan (JTAG)**—Boundary Scan provides in-circuit testing of components on printed circuit boards. The Intel Boundary Scan implementation conforms with the IEEE Standard Test Access Port and Boundary Scan Architecture.
- **Intel System Management Mode (SMM)**—A unique Intel architecture operating mode provides a dedicated special purpose interrupt and address space that can be used to implement intelligent power management and other enhanced functions in a manner that is completely transparent to the operating system and applications software.
- **I/O Restart**—An I/O instruction interrupted by a System Management Interrupt (SMI#) can automatically be restarted following the execution of the RSM instruction.

- **Stop Clock**—The embedded ULP Intel486 SX processor has a stop clock control mechanism that provides two low-power states: a Stop Grant state (40–85 mW typical, depending on input clock frequency) and a Stop Clock state (~60 μW typical, with input clock frequency of 0 MHz).
- **Auto HALT Power Down**—After the execution of a HALT instruction, the embedded ULP Intel486 SX processor issues a normal Halt bus cycle and the clock input to the processor core is automatically stopped, causing the processor to enter the Auto HALT Power Down state (40–85 mW typical, depending on input clock frequency).

The embedded ULP Intel486 SX processor differs from the Intel486 SX processor in the following areas:

- **Processor Upgrade Removed**—The UP# signal is not provided.
- **Parity Signals Removed**—The DP3–DP0 and PCHK# signals are not provided.
- **Separate Processor-Core Power**—While the embedded ULP Intel486 SX processor requires a supply voltage of 3.3V, the processor core has dedicated V_{CC} pins and operates with a supply voltage as low as 2.4V.
- **Small, Low-Profile Package**—The 176-Lead Thin Quad Flat Pack (TQFP) package is approximately 26 mm square and only 1.5 mm in height. This is approximately the diameter and thickness of a U.S. quarter. The embedded ULP Intel486 SX processor is ideal for embedded hand-held and battery-powered applications.
- **Level Keeper Circuits**—The embedded ULP Intel486 SX processor has level-keeper circuits for its 32-bit external data bus signals. They retain valid high and low logic voltage levels when the processor is in the Stop Grant and Stop Clock states. This is a power-saving improvement from the floating data bus of the Intel486 SX processor.
- **Auto Clock Freeze**—The embedded ULP Intel486 SX processor monitors bus events and internal activity. The Auto Clock Freeze feature automatically controls internal clock distribution, turning off clocks to internal units when they are idle. This power-saving function is transparent to the embedded system.
- **Fast Clock Restart**—The embedded ULP Intel486 SX processor requires only eight clock periods to synchronize its internal clock with the CLK input signal. This provides for faster transition from the Stop Clock State to the Normal State. For 33-MHz operation, this synchronization time is only 240 ns compared with 1 ms (PLL startup latency) for the Intel486 processor.

1.2 Family Members

Table 1 shows the embedded ULP Intel486 SX processor and briefly describes its characteristics.

Table 1. The Embedded Ultra-Low Power Intel486™ SX Processor

Product	Supply Voltage (V_{CCP})	Processor Core Supply Voltage (V_{CC})	Processor Frequency (MHz)	Package
FA80486SXSF-33	3.3V	2.4V to 3.3V	25	176-Lead TQFP
		2.7V to 3.3	33	

2.0 HOW TO USE THIS DOCUMENT

The embedded ULP Intel486 SX processor has characteristics similar to the Intel486 SX processor. This document describes the new features of the embedded ULP Intel486 SX processor. Some Intel486 SX processor information is also included to minimize the dependence on the reference documents.

For a complete set of documentation related to the embedded ULP Intel486 SX processor, use this document in conjunction with the following reference documents:

- *Intel486™ Processor Family* datasheet—Order No. 242202
- *Intel486 Microprocessor Family Programmer's Reference Manual*—Order No. 240486
- Intel Application Note AP-485—*Intel Processor Identification with the CPUID Instruction*—Order No. 241618

3.0 PIN DESCRIPTIONS

3.1 Pin Assignments

The following figures and tables show the pin assignments for the 176-pin Thin Quad Flat Pack (TQFP) package of the embedded ULP Intel486 SX processor. Included are:

The tables and figures show "no-connects" as "N/C." These pins should always remain unconnected. Connecting N/C pins to V_{CC}, V_{CCP}, V_{SS}, or any other signal pin can result in component malfunction or incompatibility with future steppings of the embedded ULP Intel486 SX processor.

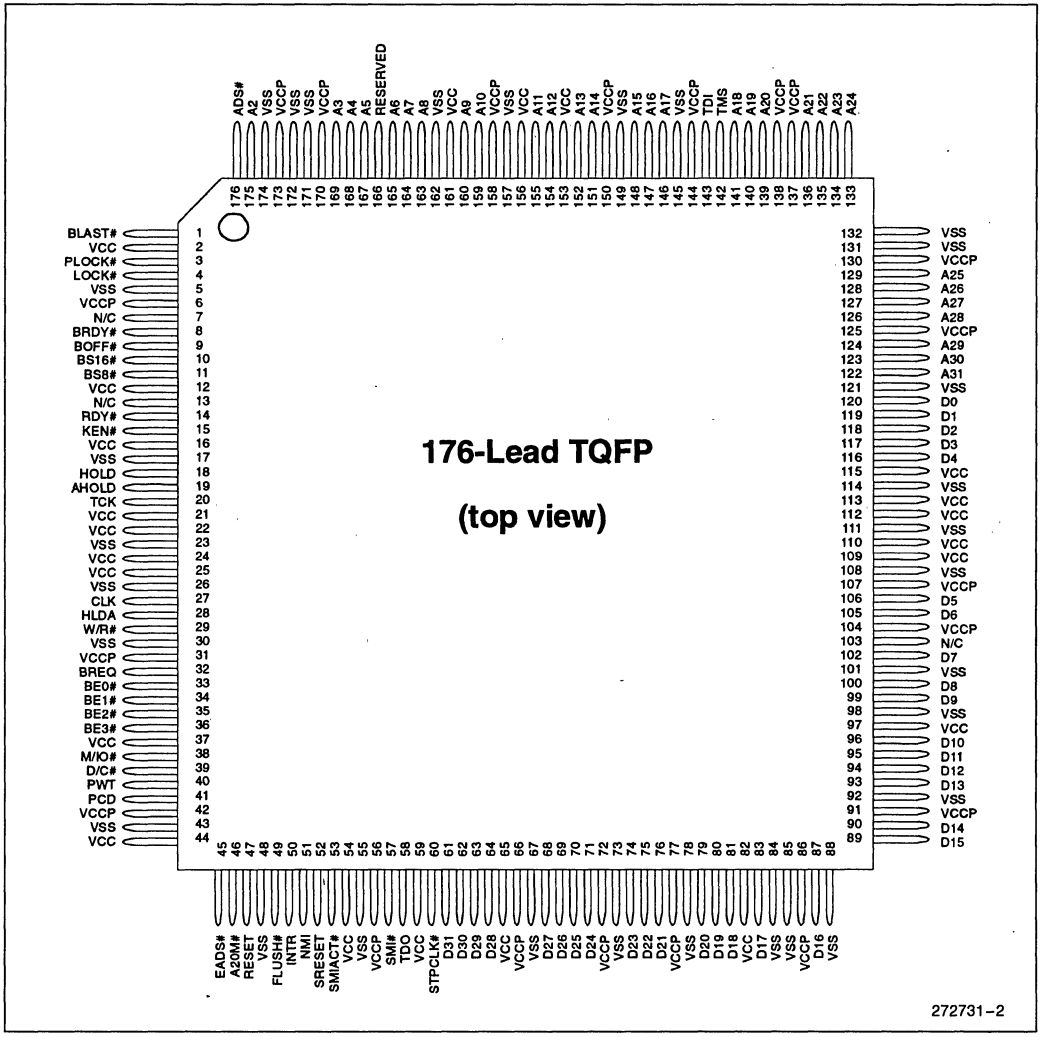


Figure 2. Package Diagram for 176-Lead TQFP Package Embedded ULP Intel486™ SX Processor

Table 2. Pin Assignment for 176-Lead TQFP Package
Embedded ULP Intel486™ SX Processor

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
1	BLAST#	45	EADS#	89	D15	133	A24
2	V_{CC}	46	A20M#	90	D14	134	A23
3	PLOCK#	47	RESET	91	V_{CCP}	135	A22
4	LOCK#	48	V_{SS}	92	V_{SS}	136	A21
5	V_{SS}	49	FLUSH#	93	D13	137	V_{CCP}
6	V_{CCP}	50	INTR	94	D12	138	V_{CCP}
7	N/C	51	NMI	95	D11	139	A20
8	BRDY#	52	SRESET	96	D10	140	A19
9	BOFF#	53	SMIACT#	97	V_{CC}	141	A18
10	BS16#	54	V_{CC}	98	V_{SS}	142	TMS
11	BS8#	55	V_{SS}	99	D9	143	TDI
12	V_{CC}	56	V_{CCP}	100	D8	144	V_{CCP}
13	N/C	57	SMI#	101	V_{SS}	145	V_{SS}
14	RDY#	58	TDO	102	D7	146	A17
15	KEN#	59	V_{CC}	103	N/C	147	A16
16	V_{CC}	60	STPCLK#	104	V_{CCP}	148	A15
17	V_{SS}	61	D31	105	D6	149	V_{SS}
18	HOLD	62	D30	106	D5	150	V_{CCP}
19	AHOLD	63	D29	107	V_{CCP}	151	A14
20	TCK	64	D28	108	V_{SS}	152	A13
21	V_{CC}	65	V_{CC}	109	V_{CC}	153	VCC
22	V_{CC}	66	V_{CCP}	110	V_{CC}	154	A12
23	V_{SS}	67	V_{SS}	111	V_{SS}	155	A11
24	V_{CC}	68	D27	112	V_{CC}	156	VCC
25	V_{CC}	69	D26	113	V_{CC}	157	V_{SS}
26	V_{SS}	70	D25	114	V_{SS}	158	V_{CCP}
27	CLK	71	D24	115	V_{CC}	159	A10
28	HLDA	72	V_{CCP}	116	D4	160	A9
29	W/R#	73	V_{SS}	117	D3	161	VCC
30	V_{SS}	74	D23	118	D2	162	V_{SS}
31	V_{CCP}	75	D22	119	D1	163	A8
32	BREQ	76	D21	120	D0	164	A7

Table 2. Pin Assignment for 176-Lead TQFP Package
Embedded ULP Intel486™ SX Processor (Continued)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
33	BE0#	77	V_{CCP}	121	V_{SS}	165	A6
34	BE1#	78	V_{SS}	122	A31	166	RESERVED
35	BE2#	79	D20	123	A30	167	A5
36	BE3#	80	D19	124	A29	168	A4
37	V_{CC}	81	D18	125	V_{CCP}	169	A3
38	M/IO#	82	V_{CC}	126	A28	170	V_{CCP}
39	D/C#	83	D17	127	A27	171	V_{SS}
40	PWT	84	V_{SS}	128	A26	172	V_{SS}
41	PCD	85	V_{SS}	129	A25	173	V_{CCP}
42	V_{CCP}	86	V_{CCP}	130	V_{CCP}	174	V_{SS}
43	V_{SS}	87	D16	131	V_{SS}	175	A2
44	V_{CC}	88	V_{SS}	132	V_{SS}	176	ADS#

Table 3. Pin Cross Reference for 176-Lead TQFP Package
Embedded ULP Intel486™ SX Processor

Address	Pin #	Data	Pin #	Control	Pin #	N/C	V_{CCP}	V_{CC}	V_{SS}
A2	175	D0	120	AHOLD	19	7	6	2	5
A3	169	D1	119	BE0#	33	13	31	12	17
A4	168	D2	118	BE1#	34	103	42	16	23
A5	167	D3	117	BE2#	35		56	21	26
A6	165	D4	116	BE3#	36		66	22	30
A7	164	D5	106	BLAST#	1		72	24	43
A8	163	D6	105	BOFF#	9		77	25	48
A9	160	D7	102	BRDY#	8		86	37	55
A10	159	D8	100	BREQ	32		91	44	67
A11	155	D9	99	BS16#	10		104	54	73
A12	154	D10	96	BS8#	11		107	59	78
A13	152	D11	95	CLK	27		125	65	84
A14	151	D12	94	D/C#	39		130	82	85
A15	148	D13	93	HLDA	28		137	97	88
A16	147	D14	90	HOLD	18		138	109	92
A17	146	D15	89	KEN#	15		144	110	98
A18	141	D16	87	LOCK#	4		150	112	101
A19	140	D17	83	M/IO#	38		158	113	108
A20	139	D18	81	PCD	41		170	115	111
A21	136	D19	80	PLOCK#	3		173	153	114
A22	135	D20	79	PWT	40			156	121
A23	134	D21	76	RESERVED	166			161	131
A24	133	D22	75	RDY#	14				132
A25	129	D23	74	TCK	20				145

Table 3. Pin Cross Reference for 176-Lead TQFP Package
Embedded ULP Intel486™ SX Processor (Continued)

Address	Pin #	Data	Pin #	Control	Pin #	N/C	V_{CCP}	V_{CC}	V_{SS}
A26	128	D24	71	W/R#	29				149
A27	127	D25	70	A20M#	46				157
A28	126	D26	69	EADS#	45				162
A29	124	D27	68	FLUSH#	49				171
A30	123	D28	64	INTR	50				172
A31	122	D29	63	NMI	51				174
		D30	62	RESET	47				
		D31	61	SMI#	57				
				SMIACT#	53				
				SRESET	52				
				STPCLK#	60				
				TDO	58				
				ADS#	176				
				TDI	143				
				TMS	142				
				W/R#	29				

3.2 Pin Quick Reference

The following is a brief pin description. For detailed signal descriptions refer to "Signal Description" in Section 9 of the *Intel486 Processor Family* datasheet.

Table 4. Embedded ULP Intel486™ SX Processor Pin Descriptions (Sheet 1 of 6)

Symbol	Type	Name and Function
CLK	I	**Clock** provides the fundamental timing and internal operating frequency for the embedded ULP Intel486 SX processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 **A3–A2**	I/O O	**Address Lines** A31–A2, together with the byte enable signals, BE3#–BE0#, define the physical area of memory or input/output space accessed. Address lines A31–A4 are used to drive addresses into the embedded ULP Intel486 SX processor to perform cache line invalidation. Input signals must meet setup and hold times t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.
BE3# **BE2#** **BE1#** **BE0#**	O O O O	**Byte Enable** signals indicate active bytes during read and write cycles. During the first cycle of a cache fill, the external system should assume that all byte enables are active. BE3#–BE0# are active LOW and are not driven during bus hold. BE3# applies to D31–D24 BE2# applies to D23–D16 BE1# applies to D15–D8 BE0# applies to D7–D0
DATA BUS		
D31–D0	I/O	**Data Lines.** D7–D0 define the least significant byte of the data bus; D31–D24 define the most significant byte of the data bus. These signals must meet setup and hold times t_{22} and t_{23} for proper operation on reads. These pins are driven during the second and subsequent clocks of write cycles.
BUS CYCLE Definition		
M/IO# **D/C#** **W/R#**	O O O	**Memory/Input-Output, Data/Control and Write/Read** lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted.

M/IO#	D/C#	W/R#	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	HALT/Special Cycle (see details below)
0	1	0	I/O Read
0	1	1	I/O Write
1	0	0	Code Read
1	0	1	Reserved
1	1	0	Memory Read
1	1	1	Memory Write

HALT/Special Cycle

Cycle Name	BE3#–BE0#	A4–A2
Shutdown	1110	000
HALT	1011	000
Stop Grant bus cycle	1011	100

4

Table 4. Embedded ULP Intel486™ SX Processor Pin Descriptions (Sheet 2 of 6)

Symbol	Type	Name and Function
LOCK#	O	**Bus Lock** indicates that the current bus cycle is locked. The embedded ULP Intel486 SX processor does not allow a bus hold when LOCK# is asserted (address holds are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the last clock of the last locked bus cycle. The last locked cycle ends when Ready is returned. LOCK# is active LOW and not driven during bus hold. Locked read cycles are not transformed into cache fill cycles when KEN# is returned active.
PLOCK#	O	**Pseudo-Lock** indicates that the current bus transaction requires more than one bus cycle to complete. For the embedded ULP Intel486 SX processor, examples of such operations are segment table descriptor reads (64 bits) and cache line fills (128 bits). The embedded ULP Intel486 SX processor drives PLOCK# active until the addresses for the last bus cycle of the transaction are driven, regardless of whether RDY# or BRDY# have been returned. PLOCK# is a function of the BS8#, BS16# and KEN# inputs. PLOCK# should be sampled only in the clock in which Ready is returned. PLOCK# is active LOW and is not driven during bus hold.
BUS CONTROL		
ADS#	O	**Address Status** output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus. ADS# is driven active in the same clock in which the addresses are driven. ADS# is active LOW and not driven during bus hold.
RDY#	I	**Non-burst Ready** input indicates that the current bus cycle is complete. RDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted data from the embedded ULP Intel486 SX processor in response to a write. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. RDY# is active during address hold. Data can be returned to the embedded ULP Intel486 SX processor while AHOLD is active. RDY# is active LOW and is not provided with an internal pull-up resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.

Table 4. Embedded ULP Intel486™ SX Processor Pin Descriptions (Sheet 3 of 6)

Symbol	Type	Name and Function
BURST CONTROL		
BRDY#	I	**Burst Ready** input performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data in response to a read or that the external system has accepted data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle. BRDY# is sampled in the second and subsequent clocks of a burst cycle. Data presented on the data bus is strobed into the embedded ULP Intel486 SX processor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. BRDY# is active LOW and is provided with a small pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.
BLAST#	O	**Burst Last** signal indicates that the next time BRDY# is returned, the burst bus cycle is complete. BLAST# is active for both burst and non-burst bus cycles. BLAST# is active LOW and is not driven during bus hold.
INTERRUPTS		
RESET	I	**Reset** input forces the embedded ULP Intel486 SX processor to begin execution at a known state. The processor cannot begin executing instructions until at least 1 ms after V_{CC}, V_{CCP}, and CLK have reached their proper DC and AC specifications. The RESET pin must remain active during this time to ensure proper processor operation. However, for warm resets, RESET should remain active for at least 15 CLK periods. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
INTR	I	**Maskable Interrupt** indicates that an external interrupt has been generated. When the internal interrupt flag is set in EFLAGS, active interrupt processing is initiated. The embedded ULP Intel486 SX processor generates two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the interrupt acknowledges have been performed to ensure processor recognition of the interrupt. INTR is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
NMI	I	**Non-Maskable Interrupt** request signal indicates that an external non-maskable interrupt has been generated. NMI is rising-edge sensitive and must be held LOW for at least four CLK periods before this rising edge. NMI is not provided with an internal pull-down resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SRESET	I	**Soft Reset** pin duplicates all functionality of the RESET pin except that the SMBASE register retains its previous value. For soft resets, SRESET must remain active for at least 15 CLK periods. SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

Table 4. Embedded ULP Intel486™ SX Processor Pin Descriptions (Sheet 4 of 6)

Symbol	Type	Name and Function
SMI#	I	**System Management Interrupt** input invokes System Management Mode (SMM). SMI# is a falling-edge triggered signal which forces the embedded ULP Intel486 SX processor into SMM at the completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# does not break LOCKed bus cycles and cannot interrupt a currently executing SMM. The embedded ULP Intel486 SX processor latches the falling edge of one pending SMI# signal while it is executing an existing SMI#. The nested SMI# is not recognized until after the execution of a Resume (RSM) instruction.
SMIACT#	O	**System Management Interrupt Active,** an active LOW output, indicates that the embedded ULP Intel486 SX processor is operating in SMM. It is asserted when the processor begins to execute the SMI# state save sequence and remains active LOW until the processor executes the last state restore cycle out of SMRAM.
STPCLK#	I	**Stop Clock Request** input signal indicates a request was made to turn off or change the CLK input frequency. When the embedded ULP Intel486 SX processor recognizes a STPCLK#, it stops execution on the next instruction boundary (unless superseded by a higher priority interrupt), empties all internal pipelines and write buffers, and generates a Stop Grant bus cycle. STPCLK# is active LOW.**STPCLK# is an asynchronous signal, but must remain active until the embedded ULP Intel486 SX processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle.**
BUS ARBITRATION		
BREQ	O	**Bus Request** signal indicates that the embedded ULP Intel486 SX processor has internally generated a bus request. BREQ is generated whether or not the processor is driving the bus. BREQ is active HIGH and is never floated.
HOLD	I	**Bus Hold Request** allows another bus master complete control of the embedded ULP Intel486 SX processor bus. In response to HOLD going active, the processor floats most of its output and input/output pins. HLDA is asserted after completing the current bus cycle, burst cycle or sequence of locked cycles. The embedded ULP Intel486 SX processor remains in this state until HOLD is de-asserted. HOLD is active HIGH and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper operation.
HLDA	O	**Hold Acknowledge** goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the embedded ULP Intel486 SX processor has given the bus to another local bus master. HLDA is driven active in the same clock that the processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.
BOFF#	I	**Backoff** input forces the embedded ULP Intel486 SX processor to float its bus in the next clock. The processor floats all pins normally floated during bus hold but HLDA is not asserted in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#; if both are returned in the same clock, BOFF# takes effect. The embedded ULP Intel486 SX processor remains in bus hold until BOFF# is negated. If a bus cycle is in progress when BOFF# is asserted the cycle is restarted. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper operation.

Table 4. Embedded ULP Intel486™ SX Processor Pin Descriptions (Sheet 5 of 6)

Symbol	Type	Name and Function
CACHE INVALIDATION		
AHOLD	I	**Address Hold** request allows another bus master access to the embedded ULP Intel486 SX processor's address bus for a cache invalidation cycle. The processor stops driving its address bus in the clock following AHOLD going active. Only the address bus is floated during address hold, the remainder of the bus remains active. AHOLD is active HIGH and is provided with a small internal pull-down resistor. For proper operation, AHOLD must meet setup and hold times t_{18} and t_{19}.
EADS#	I	**External Address**—This signal indicates that a *valid* external address has been driven onto the embedded ULP Intel486 SX processor address pins. This address is used to perform an internal cache invalidation cycle. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy setup and hold times t_{12} and t_{13} for proper operation.
CACHE CONTROL		
KEN#	I	**Cache Enable** pin is used to determine whether the current cycle is cacheable. When the embedded ULP Intel486 SX processor generates a cycle that can be cached and KEN# is active one clock before RDY# or BRDY# during the first transfer of the cycle, the cycle becomes a cache line fill cycle. Returning KEN# active one clock before RDY# during the last read in the cache line fill causes the line to be placed in the on-chip cache. KEN# is active LOW and is provided with a small internal pull-up resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	**Cache Flush** input forces the embedded ULP Intel486 SX processor to flush its entire internal cache. FLUSH# is active LOW and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition in any specific clock.
PAGE CACHEABILITY		
PWT **PCD**	O O	**Page Write-Through and Page Cache Disable** pins reflect the state of the page attribute bits, PWT and PCD, in the page table entry, page directory entry or control register 3 (CR3) when paging is enabled. When paging is disabled, the embedded ULP Intel486 SX processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT pins. PWT and PCD have the same timing as the cycle definition pins (M/IO#, D/C#, and W/R#). PWT and PCD are active HIGH and are not driven during bus hold. PCD is masked by the cache disable bit (CD) in Control Register 0.
BUS SIZE CONTROL		
BS16# **BS8#**	I I	**Bus Size 16 and Bus Size 8** pins (bus sizing pins) cause the embedded ULP Intel486 SX processor to run multiple bus cycles to complete a request from devices that cannot provide or accept 32 bits of data in a single cycle. The bus sizing pins are sampled every clock. The processor uses the state of these pins in the clock before Ready to determine bus size. These signals are active LOW and are provided with internal pull-up resistors. These inputs must satisfy setup and hold times t_{14} and t_{15} for proper operation.

Table 4. Embedded ULP Intel486™ SX Processor Pin Descriptions (Sheet 6 of 6)

Symbol	Type	Name and Function
ADDRESS MASK		
A20M#	I	**Address Bit 20 Mask** pin, when asserted, causes the embedded ULP Intel486 SX processor to mask physical address bit 20 (A20) before performing a lookup to the internal cache or driving a memory cycle on the bus. A20M# emulates the address wraparound at 1 Mbyte, which occurs on the 8086 processor. A20M# is active LOW and should be asserted only when the embedded ULP Intel486 SX processor is in real mode. This pin is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For proper operation, A20M# should be sampled HIGH at the falling edge of RESET.
TEST ACCESS PORT		
TCK	I	**Test Clock,** an input to the embedded ULP Intel486 SX processor, provides the clocking function required by the JTAG Boundary scan feature. TCK is used to clock state information (via TMS) and data (via TDI) into the component on the rising edge of TCK. Data is clocked out of the component (via TDO) on the falling edge of TCK. TCK is provided with an internal pull-up resistor.
TDI	I	**Test Data Input** is the serial input used to shift JTAG instructions and data into the processor. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and SHIFT-DR TAP controller states. During all other Test Access Port (TAP) controller states, TDI is a "don't care." TDI is provided with an internal pull-up resistor.
TDO	O	**Test Data Output** is the serial output used to shift JTAG instructions and data out of the component. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state.
TMS	I	**Test Mode Select** is decoded by the JTAG TAP to select test logic operation. TMS is sampled on the rising edge of TCK. To guarantee deterministic behavior of the TAP controller, TMS is provided with an internal pull-up resistor.
RESERVED PINS		
RESERVED	I	**Reserved** is reserved for future use. This pin MUST be connected to an external pull-up resistor circuit. The recommended resistor value is 10 kΩ.

Table 5. Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States(1)
BREQ	HIGH			Previous State
HLDA	HIGH			As per HOLD
BE3#-BE0#	LOW		•	Previous State
PWT, PCD	HIGH		•	Previous State
W/R#, M/IO#, D/C#	HIGH/LOW		•	Previous State
LOCK#	LOW		•	HIGH (inactive)
PLOCK#	LOW		•	HIGH (inactive)
ADS#	LOW		•	HIGH (inactive)
BLAST#	LOW		•	Previous State
A3-A2	HIGH	•	•	Previous State
SMIACT#	LOW			Previous State

NOTE:
1. The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

Table 6. Input/Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States(1, 2)
D31-D0	HIGH		•	Level-Keeper
A31-A4	HIGH	•	•	Previous State

NOTES:
1. The term "Level-Keeper" means that the processor maintains the most recent logic level applied to the signal pin. This conserves power by preventing the signal pin from floating. If a system component, other than the processor, temporarily drives these signal pins and then floats them, the processor forces and maintains the most recent logic levels that were applied by the system component.
2. The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

Table 7. Test Pins

Name	Input or Output	Sampled/Driven On
TCK	Input	N/A
TDI	Input	Rising Edge of TCK
TDO	Output	Failing Edge of TCK
TMS	Input	Rising Edge of TCK

Table 8. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Pull-Up/ Pull-Down
CLK			
RESET	HIGH	Asynchronous	
SRESET	HIGH	Asynchronous	Pull-Down
HOLD	HIGH	Synchronous	
AHOLD	HIGH	Synchronous	Pull-Down
EADS#	LOW	Synchronous	Pull-Up
BOFF#	LOW	Synchronous	Pull-Up
FLUSH#	LOW	Asynchronous	Pull-Up
A20M#	LOW	Asynchronous	Pull-Up
BS16#, BS8#	LOW	Synchronous	Pull-Up
KEN#	LOW	Synchronous	Pull-Up
RDY#	LOW	Synchronous	
BRDY#	LOW	Synchronous	Pull-Up
INTR	HIGH	Asynchronous	
NMI	HIGH	Asynchronous	
RESERVED			
SMI#	LOW	Asynchronous	Pull-Up
STPCLK#	LOW	Asynchronous	Pull-Up
TCK	HIGH		Pull-Up
TDI	HIGH		Pull-Up
TMS	HIGH		Pull-Up

4.0 ARCHITECTURAL AND FUNCTIONAL OVERVIEW

The embedded ULP Intel486 SX processor architecture is essentially the same as the 3.3V Intel486 SX processor with a 1X clock (CLK) input. Refer to the *Intel486™ Processor Family* datasheet (242202) for a description of the Intel486 SX processor. With some minor exceptions, the following datasheet sections apply to the embedded ULP Intel486 SX processor:

- Architectural Overview
- Real Mode Architecture
- Protected Mode Architecture
- On-Chip Cache
- System Management Mode (SMM) Architectures
- Hardware Interface
- Bus Operation
- Testability
- Debugging Support
- Instruction Set Summary
- Differences Between Intel486 Processors and Intel386 Processors

Exceptions to these sections of the datasheet are:

- The information pertaining to parity signals for the external data bus does not apply. The embedded ULP Intel486 SX processor does not have DP0#, DP1#, DP2#, DP3# and PCHK# signal pins.

- References to the Upgrade Power Down Mode do not apply. The embedded ULP Intel486 SX processor does not have the UP# signal pin and does not support the Intel OverDrive® processor.
- References to "VCC" are called "V_{CCP}" by the embedded ULP Intel486 SX processor when the supply voltage pertains to the processor's external interface drivers and receivers. The term "V_{CC}" pertains only to the processor core supply voltage of the embedded ULP Intel486 SX processor. Information about the split-supply voltage is provided in this datasheet.
- The Phase-Locked Loop (PLL) circuit of the 1X clock (CLK) input has been replaced by a proprietary Differential Delay Line (DDL) circuit that has a faster recovery time. Datasheet references to the PLL and its 1 ms recovery time are replaced with the DDL circuit and its eight-CLK recovery time. Information about the DDL circuit and recovery time is provided in this datasheet.
- The embedded ULP Intel486 SX processor has level-keeper circuits for its external 32-bit data bus signals (D31–D0). The Intel486 SX processor floats its data bus instead. More information about the level-keeper circuitry is provided in this datasheet.
- The datasheet describes the processor supply-current consumption for the Auto HALT Power Down, Stop Grant, and Stop Clock states. This supply-current consumption for the embedded ULP Intel486 SX processor is much less than that of the Intel486 SX processor. Information about power consumption and these states is provided in this datasheet.
- The CPU ID, Boundary-Scan (JTAG) ID, and boundary-scan register bits for the embedded ULP Intel486 SX processor are in this datasheet.
- The embedded ULP Intel486 SX processor has one pin reserved for possible future use. This pin is an input signal, pin 166. It is called RESERVED and must be connected to a 10 KΩ pull-up resistor.

4.1 Separate Supply Voltages

The embedded ULP Intel486 SX processor has separate voltage-supply planes for its internal core-processor circuits and its external driver/receiver circuits. The supply voltage for the internal core processor is named V_{CC} and the supply voltage for the external circuits is named V_{CCP}.

For a single-supply voltage design, the embedded ULP Intel486 SX processor is functional at 3.3V ± 0.3V. In this type of system design, the processor's V_{CC} and V_{CCP} pins must be tied to the same power plane.

Even though V_{CCP} must be 3.3V ± 0.3V, the processor's external-output circuits can drive TTL-compatible components. However, the processor's external-input circuits do not allow connection to TTL-compatible components. **Section 5.2, DC Specifications (pg. 22)** contains the DC specifications for the processor's input and output signals.

For lower-power operation, a separate, lower voltage for V_{CC} can be chosen, but V_{CCP} must be 3.3V ± 0.3V. Any voltage value between 2.4V and 3.3V can be chosen for V_{CC} for guaranteed processor operation up to 25 MHz. The embedded ULP Intel486 SX processor can also operate at 33 MHz, provided the V_{CC} value chosen is between 2.7V and 3.3V. **Section 5.2, DC Specifications (pg. 22)** defines supply voltage specifications.

In systems with separate V_{CC} and V_{CCP} power planes, the processor-core voltage supply must always be less than or equal to the processor's external-interface voltage supply; e.g., the system design must guarantee:

$$V_{CC} \leq V_{CCP}$$

Violating this relationship causes excessive power consumption. Limited testing has shown no component damage when this relationship is violated. However, prolonged violation is not recommended and component integrity is not guaranteed.

The $V_{CC} \leq V_{CCP}$ relationship must also be guaranteed by the system design during power-up and power-down sequences. Refer to Figure 3.

Even though V_{CC} must be less than or equal to V_{CCP}, it is recommended that the system's power-on sequence allows V_{CC} to quickly achieve its operational level once V_{CCP} achieves its operational level. Similarly, the power-down sequence should allow V_{CCP} to power down quickly after V_{CC} is below the operational voltage level. These recommendations are given to keep power consumption at a minimum. Deviating from the recommendations does not create a component reliability problem, but power consumption of the processor's external interface circuits increases beyond normal operating values.

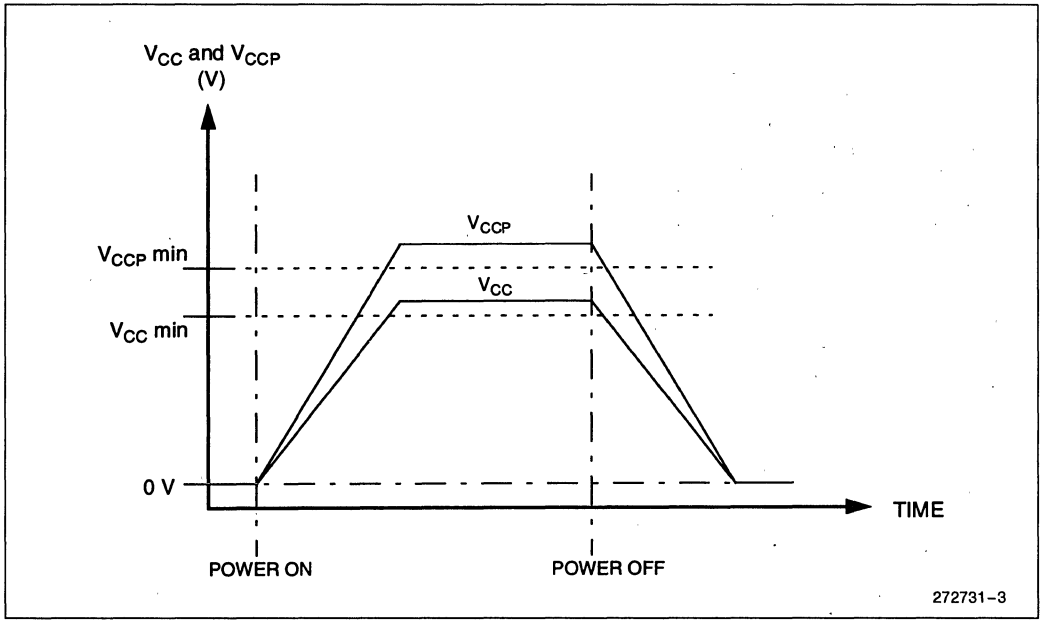


Figure 3. Example of Supply Voltage Power Sequence

4.2 Fast Clock Restart

The embedded ULP Intel486 SX processor has an integrated proprietary differential delay line (DDL) circuit for internal clock generation. The DDL is driven by the CLK input signal provided by the external system. During normal operation, the external system must guarantee that the CLK signal maintains its frequency so that the clock period deviates no more than 250 ps/CLK. This state, called the Normal State, is shown in Figure 4.

To increase or decrease the CLK frequency more quickly than this, the system must interrupt the processor with the STPCLK# signal. Once the processor indicates that it is in the Stop Grant State, the system can adjust the CLK signal to the new frequency, wait a minimum of eight CLK periods, then force the processor to return to the normal operational state by deactivating the STPCLK# interrupt.

This wait of eight CLK periods is much faster than the 1 ms wait required by earlier Intel486 SX processor products.

While in the Stop Grant State, the external system may deactivate the CLK signal to the processor. This forces the processor to the Stop Clock State—the state in which the processor consumes the least power. Once the system reactivates the CLK signal, the processor transitions to the Stop Grant State within eight CLK periods.

Normal operation can be resumed by deactivating the STPCLK# interrupt signal. Here again, the embedded ULP Intel486 SX processor recovers from the Stop Clock State much faster than the 1 ms PLL recovery of earlier Intel486 SX processors.

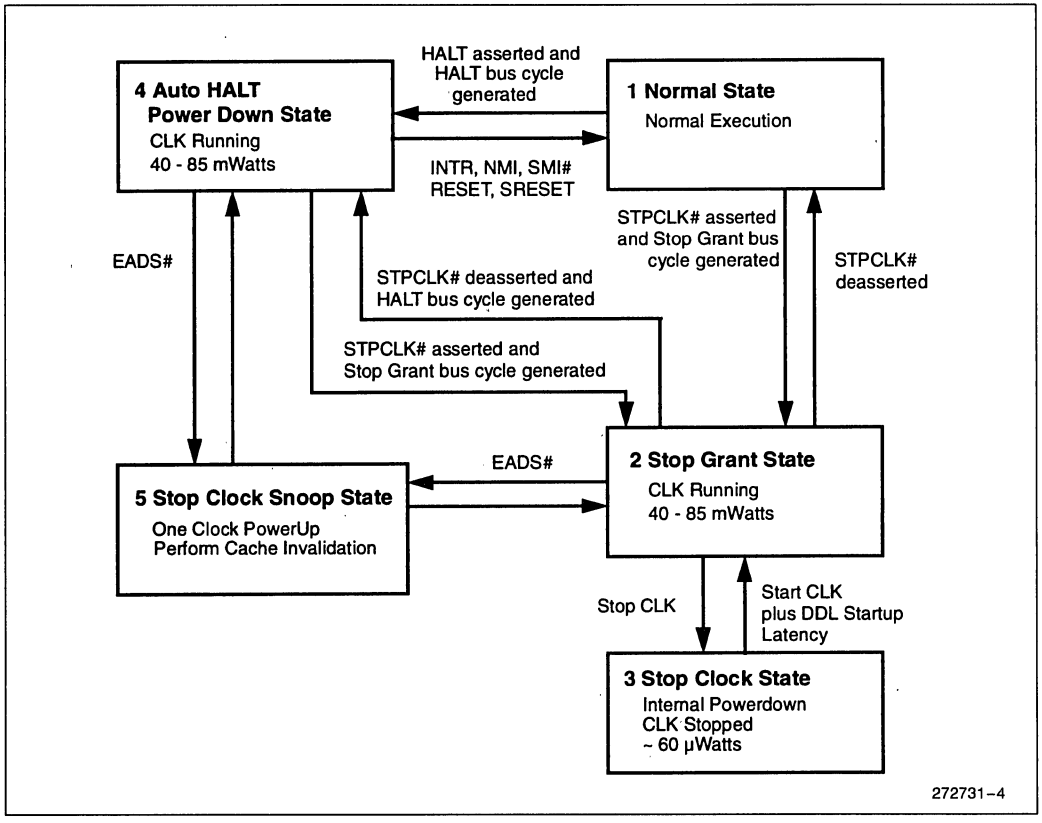


Figure 4. Stop Clock State Diagram with Typical Power Consumption Values

4.3 Level-Keeper Circuits

To obtain the lowest possible power consumption during the Stop Grant and Stop Clock states, system designers must ensure that:

- input signals with pull-up resistors are not driven LOW
- input signals with pull-down resistors are not driven HIGH

See Table 8, Input Pins (pg. 14) for the list of signals with internal pull-up and pull-down resistors.

All other input pins except A31-A4 and D31-D0 must be driven to the power supply rails to ensure lowest possible current consumption.

During the Stop Grant and Stop Clock states, most processor output signals maintain their previous condition, which is the level they held when entering the Stop Grant state. In response to HOLD driven active during the Stop Grant state when the CLK input is running, the embedded ULP Intel486 SX processor generates HLDA and floats all output and input/output signals which are floated during the HOLD/HLDA state. When HOLD is deasserted, processor signals which maintain their previous state return to the state they were in prior to the HOLD/HLDA sequence.

The data bus (D31-D0) also maintains its previous state during the Stop Grant and Stop Clock states, but does so differently, as described in the following paragraphs.

The embedded ULP Intel486 SX processor's data bus pins (D31–D0) have level keepers which maintain their previous states while in the Stop Grant and Stop Clock states. In response to HOLD driven active during the Stop Grant state when the CLK input is running, the embedded ULP Intel486 SX processor generates HLDA and floats D31–D0 throughout the HOLD/HLDA cycles. When HOLD is deasserted, the processor's D31–D0 signals return to the states they were in prior to the HOLD/HLDA sequence.

At all other times during the Stop Grant and Stop Clock states, the processor maintains the logic levels of D31–D0. When the external system circuitry drives D31–D0 to different logic levels, the processor flips its D31–D0 logic levels to match the ones driven by the external system. The processor maintains (keeps) these new levels even after the external circuitry stops driving D31–D0.

For some system designs, external resistors may not be required on D31–D0 (they are required on previous Intel486 SX processor designs). System designs that never request Bus Hold during the Stop Grant and Stop Clock states might not require external resistors. If the system design uses Bus Hold during these states, the processor disables the level-keepers and floats the data bus. This type of design would require some kind of data bus termination to minimize power consumption. It is strongly recommended that the D31–D0 pins do not have network resistors connected. External resistors used in the system design must be of a sufficient resistance value to "flip" the level-keeper circuitry and eliminate potential DC paths.

The level-keeper circuit is designed to allow an external 27-KΩ pull-up resistor to switch the D31–D0 circuits to a logic-HIGH level even though the level-keeper attempts to keep a logic-LOW level. In general, pull-up resistors smaller than 27 KΩ can be used as well as those greater than or equal to 1 MΩ. Pull-down resistors, when connected to D31–D0, should be least 800 KΩ.

4.4 Low-Power Features

As with other Intel486 processors, the embedded ULP Intel486 SX processor minimizes power consumption by providing the Auto HALT Power Down, Stop Grant, and Stop Clock states (see Figure 4). The embedded ULP Intel486 SX processor has an Auto Clock Freeze feature that further conserves power by judiciously deactivating its internal clocks while in the Normal Execution Mode. The power-conserving mechanism is designed such that it does not degrade processor performance or require changes to AC timing specifications.

4.4.1 Auto Clock Freeze

To reduce power consumption, during the following bus cycles—under certain conditions—the processor slows-up or freezes some internal clocks:

- Data-Read Wait Cycles (Memory, I/O and Interrupt Acknowledge)
- Data-Write Wait Cycles (Memory, I/O)
- HOLD/HLDA Cycles
- AHOLD Cycles
- BOFF Cycles

Power is conserved during the wait periods in these cycles until the appropriate external-system signals are sent to the processor. These signals include:

- READY
- NMI, SMI#, INTR, and RESET
- BOFF#
- FLUSH#
- EADS#
- BS8#, BS16# and KEN# transitions

The embedded ULP Intel486 SX processor also reduces power consumption by temporarily freezing the clocks of its internal logic blocks. When a logic block is idle or in a wait state, its clock is frozen.

4.5 CPUID Instruction

The embedded ULP Intel486 SX processor supports the CPUID instruction (see Table 9). Because not all Intel processors support the CPUID instruction, a simple test can determine if the instruction is supported. The test involves the processor's ID Flag, which is bit 21 of the EFLAGS register. If software can change the value of this flag, the CPUID instruction is available. The actual state of the ID Flag bit is irrelevant and provides no significance to the hardware. This bit is cleared (reset to zero) upon device reset (RESET or SRESET) for compatibility with Intel486 processor designs that do not support the CPUID instruction.

CPUID-instruction details are provided here for the embedded ULP Intel486 SX processor. Refer to Intel Application Note AP-485 *Intel Processor Identification with the CPUID Instruction* (Order No. 241618) for a description that covers all aspects of the CPUID instruction and how it pertains to other Intel processors.

4.5.1 Operation of the CPUID Instruction

The CPUID instruction requires the software developer to pass an input parameter to the processor in the EAX register. The processor response is returned in registers EAX, EBX, EDX, and ECX.

Table 9. CPUID Instruction Description

OP CODE	Instruction	Processor Core Clocks	Parameter passed in EAX (Input Value)	Description
0F A2	CPUID	9	0	Vendor (Intel) ID String
		14	1	Processor Identification
		9	> 1	Undefined (Do Not Use)

Vendor ID String—When the parameter passed in EAX is 0 (zero), the register values returned upon instruction execution are shown in the following table.

		31 – – – – – – 24	23 – – – – – – 16	15 – – – – – – 8	7 – – – – – – 0
High Value (= 1)	**EAX**	' 0000	0000	0000	0001

Vendor ID String	**EBX**	u (75)	n (6E)	e (65)	G (47)
(ASCII	**EDX**	I (49)	e (65)	n (6E)	i (69)
Characters)	**ECX**	I (6C)	e (65)	t (74)	n (6E)

The values in EBX, EDX and ECX indicate an Intel processor. When taken in the proper order, they decode to the string "GenuineIntel."

Processor Identification—When the parameter passed to EAX is 1 (one), the register values returned upon instruction execution are:

		31 – – – – 14	13, 12	11 – – – 8	7 – – – 4	3 – – – 0
Processor Signature	**EAX**	(Do Not Use) Intel Reserved	0 0 Processor Type	0 1 0 0 Family	0 0 1 0 Model	XXXX Stepping

(Intel releases information about stepping numbers as needed)

		31 – 0
Intel Reserved	**EBX**	Intel Reserved
(Do Not Use)	**ECX**	Intel Reserved

		31 – – – – – – – – – – – – – – – – 2	1	0
Feature Flags	**EDX**	0 0	1 VME	0 FPU

4.6 Identification After Reset

Processor Identification—Upon reset, the EDX register contains the processor signature:

		31 – – – – 14	13, 12	11 – – – 8	7 – – – 4	3 – – – 0
Processor Signature	**EDX**	(Do Not Use) Intel Reserved	0 0 Processor Type	0 1 0 0 Family	0 0 1 0 Model	XXXX Stepping

(Intel releases information about stepping numbers as needed)

4.7 Boundary Scan (JTAG)

4.7.1 Device Identification

Table 10 shows the 32-bit code for the embedded ULP Intel486 SX processor which is loaded into the Device Identification Register.

Table 10. Boundary Scan Component Identification Code

Version	Part Number				Mfg ID 009H = Intel	1
	V_{CC} 0=5V 1=3.3V	Intel Architecture Type	Family 0100 = Intel486 CPU Family	Model 00010 = embedded ULP Intel486 SX processor		
31 - - - 28	27	26 - - - - - 21	20 - - - - 17	16 - - - - - 12	11 - - - - - 1	0
XXXX	1	000001	0100	00010	00000001001	1

(Intel releases information about version numbers as needed)

Boundary Scan Component Identification Code = x828 2013 (Hex)

4.7.2 Boundary Scan Register Bits and Bit Order

The boundary scan register contains a cell for each pin as well as cells for control of bidirectional and three-state pins. There are "Reserved" bits which correspond to no-connect (N/C) signals of the embedded ULP Intel486 SX processor. Control registers WRCTL, ABUSCTL, BUSCTL, and MISCCTL are used to select the direction of bidirectional or three-state output signal pins. A "1" in these cells designates that the associated bus or bits are floated if the pins are three-state, or selected as input if they are bidirectional.

- WRCTL controls D31–D0
- ABUSCTL controls A31–A2
- BUSCTL controls ADS#, BLAST#, PLOCK#, LOCK#, W/R#, BE0#, BE1#, BE2#, BE3#, M/IO#, D/C#, PWT, and PCD
- MISCCTL controls HLDA, and BREQ

The following is the bit order of the embedded ULP Intel486 SX processor boundary scan register:

TDO ← A2, A3, A4, A5, RESERVED, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, Reserved, D0, D1, D2, D3, D4, D5, D6, D7, Reserved, D8, D9, D10, D11, D12, D13, D14, D15, Reserved, D16, D17, D18, D19, D20, D21, D22, D23, Reserved, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, Reserved, Reserved, SMI#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, BS8#, BS16#, BOFF#, BRDY#, Reserved, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRCTL **← TDI**

5.0 ELECTRICAL SPECIFICATIONS

5.1 Maximum Ratings

Table 11 is a stress rating only. Extended exposure to the Maximum Ratings may affect device reliability.

Furthermore, although the embedded ULP Intel486 SX processor contains protective circuitry to resist damage from electrostatic discharge, always take precautions to avoid high static voltages or electric fields.

Functional operating conditions are given in **Section 5.2, DC Specifications** and **Section 5.3, AC Specifications.**

Table 11. Absolute Maximum Ratings

Case Temperature under Bias	−65°C to +110°C
Storage Temperature	−65°C to +150°C
DC Voltage on Any Pin with Respect to Ground	−0.5V to V_{CCP} + 0.5V
Supply Voltage V_{CC} with Respect to V_{SS}	−0.5V to +4.6V
Supply Voltage V_{CCP} with Respect to V_{SS}	−0.5V to +4.6V

5.2 DC Specifications

The following tables show the operating supply voltages, DC I/O specifications, and component power consumption for the embedded ULP Intel486 SX processor.

Table 12. Operating Supply Voltages

Product	V_{CCP} Range(1)	Max. CLK Frequency	V_{CC} Range(2)	V_{CC} Fluctuation
FA80486SXSF-33	3.3V ± 0.3V	25	2.4 Vmin 3.3 Vmax	±0.2V at 2.4V ≤ V_{CC} ≤ 2.7V ±0.3V/−0.2V at 2.7V ≤ V_{CC} ≤ 3.0V
		33	2.7 Vmin 3.3 Vmax	±0.3V at 3.0V ≤ V_{CC} ≤ 3.3V

NOTES:
1. In all cases, V_{CCP} must be ≥ V_{CC}.
2. V_{CC} may be set to any voltage within the V_{CC} Range. The setting determines the allowed V_{CC} Fluctuation.

Table 13. DC Specifications

T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min.	Max.	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CCP} +0.3	V	(Note 1)
V_{IHC}	Input HIGH Voltage of CLK	V_{CCP} −0.6	V_{CCP} +0.3	V	
V_{OL}	Output LOW Voltage I_{OL} = 2.0 mA I_{OL} = 100 μA		 0.4 0.2	 V V	
V_{OH}	Output HIGH Voltage I_{OH} = −2.0 mA I_{OH} = −100 μA	 2.4 V_{CCP} −0.2		 V V	
I_{LI}	Input Leakage Current		±15	μA	(Note 2)
I_{IH}	Input Leakage Current		200	μA	(Note 3)
I_{IL}	Input Leakage Current		−400	μA	(Note 4)
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		10	pF	(Note 5)
C_{OUT}	I/O or Output Capacitance		10	pF	(Note 5)
C_{CLK}	CLK Capacitance		6	pF	(Note 5)

NOTES:
1. All inputs except CLK.
2. This parameter is for inputs without pull-up or pull-down resistors and $0V \leq V_{IN} \leq V_{CCP}$.
3. This parameter is for inputs with pull-down resistors and V_{IH} = 2.4V, and for level-keeper pins at V = 0.4V.
4. This parameter is for inputs with pull-up resistors and V_{IL} = 0.4V, and for level-keeper pins at V = 2.4V.
5. F_C=1 MHz. Not 100% tested.

4

Table 14. Active I_{CC} Values

T_{CASE} = 0°C to +85°C

Symbol	Parameter	Frequency	Supply Voltage	Typical I_{CC}	Max. I_{CC}	Notes
I_{CC1}	I_{CC} Active (V_{CC} pins)	25 MHz	V_{CC} = 2.4 + 0.2V	120 mA	195 mA	
			V_{CC} = 3.3 ± 0.3V	165 mA	260 mA	
		33 MHz	V_{CC} = 2.7 ± 0.2V	180 mA	280 mA	
			V_{CC} = 3.3 ± 0.3V	220 mA	345 mA	
I_{CC2}	I_{CC} Active (V_{CCP} pins)	25 MHz	V_{CCP} = 3.3 ± 0.3V	7 mA	25 mA	1
		33 MHz	V_{CCP} = 3.3 ± 0.3V	9 mA	32 mA	1

NOTE:
1. These parameters are for C_L = 50 pF

Table 15. Clock Stop, Stop Grant, and Auto HALT Power Down I_{CC} Values

T_{CASE} = 0°C to +85°C

Symbol	Parameter	Frequency	Supply Voltage	Typical I_{CC}	Max. I_{CC}	Notes
I_{CCS0}	I_{CC} Stop Clock (V_{CC} pins)	0 MHz	V_{CC} = 2.4 ± 0.2 V	4 μA	120 μA	(Note 1)
			V_{CC} = 2.7 ± 0.2V	4 μA	130 μA	
			V_{CC} = 3.3 ± 0.3V	5 μA	150 μA	
I_{CCS2}	I_{CC} Stop Clock (V_{CCP} pins)	0 MHz	V_{CCP} = 3.3 ± 0.3V	3 μA	80 μA	
I_{CCS1}	I_{CC} Stop Grant, Auto HALT Power Down (V_{CC} pins)	25 MHz	V_{CC} = 2.4 ± 0.2V	14 mA	25 mA	
			V_{CC} = 3.3 ± 0.3V	20 mA	30 mA	
		33 MHz	V_{CC} = 2.7 ± 0.2V	20 mA	30 mA	
			V_{CC} = 3.3 ± 0.3V	25 mA	35 mA	
I_{CCS3}	I_{CC} Stop Grant, Auto HALT Power Down (V_{CCP} pins)	25 MHz	V_{CCP} = 3.3 ± 0.3V	425 μA	1.5 mA	
		33 MHz	V_{CCP} = 3.3 ± 0.3V	610 μA	2.0 mA	

NOTE:
1. The I_{CC} Stop Clock specification refers to the I_{CC} value once the processor enters the Stop Clock state. For all input signals, the V_{IH} and V_{IL} levels must be equal to V_{CCP} and 0V, respectively, to meet the I_{CC} Stop Clock specifications.

5.3 AC Specifications

The AC specifications for the embedded ULP Intel486 SX processor are given in this section.

Table 16. AC Characteristics (Sheet 1 of 2)

valid over the operating supply voltages listed in Table 12, Operating Supply Voltages (pg. 22).
T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	2.4V ≤ V_{CC} < 2.7V		2.7V ≤ V_{CC} ≤ 3.3V		Unit	Notes
		Min	Max	Min	Max		
	Frequency	0	25	0	33	MHz	(Note 1)
t_1	CLK Period	40		30		ns	(Note 1)
t_{1a}	CLK Period Stability		250		250	ps/CLK	(Note 2)
t_2	CLK High Time	14		11		ns	at 2V
t_3	CLK Low Time	14		11		ns	at 0.8V
t_4	CLK Fall Time		4		3	ns	2V to 0.8V (Note 3)
t_5	CLK Rise Time		4		3	ns	0.8V to 2V (Note 3)
t_6	A2–A31, PWT, PCD, BE0#–BE3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT# Valid Delay	3	19	3	16	ns	
t_7	A2–A31, PWT, PCD, BE0#–BE3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, Float Delay		28		20	ns	(Note 3)
t_{8a}	BLAST#, PLOCK# Valid Delay	3	24	3	20	ns	
t_9	BLAST#, PLOCK# Float Delay		28		20	ns	(Note 3)
t_{10}	D0–D31 Write Delay	3	20	3	19	ns	
t_{11}	D0–D31 Float Delay		28		20	ns	(Note 3)
t_{12}	EADS# Setup Time	8		6		ns	
t_{13}	EADS# Hold Time	3		3		ns	
t_{14}	BS16#, BS8#, KEN# Setup Time	8		6		ns	
t_{15}	BS16#, BS8#, KEN# Hold Time	3		3		ns	
t_{16}	RDY#, BRDY# Setup Time	8		6		ns	
t_{17}	RDY#, BRDY# Hold Time	3		3		ns	
t_{18}	HOLD, AHOLD Setup Time	10		6		ns	
t_{18a}	BOFF# Setup Time	10		9		ns	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		3		ns	

4

Table 16. AC Characteristics (Sheet 2 of 2)

valid over the operating supply voltages listed in Table 12, Operating Supply Voltages (pg. 22).
T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	2.4V ≤ V_{CC} < 2.7V		2.7V ≤ V_{CC} ≤ 3.3V		Unit	Notes
		Min	Max	Min	Max		
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET Setup Time	10		6		ns	
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET Hold Time	3		3		ns	
t_{22}	D0–D31, A4–A31 Read Setup Time	6		6		ns	
t_{23}	D0–D31, A4–A31 Read Hold Time	3		3		ns	

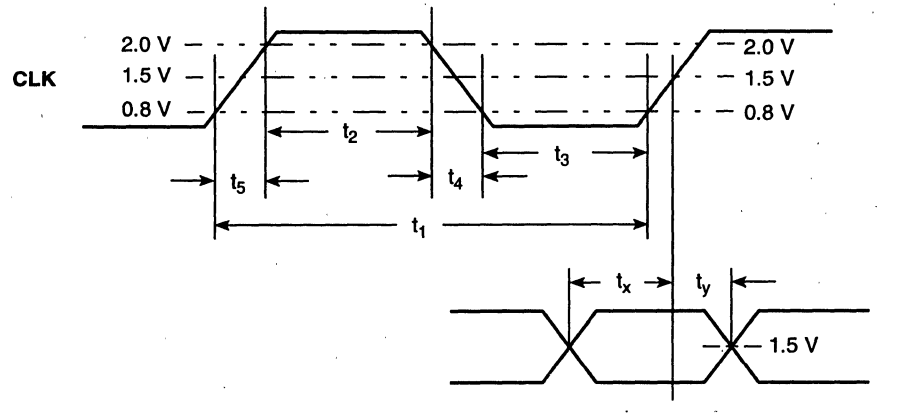
NOTES:
1. 0 Hz operation is tested and guaranteed by the STPCLK# and Stop Grant bus cycle protocol. 0 Hz < CLK < 8 MHz operation is confirmed by design characterization, but not 100% tested in production.
2. Specification t1a is available only when CLK frequency is changed without STPCLK#/STOP GRANT bus cycle protocol.
3. Not 100% tested, guaranteed by design characterization.
4. CLK reference voltage for timing measurement is 1.5V except t_2 through t_5. Other signals are measured at 1.5V.

Table 17. AC Specifications for the Test Access Port

Symbol	Parameter	2.2V ≤ V_{CC} < 3.0V		V_{CC} = 3.3V ± 0.3V		Unit	Figure	Notes
		Min	Max	Min	Max			
t_{24}	TCK Frequency		5		8	MHz	10	
t_{25}	TCK Period	200		125		ns	10	(Note 1)
t_{26}	TCK High Time	65		40		ns	10	@ 2.0V
t_{27}	TCK Low Time	65		40		ns	10	@ 0.8V
t_{28}	TCK Rise Time		15		8	ns	10	(Note 2)
t_{29}	TCK Fall Time		15		8	ns	10	(Note 2)
t_{30}	TDI, TMS Setup Time	16		8		ns	11	(Note 3)
t_{31}	TDI, TMS Hold Time	20		10		ns	11	(Note 3)
t_{32}	TDO Valid Delay	3	46	3	30	ns	11	(Note 3)
t_{33}	TDO Float Delay		52		36	ns	11	(Notes 3, 4)
t_{34}	All Outputs (except TDO) Valid Delay	3	80	3	30	ns	11	(Note 3)
t_{35}	All Outputs (except TDO) Float Delay		88		36	ns	11	(Notes 3, 4)
t_{36}	All Inputs (except TDI, TMS, TCK) Setup Time	16		8		ns	11	(Note 3)
t_{37}	All Inputs (except TDI, TMS, TCK) Hold Time	35		15		ns	11	(Note 3)

NOTES:
1. TCK period CLK period.
2. Rise/Fall Times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10 ns increase in TCK period.
3. Parameter measured from TCK.
4. Not 100% tested, guaranteed by design characterization.

4



t_x = input setup times
t_y = input hold times, output float, valid and hold times

Figure 5. CLK Waveform

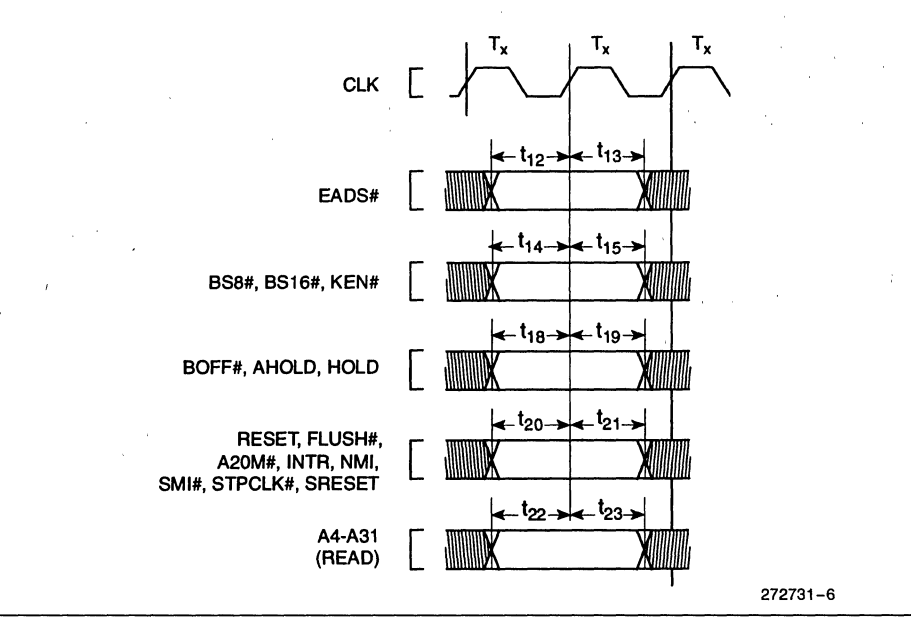


Figure 6. Input Setup and Hold Timing

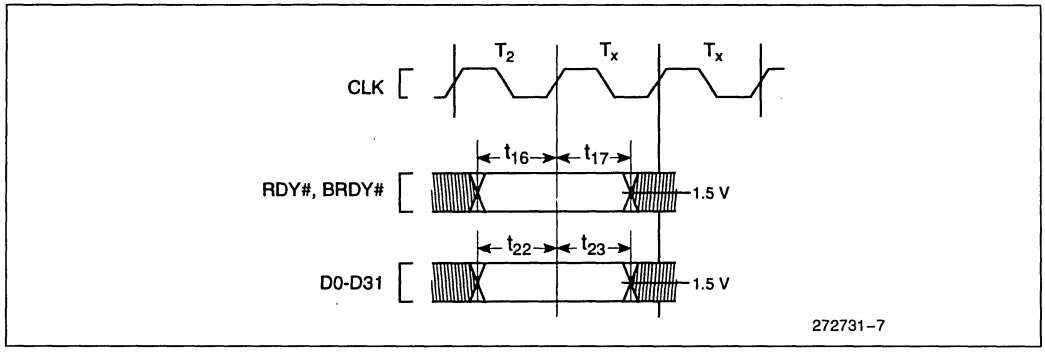


Figure 7. Input Setup and Hold Timing

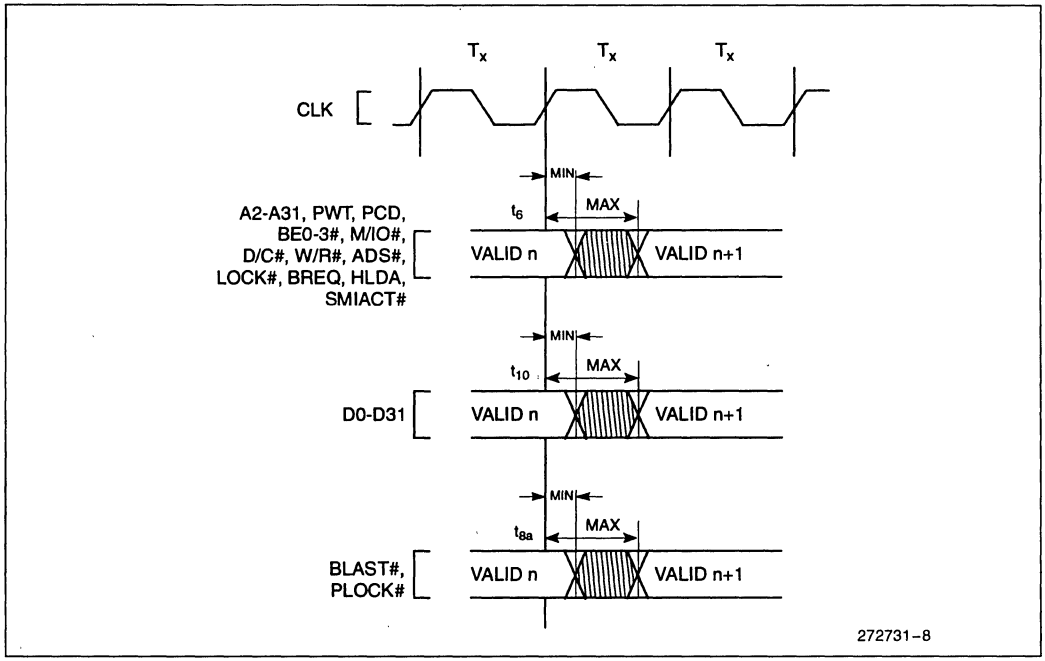


Figure 8. Output Valid Delay Timing

4

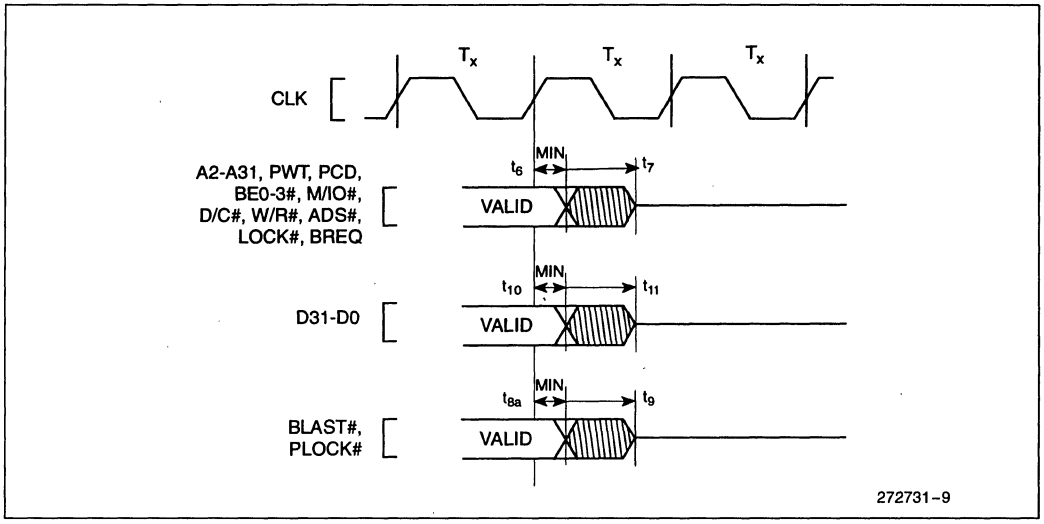


Figure 9. Maximum Float Delay Timing

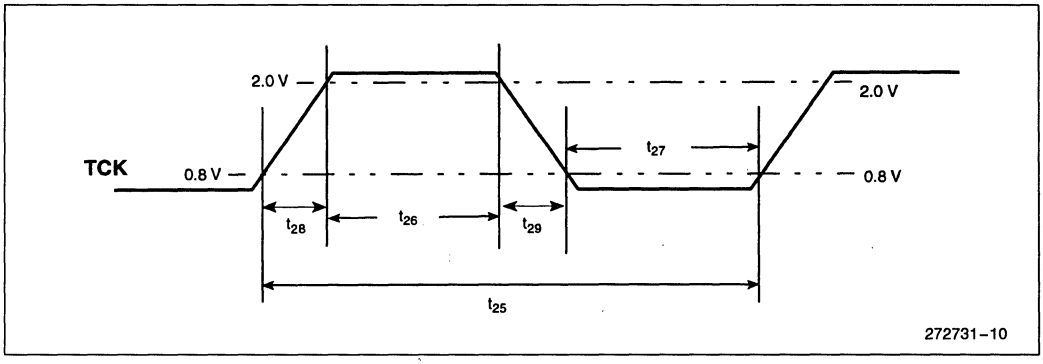


Figure 10. TCK Waveform

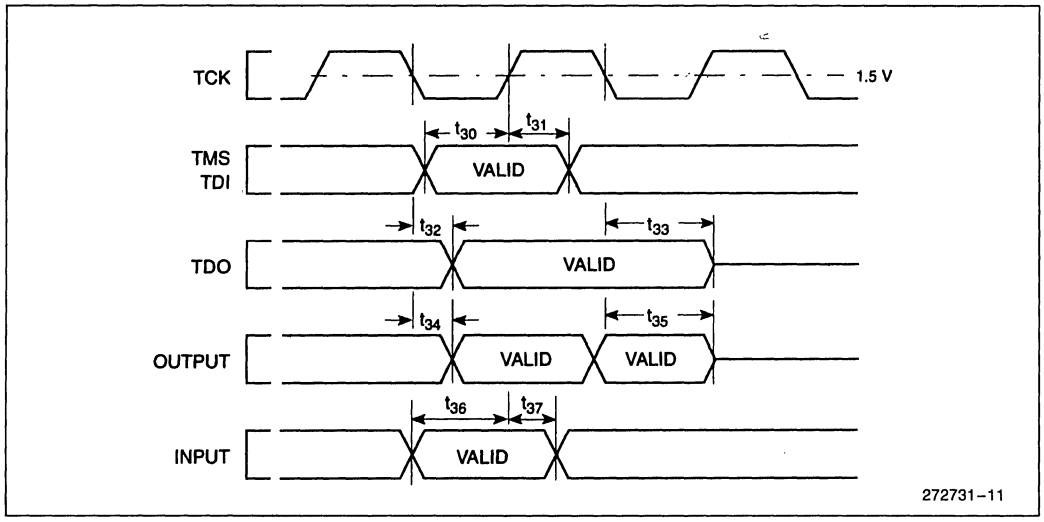
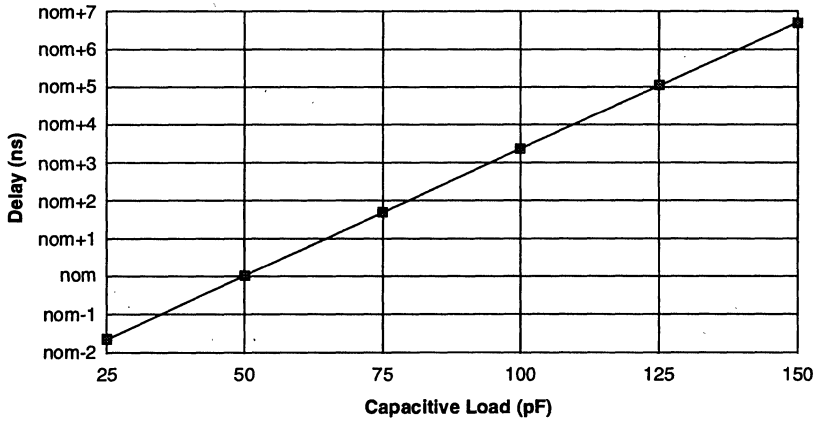


Figure 11. Test Signal Timing Diagram

4

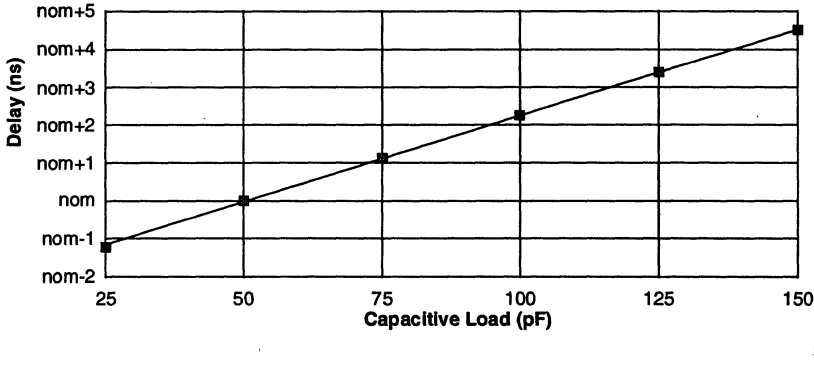
5.4 Capacitive Derating Curves

The following graphs are the capacitive derating curves for the embedded ULP Intel486 SX processor.



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 12. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



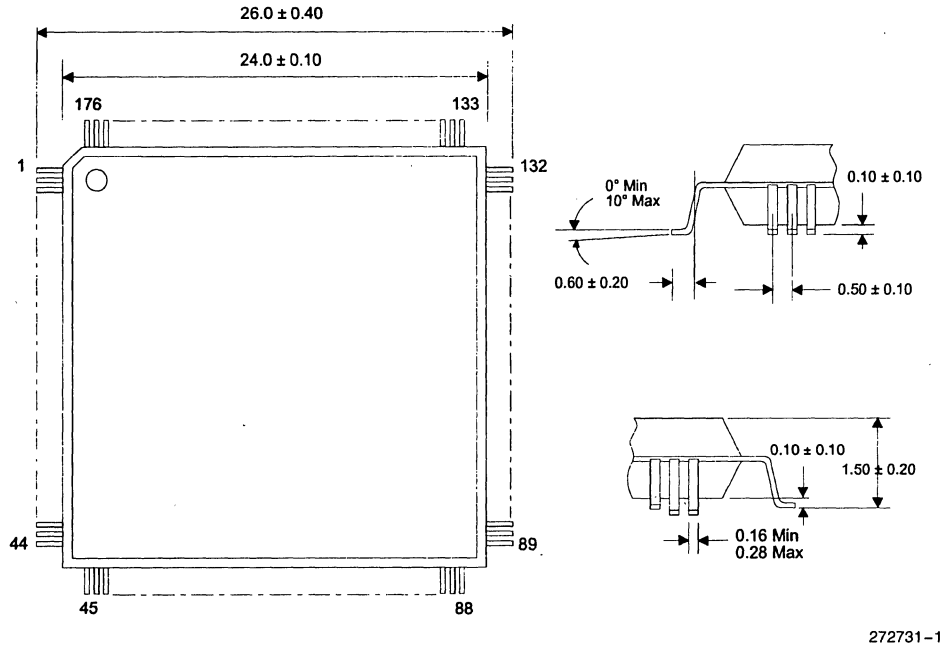
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 13. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition

6.0 MECHANICAL DATA

This section describes the packaging dimensions and thermal specifications for the embedded ULP Intel486 SX processor.

6.1 Package Dimensions



NOTE:
Height Measurements same as Width Measurements

Figure 14. Package Mechanical Specifications for the 176 Lead TQFP Package

6.2 Package Thermal Specifications

The embedded ULP Intel486 SX processor is specified for operation when the case temperature (T_C) is within the range of 0°C to 85°C. T_C may be measured in any environment to determine whether the processor is within the specified operating range.

The ambient temperature (T_A) can be calculated from θ_{JC} and θ_{JA} from the following equations:

$T_J = T_C + P * \theta_{JC}$
$T_A = T_J - P * \theta_{JA}$
$T_C = T_A + P * [\theta_{JA} - \theta_{JC}]$
$T_A = T_C - P * [\theta_{JA} - \theta_{JC}]$

Where T_J, T_A, T_C equals Junction, Ambient and Case Temperature respectively. θ_{JC}, θ_{JA} equals Junction-to-Case and Junction-to-Ambient thermal Resistance, respectively. Maximum Power Consumption (P) is defined as

$P = V\ (typ) * I_{CC}\ (max)$
$P = [V_{CC}\ (typ) * I_{CC1}\ (max)] + [V_{CCP}\ (typ) * I_{CC2}\ (max)]$

where: I_{CC1} is the V_{CC} supply current
I_{CC2} is the V_{CCP} supply current

Values for θ_{JA} and θ_{JC} are given in the following tables for each product at its maximum operating frequencies.

Table 18. Thermal Resistance

(°C/W) θ_{JC} and θ_{JA} for the 176-Lead TQFP Package

θ_{JC} (°C/W)	θ_{JA} (°C/W) with no airflow
4.3	33.6

The following table shows maximum ambient temperatures of the embedded ULP Intel486 SX processor for each product and maximum operating frequencies. These temperatures are calculated using I_{CC1} and I_{CC2} values measured during component-validation testing using V_{CCP} = 3.6V and worst-case V_{CC} values.

Table 19. Maximum Ambient Temperature (T_A)

176-Lead TQFP Package

Frequency	V_{CC}	T_A (°C) with no airflow
25 MHz	2.4V	76
	3.3V	65
33 MHz	2.7V	69
	3.3V	59

EMBEDDED Intel486™ SX PROCESSOR

- 32-Bit RISC Technology Core
- 8-Kbyte Write-Through Cache
- Four Internal Write Buffers
- Burst Bus Cycles
- Dynamic Bus Sizing for 8- and 16-bit Data Bus Devices
- SL Technology
- Data Bus Parity Generation and Checking
- Boundary Scan (JTAG)
- 3.3-Volt Processor
  — 208-Lead Shrink Quad Flat Pack (SQFP)
- 5-Volt Processor
  — 196-Lead Plastic Quad Flat Pack (PQFP)
- Binary Compatible with Large Software Base

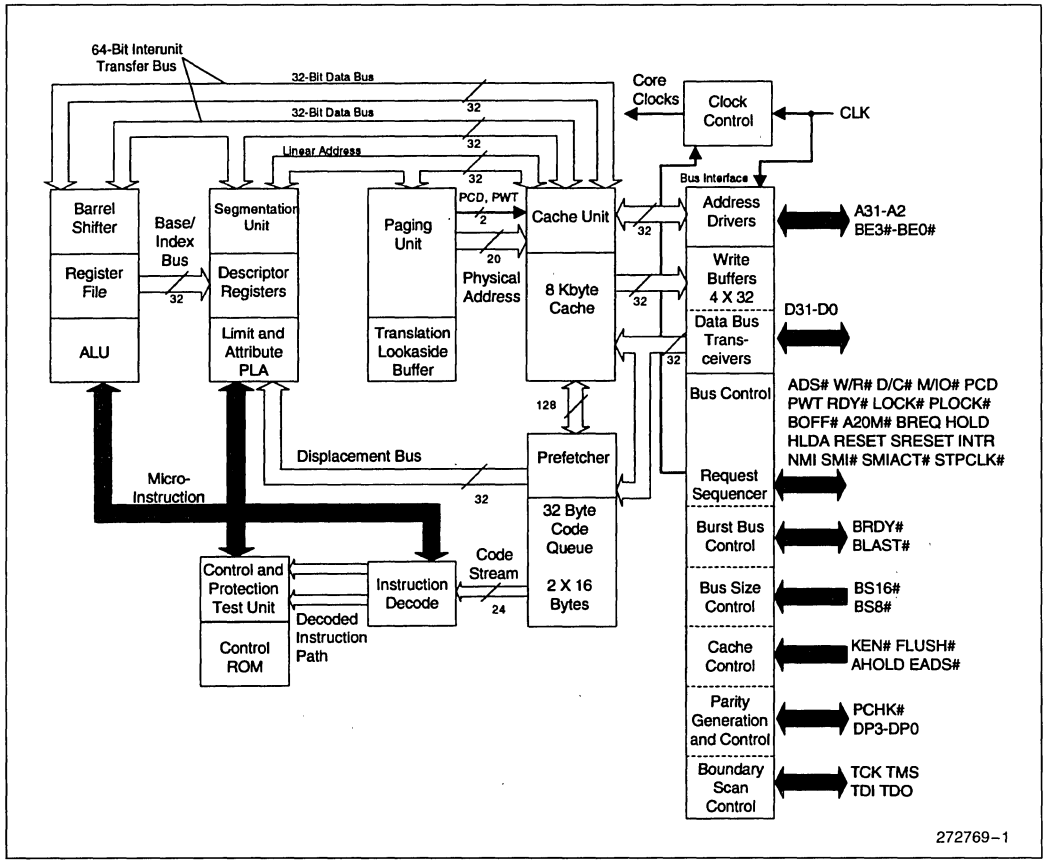


Figure 1. Embedded Intel486™ SX Processor Block Diagram

EMBEDDED Intel486™ SX PROCESSOR

CONTENTS

CONTENTS

PAGE

FIGURES

CONTENTS

PAGE

TABLES

1.0 INTRODUCTION

The embedded Intel486™ SX processor provides high performance to 32-bit, embedded applications. Designed for applications that do not need a floating-point unit, the processor is ideal for embedded designs running DOS*, Microsoft* Windows*, OS/2*, or UNIX* applications written for the Intel architecture. Projects can be completed quickly by utilizing the wide range of software tools, utilities, assemblers and compilers that are available for desktop computer systems. Also, developers can find advantages in using existing chipsets and peripheral components in their embedded designs.

The embedded Intel486 SX processor is binary compatible with the Intel386™ and earlier Intel processors. Compared with the Intel386 processor, it provides faster execution of many commonly-used instructions. It also provides the benefits of an integrated, 8-Kbyte, write-through cache for code and data. Its data bus can operate in burst mode which provides up to 106-Mbyte-per-second transfers for cache-line fills and instruction prefetches.

Intel's SL technology is incorporated in the embedded Intel486 SX processor. Utilizing Intel's System Management Mode (SMM), it enables designers to develop energy-efficient systems.

Two component packages are available. A 196-lead Plastic Quad Flat Pack (PQFP) is available for 5-Volt designs and a 208-lead Shrink Quad Flat Pack (SQFP) is available for 3.3-Volt designs. Both products operate at CLK frequencies up to 33 MHz.

1.1 Features

The embedded Intel486 SX processor offers these features:

- **32-bit RISC-Technology Core**—The embedded Intel486 SX processor performs a complete set of arithmetic and logical operations on 8-, 16-, and 32-bit data types using a full-width ALU and eight general purpose registers.
- **Single Cycle Execution**—Many instructions execute in a single clock cycle.
- **Instruction Pipelining**—Overlapped instruction fetching, decoding, address translation and execution.
- **On-Chip Cache with Cache Consistency Support**—An 8-Kbyte, write-through, internal cache is used for both data and instructions. Cache hits provide zero wait-state access times for data within the cache. Bus activity is tracked to detect alterations in the memory represented by the internal cache. The internal cache can be invalidated or flushed so that an external cache controller can maintain cache consistency.
- **External Cache Control**—Write-back and flush controls for an external cache are provided so the processor can maintain cache consistency.
- **On-Chip Memory Management Unit**—Address management and memory space protection mechanisms maintain the integrity of memory in a multitasking and virtual memory environment. Both memory segmentation and paging are supported.
- **Burst Cycles**—Burst transfers allow a new double-word to be read from memory on each bus clock cycle. This capability is especially useful for instruction prefetch and for filling the internal cache.
- **Write Buffers**—The processor contains four write buffers to enhance the performance of consecutive writes to memory. The processor can continue internal operations after a write to these buffers, without waiting for the write to be completed on the external bus.
- **Bus Backoff**—When another bus master needs control of the bus during a processor initiated bus cycle, the embedded Intel486 SX processor floats its bus signals, then restarts the cycle when the bus becomes available again.
- **Instruction Restart**—Programs can continue execution following an exception generated by an unsuccessful attempt to access memory. This feature is important for supporting demand-paged virtual memory applications.
- **Dynamic Bus Sizing**—External controllers can dynamically alter the effective width of the data bus. Bus widths of 8, 16, or 32 bits can be used.
- **Boundary Scan (JTAG)**—Boundary Scan provides in-circuit testing of components on printed circuit boards. The Intel Boundary Scan implementation conforms with the IEEE Standard Test Access Port and Boundary Scan Architecture.

Intel's SL technology provides these features:

- **Intel System Management Mode (SMM)**—A unique Intel architecture operating mode provides a dedicated special purpose interrupt and address space that can be used to implement intelligent power management and other enhanced functions in a manner that is completely transparent to the operating system and applications software.
- **I/O Restart**—An I/O instruction interrupted by a System Management Interrupt (SMI#) can automatically be restarted following the execution of the RSM instruction.
- **Stop Clock**—The embedded Intel486 SX processor has a stop clock control mechanism that provides two low-power states: a Stop Grant state (20–40 mA typical, depending on input clock frequency) and a Stop Clock state (~ 100–200 mA typical, with input clock frequency of 0 MHz).
- **Auto HALT Power Down**—After the execution of a HALT instruction, the embedded Intel486 SX processor issues a normal Halt bus cycle and the clock input to the processor core is automatically stopped, causing the processor to enter the Auto HALT Power Down state (20–40 mA typical, depending on input clock frequency).

1.2 Family Members

Table 1 shows the embedded Intel486 SX processors and briefly describes their characteristics.

Table 1. The Embedded Intel486™ SX Processor Family

Product	Supply Voltage V_{CC}	Maximum Processor Frequency	Package
SB80486SXSC33	3.3V	33 MHz	208-Lead SQFP
KU80486SXSA33	5.0V	33 MHz	196-Lead PQFP

2.0 HOW TO USE THIS DOCUMENT

For a complete set of documentation related to the embedded Intel486 SX processor, use this document in conjunction with the following reference documents:

- *Intel486™ Processor Family* datasheet—Order No. 242202
- *Intel486™ Microprocessor Family Programmer's Reference Manual*—Order No. 240486
- Intel Application Note AP-485—*Intel Processor Identification with the CPUID Instruction*—Order No. 241618

The information in the reference documents for the Intel486 SX processor, 1X Clock (CLK), applies to the embedded Intel486 SX processor. Some of the Intel486 SX processor information is duplicated in this document to minimize the dependence on the reference documents.

3.0 PIN DESCRIPTIONS

3.1 Pin Assignments

The following figures and tables show the pin assignments of each package type for the embedded Intel486 SX processor. Tables are provided showing the pin differences between the embedded Intel486 SX processor and other embedded Intel486 processor products.

208-Lead SQFP - Quad Flat Pack

196-Lead PQFP - Plastic Quad Flat Pack

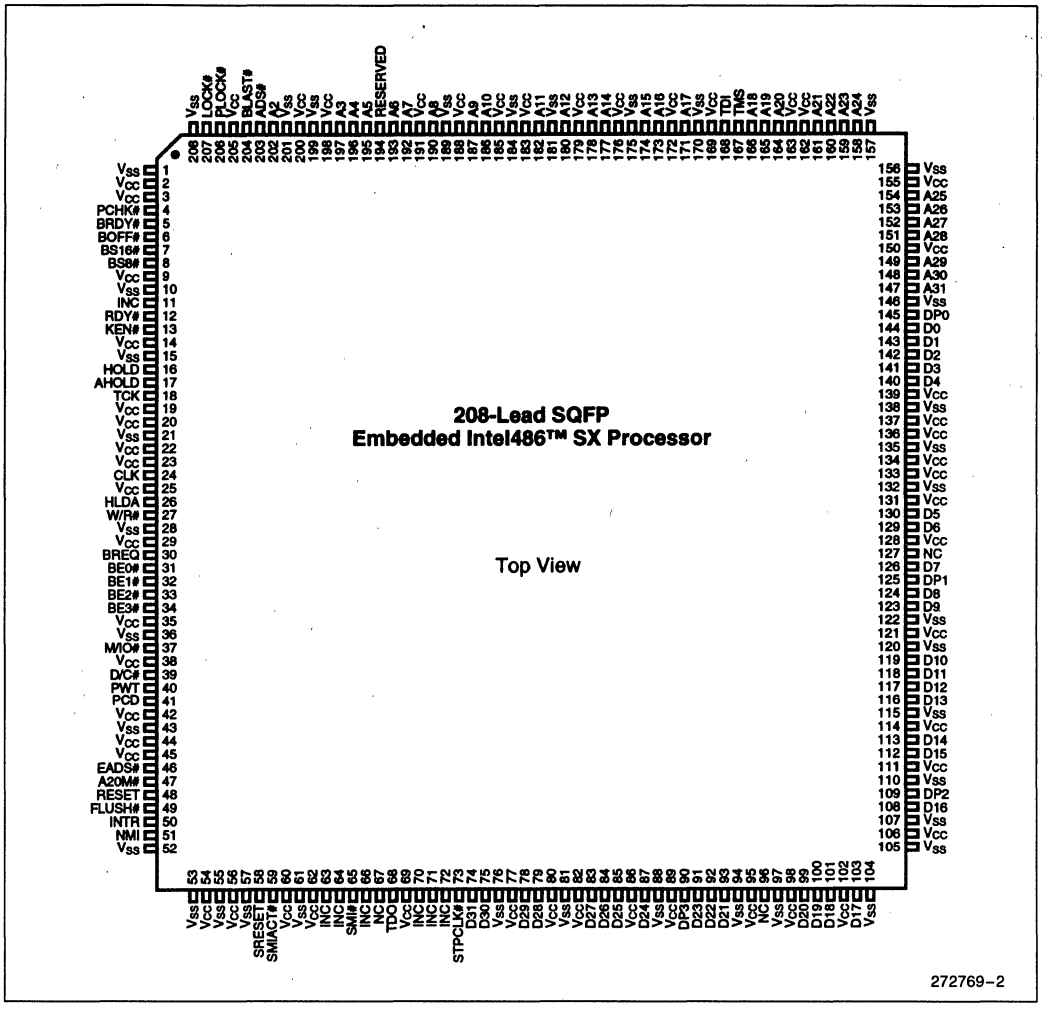


Figure 2. Package Diagram for 208-Lead SQFP Embedded Intel486™ SX Processor

Table 2. Pinout Differences for 208-Lead SQFP Package

Pin #	Embedded Intel486™ SX Processor	Embedded IntelDX2™ Processor	Embedded Write-Back Enhanced IntelDX4™ Processor
3	V_{CC}[1]	V_{CC}	V_{CC5}
11	INC[2]	INC	CLKMUL
63	INC	INC	HITM#
64	INC	INC	WB/WT#
66	INC	FERR#	FERR#
70	INC	INC	CACHE#
71	INC	INC	INV
72	INC	IGNNE#	IGNNE#

NOTES:
1. This pin location is for the V_{CC5} pin on the embedded IntelDX4 processor. For compatibility with 3.3V processors that have 5V-tolerant input buffers (i.e., embedded IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane.
2. INC. Internal No Connect. These pins are not connected to any internal pad in the embedded Intel486 SX processor. However, new signals are defined for the location of the INC pins in the embedded IntelDX2 and IntelDX4 processors. One system design can accommodate any one of these processors provided the purpose of each INC pin is understood before it is used.

Table 3. Pin Assignment for 208-Lead SQFP Package (Sheet 1 of 2)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
1	V_{SS}	53	V_{SS}	105	V_{SS}	157	V_{SS}
2	V_{CC}	54	V_{CC}	106	V_{CC}	158	A24
3	V_{CC}[1]	55	V_{SS}	107	V_{SS}	159	A23
4	PCHK#	56	V_{CC}	108	D16	160	A22
5	BRDY#	57	V_{SS}	109	DP2	161	A21
6	BOFF#	58	SRESET	110	V_{SS}	162	V_{CC}
7	BS16#	59	SMIACT#	111	V_{CC}	163	V_{CC}
8	BS8#	60	V_{CC}	112	D15	164	A20
9	V_{CC}	61	V_{SS}	113	D14	165	A19
10	V_{SS}	62	V_{CC}	114	V_{CC}	166	A18
11	INC[2]	63	INC[2]	115	V_{SS}	167	TMS
12	RDY#	64	INC[2]	116	D13	168	TDI
13	KEN#	65	SMI#	117	D12	169	V_{CC}
14	V_{CC}	66	INC[2]	118	D11	170	V_{SS}
15	V_{SS}	67	NC[3]	119	D10	171	A17
16	HOLD	68	TDO	120	V_{SS}	172	V_{CC}
17	AHOLD	69	V_{CC}	121	V_{CC}	173	A16
18	TCK	70	INC[2]	122	V_{SS}	174	A15
19	V_{CC}	71	INC[2]	123	D9	175	V_{SS}
20	V_{CC}	72	INC[2]	124	D8	176	V_{CC}
21	V_{SS}	73	STPCLK#	125	DP1	177	A14
22	V_{CC}	74	D31	126	D7	178	A13
23	V_{CC}	75	D30	127	NC[3]	179	V_{CC}
24	CLK	76	V_{SS}	128	V_{CC}	180	A12
25	V_{CC}	77	V_{CC}	129	D6	181	V_{SS}
26	HLDA	78	D29	130	D5	182	A11
27	W/R#	79	D28	131	V_{CC}	183	V_{CC}
28	V_{SS}	80	V_{CC}	132	V_{SS}	184	V_{SS}
29	V_{CC}	81	V_{SS}	133	V_{CC}	185	V_{CC}
30	BREQ	82	V_{CC}	134	V_{CC}	186	A10
31	BE0#	83	D27	135	V_{SS}	187	A9
32	BE1#	84	D26	136	V_{CC}	188	V_{CC}

Table 3. Pin Assignment for 208-Lead SQFP Package (Sheet 2 of 2)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
33	BE2#	85	D25	137	V_{CC}	189	V_{SS}
34	BE3#	86	V_{CC}	138	V_{SS}	190	A8
35	V_{CC}	87	D24	139	V_{CC}	191	V_{CC}
36	V_{SS}	88	V_{SS}	140	D4	192	A7
37	M/IO#	89	V_{CC}	141	D3	193	A6
38	V_{CC}	90	DP3	142	D2	194	RESERVED
39	D/C#	91	D23	143	D1	195	A5
40	PWT	92	D22	144	D0	196	A4
41	PCD	93	D21	145	DP0	197	A3
42	V_{CC}	94	V_{SS}	146	V_{SS}	198	V_{CC}
43	V_{SS}	95	V_{CC}	147	A31	199	V_{SS}
44	V_{CC}	96	NC[3]	148	A30	200	V_{CC}
45	V_{CC}	97	V_{SS}	149	A29	201	V_{SS}
46	EADS#	98	V_{CC}	150	V_{CC}	202	A2
47	A20M#	99	D20	151	A28	203	ADS#
48	RESET	100	D19	152	A27	204	BLAST#
49	FLUSH#	101	D18	153	A26	205	V_{CC}
50	INTR	102	V_{CC}	154	A25	206	PLOCK#
51	NMI	103	D17	155	V_{CC}	207	LOCK#
52	V_{SS}	104	V_{SS}	156	V_{SS}	208	V_{SS}

NOTES:
1. This pin location is for the V_{CC5} pin on the embedded IntelDX4 processor. For compatibility with 3.3V processors that have 5V-tolerant input buffers (i.e., embedded IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane.
2. INC. Internal No Connect. These pins are not connected to any internal pad in the embedded Intel486 SX processors. However, signals are defined for the location of the INC pins in the embedded IntelDX2 and IntelDX4 processors. One system design can accommodate any one of these processors provided the purpose of each INC pin is understood before it is used.
3. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC}, or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.

Table 4. Pin Cross Reference for 208-Lead SQFP Package (Sheet 1 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
A2	202	D0	144	A20M#	47	67	11	2	1
A3	197	D1	143	ADS#	203	96	63	3	10
A4	196	D2	142	AHOLD	17	127	64	9	15
A5	195	D3	141	BE0#	31		66	14	21
A6	193	D4	140	BE1#	32		70	19	28
A7	192	D5	130	BE2#	33		71	20	36
A8	190	D6	129	BE3#	34		72	22	43
A9	187	D7	126	BLAST#	204			23	52
A10	186	D8	124	BOFF#	6			25	53
A11	182	D9	123	BRDY#	5			29	55
A12	180	D10	119	BREQ	30			35	57
A13	178	D11	118	BS16#	7			38	61
A14	177	D12	117	BS8#	8			42	76
A15	174	D13	116	CLK	24			44	81
A16	173	D14	113	D/C#	39			45	88
A17	171	D15	112	DP0	145			54	94
A18	166	D16	108	DP1	125			56	97
A19	165	D17	103	DP2	109			60	104
A20	164	D18	101	DP3	90			62	105
A21	161	D19	100	EADS#	46			69	107
A22	160	D20	99	FLUSH#	49			77	110
A23	159	D21	93	HLDA	26			80	115
A24	158	D22	92	HOLD	16			82	120
A25	154	D23	91	INTR	50			86	122
A26	153	D24	87	KEN#	13			89	132
A27	152	D25	85	LOCK#	207			95	135
A28	151	D26	84	M/IO#	37			98	138
A29	149	D27	83	NMI	51			102	146
A30	148	D28	79	PCD	41			106	156
A31	147	D29	78	PCHK#	4			111	157
		D30	75	PLOCK#	206			114	170
		D31	74	PWT	40			121	175
				RDY#	12			128	181
				RESERVED	194			131	184
				RESET	48			133	189
				SMI#	65			134	199
				SMIACT#	59			136	201
				SRESET	58			137	208
				STPCLK#	73			139	

Table 4. Pin Cross Reference for 208-Lead SQFP Package (Sheet 2 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
				TCK	18			150	
				TDI	168			155	
				TDO	68			162	
				TMS	167			163	
				W/R#	27			169	
								172	
								176	
								179	
								183	
								185	
								188	
								191	
								198	
								200	
								205	

4

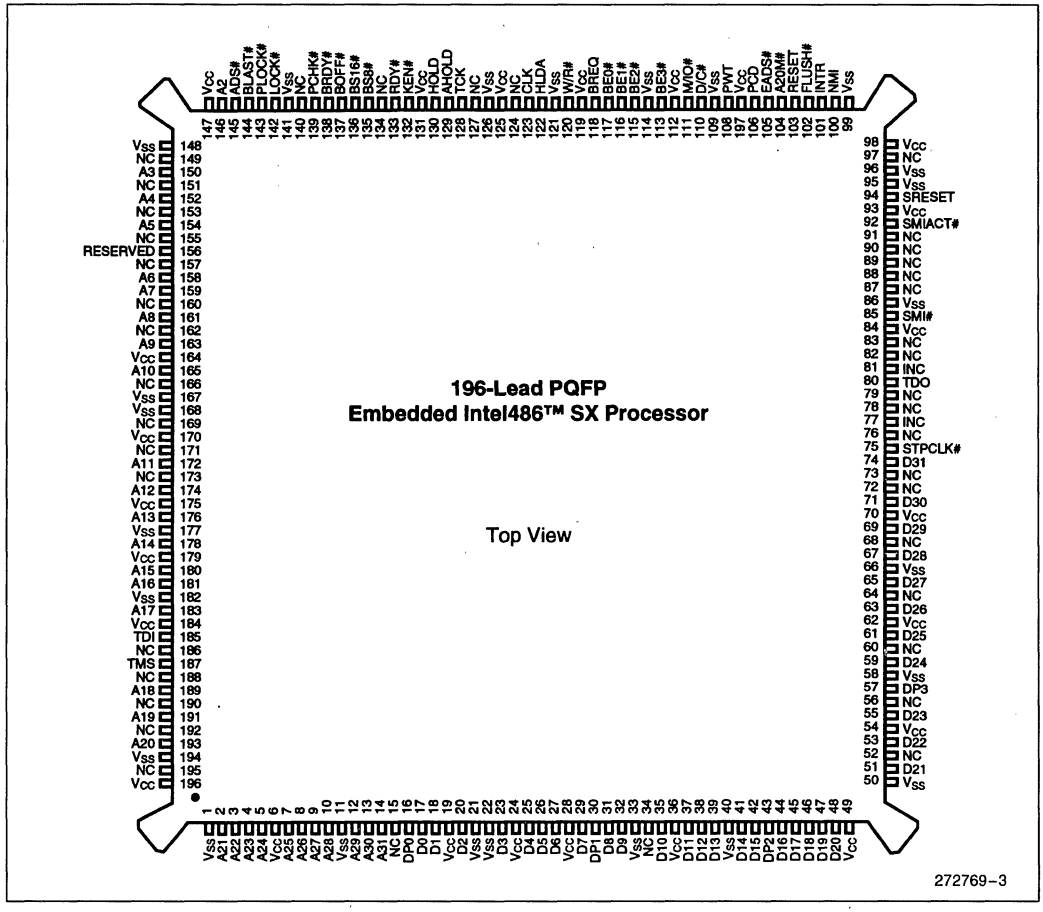


Figure 3. Package Diagram for 196-Lead PQFP Embedded Intel486™ SX Processor

Table 5. Pin Assignment for 196-Lead PQFP Package (Sheet 1 of 2)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
1	V_{SS}	50	V_{SS}	99	V_{SS}	148	V_{SS}
2	A21	51	D21	100	NMI	149	NC[1]
3	A22	52	NC[1]	101	INTR	150	A3
4	A23	53	D22	102	FLUSH#	151	NC[1]
5	A24	54	V_{CC}	103	RESET	152	A4
6	V_{CC}	55	D23	104	A20M#	153	NC[1]
7	A25	56	NC[1]	105	EADS#	154	A5
8	A26	57	DP3	106	PCD	155	NC[1]
9	A27	58	V_{SS}	107	V_{CC}	156	RESERVED
10	A28	59	D24	108	PWT	157	NC[1]
11	V_{SS}	60	NC[1]	109	V_{SS}	158	A6
12	A29	61	D25	110	D/C#	159	A7
13	A30	62	V_{CC}	111	M/IO#	160	NC[1]
14	A31	63	D26	112	V_{CC}	161	A8
15	NC[1]	64	NC[1]	113	BE3#	162	NC[1]
16	DP0	65	D27	114	V_{SS}	163	A9
17	D0	66	V_{SS}	115	BE2#	164	V_{CC}
18	D1	67	D28	116	BE1#	165	A10
19	V_{CC}	68	NC[1]	117	BE0#	166	NC[1]
20	D2	69	D29	118	BREQ	167	V_{SS}
21	V_{SS}	70	V_{CC}	119	V_{CC}	168	V_{SS}
22	V_{SS}	71	D30	120	W/R#	169	NC[1]
23	D3	72	NC[1]	121	V_{SS}	170	V_{CC}
24	V_{CC}	73	NC[1]	122	HLDA	171	NC[1]
25	D4	74	D31	123	CLK	172	A11
26	D5	75	STPCLK#	124	NC[1]	173	NC[1]
27	D6	76	NC[1]	125	V_{CC}	174	A12
28	V_{CC}	77	INC[2]	126	V_{SS}	175	V_{CC}
29	D7	78	NC[1]	127	NC[1]	176	A13
30	DP1	79	NC[1]	128	TCK	177	V_{SS}
31	D8	80	TDO	129	AHOLD	178	A14
32	D9	81	INC[2]	130	HOLD	179	V_{CC}

Table 5. Pin Assignment for 196-Lead PQFP Package (Sheet 2 of 2)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
33	V_{SS}	82	NC[1]	131	V_{CC}	180	A15
34	NC[1]	83	NC[1]	132	KEN#	181	A16
35	D10	84	V_{CC}	133	RDY#	182	V_{SS}
36	V_{CC}	85	SMI#	134	NC[1]	183	A17
37	D11	86	V_{SS}	135	BS8#	184	V_{CC}
38	D12	87	NC[1]	136	BS16#	185	TDI
39	D13	88	NC[1]	137	BOFF#	186	NC[1]
40	V_{SS}	89	NC[1]	138	BRDY#	187	TMS
41	D14	90	NC[1]	139	PCHK#	188	NC[1]
42	D15	91	NC[1]	140	NC[1]	189	A18
43	DP2	92	SMIACT#	141	V_{SS}	190	NC[1]
44	D16	93	V_{CC}	142	LOCK#	191	A19
45	D17	94	SRESET	143	PLOCK#	192	NC[1]
46	D18	95	V_{SS}	144	BLAST#	193	A20
47	D19	96	V_{SS}	145	ADS#	194	V_{SS}
48	D20	97	NC[1]	146	A2	195	NC[1]
49	V_{CC}	98	V_{CC}	147	V_{CC}	196	V_{CC}

NOTES:
1. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC}, or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.
2. INC. Internal No Connect. These pins are not connected to any internal pad in the embedded Intel486 SX processors.

Table 6. Pin Cross Reference for 196-Lead PQFP Package (Sheet 1 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
A2	146	D0	17	A20M#	104	15	77	6	1
A3	150	D1	18	ADS#	145	34	81	19	11
A4	152	D2	20	AHOLD	129	52		24	21
A5	154	D3	23	BE0#	117	56		28	22
A6	158	D4	25	BE1#	116	60		36	33
A7	159	D5	26	BE2#	115	64		49	40
A8	161	D6	27	BE3#	113	68		54	50
A9	163	D7	29	BLAST#	144	72		62	58
A10	165	D8	31	BOFF#	137	73		70	66
A11	172	D9	32	BRDY#	138	76		84	86
A12	174	D10	35	BREQ	118	78		93	95
A13	176	D11	37	BS16#	136	79		98	96
A14	178	D12	38	BS8#	135	82		107	99
A15	180	D13	39	CLK	123	83		112	109
A16	181	D14	41	D/C#	110	87		119	114
A17	183	D15	42	DP0	16	88		125	121
A18	189	D16	44	DP1	30	89		131	126
A19	191	D17	45	DP2	43	90		147	141
A20	193	D18	46	DP3	57	91		164	148
A21	2	D19	47	EADS#	105	97		170	167
A22	3	D20	48	FLUSH#	102	124		175	168
A23	4	D21	51	HLDA	122	127		179	177
A24	5	D22	53	HOLD	130	134		184	182
A25	7	D23	55	INTR	101	140		196	194
A26	8	D24	59	KEN#	132	149			
A27	9	D25	61	LOCK#	142	151			
A28	10	D26	63	M/IO#	111	153			
A29	12	D27	65	NMI	100	155			
A30	13	D28	67	PCD	106	157			
A31	14	D29	69	PCHK#	139	160			
		D30	71	PLOCK#	143	162			
		D31	74	PWT	108	166			
				RDY#	133	169			
				RESERVED	156	171			
				RESET	103	173			
				SMI#	85	186			
				SMIACT#	92	188			

Table 6. Pin Cross Reference for 196-Lead PQFP Package (Sheet 2 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
				SRESET	94	190			
				STPCLK#	75	192			
				TCK	128	195			
				TDI	185				
				TDO	80				
				TMS	187				
				W/R#	120				

3.2 Pin Quick Reference

The following is a brief pin description. For detailed signal descriptions refer to "Signal Description" in section 9 of the *Intel486™ Processor Family* datasheet.

Table 7. Embedded Intel486™ SX Processor Pin Descriptions (Sheet 1 of 6)

Symbol	Type	Name and Function
CLK	I	**Clock** provides the fundamental timing and internal operating frequency for the embedded Intel486 SX processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 **A3–A2**	I/O O	**Address Lines** A31–A2, together with the byte enable signals, BE3#–BE0#, define the physical area of memory or input/output space accessed. Address lines A31–A4 are used to drive addresses into the embedded Intel486 SX processor to perform cache line invalidation. Input signals must meet setup and hold times t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.
BE3# **BE2#** **BE1#** **BE0#**	O O O O	**Byte Enable** signals indicate active bytes during read and write cycles. During the first cycle of a cache fill, the external system should assume that all byte enables are active. BE3#–BE0# are active LOW and are not driven during bus hold. BE3# applies to D31–D24 BE2# applies to D23–D16 BE1# applies to D15–D8 BE0# applies to D7–D0
DATA BUS		
D31–D0	I/O	**Data Lines.** D7–D0 define the least significant byte of the data bus; D31–D24 define the most significant byte of the data bus. These signals must meet setup and hold times t_{22} and t_{23} for proper operation on reads. These pins are driven during the second and subsequent clocks of write cycles.
DATA PARITY		
DP3–DP0	I/O	There is one **Data Parity** pin for each byte of the data bus. Data parity is generated on all write data cycles with the same timing as the data driven by the embedded Intel486 SX processor. Even parity information must be driven back into the processor on the data parity pins with the same timing as read information to ensure that the correct parity check status is indicated by the embedded Intel486 SX processor. The signals read on these pins do not affect program execution. Input signals must meet setup and hold times t_{22} and t_{23}. DP3–DP0 must be connected to V_{CC} through a pull-up resistor in systems that do not use parity. DP3–DP0 are active HIGH and are driven during the second and subsequent clocks of write cycles.
PCHK#	O	**Parity Status** is driven on the PCHK# pin the clock after ready for read operations. The parity status is for data sampled at the end of the previous clock. A parity error is indicated by PCHK# being LOW. Parity status is only checked for enabled bytes as indicated by the byte enable and bus size signals. PCHK# is valid only in the clock immediately after read data is returned to the processor. At all other times PCHK# is inactive (HIGH). PCHK# is never floated.

Table 7. Embedded Intel486™ SX Processor Pin Descriptions (Sheet 2 of 6)

Symbol	Type	Name and Function
BUS CYCLE DEFINITION		
M/IO# **D/C#** **W/R#**	O O O	**Memory/Input-Output, Data/Control** and **Write/Read** lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted. **M/IO# D/C# W/R# Bus Cycle Initiated** 0 0 0 Interrupt Acknowledge 0 0 1 HALT/Special Cycle (see details below) 0 1 0 I/O Read 0 1 1 I/O Write 1 0 0 Code Read 1 0 1 Reserved 1 1 0 Memory Read 1 1 1 Memory Write **HALT/Special Cycle** **Cycle Name** — **BE3#-BE0#** — **A4-A2** Shutdown — 1110 — 000 HALT — 1011 — 000 Stop Grant bus cycle — 1011 — 100
LOCK#	O	**Bus Lock** indicates that the current bus cycle is locked. The embedded Intel486 SX processor does not allow a bus hold when LOCK# is asserted (address holds are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the last clock of the last locked bus cycle. The last locked cycle ends when Ready is returned. LOCK# is active LOW and not driven during bus hold. Locked read cycles are not transformed into cache fill cycles when KEN# is returned active.
PLOCK#	O	**Pseudo-Lock** indicates that the current bus transaction requires more than one bus cycle to complete. For the embedded Intel486 SX processor, examples of such operations are segment table descriptor reads (64 bits) and cache line fills (128 bits). The embedded Intel486 SX processor drives PLOCK# active until the addresses for the last bus cycle of the transaction are driven, regardless of whether RDY# or BRDY# have been returned. PLOCK# is a function of the BS8#, BS16# and KEN# inputs. PLOCK# should be sampled only in the clock in which Ready is returned. PLOCK# is active LOW and is not driven during bus hold.
BUS CONTROL		
ADS#	O	**Address Status** output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus. ADS# is driven active in the same clock in which the addresses are driven. ADS# is active LOW and not driven during bus hold.

Table 7. Embedded Intel486™ SX Processor Pin Descriptions (Sheet 3 of 6)

Symbol	Type	Name and Function
RDY#	I	**Non-burst Ready** input indicates that the current bus cycle is complete. RDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted data from the embedded Intel486 SX processor in response to a write. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. RDY# is active during address hold. Data can be returned to the embedded Intel486 SX processor while AHOLD is active. RDY# is active LOW and is not provided with an internal pull-up resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.
BURST CONTROL		
BRDY#	I	**Burst Ready** input performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data in response to a read or that the external system has accepted data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle. BRDY# is sampled in the second and subsequent clocks of a burst cycle. Data presented on the data bus is strobed into the embedded Intel486 SX processor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. BRDY# is active LOW and is provided with a small pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.
BLAST#	O	**Burst Last** signal indicates that the next time BRDY# is returned, the burst bus cycle is complete. BLAST# is active for both burst and non-burst bus cycles. BLAST# is active LOW and is not driven during bus hold.
INTERRUPTS		
RESET	I	**Reset** input forces the embedded Intel486 SX processor to begin execution at a known state. The processor cannot begin executing instructions until at least 1 ms after V_{CC}, and CLK have reached their proper DC and AC specifications. The RESET pin must remain active during this time to ensure proper processor operation. However, for warm resets, RESET should remain active for at least 15 CLK periods. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
INTR	I	**Maskable Interrupt** indicates that an external interrupt has been generated. When the internal interrupt flag is set in EFLAGS, active interrupt processing is initiated. The embedded Intel486 SX processor generates two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the interrupt acknowledges have been performed to ensure processor recognition of the interrupt. INTR is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

Table 7. Embedded Intel486™ SX Processor Pin Descriptions (Sheet 4 of 6)

Symbol	Type	Name and Function
NMI	I	**Non-Maskable Interrupt** request signal indicates that an external non-maskable interrupt has been generated. NMI is rising-edge sensitive and must be held LOW for at least four CLK periods before this rising edge. NMI is not provided with an internal pull-down resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SRESET	I	**Soft Reset** pin duplicates all functionality of the RESET pin except that the SMBASE register retains its previous value. For soft resets, SRESET must remain active for at least 15 CLK periods. SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SMI#	I	**System Management Interrupt** input invokes System Management Mode (SMM). SMI# is a falling-edge triggered signal which forces the embedded Intel486 SX processor into SMM at the completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# does not break LOCKed bus cycles and cannot interrupt a currently executing SMM. The embedded Intel486 SX processor latches the falling edge of one pending SMI# signal while it is executing an existing SMI#. The nested SMI# is not recognized until after the execution of a Resume (RSM) instruction.
SMIACT#	O	**System Management Interrupt Active,** an active LOW output, indicates that the embedded Intel486 SX processor is operating in SMM. It is asserted when the processor begins to execute the SMI# state save sequence and remains active LOW until the processor executes the last state restore cycle out of SMRAM.
STPCLK#	I	**Stop Clock Request** input signal indicates a request was made to turn off or change the CLK input frequency. When the embedded Intel486 SX processor recognizes a STPCLK#, it stops execution on the next instruction boundary (unless superseded by a higher priority interrupt), empties all internal pipelines and write buffers, and generates a Stop Grant bus cycle. STPCLK# is active LOW. **STPCLK# is an asynchronous signal, but must remain active until the embedded Intel486 SX processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle.**
BUS ARBITRATION		
BREQ	O	**Bus Request** signal indicates that the embedded Intel486 SX processor has internally generated a bus request. BREQ is generated whether or not the processor is driving the bus. BREQ is active HIGH and is never floated.
HOLD	I	**Bus Hold Request** allows another bus master complete control of the embedded Intel486 SX processor bus. In response to HOLD going active, the processor floats most of its output and input/output pins. HLDA is asserted after completing the current bus cycle, burst cycle or sequence of locked cycles. The embedded Intel486 SX processor remains in this state until HOLD is de-asserted. HOLD is active HIGH and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper operation.

Table 7. Embedded Intel486™ SX Processor Pin Descriptions (Sheet 5 of 6)

Symbol	Type	Name and Function
HLDA	O	**Hold Acknowledge** goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the embedded Intel486 SX processor has given the bus to another local bus master. HLDA is driven active in the same clock that the processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.
BOFF#	I	**Backoff** input forces the embedded Intel486 SX processor to float its bus in the next clock. The processor floats all pins normally floated during bus hold but HLDA is not asserted in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#; if both are returned in the same clock, BOFF# takes effect. The embedded Intel486 SX processor remains in bus hold until BOFF# is negated. If a bus cycle is in progress when BOFF# is asserted the cycle is restarted. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper operation.
CACHE INVALIDATION		
AHOLD	I	**Address Hold** request allows another bus master access to the embedded Intel486 SX processor's address bus for a cache invalidation cycle. The processor stops driving its address bus in the clock following AHOLD going active. Only the address bus is floated during address hold, the remainder of the bus remains active. AHOLD is active HIGH and is provided with a small internal pull-down resistor. For proper operation, AHOLD must meet setup and hold times t_{18} and t_{19}.
EADS#	I	**External Address**—This signal indicates that a *valid* external address has been driven onto the embedded Intel486 SX processor address pins. This address is used to perform an internal cache invalidation cycle. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy setup and hold times t_{12} and t_{13} for proper operation.
CACHE CONTROL		
KEN#	I	**Cache Enable** pin is used to determine whether the current cycle is cacheable. When the embedded Intel486 SX processor generates a cycle that can be cached and KEN# is active one clock before RDY# or BRDY# during the first transfer of the cycle, the cycle becomes a cache line fill cycle. Returning KEN# active one clock before RDY# during the last read in the cache line fill causes the line to be placed in the on-chip cache. KEN# is active LOW and is provided with a small internal pull-up resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	**Cache Flush** input forces the embedded Intel486 SX processor to flush its entire internal cache. FLUSH# is active LOW and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition in any specific clock.
PAGE CACHEABILITY		
PWT **PCD**	O O	**Page Write-Through** and **Page Cache Disable** pins reflect the state of the page attribute bits, PWT and PCD, in the page table entry, page directory entry or control register 3 (CR3) when paging is enabled. When paging is disabled, the embedded Intel486 SX processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT pins. PWT and PCD have the same timing as the cycle definition pins (M/IO#, D/C#, and W/R#). PWT and PCD are active HIGH and are not driven during bus hold. PCD is masked by the cache disable bit (CD) in Control Register 0.

Table 7. Embedded Intel486™ SX Processor Pin Descriptions (Sheet 6 of 6)

Symbol	Type	Name and Function
BUS SIZE CONTROL		
BS16# **BS8#**	I I	**Bus Size 16 and Bus Size 8** pins (bus sizing pins) cause the embedded Intel486 SX processor to run multiple bus cycles to complete a request from devices that cannot provide or accept 32 bits of data in a single cycle. The bus sizing pins are sampled every clock. The processor uses the state of these pins in the clock before Ready to determine bus size. These signals are active LOW and are provided with internal pull-up resistors. These inputs must satisfy setup and hold times t_{14} and t_{15} for proper operation.
ADDRESS MASK		
A20M#	I	**Address Bit 20 Mask** pin, when asserted, causes the embedded Intel486 SX processor to mask physical address bit 20 (A20) before performing a lookup to the internal cache or driving a memory cycle on the bus. A20M# emulates the address wraparound at 1 Mbyte, which occurs on the 8086 processor. A20M# is active LOW and should be asserted only when the embedded Intel486 SX processor is in real mode. This pin is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For proper operation, A20M# should be sampled HIGH at the falling edge of RESET.
TEST ACCESS PORT		
TCK	I	**Test Clock,** an input to the embedded Intel486 SX processor, provides the clocking function required by the JTAG Boundary scan feature. TCK is used to clock state information (via TMS) and data (via TDI) into the component on the rising edge of TCK. Data is clocked out of the component (via TDO) on the falling edge of TCK. TCK is provided with an internal pull-up resistor.
TDI	I	**Test Data Input** is the serial input used to shift JTAG instructions and data into the processor. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and SHIFT-DR Test Access Port (TAP) controller states. During all other TAP controller states, TDI is a "don't care." TDI is provided with an internal pull-up resistor.
TDO	O	**Test Data Output** is the serial output used to shift JTAG instructions and data out of the component. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state.
TMS	I	**Test Mode Select** is decoded by the JTAG TAP to select test logic operation. TMS is sampled on the rising edge of TCK. To guarantee deterministic behavior of the TAP controller, TMS is provided with an internal pull-up resistor.
RESERVED PINS		
RESERVED	I	**Reserved** is reserved for future use. This pin MUST be connected to an external pull-up resistor circuit. The recommended resistor value is 10 kOhms. The pull-up resistor must be connected only to the RESERVED pin. **Do not share this resistor with other pins requiring pull-ups.**

Table 8. Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States
BREQ	HIGH			Previous State
HLDA	HIGH			As per HOLD
BE3#–BE0#	LOW		•	Previous State
PWT, PCD	HIGH		•	Previous State
W/R#, M/IO#, D/C#	HIGH/LOW		•	Previous State
LOCK#	LOW		•	HIGH (inactive)
PLOCK#	LOW		•	HIGH (inactive)
ADS#	LOW		•	HIGH (inactive)
BLAST#	LOW		•	Previous State
PCHK#	LOW			Previous State
A3–A2	HIGH	•	•	Previous State
SMIACT#	LOW			Previous State

NOTE:
The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

4

Table 9. Input/Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States
D31–D0	HIGH		•	Floated
DP3–DP0	HIGH		•	Floated
A31–A4	HIGH	•	•	Previous State

NOTE:
The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

Table 10. Test Pins

Name	Input or Output	Sampled/ Driven On
TCK	Input	N/A
TDI	Input	Rising Edge of TCK
TDO	Output	Failing Edge of TCK
TMS	Input	Rising Edge of TCK

Table 11. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Pull-Up/ Pull-Down
CLK			
RESET	HIGH	Asynchronous	
SRESET	HIGH	Asynchronous	Pull-Down
HOLD	HIGH	Synchronous	
AHOLD	HIGH	Synchronous	Pull-Down
EADS#	LOW	Synchronous	Pull-Up
BOFF#	LOW	Synchronous	Pull-Up
FLUSH#	LOW	Asynchronous	Pull-Up
A20M#	LOW	Asynchronous	Pull-Up
BS16#, BS8#	LOW	Synchronous	Pull-Up
KEN#	LOW	Synchronous	Pull-Up
RDY#	LOW	Synchronous	
BRDY#	LOW	Synchronous	Pull-Up
INTR	HIGH	Asynchronous	
NMI	HIGH	Asynchronous	
RESERVED			
SMI#	LOW	Asynchronous	Pull-Up
STPCLK#	LOW	Asynchronous	Pull-Up[1]
TCK	HIGH		Pull-Up
TDI	HIGH		Pull-Up
TMS	HIGH		Pull-Up

NOTE:
1. Though STPCLK# has an internal pull-up resistor, an external 10-KΩ pull-up resistor is needed if the STPCLK# pin is not used.

4.0 ARCHITECTURAL AND FUNCTIONAL OVERVIEW

The embedded Intel486 SX processor architecture is essentially the same as the Intel486 SX processor with a 1X clock (CLK) input. Refer to the *Intel486™ Processor Family* datasheet (242202) for a description of the Intel486 SX processor. With some minor exceptions, the following datasheet sections apply to the embedded Intel486 SX processor:

- Architectural Overview
- Real Mode Architecture
- Protected Mode Architecture
- On-Chip Cache
- System Management Mode (SMM) Architectures
- Hardware Interface
- Bus Operation
- Testability
- Debugging Support
- Instruction Set Summary
- Differences Between Intel486 Processors and Intel386™ Processors

Exceptions to these sections of the datasheet are:

- References to the Upgrade Power Down Mode do not apply. The embedded Intel486 SX processor does not have the UP# signal pin and does not support the Intel OverDrive® processor.

- The embedded Intel486 SX processor has one pin reserved for possible future use. This pin, an input signal, is called RESERVED and must be connected to a 10-KΩ pull-up resistor. The pull-up resistor must be connected only to the RESERVED pin. **Do not share this resistor with other pins requiring pull-ups.**

4.1 CPUID Instruction

The embedded Intel486 SX processor supports the CPUID instruction (see Table 12). Because not all Intel processors support the CPUID instruction, a simple test can determine if the instruction is supported. The test involves the processor's ID Flag, which is bit 21 of the EFLAGS register. If software can change the value of this flag, the CPUID instruction is available. The actual state of the ID Flag bit is irrelevant and provides no significance to the hardware.

This bit is cleared (reset to zero) upon device reset (RESET or SRESET) for compatibility with Intel486 processor designs that do not support the CPUID instruction.

CPUID-instruction details are provided here for the embedded Intel486 SX processor. Refer to Intel Application Note AP-485 *Intel Processor Identification with the CPUID Instruction* (Order No. 241618) for a description that covers all aspects of the CPUID instruction and how it pertains to other Intel processors.

4.1.1 Operation of the CPUID Instruction

The CPUID instruction requires the software developer to pass an input parameter to the processor in the EAX register. The processor response is returned in registers EAX, EBX, EDX, and ECX.

Table 12. CPUID Instruction Description

OP CODE	Instruction	Processor Core Clocks	Parameter passed in EAX (Input Value)	Description
0F A2	CPUID	9	0	Vendor (Intel) ID String
		14	1	Processor Identification
		9	> 1	Undefined (Do Not Use)

Vendor ID String—When the parameter passed in EAX is 0 (zero), the register values returned upon instruction execution are shown in the following table.

		31 - - - - - - 24	23 - - - - - - 16	15 - - - - - - 8	7 - - - - - - 0
High Value (= 1)	**EAX**	0000	0000	0000	0001
Vendor ID String (ASCII Characters)	**EBX**	u (75)	n (6E)	e (65)	G (47)
	EDX	I (49)	e (65)	n (6E)	i (69)
	ECX	l (6C)	e (65)	t (74)	n (6E)

The values in EBX, EDX and ECX indicate an Intel processor. When taken in the proper order, they decode to the string `"GenuineIntel."`

Processor Identification—When the parameter passed to EAX is 1 (one), the register values returned upon instruction execution are:

		31 – – – – – – – – 14	13, 12	11 – – – 8	7 – – – 4	3 – – – 0
Processor Signature	**EAX**	(Do Not Use) Intel Reserved	0 0 Processor Type	0 1 0 0 Family	0 0 1 0 Model	XXXX Stepping

(Intel releases information about stepping numbers as needed)

		31 – 0
Intel Reserved	**EBX**	Intel Reserved
(Do Not Use)	**ECX**	Intel Reserved

		31 – 2	1	0
Feature Flags	**EDX**	31 – 0	1 VME	0 FPU

4.2 Identification After Reset

Processor Identification—Upon reset, the EDX register contains the processor signature:

		31 – – – – – – – – – 14	13, 12	11 – – – – 8	7 – – – – 4	3 – – – – 0
Processor Signature	**EDX**	(Do Not Use) Intel Reserved	0 0 Processor Type	0 1 0 0 Family	0 0 1 0 Model	XXXX Stepping

(Intel releases information about stepping numbers as needed)

4.3 Boundary Scan (JTAG)

4.3.1 Device Identification

Tables 13 and 14 show the 32-bit code for the embedded Intel486 SX processor. This code is loaded into the Device Identification Register.

Table 13. Boundary Scan Component Identification Code (3.3 Volt Processor)

Version	**Part Number**				**Mfg ID**	**1**
	V_{CC} 0=5V 1=3.3V	**Intel Architecture Type**	**Family** 0100 = Intel486 CPU Family	**Model** 00010 = embedded Intel486 SX processor	009H = Intel	
31 – – – 28	27	26 – – – – 21	20 – – – – 17	16 – – – – 12	11 – – – – 1	0
XXXX	1	000001	0100	00010	00000001001	1

(Intel releases information about version numbers as needed)
Boundary Scan Component Identification Code = x828 2013 (Hex)

Table 14. Boundary Scan Component Identification Code (5 Volt Processor)

Version	Part Number				Mfg ID	1
	V_{CC} 0=5V 1=3.3V	Intel Architecture Type	Family 0100 = Intel486 CPU Family	Model 00010 = embedded Intel486 SX processor	009H = Intel	
31 - - - 28	27	26 - - - - 21	20 - - - - 17	16 - - - - 12	11 - - - - 1	0
XXXX	0	000001	0100	00010	00000001001	1

(Intel releases information about version numbers as needed)
Boundary Scan Component Identification Code = x028 2013 (Hex)

4.3.2 Boundary Scan Register Bits and Bit Order

The boundary scan register contains a cell for each pin as well as cells for control of bidirectional and three-state pins. There are "Reserved" bits which correspond to no-connect (N/C) signals of the embedded Intel486 SX processor. Control registers WRCTL, ABUSCTL, BUSCTL, and MISCCTL are used to select the direction of bidirectional or three-state output signal pins. A "1" in these cells designates that the associated bus or bits are floated if the pins are three-state, or selected as input if they are bidirectional.

- WRCTL controls D31-D0 and DP3-DP0
- ABUSCTL controls A31-A2
- BUSCTL controls ADS#, BLAST#, PLOCK#, LOCK#, W/R#, BE0#, BE1#, BE2#, BE3#, M/IO#, D/C#, PWT, and PCD
- MISCCTL controls PCHK#, HLDA, and BREQ

The following is the bit order of the embedded Intel486 SX processor boundary scan register:

TDO ← A2, A3, A4, A5, RESERVED, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, Reserved, Reserved, SMI#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRCTL **← TDI**

5.0 ELECTRICAL SPECIFICATIONS

5.1 Maximum Ratings

Table 15 is a stress rating only. Extended exposure to the Maximum Ratings may affect device reliability.

Furthermore, although the embedded Intel486 SX processor contains protective circuitry to resist damage from electrostatic discharge, always take precautions to avoid high static voltages or electric fields.

Functional operating conditions are given in **Section 5.2, DC Specifications and Section 5.3, AC Specifications.**

Table 15. Absolute Maximum Ratings

Case Temperature under Bias	−65°C to +110°C
Storage Temperature	−65°C to +150°C
DC Voltage on Any Pin with Respect to Ground	−0.5V to V_{CC} + 0.5V
Supply Voltage V_{CC} with Respect to V_{SS}	−0.5V to +6.5V

5.2 DC Specifications

The following tables show the operating supply voltages, DC I/O specifications, and component power consumption for the embedded Intel486 SX processor.

Table 16. Operating Supply Voltages

Product	V_{CC}
SB80486SXSC33	3.3V ± 0.3V
KU80486SXSA33	5.0V ± 0.25V

Table 17. 3.3V DC Specifications

Functional Operating Range: V_{CC} = 3.3V ± 0.3V, T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} +0.3	V	Note 1
V_{IHC}	Input HIGH Voltage of CLK	V_{CC} −0.6	V_{CC} +0.3	V	
V_{OL}	Output LOW Voltage I_{OL} = 2.0 mA I_{OL} = 100 μA		 0.4 0.2	 V V	
V_{OH}	Output HIGH Voltage I_{OH} = −2.0 mA I_{OH} = −100 μA	 2.4 V_{CC} −0.2		 V V	
I_{LI}	Input Leakage Current		±15	μA	Note 2
I_{IH}	Input Leakage Current SRESET		200 300	μA μA	Note 3 Note 3
I_{IL}	Input Leakage Current		−400	μA	Note 4
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		10	pF	Note 5
C_{OUT}	I/O or Output Capacitance		10	pF	Note 5
C_{CLK}	CLK Capacitance		6	pF	Note 5

NOTES:
1. All inputs except CLK.
2. This parameter is for inputs without pull-up or pull-down resistors and $0V \leq V_{IN} \leq V_{CC}$.
3. This parameter is for inputs with pull-down resistors and V_{IH} = 2.4V.
4. This parameter is for inputs with pull-up resistors and V_{IL} = 0.4V.
5. F_C = 1 MHz. Not 100% tested.

Table 18. 3.3V I_{CC} Values

Functional Operating Range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	25 MHz 33 MHz		315 mA 415 mA	Note 1
I_{CC} Active (Thermal Design)	25 MHz 33 MHz	220 mA 289 mA	292 mA 356 mA	Notes 2, 3, 4
I_{CC} Stop Grant	25 MHz 33 MHz	20 mA 25 mA	40 mA 50 mA	Note 5
I_{CC} Stop Clock	0 MHz	100 μA	1 mA	Note 6

NOTES:
1. This parameter is for proper power supply selection. It is measured using the worst case instruction mix at V_{CC} = 3.6V.
2. The maximum current column is for thermal design power dissipation. It is measured using the worst case instruction mix at V_{CC} = 3.3V.
3. The typical current column is the typical operating current in a system. This value is measured in a system using a typical device at V_{CC} = 3.3V, running Microsoft Windows 3.1 at an idle condition. This typical value is dependent upon the specific system configuration.
4. Typical values are not 100% tested.
5. The I_{CC} Stop Grant specification refers to the I_{CC} value once the embedded Intel486 SX processor enters the Stop Grant or Auto HALT Power Down state.
6. The I_{CC} Stop Clock specification refers to the I_{CC} value once the embedded Intel486 SX processor enters the Stop Clock state. The V_{IH} and V_{IL} levels must be equal to V_{CC} and 0V, respectively, in order to meet the I_{CC} Stop Clock specifications.

Table 19. 5V DC Specifications

Functional operating range: V_{CC} = 5V ± 0.25V; T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} +0.3	V	
V_{OL}	Output LOW Voltage		0.45	V	Note 1
V_{OH}	Output HIGH Voltage	2.4		V	Note 2
I_{LI}	Input Leakage Current		±15	μA	Note 3
I_{IH}	Input Leakage Current SRESET		200 300	μA μA	Note 4 Note 4
I_{IL}	Input Leakage Current		−400	μA	Note 5
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		20	pF	Note 6
C_{OUT}	Output or I/O Capacitance		20	pF	Note 6
C_{CLK}	CLK Capacitance		20	pF	Note 6

NOTES:
1. This parameter is measured at:
   Address, Data, BEn 4.0 mA
   Definition, Control 5.0 mA
2. This parameter is measured at:
   Address, Data, BEn −1.0 mA
   Definition, Control −0.9 mA
3. This parameter is for inputs without pull-ups or pull-downs and $0V \leq V_{IN} \leq V_{CC}$.
4. This parameter is for inputs with pull-downs and V_{IH} = 2.4V.
5. This parameter is for inputs with pull-ups and V_{IL} = 0.45V.
6. F_C=1 MHz; Not 100% tested.

Table 20. 5V I_{CC} Values

Functional Operating Range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	25 MHz 33 MHz		560 mA 685 mA	Note 1
I_{CC} Active (Thermal Design)	25 MHz 33 MHz	378 mA 497 mA	535 mA 654 mA	Notes 2, 3, 4
I_{CC} Stop Grant	25 MHz 33 MHz	35 mA 40 mA	65 mA 80 mA	Note 5
I_{CC} Stop Clock	0 MHz	200 μA	2 mA	Note 6

NOTES:
1. This parameter is for proper power supply selection. It is measured using the worst case instruction mix at V_{CC} = 5.25V.
2. The maximum current column is for thermal design power dissipation. It is measured using the worst case instruction mix at V_{CC} = 5V.
3. The typical current column is the typical operating current in a system. This value is measured in a system using a typical device at V_{CC} = 5V, running Microsoft Windows 3.1 at an idle condition. This typical value is dependent upon the specific system configuration.
4. Typical values are not 100% tested.
5. The I_{CC} Stop Grant specification refers to the I_{CC} value once the embedded Intel486 SX processor enters the Stop Grant or Auto HALT Power Down state.
6. The I_{CC} Stop Clock specification refers to the I_{CC} value once the processor enters the Stop Clock state. The V_{IH} and V_{IL} levels must be equal to V_{CC} and 0V, respectively, in order to meet the I_{CC} Stop Clock specifications.

5.3 AC Specifications

The AC specifications for the embedded Intel486 SX processor are given in this section.

Table 21. AC Characteristics

T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified. (Sheet 1 of 2)

Symbol	Parameter	V_{CC} (Package)						
		3.3V (208-Lead SQFP)		5V (196-Lead PQFP)				
		Min	Max	Min	Max	Unit	Figure	Notes
	CLK Frequency	8	33	8	33	MHz		Note 1
t1	CLK Period	30	125	30	125	ns	4	
t_{1a}	CLK Period Stability		±250		±250	ps	4	Adjacent clocks
t_2	CLK High Time	11		11		ns	4	at 2V
t_3	CLK Low Time	11		11		ns	4	at 0.8V
t_4	CLK Fall Time		3		3	ns	4	2V to 0.8V
t_5	CLK Rise Time		3		3	ns	4	0.8V to 2V
t_6	A31–A2, PWT, PCD, BE3–BE0#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT# Valid Delay	3	16	3	16	ns	8	
t_7	A31–A2, PWT, PCD, BE3–BE0#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA Float Delay		20		20	ns	9	Note 2
t_8	PCHK# Valid Delay	3	22	3	22	ns	7	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	20	3	20	ns	8	
t_9	BLAST#, PLOCK# Float Delay		20		20	ns	9	Note 2
t_{10}	D31–D0, DP3–DP0 Write Data Valid Delay	3	19	3	18	ns	8	
t_{11}	D31–D0, DP3–DP0 Write Data Float Delay		20		20	ns	9	Note 2
t_{12}	EADS# Setup Time	6		5		ns	5	
t_{13}	EADS# Hold Time	3		3		ns	5	
t_{14}	KEN#, BS16#, BS8# Setup Time	6		5		ns	5	
t_{15}	KEN#, BS16#, BS8# Hold Time	3		3		ns	5	
t_{16}	RDY#, BRDY# Setup Time	6		5		ns	6	
t_{17}	RDY#, BRDY# Hold Time	3		3		ns	6	

Table 21. AC Characteristics

T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified. (Sheet 2 of 2)

Symbol	Parameter	V_{CC} (Package)						
		3.3V (208-Lead SQFP)		5V (196-Lead PQFP)				
		Min	Max	Min	Max	Unit	Figure	Notes
t_{18}	HOLD, AHOLD Setup Time	6		6		ns	5	
t_{18a}	BOFF# Setup Time	9		8		ns	5	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		3		ns	5	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET Setup Time	6		5		ns	5	Note 3
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET Hold Time	3		3		ns	5	Note 3
t_{22}	D31–D0, DP3–DP0, A31–A4 Read Setup Time	6		5		ns	6 5	
t_{23}	D31–D0, DP3–DP0, A31–A4 Read Hold Time	3		3		ns	6 5	

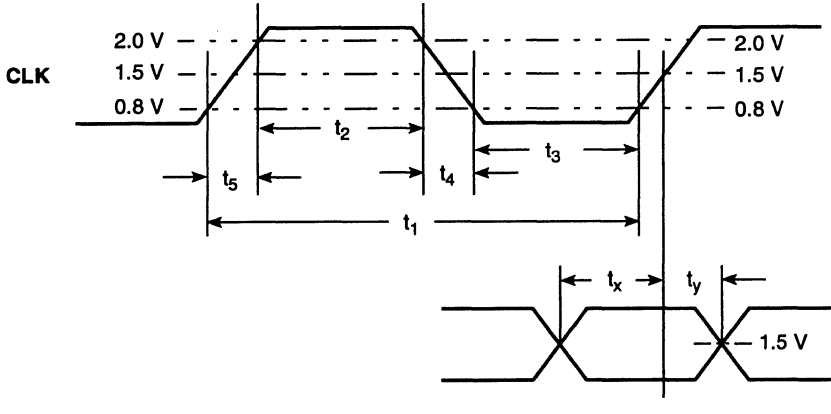
NOTES:
1. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant bus cycle protocol is used.
2. Not 100% tested, guaranteed by design characterization.
3. A reset pulse width of 15 CLK cycles is required for warm resets (RESET or SRESET). Power-up resets (cold resets) require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.

Table 22. AC Specifications for the Test Access Port

(Both 3.3V SQFP and 5V PQFP Processors)
T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{24}	TCK Frequency		8	MHz		Note 1
t_{25}	TCK Period	125		ns	10	
t_{26}	TCK High Time	40		ns	10	@ 2.0V
t_{27}	TCK Low Time	40		ns	10	@ 0.8V
t_{28}	TCK Rise Time		8	ns	10	Note 2
t_{29}	TCK Fall Time		8	ns	10	Note 2
t_{30}	TDI, TMS Setup Time	8		ns	11	Note 3
t_{31}	TDI, TMS Hold Time	10		ns	11	Note 3
t_{32}	TDO Valid Delay	3	30	ns	11	Note 3
t_{33}	TDO Float Delay		36	ns	11	Note 3
t_{34}	All Outputs (except TDO) Valid Delay	3	30	ns	11	Note 3
t_{35}	All Outputs (except TDO) Float Delay		36	ns	11	Note 3
t_{36}	All Inputs (except TDI, TMS, TCK) Setup Time	8		ns	11	Note 3
t_{37}	All Inputs (except TDI, TMS, TCK) Hold Time	10		ns	11	Note 3

NOTES:
1. TCK period ≤ CLK period.
2. Rise/Fall times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10-ns increase in TCK period.
3. Parameters t_{30}–t_{37} are measured from TCK.



t_x = input setup times
t_y = input hold times, output float, valid and hold times

Figure 4. CLK Waveform

4

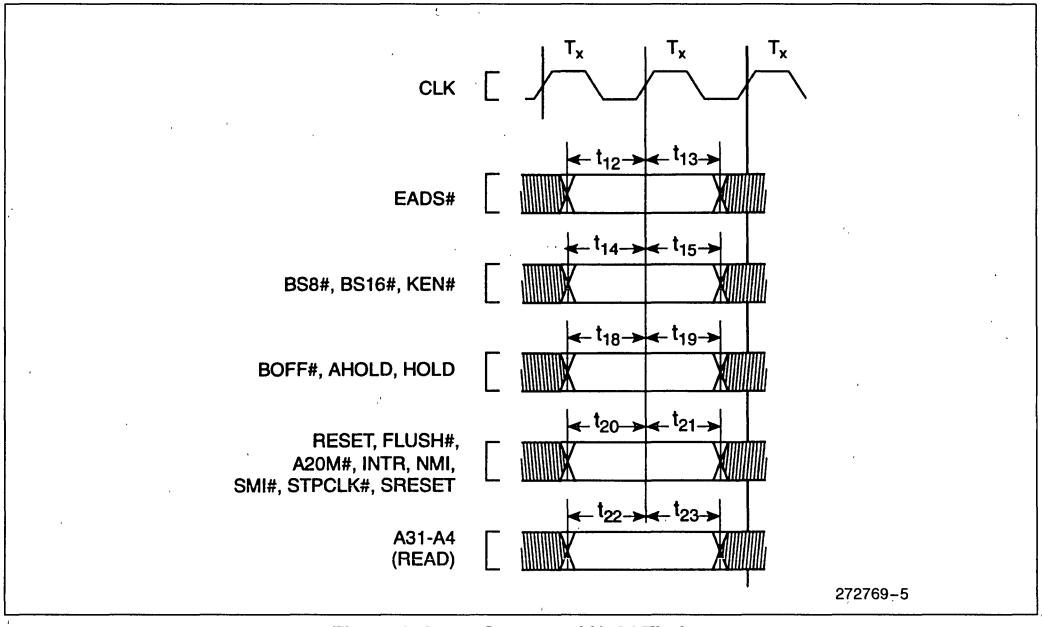


Figure 5. Input Setup and Hold Timing

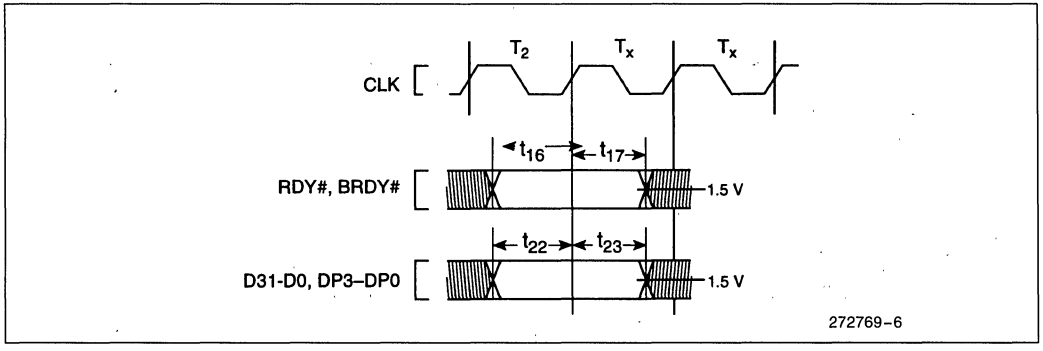


Figure 6. Input Setup and Hold Timing

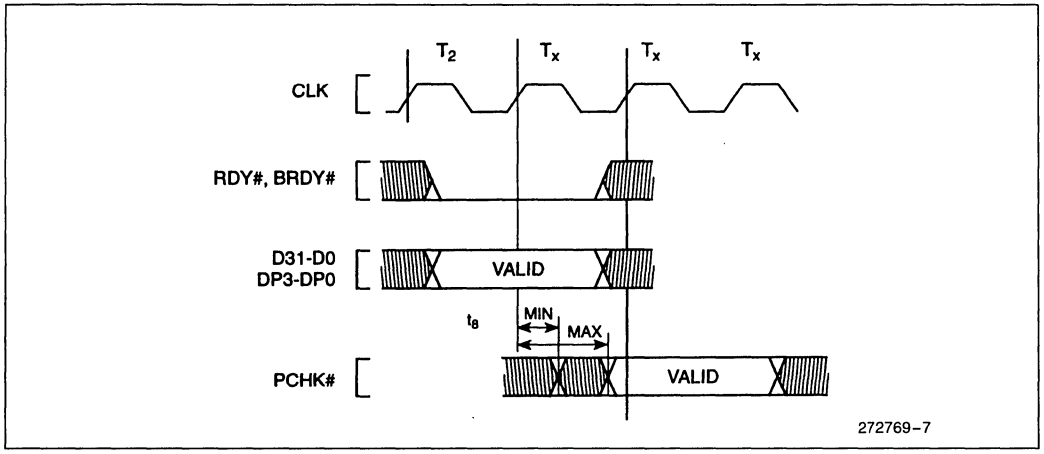


Figure 7. PCHK# Valid Delay Timing

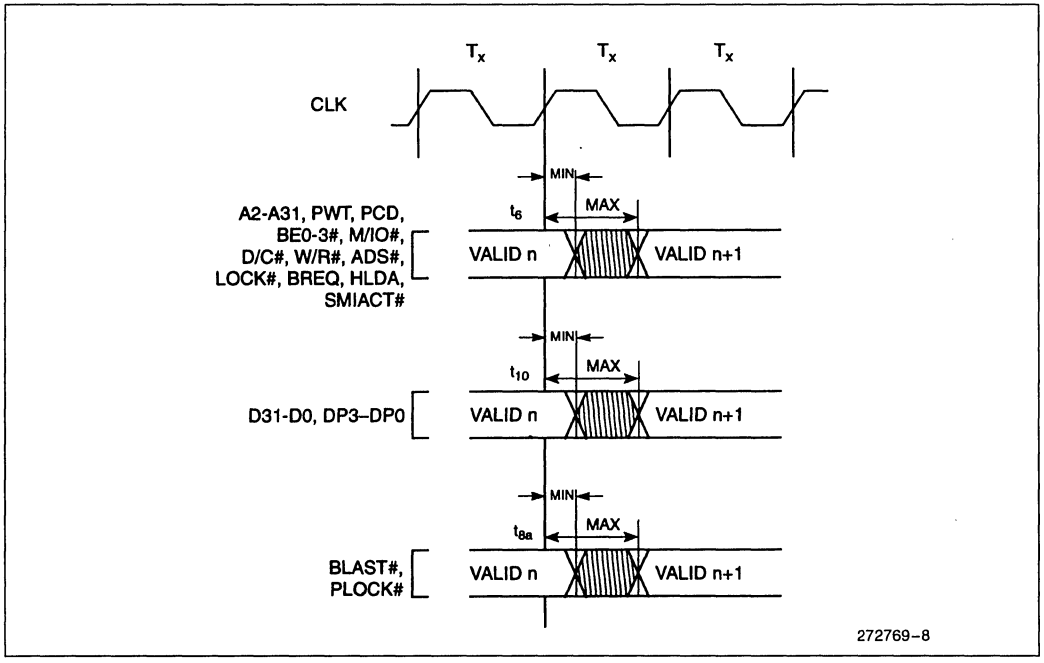


Figure 8. Output Valid Delay Timing

4

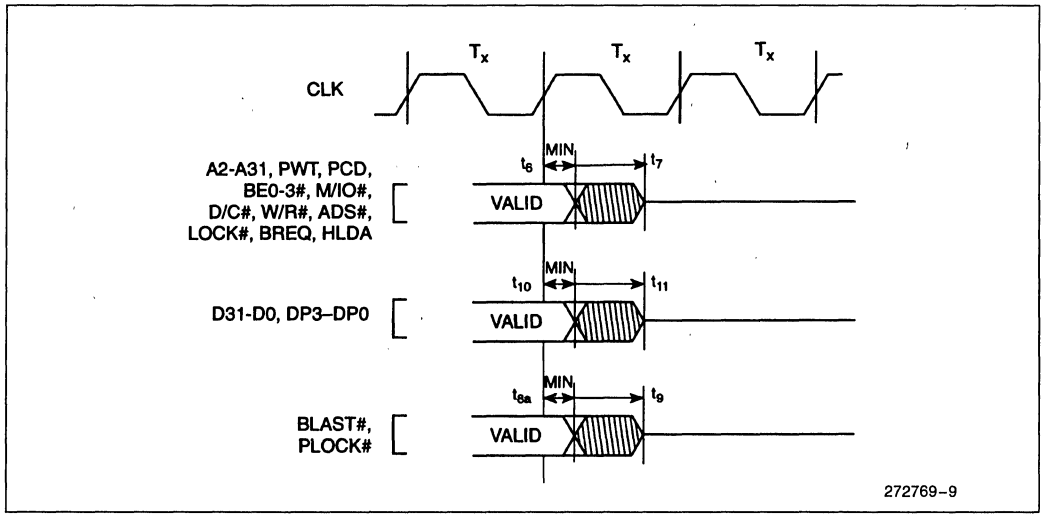


Figure 9. Maximum Float Delay Timing

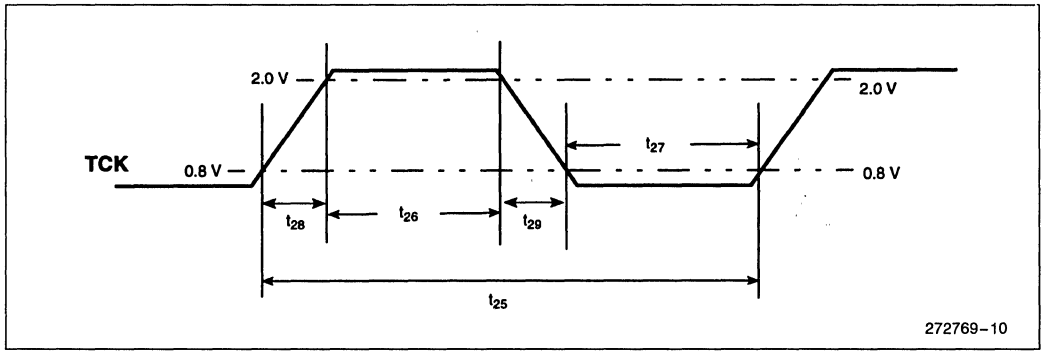


Figure 10. TCK Waveform

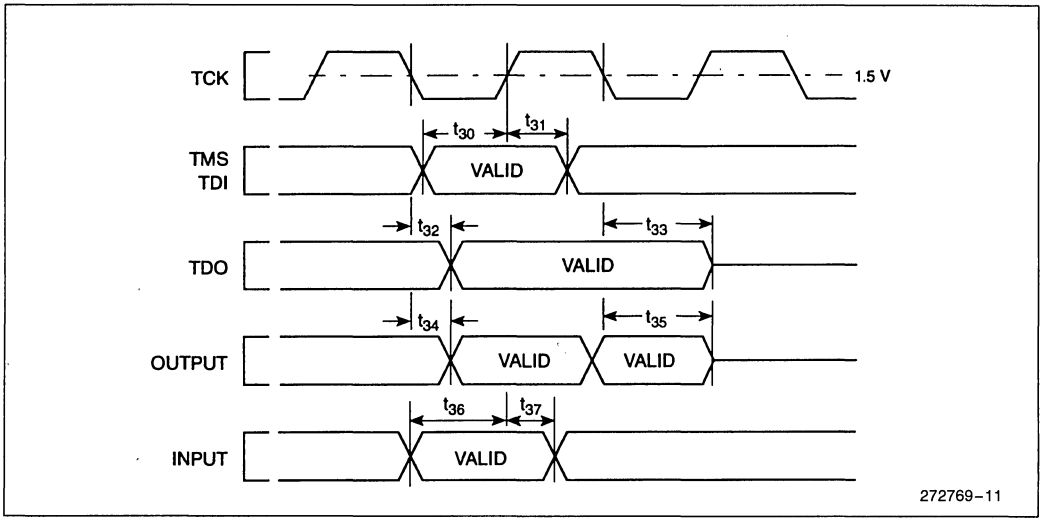
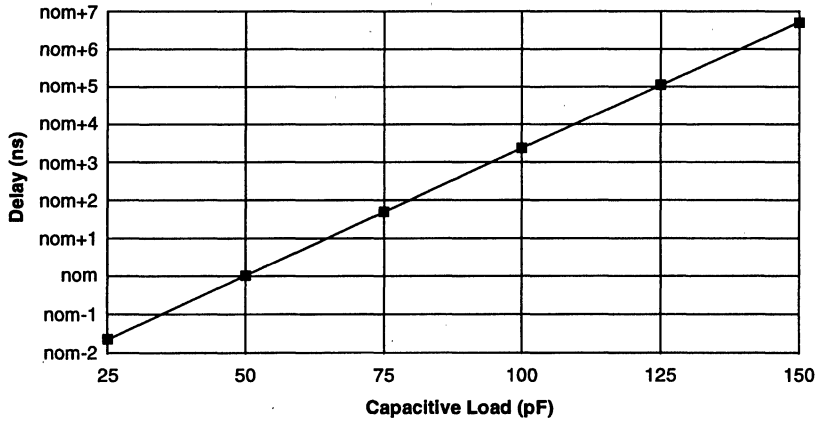


Figure 11. Test Signal Timing Diagram

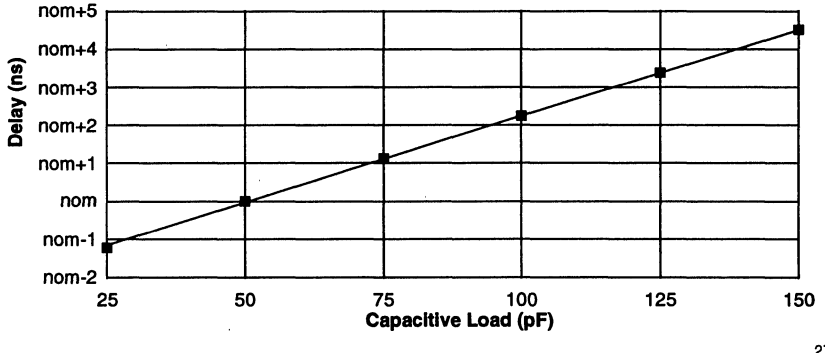
5.4 Capacitive Derating Curves

The following graphs are the capacitive derating curves for the embedded Intel486 SX processor.



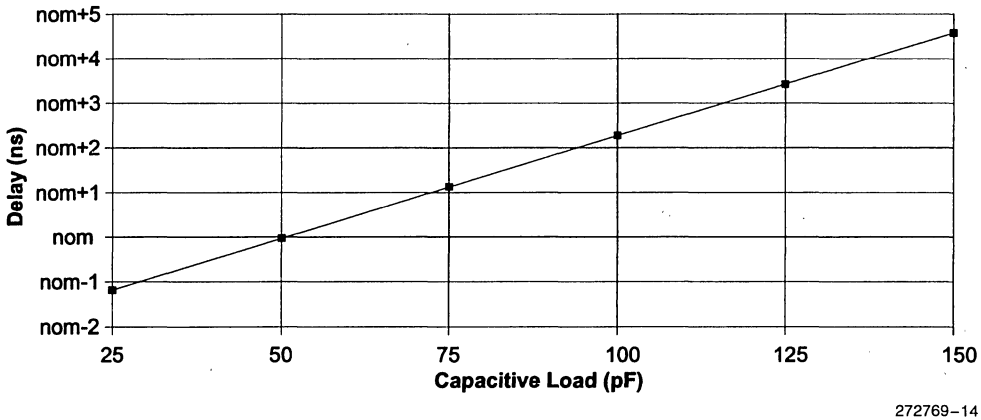
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 12. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition, 3.3V Processor



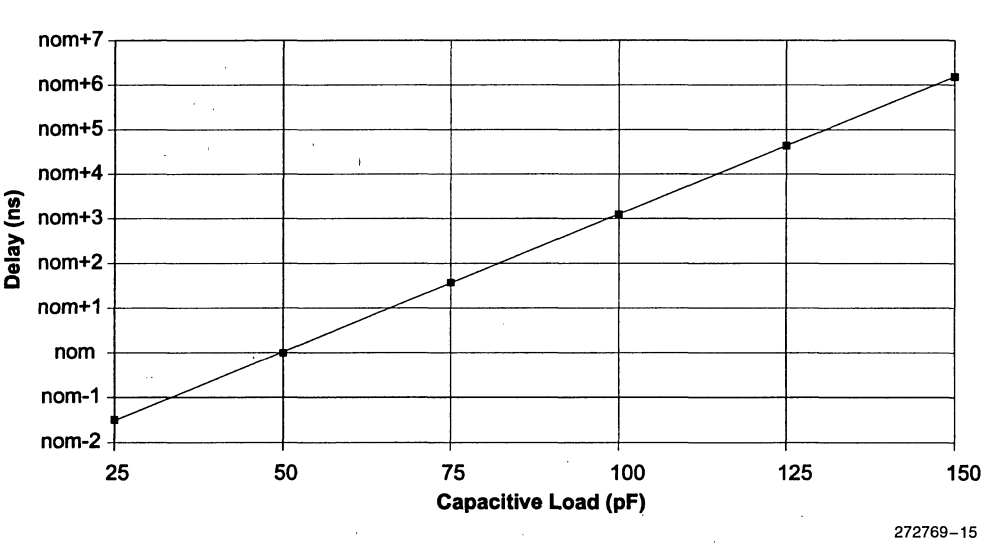
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 13. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition, 3.3V Processor



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 14. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition, 5V Processor



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 15. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition, 5V Processor

6.0 MECHANICAL DATA

This section describes the packaging dimensions and thermal specifications for the embedded Intel486 SX processor.

6.1 Package Dimensions

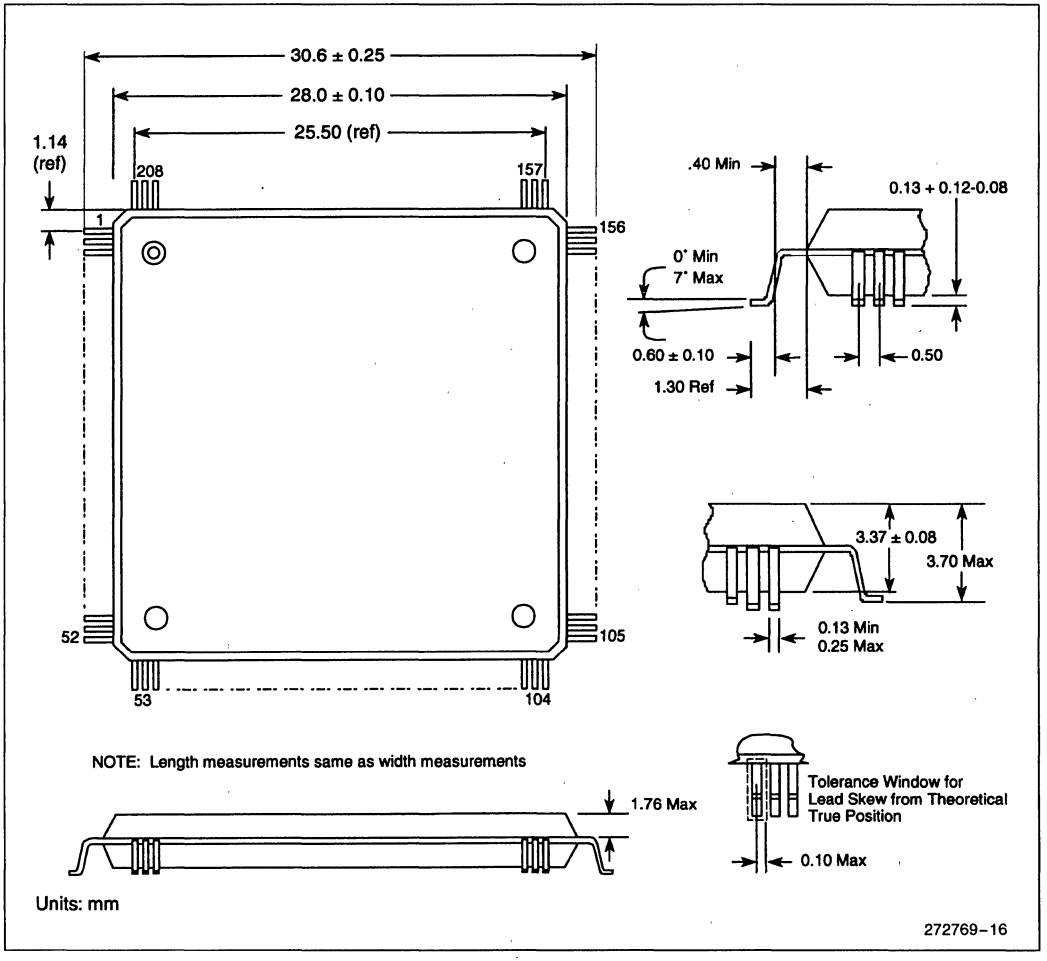
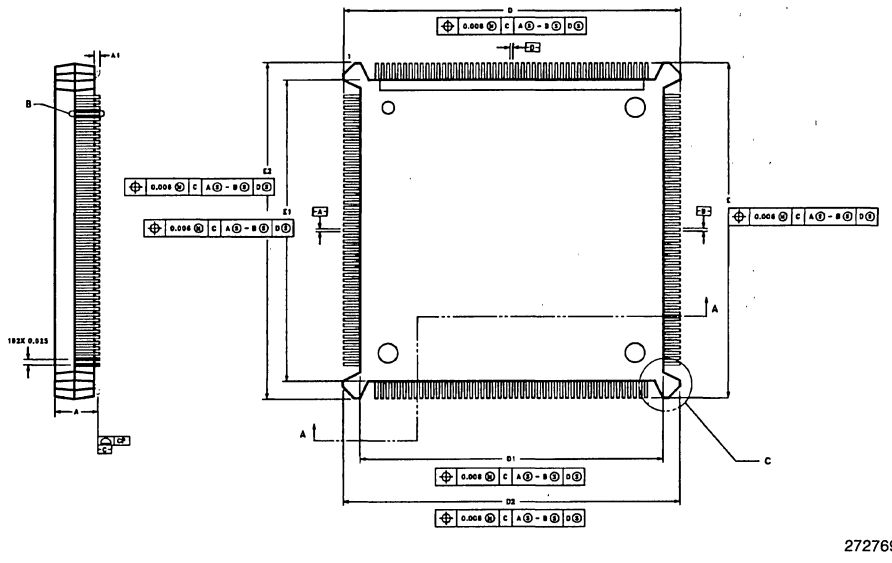


Figure 16. 208-Lead SQFP Package Dimensions



272769-17

NOTE:
Interpret dimensions and tolerances in accordance with ANSI Y14.5M-1982.

Figure 17. Principal Dimensions and Data for 196-Lead Plastic Quad Flat Pack Package

Table 23. Symbol List and Dimensions for 196-Lead PQFP Package

Symbol	Description of Dimensions	Min	Max
A	**Package Height:** Distance from the seating plane to the highest point of body.	0.160	0.175
A1	**Standoff:** The distance from the seating plane to the base plane.	0.020	0.035
D, E	**Overall Package Dimension:** Lead tip to lead tip.	1.470	1.485
D1, E1	Plastic Body Dimension	1.347	1.353
D2, E2	Bumper Distance Without FLASH With FLASH	 1.497 1.497	 1.503 1.510
CP	Seating Plane Coplanarity	0.000	0.004

NOTES:
1. All dimensions and tolerances conform to ANSI Y14.5M-1982.
2. Dimensions are in inches.

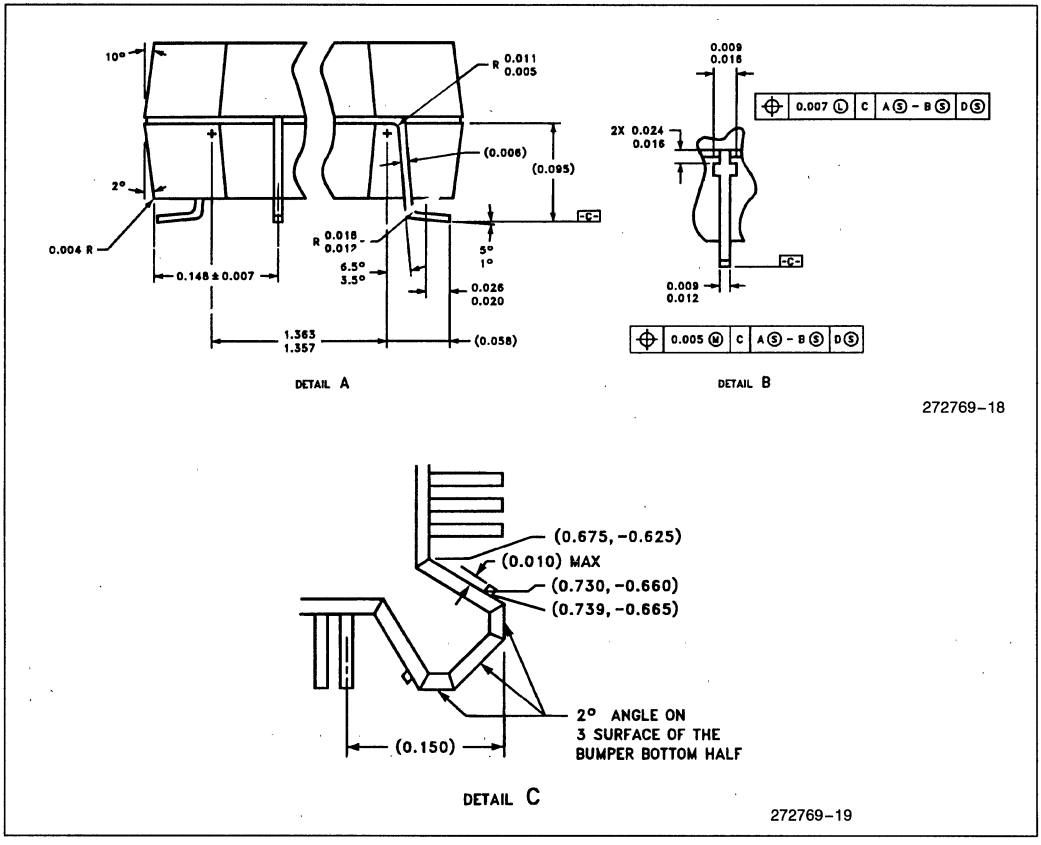


Figure 18. Typical Lead for 196-Lead PQFP Package

6.2 Package Thermal Specifications

The embedded Intel486 SX processor is specified for operation when the case temperature (T_C) is within the range of 0C to 85C. T_C may be measured in any environment to determine whether the processor is within the specified operating range.

The ambient temperature (T_A) can be calculated from θ_{JC} and θ_{JA} from the following equations:

$$T_J = T_C + P * \theta_{JC}$$
$$T_A = T_J - P * \theta_{JA}$$
$$T_C = T_A + P * [\theta_{JA} - \theta_{JC}]$$
$$T_A = T_C - P * [\theta_{JA} - \theta_{JC}]$$

Where T_J, T_A, T_C equals Junction, Ambient and Case Temperature respectively. θ_{JC}, θ_{JA} equals Junction-to-Case and Junction-to-Ambient thermal Resistance, respectively. P is defined as Maximum Power Consumption.

Values for θ_{JA} and θ_{JC} are given in the following tables for each product operating at 33 MHz. Maximum TA is shown for each product operating at 25 MHz and 33 MHz.

Table 24. Thermal Resistance, θ_{JA} (°C/W)

	θ_{JA} vs. Airflow—ft/min. (m/sec)			
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
208-Lead SQFP (3.3V) - Without Heat Sink	36.0	27.5	25.0	22.5
196-Lead PQFP (5V) - Without Heat Sink	20.5	16.5	14.0	12.5
196-Lead PQFP (5V) - With Heat Sink*	17.0	10.5	8.5	8.0

*0.350" high omnidirectional heat sink.

Table 25. Thermal Resistance, θ_{JC} (°C/W)

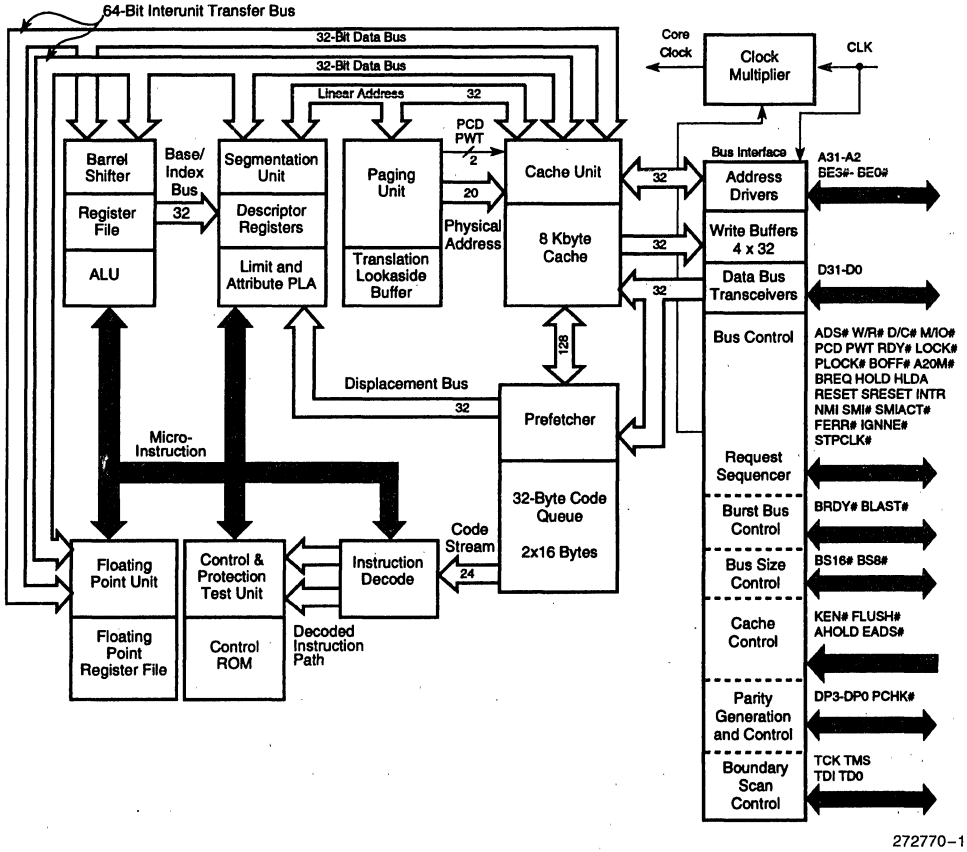
	θ_{JC} vs. Airflow—ft/min. (m/sec)			
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
	0	200	400	600
208-Lead SQFP (3.3V)	4.0	7.5	8.0	8.5
196-Lead PQFP (5V)	3.5	—	—	—

Table 26. Maximum $T_{ambient}$, T_A max (°C)

	Airflow—ft/min. (m/sec)				
	Freq. (MHz)	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
208-Lead SQFP (3.3V)	25	54	66	69	72
Without Heat Sink	33	47	62	65	69
196-Lead PQFP (5V)	25	40	50	57	61
Without Heat Sink	33	29	42	51	56
196-Lead PQFP (5V)	25	49	66	72	73
With Heat Sink	33	41	62	69	70

EMBEDDED IntelDX2™ PROCESSOR

- Integrated Floating-Point Unit
- Speed-Multiplying Technology
- 32-Bit RISC Technology Core
- 8-Kbyte Write-Through Cache
- Four Internal Write Buffers
- Burst Bus Cycles
- Dynamic Bus Sizing for 8- and 16-bit Data Bus Devices
- SL Technology
- Data Bus Parity Generation and Checking
- Boundary Scan (JTAG)
- 3.3V Processor, 50 MHz, 25 MHz CLK — 208-Lead Shrink Quad Flat Pack (SQFP)
- 5V Processor, 66 MHz, 33 MHz CLK — 168-Pin Pin Grid Array (PGA)
- Binary Compatible with Large Software Base



272770-1

Figure 1. Embedded IntelDX2™ Processor Block Diagram

December 1995
Order Number: 272770-001

EMBEDDED IntelDX2™ PROCESSOR

CONTENTS

FIGURES

CONTENTS

PAGE

TABLES

1.0 INTRODUCTION

The embedded IntelDX2™ processor provides high performance to 32-bit, embedded applications. Designed for applications that need a floating-point unit, the processor is ideal for embedded designs running DOS*, Microsoft* Windows*, OS/2*, or UNIX* applications written for the Intel architecture. Projects can be completed quickly by utilizing the wide range of software tools, utilities, assemblers and compilers that are available for desktop computer systems. Also, developers can find advantages in using existing chipsets and peripheral components in their embedded designs.

The embedded IntelDX2 processor is binary compatible with the Intel386™ and earlier Intel processors. Compared with the Intel386 processor, it provides faster execution of many commonly-used instructions. It also provides the benefits of an integrated, 8-Kbyte, write-through cache for code and data. Its data bus can operate in burst mode which provides up to 106-Mbyte-per-second transfers for cache-line fills and instruction prefetches.

Intel's SL technology is incorporated in the embedded IntelDX2 processor. Utilizing Intel's System Management Mode (SMM), it enables designers to develop energy-efficient systems.

Two component packages are available. A 168-pin Pin Grid Array (PGA) is available for 5V designs and a 208-Lead Shrink Quad Flat Pack (SQFP) is available for 3.3V designs.

The processor operates at twice the external-bus frequency. The 5V processor operates up to 66 MHz (33 MHz CLK). The 3.3V processor operates up to 50 MHz (25 MHz CLK).

1.1 Features

The embedded IntelDX2 processor offers these features:

- **32-bit RISC-Technology Core**—The embedded IntelDX2 processor performs a complete set of arithmetic and logical operations on 8-, 16-, and 32-bit data types using a full-width ALU and eight general purpose registers.
- **Single Cycle Execution**—Many instructions execute in a single clock cycle.
- **Instruction Pipelining**—Overlapped instruction fetching, decoding, address translation and execution.
- **On-Chip Floating-Point Unit**—Intel486™ processors support the 32-, 64-, and 80-bit formats specified in IEEE standard 754. The unit is binary compatible with the 8087, Intel287™, Intel387™ coprocessors, and Intel OverDrive® processor.
- **On-Chip Cache with Cache Consistency Support**—An 8-Kbyte, write-through, internal cache is used for both data and instructions. Cache hits provide zero wait-state access times for data within the cache. Bus activity is tracked to detect alterations in the memory represented by the internal cache. The internal cache can be invalidated or flushed so that an external cache controller can maintain cache consistency.
- **External Cache Control**—Write-back and flush controls for an external cache are provided so the processor can maintain cache consistency.
- **On-Chip Memory Management Unit**—Address management and memory space protection mechanisms maintain the integrity of memory in a multitasking and virtual memory environment. Both memory segmentation and paging are supported.
- **Burst Cycles**—Burst transfers allow a new double-word to be read from memory on each bus clock cycle. This capability is especially useful for instruction prefetch and for filling the internal cache.
- **Write Buffers**—The processor contains four write buffers to enhance the performance of consecutive writes to memory. The processor can continue internal operations after a write to these buffers, without waiting for the write to be completed on the external bus.
- **Bus Backoff**—When another bus master needs control of the bus during a processor initiated bus cycle, the embedded IntelDX2 processor floats its bus signals, then restarts the cycle when the bus becomes available again.
- **Instruction Restart**—Programs can continue execution following an exception generated by an unsuccessful attempt to access memory. This feature is important for supporting demand-paged virtual memory applications.

- **Dynamic Bus Sizing**—External controllers can dynamically alter the effective width of the data bus. Bus widths of 8, 16, or 32 bits can be used.
- **Boundary Scan (JTAG)**—Boundary Scan provides in-circuit testing of components on printed circuit boards. The Intel Boundary Scan implementation conforms with the IEEE Standard Test Access Port and Boundary Scan Architecture.

Intel's SL technology provides these features:

- **Intel System Management Mode (SMM)**—A unique Intel architecture operating mode provides a dedicated special purpose interrupt and address space that can be used to implement intelligent power management and other enhanced functions in a manner that is completely transparent to the operating system and applications software.
- **I/O Restart**—An I/O instruction interrupted by a System Management Interrupt (SMI#) can automatically be restarted following the execution of the RSM instruction.
- **Stop Clock**—The embedded IntelDX2 processor has a stop clock control mechanism that provides two low-power states: a Stop Grant state (20 mA–45 mA typical, depending on input clock frequency) and a Stop Clock state (~100 μA–200 μA typical, with input clock frequency of 0 MHz).
- **Auto HALT Power Down**—After the execution of a HALT instruction, the embedded IntelDX2 processor issues a normal Halt bus cycle and the clock input to the processor core is automatically stopped, causing the processor to enter the Auto HALT Power Down state (20 mA–45 mA typical, depending on input clock frequency).
- **Auto Idle Power Down**—This function allows the processor to reduce the core frequency to the bus frequency when both the core and bus are idle. Auto Idle Power Down is software transparent and does not affect processor performance. Auto Idle Power Down provides an average power savings of 10% and is only applicable to clock multiplied processors.

1.2 Family Members

Table 1 shows the embedded IntelDX2 processors and briefly describes their characteristics.

Table 1. The Embedded IntelDX2™ Processor Family

Product	Supply Voltage V_{CC}	Maximum Processor Frequency	Maximum External Bus Frequency	Package
SB80486DX2SC50	3.3V	50 MHz	25 MHz	208-Lead SQFP
A80486DX2SA66	5.0V	66 MHz	33 MHz	168-Pin PGA

2.0 HOW TO USE THIS DOCUMENT

For a complete set of documentation related to the embedded IntelDX2 processor, use this document in conjunction with the following reference documents:

- *Intel486™ Processor Family* datasheet—Order No. 242202
- *Intel486 Microprocessor Family Programmer's Reference Manual*—Order No. 240486
- Intel Application Note AP-485—*Intel Processor Identification with the CPUID Instruction*—Order No. 241618

The information in the reference documents for the IntelDX2 processor applies to the embedded IntelDX2 processor. Some of the IntelDX2 processor information is duplicated in this document to minimize the dependence on the reference documents.

3.0 PIN DESCRIPTIONS

3.1 Pin Assignments

The following figures and tables show the pin assignments of each package type for the embedded IntelDX2 processor. Tables are provided showing the pin differences between the embedded IntelDX2 processor and other embedded Intel486 processor products.

208-Lead SQFP—Quad Flat Pack

- Figure 2, Package Diagram for 208-Lead SQFP Embedded IntelDX2™ Processor (pg. 4)
- Table 2, Pinout Differences for 208-Lead SQFP Package (pg. 5)
- Table 3, Pin Assignment for 208-Lead SQFP Package (pg. 6)
- Table 4, Pin Cross Reference for 208-Lead SQFP Package (pg. 8)

168-Pin PGA - Pin Grid Array

- Figure 3, Package Diagram for 168-Pin PGA Embedded IntelDX2™ Processor (pg. 10)
- Table 5, Pinout Differences for 168-Pin PGA Package (pg. 11)
- Table 6, Pin Assignment for 168-Pin PGA Package (pg. 12)
- Table 7, Pin Cross Reference for 168-Pin PGA Package (pg. 14)

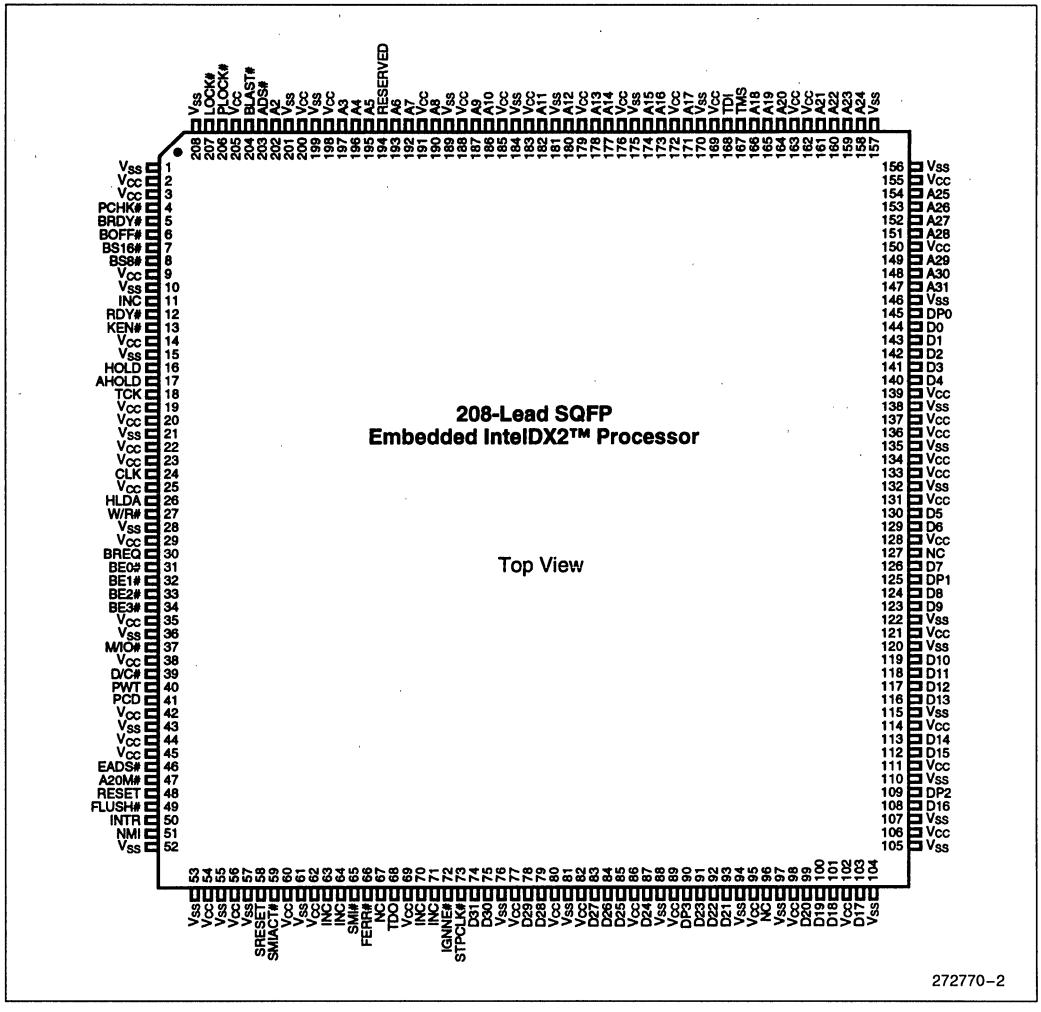


Figure 2. Package Diagram for 208-Lead SQFP Embedded IntelDX2™ Processor

Table 2. Pinout Differences for 208-Lead SQFP Package

Pin #	Embedded Intel486™ SX Processor	Embedded IntelDX2™ Processor	Embedded Write-Back Enhanced IntelDX4™ Processor
3	V_{CC1}	V_{CC}	V_{CC5}
11	INC_2	INC	CLKMUL
63	INC	INC	HITM#
64	INC	INC	WB/WT#
66	INC	FERR#	FERR#
70	INC	INC	CACHE#
71	INC	INC	INV
72	INC	IGNNE#	IGNNE#

NOTES:
1. This pin location is for the V_{CC5} pin on the embedded IntelDX4 processor. For compatibility with 3.3V processors that have 5V-tolerant input buffers (i.e., embedded IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane.
2. INC. Internal No Connect. These pins are not connected to any internal pad in the embedded IntelDX2 processor. However, new signals are defined for the location of the INC pins in the embedded IntelDX4 processor. One system design can accommodate any one of these processors provided the purpose of each INC pin is understood before it is used.

Table 3. Pin Assignment for 208-Lead SQFP Package (Sheet 1 of 2)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
1	V_{SS}	53	V_{SS}	105	V_{SS}	157	V_{SS}
2	V_{CC}	54	V_{CC}	106	V_{CC}	158	A24
3	V_{CC1}	55	V_{SS}	107	V_{SS}	159	A23
4	PCHK#	56	V_{CC}	108	D16	160	A22
5	BRDY#	57	V_{SS}	109	DP2	161	A21
6	BOFF#	58	SRESET	110	V_{SS}	162	V_{CC}
7	BS16#	59	SMIACT#	111	V_{CC}	163	V_{CC}
8	BS8#	60	V_{CC}	112	D15	164	A20
9	V_{CC}	61	V_{SS}	113	D14	165	A19
10	V_{SS}	62	V_{CC}	114	V_{CC}	166	A18
11	INC_2	63	INC_2	115	V_{SS}	167	TMS
12	RDY#	64	INC_2	116	D13	168	TDI
13	KEN#	65	SMI#	117	D12	169	V_{CC}
14	V_{CC}	66	FERR#	118	D11	170	V_{SS}
15	V_{SS}	67	NC^3	119	D10	171	A17
16	HOLD	68	TDO	120	V_{SS}	172	V_{CC}
17	AHOLD	69	V_{CC}	121	V_{CC}	173	A16
18	TCK	70	INC_2	122	V_{SS}	174	A15
19	V_{CC}	71	INC_2	123	D9	175	V_{SS}
20	V_{CC}	72	IGNNE#	124	D8	176	V_{CC}
21	V_{SS}	73	STPCLK#	125	DP1	177	A14
22	V_{CC}	74	D31	126	D7	178	A13
23	V_{CC}	75	D30	127	NC_3	179	V_{CC}
24	CLK	76	V_{SS}	128	V_{CC}	180	A12
25	V_{CC}	77	V_{CC}	129	D6	181	V_{SS}
26	HLDA	78	D29	130	D5	182	A11
27	W/R#	79	D28	131	V_{CC}	183	V_{CC}
28	V_{SS}	80	V_{CC}	132	V_{SS}	184	V_{SS}
29	V_{CC}	81	V_{SS}	133	V_{CC}	185	V_{CC}
30	BREQ	82	V_{CC}	134	V_{CC}	186	A10
31	BE0#	83	D27	135	V_{SS}	187	A9
32	BE1#	84	D26	136	V_{CC}	188	V_{CC}

Table 3. Pin Assignment for 208-Lead SQFP Package (Sheet 2 of 2)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
33	BE2#	85	D25	137	V_{CC}	189	V_{SS}
34	BE3#	86	V_{CC}	138	V_{SS}	190	A8
35	V_{CC}	87	D24	139	V_{CC}	191	V_{CC}
36	V_{SS}	88	V_{SS}	140	D4	192	A7
37	M/IO#	89	V_{CC}	141	D3	193	A6
38	V_{CC}	90	DP3	142	D2	194	RESERVED
39	D/C#	91	D23	143	D1	195	A5
40	PWT	92	D22	144	D0	196	A4
41	PCD	93	D21	145	DP0	197	A3
42	V_{CC}	94	V_{SS}	146	V_{SS}	198	V_{CC}
43	V_{SS}	95	V_{CC}	147	A31	199	V_{SS}
44	V_{CC}	96	NC_3	148	A30	200	V_{CC}
45	V_{CC}	97	V_{SS}	149	A29	201	V_{SS}
46	EADS#	98	V_{CC}	150	V_{CC}	202	A2
47	A20M#	99	D20	151	A28	203	ADS#
48	RESET	100	D19	152	A27	204	BLAST#
49	FLUSH#	101	D18	153	A26	205	V_{CC}
50	INTR	102	V_{CC}	154	A25	206	PLOCK#
51	NMI	103	D17	155	V_{CC}	207	LOCK#
52	V_{SS}	104	V_{SS}	156	V_{SS}	208	V_{SS}

NOTES:
1. This pin location is for the V_{CC5} pin on the embedded IntelDX4 processor. For compatibility with 3.3V processors that have 5V-tolerant input buffers (i.e., embedded IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane.
2. INC. Internal No Connect. These pins are not connected to any internal pad in the embedded IntelDX2 processors. However, signals are defined for the location of the INC pins in the IntelDX4 processor. One system design can accommodate any one of these processors provided the purpose of each INC pin is understood before it is used.
3. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC}, or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.

Table 4. Pin Cross Reference for 208-Lead SQFP Package (Sheet 1 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
A2	202	D0	144	A20M#	47	67	11	2	1
A3	197	D1	143	ADS#	203	96	63	3	10
A4	196	D2	142	AHOLD	17	127	64	9	15
A5	195	D3	141	BE0#	31		70	14	21
A6	193	D4	140	BE1#	32		71	19	28
A7	192	D5	130	BE2#	33			20	36
A8	190	D6	129	BE3#	34			22	43
A9	187	D7	126	BLAST#	204			23	52
A10	186	D8	124	BOFF#	6			25	53
A11	182	D9	123	BRDY#	5			29	55
A12	180	D10	119	BREQ	30			35	57
A13	178	D11	118	BS16#	7			38	61
A14	177	D12	117	BS8#	8			42	76
A15	174	D13	116	CLK	24			44	81
A16	173	D14	113	D/C#	39			45	88
A17	171	D15	112	DP0	145			54	94
A18	166	D16	108	DP1	125			56	97
A19	165	D17	103	DP2	109			60	104
A20	164	D18	101	DP3	90			62	105
A21	161	D19	100	EADS#	46			69	107
A22	160	D20	99	FERR#	66			77	110
A23	159	D21	93	FLUSH#	49			80	115
A24	158	D22	92	HLDA	26			82	120
A25	154	D23	91	HOLD	16			86	122
A26	153	D24	87	IGNNE#	72			89	132
A27	152	D25	85	INTR	50			95	135
A28	151	D26	84	KEN#	13			98	138
A29	149	D27	83	LOCK#	207			102	146
A30	148	D28	79	M/IO#	37			106	156
A31	147	D29	78	NMI	51			111	157
		D30	75	PCD	41			114	170
		D31	74	PCHK#	4			121	175
				PLOCK#	206			128	181
				PWT	40			131	184
				RDY#	12			133	189
				RESERVED	194			134	199
				RESET	48			136	201

Table 4. Pin Cross Reference for 208-Lead SQFP Package (Sheet 2 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
				SMI#	65			137	208
				SMIACT#	59			139	
				SRESET	58			150	
				STPCLK#	73			155	
				TCK	18			162	
				TDI	168			163	
				TDO	68			169	
				TMS	167			172	
				W/R#	27			176	
								179	
								183	
								185	
								188	
								191	
								198	
								200	
								205	

4

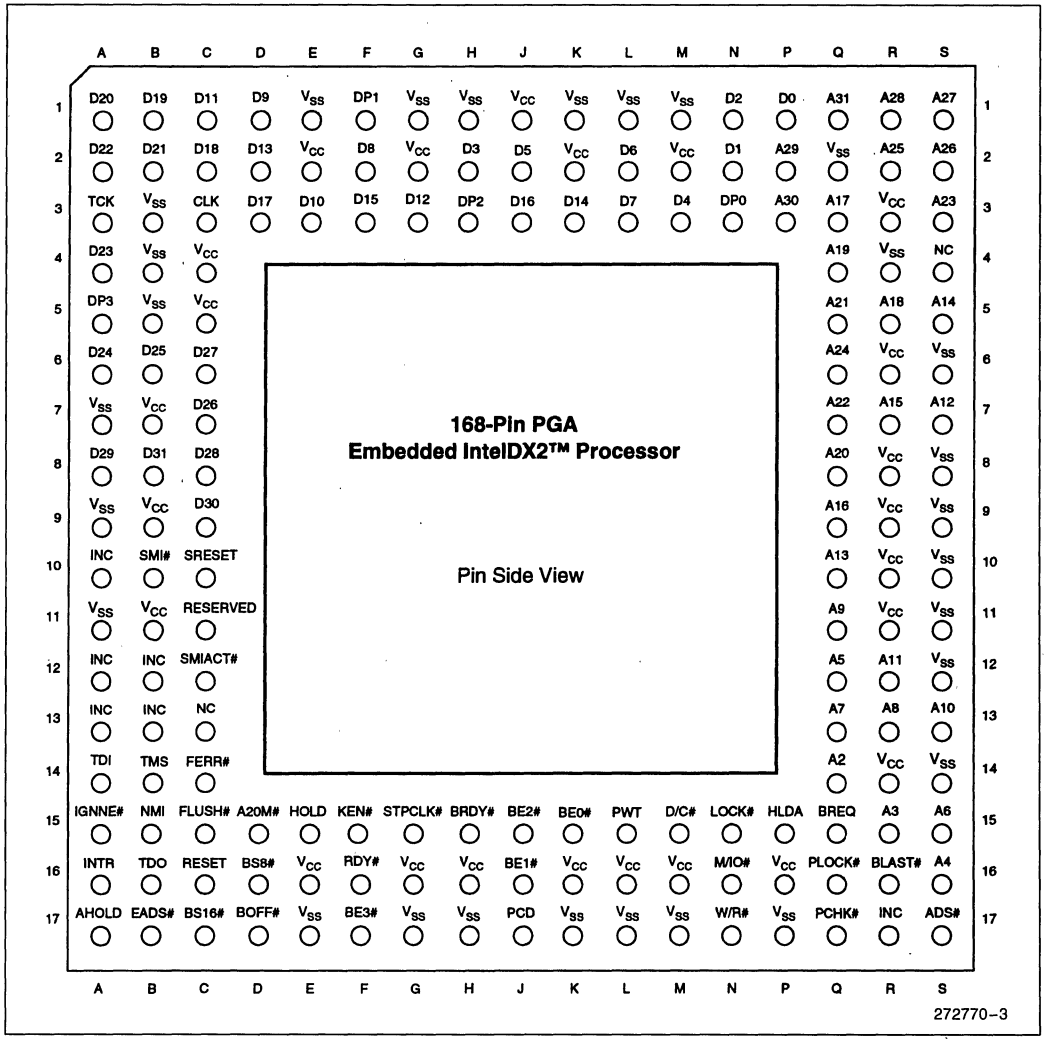


Figure 3. Package Diagram for 168-Pin PGA Embedded IntelDX2™ Processor

Table 5. Pinout Differences for 168-Pin PGA Package

Pin #	Embedded IntelDX2™ Processor	Embedded Write-Back Enhanced IntelDX4™ Processor
A10	INC	INV
A12	INC	HITM#
B12	INC	CACHE#
B13	INC	WB/WT#
J1	V_{CC}	V_{CC5}
R17	INC	CLKMUL
S4	NC	VOLDET

Table 6. Pin Assignment for 168-Pin PGA Package (Sheet 1 of 2)

Pin #	Description	Pin #	Description	Pin #	Description
A1	D20	D17	BOFF#	P2	A29
A2	D22	E1	V_{SS}	P3	A30
A3	TCK	E2	V_{CC}	P15	HLDA
A4	D23	E3	D10	P16	V_{CC}
A5	DP3	E15	HOLD	P17	V_{SS}
A6	D24	E16	V_{CC}	Q1	A31
A7	V_{SS}	E17	V_{SS}	Q2	V_{SS}
A8	D29	F1	DP1	Q3	A17
A9	V_{SS}	F2	D8	Q4	A19
A10	INC_1	F3	D15	Q5	A21
A11	V_{SS}	F15	KEN#	Q6	A24
A12	INC_1	F16	RDY#	Q7	A22
A13	INC_1	F17	BE3#	Q8	A20
A14	TDI	G1	V_{SS}	Q9	A16
A15	IGNNE#	G2	V_{CC}	Q10	A13
A16	INTR	G3	D12	Q11	A9
A17	AHOLD	G15	STPCLK#	Q12	A5
B1	D19	G16	V_{CC}	Q13	A7
B2	D21	G17	V_{SS}	Q14	A2
B3	V_{SS}	H1	V_{SS}	Q15	BREQ
B4	V_{SS}	H2	D3	Q16	PLOCK#
B5	V_{SS}	H3	DP2	Q17	PCHK#
B6	D25	H15	BRDY#	R1	A28
B7	V_{CC}	H16	V_{CC}	R2	A25
B8	D31	H17	V_{SS}	R3	V_{CC}
B9	V_{CC}	J1	V_{CC}	R4	V_{SS}
B10	SMI#	J2	D5	R5	A18
B11	V_{CC}	J3	D16	R6	V_{CC}
B12	INC_1	J15	BE2#	R7	A15
B13	INC_1	J16	BE1#	R8	V_{CC}
B14	TMS	J17	PCD	R9	V_{CC}

Table 6. Pin Assignment for 168-Pin PGA Package (Sheet 2 of 2)

Pin #	Description	Pin #	Description	Pin #	Description
B15	NMI	K1	V_{SS}	R10	V_{CC}
B16	TDO	K2	V_{CC}	R11	V_{CC}
B17	EADS#	K3	D14	R12	A11
C1	D11	K15	BE0#	R13	A8
C2	D18	K16	V_{CC}	R14	V_{CC}
C3	CLK	K17	V_{SS}	R15	A3
C4	V_{CC}	L1	V_{SS}	R16	BLAST#
C5	V_{CC}	L2	D6	R17	INC_1
C6	D27	L3	D7	S1	A27
C7	D26	L15	PWT	S2	A26
C8	D28	L16	V_{CC}	S3	A23
C9	D30	L17	V_{SS}	S4	NC_2
C10	SRESET	M1	V_{SS}	S5	A14
C11	RESERVED	M2	V_{CC}	S6	V_{SS}
C12	SMIACT#	M3	D4	S7	A12
C13	NC_2	M15	D/C#	S8	V_{SS}
C14	FERR#	M16	V_{CC}	S9	V_{SS}
C15	FLUSH#	M17	V_{SS}	S10	V_{SS}
C16	RESET	N1	D2	S11	V_{SS}
C17	BS16#	N2	D1	S12	V_{SS}
D1	D9	N3	DP0	S13	A10
D2	D13	N15	LOCK#	S14	V_{SS}
D3	D17	N16	M/IO#	S15	A6
D15	A20M#	N17	W/R#	S16	A4
D16	BS8#	P1	D0	S17	ADS#

NOTES:
1. INC. Internal No Connect. These pins are not connected to any internal pad in the embedded IntelDX2 processors. However, signals are defined for the location of the INC pins in the IntelDX4 processor. One system design can accommodate any one of these processors provided the purpose of each INC pin is understood before it is used.
2. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC}, or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.

Table 7. Pin Cross Reference for 168-Pin PGA Package (Sheet 1 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
A2	Q14	D0	P1	A20M#	D15	C13	A10	B7	A7
A3	R15	D1	N2	ADS#	S17	S4	A12	B9	A9
A4	S16	D2	N1	AHOLD	A17		A13	B11	A11
A5	Q12	D3	H2	BE0#	K15		B12	C4	B3
A6	S15	D4	M3	BE1#	J16		B13	C5	B4
A7	Q13	D5	J2	BE2#	J15		R17	E2	B5
A8	R13	D6	L2	BE3#	F17			E16	E1
A9	Q11	D7		BLAST#	R16L3			G2	E17
A10	S13	D8	F2	BOFF#	D17			G16	G1
A11	R12	D9	D1	BRDY#	H15			H16	G17
A12	S7	D10	E3	BREQ	Q15			J1	H1
A13	Q10	D11	C1	BS16#	C17			K2	H17
A14	S5	D12	G3	BS8#	D16			K16	K1
A15	R7	D13	D2	CLK	C3			L16	K17
A16	Q9	D14	K3	D/C#	M15			M2	L1
A17	Q3	D15	F3	DP0	N3			M16	L17
A18	R5	D16	J3	DP1	F1			P16	M1
A19	Q4	D17	D3	DP2	H3			R3	M17
A20	Q8	D18	C2	DP3	A5			R6	P17
A21	Q5	D19	B1	EADS#	B17			R8	Q2
A22	Q7	D20	A1	FERR#	C14			R9	R4
A23	S3	D21	B2	FLUSH#	C15			R10	S6
A24	Q6	D22	A2	HLDA	P15			R11	S8
A25	R2	D23	A4	HOLD	E15			R14	S9
A26	S2	D24	A6	IGNNE#	A15				S10
A27	S1	D25	B6	INTR	A16				S11
A28	R1	D26	C7	KEN#	F15				S12
A29	P2	D27	C6	LOCK#	N15				S14
A30	P3	D28	C8	M/IO#	N16				
A31	Q1	D29	A8	NMI	B15				
		D30	C9	PCD	J17				
		D31	B8	PCHK#	Q17				
				PLOCK#	Q16				
				PWT	L15				
				RDY#	F16				
				RESERVED	C11				
				RESET	C16				

Table 7. Pin Cross Reference for 168-Pin PGA Package (Sheet 2 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC}	V_{SS}
				SMI#	B10				
				SMIACT#	C12				
				SRESET	C10				
				STPCLK#	G15				
				TCK	A3				
				TDI	A14				
				TDO	B16				
				TMS	B14				
				W/R#	N17				

3.2 Pin Quick Reference

The following is a brief pin description. For detailed signal descriptions refer to "Signal Description" in section 9 of the *Intel486™ Processor Family datasheet.*

Table 8. Embedded IntelDX2™ Processor Pin Descriptions (Sheet 1 of 7)

Symbol	Type	Name and Function
CLK	I	**Clock** provides the fundamental timing and internal operating frequency for the embedded IntelDX2 processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 **A3–A2**	I/O O	**Address Lines** A31–A2, together with the byte enable signals, BE3#–BE0#, define the physical area of memory or input/output space accessed. Address lines A31–A4 are used to drive addresses into the embedded IntelDX2 processor to perform cache line invalidation. Input signals must meet setup and hold times t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.
BE3# **BE2#** **BE1#** **BE0#**	O O O O	**Byte Enable** signals indicate active bytes during read and write cycles. During the first cycle of a cache fill, the external system should assume that all byte enables are active. BE3#–BE0# are active LOW and are not driven during bus hold. BE3# applies to D31–D24 BE2# applies to D23–D16 BE1# applies to D15–D8 BE0# applies to D7–D0
DATA BUS		
D31–D0	I/O	**Data Lines.** D7–D0 define the least significant byte of the data bus; D31–D24 define the most significant byte of the data bus. These signals must meet setup and hold times t_{22} and t_{23} for proper operation on reads. These pins are driven during the second and subsequent clocks of write cycles.
DATA PARITY		
DP3–DP0	I/O	There is one **Data Parity** pin for each byte of the data bus. Data parity is generated on all write data cycles with the same timing as the data driven by the embedded IntelDX2 processor. Even parity information must be driven back into the processor on the data parity pins with the same timing as read information to ensure that the correct parity check status is indicated by the embedded IntelDX2 processor. The signals read on these pins do not affect program execution. Input signals must meet setup and hold times t_{22} and t_{23}. DP3–DP0 must be connected to V_{CC} through a pull-up resistor in systems that do not use parity. DP3–DP0 are active HIGH and are driven during the second and subsequent clocks of write cycles.
PCHK#	O	**Parity Status** is driven on the PCHK# pin the clock after ready for read operations. The parity status is for data sampled at the end of the previous clock. A parity error is indicated by PCHK# being LOW. Parity status is only checked for enabled bytes as indicated by the byte enable and bus size signals. PCHK# is valid only in the clock immediately after read data is returned to the processor. At all other times PCHK# is inactive (HIGH). PCHK# is never floated.

Table 8. Embedded IntelDX2™ Processor Pin Descriptions (Sheet 2 of 7)

Symbol	Type	Name and Function
BUS CYCLE DEFINITION		
M/IO# **D/C#** **W/R#**	O O O	**Memory/Input-Output, Data/Control** and **Write/Read** lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted. **M/IO# D/C# W/R# Bus Cycle Initiated** 0 0 0 Interrupt Acknowledge 0 0 1 HALT/Special Cycle (see details below) 0 1 0 I/O Read 0 1 1 I/O Write 1 0 0 Code Read 1 0 1 Reserved 1 1 0 Memory Read 1 1 1 Memory Write **HALT/Special Cycle** **Cycle Name** **BE3#–BE0#** **A4–A2** Shutdown 1110 000 HALT 1011 000 Stop Grant bus cycle 1011 100
LOCK#	O	**Bus Lock** indicates that the current bus cycle is locked. The embedded IntelDX2 processor does not allow a bus hold when LOCK# is asserted (address holds are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the last clock of the last locked bus cycle. The last locked cycle ends when Ready is returned. LOCK# is active LOW and not driven during bus hold. Locked read cycles are not transformed into cache fill cycles when KEN# is returned active.
PLOCK#	O	**Pseudo-Lock** indicates that the current bus transaction requires more than one bus cycle to complete. For the embedded IntelDX2 processor, examples of such operations are segment table descriptor reads (64 bits) and cache line fills (128 bits). For Intel486 processors with on-chip Floating-Point Unit, floating-point long reads and writes (64 bits) also require more than one bus cycle to complete. The embedded IntelDX2 processor drives PLOCK# active until the addresses for the last bus cycle of the transaction are driven, regardless of whether RDY# or BRDY# have been returned. Normally PLOCK# and BLAST# are inverse of each other. However, during the first bus cycle of a 64-bit floating-point write (for Intel486 processors with on-chip Floating-Point Unit) both PLOCK# and BLAST# are asserted. PLOCK# is a function of the BS8#, BS16# and KEN# inputs. PLOCK# should be sampled only in the clock in which Ready is returned. PLOCK# is active LOW and is not driven during bus hold.
BUS CONTROL		
ADS#	O	**Address Status** output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus. ADS# is driven active in the same clock in which the addresses are driven. ADS# is active LOW and not driven during bus hold.

Table 8. Embedded IntelDX2™ Processor Pin Descriptions (Sheet 3 of 7)

Symbol	Type	Name and Function
BUS CONTROL (Continued)		
RDY#	I	**Non-burst Ready** input indicates that the current bus cycle is complete. RDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted data from the embedded IntelDX2 processor in response to a write. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. RDY# is active during address hold. Data can be returned to the embedded IntelDX2 processor while AHOLD is active. RDY# is active LOW and is not provided with an internal pull-up resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.
BURST CONTROL		
BRDY#	I	**Burst Ready** input performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data in response to a read or that the external system has accepted data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle. BRDY# is sampled in the second and subsequent clocks of a burst cycle. Data presented on the data bus is strobed into the embedded IntelDX2 processor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. BRDY# is active LOW and is provided with a small pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.
BLAST#	O	**Burst Last** signal indicates that the next time BRDY# is returned, the burst bus cycle is complete. BLAST# is active for both burst and non-burst bus cycles. BLAST# is active LOW and is not driven during bus hold.
INTERRUPTS		
RESET	I	**Reset** input forces the embedded IntelDX2 processor to begin execution at a known state. The processor cannot begin executing instructions until at least 1 ms after V_{CC}, and CLK have reached their proper DC and AC specifications. The RESET pin must remain active during this time to ensure proper processor operation. However, for warm resets, RESET should remain active for at least 15 CLK periods. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
INTR	I	**Maskable Interrupt** indicates that an external interrupt has been generated. When the internal interrupt flag is set in EFLAGS, active interrupt processing is initiated. The embedded IntelDX2 processor generates two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the interrupt acknowledges have been performed to ensure processor recognition of the interrupt. INTR is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

Table 8. Embedded IntelDX2™ Processor Pin Descriptions (Sheet 4 of 7)

Symbol	Type	Name and Function
INTERRUPTS (Continued)		
NMI	I	**Non-Maskable Interrupt** request signal indicates that an external non-maskable interrupt has been generated. NMI is rising-edge sensitive and must be held LOW for at least four CLK periods before this rising edge. NMI is not provided with an internal pull-down resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SRESET	I	**Soft Reset** pin duplicates all functionality of the RESET pin except that the SMBASE register retains its previous value. For soft resets, SRESET must remain active for at least 15 CLK periods. SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SMI#	I	**System Management Interrupt** input invokes System Management Mode (SMM). SMI# is a falling-edge triggered signal which forces the embedded IntelDX2 processor into SMM at the completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# does not break LOCKed bus cycles and cannot interrupt a currently executing SMM. The embedded IntelDX2 processor latches the falling edge of one pending SMI# signal while it is executing an existing SMI#. The nested SMI# is not recognized until after the execution of a Resume (RSM) instruction.
SMIACT#	O	**System Management Interrupt Active,** an active LOW output, indicates that the embedded IntelDX2 processor is operating in SMM. It is asserted when the processor begins to execute the SMI# state save sequence and remains active LOW until the processor executes the last state restore cycle out of SMRAM.
STPCLK#	I	**Stop Clock Request** input signal indicates a request was made to turn off or change the CLK input frequency. When the embedded IntelDX2 processor recognizes a STPCLK#, it stops execution on the next instruction boundary (unless superseded by a higher priority interrupt), empties all internal pipelines and write buffers, and generates a Stop Grant bus cycle. STPCLK# is active LOW. **STPCLK# is an asynchronous signal, but must remain active until the embedded IntelDX2 processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle.**
BUS ARBITRATION		
BREQ	O	**Bus Request** signal indicates that the embedded IntelDX2 processor has internally generated a bus request. BREQ is generated whether or not the processor is driving the bus. BREQ is active HIGH and is never floated.
HOLD	I	**Bus Hold Request** allows another bus master complete control of the embedded IntelDX2 processor bus. In response to HOLD going active, the processor floats most of its output and input/output pins. HLDA is asserted after completing the current bus cycle, burst cycle or sequence of locked cycles. The embedded IntelDX2 processor remains in this state until HOLD is de-asserted. HOLD is active HIGH and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper operation.

Table 8. Embedded IntelDX2™ Processor Pin Descriptions (Sheet 5 of 7)

Symbol	Type	Name and Function
BUS ARBITRATION (Continued)		
HLDA	O	**Hold Acknowledge** goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the embedded IntelDX2 processor has given the bus to another local bus master. HLDA is driven active in the same clock that the processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.
BOFF#	I	**Backoff** input forces the embedded IntelDX2 processor to float its bus in the next clock. The processor floats all pins normally floated during bus hold but HLDA is not asserted in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#; if both are returned in the same clock, BOFF# takes effect. The embedded IntelDX2 processor remains in bus hold until BOFF# is negated. If a bus cycle is in progress when BOFF# is asserted the cycle is restarted. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper operation.
CACHE INVALIDATION		
AHOLD	I	**Address Hold** request allows another bus master access to the embedded IntelDX2 processor's address bus for a cache invalidation cycle. The processor stops driving its address bus in the clock following AHOLD going active. Only the address bus is floated during address hold, the remainder of the bus remains active. AHOLD is active HIGH and is provided with a small internal pull-down resistor. For proper operation, AHOLD must meet setup and hold times t_{18} and t_{19}.
EADS#	I	**External Address**—This signal indicates that a *valid* external address has been driven onto the embedded IntelDX2 processor address pins. This address is used to perform an internal cache invalidation cycle. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy setup and hold times t_{12} and t_{13} for proper operation.
CACHE CONTROL		
KEN#	I	**Cache Enable** pin is used to determine whether the current cycle is cacheable. When the embedded IntelDX2 processor generates a cycle that can be cached and KEN# is active one clock before RDY# or BRDY# during the first transfer of the cycle, the cycle becomes a cache line fill cycle. Returning KEN# active one clock before RDY# during the last read in the cache line fill causes the line to be placed in the on-chip cache. KEN# is active LOW and is provided with a small internal pull-up resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	**Cache Flush** input forces the embedded IntelDX2 processor to flush its entire internal cache. FLUSH# is active LOW and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition in any specific clock.
PAGE CACHEABILITY		
PWT **PCD**	O O	**Page Write-Through** and **Page Cache Disable** pins reflect the state of the page attribute bits, PWT and PCD, in the page table entry, page directory entry or control register 3 (CR3) when paging is enabled. When paging is disabled, the embedded IntelDX2 processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT pins. PWT and PCD have the same timing as the cycle definition pins (M/IO#, D/C#, and W/R#). PWT and PCD are active HIGH and are not driven during bus hold. PCD is masked by the cache disable bit (CD) in Control Register 0.

Table 8. Embedded IntelDX2™ Processor Pin Descriptions (Sheet 6 of 7)

Symbol	Type	Name and Function
BUS SIZE CONTROL		
BS16# **BS8#**	I I	**Bus Size 16** and **Bus Size 8** pins (bus sizing pins) cause the embedded IntelDX2 processor to run multiple bus cycles to complete a request from devices that cannot provide or accept 32 bits of data in a single cycle. The bus sizing pins are sampled every clock. The processor uses the state of these pins in the clock before Ready to determine bus size. These signals are active LOW and are provided with internal pull-up resistors. These inputs must satisfy setup and hold times t_{14} and t_{15} for proper operation.
ADDRESS MASK		
A20M#	I	**Address Bit 20 Mask** pin, when asserted, causes the embedded IntelDX2 processor to mask physical address bit 20 (A20) before performing a lookup to the internal cache or driving a memory cycle on the bus. A20M# emulates the address wraparound at 1 Mbyte, which occurs on the 8086 processor. A20M# is active LOW and should be asserted only when the embedded IntelDX2 processor is in real mode. This pin is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For proper operation, A20M# should be sampled HIGH at the falling edge of RESET.
TEST ACCESS PORT		
TCK	I	**Test Clock,** an input to the embedded IntelDX2 processor, provides the clocking function required by the JTAG Boundary scan feature. TCK is used to clock state information (via TMS) and data (via TDI) into the component on the rising edge of TCK. Data is clocked out of the component (via TDO) on the falling edge of TCK. TCK is provided with an internal pull-up resistor.
TDI	I	**Test Data Input** is the serial input used to shift JTAG instructions and data into the processor. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and SHIFT-DR Test Access Port (TAP) controller states. During all other TAP controller states, TDI is a "don't care." TDI is provided with an internal pull-up resistor.
TDO	O	**Test Data Output** is the serial output used to shift JTAG instructions and data out of the component. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state.
TMS	I	**Test Mode Select** is decoded by the JTAG TAP to select test logic operation. TMS is sampled on the rising edge of TCK. To guarantee deterministic behavior of the TAP controller, TMS is provided with an internal pull-up resistor.

Table 8. Embedded IntelDX2™ Processor Pin Descriptions (Sheet 7 of 7)

Symbol	Type	Name and Function
NUMERIC ERROR REPORTING		
FERR#	O	**The Floating Point Error** pin is driven active when a floating point error occurs. FERR# is similar to the ERROR# pin on the Intel387™ Math CoProcessor. FERR# is included for compatibility with systems using DOS type floating point error reporting. FERR# will not go active if FP errors are masked in FPU register. FERR# is active LOW, and is not floated during bus hold.
IGNNE#	I	**When the Ignore Numeric Error** pin is asserted the processor will ignore a numeric error and continue executing non-control floating point instructions, but FERR# will still be activated by the processor. When IGNNE# is de-asserted the processor will freeze on a non-control floating point instruction, if a previous floating point instruction caused an error. IGNNE# has no effect when the NE bit in control register 0 is set. IGNNE# is active LOW and is provided with a small internal pull-up resistor. IGNNE# is asynchronous but setup and hold times t_{20} and t_{21} must be met to ensure recognition on any specific clock.
RESERVED PINS		
RESERVED	I	**Reserved** is reserved for future use. This pin MUST be connected to an external pull-up resistor circuit. The recommended resistor value is 10 KΩ. The pull-up resistor must be connected only to the RESERVED pin. **Do not share this resistor with other pins requiring pull-ups.**

Table 9. Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States
BREQ	HIGH			Previous State
HLDA	HIGH			As per HOLD
BE3#–BE0#	LOW		*	Previous State
PWT, PCD	HIGH		*	Previous State
W/R#, M/IO#, D/C#	HIGH/LOW		*	Previous State
LOCK#	LOW		*	HIGH (inactive)
PLOCK#	LOW		*	HIGH (inactive)
ADS#	LOW		*	HIGH (inactive)
BLAST#	LOW		*	Previous State
PCHK#	LOW			Previous State
FERR#	LOW			Previous State
A3–A2	HIGH	*	*	Previous State
SMIACT#	LOW			Previous State

NOTE:
The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

4

Table 10. Input/Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States
D31–D0	HIGH		*	Floated
DP3–DP0	HIGH		*	Floated
A31–A4	HIGH	*	*	Previous State

NOTE:
The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

Table 11. Test Pins

Name	Input or Output	Sampled/Driven On
TCK	Input	N/A
TDI	Input	Rising Edge of TCK
TDO	Output	Failing Edge of TCK
TMS	Input	Rising Edge of TCK

Table 12. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Pull-Up/ Pull-Down
CLK			
RESET	HIGH	Asynchronous	
SRESET	HIGH	Asynchronous	Pull-Down
HOLD	HIGH	Synchronous	
AHOLD	HIGH	Synchronous	Pull-Down
EADS#	LOW	Synchronous	Pull-Up
BOFF#	LOW	Synchronous	Pull-Up
FLUSH#	LOW	Asynchronous	Pull-Up
A20M#	LOW	Asynchronous	Pull-Up
BS16#, BS8#	LOW	Synchronous	Pull-Up
KEN#	LOW	Synchronous	Pull-Up
RDY#	LOW	Synchronous	
BRDY#	LOW	Synchronous	Pull-Up
INTR	HIGH	Asynchronous	
NMI	HIGH	Asynchronous	
IGNNE#	LOW	Asynchronous	Pull-Up
RESERVED			
SMI#	LOW	Asynchronous	Pull-Up
STPCLK#	LOW	Asynchronous	Pull-Up(1)
TCK	HIGH		Pull-Up
TDI	HIGH		Pull-Up
TMS	HIGH		Pull-Up

NOTE:
1. Though STPCLK# has an internal pull-up resistor, an external 10-KΩ pull-up resistor is needed if the STPCLK# pin is unused.

4.0 ARCHITECTURAL AND FUNCTIONAL OVERVIEW

The embedded IntelDX2 processor architecture is essentially the same as the IntelDX2 processor. Refer to the *Intel486™ Processor Family* datasheet (242202) for a description of the IntelDX2 processor. With some minor exceptions, the following datasheet sections apply to the embedded IntelDX2 processor:

- Architectural Overview
- Real Mode Architecture
- Protected Mode Architecture
- On-Chip Cache
- System Management Mode (SMM) Architectures
- Hardware Interface
- Bus Operation
- Testability
- Debugging Support
- Instruction Set Summary
- Differences Between Intel486 Processors and Intel386™ Processors

Exceptions to these sections of the datasheet are:

- References to the Upgrade Power Down Mode do not apply. The embedded IntelDX2 processor does not have the UP# signal pin and does not support the Intel OverDrive® processor.
- The embedded IntelDX2 processor has one pin reserved for possible future use. This pin, an input signal, is called RESERVED and must be connected to a 10-kΩ pull-up resistor. The pull-up resistor must be connected only to the RESERVED pin. **Do not share this resistor with other pins requiring pull-ups.**

4.1 CPUID Instruction

The embedded IntelDX2 processor supports the CPUID instruction (see Table 13). Because not all Intel processors support the CPUID instruction, a simple test can determine if the instruction is supported. The test involves the processor's ID Flag, which is bit 21 of the EFLAGS register. If software can change the value of this flag, the CPUID instruction is available. The actual state of the ID Flag bit is irrelevant and provides no significance to the hardware. This bit is cleared (reset to zero) upon device reset (RESET or SRESET) for compatibility with Intel486 processor designs that do not support the CPUID instruction.

CPUID-instruction details are provided here for the embedded IntelDX2 processor. Refer to Intel Application Note AP-485 *Intel Processor Identification with the CPUID Instruction* (Order No. 241618) for a description that covers all aspects of the CPUID instruction and how it pertains to other Intel processors.

4.1.1 OPERATION OF THE CPUID INSTRUCTION

The CPUID instruction requires the software developer to pass an input parameter to the processor in the EAX register. The processor response is returned in registers EAX, EBX, EDX, and ECX.

Table 13. CPUID Instruction Description

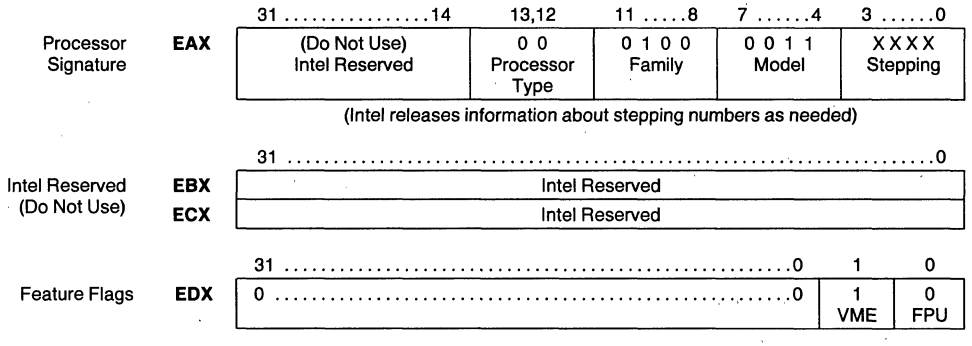
OP CODE	Instruction	Processor Core Clocks	Parameter passed in EAX (Input Value)	Description
0F A2	CPUID	9	0	Vendor (Intel) ID String
		14	1	Processor Identification
		9	>1	Undefined (Do Not Use)

Vendor ID String—When the parameter passed in EAX is 0 (zero), the register values returned upon instruction execution are shown in the following table.

		31.....24	23.....16	15.....8	7.....0
High Value (=1)	EAX	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1
Vendor ID String (ASCII Characters)	EBX	u (75)	n (6E)	e (65)	G (47)
	EDX	I (49)	e (65)	n (6E)	i (69)
	ECX	l (6C)	e (65)	t (74)	n (6E)

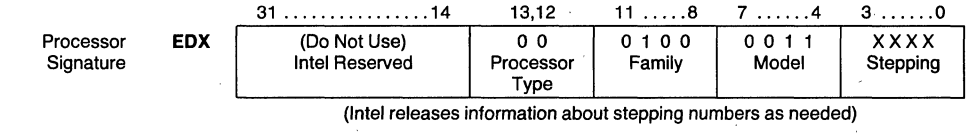
The values in EBX, EDX and ECX indicate an Intel processor. When taken in the proper order, they decode to the string "GenuineIntel."

Processor Identification—When the parameter passed to EAX is 1 (one), the register values returned upon instruction execution are:



4.2 Identification After Reset

Processor Identification—Upon reset, the EDX register contains the processor signature:



4.3 Boundary Scan (JTAG)

4.3.1 DEVICE IDENTIFICATION

Tables 14 and 15 show the 32-bit code for the embedded IntelDX2 processor. This code is loaded into the Device Identification Register.

Table 14. Boundary Scan Component Identification Code (3.3V Processor)

Version	Part Number				Mfg ID	1
	V_{CC} 0 = 5V 1 = 3.3V	Intel Architecture Type	Family 0100 = Intel486 CPU Family	Model 00101 = embedded IntelDX2 processor	009H = Intel	
31 ..28	27	26 21	20 17	16 12	11 1	0
X X X X	1	000001	0100	00101	00000001001	1

(Intel releases information about version numbers as needed)
Boundary Scan Component Identification Code = x828 5013 (Hex)

Table 15. Boundary Scan Component Identification Code (5V Processor)

Version	Part Number				Mfg ID 009H = Intel	1
	V_{CC} 0 = 5V 1 = 3.3V	Intel Architecture Type	Family 0100 = Intel486 CPU Family	Model 00101 = embedded IntelDX2 processor		
31 ..28	27	26 21	20 ..17	16 12	11 1	0
XXXX	0	000001	0100	00101	00000001001	1

(Intel releases information about version numbers as needed)
Boundary Scan Component Identification Code = x028 5013 (Hex)

4.3.2 BOUNDARY SCAN REGISTER BITS AND BIT ORDER

The boundary scan register contains a cell for each pin as well as cells for control of bidirectional and three-state pins. There are "Reserved" bits which correspond to no-connect (N/C) signals of the embedded IntelDX2 processor. Control registers WRCTL, ABUSCTL, BUSCTL, and MISCCTL are used to select the direction of bidirectional or three-state output signal pins. A "1" in these cells designates that the associated bus or bits are floated if the pins are three-state, or selected as input if they are bidirectional.

- WRCTL controls D31–D0 and DP3–DP0
- ABUSCTL controls A31–A2
- BUSCTL controls ADS#, BLAST#, PLOCK#, LOCK#, W/R#, BE0#, BE1#, BE2#, BE3#, M/IO#, D/C#, PWT, and PCD
- MISCCTL controls PCHK#, HLDA, and BREQ

The following is the bit order of the embedded IntelDX2 processor boundary scan register:

TDO ← A2, A3, A4, A5, RESERVED, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, IGNNE#, FERR#, SMI#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRCTL ← **TDI**

5.0 ELECTRICAL SPECIFICATIONS

5.1 Maximum Ratings

Table 16 is a stress rating only. Extended exposure to the Maximum Ratings may affect device reliability.

Furthermore, although the embedded IntelDX2 processor contains protective circuitry to resist damage from electrostatic discharge, always take precautions to avoid high static voltages or electric fields.

Functional operating conditions are given in **Section 5.2, DC Specifications** and **Section 5.3, AC Specifications.**

Table 16. Absolute Maximum Ratings

Case Temperature under Bias	−65°C to +110°C
Storage Temperature	−65°C to +150°C
DC Voltage on Any Pin with Respect to Ground	−0.5V to V_{CC} + 0.5V
Supply Voltage V_{CC} with Respect to V_{SS}	−0.5V to +6.5V

5.2 DC Specifications

The following tables show the operating supply voltages, DC I/O specifications, and component power consumption for the embedded IntelDX2 processor.

Table 17. Operating Supply Voltages

Product	V_{CC}
SB80486DX2SC50	3.3V ±0.3V
A80486DX2SA66	5.0V ±0.25V

Table 18. 3.3V DC Specifications

Functional Operating Range: V_{CC} = 3.3V ±0.3V, T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} + 0.3	V	Note 1
V_{IHC}	Input HIGH Voltage of CLK	V_{CC} − 0.6	V_{CC} + 0.3	V	
V_{OL}	Output LOW Voltage I_{OL} = 2.0 mA I_{OL} = 100 μA		 0.4 0.2	 V V	
V_{OH}	Output HIGH Voltage I_{OH} = −2.0 mA I_{OH} = −100 μA	 2.4 V_{CC} − 0.2		 V V	
I_{LI}	Input Leakage Current		±15	μA	Note 2
I_{IH}	Input Leakage Current SRESET		200 300	μA μA	Note 3 Note 3
I_{IL}	Input Leakage Current		−400	μA	Note 4
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		10	pF	Note 5
C_{OUT}	I/O or Output Capacitance		10	pF	Note 5
C_{CLK}	CLK Capacitance		6	pF	Note 5

NOTES:
1. All inputs except CLK.
2. This parameter is for inputs without pull-up or pull-down resistors and 0V ≤ VIN ≤ V_{CC}.
3. This parameter is for inputs with pull-down resistors and V_{IH} = 2.4V.
4. This parameter is for inputs with pull-up resistors and V_{IL} = 0.4V.
5. F_C = 1 MHz. Not 100% tested.

Table 19. 3.3V I_{CC} Values

Functional Operating Range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	40 MHz 50 MHz		450 mA 550 mA	Note 1
I_{CC} Active (Thermal Design)	40 MHz 50 MHz	318 mA 395 mA	416 mA 507 mA	Notes 2, 3, 4
I_{CC} Stop Grant	40 MHz 50 MHz	20 mA 23 mA	40 mA 50 mA	Note 5
I_{CC} Stop Clock	0 MHz	100 μA	1 mA	Note 6

NOTES:
1. This parameter is for proper power supply selection. It is measured using the worst case instruction mix at V_{CC} = 3.6V.
2. The maximum current column is for thermal design power dissipation. It is measured using the worst case instruction mix at V_{CC} = 3.3V.
3. The typical current column is the typical operating current in a system. This value is measured in a system using a typical device at V_{CC} = 3.3V, running Microsoft Windows 3.1 at an idle condition. This typical value is dependent upon the specific system configuration.
4. Typical values are not 100% tested.
5. The I_{CC} Stop Grant specification refers to the I_{CC} value once the embedded IntelDX2 processor enters the Stop Grant or Auto HALT Power Down state.
6. The I_{CC} Stop Clock specification refers to the I_{CC} value once the embedded IntelDX2 processor enters the Stop Clock state. The V_{IH} and V_{IL} levels must be equal to V_{CC} and 0V, respectively, to meet the I_{CC} Stop Clock specifications.

Table 20. 5V DC Specifications

Functional operating range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC}+0.3	V	
V_{OL}	Output LOW Voltage		0.45	V	Note 1
V_{OH}	Output HIGH Voltage	2.4		V	Note 2
I_{LI}	Input Leakage Current		±15	μA	Note 3
I_{IH}	Input Leakage Current SRESET		200 300	μA μA	Note 4 Note 4
I_{IL}	Input Leakage Current		−400	μA	Note 5
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		20	pF	Note 6
C_{OUT}	Output or I/O Capacitance		20	pF	Note 6
C_{CLK}	CLK Capacitance		20	pF	Note 6

NOTES:
1. This parameter is measured at:
   Address, Data, BE*n* 4.0 mA
   Definition, Control 5.0 mA
2. This parameter is measured at:
   Address, Data, BE*n* −1.0 mA
   Definition, Control −0.9 mA
3. This parameter is for inputs without pull-ups or pull-downs and $0V \leq V_{IN} \leq V_{CC}$.
4. This parameter is for inputs with pull-downs and V_{IH} = 2.4V.
5. This parameter is for inputs with pull-ups and V_{IL} = 0.45V.
6. F_C = 1 MHz; Not 100% tested.

4

Table 21. 5V I_{CC} Values

Functional Operating Range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	50 MHz 66 MHz		950 mA 1200 mA	Note 1
I_{CC} Active (Thermal Design)	50 MHz 66 MHz	680 mA 901 mA	906 mA 1145 mA	Notes 2, 3, 4
I_{CC} Stop Grant	50 MHz 66 MHz	35 mA 40 mA	70 mA 90 mA	Note 5
I_{CC} Stop Clock	0 MHz	200 μA	2 mA	Note 6

NOTES:

1. This parameter is for proper power supply selection. It is measured using the worst case instruction mix at V_{CC} = 5.25V.
2. The maximum current column is for thermal design power dissipation. It is measured using the worst case instruction mix at V_{CC} = 5V.
3. The typical current column is the typical operating current in a system. This value is measured in a system using a typical device at V_{CC} = 5V, running Microsoft Windows 3.1 at an idle condition. This typical value is dependent upon the specific system configuration.
4. Typical values are not 100% tested.
5. The I_{CC} Stop Grant specification refers to the I_{CC} value once the embedded IntelDX2 processor enters the Stop Grant or Auto HALT Power Down state.
6. The I_{CC} Stop Clock specification refers to the I_{CC} value once the processor enters the Stop Clock state. The V_{IH} and V_{IL} levels must be equal to V_{CC} and 0V, respectively, in order to meet the I_{CC} Stop Clock specifications.

5.3 AC Specifications

The AC specifications for the embedded IntelDX2 processor are given in this section.

Table 22. AC Characteristics

T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified. (Sheet 1 of 2)

Symbol	Parameter	V_{CC} (Package)				Unit	Figure	Notes
		3.3V (208-Lead SQFP)		5V (168-Pin PGA)				
		Min	Max	Min	Max			
	CLK Frequency	8	25	8	33	MHz		Note 1
t_1	CLK Period	40	125	30	125	ns	4	
t_{1a}	CLK Period Stability		±250		±250	ps	4	Adjacent clocks
t_2	CLK High Time	14		11		ns	4	at 2V
t_3	CLK Low Time	14		11		ns	4	at 0.8V
t_4	CLK Fall Time		4		3	ns	4	2V to 0.8V
t_5	CLK Rise Time		4		3	ns	4	0.8V to 2V
t_6	A31–A2, PWT, PCD, BE3–BE0#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT#, FERR# Valid Delay	3	19	3	16	ns	8	
t_7	A31–A2, PWT, PCD, BE3–BE0#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA Float Delay		28		20	ns	9	Note 2
t_8	PCHK# Valid Delay	3	24	3	22	ns	7	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	24	3	20	ns	8	
t_9	BLAST#, PLOCK# Float Delay		28		20	ns	9	Note 2
t_{10}	D31–D0, DP3–DP0 Write Data Valid Delay	3	20	3	18	ns	8	
t_{11}	D31–D0, DP3–DP0 Write Data Float Delay		28		20	ns	9	Note 2
t_{12}	EADS# Setup Time	8		5		ns	5	
t_{13}	EADS# Hold Time	3		3		ns	5	

Table 22. AC Characteristics

T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified. (Sheet 2 of 2)

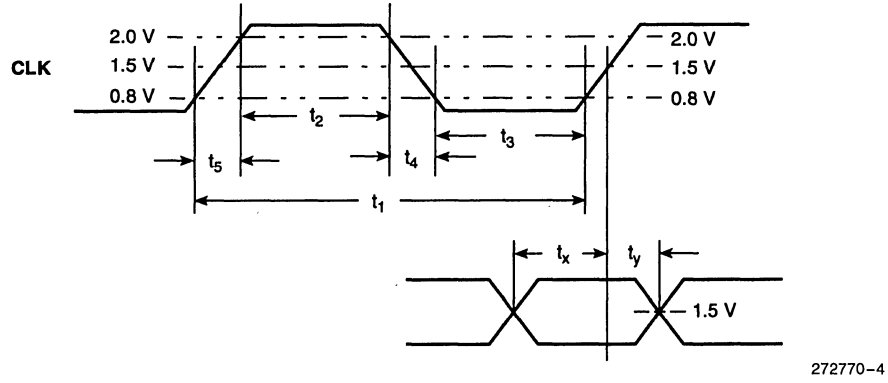
Symbol	Parameter	V_{CC} (Package)				Unit	Figure	Notes
		3.3V (208-Lead SQFP)		5V (168-Pin PGA)				
		Min	Max	Min	Max			
t_{14}	KEN#, BS16#, BS8# Setup Time	8		5		ns	5	
t_{15}	KEN#, BS16#, BS8# Hold Time	3		3		ns	5	
t_{16}	RDY#, BRDY# Setup Time	8		5		ns	6	
t_{17}	RDY#, BRDY# Hold Time	3		3		ns	6	
t_{18}	HOLD, AHOLD Setup Time	10		6		ns	5	
t_{18a}	BOFF# Setup Time	10		8		ns	5	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		3		ns	5	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Setup Time	10		5		ns	5	Note 3
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Hold Time	3		3		ns	5	Note 3
t_{22}	D31–D0, DP3–DP0, A31–A4 Read Setup Time	6		5		ns	6 5	
t_{23}	D31–D0, DP3–DP0, A31–A4 Read Hold Time	3		3		ns	6 5	

NOTES:
1. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant bus cycle protocol is used.
2. Not 100% tested, guaranteed by design characterization.
3. A reset pulse width of 15 CLK cycles is required for warm resets (RESET or SRESET). Power-up resets (cold resets) require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.

Table 23. AC Specifications for the Test Access Port
(Both 3.3V SQFP and 5V PGA Processors)
T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{24}	TCK Frequency		8	MHz		Note 1
t_{25}	TCK Period	125		ns	10	
t_{26}	TCK High Time	40		ns	10	@ 2.0V
t_{27}	TCK Low Time	40		ns	10	@ 0.8V
t_{28}	TCK Rise Time		8	ns	10	Note 2
t_{29}	TCK Fall Time		8	ns	10	Note 2
t_{30}	TDI, TMS Setup Time	8		ns	11	Note 3
t_{31}	TDI, TMS Hold Time	10		ns	11	Note 3
t_{32}	TDO Valid Delay	3	30	ns	11	Note 3
t_{33}	TDO Float Delay		36	ns	11	Note 3
t_{34}	All Outputs (except TDO) Valid Delay	3	30	ns	11	Note 3
t_{35}	All Outputs (except TDO) Float Delay		36	ns	11	Note 3
t_{36}	All Inputs (except TDI, TMS, TCK) Setup Time	8		ns	11	Note 3
t_{37}	All Inputs (except TDI, TMS, TCK) Hold Time	10		ns	11	Note 3

NOTES:
1. TCK period ≤ CLK period.
2. Rise/Fall times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10 ns increase in TCK period.
3. Parameters t_{30}–t_{37} are measured from TCK.



NOTE:
t_x = input setup times
t_y = input hold times, output float, valid hold times

Figure 4. CLK Waveform

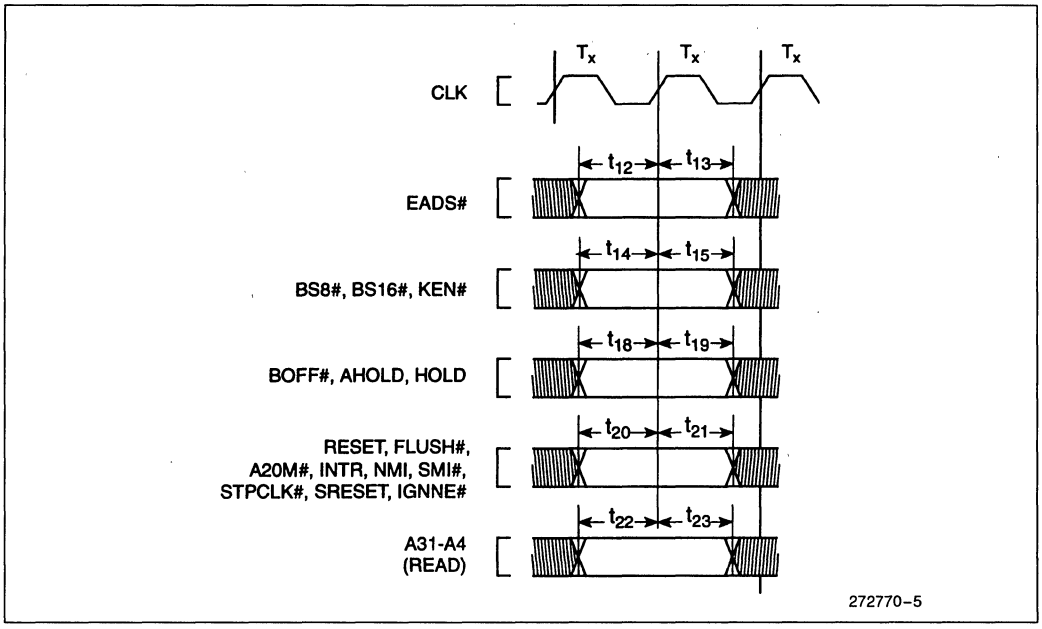


Figure 5. Input Setup and Hold Timing

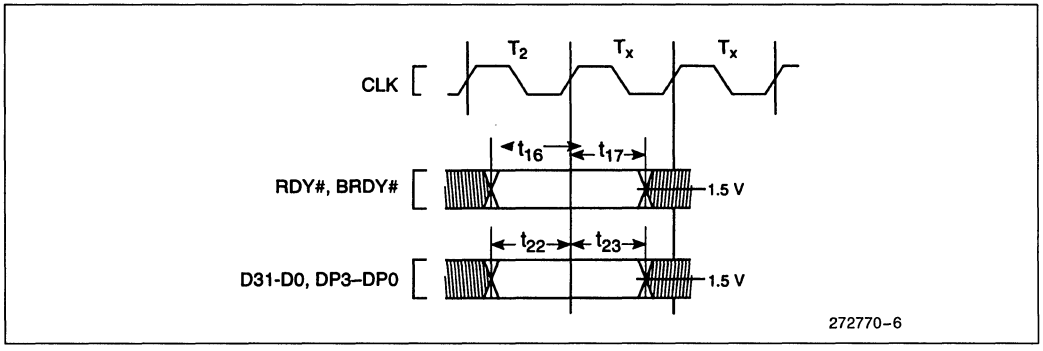


Figure 6. Input Setup and Hold Timing

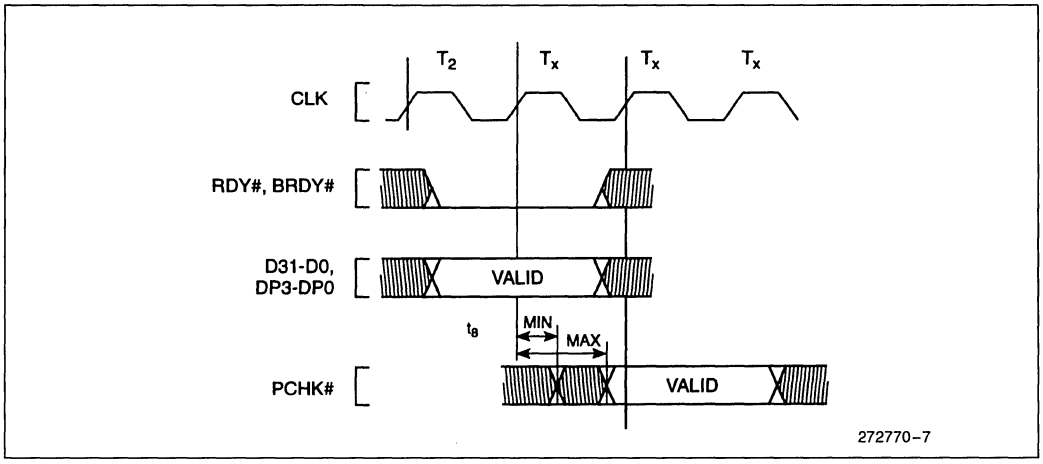


Figure 7. PCHK# Valid Delay Timing

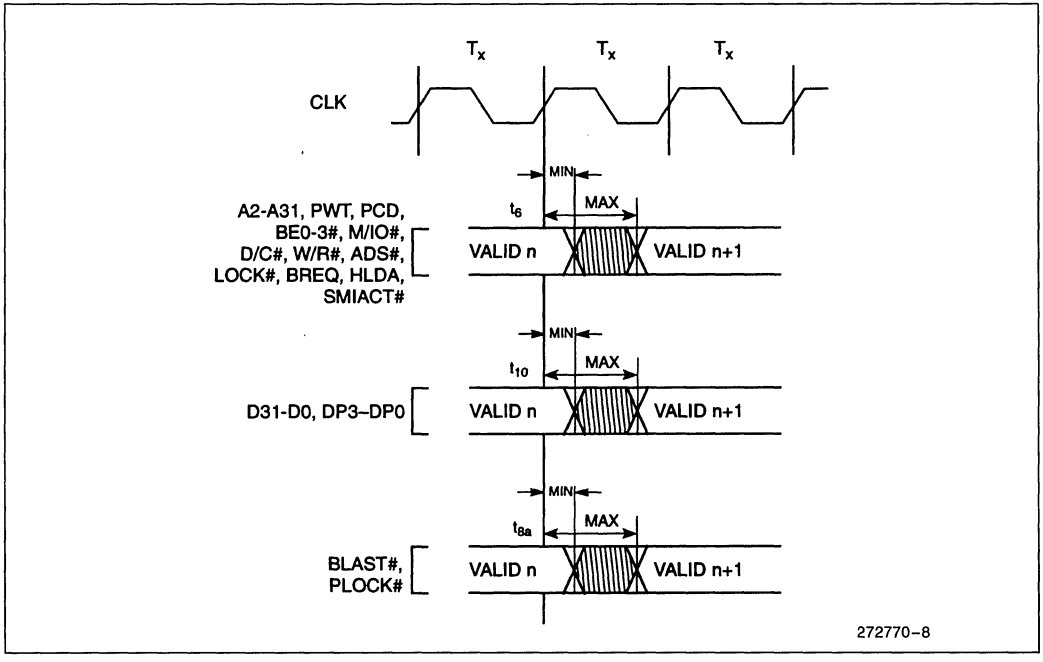


Figure 8. Output Valid Delay Timing

4

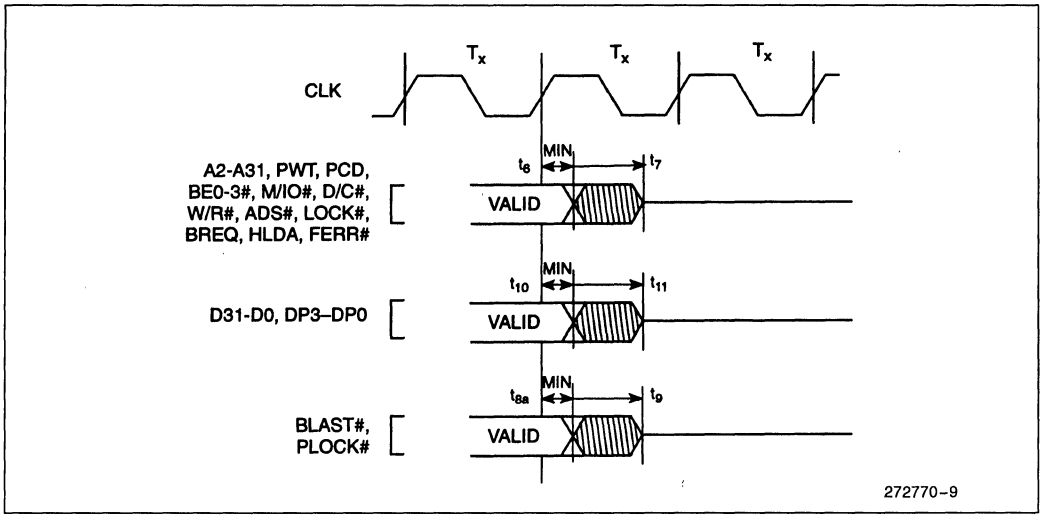


Figure 9. Maximum Float Delay Timing

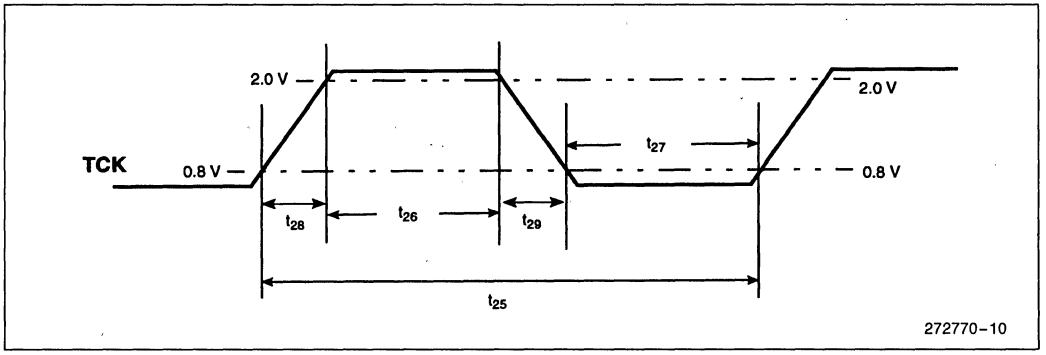


Figure 10. TCK Waveform

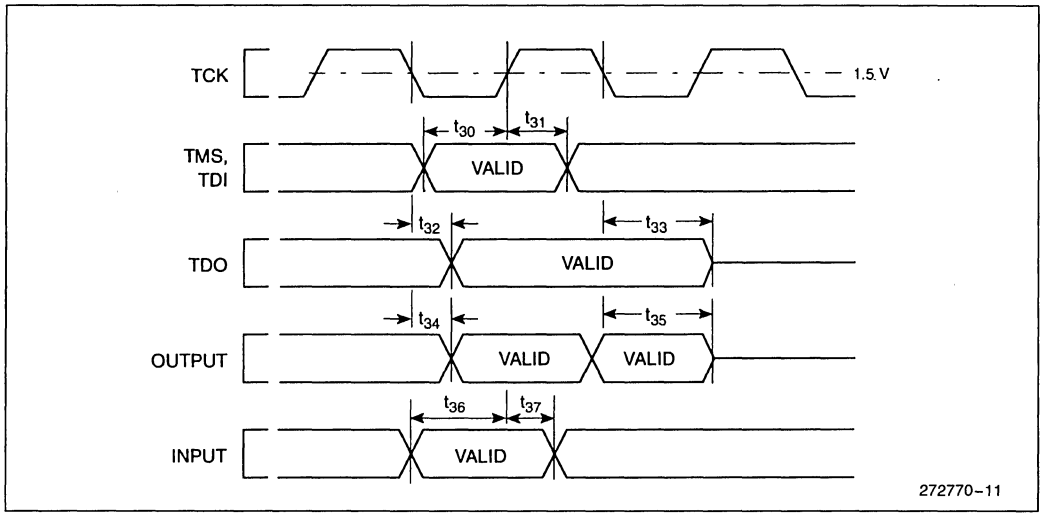
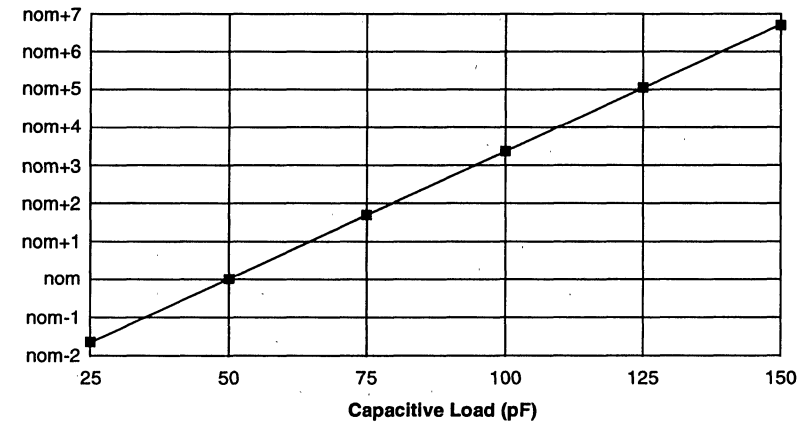


Figure 11. Test Signal Timing Diagram

4

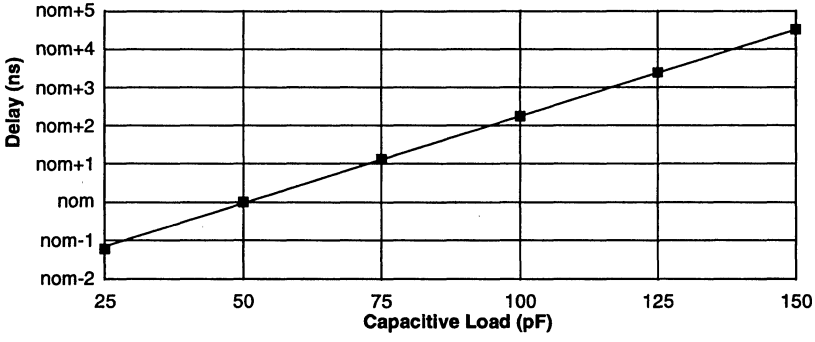
5.4 Capacitive Derating Curves

The following graphs are the capacitive derating curves for the embedded IntelDX2 processor.



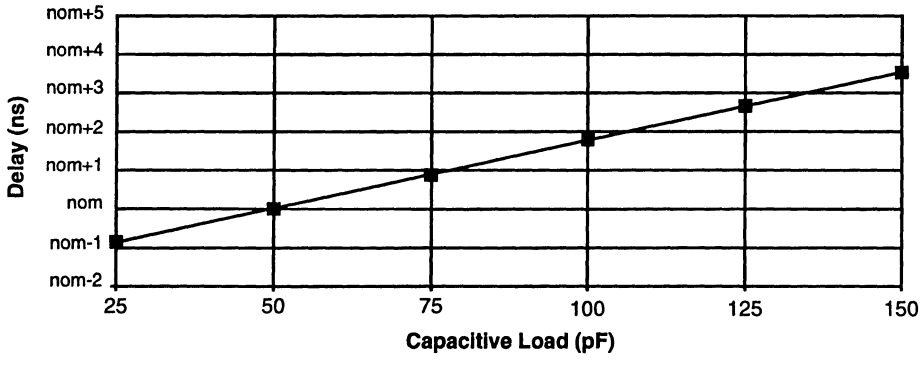
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 12. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition, 3.3V Processor



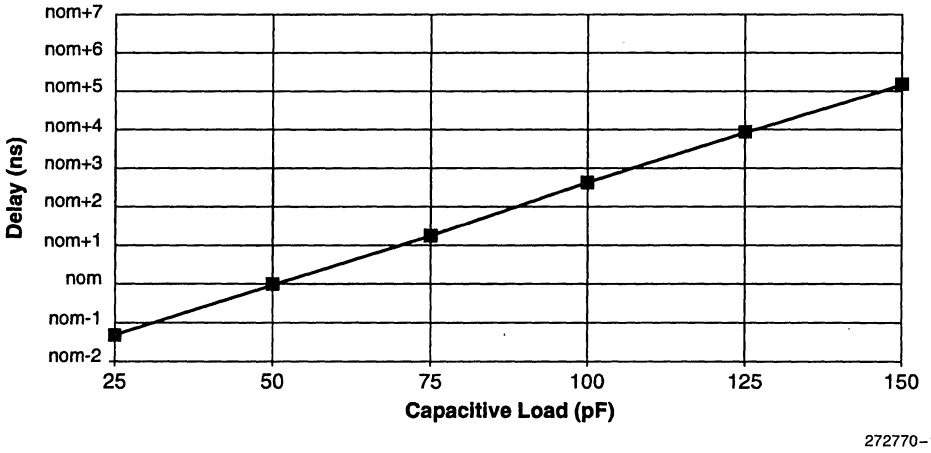
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 13. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition, 3.3V Processor



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 14. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition, 5V Processor



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 15. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition, 5V Processor

6.0 MECHANICAL DATA

This section describes the packaging dimensions and thermal specifications for the embedded IntelDX2 processor.

6.1 Package Dimensions

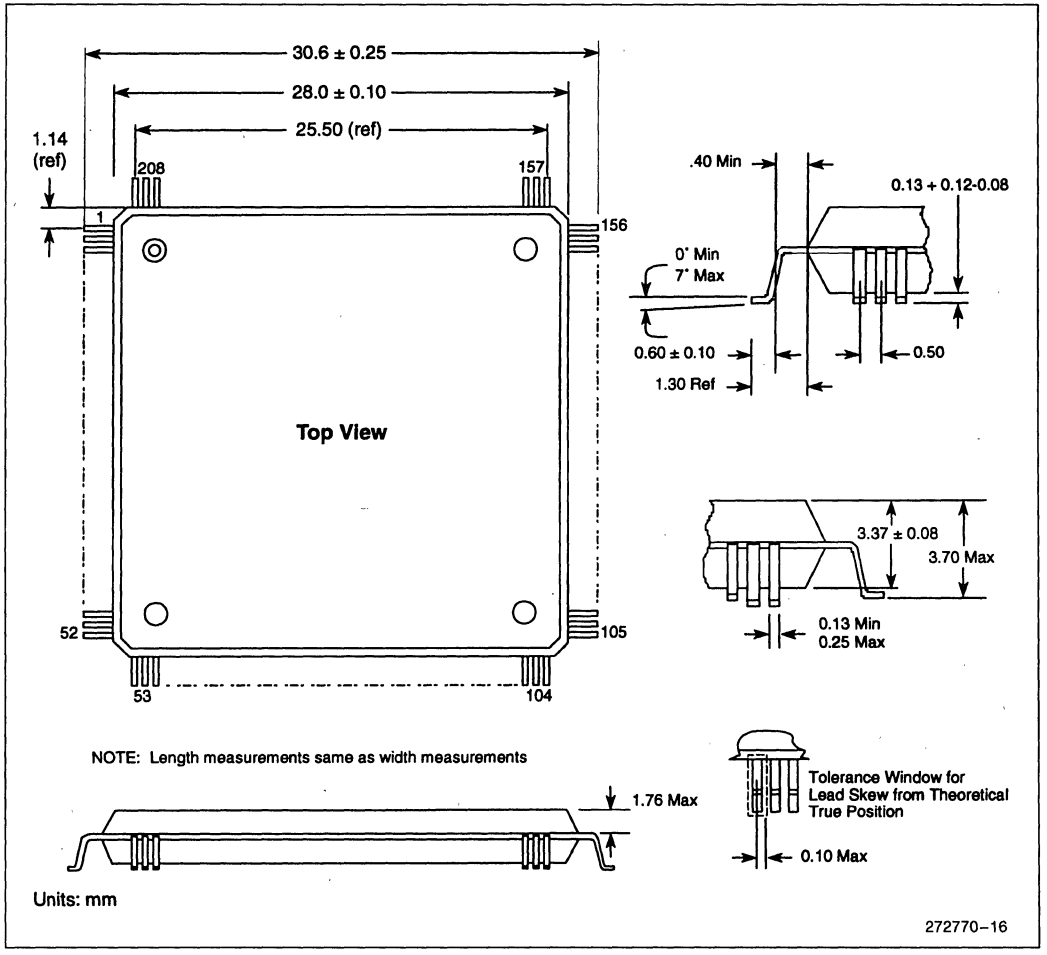


Figure 16. 208-Lead SQFP Package Dimensions

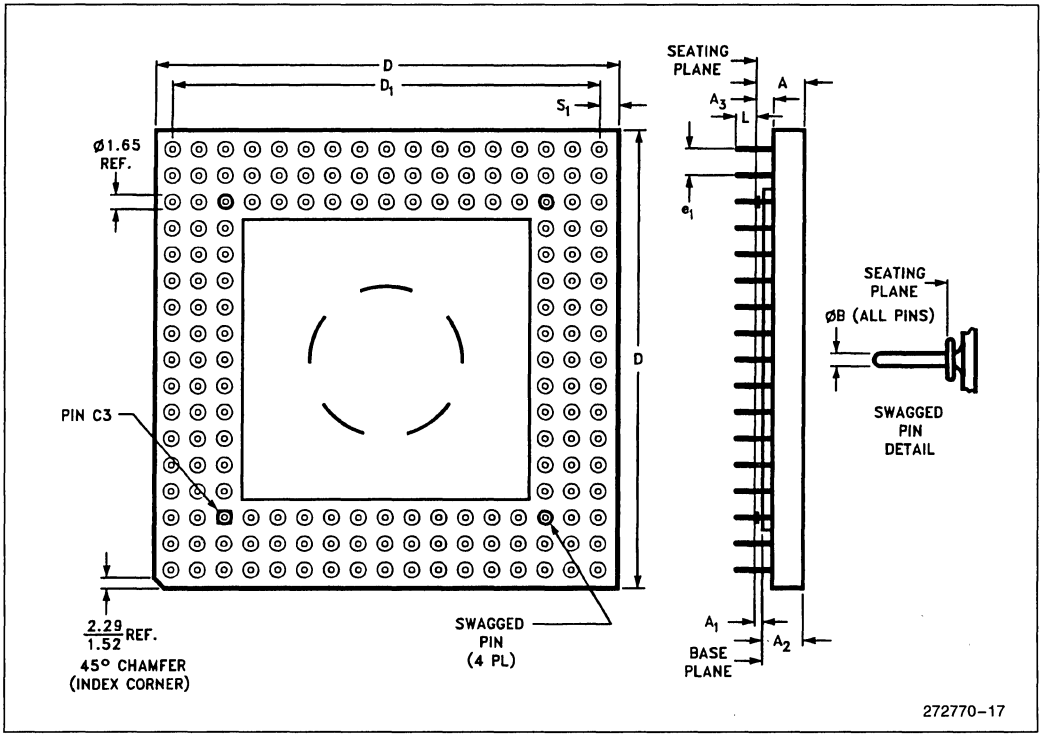


Figure 17. Principal Dimensions and Data for 168-Pin Pin Grid Array Package

Table 24. 168-Pin Ceramic PGA Package Dimensions

Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	3.56	4.57		0.140	0.180	
A_1	0.64	1.14	SOLID LID	0.025	0.045	SOLID LID
A_2	2.8	3.5	SOLID LID	0.110	0.140	SOLID LID
A_3	1.14	1.40		0.045	0.055	
B	0.43	0.51		0.017	0.020	
D	44.07	44.83		1.735	1.765	
D_1	40.51	40.77		1.595	1.605	
e_1	2.29	2.79		0.090	0.110	
L	2.54	3.30		0.100	0.130	
N	168			168		
S_1	1.52	2.54		0.060	0.100	

4

Table 25. Ceramic PGA Package Dimension Symbols

Letter or Symbol	Description of Dimensions
A	Distance from seating plane to highest point of body
A_1	Distance between seating plane and base plane (lid)
A_2	Distance from base plane to highest point of body
A_3	Distance from seating plane to bottom of body
B	Diameter of terminal lead pin
D	Largest overall package dimension of length
D_1	A body length dimension, outer lead center to outer lead center
e_1	Linear spacing between true lead position centerlines
L	Distance from seating plane to end of lead
S_1	Other body dimension, outer lead center to edge of body

NOTES:
1. Controlling dimension: millimeter.
2. Dimension "e_1" ("e") is non-cumulative.
3. Seating plane (standoff) is defined by P.C. board hole size: 0.0415 inch–0.0430 inch.
4. Dimensions "B", "B_1" and "C" are nominal.
5. Details of Pin 1 identifier are optional.

6.2 Package Thermal Specifications

The embedded IntelDX2 processor is specified for operation when the case temperature (T_C) is within the range of 0°C to 85°C. T_C may be measured in any environment to determine whether the processor is within the specified operating range.

The ambient temperature (T_A) can be calculated from θ_{JC} and θ_{JA} from the following equations:

$$T_J = T_C + P * \theta_{JC}$$
$$T_A = T_J - P * \theta_{JA}$$

$$T_C = T_A + P * [\theta_{JA} - \theta_{JC}]$$
$$T_A = T_C - P * [\theta_{JA} - \theta_{JC}]$$

Where T_J, T_A, T_C equals Junction, Ambient and Case Temperature respectively. θ_{JC}, θ_{JA} equals Junction-to-Case and Junction-to-Ambient thermal Resistance, respectively. P is defined as Maximum Power Consumption.

Values for θ_{JA} and θ_{JC} are given in the following tables for each product at its maximum operating frequencies. Maximum T_A is shown for each product operating at various processor frequencies (twice the CLK frequencies).

Table 26. Thermal Resistance, θ_{JA} (°C/W)

	θ_{JA} vs. Airflow—ft/min. (m/sec)					
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
208-Lead SQFP (3.3V)—Without Heat Sink	24.0	17.0	15.0	13.0	—	—
168-Pin PGA (5V)—Without Heat Sink	17.0	14.5	12.5	11.0	10.0	9.5
168-Pin PGA (5V)—With Heat Sink*	13.0	8.0	6.0	5.0	4.5	4.25

*0.350" high omnidirectional heat sink.

Table 27. Thermal Resistance, θ_{JC} (°C/W)

	θ_{JC} vs. Airflow—ft/min. (m/sec)			
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
	0	200	400	600
208-Lead SQFP (3.3V)	3.5	6.0	6.0	6.0
168-Pin PGA (5V)	1.5	—	—	—

Table 28. Maximum $T_{ambient}$, T_A max (°C)

	Airflow—ft/min. (m/sec)				
	Freq. (MHz)	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
IntelDX2™ Processor					
208-Lead SQFP (3.3V)	40	57	70	73	75
Without Heat Sink	50	51	67	70	73
168-Pin PGA (5V)	50	15	26	35	46
Without Heat Sink	66	−4	11	22	36
168-Pin PGA (5V)	50	33	56	65	69
With Heat Sink	66	19	48	59	65

EMBEDDED WRITE-BACK ENHANCED IntelDX4™ PROCESSOR

- Up to 100 MHz Operation
- Integrated Floating-Point Unit
- Speed-Multiplying Technology
- 32-Bit RISC Technology Core
- 16-Kbyte Write-Back Cache
- 3.3V Core Operation with 5V Tolerant I/O Buffers
- Burst Bus Cycles
- Dynamic Bus Sizing for 8- and 16-bit Data Bus Devices
- SL Technology
- Data Bus Parity Generation and Checking
- Boundary Scan (JTAG)
- 3.3-Volt Processor, 75 MHz, 25 MHz CLK
  — 208-Lead Shrink Quad Flat Pack (SQFP)
- 3.3-Volt Processor, 100 MHz, 33 MHz CLK
  — 208-Lead Shrink Quad Flat Pack (SQFP)
  — 168-Pin Pin Grid Array (PGA)
- Binary Compatible with Large Software Base

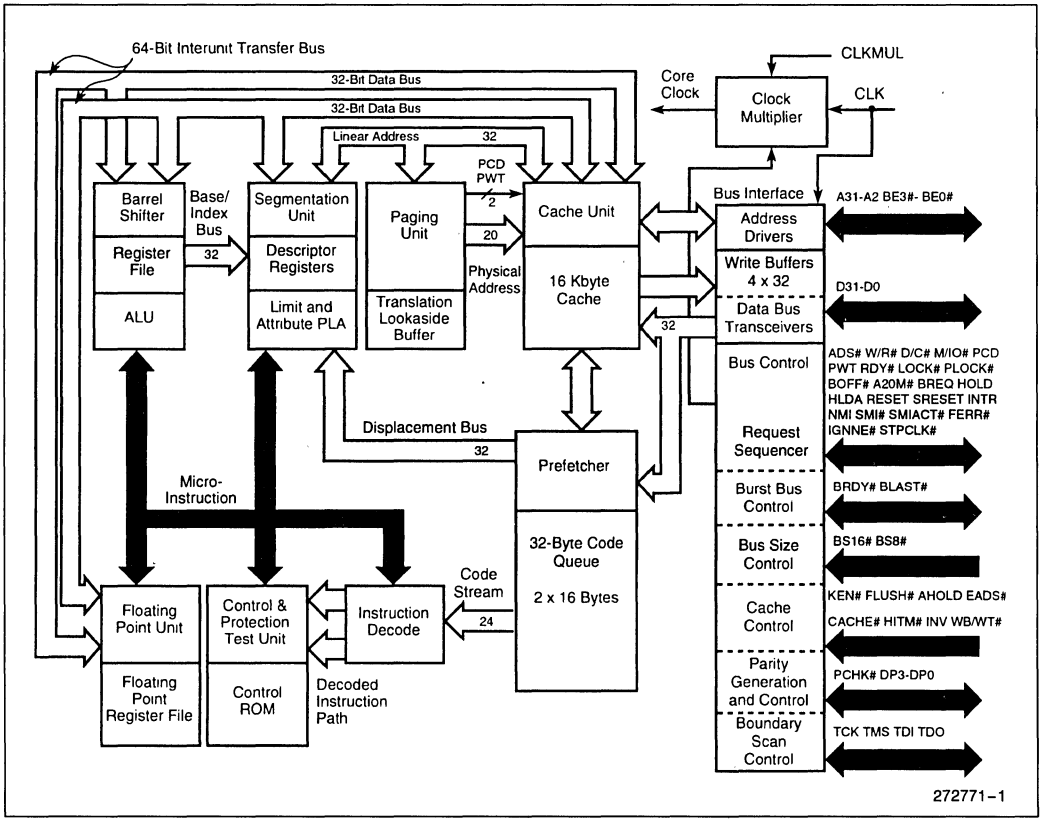


Figure 1. Embedded Write-Back Enhanced IntelDX4™ Processor Block Diagram

December 1995
Order Number: 272771-001

EMBEDDED WRITE-BACK ENHANCED IntelDX4™ PROCESSOR

CONTENTS

CONTENTS

PAGE

FIGURES

CONTENTS

PAGE

TABLES

1.0 INTRODUCTION

The embedded Write-Back Enhanced IntelDX4™ processor provides high performance to 32-bit, embedded applications. Designed for applications that need a floating-point unit, the processor is ideal for embedded designs running DOS*, Microsoft* Windows*, OS/2*, or UNIX* applications written for the Intel architecture. Projects can be completed quickly using the wide range of software tools, utilities, assemblers and compilers that are available for desktop computer systems. Also, developers can find advantages in using existing chipsets and peripheral components in their embedded designs.

The embedded Write-Back Enhanced IntelDX4 processor is binary compatible with the Intel386™ and earlier Intel processors. Compared with the Intel386 processor, it provides faster execution of many commonly-used instructions. It also provides the benefits of an integrated, 16-Kbyte, write-back cache for code and data. Its data bus can operate in burst mode which provides up to 106-Mbyte-per-second transfers for cache-line fills and instruction prefetches.

Intel's SL technology is incorporated in the embedded Write-Back Enhanced IntelDX4 processor. Utilizing Intel's System Management Mode (SMM) enables designers to develop energy-efficient systems.

Two component packages are available:

- 168-pin Pin Grid Array (PGA)
- 208-lead Shrink Quad Flat Pack (SQFP)

The processor operates at either two or three times the external bus frequency. At two times the external bus frequency the processor operates up to 66 MHz, (33-MHz CLK). At three times the external bus frequency the processor operates up to 100 MHz (33-MHz CLK).

1.1 Features

The embedded Write-Back Enhanced IntelDX4 processor offers these features:

- **32-bit RISC-Technology Core**—The embedded Write-Back Enhanced IntelDX4 processor performs a complete set of arithmetic and logical operations on 8-, 16-, and 32-bit data types using a full-width ALU and eight general purpose registers.
- **Single Cycle Execution**—Many instructions execute in a single clock cycle.
- **Instruction Pipelining**—Overlapped instruction fetching, decoding, address translation and execution.
- **On-Chip Floating-Point Unit**—Intel486™ processors support the 32-, 64-, and 80-bit formats specified in IEEE standard 754. The unit is binary compatible with the 8087, Intel287™, Intel387™ coprocessors, and Intel OverDrive® processor.
- **On-Chip Cache with Cache Consistency Support**—A 16-Kbyte internal cache is used for both data and instructions. It is configurable to be write-back or write-through on a line-by-line basis. The internal cache implements a modified MESI protocol, which is applicable to uniprocessor systems. Cache hits provide zero wait-state access times for data within the cache. Bus activity is tracked to detect alterations in the memory represented by the internal cache. The internal cache can be invalidated or flushed so that an external cache controller can maintain cache consistency.
- **External Cache Control**—Write-back and flush controls for an external cache are provided so the processor can maintain cache consistency.
- **On-Chip Memory Management Unit**—Address management and memory space protection mechanisms maintain the integrity of memory in a multitasking and virtual memory environment. Both memory segmentation and paging are supported.
- **Burst Cycles**—Burst transfers allow a new double-word to be read from memory on each bus clock cycle. This capability is especially useful for instruction prefetch and for filling the internal cache. Data written from the processor to memory can also be burst transfers.

- **Write Buffers**—The processor contains four write buffers to enhance the performance of consecutive writes to memory. The processor can continue internal operations after a write to these buffers, without waiting for the write to be completed on the external bus.
- **Bus Backoff**—When another bus master needs control of the bus during a processor initiated bus cycle, the embedded Write-Back Enhanced IntelDX4 processor floats its bus signals, then restarts the cycle when the bus becomes available again.
- **Instruction Restart**—Programs can continue execution following an exception generated by an unsuccessful attempt to access memory. This feature is important for supporting demand-paged virtual memory applications.
- **Dynamic Bus Sizing**—External controllers can dynamically alter the effective width of the data bus. Bus widths of 8, 16, or 32 bits can be used.
- **Boundary Scan (JTAG)**—Boundary Scan provides in-circuit testing of components on printed circuit boards. The Intel Boundary Scan implementation conforms with the IEEE Standard Test Access Port and Boundary Scan Architecture.
- **Enhanced Bus Mode**—The definitions of some signals have been changed to support write-back cache mode.

Intel's SL technology provides these features:

- **Intel System Management Mode (SMM)**—A unique Intel architecture operating mode provides a dedicated special purpose interrupt and address space that can be used to implement intelligent power management and other enhanced functions in a manner that is completely transparent to the operating system and applications software.
- **I/O Restart**—An I/O instruction interrupted by a System Management Interrupt (SMI#) can automatically be restarted following the execution of the RSM instruction.
- **Stop Clock**—The embedded Write-Back Enhanced IntelDX4 processor has a stop clock control mechanism that provides two low-power states: a Stop Grant state (20–50 mA typical, depending on input clock frequency) and a Stop Clock state (~600 μA typical, with input clock frequency of 0 MHz).
- **Auto HALT Power Down**—After the execution of a HALT instruction, the embedded Write-Back Enhanced IntelDX4 processor issues a normal Halt bus cycle and the clock input to the processor core is automatically stopped, causing the processor to enter the Auto HALT Power Down state (20–50 mA typical, depending on input clock frequency).
- **Auto Idle Power Down**—This function allows the processor to reduce the core frequency to the bus frequency when both the core and bus are idle. Auto Idle Power Down is software transparent and does not affect processor performance. Auto Idle Power Down provides an average power savings of 10% and is only applicable to clock multiplied processors.

4

1.2 Family Members

Table 1 shows the embedded Write-Back Enhanced IntelDX4 processors and briefly describes their characteristics.

Table 1. The Embedded Write-Back Enhanced IntelDX4™ Processor Family

Product	Supply Voltage V_{CC}	Maximum Processor Frequency	Maximum External Bus Frequency	Package
FC80486DX4WB75	3.3V	75 MHz	25 MHz	208-Lead SQFP
FC80486DX4WB100	3.3V	100 MHz	33 MHz	208-Lead SQFP
A80486DX4WB100	3.3V	100 MHz	33 MHz	168-Pin PGA

2.0 HOW TO USE THIS DOCUMENT

For a complete set of documentation related to the embedded Write-Back Enhanced IntelDX4 processor, use this document in conjunction with the following reference documents:

- *Intel486™ Processor Family* datasheet—Order No. 242202
- *Intel486 Microprocessor Family Programmer's Reference Manual*—Order No. 240486
- Intel Application Note AP-485—*Intel Processor Identification with the CPUID Instruction*—Order No. 241618

The information in the reference documents for the IntelDX4 processor applies to the embedded Write-Back Enhanced IntelDX4 processor. Some of the IntelDX4 processor information is duplicated in this document to minimize the dependence on the reference documents.

3.0 PIN DESCRIPTIONS

3.1 Pin Assignments

The following figures and tables show the pin assignments of each package type for the embedded Write-Back Enhanced IntelDX4 processor. Tables are provided showing the pin differences between the embedded Write-Back Enhanced IntelDX4 processor and other embedded Intel486 processor products.

208-Lead SQFP - Quad Flat Pack

- Figure 2, Package Diagram for 208-Lead SQFP Embedded Write-Back Enhanced IntelDX4™ Processor (pg. 8)
- Table 2, Pinout Differences for 208-Lead SQFP Package (pg. 9)
- Table 3, Pin Assignment for 208-Lead SQFP Package (pg. 10)
- Table 4, Pin Cross Reference for 208-Lead SQFP Package (pg. 12)

168-Pin PGA - Pin Grid Array

- Figure 3, Package Diagram for 168-Pin PGA Embedded Write-Back Enhanced IntelDX4™ Processor (pg. 14)
- Table 5, Pinout Differences for 168-Pin PGA Package (pg. 15)
- Table 6, Pin Assignment for 168-Pin PGA Package (pg. 16)
- Table 7, Pin Cross Reference for 168-Pin PGA Package (pg. 18)

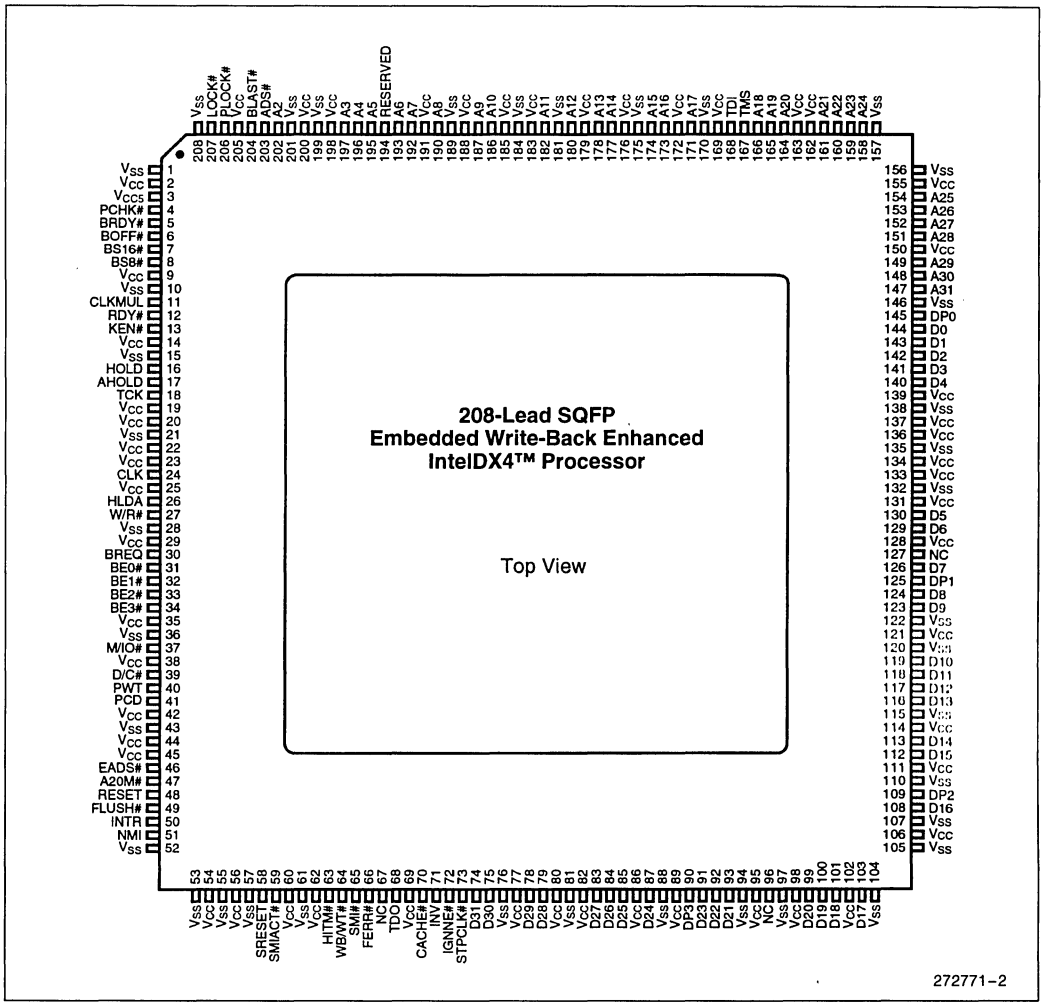


Figure 2. Package Diagram for 208-Lead SQFP Embedded Write-Back Enhanced IntelDX4™ Processor

Table 2. Pinout Differences for 208-Lead SQFP Package

Pin #	Embedded Intel486™ SX Processor	Embedded IntelDX2™ Processor	Embedded Write-Back Enhanced IntelDX4™ Processor
3	V_{CC}(1)	V_{CC}	V_{CC5}
11	INC(2)	INC	CLKMUL
63	INC	INC	HITM#
64	INC	INC	WB/WT#
66	INC	FERR#	FERR#
70	INC	INC	CACHE#
71	INC	INC	INV
72	INC	IGNNE#	IGNNE#

NOTES:
1. This pin location is for the V_{CC5} pin on the embedded IntelDX4 processor. For compatibility with 3.3V processors that have 5V-tolerant input buffers (i.e., embedded IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane.
2. INC. Internal No Connect. These pins are not connected to any internal pad. However, signals are defined for the location of the INC pins in the embedded IntelDX4 processor. One system design can accommodate any one of these processors provided the purpose of each INC pin is understood before it is used.

Table 3. Pin Assignment for 208-Lead SQFP Package (Sheet 1 of 2)

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
1	V_{SS}	53	V_{SS}	105	V_{SS}	157	V_{SS}
2	V_{CC}	54	V_{CC}	106	V_{CC}	158	A24
3	V_{CC5}	55	V_{SS}	107	V_{SS}	159	A23
4	PCHK#	56	V_{CC}	108	D16	160	A22
5	BRDY#	57	V_{SS}	109	DP2	161	A21
6	BOFF#	58	SRESET	110	V_{SS}	162	V_{CC}
7	BS16#	59	SMIACT#	111	V_{CC}	163	V_{CC}
8	BS8#	60	V_{CC}	112	D15	164	A20
9	V_{CC}	61	V_{SS}	113	D14	165	A19
10	V_{SS}	62	V_{CC}	114	V_{CC}	166	A18
11	CLKMUL	63	HITM#	115	V_{SS}	167	TMS
12	RDY#	64	WB/WT#	116	D13	168	TDI
13	KEN#	65	SMI#	117	D12	169	V_{CC}
14	V_{CC}	66	FERR#	118	D11	170	V_{SS}
15	V_{SS}	67	NC(1)	119	D10	171	A17
16	HOLD	68	TDO	120	V_{SS}	172	V_{CC}
17	AHOLD	69	V_{CC}	121	V_{CC}	173	A16
18	TCK	70	CACHE#	122	V_{SS}	174	A15
19	V_{CC}	71	INV	123	D9	175	V_{SS}
20	V_{CC}	72	IGNNE#	124	D8	176	V_{CC}
21	V_{SS}	73	STPCLK#	125	DP1	177	A14
22	V_{CC}	74	D31	126	D7	178	A13
23	V_{CC}	75	D30	127	NC(1)	179	V_{CC}
24	CLK	76	V_{SS}	128	V_{CC}	180	A12
25	V_{CC}	77	V_{CC}	129	D6	181	V_{SS}
26	HLDA	78	D29	130	D5	182	A11
27	W/R#	79	D28	131	V_{CC}	183	V_{CC}
28	V_{SS}	80	V_{CC}	132	V_{SS}	184	V_{SS}
29	V_{CC}	81	V_{SS}	133	V_{CC}	185	V_{CC}
30	BREQ	82	V_{CC}	134	V_{CC}	186	A10
31	BE0#	83	D27	135	V_{SS}	187	A9
32	BE1#	84	D26	136	V_{CC}	188	V_{CC}
33	BE2#	85	D25	137	V_{CC}	189	V_{SS}
34	BE3#	86	V_{CC}	138	V_{SS}	190	A8

Table 3. Pin Assignment for 208-Lead SQFP Package (Sheet 2 of 2)

Pin#	Description	Pin#	Description	Pin#	Description	Pin#	Description
35	V_{CC}	87	D24	139	V_{CC}	191	V_{CC}
36	V_{SS}	88	V_{SS}	140	D4	192	A7
37	M/IO#	89	V_{CC}	141	D3	193	A6
38	V_{CC}	90	DP3	142	D2	194	RESERVED
39	D/C#	91	D23	143	D1	195	A5
40	PWT	92	D22	144	D0	196	A4
41	PCD	93	D21	145	DP0	197	A3
42	V_{CC}	94	V_{SS}	146	V_{SS}	198	V_{CC}
43	V_{SS}	95	V_{CC}	147	A31	199	V_{SS}
44	V_{CC}	96	NC(1)	148	A30	200	V_{CC}
45	V_{CC}	97	V_{SS}	149	A29	201	V_{SS}
46	EADS#	98	V_{CC}	150	V_{CC}	202	A2
47	A20M#	99	D20	151	A28	203	ADS#
48	RESET	100	D19	152	A27	204	BLAST#
49	FLUSH#	101	D18	153	A26	205	V_{CC}
50	INTR	102	V_{CC}	154	A25	206	PLOCK#
51	NMI	103	D17	155	V_{CC}	207	LOCK#
52	V_{SS}	104	V_{SS}	156	V_{SS}	208	V_{SS}

NOTE:
1. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC}, or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.

Table 4. Pin Cross Reference for 208-Lead SQFP Package (Sheet 1 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	V_{CC5}	V_{CC}	V_{SS}
A2	202	D0	144	A20M#	47	67	3	2	1
A3	197	D1	143	ADS#	203	96		9	10
A4	196	D2	142	AHOLD	17	127		14	15
A5	195	D3	141	BE0#	31			19	21
A6	193	D4	140	BE1#	32			20	28
A7	192	D5	130	BE2#	33			22	36
A8	190	D6	129	BE3#	34			23	43
A9	187	D7	126	BLAST#	204			25	52
A10	186	D8	124	BOFF#	6			29	53
A11	182	D9	123	BRDY#	5			35	55
A12	180	D10	119	BREQ	30			38	57
A13	178	D11	118	BS16#	7			42	61
A14	177	D12	117	BS8#	8			44	76
A15	174	D13	116	CACHE#	70			45	81
A16	173	D14	113	CLK	24			54	88
A17	171	D15	112	CLKMUL	11			56	94
A18	166	D16	108	D/C#	39			60	97
A19	165	D17	103	DP0	145			62	104
A20	164	D18	101	DP1	125			69	105
A21	161	D19	100	DP2	109			77	107
A22	160	D20	99	DP3	90			80	110
A23	159	D21	93	EADS#	46			82	115
A24	158	D22	92	FERR#	66			86	120
A25	154	D23	91	FLUSH#	49			89	122
A26	153	D24	87	HITM#	63			95	132
A27	152	D25	85	HLDA	26			98	135
A28	151	D26	84	HOLD	16			102	138
A29	149	D27	83	IGNNE#	72			106	146
A30	148	D28	79	INTR	50			111	156
A31	147	D29	78	INV	71			114	157
		D30	75	KEN#	13			121	170
		D31	74	LOCK#	207			128	175
				M/IO#	37			131	181
				NMI	51			133	184

Table 4. Pin Cross Reference for 208-Lead SQFP Package (Sheet 2 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	V_{CC5}	V_{CC}	V_{SS}
				PCD	41			134	189
				PCHK#	4			136	199
				PLOCK#	206			137	201
				PWT	40			139	208
				RDY#	12			150	
				RESERVED	194			155	
				RESET	48			162	
				SMI#	65			163	
				SMIACT#	59			169	
				SRESET	58			172	
				STPCLK#	73			176	
				TCK	18			179	
				TDI	168			183	
				TDO	68			185	
				TMS	167			188	
				WB/WT#	64			191	
				W/R#	27			198	
								200	
								205	

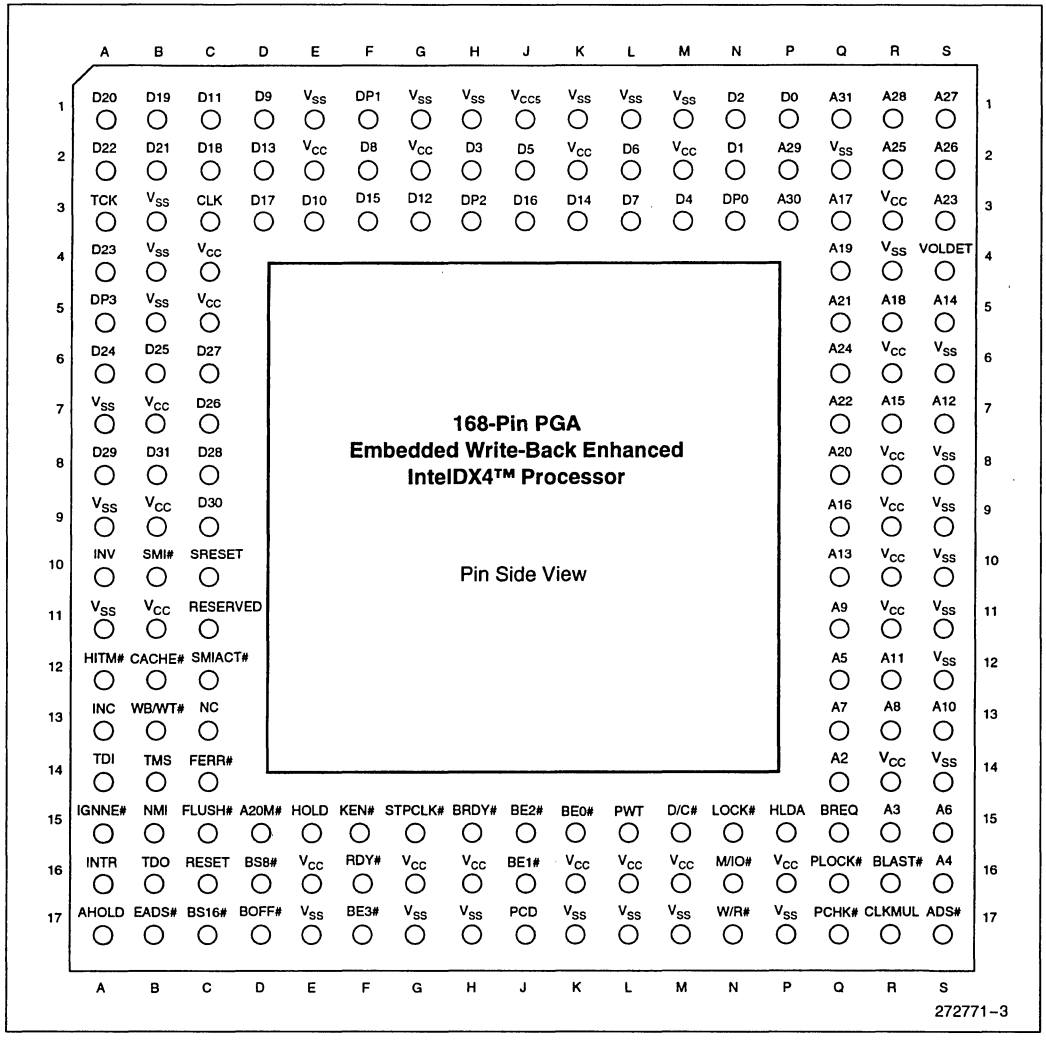


Figure 3. Package Diagram for 168-Pin PGA Embedded Write-Back Enhanced IntelDX4™ Processor

Table 5. Pinout Differences for 168-Pin PGA Package

Pin #	Embedded IntelDX2™ Processor	Embedded Write-Back Enhanced IntelDX4™ Processor
A10	INC	INV
A12	INC	HITM#
B12	INC	CACHE#
B13	INC	WB/WT#
J1	V_{CC}	V_{CC5}
R17	INC	CLKMUL
S4	NC	VOLDET

Table 6. Pin Assignment for 168-Pin PGA Package (Sheet 1 of 2)

Pin #	Description	Pin #	Description	Pin #	Description
A1	D20	D17	BOFF#	P2	A29
A2	D22	E1	V_{SS}	P3	A30
A3	TCK	E2	V_{CC}	P15	HLDA
A4	D23	E3	D10	P16	V_{CC}
A5	DP3	E15	HOLD	P17	V_{SS}
A6	D24	E16	V_{CC}	Q1	A31
A7	V_{SS}	E17	V_{SS}	Q2	V_{SS}
A8	D29	F1	DP1	Q3	A17
A9	V_{SS}	F2	D8	Q4	A19
A10	INV	F3	D15	Q5	A21
A11	V_{SS}	F15	KEN#	Q6	A24
A12	HITM#	F16	RDY#	Q7	A22
A13	INC	F17	BE3#	Q8	A20
A14	TDI	G1	V_{SS}	Q9	A16
A15	IGNNE#	G2	V_{CC}	Q10	A13
A16	INTR	G3	D12	Q11	A9
A17	AHOLD	G15	STPCLK#	Q12	A5
B1	D19	G16	V_{CC}	Q13	A7
B2	D21	G17	V_{SS}	Q14	A2
B3	V_{SS}	H1	V_{SS}	Q15	BREQ
B4	V_{SS}	H2	D3	Q16	PLOCK#
B5	V_{SS}	H3	DP2	Q17	PCHK#
B6	D25	H15	BRDY#	R1	A28
B7	V_{CC}	H16	V_{CC}	R2	A25
B8	D31	H17	V_{SS}	R3	V_{CC}
B9	V_{CC}	J1	V_{CC5}	R4	V_{SS}
B10	SMI#	J2	D5	R5	A18
B11	V_{CC}	J3	D16	R6	V_{CC}
B12	CACHE#	J15	BE2#	R7	A15
B13	WB/WT#	J16	BE1#	R8	V_{CC}
B14	TMS	J17	PCD	R9	V_{CC}
B15	NMI	K1	V_{SS}	R10	V_{CC}
B16	TDO	K2	V_{CC}	R11	V_{CC}
B17	EADS#	K3	D14	R12	A11
C1	D11	K15	BE0#	R13	A8
C2	D18	K16	V_{CC}	R14	V_{CC}
C3	CLK	K17	V_{SS}	R15	A3

Table 6. Pin Assignment for 168-Pin PGA Package (Sheet 2 of 2)

Pin #	Description	Pin #	Description	Pin #	Description
C4	V_{CC}	L1	V_{SS}	R16	BLAST#
C5	V_{CC}	L2	D6	R17	CLKMUL
C6	D27	L3	D7	S1	A27
C7	D26	L15	PWT	S2	A26
C8	D28	L16	V_{CC}	S3	A23
C9	D30	L17	V_{SS}	S4	VOLDET
C10	SRESET	M1	V_{SS}	S5	A14
C11	RESERVED	M2	V_{CC}	S6	V_{SS}
C12	SMIACT#	M3	D4	S7	A12
C13	NC	M15	D/C#	S8	V_{SS}
C14	FERR#	M16	V_{CC}	S9	V_{SS}
C15	FLUSH#	M17	V_{SS}	S10	V_{SS}
C16	RESET	N1	D2	S11	V_{SS}
C17	BS16#	N2	D1	S12	V_{SS}
D1	D9	N3	DP0	S13	A10
D2	D13	N15	LOCK#	S14	V_{SS}
D3	D17	N16	M/IO#	S15	A6
D15	A20M#	N17	W/R#	S16	A4
D16	BS8#	P1	D0	S17	ADS#

Table 7. Pin Cross Reference for 168-Pin PGA Package (Sheet 1 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC5}	V_{CC}	V_{SS}
A2	Q14	D0	P1	A20M#	D15	C13	A13	J1	B7	A7
A3	R15	D1	N2	ADS#	S17				B9	A9
A4	S16	D2	N1	AHOLD	A17				B11	A11
A5	Q12	D3	H2	BE0#	K15				C4	B3
A6	S15	D4	M3	BE1#	J16				C5	B4
A7	Q13	D5	J2	BE2#	J15				E2	B5
A8	R13	D6	L2	BE3#	F17				E16	E1
A9	Q11	D7	L3	BLAST#	R16				G2	E17
A10	S13	D8	F2	BOFF#	D17				G16	G1
A11	R12	D9	D1	BRDY#	H15				H16	G17
A12	S7	D10	E3	BREQ	Q15				K2	H1
A13	Q10	D11	C1	BS16#	C17				K16	H17
A14	J5	D12	G3	BS8#	D16				L16	K1
A15	R7	D13	D2	CLK	C3				M2	K17
A16	Q9	D14	K3	CLKMUL	R14				M16	L1
A17	Q3	D15	F3	CACHE#	B12				P16	L17
A18	R5	D16	J3	D/C#	M15				R3	M1
A19	Q4	D17	D3	DP0	N3				R6	M17
A20	Q8	D18	C2	DP1	F1				R8	P17
A21	Q5	D19	B1	DP2	H3				R9	Q2
A22	Q7	D20	A1	DP3	A5				R10	R4
A23	S3	D21	B2	EADS#	B17				R11	S6
A24	Q6	D22	A2	FERR#	C14				R14	S8
A25	R2	D23	A4	FLUSH#	C15					S9
A26	S2	D24	A6	HITM#	A12					S10
A27	S1	D25	B6	HLDA	P15					S11
A28	R1	D26	C7	HOLD	E15					S12
A29	P2	D27	C6	IGNNE#	A15					S14
A30	P3	D28	C8	INTR	A16					
A31	Q1	D29	A8	INV	A10					
		D30	C9	KEN#	F15					
		D31	B8	LOCK#	N15					
				M/IO#	N16					
				NMI	B15					

Table 7. Pin Cross Reference for 168-Pin PGA Package (Sheet 2 of 2)

Address	Pin #	Data	Pin #	Control	Pin #	NC	INC	V_{CC5}	V_{CC}	V_{SS}
				PCD	J17					
				PCHK#	Q17					
				PLOCK#	Q16					
				PWT	L15					
				RDY#	F16					
				RESERVED	C11					
				RESET	C16					
				SMI#	B10					
				SMIACT#	C12					
				SRESET	C10					
				STPCLK#	G15					
				TCK	A3					
				TDI	A14					
				TDO	B16					
				TMS	B14					
				VOLDET	S4					
				WB/WT#	B13					
				W/R#	N17					

3.2 Pin Quick Reference

The following is a brief pin description. For detailed signal descriptions refer to "Signal Description" in section 9 of the *Intel486™ Processor Family* datasheet.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 1 of 8)

Symbol	Type	Name and Function
CLK	I	**Clock** provides the fundamental timing and internal operating frequency for the embedded Write-Back Enhanced IntelDX4 processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 **A3–A2**	I/O O	**Address Lines** A31–A2, together with the byte enable signals, BE3#–BE0#, define the physical area of memory or input/output space accessed. Address lines A31–A4 are used to drive addresses into the embedded Write-Back Enhanced IntelDX4 processor to perform cache line invalidation. Input signals must meet setup and hold times t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.
BE3# **BE2#** **BE1#** **BE0#**	O O O O	**Byte Enable** signals indicate active bytes during read and write cycles. During the first cycle of a cache fill, the external system should assume that all byte enables are active. BE3#–BE0# are active LOW and are not driven during bus hold. BE3# applies to D31–D24 BE2# applies to D23–D16 BE1# applies to D15–D8 BE0# applies to D7–D0
DATA BUS		
D31–D0	I/O	**Data Lines.** D7–D0 define the least significant byte of the data bus; D31–D24 define the most significant byte of the data bus. These signals must meet setup and hold times t_{22} and t_{23} for proper operation on reads. These pins are driven during the second and subsequent clocks of write cycles.
DATA PARITY		
DP3–DP0	I/O	There is one **Data Parity** pin for each byte of the data bus. Data parity is generated on all write data cycles with the same timing as the data driven by the embedded Write-Back Enhanced IntelDX4 processor. Even parity information must be driven back into the processor on the data parity pins with the same timing as read information to ensure that the correct parity check status is indicated by the embedded Write-Back Enhanced IntelDX4 processor. The signals read on these pins do not affect program execution. Input signals must meet setup and hold times t_{22} and t_{23}. DP3–DP0 must be connected to V_{CC} through a pull-up resistor in systems that do not use parity. DP3–DP0 are active HIGH and are driven during the second and subsequent clocks of write cycles.
PCHK#	O	**Parity Status** is driven on the PCHK# pin the clock after ready for read operations. The parity status is for data sampled at the end of the previous clock. A parity error is indicated by PCHK# being LOW. Parity status is only checked for enabled bytes as indicated by the byte enable and bus size signals. PCHK# is valid only in the clock immediately after read data is returned to the processor. At all other times PCHK# is inactive (HIGH). PCHK# is never floated.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 2 of 8)

Symbol	Type	Name and Function
BUS CYCLE DEFINITION		
M/IO# **D/C#** **W/R#**	O O O	**Memory/Input-Output, Data/Control** and **Write/Read** lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted. **M/IO# D/C# W/R# Bus Cycle Initiated** 0 0 0 Interrupt Acknowledge 0 0 1 HALT/Special Cycle (see details below) 0 1 0 I/O Read 0 1 1 I/O Write 1 0 0 Code Read 1 0 1 Reserved 1 1 0 Memory Read 1 1 1 Memory Write **HALT/Special Cycle** **Cycle Name BE3#-BE0# A4-A2** Shutdown 1110 000 HALT 1011 000 Stop Grant bus cycle 1011 100
LOCK#	O	**Bus Lock** indicates that the current bus cycle is locked. The embedded Write-Back Enhanced IntelDX4 processor does not allow a bus hold when LOCK# is asserted (address holds are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the last clock of the last locked bus cycle. The last locked cycle ends when Ready is returned. LOCK# is active LOW and not driven during bus hold. Locked read cycles are not transformed into cache fill cycles when KEN# is returned active.
PLOCK#	O	**Pseudo-Lock** indicates that the current bus transaction requires more than one bus cycle to complete. For the embedded Write-Back Enhanced IntelDX4 processor, examples of such operations are segment table descriptor reads (64 bits) and cache line fills (128 bits). For Intel486 processors with on-chip Floating-Point Unit, floating-point long reads and writes (64 bits) also require more than one bus cycle to complete. The embedded Write-Back Enhanced IntelDX4 processor drives PLOCK# active until the addresses for the last bus cycle of the transaction are driven, regardless of whether RDY# or BRDY# have been returned. Normally PLOCK# and BLAST# are inverse of each other. However, during the first bus cycle of a 64-bit floating-point write (for Intel486 processors with on-chip Floating-Point Unit) both PLOCK# and BLAST# are asserted. PLOCK# is a function of the BS8#, BS16# and KEN# inputs. PLOCK# should be sampled only in the clock in which Ready is returned. PLOCK# is active LOW and is not driven during bus hold.
BUS CONTROL		
ADS#	O	**Address Status** output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus. ADS# is driven active in the same clock in which the addresses are driven. ADS# is active LOW and not driven during bus hold.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 3 of 8)

Symbol	Type	Name and Function
RDY#	I	**Non-burst Ready** input indicates that the current bus cycle is complete. RDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted data from the embedded Write-Back Enhanced IntelDX4 processor in response to a write. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. RDY# is active during address hold. Data can be returned to the embedded Write-Back Enhanced IntelDX4 processor while AHOLD is active. RDY# is active LOW and is not provided with an internal pull-up resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.
BURST CONTROL		
BRDY#	I	**Burst Ready** input performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data in response to a read or that the external system has accepted data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle. BRDY# is sampled in the second and subsequent clocks of a burst cycle. Data presented on the data bus is strobed into the embedded Write-Back Enhanced IntelDX4 processor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. BRDY# is active LOW and is provided with a small pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.
BLAST#	O	**Burst Last** signal indicates that the next time BRDY# is returned, the burst bus cycle is complete. BLAST# is active for both burst and non-burst bus cycles. BLAST# is active LOW and is not driven during bus hold.
INTERRUPTS		
RESET	I	**Reset** input forces the embedded Write-Back Enhanced IntelDX4 processor to begin execution at a known state. The processor cannot begin executing instructions until at least 1 ms after V_{CC}, and CLK have reached their proper DC and AC specifications. The RESET pin must remain active during this time to ensure proper processor operation. However, for warm resets, RESET should remain active for at least 15 CLK periods. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
INTR	I	**Maskable Interrupt** indicates that an external interrupt has been generated. When the internal interrupt flag is set in EFLAGS, active interrupt processing is initiated. The embedded Write-Back Enhanced IntelDX4 processor generates two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the interrupt acknowledges have been performed to ensure processor recognition of the interrupt. INTR is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 4 of 8)

Symbol	Type	Name and Function
NMI	I	**Non-Maskable Interrupt** request signal indicates that an external non-maskable interrupt has been generated. NMI is rising-edge sensitive and must be held LOW for at least four CLK periods before this rising edge. NMI is not provided with an internal pull-down resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SRESET	I	**Soft Reset** pin duplicates all functionality of the RESET pin except that the SMBASE register retains its previous value. For soft resets, SRESET must remain active for at least 15 CLK periods. SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SMI#	I	**System Management Interrupt** input invokes System Management Mode (SMM). SMI# is a falling-edge triggered signal which forces the embedded Write-Back Enhanced IntelDX4 processor into SMM at the completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# does not break LOCKed bus cycles and cannot interrupt a currently executing SMM. The embedded Write-Back Enhanced IntelDX4 processor latches the falling edge of one pending SMI# signal while it is executing an existing SMI#. The nested SMI# is not recognized until after the execution of a Resume (RSM) instruction.
SMIACT#	O	**System Management Interrupt Active,** an active LOW output, indicates that the embedded Write-Back Enhanced IntelDX4 processor is operating in SMM. It is asserted when the processor begins to execute the SMI# state save sequence and remains active LOW until the processor executes the last state restore cycle out of SMRAM.
STPCLK#	I	**Stop Clock Request** input signal indicates a request was made to turn off or change the CLK input frequency. When the embedded Write-Back Enhanced IntelDX4 processor recognizes a STPCLK#, it stops execution on the next instruction boundary (unless superseded by a higher priority interrupt), empties all internal pipelines and write buffers, and generates a Stop Grant bus cycle. **STPCLK# is active LOW. STPCLK# is an asynchronous signal, but must remain active until the embedded Write-Back Enhanced IntelDX4 processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle.**
BUS ARBITRATION		
BREQ	O	**Bus Request** signal indicates that the embedded Write-Back Enhanced IntelDX4 processor has internally generated a bus request. BREQ is generated whether or not the processor is driving the bus. BREQ is active HIGH and is never floated.
HOLD	I	**Bus Hold Request** allows another bus master complete control of the embedded Write-Back Enhanced IntelDX4 processor bus. In response to HOLD going active, the processor floats most of its output and input/output pins. HLDA is asserted after completing the current bus cycle, burst cycle or sequence of locked cycles. The embedded Write-Back Enhanced IntelDX4 processor remains in this state until HOLD is de-asserted. HOLD is active HIGH and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper operation.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 5 of 8)

Symbol	Type	Name and Function
HLDA	O	**Hold Acknowledge** goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the embedded Write-Back Enhanced IntelDX4 processor has given the bus to another local bus master. HLDA is driven active in the same clock that the processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.
BOFF#	I	**Backoff** input forces the embedded Write-Back Enhanced IntelDX4 processor to float its bus in the next clock. The processor floats all pins normally floated during bus hold but HLDA is not asserted in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#; if both are returned in the same clock, BOFF# takes effect. The embedded Write-Back Enhanced IntelDX4 processor remains in bus hold until BOFF# is negated. If a bus cycle is in progress when BOFF# is asserted the cycle is restarted. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper operation.
CACHE INVALIDATION		
AHOLD	I	**Address Hold** request allows another bus master access to the embedded Write-Back Enhanced IntelDX4 processor's address bus for a cache invalidation cycle. The processor stops driving its address bus in the clock following AHOLD going active. Only the address bus is floated during address hold, the remainder of the bus remains active. AHOLD is active HIGH and is provided with a small internal pull-down resistor. For proper operation, AHOLD must meet setup and hold times t_{18} and t_{19}.
EADS#	I	**External Address**—This signal indicates that a *valid* external address has been driven onto the embedded Write-Back Enhanced IntelDX4 processor address pins. This address is used to perform an internal cache invalidation cycle. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy setup and hold times t_{12} and t_{13} for proper operation.
CACHE CONTROL		
KEN#	I	**Cache Enable** pin is used to determine whether the current cycle is cacheable. When the embedded Write-Back Enhanced IntelDX4 processor generates a cycle that can be cached and KEN# is active one clock before RDY# or BRDY# during the first transfer of the cycle, the cycle becomes a cache line fill cycle. Returning KEN# active one clock before RDY# during the last read in the cache line fill causes the line to be placed in the on-chip cache. KEN# is active LOW and is provided with a small internal pull-up resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	**Cache Flush** input forces the embedded Write-Back Enhanced IntelDX4 processor to flush its entire internal cache. FLUSH# is active LOW and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition in any specific clock.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 6 of 8)

Symbol	Type	Name and Function
PAGE CACHEABILITY		
PWT **PCD**	O O	**Page Write-Through** and **Page Cache Disable** pins reflect the state of the page attribute bits, PWT and PCD, in the page table entry, page directory entry or control register 3 (CR3) when paging is enabled. When paging is disabled, the embedded Write-Back Enhanced IntelDX4 processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT pins. PWT and PCD have the same timing as the cycle definition pins (M/IO#, D/C#, and W/R#). PWT and PCD are active HIGH and are not driven during bus hold. PCD is masked by the cache disable bit (CD) in Control Register 0.
BUS SIZE CONTROL		
BS16# **BS8#**	I I	**Bus Size 16** and **Bus Size 8** pins (bus sizing pins) cause the embedded Write-Back Enhanced IntelDX4 processor to run multiple bus cycles to complete a request from devices that cannot provide or accept 32 bits of data in a single cycle. The bus sizing pins are sampled every clock. The processor uses the state of these pins in the clock before Ready to determine bus size. These signals are active LOW and are provided with internal pull-up resistors. These inputs must satisfy setup and hold times t_{14} and t_{15} for proper operation.
ADDRESS MASK		
A20M#	I	**Address Bit 20 Mask** pin, when asserted, causes the embedded Write-Back Enhanced IntelDX4 processor to mask physical address bit 20 (A20) before performing a lookup to the internal cache or driving a memory cycle on the bus. A20M# emulates the address wraparound at 1 Mbyte, which occurs on the 8086 processor. A20M# is active LOW and should be asserted only when the embedded Write-Back Enhanced IntelDX4 processor is in real mode. This pin is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For proper operation, A20M# should be sampled HIGH at the falling edge of RESET.
TEST ACCESS PORT		
TCK	I	**Test Clock**, an input to the embedded Write-Back Enhanced IntelDX4 processor, provides the clocking function required by the JTAG Boundary scan feature. TCK is used to clock state information (via TMS) and data (via TDI) into the component on the rising edge of TCK. Data is clocked out of the component (via TDO) on the falling edge of TCK. TCK is provided with an internal pull-up resistor.
TDI	I	**Test Data Input** is the serial input used to shift JTAG instructions and data into the processor. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and SHIFT-DR Test Access Port (TAP) controller states. During all other TAP controller states, TDI is a "don't care." TDI is provided with an internal pull-up resistor.
TDO	O	**Test Data Output** is the serial output used to shift JTAG instructions and data out of the component. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state.
TMS	I	**Test Mode Select** is decoded by the JTAG TAP to select test logic operation. TMS is sampled on the rising edge of TCK. To guarantee deterministic behavior of the TAP controller, TMS is provided with an internal pull-up resistor.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 7 of 8)

Symbol	Type	Name and Function
NUMERIC ERROR REPORTING		
FERR#	O	The **Floating Point Error** pin is driven active when a floating point error occurs. FERR# is similar to the ERROR# pin on the Intel387™ Math CoProcessor. FERR# is included for compatibility with systems using DOS type floating point error reporting. FERR# will not go active if FP errors are masked in FPU register. FERR# is active LOW, and is not floated during bus hold.
IGNNE#	I	When the **Ignore Numeric Error** pin is asserted the processor will ignore a numeric error and continue executing non-control floating point instructions, but FERR# will still be activated by the processor. When IGNNE# is de-asserted the processor will freeze on a non-control floating point instruction, if a previous floating point instruction caused an error. IGNNE# has no effect when the NE bit in control register 0 is set. IGNNE# is active LOW and is provided with a small internal pull-up resistor. IGNNE# is asynchronous but setup and hold times t_{20} and t_{21} must be met to ensure recognition on any specific clock.
WRITE-BACK ENHANCED MODE		
CACHE#	O	The **CACHE#** output indicates internal cacheability on read cycles and burst write-back on write cycles. CACHE# is asserted for cacheable reads, cacheable code fetches and write-backs. It is driven inactive for non-cacheable reads, I/O cycles, special cycles, and write-through cycles.
FLUSH#	I	**Cache FLUSH#** is an existing pin that operates differently if the processor is configured as Enhanced Bus mode (write-back). FLUSH# causes the processor to write back all modified lines and flush (invalidate) the cache. FLUSH# is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
HITM#	O	The **Hit/Miss to a Modified Line** pin is a cache coherency protocol pin that is driven only in Enhanced Bus mode. When a snoop cycle is run, HITM# indicates that the processor contains the snooped line and that the line has been modified. Assertion of HITM# implies that the line will be written back in its entirety, unless the processor is already in the process of doing a replacement write-back of the same line.
INV	I	The **Invalidation Request** pin is a cache coherency protocol pin that is used only in the Enhanced Bus mode. It is sampled by the processor on EADS#-driven snoop cycles. It is necessary to assert this pin to get the effect of the processor invalidate cycle on write-through-only lines. INV also invalidates the write-back lines. However, if the snooped line is modified, the line will be written back and then invalidated. INV must satisfy setup and hold times t_{12} and t_{13} for proper operation.
PLOCK#	O	In the Enhanced bus mode, **Pseudo-Lock Output** is always driven inactive. In this mode, a 64-bit data read (caused by an FP operand access or a segment descriptor read) is treated as a multiple cycle read request, which may be a burst or a non-burst access based on whether BRDY# or RDY# is returned by the system. Because only write-back cycles (caused by snoop write-back or replacement write-back) are write burstable, a 64-bit write will be driven out as two non-burst bus cycles. BLAST# is asserted during both writes.

Table 8. Embedded Write-Back Enhanced IntelDX4™ Processor Pin Descriptions (Sheet 8 of 8)

Symbol	Type	Name and Function
SRESET	I	For the embedded Write-Back Enhanced IntelDX4 processor, **Soft RESET** operates similar to other the Intel486 processors. On SRESET, the internal SMRAM base register retains its previous value, does not flush, write-back or disable the internal cache. Because SRESET is treated as an interrupt, it is possible to have a bus cycle while SRESET is asserted. SRESET is serviced only on an instruction boundary. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
WB/WT#	I	The **Write-Back/Write-Through** pin enables Enhanced Bus mode (write-back cache). It also defines a cached line as write-through or write-back. For cache configuration, WB/WT# must be valid during RESET and be active for at least two clocks before and two clocks after RESET is de-asserted. To define write-back or write-through configuration of a line, WB/WT# is sampled in the same clock as the first RDY# or BRDY# is returned during a line fill (allocation) cycle.
CLKMUL, V_{CC5}, AND VOLDET		
CLKMUL	I	The **Clock Multiplier** input, defined during device RESET, defines the ratio of internal core frequency to external bus frequency. If sampled low, the core frequency operates at twice the external bus frequency (speed doubled mode). If driven high or left floating, speed triple mode is selected. CLKMUL has an internal pull-up speed to V_{CC} and may be left floating in designs that select speed tripled clock mode.
V_{CC5}	I	The **5V reference voltage** input is the reference voltage for the 5V-tolerant I/O buffers. This signal should be connected to +5V ±5% for use with 5V logic. If all inputs are from 3V logic, this pin should be connected to 3.3V.
VOLDET	O	A **Voltage Detect** signal allows external system logic to distinguish between a 5V Intel486 processor and the 3.3V IntelDX4 processor. This signal is active LOW for a 3.3V IntelDX4 processor. This pin is available only on the PGA version of the embedded Write-Back Enhanced IntelDX4 processor.
RESERVED PINS		
RESERVED	I	**Reserved** is reserved for future use. This pin MUST be connected to an external pull-up resistor circuit. The recommended resistor value is 10 KΩ. The pull-up resistor must be connected only to the RESERVED pin. **Do not share this resistor with other pins requiring pull-ups.**

Table 9. Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States
BREQ	HIGH			Previous State(1)
HLDA	HIGH			As per HOLD
BE3#-BE0#	LOW		•	Previous State
PWT, PCD	HIGH		•	Previous State
W/R#, M/IO#, D/C#	HIGH/LOW		•	Previous State
LOCK#	LOW		•	HIGH (inactive)
PLOCK#	LOW		•	HIGH (inactive)
ADS#	LOW		•	HIGH (inactive)
BLAST#	LOW		•	Previous State
PCHK#	LOW			Previous State
FERR#	LOW			Previous State
A3-A2	HIGH	•	•	Previous State
SMIACT#	LOW			Previous State
CACHE#	LOW	•	•	HIGH(2)
HITM#	LOW	•	•	HIGH(2)
VOLDET	LOW			LOW

NOTES:
1. The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.
2. For the case of snoop cycles (via EADS#) during Stop Grant state, CACHE# and HITM# can go active depending on the snoop hit in the internal cache.

Table 10. Input/Output Pins

Name	Active Level	Output Signal		
		Floated During Address Hold	Floated During Bus Hold	During Stop Grant and Stop Clock States
D31-D0	HIGH		•	Floated
DP3-DP0	HIGH		•	Floated
A31-A4	HIGH	•	•	Previous State

NOTE:
The term "Previous State" means that the processor maintains the logic level applied to the signal pin just before the processor entered the Stop Grant state. This conserves power by preventing the signal pin from floating.

Table 11. Test Pins

Name	Input or Output	Sampled/Driven On
TCK	Input	N/A
TDI	Input	Rising Edge of TCK
TDO	Output	Failing Edge of TCK
TMS	Input	Rising Edge of TCK

Table 12. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Pull-Up/ Pull-Down
CLK			
RESET	HIGH	Asynchronous	
SRESET	HIGH	Asynchronous	Pull-Down
HOLD	HIGH	Synchronous	
AHOLD	HIGH	Synchronous	Pull-Down
EADS#	LOW	Synchronous	Pull-Up
BOFF#	LOW	Synchronous	Pull-Up
FLUSH#	LOW	Asynchronous	Pull-Up
A20M#	LOW	Asynchronous	Pull-Up
BS16#, BS8#	LOW	Synchronous	Pull-Up
KEN#	LOW	Synchronous	Pull-Up
RDY#	LOW	Synchronous	
BRDY#	LOW	Synchronous	Pull-Up
INTR	HIGH	Asynchronous	
NMI	HIGH	Asynchronous	
IGNNE#	LOW	Asynchronous	Pull-Up
RESERVED			
SMI#	LOW	Asynchronous	Pull-Up
STPCLK#	LOW	Asynchronous	Pull-Up(1)
INV	HIGH	Synchronous	Pull-Up
WB/WT#	HIGH/LOW	Synchronous	Pull-Down
CLKMUL	HIGH		Pull-Up(1)
TCK	HIGH		Pull-Up
TDI	HIGH		Pull-Up
TMS	HIGH		Pull-Up

NOTE:
1. Even though STPCLK# and CLKMUL have internal pull-up resistors, they cannot be left floating. An external 10-KΩ pull-up resistor is needed if the STPCLK# pin is unused. CLKMUL must be driven to a valid logic level. If tied HIGH, an external 10-KΩ pull-up resistor is recommended.

4.0 ARCHITECTURAL AND FUNCTIONAL OVERVIEW

The embedded Write-Back Enhanced IntelDX4 processor architecture is essentially the same as the IntelDX4 processor. Refer to the *Intel486™ Processor Family* datasheet (242202) for a description of the IntelDX4 processor. With some minor exceptions, the following datasheet sections apply to the embedded Write-Back Enhanced IntelDX4 processor:

- Architectural Overview
- Real Mode Architecture
- Protected Mode Architecture
- On-Chip Cache
- System Management Mode (SMM) Architectures
- Hardware Interface
- Bus Operation
- Testability
- Debugging Support
- Instruction Set Summary
- Differences Between Intel486 Processors and Intel386™ Processors

Exceptions to these sections of the datasheet are:

- References to the Upgrade Power Down Mode do not apply. The embedded Write-Back Enhanced IntelDX4 processor does not have the UP# signal pin and does not support the Intel OverDrive® processor.
- The embedded Write-Back Enhanced IntelDX4 processor has one pin reserved for possible future use. This pin, an input signal, is called RESERVED and must be connected to a 10-KΩ pull-up resistor. The pull-up resistor must be connected only to the RESERVED pin. **Do not share this resistor with other pins requiring pull-ups.**

4.1 CPUID Instruction

The embedded Write-Back Enhanced IntelDX4 processor supports the CPUID instruction (see Table 13). Because not all Intel processors support the CPUID instruction, a simple test can determine if the instruction is supported. The test involves the processor's ID Flag, which is bit 21 of the EFLAGS register. If software can change the value of this flag, the CPUID instruction is available. The actual state of the ID Flag bit is irrelevant and provides no significance to the hardware. This bit is cleared (reset to zero) upon device reset (RESET or SRESET) for compatibility with Intel486 processor designs that do not support the CPUID instruction.

CPUID-instruction details are provided here for the embedded Write-Back Enhanced IntelDX4 processor. Refer to Intel Application Note AP-485 *Intel Processor Identification with the CPUID Instruction* (Order No. 241618) for a description that covers all aspects of the CPUID instruction and how it pertains to other Intel processors.

4.1.1 OPERATION OF THE CPUID INSTRUCTION

The CPUID instruction requires the software developer to pass an input parameter to the processor in the EAX register. The processor response is returned in registers EAX, EBX, EDX, and ECX.

Table 13. CPUID Instruction Description

OP CODE	Instruction	Processor Core Clocks	Parameter passed in EAX (Input Value)	Description
0F A2	CPUID	9	0	Vendor (Intel) ID String
		14	1	Processor Identification
		9	> 1	Undefined (Do Not Use)

Vendor ID String—When the parameter passed in EAX is 0 (zero), the register values returned upon instruction execution are shown in the following table.

		31 – – – – – – 24	23 – – – – – – 16	15 – – – – – – 8	7 – – – – – – 0
High Value (= 1)	**EAX**	0000	0000	0000	0001
Vendor ID String	**EBX**	u (75)	n (6E)	e (65)	G (47)
(ASCII	**EDX**	I (49)	e (65)	n (6E)	i (69)
Characters)	**ECX**	l (6C)	e (65)	t (74)	n (6E)

The values in EBX, EDX and ECX indicate an Intel processor. When taken in the proper order, they decode to the string "GenuineIntel."

The state of the WB/WT# input pin is sampled by the processor on the falling edge of the RESET signal. If WB/WT# is LOW, the processor is configured to operate in Write-Through/Standard Bus mode. If HIGH, it is configured to operate in Write-Back/Enhanced Bus mode. The value of the "Model" field of the processor signature register depends on the bus mode for which the processor is configured.

Processor Identification—When the parameter passed to EAX is 1 (one), the register values returned upon instruction execution are:

		31 – – – – – – – – 14	13, 12	11 – – – 8	7 – – – 4	3 – – – 0
Processor Signature for Write-Through/Standard Bus mode	**EAX**	(Do Not Use) Intel Reserved	00 Processor Type	0100 Family	1000 Model	XXXX Stepping
Processor Signature for WriteBack/Enhanced Bus mode		(Do Not Use) Intel Reserved	00 Processor Type	0100 Family	1001 Model	XXXX Stepping

(Intel releases information about stepping numbers as needed)

		31 – 0
Intel Reserved	**EBX**	Intel Reserved
(Do Not Use)	**ECX**	Intel Reserved

		31 – 2	1	0
Feature Flags	**EDX**	0 – 0	1 VME	0 FPU

4.2 Identification After Reset

Processor Identification—Upon reset, the EDX register contains the processor signature:

		31 – – – – – – – – 14	13, 12	11 – – – 8	7 – – – 4	3 – – – 0
Processor Signature for Write-Through/Standard Bus mode	**EDX**	(Do Not Use) Intel Reserved	0 0 Processor Type	0 1 0 0 Family	1 0 0 0 Model	XXXX Stepping
Processor Signature for WriteBack/Enhanced Bus mode		(Do Not Use) Intel Reserved	0 0 Processor Type	0 1 0 0 Family	1 0 0 1 Model	XXXX Stepping

(Intel releases information about stepping numbers as needed)

4.3 Boundary Scan (JTAG)

4.3.1 Device Identification

Tables 14 and 15 show the 32-bit code for the embedded Write-Back Enhanced IntelDX4 processor. This code is loaded into the Device Identification Register.

Table 14. Boundary Scan Component Identification Code (Write-Through/Standard Bus Mode)

Version	Part Number				Mfg ID	1
	V_{CC} 1 = 3.3V	Intel Architecture Type	Family 0100 = Intel486 CPU Family	Model 00010 = embedded Write-Back Enhanced IntelDX4 processor	009H = Intel	
31 – – – 28	27	26 – – – – 21	20 – – – – 17	16 – – – – 12	11 – – – – 1	0
XXXX	1	000001	0100	01000	00000001001	1

(Intel releases information about version numbers as needed)
Boundary Scan Component Identification Code = x828 8013 (Hex)

Table 15. Boundary Scan Component Identification Code (Write-Back/Enhanced Bus Mode)

Version	Part Number				Mfg ID	1
	V_{CC} 1 = 3.3V	Intel Architecture Type	Family 0100 = Intel486 CPU Family	Model 01001 = embedded Write-Back Enhanced IntelDX4 processor	009H = Intel	
31 – – – 28	27	26 – – – – 21	20 – – – – 17	16 – – – – 12	11 – – – – 1	0
XXXX	1	000001	0100	01001	00000001001	1

(Intel releases information about version numbers as needed)
Boundary Scan Component Identification Code = x828 9013 (Hex)

4.3.2 Boundary Scan Register Bits and Bit Order

The boundary scan register contains a cell for each pin as well as cells for control of bidirectional and three-state pins. There are "Reserved" bits which correspond to no-connect (N/C) signals of the embedded Write-Back Enhanced IntelDX4 processor. Control registers WRCTL, ABUSCTL, BUSCTL, and MISCCTL are used to select the direction of bidirectional or three-state output signal pins. A "1" in these cells designates that the associated bus or bits are floated if the pins are three-state, or selected as input if they are bidirectional.

- WRCTL controls D31-D0 and DP3–DP0
- ABUSCTL controls A31-A2
- BUSCTL controls ADS#, BLAST#, PLOCK#, LOCK#, W/R#, BE0#, BE1#, BE2#, BE3#, M/IO#, D/C#, PWT, PCD, and CACHE#
- MISCCTL controls PCHK#, HLDA, BREQ, and HITM#

The following is the bit order of the embedded Write-Back Enhanced IntelDX4 processor boundary scan register:

TDO ← A2, A3, A4, A5, RESERVED, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, IGNNE#, INV, CACHE#, FERR#, SMI#, WB/WT#, HITM#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, AHOLD, HOLD, KEN#, RDY#, CLKMUL, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRCTL **← TDI**

5.0 ELECTRICAL SPECIFICATIONS

5.1 Maximum Ratings

Table 16 is a stress rating only. Extended exposure to the Maximum Ratings may affect device reliability.

Furthermore, although the embedded Write-Back Enhanced IntelDX4 processor contains protective circuitry to resist damage from electrostatic discharge, always take precautions to avoid high static voltages or electric fields.

Functional operating conditions are given in **Section 5.2, DC Specifications** and **Section 5.3, AC Specifications.**

Table 16. Absolute Maximum Ratings

Case Temperature under Bias	−65 °C to +110 °C
Storage Temperature	−65 °C to +150 °C
DC Voltage on Any Pin with Respect to Ground	−0.5V to V_{CC5} + 0.5V
Supply Voltage V_{CC} with Respect to V_{SS}	−0.5V to +4.6V
Reference Voltage V_{CC5} with Respect to V_{SS}	−0.5V to +6.5V
Transient Voltage on any Input	The lesser of: V_{CC5} + 1.6V or 6.5V
Current Sink on V_{CC5}	55 mA

5.2 DC Specifications

The following tables show the operating supply voltages, DC I/O specifications, and component power consumption for the embedded Write-Back Enhanced IntelDX4 processor.

Table 17. Operating Supply Voltages

Product	V_{CC}
FC80486DX4WB75	3.3V ± 0.3V
FC80486DX4WB100	3.3V ± 0.3V
A80486DX4WB100	3.3V ± 0.3V

4

Table 18. DC Specifications

Functional Operating Range: V_{CC} = 3.3V ± 0.3V; V_{CC5} = 5V ± 0.25V (Note 1); T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Typ	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3		+0.8	V	
V_{IH}	Input HIGH Voltage	2.0		V_{CC5} +0.3	V	Note 2
V_{IHC}	Input HIGH Voltage of CLK	V_{CC5} −0.6		V_{CC5} +0.3	V	
V_{OL}	Output LOW Voltage I_{OL} = 4.0 mA (Address, Data, BE*n*) I_{OL} = 5.0 mA (Definition, Control) I_{OL} = 2.0 mA I_{OL} = 100 µA			 0.45 0.45 0.40 0.20	 V V V V	
V_{OH}	Output HIGH Voltage I_{OH} = −2.0 mA	 2.4			 V	
I_{CC5}	V_{CC5} Leakage Current		15	300	µA	Note 3
I_{LI}	Input Leakage Current			±15	µA	Note 4
I_{IH}	Input Leakage Current SRESET			200 300	µA µA	Note 5 Note 5
I_{IL}	Input Leakage Current			−400	µA	Note 6
I_{LO}	Output Leakage Current			±15	µA	
C_{IN}	Input Capacitance			10	pF	Note 7
C_{OUT}	I/O or Output Capacitance			14	pF	Note 7
C_{CLK}	CLK Capacitance			12	pF	Note 7

NOTES:
1. V_{CC5} should be connected to 3.3V ± 0.3V in 3.3V-only systems.
2. All inputs except CLK.
3. This parameter is for inputs without pull-up or pull-down resistors and 0V ≤ VIN ≤ V_{CC}.
4. This parameter is for V_{CC5} − V_{CC} ≤ 2.25V. Typical value is not 100% tested.
5. This parameter is for inputs with pull-down resistors and V_{IH} = 2.4V.
6. This parameter is for inputs with pull-up resistors and V_{IL} = 0.4V.
7. F_C = 1 MHz. Not 100% tested.

Table 19. I_{CC} Values

Functional Operating Range: V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ± 0.25V (Note 1); T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	75 MHz 100 MHz		1100 mA 1450 mA	Note 2
I_{CC} Active (Thermal Design)	75 MHz 100 MHz	825 mA 1075 mA	975 mA 1300 mA	Notes 3, 4, 5
I_{CC} Stop Grant	75 MHz 100 MHz	20 mA 50 mA	75 mA 100 mA	Note 6
I_{CC} Stop Clock	0 MHz	600 µA	1 mA	Note 7

NOTES:
1. V_{CC5} should be connected to 3.3V ± 0.3V in 3.3V-only systems.
2. This parameter is for proper power supply selection. It is measured using the worst case instruction mix at V_{CC} = 3.6V.
3. The maximum current column is for thermal design power dissipation. It is measured using the worst case instruction mix at V_{CC} = 3.3V.
4. The typical current column is the typical operating current in a system. This value is measured in a system using a typical device at V_{CC} = 3.3V, running Microsoft Windows 3.1 at an idle condition. This typical value is dependent upon the specific system configuration.
5. Typical values are not 100% tested.
6. The I_{CC} Stop Grant specification refers to the I_{CC} value once the embedded Write-Back Enhanced IntelDX4 processor enters the Stop Grant or Auto HALT Power Down state.
7. The I_{CC} Stop Clock specification refers to the I_{CC} value once the embedded Write-Back Enhanced IntelDX4 processor enters the Stop Clock state. The V_{IH} and V_{IL} levels must be equal to V_{CC} and 0V, respectively, in order to meet the I_{CC} Stop Clock specifications.

5.3 AC Specifications

The AC specifications for the embedded Write-Back Enhanced IntelDX4 processor are given in this section.

Table 20. AC Characteristics

V_{CC} = 3.3V ± 0.3V; V_{CC5} = 5V ± 0.25V (Note 1)
T_{CASE} = 0°C to +85°C; CL = 50 pF, unless otherwise specified. (Sheet 1 of 2)

Symbol	Parameter	Product						
		WB75		WB100				
		Min	Max	Min	Max	Unit	Figure	Notes
	CLK Frequency	8	25	8	33	MHz		Note 2
t_1	CLK Period	40	125	30	125	ns	4	
t_{1a}	CLK Period Stability		±250		±250	ps	4	Adjacent clocks Note 3
t_2	CLK High Time	14		11		ns	4	at 2V
t_3	CLK Low Time	14		11		ns	4	at 0.8V
t_4	CLK Fall Time		4		3	ns	4	2V to 0.8V
t_5	CLK Rise Time		4		3	ns	4	0.8V to 2V
t_6	A31–A2, PWT, PCD, BE3–BE0#, M/IO#, D/C#, W/R#, ADS#, LOCK#, FERR#, CACHE#, HITM#, BREQ, HLDA Valid Delay	3	19	3	14	ns	8	
t_7	A31–A2, PWT, PCD, BE3–BE0#, M/IO#, D/C#, W/R#, ADS#, LOCK#, CACHE# Float Delay		28		20	ns	9	Note 3
t_8	PCHK# Valid Delay	3	24	3	14	ns	7	
t_{8a}	BLAST#, PLOCK#, SMIACT# Valid Delay	3	24	3	14	ns	8	
t_9	BLAST#, PLOCK# Float Delay		28		20	ns	9	Note 3
t_{10}	D31–D0, DP3–DP0 Write Data Valid Delay	3	20	3	14	ns	8	
t_{11}	D31–D0, DP3–DP0 Write Data Float Delay		28		20	ns	9	Note 3
t_{12}	EADS#, INV Setup Time	8		5		ns	5	
t_{13}	EADS#, INV Hold Time	3		3		ns	5	
t_{14}	KEN#, BS16#, BS8#, WB/WT# Setup Time	8		5		ns	5	
t_{15}	KEN#, BS16#, BS8#, WB/WT# Hold Time	3		3		ns	5	
t_{16}	RDY#, BRDY# Setup Time	8		5		ns	6	

Table 20. AC Characteristics

V_{CC} = 3.3V ± 0.3V; V_{CC5} = 5V ± 0.25V (Note 1)
T_{CASE} = 0°C to +85°C; CL = 50 pF, unless otherwise specified. (Sheet 2 of 2)

Symbol	Parameter	Product						
		WB75		WB100				
		Min	Max	Min	Max	Unit	Figure	Notes
t_{17}	RDY#, BRDY# Hold Time	3		3		ns	6	
t_{18}	HOLD, AHOLD Setup Time	8		6		ns	5	
t_{18a}	BOFF# Setup Time	8		7		ns	5	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		3		ns	5	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Setup Time	8		5		ns	5	Note 4
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Hold Time	3		3		ns	5	Note 4
t_{22}	D31–D0, DP3–DP0, A31–A4 Read Setup Time	5		5		ns	6 5	
t_{23}	D31–D0, DP3–DP0, A31–A4 Read Hold Time	3		3		ns	6 5	

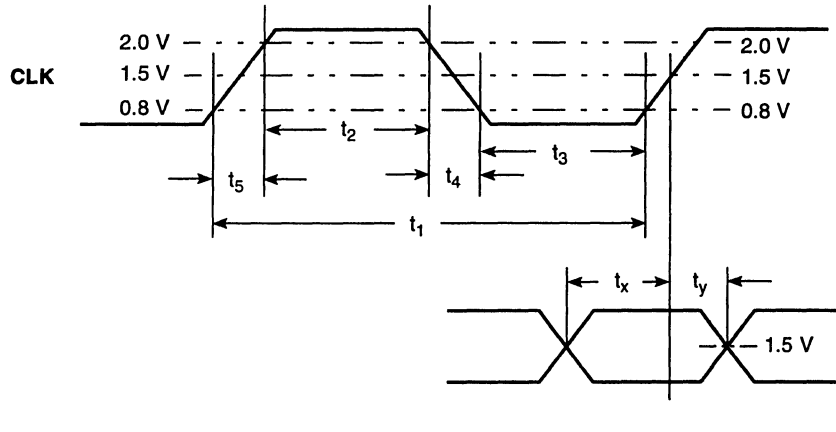
NOTES:
1. V_{CC5} should be connected to 3.3V ± 0.3V in 3.3V-only systems.
2. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant bus cycle protocol is used.
3. Not 100% tested, guaranteed by design characterization.
4. A reset pulse width of 15 CLK cycles is required for warm resets (RESET or SRESET). Power-up resets (cold resets) require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.

Table 21. AC Specifications for the Test Access Port

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1)
T_{CASE} = 0°C to +85°C; CL = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{24}	TCK Frequency		25	MHz		Note 2
t_{25}	TCK Period	40		ns	10	
t_{26}	TCK High Time	10		ns	10	@ 2.0V
t_{27}	TCK Low Time	10		ns	10	@ 0.8V
t_{28}	TCK Rise Time		4	ns	10	Note 3
t_{29}	TCK Fall Time		4	ns	10	Note 3
t_{30}	TDI, TMS Setup Time	8		ns	11	Note 4
t_{31}	TDI, TMS Hold Time	7		ns	11	Note 4
t_{32}	TDO Valid Delay	3	25	ns	11	Note 4
t_{33}	TDO Float Delay		30	ns	11	Note 4
t_{34}	All Outputs (except TDO) Valid Delay	3	25	ns	11	Note 4
t_{35}	All Outputs (except TDO) Float Delay		36	ns	11	Note 4
t_{36}	All Inputs (except TDI, TMS, TCK) Setup Time	8		ns	11	Note 4
t_{37}	All Inputs (except TDI, TMS, TCK) Hold Time	7		ns	11	Note 4

NOTES:
1. V_{CC5} should be connected to 3.3V ±0.3V in 3.3V-only systems. All inputs and outputs are TTL level.
2. TCK period ≤ CLK period.
3. Rise/Fall times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10-ns increase in TCK period.
4. Parameters t_{30}–t_{37} are measured from TCK.



t_x = input setup times
t_y = input hold times, output float, valid and hold times

Figure 4. CLK Waveform

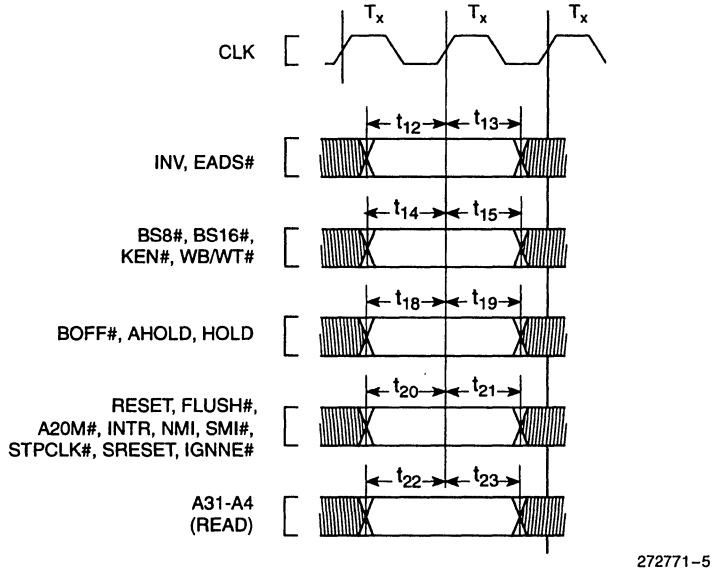


Figure 5. Input Setup and Hold Timing

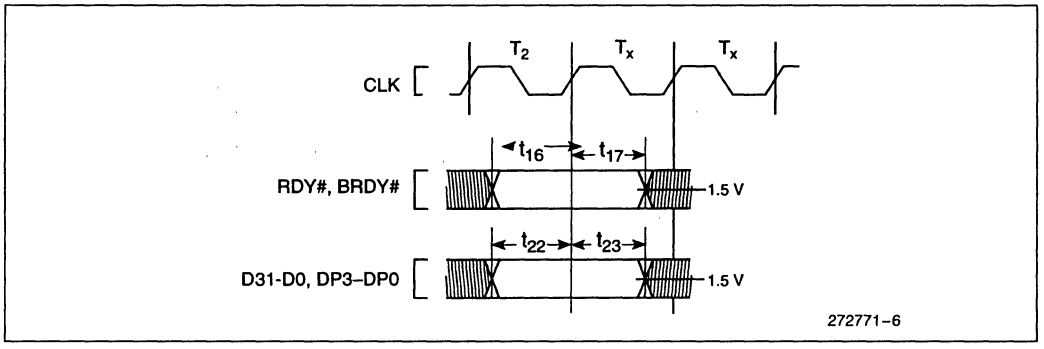


Figure 6. Input Setup and Hold Timing

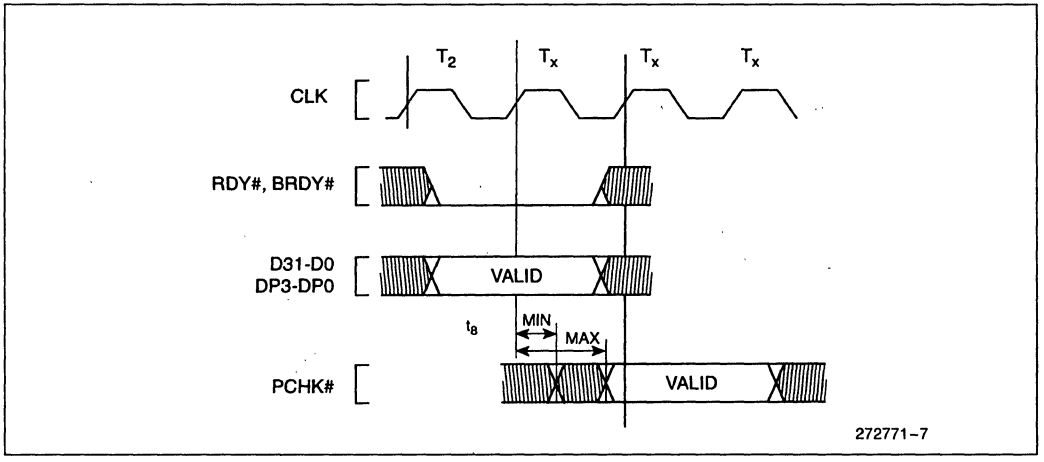


Figure 7. PCHK# Valid Delay Timing

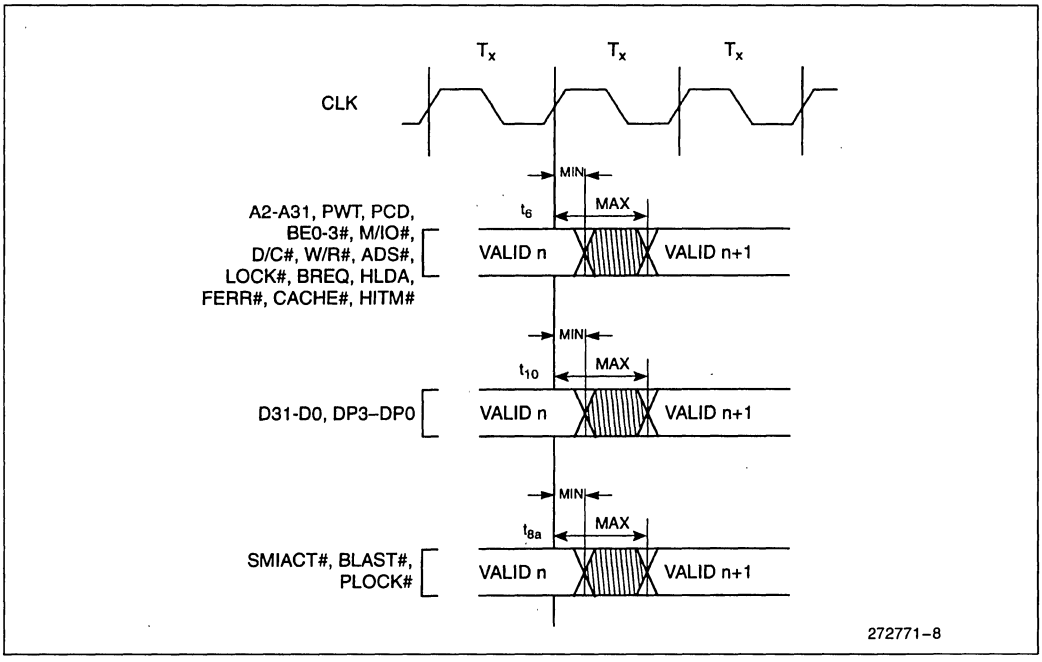


Figure 8. Output Valid Delay Timing

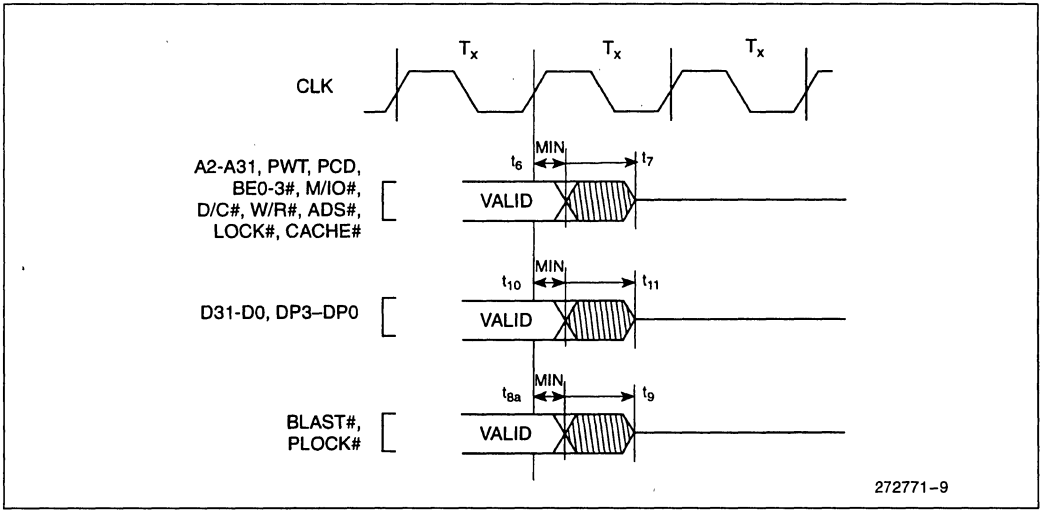


Figure 9. Maximum Float Delay Timing

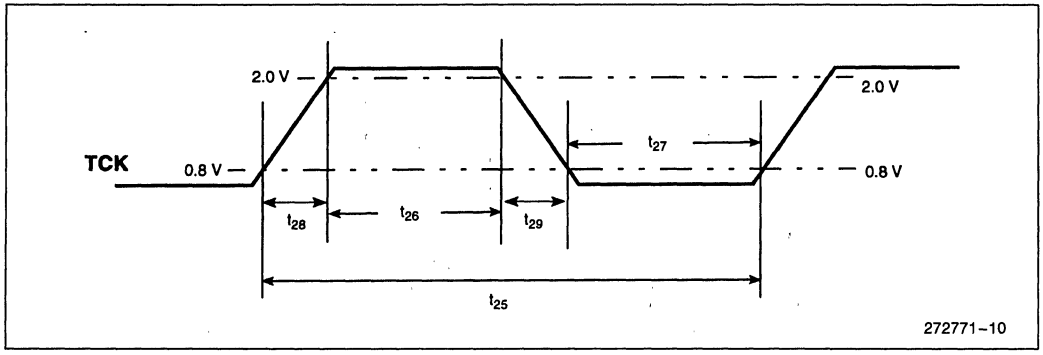


Figure 10. TCK Waveform

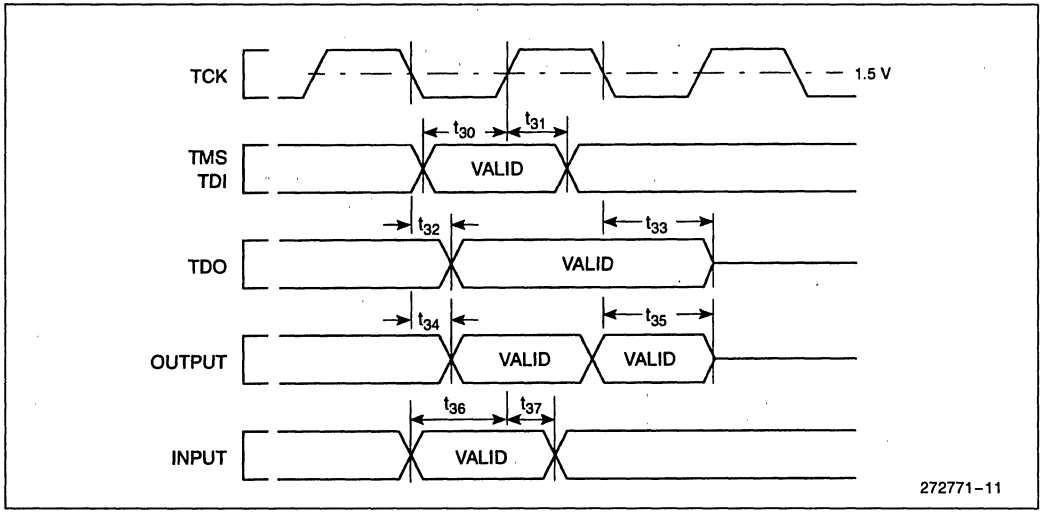
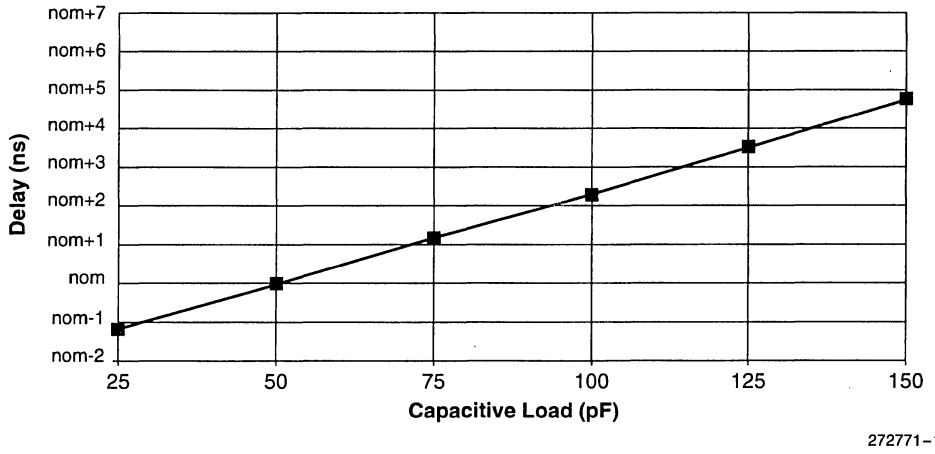


Figure 11. Test Signal Timing Diagram

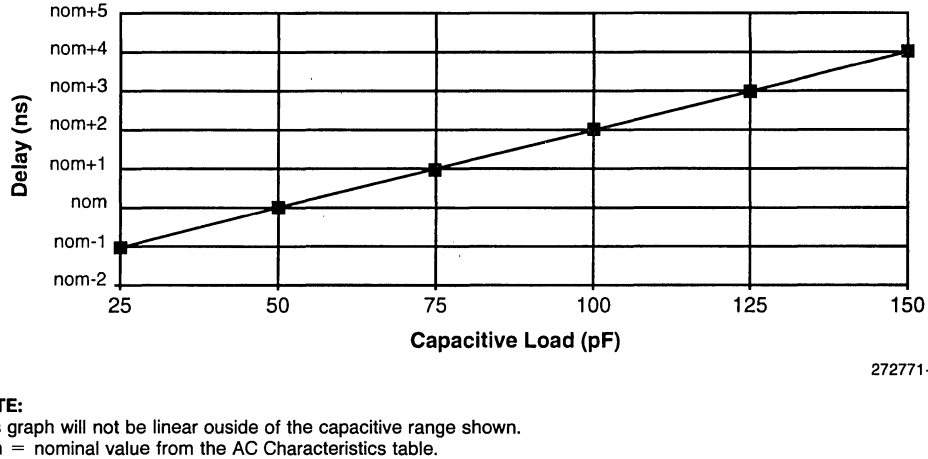
5.4 Capacitive Derating Curves

These graphs are the capacitive derating curves for the embedded Write-Back Enhanced IntelDX4 processor.



NOTE:
This graph will not be linear ouside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 12. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



NOTE:
This graph will not be linear ouside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 13. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition

6.0 MECHANICAL DATA

This section describes the packaging dimensions and thermal specifications for the embedded Write-Back Enhanced IntelDX4 processor.

6.1 Package Dimensions

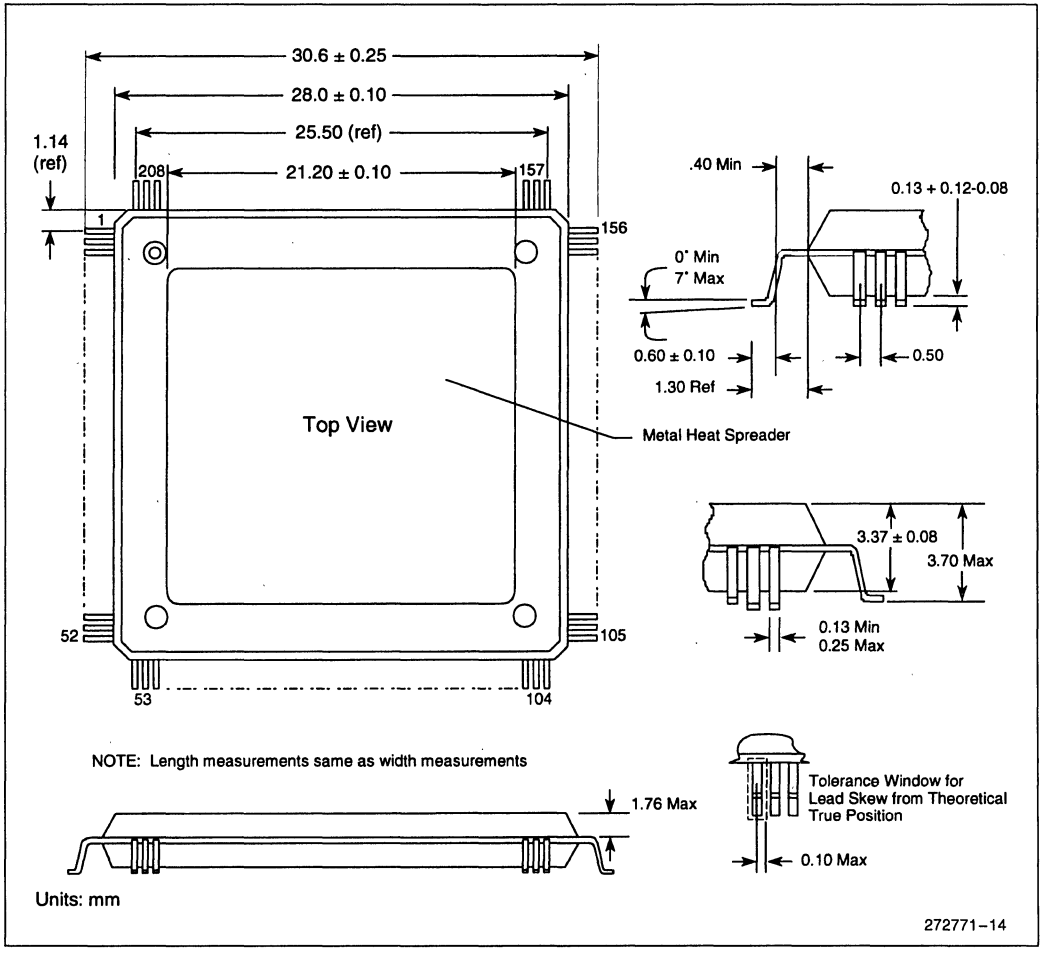


Figure 14. 208-Lead SQFP Package Dimensions

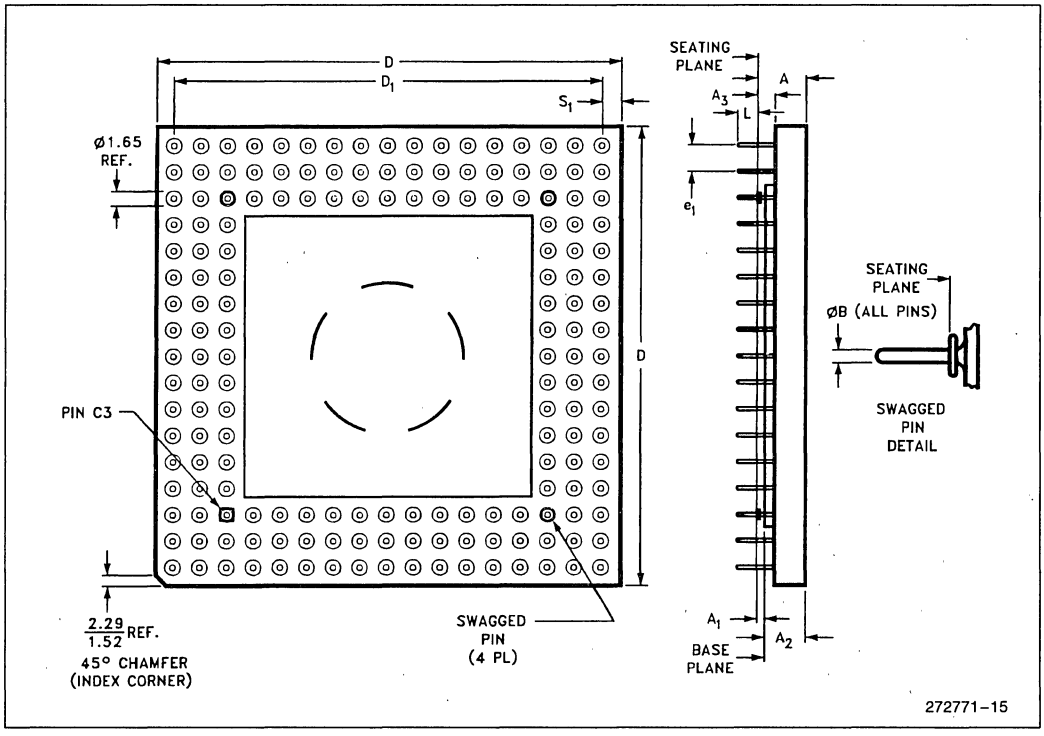


Figure 15. Principal Dimensions and Data for 168-Pin Pin Grid Array Package

Table 22. 168-Pin Ceramic PGA Package Dimensions

Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	3.56	4.57		0.140	0.180	
A_1	0.64	1.14	SOLID LID	0.025	0.045	SOLID LID
A_2	2.8	3.5	SOLID LID	0.110	0.140	SOLID LID
A_3	1.14	1.40		0.045	0.055	
B	0.43	0.51		0.017	0.020	
D	44.07	44.83		1.735	1.765	
D_1	40.51	40.77		1.595	1.605	
e_1	2.29	2.79		0.090	0.110	
L	2.54	3.30		0.100	0.130	
N	168			168		
S_1	1.52	2.54		0.060	0.100	

Table 23. Ceramic PGA Package Dimension Symbols

Letter or Symbol	Description of Dimensions
A	Distance from seating plane to highest point of body
A_1	Distance between seating plane and base plane (lid)
A_2	Distance from base plane to highest point of body
A_3	Distance from seating plane to bottom of body
B	Diameter of terminal lead pin
D	Largest overall package dimension of length
D_1	A body length dimension, outer lead center to outer lead center
e_1	Linear spacing between true lead position centerlines
L	Distance from seating plane to end of lead
S_1	Other body dimension, outer lead center to edge of body

NOTES:
1. Controlling dimension: millimeter.
2. Dimension "e_1" ("e") is non-cumulative.
3. Seating plane (standoff) is defined by P.C. board hole size: 0.0415–0.0430 inch.
4. Dimensions "B", "B_1" and "C" are nominal.
5. Details of Pin 1 identifier are optional.

6.2 Package Thermal Specifications

The embedded Write-Back Enhanced IntelDX4 processor is specified for operation when the case temperature (T_C) is within the range of 0°C to 85°C. T_C may be measured in any environment to determine whether the processor is within the specified operating range.

The ambient temperature (T_A) can be calculated from θ_{JC} and θ_{JA} from the following equations:

$$T_J = T_C + P * \theta_{JC}$$
$$T_A = T_J - P * \theta_{JA}$$
$$T_C = T_A + P * [\theta_{JA} - \theta_{JC}]$$
$$T_A = T_C - P * [\theta_{JA} - \theta_{JC}]$$

Where T_J, T_A, T_C equals Junction, Ambient and Case Temperature respectively. θ_{JC}, θ_{JA} equals Junction-to-Case and Junction-to-Ambient thermal Resistance, respectively. P is defined as Maximum Power Consumption.

Values for θ_{JA} and θ_{JC} are given in the following tables for each product at its maximum operating frequencies. Maximum T_A is shown for each product operating at its maximum processor frequency (three times the CLK frequency). Refer to the *Intel486™ Processor Family* datasheet (242202) for a description of the methods used to measure these characteristics.

Table 24. Thermal Resistance, θ_{JA}(°C/W)

Package	Heat Sink	θ_{JA}vs. Airflow—ft/min. (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
168-Pin PGA	No	17.5	15.0	13.0	11.5	10.0	9.5
168-Pin PGA	Yes	13.5	8.5	6.5	5.5	4.5	4.25
208-Lead SQFP	No	12.5	10.0	9.0	8.5		
208-Lead SQFP	Yes	10.5	6.5	5.0	4.0		

Table 25. Thermal Resistance, θ_{JC} (°C/W)

Package	Heat Sink	θ_{JC}
168-Pin PGA	No	2.0
168-Pin PGA	Yes	2.0
208-Lead SQFP	No	1.2
208-Lead SQFP	Yes	0.8

Table 26. Maximum $T_{ambient}$, T_A max (°C)

Package	Heat Sink	Freq. (MHz)	Airflow—ft/min. (m/sec)			
			0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
168-Pin PGA	No	100	18.5	29.0	37.5	44.0
168-Pin PGA	Yes	100	35.5	57.0	65.5	70.0
208-Lead SQFP	No	100	36.5	46.0	50.0	52.5
208-Lead SQFP	Yes	100	43.5	60.5	67.0	71.0
208-Lead SQFP	No	75				
208-Lead SQFP	Yes	75				

INTEL486™ PROCESSOR FAMILY

- Write-Back Enhanced IntelDX4™ Processor
  - Up to 100-MHz Operation
  - Speed-Multiplying Technology
  - 32-Bit Architecture
  - 16K-Byte On-Chip Write-Back Cache
  - Integrated Floating-Point Unit
  - 3.3V Core Operation with 5V Tolerant I/O Buffers
  - SL Technology
  - Static Design
  - IEEE 1149.1 Boundary Scan Compatibility
  - Binary Compatible with Large Software Base
- IntelDX4™ Processor
  - Up to 100-MHz Operation
  - Speed-Multiplying Technology
  - 32-Bit Architecture
  - 16K-Byte On-Chip Cache
  - Integrated Floating-Point Unit
  - 3.3V Core Operation with 5V Tolerant I/O Buffers
  - SL Technology
  - Static Design
  - IEEE 1149.1 Boundary Scan Compatibility
  - Binary Compatible with Large Software Base
- Write-Back Enhanced IntelDX2™ Processor
  - Speed-Multiplying Technology
  - 32-Bit Architecture
  - 8K-Byte On-Chip Write-Back Cache
  - Integrated Floating-Point Unit
  - SL Technology
  - Static Design
  - IEEE 1149.1 Boundary Scan Compatibility
  - Binary Compatible with Large Software Base
- IntelDX2™ Processor
  - Speed-Multiplying Technology
  - 32-Bit Architecture
  - 8K-Byte On-Chip Cache
  - Integrated Floating-Point Unit
  - SL Technology
  - Static Design
  - IEEE 1149.1 Boundary Scan Compatibility
  - Binary Compatible with Large Software Base
- IntelSX2™ Processor
  - Speed-Multiplying Technology
  - 32-Bit Architecture
  - 8K-Byte On-Chip Cache
  - SL Technology
  - Static Design
  - IEEE 1149.1 Boundary Scan Compatibility
  - Binary Compatible with Large Software Base
- Intel486™ DX Processor
  - 32-Bit Architecture
  - 8K-Byte On-Chip Cache
  - Integrated Floating-Point Unit
  - SL Technology
  - Static Design
  - IEEE 1149.1 Boundary Scan Compatibility
  - Binary Compatible with Large Software Base
- Intel486™ SX Processor
  - 32-Bit Architecture
  - 8K-Byte On-Chip Cache
  - SL Technology
  - Static Design
  - IEEE 1149.1 Boundary Scan Compatibility
  - Binary Compatible with Large Software Base

December 1995
Order Number: 242202-003

DATASHEET DESIGNATIONS

Intel uses various datasheet markings to designate each phase of the document as it relates to the product. The marking appears in the lower, inside corner of the datasheet. Following is the definition of these markings:

Datasheet Marking	Description
Product Preview	Contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product becomes available.
Advance Information	Contains information on products being sampled or in the initial production phase of development.†
Preliminary	Contains preliminary information on new products in production.†
No Marking	Contains information on products in full production.†

† Specifications within these datasheets are subject to change without notice. Verify with your local Intel sales office that you have the latest datasheet before finalizing a design.

INTEL486™ PROCESSOR FAMILY

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

1.0 INTRODUCTION

The Intel486™ processor family enables a range of low-cost, high-performance entry-level system designs capable of running the entire installed base of DOS*, Windows*, OS/2*, and UNIX* applications written for the Intel architecture. The Intel486 processor family also enables a wide range of high-performance embedded application designs. This family includes the IntelDX4™ processor, the fastest Intel486 processor (up to 50% faster than an IntelDX2™ processor). The IntelDX4 processor integrates a 16K unified cache and floating-point hardware on-chip for improved performance. The IntelDX2 processor integrates an 8K unified cache and floating-point hardware on chip. The IntelDX4 and IntelDX2 processors are also available with a write-back on-chip cache for improved entry-level performance. The IntelDX4 and IntelDX2 processors use Intel's speed-multiplying technology, allowing the processor to operate at frequencies higher than the external memory bus. The Intel486 DX processor offers the features of the IntelDX2 processors without speed-multiplying. The Intel486 SX processor offers the features of the Intel486 DX processor without floating-point hardware and the IntelSX2 processor adds speed-multiplying to the Intel486 SX processor. The entire Intel486 processor family incorporates energy efficient "SL Technology" for mobile and desktop computing.

SL Technology enables desktop system designs that exceed the Environment Protection Agency's (EPA) Energy Star program guidelines without compromising performance. It also increases system design flexibility and improves battery life in all Intel486 processor-based notebooks. SLTechnology allows system designers to differentiate their power management schemes with a variety of energy-efficient or battery-life preserving features. Intel486 processors provide power management features that are transparent to application and operating system software. Stop Clock, Auto HALT Power Down, and Auto Idle power down allow software transparent control over processor power management. Equally important is the capability of the processor to manage system power consumption. Intel486 processor System Management Mode (SMM) incorporates a non-maskable System Management Interrupt (SMI#), a corresponding Resume (RSM) instruction and a new memory space for system management code. Intel's SMM ensures seamless power control of the processor core, system logic, main memory, and one or more peripheral devices, that is transparent to any application or operating system.

Intel486 processors are available in a full range of speeds (25 MHz to 100 MHz), packages (PGA, SQFP PQFP), and voltages (5V, 3.3V) to meet any system design requirements.

1.1 Processor Features

All of the Intel486 processors consist of a 32-bit integer processing unit, an on-chip cache, and a memory management unit. This ensures full binary compatibility with the 8086, 8088, 80186, 80286, Intel386™ SX, Intel386 DX, and all versions of Intel486 processors. All of the Intel486 processors offer the following features:

- *32-bit RISC integer core*—The Intel486 processor performs a complete set of arithmetic and logical operations on 8-, 16-, and 32-bit data types using a full-width ALU and eight general purpose registers.
- *Single Cycle Execution*—Many instructions execute in a single clock cycle.
- *Instruction Pipelining*—The fetching, decoding, address translation and execution of instructions are overlapped within the Intel486 processor.
- *On-Chip Floating-Point Unit*—Intel486 processors support the 32-, 64-, and 80-bit formats specified in IEEE standard 754. The unit is binary compatible with the 8087, Intel287™, Intel387™ coprocessors, and Intel OverDrive® processor.
- *On-Chip Cache with Cache Consistency Support*—An 8-Kbyte (16 Kbyte on the IntelDX4 processor) internal cache is used for both data and instructions. Cache hits provide zero wait-state access times for data within the cache. Bus activity is tracked to detect alterations in the memory represented by the internal cache. The internal cache can be invalidated or flushed so that an external cache controller can maintain cache consistency.
- *External Cache Control*—Write-back and flush controls for an external cache are provided so the processor can maintain cache consistency.
- *On-Chip Memory Management Unit*—Address management and memory space protection mechanisms maintain the integrity of memory in a multitasking and virtual memory environment. Both segmentation and paging are supported.

- *Burst Cycles*—Burst transfers allow a new double word to be read from memory on each bus clock cycle. This capability is especially useful for instruction prefetch and for filling the internal cache.
- *Write Buffers*—The processor contains four write buffers to enhance the performance of consecutive writes to memory. The processor can continue internal operations after a write to these buffers, without waiting for the write to be completed on the external bus.
- *Bus Backoff*—If another bus master needs control of the bus during a processor initiated bus cycle, the Intel486 processor will float its bus signals, then restart the cycle when the bus becomes available again.
- *Instruction Restart*—Programs can continue execution following an exception generated by an unsuccessful attempt to access memory. This feature is important for supporting demand-paged virtual memory applications.
- *Dynamic Bus Sizing*—External controllers can dynamically alter the effective width of the data bus. Bus widths of 8, 16, or 32 bits can be used.
- *Boundary Scan (JTAG)*—Boundary Scan provides in-circuit testing of components on printed circuit boards. The Intel Boundary Scan implementation conforms with the IEEE Standard Test Access Port and Boundary Scan Architecture.

SL Technology provides the following features:

- *Intel System Management Mode*—A unique Intel architecture operating mode provides a dedicated special purpose interrupt and address space that can be used to implement intelligent power management and other enhanced functions in a manner that is completely transparent to the operating system and applications software.
- *I/O Restart*—An I/O instruction interrupted by a System Management Interrupt (SMI#) can automatically be restarted following the execution of the RSM instruction.
- *Stop Clock*—The Intel486 processor has a stop clock control mechanism that provides two low-power states: a "fast wake-up" Stop Grant state (~20–100 mA) and a "slow wake-up" Stop Clock state with CLK frequency at 0 MHz (100–1000 μA).
- *Auto HALT Power Down*—After the execution of a HALT instruction, the Intel486 processor issues a normal Halt bus cycle and the clock input to the Intel486 processor core is automatically stopped, causing the processor to enter the Auto HALT Power Down state (~20–100 mA).
- *Upgrade Power Down Mode*—When a Intel486 processor upgrade is installed, the upgrade power down mode detects the presence of the upgrade, powers down the core, and tri-states all outputs of the original processor, so the Intel486 processor enters a very low current mode.
- *Auto Idle Power Down* —This function allows the processor to reduce the core frequency to the bus frequency when both the core and bus are idle. Auto Idle Power Down is software transparent and does not affect processor performance. Auto Idle Power Down provides an average power savings of 10% and is only applicable to clock multiplied processors.

Enhanced Bus Mode Features (for the Write-Back Enhanced IntelDX4 and the Write-Back Enhanced IntelDX2 Processors only):

- *Write Back Internal Cache*—The Write-Back Enhanced IntelDX2 processor adds write-back support to the unified cache. The on-chip cache is configurable to be write-back or write-through on a line by line basis. The internal cache implements a modified MESI protocol, which is most applicable to uniprocessor systems.
- *Enhanced Bus Mode*—The definitions of some signals have been changed to support the new Enhanced Bus mode (write-back mode).
- *Write Bursting*—Data written from the processor to memory can be bursted (zero wait state transfer).

For items which are common to both the Write-Back Enhanced IntelDX4 and Write-Back Enhanced IntelDX2 processors, the term *Write-Back Enhanced Intel486 processors* will be used.

1.2 Intel486™ Processor Product Family

Table 1-1 shows the Intel486 processors available by Clock Mode, Supply Voltage, Maximum Frequency, and Package. Likewise, an individual product will have either a 5V supply voltage or a 3.3V supply voltage, but not both. An individual product will have either a 1X clock or a 2X clock, but not both. Please contact Intel for the latest product availability and specifications.

Table 1-1. Product Options

Intel486™ Processors	V_{CC}	Processor Frequency (MHz)							168-Pin PGA	208-Lead SQFP	196-Lead PQFP
		25	33	40	50	66	75	100			
1X Clock											
Intel486 SX Processor	3.3V	✓	✓							✓	
	5V	✓	✓						✓		✓
IntelSX2™ Processor	5V				✓				✓		
Intel486 DX Processor	3.3V		✓							✓	
	5V(1)		✓		✓				✓		✓
IntelDX2™ Processor	3.3V			✓	✓					✓	
	5V				✓	✓			✓		
Write-Back Enhanced IntelDX2 Processor	3.3V			✓	✓					✓	
	5V				✓	✓			✓		
IntelDX4™ Processor	3.3V						✓	✓	✓	✓	
Write-Back Enhanced IntelDX4™ Processor	3.3V						✓	✓	✓	✓	
2X Clock											
Intel486 SX Processor(2)	3.3V	✓	✓							✓	
	5V	✓	✓								✓

NOTES:
1. The 5V 33-MHz Intel486™ DX processor is available in 168-pin PGA and 196-lead PQFP packages. The 5V 50-MHz Intel486 DX processor is available in a 168-pin PGA package only.
2. With the addition of SL Technology to the Intel486 processor family, the Low Power Intel486 SX and Low Power Intel486 DX processors have been superseded with the 3.3V Intel486 processors described in this document.

2.0 HOW TO USE THIS DOCUMENT

2.1 Introduction

This datasheet is a compilation of previously published individual data sheets for the Intel486 SX, IntelSX2, Intel486 DX, IntelDX2 and IntelDX4 processors. With the addition of the Write-Back Enhanced Intel486 processors and information previously published for the introduction of the SL Enhanced Intel486 processors, this datasheet encompasses the entire current Intel486 processor family.

This datasheet describes the Intel486 processor architecture, features and technical details. Unless otherwise stated, any description for the Intel486 processor listed in this datasheet applies to all Intel486 processors. Where architectural or other differences do occur (for example, the IntelDX4 processor has a 16-Kbyte on-chip cache, all other Intel486 processors have an 8-Kbyte on-chip cache), these differences are described in separate sections. Section 2.2 provides a brief section description, highlighting the specific sections that contain processor-unique information.

This datasheet does not detail the Intel486SL processor, the Low Power Intel486 SX or Low Power Intel486 DX directly. The Low Power Intel486 and Intel486 SL processors have been superseded by current versions of the Intel486 processors.

It is important to note that all Intel486 DX, IntelDX2, and IntelDX4 processors have an on-chip floating-point unit. The Intel486 SX and IntelSX2 processors do not have an on-chip floating-point unit and do not provide FERR# and IGNNE# floating-point error reporting signals.

The 5V 50-MHz Intel486 DX processor does not implement SL Technology and does not contain the following pins: SMIACT#, SRESET, SMI#, STPCLK#, and UP#.

Boundary Scan (JTAG) testability features, capability and associated test signals (TCK, TMS, TDI, and TDO) are standard on all Intel486 processors except the Intel486 SX processors in 168-pin PGA package.

Unless specifically noted, descriptions specified for the IntelDX2 processor also apply to the Write-Back Enhanced IntelDX2 processor, whether in Enhanced Bus Mode (Write-Back cache) or Standard Bus Mode (Write-Through cache). Similarly, descriptions specified for the IntelDX4 processor also apply to the Write-Back Enhanced IntelDX4 processor.

2.2 Section Contents and Processor Specific Information

The following is a brief description of the contents of each section:

Section 1: "Introduction." This section is an overview of the current Intel486 processor family, product features and highlights. This section also lists product frequency, voltage and package offerings.

Section 2: "How to Use This Document." This section presents information to aid in the use of this datasheet.

Section 3: "Pin Description." This section contains all of the pin configurations for the various package options (168-Pin PGA, 208-Lead SQF, and 196-Lead PQFP), package diagrams, pin assignment tables and pin assignment differences for the various processors within a package class.

The 168-Pin PGA and 208-Lead SQFP package diagrams shown are for the IntelDX2, Write-Back Enhanced IntelDX2, IntelDX4 and Write-Back Enhanced IntelDX4 processors, with differences for other members of the Intel486 processor family listed in separate tables. The 196-Lead PQFP package diagram is for the Intel486 DX processor. Differences for the Intel486 SX processor in the 196-Lead package are also listed in a separate table.

This section also provides a quick pin reference table that lists pin signals for the Intel486 processor family. The table, whenever necessary, has sections applicable to each current Intel486 processor family member.

Section 4: "Architectural Overview." This section describes the Intel486 processor architecture, including the register and instruction sets, memory organization, data types and formats, and interrupts for all Intel486 processors.

The architectural overview describes the 32-bit RISC integer core of the Intel486 processor. The on-chip floating-point unit for the Intel486 DX IntelDX2 and IntelDX4 processors is included in this section. Operational differences for the Intel486 SX and IntelSX2 processors (i.e. processors that do not contain on-chip floating-point units) are also described in detail.

Section 5: "Real Mode Architecture." This section describes the Intel486 processor real-mode architecture, including memory addressing, reserved locations, interrupts, and Shutdown and HALT. This section applies to all Intel486 processors.

Section 6: "Protected Mode Architecture." This section describes the Intel486 protected-mode architecture, including addressing mechanism, segmentation, protection, paging and virtual 8086 environment. This section applies to all Intel486 processors.

Section 7: "On-Chip Cache." This section describes the on-chip cache of the Intel486 processors. Specific information on size, features, modes, and configurations is described. The differences between the IntelDX4 processor on-chip cache (16-KByte) and other members of the Intel486 processor family on chip cache (8-KByte) are detailed.

This section also documents features, modes and operational issues specific to the Write-Back Enhanced Intel486 processors. The specifics for the Write-Back Enhanced Intel486 processors are interleaved with sections on the Standard mode (Write-Through cache) of other Intel486 processors as appropriate.

Section 8: "System Management Mode (SMM) Architectures." This section describes the System Management Mode architecture of the Intel486 processors, including system management mode interrupt processing and programming mode. Specific information to the Write-Back Enhanced Intel486 processors only are listed in appropriate sections.

This section applies to Intel486 processors except the 50 MHz Intel486 DX processor, which does not implement SL Technology.

Section 9: "Hardware Interface." This section describes the hardware interface of the current Intel486 processor family, including signal descriptions, interrupt interfaces, write buffers, reset and initialization, and clock control.

The IntelDX4 processor speed multiplying options are detailed in this section. The Write-Back Enhanced IntelDX2 processor signals (both new pins and those which have different operational functions) are detailed in this section. Reset and initialization, as it applies to all of the Intel486 processor family, is also documented here.

Use and operation of the Stop Clock, Auto HALT Power Down and other power-saving SL Technology features are described. Information specific to the Write-Back Enhanced IntelDX2 processor is also documented whenever appropriate.

Section 10: "Bus Operation." This section describes the Intel486 processor bus operation, including the data transfer mechanism and bus functional description. When in Standard Bus mode, the Write-Back Enhanced IntelDX2 processor bus operation is the same as other members of the Intel486 processor family. Specific information to the Write-Back IntelDX2 processor in Enhanced Bus mode is detailed in a separate section for ease of use.

Section 11: "Testability." This section describes the testability of the Intel486 processors, including the built-in self test (BIST), on-chip cache testing, translation lookaside buffer (TLB) testing, tri-state output test mode, and boundary scan (JTAG).

Both the IntelDX4 processors and the Write-Back Enhanced Intel486 and the IntelDX4 processors have unique cache structures that alter testing in comparison to other members of the current Intel486 processor family These processor-specific

differences are documented in this section. A complete listing of Boundary Scan ID Codes and Boundary Scan Register Bits orders are also included.

Section 12: "Debugging Support." This section describes the Intel486 processor debugging support, including the breakpoint instruction, single-step trap and debug registers. This section applies to all Intel486 processors.

Section 13: "Instruction Set Summary." This section provides clock count and instruction encoding summaries for all the Intel486 processors.

Section 14: "Differences between Intel486™ Processors and Intel386™ Processors." This section lists the differences between the Intel486 processor family and the Intel386 processor family. Also described and documented are differences between the Intel386 with an Intel387™ math coprocessors and the Intel486 processors with on-chip floating-point units. This section applies to all Intel486 processors.

Section 15: "Differences between the PGA, SQFP and PQFP Versions of the Intel486™ SX and Intel486™ DX Processors." The Low Power Intel486 SX and Intel486 DX processors have been superseded by the current Intel486 processors. This section lists the differences between the current Intel486 SX and Intel486 DX products offered in PGA, SQFP and PQFP packages. The current Intel486 SX and Intel486 DX processors in the PQFP package can operate in 2X clock mode, which is described in detail here. Electrical specifications for the Intel486 SX and Intel486 DX processors in 2X Clock mode are listed in this section.

Section 16: "OverDrive® Processor Socket." This section describes the OverDrive processor socket requirements for end-user upgradability of the Intel486 processor family. This section applies to all Intel486 processors.

Section 17: "Electrical Data." This section lists the AC and DC specifications for all Intel486 processors. Processor specific information is listed in both common and separate tables and sections as appropriate.

Section 18: "Mechanical Data." This section lists the mechanical and thermal data, including the package specifications (PGA, SQFP and PQFP) for all Intel486 processors. Processor specific information is listed in both common and separate tables and sections as appropriate.

Appendix A: "Advanced Features." This section documents the advanced features of the Intel486 processor family not covered in other sections of this datasheet.

Appendix B: "Features Determination." This section documents the CPUID function to determine the Intel486 processor family identification and processor specific information. This section applies to all Intel486 processors.

Appendix C: "IBIS Models." This section provides a detailed sample listing of the types of I/O buffer modeling information available for the Intel486 processor family. This section applies to all Intel486 processors.

Appendix D: "BSDL Listing." This section provides a sample listing of a BSDL file for the Intel486 processor family. This section applies to all Intel486 processors.

Appendix E: "System Design Notes." This section provides design notes applicable to the use of System Management Mode and SMM routines with the Intel486 processor. This section applies to all Intel486 processors, except the 50-MHz Intel486 DX processor.

2.3 Documents Replaced by This Datasheet

This datasheet contains all of the latest information for the Intel486 processor family and replaces the following documentation:

SL Enhanced Intel486™ Microprocessor Datasheet Addendum, Order No. 241696

Intel486™ SX Microprocessor Data Book, Order No. 240950

IntelSX2™ Microprocessor Datasheet, Order No. 241966

Intel486™ DX Microprocessor Data Book, Order No. 240440

Intel486™ DX2 Microprocessor Data Book, Order No. 241245

IntelDX4™ Microprocessor Data Book, Order No. 241944

Intel486™ Family of Microprocessors Low Power Version Datasheet, Order No. 241199

2.4 REVISION HISTORY

Revision -001 is the original version of this document.

Revision -002 of the Intel486 Processor Family datasheet contains updates to the original version. A revision summary of major changes is listed below.

The sections significantly revised since -001 are:

Section 3.2	Revised description for PLOCK# signal to improve readability.
Figure 4.2	Base Architecture Registers: Corrected figure for EAX (AH, AL).
Table 4-13	FPU Register Usage Differences: Corrected table for Protected Mode operation.
Figure 6-18	GDT Descriptors for Simple System: Corrected Code Descriptor bit values.
Section 11.5.5	Corrected listings of bit orders for boundary scan registers for each Intel486 processor, as A29 was missing.
Table 13-15	Clock Count Summary: Added note listings for IMUL instruction for the IntelDX4 processor and corrected display of MOV and Segment Override Prefix instruction formats.
Figure 17-15	IntelDX4 Processor Derating
Figure 17-16	Curves: Revised figures for falling
Figure 17-17	3V and 5V signals.

Revision -003 incorporates documentation for the Write-Back Enhanced IntelDX4 processor.

3.0 PIN DESCRIPTION

3.1 Pin Assignments

The following figures show the pin assignments of each package type for the Intel486 processor product family. Tables are provided showing the pin differences between existing Intel486 processor products.

168-Pin PGA—Pin Grid Array
- Package Diagram
- Pin Assignment Difference Table
- Pin Cross Reference by Pin Name

208-Lead SQFP—Quad Flat Pack
- Package Diagram
- Pin Assignment Difference Table
- Pin Assignment Table in Numerical Order

196-Lead PQFP—Plastic Quad Flat Pack
- Package Diagram
- Pin Assignment Difference Table
- Pin Assignment Table in Numerical Order

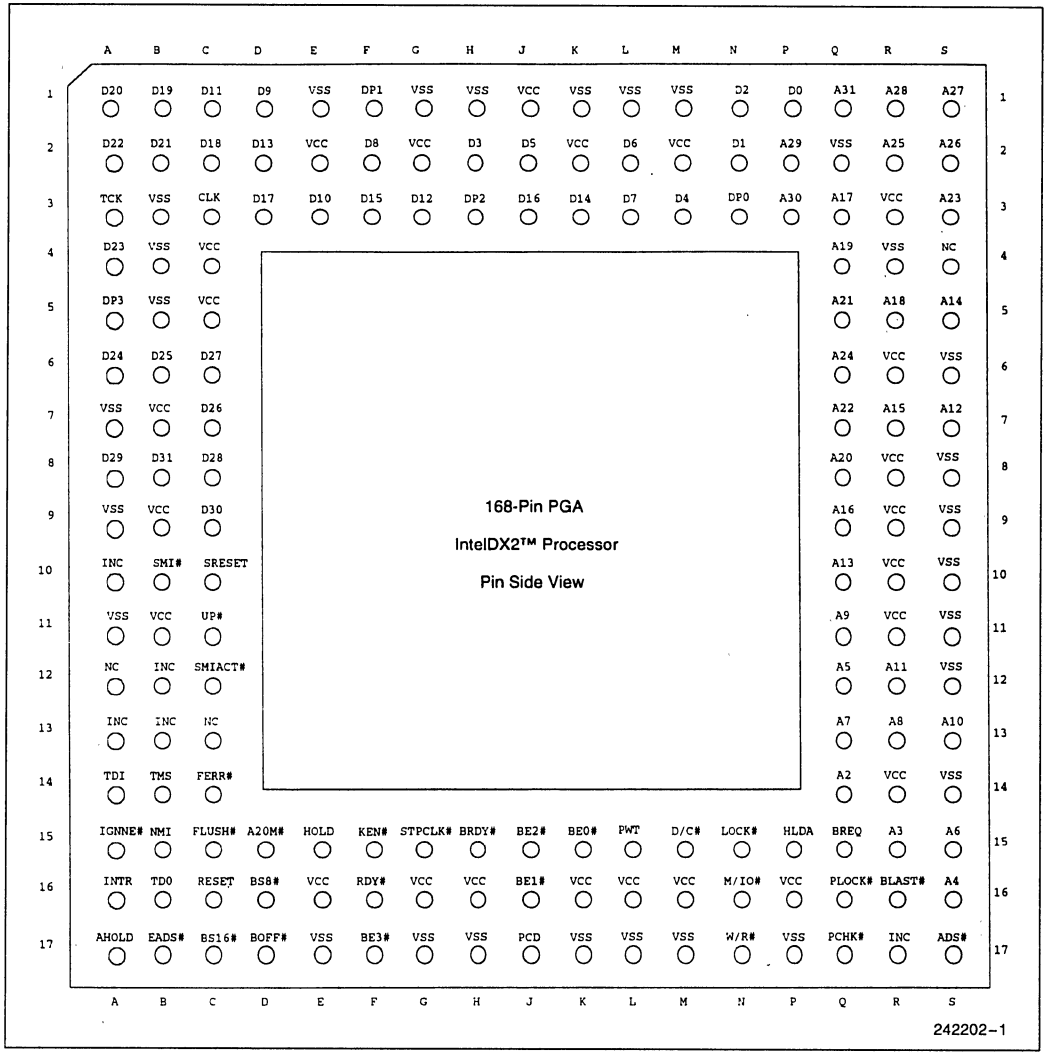


Figure 3-1. Package Diagram for 168-Pin PGA Package of the IntelDX2™ Processor

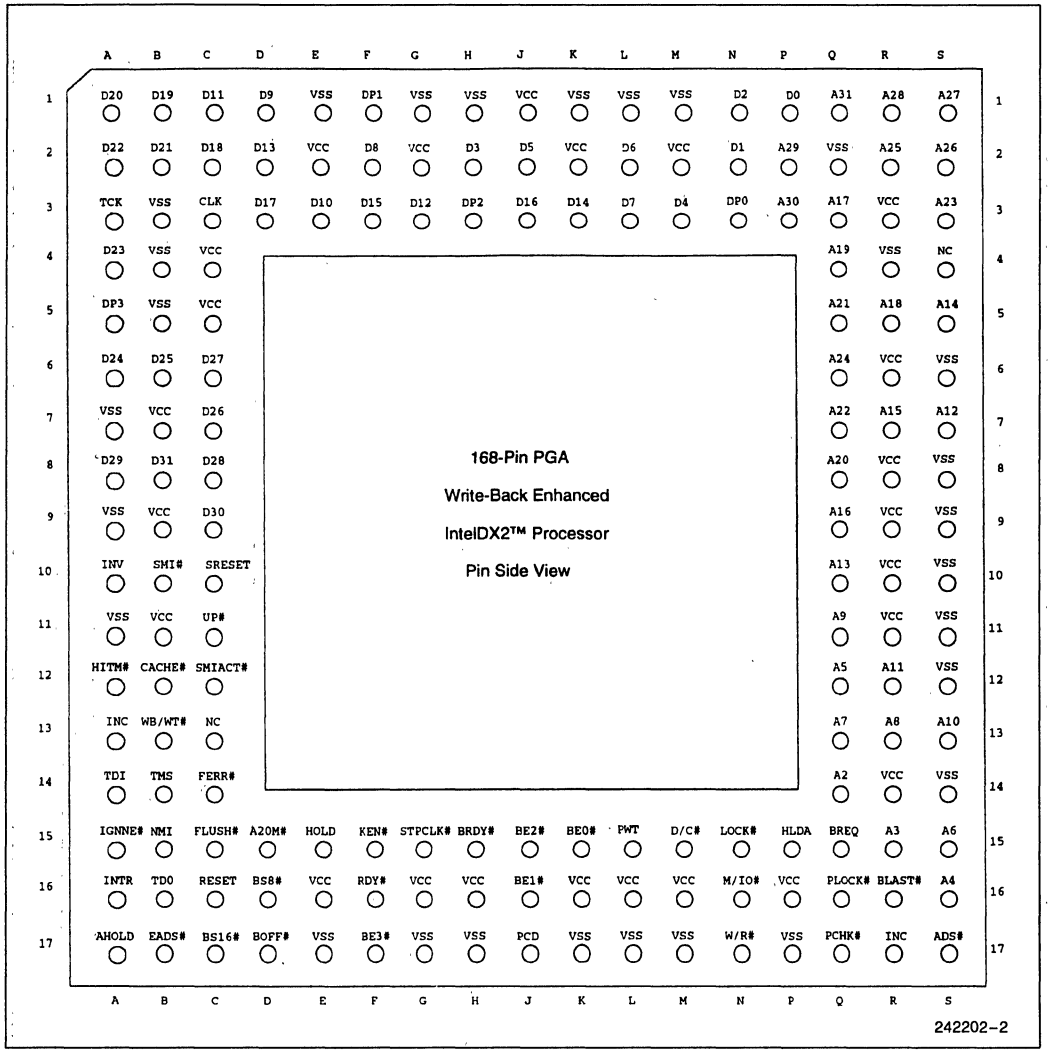


Figure 3-2. Package Diagram for 168-Pin PGA Package of the Write-Back Enhanced IntelDX2™ Processor

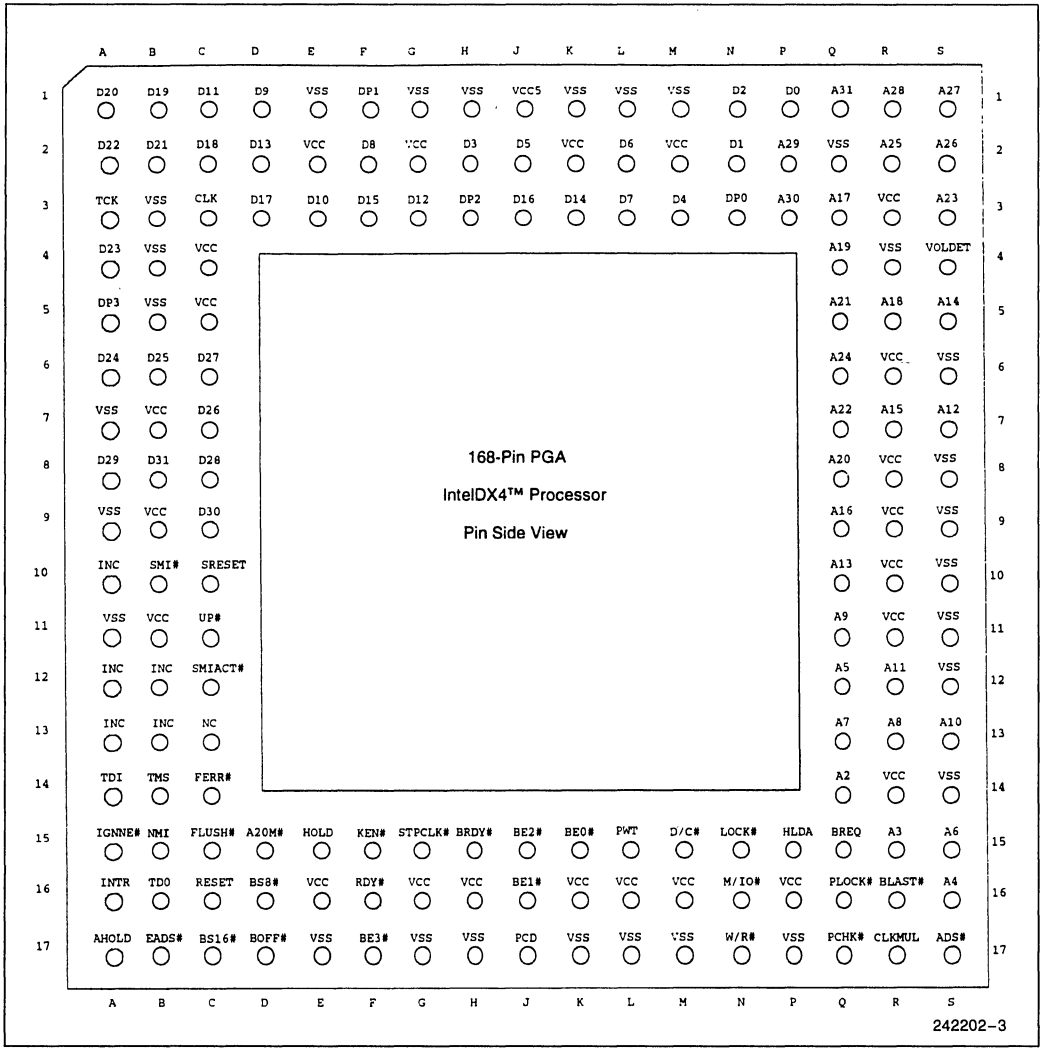


Figure 3-3. 168-Pin PGA Pinout Diagram (Pin Side) for the IntelDX4™ Processor

4

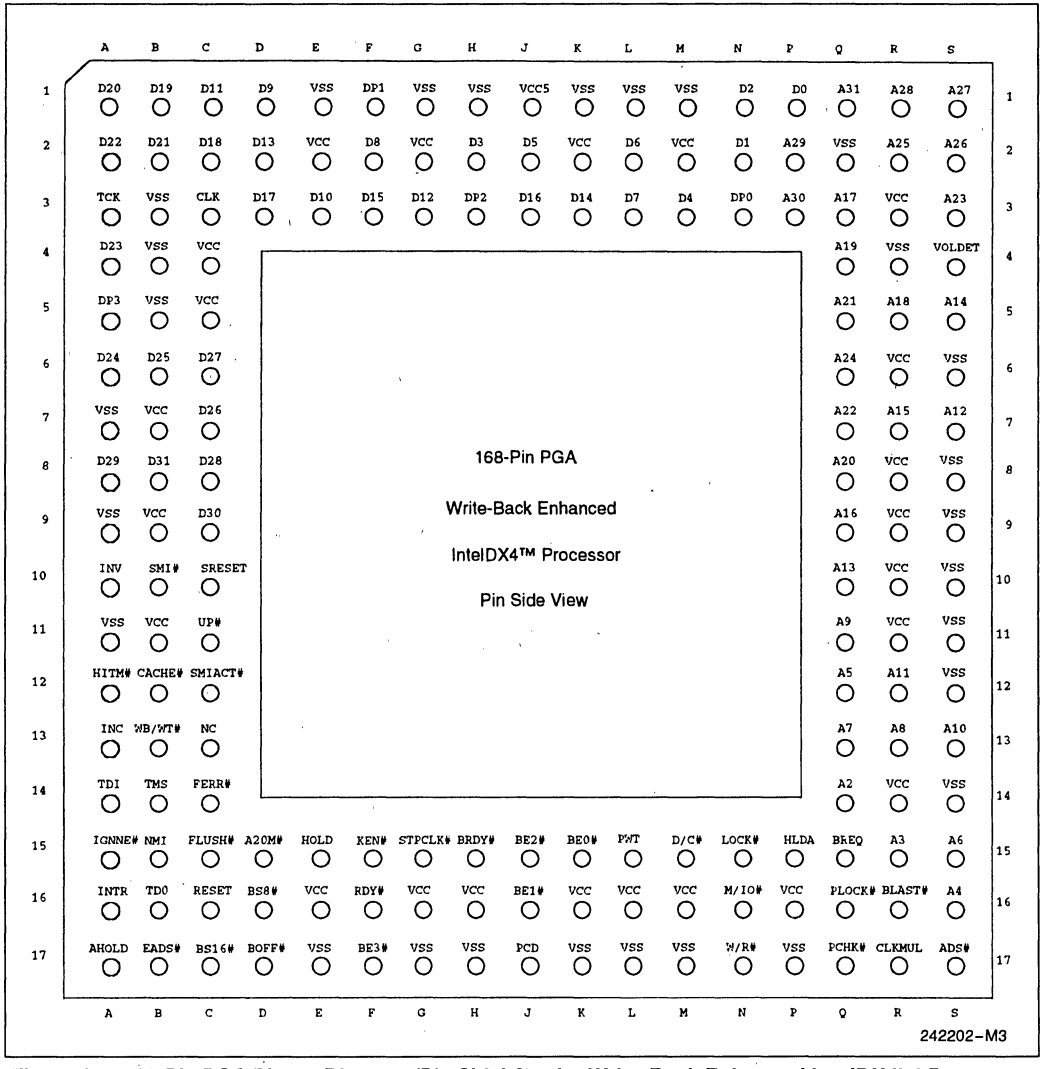


Figure 3-4. 168-Pin PGA Pinout Diagram (Pin Side) for the Write-Back Enhanced IntelDX4™ Processor

Table 3-1. Pinout Differences for 168-Pin PGA Package

Pin	Previous Intel486™ SX Processor[7]	Intel486 SX Processor	IntelSX2™ Processor	Previous Intel486 DX Processor[7]	Intel486 DX Processor	Previous IntelDX2™ Processor[7]	IntelDX2 Processor	IntelDX4™ Processor
A3	NC[1]	NC	TCK	NC TCK[4]	TCK	TCK[5]	TCK	TCK
A10	INC[2]	INC	INC	INC	INC	INC	INC INV[6]	INC INV[6]
A12	INC	INC	INC	INC	INC	INC	INC HITM#[6]	INC HITM#[6]
A13	NC	INC	INC	NC	INC	NC	INC	INC
A14	NC	NC	TDI	NC TDI[4]	TDI	TDI[5]	TDI	TDI
A15	NMI	NMI	INC	IGNNE#	IGNNE#	IGNNE#	IGNNE#	IGNNE#
B10	INC	SMI#	SMI#	INC	SMI#	INC	SMI#	SMI#
B12	INC	INC	INC	INC	INC	INC	INC CACHE#[6]	INC CACHE#[6]
B13	INC	INC	INC	INC	INC	INC	INC WB/WT#[6]	INC WB/WT#[6]
B14	NC	NC	TMS	NC TMS[4]	TMS	TMS[5]	TMS	TMS
B15	INC	INC	NMI	NMI	NMI	NMI	NMI	NMI
B16	NC	NC	TDO[5]	NC TDO[4]	TDO	TDO[5]	TDO	TDO
C10	INC	SRESET	SRESET	INC	SRESET	INC	SRESET	SRESET
C11	INC	UP#	UP#	INC	UP#	UP#	UP#	UP#
C12	INC	SMIACT#	SMIACT#	INC	SMIACT#	INC	SMIACT#	SMIACT#
C14	INC	INC	INC	FERR#	FERR#	FERR#	FERR#	FERR#
G15	INC	STPCLK#	STPCLK#	INC	STPCLK#	INC	STPCLK#	STPCLK#
J1	V_{CC}[3]	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC5}
R17	INC	INC	INC	INC	INC	INC	INC	CLKMUL
S4	NC	NC	NC	NC	NC	NC	NC	VOLDET

NOTES:

1. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486™ processors.
2. INC. Internal No Connect. These pins are not connected to any internal pad in Intel486 processors and OverDrive® processors. However, new signals are defined for the location of the INC pins in the Intel486 processor proliferations. All INC pins defined by Intel have a specific use for jumperless single socket compatibility with current and future processors. A system design could connect any signal to an INC pin without affecting the operation of the processor. However, the purpose of a specific INC pin should be understood before it is used. If not, the system design will sacrifice the ability to implement a jumperless (single socket) flexible motherboard.
3. This pin location is for the V_{CC5} pin on the IntelDX4™ processor. For compatibility with 3.3V processors that have 5V safe input buffers (i.e., IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane. See section 3.2, "Quick Pin Reference," for a description of the V_{CC5} pin on the IntelDX4 processor.
4. These pins were only available on previous 50-MHz Intel486 DX processors. These pins are now on all speeds of the Intel486 DX processor.
5. These pins were No Connects on previous Intel486 DX and IntelDX2™ processors. For compatibility with old designs, they can still be left unconnected.
6. These pins are used on the Write-Back Enhanced IntelDX4 and Write-Back Enhanced IntelDX2 processors only.
7. Previous versions of the Intel486 processor family do not implement SL Technology and are not described in this data-sheet.

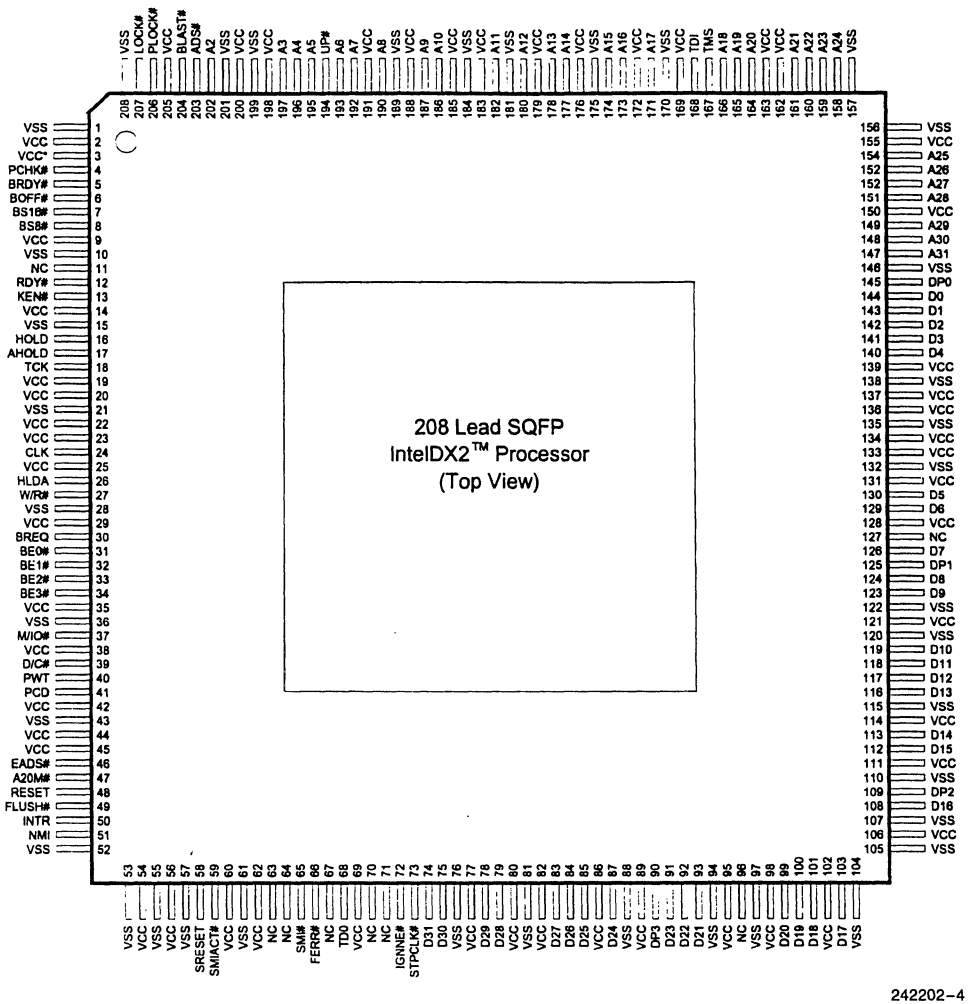
Table 3-2. Pin Cross Reference for 168-Pin PGA Package of the IntelDX2™ Processor

Address	Data	Control	INC(1)	V_{CC}	V_{SS}
A2 .....Q14	D0.....P1	A20M# ..D15	A10	B7	A7
A3 .....R15	D1 .....N2	ADS# ...S17	A12	B9	A9
A4.....S16	D2 .....N1	AHOLD ..A17	A13	B11	A11
A5 .....Q12	D3 .....H2	BE0# K15	B12	C4	B3
A6.....S15	D4 .....M3	BE1# J16	B13	C5	B4
A7 .....Q13	D5 .....J2	BE2# J15	R17	E2	B5
A8 .....R13	D6.....L2	BE3# F17	NC (2)	E16	E1
A9 .....Q11	D7.....L3	BLAST# .R16		G2	E17
A10 ....S13	D8.....F2	BOFF# ..D17	C13	G16	G1
A11 ....R12	D9 .....D1	BRDY# ..H15	S4	H16	G17
A12.....S7	D10 .....E3	BREQ ...Q15		J1	H1
A13 ....Q10	D11 .....C1	BS8# D16		K2	H17
A14.....S5	D12 .....G3	BS16# ..C17		K16	K1
A15 .....R7	D13 .....D2	CLK C3		L16	K17
A16 .....Q9	D14 .....K3	D/C# ...M15		M2	L1
A17 .....Q3	D15.....F3	DP0 .....N3		M16	L17
A18 .....R5	D16.....J3	DP1.....F1		P16	M1
A19 .....Q4	D17 .....D3	DP2 .....H3		R3	M17
A20 .....Q8	D18 .....C2	DP3 .....A5		R6	P17
A21 .....Q5	D19 .....B1	EADS# ..B17		R8	Q2
A22 .....Q7	D20 .....A1	FERR# ..C14		R9	R4
A23.....S3	D21 .....B2	FLUSH# .C15		R10	S6
A24 .....Q6	D22 .....A2	HLDA P15		R11	S8
A25 .....R2	D23 .....A4	HOLD....E15		R14	S9
A26.....S2	D24 .....A6	IGNNE# .A15			S10
A27.....S1	D25 .....B6	INTR A16			S11
A28 .....R1	D26 .....C7	KEN# ...F15			S12
A29.....P2	D27 .....C6	LOCK# ..N15			S14
A30.....P3	D28 .....C8	M/IO# ..N16			
A31 .....Q1	D29 .....A8	NMI B15			
	D30 .....C9	PCD J17			
	D31 .....B8	PCHK# ..Q17			
		PWT L15			
		PLOCK ..Q16			
		RDY# ...F16			
		RESET ..C16			
		SMI# B10			
		SMIACT# C12			
		UP# C11			
		W/R# ...N17			
		STPCLK#G15			
		SRESET .C10			
		TCK ....A3(3)			
		TDI ....A14(3)			
		TDO ...B16(3)			
		TMS ...B14(3)			

4

NOTES:

1. INC. Internal No Connect. These pins are not connected to any internal pad in Intel486™ processors and OverDrive® processors. However, new signals are defined for the location of the INC pins in the Intel486 processor proliferation. All INC pins defined by Intel have a specific use for jumperless single socket compatibility with current and future processors. A system design could connect any signal to an INC pin without affecting the operation of the processor. However, the purpose of a specific INC pin should be understood before it is used. If not, the system design will sacrifice the ability to implement a jumperless (single socket) flexible motherboard.
2. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.
3. Boundary Scan pins are not included on the 168-pin PGA package version of the Intel486 SX processor.



* Pin 3. See Note 1 for Table 3-3.

Figure 3-5. Package Diagram for 208-Lead SQFP of the IntelDX2™ Processor

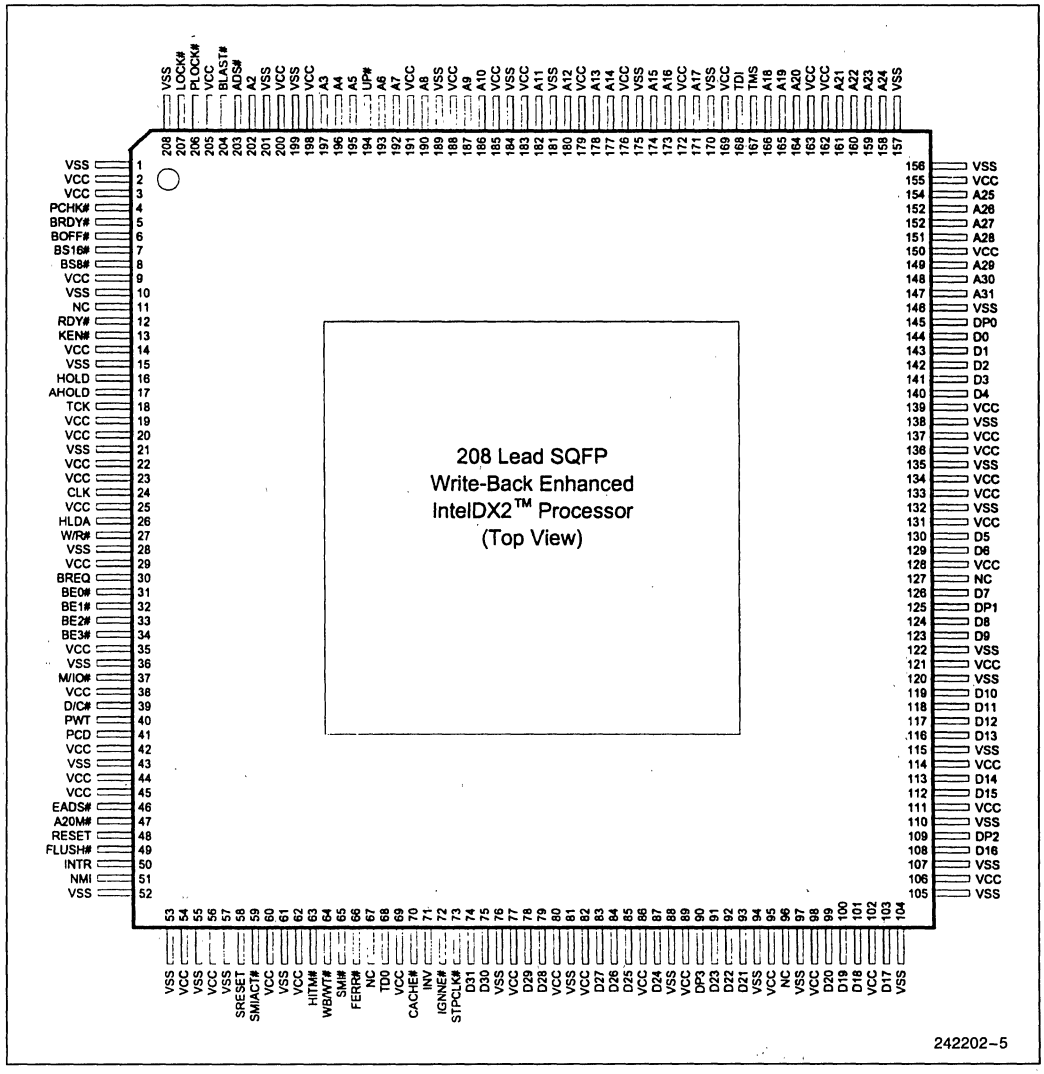


Figure 3-6. Package Diagram for 208-Lead SQFP of the Write-Back Enhanced IntelDX2™ Processor

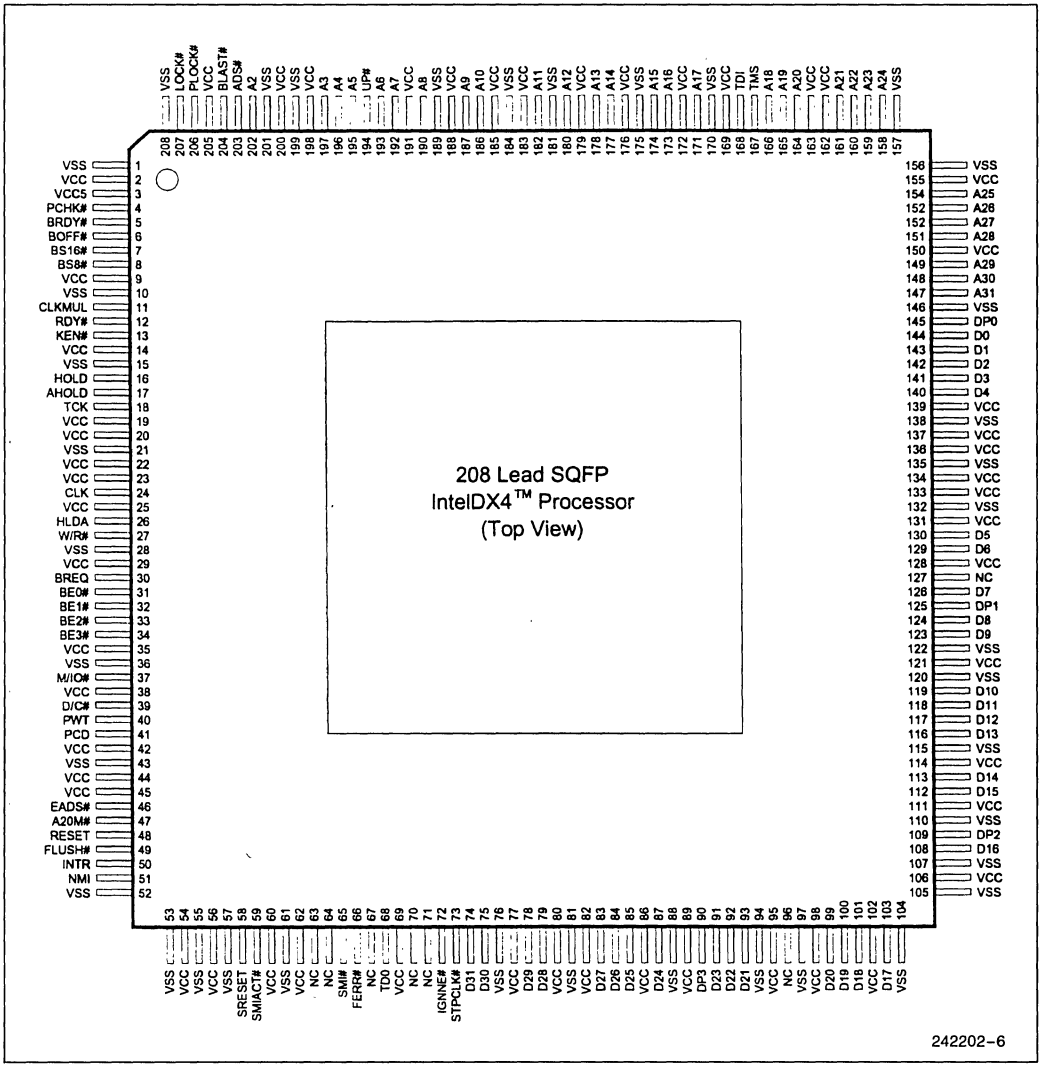


Figure 3-7. Package Diagram for 208-Lead SQFP of the IntelDX4™ Processor

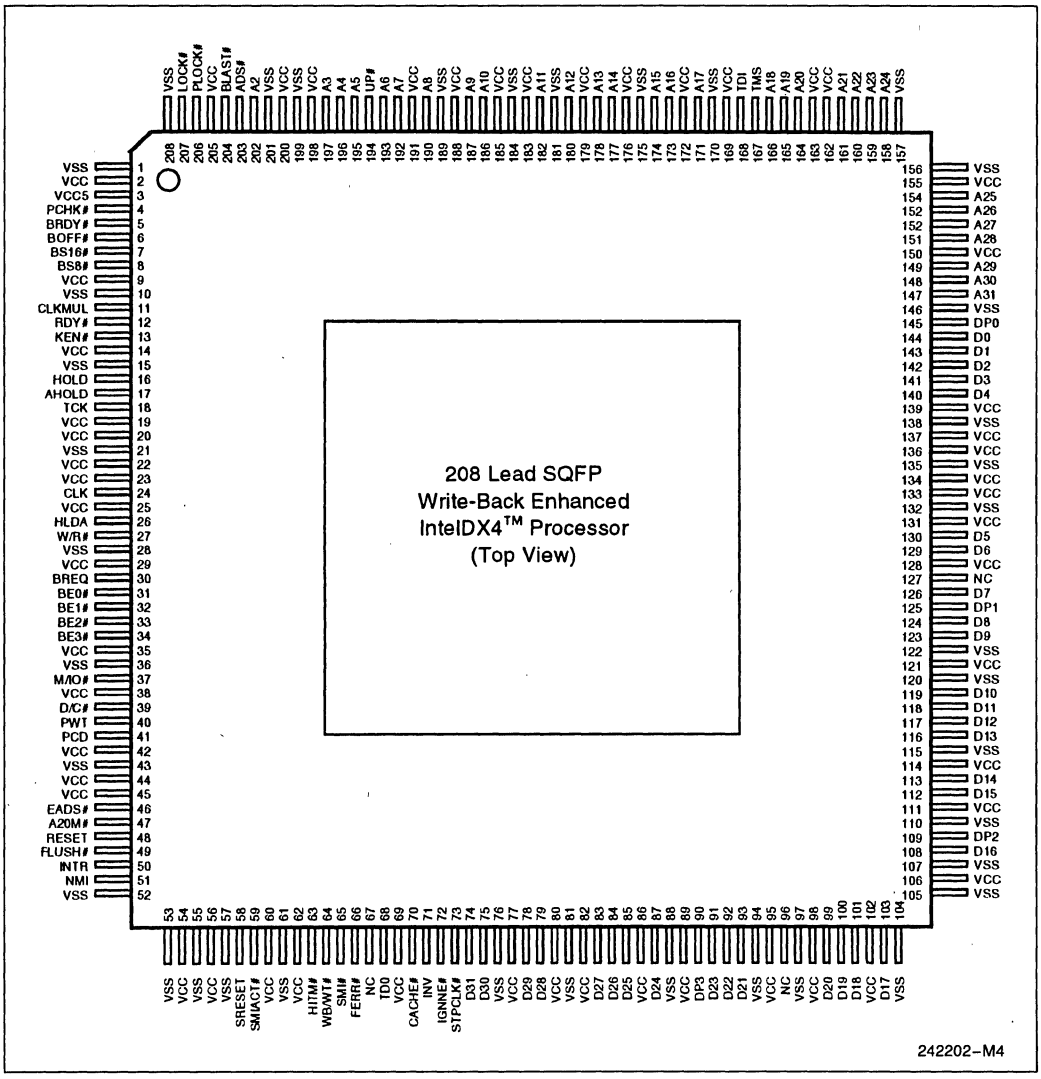


Figure 3-8. Package Diagram for 208-Lead SQFP of the Write-Back Enhanced IntelDX4™ Processor

Table 3-3. Pinout Differences for 208-Lead SQFP Package

Pin #	Intel486™ SX Processor	Intel486 DX Processor	IntelDX2™ Processor	Write-Back Enhanced IntelDX2 Processor	IntelDX4™ Processor	Write-Back Enhanced IntelDX4™ Processor
3	V_{CC}(1)	V_{CC}	V_{CC}	V_{CC}	V_{CC5}	V_{CC5}
11	INC(2)	INC	INC	INC	CLKMUL	CLKMUL
63	INC	INC	INC	HITM#	INC	HITM#
64	INC	INC	INC	WB/WT#	INC	WB/WT#
66	INC	FERR#	FERR#	FERR#	FERR#	FERR#
70	INC	INC	INC	CACHE#	INC	CACHE#
71	INC	INC	INC	INV	INC	INV
72	INC	IGNNE#	IGNNE#	IGNNE#	IGNNE#	IGNNE#

NOTES:
1. This pin location is for the V_{CC5} pin on the IntelDX4™ processor. For compatibility with 3.3V processors that have 5V safe input buffers (i.e., IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane. See section 3.2, "Quick Pin Reference," for a description of the V_{CC5} pin on the IntelDX4 processor.
2. INC. Internal No Connect. These pins are not connected to any internal pad in Intel486™ processors and OverDrive® processors. However, new signals are defined for the location of the INC pins in the Intel486 processor proliferations. All INC pins defined by Intel have a specific use for jumperless single socket compatibility with current and future processors. A system design could connect any signal to an INC pin without affecting the operation of the processor. However, the purpose of a specific INC pin should be understood before it is used. If not, the system design will sacrifice the ability to implement a jumperless (single socket) flexible motherboard.
3. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.

Table 3-4. Pin Assignment for 208-Lead SQFP Package of the IntelDX2™ Processor

Pin #	Description	Pin #	Description	Pin #	Description	Pin #	Description
1	V_{SS}	53	V_{SS}	105	V_{SS}	157	V_{SS}
2	V_{CC}	54	V_{CC}	106	V_{CC}	158	A24
3	V_{CC}(1)	55	V_{SS}	107	V_{SS}	159	A23
4	PCHK#	56	V_{CC}	108	D16	160	A22
5	BRDY#	57	V_{SS}	109	DP2	161	A21
6	BOFF#	58	SRESET	110	V_{SS}	162	V_{CC}
7	BS16#	59	SMIACT#	111	V_{CC}	163	V_{CC}
8	BS8#	60	V_{CC}	112	D15	164	A20
9	V_{CC}	61	V_{SS}	113	D14	165	A19
10	V_{SS}	62	V_{CC}	114	V_{CC}	166	A18
11	INC(2)	63	INC	115	V_{SS}	167	TMS
12	RDY#	64	INC	116	D13	168	TDI
13	KEN#	65	SMI#	117	D12	169	V_{CC}
14	V_{CC}	66	FERR#	118	D11	170	V_{SS}
15	V_{SS}	67	NC(3)	119	D10	171	A17
16	HOLD	68	TDO	120	V_{SS}	172	V_{CC}
17	AHOLD	69	V_{CC}	121	V_{CC}	173	A16
18	TCK	70	INC	122	V_{SS}	174	A15
19	V_{CC}	71	INC	123	D9	175	V_{SS}
20	V_{CC}	72	IGNNE#	124	D8	176	V_{CC}
21	V_{SS}	73	STPCLK#	125	DP1	177	A14
22	V_{CC}	74	D31	126	D7	178	A13
23	V_{CC}	75	D30	127	NC	179	V_{CC}
24	CLK	76	V_{SS}	128	V_{CC}	180	A12
25	V_{CC}	77	V_{CC}	129	D6	181	V_{SS}
26	HLDA	78	D29	130	D5	182	A11
27	W/R#	79	D28	131	V_{CC}	183	V_{CC}
28	V_{SS}	80	V_{CC}	132	V_{SS}	184	V_{SS}
29	V_{CC}	81	V_{SS}	133	V_{CC}	185	V_{CC}
30	BREQ	82	V_{CC}	134	V_{CC}	186	A10
31	BE0#	83	D27	135	V_{SS}	187	A9

Table 3-4. Pin Assignment for 208-Lead SQFP Package of the IntelDX2™ Processor (Cont'd)

Pin#	Description	Pin#	Description	Pin#	Description	Pin#	Description
32	BE1#	84	D26	136	V_{CC}	188	V_{CC}
33	BE2#	85	D25	137	V_{CC}	189	V_{SS}
34	BE3#	86	V_{CC}	138	V_{SS}	190	A8
35	V_{CC}	87	D24	139	V_{CC}	191	V_{CC}
36	V_{SS}	88	V_{SS}	140	D4	192	A7
37	M/IO#	89	V_{CC}	141	D3	193	A6
38	V_{CC}	90	DP3	142	D2	194	UP#
39	D/C#	91	D23	143	D1	195	A5
40	PWT	92	D22	144	D0	196	A4
41	PCD	93	D21	145	DP0	197	A3
42	V_{CC}	94	V_{SS}	146	V_{SS}	198	V_{CC}
43	V_{SS}	95	V_{CC}	147	A31	199	V_{SS}
44	V_{CC}	96	NC	148	A30	200	V_{CC}
45	V_{CC}	97	V_{SS}	149	A29	201	V_{SS}
46	EADS#	98	V_{CC}	150	V_{CC}	202	A2
47	A20M#	99	D20	151	A28	203	ADS#
48	RESET	100	D19	152	A27	204	BLAST#
49	FLUSH#	101	D18	153	A26	205	V_{CC}
50	INTR	102	V_{CC}	154	A25	206	PLOCK#
51	NMI	103	D17	155	V_{CC}	207	LOCK#
52	V_{SS}	104	V_{SS}	156	V_{SS}	208	V_{SS}

NOTES:
1. This pin location is for the V_{CC5} pin on the IntelDX4™ processor. For compatibility with 3.3V processors that have 5V safe input buffers (i.e., IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane. See section 3.2, "Quick Pin Reference," for a description of the V_{CC5} pin on the IntelDX4 processor.
2. INC. Internal No Connect. These pins are not connected to any internal pad in Intel486™ processors and OverDrive® processors. However, new signals are defined for the location of the INC pins in the Intel486 processor proliferations. All INC pins defined by Intel have a specific use for jumperless single socket compatibility with current and future processors. A system design could connect any signal to an INC pin without affecting the operation of the processor. However, the purpose of a specific INC pin should be understood before it is used. If not, the system design will sacrifice the ability to implement a jumperless (single socket) flexible motherboard.
3. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.

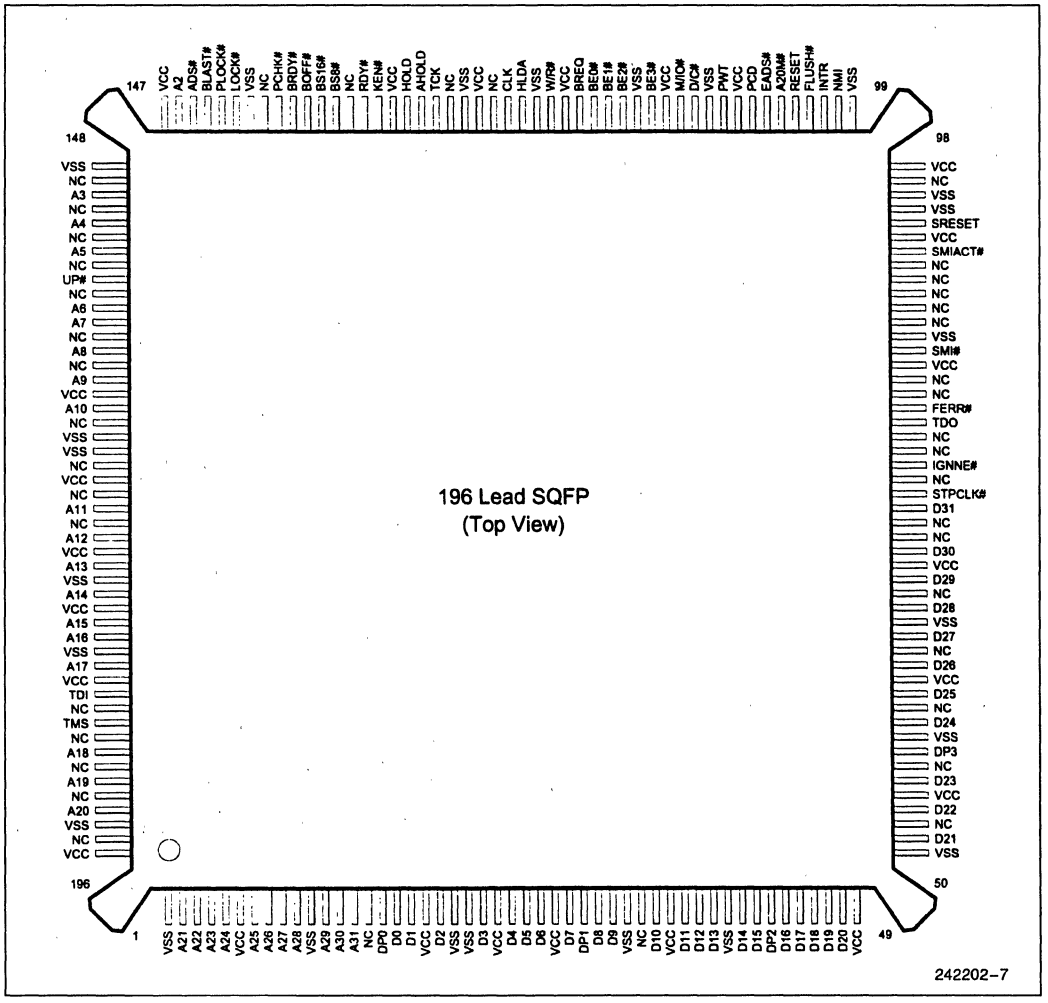


Figure 3-9. Package Diagram for 196-Lead PQFP Package of the Intel486™ DX Processor

Table 3-5. Pinout Differences for 196-Lead PQFP Package

Pin #	Previous Intel486™ SX Processor[3]	Low Power Intel486 SX Processor	Intel486 SX Processor	Previous Intel486 DX Processor[3]	Low Power Intel486 DX Processor	Intel486 DX Processor
75	INC[1]	INC	STPCLK#	INC	INC	STPCLK#
77	NC[2]	INC	INC	IGNNE#	IGNNE#	IGNNE#
81	NC	INC	INC	FERR#	FERR#	FERR#
85	INC	INC	SMI#	INC	INC	SMI#
92	INC	INC	SMIACT#	INC	INC	SMIACT#
94	INC	INC	SRESET	INC	INC	SRESET
127	INC	CLKSEL	NC	NC	CLKSEL	NC

NOTES:
1. INC. Internal No Connect. These pins are not connected to any internal pad in Intel486™ processors and OverDrive® processors. However, new signals are defined for the location of the INC pins in the Intel486 processor proliferations. All INC pins defined by Intel have a specific use for jumperless single socket compatibility with current and future processors. A system design could connect any signal to an INC pin without affecting the operation of the processor. However, the purpose of a specific INC pin should be understood before it is used. If not, the system design will sacrifice the ability to implement a jumperless (single socket) flexible motherboard.
2. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.
3. Previous versions of the Intel486 processor family do not implement SL Technology and are not described in this datasheet.

Table 3-6. Pin Assignments for Intel486™ DX Processor 196-Lead PQFP Package

Pin#	Description	Pin#	Description	Pin#	Description	Pin#	Description
1	V_{SS}	50	V_{SS}	99	V_{SS}	148	V_{SS}
2	A21	51	D21	100	NMI	149	NC
3	A22	52	NC	101	INTR	150	A3
4	A23	53	D22	102	FLUSH#	151	NC
5	A24	54	V_{CC}	103	RESET	152	A4
6	V_{CC}	55	D23	104	A20M#	153	NC
7	A25	56	NC	105	EADS#	154	A5
8	A26	57	DP3	106	PCD	155	NC
9	A27	58	V_{SS}	107	V_{CC}	156	UP#
10	A28	59	D24	108	PWT	157	NC
11	V_{SS}	60	NC	109	V_{SS}	158	A6
12	A29	61	D25	110	D/C#	159	A7
13	A30	62	V_{CC}	111	M/IO#	160	NC
14	A31	63	D26	112	V_{CC}	161	A8
15	NC	64	NC	113	BE3#	162	NC
16	DP0	65	D27	114	V_{SS}	163	A9
17	D0	66	V_{SS}	115	BE2#	164	V_{CC}
18	D1	67	D28	116	BE1#	165	A10
19	V_{CC}	68	NC	117	BE0#	166	NC
20	D2	69	D29	118	BREQ	167	V_{SS}
21	V_{SS}	70	V_{CC}	119	V_{CC}	168	V_{SS}
22	V_{SS}	71	D30	120	W/R#	169	NC
23	D3	72	NC	121	V_{SS}	170	V_{CC}
24	V_{CC}	73	NC	122	HLDA	171	NC
25	D4	74	D31	123	CLK	172	A11
26	D5	75	STPCLK#	124	NC	173	NC
27	D6	76	NC	125	V_{CC}	174	A12
28	V_{CC}	77	IGNNE#	126	V_{SS}	175	V_{CC}
29	D7	78	NC	127	NC	176	A13

Table 3-6. Pin Assignments for Intel486™ DX Processor 196-Lead PQFP Package (Continued)

Pin#	Description	Pin#	Description	Pin#	Description	Pin#	Description
30	DP1	79	NC	128	TCK	177	V_{SS}
31	D8	80	TDO	129	AHOLD	178	A14
32	D9	81	FERR#	130	HOLD	179	V_{CC}
33	V_{SS}	82	NC	131	V_{CC}	180	A15
34	NC	83	NC	132	KEN#	181	A16
35	D10	84	V_{CC}	133	RDY#	182	V_{SS}
36	V_{CC}	85	SMI#	134	NC	183	A17
37	D11	86	V_{SS}	135	BS8#	184	V_{CC}
38	D12	87	NC	136	BS16#	185	TDI
39	D13	88	NC	137	BOFF#	186	NC
40	V_{SS}	89	NC	138	BRDY#	187	TMS
41	D14	90	NC	139	PCHK#	188	NC
42	D15	91	NC	140	NC	189	A18
43	DP2	92	SMIACT#	141	V_{SS}	190	NC
44	D16	93	V_{CC}	142	LOCK#	191	A19
45	D17	94	SRESET	143	PLOCK#	192	NC
46	D18	95	V_{SS}	144	BLAST#	193	A20
47	D19	96	V_{SS}	145	ADS#	194	V_{SS}
48	D20	97	NC	146	A2	195	NC
49	V_{CC}	98	V_{CC}	147	V_{CC}	196	V_{CC}

3.2 Quick Pin Reference

The following is a brief pin description. For detailed signal descriptions refer to section 9.2, "Signal Description."

Table 3-7. Intel486™ Processor Pin Descriptions

Symbol	Type	Name and Function
CLK	I	**CLocK** provides the fundamental timing and the internal operating frequency for the Intel486 processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 **A2–A3**	I/O O	The **Address Lines.** A31–A2, together with the byte enables signals. BE0#–BE3#, define the physical area of memory or input/output space accessed. Address lines A31–A4 are used to drive addresses into the processor to perform cache line invalidations. Input signals must meet setup and hold times t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.
BE0–3#	O	The **Byte Enable** signals indicate active bytes during read and write cycles. During the first cycle of a cache fill, the external system should assume that all byte enables are active. BE3# applies to D24–D31, BE2# applies to D16–D23, BE1# applies to D8–D15 and BE0# applies to D0–D7. BE0#–BE3# are active LOW and are not driven during bus hold.
DATA BUS		
D31–D0	I/O	The **Data Lines,** D0–D7, define the least significant byte of the data bus while lines D24–D31 define the most significant byte of the data bus. These signals must meet setup and hold times t_{22} and t_{23} for proper operation on reads. These pins are driven during the second and subsequent clocks of write cycles.
DATA PARITY		
DP0–DP3	I/O	There is one **Data Parity** pin for each byte of the data bus. Data parity is generated on all write data cycles with the same timing as the data driven by the Intel486™ processor. Even parity information must be driven back into the processor on the data parity pins with the same timing as read information to insure that the correct parity check status is indicated by the Intel486 processor. The signals read on these pins do not affect program execution. Input signals must meet setup and hold times t_{22} and t_{23}. DP0–DP3 should be connected to V_{CC} through a pull-up resistor in systems that do not use parity. DP0–DP3 are active HIGH and are driven during the second and subsequent clocks of write cycles.
PCHK#	O	**Parity Status** is driven on the PCHK# pin the clock after ready for read operations. The parity status is for data sampled at the end of the previous clock. A parity error is indicated by PCHK# being LOW. Parity status is only checked for enabled bytes as indicated by the byte enable and bus size signals. PCHK# is valid only in the clock immediately after read data is returned to the processor. At all other times PCHK# is inactive (HIGH). PCHK# is never floated.

Table 3-7. Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
BUS CYCLE DEFINITION		
M/IO# **D/C#** **W/R#**	O O O	The **memory/input-output, data/control** and **write/read** lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted.

M/IO#	D/C#	W/R#	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Halt/Special Cycle
0	1	0	I/O Read
0	1	1	I/O Write
1	0	0	Code Read
1	0	1	Reserved
1	1	0	Memory Read
1	1	1	Memory Write

The bus definition signals are not driven during bus hold and follow the timing of the address bus. Refer to section 10.2.11, "Special Bus Cycles," for a description of the special bus cycles.

LOCK#	O	The **Bus Lock** pin indicates that the current bus cycle is locked. The Intel486 processor will not allow a bus hold when LOCK# is asserted (but address holds are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the last clock of the last locked bus cycle. The last locked cycle ends when ready is returned. LOCK# is active LOW and is not driven during bus hold. Locked read cycles will not be transformed into cache fill cycles if KEN# is returned active.
PLOCK#	O	The **Pseudo-Lock** pin indicates that the current bus transaction requires more than one bus cycle to complete. For the Intel486 processor, examples of such operations are segment table descriptor reads (64 bits), in addition to cache line fills (128 bits). For Intel486 processors with on-chip Floating-Point Unit, floating-point long reads and write (64 bits) also require more than one bus cycle to complete. The Intel486 processor will drive PLOCK# active until the addresses for the last bus cycle of the transaction have been driven regardless of whether RDY# or BRDY# have been returned. Normally PLOCK# and BLAST# are inverse of each other. However during the first bus cycle of a 64-bit floating-point write (for Intel486 processors with on-chip Floating-Point Unit), both PLOCK# and BLAST# will be asserted. PLOCK# is a function of the BS8#, BS16# and KEN# inputs. PLOCK# should be sampled only in the clock ready is returned. PLOCK# is active LOW and is not driven during bus hold.

Table 3-7. Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
BUS CONTROL		
ADS#	O	The **Address Status** output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus. ADS# is driven active in the same clock as the addresses are driven. ADS# is active LOW and is not driven during bus hold.
RDY#	I	The **Non-burst Ready** input indicates that the current bus cycle is complete. RDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted data from the Intel486 processor in response to a write. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. RDY# is active during address hold. Data can be returned to the processor while AHOLD is active. RDY# is active LOW, and is not provided with an internal pull-up resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.
BURST CONTROL		
BRDY#	I	The **Burst Ready** input performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data in response to a read or that the external system has accepted data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle. BRDY# is sampled in the second and subsequent clocks of a burst cycle. The data presented on the data bus will be strobed into the processor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. BRDY# is active LOW and is provided with a small pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.
BLAST#	O	The **Burst Last** signal indicates that the next time BRDY# is returned the burst bus cycle is complete. BLAST# is active for both burst and non-burst bus cycles. BLAST# is active LOW and is not driven during bus hold.
INTERRUPTS		
RESET	I	The **Reset** input forces the Intel486 processor to begin execution at a known state. The processor cannot begin execution of instructions until at least 1 ms after V_{CC} and CLK have reached their proper DC and AC specifications. The RESET pin should remain active during this time to insure proper processor operation. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

Table 3-7. Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
INTERRUPTS (Continued)		
INTR	I	The **Maskable Interrupt** indicates that an external interrupt has been generated. If the internal interrupt flag is set in EFLAGS, active interrupt processing will be initiated. The Intel486 processor will generate two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the interrupt acknowledges have been performed to assure that the interrupt is recognized. INTR is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
NMI	I	The **Non-Maskable Interrupt** request signal indicates that an external non-maskable interrupt has been generated. NMI is rising edge sensitive. NMI must be held LOW for at least four CLK periods before this rising edge. NMI is not provided with an internal pull-down resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SRESET	I	The **Soft Reset pin** duplicates all the functionality of the RESET pin with the following two exceptions: 1. The SMBASE register will retain its previous value. 2. If UP# (I) is asserted, SRESET will not have an effect on the host processor. For soft resets, SRESET should remain active for at least 15 CLK periods. SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SMI#	I	The **System Management Interrupt** input is used to invoke the System Management Mode (SMM). SMI# is a falling edge triggered signal which forces the processor into SMM at the completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# does not break LOCKed bus cycles and cannot interrupt a currently executing SMM. The processor will latch the falling edge of one pending SMI# signal while the processor is executing an existing SMI#. The nested SMI# will not be recognized until after the execution of a Resume (RSM) instruction.
SMIACT#	O	The **System Management Interrupt ACTive** is an active low output, indicating that the processor is operating in SMM. It is asserted when the processor begins to execute the SMI# state save sequence and will remain active LOW until the processor executes the last state restore cycle out of SMRAM.
STPCLK#	I	The **SToP CLocK request** input signal indicates a request has been made to turn off the CLK input. When the processor recognizes a STPCLK#, the processor will stop execution on the next instruction boundary, unless superseded by a higher priority interrupt, empty all internal pipelines and the write buffers and generate a Stop Grant acknowledge bus cycle. STPCLK# is active LOW and is provided with an internal pull-up resistor. **STPCLK# is an asynchronous signal, but must remain active until the processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle.**

Table 3-7. Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
BUS ARBITRATION		
BREQ	O	The **Bus Request** signal indicates that the Intel486 processor has internally generated a bus request. BREQ is generated whether or not the Intel486 processor is driving the bus. BREQ is active HIGH and is never floated.
HOLD	I	The **Bus Hold** *request* allows another bus master complete control of the processor bus. In response to HOLD going active the Intel486 processor will float most of its output and input/output pins. HLDA will be asserted after completing the current bus cycle, burst cycle or sequence of locked cycles. The Intel486 processor will remain in this state until HOLD is de-asserted. HOLD is active high and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper operation.
HLDA	O	**Hold Acknowledge** goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the Intel486 processor has given the bus to another local bus master. HLDA is driven active in the same clock that the Intel486 processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.
BOFF#	I	The **Backoff** input forces the Intel486 processor to float its bus in the next clock. The processor will float all pins normally floated during bus hold but HLDA will not be asserted in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#; if both are returned in the same clock, BOFF# takes effect. The processor remains in bus hold until BOFF# is negated. If a bus cycle was in progress when BOFF# was asserted the cycle will be restarted. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper operation.
CACHE INVALIDATION		
AHOLD	I	The **Address Hold** request allows another bus master access to the processor's address bus for a cache invalidation cycle. The Intel486 processor will stop driving its address bus in the clock following AHOLD going active. Only the address bus will be floated during address hold, the remainder of the bus will remain active. AHOLD is active HIGH and is provided with a small internal pull-down resistor. For proper operation AHOLD must meet setup and hold times t_{18} and t_{19}.
EADS#	I	This signal indicates that a *valid* **External Address** has been driven onto the Intel486 processor address pins. This address will be used to perform an internal cache invalidation cycle. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy setup and hold times t_{12} and t_{13} for proper operation.
CACHE CONTROL		
KEN#	I	The **Cache Enable** pin is used to determine whether the current cycle is cacheable. When the Intel486 processor generates a cycle that can be cached and KEN# is active one clock before RDY# or BRDY# during the first transfer of the cycle, the cycle will become a cache line fill cycle. Returning KEN# active one clock before RDY# during the last read in the cache line fill will cause the line to be placed in the on-chip cache. KEN# is active LOW and is provided with a small internal pull-up resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	The **Cache Flush** input forces the Intel486 processor to flush its entire internal cache. FLUSH# is active low and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition in any specific clock.

Table 3-7. Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
PAGE CACHEABILITY		
PWT **PCD**	O O	The **Page Write-Through** and **Page Cache Disable** pins reflect the state of the page attribute bits, PWT and PCD, in the page table entry, page directory entry or control register 3 (CR3) when paging is enabled. If paging is disabled, the processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT pins. PWT and PCD have the same timing as the cycle definition pins (M/IO#, D/C#, and W/R#). PWT and PCD are active HIGH and are not driven during bus hold. PCD is masked by the cache disable bit (CD) in Control Register 0.
BUS SIZE CONTROL		
BS16# **BS8#**	I I	The **Bus Size 16** and **Bus Size 8** pins (bus sizing pins) cause the Intel486 processor to run multiple bus cycles to complete a request from devices that cannot provide or accept 32 bits of data in a single cycle. The bus sizing pins are sampled every clock. The state of these pins in the clock before ready is used by the Intel486 processor to determine the bus size. These signals are active LOW and are provided with internal pull-up resistors. These inputs must satisfy setup and hold times t_{14} and t_{15} for proper operation.
ADDRESS MASK		
A20M#	I	When the **Address Bit 20 Mask** pin is asserted, the Intel486 processor masks physical address bit 20 (A20) before performing a lookup to the internal cache or driving a memory cycle on the bus. A20M# emulates the address wraparound at one Mbyte, which occurs on the 8086 processor. A20M# is active LOW and should be asserted only when the processor is in real mode. This pin is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For proper operation, A20M# should be sampled high at the falling edge of RESET.
TEST ACCESS PORT		
TCK	I	**Test ClocK** is an input to the Intel486 processor and provides the clocking function required by the JTAG Boundary scan feature. TCK is used to clock state information and data into component on the rising edge of TCK on TMS and TDI, respectively. Data is clocked out of the part on the falling edge of TCK and TDO. TCK is provided with an internal pull-up resistor.
TDI	I	**Test Data Input** is the serial input used to shift JTAG instructions and data into component. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and SHIFT-DR TAP controller states. During all other tap controller states, TDI is a "don't care." TDI is provided with an internal pull-up resistor.
TDO	O	**Test Data Output** is the serial output used to shift JTAG instructions and data out of the component. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state.
TMS	I	**Test Mode Select** is decoded by the JTAG TAP (Tap Access Port) to select the operation of the test logic. TMS is sampled on the rising edge of TCK. To guarantee deterministic behavior of the TAP controller TMS is provided with an internal pull-up resistor.

Table 3-7. Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
PERFORMANCE UPGRADE SUPPORT		
UP#	I	The **Upgrade Present** input detects the presence of the upgrade processor, then powers down the core, and tri-states all outputs of the original processor, so that the original processor consumes very low current. UP# is active LOW and sampled at all times, including after power-up and during reset.
NUMERIC ERROR REPORTING FOR Intel486 DX, INTELDX2™, AND IntelDX4™ PROCESSORS		
FERR#	O	The **Floating-Point ERRor** pin is driven active when a floating-point error occurs. FERR# is similar to the ERROR# pin on the Intel387™ Math CoProcessor. FERR# is included for compatibility with systems using DOS type floating-point error reporting. FERR# will not go active if FP errors are masked in FPU register. FERR# is active LOW, and is not floated during bus hold.
IGNNE#	I	When the **IGNore Numeric Error** pin is asserted the processor will ignore a numeric error and continue executing non-control floating-point instructions, but FERR# will still be activated by the processor. When IGNNE# is de-asserted the processor will freeze on a non-control floating-point instruction, if a previous floating-point instruction caused an error. IGNNE# has no effect when the NE bit in control register 0 is set. IGNNE# is active LOW and is provided with a small internal pull-up resistor. IGNNE# is asynchronous but setup and hold times t_{20} and t_{21} must be met to insure recognition on any specific clock.
WRITE-BACK ENHANCED IntelDX4 AND WRITE-BACK ENHANCED IntelDX2 PROCESSORS SIGNAL PINS		
CACHE#	O	The **CACHE#** output indicates internal cacheability on read cycles and burst write-back on write cycles. CACHE# is asserted for cacheable reads, cacheable code fetches and write-backs. It is driven inactive for non-cacheable reads, I/O cycles, special cycles, and write-through cycles.
FLUSH#	I	**Cache FLUSH#** is an existing pin that operates differently if the processor is configured as Enhanced Bus mode (write-back). FLUSH# will cause the processor to write back all modified lines and flush (invalidate) the cache. FLUSH# is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
HITM#	O	The **Hit/Miss to a Modified Line** pin is a cache coherency protocol pin that is driven only in Enhanced Bus mode. When a snoop cycle is run, HITM# indicates that the processor contains the snooped line and that the line has been modified. Assertion of HITM# implies that the line will be written back in its entirety, unless the processor is already in the process of doing a replacement write-back of the same line.
INV	I	The **Invalidation Request** pin is a cache coherency protocol pin that is used only in the Enhanced Bus mode. It is sampled by the processor on EADS#-driven snoop cycles. It is necessary to assert this pin to get the effect of the processor invalidate cycle on write-through-only lines. INV also invalidates the write-back lines. However, if the snooped line is modified, the line will be written back and then invalidated. INV must satisfy setup and hold times t_{12} and t_{13} for proper operation.

Table 3-7. Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
WRITE-BACK ENHANCED IntelDX4 AND WRITE-BACK ENHANCED IntelDX2 PROCESSORS SIGNAL PINS (Continued)		
PLOCK#	O	In the Enhanced bus mode, **Pseudo-Lock Output** is always driven inactive. In this mode, a 64-bit data read (caused by an FP operand access or a segment descriptor read) is treated as a multiple cycle read request, which may be a burst or a non-burst access based on whether BRDY# or RDY# is returned by the system. Because only write-back cycles (caused by Snoop write-back or replacement write-back) are write burstable, a 64-bit write will be driven out as two non-burst bus cycles. BLAST# is asserted during both writes. Refer to the Bus Functional Description section 10.3 for details on Pseudo-Locked bus cycles.
SRESET	I	For the Write-Back Enhanced IntelDX2 processor, **Soft RESET** operates similar to other Intel486 processors. On SRESET, the internal SMRAM base register retains its previous value, does not flush, write-back or disable the internal cache. Because SRESET is treated as an interrupt, it is possible to have a bus cycle while SRESET is asserted. SRESET is serviced only on an instruction boundary. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
WB/WT#	I	The **Write-Back/Write-Through** pin enables Enhanced Bus mode (write-back cache). It also defines a cached line as write-through or write-back. For cache configuration, WB/WT# must be valid during RESET and be active for at least two clocks before and two clocks after RESET is de-asserted. To define write-back or write-through configuration of a line, WB/WT# is sampled in the same clock as the first RDY# or BRDY# is returned during a line fill (allocation) cycle.
IntelDX4 PROCESSOR CLKMUL, VCC5, AND VOLDET		
CLKMUL	I	The **CLocK MULtiplier** input, defined during device RESET, defines the ratio of internal core frequency to external bus frequency. If sampled low, the core frequency operates at twice the external bus frequency (speed doubled mode). If driven high or left floating, speed triple mode is selected. CLKMUL has an internal pull-up speed to V_{CC} and may be left floating in designs that select speed tripled clock mode.
V_{CC5}	I	The **5V reference voltage** input is the reference voltage for the 5V-tolerant I/O buffers. This signal should be connected to +5V ±5% for use with 5V logic. If all inputs are from 3V logic, this pin should be connected to 3.3V.
VOLDET	O	A **VOLtage DETect** signal allows external system logic to distinguish between a 5V Intel486 processor and the 3.3V IntelDX4 processor. This signal is active low for a 3.3V IntelDX4 processor. This pin is available only on the PGA version of the IntelDX4 processor.

Table 3-8. Output Pins

Name	Active Level	When Floated
BREQ	HIGH	
HLDA	HIGH	
BE3#–BE0#	LOW	Bus Hold
PWT, PCD	HIGH/LOW	Bus Hold
W/R#, M/IO#, D/C#	HIGH/LOW	Bus Hold
LOCK#	LOW	Bus Hold
PLOCK#	LOW	Bus Hold
ADS#	LOW	Bus Hold
BLAST#	LOW	Bus Hold
PCHK#	LOW	
FERR#(1)	LOW	
A3–A2	N/A	Bus, Address Hold
SMIACT#(2)	LOW	
CACHE#(3)	LOW	Bus, Address Hold
HITM#(3)	LOW	Bus, Address Hold
VOLDET(4)	LOW	

NOTES:
1. Present on the Intel486™ DX, IntelDX2™, and IntelDX4™ processors only.
2. Not present in the 50-MHz Intel486 DX processor.
3. Present on the Write-Back Enhanced IntelDX4 and Write-Back Enhanced IntelDX2 processors only.
4. Present on the IntelDX4 processor only.

Table 3-9. Input/Output Pins(1)

Name	Active Level	When Floated
D31–D0	HIGH/LOW	Bus Hold
DP3–DP0	HIGH	Bus Hold
A31–A4	HIGH/LOW	Bus, Address Hold

NOTE:
1. All input/output signals are floated when UP# is asserted.

Table 3-10. Test Pins

Name	Input or Output	Sampled/Driven On
TCK	Input	N/A
TDI	Input	Rising Edge of TCK
TDO	Output	Falling Edge of TCK
TMS	Input	Rising Edge of TCK

NOTE:
1. The test pins are not present on the Intel486 SX™ processor in the PGA package.

Table 3-11. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Pull-Up/ Pull-Down
CLK, CLK2(1)			
RESET	HIGH	Asynchronous	
SRESET	HIGH	Asynchronous	Pull-Down
HOLD	HIGH	Synchronous	
AHOLD	HIGH	Synchronous	Pull-Down
EADS#	LOW	Synchronous	Pull-Up
BOFF#	LOW	Synchronous	Pull-Up
FLUSH#	LOW	Asynchronous	Pull-Up
A20M#	LOW	Asynchronous	Pull-Up
BS16#, BS8#	LOW	Synchronous	Pull-Up
KEN#	LOW	Synchronous	Pull-Up
RDY#	LOW	Synchronous	
BRDY#	LOW	Synchronous	Pull-Up
INTR	HIGH	Asynchronous	
NMI	HIGH	Asynchronous	
IGNNE#(2)	LOW	Asynchronous	Pull-Up
SMI#(3)	LOW	Asynchronous	Pull-Up
STPCLK(2)#	LOW	Asynchronous	Pull-Up
UP#	LOW		Pull-Up
TCK(4)	HIGH		Pull-Up
TDI(4)	HIGH		Pull-Up
TMS(4)	HIGH		Pull-Up
INV(5)	HIGH	Synchronous	Pull-Up
WB/WT#(5)	HIGH/ LOW	Synchronous	Pull-Down
CLKMUL#(6)	N/A		Pull-Up

NOTES:
1. CLK2 is present on 2X clock mode Intel486™ SX and Intel486 DX processors.
2. Present on the Intel486 DX, IntelDX2™, and IntelDX4™ processors only.
3. Not present in the 50-MHz Intel486 DX processor.
4. The test pins are not present on the Intel486 SX processor in the PGA package.
5. Present on the Write-Back Enhanced IntelDX4 and Write-Back Enhanced IntelDX2 processors only.
6. Present on the IntelDX4 processor only.

4.0 ARCHITECTURAL OVERVIEW

4.1 Introduction

The Intel486 processor family is a 32-bit architecture with on-chip memory management, Floating-Point, and cache memory units. Figure 4-1 is a block diagram of the Intel486 processor family. The Intel486 processor contains all the features of the Intel386™ processor with enhancements to increase performance.

The Intel486 processor instruction set includes the complete Intel386 processor instruction set along with extensions to serve new applications and increase performance. The on-chip memory management unit (MMU) is completely compatible with the Intel386 processor MMU. Software written for previous members of the Intel architecture family will run on the Intel486 processor without any modifications.

On-chip cache memory allows frequently used data and code to be stored on-chip reducing accesses to the external bus. RISC design techniques reduce instruction cycle times. A burst bus feature enables fast cache fills.

The memory management unit (MMU) consists of a segmentation unit and a paging unit. Segmentation allows management of the logical address space by providing easy data and code relocatibility and efficient sharing of global resources. The paging mechanism operates beneath segmentation and is transparent to the segmentation process. Paging is optional and can be disabled by system software. Each segment can be divided into one or more 4-Kbyte segments. To implement a virtual memory system, full restartability for all page and segment faults is supported.

Memory is organized into one or more variable length segments, each up to four Gbytes (232 bytes) in size. A segment can have attributes associated with it which include its location, size, type (i.e., stack, code or data), and protection characteristics. Each task on an Intel486 processor can have a maximum of 16,381 segments and each are up to four Gbytes in size. Thus, each task has a maximum of 64 terabytes (trillion bytes) of virtual memory.

The segmentation unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of software integrity.

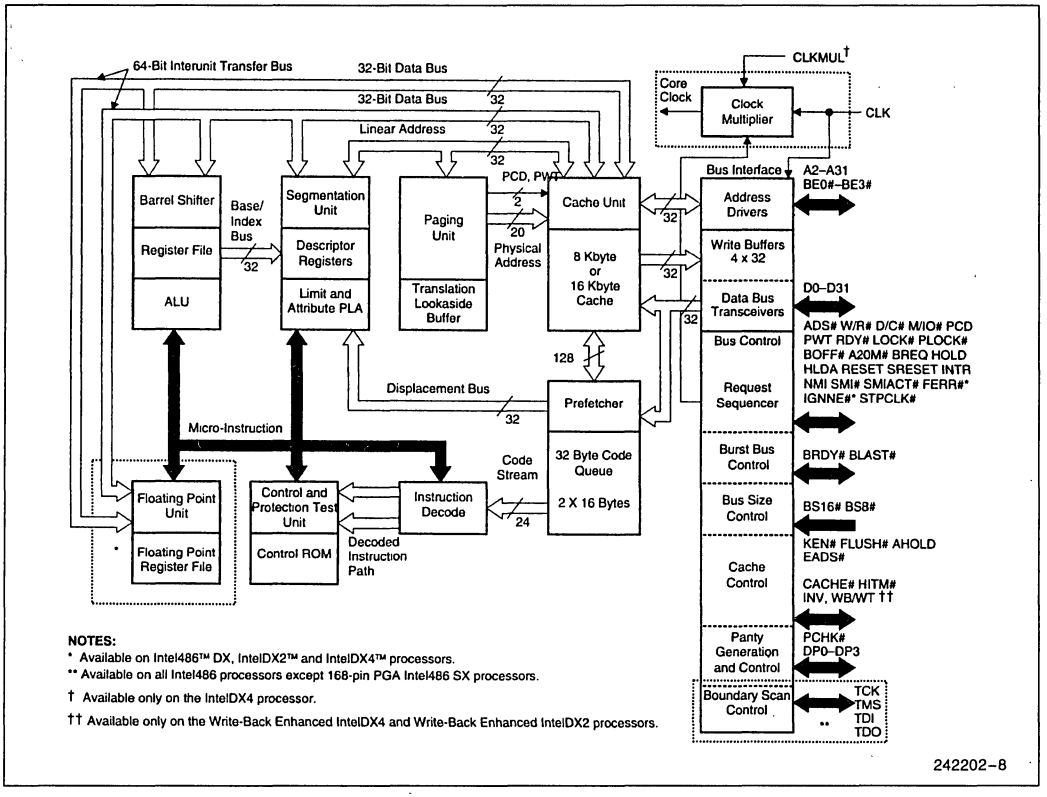
The Intel486 processor has two modes of operation: Real Address Mode (Real Mode) and Protected Mode Virtual Address Mode (Protected Mode). In Real Mode the Intel486 processor operates as a very fast 8086. Real Mode is required primarily to set up the Intel486 processor for Protected Mode operation. Protected Mode provides access to the sophisticated memory management paging and privilege capabilities of the processor.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each Virtual 8086 task behaves with 8086 semantics, allowing 8086 processor software (an application program or an entire operating system) to execute.

System Management Mode (SMM) provides the system designer with a means of adding new software controlled features to their computer products that always operate transparently to the Operating System (OS) and software applications. SMM is intended for use only by system firmware, not by applications software or general purpose systems software.

The on-chip cache is 16 Kbytes in size for the IntelDX4 processor and 8 Kbytes in size for all other members of the Intel486 processor family. It is 4-way set associative and follows a write-through policy. The on-chip cache includes features to provide flexibility in external memory system design. Individual pages can be designated as cacheable or non-cacheable by software or hardware. The cache can also be enabled and disabled by software or hardware. The Write-Back Enhanced Intel486 processors can be set to use an on-chip write-back cache policy.

The Intel486 processor also has features that facilitate high-performance hardware designs. The 1X bus clock input eases high-frequency board-level designs. The clock multiplier on IntelSX2, IntelDX2, and IntelDX4 processors improves execution performance without increasing board design complexity. The clock multiplier enhances all operations operating out of the cache and/or not blocked by external bus accesses. The burst bus feature enables fast cache fills.



NOTES:
* Available on Intel486™ DX, IntelDX2™ and IntelDX4™ processors.
** Available on all Intel486 processors except 168-pin PGA Intel486 SX processors.
† Available only on the IntelDX4 processor.
†† Available only on the Write-Back Enhanced IntelDX4 and Write-Back Enhanced IntelDX2 processors.

Figure 4-1. Intel486™ Processor Block Diagram

4.1.1 Intel486 DX, IntelDX2™, AND IntelDX4™ PROCESSOR ON-CHIP FLOATING-POINT UNIT

The Intel486 DX, IntelDX2, and IntelDX4 processors incorporate the basic Intel486 processor 32-bit architecture with on-chip memory management and cache memory units. They also have an on-chip Floating-Point Unit (FPU) that operates in parallel with the arithmetic and logic unit. The FPU provides arithmetic instructions for a variety of numeric data types and executes numerous built-in transcendental functions (e.g., tangent, sine, cosine, and log functions). The Floating-Point Unit fully conforms to the ANSI/IEEE standard 754-1985 for Floating-Point arithmetic.

All software written for the Intel386 processor, Intel387 math coprocessor and previous members of the 86/87 architectural family will run on these processors without any modifications.

4.1.2 UPGRADE POWER DOWN MODE

Upgrade Power Down Mode on the Intel486 processor is initiated by the Intel OverDrive processor using the UP# (upgrade present) pin. Upon sensing the presence of the Intel OverDrive Processor, the Intel486 processor tri-states its outputs and enters the "Upgrade Power Down Mode," lowering its power consumption. The UP# pin of the Intel486 processor is driven active (low) by the UP# pin of the Intel OverDrive processor.

4.2 Register Set

The Intel486 processor register set can be split into the following categories:

- Base Architecture Registers
  - — General Purpose Registers
  - — Instruction Pointer

- — Flags Register
- — Segment Registers
- Systems Level Registers
  - — Control Registers
  - — System Address Registers
- Debug and Test Registers

The base architecture and Floating-Point registers (see below) are accessible by the applications program. The system level registers can only be accessed at privilege level 0 and used by system level programs. The debug and test registers also can only be accessed at privilege level 0.

4.2.1 FLOATING-POINT REGISTERS

In addition to the registers listed above, the Intel486 DX, IntelDX2, and IntelDX4 processors also have the following:

- Floating-Point Registers
  - — Data Registers
  - — Tag Word
  - — Status Word
  - — Instruction and Data Pointers
  - — Control Word

4.2.2 BASE ARCHITECTURE REGISTERS

Figure 4-2 shows the Intel486 processor base architecture registers. The contents of these registers are task-specific and are automatically loaded with a new context upon a task switch operation.

The base architecture includes six directly accessible descriptors, each specifying a segment up to 4 Gbytes in size. The descriptors are indicated by the selector values placed in the Intel486 processor segment registers. Various selector values can be loaded as a program executes.

The selectors are also task-specific, so the segment registers are automatically loaded with new context upon a task switch operation.

NOTE:
In register descriptions, "set" means "set to 1," and "reset" means "reset to 0."

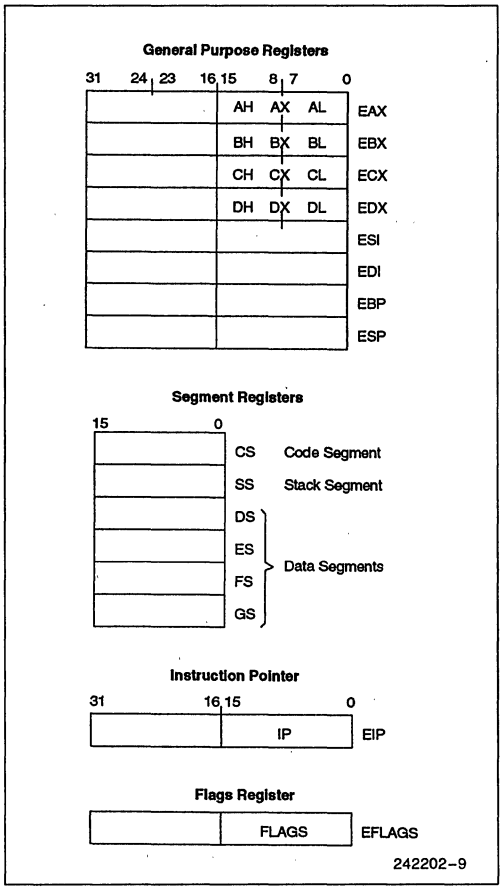


Figure 4-2. Base Architecture Registers

4.2.2.1 General Purpose Registers

The eight 32-bit general purpose registers are shown in Figure 4-2. These registers hold data or address quantities. The general purpose registers can support data operands of 1, 8, 16 and 32 bits, and bit fields of 1 to 32 bits. Address operands of 16 and 32 bits are supported. The 32-bit registers are named EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP.

The least significant 16 bits of the general purpose registers can be accessed separately by using the 16-bit names of the registers AX, BX, CX, DX, SI, DI, BP and SP. The upper 16 bits of the register are not changed when the lower 16 bits are accessed separately.

Finally, 8-bit operations can individually access the lower byte (bits 0–7) and the highest byte (bits 8–15) of the general purpose registers AX, BX, CX and DX. The lowest bytes are named AL, BL, CL and DL respectively. The higher bytes are named AH, BH, CH and DH respectively. The individual byte accessibility offers additional flexibility for data operations, but is not used for effective address calculation.

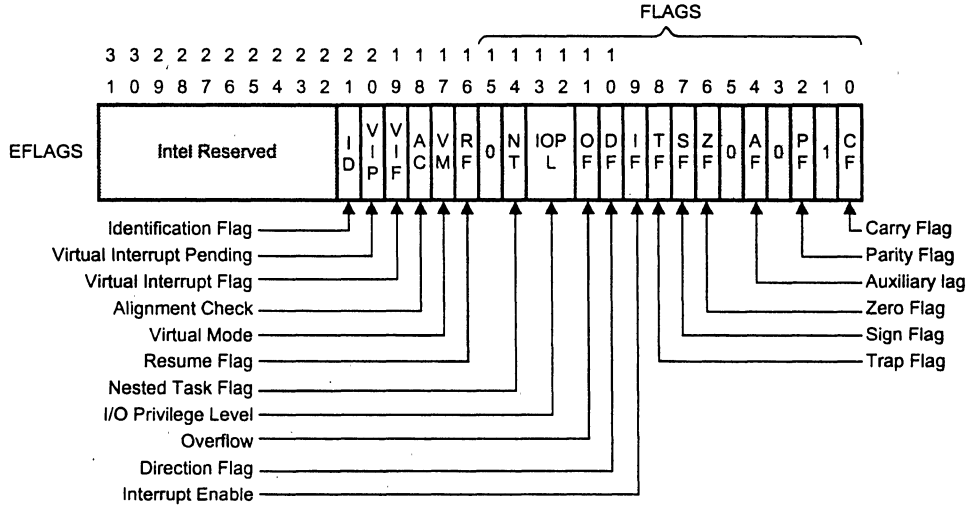
4.2.2.2 Instruction Pointer

The instruction pointer shown in Figure 4-2 is a 32-bit register named EIP. EIP holds the offset of the next instruction to be executed. The offset is always relative to the base of the code segment (CS). The lower 16 bits (bits 0–15) of the EIP contain the 16-bit instruction pointer named IP, which is used for 16-bit addressing.

4.2.2.3 Flags Register

The flags register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS control certain operations and indicate status of the Intel486 processor. The lower 16 bits (bit 0–15) of EFLAGS contain the 16-bit register named FLAGS, which is most useful when executing 8086 and 80286 processor code. EFLAGS is shown in Figure 4-3.

EFLAGS bits 1, 3, 5, 15 and 22–31 are defined as "Intel Reserved." When these bits are stored during interrupt processing or with a PUSHF instruction (push flags onto stack), a one is stored in bit 1 and zeros in bits 3, 5, 15 and 22–31.



NOTE:
See section 4.2.7 "Compatibility."

Figure 4-3. Flag Registers

ID (Identification Flag, bit 21)

The ability of a program to set and clear the ID flag indicates that the processor supports the CPUID instruction. (Refer to section 13, "Instruction Set Summary," and Appendix B, "Feature Determination: CPUID Instruction.")

VIP (Virtual Interrupt Pending Flag, bit 20)

The VIP flag together with the VIF enable each applications program in a multitasking environment to have virtualized versions of the system's IF flag. For more on the use of this flag in virtual-8086 mode and in protected mode. (Refer to Appendix A, "Advanced Features.")

VIF (Virtual Interrupt Flag, bit 19)

The VIF is a virtual image of IF (the interrupt flag) used with VIP. For more on the use of this flag in virtual-8086 mode and in protected mode. (Refer to Appendix A, "Advanced Features.")

AC (Alignment Check, bit 18)

The AC bit is defined in the upper 16 bits of the register. It enables the generation of faults if a memory reference is to a misaligned address. Alignment faults are enabled when AC is set to 1. A misaligned address is a word access an odd address, a dword access to an address that is not on a dword boundary, or an 8-byte reference to an address that is not on a 64-bit word boundary. (See section 10.1.5, "Operand Alignment.")

Alignment faults are only generated by programs running at privilege level 3. The AC bit setting is ignored at privilege levels 0, 1 and 2. Note that references to the descriptor tables (for selector loads), or the task state segment (TSS), are implicitly level 0 references even if the instructions causing the references are executed at level 3. Alignment faults are reported through interrupt 17, with an error code of 0. Table 4-1 gives the alignment required for the Intel486 processor data types.

Table 4-1. Data Type Alignment Requirements

Memory Access	Alignment (Byte Boundary)
Word	2
Dword	4
Single Precision Real	4
Double Precision Real	8
Extended Precision Real	8
Selector	2
48-Bit Segmented Pointer	4
32-Bit Flat Pointer	4
32-Bit Segmented Pointer	2
48-Bit "Pseudo-Descriptor"	4
FSTENV/FLDENV Save Area	4/2 (On Operand Size)
FSAVE/FRSTOR Save Area	4/2 (On Operand Size)
Bit String	4

IMPLEMENTATION NOTE:
Several instructions on the Intel486 processor generate misaligned references, even if their memory address is aligned. For example, on the Intel486 processor, the SGDT/SIDT (store global/interrupt descriptor table) instruction reads/writes two bytes, and then reads/writes four bytes from a "pseudo-descriptor" at the given address. The Intel486 processor will generate misaligned references unless the address is on a 2 mod 4 boundary. The FSAVE and FRSTOR instructions (Floating-Point save and restore state) will generate misaligned references for one-half of the register save/restore cycles. The Intel486 processor will not cause any AC faults if the effective address given in the instruction has the proper alignment.

VM (Virtual 8086 Mode, bit 17)

The VM bit provides Virtual 8086 Mode within Protected Mode. If set while the Intel486 processor is in Protected Mode, the Intel486 processor will switch to Virtual 8086 operation, handling segment loads as the 8086 processor does, but generating exception 13 faults on privileged opcodes. The VM bit can be set only in Protected Mode, by the IRET instruction (if current privilege level = 0) and by task switches at any privilege level. The VM bit is unaffected by POPF. PUSHF always pushes a 0 in this bit, even if executing in Virtual 8086 Mode. The EFLAGS image pushed during interrupt processing or saved during task switches will contain a 1 in this bit if the interrupted code was executing as a Virtual 8086 Task.

RF (Resume Flag, bit 16)

The RF flag is used in conjunction with the debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. When RF is set, it causes any debug fault to be ignored on the next instruction. RF is then automatically reset at the successful completion of every instruction (no faults are signaled) except the IRET instruction, the POPF instruction, (and JMP, CALL, and INT instructions causing a task switch). These instructions set RF to the value specified by the memory image. For example, at the end of the breakpoint service routine, the IRET instruction can pop an EFLAG image having the RF bit set and resume the program's execution at the breakpoint address without generating another breakpoint fault on the same location.

NT (Nested Task, bit 14)

The flag applies to Protected Mode. NT is set to indicate that the execution of this task is within another task. If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS. This bit is set or reset by control transfers to other tasks. The value of NT in EFLAGS is tested by the IRET instruction to determine whether to do an inter-task return or an intra-task return. A POPF or an IRET instruction will affect the setting of this bit according to the image popped, at any privilege level.

IOPL (Input/Output Privilege Level, bits 12–13)

This two-bit field applies to Protected Mode. IOPL indicates the numerically maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O Permission Bitmap. It also indicates the maximum CPL value allowing alteration of the IF (INTR Enable Flag) bit when new values are popped into the EFLAG register. POPF and IRET instruction can alter the IOPL field when executed at CPL = 0. Task switches can always alter the IOPL field, when the new flag image is loaded from the incoming task's TSS.

OF (Overflow Flag, bit 11)

is set if the operation resulted in a signed overflow. Signed overflow occurs when the operation resulted in carry/borrow **into** the sign bit (high-order bit) of the result but did not result in a carry/borrow **out of** the high-order bit, or vice-versa. For 8-, 16-, 32-bit operations, OF is set according to overflow at bit 7, 15, 31, respectively.

DF (Direction Flag, bit 10)

DF defines whether ESI and/or EDI registers post decrement or post increment during the string instructions. Post increment occurs if DF is reset. Post decrement occurs if DF is set.

IF (INTR Enable Flag, bit 9)

IF flag, when set, allows recognition of external interrupts signaled on the INTR pin. When IF is reset, external interrupts signaled on the INTR are not recognized. IOPL indicates the maximum CPL value allowing alteration of the IF bit when new values are popped into EFLAGS or FLAGS.

TF (Trap Enable Flag, bit 8)

TF controls the generation of exception 1 trap when single-stepping through code. When TF is set, the Intel486 processor generates an exception 1 trap after the next instruction is executed. When TF is reset, exception 1 traps occur only as a function of the breakpoint addresses loaded into debug registers DR0–DR3.

SF (Sign Flag, bit 7)

SF is set if the high-order bit of the result is set, it is reset otherwise. For 8-, 16-, 32-bit operations, SF reflects the state of bit 7, 15, 31 respectively.

ZF (Zero Flag, bit 6)

ZF is set if all bits of the result are 0. Otherwise, it is reset.

AF (Auxiliary Carry Flag, bit 4)

The Auxiliary Flag is used to simplify the addition and subtraction of packed BCD quantities. AF is set if the operation resulted in a carry out of bit 3 (addition) or a borrow into bit 3 (subtraction). Otherwise, AF is reset. AF is affected by carry out of, or borrow into bit 3 only, regardless of overall operand length: 8, 16 or 32 bits.

PF (Parity Flags, bit 2)

PF is set if the low-order eight bits of the operation contains an even number of "1's" (even parity). PF is reset if the low-order eight bits have odd parity. PF is a function of only the low-order eight bits, regardless of operand size.

CF (Carry Flag, bit 0)

CF is set if the operation resulted in a carry out of (addition), or a borrow into (subtraction) the high-order bit. Otherwise, CF is reset. For 8-, 16- or 32-bit operations, CF is set according to carry/borrow at bit 7, 15 or 31, respectively.

4.2.2.4 Segment Registers

Six 16-bit segment registers hold segment selector values identifying the currently addressable memory segments. In protected mode, each segment may range in size from one byte up to the entire linear and physical address space of the machine, 4 Gbytes (2^{32} bytes). In real address mode, the maximum segment size is fixed at 64 Kbytes (2^{16} bytes).

The six addressable segments are defined by the segment registers CS, SS, DS, ES, FS and GS. The selector in CS indicates the current code segment; the selector in SS indicates the current stack segment; the selectors in DS, ES, FS and GS indicate the current data segments.

4.2.2.5 Segment Descriptor Cache Registers

The segment descriptor cache registers are not programmer visible, yet it is very useful to understand their content. A programmer invisible descriptor cache register is associated with each programmer-visible segment register, as shown by Figure 4-4. Each descriptor cache register holds a 32-bit base address, a 32-bit segment limit, and the other necessary segment attributes.

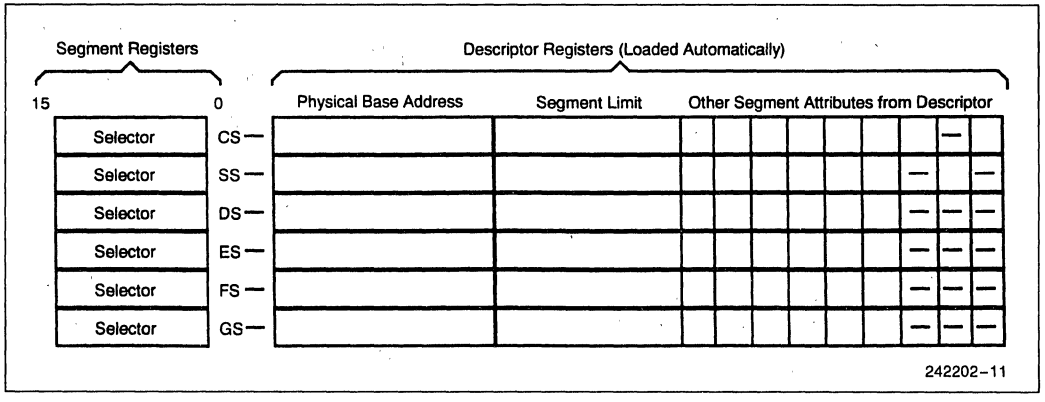


Figure 4-4. Intel486™ Processor Segment Registers and Associated Descriptor Cache Registers

When a selector value is loaded into a segment register, the associated descriptor cache register is automatically updated with the correct information. In Real Mode, only the base address is updated directly (by shifting the selector value four bits to the left), because the segment maximum limit and attributes are fixed in Real Mode. In Protected Mode, the base address, the limit, and the attributes are all updated per the contents of the segment descriptor indexed by the selector.

Whenever a memory reference occurs, the segment descriptor cache register associated with the segment being used is automatically involved with the memory reference. The 32-bit segment base address becomes a component of the linear address calculation, the 32-bit limit is used for the limit-check operation, and the attributes are checked against the type of memory reference requested.

4.2.3 SYSTEM LEVEL REGISTERS

Figure 4-5 illustrates the system level registers, which are the control operation of the on-chip cache, the on-chip Floating-Point Unit (on the Intel486 DX, IntelDX2, and IntelDX4 processors) and the segmentation and paging mechanisms. These registers are only accessible to programs running at privilege level 0, the highest privilege level.

The system level registers include three control registers and four segmentation base registers. The three control registers are CR0, CR2 and CR3. CR1 is reserved for future Intel processors. The four segmentation base registers are the Global Descriptor Table Register (GDTR), the Interrupt Descriptor Table Register (IDTR), the Local Descriptor Table Register (LDTR) and the Task State Segment Register (TR).

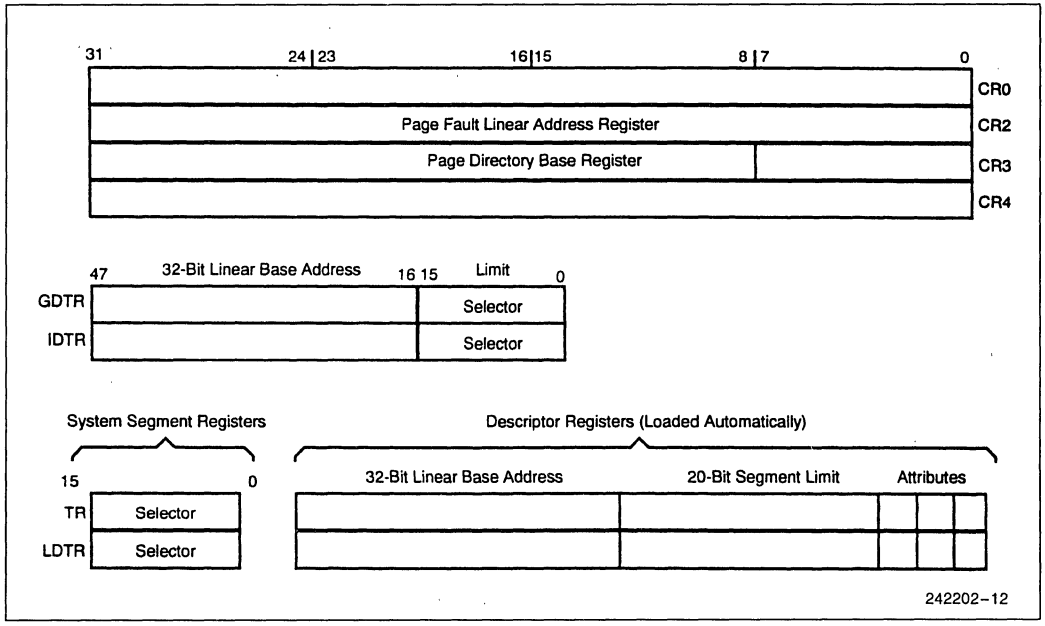
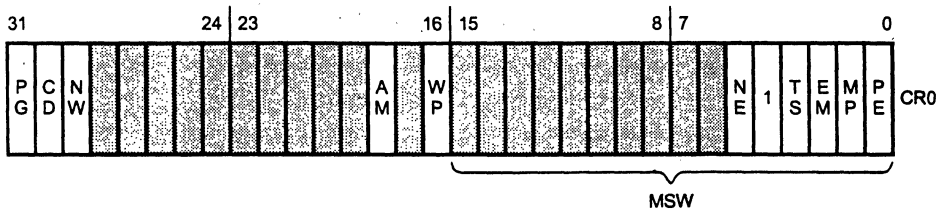


Figure 4-5. System Level Registers



NOTE:
See Section 4.2.7 "Compatibility".

Figure 4-6. Control Register 0

4.2.3.1 Control Registers

Control Register 0 (CR0)

CR0, shown in Figure 4-6, contains 10 bits for control and status purposes. The function of the bits in CR0 can be categorized as follows:

- Intel486 Processor Operating Modes: PG, PE (Table 4-2)
- On-Chip Cache Control Modes: CD, NW (Table 4-3)
- On-Chip Floating-Point Unit: NE, TS, EM, TS (Tables 4-4, 4-5, and 4-6). (Also applies for Intel486 SX and IntelSX2 processors.)
- Alignment Check Control: AM
- Supervisor Write Protect: WP

Table 4-2. Intel486™ Processor Operating Modes

PG	PE	Mode
0	0	REAL Mode. Exact 8086 processor semantics, with 32-bit extensions available with prefixes.
0	1	Protected Mode. Exact 80286 processor semantics, plus 32-bit extensions through both prefixes and "default" prefix setting associated with code segment descriptors. Also, a sub-mode is defined to support a virtual 8086 processor within the context of the extended 80286 processor protection model.
1	0	UNDEFINED. Loading CR0 with this combination of PG and PE bits will raise a GP fault with error code 0.
1	1	Paged Protected Mode. All the facilities of Protected mode, with paging enabled underneath segmentation.

Table 4-3. On-Chip Cache Control Modes

CD	NW	Operating Mode
1	1	Cache fills disabled, write-through and invalidates disabled.
1	0	Cache fills disabled, write-through and invalidates enabled.
0	1	INVALID. If CR0 is loaded with this configuration of bits, a GP fault with error code is raised.
0	0	Cache fills enabled, write-through and invalidates enabled.

The low-order 16 bits of CR0 are also known as the Machine Status Word (MSW), for compatibility with the 80286 processor protected mode. LMSW and SMSW (load and store MSW) instructions are taken as special aliases of the load and store CR0 operations, where only the low-order 16 bits of CR0 are involved. The LMSW and SMSW instructions in the Intel486 processor work in an identical fashion to the LMSW and SMSW instructions in the 80286 processor (i.e., they only operate on the low-order 16 bits of CR0 and ignore the new bits). New Intel486 processor operating systems should use the MOV CR0, Reg instruction.

NOTE:
All Intel386 and Intel486 processor CR0 bits, except for ET and NE, are upwardly compatible with the 80286 processor, because they are in register bits not defined in the 80286 processor. For strict compatibility with the 80286 processor, the load machine status word (LMSW) instruction is defined to not change the ET or NE bits.

The defined CR0 bits are described below.

PG (Paging Enable, bit 31)

PG bit is used to indicate whether paging is enabled (PG=1) or disabled (PG=0). (See Table 4-2.)

CD (Cache Disable, bit 30)

The CD bit is used to enable the on-chip cache. When CD=1, the cache will not be filled on cache misses. When CD=0, cache fills may be performed on misses. (See Table 4-3.)

The state of the CD bit, the cache enable input pin (KEN#), and the relevant page cache disable (PCD) bit determine if a line read in response to a cache miss will be installed in the cache. A line is installed in the cache only if CD=0 and KEN# and PCD are both zero. The relevant PCD bit comes from either the page table entry, page directory entry or control register 3. (Refer to section 7.6, "Page Cacheability.")

CD is set to one after RESET.

NW (Not Write-Through, bit 29)

The NW bit enables on-chip cache write-throughs and write-invalidate cycles (NW=0).

When NW=0, all writes, including cache hits, are sent out to the pins. Invalidate cycles are enabled when NW=0. During an invalidate cycle, a line will be removed from the cache if the invalidate address hits in the cache. (See Table 4-3.)

When NW=1, write-throughs and write-invalidate cycles are disabled. A write will not be sent to the pins if the write hits in the cache. With NW=1 the only write cycles that reach the external bus are cache misses. Write hits with NW=1 will never update main memory. Invalidate cycles are ignored when NW=1.

AM (Alignment Mask, bit 18)

The AM bit controls whether the alignment check (AC) bit in the flag register (EFLAGS) can allow an alignment fault. AM=0 disables the AC bit. AM=1 enables the AC bit. AM=0 is the Intel386 processor compatible mode.

Intel386 processor software may load incorrect data into the AC bit in the EFLAGS register. Setting AM=0 will prevent AC faults from occurring before the Intel486 processor has created the AC interrupt service routine.

WP (Write Protect, bit 16)

WP protects read-only pages from supervisor write access. The Intel386 processor allows a read-only page to be written from privilege levels 0–2. The Intel486 processor is compatible with the Intel386 processor when WP=0. WP=1 forces a fault on a write to a read-only page from any privilege level. Operating systems with Copy-on-Write features can be supported with the WP bit. (Refer to section 6.4.3 "Page Level Protection (R/W, U/S Bits).")

NOTE:
Refer to Tables 4-4, 4-5, and 4-6 for values and interpolation of NE, EM, TS, and MP bits, in addition to the sections below.

NE (Numerics Exception, bit 5)

Intel486 SX and IntelSX2 Processor NE Bit

For Intel486 SX and IntelSX2 processors, interrupt 7 will be generated upon encountering any Floating-Point instruction regardless of the value of the NE bit. It is recommended that NE=1 for normal operation of the Intel486 processor.

Intel486 DX, IntelDX2 and IntelDX4 Processor NE Bit

For Intel486 DX, IntelDX2, and IntelDX4 processors, the NE bit controls whether unmasked Floating-Point exceptions (UFPE) are handled through interrupt vector 16 (NE=1) or through an external interrupt (NE=0). NE=0 (default at reset) supports the DOS operating system error reporting scheme from the 8087, Intel287 and Intel387 math coprocessors. In DOS systems, math coprocessor errors are reported via external interrupt vector 13. DOS uses interrupt vector 16 for an operating system call. (Refer to sections 9.2.15, "Numeric Error Reporting (FERR#, IGNNE#)," and 10.2.14, "Floating-Point Error Handling.")

For any UFPE, the Floating-Point error output pin (FERR#) will be driven active.

For NE=0, the Intel486 DX, IntelDX2 and IntelDX4 processors work in conjunction with the ignore numeric error input (IGNNE#) and the FERR# output pins. When a UFPE occurs and the IGNNE# input is inactive, the Intel486 DX, IntelDX2, and IntelDX4 processors freeze immediately before executing the next Floating-Point instruction. An external interrupt controller will supply an interrupt vector when FERR# is driven active. The UFPE is ignored if IGNNE# is active and Floating-Point execution continues.

NOTE:
The freeze does not take place if the next instruction is one of the control instructions FNCLEX, FNINIT, FNSAVE, FNSTENV, FNSTCW, FNSTSW, FNSTSW AX, FNENI, FNDISI and FNSETPM. The freeze does occur if the next instruction is WAIT.

For NE=1, any UFPE will result in a software interrupt 16, immediately before executing the next non-control Floating-Point or WAIT instruction. The ignore numeric error input (IGNNE#) signal will be ignored.

TS (Task Switch, bit 3)

Intel486 SX and IntelSX2 Processor TS Bit

For Intel486 SX and IntelSX2 processors, the TS bit is set whenever a task switch operation is performed. Execution of Floating-Point instructions with TS=1 will cause a Device Not Available (DNA) fault (trap vector 7). With MP=0, the value of TS bit is a "don't care" for the WAIT instructions, i.e., these instructions will not generate trap 7.

Intel486 DX, IntelDX2, and IntelDX4 Processor TS Bit

For Intel486 DX, IntelDX2, and IntelDX4 processors, the TS bit is set whenever a task switch operation is performed. Execution of Floating-Point instructions with TS=1 will cause a Device Not Available (DNA) fault (trap vector 7). If TS=1 and MP=1 (monitor coprocessor in CR0), a WAIT instruction will cause a DNA fault.

EM (Emulate Coprocessor, bit 2)

Intel486 SX and IntelSX2 Processor EM Bit

For Intel486 SX and IntelSX2 processors, the EM bit should be set to one. This will cause the Intel486 SX and IntelSX2 processors to trap via interrupt vector 7 (Device Not Available) to a software exception handler whenever it encounters a Floating-Point instruction. If EM bit is 0 for the Intel486 SX and IntelSX2 processors, the system will hang. (See Tables 4-4 and 4-5.)

Intel486 DX, IntelDX2, and IntelDX4 Processor EM Bit

For the Intel486 DX, IntelDX2, and IntelDX4 processors, the EM bit determines whether Floating-Point instructions are trapped (EM=1) or executed. If EM=1, all Floating-Point instructions will cause fault 7.

If EM=0, the on-chip Floating-Point will be used.

NOTE:
WAIT instructions are not affected by the state of EM. (See Tables 4-4 and 4-6.)

MP (Monitor Coprocessor, bit 1)

Intel486 SX and IntelSX2 Processor MP Bit

For Intel486 SX and IntelSX2 processors, the MP bit must be set to zero (MP=0). The MP bit is used in conjunction with the TS bit to determine if WAIT instructions should trap. For MP=0, the value of TS is a "don't care" for these type of instructions. (See Tables 4-4 and 4-5.)

Intel486 DX, IntelDX2, and IntelDX4 Processor MP Bit

For the Intel486 DX, IntelDX2, and IntelDX4 processors, the MP is used in conjunction with the TS bit to determine if WAIT instructions cause fault 7. (See Table 4-6.) The TS bit is set to 1 on task switches by the Intel486 DX, IntelDX2, and IntelDX4 processors. Floating-point instructions are not affected by the state of the MP bit. It is recommended that the MP bit be set to one for normal processor operation.

PE (Protection Enable, bit 0)

The PE bit enables the segment based protection mechanism if PE = 1 protection is enabled. When PE = 0 the Intel486 processor operates in REAL mode, with segment based protection disabled, and addresses formed as in an 8086 processor. (Refer to Table 4-2.)

Table 4-4. Recommended Values of NE, EM, TS, and MP Bits in CR0 Register for Intel486™ SX and IntelSX2™ Processors

CR0 Bit				Instruction Type	
NE	EM	TS	MP	FP	WAIT
1	1	0	0	Trap7	Execute
1	1	1	0	Trap7	Execute

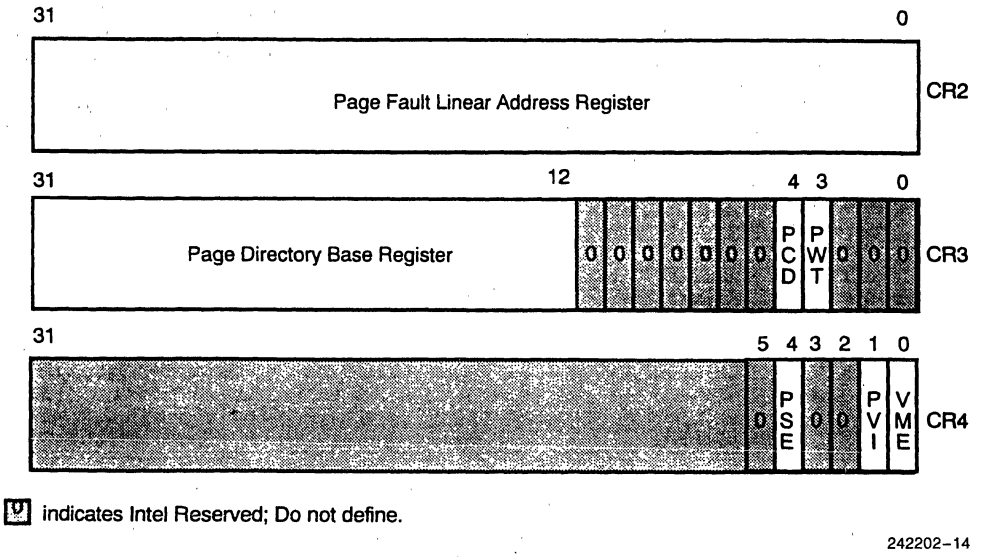
Table 4-5. Recommended Values of the Floating-Point Related Bits for All Intel486™ Processors

CR0 Bit	Intel486™ SX and IntelSX2™ Processors	Intel486 DX, IntelDX2™, and IntelDX4™ Processors
EM	1	0
MP	0	1
NE	1	0, for DOS Systems 1, for User-Defined Exception Handler

Table 4-6. Interpretation of Different Combinations of the EM, TS and MP Bits for All Intel486™ Processors

CR0 Bit			Instruction Type	
EM	TS	MP	Floating-Point	Wait
0	0	0	Execute	Execute
0	0	1	Execute	Execute
0	1	0	Exception 7	Execute
0	1	1	Exception 7	Exception 7
1	0	0	Exception 7	Execute
1	0	1	Exception 7	Execute
1	1	0	Exception 7	Execute
1	1	1	Exception 7	Exception 7

NOTE:
For Intel486™ DX, IntelDX2™ and IntelDX4™ processors, if MP = 1 and TS = 1, the processor will generate a trap 7 so that the system software can save the Floating-Point status of the old task.



NOTE:
See section 4.2.7, "Compatibility."

Figure 4-7. Control Registers 2, 3 and 4

Control Register 1 (CR1)

CR1 is reserved for use in future Intel processors.

Control Register 2 (CR2)

CR2, shown in Figure 4-7, holds the 32-bit linear address that caused the last page fault detected. The error code pushed onto the page fault handler's stack when it is invoked provides additional status information on this page fault.

Control Register 3 (CR3)

CR3, shown in Figure 4-7, contains the physical base address of the page directory table. The page directory is always page aligned (4 Kbyte-aligned). This alignment is enforced by only storing bits 12–31 in CR3.

In the Intel486 processor, CR3 contains two bits, page write-through (PWT) (bit 3) and page cache disable (PCD) (bit 4). The page table entry (PTE) and page directory entry (PDE) also contain PWT and PCD bits. PWT and PCD control page cacheability. When a page is accessed in external memory, the state of PWT and PCD are driven out on the PWT and PCD pins. The source of PWT and PCD can be CR3, the PTE or the PDE. PWT and PCD are sourced from CR3 when the PDE is being updated. When paging is disabled (PG = 0 in CR0), PCD and PWT are assumed to be 0, regardless of their state in CR3.

A task switch through a task state segment (TSS) which changes the values in CR3, or an explicit load into CR3 with any value, will invalidate all cached page table entries in the translation lookaside buffer (TLB).

The page directory base address in CR3 is a physical address. The page directory can be paged out while its associated task is suspended, but the operating system must ensure that the page directory is resident in physical memory before the task is dispatched. The entry in the TSS for CR3 has a physical address, with no provision for a present bit. This means that the page directory for a task must be resident in physical memory. The CR3 image in a TSS must point to this area, before the task can be dispatched through its TSS.

Control Register 4 (CR4)

CR4, shown in Figure 4-7, contains bits that enable virtual mode extensions and protected mode virtual interrupts.

VME (Virtual-8086 Mode Extensions, bit 0 of CR4)

Setting this bit to 1 enables support for a virtual interrupt flag in virtual-8086 mode. This feature can improve the performance of virtual-8086 applications by eliminating the overhead of faulting to a virtual-8086 monitor for emulation of certain operations. (Refer to Appendix A, "Advanced Features.")

PVI (Protected-Mode Virtual Interrupts, bit 1 of CR4)

Setting this bit to 1 enables support for a virtual interrupt flag in protected mode. This feature can enable some programs designed for execution at privilege level 0 to execute at privilege level 3. (Refer to Appendix A, "Advanced Features.")

PSE (Page Size Extensions, bit 4 of CR4)

Setting this bit to 1 enables 4-Mbyte pages. (Refer to Appendix A, "Advanced Features.")

NOTE:
Features described in CR4 (VME, PVI, and PSE) in the CPUID Feature Flag should be qualified with the CPUID instruction. The CPUID instruction and CPUID Feature Flag are specific to particular models in the Intel486 processor family. (Refer to Appendix B, "Feature Determination.")

4.2.3.2 System Address Registers

Four special registers are defined to reference the tables or segments supported by the 80286, Intel386, and Intel486 processors' protection model. These tables or segments are: GDT (Global Descriptor Table), IDT (Interrupt Descriptor Table), LDT (Local Descriptor Table), TSS (Task State Segment).

The addresses of these tables and segments are stored in special registers, the System Address and System Segment Registers, illustrated in Figure 4-5. These registers are named GDTR, IDTR, LDTR and TR respectively. Section 6, "Protected Mode Architecture," describes how to use these registers.

System Address Registers: GDTR and IDTR

The GDTR and IDTR hold the 32-bit linear-base address and 16-bit limit of the GDT and IDT, respectively.

Because the GDT and IDT segments are global to all tasks in the system, the GDT and IDT are defined by 32-bit linear addresses (subject to page translation if paging is enabled) and 16-bit limit values.

System Segment Registers: LDTR and TR

The LDTR and TR hold the 16-bit selector for the LDT descriptor and the TSS descriptor, respectively.

Because the LDT and TSS segments are task specific segments, the LDT and TSS are defined by selector values stored in the system segment registers.

NOTE:
A programmer-invisible segment descriptor register is associated with each system segment register.

4.2.4 FLOATING-POINT REGISTERS

Figure 4-8 shows the Floating-Point register set. The on-chip FPU contains eight data registers, a tag word, a control register, a status register, an instruction pointer and a data pointer.

The operation of the Intel486 DX, IntelDX2, and IntelDX4 processor on-chip Floating-Point Unit is exactly the same as the Intel387 math coprocessor. Software written for the Intel387 math coprocessor will run on the on-chip Floating-Point Unit (FPU) without any modifications.

4.2.4.1 Floating-Point Data Registers

Floating-point computations use the Intel486 DX, IntelDX2, and IntelDX4 processor FPU data registers. These eight 80-bit registers provide the equivalent capacity of twenty 32-bit registers. Each of the eight data registers is divided into "fields" corresponding to the FPU's extended-precision data type.

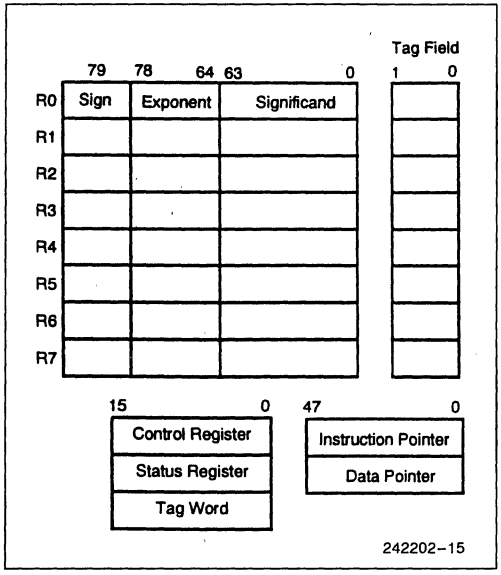


Figure 4-8. Floating-Point Registers

The FPU's register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as a fixed register set, with instructions operating on explicitly designated registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments TOP by one. Like other Intel486 DX, IntelDX2, and IntelDX4 processor stacks in memory, the FPU register stack grows "down" toward lower-addressed registers.

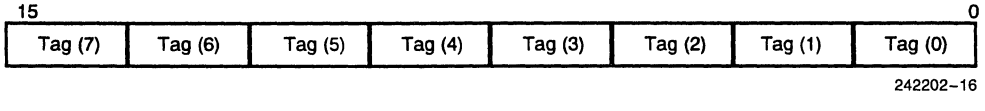
Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to use. This explicit register addressing is also relative to TOP.

4.2.4.2 Floating-Point Tag Word

The tag word marks the content of each numeric data register, as shown in Figure 4-9. Each two-bit tag represents one of the eight data registers. The principal function of the tag word is to optimize the FPU's performance and stack handling by making it possible to distinguish between empty and non-empty register locations. It also enables exception handlers to check the contents of a stack location without the need to perform complex decoding of the actual data.

4.2.4.3 Floating-Point Status Word

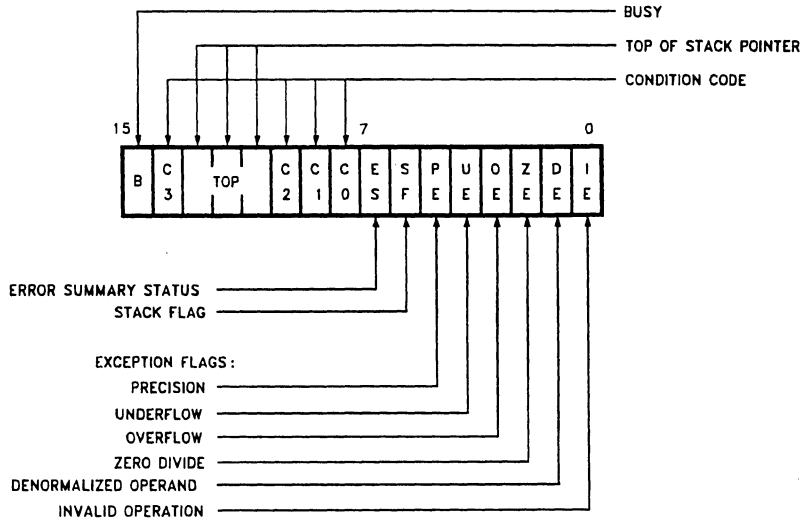
The 16-bit status word reflects the overall state of the FPU. The status word is shown in Figure 4-10 and is located in the status register.



NOTE:
The index i of tag(i) is not top-relative. A program typically uses the "top" field of Status Word to determine which tag(i) field refers to logical top of stack.

TAG VALUES:
00 = Valid
01 = Zero
10 = QNaN, SNaN, Infinity, Denormal and Unsupported Formats
11 = Empty

Figure 4-9. Floating-Point Tag Word



242202-17

ES is set if any unmasked exception bit is set; cleared otherwise.
See Table 4-7 for interpretation of condition code.
TOP values:

000 = Register 0 is Top of Stack
001 = Register 1 is Top of Stack
•
•
•
111 = Register 7 is Top of Stack

For definitions of exceptions, refer to the section entitled "Exception Handling".

NOTES:
The B bit (Busy, bit 15) is included for 8087 compatibility. The B bit reflects the contents of the ES bit (bit 7 of the status word).
Bits 13–11 (TOP) point to the FPU register that is the current top-of-stack.
The four numeric condition code bits, C0–C3, are similar to the flags in EFLAGS. Instructions that perform arithmetic operations update C0–C3 to reflect the outcome. The effects of these instructions on the condition codes are summarized in Table 4-7 through Table 4-10.

Figure 4-10. Floating-Point Status Word

Table 4-7. Floating-Point Condition Code Interpretation

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1	Three least significant bits of quotient (See Table 4-8.)			Reduction 0 = complete 1 = incomplete
	Q2	Q0	Q1 or O/U#	
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 4-9)		Zero or O/U#	Operand is not comparable
FXAM	Operand class (see Table 4-10)		Sign or O/U#	Operand class
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or O/U#	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U#	UNDEFINED
FPTAN, FSIN, FCOS, FSINCOS	UNDEFINED		Roundup or O/U#, if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FINIT	Clears these bits			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FSAVE	UNDEFINED			

NOTES:

O/U# When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).

Reduction If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.

Roundup When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.

UNDEFINED Do not rely on finding any specific value in these bits. (See Section 4.2.7, "Compatibility.")

Table 4-8. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interaction required for complete reduction	
	Q1	Q0	Q2	Q MOD8	Complete Reduction: C0, C3, and C1 contain the three least-significant bits of the quotient
0	0	0	0	0	
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 4-9. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

4

Table 4-10. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

Bit 7 is the error summary (ES) status bit. The ES bit is set if any unmasked exception bit (bits 0–5 in the status word) is set; ES is clear otherwise. The FERR# (Floating-Point error) signal is asserted when ES is set.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow. When SF is set, bit 9 (C1) distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).

Table 4-11 shows the six exception flags in bits 0–5 of the status word. Bits 0–5 are set to indicate that the FPU has detected an exception while executing an instruction.

The six exception flags in the status word can be individually masked by mask bits in the FPU control word. Table 4-11 lists the exception conditions, and their causes in order of precedence. Table 4-11 also shows the action taken by the FPU if the corresponding exception flag is masked.

An exception that is not masked by the control word will cause three things to happen: the corresponding exception flag in the status word will be set, the ES bit in the status word will be set and the FERR# output signal will be asserted. When the Intel486 DX, IntelDX2, or IntelDX4 processor attempts to execute another Floating-Point or WAIT instruction, exception 16 occurs or an external interrupt happens if the NE = 1 in control register 0. The exception condition must be resolved via an interrupt service routine. The FPU saves the address of the Floating-Point instruction that caused the exception and the address of any memory operand required by that instruction in the instruction and data pointers. (See section 4.2.4.4, "Instruction and Data Pointers.")

Note that when a new value is loaded into the status word by the FLDENV (load environment) or FRSTOR (restore state) instruction, the value of ES (bit 7) and its reflection in the B bit (bit 15) are not derived from the values loaded from memory. The values of ES and B are dependent upon the values of the exception flags in the status word and their corresponding masks in the control word. If ES is set in such a case, the FERR# output of the Intel486 DX, IntelDX2, or IntelDX4 processor is activated immediately.

Table 4-11. FPU Exceptions

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signaling NaN, unsupported format, indeterminate form (0 * ∞, 0/0, (+∞) + (−∞), etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinite
Denormalized Operand	At least one of the operands is denormalized, i.e., it has the smallest exponent but a non-zero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a non-infinite, non-zero number.	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞
Underflow	The true result is non-zero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g., 1/3); the result is rounded according to the rounding mode.	Normal processing continues

4.2.4.4 Instruction and Data Pointers

Because the FPU operates in parallel with the ALU (in the Intel486 DX, IntelDX2 and IntelDX4 processors the arithmetic and logic unit (ALU) consists of the base architecture registers), any errors detected by the FPU may be reported after the ALU has executed the Floating-Point instruction that caused it. To allow identification of the failing numeric instruction, the Intel486 DX, IntelDX2, and IntelDX4 processors contain two pointer registers that supply the address of the failing numeric instruction and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written error handlers. These registers are accessed by the FLDENV (load environment), FSTENV (store environment), FSAVE (save state) and FRSTOR (restore state) instructions. Whenever the Intel486 DX, IntelDX2, and IntelDX4 processors decode a new Floating-Point instruction, it saves the instruction (including any prefixes that may be present), the address of the operand (if present) and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the Intel486 DX, IntelDX2, and IntelDX4 processors (protected mode or real-address mode) and depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). When the Intel486 DX, IntelDX2, or IntelDX4 processor is in the virtual-86 mode, the real address mode formats are used. The four formats are shown in Figure 4-11 through Figure 4-14. The Floating-Point instructions FLDENV, FSTENV, FSAVE and FRSTOR are used to transfer these values to and from memory. Note that the value of the data pointer is undefined if the prior Floating-Point instruction did not have a memory operand.

NOTE:
The operand size attribute is the D bit in a segment descriptor.

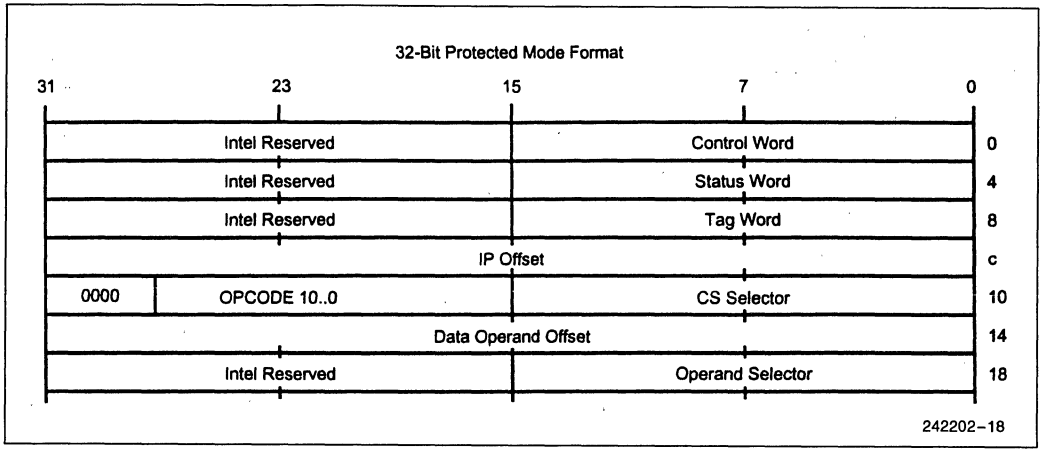


Figure 4-11. Protected Mode FPU Instructions and Data Pointer Image in Memory, 32-Bit Format

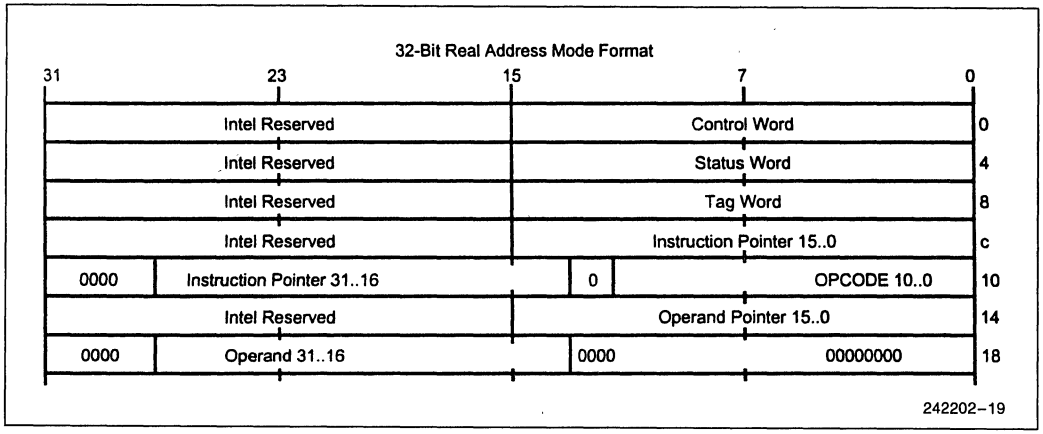


Figure 4-12. Real Mode FPU Instruction and Data Pointer Image in Memory, 32-Bit Format

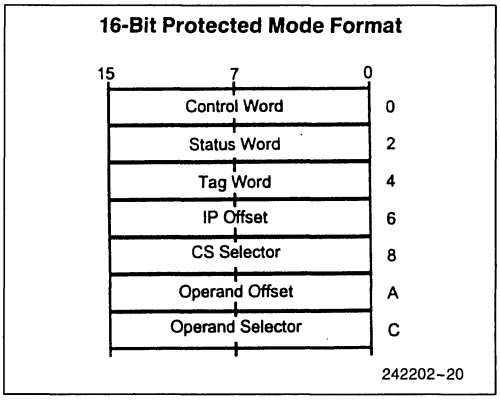


Figure 4-13. Protected Mode FPU Instruction and Data Pointer Image in Memory, 16-Bit Format

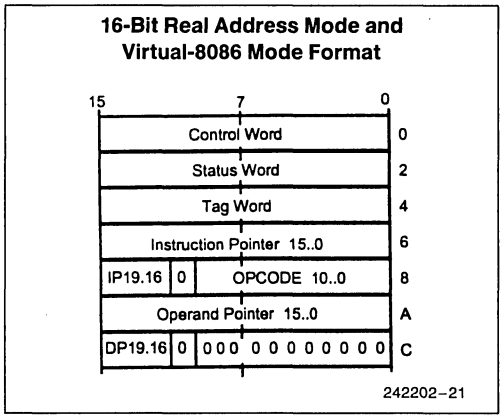
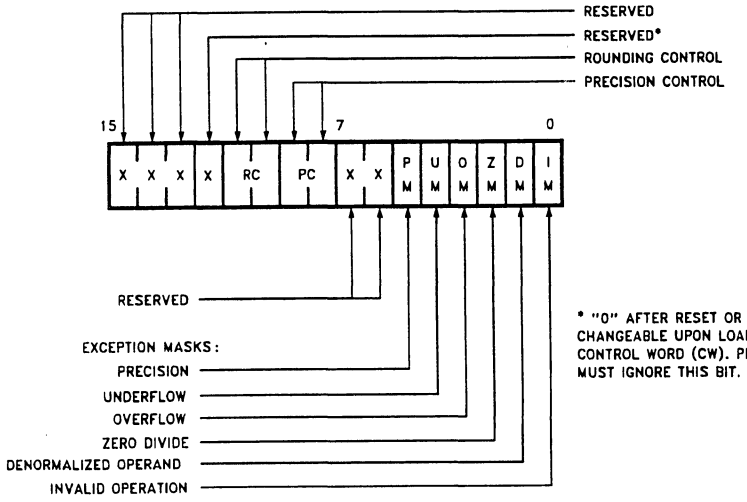


Figure 4-14. Real Mode FPU Instruction and Data Pointer Image in Memory, 16-Bit Format



Precision Control	**Rounding Control**
00-24 bits (single precision)	00-Round to nearest or even
01-(reserved)	01-Round down (toward -∞)
10-53 bits (double precision)	10-Round up (toward + ∞)
11-64 bits (extended precision)	11-Chop (truncate toward zero)

NOTE:
See section 4.2.7 "Compatibility," for RESERVED bits.

Figure 4-15. FPU Control Word

4.2.4.5 FPU Control Word

The FPU provides several processing options that are selected by loading a control word from memory into the control register. Figure 4-15 shows the format and encoding of fields in the control word.

The low-order byte of the FPU control word configures the FPU error and exception masking. Bits 0–5 of the control word contain individual masks for each of the six exceptions that the FPU recognizes.

The high-order byte of the control word configures the FPU operating mode, including precision and rounding.

RC (Rounding Control, bits 10–11)

RC bits provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS and FCHS), and all transcendental instructions.

PC (Precision Control, bits 8–9)

PC bits can be used to set the FPU internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

4.2.5 DEBUG AND TEST REGISTERS

4.2.5.1 Debug Registers

The six programmer accessible debug registers (Figure 4-16) provide on-chip support for debugging. Debug registers DR0–3 specify the four linear breakpoints. The Debug control register DR7, is used to set the breakpoints and the Debug Status Register, DR6, displays the current state of the breakpoints. The use of the Debug registers is described in section 12, "Debugging Support."

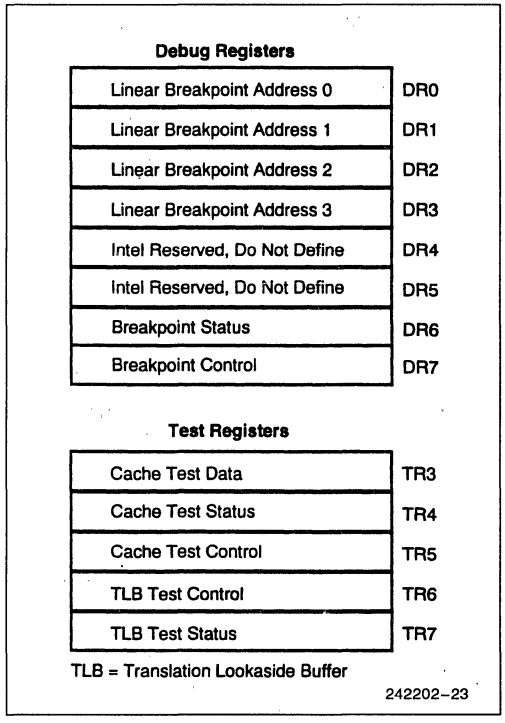


Figure 4-16. Debug and Test Registers

4.2.5.2 Test Registers

The Intel486 processor contains five test registers. The test registers are shown in Figure 4-16. TR6 and TR7 are used to control the testing of the translation look- aside buffer. TR3, TR4 and TR5 are used for testing the on-chip cache. The use of the test registers is discussed in section 11, "Testability."

4.2.6 REGISTER ACCESSIBILITY

There are a few differences regarding the accessibility of the registers in Real and Protected Mode. Table 4-12 summarizes these differences. (See section 6, "Protected Mode Architecture.")

4.2.6.1 FPU Register Usage

In addition to the differences listed in Table 4-12, Table 4-13 summarizes the differences for the on-chip FPU.

Table 4-12. Register Usage

Register	Use in Real Mode		Use in Protected Mode		Use in Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
General Registers	Yes	Yes	Yes	Yes	Yes	Yes
Segment Register	Yes	Yes	Yes	Yes	Yes	Yes
Flag Register	Yes	Yes	Yes	Yes	IOPL(1)	IOPL
Control Registers	Yes	Yes	PL = 0(2)	PL = 0	No	Yes
GDTR	Yes	Yes	PL = 0	Yes	No	Yes
IDTR	Yes	Yes	PL = 0	Yes	No	Yes
LDTR	No	No	PL = 0	Yes	No	No
TR	No	No	PL = 0	Yes	No	No
Debug Registers	Yes	Yes	PL = 0	PL = 0	No	No
Test Registers	Yes	Yes	PL = 0	PL = 0	No	No

NOTES:
1. IOPL: The PUSHF and POPF instructions are made I/O Privilege Level sensitive in Virtual 8086 Mode.
2. PL = 0: The registers can be accessed only when the current privilege level is zero.

Table 4-13. FPU Register Usage Differences

Register	Use in Real Mode		Use in Protected Mode		Use in Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
FPU Data Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Control Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Status Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Instruction Pointer	Yes	Yes	Yes	Yes	Yes	Yes
FPU Data Pointer	Yes	Yes	Yes	Yes	Yes	Yes

4.2.7 COMPATIBILITY

VERY IMPORTANT NOTE:
COMPATIBILITY WITH FUTURE PROCESSORS

In the preceding register descriptions, note certain Intel486™ processor register bits are Intel reserved. When reserved bits are called out, treat them as fully undefined. This is essential for your software compatibility with future processors! Follow the guidelines below:

1. **Do not depend on the states of any undefined bits when testing the values of defined register bits. Mask them out when testing.**
2. **Do not depend on the states of any undefined bits when storing them to memory or another register.**
3. **Do not depend on the ability to retain information written into any undefined bits.**
4. **When loading registers, always load the undefined bits as zeros.**
5. **However, registers that have been previously stored may be reloaded without masking.**

Depending upon the values of undefined register bits will make your software dependent upon the unspecified Intel486 processor handling of these bits. Depending on undefined values risks making your software incompatible with future processors that define usages for the Intel486 processor-undefined bits. AVOID ANY SOFTWARE DEPENDENCE UPON THE STATE OF UNDEFINED INTEL486 PROCESSOR REGISTER BITS.

4.3 Instruction Set

The Intel486 processor instruction set can be divided into the following categories of operations:

- Data Transfer
- Arithmetic
- Shift/Rotate
- String Manipulation
- Bit Manipulation
- Control Transfer
- High Level Language Support
- Operating System Support
- Processor Control

The Intel486 processor instructions are listed in section 13, "Instruction Set Summary."

All Intel486 processor instructions operate on either 0, 1, 2 or 3 operands; where an operand resides in a register, in the instruction itself or in memory. Most zero operand instructions (e.g., CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2-bytes long. Because the Intel486 processor has a 32-byte instruction queue, an average of 10 instructions will be prefetched. The use of two operands permits the following types of common instructions:

- Register to Register
- Memory to Register
- Memory to Memory
- Immediate to Register
- Register to Memory
- Immediate to Memory

The operands can be either 8-, 16-, or 32-bits long. As a general rule, when executing 32-bit code, operands are 8 or 32 bits; when executing existing 80286 or 8086 processor code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to all instructions that override the default length of the operands (i.e., use 32-bit operands for 16-bit code, or 16-bit operands for 32-bit code).

4.3.1 FLOATING-POINT INSTRUCTIONS

In addition to the instructions listed above, the Intel486, IntelDX2, and IntelDX4 processors have the following Floating-Point instructions. Note that all Floating-Point unit instruction mnemonics begin with an F.

- Floating-Point
- Floating-Point Control

4.4 Memory Organization

Memory on the Intel486 processor is divided up into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address, the high order byte at the high address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address, the high-order byte at the highest address. The address of a word or dword is the byte address of the low-order byte.

In addition to these basic data types, the Intel486 processor supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable-length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4-Kbyte pages. Both segmentation and paging can be combined, gaining the advantages of both systems. The Intel486 processor supports both pages and segments in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and as such is a tool for the application programmer, while pages are useful for the system programmer for managing the physical memory of a system.

4.4.1 ADDRESS SPACES

The Intel486 processor has three distinct address spaces: **logical, linear,** and **physical**. A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, DISPLACEMENT) discussed in section 4.6.3 "32-Bit Memory Addressing Modes," into an effective address. Because each task on the Intel486 processor has a maximum of 16K (2^{14} -1) selectors, and offsets can be 4 Gbytes (2^{32} bits), this gives a total of 2^{46} bits or 64 terabytes of **logical** address space per task. The programmer sees this virtual address space.

The segmentation unit translates the **logical** address space into a 32-bit **linear** address space. If the paging unit is not enabled then the 32-bit **linear** address corresponds to the **physical** address. The paging unit translates the **linear** address space into the **physical** address space. The **physical address** is what appears on the address pins.

The primary difference between Real Mode and Protected Mode is how the segmentation unit performs the translation of the **logical** address into the **linear** address. In Real Mode, the segmentation unit shifts the selector left four bits and adds the result to the offset to form the **linear** address. While in Protected Mode every selector has a **linear** base address associated with it. The **linear base** address is stored in one of two operating system tables (i.e., the Local Descriptor Table or Global Descriptor Table). The selector's **linear base** address is added to the offset to form the final **linear** address.

Figure 4-17 shows the relationship between the various address spaces.

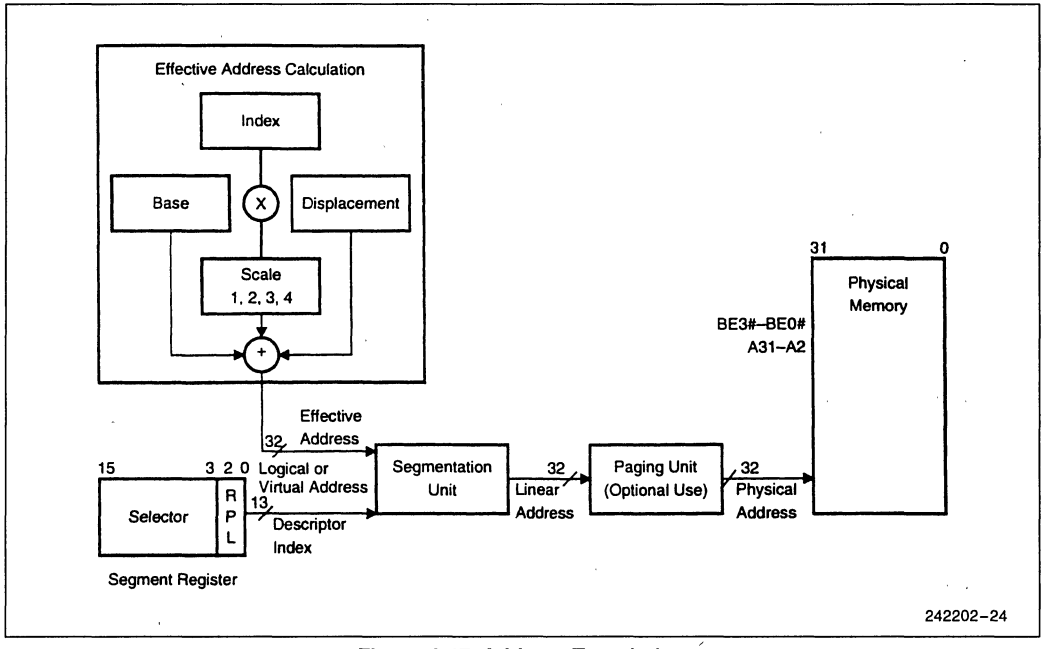


Figure 4-17. Address Translation

4.4.2 SEGMENT REGISTER USAGE

The main data structure used to organize memory is the segment. On the Intel486 processor, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments: code and data. The segments are of variable size and can be as small as 1 byte or as large as 4 Gbytes (2^{32} bytes).

In order to provide compact instruction encoding, and increase Intel486 processor performance, instructions do not need to explicitly specify which segment register is used. A default segment register is automatically chosen according to the rules of Table 4-14. In general, data references use the selector contained in the DS register; Stack references use the SS register and Instruction fetches use the CS register. The contents of the Instruction Pointer provide the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 4-14. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero and create a system with a 4-Gbyte linear address space. This creates a system where the virtual address space is the same as the linear address space. Further details of segmentation are discussed in section 6.0, "Protected Mode Architecture."

Table 4-14. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References, with Effective Address using Base Register of:		
[EAX]	DS	All
[EBX]	DS	
[ECX]	DS	
[EDX]	DS	
[ESI]	DS	
[EDI]	DS	
[EBP]	SS	
[ESP]	SS	

4.5 I/O Space

The Intel486 processor has two distinct physical address spaces: Memory and I/O. Generally, peripherals are placed in I/O space although the Intel486 processor also supports memory-mapped peripherals. The I/O space consists of 64 Kbytes, it can be divided into 64K 8-bit ports, 32K 16-bit ports, or 16K 32-bit ports, or any combination of ports which add up to less than 64 Kbytes. The 64K I/O address space refers to physical memory rather than linear address, because I/O instructions do not go through the segmentation or paging hardware. The M/IO# pin acts as an additional address line thus allowing the system designer to easily determine which address space the processor is accessing.

The I/O ports are accessed via the IN and OUT I/O instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8- and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/IO# pin to be driven low.

I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

I/O instruction code is cacheable.

I/O data is not cacheable.

I/O transfers (data or code) can be bursted.

4.6 Addressing Modes

4.6.1 ADDRESSING MODES OVERVIEW

The Intel486 processor provides a total of 11 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high-level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

4.6.2 REGISTER AND IMMEDIATE MODES

The following two addressing modes provide for instructions that operate on register or immediate operands:

- **Register Operand Mode:** The operand is located in one of the 8-, 16- or 32-bit general registers.
- **Immediate Operand Mode:** The operand is included in the instruction as part of the opcode.

4.6.3 32-BIT MEMORY ADDRESSING MODES

The remaining modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by using combinations of the following four address elements:

- **DISPLACEMENT:** An 8- or 32-bit immediate value, following the instruction.
- **BASE:** The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area.
- **INDEX:** The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters.
- **SCALE:** The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. Scaled index mode is especially useful for accessing arrays or structures.

Combinations of these four components make up the nine additional addressing modes. There is no performance penalty for using any of these addressing combinations, because the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of Base and Index components, which requires one additional clock.

As shown in Figure 4-18, the effective address (EA) of an operand is calculated according to the following formula:

EA = Base Reg + (Index Reg * Scaling) + Displacement

Direct Mode: The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit displacement.

Example: INC Word PTR [500]

Register Indirect Mode: A BASE register contains the address of the operand.

Example: MOV [ECX], EDX

Based Mode: A BASE register's contents are added to a DISPLACEMENT to form the operand's offset.

Example: MOV ECX, [EAX+24]

Index Mode: An INDEX register's contents are added to a DISPLACEMENT to form the operand's offset.

Example: ADD EAX, TABLE[ESI]

Scaled Index Mode: An INDEX register's contents are multiplied by a scaling factor which is added to a DISPLACEMENT to form the operand's offset.

Example: IMUL EBX, TABLE[ESI*4],7

Based Index Mode: The contents of a BASE register are added to the contents of an INDEX register to form the effective address of an operand.

Example: MOV EAX, [ESI] [EBX]

Based Scaled Index Mode: The contents of an INDEX register are multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operand's offset.

Example: MOV ECX, [EDX*8] [EAX]

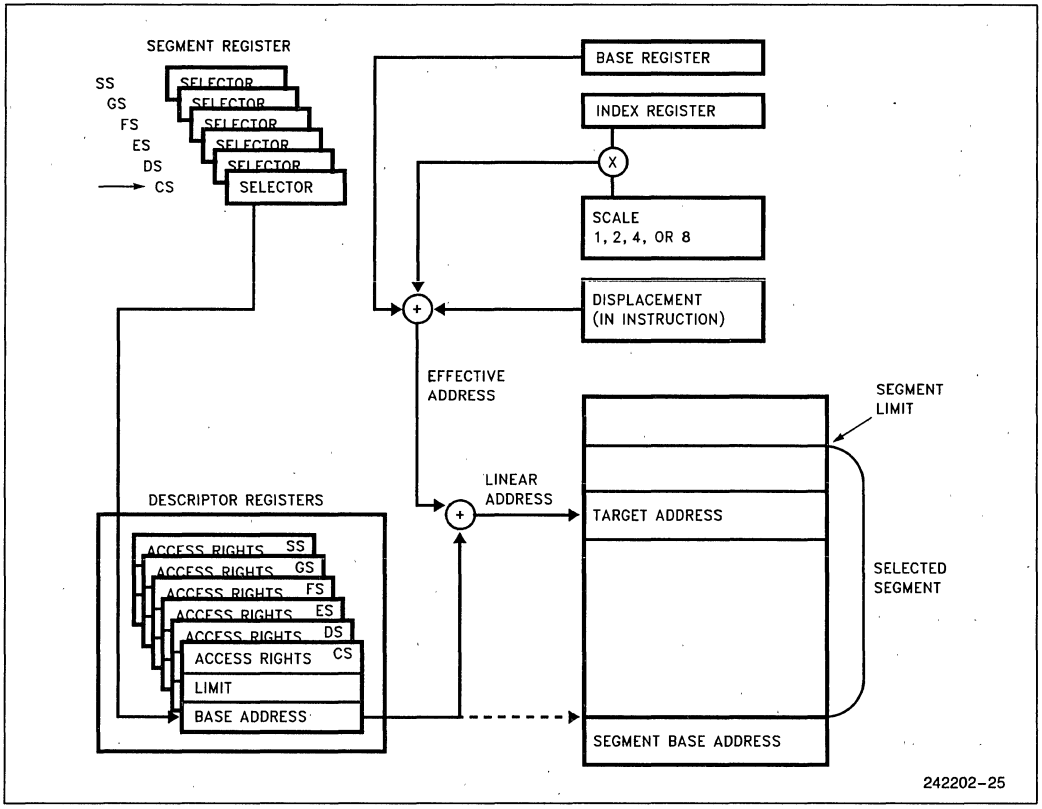


Figure 4-18. Addressing Mode Calculations

Based Index Mode with Displacement: The contents of an INDEX Register and a BASE register's contents and a DISPLACEMENT are all summed together to form the operand offset.

Example: ADD EDX, [ESI] [EBP+00FFFFF0H]

Based Scaled Index Mode with Displacement: The contents of an INDEX register are multiplied by a SCALING factor, the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.

Example: MOV EAX, LOCALTABLE[EDI*4] [EBP+80]

4.6.4 DIFFERENCES BETWEEN 16- AND 32-BIT ADDRESSES

In order to provide software compatibility with 80286 and 8086 processors, the Intel486 processor can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in the CS segment Descriptor. If the D bit is 0 then all operand lengths and effective addresses are assumed to be 16 bits long. If the D bit is 1 then the default length for operands and addresses is 32 bits. In Real Mode the default size for operands and addresses is 16-bits.

Regardless of the default precision of the operands or addresses, the Intel486 processor is able to execute either 16- or 32-bit instructions. This is

specified via the use of override prefixes. Two prefixes, the **Operand Size Prefix** and the **Address Length Prefix**, override the value of the D bit on an individual instruction basis. These prefixes are automatically added by Intel assemblers.

Example: The Intel486 processor is executing in Real Mode and the programmer needs to access the EAX registers. The assembler code for this might be MOV EAX, 32-bit MEMORYOP, ASM486 Macro Assembler automatically determines that an Operand Size Prefix is needed and generates it.

Example: The D bit is 0, and the programmer wishes to use Scaled Index addressing mode to access an array. The Address Length Prefix allows the use of MOV DX, TABLE[ESI*2]. The assembler uses an Address Length Prefix because, with D=0, the default addressing mode is 16-bits.

Example: The D bit is 1, and the program wants to store a 16-bit quantity. The Operand Length Prefix is used to specify only a 16-bit value; MOV MEM16, DX.

The OPERAND LENGTH and Address Length Prefixes can be applied separately or in combination to any instruction. The Address Length Prefix does not allow addresses over 64 Kbytes to be accessed in Real Mode. A memory address which exceeds FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional Intel486 processor addressing modes.

When executing 32-bit code, the Intel486 processor uses either 8-, or 32-bit displacements, and any register can be used as base or index registers. When executing 16-bit code, the displacements are either 8, or 16 bits, and the base and index register conform to the 80286 processor model. Table 4-15 illustrates the differences.

Table 4-15. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX,BP	Any 32-bit GP Register
INDEX REGISTER	SI,DI	Any 32-bit GP Register Except ESP
SCALE FACTOR	none	1, 2, 4, 8
DISPLACEMENT	0, 8, 16 bits	0, 8, 32 bits

4.7 Data Formats

4.7.1 DATA TYPES

The Intel486 processor can support a wide-variety of data types. In the following descriptions, the processor consists of the base architecture registers.

4.7.1.1 Unsigned Data Types

Byte: Unsigned 8-bit quantity

Word: Unsigned 16-bit quantity

Dword: Unsigned 32-bit quantity

The least significant bit (LSB) in a byte is bit 0, and the most significant bit is 7.

4.7.1.2 Signed Data Types

All signed data types assume 2's complement notation. The signed data types contain two fields, a sign bit and a magnitude. The sign bit is the most significant bit (MSB). The number is negative if the sign bit is 1. If the sign bit is 0, the number is positive. The magnitude field consists of the remaining bits in the number. (Refer to Figure 4-19.)

8-bit Integer: Signed 8-bit quantity

16-bit Integer: Signed 16-bit quantity

32-bit Integer: Signed 32-bit quantity

64-bit Integer: Signed 64-bit quantity

The integer core of the Intel486 processors only support 8-, 16- and 32-bit integers. (See section 4.7.1.4, "Floating-Point Data Types.")

4.7.1.3 BCD Data Types

The Intel486 processor supports packed and unpacked binary coded decimal (BCD) data types. A packed BCD data type contains two digits per byte, the lower digit is in bits 0–3 and the upper digit in bits 4–7. An unpacked BCD data type contains 1 digit per byte stored in bits 0–3.

The Intel486 processor supports 8-bit packed and unpacked BCD data types. (Refer to Figure 4-19.)

4.7.1.4 Floating-Point Data Types

In addition to the base registers, the Intel486 DX, IntelDX2, and IntelDX4 processors' on-chip Floating-Point Unit consists of the Floating-Point registers. The Floating-Point Unit data type contain three fields: sign, significand and exponent. The sign field is one bit and is the MSB of the Floating-Point number. The number is negative if the sign bit is 1. If the sign bit is 0, the number is positive. The significand gives the significant bits of the number. The exponent field contains the power of 2 needed to scale the significand. (Refer to Figure 4-19.)

Only the FPU supports Floating-Point data types.

Single Precision Real:	23-bit significand and 8-bit exponent. 32 bits total.
Double Precision Real:	52-bit significand and 11-bit exponent. 64 bits total.
Extended Precision Real:	64-bit significand and 15-bit exponent. 80 bits total.

Floating-Point Unsigned Data Types

The on-chip FPU does not support unsigned data types. (Refer to Figure 4-19.)

Floating-Point Signed Data Types

The on-chip FPU only supports 16-, 32- and 64-bit integers.

Floating-Point BCD Data Types

The on-chip FPU only supports 80-bit packed BCD data types.

4.7.1.5 String Data Types

A string data type is a contiguous sequence of bits, bytes, words or dwords. A string may contain between 1 byte and 4 Gbytes. (Refer to Figure 4-20.)

String data types are only supported by the CPU section of the Intel486 processor.

Byte String:	Contiguous sequence of bytes.
Word String:	Contiguous sequence of words.
Dword String:	Contiguous sequence of dwords.
Bit String:	A set of contiguous bits. In the Intel486 processor bit strings can be up to 4-gigabits long.

4.7.1.6 ASCII Data Types

The Intel486 processor supports ASCII (American Standard Code for Information Interchange) strings and can perform arithmetic operations (such as addition and division) on ASCII data. The Intel486 processor can only operate on ASCII data. (Refer to Figure 4-20.)

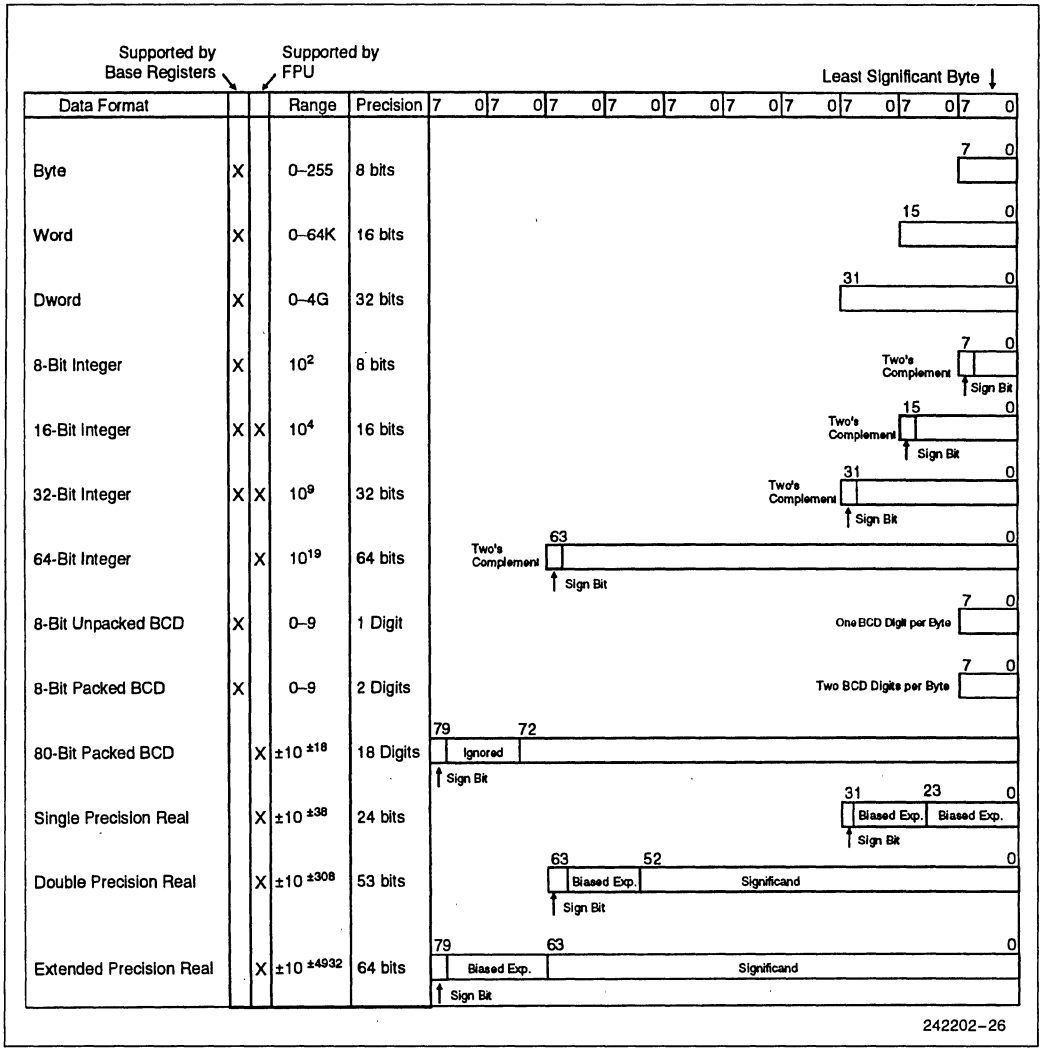


Figure 4-19. Intel486™ Processor Data Types1

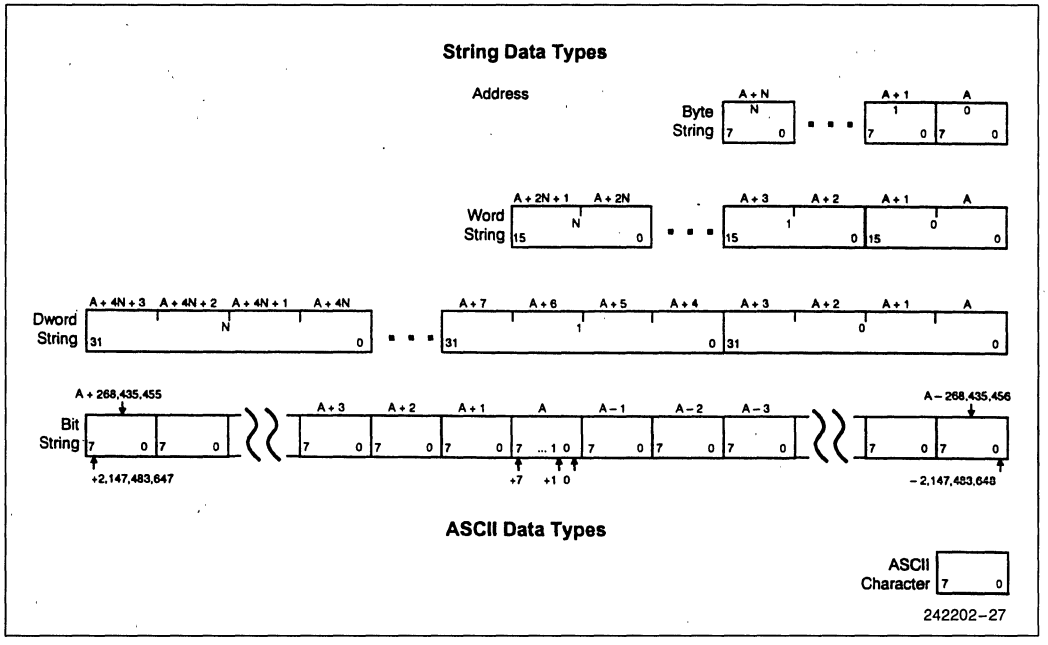


Figure 4-20. String and ASCII Data Types

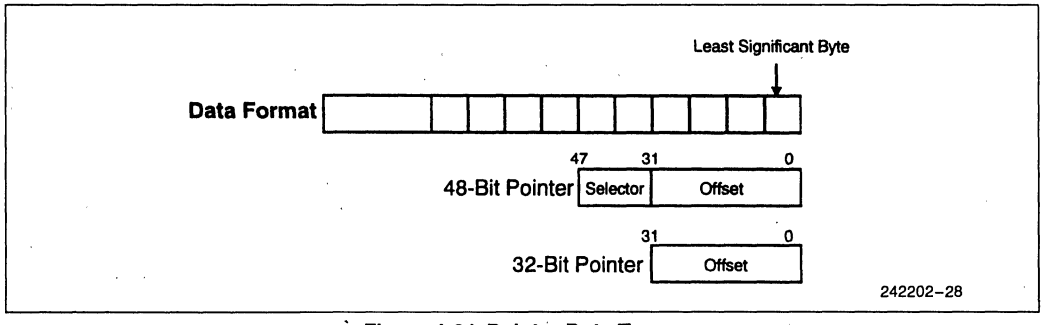


Figure 4-21. Pointer Data Types

4.7.1.7 Pointer Data Types

A pointer data type contains a value that gives the address of a piece of data. Intel486 processors support the following two types of pointers (see Figure 4-21):

- 48-bit Pointer: 16-bit selector and 32-bit offset
- 32-bit Pointer: 32-bit offset

4.7.2 LITTLE ENDIAN vs. BIG ENDIAN DATA FORMATS

The Intel486 processors, as well as all other members of the Intel architecture, use the "little-endian" method for storing data types that are larger than one byte. Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address and the high order byte at the high address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address and the high order byte at the highest address. The address of a word or dword data item is the byte address of the low-order byte.

Figure 4-22 illustrates the differences between the big-endian and little-endian formats for dwords. The 32 bits of data are shown with the low order bit numbered bit 0 and the high order bit numbered 32. Big-endian data is stored with the high-order bits at the lowest addressed byte. Little-endian data is stored with the high-order bits in the highest addressed byte.

The Intel486 processor has the following two instructions that can convert 16- or 32-bit data between the two byte orderings:

- BSWAP (byte swap) handles 4-byte values
- XCHG (exchange) handles 2-byte values

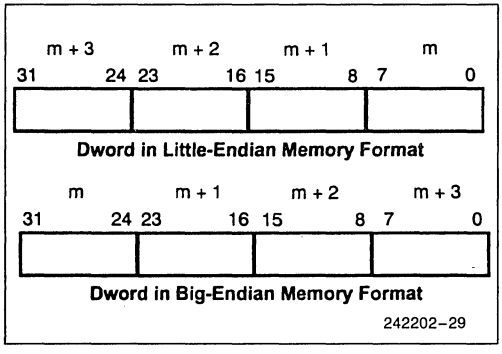


Figure 4-22. Big vs. Little Endian Memory Format

4.8 Interrupts

4.8.1 INTERRUPTS AND EXCEPTIONS

Interrupts and exceptions alter the normal program flow, in order to handle external events, to report errors or exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the Intel486 processors treat software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction. Sections 4.8.3, "Maskable Interrupt," and 4.8.4, "Non-Maskable Interrupt," discuss the differences between Maskable and Non-Maskable interrupts.

Exceptions are classified as faults, traps, or aborts, depending on the way they are reported, and whether or not restart of the instruction causing the exception is supported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. A fault would occur in a virtual memory system when the processor referenced a page or a segment that was not present. The operating system would fetch the page or segment from disk, and then the Intel486 processor would restart the instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction that caused the problem. User defined interrupts are examples of traps. **Aborts** are exceptions that do not permit the precise location of the instruction causing the exception to be determined. Aborts are used to report severe errors, such as a hardware error or illegal values in system tables.

Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point at the instruction causing the exception and include any leading instruction prefixes. Tables 4-16 and 4-17 summarize the possible interrupts for Intel486 processors and shows where the return address points.

Intel486 processors can handle up to 256 different interrupts and/or exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode (see section 5.0, "Real Mode Architecture"), the vectors are 4-byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8-byte quantities, which are put in an Interrupt Descriptor Table. (See section 6.2.3.4, "Interrupt Descriptor Table.") Of the 256 possible interrupts, 32 are reserved for use by Intel, the remaining 224 are free to be used by the system designer.

4.8.2 INTERRUPT PROCESSING

When an interrupt occurs, the following actions happen. First, the current program address and the Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the Intel486 processor which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old Intel486 processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the Intel486 processor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

4.8.3 MASKABLE INTERRUPT

Maskable interrupts are the most common way used by the Intel486 processor to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled high and the Interrupt Flag bit (IF) is enabled. The Intel486 processor only responds to interrupts between instructions, (REPeat String instructions, have an "interrupt window," between memory moves, which allows interrupts during long string moves). When an interrupt occurs, the Intel486 processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt, (one of 224 user defined interrupts). The exact nature of the interrupt sequence is discussed in section 10.2.10, "Interrupt Acknowledge."

The IF bit in the EFLAG registers is reset when an interrupt is being serviced. This effectively disables servicing additional interrupts during an interrupt service routine. However, the IF may be set explicitly by the interrupt handler, to allow the nesting of interrupts. When an IRET instruction is executed, the original state of the IF is restored.

Table 4-16. Interrupt Vector Assignments

Function	Interrupt Number	Instruction that Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	Any instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any illegal instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any instruction that can generate an exception		ABORT
Intel Reserved	9			
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	1	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Intel Reserved	15			
Alignment Check Interrupt	17	Unaligned Memory Access	YES	FAULT
Intel Reserved	18–31			
Two Byte Interrupt	0–255	INT n	NO	TRAP

*Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

Table 4-17. FPU Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Floating-Point Error	16	Floating-Point, WAIT	YES	FAULT

4.8.4 NON-MASKABLE INTERRUPT

Non-maskable interrupts provide a method of servicing very high priority interrupts. A common example of the use of a non-maskable interrupt (NMI) would be to activate a power failure routine or SMI# to activate a power saving mode. When the NMI input is pulled high, it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for an NMI.

While executing the NMI servicing procedure, the Intel486 processor will not service further NMI requests until an interrupt return (IRET) instruction is executed or the processor is reset (RSM in the case of SMI#). If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

4.8.5 SOFTWARE INTERRUPTS

A third type of interrupt/exception for the Intel486 processor is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the *n*th vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt is the single step interrupt. It is discussed in section 12.2, "Single-Step Trap."

4.8.6 INTERRUPT AND EXCEPTION PRIORITIES

Interrupts are externally-generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input or SMI# input) are recognized at instruction boundaries. When more than one interrupt or external event are **both** recognized at the **same** instruction boundary, the Intel486 processor invokes the highest priority routine first. (See list below.) If, after the NMI service routine has been invoked, maskable interrupts are still enabled, then the Intel486 processor will invoke the appropriate interrupt service routine.

Priority for Servicing External Events for All Intel486 Processors Except the Write-Back Enhanced Intel486 processors

1. RESET/SRESET
2. FLUSH#
3. SMI#
4. NMI
5. INTR
6. STPCLK#

NOTE:
STPCLK# will be recognized while in an interrupt service routine or an SMM handler.

For the Write-Back Enhanced Intel486 processors, the priority of servicing external events is modified from the standard Intel486 processor. The list below shows the priority for write-back enhanced mode.

Priority for Servicing External Events for the Write-Back Enhanced Intel486 processors

1. RESET
2. FLUSH#
3. SRESET
4. SMI#
5. NMI
6. INTR
7. STPCLK#

Exceptions are internally-generated events. Exceptions are detected by the Intel486 processor if, in the course of executing an instruction, the Intel486 processor detects a problematic condition. The IntelDX4 processor then immediately invokes the appropriate exception service routine. The state of the Intel486 processor is such that the instruction causing the exception can be restarted. If the exception service routine has taken care of the problematic condition, the instruction will execute without causing the same exception.

It is possible for a single instruction to generate several exceptions (for example, transferring a single operand could generate two page faults if the operand location spans two "not present" pages). However, only one exception is generated upon each attempt to execute the instruction. Each exception service routine should correct its corresponding exception, and restart the instruction. In this manner, exceptions are serviced until the instruction executes successfully.

As the Intel486 processor executes instructions, it follows a consistent cycle in checking for exceptions. Consider the case of the Intel486 processor having just completed an instruction. It then performs the checks listed in Table 4-18 before reaching the point where the next instruction is completed. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution. Checking for EM, TS, or FPU error status only occurs for processors with on-chip Floating-Point Units.

Table 4-18. Sequence of Exception Checking

Sequence	Description
1	Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2	Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
3	Check for external NMI and INTR.
4	Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5	Check for Page Faults that prevented fetching the entire next instruction (exception 14).
6	Check for Faults decoding the next instruction (exception 6 if illegal opcode; exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only (see section 6.5.4, "Protection and I/O Permission Bitmap"); or exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e., not at IOPL or at CPL = 0).
7	If WAIT opcode, check if TS = 1 and MP = 1 (exception 7 if both are 1).
8	If opcode for Floating-Point Unit, check if EM = 1 or TS = 1 (exception 7 if either are 1).
9	If opcode for Floating-Point Unit (FPU), check FPU error status (exception 16 if error status is asserted).
10	Check in the following order for each memory reference required by the instruction: a. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13). b. Check for Page Faults that prevent transferring the entire memory quantity (exception 14).

NOTE:
The order stated supports the concept of the paging mechanism being "underneath" the segmentation mechanism. Therefore, for any given code or data reference in memory, segmentation exceptions are generated before paging exceptions are generated.

4.8.7 INSTRUCTION RESTART

The Intel486 processor fully supports restarting all instructions after faults. If an exception is detected in the instruction to be executed (exception categories 4 through 10 in Table 4-18), the Intel486 processor invokes the appropriate exception service routine.

The Intel486 processor is in a state that permits restart of the instruction, for all cases except the following. An instruction causes a task switch to a task whose Task State Segment is **partially** "not present." (An entirely "not present" TSS is restartable.) Partially present TSSs can be avoided either by keeping the TSSs of such tasks present in memory, or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4 Kbytes or less).

NOTE:
Such cases are easily avoided by proper design of the operating system.

4.8.8 DOUBLE FAULT

A Double Fault (exception 8) results when the Intel486 processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13) but, in the process of doing so, detects an exception other than a Page Fault (exception 14).

A Double Fault (exception 8) will also be generated when the Intel486 processor attempts to invoke the Page Fault (exception 14) service routine, and detects an exception other than a second Page Fault. In any functional system, the entire Page Fault service routine must remain "present" in memory.

When a Double Fault occurs, the Intel486 processor invokes the exception service routine for exception 8.

4.8.9 FLOATING-POINT INTERRUPT VECTORS

Several interrupt vectors of the Intel486 DX, IntelDX2, and IntelDX4 processors are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 4-19 shows these interrupts and their causes.

Table 4-19. Interrupt Vectors Used by FPU

Interrupt Number	Cause of Interrupt
7	A Floating-Point instruction was encountered when EM or TS of the Intel486™ DX, IntelDX2™, and IntelDX4™ processor control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either a Floating-Point or WAIT instruction causes interrupt 7. This indicates that the current FPU context may not belong to the current task.
13	The first word or doubleword of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the Floating-Point instruction that caused the exception, including any prefixes. The FPU has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only Floating-Point and WAIT instructions can cause this interrupt. The Intel486 DX, IntelDX2, and IntelDX4 processors return address pushed onto the stack of the exception handler points to a WAIT or Floating-Point instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the FPU. The FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE instructions can not cause this interrupt.

5.0 REAL MODE ARCHITECTURE

5.1 Introduction

When the Intel486 processor is reset or powered up, it is initialized in Real Mode. Real Mode has the same base architecture as the 8086 processor, except that it allows access to the 32-bit register set of the Intel486 processor. The Intel486 processor addressing mechanism, memory size and interrupt handling are identical to those of Real Mode on the 80286 processor.

All of the Intel486 processor instructions are available in Real Mode (except those instructions listed in section 6.5.4, "Protection and I/O Permission Bitmap"). The default operand size in Real Mode is 16 bits, as in the 8086 processor. In order to use the 32-bit registers and addressing modes, override prefixes must be used. Also, the segment size on the Intel486 processor in Real Mode is 64 Kbytes, forcing 32-bit effective addresses to have a value less than 0000FFFFH. The primary purpose of Real Mode is to enable Protected Mode Operation.

The LOCK prefix on the Intel486 processor, even in Real Mode, is more restrictive than on the 80286 processor. This is due to the addition of paging on the Intel486 processor in Protected Mode and Virtual 8086 Mode. Paging makes it impossible to guarantee that repeated string instructions can be LOCKed. The Intel486 processor can not require that all pages holding the string be physically present in memory. Hence, a Page Fault (exception 14) might have to be taken during the repeated string instruction. Therefore, the LOCK prefix can not be supported during repeated string instructions.

Table 5-1 lists the only instruction forms where the LOCK prefix is legal on the Intel486 processor.

An exception 6 will be generated if a LOCK prefix is placed before any instruction form or opcode not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above. For example, even the ADD Reg, Mem is not LOCKable, because the Mem operand is not the destination (and therefore no memory read/modify/operation is being performed).

Because, on the Intel486 processor, repeated string instructions are not LOCKable, it is not possible to LOCK the bus for a long period of time. Therefore, the LOCK prefix is not IOPL-sensitive on the Intel486 processor. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed above.

Table 5-1. Instruction Forms Where LOCK Prefix Is Legal

Opcode	Operands (Dest, Source)
BIT Test and SET/RESET/COMPLEMENT	Mem, Reg/immed
XCHG	Reg, Mem
CHG	Mem, Reg
ADD, OR, ADC, SBB, AND, SUB, XOR	Mem, Reg/immed
NOT, NEG, INC, DEC	Mem
CMPXCHG, XADD	Mem, Reg

5.2 Memory Addressing

In Real Mode the maximum memory size is limited to 1 megabyte. (See Figure 5-1.) Thus, only address lines A2–A19 are active. (Exception, after RESET address lines A20–A31 are high during CS-relative memory cycles until an intersegment jump or call is executed. See section 9.5, "Reset and Initialization".)

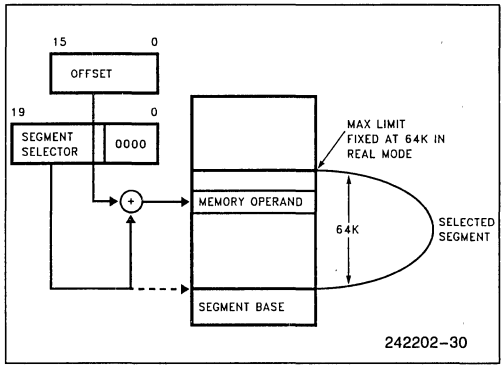


Figure 5-1. Real Address Mode Addressing

Because paging is not allowed in Real Mode, the linear addresses are the same as the physical addresses. Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register, which is shifted left by four bits to an effective address. This addition results in a physi-

cal address from 00000000H to 0010FFEFH. This is compatible with 80286 Real Mode. Because segment registers are shifted left by 4 bits, Real Mode segments always start on 16-byte boundaries.

All segments in Real Mode are exactly 64-Kbytes long, and may be read, written, or executed. The Intel486 processor will generate an exception 13 if a data operand or instruction fetch occurs past the end of a segment (i.e., if an operand has an offset greater than FFFFH, for example, a word with a low byte at FFFFH and the high byte at 0000H).

Segments may be overlapped in Real Mode. Thus, if a particular segment does not use all 64 Kbytes, another segment can be overlaid on top of the unused portion of the previous segment. This allows the programmer to minimize the amount of physical memory needed for a program.

5.3 Reserved Locations

There are two fixed areas in memory which are reserved in Real address mode: system initialization area and the interrupt table area. Locations 00000H through 003FFH are reserved for interrupt vectors. Each one of the 256 possible interrupts has a 4-byte jump vector reserved for it. Locations FFFFFFF0H through FFFFFFFFH are reserved for system initialization.

5.4 Interrupts

Many of the exceptions shown in Table 4-16 and discussed in section 4.8.3, "Maskable Interrupt," are not applicable to Real Mode operation, in particular exceptions 10, 11, 14, 17, which do not happen in Real Mode. Other exceptions have slightly different meanings in Real Mode; Table 5-2 identifies these exceptions.

5.5 Shutdown and Halt

The HALT instruction stops program execution and prevents the Intel486 processor from using the local bus until restarted. Either NMI, INTR with interrupts enabled (IF = 1), or RESET will force the Intel486 processor out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

As in the case of protected mode, the shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions, as follows:

- An interrupt or an exception occurs (exceptions 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table (i.e., there is not an interrupt handler for the interrupt).
- A CALL, INT or PUSH instruction attempts to wrap around the stack segment when SP is not even (i.e., pushing a value on the stack when SP = 0001 resulting in a stack segment greater than FFFFH).

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 0017H) and the stack has enough room to contain the vector and flag information (i.e., SP is greater than 0005H). If these conditions are not met, the Intel486 processor is unable to execute the NMI and executes another shutdown cycle. In this case, the Intel486 processor remains in the shutdown and can only exit via the RESET input.

Table 5-2. Exceptions with Different Meanings in Real Mode (see Table 4-17)

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT Vector is not within table limit	Before Instruction
CS, DS, ES, FS, GS Segment overrun exception	13	Word memory reference beyond offset = FFFFH. An attempt to execute past the end of CS segment.	Before Instruction
SS Segment overrun exception	12	Stack Reference beyond offset = FFFFH	Before Instruction

6.0 PROTECTED MODE ARCHITECTURE

The complete capabilities of the Intel486 processor are unlocked when the Intel486 processor operates in Protected Virtual Address Mode (Protected Mode). Protected Mode vastly increases the linear address space to four Gbytes (2^{32} bytes) and allows the running of virtual memory programs of almost unlimited size (64 terabytes or 2^{46} bytes). In addition Protected Mode allows the Intel486 processor to run all of the existing 8086, 80286 and Intel386 processor software, while providing a sophisticated memory management and a hardware-assisted protection mechanism. Protected Mode allows the use of additional instructions especially optimized for supporting multitasking operating systems. The base architecture of the Intel486 processor remains the same, the registers, instructions, and addressing modes described in the previous sections are retained. The main difference between Protected Mode and Real Mode from a programmer's view is the increased address space and a different addressing mechanism.

6.1 Addressing Mechanism

Like Real Mode, Protected Mode uses two components to form the logical address, a 16-bit selector is used to determine the linear base address of a segment, the base address is added to a 32-bit effective address to form a 32-bit linear address. The linear address is then either used as the 32-bit physical address, or if paging is enabled the paging mechanism maps the 32-bit linear address into a 32-bit physical address.

The difference between the two modes lies in calculating the base address. In Protected Mode the selector is used to specify an index into an operating system defined table. (See Figure 6-1.) The table contains the 32-bit base address of a given segment. The physical address is formed by adding the base address obtained from the table to the offset.

Paging provides an additional memory management mechanism which operates only in Protected Mode. Paging provides a means of managing the very large segments of the Intel486 processor. As such, paging operates beneath segmentation. The paging mechanism translates the protected linear address which comes from the segmentation unit into a physical address. Figure 6-2 shows the complete Intel486 processor addressing mechanism with paging enabled.

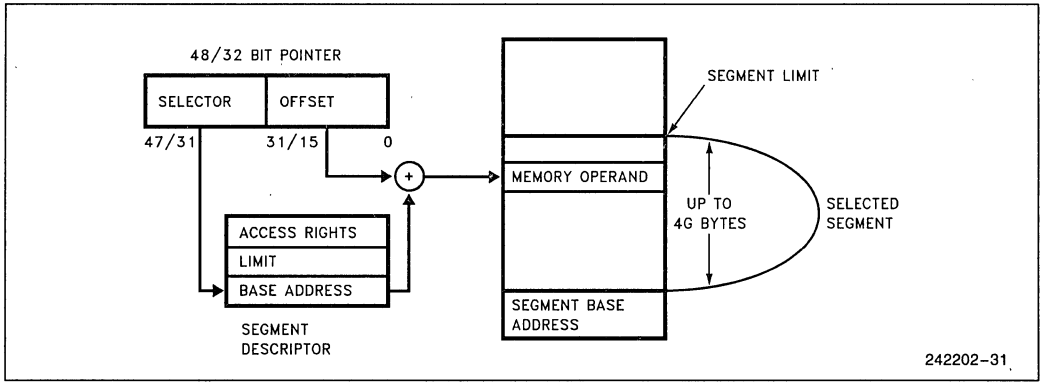


Figure 6-1. Protected Mode Addressing

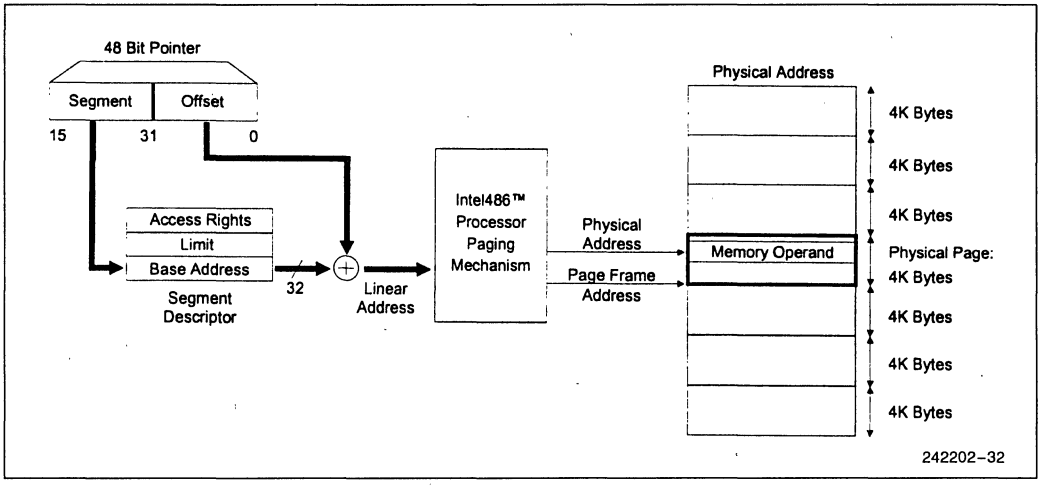


Figure 6-2. Paging and Segmentation

6.2 Segmentation

6.2.1 SEGMENTATION INTRODUCTION

Segmentation is one method of memory management. Segmentation provides the basis for protection. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about a segment is stored in an 8-byte data structure called a descriptor. All of the descriptors in a system are contained in tables recognized by hardware.

6.2.2 TERMINOLOGY

The following terms are used throughout the discussion of descriptors, privilege levels and protection:

PL: Privilege Level—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged. More privileged levels are numerically smaller than less privileged levels.

RPL: Requester Privilege Level—The privilege level of the original supplier of the selector. RPL is determined by the **least two** significant bits of a selector.

DPL: Descriptor Privilege Level—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

CPL: Current Privilege Level—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: Effective Privilege Level—The effective privilege level is the least privileged of the RPL and DPL. Because smaller privilege level **values** indicate greater privilege, EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

6.2.3 DESCRIPTOR TABLES

6.2.3.1 Descriptor Tables Introduction

The descriptor tables define all of the segments which are used in an Intel486 processor system. (See Figure 6-3.) There are three types of tables on the Intel486 processor which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Descriptor Table. All of the tables are variable length memory arrays. They can range in size between 8 bytes and 64 Kbytes. Each table can hold up to 8192 8-byte descriptors. The upper 13 bits of a selector are used as an index into the descriptor table. The tables have registers associated with them which hold the 32-bit linear base address, and the 16-bit limit of each table.

Each of the tables has a register associated with it, the GDTR, LDTR, and the IDTR (see Figure 6-3). The LGDT, LLDT, and LIDT instructions load the base and limit of the Global, Local, and Interrupt Descriptor Tables, respectively, into the appropriate register. The SGDT, SLDT, and SIDT store the base and limit values. These tables are manipulated by the operating system. Therefore, the load descriptor table instructions are privileged instructions.

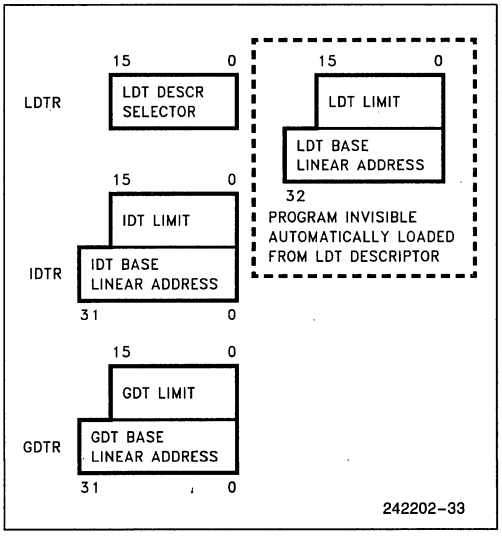


Figure 6-3. Descriptor Table Registers

6.2.3.2 Global Descriptor Table

The Global Descriptor Table (GDT) contains descriptors which are possibly available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for descriptors which are used for servicing interrupts (i.e., interrupt and trap descriptors). Every Intel486 processor system contains a GDT. Generally the GDT contains code and data segments used by the operating systems and task state segments, and descriptors for the LDTs in a system.

The first slot of the GDT corresponds to the null selector and is not used. The null selector defines a null pointer value.

6.2.3.3 Local Descriptor Table

Local Descriptor Tables contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This provides both isolation and protection for a task's segments, while still allowing global data to be shared among tasks.

Unlike the 6-byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a LDT descriptor in the GDT.

6.2.3.4 Interrupt Descriptor Table

The third table needed for Intel486 processor systems is the Interrupt Descriptor Table. (See Figure 6-4.) The IDT contains the descriptors which point to the location of up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates, and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 Intel Reserved Interrupts. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced via INT instructions, external interrupt vectors, and exceptions. (See section 4.8, "Interrupts.")

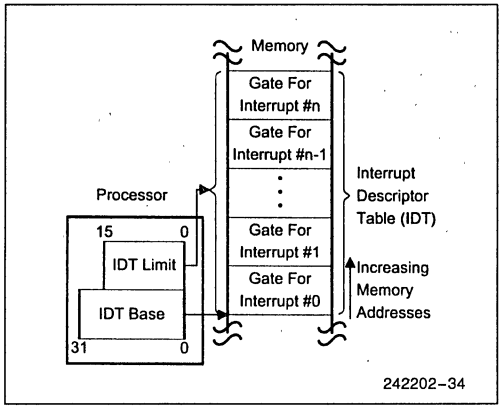


Figure 6-4. Interrupt Descriptor Table Register Use

6.2.4 DESCRIPTORS

6.2.4.1 Descriptor Attribute Bits

The object to which the segment selector points to is called a descriptor. Descriptors are eight-byte quantities that contain attributes about a given region of linear address space (i.e., a segment). These attributes include the 32-bit base linear address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write or execute privileges, the default size of the operands (16-bit or 32-bit), and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. All segments on the Intel486 processor have three attribute fields in common: the **P** bit, the **DPL** bit, and the **S** bit. The Present **P** bit is 1 if the segment is loaded in physical memory. If P=0, any attempt to access this segment will cause a not present exception (exception 11). The Descriptor Privilege Level **DPL** is a two-bit field that specifies the protection level 0–3 associated with a segment.

The Intel486 processor has two main categories of segments: system segments and non-system segments (for code and data). The segment **S** bit in the segment descriptor determines if a given segment is a system segment or a code or data segment. If the **S** bit is 1, the segment is either a code or data segment. If it is 0, the segment is a system segment.

6.2.4.2 Intel486™ Processor Code, Data Descriptors (S=1)

Figure 6-5 shows the general format of a code and data descriptor and Table 6-1 illustrates how the bits in the Access Rights Byte are interpreted. The Access Rights Bytes is bits 24–31 associated with the segment limit.

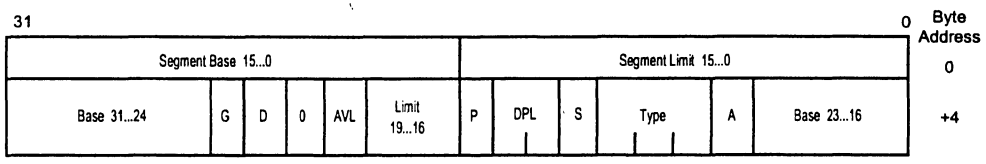
Code and data segments have several descriptor fields in common. The accessed **A** bit is set whenever the processor accesses a descriptor. The **A** bit is used by operating systems to keep usage statistics on a given segment. The **G** bit, or granularity bit, specifies if a segment length is byte-granular or page-granular. Intel486 processor segments can be one megabyte long with byte granularity (G=0) or four gigabytes with page granularity (G=1), (i.e., 2^{20} pages each page is 4 Kbytes in length). The granularity is totally unrelated to paging. An Intel486 processor system can consist of segments with byte granularity, and page granularity, whether or not paging is enabled.

The executable **E** bit tells if a segment is a code or data segment. A code segment (E=1, S=1) may be execute-only or execute/read as determined by the Read **R** bit. Code segments are execute only if R=0, and execute/read if R=1. Code segments may never be written into.

NOTE:
Code segments may be modified via aliases. Aliases are writeable data segments which occupy the same range of linear address space as the code segment.

The **D** bit indicates the default length for operands and effective addresses. If D=1 then 32-bit operands and 32-bit addressing modes are assumed. If D=0 then 16-bit operands and 16-bit addressing modes are assumed. Therefore all existing 80286 code segments will execute on the Intel486 processor assuming the D bit is set 0.

Another attribute of code segments is determined by the conforming **C** bit. Conforming segments, C=1, can be executed and shared by programs at different privilege levels. (See section 6.3, "Protection.")



BASE	Base Address of the segment
LIMIT	The length of the segment
P	Present Bit 1 = Present, 0 = Not Present
DPL	Descriptor Privilege Level 0–3
S	Segment Descriptor 0 = System Descriptor, 1 = Code or Data Segment Descriptor
TYPE	Type of Segment
A	Accessed Bit
G	Granularity Bit 1 = Segment length is page granular, 0 = Segment length is byte granular
D	Default Operation Size (recognized in code segment descriptors only) 1 = 32-bit segment, 0 = 16-bit segment
0	Bit must be zero (0) for compatibility with future processors
AVL	Available field for user or OS

NOTE:
In a maximum-size segment (i.e., a segment with G = 1 and segment limit 19. . .0 = FFFFFH), the lowest 12 bits of the segment base should be zero (i.e., segment base 11. . .000 = 000H).

Figure 6-5. Segment Descriptors

Table 6-1. Access Rights Byte Definition for Code and Data Descriptions

Bit Position	Name	Function	
7	Present (P)	P = 1	Segment is mapped into physical memory.
		P = 0	No mapping to physical memory exits, base and limit are not used.
6–5	Descriptor Privilege Level (DPL)		Segment privilege attribute used in privilege tests.
4	Segment Descriptor (S)	S = 1	Code or Data (includes stacks) segment descriptor.
		S = 0	System Segment Descriptor or Gate Descriptor.
		If Data Segment (S = 1, E = 0)	
3	Executable (E)	E = 0	Descriptor type is data segment:
2	Expansion Direction (ED)	ED = 0	Expand up segment, offsets must be ≤ limit.
		ED = 1	Expand down segment, offsets must be > limit.
1	Writeable (W)	W = 0	Data segment may not be written into.
		W = 1	Data segment may be written into.
		If Code Segment (S = 1, E = 1)	
3	Executable (E)	E = 1	Descriptor type is code segment:
2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.
1	Readable (R)	R = 0	Code segment may not be read.
		R = 1	Code segment may be read.
0	Accessed (A)	A = 0	Segment has not been accessed.
		A = 1	Segment selector has been loaded into segment register or used by selector test instructions.

Segments identified as data segments (E=0, S=1) are used for two types of Intel486 processor segments: stack and data segments. The expansion direction **(ED)** bit specifies if a segment expands downward (stack) or upward (data). If a segment is a stack segment all offsets must be greater than the segment limit. On a data segment all offsets must be less than or equal to the limit. In other words, stack segments start at the base linear address plus the maximum segment limit and grow down to the base linear address plus the limit. On the other hand, data segments start at the base linear address and expand to the base linear address plus limit.

The write **W** bit controls the ability to write into a segment. Data segments are read-only if W=0. The stack segment must have W=1.

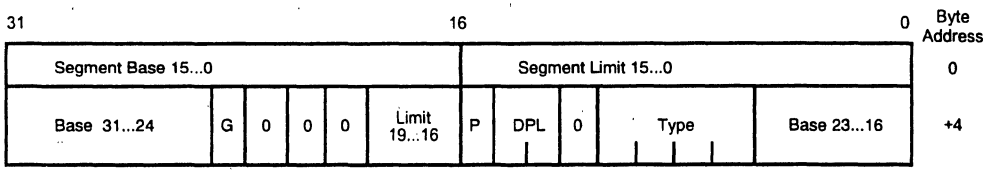
The **B** bit controls the size of the stack pointer register. If B=1, then PUSHes, POPs, and CALLs all use the 32-bit ESP register for stack references and assume an upper limit of FFFFFFFFH. If B=0, stack instructions all use the 16-bit SP register and assume an upper limit of FFFFH.

6.2.4.3 System Descriptor Formats

System segments describe information about operating system tables, tasks, and gates. Figure 6-6 shows the general format of system segment descriptors and the various types of system segments. Intel486 processor system descriptors contain a 32-bit base linear address and a 20-bit segment limit. The 80286 system descriptors have a 24-bit base address and a 16-bit segment limit. The 80286 system descriptors are identified by the upper 16 bits being all zero.

6.2.4.4 LDT Descriptors (S=0, TYPE=2)

LDT descriptors (S=0, TYPE=2) contain information about LDTs. LDTs contain a table of segment descriptors unique to a particular task. Because the instruction to load the LDTR is only available at privilege level 0, the DPL field is ignored. LDT descriptors are only allowed in the GDT.



Type	Defines
0	Invalid
1	Available 80286 TSS
2	LDT
3	Busy 80286 TSS
4	80286 call gate
5	Task Gate (for 80286, Intel486™ processor task)
6	80286 interrupt gate
7	80286 trap gate
8	Invalid
9	Available Intel486 processor TSS
A	Undefined (Intel Reserved)
B	Busy Intel486 processor TSS
C	Intel486 processor call gate
D	Undefined (Intel Reserved)
E	Intel486 processor
F	Intel486 processor

Figure 6-6. System Segment Descriptors

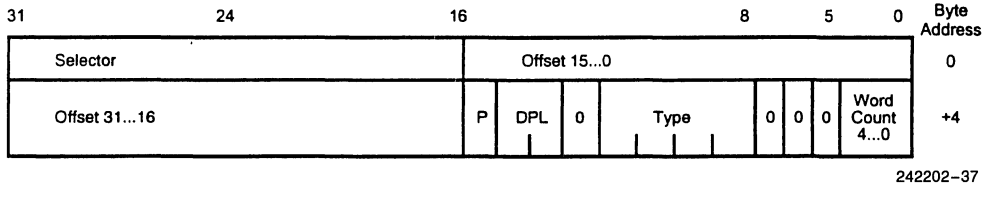
6.2.4.5 TSS Descriptors (S=0, TYPE=1, 3, 9, B)

A Task State Segment (TSS) descriptor contains information about the location, size, and privilege level of a Task State Segment (TSS). A TSS in turn is a special fixed format segment which contains all the state information for a task and a linkage field to permit nesting tasks. The TYPE field is used to indicate whether the task is currently BUSY (i.e., on a chain of active tasks) or the TSS is available. The TYPE field also indicates if the segment contains an 80286 processor TSS or an Intel486 processor TSS. The Task Register (TR) contains the selector which points to the current Task State Segment.

6.2.4.6 Gate Descriptors (S=0, TYPE=4-7, C, F)

Gates are used to control access to entry points within the target code segment. The various types of gate descriptors are **call** gates, **task** gates, **interrupt** gates, and **trap** gates. Gates provide a level of indirection between the source and destination of the control transfer. This indirection allows the processor to automatically perform protection checks. It also allows system designers to control entry points to the operating system. Call gates are used to change privilege levels (see section 6.3, "Protection"), task gates are used to perform a task switch, and interrupt and trap gates are used to specify interrupt service routines.

Figure 6-7 shows the format of the four types of gate descriptors. Call gates are primarily used to transfer program control to a more privileged level. The call gate descriptor consists of three fields: the access byte, a long pointer (selector and offset) which points to the start of a routine and a word count which specifies how many parameters are to be copied from the caller's stack to the stack of the called routine. The word count field is only used by call gates when there is a change in the privilege level, other types of gates ignore the word count field.



Gate Descriptor Fields

Name	Value	Description
Type	4	80286 call gate
	5	Task gate (for 80286 or Intel486 processor task)
	6	80286 interrupt gate
	7	80286 trap gate
	C	Intel486™ processor call gate
	E	Intel486 processor interrupt gate
	F	Intel486 processor trap gate
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid

DPL—least privileged level at which a task may access the gate. WORD COUNT 0–31—the number of parameters to copy from caller's stack to the called procedure's stack. The parameters are 32-bit quantities for Intel486 processor gates, and 16-bit quantities for 80286 gates.

DESTINATION SELECTOR	16-bit selector	Selector to the target code segment or Selector to the target task state segment for task gate
DESTINATION OFFSET	offset	Entry point within the target code segment 16-bit 80286 32-bit Intel486 processor

Figure 6-7. Gate Descriptor Formats

Interrupt and trap gates use the destination selector and destination offset fields of the gate descriptor as a pointer to the start of the interrupt or trap handler routines. The difference between interrupt gates and trap gates is that the interrupt gate disables interrupts (resets the IF bit), while the trap gate does not.

Task gates are used to switch tasks. Task gates may only refer to a task state segment. (See section 6.3.6, "Task Switching.") Therefore, only the destination selector portion of a task gate descriptor is used, and the destination offset is ignored.

Exception 13 is generated when a destination selector does not refer to a correct descriptor type, i.e., a code segment for an interrupt, trap or call gate, a TSS for a task gate.

The access byte format is the same for all gate descriptors. P=1 indicates that the gate contents are valid. P=0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the descriptor privilege level and specifies when this descriptor may be used by a task. (See section 6.3, "Protection.") The S field, bit 4 of the access rights byte, must be 0 to indicate a system control descriptor. The type field specifies the descriptor type as indicated in Figure 6-7.

6.2.4.7 Differences Between Intel486™ Processor and 80286 Descriptors

In order to provide operating system compatibility between 80286 and Intel486 processors, the Intel486 processor supports all of the 80286 segment descriptors. Figure 6-8 shows the general format of an 80286 system segment descriptor. The only differences between 80286 and Intel486 processor descriptor formats are that the values of the type fields, and the limit and base address fields have been expanded for the Intel486 processor. The 80286 system segment descriptors contained a 24-bit base address and 16-bit limit, while the Intel486 processor system segment descriptors have a 32-bit base address, a 20-bit limit field, and a granularity bit.

By supporting 80286 system segments the Intel486 processor is able to execute 80286 application programs on an Intel486 processor operating system. This is possible because the Intel486 processor automatically understands which descriptors are 80286-style descriptors and which descriptors are Intel486 processor-style descriptors. In particular, if the upper word of a descriptor is zero, then that descriptor is an 80286-style descriptor.

The only other differences between 80286-style descriptors and Intel486 processor descriptors is the interpretation of the word count field of call gates and the B bit. The word count field specifies the number of 16-bit quantities to copy for 80286 call gates and 32-bit quantities for Intel486 processor call gates. The B bit controls the size of PUSHes when using a call gate; if B=0 PUSHes are 16 bits, if B=1 PUSHes are 32 bits.

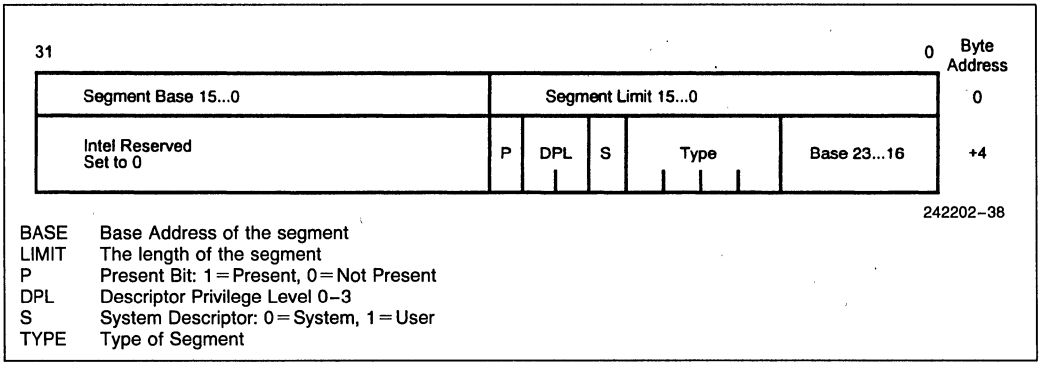


Figure 6-8. 80286 Code and Data Segment Descriptors

6.2.4.8 Selector Fields

A selector in Protected Mode has three fields: Local or Global Descriptor Table Indicator (TI), Descriptor Entry Index (Index), and Requester (the selector's) Privilege Level (RPL) as shown in Figure 6-9. The TI bits select one of two memory-based tables of descriptors (the Global Descriptor Table or the Local Descriptor Table). The Index selects one of 8K descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

6.2.4.9 Segment Descriptor Cache

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Because descriptor caches only change when a segment register is changed, programs that modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.

6.2.4.10 Segment Descriptor Register Settings

The contents of the segment descriptor cache vary depending on the mode the Intel486 processor is operating in. When operating in Real Address Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 6-10. For compatibility with the 8086 architecture, the base is set to sixteen times the current selector value, the limit is fixed at 0000FFFFH, and the attributes are fixed so as to indicate the segment is present and fully usable. In Real Address Mode, the internal "privilege level" is always fixed to the highest level, level 0, so I/O and other privileged opcodes may be executed.

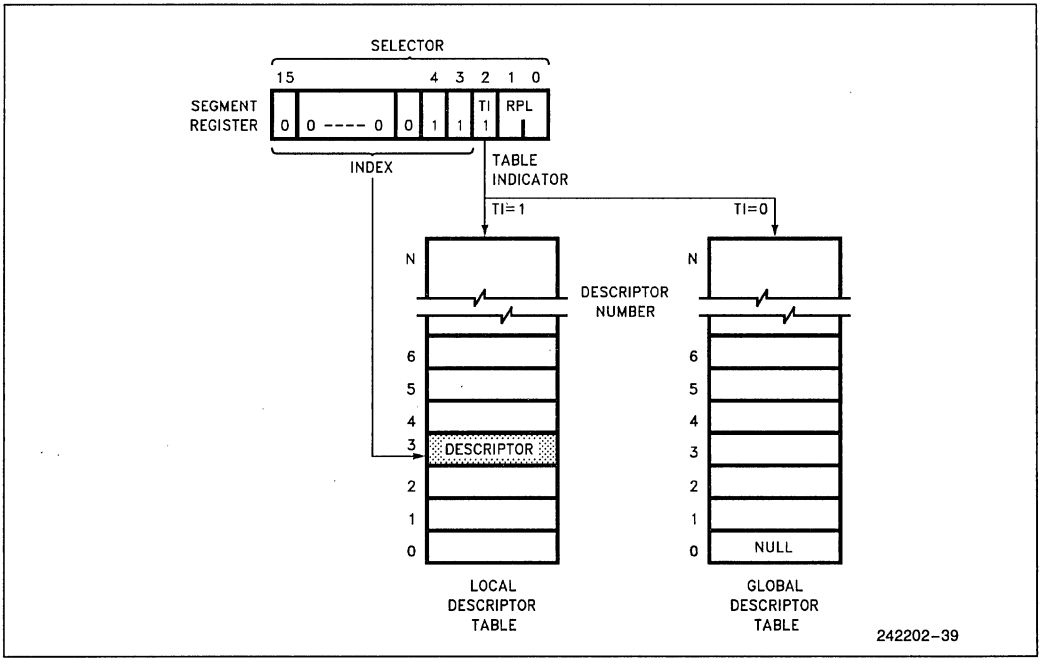
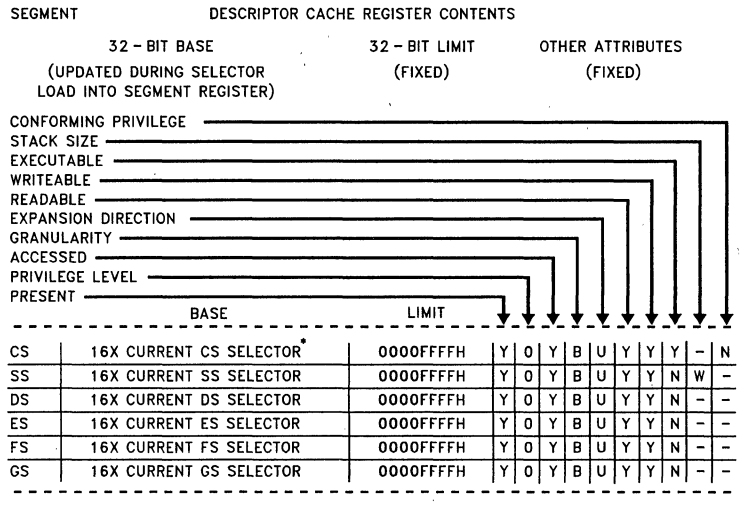


Figure 6-9. Example Descriptor Selection



*Except the 32-bit CS base is initialized to FFFFF000H after reset until first intersegment control transfer (i.e., intersegment CALL, or intersegment JMP, or INT). (See Figure 6-12 for an example.)

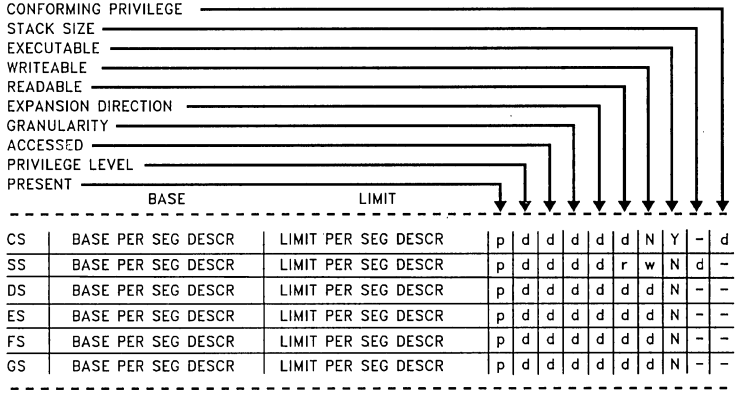
Key:

Y	= yes	D	= expand down
N	= no	B	= byte granularity
0	= privilege level 0	P	= page granularity
1	= privilege level 1	W	= push/pop 16-bit words
2	= privilege level 2	F	= push/pop 32-bit dwords
3	= privilege level 3	—	= does not apply to that segment cache register
U	= expand up		

Figure 6-10. Segment Descriptor Caches for Real Address Mode (Segment Limit and Attributes Are Fixed)

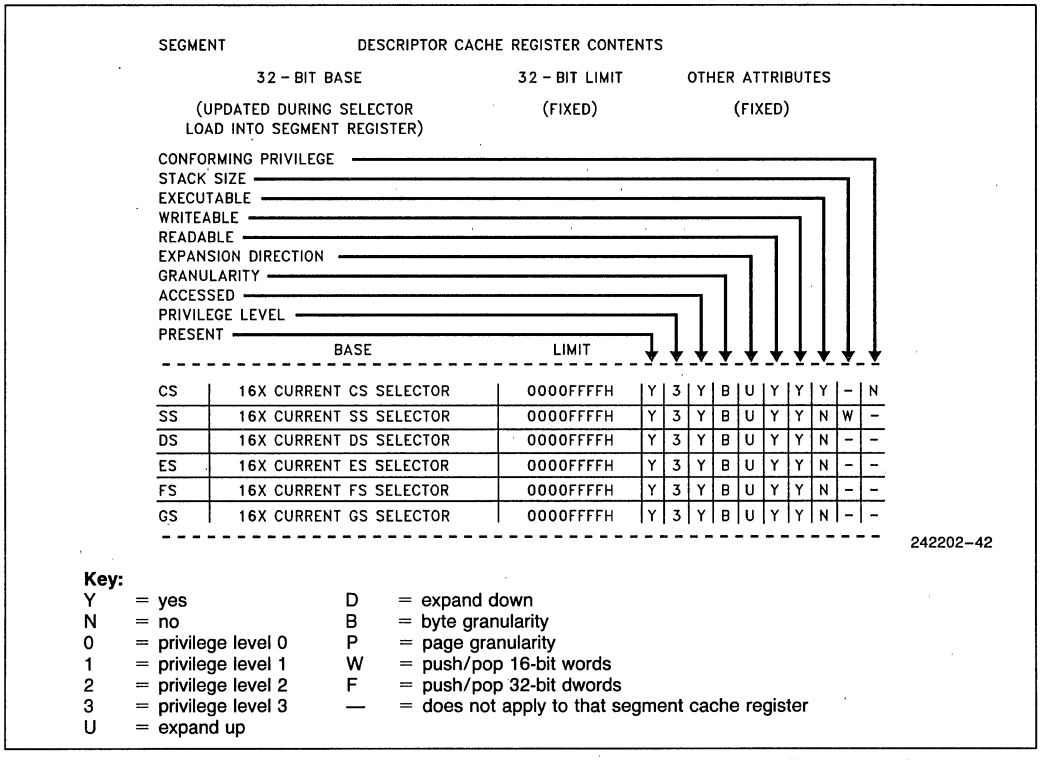
When operating in Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 6-11. In Protected Mode, each of these fields are defined according to the contents of the segment descriptor indexed by the selector value loaded into the segment register.

When operating in a Virtual 8086 Mode within the Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 6-12. For compatibility with the 8086 architecture, the base is set to sixteen times the current selector value, the limit is fixed at 0000FFFFH, and the attributes are fixed so as to indicate the segment is present and fully usable. The virtual program executes at lowest privilege level, level 3, to allow trapping of all IOPL-sensitive instructions and level-0-only instructions.



Key:
Y = fixed yes
N = fixed no
d = per segment descriptor
p = per segment descriptor; descriptor must indicate "present" to avoid exception 11 (exception 12 in case of SS)
r = per segment descriptor, but descriptor must indicate "readable" to avoid exception 13 (special case for SS)
w = per segment descriptor, but descriptor must indicate "writeable" to avoid exception 13 (special case for SS)
— = does not apply to that segment cache register

Figure 6-11. Segment Descriptor Caches for Protected Mode (Loaded per Descriptor)



Key:

Y	= yes	D	= expand down
N	= no	B	= byte granularity
0	= privilege level 0	P	= page granularity
1	= privilege level 1	W	= push/pop 16-bit words
2	= privilege level 2	F	= push/pop 32-bit dwords
3	= privilege level 3	—	= does not apply to that segment cache register
U	= expand up		

Figure 6-12. Segment Descriptor Caches for Virtual 8086 Mode within Protected Mode (Segment Limit and Attributes are Fixed)

6.3 Protection

6.3.1 PROTECTION CONCEPTS

The Intel486 processor has four levels of protection which are optimized to support the needs of a multitasking operating system to isolate and protect user programs from each other and the operating system. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. Unlike traditional processor-based systems where this protection is achieved only through the use of complex external hardware and software the Intel486 processor provides the protection as part of its integrated Memory Management Unit. The Intel486 processor offers an additional type of protection on a page basis, when paging is enabled. (See section 6.4.3, "Page Level Protection.")

The four-level hierarchical privilege system is illustrated in Figure 6-13. It is an extension of the user/supervisor privilege mode commonly used by minicomputers and, in fact, the user/supervisor mode is fully supported by the Intel486 processor paging mechanism. The privilege levels (PL) are numbered 0 through 3. Level 0 is the most privileged or trusted level.

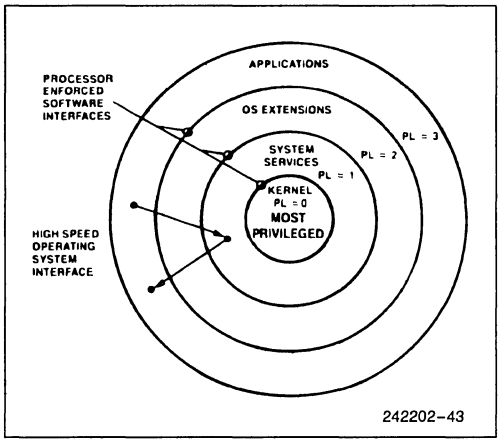


Figure 6-13. Four-Level Hierarchical Protection

6.3.2 RULES OF PRIVILEGE

The Intel486 processor controls access to both data and procedures between levels of a task, according to the following rules.

- Data stored in a segment with privilege level **p** can be accessed only by code executing at a privilege level at least as privileged as **p**.
- A code segment/procedure with privilege level **p** can only be called by a task executing at the same or a lesser privilege level than **p**.

6.3.3 PRIVILEGE LEVELS

6.3.3.1 Task Privilege

At any point in time, a task on the Intel486 processor always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies the task's privilege level. A task's CPL may only be changed by control transfers through gate descriptors to a code segment with a different privilege level. (See section 6.3.4, "Privilege Level Transfers.") Thus, an application program running at PL = 3 may call an operating system routine at PL = 1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

6.3.3.2 Selector Privilege (RPL)

The privilege level of a selector is specified by the RPL field. The RPL is the two least significant bits of the selector. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (i.e., numerically larger) level of a task's CPL and a selector's RPL. Thus, if selector's RPL = 0 then the CPL always specifies the privilege level for making an access using the selector. On the other hand if RPL = 3 then a selector can only access segments at level 3 regardless of the task's CPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Because the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

6.3.3.3 I/O Privilege and I/O Permission Bitmap

The I/O privilege level (IOPL, a 2-bit field in the EFLAG register) defines the least privileged level at which I/O instructions can be unconditionally performed. I/O instructions can be unconditionally performed when CPL ≥ IOPL. (The I/O instructions are IN, OUT, INS, OUTS, REP INS, and REP OUTS.) When CPL > IOPL, and the current task is associated with a 286 TSS, attempted I/O instructions cause an exception 13 fault. When CPL > IOPL, and the current task is associated with an Intel486 processor TSS, the I/O Permission Bitmap (part of an Intel486 processor TSS) is consulted on whether I/O to the port is allowed, or an exception 13 fault is to be generated instead. For diagrams of the I/O Permission Bitmap, refer to Figure 6-14 and Figure 6-15. For further information on how the I/O Permission Bitmap is used in Protected Mode or in Virtual 8086 Mode, refer to section 6.5.4, "Protection and I/O Permission Bitmap."

The I/O privilege level (IOPL) also affects whether several other instructions can be executed or cause an exception 13 fault instead. These instructions are called "IOPL-sensitive" instructions and they are CLI and STI. (Note that the LOCK prefix is *not* IOPL-sensitive on the Intel486 processor.)

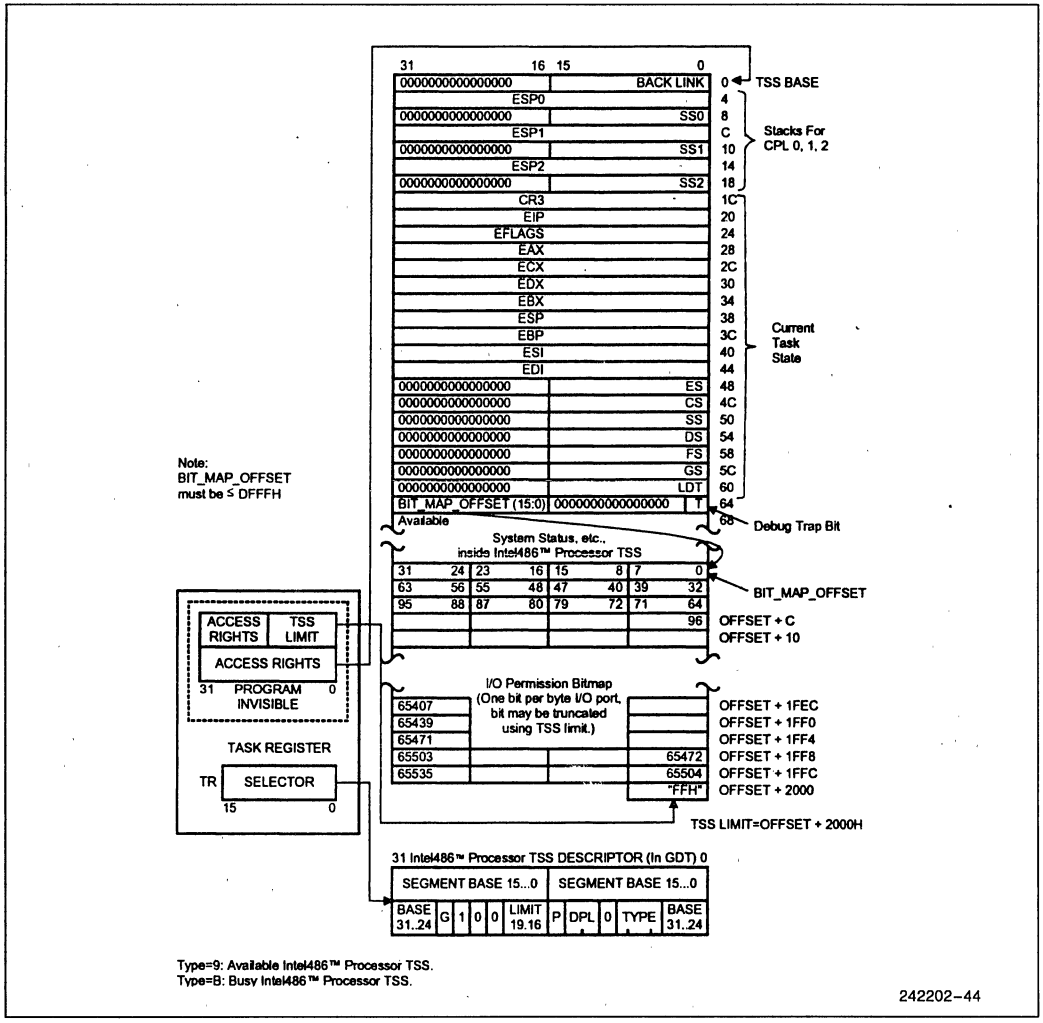


Figure 6-14. Intel486™ Processor TSS and TSS Registers

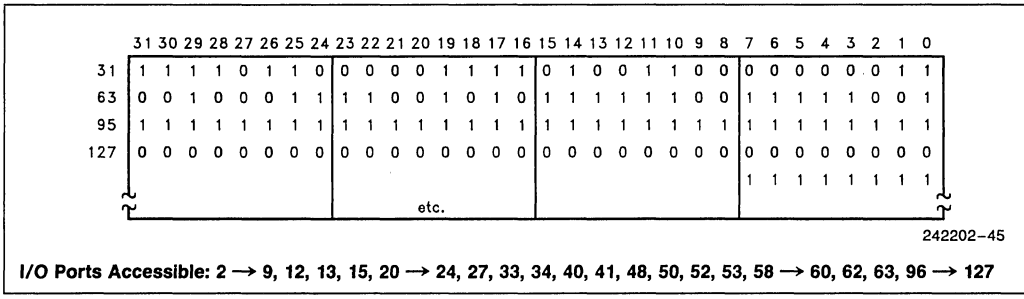


Figure 6-15. Sample I/O Permission Bit Map

The IOPL also affects whether the IF (interrupts enable flag) bit can be changed by loading a value into the EFLAGS register. When CPL ≥ IOPL, then the IF bit can be changed by loading a new value into the EFLAGS register. When CPL > IOPL, the IF bit cannot be changed by a new value POPed into (or otherwise loaded into) the EFLAGS register; the IF bit merely remains unchanged and no exception is generated.

6.3.3.4 Privilege Validation

The Intel486 processor provides several instructions to speed pointer testing and help maintain system integrity by verifying that the selector value refers to an appropriate segment. Table 6-2 summarizes the selector validation procedures available for the Intel486 processor.

This pointer verification prevents the common problem of an application at PL = 3 calling a operating systems routine at PL = 0 and passing the operating system routine a "bad" pointer which corrupts a data structure belonging to the operating system. If the operating system routine uses the ARPL instruction to ensure that the RPL of the selector has no greater privilege than that of the caller, then this problem can be avoided.

6.3.3.5 Descriptor Access

There are basically two types of segment accesses: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL as described above.

Table 6-2. Pointer Test Instructions

Instruction	Operands	Function
ARPL	Selector, Register	Adjust Requested Privilege Level: adjusts the RPL of the selector to the numeric maximum of current selector RPL value and the RPL value in the register. Set zero flag if selector RPL was changed.
VERR	Selector	VERify for Read: sets the zero flag if the segment referred to by the selector can be read.
VERW	Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL	Register, Selector	Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

Any time an instruction loads data segment registers (DS, ES, FS, GS), the Intel486 processor makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segments or readable code segments. The data access rules are specified in section 6.3.2, "Rules of Privilege." The only exception to those rules is readable conforming code segments which can be accessed at any privilege level.

Finally, the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL an exception 13 (general protection fault) is generated.

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL. All other descriptor types or a privilege-level violation will cause exception 13. A stack not-present fault causes exception 12. Note that an exception 11 is used for a not-present code or data segment.

6.3.4 PRIVILEGE LEVEL TRANSFERS

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call or a jump to another routine. There are five types of control transfers which are summarized in Table 6-3. Many of these transfers result in a privilege-level transfer. Changing privilege levels is done only via control transfers, by using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an exception 13 (e.g., JMP through a call gate, or IRET from a normal subroutine call).

In order to provide further system security, all control transfers are also subject to the privilege rules.

Table 6-3. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level	CALL	Call Gate	GDT/LDT
Interrupt within task may change CPL	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET(1)	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET(2) Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

NOTES:
1. NT (Nested Task bit of flag register) = 0
2. NT (Nested Task bit of flag register) = 1

The privilege rules require that:

- Privilege-level transitions can only occur via gates.
- JMPs can be made to a non-conforming code segment with the same privilege or to a conforming code segment with greater or equal privilege.
- CALLs can be made to a non-conforming code segment with the same privilege or via a gate to a more privileged level.
- Interrupts handled within the task obey the same privilege rules as CALLs.
- Conforming Code segments are accessible by privilege levels which are the same or less privileged than the conforming-code segment's DPL.
- Both the requested privilege level (RPL) in the selector pointing to the gate and the task's CPL must be of equal or greater privilege than the gate's DPL.
- The code segment selected in the gate must be the same or more privileged than the task's CPL.
- Return instructions that do not switch tasks can only return control to a code segment with same or less privilege.
- Task switches can be performed by a CALL, JMP, or INT which references either a task gate or task state segment whose DPL is less privileged or the same privilege as the old task's CPL.

Any control transfer that changes CPL within a task causes a change of stacks as a result of the privilege-level change. The initial values of SS:ESP for privilege levels 0, 1, and 2 are retained in the task state segment. (See section 6.3.6, "Task Switching.") During a JMP or CALL control transfer, the new stack pointer is loaded into the SS and ESP registers and the previous stack pointer is pushed onto the new stack.

When RETurning to the original privilege level, use of the lower privileged stack is restored as part of the RET or IRET instruction operation. For subroutine calls that pass parameters on the stack and cross privilege levels, a fixed number of words (as specified in the gate's word count field) are copied from the previous stack to the current stack. The inter-segment RET instruction with a stack adjustment value will correctly restore the previous stack pointer upon return.

6.3.5 CALL GATES

Gates provide protected, indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Because the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures (such as those which allocate memory, or perform I/O).

Gate descriptors follow the data access rules of privilege; that is, gates can be accessed by a task if the EPL is equal to or more privileged than the gate descriptor's DPL. Gates follow the control transfer rules of privilege and therefore may only transfer control to a more privileged level.

Call Gates are accessed via a CALL instruction and are syntactically identical to calling a normal subroutine. When an inter-level Intel486 processor call gate is activated, the following actions occur.

1. Load CS:EIP from gate check for validity
2. SS is pushed zero-extended to 32 bits
3. ESP is pushed
4. Copy Word Count 32-bit parameters from the old stack to the new stack
5. Push Return address on stack

The procedure is identical for 80286 Call gates, except that 16-bit parameters are copied and 16-bit registers are pushed.

Interrupt Gates and Trap gates work in a similar fashion as the call gates, except there is no copying of parameters. The only difference between Trap and Interrupt gates is that control transfers through an Interrupt gate disable further interrupts (i.e., the IF bit is set to 0), and Trap gates leave the interrupt status unchanged.

6.3.6 TASK SWITCHING

A very important attribute of any multitasking/multiuser operating system is its ability to rapidly switch between tasks or processes. The Intel486 processor directly supports this operation by providing a task switch instruction in hardware. The Intel486 processor task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences

execution in the new task, in about 10 microseconds. Like transfer of control via gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap, or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot.

The TSS descriptor points to a segment (see Figure 6-14) containing the entire Intel486 processor execution state while a task gate descriptor contains a TSS selector. The Intel486 processor supports both 80286 and Intel486 processor style TSSs. Figure 6-16 shows an 80286 TSS. The limit of an Intel486 processor TSS must be greater than 0064H (002BH for an 80286 TSS), and can be as large as 4 Gbytes. In the additional TSS space, the operating system is free to store additional information such as the reason the task is inactive, time the task has spent running, and open files belong to the task.

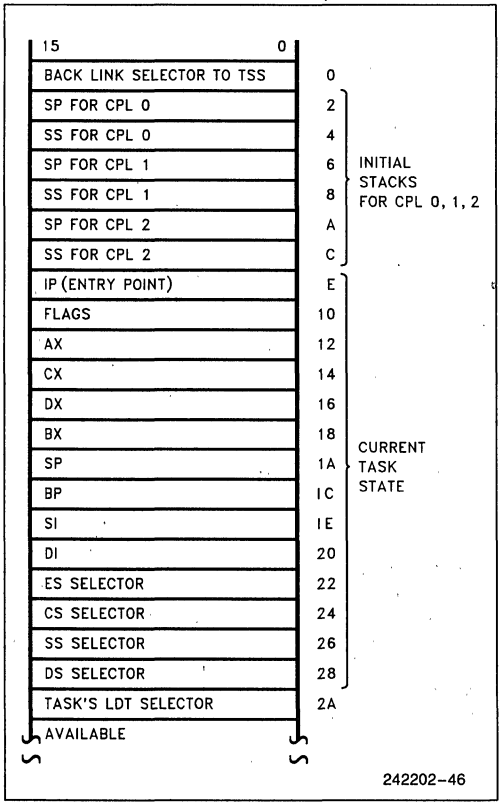


Figure 6-16. 80286 TSS

Each task must have a TSS associated with it. The current TSS is identified by a special register in the Intel486 processor called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TR are loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The current executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and machine status word (CR0) give information about the state of a task which are useful to the operating system. The Nested Task (NT) (bit 14 in EFLAGS) controls the function of the IRET instruction. If NT = 0, the IRET instruction performs the regular return; when NT = 1, IRET performs a task switch operation back to the previous task. The NT bit is set or reset in the following fashion:

When a CALL or INT instruction initiates a task switch, the new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT. (The NT bit will be restored after execution of the interrupt handler) NT may also be set or cleared by POPF or IRET instructions.

The Intel486 processor task state segment is marked busy by changing the descriptor type field from TYPE 9H to TYPE BH. An 80286 TSS is marked busy by changing the descriptor type field from TYPE 1 to TYPE 3. Use of a selector that references a busy task state segment causes an exception 13.

The Virtual Mode (VM) bit 17 is used to indicate if a task, is a virtual 8086 task. If VM = 1, then the tasks will use the Real Mode addressing mechanism. The virtual 8086 environment is only entered and exited via a task switch. (See section 6.5, "Virtual 8086 Environment.")

The **T** bit in the Intel486 processor TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1, upon entry to a new task, a debug exception 1 will be generated.

6.3.6.1 Floating-Point Task Switching

The FPU's state is not automatically saved when a task switch occurs, because the incoming task may not use the FPU. The Task Switched (TS) Bit (bit 3 in the CR0) helps deal with the FPU's state in a multitasking environment. Whenever the Intel OverDrive processors switch tasks, they set the TS bit. The Intel OverDrive processors detect the first use of a processor extension instruction after a task switch and cause the processor extension not available exception 7. The exception handler for exception 7 may then decide whether to save the state of the FPU. A processor extension not present exception (7) will occur when attempting to execute a Floating-Point or WAIT instruction if the Task Switched and Monitor coprocessor extension bits are both set (i.e., TS = 1 and MP = 1).

6.3.7 INITIALIZATION AND TRANSITION TO PROTECTED MODE

Because the Intel486 processor begins executing in Real Mode immediately after RESET, it is necessary to initialize the system tables and registers with the appropriate values.

The GDT and IDT registers must refer to a valid GDT and IDT. The IDT should be at least 256 bytes long, and GDT must contain descriptors for the initial code and data segments. Figure 6-17 shows the tables and Figure 6-18 shows the descriptors needed for a simple Protected Mode Intel486 processor system. It has a single code and single data/stack segment each 4 Gbytes long and a single privilege level PL = 0.

The actual method of enabling Protected Mode is to load CR0 with the PE bit set, via the MOV CR0, R/M instruction. This puts the Intel486 processor in Protected Mode.

After enabling Protected Mode, the next instruction should execute an intersegment JMP to load the CS register and flush the instruction decode queue. The final step is to load all of the data segment registers with the initial selector values.

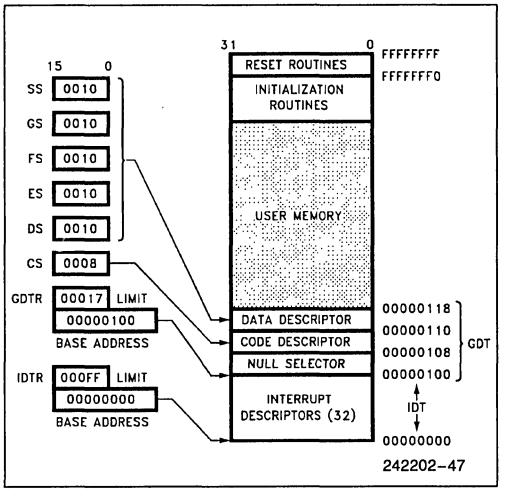


Figure 6-17. Simple Protected System

An alternate approach to entering Protected Mode, which is especially appropriate for multitasking operating systems, is to use the built-in task switch to load all of the registers. In this case the GDT would contain two TSS descriptors in addition to the code and data descriptors needed for the first task. The first JMP instruction in Protected Mode would jump to the TSS causing a task switch and loading all of the registers with the values stored in the TSS. Because a task switch saves the state of the current task in a task state segment, the TR should be initialized to point to a valid TSS descriptor.

6.4 Paging

6.4.1 PAGING CONCEPTS

Paging is another type of memory management useful for virtual memory multitasking operating systems. Unlike segmentation which modularizes programs and data into variable length segments, paging divides programs into multiple uniform size pages. Pages bear no direct relation to the logical structure of a program. While segment selectors can be considered the logical "name" of a program module or data structure, a page most likely corresponds to only a portion of a module or data structure.

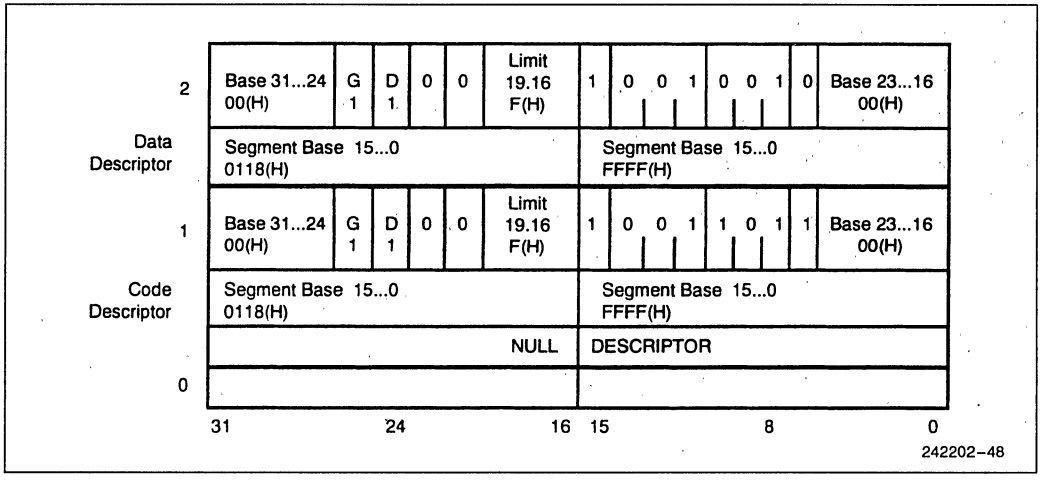


Figure 6-18. GDT Descriptors for Simple System

By taking advantage of the locality of reference displayed by most programs, only a small number of pages from each active task need be in memory at any one moment.

6.4.2 PAGING ORGANIZATION

6.4.2.1 Page Mechanism

The Intel486 processor uses two levels of tables to translate the linear address (from the segmentation unit) into a physical address. There are three components to the paging mechanism of the Intel486 processor: the page directory, the page tables, and the page itself (page frame). All memory-resident elements of the Intel486 processor paging mechanism are the same size, namely, 4 Kbytes. A uniform size for all of the elements simplifies memory allocation and reallocation schemes, because there is no problem with memory fragmentation. Figure 6-19 shows how the paging mechanism works.

6.4.2.2 Page Descriptor Base Register

CR2 is the Page Fault Linear Address register. It holds the 32-bit linear address which caused the last page fault detected.

CR3 is the Page Directory Physical Base Address Register. It contains the physical starting address of the Page Directory. The lower 12 bits of CR3 are always zero to ensure that the Page Directory is always page aligned. Loading it via a MOV CR3 reg instruction causes the Page Table Entry cache to be flushed, as will a task switch through a TSS that **changes** the value of CR0. (See section 6.4.5, "Translation Lookaside Buffer.")

6.4.2.3 Page Directory

The Page Directory is 4-Kbytes long and allows up to 1024 Page Directory Entries. Each Page Directory Entry contains the address of the next level of tables, the Page Tables and information about the page table. The contents of a Page Directory Entry are shown in Figure 6-20. The upper 10 bits of the linear address (A22–A31) are used as an index to select the correct Page Directory Entry.

6.4.2.4 Page Tables

Each Page Table is 4 Kbytes and holds up to 1024 Page Table Entries. Page Table Entries contain the starting address of the page frame and statistical information about the page. (See Figure 6-21.) Address bits A12–A21 are used as an index to select one of the 1024 Page Table Entries. The 20 upper-bit page frame address is concatenated with the lower 12 bits of the linear address to form the physical address. Page tables can be shared between tasks and swapped to disks.

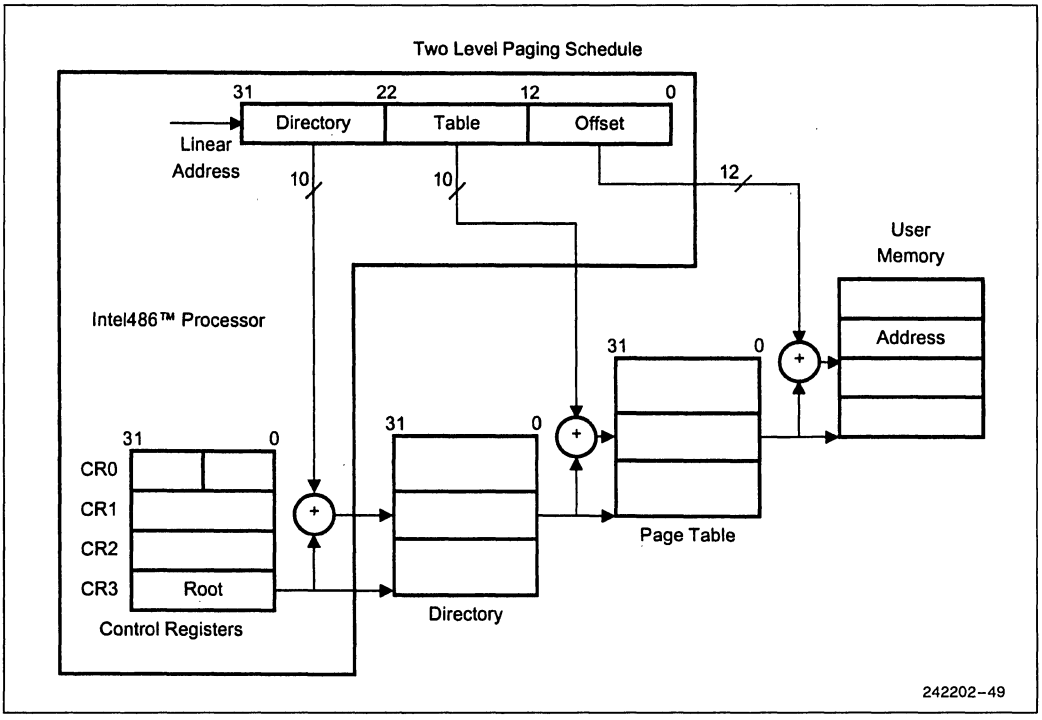


Figure 6-19. Paging Mechanism

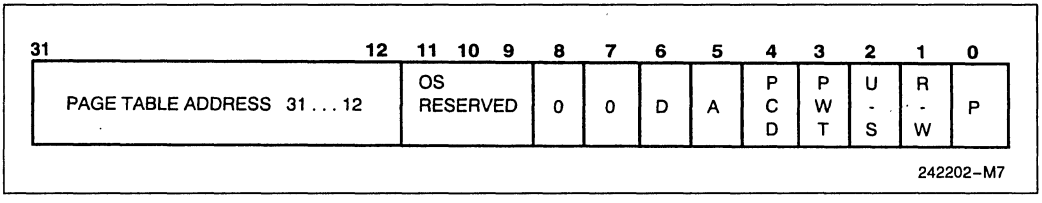


Figure 6-20. Page Directory Entry (Points to Page Table)

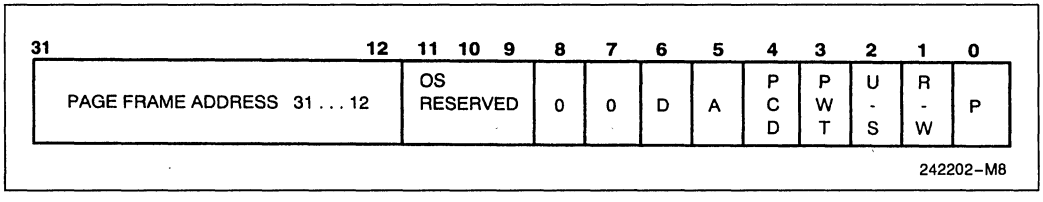


Figure 6-21. Page Table Entry (Points to Page)

6.4.2.5 Page Directory/Table Entries

The lower 12 bits of the Page Table Entries and Page Directory Entries contain statistical information about pages and page tables, respectively. The **P** (Present) bit 0 indicates if a Page Directory or Page Table entry can be used in address translation. If P = 1 the entry can be used for address translation if P = 0 the entry cannot be used for translation, and all of the other bits are available for use by the software. For example, the remaining 31 bits could be used to indicate where on the disk the page is stored.

The **A** (Accessed) bit 5 is set by the Intel486 processor for both types of entries before a read or write access occurs to an address covered by the entry. The **D** (Dirty) bit 6 is set to 1 before a write to an address covered by that page table entry occurs. The D bit is undefined for Page Directory Entries. When the P, A and D bits are updated by the Intel486 processor, a Read-Modify-Write cycle is generated which locks the bus and prevents conflicts with other processors or peripherals. Software which modifies these bits should use the LOCK prefix to ensure the integrity of the page tables in multimaster systems.

The 3 bits marked **OS Reserved** in Figure 6-20 and Figure 6-21 (bits 9–11) are software definable. OSs are free to use these bits for whatever purpose they wish. An example use of the **OS Reserved** bits would be to store information about page aging. By keeping track of how long a page has been in memory since being accessed, an operating system can implement a page replacement algorithm such as Least Recently Used.

The (User/Supervisor) U/S bit 2 and the (Read/Write) R/W bit 1 are used to provide protection attributes for individual pages.

6.4.3 PAGE LEVEL PROTECTION (R/W, U/S BITS)

The Intel486 processor provides a set of protection attributes for paging systems. The paging mechanism distinguishes between two levels of protection: User which corresponds to level 3 of the segmentation based protection, and supervisor which encompasses all of the other protection levels (0, 1, 2).

The R/W and U/S bits are used in conjunction with the WP bit in the flags register (EFLAGS). The Intel386 processor does not contain the WP bit. The WP bit has been added to the Intel486 processor to protect read-only pages from supervisor write accesses. The Intel386 processor allows a read-only page to be written from protection levels 0, 1 or 2. WP = 0 is the Intel386 processor-compatible mode. When WP = 0, the supervisor can write to a read-only page as defined by the U/S and R/W bits. When WP = 1, supervisor access to a read-only page (R/W = 0) will cause a page fault (exception 14).

Table 6-4 shows the effect of the WP, U/S and R/W bits on accessing memory. When WP = 0, the supervisor can write to pages regardless of the state of the R/W bit. When WP = 1 and R/W = 0, the supervisor cannot write to a read-only page. A user attempt to access a supervisor only page (U/S = 0) or to write to a read-only page will cause a page fault (exception 14).

Table 6-4. Page Level Protection Attributes

U/S	R/W	WP	User Access	Supervisor Access
0	0	0	None	Read/Write/Execute
0	1	0	None	Read/Write/Execute
1	0	0	Read/Execute	Read/Write/Execute
1	1	0	Read/Write/Execute	Read/Write/Execute
0	0	1	None	Read/Execute
0	1	1	None	Read/Write/Execute
1	0	1	Read/Execute	Read/Execute
1	1	1	Read/Write/Execute	Read/Write/Execute

The R/W and U/S bits provide protection from user access on a page-by-page basis because the bits are contained in the Page Table Entry and the Page Directory Table. The U/S and R/W bits in the first-level Page Directory Table apply to all entries in the page table pointed to by that directory entry. The U/S and R/W bits in the second-level Page Table Entry apply only to the page described by that entry. The most restrictive of the U/S and R/W bits from the Page Directory Table and the Page Table Entry are used to address a page.

Example: If the U/S and R/W bits for the Page Directory entry were 10 (user read/execute) and the U/S and R/W bits for the Page Table Entry were 01 (no user access at all), the access rights for the page would be 01, the numerically smaller of the two.

Note that a given segment can be easily made read-only for level 0, 1 or 2 via use of segmented protection mechanisms. (Section 6.3, "Protection".)

6.4.4 PAGE CACHEABILITY (PWT AND PCD BITS)

See section 7.6, "Page Cacheability," for a detailed description of page cacheability and the PWT and PCD bits.

6.4.5 TRANSLATION LOOKASIDE BUFFER

The Intel486 processor paging hardware is designed to support demand paged virtual memory systems. However, performance would degrade substantially if the Intel486 processor was required to access two levels of tables for every memory reference. To solve this problem, the Intel486 processor keeps a cache of the most recently accessed pages. This cache is called the Translation Lookaside Buffer (TLB). The TLB is a four-way set associative 32-entry page table cache. It automatically keeps the most commonly used Page Table Entries in the Intel486 processor. The 32-entry TLB coupled with a 4K page size, results in coverage of 128 Kbytes of memory addresses. For many common multitasking systems, the TLB will have a hit rate of about 98%. This means that the Intel486 processor will only have to access the two-level page structure on 2% of all memory references. Figure 6-22 illustrates how the TLB complements the Intel486 processor's paging mechanism.

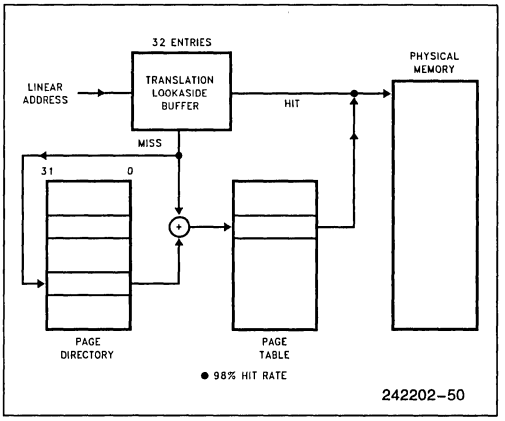


Figure 6-22. Translation Lookaside Buffer

Reading a new entry into the TLB (TLB refresh) is a two-step process handled by the Intel486 processor hardware. The sequence of data cycles to perform a TLB refresh is the following:

1. Read the correct Page Directory Entry, as pointed to by the page base register and the upper 10 bits of the linear address. The page base register is in control register 3.
   a. Optionally perform a locked read/write to set the accessed bit in the directory entry. The directory entry will actually get read twice if the Intel486 processor needs to set any of the bits in the entry. If the page directory entry changes between the first and second reads, the data returned for the second read will be used.
2. Read the correct entry in the Page Table and place the entry in the TLB.
   a. Optionally perform a locked read/write to set the accessed and/or dirty bit in the page table entry. Again, note that the page table entry will actually get read twice if the Intel486 processor needs to set any of the bits in the entry. Like the directory entry, if the data changes between the first and second read the data returned for the second read will be used.

Note that the directory entry must always be read into the Intel486 processor, because directory entries are never placed in the paging TLB. Page faults can be signaled from either the page directory read or the page table read. Page directory and page table entries may be placed in the Intel486 processor on-chip cache just like normal data.

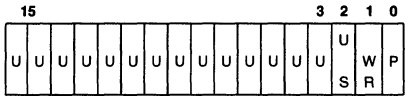
6.4.6 PAGING OPERATION

The paging hardware operates in the following fashion. The paging unit hardware receives a 32-bit linear address from the segmentation unit. The upper 20 linear address bits are compared with all 32 entries in the TLB to determine if there is a match. If there is a match (i.e., a TLB hit), then the 32-bit physical address is calculated and will be placed on the address bus.

However, if the page table entry is not in the TLB, the Intel486 processor will read the appropriate Page Directory Entry. If P = 1 on the Page Directory Entry indicating that the page table is in memory, then the Intel486 processor will read the appropriate Page Table Entry and set the Access bit. If P = 1 on the Page Table Entry indicating that the page is in memory, the Intel486 processor will update the Access and Dirty bits as needed and fetch the operand. The upper 20 bits of the linear address, read from the page table, will be stored in the TLB for future accesses. However, if P = 0 for either the Page Directory Entry or the Page Table Entry, then the Intel486 processor will generate a page fault, an Exception 14.

The Intel486 processor will also generate an exception 14 page fault if the memory reference violated the page protection attributes (i.e., U/S or R/W) (e.g., trying to write to a read-only page). CR2 will hold the linear address which caused the page fault. If a second page fault occurs, while the Intel486 processor is attempting to enter the service routine for the first, then the Intel486 processor will invoke the page fault (exception 14) handler a second time, rather than the double fault (exception 8) handler. Because Exception 14 is classified as a fault, CS:EIP will point to the instruction causing the page fault. The 16-bit error code pushed as part of the page fault handler will contain status bits which indicate the cause of the page fault.

The 16-bit error code is used by the operating system to determine how to handle the page fault. The upper portion of Figure 6-23 shows the format of the page-fault error code and the interpretation of the bits.



U/S	W/R	Access Type
0	0	Supervisor* Read
0	1	Supervisor Write
1	0	User Read
1	1	User Write

NOTES:
*Descriptor table access will fault with U/S = 0, even if the program is executing at level 3.

U: UNDEFINED

U/S: The U/S bit indicates whether the access causing the fault occurred when the Intel486 processor was executing in User Mode (U/S = 1) or in Supervisor mode (U/S = 0).

W/R: The W/R bit indicates whether the access causing the fault was a Read (W/R = 0) or a Write (W/R = 1).

P: The P bit indicates whether a page fault was caused by a not-present page (P = 0), or by a page level protection violation (P = 1).

Figure 6-23. Page Fault System Information

NOTE:
Even though the bits in the error code (U/S, W/R, and P) have similar names as the bits in the Page Directory/Table Entries, the interpretation of the error code bits is different. Figure 6-23 indicates what type of access caused the page fault.

6.4.7 OPERATING SYSTEM RESPONSIBILITIES

The Intel486 processor takes care of the page address translation process, relieving the burden from an operating system in a demand-paged system. The operating system is responsible for setting

up the initial page tables, and handling any page faults. The operating system also is required to invalidate (i.e., flush) the TLB when any changes are made to any of the page table entries. The operating system must reload CR3 to cause the TLB to be flushed.

Setting up the tables is simply a matter of loading CR3 with the address of the Page Directory, and allocating space for the Page Directory and the Page Tables. The primary responsibility of the operating system is to implement a swapping policy and handle all of the page faults.

A final concern of the operating system is to ensure that the TLB cache matches the information in the paging tables. In particular, any time the operating system sets the P present bit of page table entry to zero, the TLB must be flushed. Operating systems may want to take advantage of the fact that CR3 is stored as part of a TSS, to give every task or group of tasks its own set of page tables.

6.5 Virtual 8086 Environment

6.5.1 EXECUTING 8086 PROGRAMS

The Intel486 processor allows the execution of 8086 application programs in both Real Mode and in the Virtual 8086 Mode (Virtual Mode). Of the two methods, Virtual 8086 Mode offers the system designer the most flexibility. The Virtual 8086 Mode allows the execution of 8086 applications, while still allowing the system designer to take full advantage of the Intel486 processor protection mechanism. In particular, the Intel486 processor allows the simultaneous execution of 8086 operating systems and its applications, and an Intel486 processor operating system and both 80286 and Intel486 processor applications. Thus, in a multi-user Intel486 processor computer, one person could be running an MS-DOS* spreadsheet, another person using MS-DOS*, and a third person could be running multiple UNIX utilities and applications. Each person in this scenario would believe that he had the computer completely to himself. Figure 6-24 illustrates this concept.

6.5.2 VIRTUAL 8086 MODE ADDRESSING MECHANISM

One of the major differences between Intel486 processor Real and Protected modes is how the segment selectors are interpreted. When the Intel486 processor is executing in Virtual 8086 Mode the segment registers are used in an identical fashion to Real Mode. The contents of the segment register is shifted left 4 bits and added to the offset to form the segment base linear address.

The Intel486 processor allows the operating system to specify which programs use the 8086 style address mechanism, and which programs use Protected Mode addressing, on a per task basis. Through the use of paging, the one megabyte address space of the Virtual Mode task can be mapped to anywhere in the 4-Gbyte linear address space of the Intel486 processor. Like Real Mode, Virtual Mode effective addresses (i.e., segment offsets) that exceed 64 Kbyte will cause an exception 13. However, these restrictions should not prove to be important, because most tasks running in Virtual 8086 Mode will simply be existing 8086 application programs.

6.5.3 PAGING IN VIRTUAL MODE

The paging hardware allows the concurrent running of multiple Virtual Mode tasks, and provides protection and operating system isolation. Although it is not strictly necessary to have the paging hardware enabled to run Virtual Mode tasks, it is needed in order to run multiple Virtual Mode tasks or to relocate the address space of a Virtual Mode task to physical address space greater than one Mbyte.

The paging hardware allows the 20-bit linear address produced by a Virtual Mode program to be divided into up to 256 pages. Each one of the pages can be located anywhere within the maximum 4-Gbyte physical address space of the Intel486 processor. In addition, because CR3 (the Page Directory Base Register) is loaded by a task switch, each Virtual Mode task can use a different mapping scheme to map pages to different physical locations. Finally, the paging hardware allows the sharing of the 8086 operating system code between multiple 8086 applications. Figure 6-24 shows how the Intel486 processor paging hardware enables multiple 8086 programs to run under a virtual memory demand paged system.

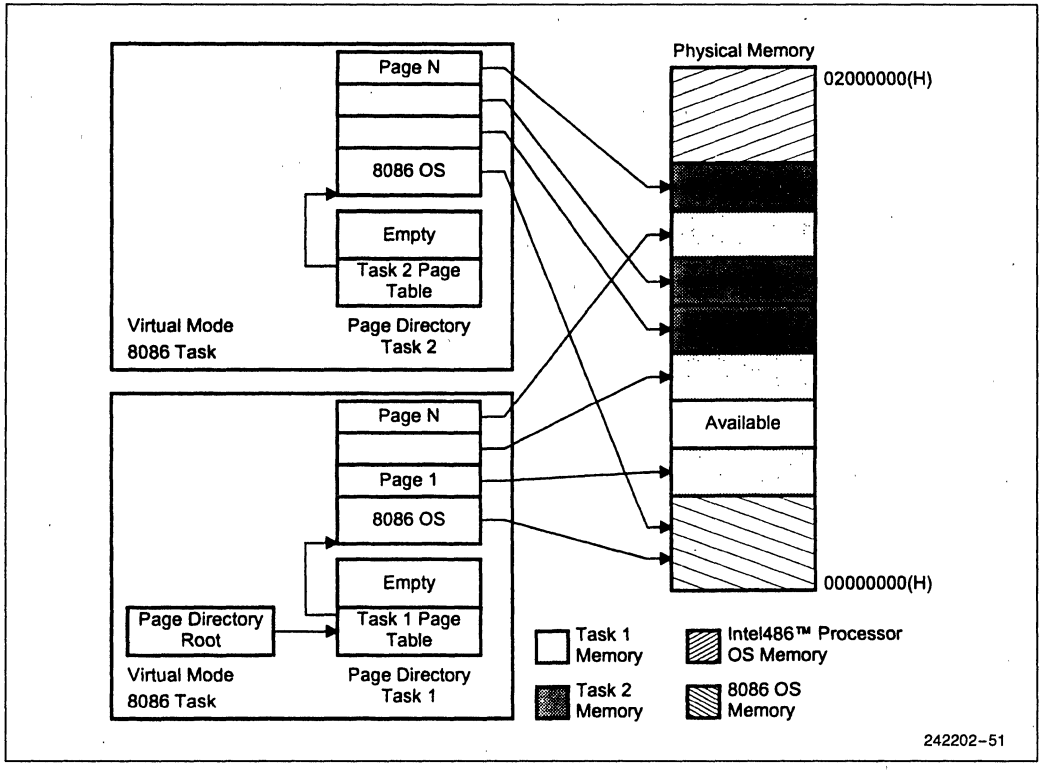


Figure 6-24. Virtual 8086 Environment Memory Management

6.5.4 PROTECTION AND I/O PERMISSION BITMAP

All Virtual 8086 Mode programs execute at privilege level 3, the level of least privilege. As such, Virtual 8086 Mode programs are subject to all of the protection checks defined in Protected Mode. (This is different from Real Mode which implicitly is executing at privilege level 0, the level of greatest privilege.) Thus, an attempt to execute a privileged instruction when in Virtual 8086 Mode will cause an exception 13 fault.

The following are privileged instructions, which may be executed only at Privilege Level 0. Therefore, attempting to execute these instructions in Virtual 8086 Mode (or anytime CPL > 0) causes an exception 13 fault:

```
LIDT; MOV DRn,reg; MOV reg,DRn;
LGDT; MOV TRn,reg; MOV reg,TRn;
LMSW; MOV CRn,reg; MOV reg,CRn.
CLTS;
HLT;
```

Several instructions, particularly those applying to the multitasking model and protection model, are available only in Protected Mode. Therefore, attempting to execute the following instructions in Real Mode or in Virtual 8086 Mode generates an exception 6 fault:

```
LTR;      STR;
LLDT;     SLDT;
LAR;      VERR;
LSL;      VERW;
ARPL.
```

The instructions that are IOPL-sensitive in Protected Mode are:

```
IN;       STI;
OUT;      CLI
INS;
OUTS;
REP INS;
REP OUTS;
```

In Virtual 8086 Mode, a slightly different set of instructions are made IOPL-sensitive. The following instructions are IOPL-sensitive in Virtual 8086 Mode:

```
INT n;    STI;
PUSHF;    CLI;
POPF;     IRET
```

The PUSHF, POPF, and IRET instructions are IOPL-sensitive in Virtual 8086 Mode only. This provision allows the IF flag (interrupt enable flag) to be virtualized to the Virtual 8086 Mode program. The INT n software interrupt instruction is also IOPL-sensitive in Virtual 8086 Mode. Note, however, that the INT 3 (opcode 0CCH), INTO, and BOUND instructions are not IOPL-sensitive in Virtual 8086 mode (they aren't IOPL sensitive in Protected Mode either).

Note that the I/O instructions (IN, OUT, INS, OUTS, REP INS, and REP OUTS) are **not** IOPL-sensitive in Virtual 8086 mode. Rather, the I/O instructions become automatically sensitive to the **I/O Permission Bitmap** contained in the **Intel486 processor Task State Segment**. The I/O Permission Bitmap, automatically used by the Intel486 processor in Virtual 8086 Mode, is illustrated by Figure 6-14 and Figure 6-15.

The I/O Permission Bitmap can be viewed as a 0–64 Kbit string, which begins in memory at offset Bit_Map_Offset in the current TSS. Bit_Map_Offset must be ≤ DFFFH so the entire bit map and the byte FFH which follows the bit map are all at offsets ≤ FFFFH from the TSS base. The 16-bit pointer Bit_Map_Offset (15:0) is found in the word beginning at offset 66H (102 decimal) from the TSS base, as shown in Figure 6-14.

Each bit in the I/O Permission Bitmap corresponds to a single byte-wide I/O port, as illustrated in Figure 6-14. If a bit is 0, I/O to the corresponding byte-wide port can occur without generating an exception. Otherwise the I/O instruction causes an exception 13 fault. Because every byte-wide I/O port must be protectable, all bits corresponding to a word-wide or dword-wide port must be 0 for the word-wide or dword-wide I/O to be permitted. If all the referenced bits are 0, the I/O will be allowed. If any referenced bits are 1, the attempted I/O will cause an exception 13 fault.

Due to the use of a pointer to the base of the I/O Permission Bitmap, the bitmap may be located anywhere within the TSS, or may be ignored completely by pointing the Bit_Map_Offset (15:0) beyond the limit of the TSS segment. In the same manner, only a small portion of the 64K I/O space need have an associated map bit, by adjusting the TSS limit to truncate the bitmap. This eliminates the commitment of 8K of memory when a complete bitmap is not required, while allowing the fully general case if desired.

Example of Bitmap for I/O Ports 0–255: Setting the TSS limit to {bit_Map_Offset + 31 + 1**} [** see note below] will allow a 32-byte bitmap for the I/O ports #0–255, plus a terminator byte of all 1's [** see note below]. This allows the I/O bitmap to control I/O Permission to I/O port 0–255 while causing an exception 13 fault on attempted I/O to any I/O port 80256 through 65,565.

> ****IMPORTANT IMPLEMENTATION NOTE:** Beyond the last byte of I/O mapping information in the I/O Permission Bitmap **must** be a byte containing all 1's. The byte of all 1's must be within the limit of the Intel486 processor TSS segment (see Figure 6-14).

6.5.5 INTERRUPT HANDLING

In order to fully support the emulation of an 8086 machine, interrupts in Virtual 8086 Mode are handled in a unique fashion. When running in Virtual Mode all interrupts and exceptions involve a privilege change back to the host Intel486 processor operating system. The Intel486 processor operating system determines if the interrupt comes from a

Protected Mode application or from a Virtual Mode program by examining the VM bit in the EFLAGS image stored on the stack.

When a Virtual Mode program is interrupted and execution passes to the interrupt routine at level 0, the VM bit is cleared. However, the VM bit is still set in the EFLAG image on the stack.

The Intel486 processor operating system in turn handles the exception or interrupt and then returns control to the 8086 program. The Intel486 processor operating system may choose to let the 8086 operating system handle the interrupt or it may emulate the function of the interrupt handler. For example, many 8086 operating system calls are accessed by PUSHing parameters on the stack, and then executing an INT n instruction. If the IOPL is set to 0 then all INT n instructions will be intercepted by the Intel486 processor operating system. The Intel486 processor operating system could emulate the 8086 operating system's call. Figure 6-25 shows how the Intel486 processor operating system could intercept an 8086 operating system's call to "Open a File."

An Intel486 processor operating system can provide a Virtual 8086 Environment which is totally transparent to the application software via intercepting and then emulating 8086 operating system's calls, and intercepting IN and OUT instructions.

6.5.6 ENTERING AND LEAVING VIRTUAL 8086 MODE

Virtual 8086 mode is entered by executing an IRET instruction (at CPL = 0), or Task Switch (at any CPL) to an Intel486 processor task whose Intel486 processor TSS has a FLAGS image containing a 1 in the VM bit position while the Intel486 processor is executing in Protected Mode. That is, one way to enter Virtual 8086 mode is to switch to a task with an Intel486 processor TSS that has a 1 in the VM bit in the EFLAGS image. The other way is to execute a 32-bit IRET instruction at privilege level 0, where the stack has a 1 in the VM bit in the EFLAGS image. POPF does not affect the VM bit, even if the Intel486 processor is in Protected Mode or level 0, and so cannot be used to enter Virtual 8086 Mode. PUSHF always pushes a 0 in the VM bit, even if the Intel486 processor is in Virtual 8086 Mode, so that a program cannot tell if it is executing in REAL mode, or in Virtual 8086 mode.

The VM bit can be set by executing an IRET instruction only at privilege level 0, or by any instruction or Interrupt which causes a task switch in Protected Mode (with VM = 1 in the new FLAGS image), and can be cleared only by an interrupt or exception in Virtual 8086 Mode. IRET and POPF instructions executed in REAL mode or Virtual 8086 mode will not change the value in the VM bit.

The transition out of virtual 8086 mode to Intel486 processor protected mode occurs only on receipt of an interrupt or exception (such as due to a sensitive instruction). In Virtual 8086 mode, all interrupts and exceptions vector through the protected mode IDT, and enter an interrupt handler in protected Intel486 processor mode. That is, as part of interrupt processing, the VM bit is cleared.

Because the matching IRET must occur from level 0, if an Interrupt or Trap Gate is used to field an interrupt or exception out of Virtual 8086 mode, the Gate must perform an inter-level interrupt only to level 0. Interrupt or Trap Gates through conforming segments, or through segments with DPL > 0, will raise a GP fault with the CS selector as the error code.

6.5.6.1 Task Switches To and From Virtual 8086 Mode

Tasks which can execute in virtual 8086 mode must be described by a TSS with the new Intel486 processor format (TYPE 9 or 11 descriptor).

A task switch out of virtual 8086 mode will operate exactly the same as any other task switch out of a task with an Intel486 processor TSS. All of the programmer visible state, including the FLAGS register with the VM bit set to 1, is stored in the TSS.

The segment registers in the TSS will contain 8086 segment base values rather than selectors.

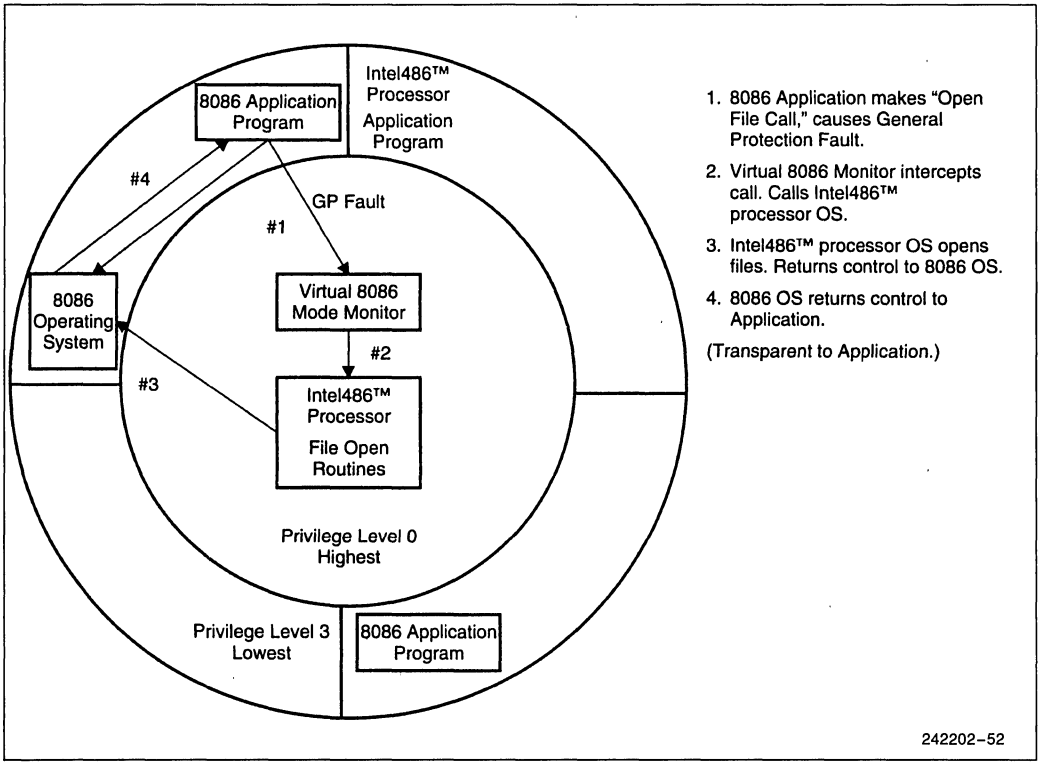


Figure 6-25. Virtual 8086 Environment Interrupt and Call Handling

A task switch into a task described by an Intel486 processor TSS will have an additional check to determine if the incoming task should be resumed in virtual 8086 mode. Tasks described by 80286 format TSSs cannot be resumed in virtual 8086 mode, so no check is required there (the FLAGS image in 80286 format TSS has only the low order 16 FLAGS bits). Before loading the segment register images from an Intel486 processor TSS, the FLAGS image is loaded, so that the segment registers are loaded from the TSS image as 8086 segment base values. The task is now ready to resume in virtual 8086 execution mode.

6.5.6.2 Transitions Through Trap and Interrupt Gates, and IRET

A task switch is one way to enter or exit virtual 8086 mode. The other method is to exit through a Trap or Interrupt gate, as part of handling an interrupt, and to enter as part of executing an IRET instruction. The transition out must use an Intel486 processor Trap Gate (Type 14), or Intel486 processor Interrupt Gate (Type 15), which must point to a non-conforming level 0 segment (DPL = 0) in order to permit the trap handler to IRET back to the Virtual 8086 program. The Gate must point to a non-conforming level 0 segment to perform a level switch to level 0 so that the matching IRET can change the VM bit. Intel486 processor gates must be used, because 80286 gates save only the low 16 bits of the FLAGS register, so that the VM bit will not be saved on transitions through the 80286 gates. Also, the 16-bit IRET (presumably) used to terminate the 80286 interrupt handler will pop only the lower 16 bits from FLAGS, and will not affect the VM bit. The action taken for an Intel486 processor Trap or Interrupt gate if an interrupt occurs while the task is executing in virtual 8086 mode is given by the following sequence.

1. Save the FLAGS register in a temp to push later. Turn off the VM and TF bits, and if the interrupt is serviced by an Interrupt Gate, turn off IF also.
2. Interrupt and Trap gates must perform a level switch from 3 (where the VM86 program executes) to level 0 (so IRET can return). This process involves a stack switch to the stack given in the TSS for privilege level 0. Save the Virtual 8086 Mode SS and ESP registers to push in a later step. The segment register load of SS will be done as a Protected Mode segment load, because the VM bit was turned off above.
3. Push the 8086 segment register values onto the new stack, in the order: GS, FS, DS, ES. These are pushed as 32-bit quantities, with undefined values in the upper 16 bits. Then load these 4 registers with null selectors (0).
4. Push the old 8086 stack pointer onto the new stack by pushing the SS register (as 32 bits, high bits undefined), then pushing the 32-bit ESP register saved above.
5. Push the 32-bit FLAGS register saved in step 1.
6. Push the old 8086 instruction pointer onto the new stack by pushing the CS register (as 32 bits, high bits undefined), then pushing the 32-bit EIP register.
7. Load up the new CS:EIP value from the interrupt gate, and begin execution of the interrupt routine in protected Intel486 processor mode.

The transition out of Virtual 8086 mode performs a level change and stack switch, in addition to changing back to protected mode. In addition, all of the 8086 segment register images are stored on the stack (behind the SS:ESP image), and then loaded with null (0) selectors before entering the interrupt handler. This will permit the handler to safely save and restore the DS, ES, FS, and GS registers as 80286 selectors. This is needed so that interrupt handlers which don't care about the mode of the interrupted program can use the same prolog and epilog code for state saving (i.e., push all registers in prolog, pop all in epilog) regardless of whether or not a "native" mode or Virtual 8086 mode program was interrupted. Restoring null selectors to these registers before executing the IRET will not cause a trap in the interrupt handler. Interrupt routines which expect values in the segment registers, or return values in segment registers, will have to obtain/return values from the 8086 register images pushed onto the new stack. They will need to know the mode of the interrupted program in order to know where to find/return segment registers, and also to know how to interpret segment register values.

The IRET instruction will perform the inverse of the above sequence. Only the extended Intel486 processor IRET instruction (operand size = 32) can be used, and must be executed at level 0 to change the VM bit to 1.

1. If the NT bit in the FLAGs register is on, an inter-task return is performed. The current state is stored in the current TSS, and the link field in the current TSS is used to locate the TSS for the interrupted task which is to be resumed. Otherwise, continue with the following sequence.
2. Read the FLAGS image from SS:8[ESP] into the FLAGS register. This will set VM to the value active in the interrupted routine.
3. Pop off the instruction pointer CS:EIP. EIP is popped first; then a 32-bit word is popped which contains the CS value in the lower 16 bits. If VM = 0, this CS load is done as a protected mode segment load. If VM = 1, this will be done as an 8086 segment load.
4. Increment the ESP register by 4 to bypass the FLAGS image which was "popped" in step 1.
5. If VM = 1, load segment registers ES, DS, FS, and GS from memory locations SS:[ESP + 8], SS:[ESP + 12], SS:[ESP + 16], and SS:[ESP + 20], respectively, where the new value of ESP stored in step 4 is used. Because VM = 1, these are done as 8086 segment register loads. If VM = 0, check that the selectors in ES, DS, FS, and GS are valid in the interrupted routine. Null out invalid selectors to trap if an attempt is made to access through them.
6. If (RPL(CS) > CPL), pop the stack pointer SS:ESP from the stack. The ESP register is popped first, followed by 32 bits containing SS in the lower 16 bits. If VM = 0, SS is loaded as a protected mode segment register load. If VM = 1, an 8086 segment register load is used.
7. Resume execution of the interrupted routine. The VM bit in the FLAGS register (restored from the interrupt routine's stack image in step 1) determines whether the Intel486 processor resumes the interrupted routine in Protected mode or Virtual 8086 mode.

7.0 ON-CHIP CACHE

All members of the Intel486 processor family, except the IntelDX4 processor, contain an on-chip 8-Kbyte cache. (See section 7.1.1, "IntelDX4 Processor On-Chip Cache," for the IntelDX4 processor cache organization.) The cache is software transparent to maintain binary compatibility with previous generations of the Intel architecture.

The on-chip cache has been designed for maximum flexibility and performance. The cache has several operating modes offering flexibility during program execution and debugging. Memory areas can be defined as non-cacheable by software and external hardware. Protocols for cache line invalidations and replacement are implemented in hardware, easing system design.

7.1 Cache Organization

The on-chip cache is a unified code and data cache. The cache is used for both instruction and data accesses and acts on physical addresses. (See section 7.1.1 for IntelDX4 processor details.)

The cache organization is 4-way set associative and each line is 16 bytes wide. The 8 Kbytes of cache memory are logically organized as 128 sets, each containing 4 lines.

The cache memory is physically split into four 2-Kbyte blocks, each containing 128 lines. (See Figure 7-1.) There are 128 21-bit tags associated with each 2-Kbyte block. There is a valid bit for each line in the cache. Each line in the cache is either valid or not valid. There are no provisions for partially valid lines.

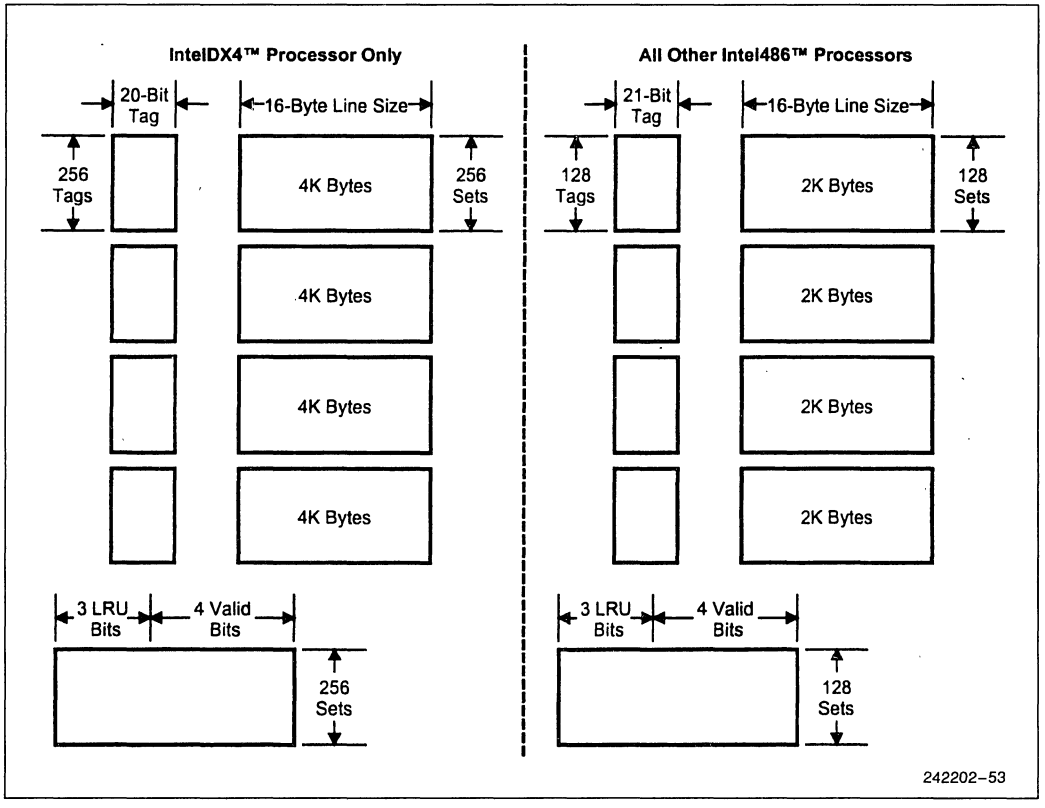


Figure 7-1. On-Chip Cache Physical Organization

For all Intel486 processors except the Write-Back Enhanced Intel486, the on-chip cache is write-through only. All writes will drive an external write bus cycle in addition to writing the information to the internal cache if the write was a cache hit. A write to an address not contained in the internal cache will only be written to external memory. Cache allocations are not made on write misses.

The Write-Back Enhanced Intel486 processors support two modes of operation with respect to internal cache configurations: Standard Bus Mode (write-through cache) and Enhanced Bus Mode (write-back cache). Standard Bus Mode operation for the Write-Back Enhanced Intel486 processors is the same as the write-through cache for all other Intel486 processors. (See section 7.1.2, "Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor Cache" and other write-back enhanced sections below for write-back cache information.)

7.1.1 IntelDX4™ PROCESSOR ON-CHIP CACHE

The IntelDX4 processor contains a 16-Kbyte write-through cache. The 16 Kbytes of cache memory are logically organized as 256 sets, each containing four lines.

The cache memory is physically split into four 4 Kbyte blocks, each containing 256 lines. (See Figure 7-1.) There are 256 20-bit tags associated with each 2-Kbyte block.

All other details listed in section 7.1 for the 8-Kbyte on-chip cache also apply to the IntelDX4 on-chip cache.

7.1.2 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSORS CACHE

The Write-Back Enhanced IntelDX2 processor implements a unified cache, with a total cache size of 8 Kbytes. The processor's on-chip cache supports a modified MESI (modified/exclusive/shared/invalid) write-back cache consistency protocol.

The Write-Back Enhanced IntelDX4 processor implements a unified cache, with a total cache size of 16 Kbytes. The processor's on-chip cache supports a modified MESI (modified / exclusive / shared / invalid) write-back cache consistency protocol.

For items which are common to both the Write-Back Enhanced IntelDX4 and Write-Back Enhanced IntelDX2 processors, the term "Write-Back Enhanced Intel486 processors" will be used.

The Write-Back Enhanced Intel486 processors internal cache is configurable as write-back or write-through on a line-by-line basis, provided the cache is enabled for write-back operation. The cache is enabled for write-back operation by driving the WB/WT# pin to a high state for at least two clocks before and two clocks after the falling edge of the RESET. Cache write-back and invalidations can be initiated by hardware or software. Protocols for cache consistency and line replacement are implemented in hardware to ease system design.

Once the cache configuration is selected, the Write-Back Enhanced Intel486 processors will continue to operate in the selected configuration and can only be changed to a different configuration by starting the **RESET** process again. Assertion of **SRESET** will not change the operating mode of the processor. **WB/WT#** has an internal pull down; If **WB/WT#** is unconnected, the processor will be in Standard Bus Mode, i.e., the on-chip cache is write-through. Table 7-1 lists the two modes of operation and the differences between the two modes.

Unless specifically noted, the following sections apply to the Write-Back Enhanced Intel486 processors in standard Bus Mode (Write-Through Cache) and all other Intel486 processors.

7.2 Cache Control

Control of the cache is provided by the CD and NW bits in CR0. CD enables and disables the cache. NW controls memory write-through and invalidates.

The CD and NW bits define four operating modes of the on-chip cache as given in Table 7-2. These modes provide flexibility in how the on-chip cache is used.

CD = 1, NW = 1

The cache is completely disabled by setting CD = 1 and NW = 1 and then flushing the cache. This mode may be useful for debugging programs where it is important to see all memory cycles at the pins. Writes that hit in the cache will not appear on the external bus.

It is possible to use the on-chip cache as fast static RAM by "pre-loading" certain memory areas into the cache and then setting CD = 1 and NW = 1. Pre-loading can be done by careful choice of memory references with the cache turned on or by use of the testability functions. (See section 11.2, "On-Chip Cache Testing.") When the cache is turned off, the memory mapped by the cache is "frozen" into the cache because fills and invalidates are disabled.

Table 7-1. Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor WB/WT# Initialization

State of WB/WT# at Falling Edge of RESET	Effect on Write-Back Enhanced IntelDX2™ Processor Operation
WB/WT# = LOW	**Processor is in Standard Bus Mode (Write-Through Cache)** 1. When **FLUSH#** is asserted, the internal cache will be invalidated in one system **CLK**. 2. No Special **FLUSH#** Acknowledge Cycles appear on the bus after the assertion of the **FLUSH#** pin. 3. All write-back specific inputs are ignored **(INV, WB/WT#)** 4. SRESET does not clear the SMBASE register. It behaves much like a RESET (invalidating the on-chip cache and resetting the CR0 register, for example). SRESET is NOT an interrupt.
WB/WT# = HIGH	**Processor is in Enhanced Bus Mode (Write-Back Cache)** 1. Write backs will be performed when a cache flush is requested (via the **FLUSH#** pin or the WBINVD instruction). The system must watch for the **FLUSH#** special cycles to determine the end of the flush. 2. The special **FLUSH#** Acknowledge Cycles will appear on the bus after the assertion of the **FLUSH#** and after all the cache write backs (if any) are completed on the bus. 3. **WB/WT#** is a sampled on a line-by-line basis to determine the state of a line to be allocated in the cache (as a Write Through (S state) or as Write Back (E state)). 4. The **WB/WT#** and **INV** inputs are no longer ignored. **HITM#** and **CACHE#** will be driven during appropriate bus cycles. 5. **PLOCK#** is always driven inactive. 6. **SRESET** is an interrupt. SRESET does not reset the SMBASE register or flush the on-chip cache. The CR0 register gets the same values as after RESET with the exception of the **CD and NW bits.** These two bits retain their previous status. (See section 9.2.18.4, "Soft Reset (SRESET)" and Table 3-7 for details on SRESET for write-back enhanced mode.)

Table 7-2. Cache Operating Modes

CD	NW	Operating Mode
1	1	Cache fills disabled, write-through and invalidates disabled
1	0	Cache fills disabled, write-through and invalidates enabled
0	1	INVALID. If CR0 is loaded with this configuration of bits, a GP fault with error code of 0 is raised.
0	0	Cache fills enabled, write-through and invalidates enabled

CD = 1, NW = 0

Cache fills are disabled but write-throughs and invalidates are enabled. This mode is the same as if the KEN# pin were strapped HIGH, disabling cache fills. Write-throughs and invalidates may still occur to keep the cache valid. This mode is useful if the software must disable the cache for a short period of time, and then re-enable it without flushing the original contents.

CD = 0, NW = 1

Invalid. If CR0 is loaded with this bit configuration, a General Protection fault with error code of 0 will occur.

CD = 0, NW = 0

This is the normal operating mode.

Completely disabling the cache is a two-step process. First, CD and NW must be set to 1, and then the cache must be flushed. If the cache is not flushed, cache hits on reads will still occur and data will be read from the cache.

7.2.1 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSOR CACHE CONTROL AND OPERATING MODES

The Write-Back Enhanced Intel486 processors retain the usage of CR0.CD and CR0.NW, in which the 1,1 state forces a cache-off condition after RESET, and the 0,0 state is the normal run state. Table 7-3 defines these control bits when the cache is enabled for write-back operation. Table 7-3 is also valid when the cache is in write-back mode and some lines are in a write-through state.

CD = 1, NW = 1

The 1,1 state is best used when no lines are allocated, which occurs naturally after RESET (but not SRESET), but must be forced (e.g., by instruction WBINVD) if entered during normal operation. In these cases, the Write-Back Enhanced Intel486 processor will operate as if it had no cache at all.

If the 1,1 state is exited, lines that are allocated as write back will be written back upon a snoop hit or replacement cycle. Lines that were allocated as write-through (and later modified while in the 1,1 state) will never appear on the bus.

CD = 1, NW = 0

The only difference from the normal 0,0 "run" state is that new line fills (and the line replacements that result from capacity limitations) do not occur. This causes the contents of the cache to be locked in, unless lines are invalidated using snoops.

7.3 Cache Line Fills

Any area of memory can be cached in the Intel486 processor. Non-cacheable portions of memory can be defined by the external system or by software. The external system can inform the Intel486 processor that a memory address is non-cacheable by returning the KEN# pin inactive during a memory access. (Refer to section 10.2.3, "Cacheable Cycles.") Software can prevent certain pages from being cached by setting the PCD bit in the page table entry.

A read request can be generated from program operation or by an instruction pre-fetch. The data will be supplied from the on-chip cache if a cache hit occurs on the read address. If the address is not in the cache, a read request for the data is generated on the external bus.

If the read request is to a cacheable portion of memory, the Intel486 processor initiates a cache line fill. During a line fill a 16-byte line is read into the Intel486 processor. Cache line fills will only be generated for read misses. Write misses will never cause a line in the internal cache to be allocated. If a cache hit occurs on a write, the line will be updated. Cache line fills can be performed over 8- and 16-bit buses using the dynamic bus sizing feature. Refer to section 10.1.2, "Dynamic Data Bus Sizing" for a description of dynamic bus sizing and section 10.2.3, "Cacheable Cycles" for further information on cacheable cycles.

Table 7-3. Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor Write-Back Cache Operating Modes

CR0 CD, NW	READ HIT	READ MISS	WRITE HIT(1)	WRITE MISS	Snoops
1,1 (state after reset)	read cache	read bus (no fill)	write cache (no write-through)	write bus	not accepted
1,0	read cache	read bus (no fill)	write cache, write bus if S	write bus	normal operation
0,1	This is a fault-protected disallowed state. A GP(0) will occur if an attempt is made to load CR0 with this state.				
0,0 (state **DURING** normal operation)	read cache	read bus, line fill	write cache, write bus if S	write bus	normal operation

NOTE:
1. Normal MESI state transitions occur on write hits in all legal states.

7.4 Cache Line Invalidations

The Intel486 processor contain both a hardware and software mechanism for invalidating lines in its internal cache. Cache line invalidations are needed to keep the Intel486 processor cache contents consistent with external memory.

Refer to section 10.2.8, "Invalidate Cycles" for further information on cache line invalidations.

7.4.1 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSOR SNOOP CYCLES AND WRITE-BACK MODE INVALIDATION

In Enhanced bus mode, the Write-Back Enhanced Intel486 processors performs invalidations differently than other Intel486 processors. Snoop Cycles are initiated by the system to determine if a line is present in the cache, and what the state is. Snoop cycles may further be classified as Inquire cycles or Invalidate cycles. Inquire cycles are driven to the Write-Back Enhanced Intel486 processors when another bus master initiates a memory read cycle, to determine if the processor cache contains the latest data. If the snooped line is in the Write-Back Enhanced Intel486 processors cache and has the most recent information, the processor must schedule a write back of the data. Inquire cycles are driven with INV = "0". Invalidate cycles are driven to the Write-Back Enhanced Intel486 processors when the other bus master initiates a memory write cycle to determine if the Write-Back Enhanced Intel486 processors cache contains the snooped line. The Invalidate cycles are driven with INV = "1", so that if the snooped line is in the on-chip cache, the line is invalidated. Snoop cycles are described in detail in the "Bus Functional Description" section.

The Write-Back Enhanced Intel486 processors has control mechanisms (including snooping) for writing back the modified write-back lines and invalidating the cache. There are special bus cycles associated with write-backs and invalidation. All of the Write-Back Enhanced Intel486 processors special cycles require acknowledgment by RDY# or BRDY#. During the special cycles, the addresses shown in the Table 7-4 are driven onto the address bus and the data bus is left undefined.

7.5 Cache Replacement

When a line needs to be placed in its internal cache the Intel486 processor first checks to see if there is a non-valid line in the set that can be replaced. If all four lines in the set are valid, a pseudo least-recently-used mechanism is used to determine which line should be replaced.

A valid bit is associated with each line in the cache. When a line needs to be placed in a set, the four valid bits are checked to see if there is a non-valid line that can be replaced. If a non-valid line is found, that line is marked for replacement.

The four lines in the set are labeled I0, I1, I2, and I3. The order in which the valid bits are checked during an invalidation is I0, I1, I2 and I3. All valid bits are cleared when the processor is reset or when the cache is flushed.

Table 7-4. Encoding of the Special Cycles for Write-Back Cache

Cycle Name	M/IO#	D/C#	W/R#	BE3#–BE0#	A4–A2
Write-Back*	0	0	1	0111	000
First Flush Ack Cycle*	0	0	1	0111	001
Flush*	0	0	1	1101	000
Second Flush Ack Cycle*	0	0	1	1101	001
Shutdown	0	0	1	1110	000
HALT	0	0	1	1011	000
Stop Grant Ack Cycle	0	0	1	1011	100

* For the Write-Back Enhanced Intel486 processors only. FLUSH is present on all Intel486 processors, but differs for Standard Mode. Refer to appropriate sections.

Replacement in the cache is handled by a pseudo LRU mechanism when all four lines in a set are valid. Three bits, B0, B1 and B2, are defined for each of the 128 sets in the cache. These bits are called the LRU bits. The LRU bits are updated for every hit or replace in the cache.

If the most recent access to the set was to I0 or I1, B0 is set to 1. B0 is set to 0 if the most recent access was to I2 or I3. If the most recent access to I0:I1 was to I0, B1 is set to 1, else B1 is set to 0. If the most recent access to I2:I3 was to I2, B2 is set to 1, else B2 to 0.

The pseudo LRU mechanism works in the following manner. When a line must be replaced, the cache will first select which of I0:I1 and I2:I3 was least recently used. Then the cache will determine which of the two lines was least recently used and mark it for replacement. This decision tree is shown in Figure 7-2.

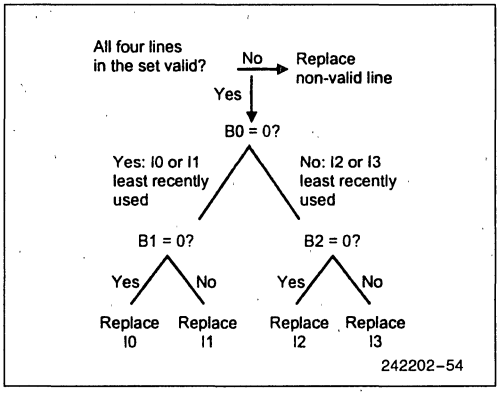


Figure 7-2. On-Chip Cache Replacement Strategy

7.6 Page Cacheability

Two bits for cache control, PWT and PCD, are defined in the page table and page directory entries. The state of these bits is driven out on the PWT and PCD pins during memory access cycles.

The PWT bit controls the write policy for second level caches used with the Intel486 processor. Setting PWT = 1 defines a write-through policy for the current page while PWT = 0 defines the possibility of write-back. The state of PWT is ignored internally by the Intel486 processor for on-chip cache in write-through mode.

The PCD bit controls cacheability on a page-by-page basis. The PCD bit is internally AND'ed with the KEN# signal to control cacheability on a cycle-by-cycle basis (see Figure 7-3). PCD = 0 enables caching while PCD = 1 forbids it. Note that cache fills are enabled when PCD = 0 AND KEN# = 0. This logical AND is implemented physically with a NOR gate.

The state of the PCD bit in the page table entry is driven on the PCD pin when a page in external memory is accessed. The state of the PCD pin informs the external system of the cacheability of the requested information. The external system then returns KEN# telling the Intel486 processor if the area is cacheable. The Intel486 processor initiates a cache line fill if PCD and KEN# indicate that the requested information is cacheable.

The PCD bit is OR'ed with the CD (cache disable) bit in control register 0 to determine the state of the PCD pin. If CD = 1, the Intel486 processor forces the PCD pin HIGH. If CD = 0, the PCD pin is driven with the value for the page table entry/directory. (See Figure 7-3.)

The PWT and PCD bits for a bus cycle are obtained from either CR3, the page directory or page table entry. These bits are assumed to be zero during real mode, whenever paging is disabled, or for cycles that bypass paging, (I/O references, interrupt acknowledge and HALT cycles).

When paging is enabled, the bits from the page table entry are cached in the TLB, and are driven any time the page mapped by the TLB entry is referenced. For normal memory cycles, PWT and PCD are taken from the page table entry. During TLB refresh cycles where the page table and directory entries are read, the PWT and PCD bits must be obtained elsewhere. During page table updates the bits are obtained from the page directory. When the page directory is updated the bits are obtained from CR3. PCD and PWT bits are initialized to zero at reset, but can be modified by level 0 software.

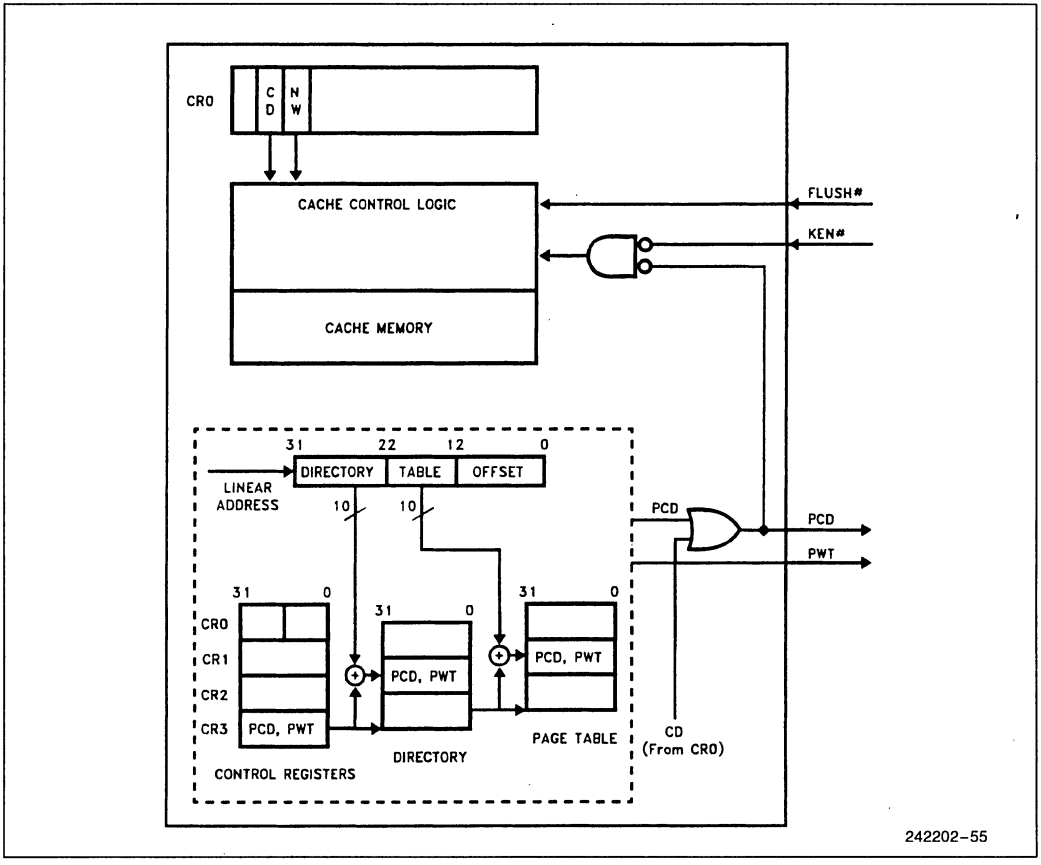


Figure 7-3. Page Cacheability

7.6.1 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSORS PAGE CACHEABILITY

In Write-Back Enhanced Intel486 processors-based system, both the processor and the system hardware must determine the cacheability and the configuration (write-back or write-through) on a line by line basis. The system hardware's cacheability is determined by KEN# and the configuration by WB/WT#. The processor's indication of cacheability is determined by PCD and the configuration by PWT. The PWT bit controls the write policy for the second level caches used with the Write-Back Enhanced Intel486 processors. Setting PWT to 1 defines a write-through policy for the current page, while clearing PWT to 0 defines a write-back policy for the current page.

7.7 Cache Flushing

The on-chip cache can be flushed by external hardware or by software instructions. Flushing the cache clears all valid bits for all lines in the cache. The cache is flushed when external hardware asserts the FLUSH# pin.

The FLUSH# pin needs to be asserted for one clock if driven synchronously or for two clocks if driven asynchronously. FLUSH# is asynchronous, but setup and hold times must be met for recognition in a particular cycle. FLUSH# should be de-asserted before the cache flush is complete. Failure to de-assert the pin will cause execution to stop as the processor will be repeatedly flushing the cache. If external hardware activates flush in response to an I/O write, FLUSH# must be asserted for at least two clocks prior to ready being returned for the I/O write. This ensures that the flush completes before the processor begins execution of the instruction following the OUT instruction.

The instructions INVD and WBINVD cause the on-chip cache to be flushed. External caches connected to the Intel486 processor are signaled to flush their contents when these instructions are executed.

WBINVD will also cause an external write-back cache to write back dirty lines before flushing its contents. The external cache is signaled using the bus cycle definition pins and the byte enables (refer to section 9.2.6 "Bus Cycle Definition" for the bus cycle definition pins and section 10.2.11 "Special Bus Cycles" for special bus cycles). Refer to the *Intel486™ Processor Programmers Reference Manual* for detailed instruction definitions.

The results of the INVD and WBINVD instructions are identical for the operation of the non-write-back enhanced Intel486 processor on-chip cache because the cache is write-through.

7.7.1 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSORS CACHE FLUSHING

The on-chip cache can be flushed by external hardware or by software instructions.

Flushing the cache through hardware is accomplished by driving the FLUSH# pin low. This causes the cache to write back all modified lines in the cache and mark the state bits invalid. The First Flush Acknowledge cycle is driven by the Write-Back Enhanced Intel486 processors followed by the Second Flush Acknowledge cycle after all write-backs and invalidations are complete. The two special cycles are issued even if there are no dirty lines to write back.

The INVD and WBINVD instructions cause the on-chip cache to be invalidated. WBINVD causes the modified lines in the internal cache to be written back, and all lines to be marked invalid. After execution of the WBINVD instruction, the Write-back and Flush special cycles are driven to indicate to any external cache that it should write back and invalidate its contents. These two special cycles are issued even if there are no dirty lines to be written back. INVD causes all lines in the cache to be invalidated, so modified lines in the cache are **not** written back. The Flush special cycle is driven after the INVD instruction is executed to indicate to any external cache that it should invalidate its contents. Care should be taken when using the INVD instruction to avoid creating cache consistency problems.

NOTE:
It is recommended to use the WBINVD instruction instead of the INVD instruction if the on-chip cache is configured in the write-back mode.

The assertion of the RESET pin invalidates the entire cache without writing back the modified lines. No special cycles are issued after the invalidation is complete.

Snoop cycles with invalidation (INV = 1) cause the Write-Back Enhanced Intel486 processors to invalidate an individual cache line. If the snooped line is a modified line, then the processor schedules a write-back cycle. Inquire cycles with no-invalidation cause the Write-Back Enhanced Intel486 processors to only write-back the line, if the inquired line is in M-state, and not invalidate the line.

SRESET, STPCLK#, INTR, NMI and SMI# are recognized and latched, but not serviced during the full-cache, modified-line write-backs, caused either by WBINVD instruction or the FLUSH#. However, BOFF#, AHOLD and HOLD are recognized **DURING** the full-cache, modified-line write-backs.

7.8 Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor Write-Back Cache Architecture

This section describes additional features pertaining to the write-back mode of the Write-Back Enhanced Intel486 processors.

7.8.1 WRITE-BACK CACHE COHERENCY PROTOCOL

The Write-Back Enhanced Intel486 processors cache protocol supports a cache line in one of the following four states:

- Whether a line is valid and defined as write-back during allocation (E-state).
- If it is valid and defined as write-through during allocation (S-state).
- If it has been modified (M-state).
- If it is invalid (I-state).

These four states are the **M** (Modified line), **E** (write-back line), **S** (write-through line) and the **I** (Invalid line) states and the protocol is referred to as the "Modified MESI protocol." A definition of the states is given below:

M - Modified: An M-state line is modified (different from main memory) and can be accessed (read/written to) without sending a cycle out on the bus.

E - Exclusive: An E-state line is a "write-back" line, but the line is not modified (i.e., it is coherent with main memory). An E-state line can be accessed (read/written to) without generating a bus cycle and a write to an E-state line will cause the line to become modified.

S - Shared: An S-state line is a "write-through" line, and is coherent with main memory. A read hit to an S-state line will not generate bus activity, but a write hit to an S-state line will generate a write-through cycle on the bus. A write to an S-state line will update the cache and the main memory.

I - Invalid: This state indicates that the line is not in the cache. A read to this line will be a miss and may cause the Write-Back Enhanced Intel486 processors to execute a line fill (fetch the whole line into the cache from main memory). A write to an invalid line will cause the Write-Back Enhanced Intel486 processors to execute a write-through cycle on the bus.

Every line in the Write-Back Enhanced Intel486 processor's cache is assigned a state dependent on both Write-Back Enhanced Intel486 processor-generated activities and activities generated by the system hardware. As the Write-Back Enhanced Intel486 processors are targeted for uniprocessor systems, a subset of MESI protocol, namely MEI, is used in the Write-Back Enhanced Intel486 processors to maintain cache coherency.

With the modified MESI protocol, it is assumed that in a uniprocessor system lines are defined as write-back or write-through at allocation time. This property associated with a line is never altered. The lines allocated as write-through go to S-state and remain in S-state. A cache line that is allocated as write-back never enters the S-state. The WB/WT# pin is sampled during line allocation and is used strictly to characterize a line as write-back or write-through.

7.8.1.1 State Transition Tables

State transitions are caused by processor-generated transactions (memory reads/writes) and by a set of external input signals and internally generated variables. The Write-Back Enhanced Intel486 processors also drive certain pins as a consequence of the Cache Consistency Protocol.

Read Cycles

Table 7-5 shows the state transitions for lines in the cache during unlocked read cycles.

Write Cycles

The state transitions of cache lines during Write-Back Enhanced Intel486 processor-generated write cycles are described in Table 7-6.

Table 7-5. Cache State Transitions for Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processors Initiated Unlocked Read Cycles

Present State	Pin Activity	Next State	Description
M	n/a	M	Read hit; data is provided to processor core by cache. No bus cycle is generated.
E	n/a	E	Read hit; data is provided to processor core by cache. No bus cycle is generated.
S	n/a	S	Read hit; Data is provided to the processor by the cache. No bus cycle is generated.
I	CACHE# low AND KEN# low AND WB/WT# high AND PWT low	E	Data item does not exist in cache (MISS). A line fill cycle (read) will be generated by the Write-Back Enhanced IntelDX2™ processor. This state transition will occur if WB/WT# is sampled high with first BRDY#.
I	CACHE# low AND KEN# low AND (WB/WT# low OR PWT high)	S	Same as previous read miss case except that WB/WT# is sampled low with first BRDY# or PWT is high.
I	CACHE# high OR KEN# high	I	KEN# pin inactive; the line is not intended to be cached in the Write-Back Enhanced Intel486 processors.

NOTES:
Locked accesses to the cache will cause the accessed line to transition to the Invalid state.
PCD can also be used by the processor to determine the cacheability, but using the CACHE# pin is recommended. The transition from I to E or S states (based on WB/WT#) occurs only if KEN# is sampled low one clock prior to the first BRDY# and then one clock prior to the last BRDY#, and the cycle is transformed into a line fill cycle. If KEN# is sampled high, the line is not cached and remains in the I state.

Table 7-6. Cache State Transitions for Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor-Initiated Write Cycles

Present State	Pin Activity	Next State	Description
M	n/a	M	Write hit; update cache. No bus cycle generated to update memory.
E	n/a	M	Write hit; update cache only. No bus cycle generated; line is now modified.
S	n/a	S	Write hit; cache updated with write data item. A write-through cycle is generated on the bus to update memory. Subsequent writes to E-state or M-state lines are held up until this write-through cycle is completed.
I	n/a	I	Write miss; a write-through cycle is generated on the bus to update external memory. No allocation is done. Subsequent writes to the E or M lines are blocked until the write-miss is completed.

Note that even though memory writes are buffered while I/O writes are not, these writes appear at the pins in the same order as they were generated by the processor. Write-Back cycles caused by the replacement of M-state lines are buffered, while Write-Backs due to Snoop hit to M-state lines are not buffered.

Cache Consistency Cycles (Snoop Cycles)

The purpose of Snoop cycles is to check whether the address being presented by another bus master is contained within the cache of the Write-Back Enhanced Intel486 processors. Snoop cycles may be initiated with or without an invalidation request (INV = 1 or 0). If a snoop cycle is initiated with INV = 0 (usually during memory read cycles by another master), it is referred to as an Inquire cycle. If a snoop cycle is initiated with INV = 1 (usually during memory write cycles), it is referred to as an Invalidate cycle. If the address hits a modified line in the cache, the HITM# pin is asserted, and the modified line is written back onto the bus. Table 7-7 describes state transitions for Snoop cycles.

7.8.2 DETECTING ON-CHIP WRITE-BACK CACHE OF THE WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSORS

The write-back policy of the on-chip cache of the Write-Back Enhanced Intel486 processors may be detected by software or hardware. The software mechanism makes use of the CPUID instruction. (See Appendix B, "Feature Determination," for use of the CPUID instruction.) The hardware mechanism makes use of a write-back-related output signal from the processor.

A software mechanism to determine if a given processor has write-back support for the on-chip cache should drive the WB/WT# pin to "1" during RESET. This pin will be sampled by the processor during the falling edge of the RESET. Execute the CPUID instruction, which returns the model number in the EAX register, EAX[7:4]. If the model number returned is 7 (Write-Back Enhanced Intel486 processors) and the family number is 4, the on-chip cache supports the write-back policy. If the model number returned is in the range 0 through 6 or 8, the on-chip cache only supports the write-through policy.

The following pseudo code/steps give an example of the initialization BIOS that can be used to detect the presence of the write-back on-chip cache:

- Boot Address Cold start.
- Load Segment Registers and null IDTR.
- Execute CPUID instruction and determine the Family ID and Model ID.
- Compare the Family ID to 4 and the Model ID returned to the values listed in Table B-2.

The hardware mechanism involves using the HITM# signal. For the Write-Back Enhanced Intel486 processors, this signal is driven inactive (high) during RESET. The chipset can sample this output on the falling edge of RESET. If HITM# is sampled high on the falling edge of RESET, the processor supports on-chip write-back cache configuration. For those processors that do not support internal write-back caching, this signal is an INC, and this output is not driven.

Table 7-7. Cache State Transitions During Snoop Cycles

Present State	Next State INV=1	Next State INV=0	Description
M	I	E	Snoop hit to a modified line indicated by HITM# pin low. Write-Back Enhanced IntelDX2 Processor schedules the write-back of the modified line to memory. The state of the line changes to E provided INV = 0 and changes to I if INV = 1.
E	I	E	Snoop hit, no bus cycle generated. State remains unaltered if INV = 0, and changes to I if INV = 1. There is no external indication of this snoop hit.
S	I	S	Snoop hit, no bus cycle generated. State remains unaltered if INV = 0, and changes to I if INV = 1. There is no external indication of this snoop hit.
I	I	I	Address not in cache.

8.0 SYSTEM MANAGEMENT MODE (SMM) ARCHITECTURES

8.1 SMM Overview

The Intel486 processor supports four modes: Real, Virtual-86, Protected, and System Management Mode (SMM). As an operating mode, SMM has a distinct processor environment, interface and hardware/software features.

SMM provides system designers with a means of adding new software-controlled features to computer products that operate transparently to the operating system and software applications. SMM is intended for use only by system firmware, not by applications software or general purpose systems software.

The SMM architectural extension consists of the following elements:

1. System Management Interrupt (SMI#) hardware interface.
2. Dedicated and secure memory space (SMRAM) for SMI# handler code and processor state (context) data with a status signal for the system to decode access to that memory space, SMIACT#. (The SMBASE address is relocatable and could also be relocated to non-cacheable address space.)
3. Resume (RSM) instruction, for exiting the System Management Mode.
4. Special Features such as I/O-Restart, for transparent power management of I/O peripherals, and Auto HALT Restart.

8.2 Terminology

The following terms are used throughout the discussion of System Management Mode.

SMM: System Management Mode. This is the operating environment that the processor (system) enters when the System Management Interrupt is being serviced.

SMI#: System Management Interrupt. This is part of the SMM interface. When SMI# is asserted (SMI# pin asserted low) it causes the processor to invoke SMM. **The SMI# pin is the only means of entering SMM.**

SMM handler: System Management Mode handler. This is the code that will be executed when the processor is in SMM. An example application that this code might implement is a power management control or a system control function.

RSM: Resume instruction. This instruction is used by the SMM handler to exit the SMM and return to the interrupted operating system or application process.

SMRAM: This is the physical memory dedicated to SMM. The SMM handler code and related data reside in this memory. This memory is also used by the processor to store its context before executing the SMM handler. The operating system and applications do not have access to this memory space.

SMBASE: Control register that contains the address of the SMRAM space.

Context: This term refers to the processor state. The SMM discussion refers to the context, or processor state, just before the processor invokes SMM. The context normally consists of the processor registers that fully represent the processor state.

Context Switch: A context switch is the process of either saving or restoring the context. The SMM discussion refers to the context switch as the process of saving/restoring the context while invoking/exiting SMM, respectively.

8.3 System Management Interrupt Processing

The system interrupts the normal program execution and invokes SMM by generating a System Management Interrupt (SMI#) to the processor. The processor will service the SMI# by executing the following sequence (see Figure 8-1):

1. The processor asserts the SMIACT# signal, indicating to the system that it should enable the SMRAM.
2. The processor saves its state (context) to SMRAM, starting at default address location 3FFFFH, proceeding downward in a stack-like fashion.
3. The processor switches to the System Management Mode processor environment (a pseudo-real mode).

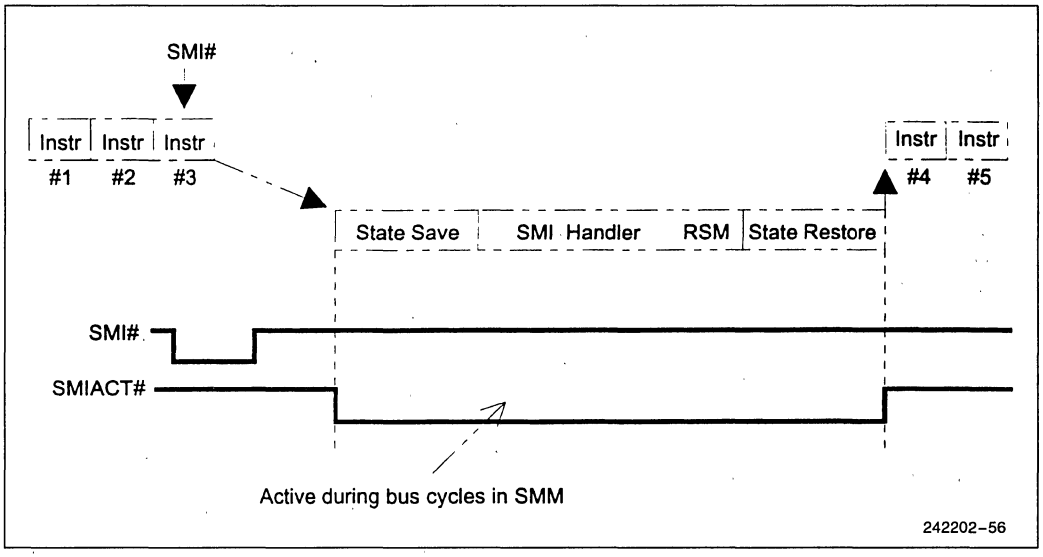


Figure 8-1. Basic SMI# Interrupt Service

4. The processor will then jump to the default absolute address of 38000H in SMRAM to execute the SMI# handler. This SMI# handler performs the system management activities.
5. The SMI# handler will then execute the RSM instruction which restores the processor's context from SMRAM, de-asserts the SMIACT# signal, and then returns control to the previously interrupted program execution.

NOTE:
The above sequence is valid for the default SMBASE value only. See the following sections for a description of the SMBASE register and SMBASE relocation.

The System Management Interrupt hardware interface consists of the SMI# interrupt request input and the SMIACT# output used by the system to decode the SMRAM.

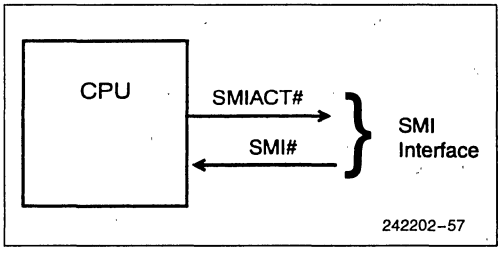


Figure 8-2. Basic SMI# Hardware Interface

8.3.1 SYSTEM MANAGEMENT INTERRUPT (SMI#)

SMI# is a falling-edge triggered, non-maskable interrupt request signal. SMI# is an asynchronous signal, but setup and hold times, t_{20} and t_{21}, must be met in order to guarantee recognition on a specific

clock. The SMI# input need not remain active until the interrupt is actually serviced. The SMI# input only needs to remain active for a single clock if the required setup and hold times are met. SMI# will also work correctly if it is held active for an arbitrary number of clocks.

The SMI# input must be held inactive for at least four external clocks after it is asserted to reset the edge triggered logic. A subsequent SMI# might not be recognized if the SMI# input is not held inactive for at least four clocks after being asserted.

SMI#, like NMI, is not affected by the IF bit in the EFLAGS register and is recognized on an instruction boundary. An SMI# will not break locked bus cycles. The SMI# has a higher priority than NMI and is not masked during an NMI. In order for SMI# to be recognized with respect to SRESET, SMI# should not be asserted until two (2) clocks after SRESET becomes inactive.

After the SMI# interrupt is recognized, the SMI# signal will be masked internally until the RSM instruction is executed and the interrupt service routine is complete. Masking the SMI# prevents recursive SMI# calls. SMI# must be de-asserted for at least 4 clocks to reset the edge triggered logic. If another SMI# occurs while the SMI# is masked, the pending SMI# will be recognized and executed on the next instruction boundary after the current SMI# completes. This instruction boundary occurs before execution of the next instruction in the interrupted application code, resulting in back to back SMM handlers. Only one SMI# can be pending while SMI# is masked.

The SMI# signal is synchronized internally and must be asserted at least three (3) CLK periods prior to asserting the RDY# signal in order to guarantee recognition on a specific instruction boundary. This is important for servicing an I/O trap with an SMI# handler. (See Figure 8-3.)

8.3.2 SMI# ACTIVE (SMIACT#)

SMIACT# indicates that the processor is operating in System Management Mode. The processor asserts SMIACT# in response to an SMI# interrupt request on the SMI# pin. SMIACT# is driven active after the processor has completed all pending write cycles (including emptying the write buffers), and before the first access to SMRAM when the processor saves (writes) its state (or context) to SMRAM. SMIACT# remains active until the last access to SMRAM when the processor restores (reads) its state from SMRAM. The SMIACT# signal does not float in response to HOLD. The SMIACT# signal is used by the system logic to decode SMRAM (See Figure 8-2).

The number of CLKs required to complete the SMM state save and restore is very dependent on-system memory performance. The values listed in Table 8-1 assume 0 wait-state memory writes (2 CLK cycles), 2-1-1-1 burst read cycles, and 0 wait-state non-burst reads (2 CLK cycles). Additionally, it is assumed that the data read during the SMM state restore sequence is not cacheable.

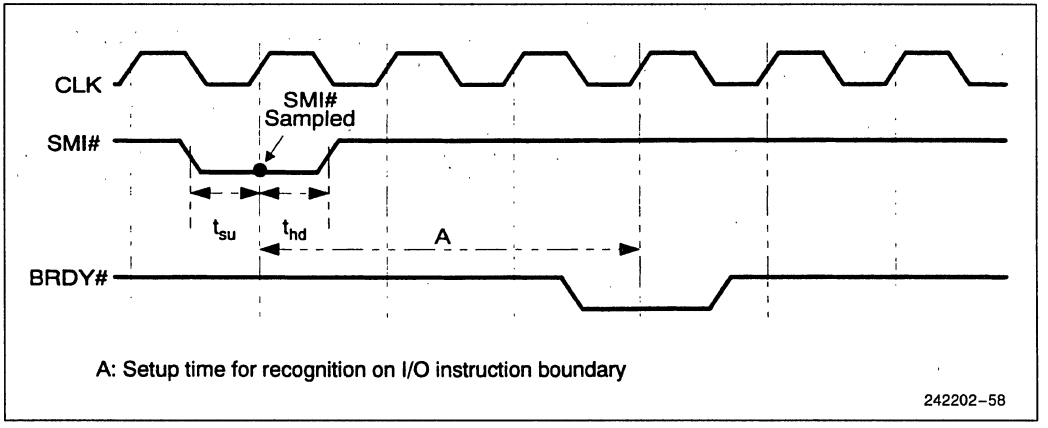


Figure 8-3. SMI# Timing for Servicing an I/O Trap

Figure 8-4 and Table 8-1 can be used for latency calculations. As shown, the minimum time required to enter an SMI# handler routine for the Intel486 DX processor (from the completion of the interrupted instruction) is given by:

Latency to beginning of SMI# handler =

A + B + C = 153 CLKs

and the minimum time required to return to the interrupted application (following the final SMM instruction before RSM) is given by:

Latency to continue interrupted application =

E + F + G = 243 CLKs

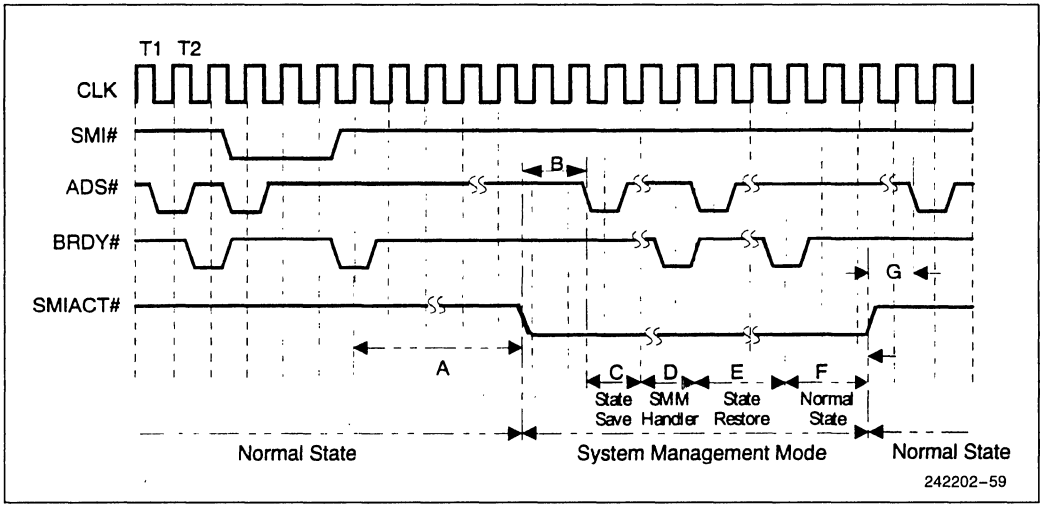


Figure 8-4. Intel486™ Processor SMIACT# Timing

Table 8-1. Intel486™ Processor SMIACT# Timing

	Intel486™ SX Processor	IntelSX2™ Processor	Intel486 DX Processor	IntelDX2™ Processor	IntelDX4™ Processor 3X	IntelDX4 Processor 2X
A:Last RDY# from non-SMM transfer to SMIACT# assertion	2 CLK minimum	1 CLK minimum	2 CLK minimum	1 CLK minimum	1 CLK minimum	1 CLK minimum
B:SMIACT# assertion to first ADS# for SMM state save	40 CLK minimum	20 CLK minimum	40 CLK minimum	20 CLK minimum	13 CLK minimum	20 CLK minimum
C:SMM state save (dependent on memory performance)	Approx 139 CLKs	Approx. 139 CLKs	Approx. 139 CLKs	Approx. 139 CLKs	Approx. 139 CLKs	Approx. 139 CLKs
D:SMM handler	User determined	User determined	User determined	User determined	User determined	User determined
E:SMM state restore (dependent on memory performance)	Approx. 236 CLKs	Approx. 236 CLKs	Approx. 236 CLKs	Approx. 236 CLKs	Approx. 236 CLKs	Approx. 236 CLKs
F:Last RDY# from SMM transfer to de-assertion of SMIACT#	4 CLK minimum	2 CLK minimum	4 CLK minimum	2 CLK minimum	1 CLK minimum	1 CLK minimum
G: SMIACT# de-assertion to first non-SMM ADS#	20 CLK minimum	10 CLK minimum	20 CLK minimum	10 CLK minimum	6 CLK minimum	10 CLK minimum

8.3.3 SMRAM

The Intel486 processor uses the SMRAM space for state save and state restore operations during an SMI# and RSM. The SMI# handler, which also resides in SMRAM, uses the SMRAM space to store code, data and stacks. In addition, the SMI# handler can use the SMRAM for system management information such as the system configuration, configuration of a powered-down device, and system designer-specific information.

The processor asserts the SMIACT# output to indicate to the memory controller that it is operating in System Management Mode. The system logic should ensure that only the processor has access to this area. Alternate bus masters or DMA devices trying to access the SMRAM space when SMIACT# is active should be directed to system RAM in the respective area.

The system logic is minimally required to decode the physical memory address range from 38000H–3FFFFH as SMRAM area. The processor will save its state to the state save area from 3FFFFH downward to 3FE00H. After saving its state the processor will jump to the address location 38000H to begin executing the SMI# handler. The system logic can choose to decode a larger area of SMRAM as needed. The size of this SMRAM can be between 32 Kbytes and 4 Gbytes.

The system logic should provide a manual method for switching the SMRAM into system memory space when the processor is **not** in SMM. This will enable initialization of the SMRAM space (i.e., loading SMI# handler) before executing the SMI# handler during SMM. (See Figure 8-5.)

8.3.3.1 SMRAM State Save Map

When the SMI# is recognized on an instruction boundary, the processor core first sets the SMIACT# signal LOW indicating to the system logic that accesses are now being made to the system-defined SMRAM areas. The processor then writes its state to the state save area in the SMRAM. The state save area starts at CS Base + [8000H + 7FFFH]. The default CS Base is 30000H, therefore the default state save area is at 3FFFFH. In this case, the CS Base can also be referred to as the SMBASE.

If SMBASE Relocation is enabled, then the SMRAM addresses can change. The following formula is used to determine the relocated addresses where the context is saved. The context will reside at CS Base + [8000H + Register Offset], where the default initial CS Base is 30000H and the Register Offset is listed in the SMRAM state save map (Table 8-2). Reserved spaces will be used to accommodate new registers in future processors. The state save area starts at 7FFFH and continues downward in a stack-like fashion.

Some of the registers in the SMRAM state save area may be read and changed by the SMI# handler, with the changed values restored to the processor registers by the RSM instruction. Some register images are read-only, and must not be modified (modifying these registers will result in unpredictable behavior). The values stored in the areas marked reserved may change in future processors. An SMM handler should not rely on any values stored in an area that is marked as reserved.

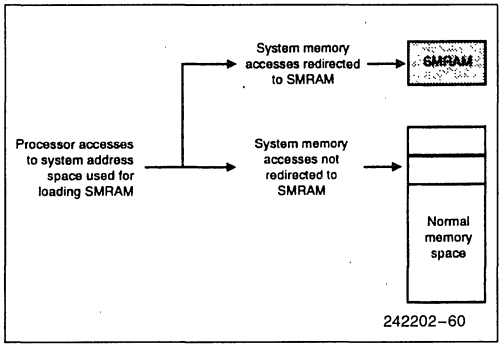


Figure 8-5. Redirecting System Memory Addresses to SMRAM

Table 8-2. SMRAM State Save Map

Register Offset	Register	Writeable?
7FFC	CR0	NO
7FF8	CR3	NO
7FF4	EFLAGS	YES
7FF0	EIP	YES
7FEC	EDI	YES
7FE8	ESI	YES
7FE4	EBP	YES
7FE0	ESP	YES
7FDC	EBX	YES
7FD8	EDX	YES
7FD4	ECX	YES
7FD0	EAX	YES
7FCC	DR6	NO
7FC8	DR7	NO
7FC4	TR*	NO
7FC0	LDTR*	NO
7FBC	GS*	NO
7FB8	FS*	NO
7FB4	DS*	NO
7FB0	SS*	NO
7FAC	CS*	NO
7FA8	ES*	NO
7FA7-7F98	Reserved	NO
7F94	IDT Base	NO
7F93-7F8C	Reserved	NO
7F88	GDT Base	NO
7F87-7F04	Reserved	NO
7F02	Auto HALT Restart Slot (Word)	YES
7F00	I/O Trap Restart Slot (Word)	YES
7EFC	SMM Revision Identifier (Dword)	NO
7EF8	SMBASE Slot (Dword)	YES
7EF7-7E00	Reserved	NO

NOTES:
*Upper two bytes are reserved.
Modifying a value that is marked as not writeable will result in unpredictable behavior.
Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address and the high-order byte in the high address.

The following registers are saved and restored (in areas of the state save that are marked reserved), but are not visible to the system software programmer:

CR1, CR2 and CR4, hidden descriptor registers for CS, DS, ES, FS, GS, and SS.

If an SMI# request is issued for the purpose of powering down the processor, the values of all reserved locations in the SMM state save must be saved to non-volatile memory.

The following registers are not automatically saved and restored by SMI# and RSM:

DR5–DR0, TR7–TR3, FPU registers: STn, FCS, FSW, tag word, FP instruction pointer, FP opcode, and operand pointer.

For all SMI# requests except for suspend/resume, these registers do not have to be saved because their contents will not change. However, during a power down suspend/resume, a resume reset will clear these registers back to their default values. In this case, the suspend SMI# handler should read these registers directly to save them and restore them during the power up resume. Anytime the SMI# handler changes these registers in the processor, it must also save and restore them.

8.3.4 EXIT FROM SMM

The **RSM** instruction is only available to the SMI# handler. The opcode of the instruction is 0FAAH. Execution of this instruction while the processor is executing outside of SMM will cause an invalid opcode error. The last instruction of the SMI# handler will be the RSM instruction.

The RSM instruction restores the state save image from SMRAM back to the processor, then returns control back to the interrupted program execution. There are three SMM features that can be enabled by writing to control "slots" in the SMRAM state save area.

Auto HALT Restart. It is possible for the SMI# request to interrupt the HALT state. The SMI# handler can tell the RSM instruction to return control to the HALT instruction **or** to return control to the instruction following the HALT instruction by appropriately setting the Auto HALT Restart slot. The default operation is to restart the HALT instruction.

I/O Trap Restart. If the SMI# interrupt was generated on an I/O access to a powered-down device, the SMI# handler can tell the RSM instruction to re-execute that I/O instruction by setting the I/O Trap Restart slot.

SMBASE Relocation. The system can relocate the SMRAM by setting the SMBASE Relocation slot in the state save area. The RSM instruction will set the SMBASE in the processor based on the value in the SMBASE relocation slot. The SMBASE must be 32K aligned.

For further details on these SMM features, see section 8.5.

If the processor detects invalid state information, it enters the shutdown state. This happens only in the following situations:

- The value stored in the SMBASE slot is not a 32-Kbyte-aligned address.
- A reserved bit of CR4 is set to 1.
- A combination of bits in CR0 is illegal; namely, (PG=1 and PE=0) or (NW=1 and CD=0).

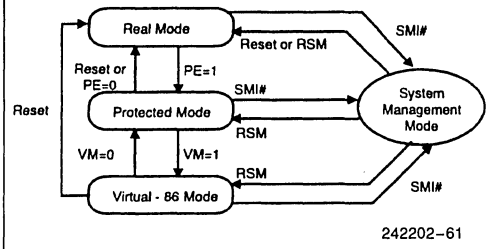
In shutdown mode, the processor stops executing instructions until an NMI interrupt is received or reset initialization is invoked. The processor generates a special bus cycle to indicate it has entered shutdown mode.

NOTE:
INTR and SMI# will also bring the processor out of a shutdown that is encountered due to invalid state information from SMM execution. Make sure that INTR and SMI# are not asserted if SMM routines are written such that a shutdown occurs.

8.4 System Management Mode Programming Model

8.4.1 ENTERING SYSTEM MANAGEMENT MODE

SMM is one of the major operating modes, on a level with Protected mode, Real address mode or virtual-86 mode. Figure 8-6 shows how the processor can enter SMM from any of the three modes and then return.



NOTE:
Reset could occur by asserting the RESET or SRESET pin.

Figure 8-6. Transition to and from System Management Mode

The external signal SMI# causes the processor to switch to SMM. The RSM instruction exits SMM. SMM is transparent to applications programs and operating systems because of the following:

- The only way to enter SMM is via a type of non-maskable interrupt triggered by an external signal.
- The processor begins executing SMM code from a separate address space, referred to earlier as system management RAM (SMRAM).
- Upon entry into SMM, the processor saves the register state of the interrupted program in a part of SMRAM called the SMM context save space.
- All interrupts normally handled by the operating system or by applications are disabled upon entry into SMM
- A special instruction, RSM, restores processor registers from the SMM context save space and returns control to the interrupted program.

SMM is similar to Real address mode in that there are no privilege levels or address mapping. An SMM program can execute all I/O and other system instructions and can address up to four Gbytes of memory.

8.4.2 PROCESSOR ENVIRONMENT

When an SMI# signal is recognized on an instruction execution boundary, the processor waits for all stores to complete, including emptying of the write buffers. The final write cycle is complete when the system returns RDY# or BRDY#. The processor then drives SMIACT# active, saves its register state to SMRAM space, and begins to execute the SMM handler.

SMI# has greater priority than debug exceptions and external interrupts. This means that if more than one of these conditions occur at an instruction boundary, only the SMI# processing occurs, not a debug exception or external interrupt. Subsequent SMI# requests are not acknowledged while the processor is in SMM. The first SMI# interrupt request that occurs while the processor is in SMM is latched, and serviced when the processor exits SMM with the RSM instruction. Only one SMI# will be latched by the processor while it is in SMM.

When the processor invokes SMM, the processor core registers are initialized as shown in Table 8-3.

Table 8-3. SMM Initial Processor Core Register Settings

Register	Contents
General Purpose Registers	Unpredictable
EFLAGS	00000002H
EIP	00008000H
CS Selector	3000H
CS Base	SMM Base (default 30000H)
DS, ES, FS, GS, SS Selectors	0000H
DS, ES, FS, GS, SS Bases	000000000H
DS, ES, FS, GS, SS Limits	0FFFFFFFFH
CR0	Bits 0,2,3 & 31 cleared (PE, EM, TS & PG); others are unmodified
DR6	Unpredictable
DR7	00000000H

The following is a summary of the key features in the SMM environment:

1. Real mode style address calculation
2. 4-Gbyte limit checking
3. IF flag is cleared

4. NMI is disabled.
5. TF flag in EFLAGS is cleared; single step traps are disabled.
6. DR7 is cleared, except for bits 12 and 13; debug traps are disabled.
7. The RSM instruction no longer generates an invalid opcode error.
8. Default 16-bit opcode, register and stack use.

All bus arbitration (HOLD, AHOLD, BOFF#) inputs and bus sizing (BS8#, BS16#) inputs operate normally while the processor is in SMM.

8.4.2.1 Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor Environment

When the Write-Back Enhanced Intel486 processors are in Enhanced Bus Mode, SMI# has greater priority than debug exceptions and external interrupts, except for FLUSH# and SRESET. (See section 4.8.6.)

8.4.3 EXECUTING SYSTEM MANAGEMENT MODE HANDLER

The processor begins execution of the SMM handler at offset 8000H in the CS segment. The CS Base is initially 30000H; however, the CS Base can be changed by using the SMM Base relocation feature.

When the SMM handler is invoked, the processor's PE and PG bits in CR0 are reset to 0. The processor is in an environment similar to Real mode, but without the 64-Kbyte limit checking. However, the default operand size and the default address size are set to 16 bits.

The EM bit is cleared so that no exceptions are generated. (If the SMM was entered from Protected mode, the Real mode interrupt and exception support is not available.) The SMI# handler should not use Floating-Point unit instructions until the FPU is properly detected (within the SMI# handler) and the exception support is initialized.

Because the segment bases (other than CS) are cleared to 0 and the segment limits are set to 4 Gbytes, the address space may be treated as a single flat 4-Gbyte linear space that is unsegmented. The processor is still in Real mode and when a segment selector is loaded with a 16-bit value, that value is then shifted left by 4 bits and loaded into the segment base cache. The limits and attributes are not modified.

In SMM, the processor can access or jump anywhere within the 4-Gbyte logical address space. The processor can also indirectly access or perform a near jump anywhere within the 4-Gbyte logical address space.

8.4.3.1 Exceptions and Interrupts within System Management Mode

When the processor enters SMM, it disables INTR interrupts, debug and single-step traps by clearing the EFLAGS, DR6 and DR7 registers. This is done to prevent a debug application from accidentally breaking into an SMM handler. This is necessary because the SMM handler operates from a distinct address space (SMRAM), and hence, the debug trap will not represent the normal system memory space.

If an SMM handler wishes to use the debug trap feature of the processor to debug SMM handler code, it must first ensure that an SMM-compliant debug handler is available. The SMM handler must also ensure DR0–DR3 is saved to be restored later. The debug registers DR0–DR3 and DR7 must then be initialized with the appropriate values.

If the processor wishes to use the single-step feature of the processor, it must ensure that an SMM-compliant single-step handler is available and then set the trap flag in the EFLAGS register.

If the system design requires the processor to respond to hardware INTR requests while in SMM, it must ensure that an SMM-compliant interrupt handler is available and then set the interrupt flag in the EFLAGS register (using the STI instruction). Software interrupts are not blocked upon entry to SMM, and the system software designer must provide an SMM compliant-interrupt handler before attempting to execute any software interrupt instructions. Note that in SMM mode, the interrupt vector table has the same properties and location as the Real mode vector table.

NMI interrupts are blocked upon entry to the SMM handler. If an NMI request occurs during the SMM handler, it is latched and serviced after the processor exits SMM. Only one NMI request will be latched during the SMM handler. If an NMI request is pending when the processor executes the RSM instruction, the NMI is serviced before the next instruction of the interrupted code sequence.

Although NMI requests are blocked when the processor enters SMM, they may be enabled through software by executing an IRET instruction. If the SMM handler requires the use of NMI interrupts, it should invoke a dummy interrupt service routine for the purpose of executing an IRET instruction. Once an IRET instruction is executed, NMI interrupt requests are serviced in the same "real mod" manner in which they are handled outside of SMM.

8.5 SMM Features

8.5.1 SMM REVISION IDENTIFIER

The SMM revision identifier is used to indicate the version of SMM and the SMM extensions that are supported by the processor. The SMM revision identifier is written during SMM entry and can be examined in SMRAM space at Register Offset 7EFCH. The lower word of the SMM revision identifier refers to the version of the base SMM architecture. The upper word of the SMM revision identifier refers to the extensions available. (See Figure 8-7.)

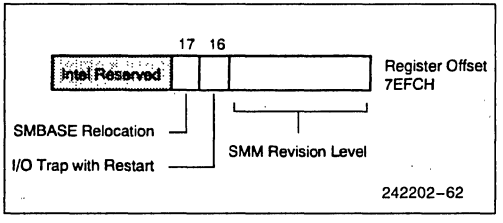


Figure 8-7. SMM Revision Identifier

Table 8-4. Bit Values for SMM Revision Identifier

Bits	Value	Comments
16	0	Processor does not support I/O Trap Restart
16	1	Processor supports I/O Trap Restart
17	0	Processor does not support SMBASE relocation
17	1	Processor supports SMBASE relocation

Bit 16 of the SMM revision identifier is used to indicate to the SMM handler that this processor supports the SMM I/O trap extension. If this bit is high, then this processor supports the SMM I/O trap extension. If this bit is low, then this processor does not support I/O trapping using the I/O trap slot mechanism. (See Table 8-4.)

Bit 17 of this slot indicates whether the processor supports relocation of the SMM jump vector and the SMRAM base address. (See Table 8-4.)

The Intel486 processor supports both the I/O Trap Restart and the SMBASE relocation features.

8.5.2 AUTO HALT RESTART

The Auto HALT Restart slot at register offset (word location) 7F02H in SMRAM indicates to the SMM handler that the SMI# interrupted the processor during a HALT state (bit 0 of slot 7F02H is set to 1 if the previous instruction was a HALT). If the SMI# did not interrupt the processor in a HALT state, then the SMI# microcode will set bit 0 of the Auto HALT Restart slot to a value of 0. If the previous instruction was a HALT, the SMM handler can choose to either set or reset bit 0. If this bit is set to 1, the RSM micro code execution will force the processor to re-enter the HALT state. If this bit is set to 0 when the RSM instruction is executed, the processor will continue execution with the instruction just after the interrupted HALT instruction. Note that if the interrupted instruction was not a HALT instruction (bit 0 is set to 0 in the Auto HALT Restart slot upon SMM entry), setting bit 0 to 1 will cause unpredictable behavior when the RSM instruction is executed. (See Figure 8-8 and Table 8-5.)

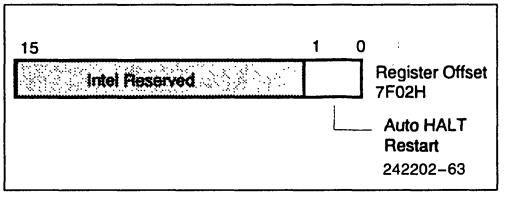


Figure 8-8. Auto HALT Restart

Table 8-5. Bit Values for Auto HALT Restart

Value of Bit 0 at Entry	Value of Bit 0 at Exit	Comments
0	0	Returns to next instruction in interrupted program.
0	1	Unpredictable
1	0	Returns to next instruction after HALT
1	1	Returns to HALT state

If the HALT instruction is restarted, the processor will generate a memory access to fetch the HALT instruction (if it is not in the internal cache), and execute a HALT bus cycle.

8.5.3 I/O INSTRUCTION RESTART

The I/O instruction restart slot (register offset 7F00H in SMRAM) gives the SMM handler the option of causing the RSM instruction to automatically re-execute the interrupted I/O instruction. When the RSM instruction is executed, if the I/O instruction restart slot contains the value 0FFH, then the processor will automatically re-execute the I/O instruction that the SMI# trapped. If the I/O instruction restart slot contains the value 00H when the RSM instruction is executed, then the processor will not re-execute the I/O instruction. The processor automatically initializes the I/O instruction restart slot to 00H during SMM entry. The I/O instruction restart slot should be written only when the processor has generated an SMI# on an I/O instruction boundary. Processor operation is unpredictable when the I/O instruction restart slot is set when the processor is servicing an SMI# that originated on a non-I/O instruction boundary. (See Figure 8-9 and Table 8-6.)

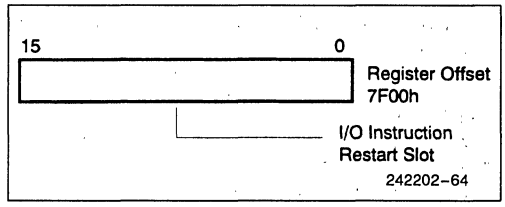


Figure 8-9. I/O Instruction Resta. t

Table 8-6 I/O Instruction Restart Value

Value at Entry	Value at Exit	Comments
00H	00H	Do not restart trapped I/O instruction
00H	0FFH	Restart trapped I/O instruction

If the system executes back-to-back SMI# requests, the second SMM handler must not set the I/O instruction restart slot (see section 8.6.6 "Nested SMI#s and I/O Restart").

8.5.4 SMM BASE RELOCATION

The Intel486 processor provides a control register, SMBASE. The address space used as SMRAM can be modified by changing the SMBASE register before exiting an SMI# handler routine. SMBASE can be changed to any 32K aligned value (values that are not 32K aligned will cause the processor to enter the shutdown state when executing the RSM instruction). SMBASE is set to the default value of 30000H on RESET, but is not changed on SRESET. If the SMBASE register is changed during an SMM handler, all subsequent SMI# requests will initiate a state save at the new SMBASE. (See Figure 8-10.)

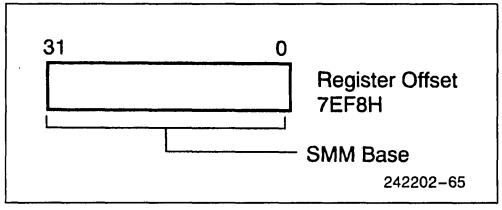


Figure 8-10. SMM Base Location

The SMBASE slot in the SMM state save area is a feature used to indicate and change the SMI# jump vector location and the SMRAM save area. When bit 17 of the SMM Revision Identifier is set then this feature exists and the SMRAM base and consequently the jump vector are as indicated by the SMM Base slot. During the execution of the RSM instruction, the processor will read this slot and initialize the processor to use the new SMBASE during the next SMI#. During an SMI#, the processor will do its context save to the new SMRAM area pointed to by the SMBASE, store the current SMBASE in the SMM Base slot (offset 7EF8H), and then start execution of the new jump vector based on the current SMBASE.

The SMBASE must be a 32-Kbyte aligned, 32-bit integer that indicates a base address for the SMRAM context save area and the SMI# jump vector. For example when the processor first powers up, the minimum SMRAM area is from 38000H–3FFFFH. The default SMBASE is 30000H. Hence the starting address of the jump vector is calculated by:

SMBASE + 8000H

While the starting address for the SMRAM state save area is calculated by:

SMM Base + [8000H + 7FFFH]

Hence, when this feature is enabled, the SMRAM register map is addressed according to the above formulas. (See Figure 8-11.)

To change the SMRAM base address and SMM jump vector location, the SMM handler should modify the SMBASE slot. Upon executing an RSM instruction, the processor will read the SMBASE slot and store it internally. Upon recognition of the next SMI# request, the processor will use the new SMBASE slot for the SMRAM dump and SMI# jump vector.

If the modified SMBASE slot does not contain a 32-Kbyte aligned value, the RSM microcode will cause the processor to enter the shutdown state.

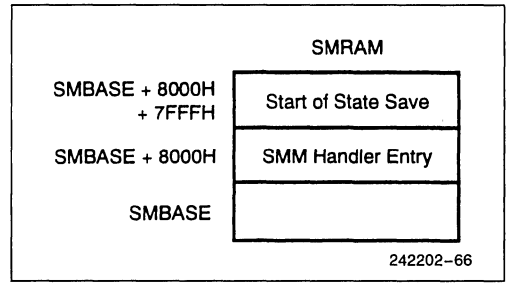


Figure 8-11. SMRAM Usage

8.6 SMM System Design Considerations

8.6.1 SMRAM INTERFACE

The hardware designed to control the SMRAM space must follow these guidelines:

1. A provision should be made to allow for initialization of SMRAM space during system boot up. This initialization of SMRAM space must happen before the first occurrence of an SMI# interrupt. Initializing the SMRAM space must include installation of an SMM handler, and may include installation of related data structures necessary for particular SMM applications. The memory controller providing the interface to the SMRAM should provide a means for the initialization code to manually open the SMRAM space.
2. A minimum initial SMRAM address space of 38000H-3FFFFH should be decoded by the memory controller.
3. Alternate bus masters (such as DMA controllers) should not be allowed to access SMRAM space. Only the processor, either through SMI# or during initialization, should be allowed access to SMRAM.
4. In order to implement a zero-volt suspend function, the system must have access to all of normal system memory from within an SMM handler routine. If the SMRAM is going to overlay normal system memory, there must be a method of accessing any system memory that is located underneath SMRAM.

There are two potential schemes for locating the SMRAM, either overlaid to an address space on top of normal system memory, or placed in a distinct address space. (See Figure 8-12.) When SMRAM is overlaid on the top of normal system memory, the processor output signal SMIACT# must be used to distinguish SMRAM from main system memory. Additionally, if the overlaid normal memory is cacheable, both the processor internal cache and any second level caches must be empty before the first read of an SMM handler routine. If the SMM memory is cacheable, the caches must be empty before the first read of normal memory following an SMM handler routine. This is done by flushing the caches, and is required to maintain cache coherency. When the default SMRAM location is used, SMRAM is overlaid on top of system main memory (at 38000H through 3FFFFH).

If SMRAM is located in its own distinct memory space, which can be completely decoded with only the processor address signals, it is said to be non-overlaid. In this case, there are no new requirements for maintaining cache coherency.

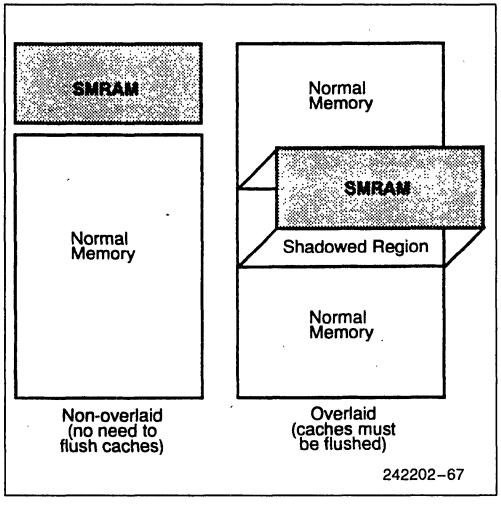


Figure 8-12. SMRAM Location

8.6.2 CACHE FLUSHES

The processor does not unconditionally flush its cache before entering SMM (this option is left to the system designer). If SMRAM is shadowed in a cacheable memory area that is visible to the application or operating system, it is necessary for the system to empty both the processor cache and any second level cache before entering SMM. That is, if SMRAM is in the same physical address location as the normal cacheable memory space, then an SMM read may hit the cache which would contain normal memory space code/data. If the SMM memory is cacheable, the normal read cycles after SMM may hit the cache, which may contain SMM code/data. In this case the cache should be empty before the first memory read cycle during SMM and before the first normal cycle after exiting SMM. (See Figure 8-13.)

The FLUSH# and KEN# signals can be used to ensure cache coherency when switching between normal and SMM modes. Cache flushing during SMM entry is accomplished by asserting the FLUSH# pin when SMI# is driven active. Cache flushing during SMM exit is accomplished by asserting the FLUSH# pin after the SMIACT# pin is deasserted (within 1 CLK). To guarantee this behavior, the constraints on setup and hold timings on the interaction of FLUSH# and SMIACT# as specified for a processor should be followed.

If the SMRAM area is overlaid over normal memory and if the system designer does not want to flush the caches upon leaving SMM then references to the SMRAM area should not be cached. It is the obligation of the system designer to ensure that the KEN# pin is sampled inactive during all references to the SMRAM area. Figures 8-14 and 8-15 illustrate a cached and non-cached SMM using FLUSH# and KEN#.

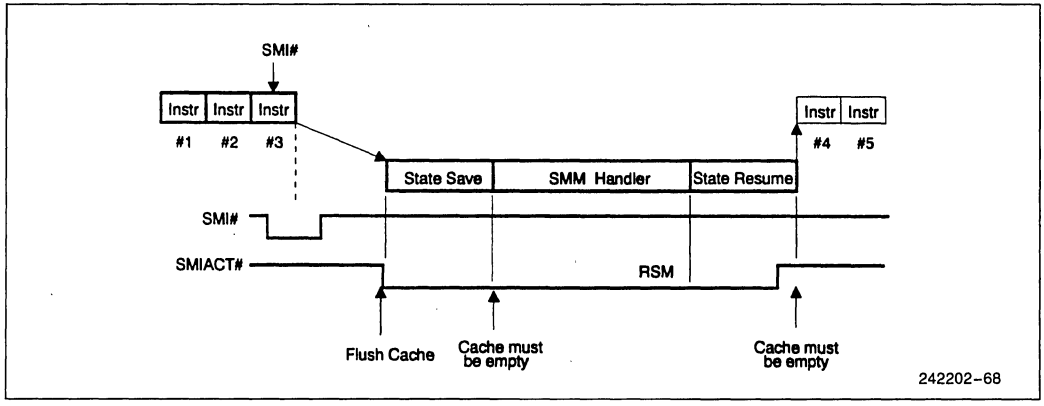


Figure 8-13. FLUSH# Mechanism during SMM

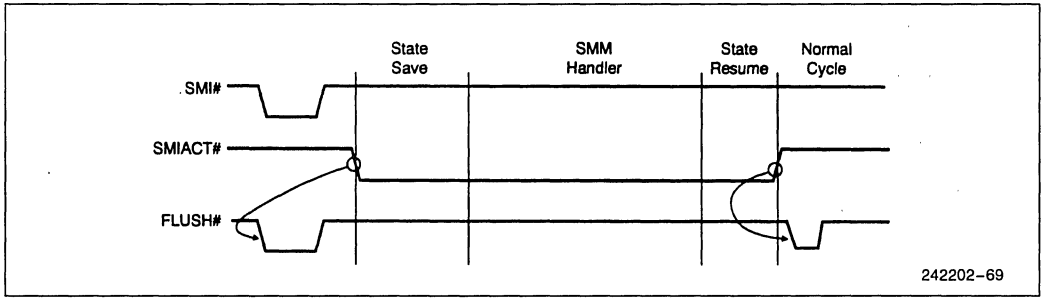


Figure 8-14. Cached SMM

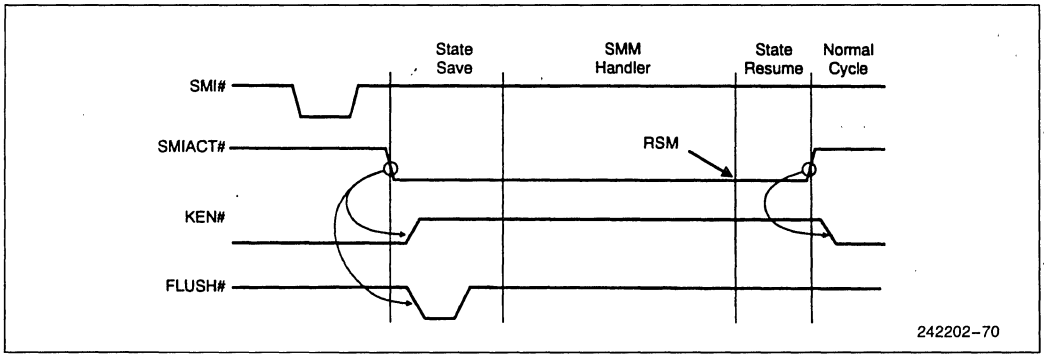


Figure 8-15. Non-Cached SMM

8.6.2.1 Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor System Management Mode and Cache Flushing

Regardless of the on-chip cache mode (i.e., either write-through or write-back) it is recommended that SMRAM be non-overlaid. This provides the greatest freedom for caching of both SMRAM and normal memory, provides a simplified memory controller design, and eliminates the performance penalty of flushing.

In general, cache flushing is not required when the SMRAM and normal memory are not overlaid. Table 8-7 gives the cache flushing requirements for entering and exiting SMM, when the SMRAM is not overlaid with normal memory space.

SMRAM can not be cached as write-back lines. If SMRAM is cached, it should be cached only as write-through lines. This is because dirty lines can not be written back to SMRAM upon exit from SMM. The de-assertion of SMIACT# signals that the processor is exiting SMM, and is used to assert FLUSH#. By the time the write back of dirty lines occurs, SMIACT# would already be inactive, so the SMRAM can no longer be decoded. When the SMRAM is cached as write-through, this problem will not occur.

Table 8-7. Cache Flushing (Non-Overlaid SMRAM)

Normal Memory Cacheable	SMRAM Cacheable	FLUSH Entering SMM
No	No	No
No	WT	No
WT	No	No
WB	No	No, but Snoop WBs must go to Normal Memory Space.
WT	WT	No
WB	WT	No, but Snoop and Replacement WBs must go to normal memory space.

Coherency requirements must be met when the normal memory is cached in write-back mode. In this case, the snoop and replacement write-backs that occur during SMM must go to normal memory, even though SMIACT# is active. This requirement is compatible with SMM security requirements, because these write backs can not decode the SMRAM, and the memory system must be able to handle this situation properly.

If SMRAM is overlaid with normal memory space, additional system design features are needed to ensure that cache coherency is maintained. Table 8-8 lists the cache flushing requirements for entering and exiting the SMM when the SMRAM is overlaid with normal memory space.

Table 8-8. Cache Flushing (Overlaid SMRAM)

Normal Memory Cacheable	SMRAM Cacheable	FLUSH Entering SMM	FLUSH Exiting SMM
No	No	No	No
No	WT	No	Yes
WT or WB	No	Yes	No
WT or WB	WT	Yes	Yes

If SMI# and FLUSH# are asserted together, the Write-Back Enhanced Intel486 processors guarantee that the FLUSH# will be recognized first, followed by the SMI#. If the cache is configured in the write-back mode, the modified lines will be written back to the normal user space, followed by the two special cycles. The SMI# will then be recognized and the transition to SMM will occur, as shown in Figure 8-16.

Cache flushing during SMM exit is accomplished by asserting the FLUSH# pin after the SMIACT# pin is de-asserted (within 1 CLK). To guarantee this behavior, the constraints on setup and hold timings on the interaction of FLUSH# and SMIACT# as specified for the Write-Back Enhanced Intel486 processors should be followed.

The WBINVD instruction should not be used to flush the cache when exiting SMM. Instead, the FLUSH# pin should be asserted after the SMIACT# pin is de-asserted (within 1 CLK). The cache coherency requirements associated with SMM and write-through vs. write-back caches apply to second level cache control designs as well. The appropriate second level cache flushing is also required upon entering and exiting the SMM.

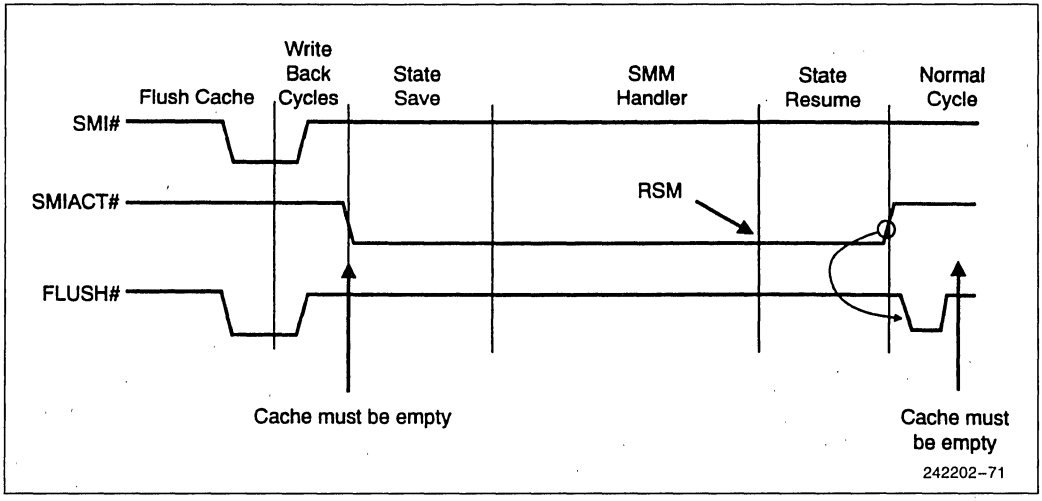


Figure 8-16. Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor Cache Flushing for Overlaid SMRAM upon Entry and Exit of Cached SMM

Snoops During SMM

Snoops cycles are allowed during SMM. However, because the SMRAM is always cached as a write-through, there can never be a snoop hit to a modified line in the SMRAM address space. Consequently, if there is a snoop hit to a modified line, it will correspond to the normal address space. In this case, even though SMIACT# is asserted, the memory controller must drive the snoop write-back cycle to the normal memory space and not to the SMRAM address space.

If the overlaid normal memory is cacheable, FLUSH# must be asserted when entering SMM, causing all modified lines of normal memory to be written back. As a result, there can not be a snoop hit to a modified line in the cacheable normal memory space that is overlaid with the SMRAM space.

If the overlaid normal memory is not cacheable, no flushing is necessary when entering SMM. If normal memory is not overlaid with SMRAM, no flushing is required upon entering SMM and it is possible that a snoop can hit a modified line cached from anywhere in normal memory space while the processor is in SMM.

8.6.3 A20M# PIN AND SMBASE RELOCATION

Systems based on a PC-compatible architecture contain a feature that enables the processor address bit A20 to be forced to 0. This limits physical memory to a maximum of 1 Mbyte, and is provided to ensure compatibility with those programs that relied on the physical address wrap around functionality of the 8088 processor. The A20M# pin on Intel486 processors provides this function. When A20M# is active, all external bus cycles will drive A20M# low, and all internal cache accesses will be performed with A20M# low.

The A20M# pin is recognized while the processor is in SMM. The functionality of the A20M# input must be recognized in the following two instances:

1. If the SMM handler needs to access system memory space above 1 Mbyte (for example, when saving memory to disk for a zero-volt suspend), the A20M# pin must be de-asserted before the memory above 1 Mbyte is addressed.
2. If SMRAM has been relocated to address space above 1 Mbyte, and A20M# is active upon entering SMM, the processor will attempt to access SMRAM at the relocated address, but with A20 low. This could cause the system to crash, because there would be no valid SMM interrupt handler at the accessed location.

In order to account for the above two situations, the system designer must ensure that A20M# is de-asserted on entry to SMM. A20M# must be driven inactive before the first cycle of the SMM state save, and must be returned to its original level after the last cycle of the SMM state restore. This can be done by blocking the assertion of A20M# whenever SMIACT# is active.

8.6.4 PROCESSOR RESET DURING SMM

The system designer should take into account the following restrictions while implementing the processor RESET logic.

1. When running software written for the 80286 processor a processor SRESET is used to switch the processor from Protected mode to Real mode. Note that SRESET has a higher interrupt priority than SMIACT#. When the processor is in SMM, the SRESET to the processor during SMM should be blocked until the processor exits SMM. SRESET must be blocked beginning from the time when SMI# is driven active and ending at least 20 CLK cycles after SMIACT# is de-asserted. Be careful not to block the global system RESET, which may be necessary to recover from a system crash.
2. During execution of the RSM instruction to exit SMM, there is a small time window between the de-assertion of SMIACT# and the completion of the RSM microcode. If SRESET is asserted during this window, it is possible that the SMRAM space will be violated. The system designer must guarantee that SRESET is blocked until at least 20 processor clock cycles after SMIACT# has been driven inactive.
3. Any request for a processor SRESET for the purpose of switching the processor from Protected mode to Real mode must be acknowledged after the processor has exited SMM. In order to maintain software transparency, the system logic must latch any SRESET signals that are blocked during SMM.

8.6.5 SMM AND SECOND LEVEL WRITE BUFFERS

Before an Intel486 processor enters SMM, it empties its internal write buffers. This is necessary so that the data in the write buffers is written to normal memory space, not SMM space. Once the processor is ready to begin writing an SMM state save to SMRAM, it asserts the SMIACT# signal. SMIACT# may be driven active by the processor before the system memory controller has had an opportunity to empty the second level write buffers.

To prevent the data from these second level write buffers from being written to the wrong location, the system memory controller needs to direct the memory write cycles to either SMM space or normal memory space. This can be accomplished by saving the status of SMIACT# along with the address for each word in the write buffers.

8.6.6 NESTED SMI#s AND I/O RESTART

Special care must be taken when executing an SMM handler for the purpose of restarting an I/O instruction. When the processor executes a RSM instruction with the I/O restart slot set, the restored EIP is modified to point to the instruction immediately preceding the SMI# request, so that the I/O instruction can be re-executed. If a new SMI# request is received while the processor is executing an SMM handler, the processor will service this SMI# request before restarting the original I/O instruction. If the I/O restart slot is set when the processor executes the RSM instruction for the second SMM handler, the RSM microcode will decrement the restored EIP again. EIP now points to an address different from the originally interrupted instruction, and the processor will begin execution of the interrupted application code at an incorrect entry point.

To prevent this from occurring, the SMM handler routine must not set the I/O restart slot during the second of two consecutive SMM handlers.

8.7 SMM Software Considerations

8.7.1 SMM CODE CONSIDERATIONS

The default operand size and the default address size are 16 bits; however, operand-size override and address-size override prefixes can be used as needed to directly access data anywhere within the 4-Gbyte logical address space.

With operand-size override prefixes, the SMM handler can use jumps, calls, and returns, to transfer control to any location within the 4-Gbyte space. Note, however, the following restrictions:

- Any control transfer that does not have an operand-size override prefix truncates EIP to 16 low-order bits.

- Due to the Real mode style of base-address formation, a far jump or call cannot transfer control to a segment with a base address of more than 20 bits (one megabyte).

8.7.2 EXCEPTION HANDLING

Upon entry into SMM, external interrupts that require handlers are disabled (the IF bit in the EFLAGS is cleared). This is necessary because, while the processor is in SMM, it is running in a separate memory space. Consequently the vectors stored in the interrupt descriptor table (IDT) for the prior mode are not applicable. Before allowing exception handling (or software interrupts), the SMM program must initialize new interrupt and exception vectors. The interrupt vector table for SMM has the same format as for Real mode. Until the interrupt vector table is correctly initialized, the SMM handler must not generate an exception (or software interrupt). Even though hardware interrupts are disabled, exceptions and software interrupts can still occur. Only a correctly written SMM handler can prevent internal exceptions. When new exception vectors are initialized, internal exceptions can be serviced. The following are the restrictions:

1. Due to the Real mode style of base address formation, an interrupt or exception cannot transfer control to a segment with a base address of more that 20 bits.
2. An interrupt or exception cannot transfer control to a segment offset of more than 16 bits (64 Kbytes).
3. If exceptions or interrupts are allowed to occur, only the low order 16 bits of the return address (EIP) are pushed onto the stack. If the offset of the interrupted procedure is greater than 64 Kbytes, it is not possible for the interrupt/exception handler to return control to that procedure. (One work-around could be to perform software adjustment of the return address on the stack.)
4. The SMBASE Relocation feature affects the way the processor will return from an interrupt or exception during an SMI# handler.

8.7.3 HALT DURING SMM

HALT should not be executed during SMM, unless interrupts have been enabled (see section 8.7.2. "Exception Handling"). Interrupts are disabled in SMM and INTR, NMI, and SMI# are the only events that take the processor out of HALT.

8.7.4 RELOCATING SMRAM TO AN ADDRESS ABOVE ONE MEGABYTE

Within SMM (or Real mode), the segment base registers can only be updated by changing the segment register. The segment registers contain only 16 bits, which allows only 20 bits to be used for a segment base address (the segment register is shifted left four bits to determine the segment base address). If SMRAM is relocated to an address above one megabyte, the segment registers can no longer be initialized to point to SMRAM.

These areas can still be accessed by using address override prefixes to generate an offset to the correct address. For example, if the SMBASE has been relocated immediately below 16M, the DS and ES registers are still initialized to 0000 0000H. We can still access data in SMRAM by using 32-bit displacement registers:

```
mov    esi, 00FFxxxxH      ;64K segment
                           ;immediately
                           ;below 16M
mov    ax,ds:[esi]
```

4

9.0 HARDWARE INTERFACE

9.1 Introduction

The Intel486 processor has separate parallel buses for data and addresses. The bidirectional data bus is 32 bits in width. The address bus consists of two components: 30 address lines (A2–A31) and 4-byte enable lines (BE0#–BE3#). The address lines form the upper 30 bits of the address and the byte enables select individual bytes within a 4-byte location. The address lines are bidirectional for use in cache line invalidations. (See Figure 9-1.)

The Intel486 processor's burst bus mechanism enables high-speed cache fills from external memory. Burst cycles can strobe data into the processor at a rate of one item every clock. Non-burst cycles have a maximum rate of one item every two clocks. Burst cycles are not limited to cache fills: all read bus cycles requiring more than a single data cycle can be bursted. The Write-Back Enhanced IntelDX2™ processor can also burst write cycles.

During bus hold, the Intel486 processor relinquishes control of the local bus by floating its address, data and control buses. The Intel486 processor has an address hold feature in addition to bus hold. During address hold, only the address bus is floated, the data and control buses can remain active. Address hold is used for cache line invalidations.

The Intel486 supports the IEEE 1149.1 boundary scan.

This section provides a brief description of the Intel486 processor input and output signals arranged by functional groups. The # symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When a # is not present after the signal name, the signal is active at high voltage level. The term "ready" is used to indicate that the cycle is terminated with RDY# or BRDY#.

This section and section 10, "Bus Operation," describe bus cycles and data cycles. A bus cycle is at least two-clocks long and begins with ADS# active in the first clock and RDY# and/or BRDY# active in the last clock. Data is transferred to or from the Intel486 processor during a data cycle. A bus cycle contains one or more data cycles.

9.2 Signal Descriptions

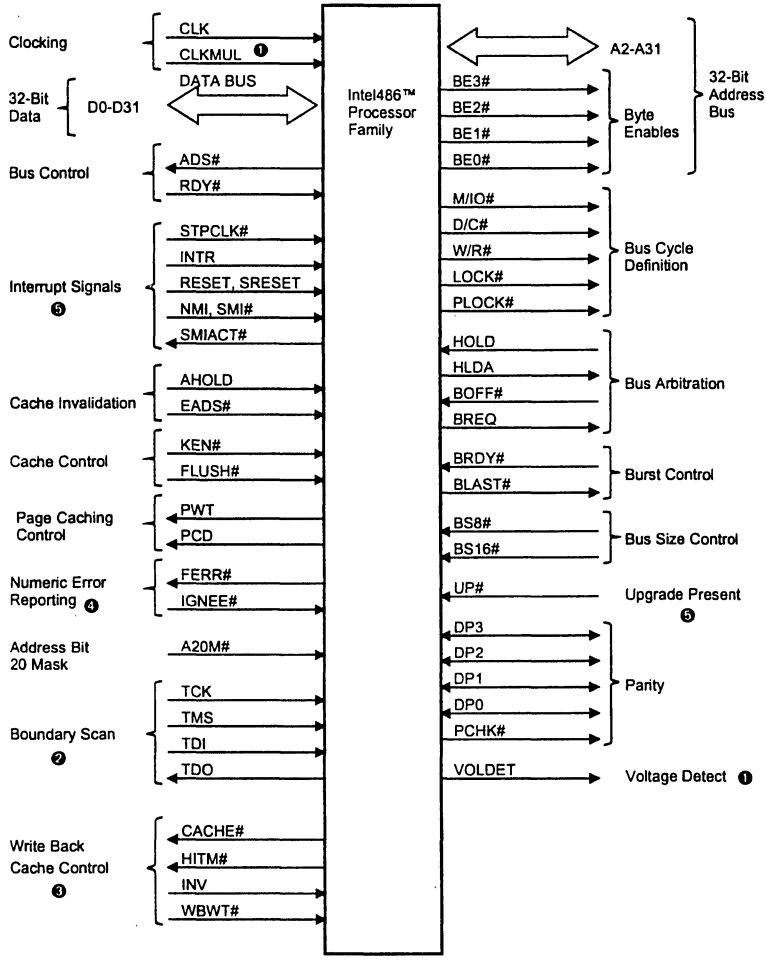
9.2.1 CLOCK (CLK)

CLK provides the fundamental timing and the internal operating frequency for the Intel486 processor. All external timing parameters are specified with respect to the **rising edge** of CLK.

The Intel486 processor can operate over a wide frequency range, however the CLK frequency cannot change rapidly while RESET is inactive. The CLK frequency must be stable for proper chip operation because a single edge of CLK is used internally to generate two phases. CLK only needs TTL levels for proper operation. Figure 9-2 illustrates the CLK waveform.

9.2.2 IntelDX4™ PROCESSOR CLOCK MULTIPLIER SELECTABLE INPUT (CLKMUL)

The IntelDX4 processor differs from the IntelDX2 processor in that it provides for two internal clock multiplier ratios: speed doubled mode and speed tripled mode. Speed doubled mode is identical to the IntelDX2 processor mode of operation where the internal core is operating at twice the external bus frequency. Selecting speed tripled mode causes the internal core frequency to operate at three times the external bus frequency. The IntelDX4 processor determines the desired clock multiplier ratio by sampling the status of the CLKMUL input during cold (power on) processor resets. **The clock multiplier ratio cannot be changed during warm resets. Also, SRESET cannot be used to select the clock multiplier ratio.**



❶ IntelDX4™ processor only.

❷ Not on Intel486™ SX processor in PGA package.

❸ Pins on Write-Back Enhanced Intel486 processors when in Enhanced Bus Mode/write back cache mode.

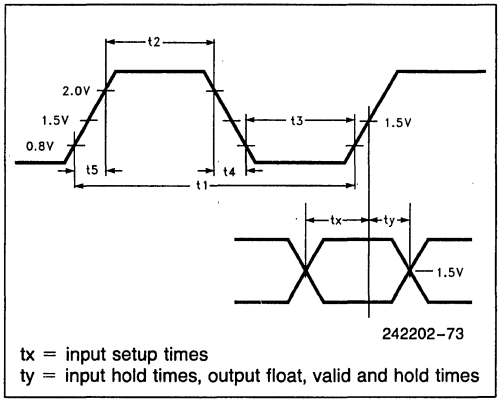
❹ Not on Intel486 SX and IntelSX2 processors.

❺ SMI#, SMIACT#, STPCLK, SRESET, UP# not on 50-MHz Intel486 DX processor.

242202-72

Figure 9-1. Functional Signal Groupings

4



tx = input setup times
ty = input hold times, output float, valid and hold times

Figure 9-2. CLK Waveform

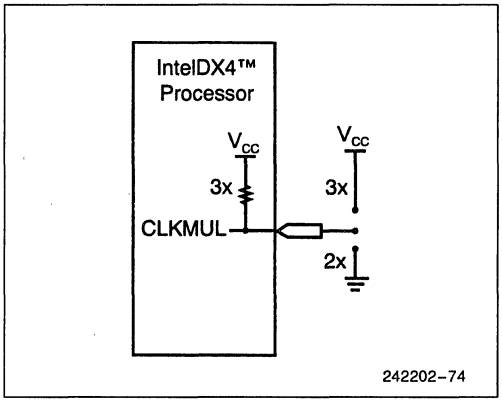


Figure 9-3. Voltage Detect (VOLDET) Sense Pin

To determine which clock multiplier is desired, the IntelDX4 processor samples the status of CLKMUL while RESET is active. If the CLKMUL input is driven low during RESET, the frequency of the core will be twice the external bus frequency (speed doubled mode). If driven high or left floating, speed tripled mode is selected. (See Table 9-1.) In order to allow maximum flexibility, CLKMUL can be jumper-configurable to either V_{CC} (speed tripled mode) or V_{SS} (speed doubled mode). (See Figure 9-3.)

Table 9-1. Clock Multiplier Selection

CLKMUL at RESET	Clock Multiplier	External Clock Freq. (MHz)	Internal Clock Freq. (MHz)
V_{CC} or Not Driven	3	25 33	75 100
V_{SS}	2	50	100

The clock multiplier selection method is fully backward compatible with Intel486 processor-based system designs. The CLKMUL signal occupies a pin which is labeled as an "INC" on other Intel486 processors. Therefore, this pin is not driven in other Intel486 processor system designs. The IntelDX4 processor contains an internal pull-up resistor on the CLKMUL signal. As shown in Table 9-1, when CLKMUL is not driven, the internal core frequency defaults to speed tripled mode.

The internal pull-up resistor on the CLKMUL pin is disabled while the IntelDX4 processor is in the Stop Grant or Stop Clock modes. This prevents a low level DC current path from drawing current while in the Stop Grant or Stop Clock states on a system with CLKMUL connected to V_{SS}.

9.2.3 ADDRESS BUS (A31–A2, BE0#–BE3#)

A31–A2 and BE0#–BE3# form the address bus and provide physical memory and I/O port addresses. The Intel486 processor is capable of addressing 4 gigabytes of physical memory space (00000000H through FFFFFFFFH), and 64 Kbytes of I/O address space (00000000H through 0000FFFFH). A31–A2 identify addresses to a 4-byte location. BE0#–BE3# identify which bytes within the 4-byte location are involved in the current transfer.

Addresses are driven back into the Intel486 processor over A31–A4 during cache line invalidations. The address lines are active HIGH. When used as inputs into the processor, A31–A4 must meet the setup and hold times, t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.

The byte enable outputs, BE0#–BE3#, determine which bytes must be driven valid for read and write cycles to external memory.

- BE3# applies to D24–D31
- BE2# applies to D16–D23
- BE1# applies to D8–D15
- BE0# applies to D0–D7

BE0#–BE3# can be decoded to generate A0, A1 and BHE# signals used in 8- and 16-bit systems (see Table 10-5). BE0#–BE3# are active LOW and are not driven during bus hold.

9.2.4 DATA LINES (D31–D0)

The bidirectional lines, D31–D0, form the data bus for the Intel486 processor D0–D7 define the least significant byte and D24–D31 the most significant byte. Data transfers to 8- or 16-bit devices are possible using the data bus sizing feature controlled by the BS8# or BS16# input pins. D31–D0 are active HIGH. For reads, D31–D0 must meet the setup and hold times, t_{22} and t_{23}. D31–D0 are not driven during read cycles and bus hold.

9.2.5 PARITY

Data Parity Input/Outputs (DP0–DP3)

DP0–DP3 are the data parity pins for the processor. There is one pin for each byte of the data bus. Even parity is generated or checked by the parity generators/checkers. Even parity means that there is an even number of HIGH inputs on the eight corresponding data bus pins and parity pin.

Data parity is generated on all write data cycles with the same timing as the data driven by the Intel486 processor. Even parity information must be driven back to the Intel486 processor on these pins with the same timing as read information to ensure that the correct parity check status is indicated by the Intel486 processor.

The values read on these pins do not affect program execution. It is the responsibility of the system to take appropriate actions if a parity error occurs.

Input signals on DP0–DP3 must meet setup and hold times t_{22} and t_{23} for proper operation.

Parity Status Output (PCHK#)

Parity status is driven on the PCHK# pin, and a parity error is indicated by this pin being LOW. PCHK# is driven by the clock after ready for read operations to indicate the parity status for the data sampled at the end of the previous clock. Parity is checked during code reads, memory reads and I/O reads. Parity is not checked during interrupt acknowledge cycles. PCHK# only checks the parity status for enabled bytes as indicated by the byte enable and bus size signals. It is valid only in the clock immediately after read data is returned to the Intel486 processor. At all other times it is inactive (HIGH). PCHK# is never floated.

Driving PCHK# is the only effect that bad input parity has on the Intel486 processor. The Intel486 processor will not vector to a bus error interrupt when bad data parity is returned. In systems that will not employ parity, PCHK# can be ignored. In systems not using parity, DP0–DP3 should be connected to V_{CC} through a pull-up resistor.

9.2.6 BUS CYCLE DEFINITION

M/IO#, D/C#, W/R# Outputs

M/IO#, D/C# and W/R# are the primary bus cycle definition signals. They are driven valid as the ADS# signal is asserted. M/IO# distinguishes between memory and I/O cycles, D/C# distinguishes between data and control cycles, and W/R# distinguishes between write and read cycles.

Bus cycle definitions as a function of M/IO#, D/C# and W/R# are given in Table 9-2. Note there is a difference between the Intel486 processor and Intel386™ processor bus cycle definitions. The halt bus cycle type has been moved to location 001 in the Intel486 processor from location 101 in the Intel386 processor. Location 101 is now reserved and will never be generated by the Intel486 processor.

Special bus cycles are discussed in section 10.2.11, "Special Bus Cycles."

Table 9-2. ADS# Initiated Bus Cycle Definitions

M/IO#	D/C#	W/R#	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Halt/Special Cycle
0	1	0	I/O Read
0	1	1	I/O Write
1	0	0	Code Read
1	0	1	Reserved
1	1	0	Memory Read
1	1	1	Memory Write

Bus Lock Output (LOCK#)

LOCK# indicates that the Intel486 processor is running a read-modify-write cycle where the external bus must not be relinquished between the read and write cycles. Read-modify-write cycles are used to implement memory-based semaphores. Multiple reads or writes can be locked.

When LOCK# is asserted, the current bus cycle is locked and the Intel486 processor should be allowed exclusive access to the system bus. LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after ready is returned indicating the last locked bus cycle.

The Intel486 processor will not acknowledge bus hold when LOCK# is asserted (though it will allow an address hold). LOCK# is active LOW and is floated during bus hold. Locked read cycles will not be transformed into cache fill cycles if KEN# is returned active. Refer to section 10.2.6, "Locked Cycles," for a detailed discussion of Locked bus cycles.

Pseudo-Lock Output (PLOCK#)

The pseudo-lock feature allows atomic reads and writes of memory operands greater than 32 bits. These operands require more than one cycle to transfer. The Intel486 processor asserts PLOCK# during segment table descriptor reads (64 bits) and cache line fills (128 bits).

When PLOCK# is asserted no other master will be given control of the bus between cycles. A bus hold request (HOLD) is not acknowledged during pseudo-locked reads and writes, with one exception. During non-cacheable non-bursted code prefetches, HOLD is recognized on memory cycle boundaries even though PLOCK# is asserted. The Intel486 processor will drive PLOCK# active until the addresses for the last bus cycle of the transaction have been driven regardless of whether BRDY# or RDY# is returned.

A pseudo-locked transfer is meaningful only if the memory operand is aligned and if it is completely contained within a single cache line.

Because PLOCK# is a function of the bus size and KEN# inputs, PLOCK# should be sampled only when the clock ready is returned. This pin is active LOW and is not driven during bus hold. Refer to section 10.2.7, "Pseudo-Locked Cycles."

9.2.6.1 PLOCK# Floating-Point Considerations

For processors with an on-chip FPU, the following must be noted for PLOCK# operation. A 64-bit Floating-Point number must be aligned to an 8-byte boundary to guarantee an atomic access. Normally PLOCK# and BLAST# are inverse of each other. However, during the first cycle of a 64-bit Floating-Point write, both PLOCK# and BLAST# will be asserted. Intel486 processors with on-chip FPUs also assert PLOCK# during Floating-Point long reads and writes (64 bits), segmentable description reads (64 bits), and code line fills (128 bits).

9.2.7 BUS CONTROL

The bus control signals allow the Intel486 processor to indicate when a bus cycle has begun, and allow other system hardware to control burst cycles, data bus width and bus cycle termination.

Address Status Output (ADS#)

The ADS# output indicates that the address and bus cycle definition signals are valid. This signal will go active in the first clock of a bus cycle and go inactive in the second and subsequent clocks of the cycle. ADS# is also inactive when the bus is idle.

ADS# is used by the external bus circuitry as the indication that the Intel486 processor has started a bus cycle. The external circuit must sample the bus cycle definition pins on the next rising edge of the clock after ADS# is driven active.

ADS# is active LOW and is not driven during bus hold.

Non-burst Ready Input (RDY#)

RDY# indicates that the current bus cycle is complete. In response to a read, RDY# indicates that the external system has presented valid data on the data pins. In response to a write request, RDY# indicates that the external system has accepted the Intel486 processor data. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. Because RDY# is sampled during address hold, data can be returned to the processor when AHOLD is active.

RDY# is active LOW, and is not provided with an internal pull-up resistor. This input must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.

9.2.8 BURST CONTROL

Burst Ready Input (BRDY#)

BRDY# performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted the Intel486 processor data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle.

During a burst cycle, BRDY# will be sampled each clock and, if active, the data presented on the data bus pins will be strobed into the Intel486 processor. ADS# is negated during the second through last data cycles in the burst, but address lines A2–A3 and byte enables will change to reflect the next data item expected by the Intel486 processor.

If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. An additional complete bus cycle will be initiated after an aborted burst cycle if the cache line fill was not complete. BRDY# is treated as a normal ready for the last data cycle in a burst transfer or for non-burstable cycles. Refer to section 10.2.2, "Multiple and Burst Cycle Bus Transfers," for burst cycle timing.

BRDY# is active LOW and is provided with a small internal pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.

Burst Last Output (BLAST#)

BLAST# indicates that the next time BRDY# is returned it will be treated as a normal RDY#, terminating the line fill or other multiple-data-cycle transfer. BLAST# is active for all bus cycles regardless of whether they are cacheable or not. This pin is active LOW and is not driven during bus hold.

9.2.9 INTERRUPT SIGNALS

The interrupt signals can interrupt or suspend execution of the processor's current instruction stream.

Reset Input (RESET)

The RESET input must be used at power-up to initialize the processor. The Reset input forces the processor to begin execution at a known state. The processor cannot begin execution of instructions until at least 1 ms after V_{CC} and CLK have reached their proper DC and AC specifications. The RESET pin should remain active during this time to ensure proper processor operation. However, for warm boot-ups RESET should remain active for at least 15 CLK periods. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

RESET will reset SMBASE to the default value of 30000H. If SMBASE relocation is not used, the RESET signal can be used as the only reset. (See section 8, "System Management Mode Architecture.")

The Intel486 processor will be placed in the Power Down Mode if UP# is sampled active at the falling edge of RESET.

Soft Reset Input (SRESET)

The SRESET (Soft RESET) input has the same functions as RESET, but does not change the SMBASE, and UP# is not sampled on the falling edge of SRESET. If SMBASE relocation is used by the system, the soft resets should be handled using the SRESET input. The SRESET signal should not be used for the cold boot-up power-on reset.

The SRESET input pin is provided to save the status of SMBASE during Intel 286 processor-compatible mode change. SRESET leaves the location of

SMBASE intact while resetting other units, including the on-chip cache. (See section 9.2.18.4, "Soft Reset," for Write-Back Enhanced Intel486 processor differences.) For compatibility, the system should use SRESET to flush the on-chip cache. The FLUSH# input pin should be used to flush the on-chip cache. SRESET should not be used to initiate test modes.

System Management Interrupt Request Input (SMI#)

SMI# is the system management mode interrupt request signal. The SMI# request is acknowledged by the SMIACT# signal. After the SMI# interrupt is recognized, the SMI# signal will be masked internally until the RSM instruction is executed and the interrupt service routine is complete. SMI# is falling-edge sensitive after internal synchronization.

The SMI# input must be held inactive for at least four clocks after it is asserted to reset the edge triggered logic. SMI# is provided with a pull-up resistor to maintain compatibility with designs which do not use this feature. SMI# is an asynchronous signal, but setup and hold times, t_{20} and t_{21}, must be met in order to guarantee recognition on a specific clock.

System Management Mode Active Output (SMIACT#)

SMIACT# indicates that the processor is operating in System Management Mode. The processor asserts SMIACT# in response to an SMI interrupt request on the SMI# pin. SMIACT# is driven active after the processor has completed all pending write cycles (including emptying the write buffers), and before the first access to SMRAM when the processor saves (writes) its state (or context) to SMRAM. SMIACT# remains active until the last access to SMRAM when the processor restores (reads) its state from SMRAM. The SMIACT# signal does not float in response to HOLD. The SMIACT# signal is used by the system logic to decode SMRAM.

Maskable Interrupt Request Input (INTR)

INTR indicates that an external interrupt has been generated. Interrupt processing is initiated if the IF flag is active in the EFLAGS register.

The Intel486 processor will generate two locked interrupt acknowledge bus cycles in response to asserting the INTR pin. An 8-bit interrupt number will be latched from an external interrupt controller at the end of the second interrupt acknowledge cycle. INTR must remain active until the interrupt acknowledges have been performed to assure program interruption. Refer to section 10.2.10, "Interrupt Acknowledge," for a detailed discussion of interrupt acknowledge cycles.

The INTR pin is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but the INTR setup and hold times, t_{20} and t_{21}, must be met to assure recognition on any specific clock.

Non-maskable Interrupt Request Input (NMI)

NMI is the non-maskable interrupt request signal. Asserting NMI causes an interrupt with an internally supplied vector value of 2. External interrupt acknowledge cycles are not generated because the NMI interrupt vector is internally generated. When NMI processing begins, the NMI signal will be masked internally until the IRET instruction is executed.

NMI is rising edge sensitive after internal synchronization. NMI must be held LOW for at least four CLK periods before this rising edge for proper operation. NMI is not provided with an internal pull-down resistor. NMI is asynchronous but setup and hold times, t_{20} and t_{21} must be met to assure recognition on any specific clock.

Stop Clock Interrupt Request Input (STPCLK#)

The Intel486 processor provides an interrupt mechanism, STPCLK#, that allows system hardware to control the power consumption of the processor by stopping the internal clock (output of the PLL) to the processor core in a controlled manner. This low-power state is called the Stop Grant state. In addition, the STPCLK# interrupt allows the system to change the input frequency within the specified range or completely stop the CLK input frequency (input to the PLL). If the CLK input is completely stopped, the processor enters into the Stop Clock state–the lowest power state. If the frequency is changed or stopped, the Intel486 processor will not return to the Stop Grant state until the CLK input has been running at a constant frequency for the time period necessary to stabilize the PLL (minimum of 1 ms).

The Intel486 processor will generate a Stop Grant bus cycle in response to the STPCLK# interrupt request. STPCLK# is active LOW and is provided

with an internal pull-up resistor. STPCLK# is an asynchronous signal, but must remain active until the processor issues the Stop Grant bus cycle. (Refer to section 10.2.11.3, "Stop Grant Indication Cycle.")

9.2.10 BUS ARBITRATION SIGNALS

This section describes the mechanism by which the processor relinquishes control of its local bus when requested by another bus master.

Bus Request Output (BREQ)

The Intel486 processor asserts BREQ whenever a bus cycle is pending internally. Thus, BREQ is always asserted in the first clock of a bus cycle, along with ADS#. Furthermore, if the Intel486 processor is currently not driving the bus (due to HOLD, AHOLD, or BOFF#), BREQ is asserted in the same clock that ADS# would have been asserted if the Intel486 processor were driving the bus. After the first clock of the bus cycle, BREQ may change state. It will be asserted if additional cycles are necessary to complete a transfer (via BS8#, BS16#, KEN#), or if more cycles are pending internally. However, if no additional cycles are necessary to complete the current transfer, BREQ can be negated before ready comes back for the current cycle. External logic can use the BREQ signal to arbitrate among multiple processors. This pin is driven regardless of the state of bus hold or address hold. BREQ is active HIGH and is never floated. During a hold state, internal events may cause BREQ to be de-asserted prior to any bus cycles.

Bus Hold Request Input (HOLD)

HOLD allows another bus master complete control of the Intel486 processor bus. The Intel486 processor will respond to an active HOLD signal by asserting HLDA and placing most of its output and input/output pins in a high impedance state (floated) after completing its current bus cycle, burst cycle, or sequence of locked cycles. In addition, if the Intel486 processor receives a HOLD request while performing a code fetch, and that cycle is backed off (BOFF#), the Intel486 processor will recognize HOLD before restarting the cycle. The code fetch can be non-cacheable or cacheable and non-burst-ed or bursted. The BREQ, HLDA, PCHK# and FERR# pins are not floated during bus hold. The Intel486 processor will maintain its bus in this state until the HOLD is de-asserted. Refer to section 10.2.9, "Bus Hold," for timing diagrams for bus hold cycles and HOLD request acknowledge during BOFF#.

Unlike the Intel386 processor, the Intel486 processor will recognize HOLD during reset. Pull-up resistors are not provided for the outputs that are floated in response to HOLD. HOLD is active HIGH and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper chip operation.

Bus Hold Acknowledge Output (HLDA)

HLDA indicates that the Intel486 processor has given the bus to another local bus master. HLDA goes active in response to a hold request presented on the HOLD pin. HLDA is driven active in the same clock that the Intel486 processor floats its bus.

HLDA will be driven inactive when leaving bus hold and the Intel486 processor will resume driving the bus. The Intel486 processor will not cease internal activity during bus hold because the internal cache will satisfy the majority of bus requests. HLDA is active HIGH and remains driven during bus hold.

Backoff Input (BOFF#)

Asserting the BOFF# input forces the Intel486 processor to release control of its bus in the next clock. The pins floated are exactly the same as in response to HOLD. The response to BOFF# differs from the response to HOLD in two ways: First, the bus is floated immediately in response to BOFF# while the Intel486 processor completes the current bus cycle before floating its bus in response to HOLD. Second the Intel486 processor does not assert HLDA in response to BOFF#.

The Intel486 processor remains in bus hold until BOFF# is negated. Upon negation, the Intel486 processor restarts the bus cycle aborted when BOFF# was asserted. To the internal execution engine the effect of BOFF# is the same as inserting a few wait states to the original cycle. Refer to section 10.2.12, "Bus Cycle Restart," for a description of bus cycle restart.

Any data returned to the Intel486 processor while BOFF# is asserted is ignored. BOFF# has higher priority than RDY# or BRDY#. If both BOFF# and ready are returned in the same clock, BOFF# takes effect. If BOFF# is asserted while the bus is idle, the Intel486 processor will float its bus in the next clock. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper chip operation.

9.2.11 CACHE INVALIDATION

The AHOLD and EADS# inputs are used during cache invalidation cycles. AHOLD conditions the Intel486 processor address lines, A4–A31, to accept an address input. EADS# indicates that an external address is actually valid on the address inputs. Activating EADS# will cause the Intel486 processor to read the external address bus and perform an internal cache invalidation cycle to the address indicated. Refer to section 10.2.8, "Invalidate Cycle," or cache invalidation cycle timing.

Address Hold Request Input (AHOLD)

AHOLD is the address hold request. It allows another bus master access to the Intel486 processor address bus for performing an internal cache invalidation cycle. Asserting AHOLD will force the Intel486 processor to stop driving its address bus in the next clock. While AHOLD is active only the address bus will be floated, the remainder of the bus can remain active. For example, data can be returned for a previously specified bus cycle when AHOLD is active. The Intel486 processor will not initiate another bus cycle during address hold. Because the Intel486 processor floats its bus immediately in response to AHOLD, an address hold acknowledge is not required. If AHOLD is asserted while a bus cycle is in progress, and no readies are returned during the time AHOLD is asserted, the Intel486 processor will redrive the same address (that it originally sent out) once AHOLD is negated.

AHOLD is recognized during reset. Because the entire cache is invalidated by reset, any invalidation cycles run during reset will be unnecessary. AHOLD is active HIGH and is provided with a small internal pull-down resistor. It must satisfy the setup and hold times t_{18} and t_{19} for proper chip operation. AHOLD also determines whether or not the built in self test features of the Intel486 processor will be exercised on assertion of RESET. (See section 11.1, "Built-In Self Test.")

External Address Valid Input (EADS#)

EADS# indicates that a valid external address has been driven onto the Intel486 processor address pins. This address will be used to perform an internal cache invalidation cycle. The external address will be checked with the current cache contents. If the address specified matches any areas in the cache, that area will immediately be invalidated.

An invalidation cycle may be run by asserting EADS# regardless of the state of AHOLD, HOLD and BOFF#. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy the setup and hold times t_{12} and t_{13} for proper chip operation.

9.2.12 CACHE CONTROL

Cache Enable Input (KEN#)

KEN# is the cache enable pin. KEN# is used to determine whether the data being returned by the current cycle is cacheable. When KEN# is active and the Intel486 processor generates a cycle that can be cached (most any memory read cycle), the cycle will be transformed into a cache line fill cycle.

A cache line is 16 bytes long. During the first cycle of a cache line fill the byte-enable pins should be ignored and data should be returned as if all four byte enables were asserted. The Intel486 processor will run between 4 and 16 contiguous bus cycles to fill the line depending on the bus data width selected by BS8# and BS16#. Refer to section 10.2.3, "Cacheable Cycles," for a description of cache line fill cycles.

The KEN# input is active LOW and is provided with a small internal pull-up resistor. It must satisfy the setup and hold times t_{14} and t_{15} for proper chip operation.

Cache Flush Input (FLUSH#)

The FLUSH# input forces the Intel486 processor to flush its entire internal cache. FLUSH# is active LOW and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition on any specific clock.

FLUSH# also determines whether or not the tri-state test mode of the Intel486 processor will be invoked on assertion of RESET. (See section 11.4, "Tri-State Output Test Mode.")

9.2.13 PAGE CACHEABILITY (PWT, PCD)

The PWT and PCD output signals correspond to two user attribute bits in the page table entry. When paging is enabled, PWT and PCD correspond to bits 3 and 4 of the page table entry, respectively. For cycles that are not paged when paging is enabled (for example I/O cycles), PWT and PCD correspond to bits 3 and 4 in control register 3. When paging is disabled, the Intel486 processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT.

PCD is masked by the CD (cache disable) bit in control register 0 (CR0). When CD = 1 (cache line fills disabled), the Intel486 processor forces PCD HIGH. When CD = 0, PCD is driven with the value of the page table entry/directory.

The purpose of PCD is to provide a cacheable/non-cacheable indication on a page-by-page basis. The Intel486 processor will not perform a cache fill to any page in which bit 4 of the page table entry is set. PWT corresponds to the write-back bit and can be used by an external cache to provide this functionality. PCD and PWT bits are assigned to be zero during real mode or whenever paging is disabled. Refer to section 7.6, "Page Cacheability," for a discussion of non-cacheable pages.

PCD and PWT have the same timing as the cycle definition pins (M/IO#, D/C#, W/R#). PCD and PWT are active HIGH and are not driven during bus hold.

NOTE:
The PWT and PCD bits function differently in the write-back mode of the Write-Back Enhanced Intel486 processors. (See section 7.6.1.)

9.2.14 UPGRADE PRESENT (UP#)

The **Upgrade Present** input detects the presence of the upgrade processor, then powers down the core, and tri-states all outputs of the original processor, so that the original processor consumes very low current. This state is known as Upgrade Power Down Mode. UP# is active LOW and sampled at all times, including after power-up and during reset.

9.2.15 NUMERIC ERROR REPORTING (FERR#, IGNNE#)

To allow PC-type Floating-Point error reporting, Intel486 DX, IntelDX2, and IntelDX4 processors provide two pins, FERR# and IGNNE#.

Floating-Point Error Output (FERR#)

The processor asserts FERR# whenever an unmasked Floating-Point error is encountered. FERR# is similar to the ERROR# pin on the Intel387 math coprocessor. FERR# can be used by external logic for PC-type Floating-Point error reporting in systems with an Intel486 DX, IntelDX2, or IntelDX4 processor. FERR# is active LOW and is not floated during bus hold.

In some cases, FERR# is asserted when the next Floating-Point instruction is encountered. In other cases, it is asserted before the next Floating-Point instruction is encountered, depending on the execution state of the instruction that caused the exception.

The following class of Floating-Point exceptions drive FERR# at the time the exception occurs (i.e., before encountering the next Floating-Point instruction):

1. The stack fault, invalid operation, and denormal exceptions on all transcendental instructions, integer arithmetic instructions, FSQRT, FSCALE, FPREM(1), FXTRACT, FBLD, and FBSTP.
2. Any exceptions on store instructions (including integer store instructions).

The following class of Floating-Point exceptions drive FERR# only after encountering the next Floating-Point instruction:

1. Exceptions other than on all transcendental instructions, integer arithmetic instructions, FSQRT, FSCALE, FPREM(1), FXTRACT, FBLD, and FBSTP.
2. Any exception on all basic arithmetic, load, compare, and control instructions (i.e., all other instructions).

In the event of a pending unmasked Floating-Point exception the FNINIT, FNCLEX, FNSTENV, FNSAVE, FNSTSW and FNSTCW instructions assert the FERR# pin. Shortly after the assertion of the pin, an interrupt window is opened during which the processor samples and services interrupts, if any. If no interrupts are sampled within this window, the processor will then execute these instructions with the pending unmasked exception. However, for the FNCLEX, FNINIT, FNSTENV and FNSAVE instructions, the FERR# pin is deasserted to enable the execution of these instructions. For details, please refer to section 19.3 in the *Intel486™ Processor Family Programmer's Reference Manual,* revision 3.

Ignore Numeric Error Input (IGNNE#)

Intel486 DX, IntelDX2, and IntelDX4 processors will ignore a numeric error and continue executing non-control Floating-Point instructions when IGNNE# is asserted and FERR# is still activated. When IGNNE# is not asserted and a pending unmasked numeric exception exists (SW.ES = 1), the Intel486 processor will behave as follows:

On encountering a Floating-Point instruction that is one of FNINIT, FNCLEX, FNSTENV, FNSAVE, FNSTSW or FNSTCW, the Intel486 processor will assert the FERR# pin. Subsequently, the processor opens an interrupt sampling window. The interrupts are checked and serviced during this window. If no interrupts are sampled within this window, the processor will then execute these instructions in spite of the pending unmasked exception. For further details, please refer to section 19.3 in the *Intel486™ Processor Family Programmer's Reference Manual,* revision 3.

On encountering any Floating-Point instruction other than FNINIT, FNCLEX, FNSTENV, FNSAVE, FNSTSW or FNSTCW, the Intel486 processor will stop execution, assert the FERR# pin and wait for an external interrupt.

IGNNE# has no effect when the NE bit in control register 0 is set.

The IGNNE# input is active LOW and provided with a small internal pull-up resistor. This input is asynchronous, but must meet setup and hold times t_{20} and t_{21} to ensure recognition on any specific clock.

9.2.16 BUS SIZE CONTROL (BS16#, BS8#)

The BS16# and BS8# inputs allow external 16- and 8-bit buses to be supported with a small number of external components. The Intel486 processor samples these pins every clock. The value sampled in the clock before ready determines the bus size. When asserting BS16# or BS8#, only 16 or 8 bits of the data bus need be valid. If both BS16# and BS8# are asserted, an 8-bit bus width is selected.

When BS16# or BS8# is asserted, the Intel486 processor will convert a larger data request to the appropriate number of smaller transfers. The byte enables will also be modified appropriately for the bus size selected.

BS16# and BS8# are active LOW and are provided with small internal pull-up resistors. BS16# and BS8# must satisfy the setup and hold times t_{14} and t_{15} for proper chip operation.

9.2.17 ADDRESS BIT 20 MASK (A20M#)

Asserting the A20M# input causes the Intel486 processor to mask physical address bit 20 before performing a lookup in the internal cache and before driving a memory cycle to the outside world. When A20M# is asserted, the Intel486 processor emulates the 1-Mbyte address wraparound that occurs on the 8086. A20M# is active LOW and must be asserted only when the processor is in real mode. The A20M# is not defined in Protected Mode. A20M# is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For correct operation of the chip, A20M# should not be active at the falling edge of RESET.

A20M# exhibits a minimum 4-clock latency, from time of assertion to masking of the A20 bit. A20M# is ignored during cache invalidation cycles. I/O writes require A20M# to be asserted a minimum of two clocks prior to RDY being returned for the I/O write. This ensures recognition of the address mask before the Intel486 processor begins execution of the instruction following OUT. If A20M# is asserted after the ADS# of a data cycle, the A20 address signal is not masked during this cycle but is masked in the next cycle. During a prefetch (cacheable or not), if A20M# is asserted after the first ADS#, A20 is not masked for the duration of the prefetch, even if BS16# or BS8# is asserted.

9.2.18 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSOR SIGNALS AND OTHER ENHANCED BUS FEATURES

This section describes the pins that interface with the system to support the Enhanced Bus mode write-back features at system level.

9.2.18.1 Cacheability (CACHE#)

The CACHE# output indicates the internal cacheability on read cycles and a burst write-back on write cycles. CACHE# is asserted for cacheable reads, cacheable code fetches and write-backs. It is driven inactive for non-cacheable reads, special cycles, I/O cycles and write-through cycles. This is different from the PCD (page cache disable) pin. The operational differences between CACHE# and PCD are listed in Table 9-3. See Table 9-4 for operational differences between CACHE# and other Intel486 processor signals.

Table 9-3. Differences between CACHE# and PCD

Bus Operation	CACHE#	PCD
All reads (1)	same as PCD(3)	same as PCD(3)
Replacement write-back	low	low
Snoop-forced write-back	low	low
S-state write-through	high	same as PCD(3)
I-state write-through (2)	high	same as PCD(3)

NOTES:
1. Includes line fills and non-cacheable reads. During locked read cycles CACHE# is inactive. The non-cacheable reads may or may not be burst.
2. Due to the non-allocate on write policy, this includes both cacheable and non-cacheable writes. PCD will distinguish between the two, but CACHE# does not.
3. This behavior is the same as the existing specification of the Intel486 processor in write-through mode.

Table 9-4. CACHE# vs. Other Intel486™ Processor Signals

Pin Symbol	Relation To This Signal
ADS#	CACHE# is driven to valid state with ADS#
RDY#, BRDY#	CACHE# is de-asserted with the first RDY# or BRDY#
HLDA, BOFF#	CACHE# floats under these signals.
KEN#	The combination of CACHE# and KEN# determines if a read miss is converted into a cache line fill.

9.2.18.2 Cache Flush (FLUSH#)

FLUSH# is an existing pin that operates differently if the processor is configured as Enhanced Bus mode (write-back). In Enhanced Bus mode, it acts similar to the WBINVD instruction. In Enhanced Bus mode, FLUSH# is treated as an interrupt. It is sampled at each clock, but is recognized only on instruction boundary. Pending writes are completed before FLUSH# is serviced, and all prefetching is stopped. Depending on the number of modified lines in the cache, the flush could take up to a minimum of 1280 bus clocks or 2560 processor clocks and a maximum of 5000+ bus clocks to scan the cache, perform the write-backs, invalidate the cache and run two special cycles. After all modified lines are written back to memory, two special bus cycles, "First Flush ACK Cycle" and "Second Flush ACK Cycle," are issued, in that order. These cycles differ from the special cycles issued after WBINVD only in that A2=1 (address line 2 = 1). SRESET, STPCLK#, INTR, NMI and SMI# are not recognized during a flush write-back, while BOFF#, AHOLD and HOLD are recognized.

FLUSH# may be asserted just for a single clock, or may be retained asserted, but should be de-asserted at or prior to the RDY# returned from the "First Flush ACK" special bus cycle. If asserted during INVD or WBINVD, FLUSH# will be recognized. If asserted simultaneously with SMI#, then SMI# is recognized after FLUSH# is serviced.

FLUSH# may be driven at any time. If driven during SRESET, it must be held for one clock after SRESET is de-asserted to be recognized.

9.2.18.3 Hit/Miss to a Modified Line (HITM#)

HITM# is a cache coherency protocol pin that is driven only in Enhanced Bus mode. When a snoop cycle is run by the system (with INV = "0" or INV = "1"), HITM# indicates if the processor contains the snooped line in the M-state. Assertion of HITM# indicates that the line will be written back in total, unless the processor is already in the process of doing a replacement write-back of the same line.

HITM# will be valid on the bus two system clocks after EADS# is asserted on the bus. If asserted, HITM# remains asserted until the last RDY# or BRDY# of the snoop write-back cycle is returned. It will be de-asserted before the next following ADS#. (See Table 9-5.)

Table 9-5. HITM# vs. Other Intel486™ Processor Signals

Pin Symbol	Relation To This Signal
EADS#	HITM# is asserted due to an EADS#-driven snoop, provided the snooped line is in the M-state in the cache.
HLDA, BOFF#	HITM# does not float under these signals.
ADS#, CACHE#	The beginning of a snoop write-back cycle is identified by the assertion of ADS#, CACHE#, and HITM#.

9.2.18.4 Soft Reset (SRESET)

When in Enhanced Bus mode, SRESET has the following differences: SRESET, unlike RESET, does not cause AHOLD, A20M#, FLUSH#, UP#, and WB/WT# pins to be sampled (i.e., special test modes and on-chip cache configuration cannot be accessed with SRESET.)

On SRESET, the internal SMRAM base register retains its previous value and the processor does not flush, write-back or disable the internal cache. CR0.CD and CR0.NW retain previous values, CR0.4 is set to "1", and the remaining bits are cleared. Because SRESET is treated as an interrupt, it is possible to have a bus cycle while SRESET is asserted. A bus cycle could be due to an ongoing instruction, emptying the write buffers of the processor, or snoop write-back cycles if there is a snoop hit to an M-state line while SRESET is asserted.

NOTE:
For both Standard Bus mode and Enhanced Bus mode:

- SMI# must be blocked during SRESET. It must also be blocked for a minimum of 2 clocks after SRESET is de-asserted.
- SRESET must be blocked during SMI#. It must also be blocked for a minimum of 20 clocks after SMIACT# is de-asserted.

9.2.18.5 Invalidation Request (INV)

INV is a cache coherency protocol pin that is used only in Enhanced Bus mode. It is sampled by the processor on EADS#-driven snoop cycles. **It is necessary to assert this pin to simulate the standard mode processor invalidate cycle on write-through-only lines.** INV also invalidates the write-back lines. However, if the snooped line is in the M-state, the line will be written back and then invalidated.

INV is sampled when EADS# is asserted. If INV is not asserted with EADS#, the snoop cycle will have no effect on a write-through-only line or a line allocated as write-back, but not yet modified. If the line is write-back and modified, it will be written back to memory, but will not be de-allocated (invalidated) from the internal cache. The address of the snooped cache line is provided on the address bus. (See Table 9-6.)

Table 9-6. INV vs. Other Intel486™ Processor Signals

Pin Symbol	Relation To This Signal
EADS#	EADS# determines when INV is sampled.
A31–A4	The address of the snooped cache line is provided on these pins.

9.2.18.6 Write-Back/Write-Through (WB/WT#)

WB/WT# enables Enhanced Bus mode (write-back cache). It also allows the system to define a cached line as write-through or write-back.

WB/WT# is sampled at the falling edge of RESET to determine if Enhanced Bus mode is enabled (WB/WT# must be driven for two clocks before and two clocks after RESET for recognition by the processor). If sampled low or floated, the Write-Back Enhanced Intel486 processors operate in the Intel486 processor standard mode. For write-through only operation, i.e. standard mode, WB/WT# does not need to be connected.

In Enhanced Bus mode, WB/WT# allows the system-hardware to force any allocated line to be treated as write-through or write-back. As with cacheability, both the processor and the external system must agree that a line may be treated as write-back for the internal cache to be allocated as write-back. The default is always write-through. **The processor's indication of write-back vs. write-through is from the PWT pin, in which function and timing are the same as in the standard mode Intel486 processor.**

To define write-back or write-through configuration of a line, WB/WT# is sampled in the same clock as the first RDY# or BRDY# is returned during a line fill (allocation) cycle. (See Table 9-7.)

Table 9-7. WB/WT# vs. Other Intel486™ Processor Signals

Pin Symbol	Relation to This Signal
RDY#, BRDY#	WB/WT# is sampled with the first RDY# or BRDY#
PWT	The combination of WB/WT# and PWT determine if the Write-Back Enhanced IntelDX2™ processor will treat the line as WB.
PCD, CACHE#; KEN#	The state of WB/WT# does not matter if PCD, CACHE# or KEN# define the line to be non-cacheable.
W/R#	WB/WT# is significant only on read fill cycles.
RESET	WB/WT# is sampled on the falling edge of RESET to define the cache configuration.

9.2.18.7 Pseudo-Lock Output (PLOCK#)

In the Enhanced bus mode, PLOCK# is always driven inactive. In this mode, a 64-bit data read (caused by an FP operand access or a segment descriptor read) is treated as a multiple cycle read request, which may be a burst or a non-burst access based on whether BRDY# or RDY# is returned by the system. Because only write-back cycles (caused by Snoop write-back or replacement write-back) are burstable, a 64-bit write will be driven out as two non-burst bus cycles. BLAST# is asserted during both writes. Refer to section 10.2, "Bus Functional Description" for details on Pseudo-Locked bus cycles.

9.2.19 IntelDX4™ PROCESSOR VOLTAGE DETECT SENSE OUTPUT (VOLDET)

A voltage detect sense pin (VOLDET) has been added to the IntelDX4 processor PGA package. This pin allows external system logic to distinguish between a 5V Intel486 DX or IntelDX2 processor and the 3.3V IntelDX4 processor. The pin passively indicates to external logic whether the installed PGA processor requires 5V (in the case of the Intel486 DX or IntelDX2 processor) or 3.3V (in the case of the IntelDX4 processor). Pin S4 has been defined as the VOLDET pin because this pin is defined as an INC pin on the Intel486 DX and IntelDX2 processor. This pin is only provided in PGA package.

To utilize this feature, a weak, external pull-up resistor should be connected to the VOLDET pin. This pin samples high (logic 1) if the installed processor is a 5V Intel486 DX or IntelDX2 processor. This pin samples low (logic 0) if a IntelDX4 processor is installed. Upon sampling the logic level of this pin, external logic can then enable the proper V_{CC} level to the processor. In power sensitive applications, an active element is preferred for the pull-up device because it could be disabled after sampling, thereby eliminating the resulting DC current path when the installed processor is the IntelDX4 processor.

Figure 9-4 shows a logical representation of the Voltage Detect sense mechanism.

This pin can remain not connected for those system designs that do not wish to utilize this voltage detect feature.

9.2.20 BOUNDARY SCAN TEST SIGNALS

The following boundary scan test signals are available on all Intel486 processors except the Intel486 SX processor in PGA packages.

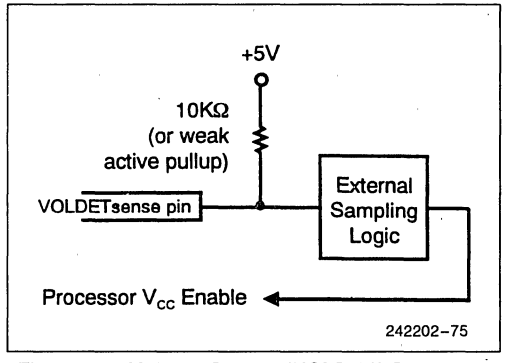


Figure 9-4. Voltage Detect (VOLDET) Sense Pin

Test Clock (TCK)

TCK is an input to the Intel486 processor and provides the clocking function required by the JTAG boundary scan feature. TCK is used to clock state information and data into and out of the component.

State select information and data are clocked into the component on the rising edge of TCK on TMS and TDI, respectively. Data is clocked out of the part on the falling edge of TCK on TDO.

In addition to using TCK as a free running clock, it may be stopped in a low, O, state, indefinitely as described in IEEE 1149.1. While TCK is stopped in the low state, the boundary scan latches retain their state.

When boundary scan is not used, TCK should be tied high or left as a NC. (This is important during power up to avoid the possibility of glitches on the TCK which could prematurely initiate boundary scan operations.) TCK is supplied with an internal pull-up resistor.

TCK is a clock signal and is used as a reference for sampling other JTAG signals. On the rising edge of TCK, TMS and TDI are sampled. On the falling edge of TCK, TDO is driven.

Test Mode Select (TMS)

TMS is decoded by the JTAG TAP (Tap Access Port) to select the operation of the test logic, as described in section 11.5.4, "Test Access Port Controller."

To guarantee deterministic behavior of the TAP controller TMS is provided with an internal pull-up resistor. If boundary scan is not used, TMS may be tied high or left unconnected. TMS is sampled on the rising edge of TCK. TMS is used to select the internal TAP states required to load boundary scan instructions to data on TDI. For proper initialization of the JTAG logic, TMS should be driven high, "1," for at least four TCK cycles following the rising edge of RESET.

Test Data Input (TDI)

TDI is the serial input used to shift JTAG instructions and data into the component. The shifting of instructions and data occurs during the SHIFT-IR and SHIFT-DR TAP controller states, respectively. These states are selected using the TMS signal as described in section 11.5.4, "Test Access Port Controller."

An internal pull-up resistor is provided on TDI to ensure a known logic state if an open circuit occurs on the TDI path. Note that when "1" is continuously shifted into the instruction register, the BYPASS instruction is selected. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and the SHIFT-DR states. During all other TAP controller states, TDI is a "don't care." TDI is only sampled when TMS and TCK have been used to select the SHIFT-IR or SHIFT-DR states in the TAP controller. For proper initialization of JTAG logic, TDI should be driven high, "1," for at least four TCK cycles following the rising edge of RESET.

Test Data Output (TDO)

TDO is the serial output used to shift JTAG instructions and data out of the component. The shifting of instructions and data occurs during the SHIFT-IR and SHIFT-DR TAP controller states, respectively. These states are selected using the TMS signal as described in section 11.5.4, "Test Access Port Controller". When not in SHIFT-IR or SHIFT-DR state, TDO is driven to a high impedance state to allow connecting TDO of different devices in parallel. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state. TDO is only driven when TMS and TCK have been used to select the SHIFT-IR or SHIFT-DR states in the TAP controller.

9.3 Interrupt and Non-Maskable Interrupt Interface

The Intel486 processor provides four asynchronous interrupt inputs: INTR (interrupt request), NMI (non-maskable interrupt input), SMI# (system management interrupt) and STPCLK# (stop clock interrupt). This section describes the hardware interface between the instruction execution unit and the pins. For a description of the algorithmic response to interrupts refer to section 4.7.6, "Interrupts". For interrupt timings refer to section 10.2.10, "Interrupt Acknowledge".

9.3.1 INTERRUPT LOGIC

The Intel486 processor contains a two-clock synchronizer on the interrupt line. An interrupt request will reach the internal instruction execution unit two clocks after the INTR pin is asserted, if proper setup is provided to the first stage of the synchronizer.

There is no special logic in the interrupt path other than the synchronizer. The INTR signal is level sensitive and must remain active for the instruction execution unit to recognize it. The interrupt will not be serviced by the Intel486 processor if the INTR signal does not remain active.

The instruction execution unit will look at the state of the synchronized interrupt signal at specific clocks during the execution of instructions (if interrupts are enabled). These specific clocks are at instruction boundaries, or iteration boundaries in the case of string move instructions. Interrupts will only be accepted at these boundaries.

An interrupt must be presented to the Intel486 processor INTR pin three clocks before the end of an instruction for the interrupt to be acknowledged. Presenting the interrupt 3 clocks before the end of an instruction allows the interrupt to pass through the two clock synchronizer leaving one clock to prevent the initiation of the next sequential instruction and to begin interrupt service. If the interrupt is not received in time to prevent the next instruction, it will be accepted at the end of next instruction, assuming INTR is still held active.

The longest latency between when an interrupt request is presented on the INTR pin and when the interrupt service begins is: longest instruction used + the two clocks for synchronization + one clock required to vector into the interrupt service microcode.

9.3.2 NMI LOGIC

The NMI pin has a synchronizer like that used on the INTR line. Other than the synchronizer, the NMI logic is different from that of the maskable interrupt.

NMI is edge triggered as opposed to the level triggered INTR signal. The rising edge of the NMI signal is used to generate the interrupt request. The NMI input need not remain active until the interrupt is actually serviced. The NMI pin only needs to remain active for a single clock if the required setup and hold times are met. NMI will operate properly if it is held active for an arbitrary number of clocks.

The NMI input must be held inactive for at least four clocks after it is asserted to reset the edge triggered logic. A subsequent NMI may not be generated if the NMI is not held inactive for at least four clocks after being asserted.

The NMI input is internally masked whenever the NMI routine is entered. The NMI input will remain masked until an IRET (return from interrupt) instruction is executed. Masking the NMI signal prevents recursive NMI calls. If another NMI occurs while the NMI is masked off, the pending NMI will be executed after the current NMI is done. Only one NMI can be pending while NMI is masked.

9.3.3 SMI# LOGIC

SMI# is edge triggered like NMI, but the interrupt request is generated on the falling-edge. SMI# is an asynchronous signal, but setup and hold times, t_{20} and t_{21}, must be met in order to guarantee recognition on a specific clock. The SMI# input need not remain active until the interrupt is actually serviced. The SMI# input only needs to remain active for a single clock if the required setup and hold times are met. SMI# will also work correctly if it is held active for an arbitrary number of clocks.

The SMI# input must be held inactive for at least four clocks after it is asserted to reset the edge triggered logic. A subsequent SMI# might not be recognized if the SMI# input is not held inactive for at least four clocks after being asserted.

SMI#, like NMI, is not affected by the IF bit in the EFLAGS register and is recognized on an instruction boundary. An SMI# will not break locked bus cycles.

The SMI# has a higher priority than NMI and is not masked during an NMI.

After the SMI# interrupt is recognized, the SMI# signal will be masked internally until the RSM instruction is executed and the interrupt service routine is complete. Masking the SMI# prevents recursive SMI# calls. The SMI# input must be de-asserted for at least 4 clocks to reset the edge triggered logic. If another SMI# occurs while the SMI# is masked, the pending SMI# will be recognized and executed on the next instruction boundary after the current SMI# completes. This instruction boundary occurs before execution of the next instruction in the interrupted application code, resulting in back to back SMM handlers. Only one SMI# can be pending while SMI# is masked.

The SMI# signal is synchronized internally and should be asserted at least three (3) CLK periods prior to asserting the RDY# signal in order to guarantee recognition on a specific instruction boundary. This is important for servicing an I/O trap with an SMI# handler.

9.3.4 STPCLK# LOGIC

STPCLK# is level triggered and active LOW. STPCLK# is an asynchronous signal, but must remain active until the processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop

Grant bus cycle. When the processor enters the Stop Grant state, the internal pull-up resistor of STPCLK#, CLKMUL (for IntelDX4 processor), and UP# are disabled so that the processor power consumption is reduced. The STPCLK# input must be driven high (not floated) in order to exit the Stop Grant state. **STPCLK# must be de-asserted for a minimum of 5 clocks after RDY# or BRDY# is returned active for the Stop Grant Bus Cycle before being asserted again.**

When the processor recognizes a STPCLK# interrupt, the processor will stop execution on the next instruction boundary (unless superseded by a higher priority interrupt), stop the pre-fetch unit, empty all internal pipelines and the write buffers, generate a Stop Grant bus cycle, and then stop the internal clock. At this point the processor is in the Stop Grant state.

The processor cannot respond to a STPCLK# request from an HLDA state because it cannot empty the write buffers and, therefore, cannot generate a Stop Grant cycle.

The rising edge of STPCLK# will tell the processor that it can return to program execution at the instruction following the interrupted instruction.

Unlike the normal interrupts, INTR and NMI, the STPCLK# interrupt does not initiate acknowledge cycles or interrupt table reads. Among external interrupts, the STPCLK# order of priority is shown in section 4.7.6.

9.4 Write Buffers

The Intel486 processor contains four write buffers to enhance the performance of consecutive writes to memory. The buffers can be filled at a rate of one write per clock until all four buffers are filled.

When all four buffers are empty and the bus is idle, a write request will propagate directly to the external bus bypassing the write buffers. If the bus is not available at the time the write is generated internally, the write will be placed in the write buffers and propagate to the bus as soon as the bus becomes available. The write is stored in the on-chip cache immediately if the write is a cache hit.

Writes will be driven onto the external bus in the same order in which they are received by the write buffers. Under certain conditions a memory read will go onto the external bus before the memory writes pending in the buffer even though the writes occurred earlier in the program execution.

A memory read will only be reordered in front of all writes in the buffers under the following conditions: If all writes pending in the buffers are cache hits and the read is a cache miss. Under these conditions the Intel486 processor will not read from an external memory location that needs to be updated by one of the pending writes.

Reordering of a read with the writes pending in the buffers can only occur once before all the buffers are emptied. Reordering read once only maintains cache consistency. Consider the following example: The processor writes to location X. Location X is in the internal cache, so it is updated there immediately. However, the bus is busy so the write out to main memory is buffered (see Figure 9-5). At this point, any reads to location X would be cache hits and most up-to-date data would be read.

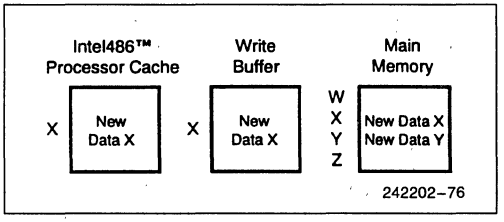


Figure 9-5. Reordering of a Reads with Write Buffers

The next instruction causes a read to location Y. Location Y is not in the cache (a cache miss). Because the write in the write buffer is a cache hit, the read is reordered. When location Y is read, it is put into the cache. The possibility exists that location Y will replace location X in the cache. If this is true, location X would no longer be cached (see Figure 9-6).

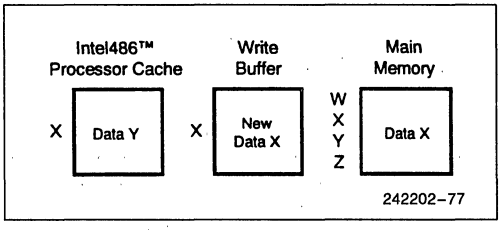


Figure 9-6. Reordering of a Reads with Write Buffers

Cache consistency has been maintained up to this point. If a subsequent read is to location X (now a cache miss) and it was reordered in front of the buffered write to location X, stale data would be read. This is why only 1 read is allowed to be reordered. Once a read is reordered, all the writes in the write buffer are flagged as cache misses to ensure that no more reads are reordered. Because one of the conditions to reorder a read is that all writes in the write buffer must be cache hits, no more reordering is allowed until all of those flagged writes propagate to the bus. Similarly, if an invalidation cycle is run all entries in the write buffer are flagged as cache misses.

For multiple processor systems and/or systems using DMA techniques, such as bus snooping, locked semaphores should be used to maintain cache consistency.

9.4.1 WRITE BUFFERS AND I/O CYCLES

Input/Output (I/O) cycles must be handled in a different manner by the write buffers.

I/O reads are never reordered in front of buffered memory writes. This ensures that the Intel486 processor will update all memory locations before reading status from an I/O device.

The Intel486 processor never buffers single I/O writes. When processing an OUT instruction, internal execution stops until the I/O write actually completes on the external bus. This allows time for the external system to drive an invalidate into the Intel486 processor or to mask interrupts before the processor progresses to the instruction following OUT. REP OUTS instructions will be buffered.

A read cycle must be explicitly generated to a non-cacheable location in memory to guarantee that a read bus cycle is performed. This read will not be allowed to proceed to the bus until after the I/O write has completed because I/O writes are not buffered. The I/O device will have time to recover to accept another write during the read cycle.

9.4.2 WRITE BUFFERS IMPLICATIONS ON LOCKED BUS CYCLES

Locked bus cycles are used for read-modify-write accesses to memory. During a read-modify-write access, a memory base variable is read, modified and then written back to the same memory location. It is important that no other bus cycles, generated by other bus masters or by the Intel486 processor itself, be allowed on the external bus between the read and write portion of the locked sequence.

During a locked read cycle, the Intel486 processor will always access external memory, it will never look for the location in the on-chip cache, but for write cycles, data is written in the internal cache (if cache hit) and in the external memory. All data pending in the Intel486 processor's write buffers will be written to memory before a locked cycle is allowed to proceed to the external bus.

The Intel486 processor will assert the LOCK# pin after the write buffers are emptied during a locked bus cycle. With the LOCK# pin asserted, the processor will read the data, operate on the data and place the results in a write buffer. The contents of the write buffer will then be written to external memory. LOCK# will become inactive after the write part of the locked cycle.

9.5 Reset and Initialization

The Intel486 processor has a built-in self-test (BIST) that can be run during reset. BIST is invoked if the AHOLD pin is asserted for one clock before and one clock after RESET is de-asserted. RESET must be active for 15 clocks with or without BIST being enabled. To ensure proper results, neither FLUSH# nor SRESET can be asserted while BIST is executing. Refer to section 11.0, "Processor Testability," for information on Intel486 processor testability.

The Intel486 processor registers have the values shown in Table 9-8 after RESET is performed. The EAX register contains information on the success or failure of the BIST if the self test is executed. The DX register always contains a component identifier at the conclusion of RESET. The upper byte of DX (DH) will contain 04 and the lower byte (DL) will contain the revision identifier. (See Table 9-9.)

RESET forces the Intel486 processor to terminate all execution and local bus activity. No instruction or bus activity will occur as long as RESET is active.

All entries in the cache are invalidated by RESET.

9.5.1 FLOATING-POINT REGISTER VALUES

In addition to the register values listed above, Intel486 DX, IntelDX2, and IntelDX4 processors have the Floating-Point register values shown in Table 9-10.

The Floating-Point registers are initialized as if the FINIT/FNINIT (initialize processor) instruction were executed if the BIST was performed. If the BIST is not executed, the Floating-Point registers are unchanged.

The Intel486 processor will start executing instructions at location FFFFFFF0H after RESET. When the first Inter-Segment Jump or Call is executed, address lines A20–A31 will drop LOW for CS-relative memory cycles, and the Intel486 processor will only execute instructions in the lower one Mbyte of physical memory. This allows the system designer to use a ROM at the top of physical memory to initialize the system and take care of RESETs.

Table 9-8. Register Values after Reset

Register	Initial Value (BIST)	Initial Value (No BIST)
EAX	Zero (Pass)	Undefined
ECX	Undefined	Undefined
EDX	0400 + Revision ID	0400 + Revision ID
EBX	Undefined	Undefined
ESP	Undefined	Undefined
EBP	Undefined	Undefined
ESI	Undefined	Undefined
EDI	Undefined	Undefined
EFLAGS	00000002h	00000002h
EIP	0FFF0h	0FFF0h
ES	0000h	0000h
CS	F000h*	F000h*
SS	0000h	0000h
DS	0000h	0000h
FS	0000h	0000h
GS	0000h	0000h
IDTR	Base = 0, Limit = 3FFh	Base = 0, Limit = 3FFh
CR0	60000010h	60000010h
DR7	00000000h	00000000h

Table 9-9. Intel486™ Processor Revision ID

Product	Component ID (DH)	Revision ID (DL)
Intel486 SX Processor	04	2x
IntelSX2™ Processor	04	5x
Intel486™ DX Processor	04	0x 1x
IntelDX2™ Processor	04	3x
Write-Back Enhanced IntelDX2 Processor	04	7x
IntelDX4™ Processor	04	8x
Write-Back Enhanced IntelDX4™ Processor	04	8x

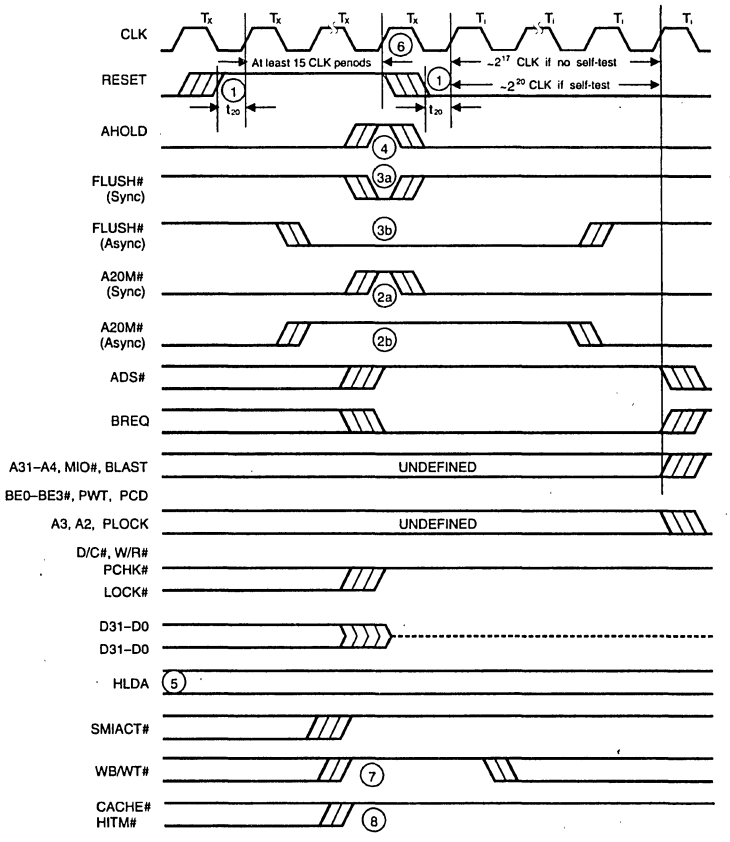
Table 9-10. Floating-Point Values after Reset

Register	Initial Value (BIST)	Initial Value (No BIST)
CW	037Fh	Unchanged
SW	0000h	Unchanged
TW	FFFFh	Unchanged
FIP	00000000h	Unchanged
FEA	00000000h	Unchanged
FCS	0000h	Unchanged
FDS	0000h	Unchanged
FOP	000h	Unchanged
FSTACK	Undefined	Unchanged

9.5.2 PIN STATE DURING RESET

The Intel486 processor recognizes and can respond to HOLD, AHOLD, and BOFF# requests regardless of the state of RESET. Thus, even though the processor is in reset, it can still float its bus in response to any of these requests.

While in reset, the Intel486 processor bus is in the state shown in Figure 9-7 if the HOLD, AHOLD and BOFF# requests are inactive. The figure shows the bus state for the Intel486 processor. Note that the address (A31–A2, BE3#–BE0#) and cycle definition (M/IO#, D/C#, W/R#) pins are undefined from the time reset is asserted up to the start of the first bus cycle. All undefined pins **(except FERR#)** assume known values at the beginning of the first bus cycle. The first bus cycle is always a code fetch to address FFFFFFF0H.



Notes:
RESET is an asynchronous input. t_{20} must be met only to guarantee recognition on a specific clock edge.
2a. When A20M# is driven synchronously, it must be driven high (inactive) for the CLK edge prior to the falling edge of RESET to ensure proper operation. A20M# setup and hold times must be met.
2b. When A20M# is driven asynchronously, it should be driven low (active) for two CLKs prior to and two CLKs after the falling edge of RESET to ensure proper operation.
3a. When FLUSH# is driven synchronously, it must be driven low (high) for the CLK edge prior to the falling edge of RESET to invoke the 3-state Output Test Mode. All outputs are guaranteed 3-stated within 10 CLKs of RESETR being deasserted. FLUSH# setup and hold times must be met.
3b. When FLUSH# is driven asynchronously, it must be driven low (active) for two CLKs prior and two CLKs after the falling edge of RESET to invoke the 3-state Output Test Mode. All outputs are guaranteed 3-stated withing 10 CLKs of RESET being deasserted.
4. AHOLD should be driven high (active) for the CLK edge prior to the falling edge of RESET to invoke the Built-In Self-Test (BIST). AHOLD setup and hold times must be met.
5. Hold is recognized normally during RESET. On power-up HLDA is indeterminate until RESET is recognized by the processor.
6. 15 CLKs RESET pulse width for warm resets. Power-up resets require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
7. WB/WT# should be driven high for at least one CLK before falling edge of RESET and at least one CLK after falling edge of RESET to enable the Enhanced Bus Mode. The Standard Bus Mode will be enabled if WB/WT# is sampled low or left floating at the falling edge of RESET.
8. The system may sample HITM# to detect the presence of the Enhanced Bus Mode. If HITM# is HIGH for one CLK after RESET is inactive, the Enhanced Bus Mode is present.

Figure 9-7. Pin States During RESET

9.5.2.1 Controlling the CLK Signal in the Processor during Power On

The power on requirements of the Intel486 processor with regard to allowable CLK input during the power-on sequence have never been specified. Clocking the processor before V_{CC} has reached its normal operating level can cause unpredictable results on Intel486 processors. While Intel will maintain original clock and power specifications (none), this section reflects what Intel considers to be a good clock design.

Intel strongly recommends that system designers ensure that a clock signal is not presented to the Intel486 processor until V_{CC} has stabilized at its normal operating level. This design recommendation can easily be met by gating the clock signal with a POWERGOOD signal. The POWERGOOD signal should reflect the status of V_{CC} at the Intel486 processor (which may be different from the power supply status in designs that provide power to the processor through the use of a voltage regulator or converter).

Most clock synthesizers and some clock oscillators contain on-board gating logic. If external gating logic is implemented, it should be done on the original clock signal output from the clock oscillator/synthesizer. Gating the clock to the processor independently of the clock to the rest of the motherboard will cause clock skew, which may violate processor or chipset timing requirements. If the clock signal to the motherboard is enabled with a POWERGOOD signal, it is also important to verify that the motherboard logic does not require a clock input prior to this POWERGOOD signal. Some chipsets also gate the clock to the processor only after a POWERGOOD signal, which inherently meets the requirements of this design note. Designs should implement this design note, so as to maintain maximum flexibility with all Intel486 processor steppings.

9.5.2.2 FERR# Pin State During Reset for Intel486™ DX, IntelDX2™, and IntelDX4™ Processors

FERR# reflects the state of the ES (error summary status) bit in the FPU status word. The ES bit is initialized whenever the FPU state is initialized. The FPU's status word register can be initialized by BIST or by executing FINIT/FNINIT instruction. Thus, after reset and before executing the first FINIT or FNINIT instruction, the values of the FERR# and the numeric status word register bits 0–7 depends on whether or not BIST is performed. Table 9-11 shows the state of FERR# signal after reset and before the execution of the FINIT/FNINIT instruction.

Table 9-11. FERR# Pin State after Reset and before FP Instructions

BIST Performed	FERR# Pin	FPU Status Word Register Bits 0–7
YES	Inactive (High)	Inactive (Low)
NO	Undefined (Low or High)	Undefined (Low or High)

After the first FINIT or FNINIT instruction, FERR# pin and the FPU status word register bits (0–7) will be inactive irrespective of the BIST.

9.5.2.3 Power-Down Mode (Upgrade Processor Support)

The Power-Down Mode on the Intel486 processor, when initiated by the upgrade processor, reduces the power consumption of the Intel486 processor (see Table 17-3, DC Specifications), as well as forces all of its output signals to be tri-stated. The UP# pin on the Intel486 processor is used for enabling the Power-Down Mode.

Once the UP# pin is driven active by the upgrade processor upon power-up, the Intel486 processor's bus is floated immediately. The Intel486 processor enters the Power-Down Mode when the UP# pin is sampled and asserted in the clock before the falling edge of RESET. The UP# pin has no effect on the power-down status, except during this edge. The Intel486 processor then remains in the Power-Down Mode until the next time the RESET signal is activated. For warm resets, with the upgrade processor in the system, the Intel486 processor will remain tri-stated and re-enter the Power-Down Mode once RESET is de-asserted. Similarly for power-up resets, if the upgrade processor is not taken out of the system, the Intel486 processor will tri-state its outputs upon sensing the UP# pin active and enter the Power-Down Mode after the falling edge of RESET.

9.6 Clock Control

The Intel486 processor provides an interrupt mechanism (STPCLK#) that allows system hardware to control the power consumption of the processor by stopping the internal clock (output of the PLL) to the processor core in a controlled manner. This low-power state is called the Stop Grant state. In addition, the STPCLK# interrupt allows the system to change the input frequency within the specified range or completely stop the CLK input frequency (input to the PLL). If the CLK input is completely stopped, the processor enters into the Stop Clock state—the lowest power state.

There are two targets for the low-power mode supply current:

- ~20–100 mA in the **Stop Grant state** (fast wake-up, frequency-and voltage-dependent), and
- ~100–1000 μA in the full **Stop Clock state** (slow wake-up, voltage-dependent).

See sections 9.6.4.2 and 9.6.4.3 for a detailed description of the Stop Grant and Stop Clock states.

9.6.1 STOP GRANT BUS CYCLE

A special Stop Grant bus cycle will be driven to the bus after the processor recognizes the STPCLK# interrupt. The definition of this bus cycle is the same as the HALT cycle definition for the standard Intel486 processor, with the exception that the Stop Grant bus cycle drives the value 0000 0010H on the address pins. The system hardware must acknowledge this cycle by returning RDY# or BRDY#. **The processor will not enter the Stop Grant state until either RDY# or BRDY# has been returned.**

The Stop Grant bus cycle is defined as follows:

M/IO# = 0, D/C# = 0, W/R# = 1, Address Bus = 0000 0010H (A_4 = 1), BE3#–BE0# = 1011, Data bus = undefined

The latency between a STPCLK# request and the Stop Grant bus cycle is dependent on the current instruction, the amount of data in the processor write buffers, and the system memory performance. (See Figure 9-8.)

9.6.2 PIN STATE DURING STOP GRANT

During the Stop Grant state, most output and input/output signals of the processor will maintain their previous condition (the level they held when entering the Stop Grant state). The data and data parity signals will be tri-stated. In response to HOLD being driven active during the Stop Grant state (when the CLK input is running), the processor will generate HLDA and tri-state all output and input/output signals that are tri-stated during the HOLD/HLDA state. After HOLD is de-asserted, all signals will return to their prior state before the HOLD/HLDA sequence.

In order to achieve the lowest possible power consumption during the Stop Grant state, the system designer must ensure the input signals with pull-up resistors are not driven LOW and the input signals with pull-down resistors are not driven HIGH. (See Table 3-11 in the "Quick Pin Reference" section for signals with internal pull-up and pull-down resistors.)

All inputs except the data bus pins must be driven to the power supply rails to ensure the lowest possible current consumption during Stop Grant or Stop Clock modes. For compatibility with future processors, data pins should be driven LOW to achieve the lowest possible power consumption. Pull-down resistors/bus keepers are needed to minimize leakage current.

If HOLD is asserted during the Stop Grant state, all pins that are normally floated during HLDA will still be floated by the processor. The floated pins should be driven to a low level. (See Table 9-12.)

9.6.3 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSOR PIN STATES DURING STOP GRANT SPECIFICS GENERIC UPDATE

During the Stop Grant state, most output signals of the processor will maintain their previous condition, which is the level they held when entering the Stop Grant state. The data bus and data parity signals also maintain their previous state. In response to HOLD being driven active during the Stop Grant state when the CLK input is running, the Write-Back Enhanced Intel486 processors will generate HLDA and tri-state all output and input/output signals that are tri-stated during the HOLD/HLDA state. After HOLD is de-asserted, all signals will return to the state they were in prior to the HOLD/HLDA sequence.

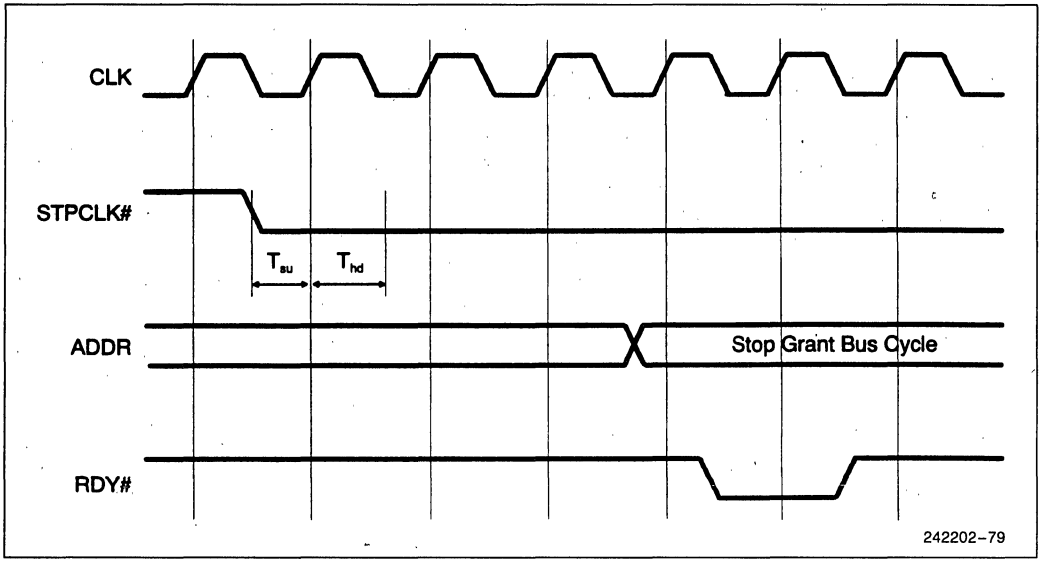


Figure 9-8. Stop Clock Protocol

Table 9-12. Pin State during Stop Grant Bus State

Signal	Type	State
A3-A2	O	Previous state
A31-A4	I/O	Previous state
D31-D0	I/O	Floated
BE3#-BE0#	O	Previous state
DP3-DP0	I/O	Floated
W/R#, D/C#, M/IO#	O	Previous state
ADS#	O	Inactive
LOCK#, PLOCK#	O	Inactive
BREQ	O	Previous state
HLDA	O	As per HOLD
BLAST#	O	Previous state
FERR#	O	Previous state
PCD, PWT	O	Previous state
PCHK#	O	Previous state
PWT, PCD	O	Previous state
SMIACT#	O	Previous state

All inputs should be driven to the power supply rails to ensure the lowest possible current consumption during the Stop Grant or Stop Clock states. (See Table 9-13.)

The Write-Back Enhanced Intel486 processors has bus keepers features. The data bus and data parity pins have bus keepers that maintain the previous state while in the Stop Grant state. External resistors are no longer required, which prevents excess current during the Stop Grant state. (If external resistors are present, they should be strong enough to "flip" the bus hold circuitry and eliminate potential DC paths. Alternately, "weak" resistors may also be added to prevent excessive current flow.) See section 17.3.3, "External Resistors Recommended to Minimize Leakage Currents," for external register values.

In order to obtain the lowest possible power consumption during the Stop Grant state, system designers must ensure that the input signals with pull-up resistors are not driven LOW, and the input signals with pull-down resistors are not driven HIGH. (See the Table 3-11 for signals with internal pull-up and pull-down resistors).

Table 9-13. Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Pin State during Stop Grant Bus Cycle

Signal	Type	State
A3-A2	O	Previous state
A31-A4	I/O	Previous state
D31-D0	I/O	Previous state
BE3#-BE0#	O	Previous state
DP3-DP0	I/O	Previous state
W/R#, D/C#, M/IO#	O	Previous state
ADS#	O	Inactive (high)
LOCK#, PLOCK#	O	Inactive (high)
BREQ	O	Previous state
HLDA	O	As per HOLD
BLAST#	O	Previous state
FERR#	O	Previous state
PCHK#	O	Previous state
PWT, PCD	O	Previous state
CACHE#	O	Inactive(1) (high)
HITM#	O	Inactive(1) (high)
SMIACT#	O	Previous state

NOTES:
1. For the case of snoop cycles (via EADS#) during Stop Grant state, both HITM# and CACHE# may go active depending on the snoop hit in the internal cache. During Stop Grant state, AHOLD, HOLD, BOFF# and EADS# are serviced normally.

9.6.4 CLOCK CONTROL STATE DIAGRAM

The following state descriptions and diagram show the state transitions during a Stop Clock cycle for the Intel486 processor. (Refer to Figure 9-9 for a Stop Clock state diagram.) Refer to section 9.6.5 for Write-Back Enhanced Intel486 processors Clock Control State specifics.

9.6.4.1 Normal State

This is the normal operating state of the processor.

9.6.4.2 Stop Grant State

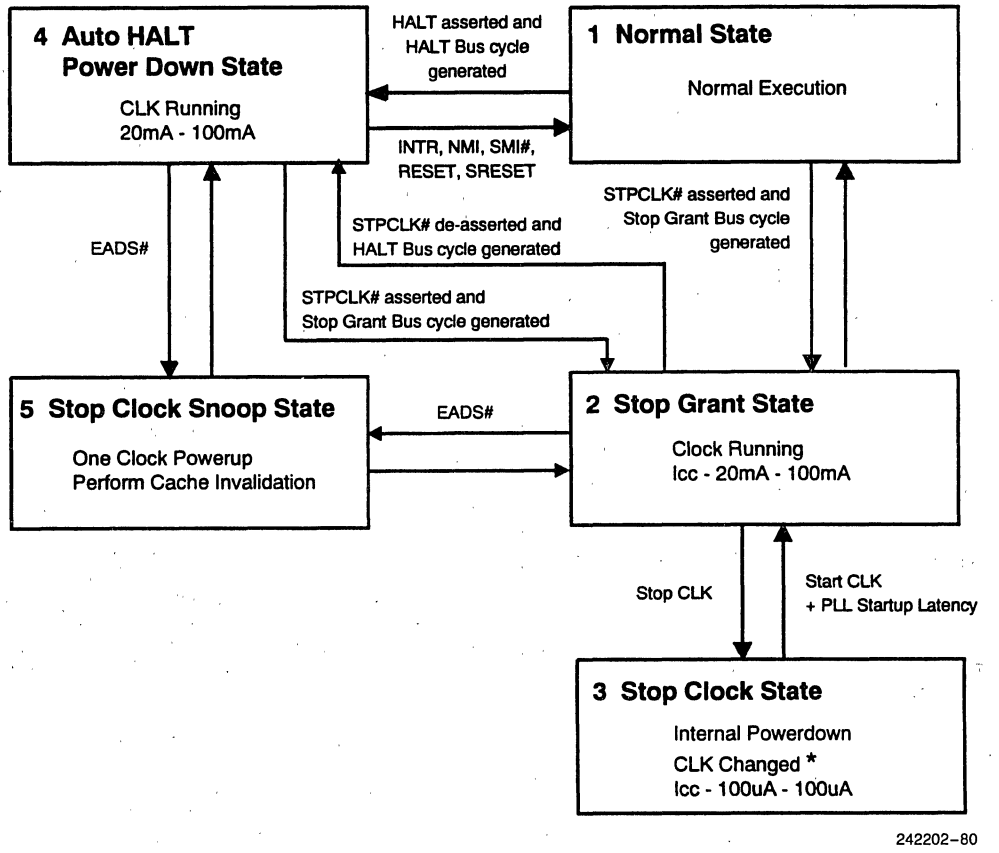
The **Stop Grant** state provides a fast wake-up state that can be entered by simply asserting the external STPCLK# interrupt pin. Once the Stop Grant bus cycle has been placed on the bus, and either RDY# or BRDY# is returned, the processor is in this state (depending on the CLK input frequency). The processor returns to the normal execution state 10–20 clock periods after STPCLK# has been de-asserted.

While in the Stop Grant state, the pull-up resistors on STPCLK#, CLKMUL (for the IntelDX4 processor) and UP# are disabled internally. The system must continue to drive these inputs to the state they were in immediately before the processor entered the Stop Grant state. For minimum processor power consumption, all other input pins should be driven to their inactive level while the processor is in the Stop Grant state.

A RESET or SRESET will bring the processor from the Stop Grant state to the Normal state. The processor will recognize the inputs required for cache invalidation's (HOLD, AHOLD, BOFF# and EADS#) as explained later in this section. The processor will not recognize any other inputs while in the Stop Grant state. Input signals to the processor will not be recognized until 1 CLK after STPCLK# is de-asserted (see Figure 9-10).

While in the Stop Grant state, the processor will not recognize transitions on the interrupt signals (SMI#, NMI, and INTR). Driving an active edge on either SMI# or NMI will not guarantee recognition and service of the interrupt request following exit from the Stop Grant state. However, if one of the interrupt signals (SMI#, NMI, or INTR) is driven active while the processor is in the Stop Grant state, and held active for at least one CLK after STPCLK# is de-asserted, the corresponding interrupt will be serviced. The Intel486 processor requires INTR to be held active until the processor issues an interrupt acknowledge cycle in order to guarantee recognition. (See Figure 9-10).

When the processor is in the Stop Grant state, the system is allowed to stop or change the CLK input. When the CLK input to the processor is stopped (or changed), the Intel486 processor requires the CLK input to be held at a constant frequency for a minimum of 1 ms before de-asserting STPCLK#. This 1-ms time period is necessary so that the PLL can stabilize, and it must be met before the processor will return to the Stop Grant state.



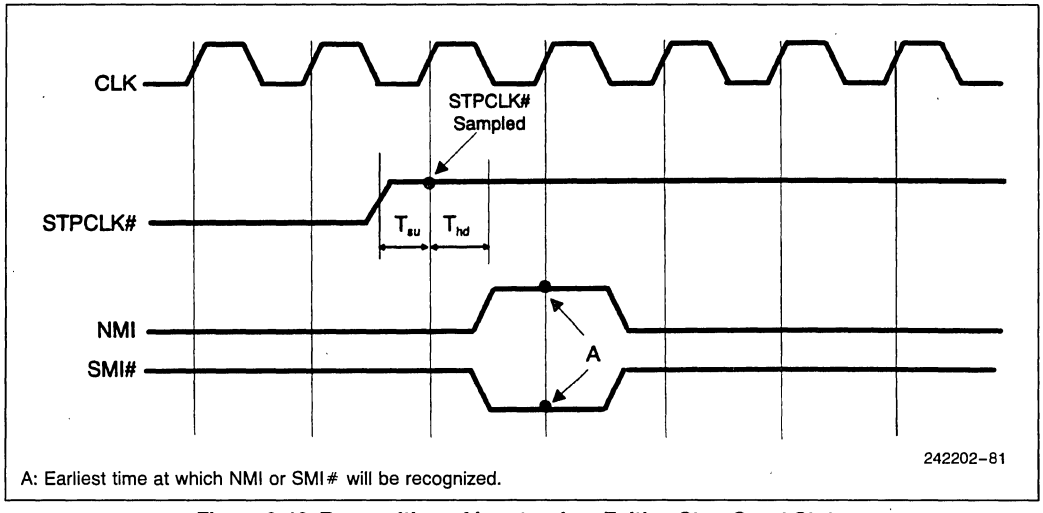
* The system can change the input frequency within the specified range or completely stop the CLK input frequency (input to PLL)

Figure 9-9. Intel486™ Processor Family Stop Clock State Machine

The processor will generate a Stop Grant bus cycle only when entering that state from the Normal or the Auto HALT Power Down state. When the processor enters the Stop Grant state from the Stop Clock state or the Stop Clock Snoop state, the processor will not generate a Stop Grant bus cycle.

9.6.4.3 Stop Clock State

Stop Clock state is entered from the Stop Grant state by stopping the CLK input (either logic high or logic low). None of the processor input signals should change state while the CLK input is stopped. Any transition on an input signal (with the exception of INTR, NMI and SMI#) before the processor has returned to the Stop Grant state will result in unpredictable behavior. If INTR is driven active while the CLK input is stopped, and held active until the processor issues an interrupt acknowledge bus cycle, it will be serviced in the normal manner. The system design must ensure the processor is in the correct state prior to asserting cache invalidation or interrupt signals to the processor.



A: Earliest time at which NMI or SMI# will be recognized.

Figure 9-10. Recognition of Inputs when Exiting Stop Grant State

The processor will return to the Stop Grant state after the CLK input has been running at a constant frequency for a period of time equal to the PLL start-up latency (see section 9.6.4.2). The CLK input can be restarted to any frequency between the minimum and maximum frequency listed in the AC timing specifications.

9.6.4.4 Auto HALT Power Down State

The execution of a HALT instruction will also cause the processor to automatically enter the **Auto HALT Power Down** state. The processor will issue a normal HALT bus cycle before entering this state. The processor will transition to the Normal state on the occurrence of INTR, NMI, SMI#, RESET, or SRESET.

The system can generate a STPCLK# while the processor is in the Auto HALT Power Down state. The processor will generate a Stop Grant bus cycle when it enters the Stop Grant state from the HALT state.

When the system de-asserts the STPCLK# interrupt, the processor will return execution to the HALT state. The processor will generate a new HALT bus cycle when it re-enters the HALT state from the Stop Grant state.

9.6.4.5 Stop Clock Snoop State (Cache Invalidations)

When the processor is in the Stop Grant state or the Auto HALT Power Down state, the processor will recognize HOLD, AHOLD, BOFF# and EADS# for cache invalidation. When the system asserts HOLD, AHOLD, or BOFF#, the processor will float the bus accordingly. When the system then asserts EADS#, the processor will transparently enter the **Stop Clock Snoop state** and will power up for 1 full core clock in order to perform the required cache snoop cycle. It will then re-freeze the clock to the processor core and return to the previous state. The processor does not generate a bus cycle when it returns to the previous state.

A FLUSH# event during the Stop Grant state or the Auto HALT Power Down state will be latched and acted upon by asserting the internal FLUSH# signal for one clock upon re-entering the Normal state.

9.6.4.6 Auto Idle Power Down State

When the chip is known to be truly idle and waiting for a RDY# or BRDY# from a memory or I/O bus cycle read, the Intel486 processor will reduce its core clock rate to be equal to the external CLK frequency without affecting performance. When any RDY# or BRDY# is asserted, the part will return to clocking the core at the specified multiplier of the external CLK frequency. This functionality is transparent to software and external hardware.

9.6.5 WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSOR CLOCK CONTROL STATE DIAGRAM

Figure 9-11 (state diagram) shows the state transitions during Stop Clock for the Write-Back Enhanced Intel486 processors.

NOTE:
The Stop Clock State Machine in the Standard bus configuration is identical to that of other Intel486 processors. (See section 9.6.4, "Clock Control State Diagram".)

9.6.5.1 Normal State

This is the normal operating state of the processor. When the processor is executing program/instruction and the STPCLK# pin is not asserted, the processor is said to be in its normal state.

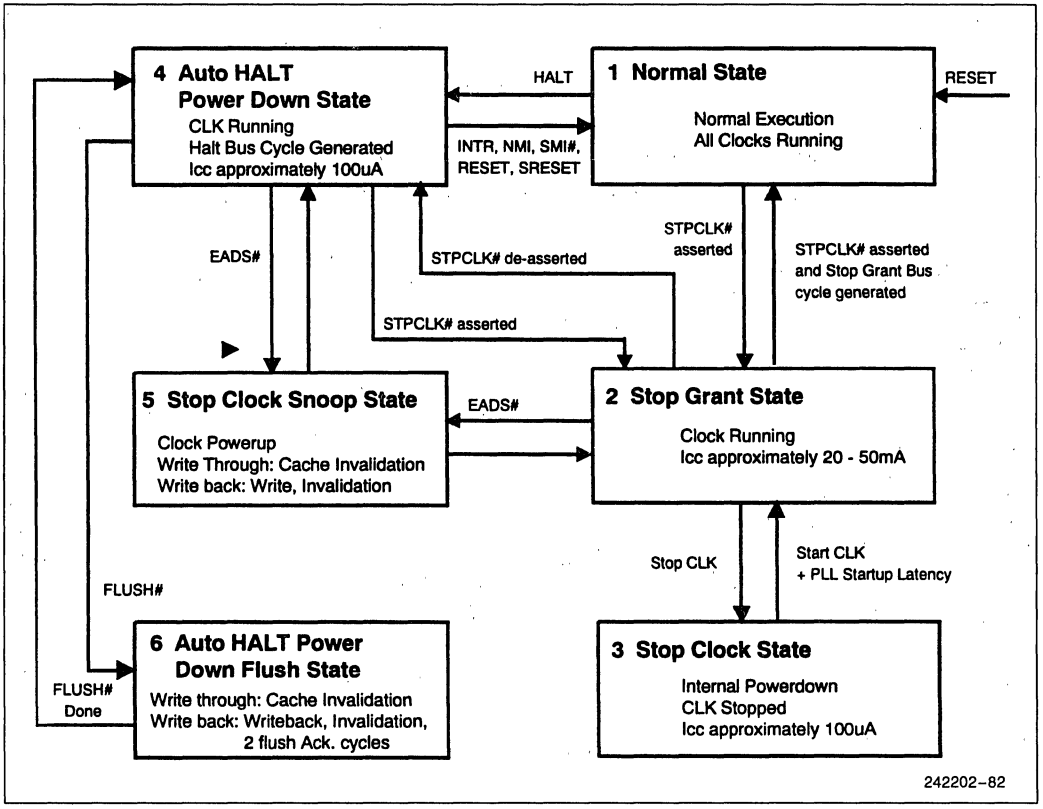


Figure 9-11. Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor Stop Clock State Machine (Enhanced Bus Configuration)

9.6.5.2 Stop Grant State

For minimum processor power consumption, all other input pins should be driven to their inactive level while the processor is in the Stop Grant state excepting data bus, data parity, WB/WT# and INV pins. WB/WT# should be driven low and INV should be driven high.

In both the Standard mode and Enhanced mode states, the following conditions exist:

- A RESET, SRESET or de-assertion of STPCLK# will bring the processor from the Stop Grant state to the Normal state.
- While in the Stop Grant state, the processor will not recognize transitions on the interrupt signals (SMI#, NMI, and INTR). This means SMI#, NMI, INTR are not Stop Break events. The external logic should de-assert STPCLK# before issuing interrupts or if an interrupt is asserted it should be kept asserted for at least 1 clock after STPCLK# is removed. (Note that the Write-Back Enhanced Intel486 processors require that INTR must be held active until the processor issues an interrupt acknowledge cycle in order to guarantee recognition).
- FLUSH# is not a Stop Break event. But if FLUSH# is asserted during the Stop Grant state, it is latched by the Write-Back Enhanced Intel486 processors and serviced later when STPCLK# is deasserted.
- The processor will latch and respond to the inputs BOFF#, EADS#, AHOLD, and HOLD. The processor will not recognize any other inputs while in the Stop Grant state except FLUSH#. Other input signals to the processor will not be recognized until the CLK following the CLK in which STPCLK# is de-asserted. (See Figure 9-11.)
- The processor will generate a Stop Grant bus cycle only when entering that state from the Normal or the Auto HALT Power Down state. The Stop Grant bus cycle is not generated when the processor enters the Stop Grant state from the Stop Clock state or the Stop Clock Snoop state.
- The processor will not enter the Stop Grant state until all the pending writes are completed, all pending interrupts are serviced and the processor is idle.

9.6.5.3 Stop Clock State

Stop Clock state is the lowest power consumption mode in the processor, because it allows removal of the external clock. It also has the longest latency for returning to normal state. The **Stop Clock** state is entered from the Stop Grant state by stopping the CLK input. In the Stop Clock state, total processor power consumption drops to 100 μA, which is approximately 200–250 times lower than the Stop Grant state. None of the processor input signals should change state while the CLK input is stopped. Any transition on an input signal before the processor has returned to the Stop Grant state will result in unpredictable behavior. If INTR is driven active, it must be held active until the processor issues an interrupt acknowledge cycle.

In the Stop Clock state, the processor is dormant. It does not respond to any transitions on any of the input pins including snoops, flushes and interrupts. It is recommended that this mode only be entered if the processor cache is coherent with main memory and the processor is not processing any interrupts. If this mode is entered with a dirty cache, no alternate master cycles can be allowed while the processor is in the Stop Clock state.

The processor will return to the Stop Grant state after the CLK input has been running at a constant frequency for a period of time equal to the PLL start-up latency. The CLK input can be restarted to any frequency between the minimum and maximum frequency listed in the AC timing specifications.

In Enhanced Bus Mode

If the processor is taken into the Stop Clock state with a dirty cache, alternate bus master cycles are not allowed while the processor remains in the Stop Clock state. In order to take the processor into the Stop Clock state with a clean cache, the cache must be flushed. During the time the cache is being flushed, the system must block interrupts to the processor. With all interrupts other than STPCLK# blocked, the processor does not write into the cache during the time from the completion of the flush and time it enters the Stop Grant state. This is necessary for the cache to be coherent. To ensure this, the system should drive KEN# inactive from the time the flush starts until the Stop Grant cycle is issued. The system can then put the processor in the Stop Clock state by stopping the CLOCK.

If the processor is already in the Stop Grant state and entering the Stop Clock state is desired, the system must de-assert STPCLK# before flushing the cache in order to ensure the cache coherency. The 5-clock de-assertion specification for STPCLK# must also be met before the above sequence can occur.

9.6.5.4 Auto HALT Power-Down State

Upon execution of a HALT instruction, the processor will automatically enter a low power state, called the **Auto HALT Power-Down state.** The processor will issue a normal HALT bus cycle when entering this state. Because interrupts are HALT break events, the processor will transition to the Normal state on the occurrence of INTR, NMI, SMI# or RESET. (SRESET is also a HALT break event.) If there is a FLUSH# while the processor is in this state, the FLUSH# will be serviced by transitioning to the Stop Clock Flush state. After the FLUSH# is completed, the processor returns back to the Auto HALT Power-Down state.

The system can generate a STPCLK# while the processor is in the Auto HALT Power-Down state. The processor will then generate a Stop Grant bus cycle and enter the Stop Grant state from the Auto HALT Power-Down state. When the system de-asserts the STPCLK# interrupt, the processor will return to the Auto HALT Power-Down state. The processor will not generate a new HALT bus cycle when it re-enters the Auto HALT Power-Down state from the Stop Grant state.

9.6.6 STOP CLOCK SNOOP STATE (CACHE INVALIDATIONS)

When the processor is in the Stop Grant state or the Auto HALT Power-Down state, the processor will recognize HOLD, AHOLD, BOFF#, and EADS# for cache invalidation. When the system asserts HOLD, AHOLD, or BOFF#, the processor will float the bus accordingly. When the system asserts EADS#, the processor will transparently enter the **Stop Clock Snoop state** and will power up in order to perform the required cache snoop cycle and write-back cycles. It will then refreeze the CLK to the processor core and return to the previous state (i.e., either the Stop Grant state or the Auto HALT Power-Down state). The processor does not generate a bus cycle when it returns to the previous state.

9.6.6.1 Auto HALT Power-Down Flush State (Cache Flush) for the Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processors

If the Write-Back Enhanced Intel486 processor is in either Standard or Enhanced mode and a FLUSH# event occurs during Auto HALT Power-Down state, the processor will transition to the Auto HALT Power-Down Flush state. If the on-chip cache is configured as a write-back cache, the CLK to the processor core is turned on until all the dirty lines are written back, the cache is invalidated, and the two flush acknowledge cycles are completed. If the on-chip cache is configured as a write-through cache, the CLK to the processor core is turned on until the cache is invalidated. The processor then refreezes the CLK and returns to the previous state (i.e., the Auto HALT Power-Down state). Auto HALT Power-Down Flush state is entered only from the Auto HALT Power-Down state and not from the Stop Grant state.

9.6.7 SUPPLY CURRENT MODEL FOR STOP CLOCK MODES AND TRANSITIONS

Figures 9-12 and 9-13 illustrate the effect of different Stop Clock state transitions on the supply current of the Intel486 processor.

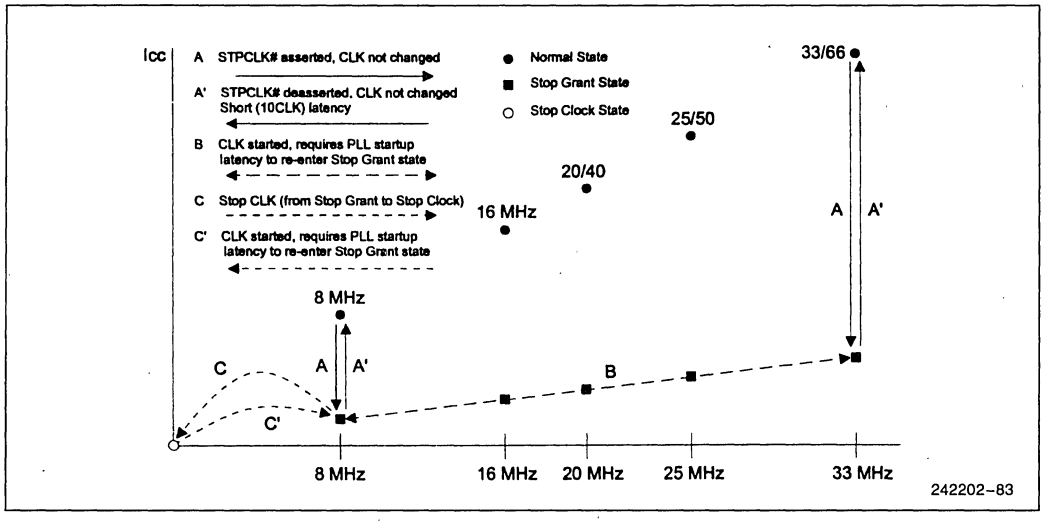


Figure 9-12. Supply Current Model for Stop Clock Modes and Transitions for the Intel486™ Processor

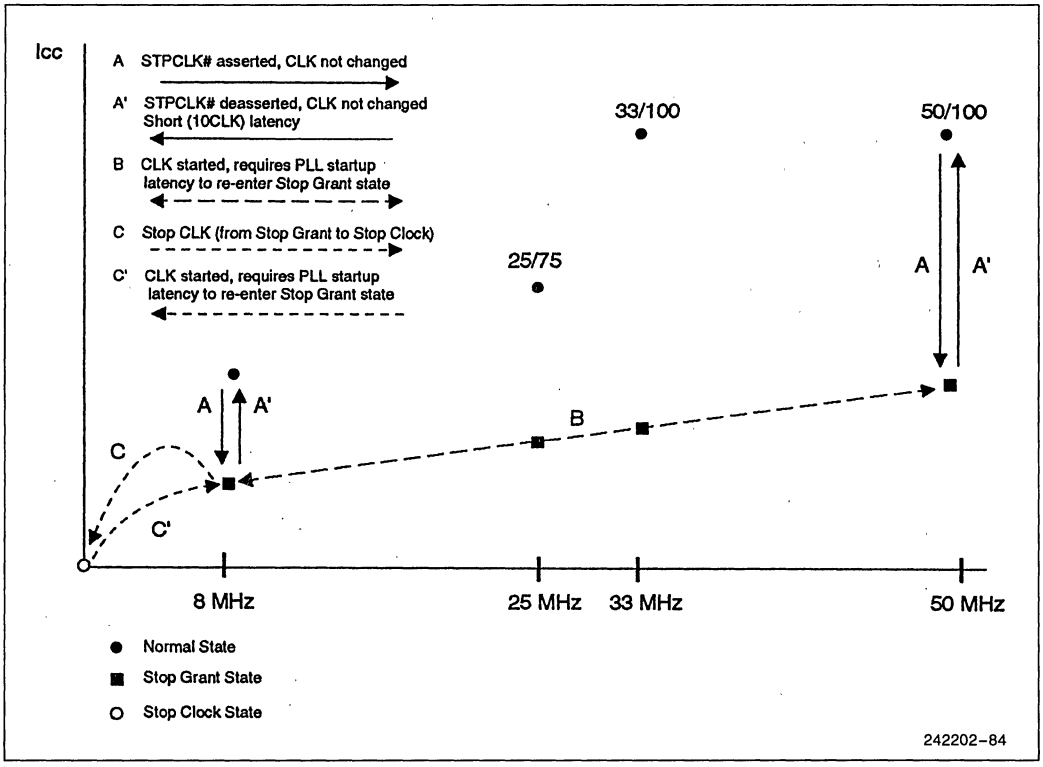


Figure 9-13. Supply Current Model for Stop Clock Modes and Transitions for the IntelDX4™ Processor

4

10.0 BUS OPERATION

All Intel486 processors operate in Standard Bus (write-through) mode. However, when the internal cache of the Write-Back Enhanced IntelDX2 processor is configured in write-back mode, the processor bus operates in the Enhanced Bus mode, which is described in section 10.3. **When the internal cache of the Write-Back Enhanced Intel486 processors is configured in write-through mode, the processor bus operates in Standard Bus mode, identical to the other Intel486 processors in Standard Bus mode.**

10.1 Data Transfer Mechanism

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte, word and doubleword lengths may be transferred without restrictions on physical address alignment. Data may be accessed at any byte boundary but two or three cycles may be required for unaligned data transfers. (See section 10.1.2, "Dynamic Data Bus Sizing," and section 10.1.5, "Operand Alignment.")

The Intel486 processor address signals are split into two components. High-order address bits are provided by the address lines, A2–A31. The byte enables, BE0#–BE3#, form the low-order address and provide linear selects for the four bytes of the 32-bit address bus.

The byte enable outputs are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 10-1. Byte enable patterns that have a negated byte enable separating two or three asserted byte enables will never occur (see Table 10-5). All other byte enable patterns are possible.

Table 10-1. Byte Enables and Associated Data and Operand Bytes

Byte Enable Signal	Associated Data Bus Signals	
BE0#	D0–D7	(byte 0–least significant)
BE1#	D8–D15	(byte 1)
BE2#	D16–D23	(byte 2)
BE3#	D24–D31	(byte 3–most significant)

Address bits A0 and A1 of the physical operand's base address can be created when necessary. Use of the byte enables to create A0 and A1 is shown in Table 10-2. The byte enables can also be decoded to generate BLE# (byte low enable) and BHE# (byte high enable). These signals are needed to address 16-bit memory systems. (See section 10.1.3, "Interfacing with 8-, 16-, and 32-Bit Memories.")

10.1.1 MEMORY AND I/O SPACES

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. Physical memory addresses range from 00000000H to FFFFFFFFH (4 gigabytes). I/O addresses range from 00000000H to 0000FFFFH (64 Kbytes) for programmed I/O. (See Figure 10-1.)

Table 10-2. Generating A0–A31 from BE0#–BE3# and A2–A31

Intel486™ Processor Address Signals								
A31	**...**	**A2**			**BE3#**	**BE2#**	**BE1#**	**BE0#**
	Physical Base Address							
A31	...	A2	A1	A0				
A31	...	A2	0	0	X	X	X	Low
A31	...	A2	0	1	X	X	Low	High
A31	...	A2	1	0	X	Low	High	High
A31	...	A2	1	1	Low	High	High	High

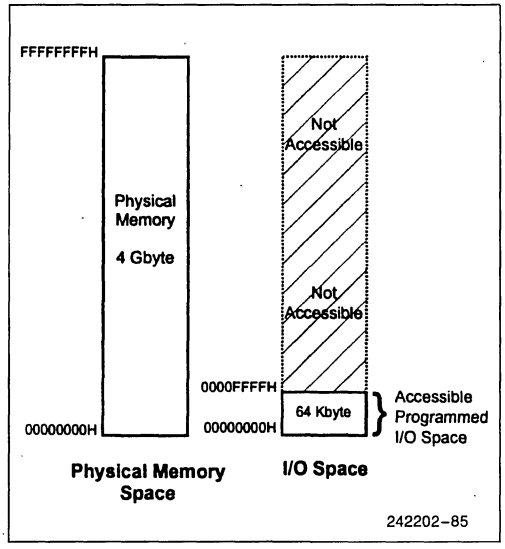


Figure 10-1. Physical Memory and I/O Spaces

10.1.1.1 Memory and I/O Space Organization

The Intel486 processor datapath to memory and input/output (I/O) spaces can be 32-, 16- or 8-bits wide. The byte enable signals, BE0#–BE3#, allow byte granularity when addressing any memory or I/O structure whether 8, 16 or 32 bits wide.

The Intel486 processor includes bus control pins, BS16# and BS8#, which allow direct connection to 16- and 8-bit memories and I/O devices. Cycles to 32-, 16- and 8-bit may occur in any sequence, since the BS8# and BS16# signals are sampled during each bus cycle.

32-bit wide memory and I/O spaces are organized as arrays of physical 4-byte words. Each memory or I/O 4-byte word has four individually addressable bytes at consecutive byte addresses (see Figure 10-2). The lowest addressed byte is associated with data signals D0–D7; the highest-addressed byte with D24–D31. Physical 4-byte words begin at addresses divisible by four.

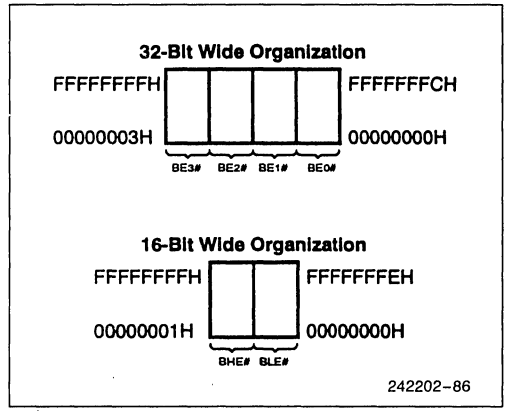


Figure 10-2. Physical Memory and I/O Space Organization

16-bit memories are organized as arrays of physical 2-byte words. Physical 2-byte words begin at addresses divisible by two. The byte enables BE0#–BE3#, must be decoded to A1, BLE# and BHE# to address 16-bit memories. (See section 10.1.3, "Interfacing with 8-, 16- and 32-Bit Memories.")

To address 8-bit memories, the two low order address bits A0 and A1, must be decoded from BE0#–BE3#. The same logic can be used for 8- and 16-bit memories, because the decoding logic for BLE# and A0 are the same. (See section 10.1.3, "Interfacing with 8-, 16-, and 32-Bit Memories.")

10.1.2 DYNAMIC DATA BUS SIZING

Dynamic data bus sizing is a feature allowing processor connection to 32-, 16- or 8-bit buses for memory or I/O. The Intel486 processor may connect to all three bus sizes. Transfers to or from 32-, 16- or 8-bit devices are supported by dynamically determining the bus width during each bus cycle. Address decoding circuitry may assert BS16# for 16-bit devices, or BS8# for 8-bit devices during each bus cycle. BS8# and BS16# must be negated when addressing 32-bit devices. An 8-bit bus width is selected if both BS16# and BS8# are asserted.

BS16# and BS8# force the Intel486 processor to run additional bus cycles to complete requests larger than 16 or 8 bits. A 32-bit transfer will be converted into two 16-bit transfers (or 3 transfers if the data is misaligned) when BS16# is asserted. Asserting BS8# will convert a 32-bit transfer into four 8-bit transfers.

Extra cycles forced by BS16# or BS8# should be viewed as independent bus cycles. BS16# or BS8# must be driven active during each of the extra cycles unless the addressed device has the ability to change the number of bytes it can return between cycles.

The Intel486 processor will drive the byte enables appropriately during extra cycles forced by BS8# and BS16#. A2–A31 will not change if accesses are to a 32-bit aligned area. Table 10-3 shows the set of byte enables that will be generated on the next cycle for each of the valid possibilities of the byte enables on the current cycle.

The dynamic bus sizing feature of the Intel486 processor is significantly different than that of the Intel386 processor. Unlike the Intel386 processor, the Intel486 processor requires that data bytes be driven on the addressed data pins. The simplest example of this function is a 32-bit aligned, BS16# read. When the Intel486 processor reads the two high-order bytes, they must be driven on the data bus pins D16–D31. The Intel486 processor expects the two low-order bytes on D0–D15. The Intel386 processor expects both the high- and low-order bytes on D0–D15. The Intel386 processor always reads or writes data on the lower 16 bits of the data bus when BS16# is asserted.

The external system must contain buffers to enable the Intel486 processor to read and write data on the appropriate data bus pins. Table 10-4 shows the data bus lines to which the Intel486 processor expects data to be returned for each valid combination of byte enables and bus sizing options.

Table 10-3. Next Byte Enable Values for BSn# Cycles

Current				Next with BS8#				Next with BS16#			
BE3#	BE2#	BE1#	BE0#	BE3#	BE2#	BE1#	BE0#	BE3#	BE2#	BE1#	BE0#
1	1	1	0	n	n	n	n	n	n	n	n
1	1	0	0	1	1	0	1	n	n	n	n
1	0	0	0	1	0	0	1	1	0	1	1
0	0	0	0	0	0	0	1	0	0	1	1
1	1	0	1	n	n	n	n	n	n	n	n
1	0	0	1	1	0	1	1	1	0	1	1
0	0	0	1	0	0	1	1	0	0	1	1
1	0	1	1	n	n	n	n	n	n	n	n
0	0	1	1	0	1	1	1	n	n	n	n
0	1	1	1	n	n	n	n	n	n	n	n

NOTE: "n" means that another bus cycle will not be required to satisfy the request.

Table 10-4. Data Pins Read with Different Bus Sizes

BE3#	BE2#	BE1#	BE0#	w/o BS8#/BS16#	w BS8#	w BS16#
1	1	1	0	D7–D0	D7–D0	D7–D0
1	1	0	0	D15–D0	D7–D0	D15–D0
1	0	0	0	D23–D0	D7–D0	D15–D0
0	0	0	0	D31–D0	D7–D0	D15–D0
1	1	0	1	D15–D8	D15–D8	D15–D8
1	0	0	1	D23–D8	D15–D8	D15–D8
0	0	0	1	D31–D8	D15–D8	D15–D8
1	0	1	1	D23–D16	D23–D16	D23–D16
0	0	1	1	D31–D16	D23–D16	D31–D16
0	1	1	1	D31–D24	D31–D24	D31–D24

Valid data will only be driven onto data bus pins corresponding to active byte enables during write cycles. Other pins in the data bus will be driven but they will not contain valid data. Unlike the Intel386 processor, the Intel486 processor will not duplicate write data onto parts of the data bus for which the corresponding byte enable is negated.

10.1.3 INTERFACING WITH 8-, 16- AND 32-BIT MEMORIES

In 32-bit physical memories, such as the one shown in Figure 10-3, each 4-byte word begins at a byte address that is a multiple of four. A2–A31 are used as a 4-byte word select. BE0#–BE3# select individual bytes within the 4-byte word. BS8# and BS16# are negated for all bus cycles involving the 32-bit array.

16- and 8-bit memories require external byte swapping logic for routing data to the appropriate data lines and logic for generating BHE#, BLE# and A1. In systems where mixed memory widths are used, extra address decoding logic is necessary to assert BS16# or BS8#.

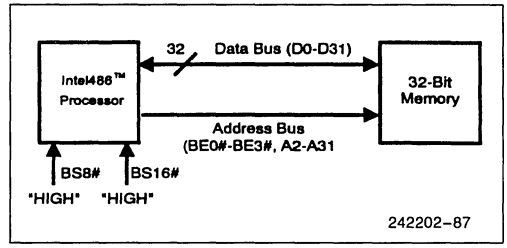


Figure 10-3. Intel486™ Processor with 32-Bit Memory

Figure 10-4 shows the Intel486 processor address bus interface to 32-, 16- and 8-bit memories. To address 16-bit memories the byte enables must be decoded to produce A1, BHE# and BLE# (A0). For 8-bit wide memories the byte enables must be decoded to produce A0 and A1. The same byte select logic can be used in 16- and 8-bit systems, because BLE# is exactly the same as A0. (See Table 10-5.)

BE0#–BE3# can be decoded as shown in Table 10-5 to generate A1, BHE# and BLE#. The byte select logic necessary to generate BHE# and BLE# is shown in Figure 10-5.

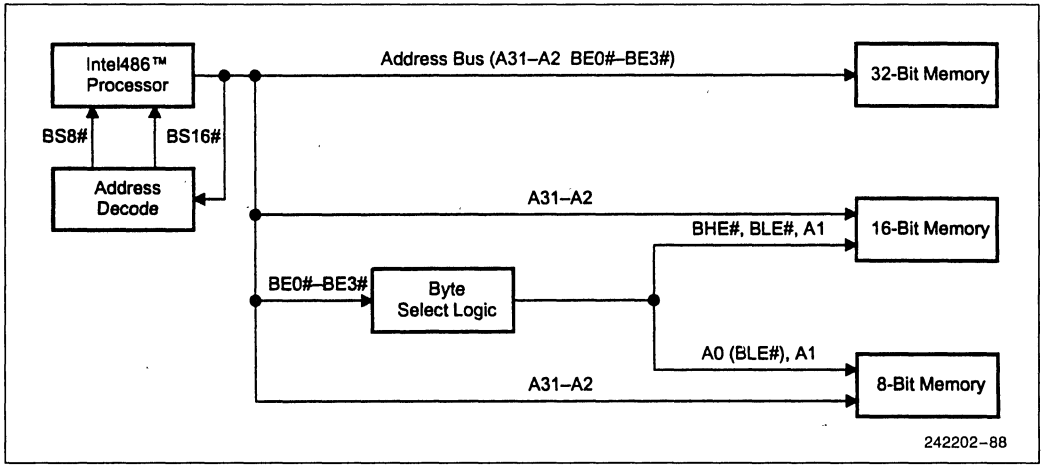


Figure 10-4. Addressing 16- and 8-Bit Memories

Table 10-5. Generating A1, BHE# and BLE# for Addressing 16-Bit Devices

Intel486™ Processor				8-, 16-Bit Bus Signals			Comments
BE3#	BE2#	BE1#	BE0#	A1	BHE#	BLE# (A0)	
H*	H*	H*	H*	x	x	x	x–no active bytes
H	H	H	L	L	H	L	
H	H	L	H	L	L	H	
H	H	L	L	L	L	L	
H	L	H	H	H	H	L	
H*	L*	H*	L*	x	x	x	x–not contiguous bytes
H	L	L	H	L	L	H	
H	L	L	L	L	L	L	
L	H	H	H	H	L	H	
L*	H*	H*	L*	x	x	x	x–not contiguous bytes
L*	H*	L*	H*	x	x	x	x–not contiguous bytes
L*	H*	L*	L*	x	x	x	x–not contiguous bytes
L		H	H	H	L	L	
L*	L*	H*	L*	x	x	x	x–not contiguous bytes
L	L	L	H	L	L	H	
L	L	L	L	L	L	L	

BLE# asserted when D0–D7 of 16-bit bus is active.
BHE# asserted when D8–D15 of 16-bit bus is active.
A1 low for all even words; A1 high for all odd words.

Key:
x = don't care H = high voltage level L = low voltage level
* = a non-occurring pattern of Byte Enables; either none are asserted or the pattern has Byte Enables asserted for non-contiguous bytes

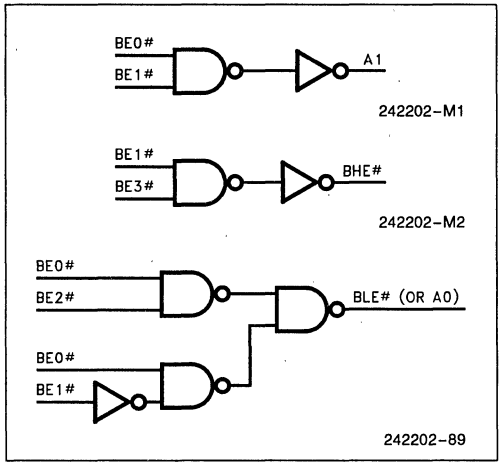


Figure 10-5. Logic to Generate A1, BHE# and BLE# for 16-Bit Buses

Combinations of BE0#–BE3# that never occur are those in which two or three asserted byte enables are separated by one or more negated byte enables. These combinations are "don't care" conditions in the decoder. A decoder can use the non-occurring BE0#–BE3# combinations to its best advantage.

Figure 10-6 shows an Intel486 processor data bus interface to 16- and 8-bit wide memories. External byte swapping logic is needed on the data lines so that data is supplied to and received from the Intel486 processor on the correct data pins (see Table 10-4).

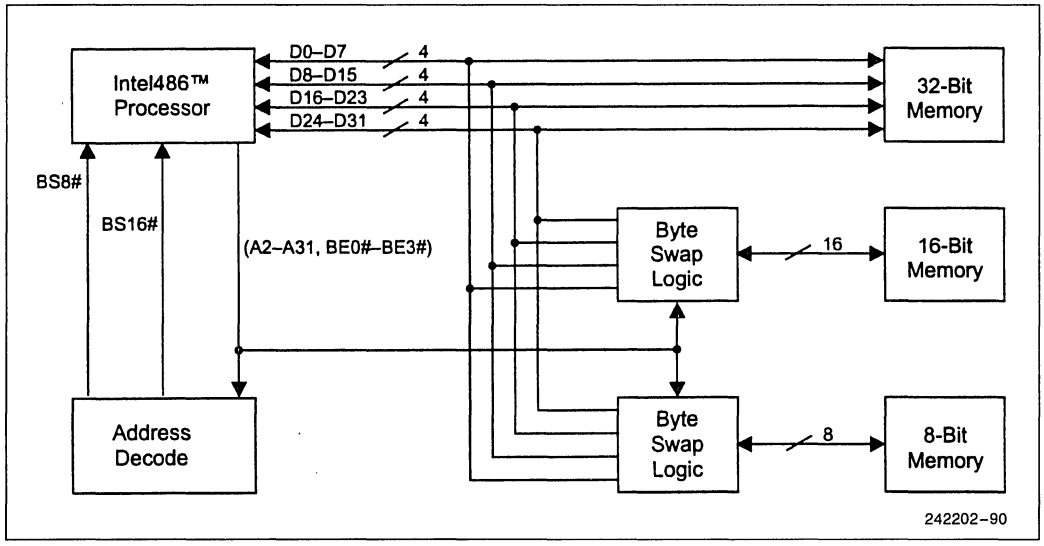


Figure 10-6. Data Bus Interface to 16- and 8-Bit Memories

10.1.4 DYNAMIC BUS SIZING DURING CACHE LINE FILLS

BS8# and BS16# can be driven during cache line fills. The Intel486 processor will generate enough 8- or 16-bit cycles to fill the cache line. This can be up to sixteen 8-bit cycles.

The external system should assume that all byte enables are active for the first cycle of a cache line fill. The Intel486 processor will generate proper byte enables for subsequent cycles in the line fill. Table 10-6 shows the appropriate A0 (BLE#), A1 and BHE# for the various combinations of the Intel486 processor byte enables on both the first and subsequent cycles of the cache line fill. The "*" marks all combinations of byte enables that will be generated by the Intel486 processor during a cache line fill.

10.1.5 OPERAND ALIGNMENT

Physical 4-byte words begin at addresses that are multiples of four. It is possible to transfer a logical operand that spans more than one physical 4-byte word of memory or I/O at the expense of extra cycles. Examples are 4-byte operands beginning at addresses that are not evenly divisible by 4, or 2-byte words split between two physical 4-byte words. These are referred to as unaligned transfers.

Operand alignment and data bus size dictate when multiple bus cycles are required. Table 10-7 describes the transfer cycles generated for all combinations of logical operand lengths, alignment, and data bus sizing. When multiple cycles are required to transfer a multibyte logical operand, the highest-order bytes are transferred first. For example, when the processor does a 4-byte unaligned read beginning at location x11 in the 4-byte aligned space, the three high order bytes are read in the first bus cycle. The low byte is read in a subsequent bus cycle.

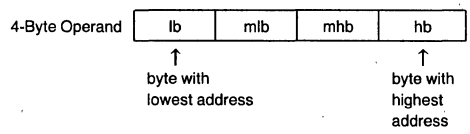
Table 10-6. Generating A0, A1 and BHE# from the Intel486™ Processor Byte Enables

				First Cache Fill Cycle			Any Other Cycle		
BE3#	BE2#	BE1#	BE0#	A0	A1	BHE#	A0	A1	BHE#
1	1	1	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
*0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	1	0	0
1	0	0	1	0	0	0	1	0	0
*0	0	0	1	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	1
*0	0	1	1	0	0	0	0	1	0
*0	1	1	1	0	0	0	1	1	0

Table 10-7. Transfer Bus Cycles for Bytes, Words and Dwords

	Byte-Length of Logical Operand								
	1	2				4			
Physical Byte Address in Memory (Low Order Bits)	xx	00	01	10	11	00	01	10	11
Transfer Cycles over 32-Bit Bus	b	w	w	w	hb lb	d	hb l3	hw lw	h3 lb
Transfer Cycles over 16-Bit Bus († = BS#16 asserted)	b	w	lb † hb†	w	hb lb	lw † hw†	hb lb† mw†	hw lw	mw † hb† lb
Transfer Cycles over 8-Bit Bus (‡ = BS8# Asserted)	b	lb ‡ hb ‡	lb ‡ hb‡	lb ‡ hb ‡	hb lb	lb ‡ mlb ‡ mhb ‡ hb ‡	hb lb ‡ mlb ‡ mhb ‡	mhb ‡ hb ‡ lb ‡ mlb ‡	mlb ‡ mhb ‡ hb ‡ kb

KEY:
b = byte transfer
w = 2-byte transfer
3 = 3-byte transfer
d = 4-byte transfer
h = high-order portion
l = low-order portion
m = mid-order portion



The function of unaligned transfers with dynamic bus sizing is not obvious. When the external system asserts BS16# or BS8# forcing extra cycles, low-order bytes or words are transferred first (opposite to the example above). When the Intel486 processor requests a 4-byte read and the external system asserts BS16#, the lower 2 bytes are read first followed by the upper 2 bytes.

In the unaligned transfer described above, the processor requested three bytes on the first cycle. If the external system asserted BS16# during this 3-byte transfer, the lower word is transferred first followed by the upper byte. In the final cycle the lower byte of the 4-byte operand is transferred as in the 32-bit example above.

10.2 Bus Functional Description

The Intel486 processor supports a wide variety of bus transfers to meet the needs of high performance systems. Bus transfers can be single cycle or multiple cycle, burst or non-burst, cacheable or non-cacheable, 8-, 16- or 32-bit, and pseudo-locked. To support multiprocessing systems there are cache invalidation cycles and locked cycles.

This section begins with basic non-cacheable non-burst single cycle transfers. It moves on to multiple cycle transfers and introduces the burst mode. Cacheability is introduced in section 10.2.3, "Cacheable Cycles." The remaining sections describe locked, pseudo-locked, invalidate, bus hold and interrupt cycles.

Bus cycles and data cycles are discussed in this section. A bus cycle is at least two clocks long and begins with ADS# active in the first clock and ready active in the last clock. Data is transferred to or from the Intel486 processor during a data cycle. A bus cycle contains one or more data cycles.

Refer to section 10.2.13, "Bus States," for a description of the bus states shown in the timing diagrams.

10.2.1 NON-CACHEABLE NON-BURST SINGLE CYCLE

10.2.1.1 No Wait States

The fastest non-burst bus cycle that the Intel486 processor supports is two clocks long. These cycles are called 2-2 cycles because reads and writes take two cycles each. The first "2" refers to reads and the second to writes.

For example, if a wait state needs to be added to the write, the cycle would be called 2-3.

Basic two clock read and write cycles are shown in Figure 10-7. The Intel486 processor initiates a cycle by asserting the address status signal (ADS#) at the rising edge of the first clock. The ADS# output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus, respectively.

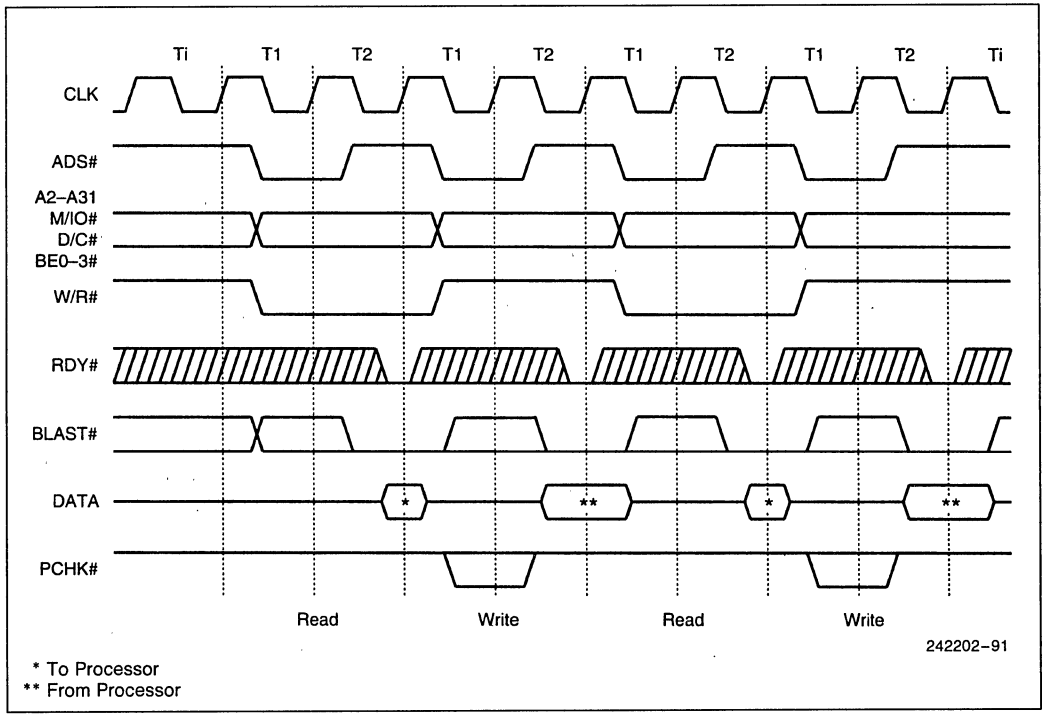


Figure 10-7. Basic 2-2 Bus Cycle

The non-burst ready input (RDY#) is returned by the external system in the second clock. RDY# indicates that the external system has presented valid data on the data pins in response to a read or the external system has accepted data in response to a write.

The Intel486 processor samples RDY# at the end of the second clock. The cycle is complete if RDY# is active (LOW) when sampled. Note that RDY# is ignored at the end of the first clock of the bus cycle.

The burst last signal (BLAST#) is asserted (LOW) by the Intel486 processor during the second clock of the first cycle in all bus transfers illustrated in Figure 10-7. This indicates that each transfer is complete after a single cycle. The Intel486 processor asserts BLAST# in the last cycle of a bus transfer.

The timing of the parity check output (PCHK#) is shown in Figure 10-7. The Intel486 processor drives the PCHK# output one clock after ready terminates a read cycle. PCHK# indicates the parity status for the data sampled at the end of the previous clock. The PCHK# signal can be used by the external system. The Intel486 processor does nothing in response to the PCHK# output.

10.2.1.2 Inserting Wait States

The external system can insert wait states into the basic 2-2 cycle by driving RDY# inactive at the end of the second clock. RDY# must be driven inactive to insert a wait state. Figure 10-8 illustrates a simple non-burst, non-cacheable signal with one wait state added. Any number of wait states can be added to an Intel486 processor bus cycle by maintaining RDY# inactive.

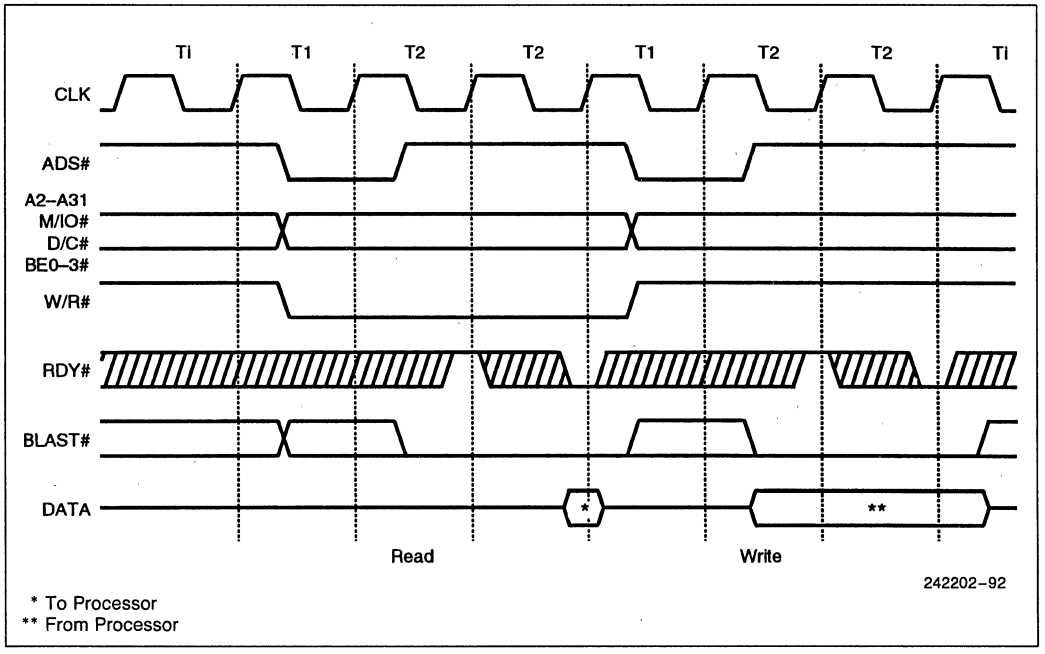


Figure 10-8. Basic 3-3 Bus Cycle

The burst ready input (BRDY#) must be driven inactive on all clock edges where RDY# is driven inactive for proper operation of these simple non-burst cycles.

10.2.2 MULTIPLE AND BURST CYCLE BUS TRANSFERS

Multiple cycle bus transfers can be caused by internal requests from the Intel486 processor or by the external memory system. An internal request for a 128-bit pre-fetch must take more than one cycle. Internal requests for unaligned data may also require multiple bus cycles. A cache line fill requires multiple cycles to complete.

The external system can cause a multiple cycle transfer when it can only supply 8- or 16-bits per cycle.

Only multiple cycle transfers caused by internal requests are considered in this section. Cacheable cycles and 8- and 16-bit transfers are covered in section 10.2.3, "Cacheable Cycles" and section 10.2.5, "8- and 16-Bit Cycles."

Internal Requests from Intel486 DX, IntelDX2, and IntelDX4 Processors

An internal request by an Intel486 DX, IntelDX2, or IntelDX4 processor for a 64-bit Floating-Point load must take more than one internal cycle.

10.2.2.1 Burst Cycles

The Intel486 processor can accept burst cycles for any bus requests that require more than a single data cycle. During burst cycles, a new data item is strobed into the Intel486 processor every clock rather than every other clock as in non-burst cycles. The fastest burst cycle requires 2 clocks for the first data item with subsequent data items returned every clock.

The Intel486 processor is capable of bursting a maximum of 32 bits during a write. Burst writes can only occur if BS8# or BS16# is asserted. For example, the Intel486 processor can burst write four 8-bit operands or two 16-bit operands in a single burst cycle. But the Intel486 processor cannot burst multiple 32-bit writes in a single burst cycle.

Burst cycles begin with the Intel486 processor driving out an address and asserting ADS# in the same manner as non-burst cycles. The Intel486 processor indicates that it is willing to perform a burst cycle by holding the burst last signal (BLAST#) inactive in the second clock of the cycle. The external system indicates its willingness to do a burst cycle by returning the burst ready signal (BRDY#) active.

The addresses of the data items in a burst cycle will all fall within the same 16-byte aligned area (corresponding to an internal Intel486 processor cache line). A 16-byte aligned area begins at location XXXXXXX0 and ends at location XXXXXXXF. During a burst cycle, only BE0–3#, A_2, and A_3 may change. A_4–A_{31}, M/IO#, D/C#, and W/R# will remain stable throughout a burst. Given the first address in a burst, external hardware can easily calculate the address of subsequent transfers in advance. An external memory system can be designed to quickly fill the Intel486 processor internal cache lines.

Burst cycles are not limited to cache line fills. Any multiple cycle read request by the Intel486 processor can be converted into a burst cycle. The Intel486 processor will only burst the number of bytes needed to complete a transfer.

For example, the Intel486 DX, IntelDX2, Write-Back Enhanced IntelDX2 or IntelDX4 processor will burst eight bytes for a 64-bit Floating-Point non-cacheable read.

The external system converts a multiple cycle request into a burst cycle by returning BRDY# active rather than RDY# (non-burst ready) in the first cycle of a transfer. For cycles that cannot be burst, such as interrupt acknowledge and halt, BRDY# has the same effect as RDY#. BRDY# is ignored if both BRDY# and RDY# are returned in the same clock. Memory areas and peripheral devices that cannot perform bursting must terminate cycles with RDY#.

10.2.2.2 Terminating Multiple and Burst Cycle Transfers

The Intel486 processor drives BLAST# inactive for all but the last cycle in a multiple cycle transfer. BLAST# is driven inactive in the first cycle to inform the external system that the transfer could take additional cycles. BLAST# is driven active in the last cycle of the transfer indicating that the next time BRDY# or RDY# is returned the transfer is complete.

BLAST# is not valid in the first clock of a bus cycle. It should be sampled only in the second and subsequent clocks when RDY# or BRDY# is returned.

The number of cycles in a transfer is a function of several factors including the number of bytes the Intel486 processor needs to complete an internal request (1, 2, 4, 8, or 16), the state of the bus size inputs (BS8# and BS16#), the state of the cache enable input (KEN#) and alignment of the data to be transferred.

When the Intel486 processor initiates a request it knows how many bytes will be transferred and if the data is aligned. The external system must indicate whether the data is cacheable (if the transfer is a read) and the width of the bus by returning the state of the KEN#, BS8# and BS16# inputs one clock before RDY# or BRDY# is returned. The Intel486 processor determines how many cycles a transfer will take based on its internal information and inputs from the external system.

BLAST# is not valid in the first clock of a bus cycle because the Intel486 processor cannot determine the number of cycles a transfer will take until the external system returns KEN#, BS8# and BS16#. BLAST# should only be sampled in the second and subsequent clocks of a cycle when the external system returns RDY# or BRDY#.

The system may terminate a burst cycle by returning RDY# instead of BRDY#. BLAST# will remain deasserted until the last transfer. However, any transfers required to complete a cache line fill will follow the burst order, e.g., if burst order was 4, 0, C, 8 and RDY# was returned at after 0, the next transfers will be from C and 8.

10.2.2.3 Non-Cacheable, Non-Burst, Multiple Cycle Transfers

Figure 10-9 illustrates a 2 cycle non-burst, non-cacheable multiple cycle read. This transfer is simply a sequence of two single cycle transfers. The Intel486 processor indicates to the external system that this is a multiple cycle transfer by driving BLAST# inactive during the second clock of the first cycle. The external system returns RDY# active indicating that it will not burst the data. The external system also indicates that the data is not cacheable by returning KEN# inactive one clock before it returns RDY# active. When the Intel486 processor samples RDY# active it ignores BRDY#.

Each cycle in the transfer begins when ADS# is driven active and the cycle is complete when the external system returns RDY# active.

The Intel486 processor indicates the last cycle of the transfer by driving BLAST# active. The next RDY# returned by the external system terminates the transfer.

10.2.2.4 Non-Cacheable Burst Cycles

The external system converts a multiple cycle request into a burst cycle by returning BRDY# active rather than RDY# in the first cycle of the transfer. This is illustrated in Figure 10-10.

There are several features to note in the burst read. ADS# is only driven active during the first cycle of the transfer. RDY# must be driven inactive when BRDY# is returned active.

BLAST# behaves exactly as it does in the non-burst read. BLAST# is driven inactive in the second clock of the first cycle of the transfer indicating more cycles to follow. In the last cycle, BLAST# is driven active telling the external memory system to end the burst after returning the next BRDY#.

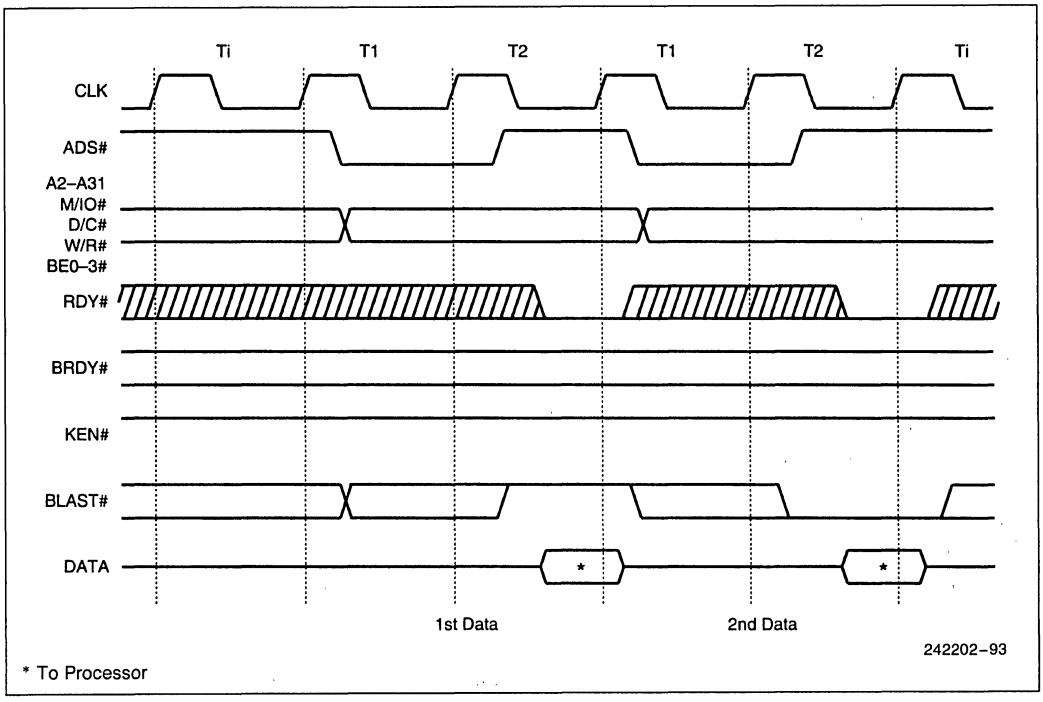


Figure 10-9. Non-Cacheable, Non-Burst, Multiple-Cycle Transfers

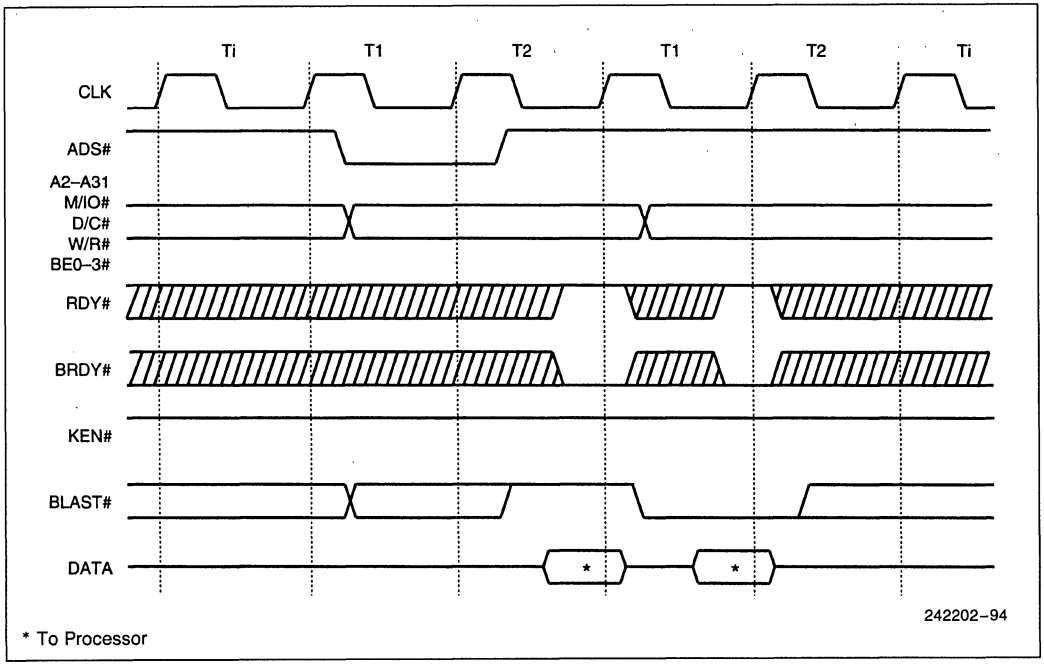


Figure 10-10. Non-Cacheable Burst Cycle

10.2.3 CACHEABLE CYCLES

Any memory read can become a cache fill operation. The external memory system can allow a read request to fill a cache line by returning KEN# active one clock before RDY# or BRDY# during the first cycle of the transfer on the external bus. Once KEN# is asserted and the remaining three requirements described below are met, the Intel486 processor will fetch an entire cache line regardless of the state of KEN#. KEN# must be returned active in the last cycle of the transfer for the data to be written into the internal cache. The Intel486 processor will only convert memory reads or prefetches into a cache fill.

KEN# is ignored during write or I/O cycles. Memory writes will only be stored in the on-chip cache if there is a cache hit. I/O space is never cached in the internal cache.

To transform a read or a prefetch into a cache line fill the following conditions must be met:

1. The KEN# pin must be asserted one clock prior to RDY# or BRDY# being returned for the first data cycle.
2. The cycle must be of the type that can be internally cached. (Locked reads, I/O reads, and interrupt acknowledge cycles are never cached).
3. The page table entry must have the page cache disable bit (PCD) set to 0. To cache a page table entry, the page directory must have PCD=0. To cache reads or prefetches when paging is disabled, or to cache the page directory entry, control register 3 (CR3) must have PCD=0.
4. The cache disable (CD) bit in control register 0 (CR0) must be clear.

External hardware can determine when the Intel486 processor has transformed a read or prefetch into a cache fill by examining the KEN#, M/IO#, D/C#, W/R#, LOCK#, and PCD pins. These pins convey to the system the outcome of conditions 1–3 in the above list. In addition, the Intel486 processor drives PCD high whenever the CD bit in CR0 is set, so that external hardware can evaluate condition 4.

Cacheable cycles can be burst or non-burst.

10.2.3.1 Byte Enables during a Cache Line Fill

For the first cycle in the line fill, the state of the byte enables should be ignored. In a non-cacheable memory read, the byte enables indicate the bytes actually required by the memory or code fetch.

The Intel486 processor expects to receive valid data on its entire bus (32 bits) in the first cycle of a cache line fill. Data should be returned with the assumption that all the byte enable pins are driven active. However if BS8# is asserted only one byte need be returned on data lines D0–D7. Similarly if BS16# is asserted two bytes should be returned on D0–D15.

The Intel486 processor will generate the addresses and byte enables for all subsequent cycles in the line fill. The order in which data is read during a line fill depends on the address of the first item read. Byte ordering is discussed in section 10.2.4, "Burst Mode Details."

10.2.3.2 Non-Burst Cacheable Cycles

Figure 10-11 shows a non-burst cacheable cycle. The cycle becomes a cache fill when the Intel486 processor samples KEN# active at the end of the first clock. The Intel486 processor drives BLAST# inactive in the second clock in response to KEN#. BLAST# is driven inactive because a cache fill requires three additional cycles to complete. BLAST# remains inactive until the last transfer in the cache line fill. KEN# must be returned active in the last cycle of the transfer for the data to be written into the internal cache.

Note that this cycle would be a single bus cycle if KEN# was not sampled active at the end of the first clock. The subsequent three reads would not have happened since a cache fill was not requested.

The BLAST# output is invalid in the first clock of a cycle. BLAST# may be active during the first clock due to earlier inputs. Ignore BLAST# until the second clock.

During the first cycle of the cache line fill the external system should treat the byte enables as if they are all active. In subsequent cycles in the burst, the Intel486 processor drives the address lines and byte enables. (See section 10.2.4.2, "Burst and Cache Line Fill Order') .

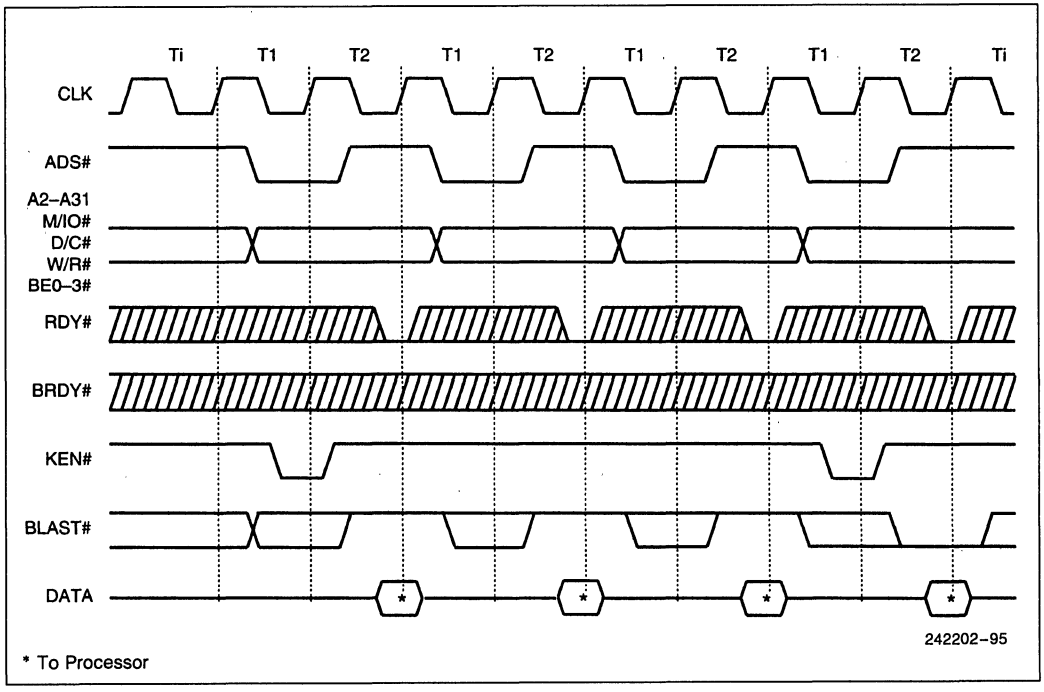


Figure 10-11. Non-Burst, Cacheable Cycles

10.2.3.3 Burst Cacheable Cycles

Figure 10-12 illustrates a burst mode cache fill. As in Figure 10-11, the transfer becomes a cache line fill when the external system returns KEN# active at the end of the first clock in the cycle.

The external system informs the Intel486 processor that it will burst the line in by driving BRDY# active at the end of the first cycle in the transfer.

Note that during a burst cycle, ADS# is only driven with the first address.

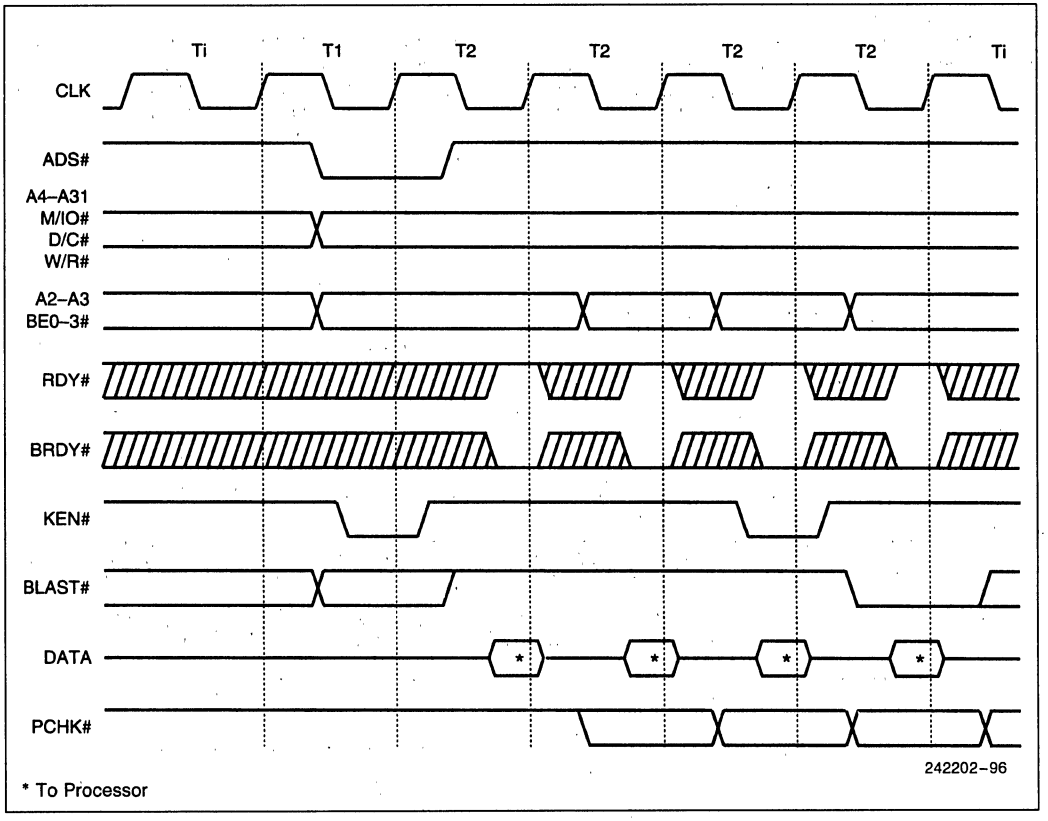


Figure 10-12. Burst Cacheable Cycle

10.2.3.4 Effect of Changing KEN# during a Cache Line Fill

KEN# can change multiple times as long as it arrives at its final value in the clock before RDY# or BRDY# is returned. This is illustrated in Figure 10-13. Note that the timing of BLAST# follows that of KEN# by one clock. The Intel486 processor samples KEN# every clock and uses the value returned in the clock before ready to determine if a bus cycle would be a cache line fill. Similarly, it uses the value of KEN# in the last cycle before early RDY# to load the line just retrieved from memory into the cache. KEN# is sampled every clock and it must satisfy setup and hold time.

KEN# can also change multiple times before a burst cycle, as long as it arrives at its final value one clock before ready is returned active.

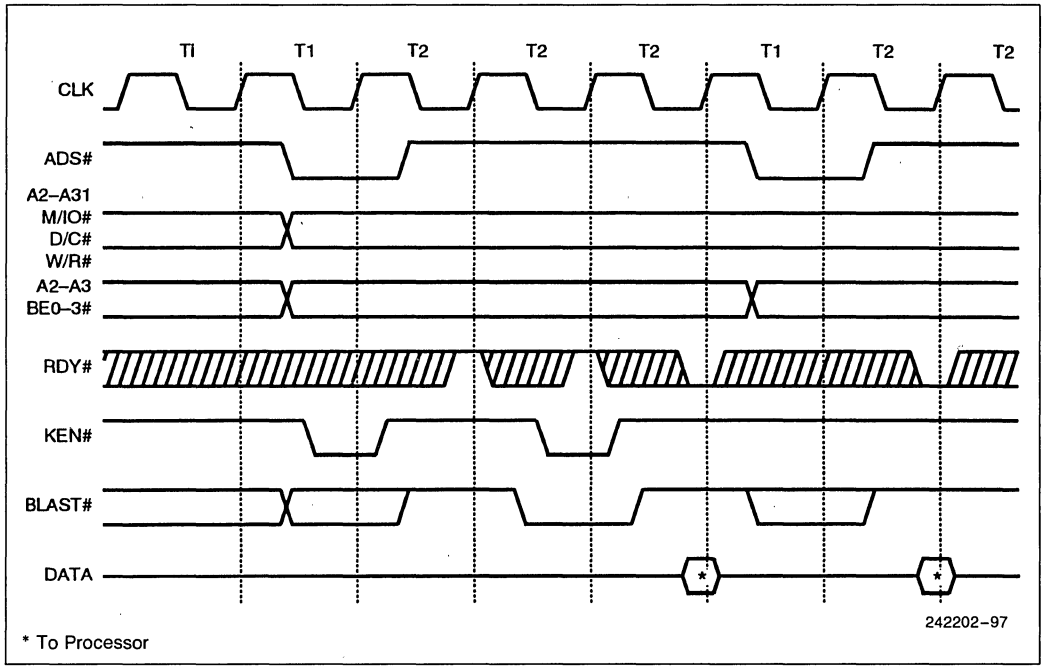


Figure 10-13. Effect of Changing KEN#

10.2.4 BURST MODE DETAILS

10.2.4.1 Adding Wait States to Burst Cycles

Burst cycles need not return data on every clock. The Intel486 processor will only strobe data into the chip when either RDY# or BRDY# is active. Driving BRDY# and RDY# inactive adds a wait state to the transfer. A burst cycle where two clocks are required for every burst item is shown in Figure 10-14.

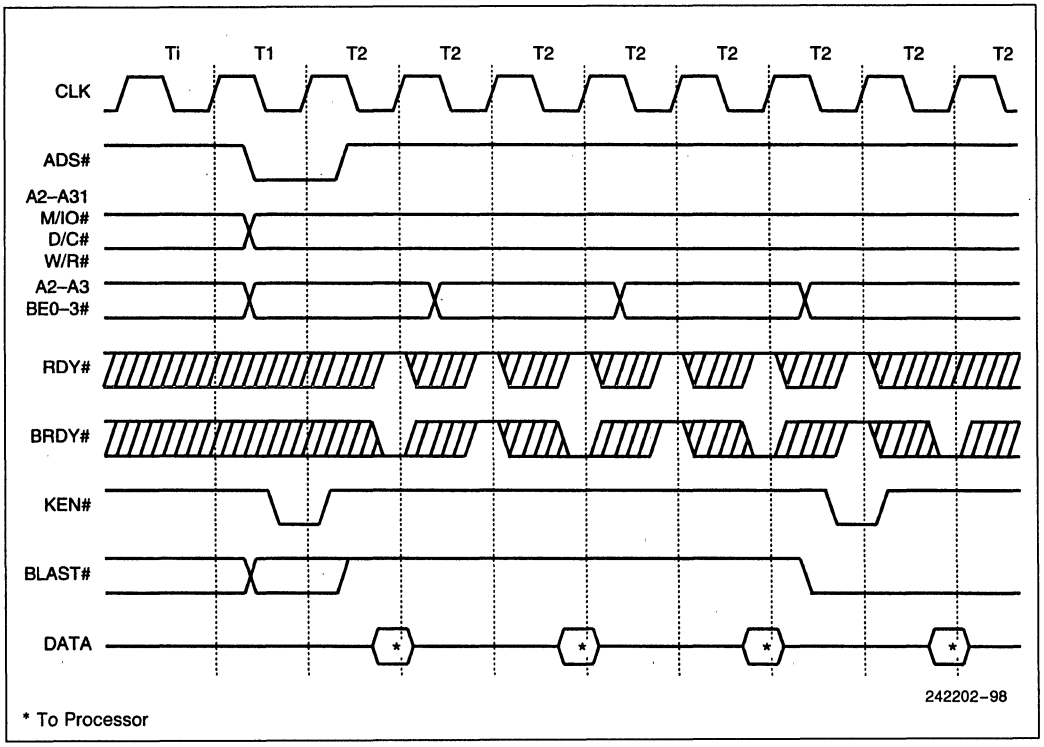


Figure 10-14. Slow Burst Cycle

10.2.4.2 Burst and Cache Line Fill Order

The burst order used by the Intel486 processor is shown in Table 10-8. This burst order is followed by any burst cycle (cache or not), cache line fill (burst or not) or code prefetch.

The Intel486 processor presents each request for data in an order determined by the first address in the transfer. For example, if the first address was 104 the next three addresses in the burst will be 100, 10C and 108. An example of burst address sequencing is shown in Figure 10-15.

Table 10-8. Burst Order (Both Read and Write Bursts)

First Addr.	Second Addr.	Third Addr.	Fourth Addr.
0	4	8	C
4	0	C	8
8	C	0	4
C	8	4	0

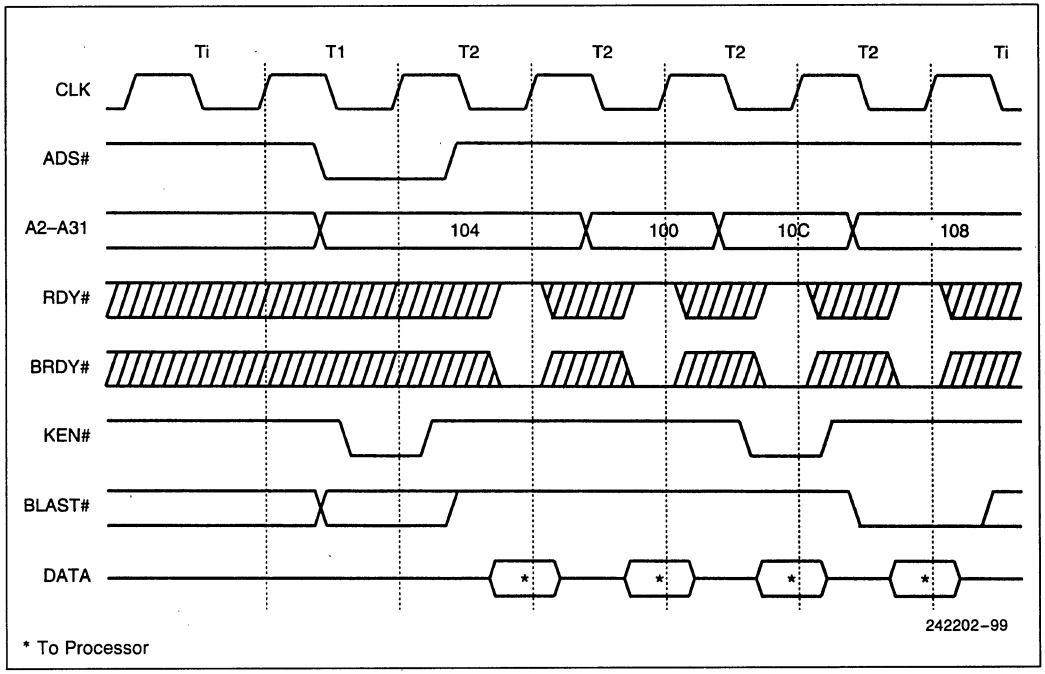


Figure 10-15. Burst Cycle Showing Order of Addresses

The sequences shown in Table 10-8 accommodate systems with 64-bit buses as well as systems with 32-bit data buses. The sequence applies to all bursts, regardless of whether the purpose of the burst is to fill a cache line, do a 64-bit read, or do a pre-fetch. If either BS8# or BS16# is returned active, the Intel486 processor completes the transfer of the current 32-bit word before progressing to the next 32-bit word. For example, a BS16# burst to address 4 has the following order: 4-6-0-2-C-E-8-A.

10.2.4.3 Interrupted Burst Cycles

Some memory systems may not be able to respond with burst cycles in the order defined in Table 10-8. To support these systems the Intel486 processor allows a burst cycle to be interrupted at any time. The Intel486 processor will automatically generate another normal bus cycle after being interrupted to complete the data transfer. This is called an interrupted burst cycle. The external system can respond to an interrupted burst cycle with another burst cycle.

The external system can interrupt a burst cycle by returning RDY# instead of BRDY#. RDY# can be returned after any number of data cycles terminated with BRDY#.

An example of an interrupted burst cycle is shown in Figure 10-16. The Intel486 processor immediately drives ADS# active to initiate a new bus cycle after RDY# is returned active. BLAST# is driven inactive one clock after ADS# begins the second bus cycle indicating that the transfer is not complete.

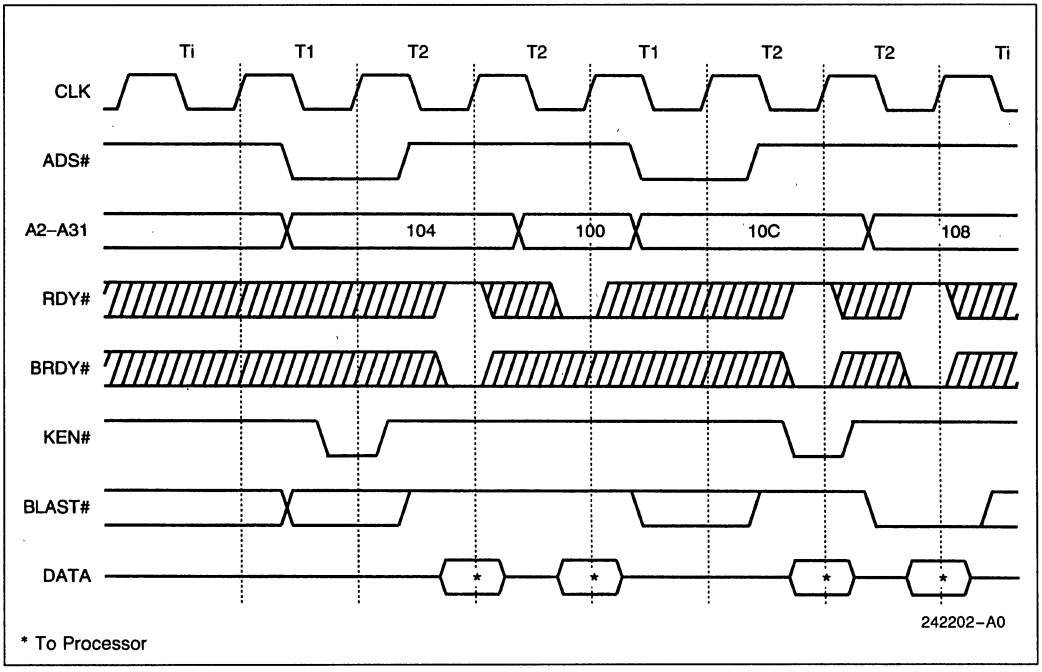


Figure 10-16. Interrupted Burst Cycle

KEN# need not be returned active in the first data cycle of the second part of the transfer in Figure 10-16. The cycle had been converted to a cache fill in the first part of the transfer and the Intel486 processor expects the cache fill to be completed. Note that the first half and second half of the transfer in Figure 10-16 are each 2-cycle burst transfers.

The order in which the Intel486 processor requests operands during an interrupted burst transfer is determined by Table 10-7. Mixing RDY# and BRDY# does not change the order in which operand addresses are requested by the Intel486 processor.

An example of the order in which the Intel486 processor requests operands during a cycle in which the external system mixes RDY# and BRDY# is shown in Figure 10-17. The Intel486 processor initially requests a transfer beginning at location 104. The transfer becomes a cache line fill when the external system returns KEN# active. The first cycle of the cache fill transfers the contents of location 104 and is terminated with RDY#. The Intel486 processor drives out a new request (by asserting ADS#) to address 100. If the external system terminates the second cycle with BRDY#, the Intel486 processor will next request/expect address 10C. The correct order is determined by the first cycle in the transfer, which may not be the first cycle in the burst if the system mixes RDY# with BRDY#.

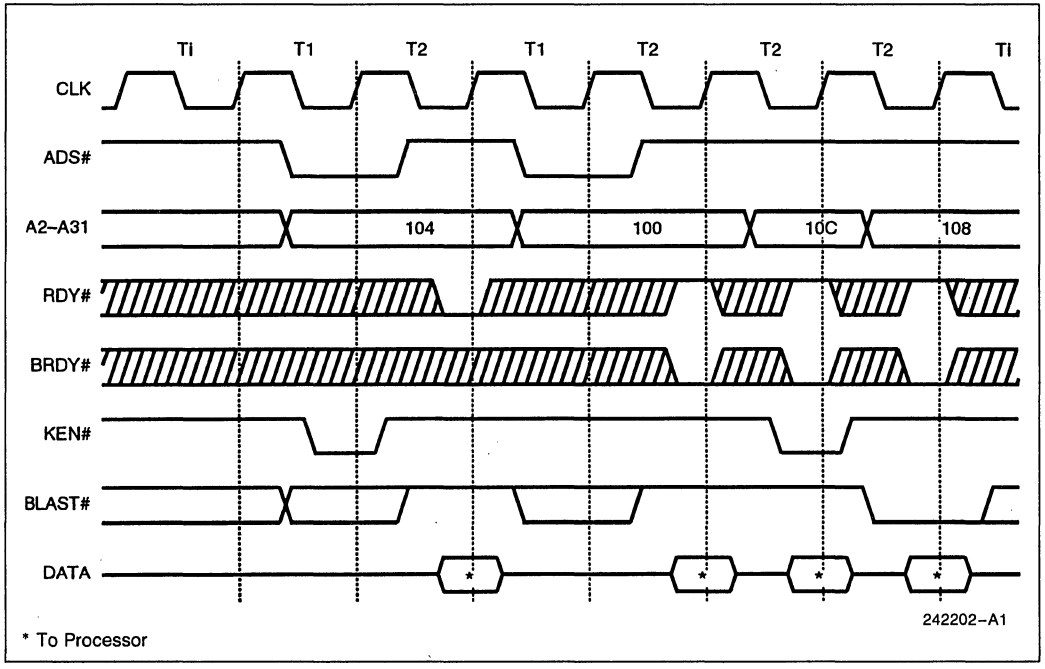


Figure 10-17. Interrupted Burst Cycle with Unobvious Order of Addresses

10.2.5 8- AND 16-BIT CYCLES

The Intel486 processor supports both 16- and 8-bit external buses through the BS16# and BS8# inputs. BS16# and BS8# allow the external system to specify, on a cycle-by-cycle basis, whether the addressed component can supply 8, 16 or 32 bits. BS16# and BS8# can be used in burst cycles as well as non-burst cycles. If both BS16# and BS8# are returned active for any bus cycle, the Intel486 processor will respond as if only BS8# were active.

The timing of BS16# and BS8# is the same as that of KEN#. BS16# and BS8# must be driven active before the first RDY# or BRDY# is driven active. Driving the BS16# and BS8# active can force the Intel486 processor to run additional cycles to complete what would have been only a single 32-bit cycle. BS8# and BS16# may change the state of BLAST# when they force subsequent cycles from the transfer.

Figure 10-18. shows an example in which BS8# forces the Intel486 processor to run two extra cycles to complete a transfer. The Intel486 processor issues a request for 24 bits of information. The external system drives BS8# active indicating that only eight bits of data can be supplied per cycle. The Intel486 processor issues two extra cycles to complete the transfer.

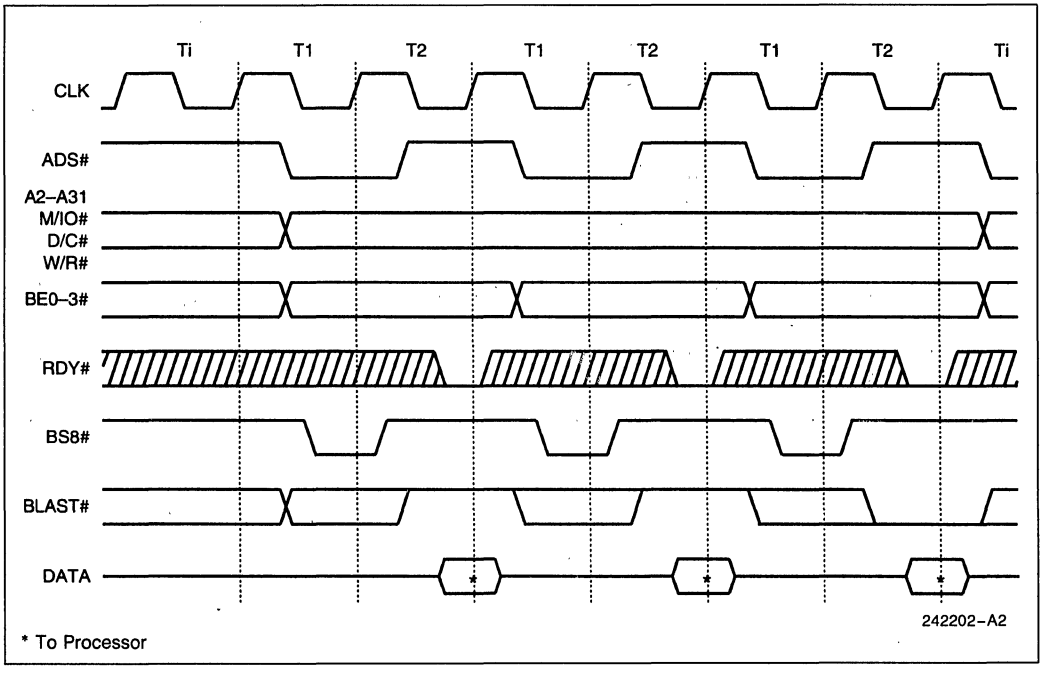


Figure 10-18. 8-Bit Bus Size Cycle

Extra cycles forced by the BS16# and BS8# should be viewed as independent bus cycles. BS16# and BS8# should be driven active for each additional cycle unless the addressed device has the ability to change the number of bytes it can return between cycles. The Intel486 processor will drive BLAST# inactive until the last cycle before the transfer is complete.

Refer to section 10.1.2, "Dynamic Data Bus Sizing," for the sequencing of addresses while BS8# or BS16# is active.

BS8# and BS16# operate during burst cycles in exactly the same manner as non-burst cycles. For example, a single non-cacheable read could be transferred by the Intel486 processor as four 8-bit burst data cycles. Similarly, a single 32-bit write could be written as four 8-bit burst data cycles. An example of a burst write is shown in Figure 10-19. Burst writes can only occur if BS8# or BS16# is asserted.

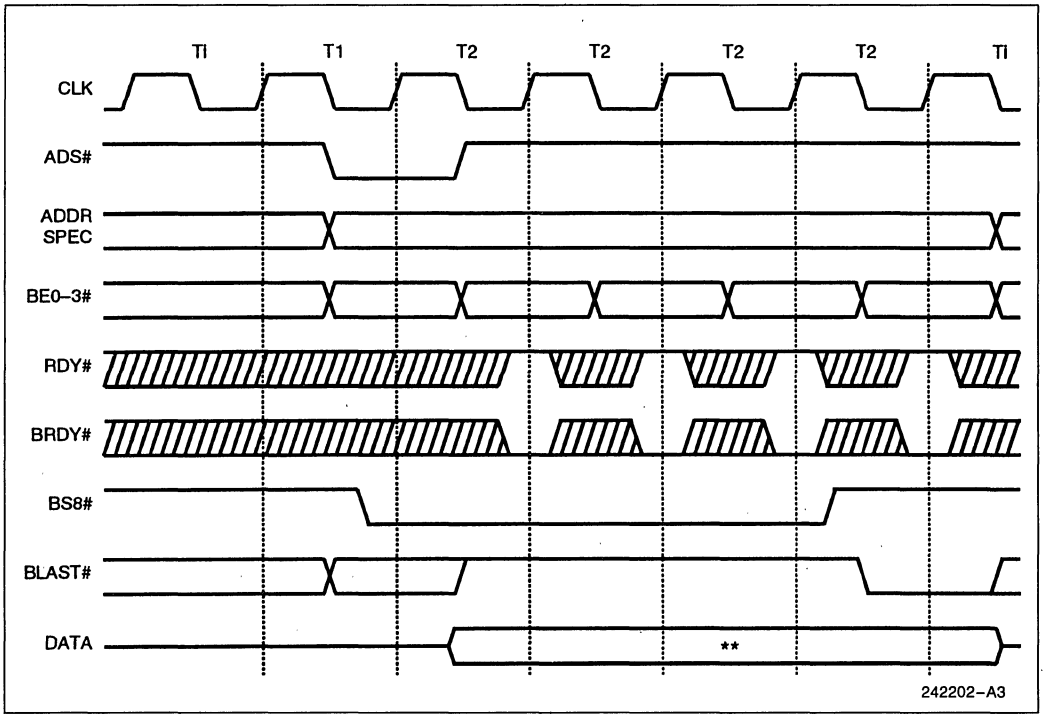


Figure 10-19. Burst Write as a Result of BS8# or BS16#

10.2.6 LOCKED CYCLES

Locked cycles are generated in software for any instruction that performs a read-modify-write operation. During a read-modify-write operation, the Intel486 processor can read and modify a variable in external memory and be assured that the variable is not accessed between the read and write.

Locked cycles are automatically generated during certain bus transfers. The xchg (exchange) instruction generates a locked cycle when one of its operands is memory-based. Locked cycles are generated when a segment or page table entry is updated and during interrupt acknowledge cycles. Locked cycles are also generated when the LOCK instruction prefix is used with selected instructions.

Locked cycles are implemented in hardware with the LOCK# pin. When LOCK# is active, the Intel486 processor is performing a read-modify-write operation and the external bus should not be relinquished until the cycle is complete. Multiple reads or writes can be locked. A locked cycle is shown in Figure 10-20. LOCK# goes active with the address and bus definition pins at the beginning of the first read cycle and remains active until RDY# is returned for the last write cycle. For unaligned 32-bit read-modify-write operation, the LOCK# remains active for the entire duration of the multiple cycle. It will go inactive when RDY# is returned for the last write cycle.

When LOCK# is active, the Intel486 processor will recognize address hold and backoff but will not recognize bus hold. It is left to the external system to properly arbitrate a central bus when the Intel486 processor generates LOCK#.

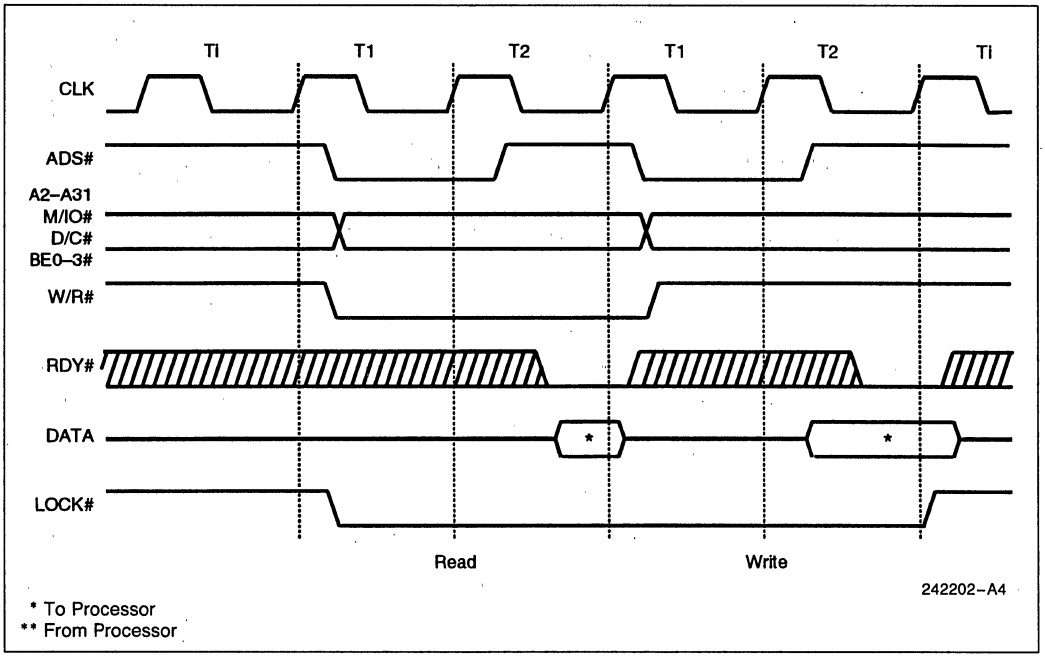


Figure 10-20. Locked Bus Cycle

10.2.7 PSEUDO-LOCKED CYCLES

Pseudo-locked cycles assure that no other master will be given control of the bus during operand transfers which take more than one bus cycle.

For the Intel486 processor, examples include 64-bit description loads and cache line fills.

Pseudo-locked transfers are indicated by the PLOCK# pin. The memory operands must be aligned for correct operation of a pseudo-locked cycle.

PLOCK# need not be examined during burst reads. A 64-bit aligned operand can be retrieved in one burst (note: this is only valid in systems that do not interrupt bursts).

The system must examine PLOCK# during 64-bit writes since the Intel486 processor cannot burst write more than 32 bits. However, burst can be used within each 32-bit write cycle if BS8# or BS16# is asserted. BLAST will be de-asserted in response to BS8# or BS16#. A 64-bit write will be driven out as two non-burst bus cycles. BLAST# is asserted during both writes since a burst is not possible. PLOCK# is asserted during the first write to indicate that another write follows. This behavior is shown in Figure 10-21.

The first cycle of a 64-bit Floating-Point write is the only case in which both PLOCK# and BLAST# are asserted. Normally PLOCK# and BLAST# are the inverse of each other.

During all of the cycles where PLOCK# is asserted, HOLD is not acknowledged until the cycle completes. This results in a large HOLD latency, especially when BS8# or BS16# is asserted. To reduce the HOLD latency during these cycles, windows are available between transfers to allow HOLD to be acknowledged during non-cacheable code prefetches. PLOCK# will be asserted since BLAST# is negated, but it is ignored and HOLD is recognized during the prefetch.

PLOCK# can change several times during a cycle settling to its final value in the clock ready is returned.

10.2.7.1 Floating-Point Read and Write Cycles

For Intel486 DX, IntelDX2, Write-Back Enhanced IntelDX2, and IntelDX4 processors, 64-bit Floating-Point read and write cycles are also examples of operand transfers that take more than one bus cycle.

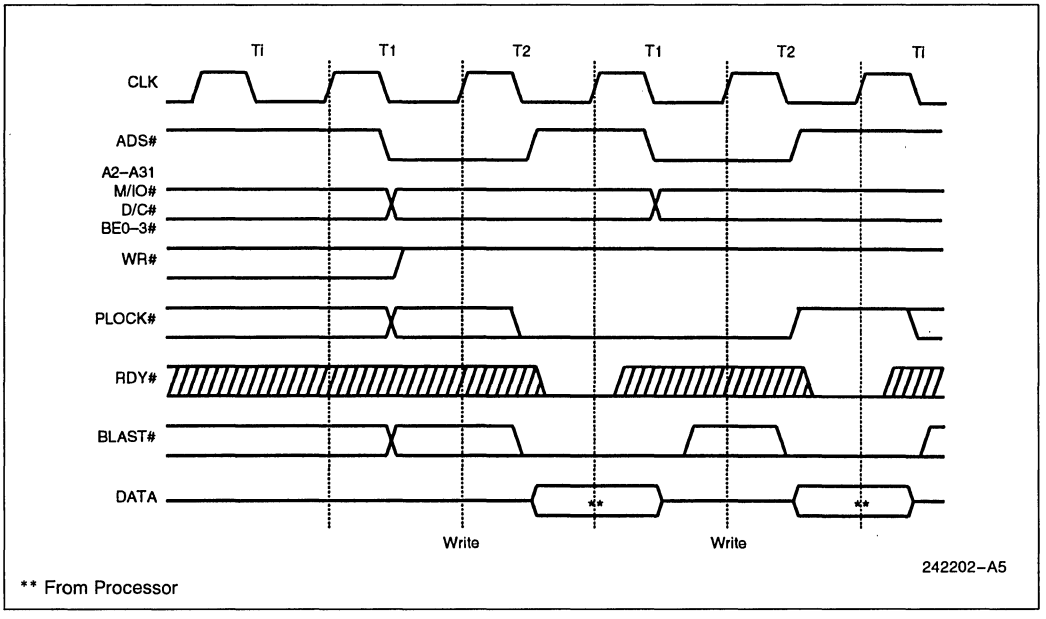


Figure 10-21. Pseudo Lock Timing

10.2.8 INVALIDATE CYCLES

Invalidate cycles are needed to keep the Intel486 processor internal cache contents consistent with external memory. The Intel486 processor contains a mechanism for listening to writes by other devices to external memory. When the Intel486 processor finds a write to a section of external memory contained in its internal cache, the Intel486 processor's internal copy is invalidated.

Invalidations use two pins, address hold request (AHOLD) and valid external address (EADS#). There are two steps in an invalidation cycle. First, the external system asserts the AHOLD input forcing the Intel486 processor to immediately relinquish its address bus. Next, the external system asserts EADS# indicating that a valid address is on the Intel486 processor address bus. Figure 10-22 shows the fastest possible invalidation cycle. The Intel486 processor recognizes AHOLD on one CLK edge and floats the address bus in response. To allow the address bus to float and avoid contention, EADS# and the invalidation address should not be driven until the following CLK edge. The Intel486 processor reads the address over its address lines. If the Intel486 processor finds this address in its internal cache, the cache entry is invalidated. Note that the Intel486 processor address bus is input/output, unlike the Intel386 processor's bus, which is output only.

The Intel486 processor immediately relinquishes its address bus in the next clock upon assertion of AHOLD. For example, the bus could be 3 wait states into a read cycle. If AHOLD is activated, the Intel486 processor will immediately float its address bus before ready is returned terminating the bus cycle.

When AHOLD is asserted only the address bus is floated, the data bus can remain active. Data can be returned for a previously specified bus cycle during address hold. (See Figure 10-22 and Figure 10-23.)

EADS# is normally asserted when an external master drives an address onto the bus. AHOLD need not be driven for EADS# to generate an internal invalidate. If EADS# alone is asserted while the Intel486 processor is driving the address bus, it is possible that the invalidation address will come from the Intel486 processor itself.

Note that it is also possible to run an invalidation cycle by asserting EADS# when BOFF# is asserted or after HLDA has been returned, following the assertion of HOLD.

Running an invalidate cycle prevents the Intel486 processor cache from satisfying other internal requests, so invalidations should be run only when necessary. The fastest possible invalidate cycle is shown in Figure 10-22, while a more realistic invalidation cycle is shown in Figure 10-23. Both of the examples take one clock of cache access from the Intel486 processor.

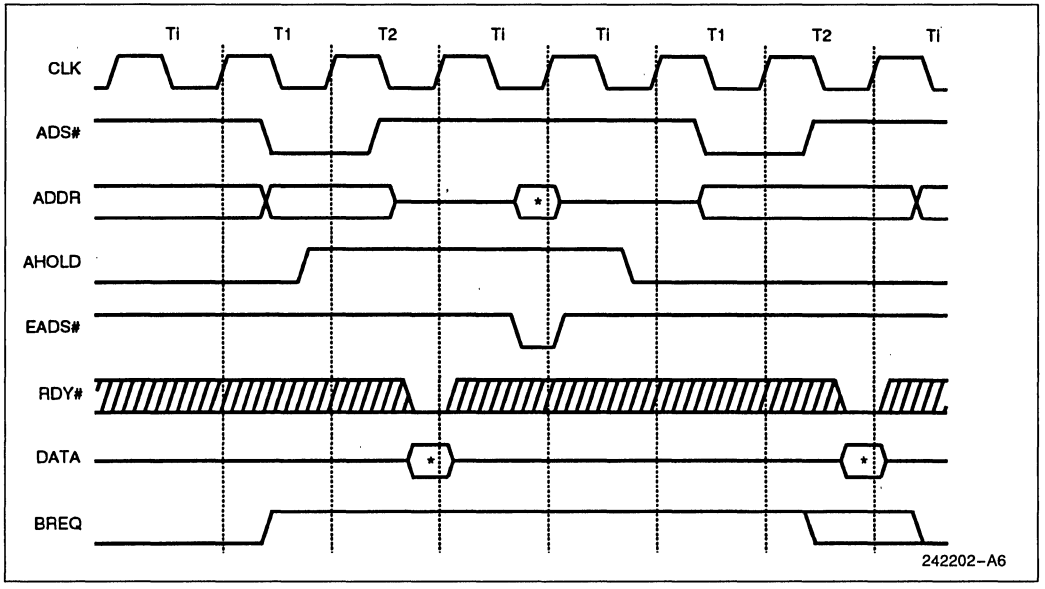


Figure 10-22. Fast Internal Cache Invalidation Cycle

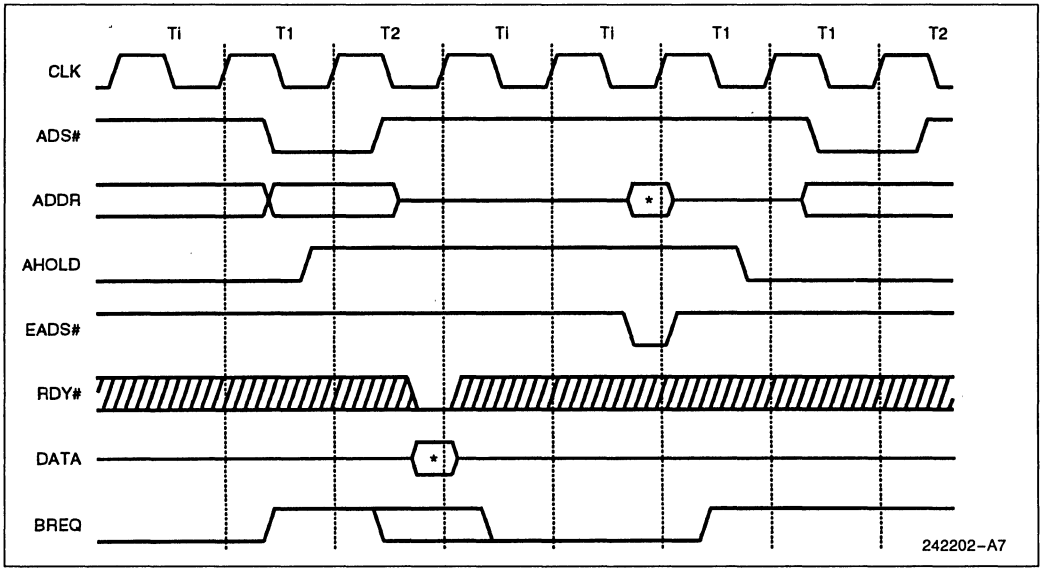


Figure 10-23. Typical Internal Cache Invalidation Cycle

10.2.8.1 Rate of Invalidate Cycles

The Intel486 processor can accept one invalidate per clock except in the last clock of a line fill. One invalidate per clock is possible as long as EADS# is negated in ONE or BOTH of the following cases:

1. In the clock RDY# or BRDY# is returned for the last time.
2. In the clock following RDY# or BRDY# being returned for the last time.

This definition allows two system designs. Simple designs can restrict invalidates to one every other clock. The simple design need not track bus activity. Alternatively, systems can request one invalidate per clock provided that the bus is monitored.

10.2.8.2 Running Invalidate Cycles Concurrently with Line Fills

Precautions are necessary to avoid caching stale data in the Intel486 processor cache in a system with a second level cache. An example of a system with a second level cache is shown in Figure 10-24.

An external device can be writing to main memory over the system bus while the Intel486 processor is retrieving data from the second level cache. The Intel486 processor will need to invalidate a line in its internal cache if the external device is writing to a main memory address also contained in the Intel486 processor cache.

A potential problem exists if the external device is writing to an address in external memory, and at the same time the Intel486 processor is reading data from the same address in the second level cache. The system must force an invalidation cycle to invalidate the data that the Intel486 processor has requested during the line fill.

If the system asserts EADS# before the first data in the line fill is returned to the Intel486 processor, the system must return data consistent with the new data in the external memory upon resumption of the line fill after the invalidation cycle. This is illustrated by the asserted EADS# signal labeled 1 in Figure 10-25.

If the system asserts EADS# at the same time or after the first data in the line fill is returned (in the same clock that the first RDY# or BRDY# is returned or any subsequent clock in the line fill) the data will be read into the Intel486 processor input buffers but it will not be stored in the on-chip cache. This is illustrated by asserted EADS# signal labeled 2 in Figure 10-25. The stale data will be used to satisfy the request that initiated the cache fill cycle.

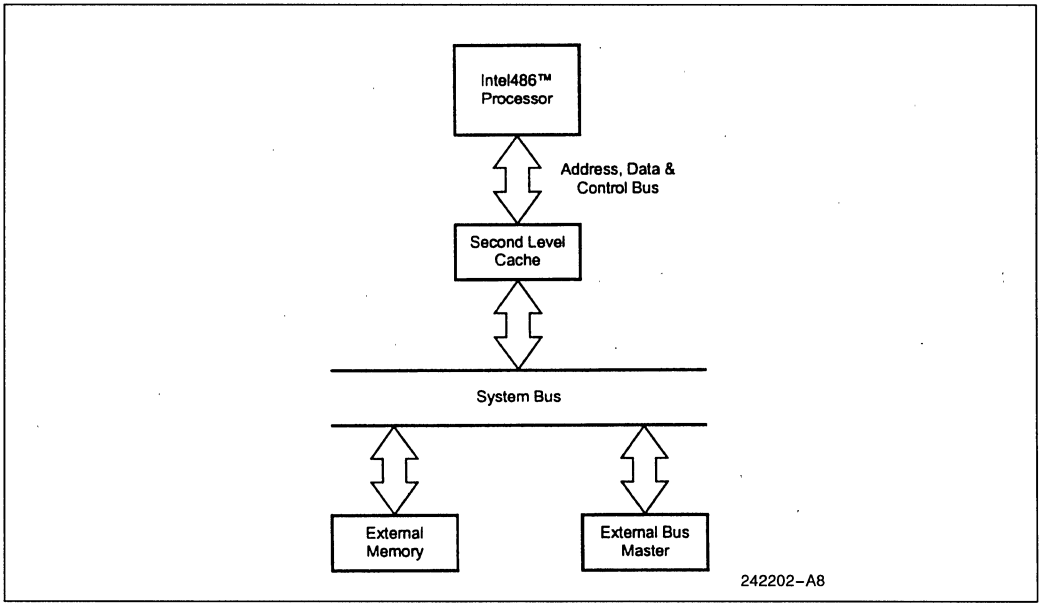
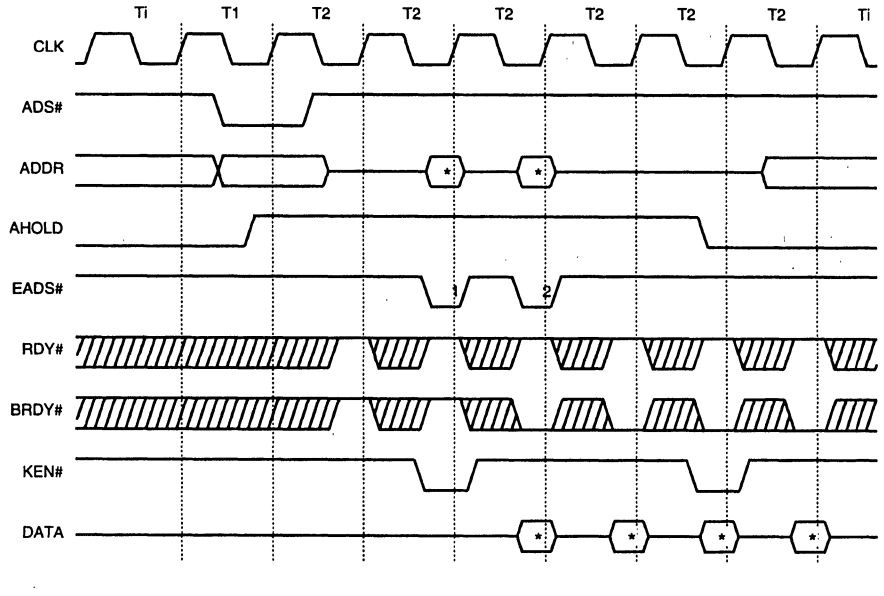


Figure 10-24. System with Second Level Cache



NOTES:
1. Data returned must be consistent if its address equals the invalidation address in this clock
2. Data returned will not be cached if its address equals the invalidation address in this clock

Figure 10-25. Cache Invalidation Cycle Concurrent with Line Fill

10.2.9 BUS HOLD

The Intel486 processor provides a bus hold, hold acknowledge protocol using the bus hold request (HOLD) and bus hold acknowledge (HLDA) pins. Asserting the HOLD input indicates that another bus master desires control of the Intel486 processor bus. The Intel486 processor will respond by floating its bus and driving HLDA active when the current bus cycle, or sequence of locked cycles is complete. An example of a HOLD/HLDA transaction is shown in Figure 10-26. Unlike the Intel386 processor, the Intel486 processor can respond to HOLD by floating its bus and asserting HLDA while RESET is asserted.

Note that HOLD will be recognized during unaligned writes (less than or equal to 32-bits) with BLAST# being active for each write. For greater than 32-bit or unaligned write, HOLD# recognition is prevented by PLOCK# getting asserted. However, HOLD is recognized during non-cacheable, non-burstable code prefetches even though PLOCK# is active.

For cacheable and non-bursted or bursted cycles, HOLD is acknowledged during backoff only if HOLD and BOFF# are asserted during an active bus cycle (after ADS# asserted) and before the first RDY# or BRDY# has been returned (see Figure 10-27). The order in which HOLD and BOFF# go active is unimportant (so long as both are active prior to the first RDY#/BRDY# returned by the system).

Figure 10-27 shows the case where HOLD is asserted first; HOLD could be asserted simultaneously or after BOFF# and still be acknowledged.

The pins floated during bus hold are: BE0#–BE3#, PCD, PWT, W/R#, D/C#, M/IO#, LOCK#, PLOCK#, ADS#, BLAST#, D0–D31, A2–A31, DP0–DP3.

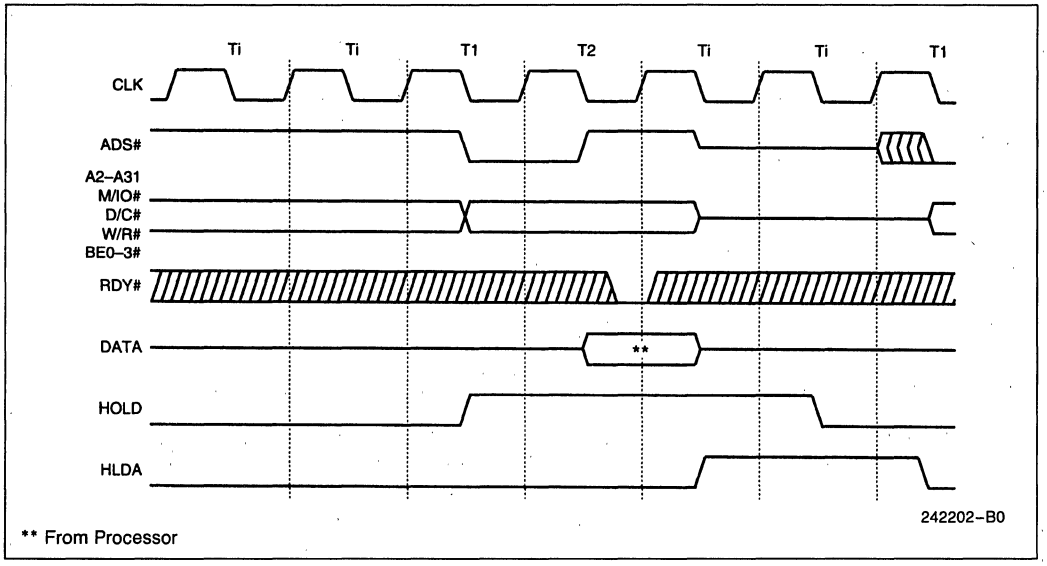


Figure 10-26. HOLD/HLDA Cycles

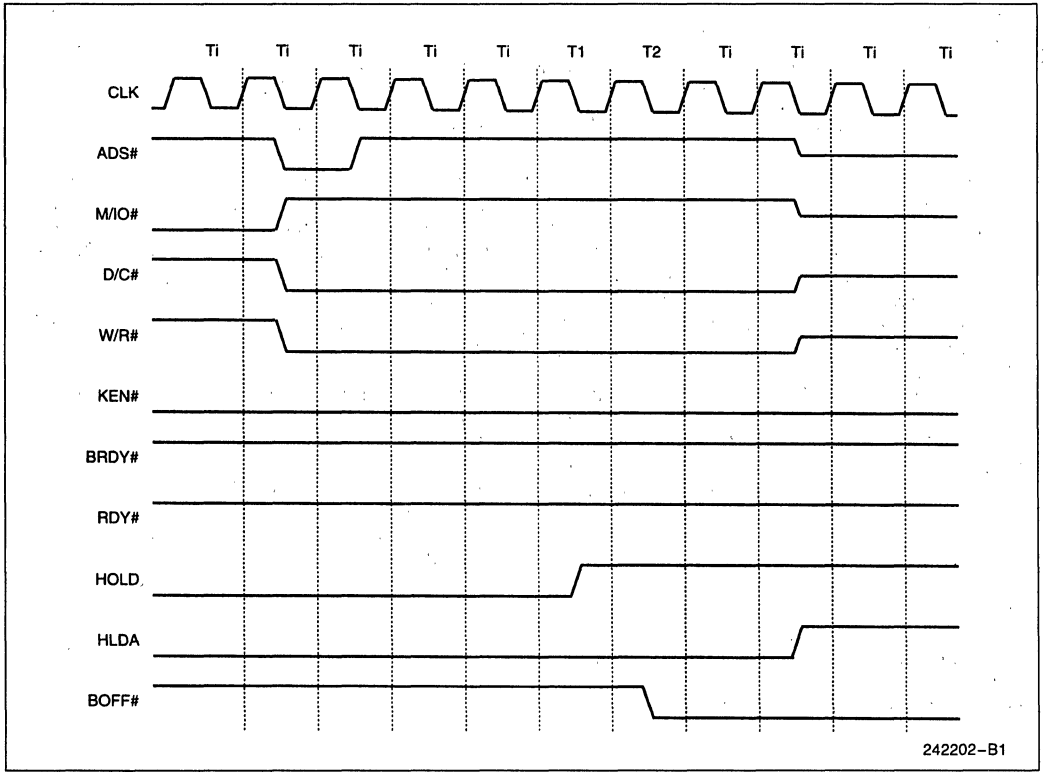


Figure 10-27. HOLD Request Acknowledged during BOFF#

10.2.10 INTERRUPT ACKNOWLEDGE

The Intel486 processor generates interrupt acknowledge cycles in response to maskable interrupt requests generated on the interrupt request input (INTR) pin. Interrupt acknowledge cycles have a unique cycle type generated on the cycle type pins.

An example of an interrupt acknowledge transaction is shown in Figure 10-28. Interrupt acknowledge cycles are generated in locked pairs. Data returned during the first cycle is ignored. The interrupt vector is returned during the second cycle on the lower 8 bits of the data bus. The Intel486 processor has 256 possible interrupt vectors.

The state of A2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A31–A3 low, A2 high, BE3#–BE1# high, and BE0# low). The address driven during the second interrupt acknowledge cycle is 0 (A31–A2 low, BE3#–BE1# high, BE0# low).

Each of the interrupt acknowledge cycles are terminated when the external system returns RDY# or BRDY#. Wait states can be added by withholding RDY# or BRDY#. The Intel486 processor automatically generates four idle clocks between the first and second cycles to allow for 8259A recovery time.

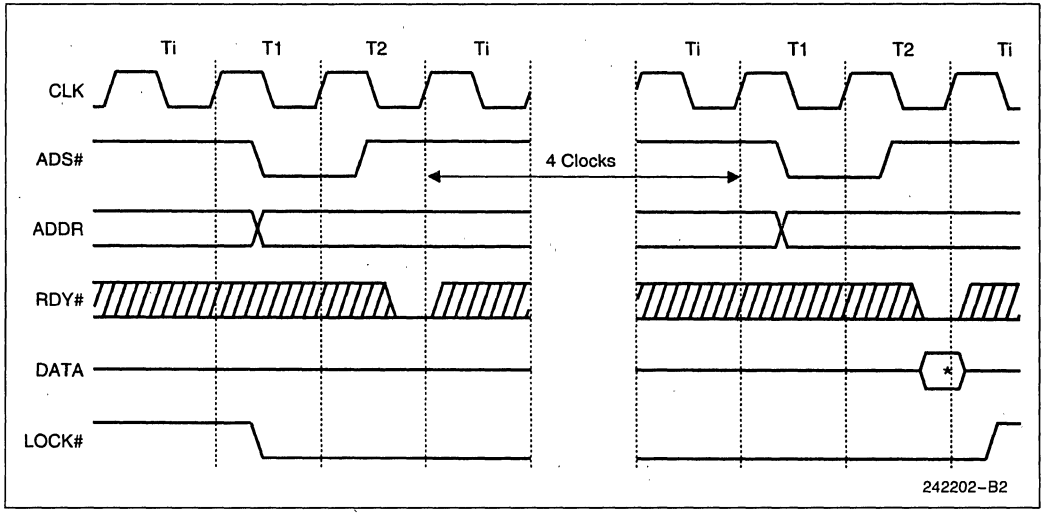


Figure 10-28. Interrupt Acknowledge Cycles

10.2.11 SPECIAL BUS CYCLES

The Intel486 processor provides special bus cycles to indicate that certain instructions have been executed, or certain conditions have occurred internally. The special bus cycles in Table 10-9 are defined when the bus cycle definition pins are in the following state: M/IO# = 0, D/C# = 0 and W/R# = 1.

During these cycles the address bus is driven low while the data bus is undefined.

Two of the special cycles indicate halt or shutdown. Another special cycle is generated when the Intel486 processor executes an INVD (invalidate data cache) instruction and could be used to flush an external cache. The Write-Back cycle is generated when the Intel486 processor executes the WBINVD (write-back invalidate data cache) instruction and could be used to synchronize an external write-back cache.

The external hardware must acknowledge these special bus cycles by returning RDY# or BRDY#.

Table 10-9. Special Bus Cycle Encoding

Cycle Name	M/IO#	D/C#	W/R#	BE3#-BE0#	A4-A2
Write-Back(1)	0	0	1	0111	000
First Flush Ack Cycle(1)	0	0	1	0111	001
Flush(1)	0	0	1	1101	000
Second Flush Ack Cycle(1)	0	0	1	1101	001
Shutdown	0	0	1	1110	000
HALT	0	0	1	1011	000
Stop Grant Ack Cycle(2)	0	0	1	1011	001

NOTES:
1. These cycles are specific to the Write-Back Enhanced Intel486 processor. (See section 7.4.1, "Snoop Cycles and Write-Back Invalidation.") The FLUSH# cycle is applicable to all Intel486 processors. See appropriate sections.
2. See section 9.6.1, "Stop Grant Bus Cycle," for details.

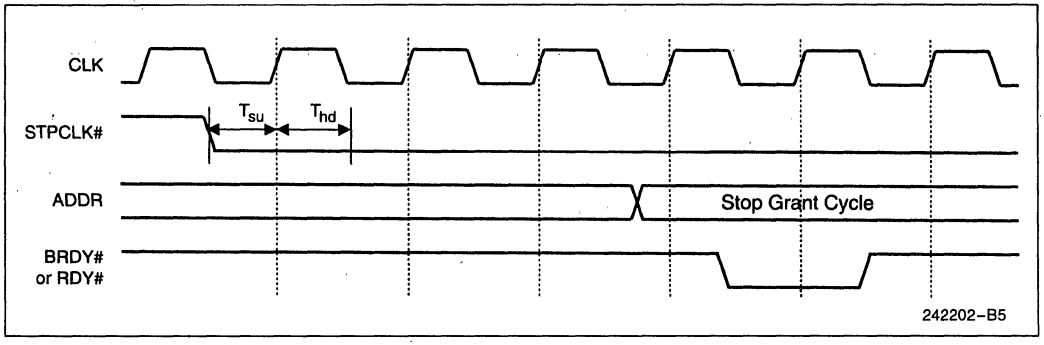


Figure 10-29. Stop Grant Bus Cycle

10.2.11.1 HALT Indication Cycle

The Intel486 processor halts as a result of executing a HALT instruction. Signaling its entrance into the HALT state, a HALT indication cycle is performed. The HALT indication cycle is identified by the bus definition signals in special bus cycle state and a byte address of 2. BE0# and BE2# are the only signals distinguishing HALT indication from shutdown indication, which drives an address of 0. During the HALT cycle, undefined data is driven on D0–D31. The HALT indication cycle must be acknowledged by RDY# asserted.

A halted Intel486 processor resumes execution when INTR (if interrupts are enabled) or NMI or RESET is asserted.

10.2.11.2 Shutdown Indication Cycle

The Intel486 processor shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the bus definition signals in special bus cycle state and a byte address of 0.

10.2.11.3 Stop Grant Indication Cycle

A special Stop Grant bus cycle will be driven to the bus after the processor recognizes the STPCLK# interrupt. The definition of this bus cycle is the same as the HALT cycle definition for the Intel486 processor, with the exception that the Stop Grant bus cycle drives the value 0000 0010H on the address pins. The system hardware must acknowledge this cycle by returning RDY# or BRDY#. The processor will not enter the Stop Grant state until either RDY# or BRDY# has been returned. (See Figure 10-31.)

The Stop Grant Bus Cycle is defined as follows:

M/IO# = 0, D/C# = 0, W/R# = 1, Address Bus = 0000 0010H (A_4 = 1), BE3#–BE0# = 1011, Data bus = undefined.

The latency between a STPCLK# request and the Stop Grant bus cycle is dependent on the current instruction, the amount of data in the processor write buffers, and the system memory performance.

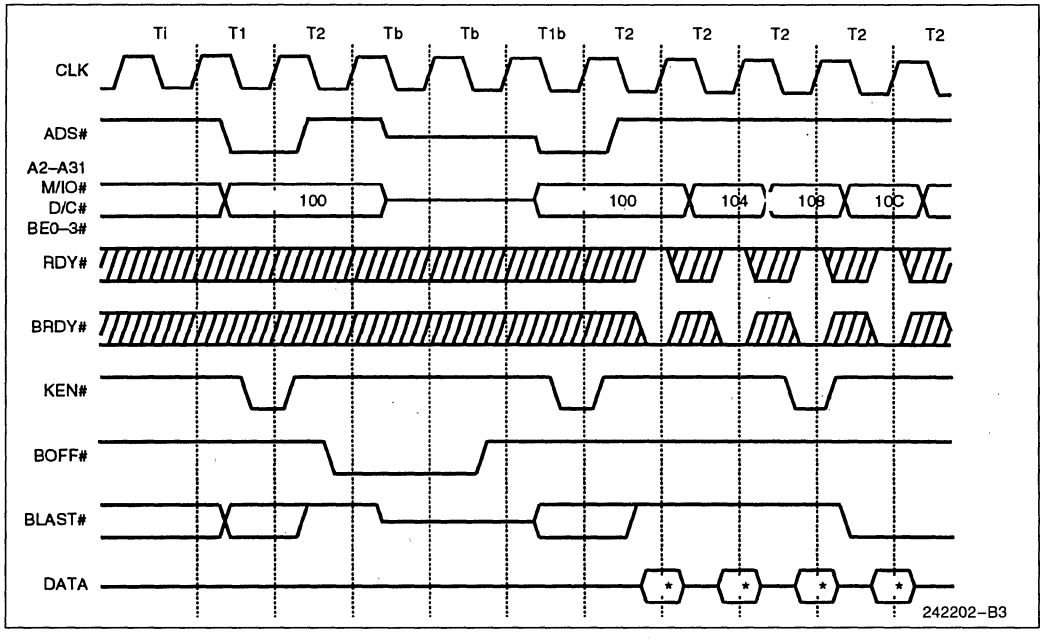


Figure 10-30. Restarted Read Cycle

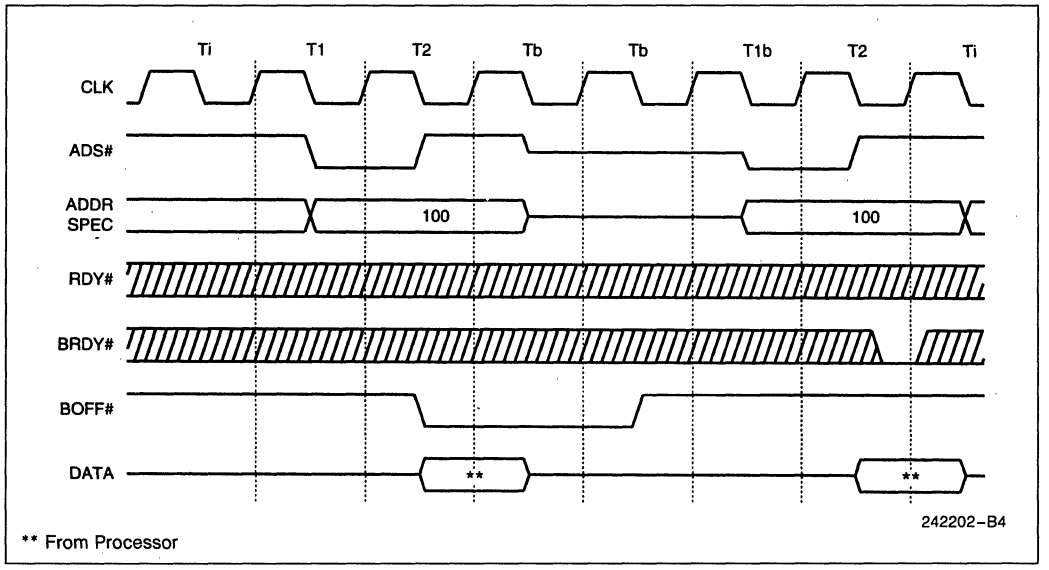


Figure 10-31. Restarted Write Cycle

10.2.12 BUS CYCLE RESTART

In a multi-master system another bus master may require the use of the bus to enable the Intel486 processor to complete its current bus request. In this situation the Intel486 processor will need to restart its bus cycle after the other bus master has completed its bus transaction.

A bus cycle may be restarted if the external system asserts the backoff (BOFF#) input. The Intel486 processor samples the BOFF# pin every clock. The Intel486 processor will immediately (in the next clock) float its address, data and status pins when BOFF# is asserted (see Figures 10-30 and 10-31). Any bus cycle in progress when BOFF# is asserted is aborted and any data returned to the processor is ignored. The same pins are floated in response to BOFF# as are floated in response to HOLD. HLDA is not generated in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#. If either RDY# or BRDY# is returned in the same clock as BOFF#, BOFF# takes effect.

The device asserting BOFF# is free to run any cycles it wants while the Intel486 processor bus is in its high impedance state. If backoff is requested after the Intel486 processor has started a cycle, the new master should wait for memory to return RDY# or BRDY# before assuming control of the bus. Waiting for ready provides a handshake to ensure that the memory system is ready to accept a new cycle. If the bus is idle when BOFF# is asserted, the new master can start its cycle two clocks after issuing BOFF#.

The external memory can view BOFF# in the same manner as BLAST#. Asserting BOFF# tells the external memory system that the current cycle is the last cycle in a transfer.

The bus remains in the high impedance state until BOFF# is negated. Upon negation, the Intel486 processor restarts its bus cycle by driving out the address and status and asserting ADS#. The bus cycle then continues as usual.

Asserting BOFF# during a burst, BS8# or BS16# cycle will force the Intel486 processor to ignore data returned for that cycle only. Data from previous cycles will still be valid. For example, if BOFF# is asserted on the third BRDY# of a burst, the Intel486 processor assumes the data returned with the first and second BRDY# is correct and restarts the burst beginning with the third item. The same rule applies to transfers broken into multiple cycle by BS8# or BS16#.

Asserting BOFF# in the same clock as ADS# will cause the Intel486 processor to float its bus in the next clock and leave ADS# floating low. Because ADS# is floating low, a peripheral may think that a new bus cycle has begun even-though the cycle was aborted. There are two possible solutions to this problem. The first is to have all devices recognize this condition and ignore ADS# until ready comes back. The second approach is to use a "two clock" backoff: in the first clock AHOLD is asserted, and in the second clock BOFF# is asserted. This guarantees that ADS# will not be floating low. This is only necessary in systems where BOFF# may be asserted in the same clock as ADS#.

10.2.13 BUS STATES

A bus state diagram is shown in Figure 10-32. A description of the signals used in the diagram is given in Table 10-10.

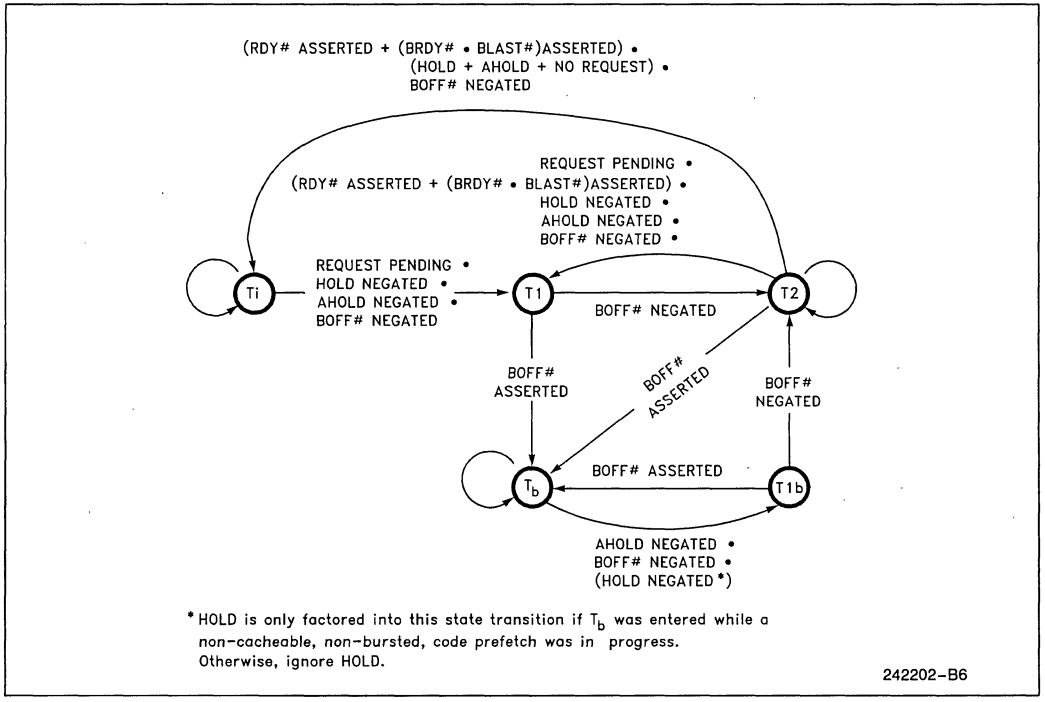


Figure 10-32. Bus State Diagram

Table 10-10. Bus State Description

State	Means
Ti	Bus is idle. Address and status signals may be driven to undefined values, or the bus may be floated to a high impedance state.
T1	First clock cycle of a bus cycle. Valid address and status are driven and ADS# is asserted.
T2	Second and subsequent clock cycles of a bus cycle. Data is driven if the cycle is a write, or data is expected if the cycle is a read. RDY# and BRDY# are sampled.
T1b	First clock cycle of a restarted bus cycle. Valid address and status are driven and ADS# is asserted.
Tb	Second and subsequent clock cycles of an aborted bus cycle.

10.2.14 FLOATING-POINT ERROR HANDLING FOR THE Intel486™ DX, IntelDX2™, AND IntelDX4™ PROCESSORS

The Intel486 DX, IntelDX2, and IntelDX4 processors provide two options for reporting Floating-Point errors. The simplest method is to raise interrupt 16 whenever an unmasked Floating-Point error occurs. This option may be enabled by setting the NE bit in control register 0 (CR0).

The Intel486 DX, IntelDX2, and IntelDX4 processors also provide the option of allowing external hardware to determine how Floating-Point errors are reported. This option is necessary for compatibility with the error reporting scheme used in DOS based systems. The NE bit must be cleared in CR0 to enable user-defined error reporting. User-defined error reporting is the default condition because the NE bit is cleared on reset.

Two pins, Floating-Point error (FERR#) and ignore numeric error (IGNNE#), are provided to direct the actions of hardware if user-defined error reporting is used. The Intel486 DX, IntelDX2, and IntelDX4 processors assert the FERR# output to indicate that a Floating-Point error has occurred. FERR# corresponds to the ERROR# pin on the Intel387™ math coprocessor. However, there is a difference in the behavior of the two.

In some cases FERR# is asserted when the next Floating-Point instruction is encountered, and in other cases it is asserted before the next Floating-Point instruction is encountered depending upon the execution state of the instruction causing the exception.

The following class of Floating-Point exceptions drive FERR# at the time the exception occurs (i.e., before encountering the next Floating-Point instruction).

1. The stack fault, invalid operation, and denormal exceptions on all transcendental instructions, integer arithmetic instructions, FSQRT, FSEALE, FPREM(1), FXTRACT, FBLD, and FBSTP.
2. Any exceptions on store instructions (including integer store instructions).

The following class of Floating-Point exceptions drive FERR# only after encountering the next Floating-Point instruction.

1. Exceptions other than on all transcendental instructions, integer arithmetic instructions, FSQRT, FSCALE, FPREM(1), FXTRACT, FBLD, and FBSTP.
2. Any exception on all basic arithmetic, load, compare, and control instructions (i.e., all other instructions).

For both sets of exceptions above, the Intel387 math coprocessor asserts ERROR# when the error occurs and does not wait for the next Floating-Point instruction to be encountered.

IGNNE# is an input to the Intel486 DX, IntelDX2, and IntelDX4 processors. When the NE bit in CR0 is cleared, and IGNNE# is asserted, the Intel486 DX, IntelDX2, and IntelDX4 processors will ignore a user Floating-Point error and continue executing Floating-Point instructions. When IGNNE# is negated, the IGNNE# is an input to these processors that will freeze on Floating-Point instructions which get errors (except for the control instructions FNCLEX, FNINIT, FNSAVE, FNSTENV, FNSTCW, FNSTSW, FNSTSW AX, FNENI, FNDISI and FNSETPM). IGNNE# may be asynchronous to the Intel486 DX, IntelDX2, and IntelDX4 processor clock.

In systems with user-defined error reporting, the FERR# pin is connected to the interrupt controller. When an unmasked Floating-Point error occurs, an interrupt is raised. If IGNNE# is high at the time of this interrupt, the Intel486 DX, IntelDX2, and IntelDX4 processors will freeze (disallowing execution of a subsequent Floating-Point instruction) until the interrupt handler is invoked. By driving the IGNNE# pin low (when clearing the interrupt request), the interrupt handler can allow execution of a Floating-Point instruction, within the interrupt handler, before the error condition is cleared (by FNCLEX, FNINIT, FNSAVE or FNSTENV). If execution of a non-control Floating-Point instruction, within the Floating-Point interrupt handler, is not needed, the IGNNE# pin can be tied HIGH.

10.2.15 Intel486™ DX, IntelDX2™, AND IntelDX4™ PROCESSORS FLOATING-POINT ERROR HANDLING IN AT-COMPATIBLE SYSTEMS

The Intel486 DX, IntelDX2, and IntelDX4 processors provide special features to allow the implementation of an AT-compatible numerics error reporting scheme. These features DO NOT replace the external circuit. Logic is still required that decodes the OUT F0 instruction and latches the FERR# signal. What follows is a description of the use of these Intel Processor features.

The features provided by the Intel486 DX, IntelDX2, and IntelDX4 processors are the NE bit in the Machine Status Register, the IGNNE# pin, and the FERR# pin.

The NE bit determines the action taken by the Intel486 DX, IntelDX2, and IntelDX4 processors when a numerics error is detected. When set this bit signals that non-DOS compatible error handling will be implemented. In this mode the Intel486 DX, IntelDX2, and IntelDX4 processors take a software exception (16) if a numerics error is detected.

If the NE bit is reset, the Intel486 DX, IntelDX2, and IntelDX4 processors use the IGNNE# pin to allow an external circuit to control the time at which non-control numerics instructions are allowed to execute. Note that Floating-Point control instructions such as FNINIT and FNSAVE can be executed during a Floating-Point error condition regardless of the state of IGNNE#.

To process a Floating-Point error in the DOS environment the following sequence must take place:

1. The error is detected by the Intel486 DX, IntelDX2, and IntelDX4 processor that activates the FERR# pin.
2. FERR# is latched so that it can be cleared by the OUT F0 instruction.
3. The latched FERR# signal activates an interrupt at the interrupt controller. This interrupt is usually handled on IRQ13.
4. The Interrupt Service Routine (ISR) handles the error and then clears the interrupt by executing an OUT instruction to port F0. The address F0 is decoded externally to clear the FERR# latch. The IGNNE# signal is also activated by the decoder output.
5. Usually the ISR then executes an FNINIT instruction or other control instruction before restarting the program. FNINIT clears the FERR# output.

Figure 10-33 illustrates a sample circuit that will perform the function described above. Note that this circuit has not been tested and is included as an example of required error handling logic.

Note that the IGNNE# input allows non-control instructions to be executed prior to the time the FERR# signal is reset by the Intel486 DX, IntelDX2, and IntelDX4 processors. This function is implemented to allow exact compatibility with the AT implementation. Most programs reinitialize the Floating-Point unit before continuing after an error is detected. The Floating-Point unit can be reinitialized using one of the following four instructions: FCLEX, FINIT, FSAVE and FSTENV.

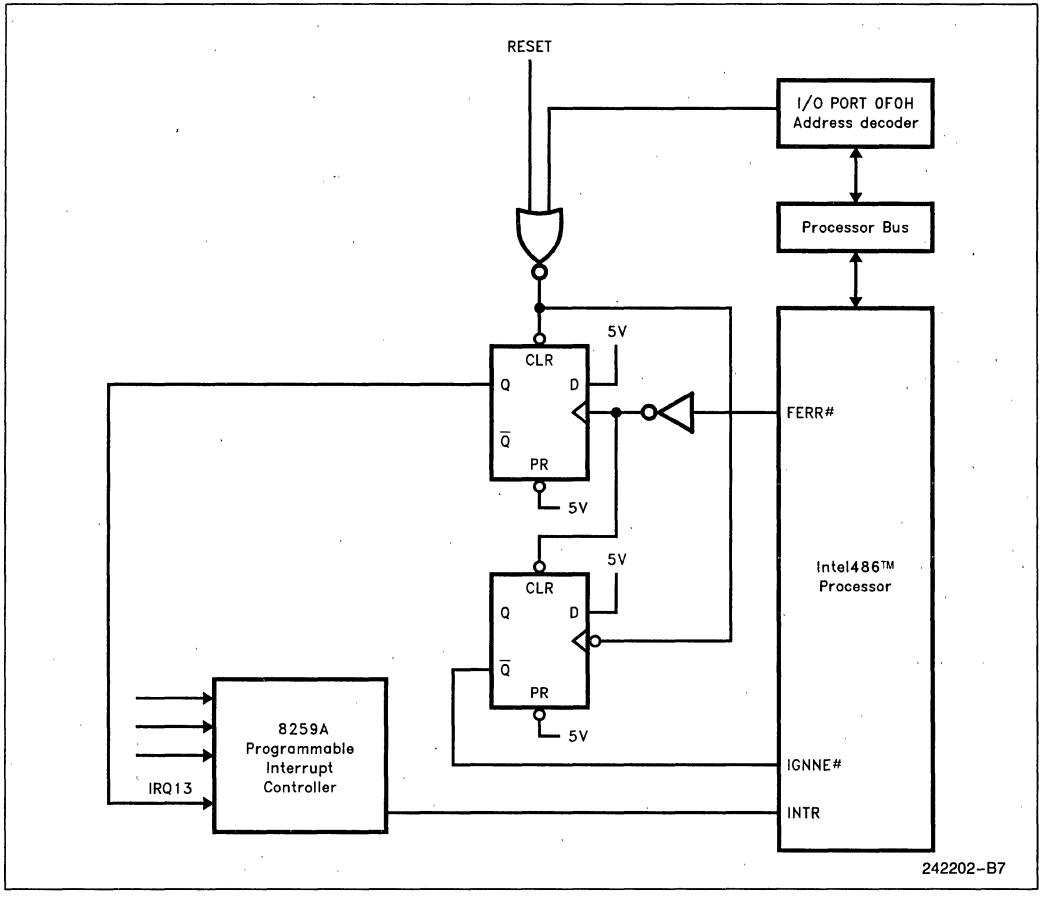


Figure 10-33. DOS-Compatible Numerics Error Circuit

10.3 Enhanced Bus Mode Operation (Write-Back Mode) for the Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processors

This section describes how the Write-Back Enhanced Intel486 processor bus operation changes for the Enhanced Bus mode when the internal cache is configured in write-back mode.

10.3.1 SUMMARY OF BUS DIFFERENCES

The following is a list of the differences between the Enhanced and Standard Bus modes:

1. Burst write capability is extended to four double-word burst cycles (for write-back cycles only).
2. Four new signals: INV, WB/WT#, HITM#, and CACHE#, have been added to support the write-back operation of the internal cache. These signals function the same as the equivalent signals on the Pentium® OverDrive® Processor pins.
3. The SRESET signal has been modified so that it neither writes back, invalidates, nor disables the cache. Special test modes are also not initiated through SRESET.
4. The FLUSH# signal behaves the same as the WBINVD instruction. Upon assertion, FLUSH# writes back all modified lines, invalidates the cache, and issues two special bus cycles.
5. The PLOCK# signal remains inactive in the Enhanced Bus mode.

10.3.2 BURST CYCLES

Figure 10-34 shows a basic burst read cycle of the Write-Back Enhanced Intel486 processors. In the Enhanced Bus mode, both PCD and CACHE# are asserted if the cycle is internally cacheable. The Write-Back Enhanced Intel486 processors sample KEN# in the clock before the first BRDY#. If KEN# is returned active by the system, this cycle is transformed into a multiple-transfer cycle. With each data item returned from external memory, the data is "cached" only if KEN# is returned active again in the clock before the last BRDY# signal. Data is sampled only in the clock in which BRDY# is returned. If the data is not sent to the processor every clock, it causes a "slow burst" cycle.

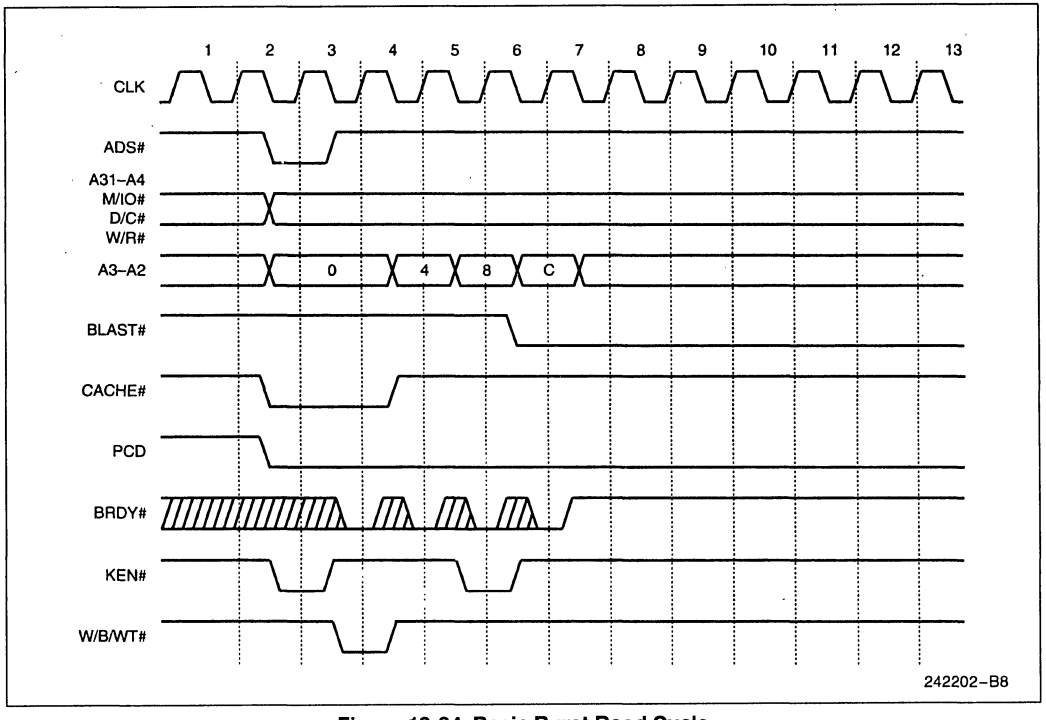


Figure 10-34. Basic Burst Read Cycle

10.3.2.1 Non-Cacheable Burst Operation

If CACHE# is asserted on a read cycle, it indicates that the processor will follow with BLAST# high if KEN# is returned active. However, the converse is not true. The Write-Back Enhanced Intel486 processors may elect to read-burst data, which are identified as non-cacheable by either CACHE# or KEN#. In this case, BLAST# is also high in the same cycle as the first BRDY# (in clock four). To improve performance, the memory controller should try to complete the cycle as a burst cycle.

The assertion of CACHE# on a write cycle signifies a replacement or snoop write-back cycle. These cycles consist of four doubleword transfers (either bursts or non-burst). The signals KEN# and WB/WT# are not sampled during write-back cycles because the processor does not attempt to redefine the cacheability of the line.

10.3.2.2 Burst Cycle Signal Protocol

The signals from ADS# through BLAST#, which are shown in Figure 10-34, have the same function and timing in both Standard and Enhanced Bus modes. Burst cycles can be up to 16-bytes long (four aligned doublewords) and can start with any one of the four doublewords. The sequence of the addresses are determined by the first address and the sequence follows the order shown previously in Table 10-8. The burst order for reads is the same as the burst order for writes. (See section 10.2.4.2, "Burst and Cache Line Fills.")

An attempted line fill, which is caused by a read miss, is indicated by the assertion of CACHE# and W/R# to low. For a line fill to occur, the system must assert KEN# twice: one clock prior to the first BRDY# and one clock prior to last BRDY#. It takes only one assertion of KEN# to mark the line as non-cacheable. A write-back cycle of a cache line, due to replacement or snoop, is indicated by the assertion of CACHE# low and W/R# high. KEN# has no effect during write-back cycles. CACHE# is valid from the assertion of ADS# through the clock in which the first RDY# or BRDY# is returned. CACHE# is inactive at all other times. PCD behaves the same in Enhanced Bus mode as in Standard Bus mode, **except that it is low during write-back cycles.**

The Write-Back Enhanced Intel486 processors samples WB/WT# once, in the *same* clock as the first BRDY#. This sampled value of WB/WT# is combined with PWT to bring the line into the internal cache, either as a write-back line or write-through line.

10.3.3 CACHE CONSISTENCY CYCLES

The system performs snooping to maintain cache consistency. Snoop cycles can be performed under AHOLD, BOFF#, or HOLD, described in Table 10-11.

The snoop cycle begins by checking whether a particular cache line has been "cached" and invalidates the line based on the state of the INV pin. If the Write-Back Enhanced Intel486 processors is configured in Enhanced Bus mode, the system must drive INV high to invalidate a particular cache line. The Write-Back Enhanced Intel486 processors do not have an output pin to indicate a snoop hit to an S-state line or an E-state line. However, the Write-Back Enhanced Intel486 processors will invalidate the line if the system snoop hits an S-state, E-state, or M-state line, provided INV was driven high during snooping. If INV is driven low during a snoop, a modified line will be written back to memory and will remain in the cache as a write-back line; a write-through line also will remain in the cache as a write-through line.

Table 10-11. Snoop Cycles under AHOLD, BOFF#, or HOLD

AHOLD	Floats the address bus. ADS# is asserted under AHOLD only to initiate a snoop write-back cycle. An ongoing burst cycle is completed under AHOLD. For non-burst cycles, a specific non-burst transfer (ADS#-RDY# transfer) is completed under AHOLD and fractured before the next assertion of ADS#. A snoop write-back cycle is reordered ahead of a fractured non-burst cycle and the non-burst cycle is completed only after the snoop write-back cycle is completed, provided there are no other snoop write-back cycles scheduled.
BOFF#	Overrides AHOLD and takes effect in the next clock. On-going bus cycles will stop in the clock following the assertion of BOFF# and resume when BOFF# is de-asserted. A snoop is the only bus cycle the Write-Back Enhanced IntelDX2™ processor responds to under BOFF#. Snoop write-back will be reordered ahead of the backed-off cycle. The snoop write-back cycle begins after BOFF# is de-asserted followed by the backed-off cycle.
HOLD	HOLD is acknowledged only between bus cycles, except for a non-cacheable, non-bursted code prefetch cycle. In a non-cacheable, non-bursted code prefetch cycle, HOLD is acknowledged after the system returns RDY#. Once HOLD is active, the processor blocks all bus activities until the system releases the bus (by de-asserting HOLD).

After asserting AHOLD or BOFF#, the external bus master driving the snoop cycle must wait for two clocks before driving the snoop address and asserting EADS#. If snooping is done under HOLD, the master performing the snoop must wait for at least one clock cycle before driving the snoop addresses and asserting EADS#. **INV should be driven low during read operations to minimize invalidations, and INV should be driven high to invalidate a cache line during write operations.** The Write-Back Enhanced Intel486 processors assert HITM# if the cycle hits a modified line in the cache. This output signal becomes valid two clock periods after EADS# is valid on the bus. HITM# remains asserted until the modified line is written back and will remain asserted until the RDY# or BRDY# of the snoop cycle is returned. Snoop operations could interrupt an ongoing bus operation in both the Standard Bus and Enhanced Bus modes. **The Write-Back Enhanced Intel486 processors can accept EADS# in every clock period while in Standard Bus mode. In Enhanced Bus mode, the Write-Back Enhanced Intel486 processors can accept EADS# every other clock period or until a snoop hits an M-state line.** The Write-Back Enhanced Intel486 processors will not accept any further snoop cycles input until the previous snoop write-back operation is completed.

All write-back cycles adhere to the burst address sequence of 0-4-8-C. The CACHE#, PWT, and PCD output pins are asserted and the KEN# and WB/WT# input pins are ignored. Write-back cycles can be either bursted or non-bursted. All write-back operations write 16 bytes of data to memory corresponding to the modified line that hit during the snoop. **Note that the Write-Back Enhanced Intel486 processors will accept BS8# and BS16# line-fill cycles, but not on replacement or snoop-forced write-back cycles.**

10.3.3.1 Snoop Collision with a Current Cache Line Operation

The system can also perform snooping concurrent with a cache access and may collide with a current cache bus cycle. Table 10-12 lists some scenarios and the results of a snoop operation colliding with an on-going cache fill or replacement cycle.

Table 10-12. Various Scenarios of a Snoop Write-Back Cycle Colliding with an On-Going Cache Fill or Replacement Cycle

Arbitration Control	Snoop to the Line That Is Being Filled	Snoop to a Different Line from the Line Being Filled	Snoop to the Line That Is Being Replaced	Snoop to a Different Line from the Line Being Replaced
AHOLD	Read all line fill data into cache line buffer. Update cache only if snoop occurred with INV = "0" No write-back cycle because the line has not been modified yet	Complete fill if the cycle is bursted. Start snoop write-back. If the cycle is non-bursted, the snoop write-back will be reordered ahead of the line fill. After the snoop write-back cycle is completed, continue with line fill	Complete replacement write-back if the cycle is bursted. Processor does not initiate a snoop write-back, but asserts HITM# until the replacement write-back is completed. If the replacement cycle is non-bursted, the snoop write-back is re-ordered ahead of the replacement write-back cycle. The processor does not continue with the replacement write-back cycle.	Complete replacement write-back if it is a burst cycle. Initiate snoop write-back. If the replacement write-back is a non-burst cycle, the snoop write-back cycle is re-ordered in front of the replacement cycle. After the snoop write-back, the replacement write-back is continued from the interrupt point.
BOFF#	Stop reading line fill data Wait for BOFF# to go inactive. Continue read from backed off point Update cache only if snoop occurred with INV = "0"	Stop fill Wait for BOFF# to go inactive Do snoop write-back Continue fill from interrupt point	Stop replacement write-back Wait for BOFF# to go inactive Initiate snoop write-back Processor does not continue replacement write-back	Stop replacement write-back Wait for BOFF# to be de-asserted Initiate snoop write-back Continue replacement write-back from point of interrupt
HOLD	HOLD is not acknowledged until the current bus cycle (i.e., the line operation) is completed, except for a non-cacheable, non-bursted code prefetch cycle. Consequently there can be no collision with the snoop cycles using HOLD, except as mentioned earlier. In this case the snoop write-back is re-ordered ahead of an on-going non-burst, non-cached code prefetch cycle. After the write-back cycle is completed, the code prefetch cycle continues from the point of interrupt.			

10.3.3.2 Snoop under AHOLD

Snooping under AHOLD begins by asserting AHOLD to force the Write-Back Enhanced Intel486 processors to float its address bus, as shown in Figure 10-35. **The ADS# for the write-back cycle is guaranteed to occur no sooner than the second clock following the assertion of HITM# (i.e., there is a dead clock between the assertion of HITM# and the first ADS# of the snoop write-back cycle.)**

When a line is written back, KEN#, WB/WT#, BS8#, and BS16# are ignored, and PWT and PCD are always low during write-back cycles.

The next ADS# for a new cycle can occur immediately after the last RDY# or BRDY# of the write-back cycle. The Write-Back Enhanced Intel486 processors do not guarantee a dead clock between cycles **unless** the **second** cycle is a snoop-forced write-back cycle. This allows snoop-forced write-backs to be backed off (BOFF#) when snooping under AHOLD.

HITM# is guaranteed to remain asserted until the RDY# or BRDY# corresponding to the last double-word of the write-back cycle is returned. HITM# will be de-asserted from the clock edge in which the last BRDY# or RDY# for the snoop write-back cycle is returned. The write-back cycle could be a bursted or non-bursted. In either case, 16 bytes of data corresponding to the modified line that has a snoop hit is written back.

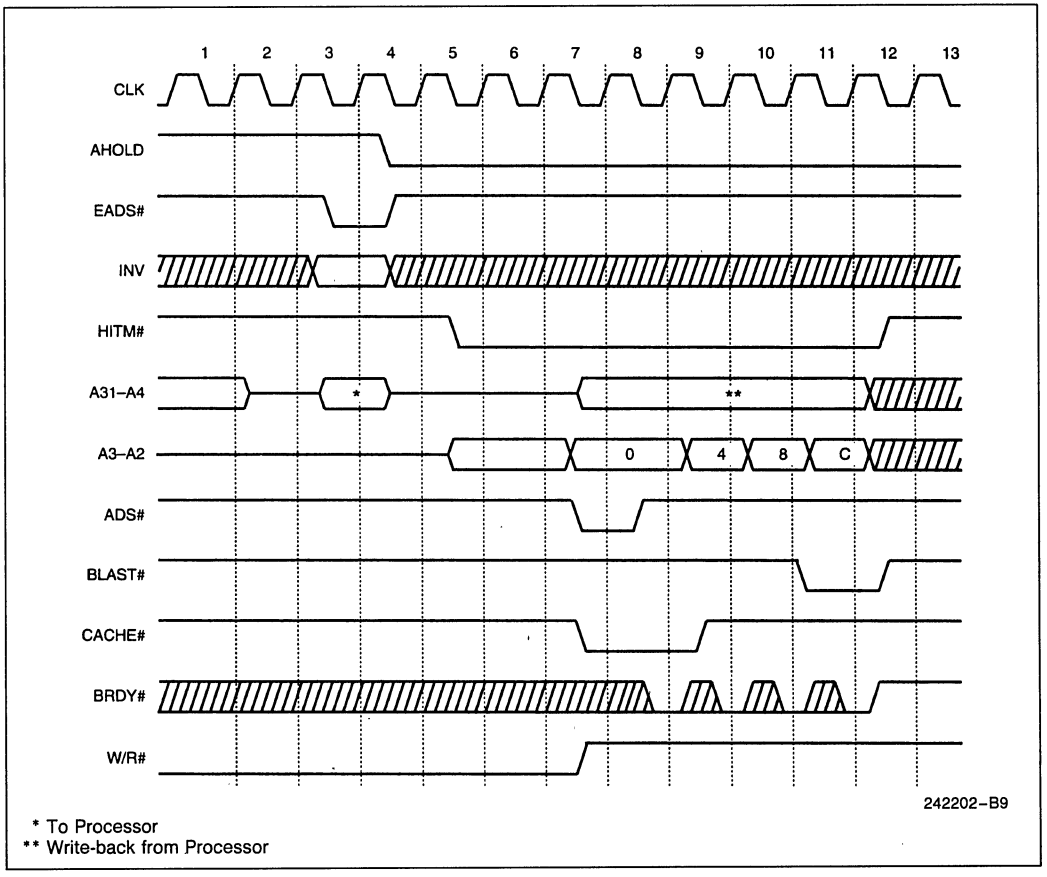


Figure 10-35. Snoop Cycle Invalidating a Modified Line

Snoop under AHOLD Overlaying a Line-Fill Cycle

The assertion of AHOLD during a line fill is allowed on the Write-Back Enhanced Intel486 processors. In this case, when a snoop cycle is overlaid by an on-going line-fill cycle, the chipset must generate the burst addresses internally for the line fill to complete, because the address bus will have the valid snoop address. The write-back mode is more complex compared to the write-through mode because of the possibility of a line being written back. Figure 10-36 shows a snoop cycle overlaying a line-fill cycle, when the snooped line is not the same as the line being filled.

In Figure 10-36, the snoop to an M-state line causes a snoop write-back cycle. The Write-Back Enhanced Intel486 processors will assert HITM# two clocks after the EADS#, but will delay the snoop write-back cycle until the line fill is completed, because the line fill shown in Figure 10-36 is a burst cycle. In this figure, AHOLD is asserted one clock after ADS#. In the clock after AHOLD is asserted, the Write-Back Enhanced Intel486 processors will float the address bus (not the Byte Enables). Hence, the memory controller must determine burst addresses in this period. The chipset must comprehend the special ordering required by all burst sequences of the Write-Back Enhanced Intel486 processors. HITM# is guaranteed to remain active until the write-back cycle completes.

If AHOLD continues to be asserted over the forced write-back cycle, the memory controller also must supply the write-back addresses to the memory. The Write-Back Enhanced Intel486 processors always run the write-back with an address sequence of 0-4-8-C.

In general, if the snoop cycle overlays any burst cycle (not necessarily a line-fill cycle) the snoop write-back will be delayed because of the on-going burst cycle. First, the burst cycle goes to completion and only then does the snoop write-back cycle start.

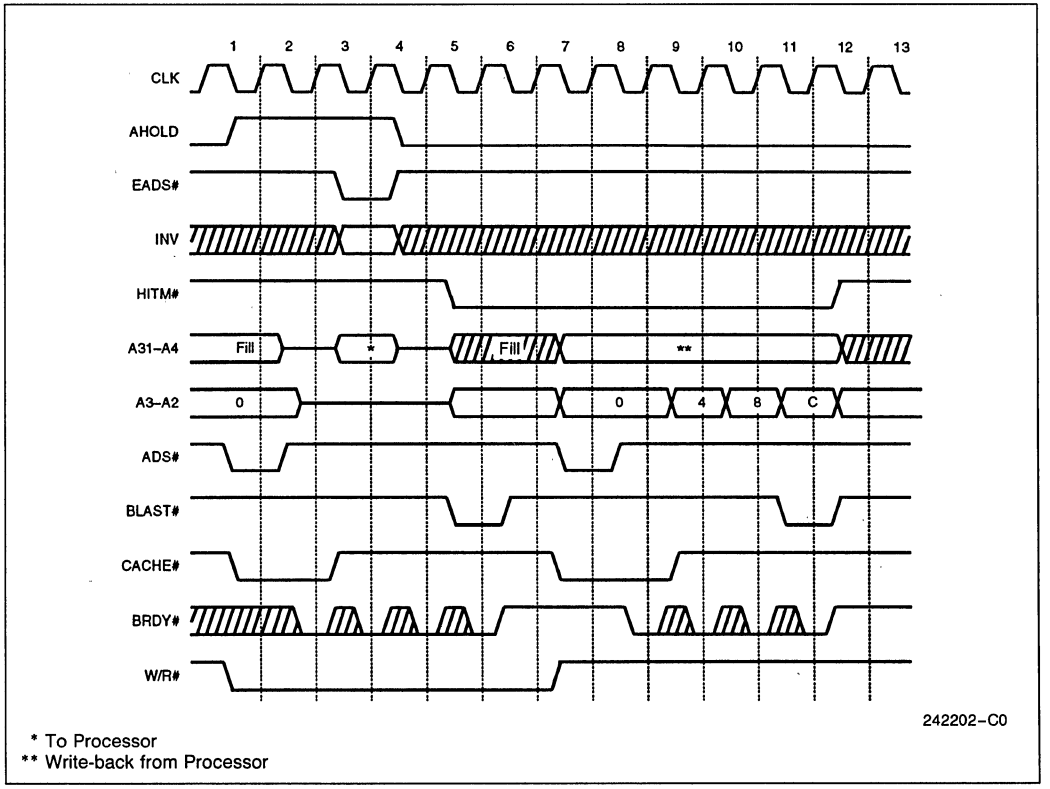


Figure 10-36. Snoop Cycle Overlaying a Line-Fill Cycle

AHOLD Snoop Overlaying a Non-Burst Cycle

When AHOLD overlays a non-burst cycle, snooping is based on the completion of the current non-bursted transfer (ADS#-RDY# transfer). Figure 10-37 shows a snoop cycle under AHOLD overlaying a non-burst line-fill cycle. HITM# is asserted two clocks after EADS#, and the non-burst cycle is fractured after the RDY# for a specific single transfer is returned. The snoop write-back cycle is re-ordered ahead of an ongoing non-burst cycle. After the write-back cycle is completed, the fractured non-burst cycle will continue. The snoop write-back ALWAYS precedes the completion of a fractured cycle, regardless of the point at which AHOLD is de-asserted, and AHOLD must be de-asserted before the fractured non-burst cycle can complete.

AHOLD Snoop to the Same Line That Is Being Filled

A system snoop will not cause a write-back cycle to occur if the snoop hits a line while the line is being filled. The processor does not allow a line to be modified until the fill is completed (and a snoop will only produce a write-back cycle for a modified line). Although a snoop to a line that is being filled will not produce a write-back cycle, the snoop still has an effect based on the following rules:

1. The processor always snoops the line being filled.
2. In **all** cases, the processor uses the operand that triggered the line fill.
3. If the snoop occurs when INV = "1", the processor never updates the cache with the fill data.
4. If the snoop occurs when INV = "0", the processor loads the line into the internal cache.

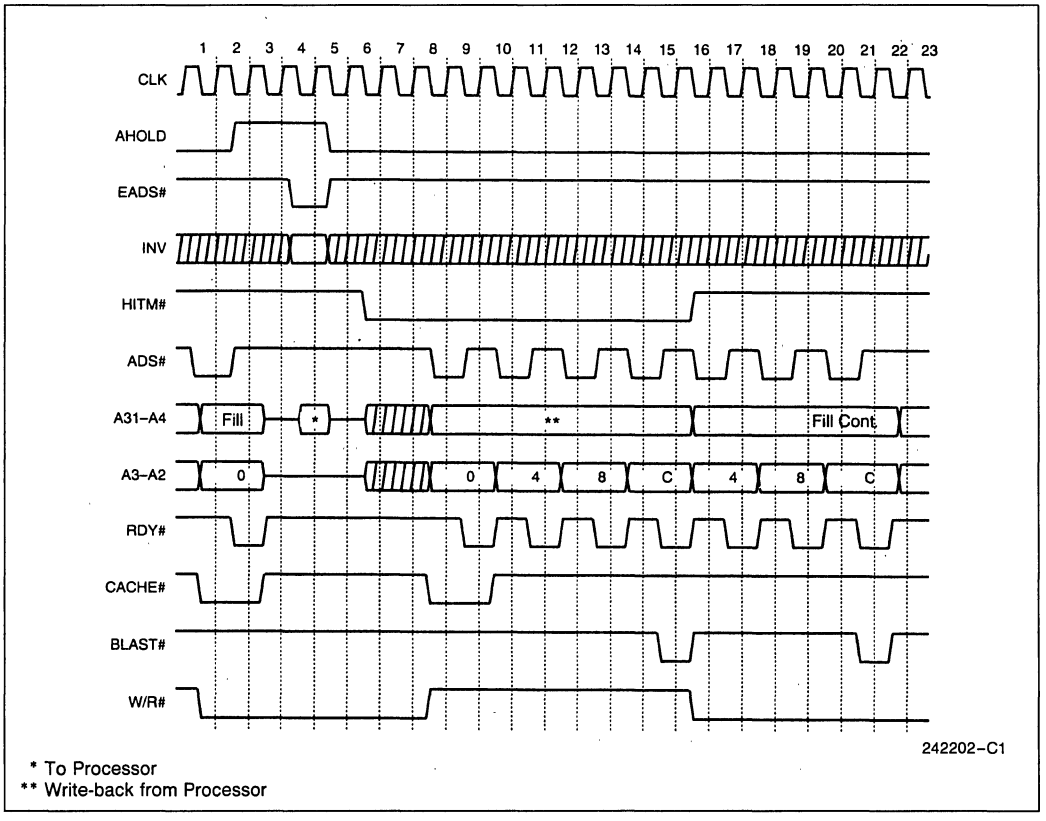


Figure 10-37. Snoop Cycle Overlaying a Non-Burst Cycle

Snoop during Replacement Write-Back

If the cache contains valid data during a line fill, one of the cache lines may be replaced as determined by the LRU algorithm. If the line being replaced is modified, this line will be written back to maintain cache coherency. When a replacement write-back cycle is in progress, it might be necessary to snoop the line that is being written back. (See Figure 10-38.)

If the replacement write-back cycle is bursted and there is a snoop hit to the same line as the line that is being replaced, the on-going replacement cycle runs to completion. HITM# is asserted until the line is written back and the snoop write-back will not be initiated. In this case, the replacement write-back is converted to the snoop write-back, and HITM# is asserted and de-asserted without a specific ADS# to initiate the write-back cycle.

If there is a snoop hit to a different line from the line being replaced, and if the replacement write-back cycle is bursted, the replacement cycle goes to completion. Only then is the snoop write-back cycle initiated.

If the replacement write-back cycle is a non-burst cycle, and if there is a snoop hit to the same line as the line being replaced, it will fracture the replacement write-back cycle after the RDY# for the current non-burst transfer is returned. The snoop write-back cycle will be reordered in front of the fractured replacement write-back cycle and will be completed under HITM#. However, after AHOLD is de-asserted the replacement write-back cycle is not completed.

If there is a snoop hit to the line that is different from the one being replaced, the non-burst replacement write-back cycle will be fractured, and the snoop write-back cycle will be reordered ahead of the replacement write-back cycle. After the snoop write-back is completed, the replacement write-back cycle will continue.

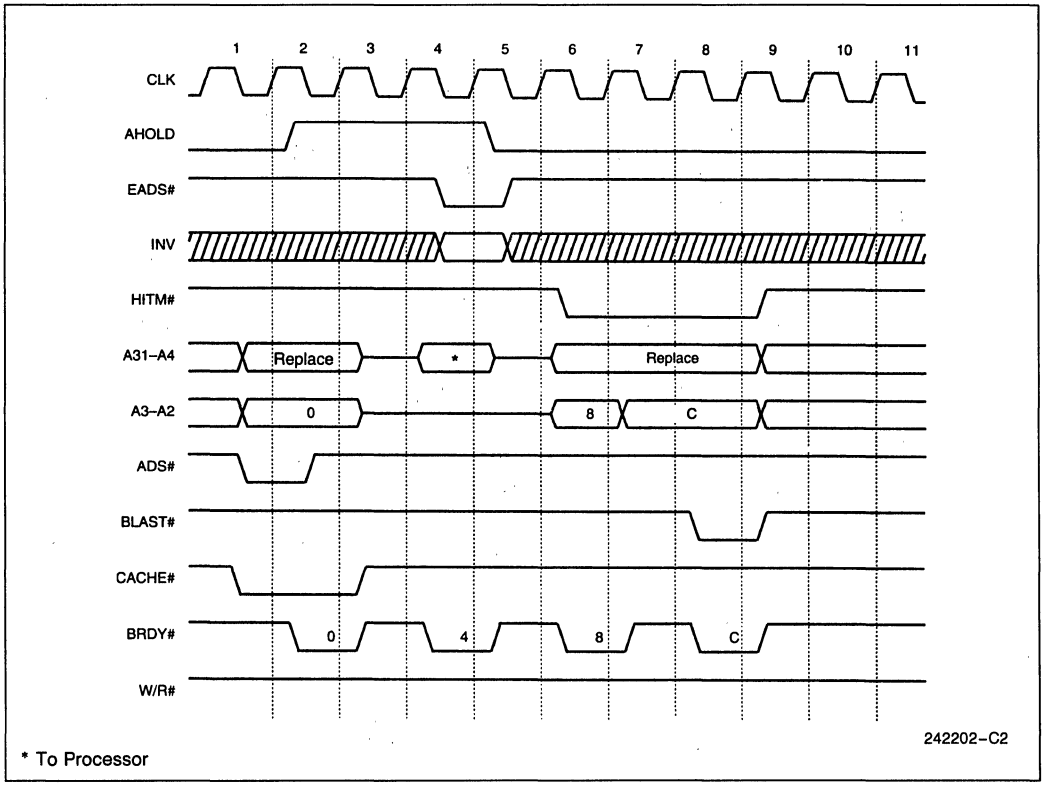


Figure 10-38. Snoop to the Line That Is Being Replaced

10.3.3.3 Snoop under BOFF#

BOFF# is capable of fracturing any transfer, burst or non-burst. The output pins (see Table 3-8 and Table 3-9) of the Write-Back Enhanced Intel486 processors will be floated in the clock period following the assertion of BOFF#. If the system snoop hits a modified line using BOFF#, the snoop write-back cycle will be reordered ahead of the current cycle. BOFF# must be de-asserted for the processor to perform a snoop write-back cycle and resume the fractured cycle. The fractured cycle resumes with a new ADS# and begins with the first uncompleted transfer. Snoops are permitted under BOFF#, but write-back cycles will not be started until BOFF# is de-asserted. Consequently, multiple snoop cycles can occur under a continuously asserted BOFF#, but only up to the first asserted HITM#.

Snoop under BOFF# during Cache Line Fill

As shown in Figure 10-39, BOFF# fractured the second transfer of a non-burst cache line-fill cycle. The system begins snooping by driving EADS# and INV in clock six. The assertion of HITM# in clock eight indicates that the snoop cycle hit a modified line and the cache line will be written back to memory. The assertion of HITM# in clock eight and CACHE# and ADS# in clock ten identifies the beginning of the snoop write-back cycle. ADS# is guaranteed to be asserted no sooner than two clock periods after the assertion of HITM#. Write-back cycles always use the four-doubleword address sequence of 0-4-8-C (burst or non-burst). The snoop write-back cycle begins upon the de-assertion of BOFF# with HITM# asserted throughout the duration of the snoop write-back cycle.

If the snoop cycle hits a line that is different from the line being filled, the cache line fill will resume after the snoop write-back cycle completes, as shown in Figure 10-39.

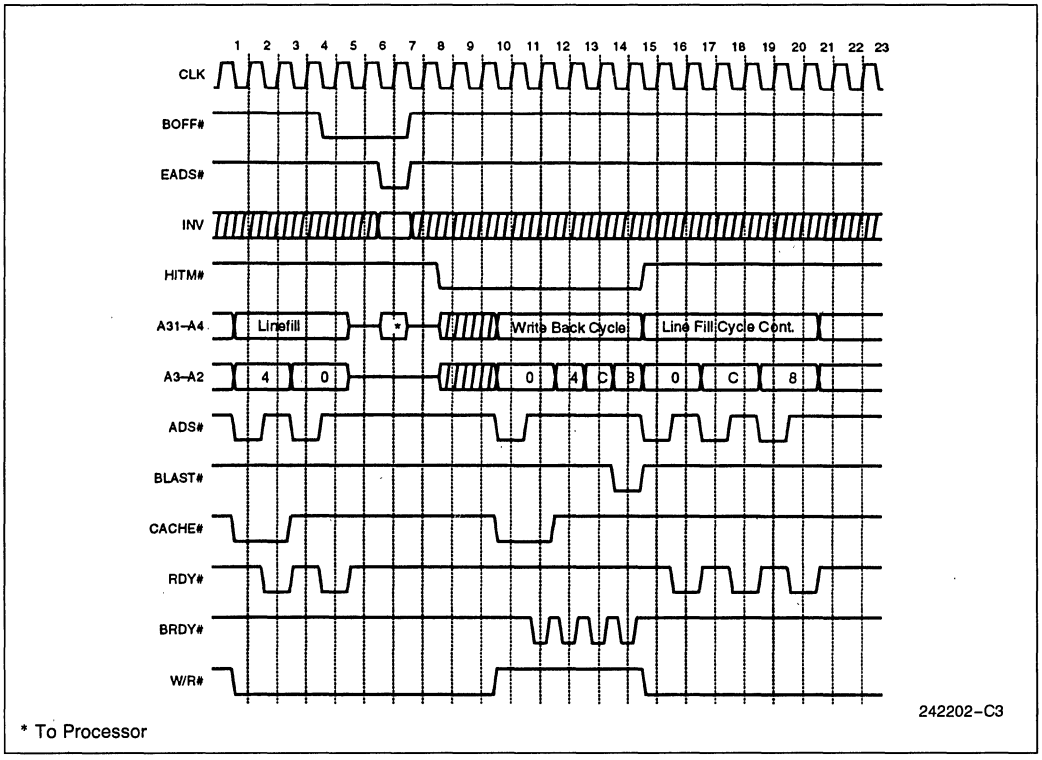


Figure 10-39. Snoop under BOFF# during a Cache Line-Fill Cycle

An ADS# is always issued when a cycle resumes after being fractured by BOFF#. The address of the fractured data transfer is reissued under this ADS#, and CACHE# is not issued unless the fractured operation resumes from the first transfer (e.g., first doubleword). If the system asserts BOFF# and RDY# simultaneously, as shown in clock four on Figure 10-39, BOFF# dominates and RDY# is ignored. Consequently, the Write-Back Enhanced Intel486 processors accept only up to the x4h doubleword, and the line fill resumes with the x0h doubleword. ADS# initiates the resumption of the line-fill operation in clock period 15. HITM# is de-asserted in the clock period following the clock period in which the last RDY# or BRDY# of the write-back cycle is returned. Hence, HITM# is guaranteed to be de-asserted before the ADS# of the next cycle.

Figure 10-39 also shows the system returning RDY# to indicate a non-burst line-fill cycle. Bursted cache line-fill cycles behave similar to non-bursted cache line-fill cycles when snooping using BOFF#. If the system snoop hits the same line as the line being filled (burst or non-burst), the Write-Back Enhanced Intel486 processors will not assert HITM# and will not issue a snoop write-back cycle, because it did not modify the line, and the line fill resumes upon the de-assertion of BOFF#. However, the line fill will be cached only if INV is driven low during the snoop cycle.

Snoop under BOFF# during Replacement Write-Back

If the system snoop under BOFF# hits the line that is currently being replaced (burst or non-burst), the entire line is written back as a snoop write-back line, and the replacement write-back cycle is not continued. However, if the system snoop hits to a different line than the one currently being replaced, the replacement write-back cycle will continue after the snoop write-back cycle has been completed. Figure 10-40 shows a system snoop hit to the same line as the one being replaced (non-burst).

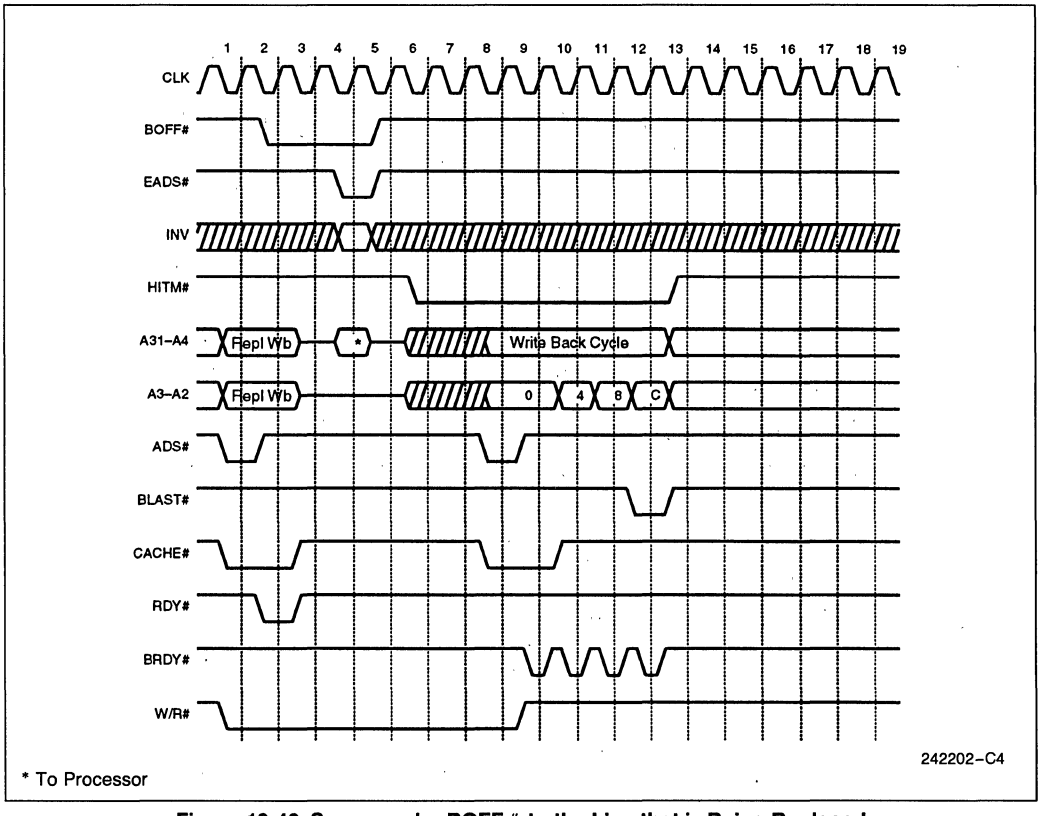


Figure 10-40. Snoop under BOFF# to the Line that is Being Replaced

10.3.3.4 Snoop under HOLD

HOLD can only fracture a non-cacheable, non-burst-ed code prefetch cycle. For all other cycles, the Write-Back Enhanced Intel486 processors will not assert HLDA until the entire current cycle is complet-ed. If the system snoop hits a modified line under HLDA during a non-cacheable, non-burstable code prefetch, the snoop write-back cycle will be reor-dered ahead of the fractured cycle. The fractured non-cacheable, non-bursted code prefetch resumes with an ADS# and begins with the first uncompleted transfer. Snoops are permitted under HLDA, but write-back cycles will not occur until HOLD is de-as-serted. Consequently, multiple snoop cycles are per-mitted under a continuously asserted HLDA only up to the first asserted HITM#.

Snoop under HOLD during Cache Line Fill

As shown in Figure 10-41, HOLD (asserted in clock two) does not fracture the bursted cache line-fill cy-cle until the line fill is completed (in clock five). Upon completing the line fill in clock five, the Write-Back Enhanced Intel486 processors assert HLDA and the system begins snooping by driving EADS# and INV in the following clock period. The assertion of HITM# in clock nine indicates that the snoop cycle has hit a modified line and the cache line is written back to memory. The assertion of HITM# in clock nine and CACHE# and ADS# in clock 11 identifies the beginning of the snoop write-back cycle. The snoop write-back cycle begins upon the de-asser-tion of HOLD, and HITM# is asserted throughout the duration of the snoop write-back cycle.

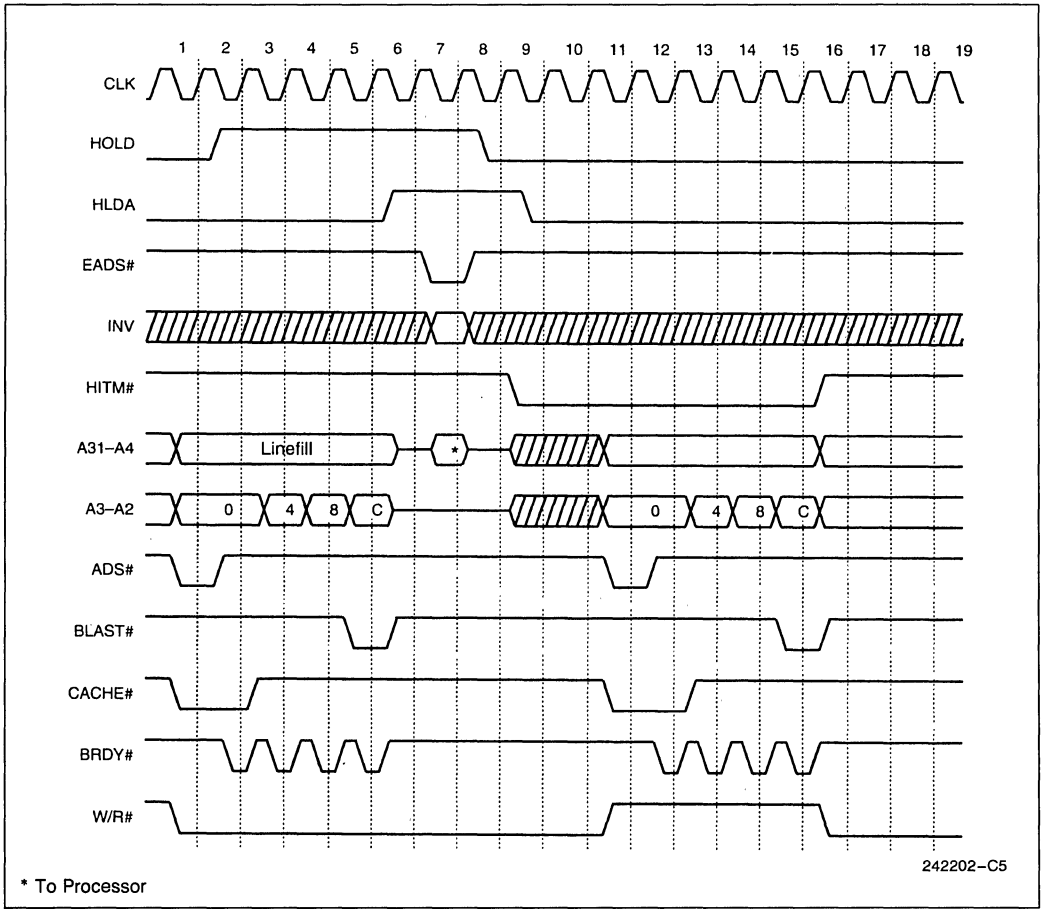


Figure 10-41. Snoop under HOLD during Line Fill

If HOLD is asserted during a non-cacheable, non-bursted code prefetch cycle, as shown in Figure 10-42, the Write-Back Enhanced Intel486 processors will issue HLDA in clock seven (which is the clock period in which the next RDY# is returned). If the system snoop hits a modified line, the snoop write-back cycle will begin after HOLD is released. After the snoop write-back cycle is completed, an ADS# is issued and the code prefetch cycle resumes.

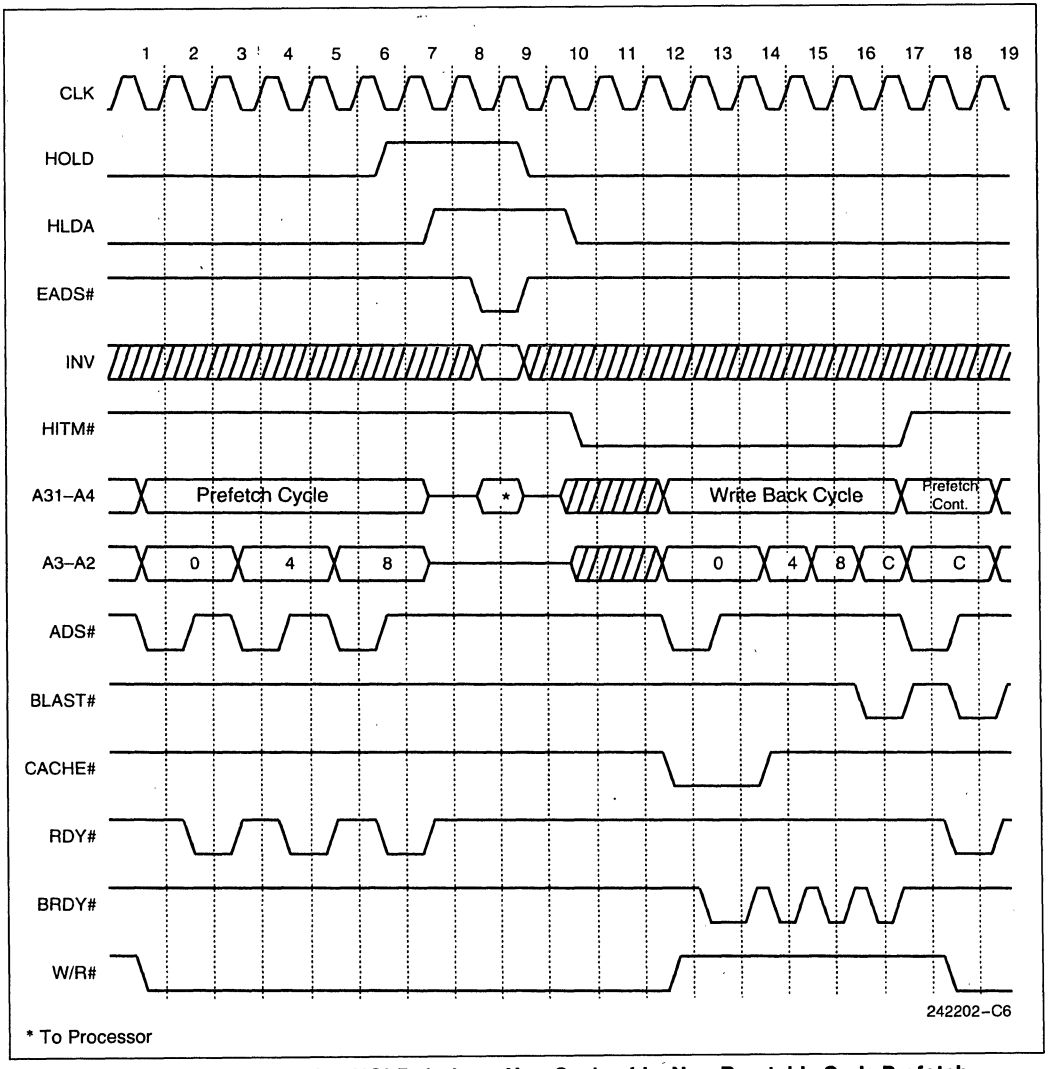


Figure 10-42. Snoop using HOLD during a Non-Cacheable, Non-Burstable Code Prefetch

Snoop under HOLD during Replacement Write-Back

Collision of snoop cycles under a HOLD during the replacement write-back cycle can never occur, because HLDA is asserted only after the replacement write-back cycle (bursted or non-bursted) is completed.

10.3.4 LOCKED CYCLES

In both Standard and Enhanced Bus modes, the Write-Back Enhanced Intel486 processors architecture supports atomic memory access. A programmer can modify the contents of a memory variable and be assured that the variable will not be accessed by another bus master between the read of the variable and the update of that variable. This function is provided for instructions that contain a LOCK prefix, and also for instructions that implicitly perform locked read modify write cycles. In hardware, the LOCK function is implemented through the LOCK# pin, which indicates to the system that the processor is performing this sequence of cycles, and that the processor should be allowed atomic access for the location accessed during the first locked cycle.

A locked operation is a combination of one or more read cycles followed by one or more write cycles with the LOCK# pin asserted. Before a locked read cycle is run, the processor first determines if the corresponding line is in the cache. If the line is present in the cache, and is in an E or S state, it is invalidated. If the line is in the M state, the processor does a write-back and then invalidates the line. A locked cycle to an M, S, or E state line is always forced out to the bus. If the operand is misaligned across cache lines, the processor could potentially run two write back cycles before starting the first locked read. In this case the sequence of bus cycles is: write back, write back, locked read, locked read, locked write and the final locked write. Note that although a total of six cycles are generated, the LOCK# pin will be active only during the last four cycles, as shown in Figure 10-43.

LOCK# will not be de-asserted if AHOLD is asserted in the middle of a locked cycle. LOCK# will remain asserted even if there is a snoop write-back during a locked cycle. LOCK# will be floated if BOFF# is asserted in the middle of a locked cycle. However, it will be driven LOW again when the cycle restarts after BOFF#. Locked read cycles are never transformed into line fills, even if KEN# is returned active. **If there are back to back locked cycles, the Write-Back Enhanced Intel486 processors do not insert a dead clock between these two cycles.** HOLD is recognized if there are two back to back locked cycles, and LOCK# will float when HLDA is asserted.

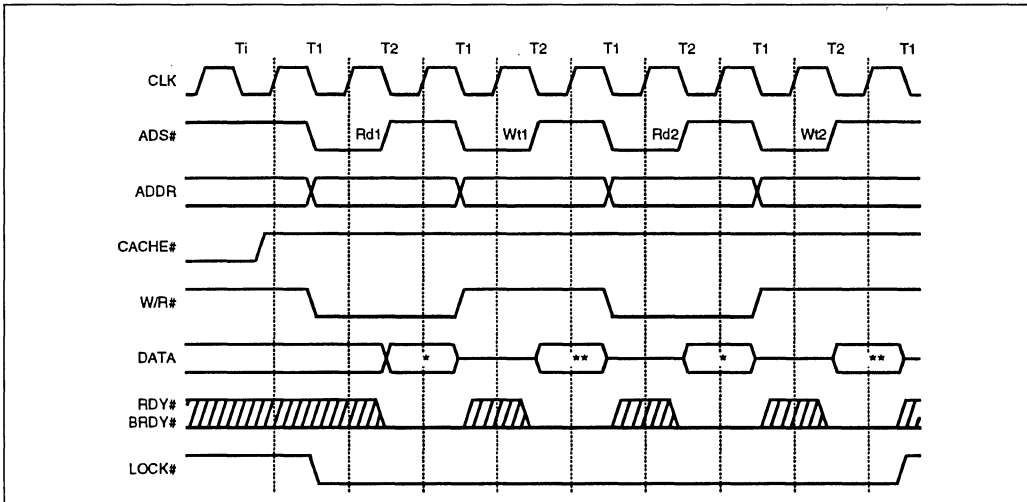


Figure 10-43. Locked Cycles (Back to Back)

10.3.4.1 Snoop/Lock Collision

If there is a snoop cycle overlaying a locked cycle, the snoop write-back cycle will fracture the locked cycle. As shown in Figure 10-44, after the read portion of the locked cycle is completed, the snoop write-back starts under HITM#. After the write-back is completed the locked cycle will continue. But during all this time (including the write-back cycle), the LOCK# signal remains asserted.

Because HOLD is not acknowledged if LOCK# is asserted, snoop-lock collisions are restricted to AHOLD and BOFF# snooping.

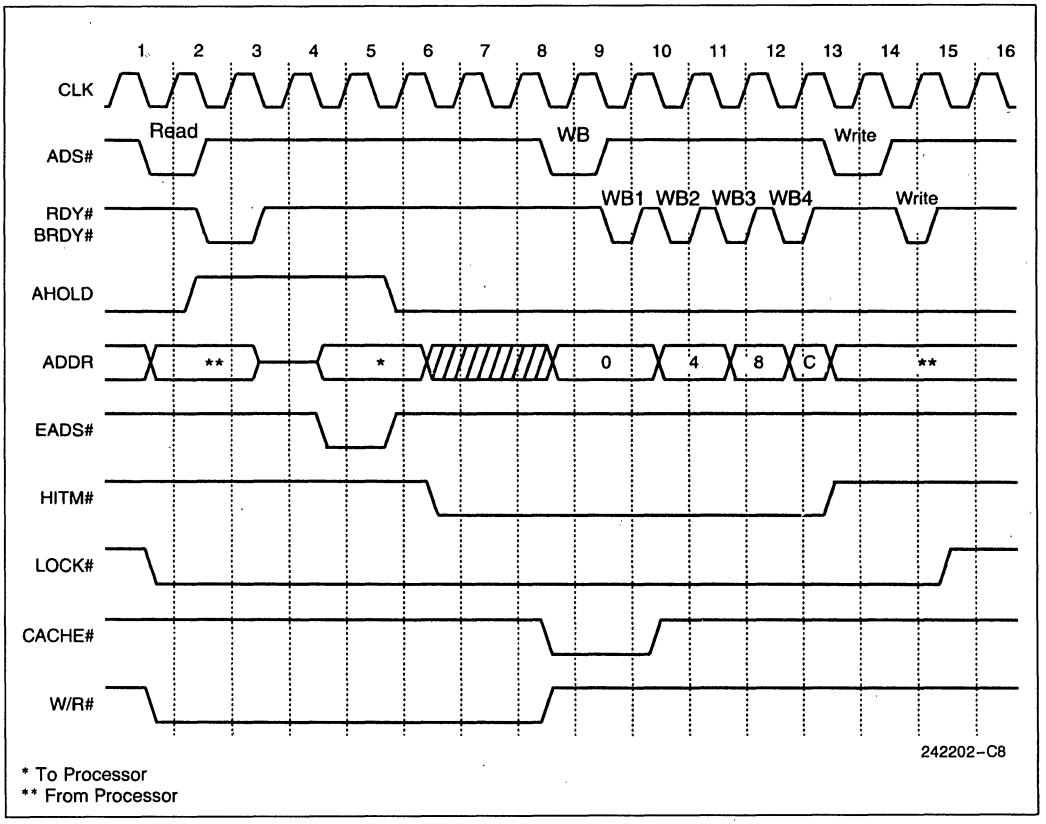


Figure 10-44. Snoop Cycle Overlaying a Locked Cycle

10.3.5 FLUSH OPERATION

The Write-Back Enhanced Intel486 processors execute a flush operation when the FLUSH# pin is active, and no outstanding bus cycles, such as a line fill or write back, are being processed. In the Enhanced Bus mode, the processor first writes back all the modified lines to external memory. After the write-back is completed, two special cycles are generated, indicating to the external system that the write-back is done. All lines in the internal cache are invalidated after all the write-back cycles are done. Depending on the number of modified lines in the cache, the flush could take a minimum of 1280 bus clocks (2560 processor clocks) and up to a maximum of 5000+ bus clocks to scan the cache, perform the write-backs, invalidate the cache, and run the flush acknowledge cycles. FLUSH# is implemented as an interrupt in the Enhanced Bus mode, and will be recognized only on an instruction boundary. Write-back system designs should look for the flush acknowledge cycles to recognize the end of the flush operation. Figure 10-45 shows the flush operation of the Write-Back Enhanced Intel486 processors, when configured in the Enhanced Bus mode.

If the processor is in Standard Bus mode, the processor will not issue special acknowledge cycles in response to the FLUSH# input, although the internal cache is invalidated. The invalidation of the cache, in this case, takes only two bus clocks.

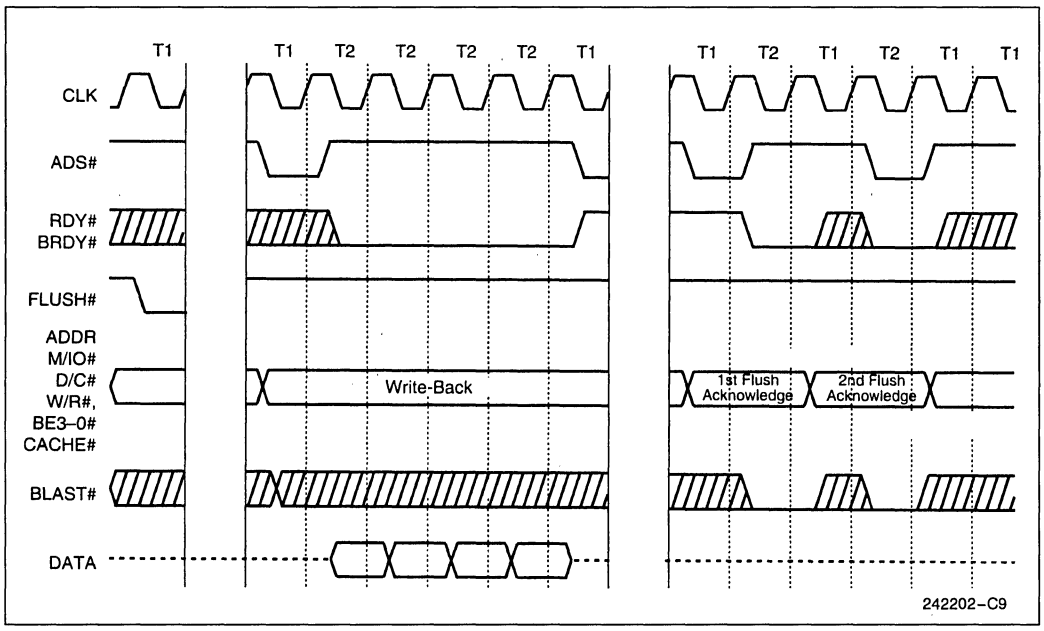


Figure 10-45. Flush Cycle

10.3.6 PSEUDO LOCKED CYCLES

In Enhanced Bus mode, PLOCK# is always driven inactive for both burst and non-burst cycles. Hence, it is possible for other bus masters to gain control of the bus during operand transfers that take more than one bus cycle. A 64-bit aligned operand can be read in one burst cycle or two non-burst cycles if BS8# and BS16# are not asserted. Figure 10-46 shows a 64-bit floating point operand or Segment Descriptor read cycle, which is burst by the system returning BRDY#.

10.3.6.1 Snoop under AHOLD during Pseudo-Locked Cycles

AHOLD can fracture a 64-bit transfer if it is a non-burst cycle. If the 64-bit cycle is burst, as shown in Figure 10-46, the entire transfer goes to completion and only then does the snoop write-back cycle start.

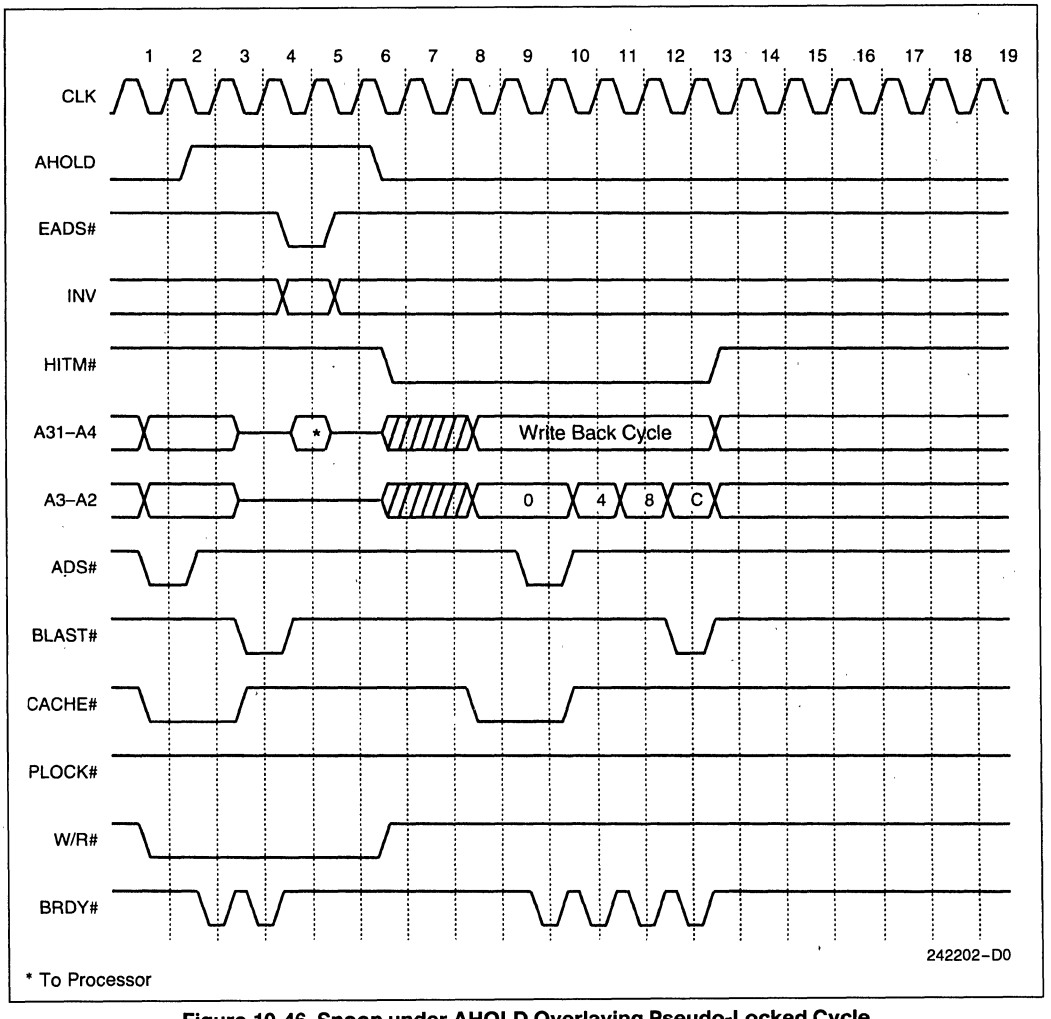


Figure 10-46. Snoop under AHOLD Overlaying Pseudo-Locked Cycle

10.3.6.2 Snoop under Hold during Pseudo-Locked Cycles

As shown in Figure 10-47, HOLD will not fracture the 64-bit burst transfer. The Write-Back Enhanced Intel486 processors will not issue HLDA until clock four. After the 64-bit transfer is completed, the Write-Back Enhanced Intel486 processors writes back the modified line to memory (if snoop hits to modified line). If the 64-bit transfer is non-bursted, the Write-Back Enhanced Intel486 processors can issue HLDA in between bus cycles for a 64-bit transfer.

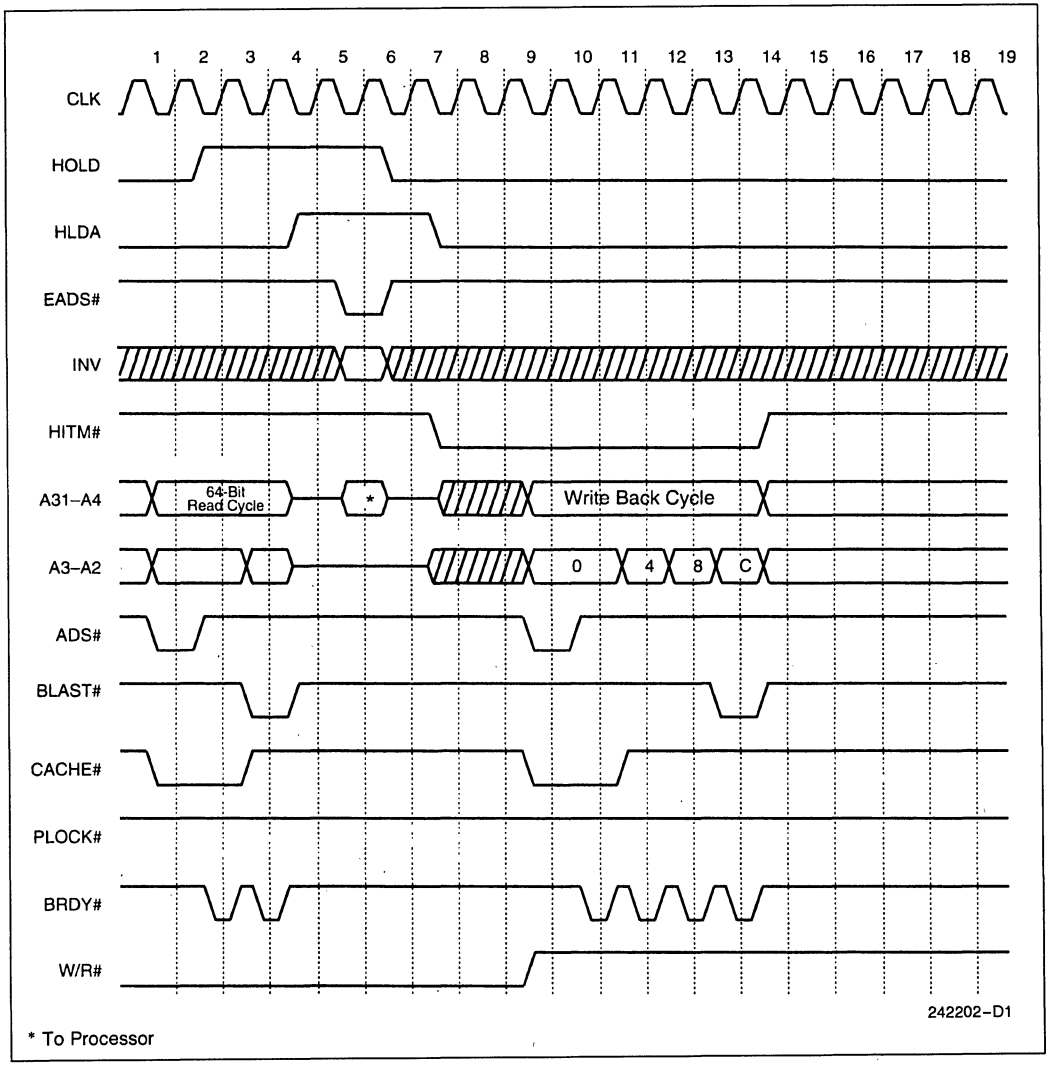


Figure 10-47. Snoop under HOLD Overlaying Pseudo-Locked Cycle

10.3.6.3 Snoop under BOFF# Overlaying a Pseudo-Locked Cycle

BOFF# is capable of fracturing any bus operation. As shown in Figure 10-48, BOFF# fractured a current 64-bit read cycle in clock four. If there is a snoop hit under BOFF#, the snoop write-back operation will begin after BOFF# is de-asserted. The 64-bit write cycle resumes after the snoop write-back operation completes.

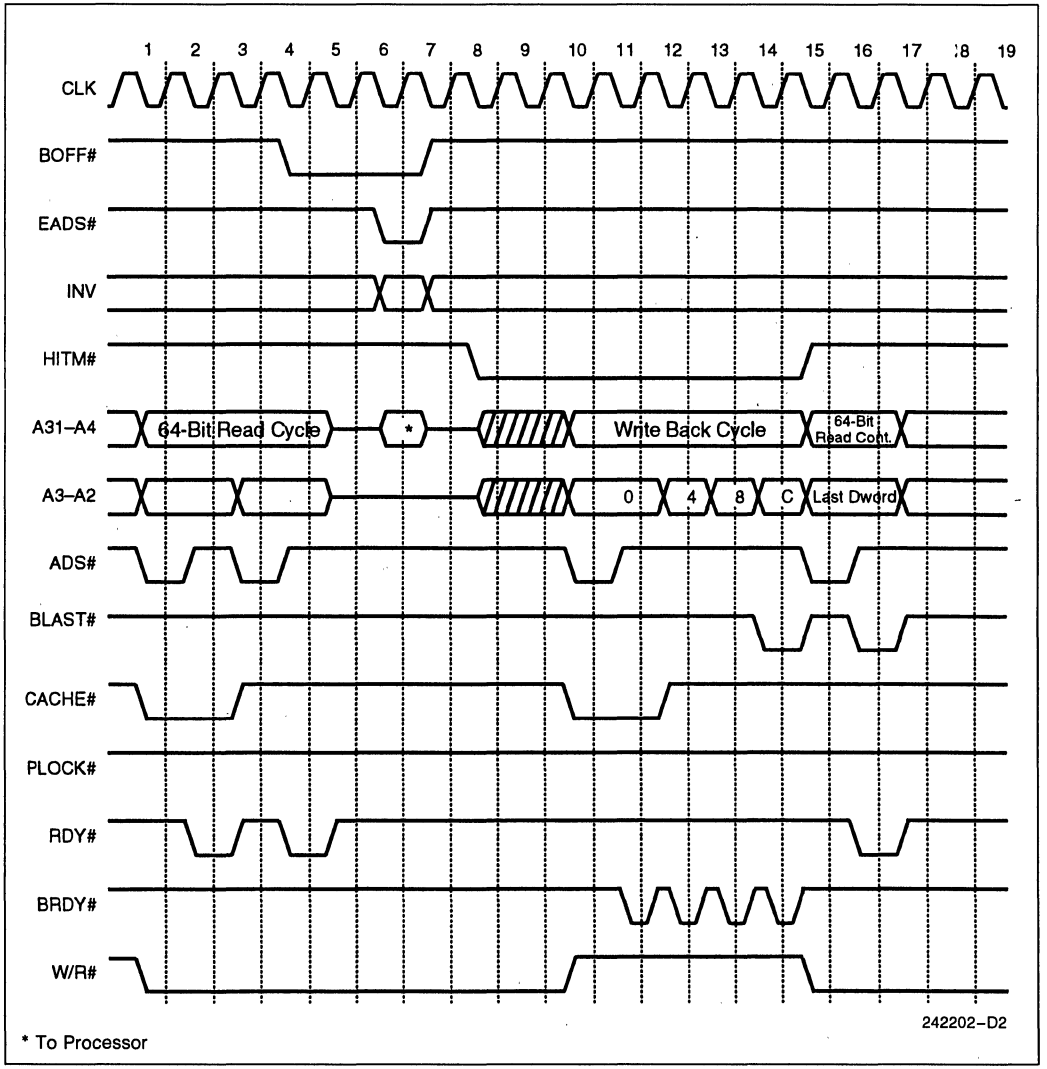


Figure 10-48. Snoop under BOFF# Overlaying a Pseudo-Locked Cycle

11.0 TESTABILITY

Testing in the Intel486 processor can be divided into two categories: Built-in Self Test (BIST) and external testing. The BIST tests the non-random logic, control ROM (CROM), translation lookaside buffer (TLB) and on-chip cache memory. External tests can be run on the TLB and the on-chip cache. The Intel486 processor also has a test mode in which all outputs are tri-stated.

11.1 Built-In Self Test (BIST)

The BIST is initiated by holding the AHOLD (address hold) HIGH for 1 CLK after RESET goes from HIGH to LOW, as shown in Figure 9.6. No bus cycles will be run by the Intel486 processor until the BIST is concluded. Note that for the Intel486 processor, the RESET must be active for 15 clocks with or without BIST enabled for warm resets. SRESET should not be driven active (i.e., high) when entering or during BIST. See Table 11-1 for approximate clocks and maximum completion times for different Intel486 processors.

The results of BIST is stored in the EAX register. The Intel486 processor has successfully passed the BIST if the contents of the EAX register are zero. If the results in EAX are not zero, then the BIST has detected a flaw in the Intel486 processor. The Intel486 processor performs reset and begins normal operation at the completion of the BIST.

The non-random logic, control ROM, on-chip cache and translation lookaside buffer (TLB) are tested during the BIST.

The cache portion of the BIST verifies that the cache is functional and that it is possible to read and write to the cache. The BIST manipulates test registers TR3, TR4 and TR5 while testing the cache. These test registers are described in section 11.2, "On-Chip Cache Testing."

The cache testing algorithm writes a value to each cache entry, reads the value back, and checks that the correct value was read back. The algorithm may be repeated more than once for each of the 512 cache entries using different constants. The IntelDX4 processor has 1024 cache entries. All other Intel486 processors have 512 cache entries.

The TLB portion of the BIST verifies that the TLB is functional and that it is possible to read and write to the TLB. The BIST manipulates test registers TR6 and TR7 while testing the TLB. TR6 and TR7 are described in section 11.3.2, "TLB Test Registers TR6 and TR7."

11.2 On-Chip Cache Testing

The on-chip cache testability hooks are designed to be accessible during the BIST and for assembly language testing of the cache.

The Intel486 processor contains a cache fill buffer and a cache read buffer. For testability writes, data must be written to the cache fill buffer before it can be written to a location in the cache. Data must be read from a cache location into the cache read buffer before the processor can access the data. The cache fill and cache read buffer are both 128 bits wide.

11.2.1 CACHE TESTING REGISTERS TR3, TR4 AND TR5

Figure 11-1 shows the three cache testing registers: the Cache Data Test Register (TR3), the Cache Status Test Register (TR4) and the Cache Control Test Register (TR5). External access to these registers is provided through MOV reg, TREG and MOV TREG, reg instructions.

Table 11-1. Maximum BIST Completion Time

Processor Type	Core Clock Freq.	Approximate Clocks	Approximate Time for Completions
Intel486™ SX	25 MHz	1.05 million	42 milliseconds
IntelSX2™	50 MHz	0.6 million	24 milliseconds
Intel486 DX	33 MHz	1.05 million	32 milliseconds
IntelDX2™	50 MHz	0.6 million	24 milliseconds
IntelDX4™	75 MHz	1.6 million	22 milliseconds

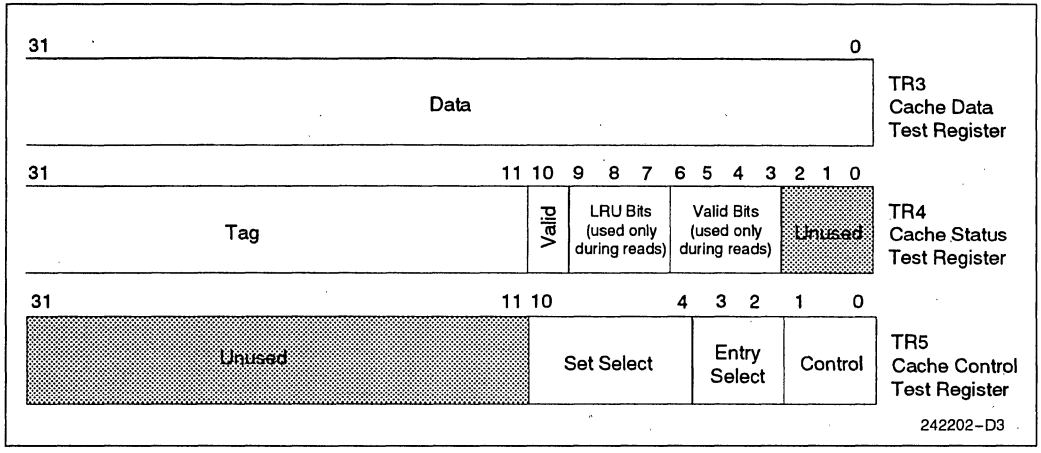


Figure 11-1. Cache Test Registers (All Intel486™ Processors Except the IntelDX4™ Processor)

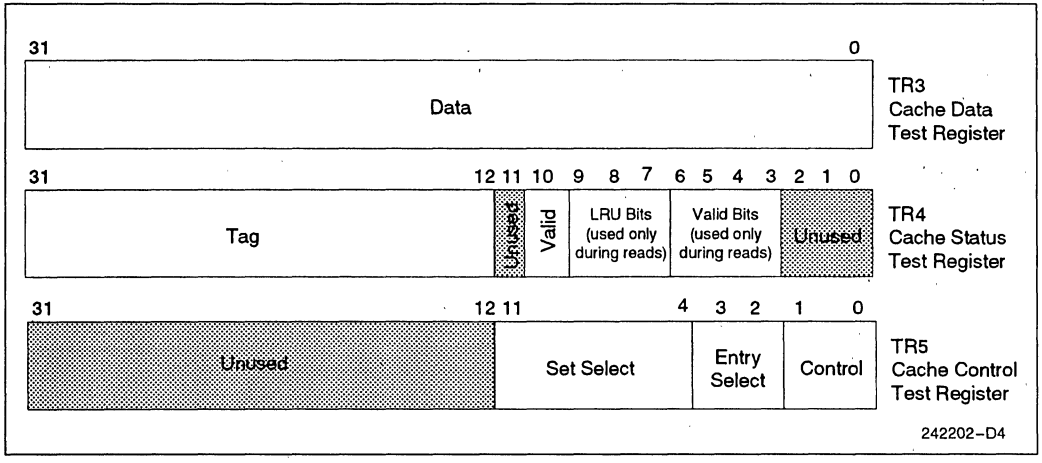


Figure 11-2. IntelDX4™ Processor Cache Test Registers

Cache Data Test Register: TR3

The cache fill buffer and the cache read buffer can only be accessed through TR3. Data to be written to the cache fill buffer must first be written to TR3. Data read from the cache read buffer must be loaded into TR3.

TR3 is 32 bits wide while the cache fill and read buffers are 128 bits wide. 32 bits of data must be written to TR3 four times to fill the cache fill buffer. 32 bits of data must be read from TR3 four times to empty the cache read buffer. The entry select bits in TR5 determine which 32 bits of data TR3 will access in the buffers.

Cache Status Test Register: TR4

TR4 handles tag, LRU and valid bit information during cache tests. TR4 must be loaded with a tag and a valid bit before a write to the cache. After a read from a cache entry, TR4 contains the tag and valid bit from that entry, and the LRU bits and four valid bits from the accessed set. Note that the IntelDX4 processor has one less bit in the TR4 TAG field. (See Figure 11-1.)

Cache Control Test Register: TR5

TR5 specifies which testability operation will be performed and the set and entry within the set which will be accessed. The set select field determines which will be accessed. Note that the IntelDX4 processor has an 8-bit set select field and 256 sets. All other Intel486 processors have a 7-bit set select field and 128 sets. (See Figure 11-1.)

The function of the two entry select bits depends on the state of the control bits. When the fill or read buffers are being accessed, the entry select bits point to the 32-bit location in the buffer being accessed. When a cache location is specified, the entry select bits point to one of the four entries in a set. (Refer to Table 11-2.)

Five testability functions can be performed on the cache. The two control bits in TR5 specify the operation to be executed. The five operations are:

1. Write cache fill buffer
2. Perform a cache testability write
3. Perform a cache testability read
4. Read the cache read buffer
5. Perform a cache flush

Table 11-2 shows the encoding of the two control bits in TR5 for the cache testability functions. Table 11-2 also shows the functionality of the entry and set select bits for each control operation.

The cache tests attempt to use as much of the normal operating circuitry as possible. Therefore, when cache tests are being performed, the cache must be disabled (the CD and NW bits in control register 0 (CR0) must be set to 1 to disable the cache. (See section 7.0, "On-Chip Cache.")

11.2.2 CACHE TESTING REGISTERS FOR THE IntelDX4™ PROCESSOR

The cache testing registers for the IntelDX4 processor differ slightly from the other Intel486 processors. TR3 in the IntelDX4 processor is identical to other Intel486 processors. TR4 in the IntelDX4 processor uses bits 31 to 12 for the Tag field, and bit 11 is unused. TR5 uses bits 11 to 4 for the Set Select field. The Test Registers for the IntelDX4 processor are shown in Figure 11-2.

NOTE:
Software written for the Intel486 processor for testing the cache using the Test Register will produce failures due to the changes in the TAG bits and Set Select bits for the IntelDX4 processor.

Rewrite the code to take into account the 20 TAG bits and 8 Set Select bits to address the larger cache.

11.2.3 CACHE TESTABILITY WRITE

A testability write to the cache is a two step process. First the cache fill buffer must be loaded with 128 bits of data and TR4 loaded with the tag and valid bit. Next the contents of the fill buffer are written to a cache location.

Loading the fill buffer is accomplished by first writing to the entry select bits in TR5 and setting the control bits in TR5 to 00. The entry select bits identify one of four 32-bit locations in the cache fill buffer to put 32 bits of data. Following the write to TR5, TR3 is written with 32 bits of data which are immediately placed in the cache fill buffer. Writing to TR3 initiates the write to the cache fill buffer. The cache fill buffer is loaded with 128 bits of data by writing to TR5 and TR3 four times using a different entry select location each time.

Table 11-2. Cache Control Bit Encoding and Effect of Control Bits on Entry Select and Set Select Functionality

Control Bits		Operation	Entry Select Bits Function	Set Select Bits
Bit 1	Bit 0			
0	0	Enable: Fill Buffer Write Read Buffer Read	Select 32-bit location in fill/read buffer	—
0	1	Perform Cache Write	Select an entry in set	Select a set to write to
1	0	Perform Cache Read	Select an entry in set	Select a set to read from
1	1	Perform Cache Flush	—	—

TR4 must be loaded with the tag and valid bit (bit 10 in TR4) before the contents of the fill buffer are written to a cache location. **The IntelDX4 processor has a 20-bit tag in TR4. All other Intel486 processors use a 21-bit tag in TR4.**

The contents of the cache fill buffer are written to a cache location by writing TR5 with a control field of 01 along with the set select and entry select fields. The set select and entry select field indicate the location in the cache to be written. The normal cache LRU update circuitry updates the internal LRU bits for the selected set.

Note that a cache testability write can only be done when the cache is disabled for replaces (the CD bit in control register 0 is reset to 1). Care must be taken when directly writing to entries in the cache. If the entry is set to overlap an area of memory that is being used in external memory, that cache entry could inadvertently be used instead of the external memory. This is exactly the type of operation that one would desire if the cache were to be used as a high speed RAM. Also, a memory reference (or any external bus cycle) should not occur in between the move to TR4 and the move to TR5, in order to avoid having the value in TR4 change due to the memory reference.

11.2.4 CACHE TESTABILITY READ

A cache testability read is a two-step process. First, the contents of the cache location are read into the cache read buffer. Next, the data is examined by reading it out of the read buffer.

Reading the contents of a cache location into the cache read buffer is initiated by writing TR5 with the control bits set to 10 and the desired set select and two-bit entry select. **The IntelDX4 processor has an 8-bit select field. All other Intel486 processors have a 7-bit select field.** In response to the write to TR5, TR4 is loaded with the 21-bit tag field and the single valid bit from the cache entry read. TR4 is also loaded with the three LRU bits and four valid bits corresponding to the cache set that was accessed. The cache read buffer is filled with the 128-bit value which was found in the data array at the specified location.

The contents of the read buffer are examined by performing four reads of TR3. Before reading TR3 the entry select bits in TR5 must be loaded to indicate which of the four 32-bit words in the read buffer to transfer into TR3 and the control bits in TR5 must be loaded with 00. The register read of TR3 will initiate the transfer of the 32-bit value from the read buffer to the specified general purpose register.

Note that it is very important that the entire 128-bit quantity from the read buffer and also the information from TR4 be read before any memory references are allowed to occur. If memory operations are allowed to happen, the contents of the read buffer will be corrupted. This is because the testability operations use hardware that is used in normal memory accesses for the Intel486 processor whether the cache is enabled or not.

11.2.5 FLUSH CACHE

The control bits in TR5 must be written with 11 to flush the cache. None of the other bits in TR5 have any meaning when 11 is written to the control bits. Flushing the cache will reset the LRU bits and the valid bits to 0, but will not change the cache tag or data arrays.

When the cache is flushed by writing to TR5, the special bus cycle indicating a cache flush to the external system is not run. (See section 10.2.11, "Special Bus Cycles.") For normal operation, the cache should be flushed with the instruction INVD (Invalidate Data Cache) instruction or the WBINVD (Write-back and Invalidate Data Cache) instruction.

11.2.6 ADDITIONAL CACHE TESTING FEATURES FOR ENHANCED WRITE-BACK Intel486™ PROCESSORS

11.2.6.1 Write-Back Enhanced IntelDX2™ Processor Cache Testing

When in Enhanced Bus (write-back) mode, the Write-Back Enhanced IntelDX2 cache testing is a superset of the Standard Bus (write-through) mode. The additional cache testing features for the Write-Back Enhanced IntelDX2 processor are summarized below.

There are two state bits per cache line (VH and VL) instead of one (V). The assignment of VH and VL state bits is listed in the Table 11-3.

Table 11-3. State Bit Assignments for the Write-Back Enhanced IntelDX2™ Processor

State	VH, VL
M	1, 1
E	0, 1
S	1, 0
I	0, 0

When the Write-Back Enhanced IntelDX2 processor is in standard mode, the VH state assignments are identical to the V state assignments of the IntelDX2 processor, which only support S and I states.

TR3 is the same as described above for both Standard and Enhanced Bus modes.

TR4 is the same as described above for the Intel486 processor in Standard Mode. However, in Enhanced Bus mode, the cache line state bits of all four lines of the set are no longer available, to avoid a conflicting definition of state bits for the selected entry. The entry's state bits are moved to positions 0 and 1. Bit 10 is reserved for the possible extension of the tag. The changes to TR4 for Enhanced Bus mode are shown in Figure 11-3.

TR5 is the same as it is for the IntelDX2 processor in standard mode. In Enhanced Bus mode, control bit TR5.SLF (bit 13) is added to allow 1,1 of TR5.CTL (bits 0 and 1) to perform two different kinds of cache flushes. When SLF = 0, CTL = 1,1 performs a single-clock invalidate of all lines in the cache, which will ***not*** write back M-state lines. In the M state, if SLF = 1, the specific line addressed will be written back and invalidated. The state of SLF is significant only when CTL = 1,1. The changes to TR5 for Enhanced Bus mode are shown in Figure 11-4.

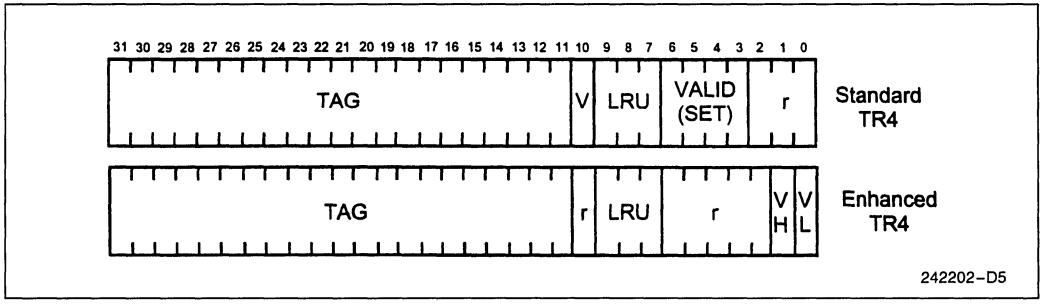


Figure 11-3. TR4 Definition for Standard and Enhanced Bus Modes for the Write-Back Enhanced IntelDX2™ Processor

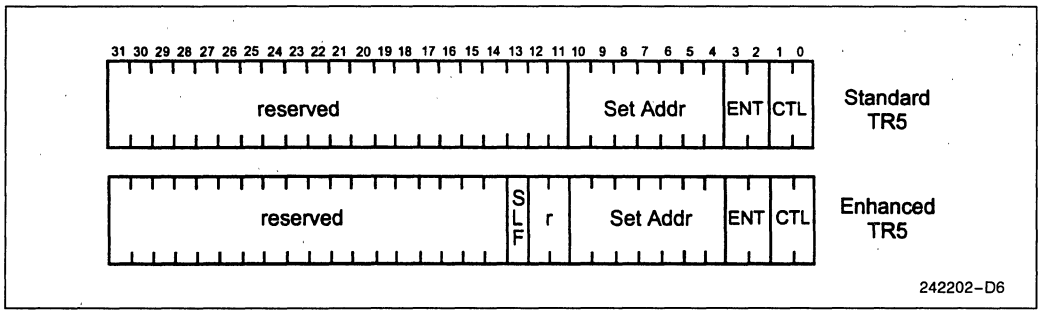


Figure 11-4. TR5 Definition for Standard and Enhanced Bus Modes for the Write-Back Enhanced IntelDX2™ Processor

11.2.6.2 Write-Back Enhanced IntelDX4™ Processor

When in Enhanced Bus (Write-Back) mode, the Write-Back Enhanced IntelDX4 cache testing is a superset of the Standard Bus (Write-Through) mode. The additional cache testing features for the Write-Back Enhanced IntelDX4 processor are summarized below.

There are two state bits per cache line (VH and VL) instead of one (V). The assignment of VH and VL state bits is shown in Table 11-4.

Table 11-4. State Bit Assignments for the Write Back Enhanced IntelDX4™ Processor

State	VH, VL
M	1, 1
E	0, 1
S	1, 0
I	0, 0

The state assignments have been chosen so that VH is identical to the V-state of the IntelDX4 processor, when the Write-Back Enhanced IntelDX4 is in Standard Bus mode and where only S and I states are possible.

There are no changes to TR3 between the Standard Bus mode and the Enhanced Bus mode. The TR4 definition remains the same in Standard Bus mode as in Intel486 processors. The changes to TR4 in Enhanced Bus mode are shown in Figure 11-5.

In Enhanced Bus mode, the cache line state bits of all four lines of the set are no longer available, which eliminates the possibility of a conflicting definition of state bits for the selected entry. The entry's state bits are moved to positions 0 and 1.

TR5 is also the same in Standard Bus mode as in standard Intel486 processors. A minor change to TR5 in Enhanced Bus mode is illustrated in Figure 11-6.

In Enhanced Bus mode, control bit TR5.SLF (bit 13) is added to allow 1,1 of TR5.CTL (bits 1-0) to perform two different kinds of cache flushes. When SLF = 0, CTL = 1,1 performs a single clock invalidate of all lines in the cache, which will **NOT** write-back M-State lines. If SLF = 1, the specific line addressed will be written back (IF in M-State) and invalidated. The state of SLF is significant only when CTL = 1,1.

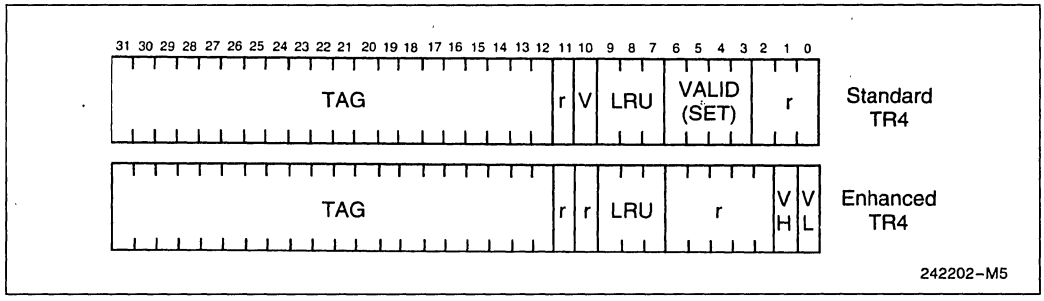


Figure 11-5. TR4 Definition for Standard and Enhanced Bus Modes for the Write-Back Enhanced IntelDX4™ Processor

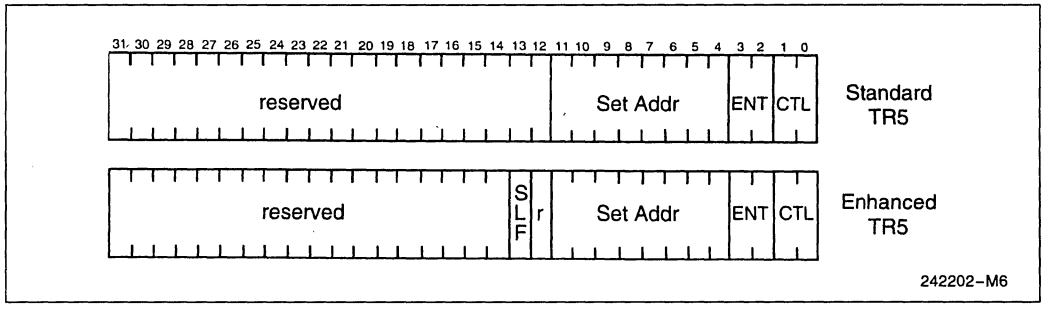


Figure 11-6. TR5 Definition for Standard and Enhanced Bus Modes for the Write-Back Enhanced IntelDX4™ Processor

11.3 Translation Lookaside Buffer (TLB) Testing

The Intel486 processor TLB testability hooks are similar to those in the Intel386 processor. The testability hooks have been enhanced to provide added test features and to include new features in the Intel486 processor. The TLB testability hooks are designed to be accessible during the BIST and for assembly language testing of the TLB.

11.3.1 TRANSLATION LOOKASIDE BUFFER ORGANIZATION

The Intel486 processor TLB is 4-way set associative and has space for 32 entries. The TLB is logically split into three blocks shown in Figure 11-7.

The data block is physically split into four arrays, each with space for eight entries. An entry in the data block is 22 bits wide containing a 20-bit physical address and two bits for the page attributes. The page attributes are the PCD (page cache disable) bit and the PWT (page write-through) bit. Refer to section 7.6, "Page Cacheability," for a discussion of the PCD and PWT bits.

The tag block is also split into four arrays, one for each of the data arrays. A tag entry is 21 bits wide containing a 17-bit linear address and four protection bits. The protection bits are valid (V), user/supervisor (U/S), read/write (R/W) and dirty (D).

The third block contains eight three bit quantities used in the pseudo least recently used (LRU) replacement algorithm. These bits are called the LRU bits. Unlike the on-chip cache, the TLB will replace a valid line even when there is an invalid line in a set.

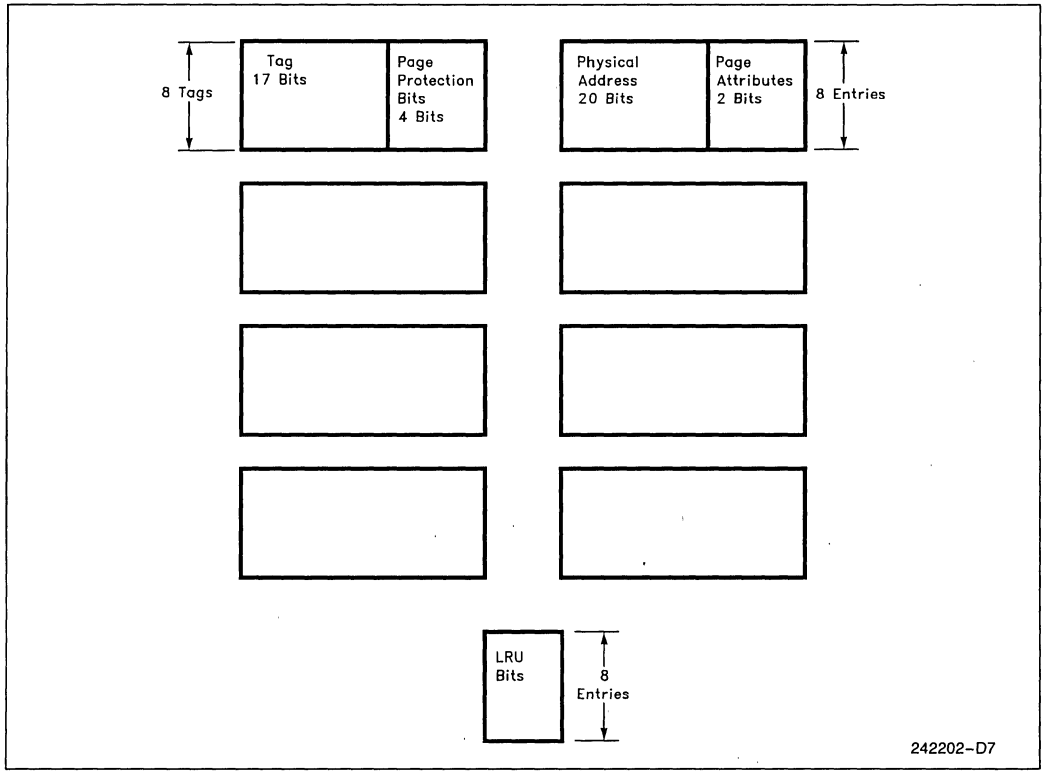


Figure 11-7. TLB Organization

11.3.2 TLB TEST REGISTERS TR6 AND TR7

The two TLB test registers are shown in Figure 11-8. TR6 is the command test register and TR7 is the data test register. External access to these registers is provided through MOV reg,TREG and MOV TREG,reg instructions.

Command Test Register: TR6

TR6 contains the tag information and control information used in a TLB test. Loading TR6 with tag and control information initiates a TLB write or lookup test.

TR6 contains three bit fields, a 20-bit linear address (bits 12–31), seven bits for the TLB tag protection bits (bits 5-11) and one bit (bit 0) to define the type of operation to be performed on the TLB.

The 20-bit linear address forms the tag information used in the TLB access. The lower three bits of the linear address select which of the eight sets are accessed. The upper 17 bits of the linear address form the tag stored in the tag array.

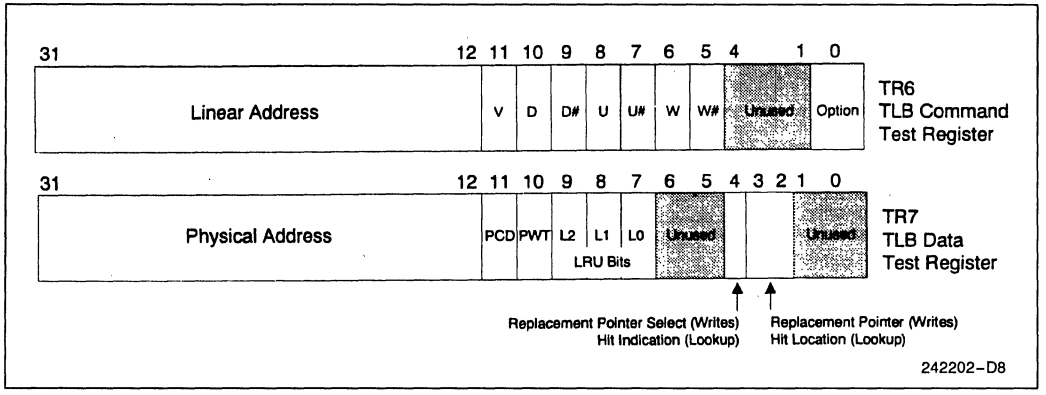


Figure 11-8. TLB Test Registers

The seven TLB tag protection bits are described below.

V: The valid bit for this TLB entry

D,D#: The dirty bit for/from the TLB entry

U,U#: The user/supervisor bit for/from the TLB entry

W,W#: The read/write bit for/from the TLB entry

Two bits are used to represent the D, U/S and R/W bits in the TLB tag to permit the option of a forced miss or hit during a TLB lookup operation. The forced miss or hit will occur regardless of the state of the actual bit in the TLB. The meaning of these pairs of bits is given in Table 11-5.

The operation bit in TR6 determines if the TLB test operation will be a write or a lookup. The function of the operation bit is given in Table 11-6.

Table 11-5. Meaning of a Pair of TR6 Protection Bits

TR6 Protection Bit (B)	TR6 Protection Bit# (B#)	Meaning on TLB Write Operation	Meaning on TLB Lookup Operation
0	0	Undefined	Miss any TLB TAG Bit B
0	1	Write 0 to TLB TAG Bit B	Match TLB TAG Bit B if 0
1	0	Write 1 to TLB TAG Bit B	Match TLB TAG Bit B if 1
1	1	Undefined	Match any TLB TAG Bit B

Table 11-6. TR6 Operation Bit Encoding

TR6 Bit 0	TLB Operation to Be Performed
0	TLB Write
1	TLB Lookup

Data Test Register: TR7

TR7 contains the information stored or read from the data block during a TLB test operation. Before a TLB test write, TR7 contains the physical address and the page attribute bits to be stored in the entry. After a TLB test lookup hit, TR7 contains the physical address, page attributes, LRU bits and entry location from the access.

TR7 contains a 20-bit physical address (bits 12–31), PLD bit (bit 11), PWT bit (bit 10), and three bits for the LRU bits (bits 7–9). The LRU bits in TR7 are only used during a TLB lookup test. The functionality of TR7 bit 4 differs for TLB writes and lookups. The encoding of bit 4 is defined in Table 11-7 and Table 11-8. Finally, TR7 contains two bits (bits 2–3) to specify a TLB replacement pointer or the location of a TLB hit.

Table 11-7. Encoding of Bit 4 of TR7 on Writes

TR7 Bit 4	Replacement Pointer Used on TLB Write
0	Pseudo-LRU Replacement Pointer
1	Data Test Register Bits 3:2

A replacement pointer is used during a TLB write. The pointer indicates which of the four entries in an accessed set is to be written. The replacement pointer can be specified to be the internal LRU bits or bits 2–3 in TR7. The source of the replacement pointer is specified by TR7 bit 4. The encoding of bit 4 during a write is given by Table 11-7.

Note that both testability writes and lookups affect the state of the internal LRU bits regardless of the replacement pointer used. All TLB write operations (testability or normal operation) cause the written entry to become the most recently used. For example, during a testability write with the replacement pointer specified by TR7 bits 2–3, the indicated entry is written and that entry becomes the most recently used as specified by the internal LRU bits.

There are two TLB testing operations: write entries into the TLB, and perform TLB lookups. One major enhancement over TLB testing in the Intel386 processor is that paging need not be disabled while executing testability writes or lookups.

Note that any time one TLB set contains the same linear address in more than one of its entries, looking up that linear address will give unpredictable results. Therefore a single linear address should not be written to one TLB set more than once.

Table 11-8. Encoding of Bit 4 of TR7 on Lookups

TR7 Bit 4	Meaning after TLB Lookup Operation
0	TLB Lookup Resulted in a Miss
1	TLB Lookup Resulted in a Hit

11.3.3 TLB WRITE TEST

To perform a TLB write TR7 must be loaded followed by a TR6 load. The register operations must be performed in this order because the TLB operation is triggered by the write to TR6.

TR7 is loaded with a 20-bit physical address and values for PCD and PWT to be written to the data portion of the TLB. In addition, bit 4 of TR7 must be loaded to indicate whether to use TR7 bits 3-2 or the internal LRU bits as the replacement pointer on the TLB write operation. Note that the LRU bits in TR7 are not used in a write test.

TR6 must be written to initiate the TLB write operation. Bit 0 in TR6 must be reset to zero to indicate a TLB write. The 20-bit linear address and the seven page protection bits must also be written in TR6 to specify the tag portion of the TLB entry. Note that the three least significant bits of the linear address specify which of the eight sets in the data block will be loaded with the physical address data. Thus only 17 of the linear address bits are stored in the tag array.

11.3.4 TLB LOOKUP TEST

To perform a TLB lookup it is only necessary to write the proper tags and control information into TR6. Bit 0 in TR6 must be set to 1 to indicate a TLB lookup. TR6 must be loaded with a 20-bit linear address and the seven protection bits. To force misses and matches of the individual protection bits on TLB lookups, set the seven protection bits as specified in Table 11-5.

A TLB lookup operation is initiated by the write to TR6. TR7 will indicate the result of the lookup operation following the write to TR6. The hit/miss indication can be found in TR7 bit 4 (see Table 11-8).

TR7 will contain the following information if bit 4 indicated that the lookup test resulted in a hit. Bits 2–3 will indicate in which set the match occurred. The 22 most significant bits in TR7 will contain the physical address and page attributes contained in the entry. Bits 9–7 will contain the LRU bits associated with the accessed set. The state of the LRU bits is previous to their being updated for the current lookup.

If bit 4 in TR7 indicated that the lookup test resulted in a miss the remaining bits in TR7 are undefined.

Again it should be noted that a TLB testability lookup operation affects the state of the LRU bits. The LRU bits will be updated if a hit occurred. The entry which was hit will become the most recently used.

11.4 Tri-State Output Test Mode

The Intel486 processor provides the ability to float all its outputs and bidirectional pins, except for the VOLDET pin in the IntelDX4 processor. This includes all pins floated during bus hold as well as pins which are never floated in normal operation of the chip (HLDA, BREQ, FERR# and PCHK#). When the Intel486 processor is in the tri-state output test mode external testing can be used to test board connections.

The tri-state test mode is invoked if FLUSH# is sampled active at the falling edge of RESET. FLUSH# is an asynchronous signal. When driven, FLUSH# should be asserted for 2 clocks before and 2 clocks after RESET is de-asserted. If FLUSH# is driven synchronously, the tri-state output test mode is initiated by driving FLUSH# so that it is sampled active in the clock prior to RESET going low and ensuring that specified setup and hold times are met. The outputs are guaranteed to tri-state no later than 10 clocks after RESET goes low (see Figure 9.6). The Intel486 processor remains in the tri-state test mode until the next RESET.

11.5 Intel486™ Processor Boundary Scan (JTAG)

The Intel486 processor provides additional testability features compatible with the IEEE Standard Test Access Port and Boundary Scan Architecture (IEEE Std. 1149.1). (Note that the Intel486 SX processor in PGA package does *not* have JTAG capability.) The test logic provided allows for testing to insure that components function correctly, that interconnections between various components are correct, and that various components interact correctly on the printed circuit board.

The boundary scan test logic consists of a boundary scan register and support logic that are accessed through a test access port (TAP). The TAP provides a simple serial interface that makes it possible to test all signal traces with only a few probes.

The TAP can be controlled via a bus master. The bus master can be either automatic test equipment or a component (PLD) that interfaces to the four-pin test bus.

11.5.1 BOUNDARY SCAN ARCHITECTURE

The boundary scan test logic contains the following elements:

- Test access port (TAP), consisting of input pins TMS, TCK, and TDI; and output pin TDO.
- TAP controller, which interprets the inputs on the test mode select (TMS) line and performs the corresponding operation. The operations performed by the TAP include controlling the instruction and data registers within the component.
- Instruction register (IR), which accepts instruction codes shifted into the test logic on the test data input (TDI) pin. The instruction codes are used to select the specific test operation to be performed or the test data register to be accessed.
- Test data registers: The Intel486 processor contains three test data registers: Bypass register (BPR), Device Identification register (DID), and Boundary Scan register (BSR).

The instruction and test data registers are separate shift-register paths connected in parallel and have a common serial data input and a common serial data output connected to the TAP signals, TDI and TDO, respectively.

11.5.2 DATA REGISTERS

The Intel486 processor contains the two required test data registers; bypass register and boundary

scan register. In addition, they also have a device identification register.

Each test data register is serially connected to TDI and TDO, with TDI connected to the most significant bit and TDO connected to the least significant bit of the test data register.

Data is shifted one stage (bit position within the register) on each rising edge of the test clock (TCK). In addition the Intel486 processor contains a runbist register to support the RUNBIST boundary scan instruction.

11.5.2.1 Bypass Register

The Bypass Register is a one-bit shift register that provides the minimal length path between TDI and TDO. This path can be selected when no test operation is being performed by the component to allow rapid movement of test data to and from other components on the board. While the bypass register is selected data is transferred from TDI to TDO without inversion.

11.5.2.2 Boundary Scan Register

The Boundary Scan Register is a single shift register path containing the boundary scan cells that are connected to all input and output pins of the Intel486 processor. Figure 11-9 shows the logical structure of the boundary scan register. While output cells determine the value of the signal driven on the corresponding pin, input cells only capture data; they do not affect the normal operation of the device. Data is transferred without inversion from TDI to TDO through the boundary scan register during scanning. The boundary scan register can be operated by the EXTEST and SAMPLE instructions. The boundary scan register order is described in section 11.5.5 "Boundary Scan Register Bits and Bit Orders."

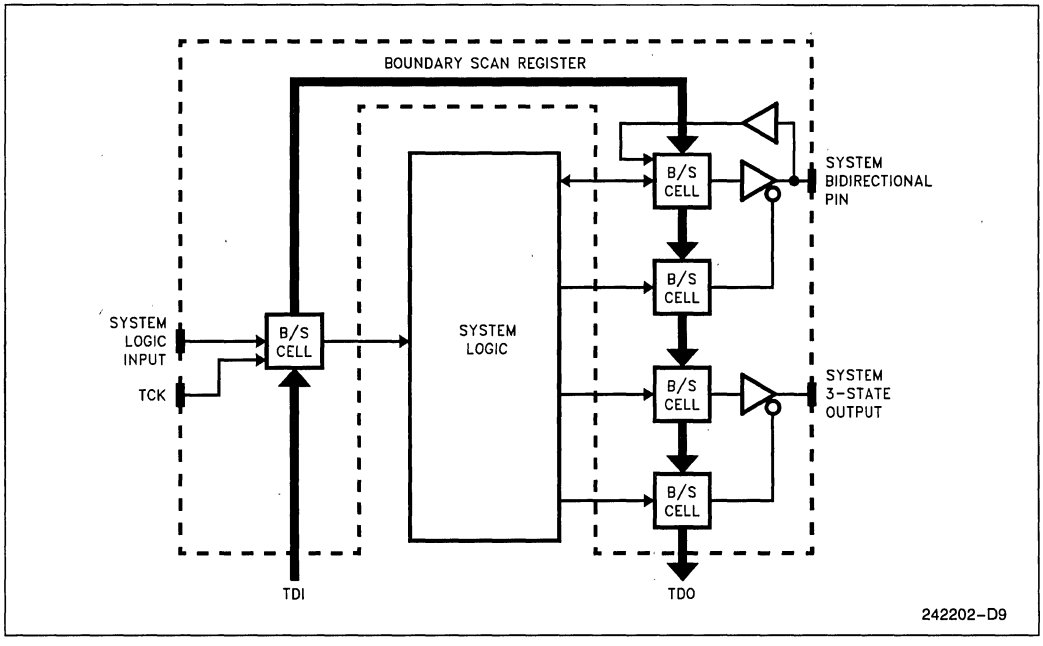


Figure 11-9. Logical Structure of Boundary Scan Register

11.5.2.3 Device Identification Register

The Device Identification Register contains the manufacturer's identification code, part number code, and version code. Table 11-9 lists the codes corresponding to the Intel486 processor.

11.5.2.4 Runbist Register

The Runbist Register is a one bit register used to report the results of the Intel486 processor BIST when it is initiated by the RUNBIST instruction. This register is loaded with a "1" prior to invoking the BIST and is loaded with "0" upon successful completion.

11.5.3 INSTRUCTION REGISTER

The Instruction Register (IR) allows instructions to be serially shifted into the device. The instruction selects the particular test to be performed, the test data register to be accessed, or both. The instruction register is four (4) bits wide. The most significant bit is connected to TDI and the least significant bit is connected to TDO. There are no parity bits associated with the Instruction register. Upon entering the Capture-IR TAP controller state, the Instruction register is loaded with the default instruction "0001," SAMPLE/PRELOAD. Instructions are shifted into the instruction register on the rising edge of TCK while the TAP controller is in the SHIFT-IR state.

11.5.3.1 Boundary Scan Instruction Set

The Intel486 processor supports all three mandatory boundary scan instructions (BYPASS, SAMPLE/PRELOAD, and EXTEST) along with two optional instructions (IDCODE and RUNBIST). Table 11-10 lists the Intel486 processor boundary scan instruction codes. The instructions listed as PRIVATE cause TDO to become enabled in the Shift-DR state and cause "0" to be shifted out of TDO on the rising edge of TCK. Execution of the PRIVATE instructions will not cause hazardous operation of the Intel486 processor.

EXTEST The instruction code is "0000." The EXTEST instruction allows testing of circuitry external to the component package, typically board interconnects. It does so by driving the values loaded into the Intel486 processor's boundary scan register out on the output pins corresponding to each boundary scan cell and capturing the values on Intel486 processor input pins to be loaded into their corresponding boundary scan register locations. I/O pins are selected as input or output, depending on the value loaded into their control setting locations in the boundary scan register. Values shifted into input latches in the boundary scan register are never used by the internal logic of the Intel486 processor.

Table 11-9. Boundary Scan Component Identification Codes

Processor Type	Version	V_{CC} 1 = 3.3V 0 = 5V	Intel Architecture Type	Family	Model	MFG ID Intel = 009H	1st Bit	Boundary Scan ID (Hex)
Intel486™ SX processor (3.3V)	xxxx*	1	000001	0100	00010	00000001001	1	x8282013H
Intel486 SX processor (3.3V, 2X CLK)	xxxx*	1	000001	0100	00010	00000001001	1	x8282013H
Intel486 SX processor (5V)	xxxx*	0	000001	0100	00010	00000001001	1	x0282013H
Intel486 SX processor (5V, 2X CLK)	xxxx*	0	000001	0100	00010	00000001001	1	x0282013H
IntelSX2™ processor	xxxx*	0	000001	0100	00101	00000001001	1	x0286013H
Intel486 DX processor (3.3V)	xxxx*	1	000001	0100	00001	00000001001	1	x8281013H
Intel486 DX processor (3.3V, 2X CLK)	xxxx*	1	000001	0100	00001	00000001001	1	x8281013H
Intel486 DX processor (5V)	xxxx*	0	000001	0100	00001	00000001001	1	x0281013H
Intel486 DX processor (5V, 2X CLK)	xxxx*	0	000001	0100	00001	00000001001	1	x0281013H
IntelDX2™ processor (3.3V)	xxxx*	1	000001	0100	00101	00000001001	1	x8285013H
IntelDX2 processor (5V)	xxxx*	0	000001	0100	00101	00000001001	1	x0285013H
Write-Back Enhanced IntelDX2 processor (3.3V)	xxxx*	1	000001	0100	00111	00000001001	1	x8287013H
Write-Back Enhanced IntelDX2 processor (5V)	xxxx*	0	000001	0100	00111	00000001001	1	x0287013H
IntelDX4™ processor (3.3V)	xxxx*	1	000001	0100	01000	00000001001	1	x8288013H
Write-Back Enhanced IntelDX4 processor (3.3V)	xxxx*	1	000001	0100	01001	00000001001	1	X8289013H

NOTE:
*Contact Intel for details

Table 11-10. Boundary Scan Instruction Codes

Instruction Code	Instruction Name
0000	EXTEST
0001	SAMPLE
0010	IDCODE
0011	PRIVATE
0100	PRIVATE
0101	PRIVATE
0110	PRIVATE
0111	PRIVATE
1000	RUNBIST
1001	PRIVATE
1010	PRIVATE
1011	PRIVATE
1100	PRIVATE
1101	PRIVATE
1110	PRIVATE
1111	BYPASS

NOTE:
After using the EXTEST instruction, the Intel486 processor must be reset before normal (non-boundary scan) use.

SAMPLE/PRELOAD The instruction code is "0001." The SAMPLE/PRELOAD has two functions that it performs. When the TAP controller is in the Capture-DR state, the SAMPLE/PRELOAD instruction allows a "snap-shot" of the normal operation of the component without interfering with that normal operation. The instruction causes boundary scan register cells associated with outputs to sample the value being driven by the Intel486 processor. It causes the cells associated with inputs to sample the value being driven into the Intel486 processor. On both outputs and inputs the sampling occurs on the rising edge of TCK. When the TAP controller is in the Update-DR state, the SAMPLE/PRELOAD instruction preloads data to the device pins to be driven to the board by executing the EXTEST instruction. Data is preloaded to the pins from the boundary scan register on the falling edge of TCK.

IDODE The instruction code is "0010." The IDCODE instruction selects the device identification register to be connected to TDI and TDO, allowing the device identification code to be shifted out of the device on TDO. Note that the device identification register is not altered by data being shifted in on TDI.

BYPASS The instruction code is "1111." The BYPASS instruction selects the bypass register to be connected to TDI and TDO, effectively bypassing the test logic on the Intel486 processor by reducing the shift length of the device to one bit. Note that an open circuit fault in the board level test data path will cause the bypass register to be selected following an instruction scan cycle due to the pull-up resistor on the TDI input. This has been done to prevent any unwanted interference with the proper operation of the system logic.

RUNBIST The instruction code is "1000." The RUNBIST instruction selects the one (1) bit runbist register, loads a value of "1" into the runbist register, and connects it to TDO. It also initiates the built-in self test (BIST) feature of the Intel486 processor, which is able to detect approximately 60% of the stuck-at faults on the Intel486 processor. The Intel486 processor ac/dc specifications for V_{CC} and CLK must be met and RESET must have been asserted at least once prior to executing the RUNBIST boundary scan instruction. After loading the RUNBIST instruction code in the instruction register, the TAP controller must be placed in the Run-Test/Idle state. BIST begins on the first rising edge of TCK after entering the Run-Test/Idle state. The TAP controller must remain in the Run-Test/Idle state until BIST is completed. It requires 1.2 million clock (CLK) cycles to complete BIST and report the result to the runbist register. After completing the 1.2 million clock (CLK) cycles, the value in the runbist register should be shifted out on

TDO during the Shift-DR state. A value of "0" being shifted out on TDO indicates BIST successfully completed. A value of "1" indicates a failure occurred. After executing the RUNBIST instruction, the Intel486 processor must be reset prior to normal operation.

11.5.4 TEST ACCESS PORT (TAP) CONTROLLER

The TAP controller is a synchronous, finite state machine. It controls the sequence of operations of the test logic. The TAP controller changes state only in response to the following events:

1. a rising edge of TCK
2. power-up.

The value of the test mode state (TMS) input signal at a rising edge of TCK controls the sequence of the state changes. The state diagram for the TAP controller is shown in Figure 11-8. Test designers must consider the operation of the state machine in order to design the correct sequence of values to drive on TMS.

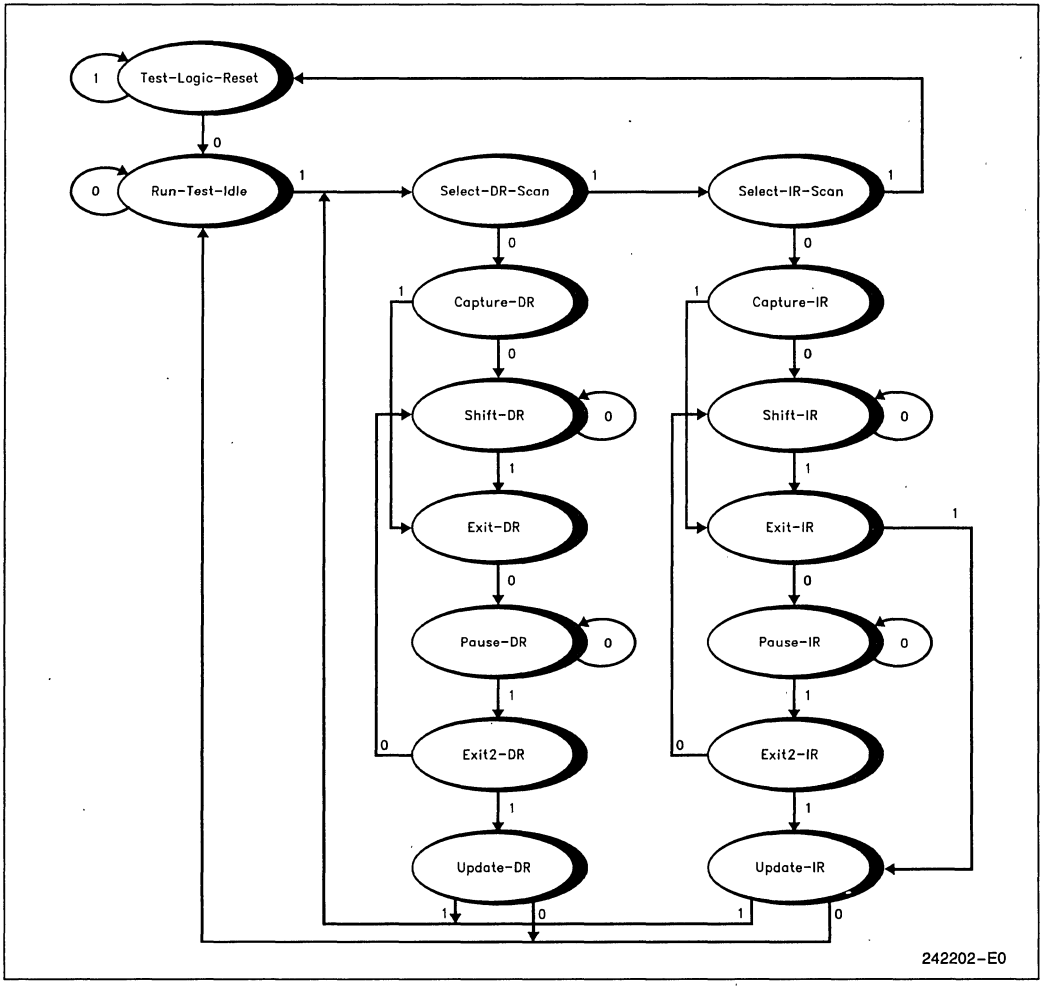


Figure 11-10. TAP Controller State Diagram

11.5.4.1 Test-Logic-Reset State

In this state, the test logic is disabled so that normal operation of the device can continue unhindered. This is achieved by initializing the instruction register such that the IDCODE instruction is loaded. No matter what the original state of the controller, the controller enters Test-Logic-Reset state when the TMS input is held high (1) for at least five rising edges of TCK. The controller remains in this state while TMS is high. The TAP controller is also forced to enter this state at power-up.

11.5.4.2 Run-Test/Idle State

A controller state between scan operations. Once in this state, the controller remains in this state as long as TMS is held low. In devices supporting the RUNBIST instruction, the BIST is performed during this state and the result is reported in the runbist register. For instruction not causing functions to execute during this state, no activity occurs in the test logic. The instruction register and all test data registers retain their previous state. When TMS is high and a rising edge is applied to TCK, the controller moves to the Select-DR state.

11.5.4.3 Select-DR-Scan State

This is a temporary controller state. The test data register selected by the current instruction retains its previous state. If TMS is held low and a rising edge is applied to TCK when in this state, the controller moves into the Capture-DR state, and a scan sequence for the selected test data register is initiated. If TMS is held high and a rising edge is applied to TCK, the controller moves to the Select-IR-Scan state.

The instruction does not change in this state.

11.5.4.4 Capture-DR State

In this state, the boundary scan register captures input pin data if the current instruction is EXTEST or SAMPLE/PRELOAD. The other test data registers, which do not have parallel input, are not changed.

The instruction does not change in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-DR state if TMS is high or the Shift-DR state if TMS is low.

11.5.4.5 Shift-DR State

In this controller state, the test data register connected between TDI and TDO as a result of the current instruction shifts data one stage toward its serial output on each rising edge of TCK.

The instruction does not change in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-DR state if TMS is high or remains in the Shift-DR state if TMS is low.

11.5.4.6 Exit1-DR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-DR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Pause-DR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

11.5.4.7 Pause-DR State

The pause state allows the test controller to temporarily halt the shifting of data through the test data register in the serial path between TDI and TDO. An example of using this state could be to allow a tester to reload its pin memory from disk during application of a long test sequence.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

The controller remains in this state as long as TMS is low. When TMS goes high and a rising edge is applied to TCK, the controller moves to the Exit2-DR state.

11.5.4.8 Exit2-DR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-DR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Shift-DR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

11.5.4.9 Update-DR State

The boundary scan register is provided with a latched parallel output to prevent changes at the parallel output while data is shifted in response to the EXTEST and SAMPLE/PRELOAD instructions. When the TAP controller is in this state and the boundary scan register is selected, data is latched onto the parallel output of this register from the shift-register path on the falling edge of TCK. The data held at the latched parallel output does not change other than in this state.

All test data registers selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

11.5.4.10 Select-IR-Scan State

This is a temporary controller state. The test data register selected by the current instruction retains its previous value. If TMS is held low and a rising edge is applied to TCK when in this state, the controller moves into the Capture-IR state, and a scan sequence for the instruction register is initiated. If TMS is held high and a rising edge is applied to TCK, the controller moves to the Test-Logic-Reset state.

The instruction does not change in this state.

11.5.4.11 Capture-IR State

In this controller state the shift register contained in the instruction register loads the fixed value "0001" on the rising edge of TCK.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state. When the controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-IR state if TMS is held high, or the Shift-IR state if TMS is held low.

11.5.4.12 Shift-IR State

In this state the shift register contained in the instruction register is connected between TDI and TDO and shifts data one stage towards its serial output on each rising edge of TCK.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

When the controller is in this state and a rising edge is applied to TCK, the controller enters the Exit1-IR state if TMS is held high, or remains in the Shift-IR state if TMS is held low.

11.5.4.13 Exit1-IR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-IR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Pause-IR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

11.5.4.14 Pause-IR State

The pause state allows the test controller to temporarily halt the shifting of data through the instruction register.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

The controller remains in this state as long as TMS is low. When TMS goes high and a rising edge is applied to TCK, the controller moves to the Exit2-IR state.

11.5.4.15 Exit2-IR State

This is a temporary state. While in this state, if TMS is held high, a rising edge applied to TCK causes the controller to enter the Update-IR state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the Shift-IR state.

The test data register selected by the current instruction retains its previous value during this state. The instruction does not change in this state.

11.5.4.16 Update-IR State

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of TCK. Once the new instruction has been latched, it becomes the current instruction.

Test data registers selected by the new current instruction retain the previous value.

11.5.5 BOUNDARY SCAN REGISTER BITS AND BIT ORDERS

The boundary scan register contains a cell for each pin, as well as cells for control of I/O and tri-state pins.

Intel486™ SX and IntelSX2™ Processor Boundary Scan Register Bits

The following is the bit order of the Intel486 SX and IntelSX2 processor boundary scan register (from left to right and top to bottom. See notes below):

TDO ← A2, A3, A4, A5, UP#, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, Reserved, Reserved, SMI#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRTL ← TDI

Intel486™ DX and IntelDX2™ Processor Boundary Scan Register Bits

The following is the bit order of the Intel486 DX and IntelDX2 processor boundary scan register (from left to right and top to bottom. See notes below):

TDO ← A2, A3, A4, A5, UP#, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, IGNNE#, FERR#, SMI#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRTL ← TDI

50-MHz Intel486™ DX Processor Boundary Scan Register Bits

The following is the bit order of the 50-MHz Intel486 DX processor boundary scan register (from left to right and top to bottom. See notes below):

TDO ← A2, A3, A4, A5, UP#, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, IGNNE#, FERR#, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRTL ← TDI

Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processor Boundary Scan Register Bits

The following is the bit order of the Write-Back Enhanced IntelDX2 processor boundary scan register (from left to right and top to bottom. See notes below).

TDO ← A2, A3, A4, A5, UP#, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, IGNNE#, INV, CACHE#, FERR#, SMI#, WB/WT#, HITM#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, Reserved, AHOLD, HOLD, KEN#, RDY#, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRTL ← TDI

IntelDX4™ Processor Boundary Scan Register Bits

The following is the bit order of the IntelDX4 processor boundary scan register (from left to right and top to bottom. See notes below).

TDO ← A2, A3, A4, A5, UP#, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, IGNNE#, FERR#, SMI#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3#, BE2#, BE1#, BE0#, BREQ, W/R#, HLDA, CLK, AHOLD, HOLD, KEN#, RDY#, CLKMUL, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRTL ← TDI

11.5.6 WRITE-BACK ENHANCED IntelDX4™ PROCESSORS BOUNDARY SCAN REGISTER BITS

The following is the bit order of the Write-Back Enhanced IntelDX4 processor boundary scan register (from left to right and top to bottom. See notes below).

TDO ← A2, A3, A4, A5, UP#, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24, A25, A26, A27, A28, A29, A30, A31, DP0, D0, D1, D2, D3, D4, D5, D6, D7, DP1, D8, D9, D10, D11, D12, D13, D14, D15, DP2, D16, D17, D18, D19, D20, D21, D22, D23, DP3, D24, D25, D26, D27, D28, D29, D30, D31, STPCLK#, IGNNE#, INV, CACHE#, FERR#, SMI#, WB/WT#, HITM#, SMIACT#, SRESET, NMI, INTR, FLUSH#, RESET, A20M#, EADS#, PCD, PWT, D/C#, M/IO#, BE3, BE2, BE1, BE0, BREQ, W/R#, HLDA, CLK, AHOLD, HOLD, KEN#, RDY#, CLKMUL, BS8#, BS16#, BOFF#, BRDY#, PCHK#, LOCK#, PLOCK#, BLAST#, ADS#, MISCCTL, BUSCTL, ABUSCTL, WRcTL ← TDI

NOTES:

"Reserved" corresponds to no connect "NC" or "INC" signals on the Intel486 processor.

All the *CTL cells are control cells that are used to select the direction of bidirectional pins or tri-state output pins. If "1" is loaded into the control cell (*CTL), the associated pin(s) are tri-stated or selected as input. The following lists the control cells and their corresponding pins.

1. WRCTL controls the D31–D0 and DP3–DP0 pins.
2. ABUSCTL controls the A31–A2 pins.
3. BUSCTL controls the ADS#, BLAST#, PLOCK#, LOCK#, WR#, BE0#, BE1#, BE2#, BE3#, MIO#, DC#, PWT, and PCD pins.
4. MISCCTL controls the PCHK#, HLDA, and BREQ pins.

11.5.7 TAP CONTROLLER INITIALIZATION

The TAP controller is automatically initialized when a device is powered up. In addition, the TAP controller can be initialized by applying a high signal level on the TMS input for five TCK periods.

11.5.8 BOUNDARY SCAN DESCRIPTION LANGUAGE (BSDL) FILES

See Appendix D for an example of a BSDL file for Intel486 processors.

12.0 DEBUGGING SUPPORT

The Intel486 processor provides several features that simplify the debugging process. The three categories of on-chip debugging aids are:

1. Code execution breakpoint opcode (0CCH),
2. Single-step capability provided by the TF bit in the flag register, and
3. Code and data breakpoint capability provided by the Debug Registers DR0–3, DR6, and DR7.

12.1 Breakpoint Instruction

A single-byte-opcode breakpoint instruction is available for use by software debuggers. The breakpoint opcode is 0CCH, and generates an exception 3 trap when executed. In typical use, a debugger program can "plant" the breakpoint instruction at all desired code execution breakpoints. The single-byte breakpoint opcode is an alias for the two-byte general software interrupt instruction, INT n, where n=3. The only difference between INT 3 (0CCh) and INT n is that INT 3 is never IOPL-sensitive, while INT n is IOPL-sensitive in Protected Mode and Virtual 8086 Mode.

12.2 Single-Step Trap

If the single-step flag (TF, bit 8) in the EFLAG register is found to be set at the end of an instruction, a single-step exception occurs. The single-step exception is auto vectored to exception number 1. Precisely, exception 1 occurs as a trap after the instruction following the instruction which set TF. In typical practice, a debugger sets the TF bit of a flag register image on the debugger's stack. It then typically transfers control to the user program and loads the flag image with a signal instruction, the IRET instruction. The single-step trap occurs after executing one instruction of the user program.

Because exception 1 occurs as a trap (that is, it occurs after the instruction has already executed), the CS:EIP pushed onto the debugger's stack points to the next unexecuted instruction of the program being debugged. An exception 1 handler, merely by ending with an IRET instruction, can therefore efficiently support single-stepping through a user program.

12.3 Debug Registers

The Debug Registers are an advanced debugging feature of the Intel486 processor. They allow data access breakpoints as well as code execution breakpoints. Because the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT3 breakpoint opcode.

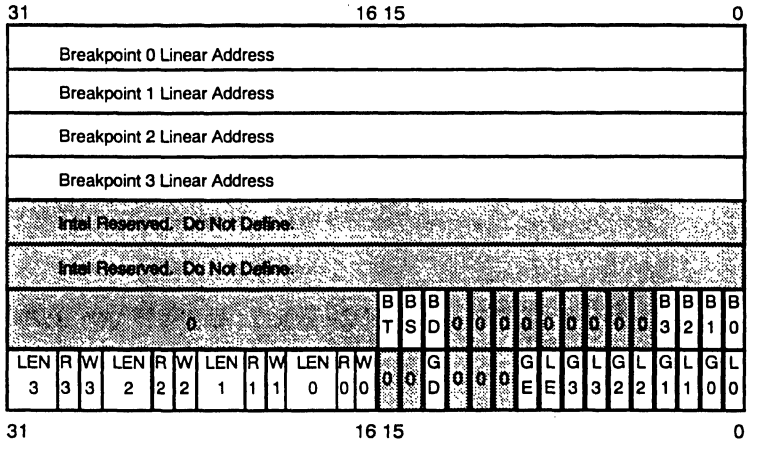
The Intel486 processor contains six Debug Registers, providing the ability to specify up to four distinct breakpoints addresses, breakpoint control options, and read breakpoint status. Initially after reset, breakpoints are in the disabled state. Therefore, no breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are auto vectored to exception number 1.

12.3.1 LINEAR ADDRESS BREAKPOINT REGISTERS (DR0–DR3)

Up to four breakpoint addresses can be specified by writing into Debug Registers DR0–DR3, shown in Figure 12-1. The breakpoint addresses specified are 32-bit linear addresses. Intel486 processor hardware continuously compares the linear breakpoint addresses in DR0–DR3 with the linear addresses generated by executing software (a linear address is the result of computing the effective address and adding the 32-bit segment base address). Note that if paging is not enabled the linear address equals the physical address. If paging is enabled, the linear address is translated to a physical 32-bit address by the on-chip paging unit. Regardless of whether paging is enabled or not, however, the breakpoint registers hold linear addresses.

12.3.2 DEBUG CONTROL REGISTER (DR7)

A Debug Control Register, DR7 shown in Figure 12-1, allows several debug control functions such as enabling the breakpoints and setting up other control options for the breakpoints. The fields within the Debug Control Register, DR7, are as follows:



NOTE:
See section 4.2.7

Figure 12-1. Debug Registers

4

LENi (breakpoint length specification bits)

A 2-bit LEN field exists for each of the four breakpoints. LEN specifies the length of the associated breakpoint field. The choices for data breakpoints are: 1 byte, 2 bytes, and 4 bytes. Instruction execution breakpoints must have a length of 1 (LENi = 00). Encoding of the LENi field is as described in Table 12-1.

The LENi field controls the size of breakpoint field i by controlling whether all low-order linear address bits in the breakpoint address register are used to detect the breakpoint event. Therefore, all breakpoint fields are aligned; 2-byte breakpoint fields begin on Word boundaries, and 4-byte breakpoint fields begin on Dword boundaries.

Figure 12-2 is an example of various size breakpoint fields. Assume the breakpoint linear address in DR2 is 00000005H. In that situation, the Figure 12-2 indicates the region of the breakpoint field for lengths of 1, 2, or 4 bytes.

RWi (memory access qualifier bits)

A 2-bit RW field exists for each of the four breakpoints. The 2-bit RW field specifies the type of usage which must occur in order to activate the associated breakpoint.

Table 12-1. LENi Encoding

LENi Encoding	Breakpoint Field Width	Usage of Least Significant Bits in Breakpoint Address Register i, (i=0-3)
00	1 byte	All 32-bits used to specify a single-byte breakpoint field.
01	2 bytes	A1–A31 used to specify a two-byte, word-aligned breakpoint field. A0 in Breakpoint Address Register is not used.
10	Undefined—do not use this encoding	
11	4 bytes	A2–A31 used to specify a four-byte, dword-aligned breakpoint field. A0 and A1 in Breakpoint Address Register are not used.

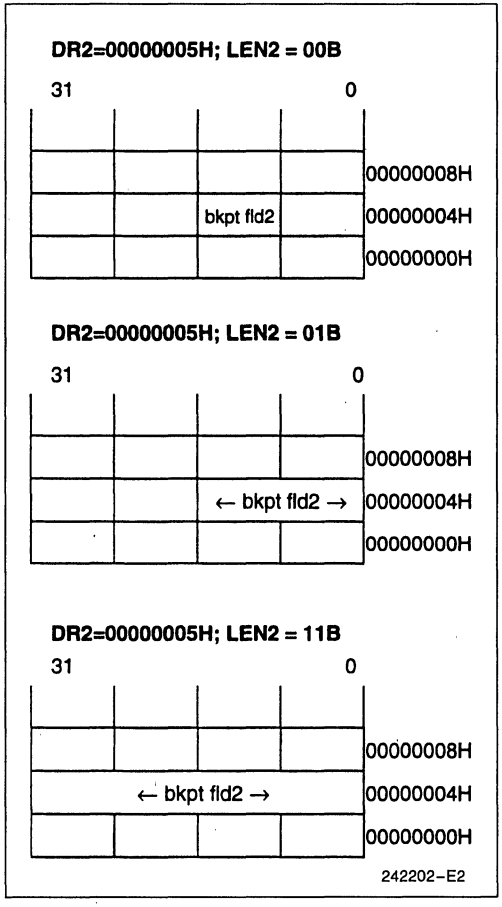


Figure 12-2. Size Breakpoint Fields

Table 12-2. RW Encoding

RW Encoding	Usage Causing Breakpoint
00	Instruction execution only
01	Data writes only
10	Undefined-do not use this encoding
11	Data reads and writes only

RW encoding 00 is used to set up an instruction execution breakpoint. RW encodings 01 or 11 are used to set up write-only or read/write data breakpoints.

Note that **instruction execution breakpoints are taken as faults** (i.e., before the instruction executes), but **data breakpoints are taken as traps** (i.e., after the data transfer takes place).

Using LENi and RWi to Set Data Breakpoint i

A data breakpoint can be set up by writing the linear address into DRi (i = 0–3). For data breakpoints, RWi can = 01 (write-only) or 11 (write/read). LEN can = 00, 01, or 11.

If a data access entirely or partly falls within the data breakpoint field, the data breakpoint condition has occurred, and if the breakpoint is enabled, an exception 1 trap will occur.

Using LENi and RWi to Set Instruction Execution Breakpoint i

An instruction execution breakpoint can be set up by writing address of the beginning of the instruction (including prefixes if any) into DRi (i = 0–3). RWi must = 00 and LEN must = 00 for instruction execution breakpoints.

If the instruction beginning at the breakpoint address is about to be executed, the instruction execution breakpoint condition has occurred, and if the breakpoint is enabled, an exception 1 fault will occur before the instruction is executed.

Note that an instruction execution breakpoint address must be equal to the **beginning** byte address of an instruction (including prefixes) in order for the instruction execution breakpoint to occur.

GD (Global Debug Register access detect)

The Debug Registers can only be accessed in Real Mode or at privilege level 0 in Protected Mode. The GD bit, when set, provides extra protection against **any** Debug Register access even in Real Mode or at privilege level 0 in Protected Mode. This additional protection feature is provided to guarantee that a software debugger can have full control over the Debug Register resources when required. The GD bit, when set, causes an exception 1 fault if an instruction attempts to read or write any Debug Register. The GD bit is then automatically cleared when the exception 1 handler is invoked, allowing the exception 1 handler free access to the debug registers.

GE and LE (Exact data breakpoint match, global and local)

The breakpoint mechanism of the Intel486 processor differs from that of the Intel386 processor. The Intel486 processor always does exact data breakpoint matching, regardless of GE/LE bit settings. Any data breakpoint trap will be reported exactly after completion of the instruction that caused the operand transfer. Exact reporting is provided by forcing the Intel486 processor execution unit to wait for completion of data operand transfers before beginning execution of the next instruction.

When the Intel486 processor performs a task switch, the LE bit is cleared. Thus, the LE bit supports fast task switching out of tasks, that have enabled the exact data breakpoint match for their task-local breakpoints. The LE bit is cleared by the Intel486 processor during a task switch, to avoid having exact data breakpoint match enabled in the new task. Note that exact data breakpoint match must be re-enabled under software control.

The Intel486 processor GE bit is unaffected during a task switch. The GE bit supports exact data breakpoint match that is to remain enabled during all tasks executing in the system.

Note that **instruction execution** breakpoints are always reported exactly.

Gi and Li (breakpoint enable, global and local)

If either Gi or Li is set then the associated breakpoint (as defined by the linear address in DRi, the length in LENi and the usage criteria in RWi) is enabled. If either Gi or Li is set, and the Intel486 processor detects the ith breakpoint condition, then the exception 1 handler is invoked.

When the Intel486 processor performs a task switch to a new Task State Segment (TSS), all Li bits are cleared. Thus, the Li bits support fast task switching out of tasks that use some task-local breakpoint registers. The Li bits are cleared by the Intel486 processor during a task switch, to avoid spurious exceptions in the new task. Note that the breakpoints must be re-enabled under software control.

All Intel486 processor Gi bits are unaffected during a task switch. The Gi bits support breakpoints that are active in all tasks executing in the system.

12.3.3 DEBUG STATUS REGISTER (DR6)

A Debug Status Register, DR6 shown in Figure 12-1, allows the exception 1 handler to easily determine why it was invoked. Note the exception 1 handler can be invoked as a result of one of several events:

1. DR0 Breakpoint fault/trap.
2. DR1 Breakpoint fault/trap.
3. XDR2 Breakpoint fault/trap.
4. XDR3 Breakpoint fault/trap.
5. XSingle-step (TF) trap.
6. XTask switch trap.
7. XFault due to attempted debug register access when GD=1.

The Debug Status Register contains single-bit flags for each of the possible events invoking exception 1. Note below that some of these events are faults (exception taken before the instruction is executed), while other events are traps (exception taken after the debug events occurred).

The flags in DR6 are set by the hardware but never cleared by hardware. Exception 1 handler software should clear DR6 before returning to the user program to avoid future confusion in identifying the source of exception 1.

The fields within the Debug Status Register, DR6, are as follows:

Bi (debug fault/trap due to breakpoint 0–3)

Four breakpoint indicator flags, B0–B3, correspond one-to-one with the breakpoint registers in DR0–DR3. A flag Bi is set when the condition described by DRi, LENi, and RWi occurs.

If Gi or Li is set, and if the ith breakpoint is detected, the Intel486 processor will invoke the exception 1 handler. The exception is handled as a fault if an instruction execution breakpoint occurred, or as a trap if a data breakpoint occurred.

IMPORTANT NOTE:

A flag Bi is set whenever the hardware detects a match condition on **enabled** breakpoint i. Whenever a match is detected on at least one **enabled** breakpoint i, the hardware immediately sets all Bi bits corresponding to breakpoint conditions matching at that instant, whether enabled **or not.** Therefore, the exception 1 handler may see that multiple Bi bits are set, but only set Bi bits corresponding to **enabled** breakpoints (Li or Gi set) are **true** indications of why the exception 1 handler was invoked.

BD (debug fault due to attempted register access when GD bit set)

This bit is set if the exception 1 handler was invoked due to an instruction attempting to read or write to the debug registers when GD bit was set. If such an event occurs, then the GD bit is automatically cleared when the exception 1 handler is invoked, allowing handler access to the debug registers.

BS (debug trap due to single-step)

This bit is set if the exception 1 handler was invoked due to the TF bit in the flag register being set (for single-stepping).

BT (debug trap due to task switch)

This bit is set if the exception 1 handler was invoked due to a task switch occurring to a task having an Intel486 processor TSS with the T bit set. Note the task switch into the new task occurs normally, but before the first instruction of the task is executed, the exception 1 handler is invoked. With respect to the task switch operation, the operation is considered to be a trap.

12.3.4 USE OF RESUME FLAG (RF) IN FLAG REGISTER

The Resume Flag (RF) in the flag word can suppress an instruction execution breakpoint when the exception 1 handler returns to a user program at a user address which is also an instruction execution breakpoint.

13.0 INSTRUCTION SET SUMMARY

This section describes the Intel486 processor instruction set. Detailed information on the CPUID instruction can be found in Appendix B: Feature Determination. Further details of the instruction encoding are then provided in section 13.1, which describes the entire encoding structure and the definition of all fields occurring within the Intel486 processor instructions.

13.1 Instruction Encoding

13.1.1 OVERVIEW

All instruction encodings are subsets of the general instruction format shown in Figure 13-1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the "mod r/m" byte and "scaled index" byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16 or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure 13-1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table 13-1 is a complete list of all fields appearing in the Intel486 processor instruction set. Following Table 13-1 are detailed tables for each field.

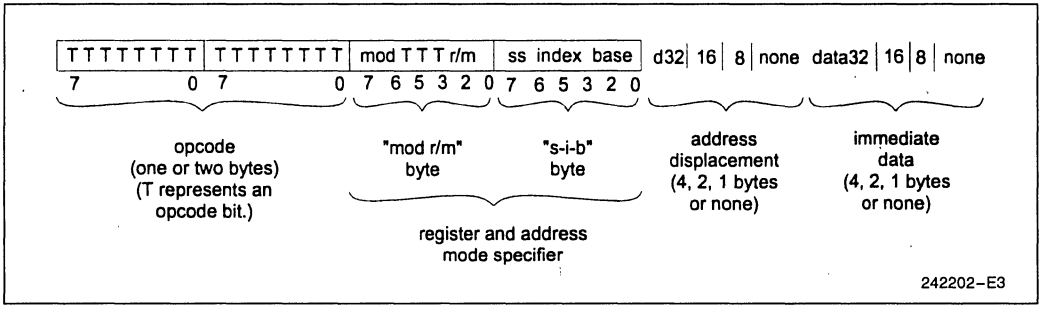


Figure 13-1. General Instruction Format

Table 13-1. Fields within Intel486™ Processor Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 Bits)	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field Must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, Specifies a Condition Asserted or a Condition Negated	4

NOTE:
Table 13-15 through Table 13-19 show encoding of individual instructions.

13.1.2 32-BIT EXTENSIONS OF THE INSTRUCTION SET

With the Intel486 processor, the 8086/80186/80286 instruction set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types, and 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

Whether the instruction defaults to operations of 16 bits or 32 bits depends on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 bits or 16 bits) for both operands and effective addresses when executing that code segment. In the Real Address Mode or Virtual 8086 Mode, no code segment descriptors are used, but a D value of 0 is assumed internally by the Intel486 processor when operating in those modes (for 16-bit default sizes compatible with the 8086/80186/80286).

Two prefixes, the Operand Size Prefix and the Effective Address Size Prefix, allow overriding individually the Default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the opcode bytes. The presence of the Operand Size Prefix and the Effective Address Prefix will toggle the operand size or the effective address size, respectively, to the value "opposite" from the Default setting. For example, if the default operand size is for 32-bit data operations, then presence of the Operand Size Prefix toggles the instruction to 16-bit data operation. As another example, if the default effective address size is 16 bits, presence of the Effective Address Size prefix toggles the instruction to use 32-bit effective address computations.

These 32-bit extensions are available in all Intel486 processor modes, including the Real Address Mode or the Virtual 8086 Mode. In these modes the default is always 16 bits, so prefixes are needed to specify 32-bit operands or addresses. For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

13.1.3 ENCODING OF INTEGER INSTRUCTION FIELDS

Within the instruction are several fields indicating register selection, addressing mode and so on. The exact encodings of these fields are defined in the following paragraphs.

13.1.3.1 Encoding of Operand Length (w) Field

For any given instruction performing a data operation, the instruction is executing as a 32-bit operation or a 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

Table 13-2. Encoding of Operand Length (w) Field

w Field	Operand Size during 16-Bit Data Operations	Operand Size during 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

13.1.3.2 Encoding of the General Register (reg) Field

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

Table 13-3. Encoding of reg Field when the w Field Is Not Present in Instruction

reg Field	Register Selected during 16-Bit Data Operations	Register Selected during 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI

Table 13-4. Encoding of reg Field when the w Field Is Present in Instruction

Register Specified by reg Field during 16-Bit Data Operations:		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI
Register Specified by reg Field during 32-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

13.1.3.3 Encoding of the Segment Register (sreg) Field

The sreg field in certain instructions is a 2-bit field allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the Intel486 processor FS and GS segment registers to be specified.

Table 13-5. 2-Bit sreg2 Field

2-bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

Table 13-6. 3-Bit sreg3 Field

3-bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

13.1.3.4 Encoding of Address Mode

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte, can be specified.

The s-i-b byte (scale-index-base byte) is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01 or 10. When the sib byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure 13-1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

Tables 13-7, 13-8, and 13-9 define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Table 13-7. Encoding of 16-Bit Address Mode with "mod r/m" Byte

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	SS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]

mod r/m	Effective Address
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP + DI + d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by r/m during 16-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m during 32-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Table 13-8. Encoding of 32-Bit Address Mode with "mod r/m" Byte (No "s-i-b" Byte Present)

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]

mod r/m	Effective Address
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	register—see below
11 001	register—see below
11 010	register—see below
11 011	register—see below
11 100	register—see below
11 101	register—see below
11 110	register—see below
11 111	register—see below

Register Specified by reg or r/m during 16-Bit Data Operations:

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m during 32-Bit Data Operations:

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Table 13-9. Encoding of 32-Bit Address Mode ("mod r/m" Byte and "s-i-b" Byte Present)

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

NOTE:
Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8
Index	**Index Register**
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg**
101	EBP
110	ESI
111	EDI

****IMPORTANT NOTE:**
When index field is 100, indicating "no index register," then ss field MUST equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

13.1.3.5 Encoding of Operation Direction (d) Field

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

Table 13-10. Encoding of Operation Direction (d) Field

d	Direction of Operation
0	Register/Memory ← Register "reg" Field Indicates Source Operand; "mod r/m" or "mod ss index base" Indicates Destination Operand
1	Register ← Register/Memory "reg" Field Indicates Destination Operand; "mod r/m" or "mod ss index base" Indicates Source Operand

13.1.3.6 Encoding of Sign-Extend (s) Field

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

Table 13-11. Encoding of Sign-Extend (s) Field

S	Effect on Immediate Data 8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extend Data 8 to Fill 16-bit or 32-bit Destination	None

13.1.3.7 Encoding of Conditional Test (tttn) Field

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n = 0) or its negation (n = 1), and ttt giving the condition to test.

Table 13-12. Encoding of Conditional Test (tttn) Field

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111

13.1.3.8 Encoding of Control or Debug or Test Register (eee) Field

For the loading and storing of the Control, Debug and Test registers.

Table 13-13. Encoding of Control or Debug or Test Register (eee) Field

eee Code	Reg Name
When Interpreted as Control Register Field:	
000	CR0
010	CR2
011	CR3
When Interpreted as Debug Register Field:	
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
When Interpreted as Test Register Field:	
011	TR3
100	TR4
101	TR5
110	TR6
111	TR7

Do not use any other encoding

Table 13-14. Encoding of Floating-Point Instruction Fields

	Instruction									Optional	
	First Byte				Second Byte					Fields	
1	11011	OPA		1	mod		1	OPB	r/m	s-i-b	disp
2	11011	MF		OPA	mod		OPB		r/m	s-i-b	disp
3	11011	d	P	OPA	1	1	OPB		ST(i)		
4	11011	0	0	1	1	1	1	OP			
5	11011	0	1	1	1	1	1	OP			
	15–11	10	9	8	7	6	5	4 3 2 1 0			

13.1.4 ENCODING OF FLOATING-POINT INSTRUCTION FIELDS

Instructions for the FPU assume one of the five forms shown in the following table. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B.

OP = Instruction opcode, possible split into two fields OPA and OPB

MF = Memory Format
00–32-bit real
01–32-bit integer
10–64-bit real
11–16-bit integer

P = Pop
0–Do not pop stack
1–Pop stack after operation

d = Destination
0–Destination is ST(0)
1–Destination is ST(i)

R XOR d = 0–Destination (op) Source

R XOR d = 1–Source (op) Destination

ST(i) = Register stack element *i*
000 = Stack top
001 = Second stack element
111 = Eighth stack element

mod (Mode field) and r/m (Register/Memory specifier) have the same interpretation as the corresponding fields of the integer instructions.

s-i-b (Scale Index Base) byte and disp (displacement) are optionally present in instructions that have mod and r/m fields. Their presence depends on the values of mod and r/m, as for integer instructions.

13.2 Clock Count Summary

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 13-15 through Table 13-19 by the processor core clock period (e.g., 10 ns for a 100-MHz IntelDX4 processor).

13.2.1 INSTRUCTION CLOCK COUNT ASSUMPTIONS

The Intel486 processor instruction core clock count tables give clock counts assuming data and instruction accesses hit in the cache. The combined instruction and data cache hit rate is over 90%.

A cache miss will force the Intel486 processor to run an external bus cycle. The Intel486 processor 32-bit burst bus is defined as r-b-w.

Where:

r = The number of bus clocks in the first cycle of a burst read or the number of clocks per data cycle in a non-burst read.

b = The number of bus clocks for the second and subsequent cycles in a burst read.

w = The number of bus clocks for a write.

The clock counts in the cache miss penalty column assume a 2-1-2 bus. For slower buses add r-2 clocks to the cache miss penalty for the first dword accessed. Other factors also affect instruction clock counts.

Instruction Clock Count Assumptions

1. The external bus is available for reads or writes at all times. Else add bus clocks to reads until the bus is available.
2. Accesses are aligned. Add three core clocks to each misaligned access.
3. Cache fills complete before subsequent accesses to the same line. If a read misses the cache during a cache fill due to a previous read or prefetch, the read must wait for the cache fill to complete. If a read or write accesses a cache line still being filled, it must wait for the fill to complete.
4. If an effective address is calculated, the base register is not the destination register of the preceding instruction. If the base register is the destination register of the preceding instruction add 1 to the core clock counts shown. Back-to-back PUSH and POP instructions are not affected by this rule.
5. An effective address calculation uses one base register and does not use an index register. However, if the effective address calculation uses an index register, 1 core clock **may** be added to the clock count shown.
6. The target of a jump is in the cache. If not, add r clocks for accessing the destination instruction of a jump. If the destination instruction is not completely contained in the first dword read, add a maximum of 3b bus clocks. If the destination instruction is not completely contained in the first 16 byte burst, add a maximum of another r+3b bus clocks.
7. If no write buffer delay, w bus clocks are added only in the case in which all write buffers are full.
8. Displacement and immediate not used together. If displacement and immediate used together, 1 core clock **may** be added to the core clock count shown.
9. No invalidate cycles. Add a delay of 1 bus clock for each invalidate cycle if the invalidate cycle contends for the internal cache/external bus when the Intel486 processor needs to use it.
10. Page translation hits in TLB. A TLB miss will add 13, 21 or 28 bus clocks + 1 possible core clock to the instruction depending on whether the Accessed and/or Dirty bit in neither, one or both of the page entries needs to be set in memory. This assumes that neither page entry is in the data cache and a page fault does not occur on the address translation.
11. No exceptions are detected during instruction execution. Refer to Interrupt core Clock Counts Table for extra clocks if an interrupt is detected.
12. Instructions that read multiple consecutive data items (i.e. task switch, POPA, etc.) and miss the cache are assumed to start the first access on a 16-byte boundary. If not, an extra cache line fill may be necessary which may add up to (r+3b) bus clocks to the cache miss penalty.

Table 13-15. Clock Count Summary

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
INTEGER OPERATIONS				
MOV = Move:				
reg1 to reg2	1000 100w : 11 reg1 reg2	1		
reg2 to reg1	1000 101w : 11 reg1 reg2	1		
memory to reg	1000 100w : mod reg r/m	1	2	
Immediate to reg	1100 011w : 11000 reg : immediate data	1		
or	1011W reg : immediate data	1		
Immediate to Memory	1100 01w : mod 000 r/m : displacement immediate	1		
Memory to Accumulator	1010 000w : full displacement	1	2	
Accumulator to Memory	1010 001w : full displacement	1		
MOVSX/MOVZX = Move with Sign/Zero Extension				
reg2 to reg1	0000 1111 : 1011 z11w : 11 reg1 reg2	3		
memory to reg	0000 1111 : 1011 z11w : mod reg r/m	3	2	
z instruction 0 MOVZX 1 MOVSX				
PUSH = Push				
reg	1111 1111 : 11 110 reg	4		
or	01010 reg	1		
memory	1111 1111 : mod 110 r/m	4	1	1
immediate	0110 10s0 : immediate data	1		
PUSHA = Push All	0110 0000	11		
POP = Pop				
reg	1000 1111 : 11 000 reg	4	1	
or	01011 reg	1	2	
memory	1000 1111 : mod 000 r/m	5	2	1
POPA = Pop All	0110 0001	9	7/15	16/32
XCHG = Exchange				
reg1 with reg2	1000 011w : 11 reg1 reg2	3		2
Accumulator with reg	10010 reg	3		2
Memory with reg	1000 011w : mod reg r/m	5		2

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
INTEGER OPERATIONS (Continued)				
NOP = No Operation	1001 0000	1		
LEA = Load EA to Register	1000 1101 : mod reg r/m			
no index register		1		
with index register		2		
Instruction	TTT			
ADD = Add	000			
ADC = Add with Carry	010			
AND = Logical AND	100			
OR = Logical OR	001			
SUB = Subtract	101			
SBB = Subtract with Borrow	011			
XOR = Logical Exclusive OR	110			
reg1 to reg2	00TT T00w : 11 reg1 reg2	1		
reg2 to reg1	00TT T01w : 11 reg1 reg2	1		
memory to register	00TT T01w : mod reg r/m	2	2	
register to memory	00TT T00w : mod reg r/m	3	6/2	U/L
immediate to register	1000 00sw : 11 TTT reg : immediate register	1		
immediate to Accumulator	00TT T10w : immediate data	1		
immediate to memory	1000 00sw : mod TTT r/m : immediate data	3	6/2	U/L
Instruction	TTT			
INC = Increment	000			
DEC = Decrement	001			
reg	1111 111w : 11 TTT reg	1		
or	01TTT reg	1		
memory	1111 111w : mod TTT r/m	3	6/2	U/L
Instruction	TTT			
NOT = Logical Complement	010			
NEG = Negate	011			
reg	1111 011w : 11 TTT reg	1		
memory	1111 011w : mod TTT r/m	3	6/2	U/L

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
INTEGER OPERATIONS (Continued)				
CMP = Compare				
reg1 with reg2	0011 100w : 11 reg1 reg2	1		
reg2 with reg1	0011 101w : 11 reg1 reg2	1		
memory with register	0011 100w : mod reg r/m	2	2	
register with memory	0011 101w : mod reg r/m	2	2	
immediate with register	1000 00sw : 11 111 reg : immediate data	1		
immediate with acc.	0011 110w : immediate data	1		
immediate with memory	1000 00sw : mod 111 r/m : immediate data	2	2	
TEST = Logical Compare				
reg1 and reg2	1000 010w : 11 reg1 reg2	1		
memory and register	1000 010w : mod reg r/m	2	2	
immediate and register	1111 011w : 11 000 reg : immediate data	1		
immediate and acc.	1010100w : immediate data	1		
immediate and memory	1111 011w : mod 000 r/m : immediate data	2	2	
MUL = Multiply (unsigned)				
acc. with register	1111 011w : 11 100 reg			
Multiplier-Byte		13/18		MN/MX,3
Word		13/26		MN/MX,3
Dword		13/42		MN/MX,3
acc. with memory	1111 011w : mod 100 r/m			
Multiplier-Byte		13/18	1	MN/MX,3
Word		13/26	1	MN/MX,3
Dword		13/42	1	MN/MX,3
IMUL = Integer Multiply (unsigned)				
acc. with register	1111 011w : 11 101 reg			
Multiplier-Byte		13/18		MN/MX,3
Word		13/26		MN/MX,3
Dword		13/42		MN/MX,3
acc. with memory	1111 011w : mod 101 r/m			
Multiplier-Byte		13/18		MN/MX,3
Word		13/26		MN/MX,3
Dword		13/42		MN/MX,3
reg1 with reg2	0000 1111 : 10101111 : 11 reg1 reg2			
Multiplier-Byte		13/18		MN/MX,3
Word		13/26		MN/MX,3
Dword		13/42		MN/MX,3

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
INTEGER OPERATIONS (Continued)				
IMUL = Integer Multiply (unsigned), (Continued)				
register with memory	0000 1111 : 10101111 : mod reg r/m			
Multiplier-Byte		13/18	1	MN/MX,3
Word		13/26	1	MN/MX,3
Dword		13/42	1	MN/MX,3
reg1 with imm. to reg2	0110 10s1 : 11 reg1 reg2 : immediate data			
Multiplier-Byte		13/18		MN/MX,3
Word		13/26		MN/MX,3
Dword		13/42		MN/MX,3
mem. with imm. to reg.	0110 10s1 : mod reg r/m : immediate data			
Multiplier-Byte		13/18		MN/MX,3
Word		13/26		MN/MX,3
Dword		13/42		MN/MX,3
For the IntelDX4™ Processor Only:				
IMUL = Integer Multiply (signed)				
acc. with register	1111 011w : 11101 reg			
Multiplier-Byte		5/5		MN/MX,3
Word		5/6		MN/MX,3
Dword		6/12		MN/MX,3
acc. with memory	1111 011w : mod 101 r/m			
Multiplier-Byte		5/5		MN/MX,3
Word		5/6		MN/MX,3
Dword		6/12		MN/MX,3
reg1 with reg2	0000 1111 : 10101111 : 11 reg1 reg2			
Multiplier-Byte		5/5		MN/MX,3
Word		5/6		MN/MX,3
Dword		6/12		MN/MX,3
register with memory	0000 1111 : 10101111 : mod reg r/m			
Multiplier-Byte		5/5		MN/MX,3
Word		5/6		MN/MX,3
Dword		6/12		MN/MX,3
reg1 with imm. to reg2	0110 10s1 : 11 reg1 reg2 : immediate data			
Multiplier-Byte		5/5		MN/MX,3
Word		5/6		MN/MX,3
Dword		6/12		MN/MX,3
mem. with imm. to reg.	0110 10s1 : mod reg r/m : immediate data			
Multiplier-Byte		5/5		MN/MX,3
Word		5/6		MN/MX,3
Dword		6/12		MN/MX,3

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
INTEGER OPERATIONS (Continued)				
DIV = Divide (unsigned)				
acc. by register	1111 011w : 11110 reg			
Divisor-Byte		16		
Word		24		
Dword		40		
acc. by memory	1111 011w : mod 110 r/m			
Divisor-Byte		16		
Word		24		
Dword		40		
IDIV = Integer Divide (signed)				
acc. by register	1111 011w : 11111 reg			
Divisor-Byte		19		
Word		27		
Dword		43		
acc. by memory	1111 011w : mod 111 r/m			
Divisor-Byte		20		
Word		28		
Dword		44		
CBW = Convert Byte to Word	1001 1000	3		
CWD = Convert Word to Dword	1001 1001	3		
Instruction / TTT				
ROL = Rotate Left	000			
ROR = Rotate Right	001			
RCL = Rotate Through Carry Left	010			
RDR = Rotate Through Carry Right	011			
SHL/SAL = Shift Logical/ Arithmetic Left	100			
SHR = Shift Logical Right	101			
SAR = Shift Arithmetic Right	111			
Not Through Carry (ROL, ROR, SAR, SHL, and SHR)				
reg by 1	1101 000w : 11 TTT reg	3		
memory by 1	1101 000w : mod TTT r/m	4	6	
reg by CL	1101 001w : 11 TTT reg	3		
memory by CL	1101 001w : mod TTT r/m	4	6	
reg by immediate count	1100 000w : 11 TTT reg : imm. 8-bit data	2		
mem by immediate count	1100 000w : mod TTT r/m : imm. 8-bit data	4	6	

4

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
INTEGER OPERATIONS (Continued)				
Through Carry (RCL and RCR)				
reg by 1	1101 000w : 11 TTT reg	3		
memory by 1	1101 000w : mod TTT r/m	4	6	
reg by CL	1101 001w : 11 TTT reg	8/30		MN/MX,4
memory by CL	1101 001w : mod TTT r/m	9/31		MN/MX,5
reg by immediate count	1100 000w : 11 TTT reg : imm. 8-bit data	8/30		MN/MX,4
mem by immediate count	1100 000w : mod TTT r/m : imm. 8-bit data	9/31		MN/MX,5
Instruction SHLD = Shift Left Double SHRD = Shift Right Double	TTT 100 101			
register with immediate	0000 1111 : 10TT T100 : 11 reg2 reg1 : imm. 8-bit data	2		
memory with immediate	0000 1111 : 10TT T100 : mod reg r/m : imm. 8-bit data	3	6	
register by CL	0000 1111 : 10TT T101 : 11 reg2 reg1	3		
memory by CL	0000 1111 : 10TT T101 : mod reg r/m	4	5	
BSWAP = Byte Swap	0000 1111 : 11001 reg	1		
XADD = Exchange and Add				
reg1, reg2	0000 1111 : 1100 000w : 11 reg2 reg1	3		
memory, reg	0000 1111 : 1100 000w : mod reg r/m	4	6/2	U/L
CMPXCHG = Compare and Exchange				
reg1, reg2	0000 1111 : 1011 000w : 11 reg2 reg1	6		
memory, reg	0000 1111 : 1011 000w : mod reg r/m	7/10	2	6
CONTROL TRANSFER (within segment)				
Note: Times are jump taken/not taken **J_{cccc} = Jump on cccc**				
8-bit displacement	0111 tttn : 8-bit disp.	3/1		T/NT,23
full displacement	0000 1111 : 1000 tttn : full displacement	3/1		T/NT,23
Note: Times are jump taken/not taken **SET_{cccc} = Set Byte on cccc (Times are cccc true/false)**				
reg	0000 1111 : 1001 tttn : 11 000 reg	4/3		
memory	0000 1111 : 1001 tttn : mod 0000 r/m	3/4		

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
CONTROL TRANSFER (within segment) (Continued)				
Mnemonic cccc / Condition / tttn				
O — Overflow — 0000				
NO — No Overflow — 0001				
B/NAE — Below/Not Above or Equal — 0010				
NB/AE — Not Below/Above or Equal — 0011				
E/Z — Equal Zero — 0100				
NE/NZ — Not Equal/Not Zero — 0101				
BE/NA — Below or Equal/Not Above — 0110				
NBE/A — Not Below or Equal/Above — 0111				
S — Sign — 1000				
NS — Not Sign — 1001				
P/PE — Parity/Parity Even — 1010				
NP/PO — Not Parity/Parity Odd — 1011				
L/NGE — Less Than/Not Greater or Equal — 1100				
NL/GE — Not Less Than/Greater or Equal — 1101				
LE/NG — Less Than or Equal/Greater Than — 1110				
NLE/G — Not Less Than or Equal/Greater Than — 1111				
LOOP = LOOP CX Times	1110 0010 : 8-bit disp.	7/6		L/NL,23
LOOPZ/LOOPE = Loop with Zero/Equal	1110 0001 : 8-bit disp.	9/6		L/NL,23
LOOPNZ/LOOPNE = Loop While Not Zero	1110 0000 : 8-bit disp.	9/6		L/NL,23
JCXZ = Jump on CX Zero	1110 0011 : 8-bit disp.	8/5		T/NT,23
JECXZ = Jump on ECX Zero (Address Size Prefix Differentiates JCXZ for JECXZ)	1110 0011 : 8-bit disp.	8/5		T/NT,23
JMP = Unconditional Jump (within segment)				
Short	1110 1011 : 8-bit disp.	3		7,23
Direct	1110 1001 : full displacement	3		7,23
Register Indirect	1111 1111 : 11 100 reg	5		7,23
Memory Indirect	1111 1111 : mod 100 r/m	5	5	7
CALL = Call (within segment)				
Direct	1110 1000 : full displacement	3		7,23
Register Indirect	1111 1111 : 11 010 reg	5		7,23
Memory Indirect	1111 1111 : mod 010 reg	5	5	7
RET = Return from CALL (within segment)	1100 0011	5	5	
Adding Immediate to SP	1100 0010 : 16-bit disp.	5	5	
ENTER = Enter Procedure	1100 1000 : 16-bit disp., 8-bit level			
Level = 0		14		
Level = 1		17		
Level (L) > 1		17+3L		8
LEAVE = Leave Procedure	1100 1001	5	1	

4

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
MULTIPLE-SEGMENT INSTRUCTIONS				
MOV = Move				
reg. to segment reg.	1000 1110 : 11 sreg3 reg	3/9	0/3	RV/P,9
memory to segment reg.	1000 1110 : mod sreg3 r/m	3/9	2/5	RV/P,9
segment reg. to reg.	1000 1100 : 11 sreg3 reg	3		
segment reg. to memory	1000 1100 : mod sreg3 r/m	3		
PUSH = Push				
segment reg. (ES, CS, SS, or DS)	000sreg 2110	3		
segment reg. (FS or GS)	0000 1111 : 10 sreg3001	3		
POP = Pop				
segment reg. (ES, CS, SS, or DS)	000sreg 2111	3/0	2/5	RV/P,9
segment reg. (FS or GS)	0000 1111 : 10 sreg3001	3/9	2/5	RV/P,9
LDS = Load Pointer to DS	1100 0101 : mod reg r/m	6/12	7/10	RV/P,9
LES = Load Pointer to ES	1100 0100 : mod reg r/m	6/12	7/10	RV/P,9
LFS = Load Pointer to FS	0000 1111 : 1011 0100 : mod reg r/m	6/12	7/10	RV/P,9
LGS = Load Pointer to GS	0000 1111 : 1011 0101 : mod reg r/m	6/12	7/10	RV/P,9
LSS = Load Pointer to SS	0000 1111 : 1011 0010 : mod reg r/m	6/12	7/10	RV/P,9
CALL = Call				
Direct intersegment	1001 1010 : unsigned full offset, selector	18	2	R,7,22
to same level		20	3	P,9
thru Gate to same level		35	6	P,9
to inner level, no parameters		69	17	P,9
to inner level, x parameters (d) words		77+4X	17+n	P,11,9
to TSS		37+TS	3	P,10,9
thru Task Gate		38+TS	3	P,10,9
Indirect intersegment	1111 1111 : mod 011 r/m	17	8	R,7
to same level		20	10	P,9
thru Gate to same level		35	13	P,9
to inner level, no parameters		69	24	P,9
to inner level, x parameters (d) words		77+4X	24+n	P,11,9
to TSS		37+TS	10	P,10,9
thru Task Gate		38+TS	10	P,10,9
RET = Return from CALL				
intersegment	1100 1010	13	8	R,7
to same level		17	9	P,9
to outet lever		35	12	P,9
intersegment adding imm. to SP	1100 1010 : 16-bit disp.	14	8	R,7
to same level		18	9	P,9
to outer level		36	12	P,9

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
MULTIPLE-SEGMENT INSTRUCTIONS (Continued)				
JMP = Unconditional Jump				
Direct intersegment	1110 1010 : unsigned full offset, selector	17	2	R,7,22
to same level		19	3	P,9
thru Call Gate to same level		32	6	P,9
thru TSS		42+TS	3	P,10,9
thru Task Gate		43+TS	3	P,10,9
Indirect intersegment	1111 1111 : mod 011 r/m	13	9	R,7,9
to same level		18	10	P,9
thru Call Gate to same level		31	13	P,9
thru TSS		41+TS	10	P,10,9
thru Task Gate		42+TS	10	P,10,9
BIT MANIPULATION				
BT = Test Bit				
register, immediate	0000 1111 : 1011 1010 : 11 100 reg : imm. 8-bit data	3		
memory, immediate	0000 1111 : 1011 1010 : mod 100 r/m : imm. 8-bit data	3	1	
reg1, reg2	0000 1111 : 1010 0011 : 11 reg2 reg1	3		
memory, reg	0000 1111 : 1010 0011 : mod reg r/m	8	2	
Instruction / TTT BTS = Test Bit and Set / 101 BTR = Test Bit and Reset / 110 BTC = Test Bit and Compliment / 111				
register, immediate	0000 1111 : 1011 1010 : 11 TTT reg imm. 8-bit data	6		
memory, immediate	0000 1111 : 1011 1010 : mod TTT r/m imm. 8-bit data	8		U/L
reg1, reg2	0000 1111 : 10TT T011 : 1 1 reg2 reg1	6		
memory, reg	0000 1111 : 10TT T011 : mod reg r/m	13		U/L
BSF = Scan Bit Forward				
reg1, reg2	0000 1111 : 1011 1100 : 11 reg2 reg1	6/42		MN/MX, 12
memory, reg	0000 1111 : 1011 1100 : mod reg r/m	7/43	2	MN/MX, 15
BSR = Scan Bit Reverse				
reg1, reg2	0000 1111 : 1011 1101 : 11 reg2 reg1	6/103		MN/MX, 14
memory, reg	0000 1111 : 1011 1101 : mod reg r/m	7/104	1	MN/MX, 15

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
STRING INSTRUCTIONS				
CMPS = Compare Byte Word	1010 011w	8	6	16
LODS = Load Byte/Word to AL/AX/EAX	1010 111w	5	2	
MOVS = Move Byte/Word	1010 010w	7	2	16
SCAS = Scan Byte/Word	1010 111w	6	2	
STOS = Store Byte/Word from AL/AX/EX	1010 101w	5		
XLAT = Translate String	1101 0111	4	2	
REPEATED STRING INSTRUCTIONS Repeated by Count in CX or ECX (C = Count in CX or ECX)				
REPE CMPS = Compare String (Find Non-match)	1111 0011 : 1010 011w			
C = 0		5		16, 17
C > 0		7+7c		
REPNE CMPS = Compare String (Find Match)	1111 0010 : 1010 011w			
C = 0		5		16, 17
C > 0		7+7c		
REP LODS = Load String	1111 0010 : 1010 110w			
C = 0		5		16, 18
C > 0		7+4c		
REP MOVS = Move String	1111 0010 : 1010 010w			
C = 0		5	1	16
C = 1		13		16, 19
C > 1		12+3c		
REPE SCAS = Scan String (Find Non-AL/AX/EAX)	1111 0011 : 1010 111w			
C = 0		5		
C > 0		7+5c		20
REPNE SCAS = Scan String (Find AL/AX/EAX)	1111 0010 : 1010 111w			
C = 0		5		
C > 0		7+5c		20
REP STOS = Store String	1111 0010 : 1010 101w			
C = 0		5		
C > 0		7+4c		

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
FLAG CONTROL				
CLC = Clear Carry Flag	1111 1000	2		
STC = Set Carry Flag	1111 1001	2		
CMC = Complement Carry Flag	1111 0101	2		
CLD = Clear Direction Flag	1111 1100	2		
STD = Set Direction Flag	1111 1101	2		
CLI = Clear Interrupt Enable Flag	1111 1010	5		
STI = Set Interrupt Enable Flag	1111 1011	5		
LAHF = Load AH into Flag	1001 1111	3		
SAHF = Store AH into Flag	1001 1110	2		
PUSHF = Push Flags	1001 1100	4/3		RV/P
POFF = Pop Flags	1001 1101	9/6		RV/P
DECIMAL ARITHMETIC				
AAA = ASCII Adjust to Add	0011 0111	3		
AAS = ASCII Adjust for Subtract	0011 1111	3		
AAM = ASCII Adjust for Multiply	1101 0100 : 0000 1010	15		
AAD = ASCII Adjust for Divide	1101 0101 : 0000 1010	14		
DAA = Decimal Adjust for Add	0010 0111	2		
DAS = Decimal Adjust for Subtract	0010 1111	2		
PROCESSOR CONTROL INSTRUCTIONS				
HLT = Halt	1111 0100	4		
MOV = Move To and From Control/Debug/Test Registers				
CR0 from register	0000 1111 : 0010 0010 : 11 000 reg	17	2	
CR2/CR3 from register	0000 1111 : 0010 0010 : 11 eee reg	4		
Reg from CR0-3	0000 1111 : 0010 0000 : 11 eee reg	4		
DR0-3 from register	0000 1111 : 0010 0011 : 11 eee reg	10		
DR6-7 from register	0000 1111 : 0010 0011 : 11 eee reg	10		
Register from DR6-7	0000 1111 : 0010 0001 : 11 eee reg	9		
Register from DR0-3	0000 1111 : 0010 0001 : 11 eee reg	9		
TR3 from register	0000 1111 : 0010 0110 : 11 011 reg	4		
TR4-7 from register	0000 1111 : 0010 0110 : 11 eee reg	4		
Register from TR3	0000 1111 : 0010 0100 : 11 011 reg	3		
Register from TR4-7	0000 1111 : 0010 0100 : 11 eee reg	4		

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
PROCESSOR CONTROL INSTRUCTIONS (Continued)				
CPUID = CPU Identification	0000 1111 : 1010 0010			
EAX = 1		14		
EAX = 0, >1		9		
CLTS = Clear Task Switched Flag	0000 1111 : 0000 0110	7	2	
INVD = Invalidate Data Cache	0000 1111 : 0000 1000	4		
WBINVD = Write-Back and Invalidate Data Cache	0000 1111 : 0000 1001	5		
INVLPG = Invalidate TLB Entry				
INVLPG memory	0000 1111 : 0000 0001 : mod 111 r/m	12/11		H/NH
PREFIX BYTES				
Address Size Prefix	0110 0111	1		
LOCK = Bus Lock Prefix	1111 0000	1		
Operand Size Prefix	0110 0110	1		
Segment Override Prefix				
CS:	0010 1110	1		
DS:	0011 1110	1		
ES:	0010 0110	1		
FS:	0110 0100	1		
GS:	0110 0101	1		
SS:	0011 0110	1		
PROTECTION CONTROL				
ARPL = Adjust Requested Privilege Level				
From register	0110 0011 : 11 reg1 reg2	9		
From memory	0110 0011 : mod reg r/m	9		
LAR = Load Access Rights				
From register	0000 1111 : 0000 0010 : 11 reg1 reg2	11	3	
From memory	0000 1111 : 0000 0010 : mod reg r/m	11	5	
LGDT = Load Global Descriptor				
Table register	0000 1111 : 0000 0001 : mod 010 r/m	12	5	
LIDT = Load Interrupt Descriptor				
Table register	0000 1111 : 0000 0001 : mod 011 r/m	12	5	
LLDT = Load Local Descriptor				
Table register from reg.	0000 1111 : 0000 0000 : 11 010 reg	11	3	
Table register from mem.	0000 1111 : 0000 0000 : mod 010 r/m	11	6	

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
PROTECTION CONTROL (Continued)				
LMSW = Load Machine Status Word				
From register	0000 1111 : 0000 0001 : 11 110 reg	13		
From memory	0000 1111 : 0000 0001 : mod 110 r/m	13	1	
LSL = Load Segment Limit				
From register	0000 1111 : 0000 0011 : 11 reg1 reg2	10	3	
From memory	0000 1111 : 0000 0011 : mod reg r/m	10	6	
LTR = Load Task Register				
From register	0000 1111 : 0000 0000 : 11 011 reg	20		
From memory	0000 1111 : 0000 0000 : mod 011 r/m	20		
SGDT = Store Global Descriptor Table	0000 1111 : 0000 0001 : mod 000 r/m	10		
SIDT = Store Interrupt Descriptor Table	0000 1111 : 0000 0001 : mod 001 r/m	2		
SLDT = Store Local Descriptor Table				
To register	0000 1111 : 0000 0000 : 11 000 reg	2		
To memory	0000 1111 : 0000 0001 : mod 000 r/m	3		
SMSW = Store Machine Status Word				
To register	0000 1111 : 0000 0001 : 11 000 reg	2		
To memory	0000 1111 : 0000 0001 : mod 100 r/m	3		
STR = Store Task Register				
To register	0000 1111 : 0000 0000 : 11 001 r/m	2		
To memory	0000 1111 : 0000 0000 : mod 001 r/m	3		
VERR = Verify Read Access				
Register	0000 1111 : 0000 0000 : 11 100 r/m	11	3	
Memory	0000 1111 : 0000 0000 : mod 100 r/m	11	7	
VERW = Verify Write Access				
To register	0000 1111 : 0000 0000 : 11 101 r/m	11	3	
To memory	0000 1111 : 0000 0000 : mod 101 r/m	11	7	
INTERRUPT INSTRUCTIONS				
INTn = Interrupt Type n	1100 1101 : type	INT+ 4/0		RV/P, 21
INT3 = Interrupt Type 3	1100 1100	INT+0		21

4

Table 13-15. Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
INTERRUPT INSTRUCTIONS (Continued)				
INTO = Interrupt 4 if Overflow Flag Set	1100 1110			
Taken		INT+2		21
Not Taken		3		21
BOUND = Interrupt 5 if Detect Value Out Range	0110 0010 : mod reg r/m			
If in range		7	7	21
If out of range		INT+ 24	7	21
IRET = Interrupt Return	1100 1111			
Real Mode/Virtual Mode		15	8	
Protected Mode				
To same level		20	11	9
To outer level		36	19	9
To nested task		TS+32	4	9,10
(EFLAGS.NT=1)				
RSM = Exit System Management Mode	0000 1111 : 1010 1010			
SMBASE Relocation		452		
Auto HALT Restart		456		
I/O Trap Restart		465		
External Interrupt		INT+ 11		21
NMI = Non-Maskable Interrupt		INT+3		21
Page Fault		INT+ 24		21
VM86 Exceptions				
CLI		INT+8		21
STI		INT+8		21
INTn		INT+9		
PUSHF		INT+9		21
POPF		INT+8		21
IRET		INT+9		
IN				
Fixed Port		INT+50		21
Variable Port		INT+51		21
OUT				
Fixed Port		INT+50		21
Variable Port		INT+51		21
INS		INT+50		21
OUTS		INT+50		21
REP INS		INT+51		21
REP OUTS		INT+51		21

Table 13-16. Task Switch Clock Counts

Method	Value for TS	
	Cache Hit	Miss Penalty
VM/Intel486™ Processor/286 TSS to Intel486 Processor TSS	162	55
VM/Intel486 Processor/286 TSS to 286 TSS	144	31
VM/Intel486 Processor/286 TSS to VM TSS	140	37

Table 13-17. Interrupt Clock Counts

Method	Value for INT		
	Cache Hit	Miss Penalty	Notes
Real Mode	26	2	
Protected Mode			
Interrupt/Trap gate, same level	44	6	9
Interrupt/Trap gate, different level	71	17	9
Task Gate	37 + TS	3	9, 10
Virtual Mode			
Interrupt/Trap gate, different level	82	17	
Task Gate	37 + TS	3	10

Abbreviations	Definition
16/32	16/32 bit modes
U/L	unlocked/locked
MN/MX	minimum/maximum
L/NL	loop/no loop
RV/P	real and virtual mode/protected mode
R	real mode
P	protected mode
T/NT	taken/not taken
H/NH	hit/no hit

NOTES (for Tables 13-17 through 13-19):

1. Assuming that the operand address and stack address fall in different cache sets.
2. Always locked, no cache hit case.
3. Clocks = 10 + max($\log_2(|m|)$,n)
4. Clocks = {qoutient(couint/operand length)}*7+9
   = 8 if count ≤ operand length (8/16/32)
5. Clocks = {qoutient(couint/operand length)}*7+9
   = 9 if count ≤ operand length (8/16/32)
6. Equal/not equal cases (penalty is the same regardless of lock)
7. Assuming that addresses for memory read (for indirection), stack puch/pop and branch fall in different cache sets.
8. Penalty for cache miss: add 6 clocks for every 16 bytes copied to new stack frame.
9. Add 11 clocks for each unaccessed descriptor load.
10. Refer to task switch clock counts table for value of TS.
11. Add 4 extra clocks to the cache miss penalty for each 16 bytes.

For notes 12-13: (b=0-3, non-zero byte number);
(i=0-1, non-zero nibble number);
(n=0-3, non-bit number in nibble);

12. Clocks = 8 + 4 (b+1) + 3(i+1) + 3(n+1)
= 6 if second operand = 0
13. Clocks = 9 + 4 (b+1) + 3(i+1) + 3(n+1)
= 7 if second operand = 0

For notes 14-15: (n=bit position 0-31)

14. Clocks = 7 + 3(32-n)
= 6 if second operand = 0
15. Clocks = 8 + 3(32-n)
= 7 if second operand = 0
16. Assuming that the two string addresses fall in different cache sets.
17. Cache miss penalty: add 6 clocks for every 16 bytes compared. Entire penalty on first compare.
18. Cache miss penalty: add 2 clocks for every 16 bytes of data. Entire penalty on first load.
19. Cache miss penalty: add 4 clocks for every 16 bytes moved. (1 clock for the first operation and 3 for the second)
20. Cache miss penalty: add 4 clocks for every 16 bytes scanned. (2 clocks each for first and second operations)
21. Refer to interrupt clock counts table for value of INT.
22. Clock count includes one clock for using both displacement and immediate.
23. Refer to assumption 6 in the case of a cache miss.
24. Virtual Mode Extensions are disabled.
25. Protected Virtual Interrupts are disabled.

Table 13-18. I/O Instructions Clock Count Summary

Instruction	Format	Real Mode	Protected Mode (CPL≤IOPL)	Protected Mode (CPL>IOPL)	Virtual 86 Mode	Notes
IN = Input from:						
Fixed Port	1110 010w : port number	14	9	29	27	
Variable Port	1110 110w	14	8	28	27	
OUT = Output to:						
Fixed Port	1110 011w : port number	16	11	31	29	
Variable Port	1110 110w	16	10	30	29	
INS = Input Byte/Word from DX Port	0110 110w	17	10	32	30	
OUTS = Output Byte/Word to DX Port	0110 111w	17	10	32	30	1
REP INS = Input String	1111 0010 : 0110 110w	16+8c	10+8c	30+8c	29+8c	2
REP OUTS = Output String	1111 0010 : 0110 111w	17+5c	11+5c	31+5c	30+5c	3

NOTES:
1. Two clock cache miss penalty in all cases.
2. c = count in CX or ECX.
3. Cache miss penalty in all modes: Add 2 clocks for every 16 bytes. Entire penalty on second operation.

Table 13-19. Floating-Point Clock Count Summary

Instruction	Format	Cache Hit	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
DATA TRANSFER					
FLD = Real Load to ST(0)					
32-bit memory	11011 001 : mod 000 r/m : s-i-b/disp.	3	2		
64-bit memory	11011 101 : mod 000 r/m : s-i-b/disp.	3	3		
80-bit memory	11011 011 : mod 101 r/m : s-i-b/disp.	6	4		
ST(i)	11011 001 : 11000 ST(i)	4			
FILD = Integer Load to ST(0)					
16-bit memory	11011 111 : mod 000 r/m : s-i-b/disp.	14.5(13-16)	2	4	
32-bit memory	11011 011 : mod 000 r/m : s-i-b/disp.	11.5(9-12)	2	4(2-4)	
64-bit memory	11011 111 : mod 101 r/m : s-i-b/disp.	16.8(10-18)	3	7.8(2-8)	
FBLD = BCD Load to ST(0)					
	11011 111 : mod 100 r/m : s-i-b/disp.	75(70-103)	4	7.7(2-8)	
FST = Store Real from ST(0)					
32-bit memory	11011 011 : mod 010 r/m : s-i-b/disp.	7			1
64-bit memory	11011 101 : mod 010 r/m : s-i-b/disp.	8			2
ST(i)	11011 101 : 11001 ST(i)	3			
FSTP = Store Real from ST(0) and Pop					
32-bit memory	11011 011 : mod 011 r/m : s-i-b/disp.	7			1
64-bit memory	11011 101 : mod 011 r/m : s-i-b/disp.	8			2
80-bit memory	11011 011 : mod 111 r/m : s-i-b/disp.	6			
ST(i)	11011 101 : 11001 ST(i)	3			
FIST = Store Integer from ST(0)					
16-bit memory	11011 111 : mod 010 r/m : s-i-b/disp.	33.4(29-34)			
32-bit memory	11011 011 : mod 010 r/m : s-i-b/disp.	32.4(28-34)			
FISTP = Store Integer from ST(0) and Pop					
16-bit memory	11011 111 : mod 011 r/m : s-i-b/disp.	33.4(29-34)			
32-bit memory	11011 011 : mod 011 r/m : s-i-b/disp.	33.4(29-34)			
64-bit memory	11011 111 : mod 111 r/m : s-i-b/disp.	33.4(29-34)			
FBSTP = Store BCD from ST(0) and Pop					
	11011 111 : mod 110 r/m : s-i-b/disp.	175(172-176)			
FXCH = Exchange ST(0) and ST(i)					
	11011 001 : 11001 ST(i)	4			

4

Table 13-19. Floating-Point Clock Count Summary (Continued)

Instruction	Format	Cache Hit: Avg (Lower Range ... Upper Range)	Penalty if Cache Miss	Concurrent Execution: Avg (Lower Range ... Upper Range)	Notes
COMPARISON INSTRUCTIONS					
FCOM = Compare ST(0) with Real					
32-bit memory	11011 000 : mod 010 r/m : s-i-b/disp.	4	2	1	
64-bit memory	11011 100 : mod 010 r/m : s-i-b/disp.	4	3	1	
ST(i)	11011 000 : 11010 ST(i)	4			
FCOMP = Compare ST(0) with Real and Pop					
32-bit memory	11011 000 : mod 011 r/m : s-i-b/disp.	4	2	1	
64-bit memory	11011 100 : mod 011 r/m : s-i-b/disp.	4	3	1	
ST(i)	11011 000 : 11011 ST(i)	4		1	
FCOMPP = Compare ST(0) with ST(1) and Pop Twice					
	11011 110 : 1101 1001	5		1	
FICOM = Compare ST(0) with Integer					
16-bit memory	11011 110 : mod 010 r/m : s-i-b/disp.	18(16-20)	2	1	
32-bit memory	11011 010 : mod 010 r/m : s-i-b/disp.	16.5(15-17)	2	1	
FICOMP = Compare ST(0) with Integer					
16-bit memory	11011 110 : mod 011 r/m : s-i-b/disp.	18(16-20)	2	1	
32-bit memory	11011 010 : mod 011 r/m : s-i-b/disp.	16.5(15-17)	2	1	
FTST = Compare ST(0) with 0.0					
	11011 011 : 1110 0100	4		1	
FUCOM = Unordered compare ST(0) with ST(i)					
	11011 101 : 11100 ST(i)	4		1	
FUCOMP = Unordered compare ST(0) with ST(i) and Pop					
	11011 101 : 11101 ST(i)	4		1	
FUCOMPP = Unordered compare ST(0) with ST(1) and Pop Twice					
	11011 101 : 11101 1001	5		1	
FXAM = Examine ST(0)					
	11011 001 : 1110 0101	8			

Table 13-19. Floating-Point Clock Count Summary (Continued)

Instruction Format	Cache Hit: Avg (Lower Range ... Upper Range)	Penalty if Cache Miss	Concurrent Execution: Avg (Lower Range ... Upper Range)	Notes
CONSTANTS				
FLDZ = Load +0.0 Into ST(0) 11011 001 : 1110 1110 :	4			
FLD1 = Load +1.0 Into ST(0) 11011 001 : 1110 1000 :	4			
FLDP1 = Load π Into ST(0) 11011 001 : 1110 1011 :	8		2	
FLDL2T = Load $\log_2(10)$ Into ST(0) 11011 001 : 1110 1001 :	8		2	
FLDL2E = Load $\log_2(e)$ Into ST(0) 11011 001 : 1110 1010 :	8		2	
FLDLG2 = Load $\log_{10}(2)$ Into ST(0) 11011 001 : 1110 1100 :	8		2	
FLDLN2 = Load $\log_e(2)$ Into ST(0) 11011 001 : 1110 1101 :	8		2	
ARITHMETIC				
FADD = Add Real with ST(0)				
ST(0) ← ST(0) + 32-bit memory 11011 000 : mod 000 r/m : s-i-b/disp.	10(8-20)	2	7(5-17)	
ST(0) ← ST(0) + 64-bit memory 11011 100 : mod 000 r/m : s-i-b/disp.	10(8-20)	3	7(5-17)	
ST(d) ← ST(0) + ST(i) 11011 d00 : 11000 ST(i)	10(8-20)		7(5-17)	
FADDP = Add real with ST(0) and Pop (ST(i) ← ST(0) +ST(i)) 11011 110 : 11000 ST(i) :	10(8-20)		7(5-17)	

4

Table 13-19. Floating-Point Clock Count Summary (Continued)

Instruction Format	Cache Hit: Avg (Lower Range ... Upper Range)	Penalty if Cache Miss	Concurrent Execution: Avg (Lower Range ... Upper Range)	Notes
ARITHMETIC (Continued)				
FSUB = Subtract Real from ST(0)				
ST(0) ← ST(0) − 32-bit memory				
11011 000 : mod 100 r/m : s-i-b/disp.	10(8-20)	2	7(5-17)	
ST(0) ← ST(0) − 64-bit memory				
11011 100 : mod 100 r/m : s-i-b/disp.	10(8-20)	3	7(5-17)	
ST(d) ← ST(0) − ST(i)				
11011 d00 : 11001 ST(i)	10(8-20)		7(5-17)	
FSUBP = Subtract real from ST(0) and Pop (ST(i) ← ST(0)−ST(i))				
11011 110 : 11001 ST(i)	10(8-20)		7(5-17)	
FSUBR = Subtract Real reversed (Subtract ST(0) from Real)				
ST(0) ← 32-bit memory − ST(0)				
11011 000 : mod 101 r/m : s-i-b/disp.	10(8-20)	2	7(5-17)	
ST(0) ← 64-bit memory − ST(0)				
11011 100 : mod 101 r/m : s-i-b/disp.	10(8-20)	3	7(5-17)	
ST(d) ← ST(i) − ST(0)				
11011 d00 : 11001 ST(i)	10(8-20)		7(5-17)	
FSUBRP = Subtract Real reversed and Pop (ST(i) ← ST(i)−ST(0))				
11011 110 : 11100 ST(i)	10(8-20)		7(5-17)	
FMUL = Multiply Real with ST(0)				
ST(0) ← ST(0) X 32-bit memory				
11011 000 : mod 001 r/m : s-i-b/disp.	11	2	8	
ST(0) ← ST(0) X 64-bit memory				
11011 100 : mod 001 r/m : s-i-b/disp.	14	3	11	
ST(d) ← ST(0) X ST(i)				
11011 d00 : 11001 ST(i)	16		13	
FMULP = Multiply ST(0) with ST(i) and Pop (ST(i) ← ST(0)×ST(i))				
11011 110 : 11001 ST(i)	16		13	
FDIV = Divide ST(0) by Real				
ST(0) ← ST(0)/ 32-bit memory				
11011 000 : mod 110 r/m : s-i-b/disp.	73	2	70	3
ST(0) ← ST(0)/ 64-bit memory				
11011 100 : mod 110 r/m : s-i-b/disp.	73	3	70	3
ST(d) ← ST(0)/ ST(i)				
11011 d00 : 11111 ST(i)	73		70	3
FDIVP = Divide ST(0) by ST(i) and Pop (ST(i) ← ST(0)/ST(i))				
11011 110 : 11111 ST(i)	73		70	3

Table 13-19. Floating-Point Clock Count Summary (Continued)

Instruction Format	Cache Hit: Avg (Lower Range ... Upper Range)	Penalty if Cache Miss	Concurrent Execution: Avg (Lower Range ... Upper Range)	Notes
ARITHMETIC (Continued)				
FDIVR = Divide real reversed (Real/ST(0))				
ST(0) ← 32-bit memory/ ST(0)				
11011 000 : mod 111 r/m : s-i-b/disp.	73	2	70	3
ST(0) ← 64-bit memory/ ST(0)				
11011 100 : mod 111 r/m : s-i-b/disp.	73	3	70	3
ST(d) ← ST(i)/ ST(0)				
11011 d00 : 11110 ST(i)	73		70	3
FDIVRP = Divide real reversed and Pop (ST(i) ← ST(i)/ ST(0))				
11011 110 : 11110 ST(i)	73		70	3
FIADD = Add Integer to ST(0)				
ST(0) ← ST(0) + 16-bit memory				
11011 110 : mod 000 r/m : s-i-b/disp.	24(20-35)	2	7(5-17)	
ST(0) ← ST(0) + 32-bit memory				
11011 010 : mod 000 r/m : s-i-b/disp.	22.5(19-32)	2	7(5-17)	
FISUB = Subtract Integer from ST(0)				
ST(0) ← ST(0) − 16-bit memory				
11011 110 : mod 100 r/m : s-i-b/disp.	24(20-35)	2	7(5-17)	
ST(0) ← ST(0) − 32-bit memory				
11011 010 : mod 100 r/m : s-i-b/disp.	22.5(19-32)	2	7(5-17)	
FISUBR = Integer Subtract Reversed				
ST(0) ← 16-bit memory − ST(0)				
11011 110 : mod 101 r/m : s-i-b/disp.	24(20-35)	2	7(5-17)	
ST(0) ← 32-bit memory − ST(0)				
11011 010 : mod 101 r/m : s-i-b/disp.	22.5(19-32)	2	7(5-17)	
FIMUL = Multiply Integer with ST(0)				
ST(0) ← ST(0) X 16-bit memory				
11011 110 : mod 101 r/m : s-i-b/disp.	25(23-27)	2	8	
ST(0) ← ST(0) X 32-bit memory				
11011 010 : mod 001 r/m : s-i-b/disp.	23.5(19-32)	2	8	

Table 13-19. Floating-Point Clock Count Summary (Continued)

Instruction Format	Cache Hit Avg (Lower Range... Upper Range)	Penalty if Cache Miss	Concurrent Execution Avg (Lower Range... Upper Range)	Notes
ARITHMETIC (Continued)				
FIDIV = Integer Divide				
ST(0) ← ST(0)/ 16-bit memory				
11011 110 : mod 110 r/m : s-i-b/disp.	87(85-89)	2	70	3
ST(0) ← ST(0)/ 32-bit memory				
11011 010 : mod 110 r/m : s-i-b/disp.	85.5(84-86)	2	70	3
FIDVR = Integer Divide Reversed				
ST(0) ← 16-bit memory/ST(0)				
11011 110 : mod 111 r/m : s-i-b/disp.	87(85-89)	2	70	3
ST(0) ← 32-bit memory/ST(0)				
11011 010 : mod 111 r/m : s-i-b/disp.	85.5(84-86)	2	70	3
FSQRT = Square Root				
11011 001 : 1111 1010	85.5(83-87)		70	
FSCALE = Scale ST(0) by ST(1)				
11011 001 : 1111 1101	31(30-32)		2	
FXTRACT = Extract Components of ST(0)				
11011 001 : 1111 0100	19(16-20)		4(2-4)	
FPREM = Partial Reminder				
11011 001 : 1111 1000	84(70-138)		2(2-8)	
FPREM1 = Partial Reminder (IEEE)				
11011 001 : 1111 0101	94.5(72-167)		5.5(2-18)	
FRNDINT = Round ST(0) to Integer				
11011 001 : 1111 1100	29.1(21-30)		7.4(2-8)	
FABS = Absolute value of ST(0)				
11011 001 : 1110 0001	3			
FCHS = Change Sign of ST(0)				
11011 001 : 1110 0000	6			
TRANSCENDENTAL				
FCOS = Cosine of ST(0)				
11011 001 : 1111 1111	241(193-279)		2	6,7
FPTAN = Partial Tangent of ST(0)				
11011 001 : 1111 0010	244(200-273)		70	6,7
FPATAN = Partial Arctangent				
11011 001 : 1111 0011	289(218-303)		5(2-17)	6

Table 13-19. Floating-Point Clock Count Summary (Continued)

Instruction Format	Cache Hit: Avg (Lower Range ... Upper Range)	Penalty if Cache Miss	Concurrent Execution: Avg (Lower Range ... Upper Range)	Notes
TRANSCENDENTAL (Continued)				
FSIN = Sine of ST(0) 11011 001 : 1111 1110	241(193-279)		2	6,7
FSINCOS = Sine and Cosine of ST(0) 11011 001 : 1111 1011	291(243-329)		2	6,7
F2XM1 = $2^{ST(0)} - 1$ 11011 001 : 1111 0000	242(140-279)		2	6
FYL2X = ST(1) x $\log_2$(ST(0)) 11011 001 : 1111 0001	311(196-329)		13	6
FYL2XP1 = ST(1) x $\log_2$(ST(0) + 1.0) 11011 001 : 1111 1001	313(171-326)		13	6
PROCESSOR CONTROL				
FINIT = Initialize FPU 11011 001 : 1110 0011	17			4
FSTSW AX = Store status word into AX 11011 111 : 1110 0000	3			5
FSTSW = Store status word into memory 11011 101 : mod 111 r/m : s-i-b/disp.	3			5
FLDCW = Load control word 11011 001 : mod 101 r/m : s-i-b/disp.	4	2		
FSTCW = Store control word 11011 001 : mod 111 r/m : s-i-b/disp.	3			5
FCLEX = Clear exceptions 11011 011 : 1110 0010	7			4
FSTENV = Store environment 11011 011 : mod 110 r/m : s-i-b/disp.				
Real and Virtual Modes 16-bit address	67			4
Real and Virtual Modes 32-bit address	67			4
Protected Mode 16-bit address	56			4
Protected Mode 32-bit address	56			4
FLDENV = Load Environment 11011 011 : mod 100 r/m : s-i-b/disp.				
Real and Virtual Modes 16-bit address	44	2		
Real and Virtual Modes 32-bit address	44	2		
Protected Mode 16-bit address	34	2		
Protected Mode 32-bit address	34	2		

Table 13-19. Floating-Point Clock Count Summary (Continued)

Instruction Format	Cache Hit: Avg (Lower Range ... Upper Range)	Penalty if Cache Miss	Concurrent Execution: Avg (Lower Range ... Upper Range)	Notes
PROCESSOR CONTROL (Continued)				
FSAVE = Save State				
11011 101 : mod 110 r/m : s-i-b/disp.				
Real and Virtual Modes 16-bit address	154			4
Real and Virtual Modes 32-bit address	154			4
Protected Mode 16-bit address	143			4
Protected Mode 32-bit address	143			4
FRSTOR = Restore State				
11011 101 : mod 100 r/m : s-i-b/disp.				
Real and Virtual Modes 16-bit address	131	23		
Real and Virtual Modes 32-bit address	131	27		
Protected Mode 16-bit address	120	23		
Protected Mode 32-bit address	120	27		
FINCSTP = Increment Stack Pointer				
11011 001 : 1111 0111	3			
FDECSTP = Decrement Stack Pointer				
11011 001 : 1111 0110	3			
FFREE = Free ST(i)				
11011 101 : 11000 ST(i)	3			
FNOP = No Operations				
11011 101 : 1101 0000	3			
WAIT = Wait until FPU ready (min/max)				
10011011	1/3			

NOTES:
1. If operand is 0 clock counts = 27.
2. If operand is 0 clock counts = 28.
3. If CW.PC indicates 24-bit precision then subtract 38 clocks.
   If CW.PC indicates 53-bit precision then subtract 11 clocks.
4. If there is a numeric error pending from a previous instruction add 17 clocks.
5. If there is a numeric error pending from a previous instruction add 18 clocks.
6. The INT pin is polled several times while this function is executing to assure short interrupt latency.
7. If ABS(operand) is greater than $\pi/4$ then add n clocks, where n=(operand/($\pi/4$)).

14.0 DIFFERENCES BETWEEN Intel486™ PROCESSORS AND Intel386™ PROCESSORS

The differences between Intel486 processors and Intel386 processors are due to performance enhancements. The differences are listed below.

1. Instruction clock counts have been reduced to achieve higher performance. (See section 13.0, "Instruction Set Summary.")
2. The Intel486 processor bus is significantly faster than the Intel386 processor bus. Differences include a 1X clock, parity support, burst cycles, cacheable cycles, cache invalidate cycles and 8-bit bus support. The Hardware Interface and Bus Operation sections (sections 9.0 and 10.0) of the data sheet should be carefully read to understand the Intel486 processor bus functionality.
3. To support the on-chip cache bits have been added to control register 0 (CD and NW) (see section 4.2.3.1, "Control Registers"), new pins have been added to the bus (see section 9.0, "Hardware Interface") and new bus cycle types have been added (see section 10.0, "Bus Operation"). The on-chip cache needs to be enabled after reset by clearing the CD and NW bit in CR0.
4. Eight new instructions have been added:
   - Byte Swap (BSWAP)
   - Exchange-and-Add (XADD)
   - Compare and Exchange (CMPXCHG)
   - Invalidate Data Cache (INVD)
   - Write-back and Invalidate Data Cache (WBINVD)
   - Invalidate TLB Entry (INVLPG)
   - Processor Identification (CPUID)
   - Resume (RSM)
5. Two bits defined in control register 3, the page table entries and page directory entries (PCD and PWT). (See section 6.4.2.5, "Page Directory/Table Entries.")
6. A page protection feature has been added. This feature required a new bit in control register 0 (WP) (See sections 4.2.3.1 "Control Registers" and 6.4.3 "Page Level Protection.")
7. An Alignment Check feature has been added. This feature required a bit in the flags register (AC) (section 4.2.2.3 "Flags Register") and a bit in control register 0 (AM) (section 4.2.3.1 "Control Registers").
8. The replacement algorithm for the translation lookaside buffer has been changed from a random algorithm to a pseudo least recently used algorithm like that used by the on-chip cache. (See section 7.5 "Cache Replacement" for a description of the algorithm.)
9. Three testability registers, TR3, TR4 and TR5, have been added for testing the on-chip cache. TLB testability has been enhanced. (See section 11.0, "Testability.")
10. The prefetch queue has been increased from 16 bytes to 32 bytes. A jump always needs to execute after modifying code to guarantee correct execution of the new instruction.
11. After reset, the ID in the upper byte of the DX register is 04.

14.1 Differences Between the Intel386™ Processor with an Intel387™ Math CoProcessor and Intel486™ DX, IntelDX2™ and IntelDX4™ Processors

In addition to the previously mentioned enhancements, the Intel486 DX, IntelDX2 and IntelDX4 processors offer the following features:

1. The complete Intel387 math coprocessor instruction set and register set have been added. No I/O cycles are performed during FLOATING-POINT instructions. The instruction and data pointers are set to 0 after FINIT/FSAVE. Interrupt 9 can no longer occur, interrupt 13 occurs instead.
2. Support for FLOATING-POINT error reporting modes to guarantee DOS compatibility. These modes require a bit in control register 0 (NE) (see section 4.2.3.1, "Control Registers") and pins (FERR# and IGNNE#). (See sections 9.2.15, "Numeric Error Reporting" and 10.2.14 "FLOATING-POINT Error Handling.")
3. In some cases FERR# is asserted when the next FLOATING-POINT instruction is encountered and in other cases it is asserted before the next FLOATING-POINT instruction is encountered, depending upon the execution state the instruction causing exception. (See sections 9.2.15, "Numeric Error Reporting" and 10.2.14, "FLOATING-POINT Error Handling.") For both of these cases, the Intel387 math coprocessor asserts ERROR# when the error occurs and does not wait for the next FLOATING-POINT instruction to be encountered.
4. The contents of the base registers ***including the FLOATING-POINT registers*** may be different after reset.

15.0 DIFFERENCES BETWEEN THE PGA, SQFP AND PQFP VERSIONS OF THE Intel486™ SX AND Intel486 DX PROCESSORS

The section lists the differences between PGA, SQFP, and PQFP packages of the Intel486 SX and Intel486 DX processor. It also provides a quick pin reference table that is useful for converting a system design from one that uses a PGA package to one that uses an SQFP or PQFP package.

NOTE:
The boundary scan feature is not supported in the Intel486 SX processor in 168-pin PGA package. See sections 3.0, "Pin Description," and 11.5, "Intel486 Processor Boundary Scan," for pinout differences.

15.1 2X Clock Mode

The Intel486 processors offer 2X clock mode for systems that rely on dynamic frequency scaling for processor power management. **This product is not intended for the desktop computer.** This 2X clock processor differs from the 1X clock processor in the following ways:

Pin Assignment/Function: The 2X clock product has a CLK2 input, rather than the 1X clock product's CLK input. The CLK2 input must be synchronized to the system phase using the falling edge of RESET. (For reference, the pinout change from the existing Low Power Intel486 DX and SX processors is also shown. The CLKSEL pin is not used on the Intel486 processors, as it is on the existing Low Power Intel486 DX and SX processors.)

Clock Control: The CLK2 input can be changed dynamically. The Stop Clock interrupt is handled in a different manner.

AC Specifications: In general, the AC specifications for the 2X clock device will have slightly longer setups, holds, and maximum valid delays. This is consistent with the existing Low Power Intel486 DX and Intel486 SX processors. See section 15.1.5, "AC Specifications," for 2X clock mode AC specifications.

Upgrades: There are no end user upgrade products planned for the 2X clock mode product. The UP# function is still provided for use by system designers that offer Intel486 SX to Intel486 DX processor upgrade cards.

This section will explain the differences between the processor with the 2X clock mode and the processor with the 1X clock mode.

15.1.1 PIN ASSIGNMENTS

The Intel486 processor with the 2X clock option is available in the 208-lead SQFP and 196-lead PQFP packages. The pinout is identical to the Intel486 processor with the 1X clock option with the exception of the name of the clock input. **The 1X clock input is called CLK and the 2X clock input is called CLK2.**

Table 15-1 shows the changes between the existing products and new products.

Table 15-2 is a list of pin descriptions.

Table 15-1. Pinout Differences for the 2X Clock Mode (Low Power) Processors (196-Lead PQFP Package)

Pin	Low Power Intel486™ SX Processor	Intel486 SX Processor	Low Power Intel486 DX Processor	Intel486 DX Processor
75	NC	STPCLK#	NC	STPCLK#
77	NC	NC	IGNNE#	IGNNE#
81	NC	NC	FERR#	FERR#
85	NC	SMI#	N C	SMI#
92	NC	SMIACT#	NC	SMIACT#
94	NC	SRESET	NC	SRESET
127	CLKSEL	NC	CLKSEL	NC

15.1.2 QUICK PIN REFERENCE

Table 15-2. Pin Descriptions

Symbol	Type	Name and Function
CLK2	I	**CLK2** provides the fundamental timing for the processor. Both of the internal timing phases, phase-1 (ph1) and phase-2 (ph2), are provided by the external CLK2 input. All external timing parameters are specified with respect to the phase-1 rising edge of CLK2. For the 2X clock mode the CLK frequency is twice the frequency of the processor.
RESET	I	The **RESET** input forces the processor to begin execution at a known state. The processor cannot begin execution of instructions until at least 1 ms after V_{CC} and CLK2 have reached their proper AC and DC specifications. However, for soft resets, RESET should remain active for at least 30 CLK2 periods (equal to 15 internal processor CLK). The RESET pin should remain active during this time to insure proper processor operation. RESET is active HIGH. Reset is asynchronous, but must meet setup and hold times t_{20}, t_{20a} and t_{21} for recognition in any specific clock. RESET sets the SMBASE descriptor to default address of 30000H. If the system uses SMBASE relocation, then the SRESET pin should be used for soft resets. For the 2X clock mode, the falling edge of RESET synchronizes the processor internal clock phase. RESET must be used at power up and anytime the phase of the processor clock must be re-synchronized to the system phase.
SRESET	I	The SRESET pin duplicates all the functionality of the RESET pin with the following two exceptions: 1. The SMBASE register will retain its previous value. 2. If UP# is asserted, SRESET will not have an effect on the host processor. For soft resets, SRESET should remain active for at least 30 CLK2 periods (equal to 15 internal processor CLK). SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

15.1.3 CLOCK CONTROL

15.1.3.1 Clock Generation

The frequency of **CLK2** is twice the internal frequency of the processor. The internal clock is comprised of two phases, "PH1" and "PH2". Each CLK2 period is a phase of the internal clock. Figure 15-1 illustrates the relationship between the CLK2 input and the internal phases. All set-up, hold, float-delay and valid delay timings are referenced to the rising edge of phase 1 of CLK2. Thus it is important to synchronize the external circuitry with the phase of the CLK2 input. The internal processor clock phase is determined at the falling edge of the RESET input. RESET must meet the specified setup and hold times to correctly synchronize the internal clock phase. See Figure 15-2.

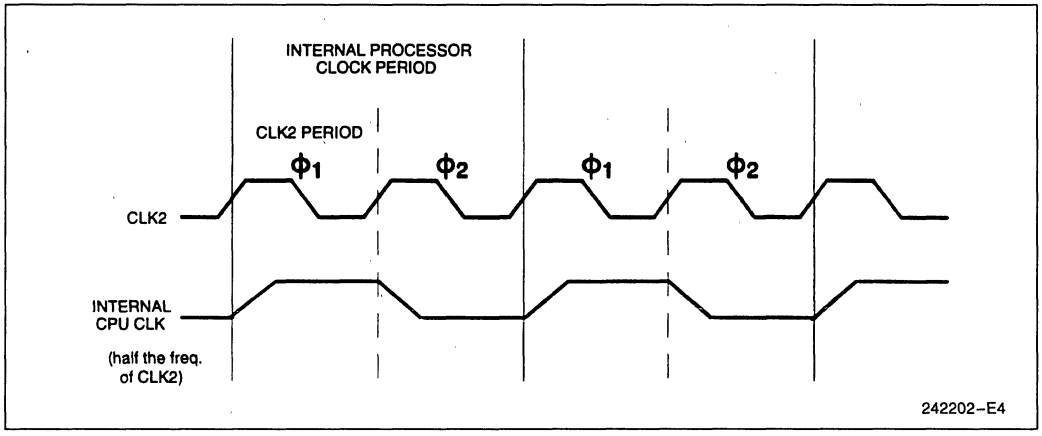


Figure 15-1. CLK2 Signal and Internal Processor Clock

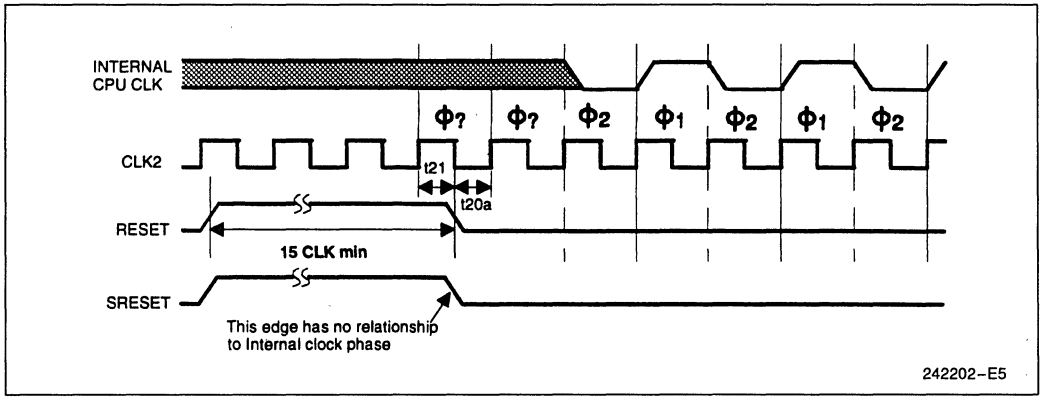


Figure 15- 2. CLK2 and Internal Processor CLK vs. SRESET and RESET Timings

15.1.3.2 Stop Clock

The processor with the 2X clock option does not rely on an internal Phase Lock Loop to generate the internal phase clocks. Therefore, the frequency of the CLK2 input can be changed dynamically or "on-the-fly."

The 2X clock mode, Intel486 processor provides an interrupt mechanism, STPCLK#, that places the processor into a known state. Although the frequency of the CLK2 input can be dynamically changed between 0 MHz and the maximum operating frequency of the processor, operation between 0 MHz and 8 MHz is not tested. Stopping the CLK2 input with the processor in a known state requires use of the STPCLK# mechanism. When the processor recognizes a STPCLK# request, the processor will stop execution on the next instruction boundary, stop the prefetcher, empty all internal pipelines and the write buffers, and then generate a Stop Grant bus cycle. At this point the processor is in the Stop Grant state.

The rising edge of STPCLK# will tell the processor that it can return to program execution at the instruction following the interrupted instruction.

Unlike the normal interrupts, INTR and NMI, the STPCLK# interrupt does not initiate interrupt acknowledge cycles or interrupt vector table reads.

STPCLK# is active LOW and is provided with an internal pull-up resistor. STPCLK# is an asynchronous signal, but must remain active until the processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle. Note that STPCLK# should NOT be de-asserted before the processor has issued the Stop Grant bus cycle. STPCLK# must be de-asserted for a minimum of 5 clocks after RDY# is returned active for the Stop Grant bus cycle before being asserted again.

15.1.3.3 Clock Control State Diagram

The state diagram in Figure 15-3 shows the Stop Clock state transitions for the 2X clock mode.

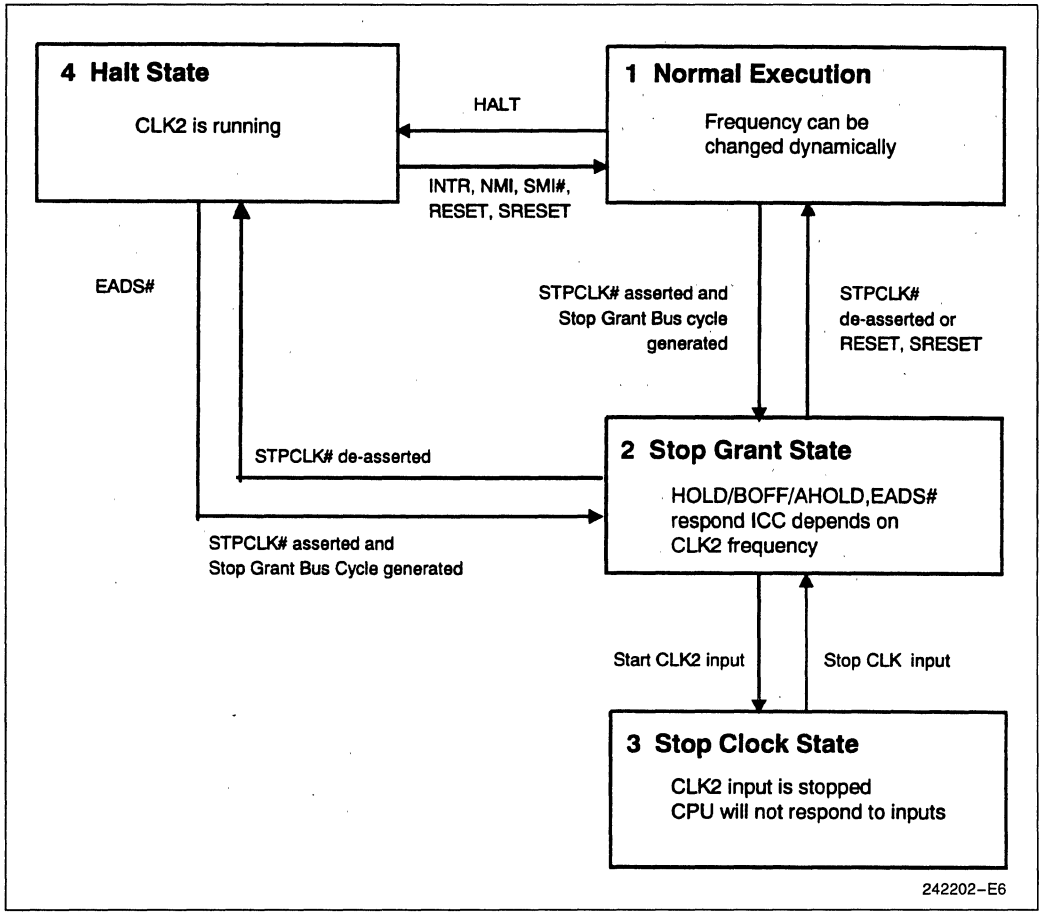


Figure 15-3. Stop Clock State Machine—2X Clock Mode

Normal State

This is the Normal operating state of the processor. During this state, the CLK2 input frequency can be changed dynamically or "on-the-fly" for power consumption control with no clock control latency. This capability provides a wide range of performance/power consumption options. Operation of the processor is tested between 8 MHz and the maximum operating frequency of the processor. Operation below 8 MHz is guaranteed by design, though is not 100% tested. Operation at 0 MHz is tested when the stop clock protocol (STPCLK#) is used.

Stop Grant State

The processor enters the Stop Grant state in response to a STPCLK# interrupt. The processor will generate a Stop Grant bus cycle when it enters this state from the Normal state or the HALT state. The processor will not generate a Stop Grant bus cycle when it enters the Stop Grant state from the Stop Clock state.

While in the **Stop Grant state**, the pull-up resistors on STPCLK# and UP# are disabled internally. The system must continue to drive these inputs to the state they were in immediately before the processor entered the Stop Grant state. For minimum processor power consumption, all other input pins should be driven to their inactive level while the processor is in the Stop Grant state.

During the Stop Grant state, the processor will respond to HOLD, AHOLD and BOFF# normally and can perform cache invalidates. An active edge on either the SMI# or NMI interrupts will be latched and will be serviced after the rising edge of STPCLK#. An INTR request will be serviced after the processor returns to the normal state as long as INTR is held active until the processor issues an interrupt acknowledge bus cycle.

Stop Clock State

The processor enters the Stop Clock state when the system stops the CLK2 input. The system can stop the CLK2 input on either a logic high or a logic low. The CLK2 input must be restarted in the same state as when it was stopped. In other words, any CLK2 input state can be stretched. (See Figure 15-4 for details.) Processor operation at 0 MHz is tested only when the processor is in the Stop Clock state.

In the Stop Clock state, the processor does not respond to any stimulus. The processor must re-enter the Stop Grant state (CLK2 input must be restarted) in order to perform any bus actions such as HOLD/HLDA cycles, invalidates (AHOLD/EADS# or FLUSH# cycles), and BOFF#. It is recommended that CLK2 be restarted 2 clocks before and continue until 2 clocks after the transition of the HOLD, AHOLD, EADS#, FLUSH#, or BOFF# signals.

The interrupt signals (SMI#, NMI, and INTR) will be recognized and serviced correctly if the input is held in the active state until the processor returns to the Stop Grant state. The Intel486 processor family requires that INTR be held active until the processor issues an interrupt acknowledge cycle in order to guarantee recognition. This condition also applies to the existing Intel486 processors.

HALT State

The processor enters the HALT state from the Normal state on a HALT instruction. The system can place the processor into the Stop Grant state from the HALT state by asserting the STPCLK# input. The processor will generate a Stop Grant bus cycle when it enters the Stop Grant state. If the processor entered the Stop Grant state from the HALT state then it will return to the HALT state when the STPCLK# interrupt is de-asserted. When the processor re-enters the HALT state it will generate a HALT bus cycle.

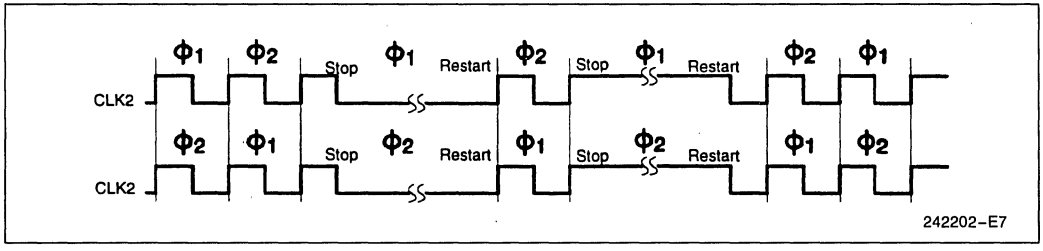


Figure 15-4. CLK2 Phase Coherence in CLK2 Stop and Restart

15.1.3.3 Supply Current Model for Stop Clock Modes and Transitions

Figure 15-5 illustrates the effect of different Stop Clock state transitions on the supply current.

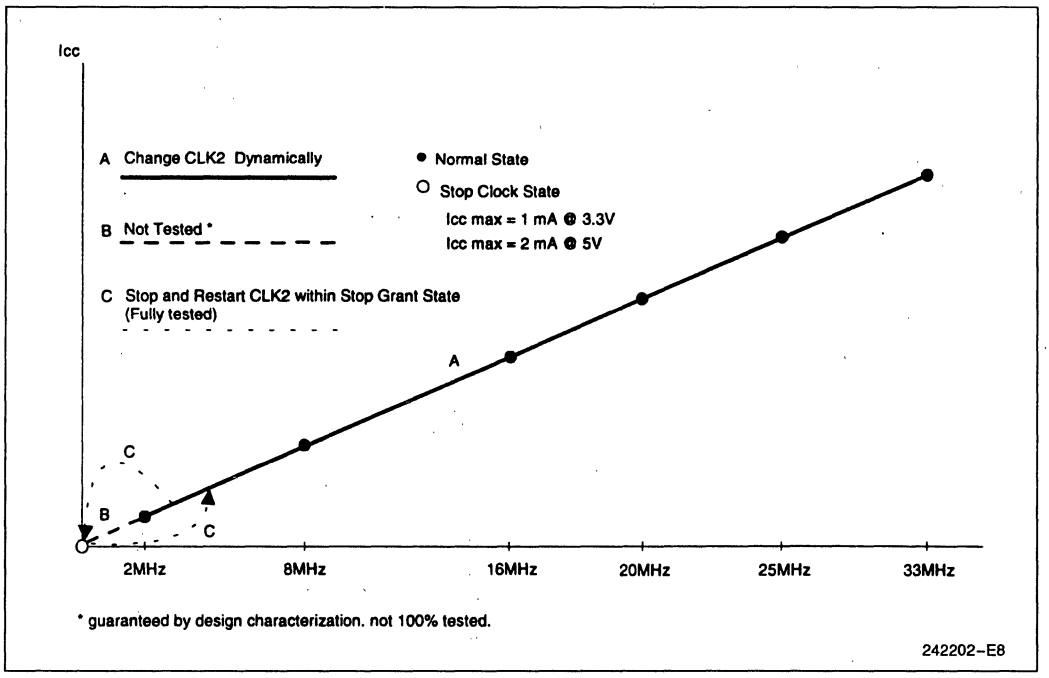


Figure 15-5. Supply Current Model for Stop Clock Modes and Transitions

15.1.4 DC SPECIFICATIONS FOR 2X CLOCK OPTION

For 2X clock DC specifications, refer to the 1X clock DC specifications. (See section 17.3, "DC Specifications.")

15.1.5 AC SPECIFICATIONS FOR 2X CLOCK OPTION

The AC specifications given in the tables of this section consist of output delays, input setup requirements and input hold requirements. All AC specifications are relative to the rising edge of the phase 1 of the input system clock (CLK2), unless otherwise specified. (See Figures 15-6 through 15-8.)

15.1.5.1 3.3V AC Characteristic

Table 15-3 is for 25- and 33-MHz Intel486 SX and 33-MHz Intel486 DX processors in 2X Clock Mode.

Table 15-3. 3.3V AC Characteristics (2X Clock)

Functional operating range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified

Symbol	Parameter	Min	Max	Min	Max	Unit	Notes
	Frequency		**25**		**33**	MHz	Note 1
	CLK2 Frequency		**50**		**66**	MHz	Note 1
t_1	CLK2 Period	20		15		ns	
t_2	CLK2 High Time	7		5		ns	at 2V
t_3	CLK2 Low Time	7		5		ns	at 0.8V
t_4	CLK2 Fall Time		2		2	ns	2V to 0.8V, Note 2
t_5	CLK2 Rise Time		2		2	ns	0.8V to 2V, Note 2
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT#, FERR# Valid Delay	3	19	3	17	ns	Note 3
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA Float Delay		28		21	ns	Note 2
t_8	PCHK# Valid Delay	3	24	3	23	ns	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	24	3	21	ns	
t_9	BLAST#, PLOCK# Float Delay		28		21	ns	Note 2
t_{10}	D0–D31, DP0–DP3 Write Data Valid Delay	3	20	3	19	ns	
t_{11}	D0–D31, DP0–DP3 Write Data Float Delay		28		21	ns	Note 2
t_{12}	EADS# Setup Time	9		6		ns	

Table 15-3. 3.3V AC Characteristics (2X Clock) (Cont'd)

Functional operating range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified

Symbol	Parameter	Min	Max	Min	Max	Unit	Notes
t_{13}	EADS# Hold Time	4		4		ns	
t_{14}	KEN#, BS16#, BS8# Setup Time	9		6		ns	
t_{15}	KEN#, BS16#, BS8# Hold Time	4		4		ns	
t_{16}	RDY#, BRDY# Setup Time	9		6		ns	
t_{17}	RDY#, BRDY# Hold Time	4		4		ns	
t_{18}	HOLD, AHOLD Setup Time	11		7		ns	
t_{18a}	BOFF# Setup Time	11		9		ns	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	4		4		ns	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Setup Time	11		6		ns	Note 3
t_{20a}	RESET Falling Edge Setup Time	9		5		ns	
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Hold Time	4		4		ns	Note 3
t_{22}	D0–D31, DP0–DP3, A4–A31 Read Setup Time	6		6		ns	
t_{23}	D0–D31, DP0–DP3, A4–A31 Read Hold Time	4		4		ns	

NOTES:
1. 0-MHz operation is tested using the STPCLK# and Stop Grant bus cycle protocol. Operation between 0 MHz < CLK2 < 8 MHz is guaranteed by design characterization, but is not 100% tested.
2. Not 100% tested, guaranteed by design characterization.
3. FERR# and IGNNE# are present only in Intel486 DX processors.

15.1.5.2 5V AC Characteristics

Table 15-4 is for 25- and 33-MHz Intel486 SX and 33-MHz Intel486 DX processors in 2X Clock Mode.

Table 15-4. 5V AC Characteristics (2X Clock)
Functional operating range: V_{CC} = 5V ±5%; T_{CASE} = 0°C +85°C; C_L = 50 pF, unless otherwise specified.

Symbol	Parameter	Min	Max	Min	Max	Unit	Notes
	Frequency		**25**		**33**	MHz	Note 1
	CLK2 Frequency		**50**		**66**	MHz	Note 1
t_1	CLK2 Period	20		15		ns	
t_2	CLK2 High Time	7		5		ns	at 2V
t_3	CLK2 Low Time	7		5		ns	at 0.8V
t_4	CLK2 Fall Time		2		2	ns	2V to 0.8V, Note 2
t_5	CLK2 Rise Time		2		2	ns	0.8V to 2V, Note 2
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT#, FERR# Valid Delay	3	19	3	17	ns	Note 3
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA Float Delay		28		21	ns	Note 2
t_8	PCHK# Valid Delay	3	24	3	23	ns	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	24	3	21	ns	
t_9	BLAST#, PLOCK# Float Delay		28		21	ns	Note 2
t_{10}	D0–D31, DP0–DP3 Write Data Valid Delay	3	20	3	19	ns	
t_{11}	D0–D31, DP0–DP3 Write Data Float Delay		28		21	ns	Note 2
t_{12}	EADS# Setup Time	9		6		ns	
t_{13}	EADS# Hold Time	4		4		ns	
t_{14}	KEN#, BS16#, BS8# Setup Time	9		6		ns	
t_{15}	KEN#, BS16#, BS8# Hold Time	4		4		ns	
t_{16}	RDY#, BRDY# Setup Time	9		6		ns	

Table 15-4. 5V AC Characteristics (2X Clock) (Continued)
Functional operating range: V_{CC} = 5V ±5%; T_{CASE} = 0°C + 85°C; C_L = 50 pF, unless otherwise specified.

Symbol	Parameter	Min	Max	Min	Max	Unit	Notes
t_{17}	RDY#, BRDY# Hold Time	4		4		ns	
t_{18}	HOLD, AHOLD Setup Time	11		7		ns	
t_{18a}	BOFF# Setup Time	11		9		ns	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	4		4		ns	
t_{20}	FLUSH#, A20M#, NMI, INTR SMI#, STPCLK#, SRESET, RESET, IGNNE#, Setup Time	11		6		ns	Note 3
t_{20a}	RESET Falling Edge Setup Time	9		5		ns	
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Hold Time	4		4		ns	Note 3
t_{22}	D0–D31, DP0–DP3, A4–A31 Read Setup Time	6		6		ns	Note 3
t_{23}	D0–D31, DP0–DP3, A4–A31 Read Hold Time	4		4		ns	

NOTES:
1. 0-MHz-operation is tested using the STPCLK# and Stop Grant bus cycle protocol. Operation between 0 MHz < CLK2 < 8 MHz is guaranteed by design characterization, but is not 100% tested.
2. Not 100% tested, guaranteed by design characterization.
3. FERR# and IGNNE# are present only in Intel486 DX processors.

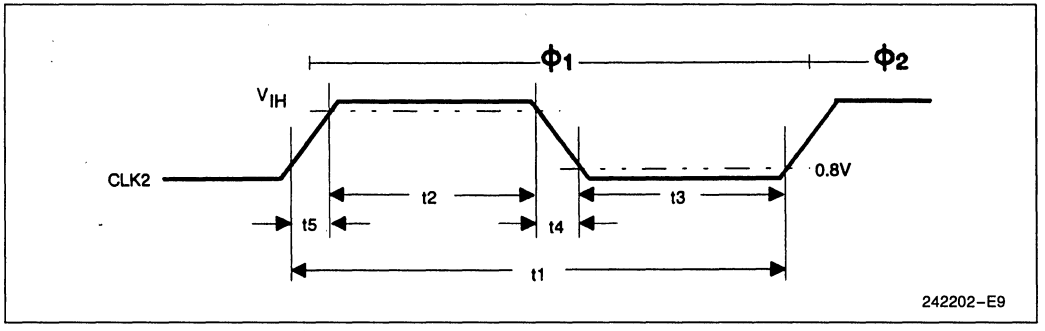
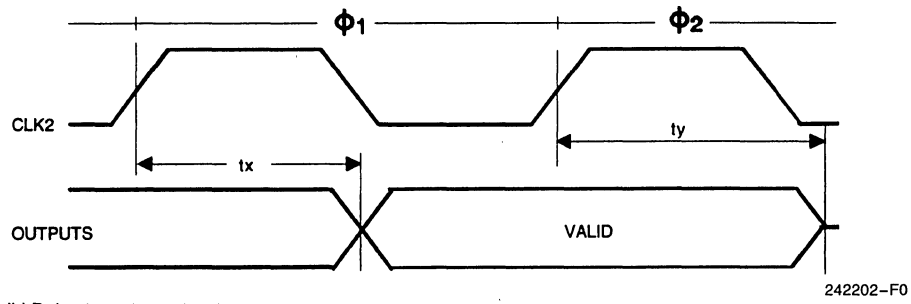
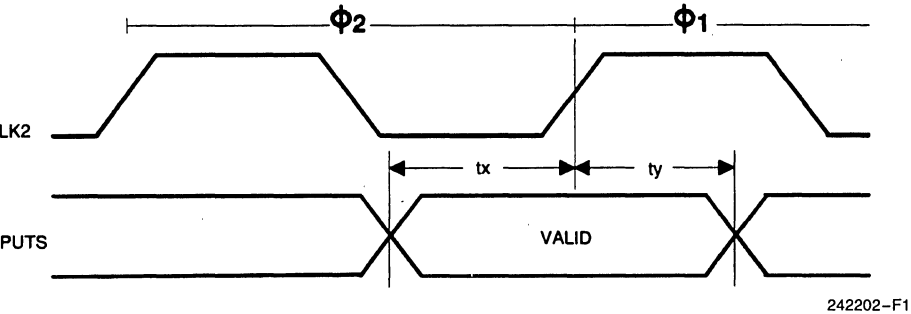


Figure 15-6. CLK2 Waveform



Output Valid Delay tx = t_6, t_8, t_{8a}, t_{10}
Output Float Delay ty = t_7, t_9, t_{11}

Figure 15-7. Valid and Float Delay Timings



tx = t_{12}, t_{14}, t_{16}, t_{18}, t_{20}, t_{20a}, t_{22}, t_{18a}
ty = t_{13}, t_{15}, t_{17}, t_{19}, t_{21}, t_{23}

Figure 15-8. Setup and Hold Timings

16.0 OverDrive® PROCESSOR SOCKET

This section contains the specifications for the OverDrive processor socket for systems based on Intel486 processors. All of the specifications described herein are based on the specifications of the Intel486 processors.

One of the most important features of the Intel486 family architecture, compared with previous Intel architectures, is its upgradability via the OverDrive processor socket. Inclusion of the OverDrive processor socket in systems based on the Intel486 family of microprocessors provides the end user with an easy and cost-effective way to increase system performance. The paradigm of simply installing an additional component into an empty OverDrive processor socket to achieve enhanced system performance is familiar to the millions of end users and dealers who have purchased Intel math coprocessor upgrades to boost system floating-point performance. The OverDrive processor provides improvement over the base system. The OverDrive processor takes advantage of Intel's next generation processor technology to provide this performance improvement.

The OverDrive processor socket described in this chapter is designed for the Pentium® OverDrive processor. The Pentium OverDrive processor is designed with Pentium processor core technology. This socket is also backwards compatible for the IntelDX2 and IntelDX4 OverDrive processors. Support for the future 3.3V Pentium OverDrive processor upgrade for IntelDX4 processor-based systems is also specified.

The Pentium OverDrive processor implements a superset of Intel486 processor signals. The new signals for the Pentium OverDrive processor socket, in addition to Intel486 processor signals, support a write-back protocol for the on-chip cache. Implementation of the cache writeback capability for the OverDrive processor socket is optional, although implementation of the on-chip writeback protocol enables maximum performance gain. For more information, please contact Intel.

As a new system architecture feature, the provision of the OverDrive processor socket as a means for PC users to take advantage of the ever more rapid advances in software and hardware technology helps to maintain the competitiveness of Intel PC-compatible systems over other architectures.

The majority of upgrade installations which take advantage of the OverDrive processor socket will be performed by end users and resellers. Therefore, it is important that the design minimize the amount of training and technical expertise required to install the OverDrive processors. Upgrade installation instructions should be clearly described in the system user's manual. In addition, by making installation simple and foolproof, PC manufacturers can reduce the risk of system damage, warranty claims and service calls. Feedback from Intel's upgrade customers highlights three main characteristics of end user easy designs: accessible OverDrive processor socket location, clear indication of upgrade component orientation, and minimization of insertion force.

OverDrive® Socket Location

The OverDrive processor socket can be located on either the motherboard or modular processor card. The OverDrive processor socket should be easily accessible for installation and readily visible when the PC case is removed. The OverDrive processor socket should not be located in a position that requires removal of any other hardware (such as hard disk drives) in order to install the OverDrive processor.

Component Orientation

The most common mistake made by end users and resellers when installing upgrades is incorrect orientation of the chip. This can result in irreversible damage to the chip. To solve this problem, Intel has designed OverDrive processor sockets and OverDrive processors with keying mechanisms to ensure that upgrade components fit in the right socket, with the right orientation. There are two OverDrive processor sockets, presented in this chapter, that can accept the OverDrive processor for Intel486 processor-based systems, designated as socket 3 and socket 6. The two sockets and the keying mechanism are illustrated in Figure 16-1.

Socket 3 is a 237-pin socket and accepts both 5V and 3.3V OverDrive processors. The keying mechanism consists of one Key Pin (A1) and four missing pins (B1, C1, A2 and A3). To be effective as a keying mechanism, the locations in the socket corresponding to the four missing pins must be plugged. This socket is designed to be backwards compatible with the 169-pin IntelDX2 and IntelDX4 OverDrive pro-

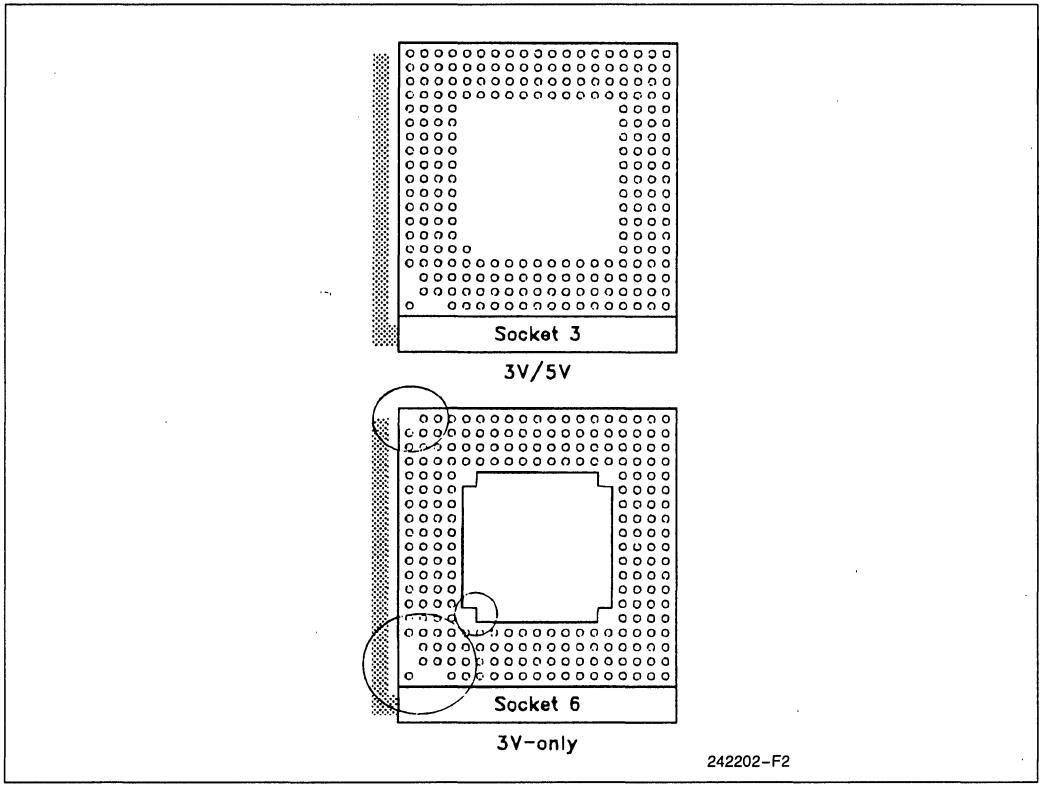


Figure 16-1. OverDrive® Processor Sockets for Intel486™ Processor-Based Systems

cessors. In order to maintain compatibility, socket 3 includes the Key Pin at location E5.

Socket 6 is a 235-pin socket and accepts 3.3V OverDrive processors only. The keying mechanism consists of one Key Pin (A1) and five missing pins (B1, C1, A2, A3 and A19). Being designed for 3.3V OverDrive processors only, socket 6 must not accept the 169-pin IntelDX2 and IntelDX4 OverDrive processors, therefore, the Key Pin E5 is missing. To be effective as a keying mechanism, the locations in the socket corresponding to the six missing pins must be plugged.

In addition, the location of the pin 1 corner should be clearly marked on the motherboard, for example by silk screening.

Insertion Force

The third major concern voiced by end users refers to how much pressure should be exerted on the upgrade chip and PC board for proper installation without damage. This becomes even more of a concern with the larger components which require up to 200 pounds of pressure for insertion into a standard screw machine socket. This level of pressure can easily result in cracked traces and stress to solder joints. To minimize the risk of system damage, it is recommended that a Zero Insertion Force (ZIF)

socket be used for the OverDrive processor socket. Designing with a ZIF socket eliminates the need to design in additional structural support to prevent flexing of the PC board during installation, and results in improved end user and reseller product satisfaction due to easy "drop-in" installation. If a LIF socket is used, sufficient support should be provided directly under the OverDrive processor socket. This will minimize the possibility of damage to the motherboard during insertion of the OverDrive processor.

16.1 OverDrive® Processor Socket Overview

The circuit design requirements for the OverDrive processor socket are discussed in section 16.2, "OverDrive Processor Circuit Design". In addition to the OverDrive processor socket circuits, there are layout considerations for the OverDrive processor socket and processor spatial requirements. These issues are discussed in section 16.3, "Socket Layout". Section 16.4, "Thermal Design Consideration", discusses the thermal considerations in the system design. Because the system must operate correctly with any OverDrive processor without a BIOS change, BIOS and software restrictions and recommendations are provided in section 16.5, "BIOS and Software". Section 16.6, "Test Requirements", discusses OverDrive processor socket test requirements. Section 16.7, "OverDrive Processor Socket Pinout", and 16.8, "3.3V Socket Specifications", specify the pinout specifications for the 5V and 3.3V OverDrive processor sockets, respectively. Finally, section 16.9, "DC/AC Specification", specifies the electrical characteristics of the OverDrive processor socket.

16.2 OverDrive® Processor Circuit Design

The Intel486 processors in the 168-pin PGA package fit in the three inner rows of the 237-pin OverDrive socket. The corresponding pins of the Intel486 processor and the OverDrive socket define identical signals, with the following exceptions:

- The pins corresponding to the signals TCK, TDI and TDO on Intel486 processors are defined as Internal No Connect (INC) on the OverDrive processor socket, and the pin corresponding to TMS is defined as UP# (pin C15 of the OverDrive processor socket). For compatibility, the system should not do boundary scan testing when the OverDrive processor is installed.
- On Intel486 processors, pin C14 defines FERR# and pin A13 is INC, while on the OverDrive processor socket pin D15 is INC and pin B14 defines FERR#.
- On Intel486 processors pin C10 defines SRESET, while on the OverDrive processor socket the corresponding pin D11 is INC.

In a system with a single socket motherboard design, the processor and the OverDrive processor share the same socket. The Intel486 processor occupies the three innermost rows of pins of the 237-pin OverDrive processor socket, with "pin 1" (notched corner) oriented toward "pin 1" corner of the socket. The processor will be replaced with the OverDrive processor when the system is upgraded. In a jumperless, plug-and-play, upgradable system, pins D15 and B14 of the OverDrive processor socket should be connected together to the FERR# signal line, and pins F19 (INIT) and D11 should be connected together to SRESET. When the OverDrive processor is installed, pin C15 (UP#) must be isolated from TMS (see Figure 16-2).

16.2.1 BACKWARD COMPATIBILITY

The Pentium OverDrive processor socket for Intel486 processor-based systems is designed to be compatible with the other OverDrive processors.

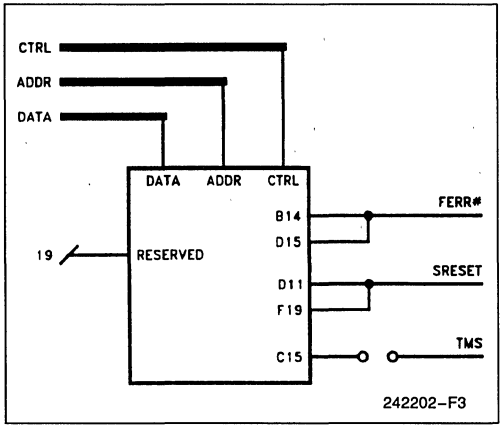


Figure 16-2. OverDrive® Processor Socket Circuit Diagram-Single Socket Design

The Pentium OverDrive processor socket has a fourth row of contacts around the outside of the 169 contacts defined for the IntelDX2 and IntelDX4 OverDrive processors. The three inner rows of the Pentium OverDrive processor socket, with the inner key pin, are 100% compatible with the 169-pin PGA OverDrive processors. For backward compatibility, the inner row key pin location (E5) must be included in the Pentium OverDrive processor socket.

16.3 Socket Layout

This section discusses three aspects for the OverDrive processor socket: size, upgradability, and vendors.

16.3.1 MECHANICAL DESIGN CONSIDERATIONS

The Pentium OverDrive processor for Intel486 processor-based systems is designed to fit in a standard 240-lead (19 x 19) PGA socket with the appropriate key holes and plugs. The Pentium OverDrive processor uses a fan/heatsink, and therefore requires vertical clearance to allow adequate air circulation.

16.3.1.1 Fan Heatsink Design

The maximum and minimum dimensions of the Pentium OverDrive processor package with a fan/heatsink are shown in Table 16-1. The fan/heatsink unit space requirement is divided into the size of the actual heatsink, and the required free space above the heatsink. The total height required for the Pentium OverDrive processor from the motherboard will depend on the height of the PGA socket. The total external height given in Table 16-1 is only measured from the PGA pin stand-offs. Table 16-1 also details the minimum clearance needed around all four sides of the PGA package.

Since the Pentium OverDrive processor dissipates more power than the Intel486 processor family members, it requires a larger cooling capacity. To facilitate the task of cooling the Pentium OverDrive processor, Intel will ship the product with a fan/heatsink. No external connections (i.e., power) will be required for the fan/heatsink. All the needed connections will be made through the pins on the outer row of the processor.

16.3.1.2 Passive Heatsink Design

The space required for the passive heatsink OverDrive processors is less than the space requirements shown in Table 16-1. The passive heatsink OverDrive processors include the IntelDX2 and IntelDX4 OverDrive processors.

16.3.2 DESIGN RECOMMENDATIONS

PC buyers value easy and safe upgrade installation. PC manufacturers can make upgrade component installation in the OverDrive processor socket simple and foolproof for the end user and reseller by implementing the suggestions listed in Table 16-2.

Table 16-1. Pentium® OverDrive® Processor, 236-Pin, PGA Package Dimensions with Fan/Heatsink Attached

Component	Length and Width (inches)		Height (inches)	
	Minimum	Maximum	Minimum	Maximum
PGA Package	1.950	1.975	0.140	0.180
Adhesive	N/A	N/A	0.008	0.012
Fan/Heatsink Unit	1.700	1.820	0.790	0.810
Required Airspace	0.200	N/A	0.400	N/A
Total Package	**1.950**	**1.975**	**0.938**	**1.002**
Total Package with Airspace	2.350	N/A	1.338	N/A

Table 16-2. Socket and Layout Considerations

Design Considerations	Implementation
Visible OverDrive® processor socket	The OverDrive processor socket should be easily visible when the PC's cover is removed. Label the OverDrive processor socket and the location of pin 1 by silk screening this information on the PC board.
Accessible OverDrive processor socket	Make the OverDrive processor socket easily accessible to the end user (i.e., do not place the OverDrive processor socket under a disk drive). Be sure to leave enough clearance to open the Zero Insertion Force (ZIF) socket.
Foolproof Chip Orientation	This OverDrive processor socket must ensure proper orientation of the OverDrive processor. The PGA package of the Pentium OverDrive processor for Intel486 processor-based systems is oriented by the four corner pins that have been removed and the "KEY" pin from the "pin 1" corner. The four contacts (A2, A3, B1 and C1) in the socket should be plugged, such that PGA pins cannot be inserted, to assure correct orientation.
Zero Insertion Force OverDrive processor socket	The high pin count of the OverDrive processor makes the insertion force required for installation into a screw machine PGA socket excessive. Even most Low Insertion Force (LIF) sockets often require more than 60 lbs. of insertion force. A Zero Insertion Force (ZIF) socket ensures that the chip insertion force does not damage the PC board. Be sure to allow enough clearance for the ZIF socket handle. Do not use a LIF or screw machine socket.
"Plug and Play"	Jumper or switch changes should not be needed to electrically configure the system for the OverDrive processor.
Thorough Documentation	Describe the OverDrive processor socket and the OverDrive processor installation procedure in the PC's User's Manual.

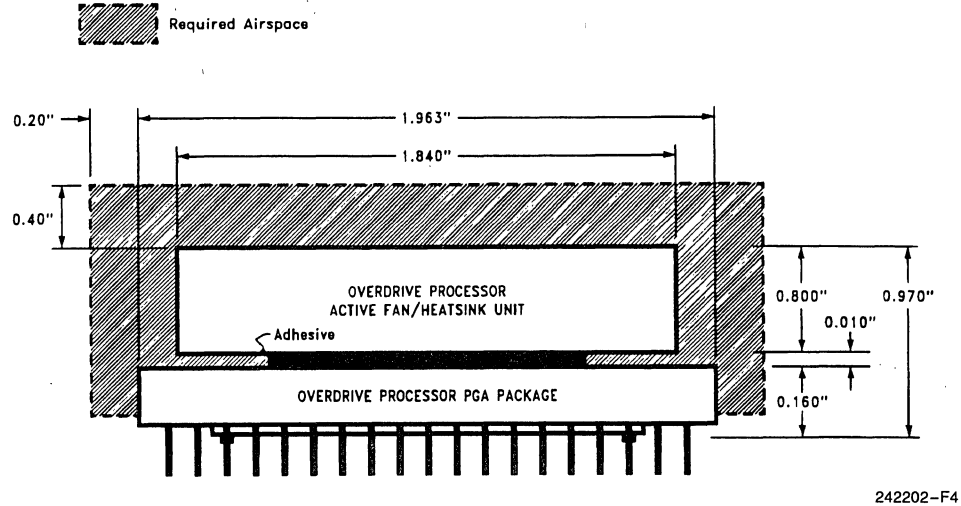


Figure 16-3. Pentium® OverDrive® Processor PGA Package with Heatsink Attached (Normal Dimensions)

16.3.3 ZIF SOCKET VENDORS

Intel has a list of qualified ZIF socket vendors. Contact Intel for more information.

16.4 Thermal Design Considerations

16.4.1 ACTIVE HEATSINK THERMAL DESIGN

The Pentium OverDrive processor for Intel486 processor-based systems will have an active fan/heatsink for thermal dissipation. The maximum allowable temperature of the air entering the fan/heatsink can not exceed 55°C under the worst case operating conditions specified for the system. The fan/heatsink reduces the need for high airflow. However, the system must provide adequate ventilation to prevent localized heating above the specified ambient.

16.4.2 PASSIVE HEATSINK THERMAL DESIGN

Passive heatsinks are used on the IntelDX2 and IntelDX4 OverDrive processors. The thermal efficiency of passive heatsink processors is dependent on system airflow. Please refer to the individual OverDrive processor datasheet for airflow requirements.

16.5 BIOS and Software

The following should be considered when designing the OverDrive processor socket for an Intel486 processor-based system.

16.5.1 OverDrive® PROCESSOR DETECTION

The component identifier and stepping/revision identifier for the OverDrive processor is readable in the DH and DL registers respectively, immediately after RESET. See Table 16-3. These values can also be obtained using the CPU_ID instruction.

As with the Intel486 processor specification, it is recommended that the BIOS save the contents of the DX register, immediately after RESET, so that this information can be used later, if required.

Table 16-3. DX Register Contents after Reset

OverDrive® Processor	Component ID (DH)	Stepping ID (DL)
Future Pentium® OverDrive processor (3.3V)	15h	5xh
Pentium OverDrive processor (5V)	15h	3xh
IntelDX4™ OverDrive processor	14h or 04h	8xh
IntelDX2™ OverDrive processor	04h	3xh

16.5.2 TIMING DEPENDENT LOOPS

The OverDrive processor for Intel486 processor-based systems executes instructions at a multiple of the frequency of the input clock. The OverDrive processor will also use advanced design techniques to decrease the number of clocks per instruction (cpi) from that of Intel486 processors. Thus software, such as instruction-based timing loops, will execute faster on the OverDrive processor than on the Intel486 processor at the same input clock frequency. Instructions such as NOP, LOOP, and JMP $+2 are frequently used by the BIOS to implement timing loops that are required, for example, to enforce recovery time between consecutive accesses for I/O devices. These instruction-based, timing-loop implementations may require modification to be compatible with this OverDrive processor socket.

In order to avoid any incompatibilities, timing loops can be implemented in hardware rather than in software. This provides transparency and also does not require any change in BIOS or I/O device drivers in the future when moving to higher processor clock speeds.

As an example, a timing loop may be implemented as follows: The software performs a dummy I/O instruction to an unused I/O port. The hardware for the bus controller logic recognizes this I/O instruction and delays the termination of the I/O cycle by keeping RDY# or BRDY# deasserted for the appropriate amount of time.

16.6 Test Requirements

The electrical functionality of the OverDrive processor socket can be verified by fully testing the PC with a populated OverDrive processor socket. Intel recommends that the system be tested with all available OverDrive processors that are compatible with the OverDrive socket type and power supply voltage, to ensure that there are no BIOS issues. The BIOS requirements to maintain compatibility with all OverDrive processors are discussed in section 16.5, "BIOS and Software," of this document. All OverDrive processors undergo thorough application software compatibility testing prior to their introduction.

16.7 OverDrive® Processor Socket Pinout

Socket 3 can accept all 5V OverDrive processors and the future 3.3V Pentium OverDrive processor for the IntelDX4 processor. The socket 3 pinout is shown in Figure 16-4 and Figure 16-5. Socket 6 is discussed in section 16.8, "3.3V Socket Specification."

16.7.1 PIN DESCRIPTION

The signal pin descriptions for the OverDrive processor are identical to the pin descriptions for the Intel486 processor except for those shown in Table 16-5.

16.7.2 RESERVED PIN SPECIFICATION

Many pins in the OverDrive processor socket are defined as reserved (RES). The function of these pins is documented separately. These pins must not be connected unless they are used to implement these functions, as documented in the information available separately. To insure proper operation, pins marked as NC must be left unconnected as well. For more information contact Intel.

Section 16.7.4, "Shared Write-Back Pins," discusses the Pentium OverDrive processor compatibility with the Write-Back Enhanced IntelDX2 processor.

16.7.3 INC "Internal No Connect" PIN SPECIFICATIONS

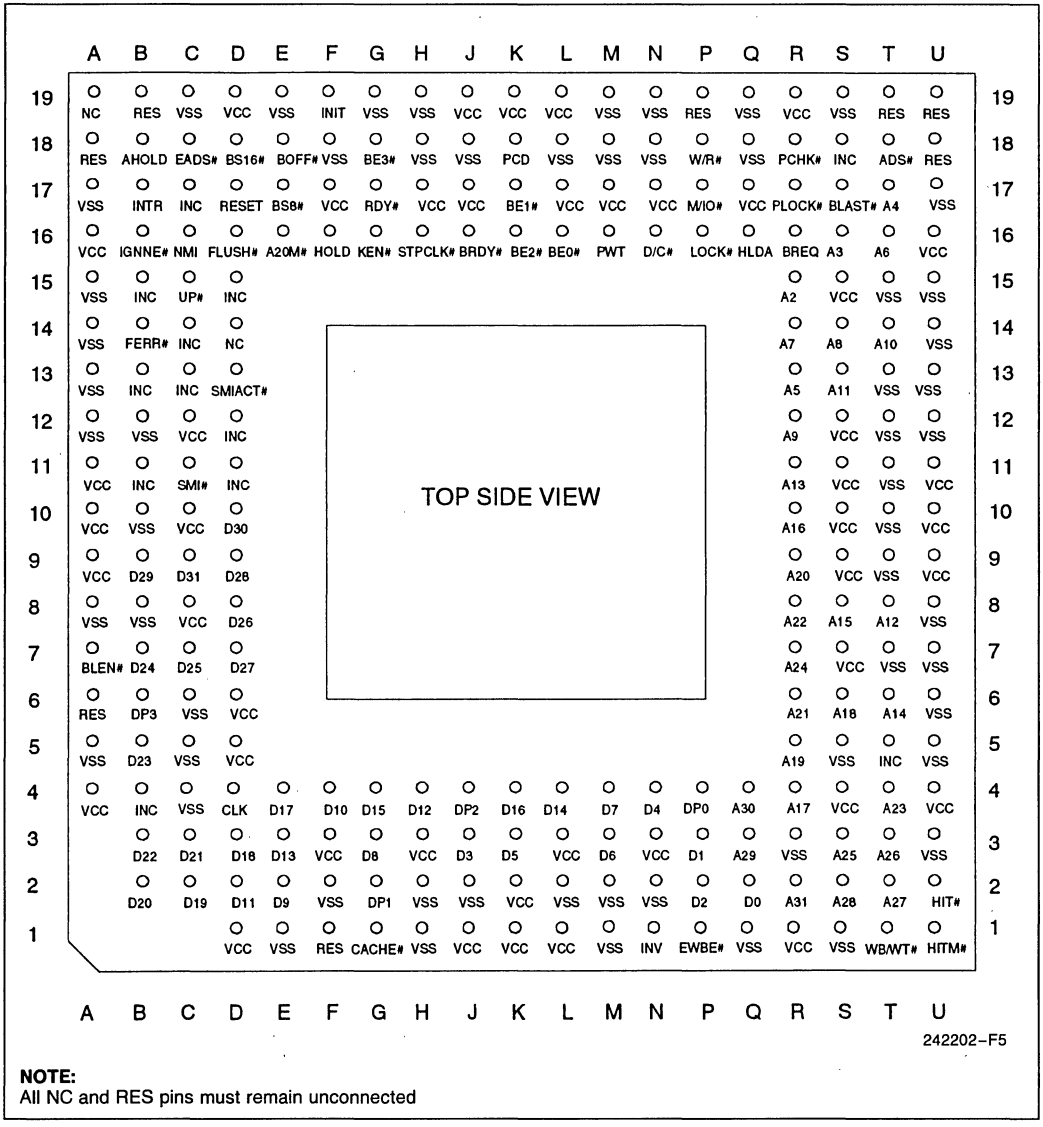
INC Pins are defined as Internal No-Connects. This means that the pin is not connected to the processor internally. Since the pin is inert and floating, it may be used in any manner seen fit, but must meet the **INC** pin specifications. In general, all **INC** pins have an intended use to implement a single processor socket system design. The **INC** pin will never be used for any other function. The 6 signals which use the INC pins to simplify single-socket board design are shown in Table 16-6.

16.7.4 SHARED WRITE-BACK PINS

There are several signals that the Pentium OverDrive processor socket has in common with the Write-Back Enhanced IntelDX2 processor, but which are located on different pins. An example of this would be the **HITM#** signal. On the Pentium OverDrive processor socket, it is located in the outer row of pins, while on the Write-Back Enhanced IntelDX2 processor, it is located on one of the inner rows. Single socket designs require that these signals be tied together so that the use of a jumper to reroute the signal is unnecessary. This is done through the use of the **INC** pin.

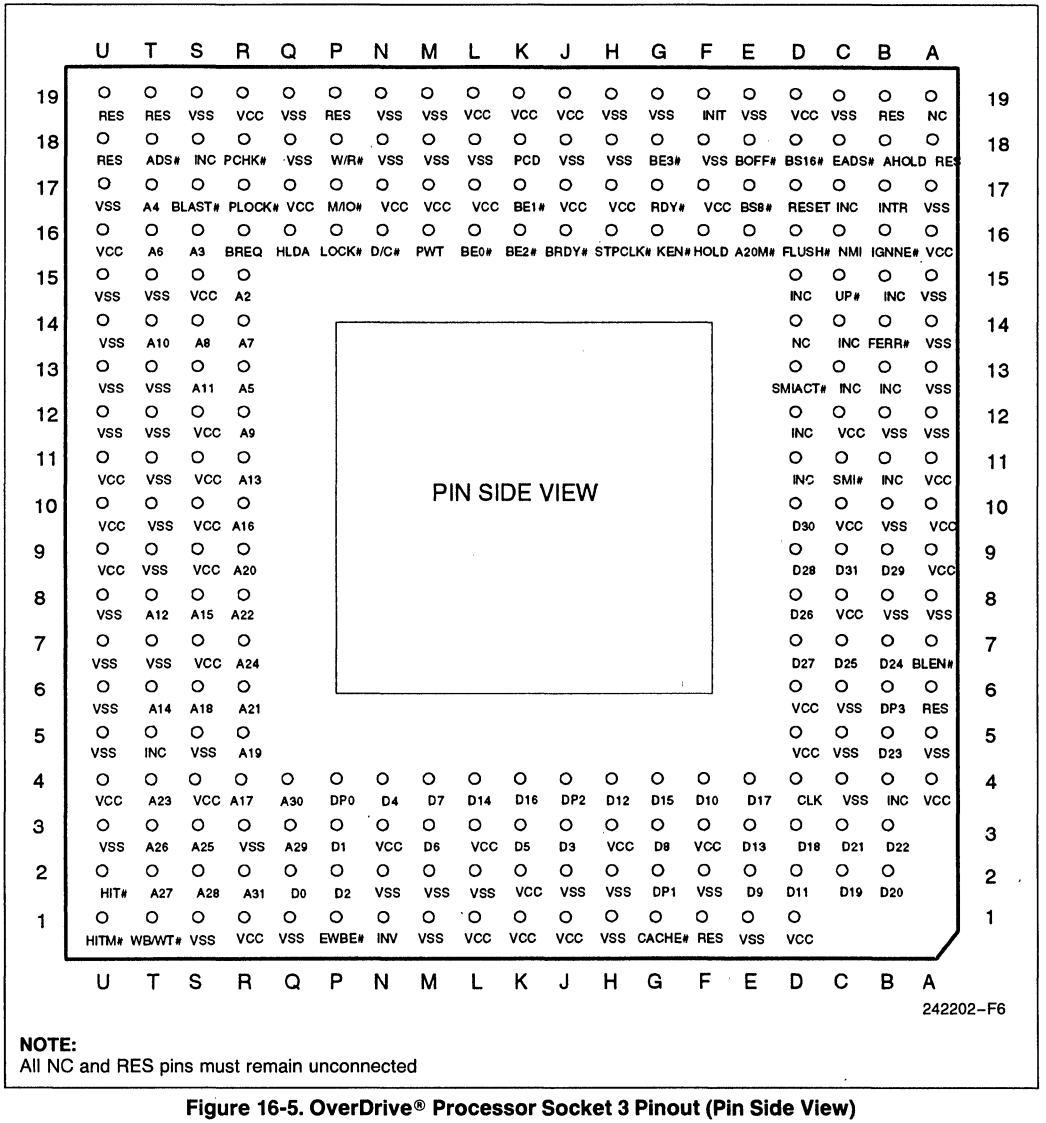
Figure 16-6 shows an example of how the INC pins shown in Table 16-6 should be connected together to allow single socket compatibility between the Write-Back Enhanced IntelDX2 processor and the OverDrive processor socket. The figure is provided as an example only and is not intended to be guide for how the signals should actually be routed on a motherboard.

16.7.5 PINOUT



NOTE:
All NC and RES pins must remain unconnected

Figure 16-4. OverDrive® Processor Socket 3 Pinout (Top Side View)



NOTE:
All NC and RES pins must remain unconnected

Figure 16-5. OverDrive® Processor Socket 3 Pinout (Pin Side View)

Table 16-4. OverDrive® Processor Socket Pin Cross Reference

Address		Data		Control		Control		Res(1)	V_{CC}		V_{SS}	
A2	R15	D0	Q2	A20M#	E16	PCD	K18	A6	A4(3)	N3	A5	M18
A3	S16	D1	P3	ADS#	T18	PCHK#	R18	A18	A9	N17	A8	M19
A4	T17	D2	P2	AHOLD	B18	PLOCK#	R17	B19	A10	Q17	A12	N2
A5	R13	D3	J3	BE0#	L16	PWT	M16	F1	A11	R1	A13	N18
A6	T16	D4	N4	BE1#	K17	RDY#	G17	P19	A16	R19	A14	N19
A7	R14	D5	K3	BE2#	K16	RESET	D17	T19	C8	S4	A15	Q1
A8	S14	D6	M3	BE3#	G18	SMI#	C11	U18	C10	S7	A17	Q18
A9	R12	D7	M4	BLAST#	S17	SMIACT#	D13	U19	C12	S9	B8	Q19
A10	T14	D8	G3	BLEN#	A7	STPCLK#	H16	**N/C(1)**	D1	S10	B10	R3
A11	S13	D9	E2	BOFF#	E18	UP#	C15	A19	D5	S11	B12	S1
A12	T8	D10	F4	BRDY#	J16	VOLDET(2)	T5	D14	D6	S12	C4	S5
A13	R11	D11	D2	BREQ	R16	W/R#	P18	**INC**	D19	S15	C5	S19
A14	T6	D12	H4	BS8#	E17	WB/WT#	T1	B11	F3	U4	C6	T7
A15	S8	D13	E3	BS16#	D18	**Position**		B13	F17	U9	C19	T9
A16	R10	D14	L4	CACHE#	G1	KEY	E5	B4	H3	U10	E1	T10
A17	R4	D15	G4	CLK	D4	KEY	A1	B15	H17	U11	E19	T11
A18	S6	D16	K4	CLKMUL(2)	S18	PLUG	A2	C17	J17	U16	F2	T12
A19	R5	D17	E4	D/C#	N16	PLUG	A3	C13	J19	**V_{CC5P}(3)**	F18	T13
A20	R9	D18	D3	DP0	P4	PLUG	B1	C14	K19	J1	G19	T15
A21	R6	D19	C2	DP1	G2	PLUG	C1	D11	L3	K1	H1	U3
A22	R8	D20	B2	DP2	J4	PLUG	E6	D12	L17	L1	H2	U5
A23	T4	D21	C3	DP3	B6	PLUG	E14	D15	L19	**V_{CC5}(5)**	H18	U6
A24	R7	D22	B3	EADS#	C18	PLUG	E15	S18	M17	K2	H19	U7
A25	S3	D23	B5	EWBE#	P1	PLUG	F5	T5			J2	U8
A26	T3	D24	B7	FERR#	B14	PLUG	F15				J18	U12
A27	T2	D25	C7	FLUSH#	D16	PLUG	P5				L2	U13
A28	S2	D26	D8	HIT#	U2	PLUG	P15				L18	U14
A29	Q3	D27	D7	HITM#	U1	PLUG	Q5				M1	U15
A30	Q4	D28	D9	HLDA	Q16	PLUG	Q6				M2	U17
A31	R2	D29	B9	HOLD	F16	PLUG	Q14					
		D30	D10	IGNNE#	B16	PLUG	Q15					
		D31	C9	INIT	F19							
				INTR	B17							
				INV	N1							
				KEN#	G16							
				LOCK#	P16							
				M/IO#	P17							
				NMI	C16							

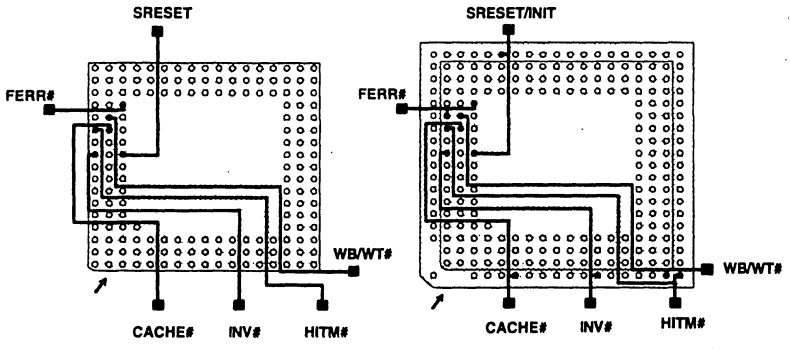
NOTES:
1. All RES pins are reserved for later use by Intel. To ensure proper operation of the microprocessor, all RES and N/C pins should be left unconnected. Please contact Intel for design information.
2. These pins are valid for the future Pentium® OverDrive® processor (3.3V) only.
3. These pins should be tied to 5 volts for both the Pentium OverDrive processor and the future Pentium OverDrive processors.
4. If designing for single socket compatibility with future Pentium OverDrive processors, pin K2 may be connected to V_{CC} via a circuit to limit the current through the pin. Please contact Intel for more information about compatibility with future Pentium OverDrive processors.
5. This pin may be connected to V_{CC} or left unconnected.

Table 16-5. OverDrive® Processor Socket Pin Description

Symbol	Type	Name and Function
Intel486™ PROCESSOR INTERFACE		
UP#	O	The *Upgrade Present* pin is used to signal Intel486 processor to float its outputs and stop driving the bus in a dual socket system design. It is active low and is never floated. UP# is driven low at power-up and remains active for the entire duration of the OverDrive® processor operation.
OverDrive® PROCESSOR INTERFACE		
V_{CC5P}	I	The V_{CC5P} pin supplies power to the OverDrive processor's fan/heatsink and should be connected to $+5V \pm 10\%$ regardless of the system design. Failure to connect V_{CC5P} to 5V will cause the component to overheat.
INIT	I	The INIT input will force the OverDrive processor to begin execution in a known state. The processor state after INIT is the same as the state after RESET, except that the internal caches, floating-point register and SMM base register retain whatever values they had prior to INIT. INIT may not be used in lieu of RESET after power-up.
KEY PIN		
KEY		The Key pin is an electrically non-functional pin which ensures correct orientation for the OverDrive processor. Socket plugs are also used to ensure correct orientation.

Table 16-6. Single Socket Compatibility Signals

Signal	Write-Back Enhanced IntelDX2™ Processor Signal Pin	Pentium® OverDrive® Processor Socket Signal Pin	Pentium OverDrive Processor Socket INC Pins
INV	A10	N1	B11
HITM#	A12	U1	B13
CACHE#	B12	G1	C13
WB/WT#	B13	T1	C14
INIT	C10	F19	D11
FERR#	C14	B14	D15



168 - Pin Write-Back Enhanced IntelDX2™ Processor Signal Layout

237-Pin Single Socket Design Using INC Pins

242202-F7

Figure 16-6. Sample Routing of INC Pins

16.8 3.3V Socket Specification

Socket 6 is a 235-pin socket and accepts 3.3V OverDrive processors only. The keying mechanism consists of a Key pin (A1) and five missing pins (B1, C1, A2, A3, and A19). Since it is designed for 3.3V OverDrive processors only, socket 6 must not accept the 169-pin OverDrive processors; therefore, the key pin, E5, is missing. To be effective as a keying mechanism, the locations in the socket corresponding to the six missing pins must be plugged.

In addition, the location of the pin 1 corner should be clearly marked on the motherboard.

Systems designed with the OverDrive processor socket 3 could use the future Pentium OverDrive processor for IntelDX4 processor-based systems, as well as the Pentium OverDrive processor. However, the OverDrive processor for IntelDX4 processor-based systems requires a 3.3V supply, while the Pentium OverDrive processor requires a 5V supply. Therefore, the supply voltage to the socket 3 (V_{CC}) must be 3.3V when it is used with the future Pentium OverDrive processor for IntelDX4 processor-based systems, and 5V when it is used with the Pentium OverDrive processor.

However, the fan/heatsink does require a 5V power supply, V_{CC5P}, as specified in Tables 16-4 and 16-5. To ensure adequate air circulation, the additional clearance specified in Table 16-1 must be provided.

16.9 DC/AC Specifications

The electrical specifications in this section represent the electrical interface of the OverDrive processor for Intel486 processor-based systems. The OverDrive processor will be compatible to the maximum ratings and AC Specifications of Intel486 processors. Tables 16-7 and 16-8 provide the unique DC Operating Conditions for the OverDrive processors.

Table 16-7. Pentium® OverDrive® Processor Socket (5V) DC Parametric Values

Functional Operating Range: V_{CC} = 5V ± 5%; T_{SINK} = 0°C to +85°C.

Symbol	Parameter	Min	Max	Unit	Notes
I_{CC}	Power Supply Current CLK = 25 MHz CLK = 33 MHz		 1900 2500	 mA mA	
C_{IN}	Input Capacitance		13	pF	
C_O	I/O or Output Capacitance		17	pF	
C_{CLK}	CLK Capacitance		15	pF	

Table 16-8. Future Pentium® OverDrive® Processor Socket (3.3V) DC Parametric Values

Functional Operating Range: V_{CC} = 3.3V ±0.3V; T_{SINK} = 0°C to +85°C.

Symbol	Parameter	Min	Max	Unit	Notes
I_{CC}	Power Supply Current		3000	mA	
I_{CC5}	Reference Supply Current		100	A	(Note 1)
I_{CC5P}	Fan/Heatsink Supply Current		200	mA	1_{watt} @ 5V
C_{IN}	Input Capacitance		13	pF	
C_O	I/O or Output Capacitance		17	pF	
C_{CLK}	CLK Capacitance		15	pF	

NOTES:

1. To avoid damaging the OverDrive® processor when the 3.3V power supply is accidentally connected to V_{SS}, the system must limit this current to less than 55 mA.

17.0 ELECTRICAL DATA

The following sections describe recommended electrical connections and electrical specifications for the Intel486 processor.

NOTE:
Unless otherwise specified, all values for electrical data for the IntelDX2 and IntelDX4 processors are also valid for the Write-Back Enhanced IntelDX2 and Write-Back Enhanced IntelDX4 processors, respectively.

17.1 Power and Grounding

17.1.1 POWER CONNECTIONS

The Intel486 processor is implemented in CHMOS technology and has modest power requirements. However, the high clock frequency output buffers can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, multiple V_{CC} and V_{SS} pins feed the Intel486 processor.

Power and ground connections must be made to all external V_{CC} and GND pins of the Intel486 processor. On the circuit board, all V_{CC} pins must be connected on a V_{CC} plane. All V_{SS} pins must be likewise connected on a GND plane.

17.1.2 Intel486™ PROCESSOR POWER DECOUPLING RECOMMENDATIONS

Liberal decoupling capacitance should be placed near the Intel486 processor. The Intel486 processor, driving its 32-bit parallel address and data buses at high frequencies, can cause transient power surges, particularly when driving large capacitive loads. Low inductance capacitors (i.e., surface-mount capacitors) and interconnects are recommended for the best high-frequency electrical performance. Inductance can be reduced by connecting capacitors directly to the V_{CC} and V_{SS} planes, with minimal trace length between the component pads and vias to the plane. These capacitors should be evenly distributed around each component on the V_{CC} power plane.

Capacitor values should be chosen to ensure they eliminate both low and high frequency noise components.

The recommendation for the Intel486 processor is 9 x 0.01 μF and 9 x 0.1 μF capacitors.

The power consumption can transition from a low level of power to a much higher level (or high to low power) very rapidly. A typical example would be entering or exiting the Stop Grant state. Another example would be executing a HALT instruction, causing the Intel486 processor to enter the Auto HALT Power Down state, or transitioning from HALT to the Normal state. All of these examples may cause abrupt changes in the power being consumed by the Intel486 processor. Bulk storage capacitors with a low ESR (Effective Series Resistance) in the 10 to 100 microfarad range are required to maintain a regulated supply voltage during the interval between the time the current load changes and the point that the regulated power supply output can react to the change in load. In order to reduce the ESR, it may be necessary to place several bulk storage capacitors in parallel. These capacitors should be placed near the Intel486 processor (on the processor power plane) to ensure that the supply voltage stays within specified limits during changes in the supply current while in operation.

17.1.3 V_{CC5} AND V_{CC} POWER SUPPLY REQUIREMENTS FOR THE IntelDX4 PROCESSOR

In mixed voltage systems that will be driving IntelDX4 processor inputs in excess of 3.3V, the V_{CC5} pin must be connected to the system 5V supply. In order to limit current flow into the V_{CC5} pin, there is a limit to the voltage differential between the V_{CC5} pin and the other V_{CC} pins. The voltage differential between the V_{CC5} pin of the IntelDX4 processor and its 3.3V V_{CC} pins should never exceed 2.25V. The 2.25V limit applies to power up, power down and steady state operation. Table 17-1 outlines this requirement.

Meeting this requirement ensures proper operation of the IntelDX4 processor and guarantees that the current draw into the V_{CC5} pin will not exceed the I_{CC5} specification (see section 17.3.1, "DC Specifications"). If the voltage difference requirement cannot be met due to system design limitations, then an alternate solution may be employed. A minimum of a 100Ω series resistor may be used to limit the current into the V_{CC5} pin. This resistor will ensure that current drawn by the V_{CC5} pin will not exceed the maximum rating of 55 mA for this pin (see section 17.2, "Maximum Ratings").

Table 17-1. Dual Power Supply Requirements for the IntelDX4™ Processor

Symbol	Parameter	Min	Max	Unit	Notes
VDIFF	V_{CC5}–V_{CC} Difference		2.25	V	V_{CC5} input should not exceed V_{CC} by more than 2.25V during power-up, power-down or during operation.

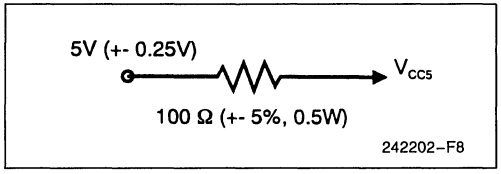


Figure 17-1. IntelDX4™ Processor V_{CC5} Current Limiting Resistor

Note that this resistor is not necessary if the system can guarantee that the voltage difference between V_{CC5} and V_{CC} is always limited to 2.25V, even during power up and power down.

In 3.3V-only systems and systems that will be driving all IntelDX4 processor inputs and I/Os from 3.3V logic, the V_{CC5} pin should be connected directly to the 3.3V V_{CC} plane. This will guarantee the voltage difference specification is met and will eliminate the current draw into the V_{CC5} pin. In a 3.3V-only system, the V_{CC5} may be connected to the 5V supply as described previously, as long as the voltage differential in Table 17-1 is met, and assuming the current drawn by the V_{CC5} pin is of little consequence to the system design.

17.1.4 SYSTEM CLOCK RECOMMENDATIONS

It is recommended that the CLK input to the Intel486 processor should not be driven until V_{CC} has reached its normal operating level (either 3.3V or 5V). The CLK input may be grounded or allowed to ramp with V_{CC} during this period. Once V_{CC} has reached its normal operating level, the Intel486 processor can handle the clock frequency for which it is specified and the oscillator/clock driver should have locked onto its desired frequency.

17.1.5 OTHER CONNECTION RECOMMENDATIONS

NC pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with other steppings of the Intel486 processor family.

INC (Internal No Connect) pins are not connected to any internal pad in Intel486 and OverDrive® processors. However, new signals are defined for the location of the INC pins in the Intel486 processor proliferations. All INC pins defined by Intel have a specific use for jumperless single socket compatibility with current and future processors. A system design could connect any signal to an INC pin without affecting the operation of the processor. However, the purpose of a specific INC pin should be understood before it is used. If not, the system design will sacrifice the ability to implement a jumperless (single socket) flexible motherboard.

For reliable operation, always connect unused inputs to an appropriate signal level. Active LOW inputs should be connected to V_{CC} through a pull-up resistor. Pull-ups in the range of 20 KΩ are recommended. Active HIGH inputs should be connected to GND.

17.2 Maximum Ratings

Table 17-2 is a stress rating only, and functional operation at the maximums is not guaranteed. Function operating conditions are given in Table 17-3 for 3.3V processor DC Specifications, Table 17-9 for 5V DC Specifications, Tables 17-17 through 17-20 for 3.3V processor AC specifications, and Tables 17-23 through 17-25 for 5V processor AC specifications.

Extended exposure to the Maximum Ratings may affect device reliability. Furthermore, although the Intel486 processor contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

Table 17-2. Absolute Maximum Ratings

Case Temperature under Bias	−65°C to +110°C
Storage Temperature	−65°C to +150°C
DC Voltage on Any Pin with Respect to Ground	−0.5 to V_{CC} + 0.5V −0.5 to V_{CC5} + 0.5V(1)
Supply Voltage with Respect to V_{SS}	V_{CC} −0.5V to +6.5V(2) V_{CC} −0.5V to +4.6V(1) V_{CC5}(1) −0.5V to +6.5V(1)
Transient Voltage on Any Input	−1.6V to V_{CC5} +1.6V(1,3)
Maximum Allowable Current Sink on V_{CC5}(1)	55 mA

NOTES:
1. For IntelDX4™ processor only.
2. All Intel486™ processors except IntelDX4 processor.
3. Maximum voltage on any pin with respect to ground is the lesser of Vcc5 + 1.6V or 6.5V for the IntelDX4 processor.

17.3 DC Specifications

17.3.1 3.3V DC CHARACTERISTICS

Table 17-3 is for Intel486 SX, Intel486 DX, IntelDX2™, Write-Back Enhanced IntelDX2, IntelDX4, and Write-Back Enhanced IntelDX4 processors.

Table 17-3. 3.3V DC Specifications

Functional operating range: V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 17); T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Typ	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3		+0.8	V	
V_{IH}	Input HIGH Voltage	2.0 2.0		V_{CC}+0.3 V_{CC5}+0.3	V	1 10
V_{IHC}	Input HIGH Voltage of CLK, CLK2	V_{CC}−0.6		V_{CC}+0.3	V	
V_{OL}	Output LOW Voltage I_{OL} = 2.0 mA I_{OL} = 100 µA			 0.40 0.20	 V V	 7
				0.45	V	
V_{OH}	Output HIGH Voltage I_{OH} = −2.0 mA I_{OH} = −100 µA	 2.4 V_{CC}−0.2			 V V	 16
I_{CC5}	V_{CC5} Leakage Current		15	300	µA	8, 9
I_{CCU}	UP# Active Supply Current		15	35 50	mA mA	2 2, 10
I_{LI}	Input Leakage Current			±15	µA	3
I_{IH}	Input Leakage Current			200 300	µA µA	4 15
I_{IL}	Input Leakage Current			−400	µA	5
I_{LO}	Output Leakage Current			±15	µA	
C_{IN}	Input Capacitance			10	pF	6
C_{OUT}	Output or I/O Capacitance			10 14	pF pF	6 6, 10
C_{CLK}	CLK Capacitance			6 12	pF pF	6 6, 10
I_{BHL}	Bus Hold Low Sustaining Current			17	µA	11
I_{BHH}	Bus Hold High Sustaining Current			−20	µA	12
I_{BHLO}	Bus Hold Low OverDrive® Current	210			µA	13
I_{BHHO}	Bus Hold High OverDrive Current	−350			µA	14

NOTES:
1. All inputs except CLK, CLK2. (For all Intel486 processors except the IntelDX4™ processor.)
2. When the processor is in Stop Grant state, the I_{CCU} of the host processor is less than 2 mA.
3. This parameter is for inputs without internal pull-ups or pull downs and $0V \leq V_{IN} \leq V_{CC}$.
4. This parameter is for inputs with internal pull-downs and $V_{IH} = 2.4V$.
5. This parameter is for inputs with internal pull-ups and $V_{IL} = 0.4V$.
6. $F_C = 1$ MHz; Not 100% tested.
7. For the IntelDX4 processor, this parameter is measured at: Address, Data, BEn = 4.0 mA
   Definition, Control = 5.0 mA
8. Typical values are not 100% tested.
9. This parameter is for $V_{CC5}-V_{CC} \leq 2.25V$. (IntelDX4 processor only.)
10. For the IntelDX4 processor only.
11. This is the maximum current the bus hold circuit can sink without raising the node above V_{IL}max. IB_{HL} should be measured after lowering V_{IN} to ground and then raising to V_{IL}max. ($V_{IN} = 0.8V$). (Write-Back Enhanced IntelDX2 processor only.)
12. This is the maximum current the bus hold circuit can source without lowering the node voltage below V_{IH}min. IB_{HH} should be measured after raising V_{IN} to V_{CC} (3.3V) and then lowering to V_{IH}min. ($V_{IN} = 2.0V$). (Write-Back Enhanced IntelDX2 processor only.)
13. An external driver must source at least I_{BHLO} to switch this node from low to high. ($V_{IN} \geq 1.3V$) (Write-Back Enhanced IntelDX2 processor only.)
14. An external driver must source at least I_{BHHO} to switch this node from high to low. ($V_{IN} \leq 1.3V$) (Write-Back Enhanced IntelDX2 processor only.)
15. This parameter is for inputs with pull-downs and $V_{IH} = 2.4V$. (Write-Back Enhanced IntelDX2 processor only.)
16. All Intel486 processors except the IntelDX4 processor.
17. V_{CC5} should be connected to $3.3V \pm 0.3V$ in 3.3V-only systems (IntelDX4 processor only.)

Table 17-4. 3.3V I_{CC} Values for Intel486™ SX Processor

Functional Operating Range: $V_{CC} = 3.3V \pm 0.3V$; $T_{CASE} = 0°C$ to $+85°C$

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	25 MHz 33 MHz		315 mA 415 mA	1
I_{CC} Active (Thermal Design)	25 MHz 33 MHz	220 mA 289 mA	292 mA 356 mA	2, 3, 4
I_{CC} Stop Grant	25 MHz 33 MHz	20 mA 25 mA	40 mA 50 mA	5
I_{CC} Stop Clock	0 MHz	100 μA	1 mA	6

Table 17-5. 3.3V I_{CC} Values for Intel486™ DX Processor
Functional Operating Range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	33 MHz		415 mA	1
I_{CC} Active (Thermal Design)	33 MHz	290 mA	383 mA	2, 3, 4
I_{CC} Stop Grant	33 MHz	25 mA	50 mA	5
I_{CC} Stop Clock	0 MHz	100 μA	1 mA	6

Table 17-6. 3.3V I_{CC} Values for IntelDX2™ Processor
Functional Operating Range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	40 MHz 50 MHz		450 mA 550 mA	1
I_{CC} Active (Thermal Design)	40 MHz 50 MHz	318 mA 395 mA	416 mA 507 mA	2, 3, 4
I_{CC} Stop Grant	40 MHz 50 MHz	20 mA 23 mA	40 mA 50 mA	5
I_{CC} Stop Clock	0 MHz	100 μA	1 mA	6

Table 17-7. 3.3V I_{CC} Values for Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processors
Functional Operating Range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	40 MHz 50 MHz		515 mA 630 mA	1
I_{CC} Active (Thermal Design)	40 MHz 50 MHz	309 mA 384 mA	475 mA 581 mA	2, 3, 4
I_{CC} Stop Grant	40 MHz 50 MHz	20 mA 23 mA	40 mA 50 mA	5
I_{CC} Stop Clock	0 MHz	100 μA	1 mA	6

Table 17-8. 3.3V I_{CC} Values for IntelDX4™ Processor

Functional Operating Range: V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 7); T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	100 MHz 75 MHz		1450 mA 1100 mA	1
I_{CC} Active (Thermal Design)	100 MHz 75 MHz	1075 mA 825 mA	1300 mA 975 mA	2, 3, 4
I_{CC} Stop Grant	100 MHz 75 MHz	50 mA 20 mA	100 mA 75 mA	5
I_{CC} Stop Clock	0 MHz	600 μA	1 mA	6

NOTES FOR TABLES 17-4 THROUGH 17-8:

1. This parameter is for proper power supply selection. It is measured using the worst-case instruction mix at V_{CC} = 3.6V. In order to support the OverDrive® processor, care should be taken to accommodate the Maximum Power Supply Current Value in section 16, "OverDrive® Processor Socket."
2. The maximum current column is for thermal design power dissipation. It is measured using the worst-case instruction mix at V_{CC} = 3.3V.
3. The typical current column is the typical operating current in a system. This value is measured in a system using a typical device at V_{CC} = 3.3V, running Microsoft Windows 3.1 at an idle condition. This typical value is dependent upon the specific system configuration.
4. Typical values are not 100% tested.
5. The I_{CC} Stop Grant specification refers to the I_{CC} value once the Intel486 processor enters the Stop Grant or Auto HALT Power-Down state.
6. The I_{CC} Stop Clock specification refers to the I_{CC} value once the processor enters the Stop Clock state. The V_{IH} and V_{IL} levels must be equal to V_{CC} and 0V, respectively, in order to meet the I_{CC} Stop Clock specifications.
7. V_{CC5} should be connected to 3.3V ±0.3V in 3.3V-only systems.

17.3.2 5V DC CHARACTERISTICS

Table 17-9 is for Intel486 SX, IntelSX2, Intel486 DX, IntelDX2 Processors, and Write-Back Enhanced IntelDX2 processors.

Table 17-9. 5V DC Specifications

Functional operating range: V_{CC} = 5V ± 0.25V; T_{CASE} = 0°C to +85°C

Symbol	Parameter	Min	Typ	Max	Unit	Notes
V_{IL}	Input LOW Voltage	−0.3		+0.8	V	
V_{IH}	Input HIGH Voltage	2.0		V_{CC}+0.3	V	
V_{OL}	Output LOW Voltage			0.45	V	1
V_{OH}	Output HIGH Voltage	2.4			V	2
I_{CCU}	UP# Active Supply Current		25	50	mA	6
I_{LI}	Input Leakage Current			±15	µA	3
I_{IH}	Input Leakage Current			200 300	µA µA	4 8
I_{IL}	Input Leakage Current			−400	µA	5
I_{LO}	Output Leakage Current			±15	µA	
C_{IN}	Input Capacitance PGA PQFP			 20 10	 pF pF	 7
C_{OUT}	Output or I/O Capacitance PGA PQFP			 20 10	 pF pF	 7
C_{CLK}	CLK Capacitance PGA PQFP			 20 6	 pF pF	 7
I_{BHL}	Bus Hold Low Sustaining Current			33	µA	9
I_{BHH}	Bus Hold High Sustaining Current			−80	µA	10
I_{BHLO}	Bus Hold Low OverDrive® Current	330			µA	11
I_{BHHO}	Bus Hold High OverDrive Current	−550			µA	12

4

NOTES:
1. This parameter is measured at: Address, Data, BEn 4.0 mA
   Definition, Control 5.0 mA
2. This parameter is measured at: Address, Data, BEn −1.0 mA
   Definition, Control −0.9 mA
3. This parameter is for inputs without pull-ups or pull-downs and $0V \leq V_{IN} \leq V_{CC}$.
4. This parameter is for inputs with pull-downs and $V_{IH} = 2.4V$.
5. This parameter is for inputs with pull-ups and $V_{IL} = 0.45V$.
6. When the processor is in Stop Grant state, the I_{CCU} of the host processor is less than 2 mA.
7. $F_C = 1$ MHz; Not 100% tested.
8. This parameter is for inputs with pull-downs and VIH = 2.4V. (SRESET pin only.)
9. This is the maximum current the bus hold circuit can sink without raising the node above V_{IL}max. IB_{HL} should be measured after lowering V_{IN} to ground and then raising to V_{IL}max. ($V_{IN} = 0.8V$) (Write-Back Enhanced IntelDX2 processor only.)
10. This is the maximum current the bus hold circuit can source without lowering the node voltage below V_{IH} min. IB_{HH} should be measured after raising V_{IN} to V_{CC} (5V) and then lowering to V_{IH} min. ($V_{IN} = 2.0V$) (Write-Back Enhanced IntelDX2 processor only.)
11. An external driver must source at least I_{BHLO} to switch this node from low to high. ($V_{IN} \geq 1.6V$) (Write-Back Enhanced IntelDX2 processor only.)
12. An external driver must source at least I_{BHHO} to switch this node from high to low. ($V_{IN} \leq 1.6V$) (Write-Back Enhanced IntelDX2 processor only.)

Table 17-10. 5V I_{CC} Values for Intel486™ SX Processor

Functional Operating Range: $V_{CC} = 5V \pm 0.25V$; $T_{CASE} = 0°C$ to $+85°C$

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	25 MHz 33 MHz		560 mA 685 mA	1
I_{CC} Active (Thermal Design)	25 MHz 33 MHz	378 mA 497 mA	535 mA 654 mA	2, 3, 4
I_{CC} Stop Grant	25 MHz 33 MHz	35 mA 40 mA	65 mA 80 mA	5
I_{CC} Stop Clock	0 MHz	200 μA	2 mA	6

Table 17-11. 5V I_{CC} Values for IntelSX2™ Processor

Functional Operating Range: $V_{CC} = 5V \pm 0.25V$; $T_{CASE} = 0°C$ to $+85°C$

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	50 MHz		855 mA	1
I_{CC} Active (Thermal Supply)	50 MHz	615 mA	815 mA	2, 3, 4
I_{CC} Stop Grant	50 MHz	35 mA	70 mA	5
I_{CC} Stop Clock	0 MHz	200 μA	2 mA	6

Table 17-12. 5V I_{CC} Values for Intel486™ DX Processor

Functional Operating Range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	33 MHz 50 MHz		630 mA 1000 mA	1
I_{CC} Active (Thermal Supply)	33 MHz 50 MHz	499 mA 800 mA	602 mA 956 mA	2, 3, 4
I_{CC} Stop Grant	33 MHz 50 MHz	40 mA N/A	80 mA N/A	5 7
I_{CC} Stop Clock	0 MHz	200 μA	2 mA	6, 7

Table 17-13. 5V I_{CC} Values for IntelDX2™ Processor

Functional Operating Range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	50 MHz 66 MHz		950 mA 1200 mA	1
I_{CC} Active (Thermal Supply)	50 MHz 66 MHz	680 mA 901 mA	906 mA 1145 mA	2, 3, 4
I_{CC} Stop Grant	50 MHz 66 MHz	35 mA 45 mA	70 mA 90 mA	5
I_{CC} Stop Clock	0 MHz	200 μA	2 mA	6

Table 17-14. 5V I_{CC} Values for Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processors

Functional Operating Range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C

Parameter	Operating Frequency	Typ	Maximum	Notes
I_{CC} Active (Power Supply)	50 MHz 66 MHz		1025 mA 1350 mA	1
I_{CC} Active (Thermal Supply)	50 MHz 66 MHz	659 mA 872 mA	928 mA 1287 mA	2, 3, 4
I_{CC} Stop Grant	50 MHz 66 MHz	35 mA 45 mA	70 mA 90 mA	5
I_{CC} Stop Clock	0 MHz	200 μA	2 mA	6

NOTES FOR TABLES 17-10 THROUGH 17-14:

1. This parameter is for proper power supply selection. It is measured using the worst-case instruction mix at V_{CC} = 5.25V.
2. The maximum current column is for thermal design power dissipation. It is measured using the worst-case instruction mix at V_{CC} = 5V.
3. The typical current column is the typical operating current in a system. This value is measured in a system using a typical device at V_{CC} = 5V, running Microsoft Windows 3.1 at an idle condition. This typical value is dependent upon the specific system configuration.
4. Typical values are not 100% tested.
5. The I_{CC} Stop Grant specification refers to the I_{CC} value once the Intel486 processor enters the Stop Grant or Auto HALT Power-Down state.
6. The I_{CC} Stop Clock specification refers to the I_{CC} value once the processor enters the Stop Clock state. The V_{IH} and V_{IL} levels must be equal to V_{CC} and 0V, respectively, in order to meet the I_{CC} Stop Clock specifications.
7. The 50 MHz Intel486 DX does not implement SL Technology and cannot utilize Stop Grant or Stop Clock functions.

17.3.3 EXTERNAL RESISTORS RECOMMENDED TO MINIMIZE LEAKAGE CURRENTS FOR THE WRITE-BACK ENHANCED IntelDX4™ AND WRITE-BACK ENHANCED IntelDX2™ PROCESSORS

The data bus and data parity pins of the Write-Back Enhanced Intel486 processors employ internal bus hold circuitry to maintain their previous logic level while in the Stop Grant state; external resistors are not required to prevent excessive current during the Stop Grant state for the Write-Back Enhanced IntelDX2 processors. See Table 17-15 for specifications of the bus hold circuitry. **If resistors are present, they should be strong enough to "flip" the logic level of the bus hold circuitry to minimize any potential DC paths (i.e., leakage currents. If external resistors are not strong enough to "flip" the logic level, the estimated leakage current on the data bus and the data bus parity pins is approximately 2 mA.)**

According to section 9.6.2, "Pin State During Stop Grant," data pins must be driven low to achieve the lowest possible power consumption. If the Write-Back Enhanced Intel486 processors are installed in an existing Intel486 processor socket, and the socket contains existing pull-down resistors, these resistors should be changed to the suggested values, listed in Table 17-16. The values listed are recommended to minimize leakage current.

Table 17-15. Write-Back Enhanced—IntelDX2™ Processors DC Keeper Specifications

Parameter	Description	3.3V	5V	Condition
I_{BHL}	Current Required to Sustain Bus Hold LOW	17 μA (max)	33 μA (max)	V_{IN} @ 0.8V
I_{BHH}	Current Required to Sustain Bus Hold HIGH	−20 μA (max)	−80 μA (max)	V_{IN} @ 2.0V
I_{BHLO}	Current Required to "Flip" Bus Hold HIGH	210 μA (min)	330 μA (min)	$V_{IN} \geq 1.3V$ @ $V_{CC}=3.3V$ $V_{IN} \geq 1.6V$ @ $V_{CC}=5V$
I_{BHHO}	Current Required to "Flip" Bus Hold LOW	−350 μA (min)	−550 μA (min)	$V_{IN} \leq 1.3V$ @ $V_{CC}=3.3V$ $V_{IN} \leq 1.6V$ @ $V_{CC}=5V$

Table 17-16. Write-Back Enhanced IntelDX4™ and Write-Back Enhanced IntelDX2™ Processors Recommended Values for Bus Keeper

V_{CC}	Calculation	Suggested
3.3V	$\frac{1.3V}{350\ \mu A} = 3.7\ K\Omega$	3 KΩ
5V	$\frac{1.6V}{550\ \mu A} = 2.9\ K\Omega$	2.7 KΩ

17.4 AC Specifications

The AC specifications given in the tables in this section consist of output delays, input setup requirements and input hold requirements. All AC specifications are relative to the rising edge of the input system clock (CLK) unless otherwise specified.

17.4.1 3.3V AC CHARACTERISTICS

Table 17-17 is for 25- and 33-MHz Intel486 SX, 33-MHz Intel486 DX, 40-MHz IntelDX2 (20-MHz Max.), 50-MHz IntelDX2 (25-MHz Max.), 40-MHz Write-Back Enhanced IntelDX2 (20-MHz Max.), and 50-MHz Write-Back Enhanced IntelDX2 Processors (25-MHz Max.).

Table 17-17. 3.3V AC Characteristics

Functional operating range: V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified.

Symbol	Parameter	Bus Speed						Unit	Figure	Notes
		20 MHz		25 MHz		33 MHz				
		Min	Max	Min	Max	Min	Max			
	Frequency	8	**20**	8	**25**	8	**33**	MHz		1
t_1	CLK Period	50	125	40	125	30	125	ns	17-2	
t_{1a}	CLK Period Stability		±250		±250		±250	ps	17-2	Adjacent clocks
t_2	CLK High Time	16		14		11		ns	17-2	at 2V
t_3	CLK Low Time	16		14		11		ns	17-2	at 0.8V
t_4	CLK Fall Time		6		4		3	ns	17-2	2V to 0.8V
t_5	CLK Rise Time		6		4		3	ns	17-2	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, CACHE#, HITM#, SMIACT#, FERR# Valid Delay	3	23	3	19	3	16	ns	17-6	2
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, CACHE# Float Delay		37		28		20	ns	17-7	3
t_8	PCHK# Valid Delay	3	28	3	24	3	22	ns	17-5	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	28	3	24	3	20	ns	17-6	
t_9	BLAST#, PLOCK# Float Delay		37		28		20	ns	17-7	3

Table 17-17. 3.3V AC Characteristics (Continued)

Functional operating range: V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 50 pF, unless otherwise specified.

Symbol	Parameter	Bus Speed						Unit	Figure	Notes
		20 MHz		25 MHz		33 MHz				
		Min	Max	Min	Max	Min	Max			
t_{10}	D0–D31, DP0–DP3 Write Data Valid Delay	3	26	3	20	3	19	ns	17-6	
t_{11}	D0–D31, DP0–DP3 Write Data Float Delay		37		28		20	ns	17-7	3
t_{12}	EADS#, INV Setup Time	10		8		6		ns	17-3	6
t_{13}	EADS#, INV Hold Time	3		3		3		ns	17-3	6
t_{14}	KEN#, BS16#, BS8#, WB/WT# Setup Time	10		8		6		ns	17-3	6
t_{15}	KEN#, BS16#, BS8#, WB/WT# Hold Time	3		3		3		ns	17-3	6
t_{16}	RDY#, BRDY# Setup Time	10		8		6		ns	17-4	
t_{17}	RDY#, BRDY# Hold Time	3		3		3		ns	17-4	
t_{18}	HOLD, AHOLD Setup Time	12		10		6		ns	17-3	
t_{18a}	BOFF# Setup Time	12		10		9		ns	17-3	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		3		3		ns	17-3	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Setup Time	12		10		6		ns	17-3	2
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Hold Time	3		3		3		ns	17-3	2
t_{22}	D0–D31, DP0–DP3, A4–A31 Read Setup Time	6		6		6		ns	17-3 17-4	
t_{23}	D0–D31, DP0–DP3, A4–A31 Read Hold Time	3		3		3		ns	17-3 17-4	

4

NOTES:
1. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant bus cycle protocol is used.
2. IGNNE# and FERR# are present only in the Intel486™ DX, IntelDX2™, and Write-Back Enhanced IntelDX2™ processors.
3. Not 100% tested; guaranteed by design characterization.
4. All timing specifications assume C_L = 50 pF. See capacitive derating charts for additional timing delays due to loading.
5. A reset pulsewidth of 15 CLK cycles is required for warm resets (RESET or SRESET). Power-up resets (cold resets) require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. CACHE#, WB/WT#, HITM#, and INV are present only in the Write-Back Enhanced IntelDX2 processor.

Tables 17-18 through 17-20 are for the IntelDX4 and Write-Back Enhanced IntelDX4 processors.

Table 17-18. 3.3V AC Characteristics for the 75/25-MHz IntelDX4™ Processor

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	CLK Frequency	8	25	MHz		2
t_1	CLK Period	40	125	ns	17-2	
t_{1a}	CLK Period Stability		±250	ps		3, 6
t_2	CLK High Time	14		ns	17-2	at 2V
t_3	CLK Low Time	14		ns	17-2	at 0.8V
t_4	CLK Fall Time		4	ns	17-2	2V to 0.8V
t_5	CLK Rise Time		4	ns	17-2	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, FERR#, BREQ, HLDA, CACHE#, HITM# Valid Delay	3	19	ns	17-6	7
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, CACHE# Float Delay		28	ns	17-7	3, 7
t_8	PCHK# Valid Delay	3	24	ns	17-5	
t_{8a}	BLAST#, PLOCK# SMIACT# Valid Delay	3	24	ns	17-6	
t_9	BLAST#, PLOCK# Float Delay		28	ns	17-7	3
t_{10}	D0–D31, DP0–3 Write Data Valid Delay	3	20	ns	17-6	

Table 17-18. 3.3V AC Characteristics for the 75/25-MHz IntelDX4™ Processor (Cont'd)

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{11}	D0–D31, DP0–3 Write Data Float Delay		28	ns	17-7	3
t_{12}	EADS#, INV Setup Time	8		ns	17-3	
t_{13}	EADS#, INV Hold Time	3		ns	17-3	7
t_{14}	KEN#, BS16#, BS8#, WB/WT# Setup Time	8		ns	17-3	7
t_{15}	KEN#, BS16#, BS8#, WB/WT# Hold Time	3		ns	17-3	7
t_{16}	RDY#, BRDY# Setup Time	8		ns	17-4	
t_{17}	RDY#, BRDY# Hold Time	3		ns	17-4	
t_{18}	HOLD, AHOLD Setup Time	8		ns	17-3	
t_{18a}	BOFF# Setup Time	8		ns	17-3	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		ns	17-3	
t_{20}	RESET, FLUSH#, A20M#, NMI, INTR, IGNNE# SRESET, STPCLK#, SMI# Setup Time	8		ns	17-3	5
t_{21}	RESET, FLUSH#, A20M#, NMI, INTR, IGNNE# SRESET, STPCLK#, SMI# Hold Time	3		ns	17-3	5
t_{22}	D0–D31, DP0–3, A4–A31 Read Setup Time	5		ns	17-3, 17-4	
t_{23}	D0–D31, DP0–3, A4–A31 Read Hold Time	3		ns	17-3, 17-4	

NOTES:
1. V_{CC5} should be connected to 3.3V ±0.3V in 3.3V-only systems.
2. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant Acknowledge protocol is used.
3. Not 100% tested; guaranteed by design characterization.
4. All timing specifications assume C_L = 50 pF. See capacitive derating charts for additional timing delays due to loading.
5. A reset pulsewidth of 15 CLK cycles is required for warm resets (RESET or SRESET). Power-up resets (cold resets) require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. For adjacent clocks; assumes frequency of operation is constant. STPCLK# input should be used to change frequency of operation.
7. CACHE#, WB/WT#, HITM#, and INV are present only in the Write-Back Enhanced IntelDX4 processor.

Table 17-19. 3.3V AC Characteristics for the 100/33-MHz IntelDX4™ Processors

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	CLK Frequency	8	33	MHz		2
t_1	CLK Period	30	125	ns	17-2	
t_{1a}	CLK Period Stability		±250	ps		3, 6
t_2	CLK High Time	11		ns	17-2	at 2V
t_3	CLK Low Time	11		ns	17-2	at 0.8V
t_4	CLK Fall Time		3	ns	17-2	2V to 0.8V
t_5	CLK Rise Time		3	ns	17-2	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, FERR#, BREQ, HLDA, CACHE#, HITM# Valid Delay	3	14	ns	17-6	7
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, CACHE# Float Delay		20	ns	17-7	3, 7
t_8	PCHK# Valid Delay	3	14	ns	17-5	
t_{8a}	BLAST#, PLOCK#, SMIACT# Valid Delay	3	14	ns	17-6	
t_9	BLAST#, PLOCK# Float Delay		20	ns	17-7	3
t_{10}	D0–D31, DP0–3 Write Data Valid Delay	3	14	ns	17-6	
t_{11}	D0–D31, DP0–3 Write Data Float Delay		20	ns	17-7	3
t_{12}	EADS#, INV Setup Time	5		ns	17-3	7
t_{13}	EADS#, INV Hold Time	3		ns	17-3	7
t_{14}	KEN#, BS16#, BS8#, WB/WT# Setup Time	5		ns	17-3	7
t_{15}	KEN#, BS16#, BS8#, WB/WT# Hold Time	3		ns	17-3	7
t_{16}	RDY#, BRDY# Setup Time	5		ns	17-4	
t_{17}	RDY#, BRDY# Hold Time	3		ns	17-4	
t_{18}	HOLD, AHOLD Setup Time	6		ns	17-3	
t_{18a}	BOFF# Setup Time	7		ns	17-3	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		ns	17-3	

Table 17-19. 3.3V AC Characteristics for the 100/33-MHz IntelDX4™ Processors (Cont'd)

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{20}	RESET, FLUSH#, A20M#, NMI, INTR, IGNNE#, SRESET, STPCLK#, SMI# Setup Time	5		ns	17-3	5
t_{21}	RESET, FLUSH#, A20M#, NMI, INTR, IGNNE#, SRESET, STPCLK#, SMI# Hold Time	3		ns	17-3	5
t_{22}	D0–D31, DP0–3, A4–A31 Read Setup Time	5		ns	17-3, 17-4	
t_{23}	D0–D31, DP0–3, A4–A31 Read Hold Time	3		ns	17-3, 17-4	

NOTES:
1. V_{CC5} should be connected to 3.3V ±0.3V in 3.3V-only systems.
2. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant Acknowledge protocol is used.
3. Not 100% tested; guaranteed by design characterization.
4. All timing specifications assume C_L = 50 pF. See capacitive derating charts for additional timing delays due to loading.
5. A reset pulsewidth of 15 CLK cycles is required for warm resets (RESET or SRESET). Power-up resets (cold resets) require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. For adjacent clocks; assumes frequency of operation is constant. STPCLK# input should be used to change frequency of operation.
7. CACHE#, WB/WT#, HITM#, and INV are present only in the Write-Back Enhanced IntelDX4 processor.

Table 17-20. 3.3V AC Characteristics for the 100/50-MHz IntelDX4™ Processor

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	CLK Frequency	16	50	MHz		2
t_1	CLK Period	20	62.5	ns	17-2	
t_{1a}	CLK Period Stability		±250	ps		3, 6
t_2	CLK High Time	7		ns	17-2	at 2V
t_3	CLK Low Time	7		ns	17-2	at 0.8V
t_4	CLK Fall Time		2	ns	17-2	2V to 0.8V
t_5	CLK Rise Time		2	ns	17-2	0.8V to 2V
t_{6a}	A20–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, FERR#, BREQ, HLDA, CACHE#, HITM# Valid Delay	2	12	ns	17-6	7
t_{6b}	A2–A19	2	10.5	ns	17-6	
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, CACHE# Float Delay		18	ns	17-7	3, 7
t_8	PCHK# Valid Delay	2	14	ns	17-5	
t_{8a}	BLAST#, PLOCK#, SMIACT# Valid Delay	2	12	ns	17-6	
t_9	BLAST#, PLOCK# Float Delay		18	ns	17-7	3
t_{10}	D0–D31, DP0–3 Write Data Valid Delay	3	12	ns	17-6	
t_{11}	D0–D31, DP0–3 Write Data Float Delay		18	ns	17-7	3
t_{12}	EADS#, INV Setup Time	5		ns	17-3	7
t_{13}	EADS#, INV Hold Time	2		ns	17-3	7
t_{14}	KEN#, BS16#, BS8#, WB/WT# Setup Time	5		ns	17-3	7
t_{15}	KEN#, BS16#, BS8#, WB/WT# Hold Time	2		ns	17-3	7
t_{16}	RDY#, BRDY# Setup Time	5		ns	17-4	
t_{17}	RDY#, BRDY# Hold Time	2		ns	17-4	
t_{18}	HOLD, AHOLD, Setup Time	5		ns	17-3	
t_{18a}	BOFF# Setup Time	5		ns	17-3	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	2		ns	17-3	

Table 17-20. 3.3V AC Characteristics for the 100/50-MHz IntelDX4™ Processor (Cont'd)

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{20}	RESET, FLUSH#, A20M#, NMI, INTR, IGNNE#, SRESET, STPCLK#, SMI# Setup Time	5		ns	17-3	5
t_{21}	RESET, FLUSH#, A20M#, NMI, INTR, IGNNE#, SRESET, STPCLK#, SMI# Hold Time	2		ns	17-3	5
t_{22}	D0–D31, DP0–3, A4–A31 Read Setup Time	4		ns	17-3, 17-4	
t_{23}	D0–D31, DP0–3, A4–A31 Read Hold Time	2		ns	17-3, 17-4	

NOTES:
1. V_{CC5} should be connected to 3.3V ±0.3V in 3.3V-only systems.
2. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant Acknowledge protocol is used.
3. Not 100% tested; guaranteed by design characterization.
4. All timing specifications assume C_L = 0 pF. I/O buffer modeling should be used to calculate additional timing delays due to loading.
5. A reset pulsewidth of 15 CLK cycles is required for warm resets (RESET or SRESET). Power-up resets (cold resets) require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. For adjacent clocks; assumes frequency of operation is constant. STPCLK# input should be used to change frequency of operation.
7. CACHE#, WB/WT#, HITM#, and INV are present only in the Write-Back Enhanced IntelDX4 processor.

Table 17-21. 3.3V Intel486™ Processor AC Specifications for the Test Access Port (All Intel486 Processors and Frequencies except the IntelDX4™ Processors)

V_{CC} = 3.3V ±0.3V; T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Notes
t_{24}	TCK Frequency		8	MHz	1
t_{25}	TCK Period	125		ns	
t_{26}	TCK High Time	40		ns	at 2.0V
t_{27}	TCK Low Time	40		ns	at 0.8V
t_{28}	TCK Rise Time		8	ns	2
t_{29}	TCK Fall Time		8	ns	2
t_{30}	TDI, TMS Setup Time	8		ns	3
t_{31}	TDI, TMS Hold Time	10		ns	3
t_{32}	TDO Valid Delay	3	30	ns	3
t_{33}	TDO Float Delay		36	ns	3
t_{34}	All Outputs (Non-Test) Valid Delay	3	30	ns	3
t_{35}	All Outputs (Non-Test) Float Delay		36	ns	3
t_{36}	All Inputs (Non-Test) Setup Time	8		ns	3
t_{37}	All Inputs (Non-Test) Hold Time	10		ns	3

NOTES:
1. TCK period ≤ CLK period.
2. Rise/Fall times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10-ns increase in TCK period.
3. Parameters t_{30}–t_{37} are measured from TCK.
4. Refer to Figure 17-8 for signal waveforms.

Table 17-22. 3.3V IntelDX4™ Processor AC Specifications for the Test Access Port (All IntelDX4 Processor Frequencies)

V_{CC} = 3.3V ±0.3V; V_{CC5} = 5V ±0.25V (Note 1); T_{CASE} = 0°C to +85°C; C_L = 0 pF

Symbol	Parameter	Min	Max	Unit	Figure
t_{24}	TCK Frequency		25	MHz	
t_{25}	TCK Period	40		ns	
t_{26}	TCK High Time	10		ns	
t_{27}	TCK Low Time	10		ns	
t_{28}	TCK Rise Time		4	ns	
t_{29}	TCK Fall Time		4	ns	
t_{30}	TDI, TMS Setup Time	8		ns	17-8
t_{31}	TDI, TMS Hold Time	7		ns	17-8
t_{32}	TDO Valid Delay	3	25	ns	17-8
t_{33}	TDO Float Delay		30	ns	
t_{34}	All Outputs (Non-Test) Valid Delay	3	25	ns	17-8
t_{35}	All Outputs (Non-Test) Float Delay		36	ns	17-8
t_{36}	All Inputs (Non-Test) Setup Time	8		ns	17-8
t_{37}	All Inputs (Non-Test) Hold Time	7		ns	17-8

NOTES:
1. V_{CC5} should be connected to 3.3V ± 0.3V in 3.3V-only systems.
2. All inputs and outputs are TTL Level.
3. Rise/Fall times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10-ns increase in TCK period.
4. TCK period ≤ CLK period.
5. Parameters t_{30}–t_{37} are measured from TCK.

17.4.2 5V AC CHARACTERISTICS

Table 17-23 is for 25- and 33-MHz Intel486 SX, 33-MHz Intel486 DX, 50-MHz IntelSX2 (25-MHz Max.), 50-MHz IntelDX2 (25-MHz Max.), 66-MHz IntelDX2 (33-MHz Max.), 50-MHz Write-Back Enhanced IntelDX2 (25-MHz Max.) and 66-MHz Write-Back Enhanced IntelDX2 (33-MHz Max.) processors.

Table 17-23. 5V AC Characteristics

Functional operating range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C; C_L = 50 pF unless otherwise specified. (See also Table 17-24).

Symbol	Parameter	Bus Speed				Unit	Figure	Notes
		25 MHz		33 MHz				
		Min	Max	Min	Max			
	Frequency	8	**25**	8	**33**	MHz		1
t_1	CLK Period	40	125	30	125	ns	17-2	
t_{1a}	CLK Period Stability		±250		±250	ps	17-2	Adjacent clocks
t_2	CLK High Time	14		11		ns	17-2	at 2V
t_3	CLK Low Time	14		11		ns	17-2	at 0.8V
t_4	CLK Fall Time		4		3	ns	17-2	2V to 0.8V
t_5	CLK Rise Time		4		3	ns	17-2	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT#, FERR#, CACHE#, HITM# Valid Delay	3	19	3	16	ns	17-6	3, 4
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, CACHE# Float Delay		28		20	ns	17-7	2, 4
t_8	PCHK# Valid Delay	3	24	3	22	ns	17-5	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	24	3	20	ns	17-6	
t_9	BLAST#, PLOCK# Float Delay		28		20	ns	17-7	2
t_{10}	D0–D31, DP0–DP3 Write Data Valid Delay	3	20	3	18	ns	17-6	
t_{11}	D0–D31, DP0–DP3 Write Data Float Delay		28		20	ns	17-7	2
t_{12}	EADS#, INV Setup Time	8		5		ns	17-3	4
t_{13}	EADS#, INV Hold Time	3		3		ns	17-3	4
t_{14}	KEN#, BS16#, BS8#, WB/WT# Setup Time	8		5		ns	17-3	4

Table 17-23. 5V AC Characteristics (Cont'd)

Functional operating range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C; C_L = 50 pF unless otherwise specified. (See also Table 17-24).

Symbol	Parameter	Bus Speed				Unit	Figure	Notes
		25 MHz		33 MHz				
		Min	Max	Min	Max			
t_{15}	KEN#, BS16#, BS8#, WB/WT# Hold Time	3		3		ns	17-3	4
t_{16}	RDY#, BRDY# Setup Time	8		5		ns	17-4	
t_{17}	RDY#, BRDY# Hold Time	3		3		ns	17-4	
t_{18}	HOLD, AHOLD Setup Time	10		6		ns	17-3	
t_{18a}	BOFF# Setup Time	10		8		ns	17-3	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		3		ns	17-3	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Setup Time	10		5		ns	17-3	3
t_{21}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Hold Time	3		3		ns	17-3	3
t_{22}	D0–D31, DP0–DP3, A4–A31 Read Setup Time	5		5		ns	17-3, 17-4	
t_{23}	D0–D31, DP0–DP3, A4–A31 Read Hold Time	3		3		ns	17-3, 17-4	

NOTES:
1. 0-MHz operation is guaranteed when the STPCLK# and Stop Grant bus cycle protocol is used.
2. Not 100% tested, guaranteed by design characterization.
3. IGNNE# and FERR# are present only in the Intel486™ DX, IntelDX2™, and Write-Back Enhanced IntelDX2 processors.
4. CACHE#, WB/WT#, HITM#, and INV are present only in the Write-Back Enhanced IntelDX2 processor.

The following specifications are different for existing IntelSX2, IntelDX2 and Write-Back Enhanced IntelDX2 processors. A system board that will support all of the Intel486 processors should be designed to the worst-case specifications of the 25- and 33-MHz local bus timings.

Table 17-24 is for 50-MHz IntelSX2 (25-MHz Max.), 50-MHz IntelDX2 (25-MHz Max.), 66-MHz IntelDX2 (33-MHz Max.), 50-MHz Write-Back Enhanced IntelDX2 (25-MHz Max.) and 66-MHz Write-Back Enhanced IntelDX2 (33-MHz Max.) processors.

Table 17-24. 5V AC Characteristics

Functional operating range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C; C_L = 50 pF unless otherwise specified. (See also Table 17-23).

Symbol	Parameter	Bus Speed				Unit	Notes
		25 MHz		33 MHz			
		Min	Max	Min	Max		
	Frequency		50		66	MHz	
	CLK Frequency	8	25	8	33	MHz	
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, SMIACT#, FERR#, CACHE#, HITM# Valid Delay				14	ns	3, 4
t_8	PCHK# Valid Delay				14	ns	
t_{8a}	BLAST#, PLOCK# Valid Delay				14	ns	
t_{10}	D0–D31, DP0–DP3 Write Data Valid Delay				14	ns	
t_{18}	HOLD, AHOLD Setup Time	8				ns	
t_{18a}	BOFF# Setup Time	8		7		ns	
t_{20}	FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, RESET, IGNNE# Setup Time	8				ns	

NOTES:
1. 0-MHz operation is guaranteed when the STPCLK# and STOP GRANT bus cycle protocol is used.
2. Not 100% tested; guaranteed by design characterization.
3. IGNNE# and FERR# are present in the IntelDX2™ and Write-Back Enhanced IntelDX2 processors only.
4. CACHE#, WB/WT#, HITM#, and INV are present only in the Write-Back Enhanced IntelDX2 processor.

Table 17-25. 5V AC Characteristics 50-MHz Intel486™ DX Processors

Functional operating range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C; C_L = (Note 1).

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	16	**50**	MHz		1
t_1	CLK Period	20	62.5	ns	17-2	
t_{1a}	CLK Period Stability		±250	ps	17-2	Adjacent clocks
t_2	CLK High Time	7		ns	17-2	at 2V
t_3	CLK Low Time	7		ns	17-2	at 0.8V
t_4	CLK Fall Time		2	ns	17-2	2V to 0.8V
t_5	CLK Rise Time		2	ns	17-2	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA, FERR# Valid Delay	3	12	ns	17-6	
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA Float Delay		18	ns	17-7	2
t_8	PCHK# Valid Delay	3	14	ns	17-5	
t_{8a}	BLAST#, PLOCK# Valid Delay	3	12	ns	17-6	
t_9	BLAST#, PLOCK# Float Delay		18	ns	17-7	2
t_{10}	D0–D31, DP0–DP3 Write Data Valid Delay	3	12	ns	17-6	
t_{11}	D0–D31, DP0–DP3 Write Data Float Delay		18	ns	17-7	2
t_{12}	EADS# Setup Time	5		ns	17-3	
t_{13}	EADS# Hold Time	2		ns	17-3	
t_{14}	KEN#, BS16#, BS8# Setup Time	5		ns	17-3	
t_{15}	KEN#, BS16#, BS8# Hold Time	2		ns	17-3	
t_{16}	RDY#, BRDY# Setup Time	5		ns	17-4	
t_{17}	RDY#, BRDY# Hold Time	2		ns	17-4	
t_{18}	HOLD, AHOLD, BOFF# Setup Time	5		ns	17-3	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	2		ns	17-3	
t_{20}	FLUSH#, A20M#, NMI, INTR, RESET, IGNNE# Setup Time	5		ns	17-3	
t_{21}	FLUSH#, A20M#, NMI, INTR, RESET, IGNNE# Hold Time	2		ns	17-3	

Table 17-25. 5V AC Characteristics 50-MHz Intel486™ DX Processors (Cont'd)

Functional operating range: V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C; C_L = (Note 1).

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{22}	D0–D31, DP0–DP3, A4–A31 Read Setup Time	4		ns	17-3, 17-4	
t_{23}	D0–D31, DP0–DP3, A4–A31 Read Hold Time	2		ns	17-3, 17-4	

NOTES:
1. Specifications assume C_L = 0 pF. I/O buffer model must be used to determine delays due to loading (trace and component). First order I/O buffer models for the Intel486™ processor are available. Contact Intel for the latest release.
2. Not 100% tested; guaranteed by design characterization.

Table 17-26. 5V Intel486™ Processor AC Specifications for the Test Access Port (All Processors and Frequencies)

V_{CC} = 5V ±0.25V; T_{CASE} = 0°C to +85°C; C_L = 50 pF

Symbol	Parameter	Min	Max	Unit	Notes
t_{24}	TCK Frequency		8	MHz	1
t_{25}	TCK Period	125		ns	
t_{26}	TCK High Time	40		ns	at 2.0V
t_{27}	TCK Low Time	40		ns	at 0.8V
t_{28}	TCK Rise Time		8	ns	2
t_{29}	TCK Fall Time		8	ns	2
t_{30}	TDI, TMS Setup Time	8		ns	3
t_{31}	TDI, TMS Hold Time	10		ns	3
t_{32}	TDO Valid Delay	3	30	ns	3
t_{33}	TDO Float Delay		36	ns	3
t_{34}	All Outputs (Non-Test) Valid Delay	3	30	ns	3
t_{35}	All Outputs (Non-Test) Float Delay		36	ns	3
t_{36}	All Inputs (Non-Test) Setup Time	8		ns	3
t_{37}	All Inputs (Non-Test) Hold Time	10		ns	3

NOTES:
1a. V_{CC} should be connected to 3.3V ±0.3V in 3.3V-only systems.
1. TCK period ≤ CLK period.
2. Rise/Fall times are measured between 0.8V and 2.0V. Rise/Fall times can be relaxed by 1 ns per 10-ns increase in TCK period.
3. Parameters t_{30}–t_{37} are measured from TCK.
4. Refer to Figure 17-8 for signal waveforms.

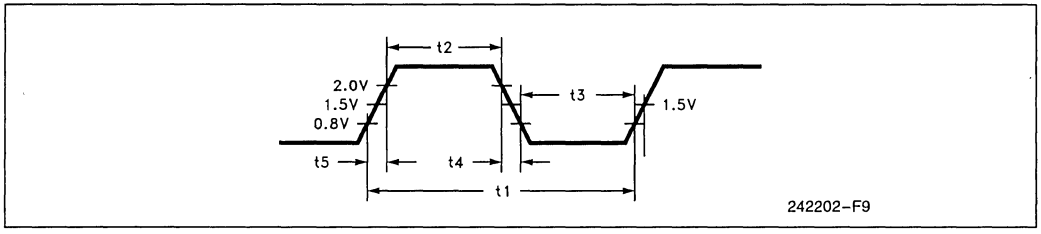


Figure 17-2. CLK Waveforms

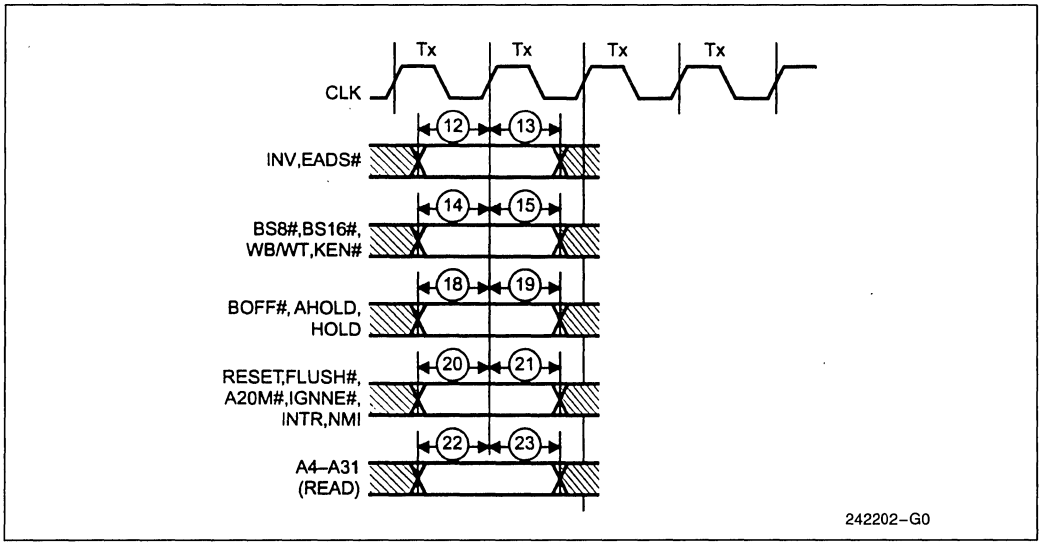


Figure 17-3. Input Setup and Hold Timing

4

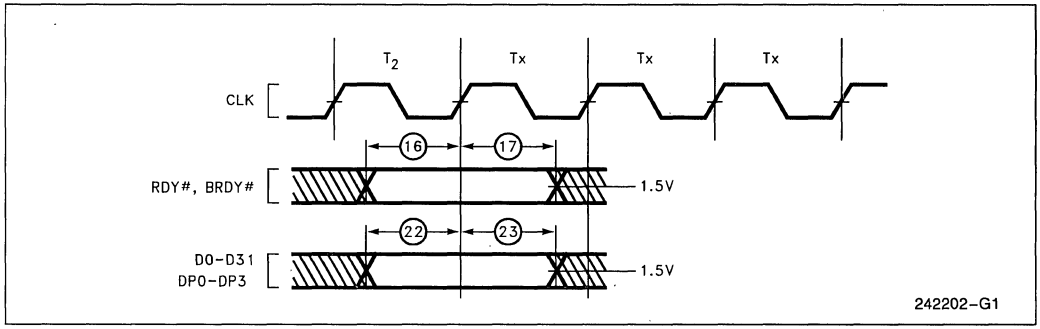


Figure 17-4. Input Setup and Hold Timing

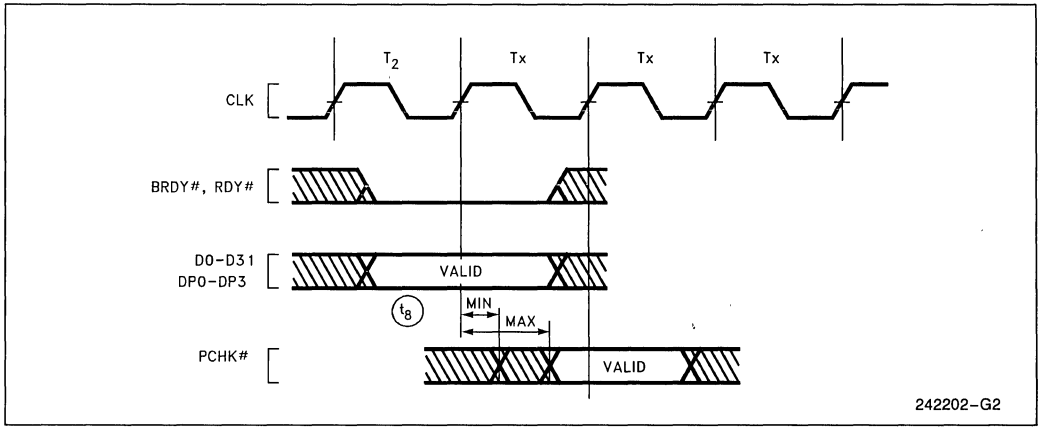


Figure 17-5. PCHK# Valid Delay Timing

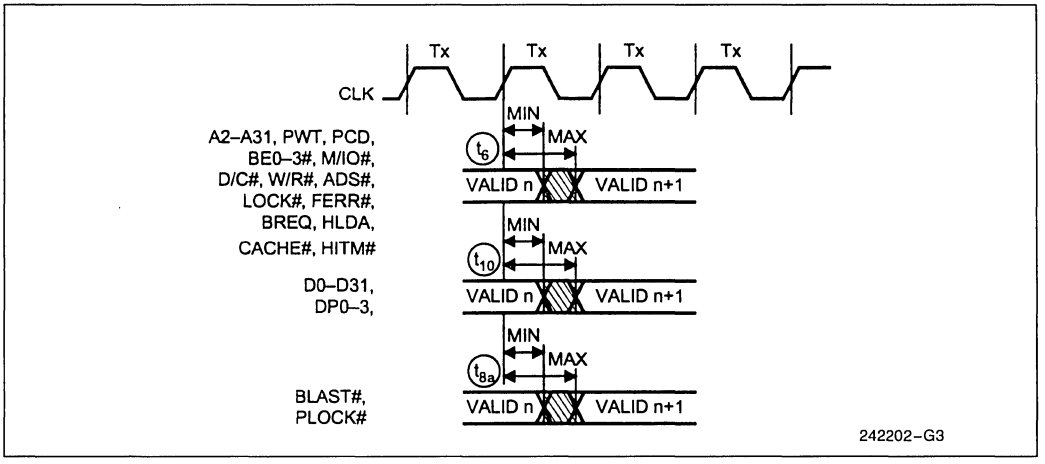


Figure 17-6. Output Valid Delay Timing

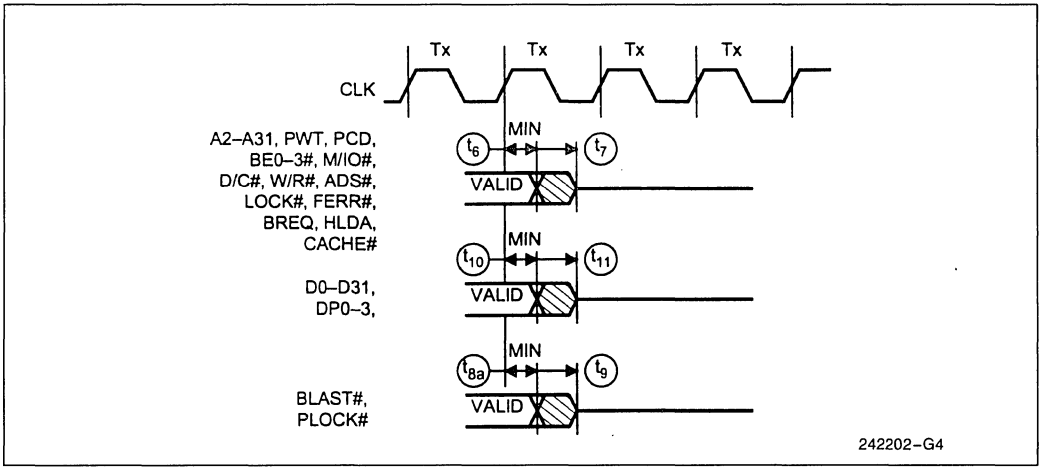


Figure 17-7. Maximum Float Delay Timing

4

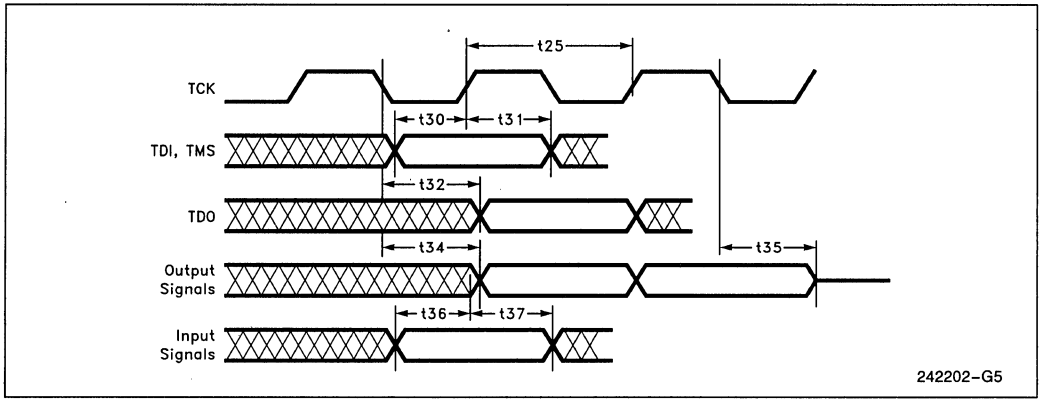
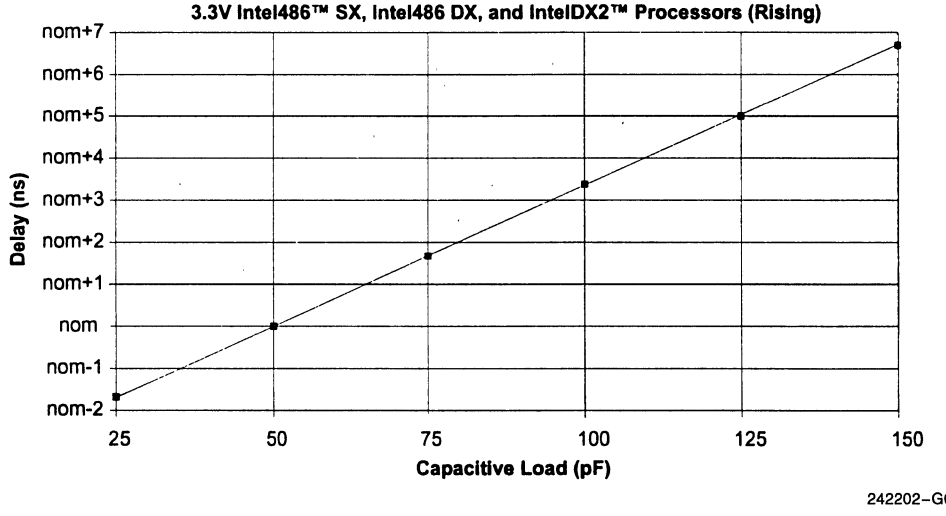


Figure 17-8. Test Signal Timing Diagram

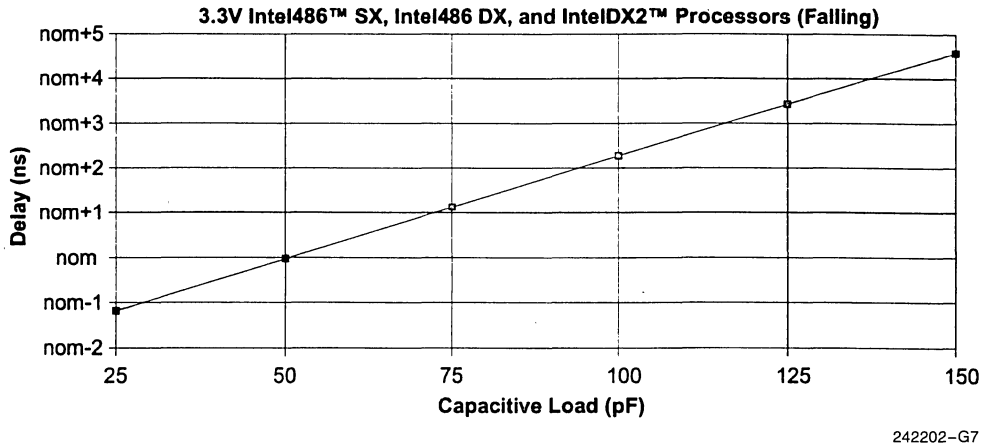
17.5 Capacitive Derating Curves

The capacitive derating curves illustrate output delay versus capacitive load for 3.3V and 5V Intel486 processors. The derating curves show the delays for the rising and falling edges under worst-case conditions. Figure 17-9 and Figure 17-10 apply to all 3.3V Intel486 SX, Intel486 DX, and IntelDX2 processors. Figure 17-11 and Figure 17-12 apply to 5V Intel486 DX and IntelDX2 processors. Figure 17-13 and Figure 17-14 apply to 5V Intel486 SX and IntelSX2 processors. Figures 17-15 through 17-17 apply to the IntelDX4 processor and Figures 17-18 through 17-20 apply to the Write-Back Enhanced IntelDX4 processor. The figures apply to all frequencies specified for each corresponding product. Refer to Appendix C for bus frequencies above 33 MHz for Intel486 processors.



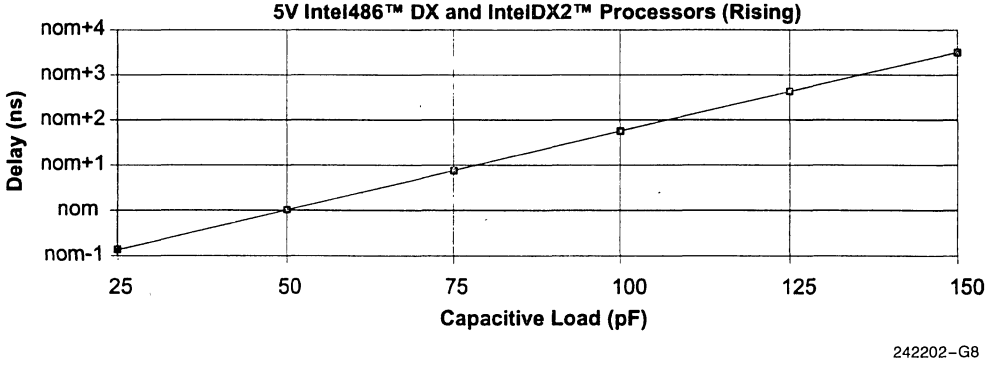
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17-9. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



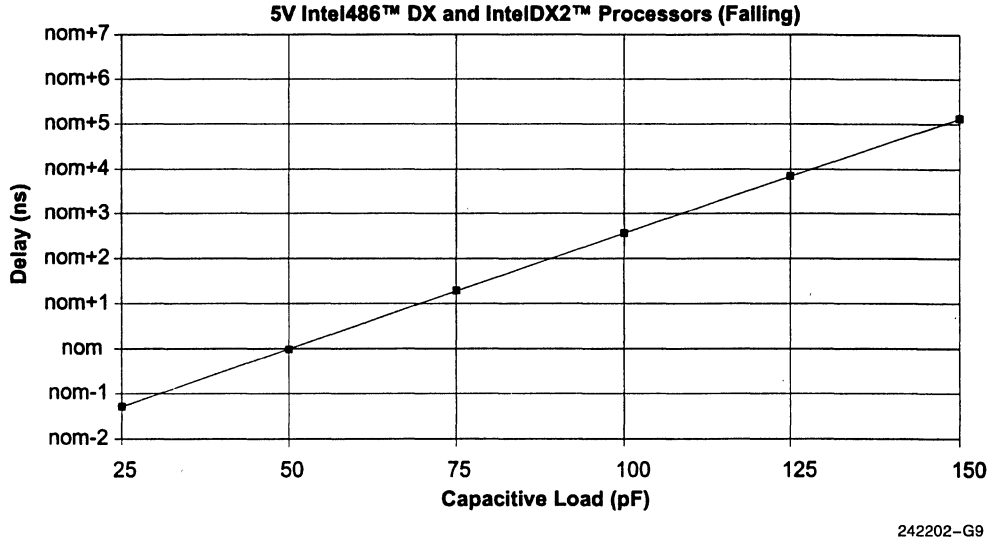
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17-10. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition



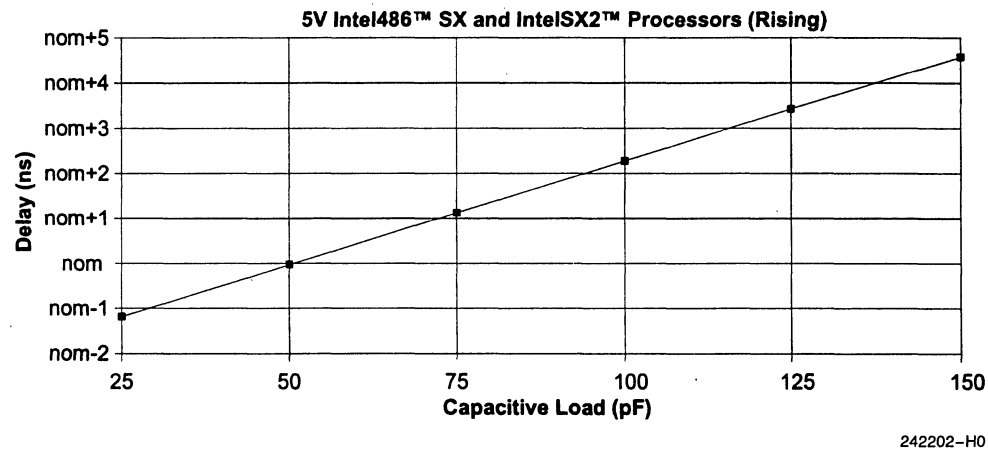
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17-11. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



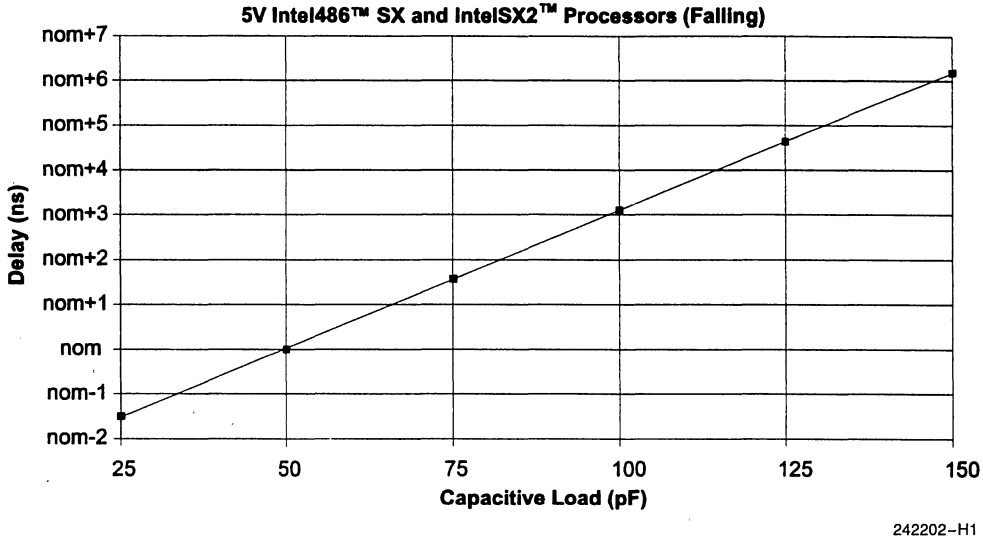
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17-12. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17-13. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17-14. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition

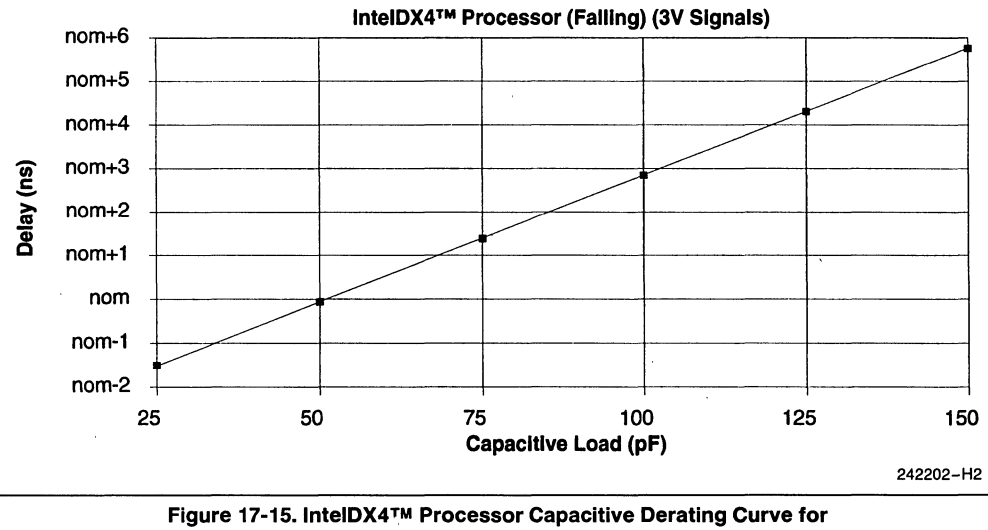


Figure 17-15. IntelDX4™ Processor Capacitive Derating Curve for High-to-Low Transitions (3V Signals)

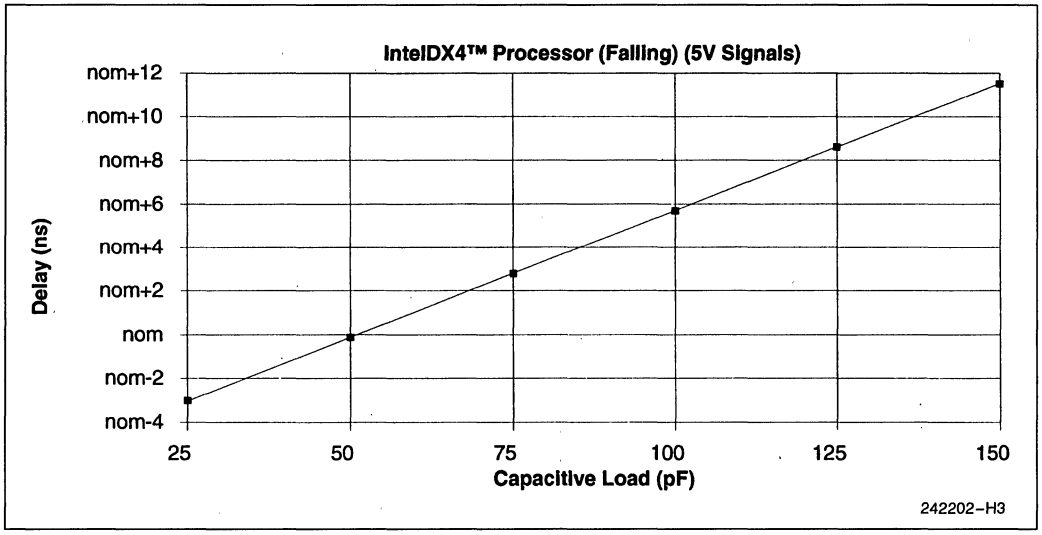


Figure 17-16. IntelDX4™ Processor Capacitive Derating Curve for Low-to-High Transitions (5V Signals)

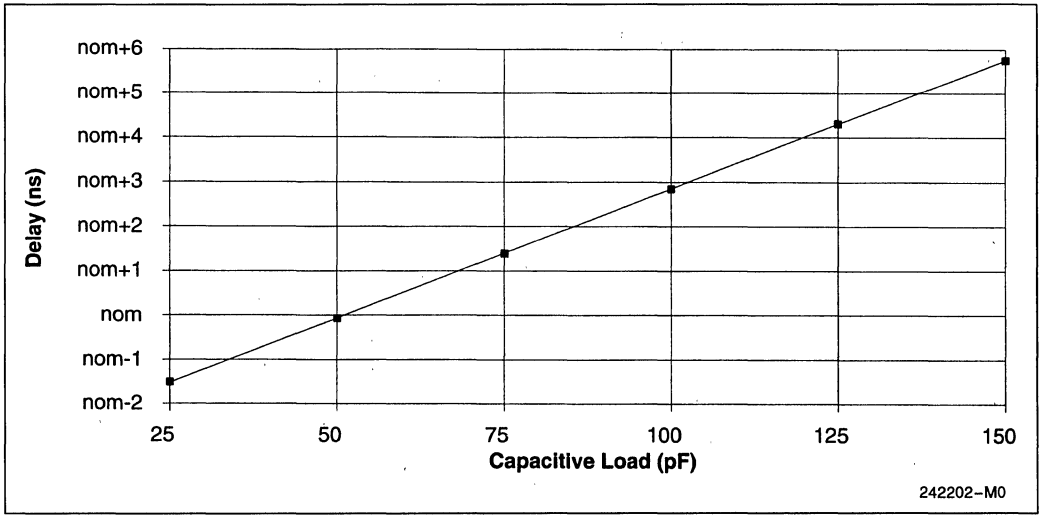


Figure 17-17. IntelDX4™ Processor Capacitive Derating Curve for Low-to-High Transitions (3V/5V Signals)

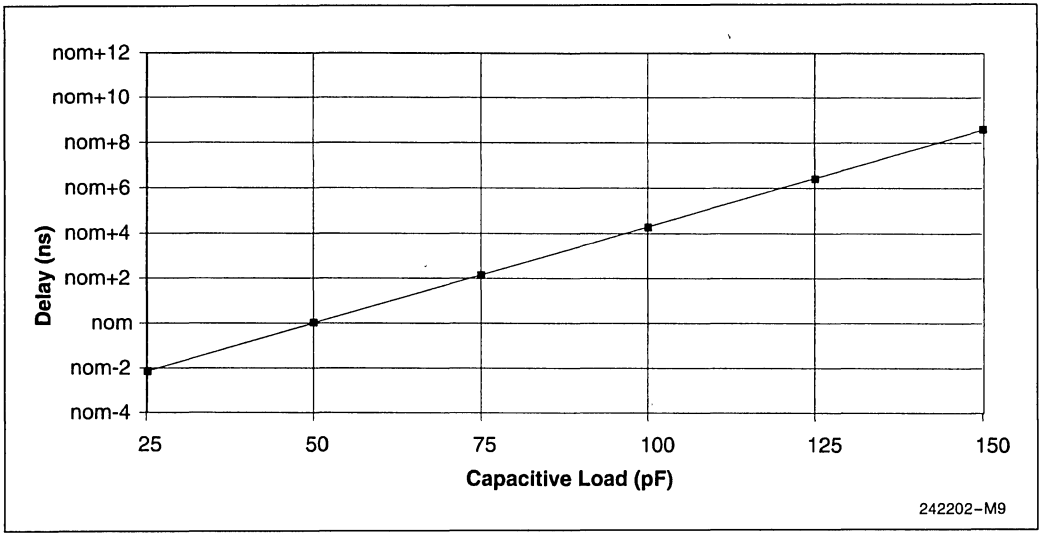


Figure 17-18. Write-Back Enhanced IntelDX4™ Processor (Falling) (5V Signals)

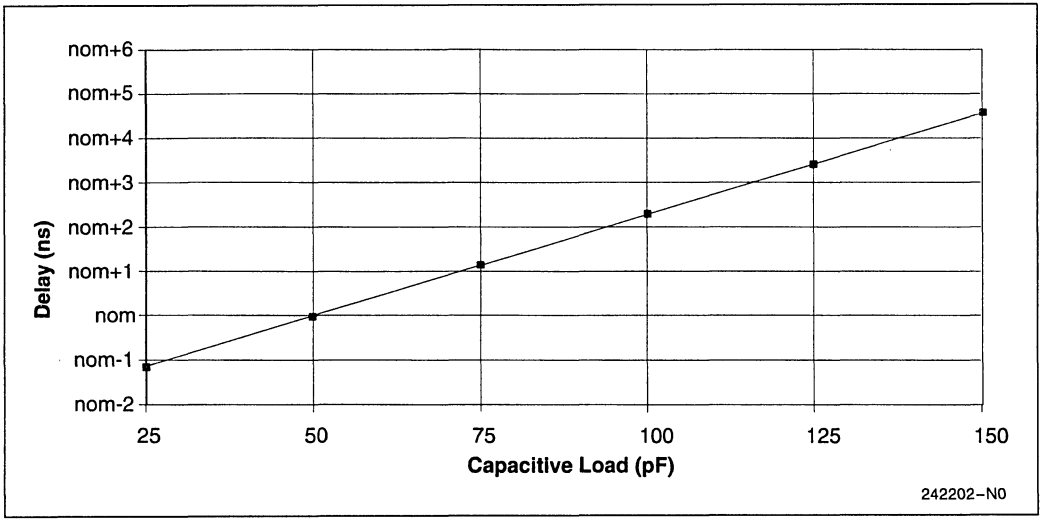


Figure 17-19. Write-Back Enhanced IntelDX4™ Processor (Rising) (3V/5V Signals)

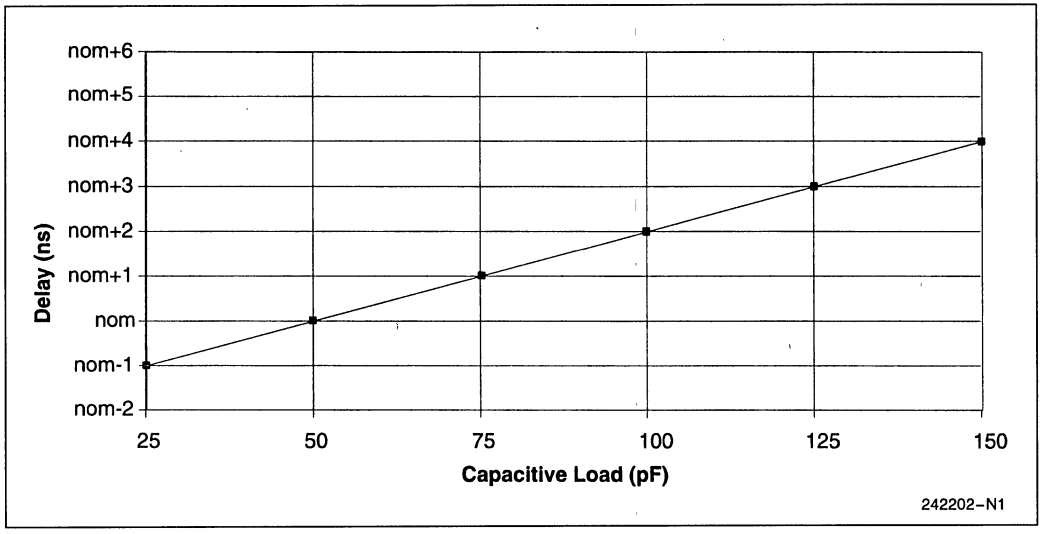


Figure 17-20. Write-Back Enhanced IntelDX4™ Processor (Falling) (3V Signals)

18.0 MECHANICAL DATA

This section describes the package dimensions and thermal specifications for all processors in the Intel486 processor family.

NOTE:
For further details about thermal and mechanical package specifications and methodologies, refer to the 1994 Packaging Handbook (order number 240800).

18.1 Intel486™ Processor Package Dimensions

The processor dimensions are listed in the following order:

- 168-pin PGA package
- 208-lead SQFP package
- 196-lead PQFP package.

18.1.1 168-PIN PGA PACKAGE

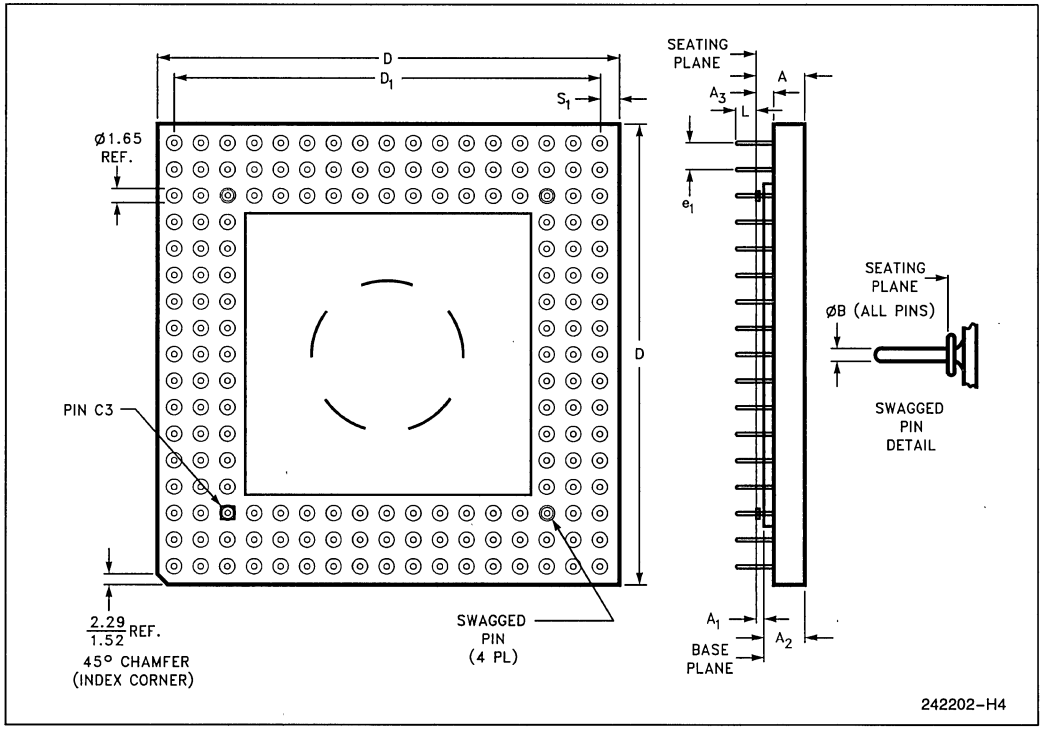


Figure 18-1. 168-Pin Ceramic PGA Package Dimensions

Table 18-1. 168-Pin Ceramic PGA Package Dimensions

Family: Ceramic Pin Grid Array Package						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	3.56	4.57		0.140	0.180	
A_1	0.64	1.14	SOLID LID	0.025	0.045	SOLID LID
A_2	2.8	3.5	SOLID LID	0.110	0.140	SOLID LID
A_3	1.14	1.40		0.045	0.055	
B	0.43	0.51		0.017	0.020	
D	44.07	44.83		1.735	1.765	
D_1	40.51	40.77		1.595	1.605	
e_1	2.29	2.79		0.090	0.110	
L	2.54	3.30		0.100	0.130	
N	168			168		
S_1	1.52	2.54		0.060	0.100	
ISSUE	IWS REV X 7/15/88					

Table 18-2. Ceramic PGA Package Dimension Symbols

Letter or Symbol	Description of Dimensions
A	Distance from seating plane to highest point of body
A_1	Distance between seating plane and base plane (lid)
A_2	Distance from base plane to highest point of body
A_3	Distance from seating plane to bottom of body
B	Diameter of terminal lead pin
D	Largest overall package dimension of length
D_1	A body length dimension, outer lead center to outer lead center
e_1	Linear spacing between true lead position centerlines
L	Distance from seating plane to end of lead
S_1	Other body dimension, outer lead center to edge of body

NOTES:
1. Controlling dimension: millimeter.
2. Dimension "e_1" ("e") is non-cumulative.
3. Seating plane (standoff) is defined by P.C. board hole size: 0.0415–0.0430 inch.
4. Dimensions "B", "B_1" and "C" are nominal.
5. Details of Pin 1 identifier are optional.

18.1.2 208-LEAD SQFP PACKAGE

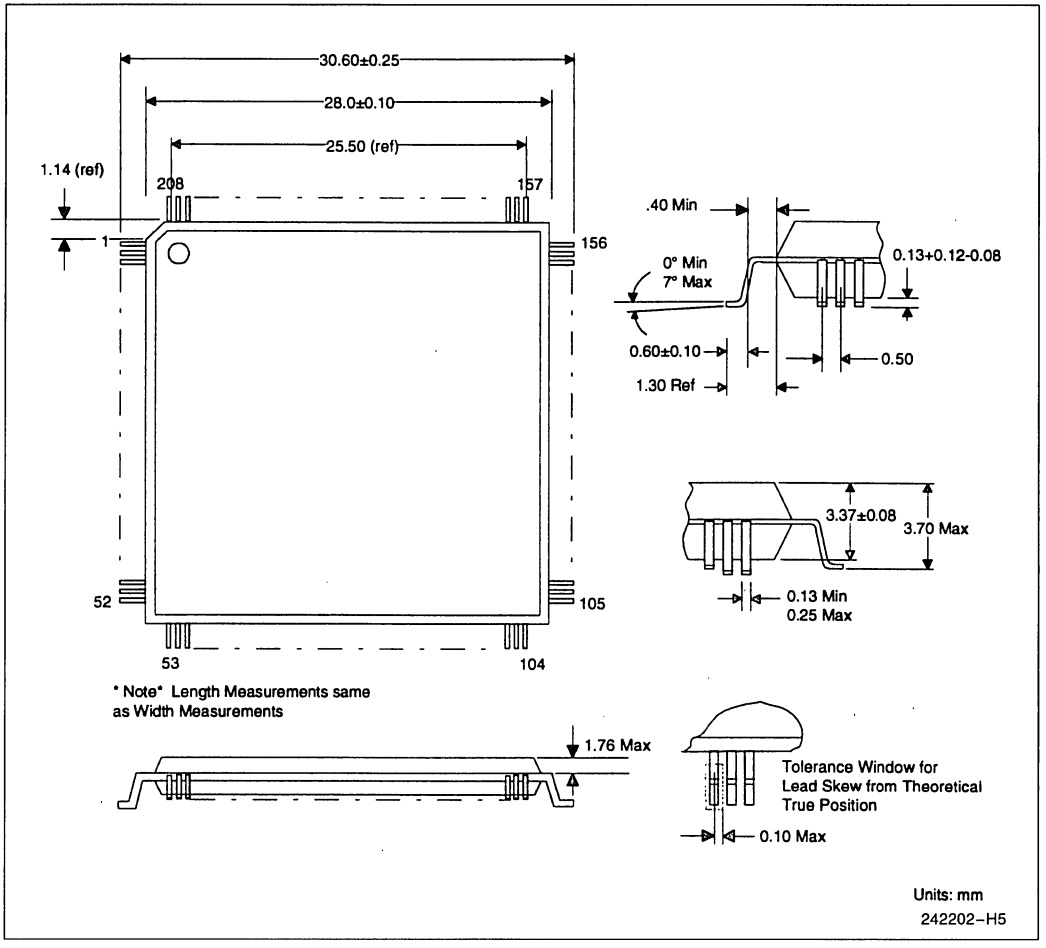
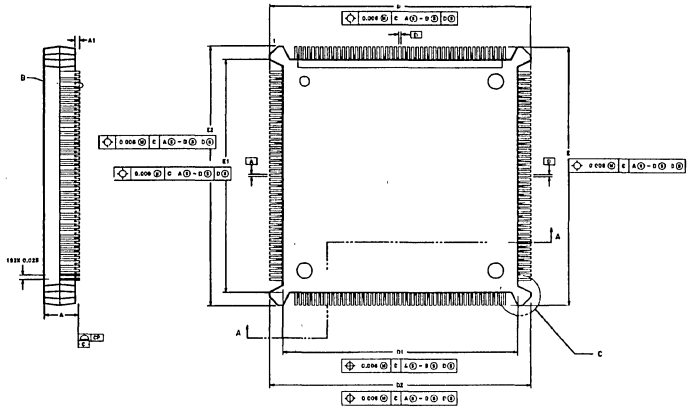


Figure 18-2. 208-Lead SQFP Package Dimensions

18.1.3 196-LEAD PQFP PACKAGE



242202–H6

NOTE:
Interpret dimensions and tolerances in accordance with ANSI Y14.5M-1982.

Figure 18-3. Principal Dimensions and Data for 196-Lead Plastic Quad Flat Pack Package

Table 18-3. Symbol List and Dimensions for 196-Lead PQFP Package

Symbol	Description of Dimensions	Min	Max
A	**Package Height:** Distance from the seating plane to the highest point of body.	0.160	0.175
A1	**Standoff:** The distance from the seating plane to the base plane.	0.020	0.035
D, E	**Overall Package Dimension:** Lead tip to lead tip.	1.470	1.485
D1, E1	**Plastic Body Dimension**	1.347	1.353
D2, E2	**Bumper Distance** Without FLASH With FLASH	 1.497 1.497	 1.503 1.510
CP	**Seating Plane Coplanarity**	0.000	0.004

NOTES:
1. All dimensions and tolerances conform to ANSI Y14.5M-1982.
2. Dimensions are in inches.

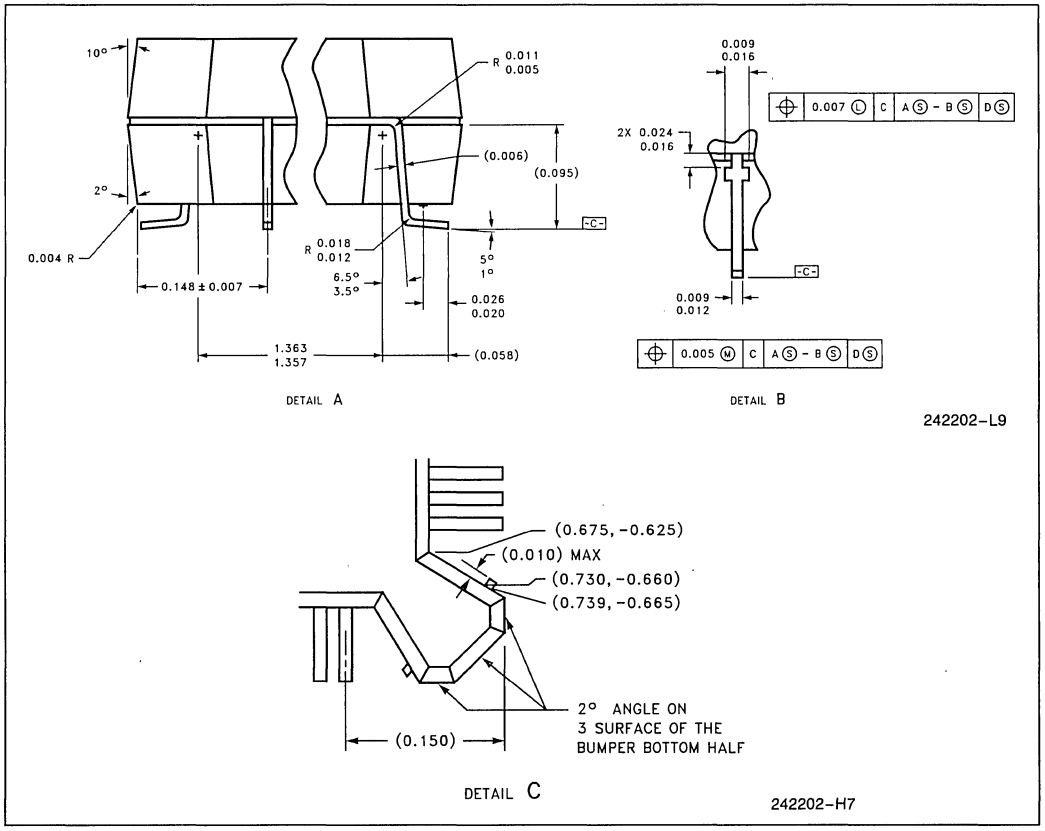


Figure 18-4. Typical Lead

18.2 Package Thermal Specifications

The Intel486 processors are specified for operation when T_C (the case temperature) is within the range of 0°C–85°C. T_C may be measured in any environment to determine whether the Intel486 processor is within the specified operating range. The case temperature, with and without heat sink should be measured using a 0.005" diameter (AWG#36) thermocouple with a 90° angle adhesive bond at the center of the package top surface, opposite the pins. Figure 18-5 and Figure 18-6 illustrate this methodology.

The ambient temperature (T_A) is guaranteed as long as T_C is not violated. The ambient temperature can be calculated from θ_{JC} and θ_{JA} from the following equations.

$$T_J = T_C + P * \theta_{JC}$$

$$T_A = T_J - P * \theta_{JA}$$

$$T_C = T_A + P * [\theta_{JA} - \theta_{JC}]$$

Where:

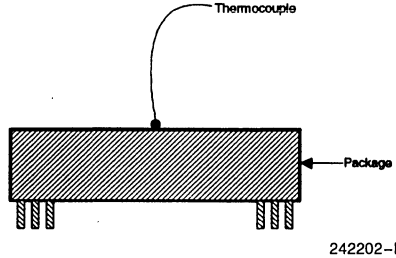
T_J, T_A, T_C = Junction, Ambient and Case Temperature, respectively.

θ_{JC}, θ_{JA} = Junction-to-Case and Junction-to-Ambient thermal Resistance, respectively.

P = Maximum Power Consumption

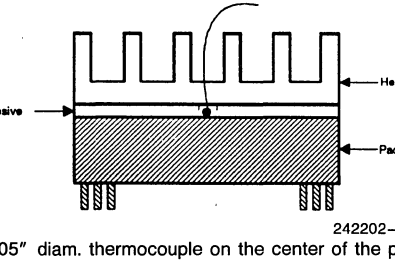
The values for θ_{JA} and θ_{JC} are given below for the packaging and operating frequencies.

Note that T_A is greatly improved by attaching "fins" or a "heat sink" to the package. P (the maximum power consumption) is calculated by using the maximum I_{CC} at nominal V_{CC} (either 3.3V or 5V) as tabulated in the DC Characteristics in section 17.



0.005″ diam. thermocouple on the center of the package top surface with a 90° angle adhesive bond

Figure 18-5. Case Temperature Measurement without Heat Sink



0.005″ diam. thermocouple on the center of the package top surface with a 90° angle adhesive bond through a hole drilled at the heat sink base

Figure 18-6. Case Temperature Measurement with Heat Sink

Refer to section 16, "OverDrive® Processor Socket," for OverDrive processor thermal specifications.

18.2.1 168-PIN PGA PACKAGE THERMAL CHARACTERISTICS FOR 3.3V IntelDX4™ PROCESSOR

Desktop Applications

All thermal measurements for the PGA package were taken with part in a socket mounted to a 4.5″ x 4.5″ printed circuit board with 2 power plane layers and 2 signal layers in a test chamber.

Table 18-5 shows maximum ambient temperatures of IntelDX4 processor for each core operating frequency. The maximum ambient temperature is measured as ambient air temperature in the system case. These maximum ambient temperatures are not valid for the OverDrive processor.

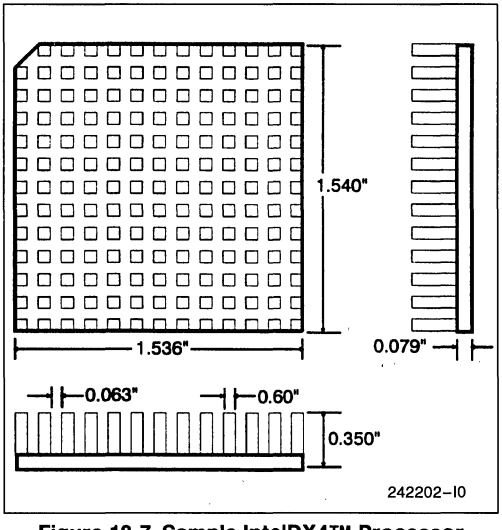


Figure 18-7. Sample IntelDX4™ Processor PGA Heat Sink

Table 18-4. Desktop PGA Package Thermal Resistance (°C/W)–θ_{JC} and θ_{JA} for the Write-Back Enhanced IntelDX4 and IntelDX4™ Processors

		θ_{JA} vs Airflow—ft/min (m/sec)					
	θ_{JC}	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
With Heat Sink*	2	13.5	8.5	6.5	5.5	4.5	4.25
Without Heat Sink	2	17.5	15	13	11.5	10.0	9.5

*0.350″ high omnidirectional heat sink.

Table 18-5. Desktop PGA Package Maximum Ambient Temperature for the Write-Back Enhanced IntelDX4 and IntelDX4™ Processors

		Airflow—ft/min (m/sec)			
	Freq. (MHz)	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
$T_{ambient}$ °C with Heat Sink*	100	35.5	57	65.5	70
$T_{ambient}$ °C without Heat Sink	100	18.5	29	37.5	44

*0.350″ high omnidirectional heat sink.

Mobile Applications for IntelDX4 Processor

The thermal resistances and maximum ambient temperatures for both the Write-Back Enhanced IntelDX4 and the IntelDX4 processors in the PGA package are given in Table 18-6.

NOTE:
These values should be used as guidelines only, and are highly system dependent. Mobile Applications assume no airflow (if airflow exists in the system, the Desktop Application values should be used). All thermal measurements for the PGA package were taken with part in a socket mounted to a 4.5″ x 4.5″ printed circuit board with 2 power plane layers and 2 signal layers in a test chamber. Final system verification should always refer to the case temperature specification.

Table 18-6. Mobile PGA Package Thermal Resistance (°C/W)—θ_{JC} and θ_{JA} and $T_{ambient}$

	θ_{JC}	θ_{JA}	T_A (°C) (Ext)
With Heat Sink*			
75 MHz	2	13.5	48
100 MHz	2	13.5	35.5
Without Heat Sink			
75 MHz	2	17.5	35
100 MHz	2	17.5	18.5

*0.350″ high omnidirectional heat sink.

18.2.2 168-PIN PGA PACKAGE THERMAL CHARACTERISTICS FOR 5V Intel486™ PROCESSORS

Table 18-7. Thermal Resistance (°C/W) θ_{JC} and θ_{JA} for the 168-Pin PGA Package of the Intel486™ Processor

		θ_{JA} vs. Airflow—ft/min. (m/sec)					
	θ_{JC}	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
With Heat Sink*	1.5	13	8.0	6.0	5.0	4.5	4.25
Without Heat Sink	1.5	17	14.5	12.5	11.0	10.0	9.5

*0.350″ high omnidirectional heat sink.

Table 18-8. Maximum $T_{ambient}$ for the 5V, 168-Pin PGA Intel486™ Processor Family

	Freq. (MHz)	Airflow—ft/min. (m/sec)			
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
Intel486™ SX Processor					
$T_{ambient}$ °C with Heat Sink*	25	54	68	73	76
	33	47	64	70	74
$T_{ambient}$ °C without Heat Sink	25	44	50	56	62
	33	34	42	49	57
IntelSX2™ Processor					
$T_{ambient}$ °C with Heat Sink*	50	38	59	67	71
$T_{ambient}$ °C without Heat Sink	50	22	32	40	50
Intel486 DX and IntelDX2™ Processors**					
$T_{ambient}$ °C with Heat Sink*	33	50	65	71	74
	50	33	56	65	69
	66	19	48	59	65
$T_{ambient}$ °C without Heat Sink	33	38	46	52	59
	50	15	26	35	46
	66	−4	11	22	36
Write-Back Enhanced IntelDX2 Processors					
$T_{ambient}$ °C with Heat Sink	50	29	53	63	68
	66	11	43	56	62
$T_{ambient}$ °C without Heat Sink	50	9	21	31	43
	66	−15	1	14	30

NOTE:
* 0.350″ high omnidirectional heat sink.
** For 33- and 50-MHz Intel486 DX processors and 50- and 66-MHz IntelDX2 processors.

18.2.3 THERMAL SPECIFICATIONS FOR 208-LEAD SQFP PACKAGE

Table 18-9. Thermal Resistance (°C/W) θ_{JA} for the Intel486™ Processor (for SQFP)

	θ_{JA} vs. Airflow —ft/min. (m/sec)			
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
Intel486™ SX Processor without Heat Sink	36.0	27.5	25.0	22.5
Intel486 DX Processor without Heat Sink	25.0	17.5	15.0	13.0
IntelDX2™ Processor without Heat Sink	24.0	17.0	15.0	13.0

Table 18-10. Thermal Resistance (°C/W) θ_{JC} for the Intel486™ Processor (for SQFP)

	θ_{JC} vs. Airflow —ft/min. (m/sec)			
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
	0	200	400	600
Intel486™ SX Processor	4.0	7.5	8.0	8.5
Intel486 DX and IntelDX2™ Processors	3.5	6.0	6.0	6.0

Table 18-11. Maximum $T_{ambient}$ for the 3.3V, 208-Lead SQFP Intel486™ Processor Family

	Freq. (MHz)	Airflow—ft/min. (m/sec)			
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
Intel486™ SX Processor					
$T_{ambient}$ °C without Heat Sink	25	54	66	69	72
	33	47	62	65	69
Intel486 DX Processor					
$T_{ambient}$ °C without Heat Sink	33	58	70	74	76
IntelDX2™ Processor					
$T_{ambient}$ °C without Heat Sink	40	57	70	73	75
	50	51	67	70	73
Write-Back Enhanced IntelDX2 Processor					
$T_{ambient}$ °C without Heat Sink	40	53	68	71	74
	50	46	64	68	72

18.2.3.1 SQFP Package Thermal Characteristics for the 3.3V Write-Back Enhanced IntelDX4™ and IntelDX4 Processors

Desktop Applications

All thermal measurements for the SQFP package were taken with part soldered to a 4.5″ x 4.5″ printed circuit board with two power plane layers and two signal layers in a test chamber.

Table 18-14 shows maximum ambient temperatures of IntelDX4 processor for each core operating frequency. The maximum ambient temperature is measured as ambient air temperature in the system case.

Mobile Applications

The thermal resistances and maximum ambient temperatures for the IntelDX4 processor in the SQFP package are given below.

NOTE:
These values should be used as guidelines only, and are highly system dependent. Mobile Applications assume no airflow (if airflow exists, the Desktop Application values should be used). Data was taken on a 2″ x 2″, 4-layer circuit board in a test chamber. These values have been correlated to a typical 8.5″ x 11″ x 1.5″ notebook PC **with the ambient temperature measured external to the system.** Final system verification should always refer to the case temperature specification.

Table 18-12. Mobile SQFP Package Thermal Resistance (°C/W)—θ_{JC} and θ_{JA} and $T_{ambient}$

	θ_{JC}	θ_{JA}	T_A (°C) (Ext)
With Heat Sink			
75 MHz	1.0	14	43
100 MHz	1.0	13.5	31
Without Heat Sink			
75 MHz	1.5	18	31.5
100 MHz	1.5	17.5	16

Table 18-13. Desktop SQFP Package Thermal Resistance (°C/W)—θ_{JC} and θ_{JA} for the IntelDX4™ Processor

		θ_{JA} vs Airflow—ft/min (m/sec)			
	θ_{JC}	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
With Heat Sink	0.8	10.5	6.5	5	4
Without Heat Sink	1.2	12.5	10	9	8.5

Table 18-14. Desktop SQFP Package Maximum Ambient Temperature for the IntelDX4™ and Write-Back Enhanced IntelDX4 Processors

		Airflow—ft/min (m/sec)			
	Freq. (MHz)	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
$T_{ambient}$ °C with Heat Sink	100	43.5	60.5	67	71
$T_{ambient}$ °C without Heat Sink	100	36.5	46	50	52.5

NOTE:
The SQFP Package is intended primarily for the Mobile Market.

18.2.4 THERMAL SPECIFICATIONS FOR 196-LEAD PQFP PACKAGE

Table 18-15. Thermal Resistance (°C/W) θ_{JC} and θ_{JA}

	θ_{JC}	θ_{JA} vs Airflow—ft/min (m/sec)			
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
With Heat Sink*	3.5	17.0	10.5	8.5	8.0
Without Heat Sink	3.5	20.5	16.5	14.0	12.5

*0.350" high omnidirectional heat sink.

Table 18-16. Maximum $T_{ambient}$ for the 5V, 196-Lead PQFP Intel486™ Processor Family

	Freq. (MHz)	Airflow—ft/min (m/sec)			
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)
Intel486™ SX Processor					
$T_{ambient}$ °C with Heat Sink*	25	49	66	72	73
	33	41	62	69	70
$T_{ambient}$ °C without Heat Sink	25	40	50	57	61
	33	29	42	51	56
Intel486 DX Processor					
$T_{ambient}$ °C with Heat Sink	33	44	64	70	71
$T_{ambient}$ °C without Heat Sink	33	34	46	53	58

APPENDIX A
ADVANCED FEATURES

Some non-essential information regarding the Intel486 processor is considered Intel confidential and proprietary and is not documented in this publication. This information is provided in the *Supplement to the Pentium® Processor Family User's Manual* and is available with the appropriate non-disclosure agreements in place. Please contact Intel Corporation for details.

The *Supplement to the Pentium® Family User's Manual* contains architecture extensions for the Intel486 and Pentium processors that are confidential and non-essential for standard applications. These extensions include low-level registers that provide access to features such as page extensions, virtual mode extensions, testing, and performance monitoring.

This information is specifically targeted at software developers who develop the following types of low-level software:

- operating system kernels
- virtual memory managers
- BIOS and processor test software
- performance monitoring tools

For software developers designing other categories of software, this information does not apply. All of the required program development details are provided in the *Intel486™ Microprocessor Family Programmer's Reference Manual*, which is publicly available from the Intel Corporation Literature Center. To obtain this document, contact the Intel Corporation Literature Center at:

Intel Corporation Literature Center
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683 and reference Order Number 240486.

APPENDIX B
FEATURE DETERMINATION

CPUID Instruction

The Intel486 processor implements the CPUID instruction that makes information available to the system software about the family, model, and stepping of the processor on which it is executing. Support of this instruction is indicated by the ability of system software to write and read the bit in position EFLAGS.21, referred to as the EFLAGS.ID bit. The actual state of the EFLAGS.ID bit is irrelevant and provides no significance to the hardware. This bit is reset to zero upon device reset (RESET and SRESET) for compatibility with older Intel486 processor designs.

Operation

The CPUID instruction requires the software developer to pass an input parameter to the processor in the EAX register. The processor response is returned in registers EAX, EBX, ECX, and EDX.

1. When the parameter passed to EAX is zero, the register values returned upon instruction execution are:

   EAX[31:0] ← 1

   EBX[31:0] ← 756E6547—"Genu", with "G" in the low nibble of BL

   EDX[31:0] ← 49656E69—"inel", with "i' in the low nibble of DL

   ECX[31:0] ← 6C65746E—"ntel", with "n" in the low nibble of CL

   The values in EBX, ECX, and EDX indicate an Intel processor. When taken in the proper order, they decode to the string "GenuineIntel."

2. When the parameter passed to EAX is one, the register values returned upon instruction execution are:

   EAX[3:0] ← xxxx—Stepping ID

   EAX[7:4] ← xxxx—Model (See Table B-2.)

   EAX[11:8] ← 0100—Family

   EAX[15:12] ← 0000

   EAX[31:16] ← Intel Reserved

   EBX[31:0] ← 00000000

   ECX[31:0] ← 00000000

   EDX[0:0] ← 1—FPU on-chip

   EDX[3:1] ← 1—For more information on these bits, see Appendix A

   EDX[31:4] ← Intel Reserved

   The value returned in EAX after CPUID instruction execution is identical to the value loaded into EDX upon device reset. Software must avoid any dependency upon the state of reserved processor bits.

3. When the parameter in EAX is greater than one, the register values returned upon instruction execution are:

   EAX[31:0] ← 00000000

   EBX[31:0] ← 00000000

   EDX[31:0] ← 00000000

   ECX[31:0] ← 00000000

Flags Affected

No flags are affected.

Exceptions

None.

For More Information

Refer to the Intel application note AP-485, *Intel Processor Identification with the CPUID Instruction* for more details.

Table B-1. CPUID Instruction Description

OPCODE	Instruction	Processor Core Clocks	EAX Input Value	Description
0F A2	CPUID	14 9	1 0 or greater than 1	Processor Identification Intel String/Null Registers

Table B-2. Intel486™ Processor Signatures

Family	Model	Stepping[1]	Description
0100	0000 and 0001	xxxx	Intel486™ DX Processors
0100	0010	xxxx	Intel486 SX Processors
0100	0011	xxxx	Intel487™ Processors[2]
0100	0011	xxxx	IntelDX2™ and Intel DX2 OverDrive® Processors
0100	0100	xxxx	Intel486 SL Processor[2]
0100	0101	xxxx	IntelSX2™ Processors
0100	0111	xxxx	Write-Back Enhanced IntelDX2 Processors
0100	1000	xxxx	IntelDX4™ and IntelDX4 OverDrive Processors
0100	1001	xxxx	Write-Back Enhanced IntelDX4 Processor
0101	0101	xxxx	Reserved for Pentium® OverDrive Processor for IntelDX4 Processor

NOTES:
1. Intel releases information about stepping numbers as needed.
2. This processor does not implement the CPUID instruction.

APPENDIX C
I/O BUFFER MODELS

For processor bus speeds above 33 MHz (e.g., 50 MHz), the capacitive derating curves are not guaranteed. For bus speeds of 50 MHz, I/O buffer modeling techniques should be used to accurately simulate (and predict) the behavior of processor signals in a particular environment.

This appendix presents sample I/O buffer model parameters for the Write-Back Enhanced IntelDX4 processor. A text listing of the data is presented in the IBIS format.

I/O buffer model information is available for all Intel486 processors described in this datasheet. Contact your Intel representative for the latest I/O buffer models for the Write-Back Enhanced IntelDX4 processor and other members of the Intel486 processor family.

Sample IBIS Files for the Write-Back Enhanced IntelDX4™ Processor

The following pages present sample IBIS file outputs for the Write-Back Enhanced IntelDX4 processor.

Sample Text Listing of IBIS Files for the Write-Back Enhanced IntelDX4™ Processor

```
|**************************************************************************
|
[IBIS Ver]      1.1
[File name]     inteldx4pg.ibs
[File Rev]      2.0
[Date]          3/23/94
[Source]        File originated at Intel Corporation
[Notes]         The following information corresponds to the INTELDX4(TM)
|               processor and has been correlated with silicon.  This file
                is for the PGA package only.IntelDX4 processor
[Disclaimer]    This information is for modeling purposes only, and is not
|               guaranteed.
|
|**************************************************************************
|
|
[Component]       INTELDX4 PROCESSOR
[Manufacturer]    Intel
[Package]
|
|      typ   min          max
R_pkg        2329m 728m  3930m
L_pkg        17.79nH     8.56nH      27.01nH
C_pkg        6.03pF      1.89pF      10.16pF
|
|
|**************************************************************************
|
[Pin]    signal_name      model_name   R_pin    L_pin    C_pin
 A01     D20                 I/O2        1866m    15.54n    3.88p
 A02     D22                 I/O2        1808m    13.70n    5.60p
 A04     D23                 I/O2        1468m    13.33n    3.14p
 A05     DP3                 I/O2        1406m    11.59n    4.50p
 A06     D24                 I/O2        1412m    13.02n    3.04p
 A08     D29                 I/O2        1274m    10.90n    4.14p
 A15     IGNNE#              Input1      1858m    13.96n    5.74p
 A16     INTR                Input1      1956m    14.47n    6.00p
 A17     AHOLD               Input1      3414m    22.11n    9.99p
 B01     D19                 I/O2        1762m    13.46n    5.47p
 B02     D21                 I/O2        1636m    14.27n    3.45p
 B06     D25                 I/O2        1178m    11.72n    2.61p
 B08     D31                 I/O2        1050m     9.73n    3.52p
 B10     SMI#                Input1      948m     10.45n    2.19p
 B15     NMI                 Input1      1430m    13.12n    3.07p
 B17     EADS#               Input1      1728m    14.78n    3.62p
 C01     D11                 I/O1        2440m    18.73n    4.93p
 C02     D18                 I/O2        1446m    13.21n    3.10p
 C03     CLK                 Clockbuffer 2486m    18.99n    5.02p
 C06     D27                 I/O2        964m     10.54n    2.21p
 C07     D26                 I/O2        892m     8.90n     3.09p
 C08     D28                 I/O2        752m     9.36n     1.82p
 C09     D30                 I/O2        728m     9.22n     1.78p
 C10     SRESET              Input1      784m     9.54n     1.88p
 C12     SMIACT#             Output1     1270m    12.23n    2.78p
```

242202-J3

C14	FERR#	Output1	1904m	15.76n	3.95p
C15	FLUSH#	Input1	1342m	11.26n	4.32p
C16	RESET	Input1	1480m	11.98n	4.70p
C17	BS16#	Input1	2756m	20.49n	5.52p
D01	D9	I/O1	2718m	18.47n	8.09p
D02	D13	I/O1	2100m	16.84n	4.31p
D03	D17	I/O2	1156m	11.60n	2.57p
D15	A20M#	Input1	1148m	11.56n	2.55p
D16	BS8#	Input1	3474m	22.43n	10.16p
D17	BOFF#	Input1	3452m	22.31n	10.10p
E03	D10	I/O1	2356m	16.57n	7.10p
E15	HOLD	Input1	1920m	15.85n	3.98p
F01	DP1	I/O1	2394m	16.77n	7.20p
F02	D8	I/O1	1864m	15.53n	3.87p
F03	D15	I/O1	858m	9.95n	2.02p
F15	KEN#	Input1	2404m	16.82n	7.23p
F16	RDY#	Input1	1804m	15.20n	3.76p
F17	BE3#	Output2	3336m	23.71n	6.58p
G03	D12	I/O1	1912m	14.24n	5.88p
G15	STPCLK#	Input1	3930m	27.01n	7.68p
H02	D3	I/O1	1708m	14.67n	3.59p
H03	DP2	I/O1	928m	9.09n	3.19p
H15	BRDY#	Input1	2134m	17.03n	4.37p
J02	D5	I/O1	1528m	13.67n	3.25p
J03	D16	I/O1	1614m	14.14n	3.41p
J15	BE2#	Output2	848m	8.67n	2.97p
J16	BE1#	Output2	1002m	10.75n	2.28p
J17	PCD	Output2	2266m	17.77n	4.61p
K03	D14	I/O1	1160m	10.30n	3.83p
K15	BE0#	Output2	954m	9.22n	3.26p
L02	D6	I/O1	1432m	11.73n	4.57p
L03	D7	I/O1	1048m	11.00n	2.37p
L15	PWT	Output2	1548m	12.34n	4.89p
M03	D4	I/O1	1000m	9.47n	3.39p
M15	D/C#	Output2	1442m	13.19n	3.10p
N01	D2	I/O1	1448m	11.81n	4.61p
N02	D1	I/O1	1198m	11.84n	2.65p
N03	DP0	I/O1	1038m	10.95n	2.35p
N15	LOCK#	Output1	1118m	10.09n	3.71p
N16	M/IO#	Output2	1744m	14.87n	3.65p
N17	W/R#	Output2	1676m	13.01n	5.24p
P01	D0	I/O1	1532m	12.25n	4.84p
P02	A29	I/O3	1292m	12.36n	2.82p
P03	A30	I/O3	1194m	10.48n	3.92p
P15	HLDA	Output1	1568m	13.89n	3.33p
Q01	A31	I/O3	1608m	14.11n	3.40p
Q03	A17	I/O3	2016m	16.38n	4.15p
Q04	A19	I/O3	1584m	12.53n	4.99p
Q05	A21	I/O3	956m	10.49n	2.20p
Q06	A24	I/O3	920m	9.05n	3.17p
Q07	A22	I/O3	894m	8.91n	3.10p
Q08	A20	I/O3	788m	9.56n	1.89p
Q09	A16	I/O3	850m	8.68n	2.98p
Q10	A13	I/O3	836m	9.82n	1.98p
Q11	A9	I/O3	914m	9.02n	3.15p
Q12	A5	I/O3	966m	9.29n	3.29p

242202-J4

```
Q13     A7                  I/O3        1084m    11.20n    2.44p
Q14     A2                  Output1     1134m    11.48n    2.53p
Q15     BREQ                Output2     1872m    15.58n    3.89p
Q16     PLOCK#              Output1     1630m    14.23n    3.44p
Q17     PCHK#               Output2     1616m    12.69n    5.07p
R01     A28                 I/O3        1708m    14.67n    3.59p
R02     A25                 I/O3        1604m    12.63n    5.04p
R05     A18                 I/O3        1498m    13.50n    3.20p
R07     A15                 I/O3        1148m    11.56n    2.55p
R12     A11                 I/O3        1274m    10.90n    4.14p
R13     A8                  I/O3        1252m    12.13n    2.75p
R15     A3                  Output1     1504m    12.11n    4.77p
R16     BLAST#              Output1     1698m    14.61n    3.57p
S01     A27                 I/O3        1900m    14.18n    5.85p
S02     A26                 I/O3        1800m    15.18n    3.75p
S03     A23                 I/O3        1756m    14.93n    3.67p
S05     A14                 I/O3        1842m    13.88n    5.69p
S07     A12                 I/O3        1530m    12.24n    4.84p
S13     A10                 I/O3        1444m    13.20n    3.10p
S15     A6                  I/O3        1722m    13.25n    5.36p
S16     A4                  I/O3        1792m    15.13n    3.74p
S17     ADS#                Output1     1888m    14.12n    5.82p
|
|*****************************************************************************
|
[Model]  Output1
Model_type Output
Polarity Non-Inverting
Enable  Active-Low
|signals ADS#,BLAST#,SMIACT#,A<3:2>,FERR#,HLDA,PLOCK#,LOCK#
|
|          typ   min   max
C_comp     3.05pF  2.9pF   3.2pF
[Voltage range] 3.3V  3.0V  3.6V
|
|*****************************************************************************
|
[Pulldown]
| voltage  I(typ)    I(min)    I(max)
 -5.0V    -960.0mA  -580.0mA  -1410mA
 -2.0V    -190.0mA  -99.0mA  -292.0mA
 -1.0V    -21.0mA  -16.7mA  -27.1mA
 -0.5V    -13.30mA  -8.7mA   -20.5mA
 0.0V    0.0    0.0    0.0
 0.5V    12.9mA    8.3mA    20.1mA
 1.0V    23.7mA    15.1mA    37.2mA
 1.5V    31.5mA    19.7mA    49.7mA
 2.0V    35.4mA    21.6mA    56.6mA
 2.5V    36.3mA    22.0mA    58.6mA
 3.0V    36.6mA    22.2mA    59.0mA
 3.5V    36.9mA    22.3mA    59.4mA
 4.0V    37.0mA    22.4mA    59.7mA
 4.5V    37.1mA    22.5mA    59.8mA
 5.0V    37.2mA    22.5mA    60.0mA
 6.0V    37.2mA    22.5mA    60.0mA
 10.0V    37.2mA    22.5mA    60.0mA
```

242202-J5

```
|
|*************************************************************************
| Note that the pullup voltage in the data table is derived from
| the equation:
|  Vtable = Vcc - Voutput
| For the 8.3V in the table, it is actually 8.3V below Vcc and -5V
| with respected to Ground.
|*************************************************************************
|
[Pullup]
| voltage  I(typ)  I(min)  I(max)
|
 8.3V   -1410mA  -976.25mA -1992.8mA
 4.3V  -79.8mA  -55.6mA  -229.06mA
 3.3V   -47.1mA  -40.80mA -72.66mA
 2.8V   -46.1mA  -29.42mA -70.96mA
 2.3V   -44.8mA  -28.64mA -68.34mA
 1.8V   -42.1mA  -27.14mA -61.94mA
 1.3V   -35.1mA  -23.34mA -50.34mA
 0.8V  -24.1mA  -16.08mA -33.08mA
 0.3V   -9.5mA   -6.3mA  -12.86mA
 -0.2V   6.5mA   4.76mA  9.02mA
 -0.7V   24.3mA   17.3mA  33.04mA
 -1.2V   41.7mA   30.48mA  55.6mA
 -1.7V   56.4mA   42.26mA  74.14mA
 -2.2V   67.3mA   51.26mA  88.4mA
 -2.7V   74.6mA   56.96mA  96.58mA
 -3.7V   79.8mA   61.33mA  104.5mA
 -6.7V   82.4mA   63.55mA  109.5mA
|
[GND_clamp]
| Voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  -0.2mA  NA  NA
 -0.6V  -1.1mA  NA  NA
 -0.7V  -3.0mA  NA  NA
 -0.8V  -6.0mA  NA  NA
 -0.9V  -11.0mA  NA  NA
 -1.0V  -30.0mA  NA  NA
 -1.2V  -120.0mA NA  NA
 -2.0V  -180.0mA NA  NA
 -5.0V  -420.0mA NA  NA
|
|*************************************************************************
|  The data in the following POWER_clamp table is listed
| as "Vcc-relative", meaning that the voltage values are
| referenced to the Vcc pin.  The voltages in the data tables
|  are derived from the equation:
|   Vtable = Vcc - Voutput
| In this case, assuming that Vcc  is referenced to 3.3V.
| 0V in the table actually means 3.3V with respected to
| Ground and 0V above Vcc.
|*************************************************************************
|
[POWER_clamp]
```

242202-J6

4

```
| voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  0mA  NA  NA
 -0.6V  0mA  NA  NA
 -0.7V  0.1mA  NA  NA
 -0.8V  1.0mA  NA  NA
 -0.9V  8.0mA  NA  NA
 -1.0V  14.0mA  NA  NA
 -2.0V  100mA  NA  NA
|
|*******************************************************************************
|
[Ramp]
|  typ  min  max
dV/dt_r  1.13/0.749n 0.93/0.868n 1.35/0.642n
dV/dt_f  0.99/0.447n 0.75/0.543n 1.27/0.387n
|
|*******************************************************************************
|
[Model]  Output2
Model_type Output
Polarity Non-Inverting
Enable  Active-Low
|signal  BE<3:0>#,BREQ,D/C#,M/IO#,PWT,PCD,PCHK#,W/R#
|
|          typ  min  max
C_comp     2.7pF      2.7pF      2.7pF
[Voltage range] 3.3V  3.0V  3.6V
|
|*******************************************************************************
|
[Pulldown]
| voltage  I(typ)  I(min)  I(max)
 -5.0V   -960.0mA  -580.0mA  -1410mA
 -2.0V   -190.0mA  -99.0mA  -292.0mA
 -1.0V   -21.0mA  -16.7mA  -27.1mA
 -0.5V   -13.30mA  -8.7mA   -20.5mA
 0.0V   0.0   0.0   0.0
 0.5V   12.9mA   8.3mA   20.1mA
 1.0V   23.7mA   15.1mA   37.2mA
 1.5V   31.5mA   19.7mA   49.7mA
 2.0V   35.4mA   21.6mA   56.6mA
 2.5V   36.3mA   22.0mA   58.6mA
 3.0V   36.6mA   22.2mA   59.0mA
 3.5V   36.9mA   22.3mA   59.4mA
 4.0V   37.0mA   22.4mA   59.7mA
 4.5V   37.1mA   22.5mA   59.8mA
 5.0V   37.2mA   22.5mA   60.0mA
 6.0V   37.2mA   22.5mA   60.0mA
 10.0V   37.2mA   22.5mA   60.0mA
|
|*****************************************************************
| Note that the pullup voltage in the data table is derived from
| the equation:
|  Vtable = Vcc - Voutput
```

242202-J7

```
| For the 8.3V in the table, it is actually 8.3V below Vcc and -5V
| with respected to Ground.
|*****************************************************************
|
[Pullup]
| Voltage  I(typ)  I(min)  I(max)
|
 8.3V   -1410mA  -976.25mA -1992.8mA
 4.3V  -79.8mA  -55.6mA  -229.06mA
 3.3V   -47.1mA  -40.80mA -72.66mA
 2.8V   -46.1mA  -29.42mA -70.96mA
 2.3V   -44.8mA  -28.64mA -68.34mA
 1.8V   -42.1mA  -27.14mA -61.94mA
 1.3V   -35.1mA  -23.34mA -50.34mA
 0.8V  -24.1mA  -16.08mA -33.08mA
 0.3V   -9.5mA   -6.3mA  -12.86mA
-0.2V    6.5mA   4.76mA  9.02mA
-0.7V    24.3mA   17.3mA  33.04mA
-1.2V    41.7mA   30.48mA  55.6mA
-1.7V    56.4mA   42.26mA  74.14mA
-2.2V    67.3mA   51.26mA  88.4mA
-2.7V    74.6mA   56.96mA  96.58mA
-3.7V    79.8mA   61.33mA  104.5mA
-6.7V    82.4mA   63.55mA  109.5mA
|
[GND_clamp]
|
| Voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
-0.4V  0mA  NA  NA
-0.5V  -0.2mA  NA  NA
-0.6V  -1.1mA  NA  NA
-0.7V  -3.0mA  NA  NA
-0.8V  -6.0mA  NA  NA
-0.9V  -11.0mA  NA  NA
-1.0V  -30.0mA  NA  NA
-1.2V  -120.0mA NA  NA
-2.0V  -180.0mA NA  NA
-5.0V  -420.0mA NA  NA
|
|*****************************************************************
|  The data in the following POWER_clamp table is listed
| as "Vcc-relative", meaning that the voltage values are
| referenced to the Vcc pin.  The voltages in the data tables
|  are derived from the equation:
|   Vtable = Vcc - Voutput
| In this case, assuming that Vcc  is referenced to 3.3V.
| 0V in the table actually means 3.3V with respected to
| Ground and 0V above Vcc.
|*****************************************************************
|
[POWER_clamp]
| voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
-0.4V  0mA  NA  NA
-0.5V  0mA  NA  NA
```

242202-J8

4

```
 -0.6V   0mA   NA   NA
 -0.7V   0.1mA   NA   NA
 -0.8V   1.0mA   NA   NA
 -0.9V   8.0mA   NA   NA
 -1.0V   14.0mA   NA   NA
 -2.0V   100mA   NA   NA
|
[Ramp]
|   typ   min   max
dV/dt_r   1.13/0.749n 0.93/0.868n 1.35/0.642n
dV/dt_f   0.99/0.447n 0.75/0.543n 1.27/0.387n
|
|*********************************************************************
|
[Model]  I/O1
Model_type I/O
Polarity Non-Inverting
Enable   Active-Low
Vinl = 0.8v
Vinh = 2.0v
|signal  DBUS<16:0>,DP<2:0>
|
|           typ   min   max
C_comp      2.7pF       2.7pF       2.7pF
[Voltage range] 3.3V  3.0V  3.6V
|
|*********************************************************************
|
[Pulldown]
|  voltage   I(typ)   I(min)   I(max)
 -5.0V    -960.0mA   -580.0mA   -1410mA
 -2.0V    -190.0mA   -99.0mA   -292.0mA
 -1.0V    -21.0mA   -16.7mA   -27.1mA
 -0.5V    -13.30mA   -8.7mA    -20.5mA
 0.0V    0.0    0.0    0.0
 0.5V    12.9mA    8.3mA    20.1mA
 1.0V    23.7mA    15.1mA    37.2mA
 1.5V    31.5mA    19.7mA    49.7mA
 2.0V    35.4mA    21.6mA    56.6mA
 2.5V    36.3mA    22.0mA    58.6mA
 3.0V    36.6mA    22.2mA    59.0mA
 3.5V    36.9mA    22.3mA    59.4mA
 4.0V    37.0mA    22.4mA    59.7mA
 4.5V    37.1mA    22.5mA    59.8mA
 5.0V    37.2mA    22.5mA    60.0mA
 6.0V    37.2mA    22.5mA    60.0mA
 10.0V    37.2mA    22.5mA    60.0mA
|
|*********************************************************************
| Note that the pullup voltage in the data table is derived from
| the equation:
|  Vtable = Vcc - Voutput
| For the 8.3V in the table, it is actually 8.3V below Vcc and -5V
| with respected to Ground.
|*********************************************************************
|
```

242202-J9

```
[Pullup]
| voltage  I(typ)  I(min)  I(max)
|
 8.3V   -1410mA  -976.25mA -1992.8mA
 4.3V  -79.8mA  -55.6mA  -229.06mA
 3.3V   -47.1mA  -40.80mA -72.66mA
 2.8V   -46.1mA  -29.42mA -70.96mA
 2.3V   -44.8mA  -28.64mA -68.34mA
 1.8V   -42.1mA  -27.14mA -61.94mA
 1.3V   -35.1mA  -23.34mA -50.34mA
 0.8V  -24.1mA  -16.08mA -33.08mA
 0.3V   -9.5mA   -6.3mA  -12.86mA
 -0.2V   6.5mA   4.76mA  9.02mA
 -0.7V   24.3mA   17.3mA  33.04mA
 -1.2V   41.7mA   30.48mA  55.6mA
 -1.7V   56.4mA   42.26mA  74.14mA
 -2.2V   67.3mA   51.26mA  88.4mA
 -2.7V   74.6mA   56.96mA  96.58mA
 -3.7V   79.8mA   61.33mA  104.5mA
 -6.7V   82.4mA   63.55mA  109.5mA
|
[GND_clamp]
| Voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  -0.2mA  NA  NA
 -0.6V  -1.1mA  NA  NA
 -0.7V  -3.0mA  NA  NA
 -0.8V  -6.0mA  NA  NA
 -0.9V  -11.0mA  NA  NA
 -1.0V  -30.0mA  NA  NA
 -1.2V  -120.0mA NA  NA
 -2.0V  -180.0mA NA  NA
 -5.0V  -420.0mA NA  NA
|
|*************************************************************************
|  The data in the following POWER_clamp table is listed
| as "Vcc-relative", meaning that the voltage values are
| referenced to the Vcc pin.  The voltages in the data tables
|  are derived from the equation:
|   Vtable = Vcc - Voutput
| In this case, assuming that Vcc  is referenced to 3.3V.
| 0V in the table actually means 3.3V with respected to
| Ground and 0V above Vcc.
|*************************************************************************
|
[POWER_clamp]
| voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  0mA  NA  NA
 -0.6V  0mA  NA  NA
 -0.7V  0.1mA  NA  NA
 -0.8V  1.0mA  NA  NA
 -0.9V  8.0mA  NA  NA
 -1.0V  14.0mA  NA  NA
```

242202-K0

```
 -2.0V  100mA  NA  NA
|
[Ramp]
|  typ  min  max
dV/dt_r  1.13/0.749n 0.93/0.868n 1.35/0.642n
dV/dt_f  0.99/0.447n 0.75/0.543n 1.27/0.387n
|
|*********************************************************************
|
[Model]  I/O2
Model_type I/O
Polarity Non-Inverting
Enable  Active-Low
Vinl = 3.3v
Vinh = 6.0v
|siganl  DBUS<31:17>,DP<3>
|
|  typ  min  max
C_comp  3.05pF  2.9pF  3.2pF
[Voltage range]  3.3V 3.0V 3.6V
|
|*********************************************************************
|
[Pulldown]
| voltage  I(typ)  I(min)  I(max)
 -5.0V   -960.0mA  -580.0mA  -1410mA
 -2.0V   -190.0mA  -99.0mA  -292.0mA
 -1.0V   -21.0mA  -16.7mA  -27.1mA
 -0.5V   -13.30mA  -8.7mA   -20.5mA
 0.0V   0.0   0.0   0.0
 0.5V   12.9mA   8.3mA   20.1mA
 1.0V   23.7mA   15.1mA   37.2mA
 1.5V   31.5mA   19.7mA   49.7mA
 2.0V   35.4mA   21.6mA   56.6mA
 2.5V   36.3mA   22.0mA   58.6mA
 3.0V   36.6mA   22.2mA   59.0mA
 3.5V   36.9mA   22.3mA   59.4mA
 4.0V   37.0mA   22.4mA   59.7mA
 4.5V   37.1mA   22.5mA   59.8mA
 5.0V   37.2mA   22.5mA   60.0mA
 6.0V   37.2mA   22.5mA   60.0mA
 10.0V   37.2mA   22.5mA   60.0mA
|
|*********************************************************************
| Note that the pullup voltage in the data table is derived from
| the equation:
|  Vtable = Vcc - Voutput
| For the 8.3V in the table, it is actually 8.3V below Vcc and -5V
| with respected to Ground.
|*********************************************************************
|
[Pullup]
| voltage  I(typ)  I(min)  I(max)
|
 8.3V   -1410mA  -976.25mA -1992.8mA
 4.3V  -79.8mA  -55.6mA  -229.06mA
```

242202-K1

```
 3.3V   -47.1mA   -40.80mA -72.66mA
 2.8V   -46.1mA   -29.42mA -70.96mA
 2.3V   -44.8mA   -28.64mA -68.34mA
 1.8V   -42.1mA   -27.14mA -61.94mA
 1.3V   -35.1mA   -23.34mA -50.34mA
 0.8V  -24.1mA   -16.08mA -33.08mA
 0.3V   -9.5mA    -6.3mA   -12.86mA
-0.2V    6.5mA    4.76mA   9.02mA
-0.7V    24.3mA    17.3mA   33.04mA
-1.2V    41.7mA    30.48mA   55.6mA
-1.7V    56.4mA    42.26mA   74.14mA
-2.2V    67.3mA    51.26mA   88.4mA
-2.7V    74.6mA    56.96mA   96.58mA
-3.7V    79.8mA    61.33mA   104.5mA
-6.7V    82.4mA    63.55mA   109.5mA
|
[GND_clamp]
|
| Voltage  I(typ)   I(min)   I(max)
 0.0V   0mA   NA   NA
 -0.4V   0mA   NA   NA
 -0.5V   -0.2mA   NA   NA
 -0.6V   -1.1mA   NA   NA
 -0.7V   -3.0mA   NA   NA
 -0.8V   -6.0mA   NA   NA
 -0.9V   -11.0mA   NA   NA
 -1.0V   -30.0mA   NA   NA
 -1.2V   -120.0mA NA   NA
 -2.0V   -180.0mA NA   NA
 -5.0V   -420.0mA NA   NA
|
|*************************************************************************
|  The data in the following POWER_clamp table is listed
| as "Vcc-relative", meaning that the voltage values are
| referenced to the Vcc pin.  The voltages in the data tables
|  are derived from the equation:
|   Vtable = Vcc - Voutput
| In this case, assuming that Vcc  is referenced to 3.3V.
| 0V in the table actually means 3.3V with respected to
| Ground and 0V above Vcc.
|*************************************************************************
|
[POWER_clamp]
| voltage  I(typ)   I(min)   I(max)
 0.0V   0mA   NA   NA
 -0.4V   0mA   NA   NA
 -0.5V   0mA   NA   NA
 -0.6V   0mA   NA   NA
 -0.7V   0.1mA   NA   NA
 -0.8V   1.0mA   NA   NA
 -0.9V   8.0mA   NA   NA
 -1.0V   14.0mA   NA   NA
 -2.0V   100mA   NA   NA
|
|*************************************************************************
|
```

242202-K2

```
[Ramp]
|  typ  min  max
dV/dt_r  1.13/0.749n 0.93/0.868n 1.35/0.642n
dV/dt_f  0.99/0.447n 0.75/0.543n 1.27/0.387n
|
|*****************************************************************************
|
[Model]  I/O3
Model_type I/O
Polarity Non-Inverting
Enable  Active-Low
Vinl = 0.8v
Vinh = 2.0v
|
|signal  ABUS<31:4>
|
|  typ min max
C_comp  2.9pF 2.9pF 2.9pF
[Voltage range]  3.3V 3.0V 3.6V
|
|*****************************************************************************
|
[Pulldown]
| voltage  I(typ)  I(min)  I(max)
 -5.0V   -960.0mA  -580.0mA  -1410mA
 -2.0V   -190.0mA  -99.0mA  -292.0mA
 -1.0V   -21.0mA  -16.7mA  -27.1mA
 -0.5V   -13.30mA  -8.7mA   -20.5mA
 0.0V   0.0   0.0   0.0
 0.5V   12.9mA   8.3mA   20.1mA
 1.0V   23.7mA   15.1mA   37.2mA
 1.5V   31.5mA   19.7mA   49.7mA
 2.0V   35.4mA   21.6mA   56.6mA
 2.5V   36.3mA   22.0mA   58.6mA
 3.0V   36.6mA   22.2mA   59.0mA
 3.5V   36.9mA   22.3mA   59.4mA
 4.0V   37.0mA   22.4mA   59.7mA
 4.5V   37.1mA   22.5mA   59.8mA
 5.0V   37.2mA   22.5mA   60.0mA
 6.0V   37.2mA   22.5mA   60.0mA
 10.0V   37.2mA   22.5mA   60.0mA
|
|*****************************************************************
| Note that the pullup voltage in the data table is derived from
| the equation:
|  Vtable = Vcc - Voutput
| For the 8.3V in the table, it is actually 8.3V below Vcc and -5V
| with respected to Ground.
|*****************************************************************
|
[Pullup]
| voltage  I(typ)  I(min)  I(max)
|
 8.3V   -1410mA  -976.25mA -1992.8mA
 4.3V  -79.8mA  -55.6mA  -229.06mA
 3.3V   -47.1mA  -40.80mA -72.66mA
```

242202-K3

```
 2.8V   -46.1mA   -29.42mA -70.96mA
 2.3V   -44.8mA   -28.64mA -68.34mA
 1.8V   -42.1mA   -27.14mA -61.94mA
 1.3V   -35.1mA   -23.34mA -50.34mA
 0.8V  -24.1mA   -16.08mA -33.08mA
 0.3V   -9.5mA    -6.3mA   -12.86mA
-0.2V    6.5mA    4.76mA   9.02mA
-0.7V    24.3mA    17.3mA   33.04mA
-1.2V    41.7mA    30.48mA   55.6mA
-1.7V    56.4mA    42.26mA   74.14mA
-2.2V    67.3mA    51.26mA   88.4mA
-2.7V    74.6mA    56.96mA   96.58mA
-3.7V    79.8mA    61.33mA   104.5mA
-6.7V    82.4mA    63.55mA   109.5mA
|
[GND_clamp]
|
| Voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  -0.2mA  NA  NA
 -0.6V  -1.1mA  NA  NA
 -0.7V  -3.0mA  NA  NA
 -0.8V  -6.0mA  NA  NA
 -0.9V  -11.0mA  NA  NA
 -1.0V  -30.0mA  NA  NA
 -1.2V  -120.0mA NA  NA
 -2.0V  -180.0mA NA  NA
 -5.0V  -420.0mA NA  NA
|
|*************************************************************************
|  The data in the following POWER_clamp table is listed
| as "Vcc-relative", meaning that the voltage values are
| referenced to the Vcc pin.  The voltages in the data tables
|  are derived from the equation:
|   Vtable = Vcc - Voutput
| In this case, assuming that Vcc  is referenced to 3.3V.
| 0V in the table actually means 3.3V with respected to
| Ground and 0V above Vcc.
|*************************************************************************
|
[POWER_clamp]
| voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  0mA  NA  NA
 -0.6V  0mA  NA  NA
 -0.7V  0.1mA  NA  NA
 -0.8V  1.0mA  NA  NA
 -0.9V  8.0mA  NA  NA
 -1.0V  14.0mA  NA  NA
 -2.0V  100mA  NA  NA
|
|*************************************************************************
[Ramp]
|  typ  min  max
```

242202-K4

4

```
dV/dt_r  1.13/0.749n 0.93/0.868n 1.35/0.642n
dV/dt_f  0.99/0.447n 0.75/0.543n 1.27/0.387n
|
|*****************************************************************************
|
[Model]  Input1
Model_type Input
Polarity Non-Inverting
Enable  Active-Low
Vinl = 0.8v
Vinh = 2.0v
|signal  A20M#,AHOLD,BOFF#,BRDY#,BS16#,BS8#,FLUSH#,
|  HOLD,IGNNE#,INTR,KEN#,NMI,RDY#,RESET,SRESET,SMI#,STPCLK#
|  typ  min  max
C_comp  2.0pF  2.0pF  2.0pF
[Voltage range]  3.3V 3.0V 3.6V
|
|*****************************************************************************
|
[GND_clamp]
| Voltage  I(typ)  I(min)  I(max)
|
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  -0.2mA  NA  NA
 -0.6V  -1.1mA  NA  NA
 -0.7V  -3.0mA  NA  NA
 -0.8V  -6.0mA  NA  NA
 -0.9V  -11.0mA  NA  NA
 -1.0V  -30.0mA  NA  NA
 -1.2V  -120.0mA NA  NA
 -2.0V  -180.0mA NA  NA
 -5.0V  -420.0mA NA  NA
|
|*****************************************************************************
|  The data in the following POWER_clamp table is listed
| as "Vcc-relative", meaning that the voltage values are
| referenced to the Vcc pin.  The voltages in the data tables
|  are derived from the equation:
|   Vtable = Vcc - Voutput
| In this case, assuming that Vcc  is referenced to 3.3V.
| 0V in the table actually means 3.3V with respected to
| Ground and 0V above Vcc.
|*****************************************************************************
|
[POWER_clamp]
| voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  0mA  NA  NA
 -0.6V  0mA  NA  NA
 -0.7V  0.1mA  NA  NA
 -0.8V  1.0mA  NA  NA
 -0.9V  8.0mA  NA  NA
 -1.0V  14.0mA  NA  NA
 -2.0V  100mA  NA  NA
```

242202-K5

```
|
|*******************************************************************************
|
[Model]  Clockbuffer
Model_type Input
Polarity Non-Inverting
Enable  Active-Low
Vinl = 0.8V
Vinh = 2.0V
|signal  CLK
|  typ  min  max
C_comp  2.0pF  2.0pF  2.0pF
[Voltage range]  3.3V 3.0V 3.6V
|
|*******************************************************************************
|
[GND_clamp]
| Voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  -0.2mA  NA  NA
 -0.6V  -1.1mA  NA  NA
 -0.7V  -3.0mA  NA  NA
 -0.8V  -6.0mA  NA  NA
 -0.9V  -11.0mA  NA  NA
 -1.0V  -30.0mA  NA  NA
 -1.2V  -120.0mA NA  NA
 -2.0V  -180.0mA NA  NA
 -5.0V  -420.0mA NA  NA
|
|*******************************************************************************
|  The data in the following POWER_clamp table is listed
| as "Vcc-relative", meaning that the voltage values are
| referenced to the Vcc pin.  The voltages in the data tables
|  are derived from the equation:
|   Vtable = Vcc - Voutput
| In this case, assuming that Vcc  is referenced to 3.3V.
| 0V in the table actually means 3.3V with respected to
| Ground and 0V above Vcc.
|*******************************************************************************
|
[POWER_clamp]
| voltage  I(typ)  I(min)  I(max)
 0.0V  0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  0mA  NA  NA
 -0.6V  0mA  NA  NA
 -0.7V  0.1mA  NA  NA
 -0.8V  1.0mA  NA  NA
 -0.9V  8.0mA  NA  NA
 -1.0V  14.0mA  NA  NA
 -2.0V  100mA  NA  NA
|
|*******************************************************************************
|
[End]
```

242202-K6

APPENDIX D
BSDL LISTINGS

Below is a listing of a boundary scan description language (BSDL) file for the Write-Back Enhanced IntelDX4 processor.

This file is provided as an example. Contact Intel for design information for this and other Intel486 processors. See section 11.5, "Intel486 Processor Boundary Scan," for a complete description of BSDL instructions and usage.

Write-Back Enhanced IntelDX4™ Processor Listing

```
-- Copyright Intel Corporation 1993
--*****************************************************************************
-- 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.
--*****************************************************************************
-- Boundary-Scan Description Language (BSDL Version 0.0) is a de-facto
-- standard means of describing essential features of ANSI/IEEE 1149.1-1990
-- compliant devices.  This language is under consideration by the IEEE for
-- formal inclusion within a supplement to the 1149.1-1990 standard.  The
-- generation of the supplement entails an extensive IEEE review and a formal
-- acceptance balloting procedure which may change the resultant form of the
-- language.  Be aware that this process may extend well into 1993, and at
-- this time the IEEE does not endorse or hold an opinion on the language.
--*****************************************************************************
--
-- Write-Back Enhanced IntelDX4(TM) processor BSDL description
-- This file has been electrically verified.
-- ----------------------------------------------------------
-- Rev: 1.3 4/26/95


entity WBE_INTELDX4 is
   generic(PHYSICAL_PIN_MAP : string := "PGA_17x17");

   port (A20M      : in    bit;
         ABUS2     : out   bit;
         ABUS3     : out   bit;
         ABUS      : inout bit_vector (4 to 31); -- Address bus (words)
         ADS       : out   bit;
         AHOLD     : in    bit;
         BE        : out   bit_vector(0 to 3);
         BLAST     : out   bit;
         BOFF      : in    bit;
         BRDY      : in    bit;
         BREQ      : out   bit;
         BS8       : in    bit;
         BS16      : in    bit;
         CLK       : in    bit;
         CLKMUL    : in    bit;
```

242202-K7

```
      DBUS      : inout bit_vector(0 to 31);  -- Data bus
      DC        : out   bit;
      DP        : inout bit_vector(0 to 3);
      EADS      : in    bit;
      FERR      : out   bit;
      FLUSH     : in    bit;
      HLDA      : out   bit;
      HOLD      : in    bit;
      IGNNE     : in    bit;
      INC_PGA   : linkage bit; --Internal NC PGA
      INTR      : in    bit;
      KEN       : in    bit;
      LOCK      : out   bit;
      MIO       : out   bit;
      NC_PGA    : linkage bit; -- No Connect for PGA
      NMI       : in    bit;
      PCD       : out   bit;
      PCHK      : out   bit;
      PLOCK     : out   bit;
      PWT       : out   bit;
      RDY       : in    bit;
      RESET     : in    bit;
      SMI       : in    bit;  -- new
      SMIACT    : out   bit;  -- new
      SRESET    : in    bit;  -- new
      STPCLK    : in    bit;  -- new
      TCK, TMS, TDI : in bit;         -- Scan Port inputs
      TDO       : out   bit;          -- Scan Port output
      UP        : in    bit;
      VCC_PGA   : linkage bit_vector(1 to 23);   -- VCC
      VCC5      : linkage bit; --Reference Voltage
      VOLDET    : linkage bit; --Voltage Detect Pin
      VSS_PGA   : linkage bit_vector(1 to 28);   -- VSS
      WR        : out   bit;
      CACHE     : out   bit;  --New pin
      HITM      : out   bit;  --New pin
      INV       : in    bit;  --New pin
      WB_WT     : in    bit);  --New pin

use STD_1149_1_1990.all;

attribute PIN_MAP of WBE_INTELDX4 : entity is PHYSICAL_PIN_MAP;

constant PGA_17x17 : PIN_MAP_STRING :=        -- Define Pin Out of PGA
     "A20M     : D15, "  &
     "ABUS2    : Q14, "  &
     "ABUS3    : R15, "  &
     "ABUS     : (S16, Q12, S15, Q13, R13, Q11, S13, R12,"  &
     "             S7,  Q10, S5,  R7,  Q9,  Q3,  R5,  Q4,  Q8,  Q5,"  &
     "             Q7,  S3,  Q6,  R2,  S2,  S1,  R1,  P2,  P3,  Q1)," &
     "ADS      : S17, "  &
     "AHOLD    : A17, "  &
     "BE       : (K15, J16, J15, F17), "  &
     "BLAST    : R16, "  &
     "BOFF     : D17, "  &
     "BRDY     : H15, "  &
     "BREQ     : Q15, "  &
```

242202-K8

```
        "BS8       : D16, "  &
        "BS16      : C17, "  &
        "CACHE     : B12, "  &
        "CLK       : C3,  "  &
        "CLKMUL    : R17, "  &
        "DBUS      : (P1, N2, N1, H2, M3, J2, L2, L3, F2, D1, E3, "  &
        "             C1, G3, D2, K3, F3, J3, D3, C2, B1, A1, B2, "  &
        "             A2, A4, A6, B6, C7, C6, C8, A8, C9, B8),"  &
        "DC        : M15, "  &
        "DP        : (N3,  F1,  H3,  A5), "  &
        "EADS      : B17, "  &
        "FERR      : C14, "  &
        "FLUSH     : C15, "  &
        "HLDA      : P15, "  &
        "HOLD      : E15, "  &
        "IGNNE     : A15, "  &
        "INC_PGA   : A13, "  &
        "INTR      : A16, "  &
        "KEN       : F15, "  &
        "LOCK      : N15, "  &
        "MIO       : N16, "  &
        "NC_PGA    : C13, "  &
        "NMI       : B15, "  &
        "PCD       : J17, "  &
        "PCHK      : Q17, "  &
        "PLOCK     : Q16, "  &
        "PWT       : L15, "  &
        "RDY       : F16, "  &
        "RESET     : C16, "  &
        "SMI       : B10, "  &
        "SMIACT    : C12, "  &
        "SRESET    : C10, "  &
        "STPCLK    : G15, "  &
        "TCK       : A3,  "  &
        "TDI       : A14, "  &
        "TDO       : B16, "  &
        "TMS       : B14, "  &
        "UP        : C11, "  &
        "VCC_PGA   : (R8,  R9,  R10, R11, R14, P16, M2,  M16, L16, K2,  K16, "
&
        "             H16, G2,  G16, E2,  E16, C4,  C5,  B7,  B9,  B11, "
&
        "             R3,  R6)," &
        "VCC5      : J1,  "  &
        "VOLDET    : S04, "  &
        "VSS_PGA      : (S6,  S8,  S9,  S10, S11, S12, S14, R4,  Q2,  P17, M1,  "
&
        "             M17, L1,  L17, K1,  K17, H1,  H17, G1,  G17, E1,  E17, "
&
        "             B3,  B4,  B5,  A7,  A9,  A11), "  &
        "WB_WT     : B13, "  &
        "HITM      : A12, "  &
        "INV       : A10, "  &
        "WR        : N17  ";

    attribute Tap_Scan_In of    TDI       : signal is true;
    attribute Tap_Scan_Mode of  TMS       : signal is true;
```

242202-K9

```
   attribute Tap_Scan_Out of   TDO      : signal is true;
   attribute Tap_Scan_Clock of TCK      : signal is (25.0e6, BOTH);

   attribute Instruction_Length of WBE_INTELDX4: entity is 4;

   attribute Instruction_Opcode of WBE_INTELDX4: entity is
      "BYPASS  (1111),"    &
      "EXTEST  (0000),"    &
      "SAMPLE  (0001),"    &
      "IDCODE  (0010),"    &
      "RUNBIST (1000),"    &
      "PRIVATE (0011,0100,0101,0110,0111,1001,1010,1011,1100,1101,1110)";

   attribute Instruction_Capture of WBE_INTELDX4: entity is "0001";
   -- there is no Instruction_Disable attribute for WBE_INTELDX4

   attribute Instruction_Private of WBE_INTELDX4: entity is "private";

   attribute Instruction_Usage of WBE_INTELDX4: entity is
      "RUNBIST (registers BIST; "      &
      "IR SCAN MUST BE DONE WITH DEVICE IN HARD RESET;" &
      "Assert RESET and do IR Scan, then release RESET and do DR Scan;" &
      "result 0=pass, 1=fail;"         &
      "clock CLK in Run_Test_Idle;"    &
      "IR End State = Pause-IR;"       &
      "DR End State = Run-Test/Idle;" &
      "RINBIST length in hex = 125000H x ratio of TCK/CPUCLK)";


   attribute Idcode_Register of WBE_INTELDX4: entity is
      "0001"                  &  --version
      "1000001010001000"      &  -- WBE_INTELDX4 part ID
      "00000001001"           &  --manufacturers identity
      "1";                       --required by the standard
--
--
   attribute Register_Access of WBE_INTELDX4: entity is
      "BIST[1] (RUNBIST)" ;

--(*********************************************************************)
--(  The first cell is closest to TDO                                   )
--(*********************************************************************)


   attribute Boundary_Cells of WBE_INTELDX4: entity is "BC_2, BC_1, BC_6";
   attribute Boundary_Length of WBE_INTELDX4: entity is 113;
   attribute Boundary_Register of WBE_INTELDX4: entity is
      "0   (BC_2,  ABUS2,      output3, X,  111,  1,  Z)," &
      "1   (BC_2,  ABUS3,      output3, X,  111,  1,  Z)," &
      "2   (BC_6,  ABUS(4),    bidir,  X,  111,  1,  Z)," &
      "3   (BC_6,  ABUS(5),    bidir,  X,  111,  1,  Z)," &
      "4   (BC_1,  UP ,        input,  X)," &
      "5   (BC_6,  ABUS(6),    bidir,  X,  111,  1,  Z)," &
      "6   (BC_6,  ABUS(7),    bidir,  X,  111,  1,  Z)," &
      "7   (BC_6,  ABUS(8),    bidir,  X,  111,  1,  Z)," &
      "8   (BC_6,  ABUS(9),    bidir,  X,  111,  1,  Z)," &
      "9   (BC_6,  ABUS(10),   bidir,  X,  111,  1,  Z)," &
```

242202-L0

```
"10: (BC_6,  ABUS(11),   bidir,  X,  111,  1,  Z)," &
"11  (BC_6,  ABUS(12),   bidir,  X,  111,  1,  Z)," &
"12  (BC_6,  ABUS(13),   bidir,  X,  111,  1,  Z)," &
"13  (BC_6,  ABUS(14),   bidir,  X,  111,  1,  Z)," &
"14  (BC_6,  ABUS(15),   bidir,  X,  111,  1,  Z)," &
"15  (BC_6,  ABUS(16),   bidir,  X,  111,  1,  Z)," &
"16  (BC_6,  ABUS(17),   bidir,  X,  111,  1,  Z)," &
"17  (BC_6,  ABUS(18),   bidir,  X,  111,  1,  Z)," &
"18  (BC_6,  ABUS(19),   bidir,  X,  111,  1,  Z)," &
"19  (BC_6,  ABUS(20),   bidir,  X,  111,  1,  Z)," &
"20  (BC_6,  ABUS(21),   bidir,  X,  111,  1,  Z)," &
"21  (BC_6,  ABUS(22),   bidir,  X,  111,  1,  Z)," &
"22  (BC_6,  ABUS(23),   bidir,  X,  111,  1,  Z)," &
"23  (BC_6,  ABUS(24),   bidir,  X,  111,  1,  Z)," &
"24  (BC_6,  ABUS(25),   bidir,  X,  111,  1,  Z)," &
"25  (BC_6,  ABUS(26),   bidir,  X,  111,  1,  Z)," &
"26  (BC_6,  ABUS(27),   bidir,  X,  111,  1,  Z)," &
"27  (BC_6,  ABUS(28),   bidir,  X,  111,  1,  Z)," &
"28  (BC_6,  ABUS(29),   bidir,  X,  111,  1,  Z)," &
"29  (BC_6,  ABUS(30),   bidir,  X,  111,  1,  Z)," &
"30  (BC_6,  ABUS(31),   bidir,  X,  111,  1,  Z)," &
"31  (BC_6,  DP(0),      bidir,  X,  112,  1,  Z)," &
"32  (BC_6,  DBUS(0),    bidir,  X,  112,  1,  Z)," &
"33  (BC_6,  DBUS(1),    bidir,  X,  112,  1,  Z)," &
"34  (BC_6,  DBUS(2),    bidir,  X,  112,  1,  Z)," &
"35  (BC_6,  DBUS(3),    bidir,  X,  112,  1,  Z)," &
"36  (BC_6,  DBUS(4),    bidir,  X,  112,  1,  Z)," &
"37  (BC_6,  DBUS(5),    bidir,  X,  112,  1,  Z)," &
"38  (BC_6,  DBUS(6),    bidir,  X,  112,  1,  Z)," &
"39  (BC_6,  DBUS(7),    bidir,  X,  112,  1,  Z)," &
"40  (BC_6,  DP(1),      bidir,  X,  112,  1,  Z)," &
"41  (BC_6,  DBUS(8),    bidir,  X,  112,  1,  Z)," &
"42  (BC_6,  DBUS(9),    bidir,  X,  112,  1,  Z)," &
"43  (BC_6,  DBUS(10),   bidir,  X,  112,  1,  Z)," &
"44  (BC_6,  DBUS(11),   bidir,  X,  112,  1,  Z)," &
"45  (BC_6,  DBUS(12),   bidir,  X,  112,  1,  Z)," &
"46  (BC_6,  DBUS(13),   bidir,  X,  112,  1,  Z)," &
"47  (BC_6,  DBUS(14),   bidir,  X,  112,  1,  Z)," &
"48  (BC_6,  DBUS(15),   bidir,  X,  112,  1,  Z)," &
"49  (BC_6,  DP(2),      bidir,  X,  112,  1,  Z)," &
"50  (BC_6,  DBUS(16),   bidir,  X,  112,  1,  Z)," &
"51  (BC_6,  DBUS(17),   bidir,  X,  112,  1,  Z)," &
"52  (BC_6,  DBUS(18),   bidir,  X,  112,  1,  Z)," &
"53  (BC_6,  DBUS(19),   bidir,  X,  112,  1,  Z)," &
"54  (BC_6,  DBUS(20),   bidir,  X,  112,  1,  Z)," &
"55  (BC_6,  DBUS(21),   bidir,  X,  112,  1,  Z)," &
"56  (BC_6,  DBUS(22),   bidir,  X,  112,  1,  Z)," &
"57  (BC_6,  DBUS(23),   bidir,  X,  112,  1,  Z)," &
"58  (BC_6,  DP(3),      bidir,  X,  112,  1,  Z)," &
"59  (BC_6,  DBUS(24),   bidir,  X,  112,  1,  Z)," &
"60  (BC_6,  DBUS(25),   bidir,  X,  112,  1,  Z)," &
"61  (BC_6,  DBUS(26),   bidir,  X,  112,  1,  Z)," &
"62  (BC_6,  DBUS(27),   bidir,  X,  112,  1,  Z)," &
"63  (BC_6,  DBUS(28),   bidir,  X,  112,  1,  Z)," &
"64  (BC_6,  DBUS(29),   bidir,  X,  112,  1,  Z)," &
"65  (BC_6,  DBUS(30),   bidir,  X,  112,  1,  Z)," &
"66  (BC_6,  DBUS(31),   bidir,  X,  112,  1,  Z)," &
```

242202-L1

```
      "67  (BC_1,  STPCLK,     input, X)," &
      "68  (BC_1,  IGNNE,      input, X)," &
      "69  (BC_1,  INV,        input, X)," &
      "70  (BC_2,  CACHE,      output3, X,  110,  1,  Z)," &
      "71  (BC_2,  FERR,       output3, X,  109,  1,  Z)," &
      "72  (BC_1,  SMI ,       input, X)," &
      "73  (BC_1,  WB_WT ,      input, X)," &
      "74  (BC_2,  HITM ,      output3, X,  109,  1,  Z)," &
      "75  (BC_2,  SMIACT ,    output3, X,  109,  1,  Z)," &
      "76  (BC_1,  SRESET ,    input, X)," &
      "77  (BC_1,  NMI,        input, X)," &
      "78  (BC_1,  INTR,       input, X)," &
      "79  (BC_1,  FLUSH,      input, X)," &
      "80  (BC_1,  RESET,      input, X)," &
      "81  (BC_1,  A20M,       input, X)," &
      "82  (BC_1,  EADS,       input, X)," &
      "83  (BC_2,  PCD,        output3, X,  110,  1,  Z)," &
      "84  (BC_2,  PWT,        output3, X,  110,  1,  Z)," &
      "85  (BC_2,  DC,         output3, X,  110,  1,  Z)," &
      "86  (BC_2,  MIO,        output3, X,  110,  1,  Z)," &
      "87  (BC_2,  BE(3),      output3, X,  110,  1,  Z)," &
      "88  (BC_2,  BE(2),      output3, X,  110,  1,  Z)," &
      "89  (BC_2,  BE(1),      output3, X,  110,  1,  Z)," &
      "90  (BC_2,  BE(0),      output3, X,  110,  1,  Z)," &
      "91  (BC_2,  BREQ,       output3, X,  109,  1,  Z)," &
      "92  (BC_2,  WR,         output3, X,  110,  1,  Z)," &
      "93  (BC_2,  HLDA,       output3, X,  109,  1,  Z)," &
      "94  (BC_1,  CLK,         input, X)," &
      "95  (BC_1,  AHOLD,       input, X)," &
      "96  (BC_1,  HOLD,        input, X)," &
      "97  (BC_1,  KEN,         input, X)," &
      "98  (BC_1,  RDY,         input, X)," &
      "99  (BC_1,  CLKMUL,      input, X)," &
      "100 (BC_1,  BS8,         input, X)," &
      "101 (BC_1,  BS16,        input, X)," &
      "102 (BC_1,  BOFF,        input, X)," &
      "103 (BC_1,  BRDY,        input, X)," &
      "104 (BC_2,  PCHK,       output3, X,  109,  1,  Z)," &
      "105 (BC_2,  LOCK,       output3, X,  110,  1,  Z)," &
      "106 (BC_2,  PLOCK,      output3, X,  110,  1,  Z)," &
      "107 (BC_2,  BLAST,      output3, X,  110,  1,  Z)," &
      "108 (BC_2,  ADS,        output3, X,  110,  1,  Z)," &
      "109 (BC_2,  *,          control, 1)," &            -- DISMISC
      "110 (BC_2,  *,          control, 1)," &            -- DISBUS
      "111 (BC_2,  *,          control, 1)," &            -- DISABUS
      "112 (BC_2,  *,          control, 1)";              -- DISWR

end WBE_INTELDX4;
```

242202-L2

4

APPENDIX E
SYSTEM DESIGN NOTES

SMM Environment Initialization

When the Intel486 processors are operating in Real Mode, the physical address at which instructions and data are fetched is determined by the segment register and an offset (i.e., CS and IP for instructions). When a new value is loaded into a segment register, the new value is shifted to the left by four bits and stored in a segment base register that corresponds to that particular segment (CSBASE, DSBASE, ESBASE, etc.). It is the value stored in the segment base register that is actually used to generate a physical address. For example, the linear address to be used for fetching instructions is determined by adding the value contained in the CS segment base register with the value in the IP register.

When the processor is in Protected Mode, the segment registers are used as selectors to a descriptor table. Each descriptor in a descriptor table contains information about the segment in use, including the segments BASE address (i.e., CSBASE), the limit (or size of the segment), as well as protection level, privileges, operand sizes, and the segment type. In Protected Mode, the linear address is determined by adding the base portion of the descriptor to the appropriate offset.

When in System Management Mode, the processor operates in a pseudo-Real Mode, with address calculation performed in the Real Mode manner. However, the processor adds the value in the segment base register with the value in the EIP register, rather than the IP register, so there are no limits as to the segment size. The physical address of an instruction is obtained by adding the value in CSBASE to the value in EIP.

When entering SMM, it may be necessary to initialize the segment registers to point to SMRAM (see section 8.4.2, "Processor Environment," for their value on SMM entry). If SMBASE has not been relocated, then the necessary segment registers can be initialized to point to SMRAM by using the value in the CS register, 3000H, which points to the SMRAM address space.

When an SMI# occurs after SMBASE has been modified, CSBASE is loaded with the new value of SMBASE. **However, the CS selector register still contains the value 3000H, not the value corresponding to the new SMBASE.**

To initialize segment registers to point to the new SMRAM area, read the SMBASE value from the SMM state that was saved in memory. Because the data segment registers are initialized to 0, do not use them to access the SMM state save area. Instead, perform a read relative to the CS register by using a CS override prefix to a normal memory read. Although CS still contains 3000H, CSBASE contains the value of SMBASE, and CSBASE is used for the address generation.

Once the value of SMBASE is obtained, it must be shifted to the right by four bits to get the appropriate value to be placed in the segment registers. The CS register itself can be initialized by executing a far jump instruction to an address within SMBASE, which causes CS to be reloaded with a value corresponding to SMBASE.

Example E-1 describes one method of initializing the segment registers when SMBASE has been relocated. This method works if SMBASE is less than 1 Megabyte.

```
;read the value of SMBASE from the state save area
      mov   si,FEF8H     ;SMBASE slot in SMM state save area
      mov   eax,cs:[si]  ;copy SMBASE from SMBASE:FEF8H to eax

;scale the SMBASE value to a 16-bit quantity
      mov   cl,4
      ror   eax,cl       ;scaled value of SMBASE now in ax

;to load cs, execute a far jump to an address that has been stored
;at memory location PTR_ADDR

;store the SMBASE value and an offset to a memory location that can be used as
;an indirect jump address

      mov   di,PTR_ADDR  ;PTR_ADDR is the location used to
                         ;store the jump address
      mov   bx,OFFSET    ;OFFSET is the address where
                         ;execution continues after the
                         ;far jump

      mov   cs:[di],bx   ;store the offset for the far jump
      inc   di
      inc   di
      mov   cs:[di],ax   ;store the segment address for the
                         ;far jump, which is SMBASE
      mov   bx,PTR_ADDR  ;bx now contains the address of the
                         ;location holding the jump address

;initialize DS and ES with the correct address of SMBASE
      mov   ds,ax
      mov   es,ax

;execute a far jump instruction to load the CS register
      jmp   far [bx]     ;jump to address stored at memory
                         ;location pointed to by bx

;CS now contains the correct value of SMBASE, and execution continues from the
;address SMBASE:OFFSET
```

242202-N2

Example E-1. Initialization of Segment Registers within SMM

Accessing SMRAM

LOADING SMRAM WITH AN INITIAL SMI HANDLER

Under normal conditions, the SMRAM address space should only be accessible by the processor while it is in SMM mode. However, some provision must be made for providing the initial SMM interrupt handler routine.

Because System Management Mode must be transparent to all operating systems, the initial SMM handler must be loaded by the system BIOS. At some time during the power on sequence, the system BIOS will need to move the SMM handler routine from the BIOS ROM to the SMRAM. The system designer must provide a hardware mechanism that allows access to SMRAM while SMIACT# from the processor is inactive. One method would be to provide an I/O port in the memory controller that forces memory cycles at a given address to be relocated to the SMRAM. Once the initial SMM handler has been loaded to SMRAM, the I/O port would be disabled to protect against accidental accesses to SMRAM.

The system BIOS must provide an SMM handler at the address 38000H. If the system designer has chosen to take advantage of the SMRAM relocation feature of the processor, this handler must change the SMBASE register in the SMM state save. Next, the BIOS must move the full featured SMM handler to the new address. An SMI# must be generated in order to change the SMBASE register before the BIOS passes control to the operating system.

SMRAM HIDDEN FROM DMA AND BUS MASTERS

In a system that allows DMA or other devices to take control of the system bus, care must be taken to ensure that only the master processor can access SMRAM. If an external bus master requests use of the system bus (by asserting HOLD or BOFF#) while the processor is executing an SMM handler routine, the processor would respond by passing control of the bus to the requesting device. The system memory controller must redirect any memory accesses that are not generated by the processor to normal system memory as if SMIACT# was inactive.

DMA accesses to the SMRAM area must be redirected to the correct address space when the initialization routine is loading SMRAM, as well as when the processor is in SMM.

It is not recommended to block bus control requests when in SMM, because the increased bus access latency could cause compatibility issues with some software or expansion hardware.

ACCESSING SYSTEM MEMORY FROM WITHIN SMM

In order to enter a suspend state where power is removed from some or all of system memory, it is necessary for the processor to have access to the entire system address space from within SMM. Access to system memory from within SMM requires that the memory controller decode both SMIACT# and the processor address to determine accesses to SMRAM. Only those memory addresses that are defined as being SMRAM space would be directed to SMRAM. If SMRAM is located at an address that overlays normal system memory address space (see section 8.6.2, "SMRAM Interface"), the processor must have a method of accessing both SMRAM (for code reads) and system memory simultaneously.

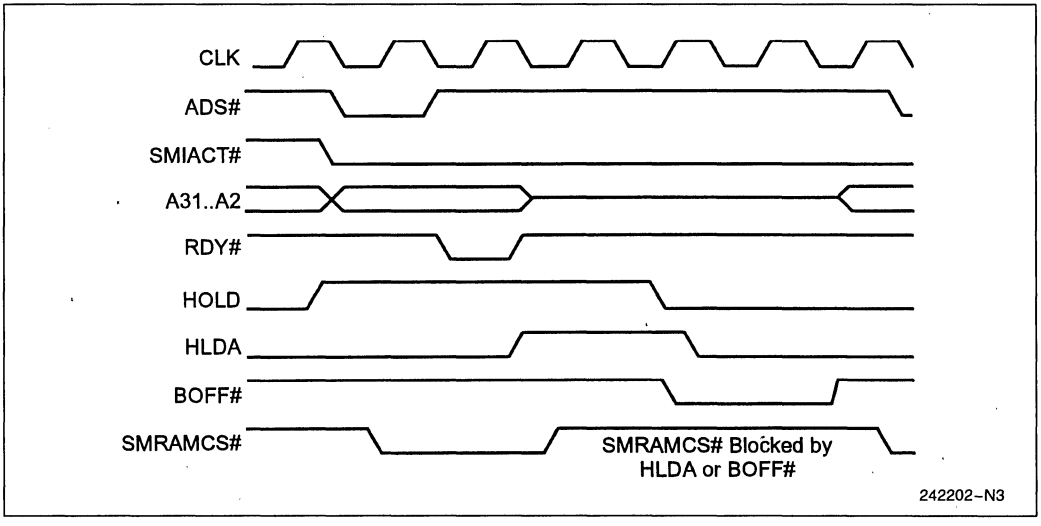


Figure E-1. Blocking Other Bus Masters from Accessing SMRAM

Ideally, a method of accessing system memory that is mapped underneath SMRAM would be provided by the system memory controller. The memory controller would provide a register that allows system memory at a given address to be remapped to a different address, which is not overlaid by SMRAM. When the SMM handler implements a suspend, it would first move all of system memory that is not underneath SMRAM to a non-volatile medium (such as a hard disk drive). Next, the SMRAM image would be transferred to the non-volatile medium. Finally, the memory underneath SMRAM would be accessed and copied to the non-volatile medium with a processor read to the remap address space, which is redirected to the overlaid system memory (see Figure E-2).

If the memory controller does not provide a method of accessing overlaid system memory, it is possible to implement a software procedure to accomplish the same goal. However, the software method is quite complex, and a hardware method is preferred. A description of the software method follows.

The ability to access the system memory that is located in the address space under SMRAM requires a method of resuming from SMM to a predetermined address space. This can be accomplished with the following procedure.

When resuming from SMM, the processor continues execution at the address contained in the CS and EIP slots within the SMM state save. However, the resume address cannot be changed by simply modifying the CS and EIP slots, because the processor will use the CS descriptor to determine the actual resume address. The descriptor registers are stored in reserved slots in the SMM state save, and they cannot be directly modified.

By replacing the suspend state save with a previously obtained image of a state save that returns to a known location, the SMM suspend handler can force a return to a given address:

1. During initial system power up, execute an SMI# from a predetermined address (the address immediately preceding the address to which you later wish to resume). This can be accomplished by generating an SMI# in response to an I/O instruction or executing a halt instruction and using an SMI# to exit the halt state.
2. Save the state save from this SMM to a safe location (SMRAM).
3. When the system needs to resume to a given address from some other SMI#, the stored state save can be substituted for the state save generated from that particular SMM.

4

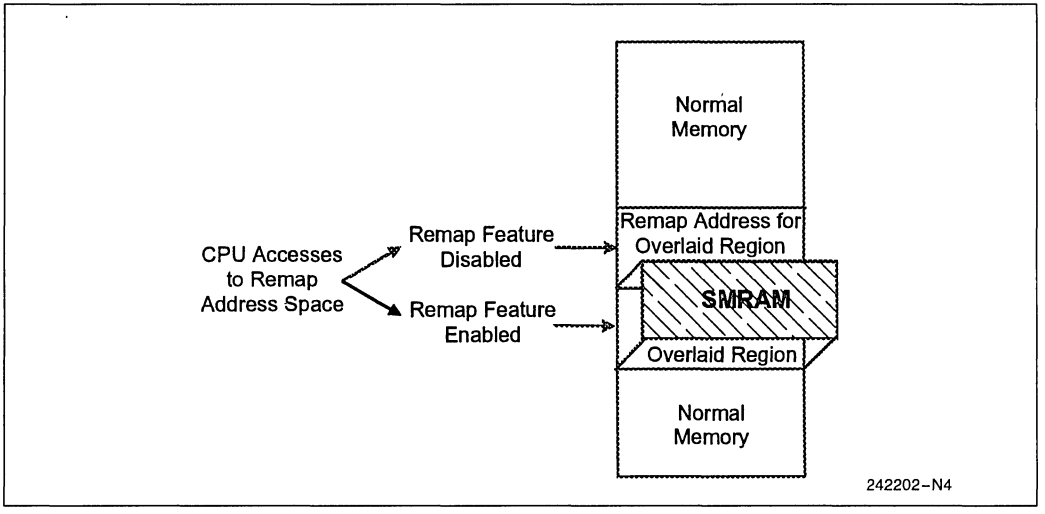


Figure E-2. Remapping Memory That Is Overlaid by SMRAM

Now that SMM can be resumed at a predetermined address, access the entire system memory space from within SMM before executing a suspend:

1. During a suspend SMM, save all system memory except that which is located underneath SMRAM to a specified (and reserved) section of the hard disk. The ability to access system memory requires the memory controller to decode both SMIACT# and the processor address, and direct a limited section (maybe 64K or 128K) of the processor address space to SMRAM. All other processor memory accesses should go to normal system memory.
2. Save the contents of the SMM state save to the hard disk.
3. Modify the SMM state save so that the RSM instruction will return to a predefined address, which is not in the application that was interrupted. The code at this address must contain the remainder of the suspend SMM handler. The predefined address can be anywhere in the processor address space, because the contents of system memory have already been saved to disk.
4. Execute an RSM instruction, which exits SMM and returns control to a predetermined address (which must contain the rest of the SMM suspend handler).
5. Save the rest of system memory (that which is located underneath SMRAM) to the hard disk. This address space can now be accessed with normal move instructions, because we are no longer in SMM.
6. Save a flag (in CMOS memory) indicating that the next reset should cause a resume from suspend.
7. Powerdown the memory (and possibly the processor).
8. When power is restored, the processor is reset and begins execution of the POST in BIOS. Early in the POST, the system should check the status of the suspend flag.
9. Load a preliminary SMM handler to location 38000H and generate an SMI#. The SMM handler should read the SMBASE slot from the SMM state save that was stored to hard disk. SMBASE is then modified to point to the final SMRAM location and the system resumes from SMM back to the system BIOS.
10. Restore the contents of system memory located underneath SMRAM from the hard disk.
11. Generate a second SMI#, which executes an SMM handler at the original value of SMBASE (before the suspend SMM). The SMM handler restores the contents of the rest of system memory from the hard disk, and then restores the original SMM state save to the SMM state save area in SMRAM, discarding the most recent SMM state save.
12. Execute an RSM instruction, which returns execution to the application that was interrupted by the suspend request.

Interrupts and Exceptions During SMM Handler Routines

To ensure transparency to existing system software, the SMM handler should not depend on interrupt or exception handlers provided by the operating system. However, in some cases it may be necessary to service interrupts or exceptions while in System Management Mode. In these cases, SMM compliant interrupt and exception handlers, as well as an SMM compliant interrupt vector table, should be provided.

SMM COMPLIANT VECTOR TABLES

An SMI# interrupt request can be generated while code is running under any of the other three processor operating modes (Real, Virtual-86, or Protected). When entering the SMM handler, the processor enters a pseudo-real mode, and the beginning of the interrupt vector table must be located at the address 00000000H. Before allowing any interrupts or exceptions to occur, the SMM handler routine must provide a valid interrupt vector table. Any code that is executed before setting up an SMM compliant interrupt vector table must be written carefully to ensure that no exceptions are generated.

The system memory controller could relocate accesses to the SMM interrupt vector table to a location within SMRAM. In this case, when SMIACT# is active, all accesses to the lowest 1 Kbyte of the processor address space would be redirected to SMRAM, which would contain an SMM compliant vector table that has already been initialized.

If the system memory controller does not redirect interrupt vector table reads to an address within SMRAM, there are three steps required to provide an SMM compliant interrupt vector table:

1. Save the contents of memory at address 00000000H to SMRAM.
2. Provide vectors for any possible interrupts or exceptions at the appropriate location in the vector table.
3. Restore the original memory contents from SMRAM before exiting the SMM handler routine.

INTERRUPTS AND SUBROUTINES WITH SMRAM RELOCATION

There is an additional issue that must be considered if interrupts or exceptions are to be executed within SMM and SMRAM has been relocated. Interrupt or subroutine calls from within SMM operate in a manner similar to Real Mode. When a subroutine is called or an interrupt is recognized, the 16-bit CS and IP registers are pushed onto the stack to provide a return address.

When SMRAM is relocated to an address space above 1M and an interrupt or subroutine call occurs, only 16 bits of the EIP register are pushed onto the stack. When returning from the subroutine or interrupt, the processor will vector to a location where the upper 16 bits of EIP are zero. This can be avoided for subroutines by using an address size override before calling the subroutine. However, the issue remains for interrupts.

Intel486™ DX, IntelDX2™, and IntelDX4™ Processor Floating-Point Operation and SMM

THE NEED TO SAVE THE FPU ENVIRONMENT

When the processor enters System Management Mode, the context information for the interrupted application is automatically saved to a specific state save address. When the SMM handler returns control to the interrupted application by executing the RSM instruction, the context information from the interrupted application is restored to the processor by reading from the state save location. This mechanism allows the SMM handler routine to modify most of the processor registers without the need to explicitly save them to memory. However, the registers in the processor's Floating-Point Unit (FPU) are not automatically saved when the processor enters SMM. If the SMM handler needs to modify any of the registers in the FPU, or if the register data will be lost due to entering a power down state, the SMM handler must first explicitly save the FPU state as it existed in the interrupted application.

There are two instances in which an SMM handler routine must be aware of the Floating-Point Unit (FPU):

1. When removing power from the processor / FPU for the purpose of executing a suspend sequence.
2. When the SMM handler uses FPU instructions.

In both of these cases, the SMM handler must save the state of the FPU as it was left by the interrupted application.

The information stored by the FPU state save instructions (FSAVE, FNSAVE, FSTENV, and FNSTENV) is dependent on the operating mode of the processor. The FPU state save instructions store the FPU state information in one of four formats: 16-bit Real Mode, 32-bit Real Mode, 16-bit Protected Mode, or 32-bit Protected Mode, depending on the processor operating mode. The content of the information saved also varies slightly, depending on the processor operating mode in which the save instruction was executed. For example, the 32-bit Protected Mode FNSAVE instruction saves the address of the last executed FPU instruction and its operands in the form of a segment selector and a 32-bit offset. In contrast, the 16-bit Real Mode FNSAVE instruction saves the address information in the form of a 20-bit physical address. Because the format with which the FPU state restore instructions (FRSTOR and FLDENV) recall the information is also dependent on the operating mode of the processor, the save and restore instructions must be executed from the same processor operating mode.

SAVING THE STATE OF THE FLOATING-POINT UNIT

When an SMM handler routine needs to save the state of the Floating-Point Unit, it must save all FPU state information necessary for the interrupted application to continue processing. This state information includes the contents of the Floating-Point Unit stack, which requires use of the FNSAVE or FSAVE instruction (FSTENV does not save the contents of the FPU stack). If the last executed non-control Floating-Point instruction caused an error (such as a divide by 0), the saved information must also include the address of the failing instruction and the addresses of any operands for that instruction. Without these addresses, it would be impossible for the FPU exception handler of the interrupted application to correct the error and restart the instruction.

The FNSAVE and FSAVE instructions differ in that FNSAVE does not wait for the FPU to check for an existing error condition before storing the FPU environment information. If there is an unmasked FPU exception condition pending, execution of the FSAVE instruction will force the processor to wait until the error condition is cleared by the software exception handler. Because the processor is in System Management Mode, the appropriate exception handler will not be available, and the FPU error would not be corrected in the manner expected by the interrupted application program. For this reason, the FNSAVE instruction should be used when saving the environment of the FPU within SMM.

Because the SMM handler does not know the processor mode in which the interrupted application was executing (16- or 32-bit, Real or Protected), the SMM handler must execute the FNSAVE instruction in a mode in which all FPU state information is stored. The 32-bit Protected Mode format of the FNSAVE instruction is a super set of all other formats of the FNSAVE instruction. Therefore, executing the 32-bit Protected Mode FNSAVE instruction ensures that all FPU state information will be saved.

Executing the FNSAVE instruction in 32-bit Protected Mode requires that the processor be temporarily placed in Protected Mode. Rather than perform all of the setup details and overhead necessary to place the processor into Protected Mode, including the initialization of all descriptors and descriptor tables, it is possible to temporarily place the processor into Protected Mode for the purpose of executing only a few carefully written instructions. This can be accomplished by setting the PE bit in the CR0 register, and then executing a short jump to clear the instruction pipelines.

It is important to note that any instruction that modifies a segment register will cause the processor to attempt to load a new descriptor from the descriptor table. (The occurrence of an interrupt or an exception would cause the processor to load a new descriptor, so interrupts must be disabled during this sequence.) Because neither the descriptors nor the descriptor table have been initialized, this would cause the system to crash. Therefore, all segment registers that are to be used in the FPU state save instructions must be initialized before entering Protected Mode.

Example E-2 gives an example of the code that can be used to place the processor in Protected Mode and save the FPU state.

Note that the no wait form (FNSAVE) of the save instruction must be used. In the event that the previous FPU instruction caused a Floating-Point error, we do not want to wait for this error to be serviced before executing the save instruction. Additionally, if the FSAVE instruction were used, the operand size override prefix would be incorrectly applied to the implicit WAIT instruction which precedes FSAVE, rather than to the save instruction itself.

Before exiting the SMM handler and returning to the interrupted application, the register contents of the Floating-Point Unit must be returned to their previous values. This can be accomplished by executing the 32-bit Protected Mode format of the FRSTOR instruction. Example E-3 gives an example code segment that can be used to restore the FPU to the state in which it was interrupted by the SMI request.

Note that the no wait form (FNRSTOR) of the restore instruction must be used. If the FRSTOR instruction were used, the operand size override prefix would be incorrectly applied to the implicit WAIT instruction which precedes FRSTOR, rather than to the save instruction itself.

Support for Power-Managed Peripherals

SHADOW REGISTERS

Before power is removed from any device, the state of that device must be saved in a protected memory space so that the device can be reinitialized to its previous state. If a peripheral contains a write only register, the value in that register can be recovered by providing shadow registers that are both readable and writeable.

These shadow registers should be updated every time the peripheral registers are written, but they have no function other than tracking the data written to a particular register.

```
;first initialize the registers used to store the state save information
      mov   dx,SEGMENT          ;SEGMENT is the segment to be used by
                                ;the save instruction,
      mov   ds,dx               ; normally it should point to SMRAM
      mov   si,OFFSET           ;OFFSET is the offset used in the save
                                ;instruction

;set the PE bit in CR0
      mov   eax,cr0             ;read the old value of CR0
      or    eax,00000001H       ;set the PE bit
      mov   cr0, eax

;enter protected mode by executing a short jump to clear the prefetch queue
      jmp   protect
protect:

;we can now save the state of the FPU in the protected mode format

      db    66H                 ;use an operand size override prefix
                                ;to set 32-bit format
      fnsave[si]                ;FPU state saved to SEGMENT:OFFSET

;now return to real mode to continue with the SMM handler (no jump is
;required)

      mov   eax,cr0             ;clear the PE bit in CR0
      and   eax,0FFFFFFFEH
      mov   cr0,eax
```

242202-N5

Example E-2. Saving the FPU State in 32-Bit Protected Mode

In addition to the write only registers in a system, there are several other registers that must be shadowed. Any device that requires registers to be programmed in a particular sequence must also have its registers shadowed. Examples in a typical personal computer include the programmable interrupt controller, the DMA controller, and the programmable timer/counter.

It is also possible to perform shadowing of some write only registers using SMM. Any time a write cycle is generated to a write only register, the system can generate an SMI#. The SMM handler can use the processor state information saved in the SMM state save to save the data from the interrupted I/O cycle to a predetermined location in the SMRAM space.

```
;first initialize the registers used to recall the state save information

      mov   dx,SEGMENT          ;SEGMENT is the segment to be used by
                                ;the restore instruction,
      mov   ds,dx               ;normally it should point to SMRAM
      mov   si,OFFSET           ;OFFSET is the offset used in the
                                ;restore instruction

;set the PE bit in CR0

      mov   eax,cr0             ;read the old value of CR0
      or    eax,00000001H       ;set the PE bit
      mov   cr0, eax

;enter protected mode by executing a short jump to clear the prefetch queue

      jmp   protect
protect:

;we can now recall the state of the FPU from the previous FNSAVE instruction
;(in the protected mode format)

      db    66H                 ;use an operand size override prefix
                                ;to set 32-bit format
      fnrstor     [si]          ;FPU state restored from
                                ;SEGMENT:OFFSET

;now return to real mode to continue with the SMM handler (no jump is
;required)

      mov   eax,cr0     ;clear the PE bit in CR0
      and   eax,FFFFFFFEH
      mov   cr0,eax
```

242202-N6

Example E-3. Restoring the FPU State from a 32-Bit Protected Mode Save

The information contained in the SMM state save can be used (with the knowledge that the SMI# was in response to an I/O write instruction) to determine both the address and the data of the interrupted write instruction. The SMM handler can examine the OPCODEs of previous instructions by decrementing the IP (or EIP) register. Once the correct OPCODE is determined, it can be used with the values in the EAX and DX slots of the SMM state save to update the information in the memory used to shadow the I/O register. I/O write instructions occur in one of three forms: 1) a write to an address that is specified in the OPCODE; 2) a write to an address contained in the DX register; or 3) a string write to an address contained in the DX register.

The I/O write instructions have the following OPCODEs:

Table E-1. I/O Write Instruction OPCODEs

Instruction	OPCODE	Notes
OUT x,al	E6x	x is the address of the I/O port
OUT x,ax	E7x	x is the address of the I/O port
OUT x,eax	E7x	x is the address of the I/O port
OUT dx,al	EE	
OUT dx,ax	EF	
OUT dx,eax	EF	
OUTSB	6E	
OUTSW	6F	
OUTSD	6F	

The SMM handler must know whether a particular I/O port is 16 or 32 bits in order to distinguish between 16- and 32-bit I/O write cycles.

The SMM handler can decrement the value of IP contained in the state save, and then examine the memory contents at that address. If the SMM handler knows that the last instruction was an I/O write instruction, and writes to I/O addresses 6EH, 6FH, 0EEH, and 0EFH will not cause an SMI#, it can use the SMM state save data for EAX and EDX to reconstruct the last instruction.

HANDLING INTERRUPTED I/O WRITE SEQUENCES

In a typical personal computer, there are several hardware devices that require the control registers for that device to be programmed in a particular order. For example, the interrupt controller, the DMA controller, the programmable timer/counter, the keyboard controller, and the real time clock all require a series of accesses to properly initialize the registers in that particular device. Some of these devices may require successive accesses to registers located at different addresses, while others may require several control registers to be programmed through write cycles to the same address.

If an SMI request interrupts an application that is in the process of initializing the registers in one of these devices, special care must be taken to ensure that the peripheral is returned to its original state when control is returned to the interrupted program. For some SMM handler events, it may be necessary to power down the device or change the state of a register within the device. In these cases, the SMM handler must return control to the interrupted application in such a way that the application can continue with the correct sequential access in the interrupted sequence.

To accomplish this, the SMM handler must restore the original values of all registers in the device, and restart the interrupted sequence so that the application may continue where it left off. This requires system hardware to shadow all registers that need to be accessed in the sequence, keep an index indicating which position in the sequence the register occupies, and keep a pointer so that SMM software knows to which register the last access was directed. This pointer would indicate the last register of each sequence that was programmed in the particular peripheral.

For example, programming the master interrupt controller requires a write to I/O port 20H (ICW1) followed by four write cycles to I/O port 21H (ICW2, ICW3, ICW4, and OCW1). If this sequence is interrupted by an SMI request, and the resulting SMM handler either modifies or powers down the interrupt controller, the SMM handler must return control to the interrupted application such that the following access to the interrupt controller would access the correct register in the sequence. System hardware must save the contents of each of the registers, as well as a pointer indicating which register was last written.

Before returning control to the interrupted application, the SMM handler must initialize ICW1–ICW4 and OCW1 to their previous values. It would then rewrite the appropriate registers so that the first access by the application program would be to the location in the sequence following the last location it programmed before it was interrupted by the SMI request.

A similar procedure must be followed for each of the peripherals that require control registers to be initialized in a particular order.

intel®

5

Memories and Peripherals

28F016XS 16-MBIT (1 MBIT x 16, 2 MBIT x 8) SYNCHRONOUS FLASH MEMORY

- **Effective Zero Wait-State Performance up to 33 MHz**
  — Synchronous Pipelined Reads
- **SmartVoltage Technology**
  — User-Selectable 3.3V or 5V V_{CC}
  — User-Selectable 5V or 12V V_{PP}
- **0.33 MB/sec Write Transfer Rate**
- **Configurable x8 or x16 Operation**
- **56-Lead TSOP and SSOP Type I Package**
- **Backwards-Compatible with 28F008SA Command-Set**
- **2 μA Typical Deep Power-Down**
- **1 mA Typical Active I_{CC} Current in Static Mode**
- **16 Separately-Erasable/Lockable 128-Kbyte Blocks**
- **1 Million Erase Cycles per Block**
- **State-of-the-Art 0.6 μm ETOX™ IV Flash Technology**

Intel's 28F016XS 16-Mbit Flash memory is a revolutionary architecture which is the ideal choice for designing truly revolutionary high-performance products. Combining very high read performance with the intrinsic non-volatility of flash memory, the 28F016XS eliminates the traditional redundant memory paradigm of shadowing code from a slow nonvolatile storage source to a faster execution memory, such as DRAM, for improved system performance. The innovative capabilities of the 28F016XS enable the design of direct-execute code and mass storage data/file flash memory systems.

The 28F016XS is the highest performance high density nonvolatile read/write flash memory solution available today. Its synchronous pipelined read interface, flexible V_{CC} and V_{PP} voltages, extended cycling, fast write and read performance, symmetrically blocked architecture, and selective block locking provide a highly flexible memory component suitable for resident flash component arrays on the system board or SIMMs. The synchronous pipelined interface and x8/x16 architecture of the 28F016XS allow easy interface with minimal glue logic to a wide range of processors/buses, providing effective zero wait-state read performance up to 33 MHz. The 28F016XS's dual read voltage allows the same component to operate at either 3.3V or 5.0V V_{CC}. Programming voltage at 5V V_{PP} minimizes external circuitry in minimal-chip, space critical designs, while the 12V V_{PP} option maximizes write/erase performance. Its high read performance combined with flexible block locking enable both storage and execution of operating systems/application software and fast access to large data tables. The 28F016XS is manufactured on Intel's 0.6 μm ETOX IV process technology.

5

290573-1

The complete datasheet can be ordered by calling 1-800-548-4725. Refer to order number 290532.

November 1995
Order Number: 290573-001

1.0 INTRODUCTION

The documentation of the Intel 28F016XS Flash memory device includes this datasheet, a detailed user's manual, a number of application notes and design tools, all of which are referenced at the end of this datasheet.

The datasheet is intended to give an overview of the chip feature-set and of the operating AC/DC specifications. The 16-Mbit Flash Product Family User's Manual provides complete descriptions of the user modes, system interface examples and detailed descriptions of all principles of operation. It also contains the full list of software algorithm flowcharts, and a brief section on compatibility with the Intel 28F008SA.

Significant 28F016XS feature revisions occurred between datasheet revisions 290532-001 and 290532-002. These revisions center around removal of the following features:

- All page buffer operations (read, write, programming, Upload Device Information)
- Command queuing
- Software Sleep and Abort
- Erase all Unlocked Blocks and Two-Byte Write
- RY/BY# Configuration options

In addition, a significant 28F016XS change occurred between datasheet revisions 290532-002 and 290532-003. This change centers around the addition of a 3/5# in to the device's pinout configuration.

Intel recommends that all customers obtain the latest revisions of 28F016XS documentation.

1.1 Product Overview

The 28F016XS is a high-performance, 16-Mbit (16,777,216-bit) block erasable nonvolatile random access memory organized as either 1 Mword x 16 or 2 Mbyte x 8, subdivided into even and odd banks. Address A_1 makes the bank selection. The 28F016XS includes sixteen 128-Kbyte (131,072 byte) blocks or sixteen 64-Kword (65,536 word) blocks.

The implementation of a new architecture, with many enhanced features, will improve the device operating characteristics and result in greater product reliability and ease-of-use as compared to other flash memories. Significant features of the 28F016XS as compared to previous asynchronous flash memories include:

- Synchronous Pipelined Read Interface
- Significantly Improved Read and Write Performance
- SmartVoltage Technology
  - — Selectable 3.3V or 5.0 V_{CC}
  - — Selectable 5.0V or 12.0 V_{PP}
- Block Write/Erase Protection

The 28F016XS's synchronous pipelined interface dramatically raises read performance far beyond previously attainable levels. Addresses are synchronously latched and data is read from a 28F016XS bank every 30 ns (5V V_{CC}, SFI Configuration = 2). This capability translates to 0-wait-state reads at clock rates up to 33 MHz at 5V V_{CC}, after an initial address pipeline fill delay and assuming even and odd banks within the flash memory are alternately accessed. Data is latched and driven valid 20 ns (t_{CHQV}) after a rising CLK edge. The 28F016XS is capable of operating up to 66 MHz (5V V_{CC}); its programmable SFI Configuration enables system design flexibility, optimizing the 28F016XS to a specific system clock frequency.

The SFI Configuration optimizes the 28F016XS for a wide range of system operating frequencies. The default SFI Configuration is 4, which allows system boot from the 28F016XS at any frequency up to 66 MHz at 5V V_{CC}. After initiating an access, data is latched and begins driving on the data outputs after a CLK count corresponding to the SFI Configuration has elapsed. The 28F016XS will hold data valid until CE# or OE# is deactivated or a CLK count corresponding to the SFI Configuration for a subsequent access has elapsed.

The CLK and ADV# inputs, new to the 28F016XS in comparison to previous flash memories, control address latching and device synchronization during read operations. The CLK input controls the device latencies, times out the SFI Configuration counter and synchronizes data outputs. ADV# indicates the presence of a valid address on the 28F016XS

address inputs. During read operations, addresses are latched and accesses are initiated on a rising CLK edge in conjunction with ADV# low. Both CLK and ADV# are ignored by the 28F016XS during command/data write sequences.

The 28F016XS incorporates SmartVoltage technology, providing V_{CC} operation at both 3.3V and 5.0V and program and erase capability at V_{PP} = 12.0V or 5.0V. Operating at V_{CC} = 3.3V, the 28F016XS consumes less than one half the power consumption at 5.0V V_{CC}, while 5.0V V_{CC} provides highest read performance capability. V_{PP} operation at 5.0V eliminates the need for a separate 12.0V converter, while the V_{PP} = 12.0V option maximizes write/erase performance. In addition to the flexible program and erase voltages, the dedicated V_{PP} gives complete code protection with $V_{PP} \leq V_{PPLK}$.

A 3/5# input pin configures the device's internal circuitry for optimal 3.3V or 5.0V read/write operation.

A Command User Interface (CUI) serves as the system interface between the microprocessor or microcontroller and the internal memory operation.

Internal Algorithm Automation allows byte/word writes and block erase operations to be executed using a Two-Write command sequence to the CUI in the same way as the 28F008SA 8-Mbit FlashFile™ memory.

Software locking of memory blocks is an added feature of the 28F016XS as compared to the 28F008SA. The 28F016XS provides selectable block locking to protect code or data such as direct-executable operating systems or application code. Each block has an associated nonvolatile lock-bit which determines the lock status of the block. In addition, the 28F016XS has a master Write Protect pin (WP#) which prevents any modifications to memory blocks whose lock-bits are set.

Writing of memory data is performed in either byte or word increments, typically within 6 µs at 12.0V V_{PP}, which is a 33% improvement over the 28F008SA. A block erase operation erases one of the 16 blocks in typically 1.2 sec, independent of the other blocks.

Each block can be written and erased a minimum of 100,000 cycles. Systems can achieve one million Block Erase Cycles by providing wear-leveling algorithms and graceful block retirement. These techniques have already been employed in many flash file systems and hard disk drive designs.

All operations are started by a sequence of Write commands to the device. Three Status Registers (described in detail later in this datasheet) and a RY/BY# output pin provide information on the progress of the requested operation.

The following Status Registers are used to provide device and WSM operation information to the user:

- A Compatible Status Register (CSR) which is 100% compatible with the 28F008SA FlashFile memory Status Register. The CSR, when used alone, provides a straightforward upgrade capability to the 28F016XS from a 28F008SA-based design.
- A Global Status Register (GSR) which also informs the system of overall Write State Machine (WSM) status.
- 16 Block Status Registers (BSRs) which provide block-specific status information such as the block lock-bit status.

The 28F016XS incorporates an open drain RY/BY# output pin. This feature allows the user to OR-tie many RY/BY# pins together in a multiple memory configuration such as a Resident Flash Array.

The 28F016XS also incorporates a dual chip-enable function with two input pins, CE_0# and CE_1#. These pins have exactly the same functionality as the regular chip-enable pin, CE#, on the 28F008SA. For minimum chip designs, CE_1# may be tied to ground and system logic may use CE_0# as the chip enable input. The 28F016XS uses the logical combination of these two signals to enable or disable the entire chip. Both CE_0# and CE_1# must be active low to enable the device. If either one becomes inactive, the chip will be disabled. This feature, along with the open drain RY/BY# pin, allows the system designer to reduce the number of control pins used in a large array of 16-Mbit devices.

The BYTE# pin allows either x8 or x16 read/writes to the 28F016XS. BYTE# at logic low selects 8-bit mode with address A_0 selecting between low byte and high byte. On the other hand, BYTE# at logic high enables 16-bit operation with address A_1 becoming the lowest order address and address A_0 is not used (don't care). A device block diagram is shown in Figure 1.

The 28F016XS incorporates an Automatic Power Saving (APS) feature, which substantially reduces the active current when the device is in static mode of operation (addresses not switching). In APS mode, the typical I_{CC} current is 1 mA at 5.0V (3 mA at 3.3V).

A deep power-down mode of operation is invoked when the RP# (called PWD# on the 28F008SA) pin transitions low. This mode brings the device power consumption to less than 2.0 μA, typically, and provides additional write protection by acting as a device reset pin during power transitions. A reset time of 300 ns (5V V_{CC}) is required from RP# switching high before latching an address into the 28F016XS. In the deep power-down state, the WSM is reset (any current operation will abort) and the CSR, GSR and BSR registers are cleared.

A CMOS standby mode of operation is enabled when either CE_0# or CE_1# transitions high and RP# stays high with all input control pins at CMOS levels. In this mode, the device typically draws an I_{CC} standby current of 70 μA at 5V V_{CC}.

The 28F016XS is available in 56-Lead, 1.2 mm thick, 14 mm x 20 mm TSOP and 1.8 mm thick, 16 mm x 23.7 mm SSOP Type I packages. The form factor and pinout of these two packages allow for very high board layout densities.

ADDITIONAL INFORMATION

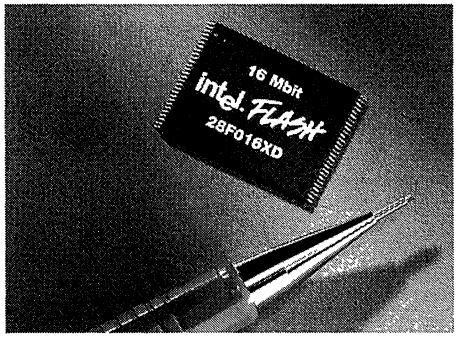
Order Number	Document/Tool
297372	16-Mbit Flash Product Family User's Manual
292126	AP-360, "28F008SA Software Drivers"
292147	AP-398, "Designing with the 28F016XS"
292146	AP-600, "Performance Benefits and Power/Energy Savings of 28F016XS-Based System Designs"
292163	AP-610, "Flash Memory In-System Code and Data Update Techniques"
292165	AB-62, "Compiling Optimized Code for Flash Memories"
297500	"Interfacing the 28F016XS to the i960® Microprocessor Family"
297504	"Interfacing the 28F016XS to the Intel486™ Microprocessor Family"
294016	ER-33, "ETOX™ Flash Memory Technology—Insight to Intel's Fourth Generation Process Innovation"
297508	FLASHBuilder Utility
Contact Intel/Distribution Sales Office	28F016XS Benchmark Utility
Contact Intel/Distribution Sales Office	Flash Cycling Utility
Contact Intel/Distribution Sales Office	28F016XS iBIS Model
Contact Intel/Distribution Sales Office	28F016XS VHDL Model
Contact Intel/Distribution Sales Office	28F016XS Timing Designer Library Files
Contact Intel/Distribution Sales Office	28F016XS Orcad/Viewlogic Schematic Symbols

28F016XD
16-MBIT (1 MBIT x 16)
DRAM-INTERFACE FLASH MEMORY

- **85 ns Access Time (t_{RAC})**
  - **— Supports both Standard and Fast-Page-Mode Accesses**
- **Multiplexed Address Bus**
  - **— RAS# and CAS# Control Inputs**
- **No-Glue Interface to Many Memory Controllers**
- **SmartVoltage Technology**
  - **— User-Selectable 3.3V or 5V V_{CC}**
  - **— User-Selectable 5V or 12V V_{PP}**
- **0.33 MB/sec Write Transfer Rate**
- **x16 Architecture**
- **56-Lead TSOP Type I Package**
- **Backwards-Compatible with 28F008SA Command Set**
- **2 μA Typical Deep Power-Down Current**
- **1 mA Typical I_{CC} Active Current in Static Mode**
- **32 Separately-Erasable/Lockable 64-Kbyte Blocks**
- **1 Million Erase Cycles per Block**
- **State-of-the-Art 0.6 μm ETOX™ IV Flash Technology**

Intel's 28F016XD 16-Mbit Flash memory is a revolutionary architecture which is the ideal choice for designing truly revolutionary high-performance products. Combining its DRAM-like read performance and interface with the intrinsic nonvolatility of flash memory, the 28F016XD eliminates the traditional redundant memory paradigm of shadowing code from a slow nonvolatile storage source to a faster execution memory, such as DRAM, for improved system performance. The innovative capabilities of the 28F016XD enable the design of direct-execute code and mass storage data/file flash memory systems.

The 28F016XD's DRAM-like interface with a multiplexed address bus, flexible V_{CC} and V_{PP} voltages, power saving features, extended cycling, fast write and read performance, symmetrically blocked architecture, and selective block locking provide a highly flexible memory component suitable for resident flash component arrays on the system board or SIMMs. The DRAM-like interface with RAS# and CAS# control inputs allows for easy migration to flash memory in existing DRAM-based systems. The 28F016XD's dual read voltage allows the same component to operate at either 3.3V or 5.0V V_{CC}. Programming voltage at 5V V_{PP} minimizes external circuitry in minimal-chip, space critical designs, while the 12V V_{PP} option maximizes write/erase performance. The x16 architecture allows optimization of the memory-to-processor interface. Its high read performance combined with flexible block locking enable both storage and execution of operating systems/application software and fast access to large data tables. The 28F016XD is manufactured on Intel's 0.6 μm ETOX™ IV process technology.



290574-1

The complete datasheet can be ordered by calling 1-800-548-4725. Refer to order number 290533.

November 1995
Order Number: 290574-001

1.0 INTRODUCTION

The documentation of the Intel 28F016XD flash memory device includes this datasheet, a detailed user's manual, and a number of application notes and design tools, all of which are referenced at the end of this datasheet.

The datasheet is intended to give an overview of the chip feature-set and of the operating AC/DC specifications. The 16-Mbit Flash Product Family User's Manual provides complete descriptions of the user modes, system interface examples and detailed descriptions of all principles of operation. It also contains the full list of software algorithm flowcharts, and a brief section on compatibility with the Intel 28F008SA.

Significant 28F016XD feature revisions occurred between datasheet revisions 290533-001 and 290533-002. These revisions center around removal of the following features:

- All page buffer operations (read, write, programming, Upload Device Information)
- Command queuing
- Software Sleep and Abort
- Erase all Unlocked Blocks
- Device Configuration command

In addition, a significant 28F016XD change occurred between datasheet revisions 290532-002 and 290532-003. This change centers around the addition of a 3/5# pin to the device's pinout configuration.

Intel recommends that all customers obtain the latest revisions of 28F016XD documentation.

1.1 Product Overview

The 28F016XD is a high-performance, 16-Mbit (16,777,216-bit) block erasable, nonvolatile random access memory, organized as 1 Mword x 16. The 28F016XD includes thirty-two 32-KW (32,768 word) blocks.

The implementation of a new architecture, with many enhanced features, will improve the device operating characteristics and result in greater product reliability and ease-of-use as compared to other flash memories. Significant features of the 28F016XD include:

- No-Glue Interface to Memory Controllers
- Improved Word Write Performance
- SmartVoltage Technology
  - — Selectable 3.3V or 5.0V V_{CC}
  - — Selectable 5.0V or 12.0V V_{PP}
- Block Write/Erase Protection

The 28F016XD's multiplexed address bus with RAS# and CAS# inputs allows for a "No Glue" interface to many existing in-system memory controllers. As such, 28F016XD-based SIMMs (72-pin JEDEC Standard) offer attractive advantages over their DRAM counterparts in many applications. For more information on 28F016XD-based SIMM designs, see the application note referenced at the end of this datasheet.

The 28F016XD incorporates SmartVoltage technology, providing V_{CC} operation at both 3.3V and 5.0V and program and erase capability at V_{PP} = 12.0V or 5.0V. Operating at V_{CC} = 3.3V, the 28F016XD consumes less than 60% of the power consumption at 5.0V V_{CC}, while 5.0V V_{CC} provides the highest read performance capability. V_{PP} = 5.0V operation eliminates the need for a separate 12.0V converter, while V_{PP} = 12.0V maximizes write/erase performance. In addition to the flexible program and erase voltages, the dedicated V_{PP} gives complete code protection with $V_{PP} \leq V_{PPLK}$.

A 3/5# input pin configures the device's internal circuitry for optimal 3.3V or 5.0V read/write operation.

A Command User Interface (CUI) serves as the system interface between the microprocessor or microcontroller and the internal memory operation.

Internal Algorithm Automation allows word writes and block erase operations to be executed using a Two-Write command sequence to the CUI in the same way as the 28F008SA 8-Mbit FlashFile™ memory.

Software Locking of Memory Blocks is an added feature of the 28F016XD as compared to the 28F008SA. The 28F016XD provides selectable block locking to protect code or data such as direct-executable operating systems or application code.

Each block has an associated nonvolatile lock-bit which determines the lock status of the block. In addition, the 28F016XD has a master Write Protect pin (WP#) which prevents any modifications to memory blocks whose lock-bits are set.

Writing of memory data is performed in word increments typically within 6 μs (12.0V V_{PP})—a 33% improvement over the 28F008SA. A block erase operation erases one of the 32 blocks in typically 0.6 sec (12.0V V_{PP}), independent of the other blocks, which is about a 65% improvement over the 28F008SA.

Each block can be written and erased a minimum of 100,000 cycles. Systems can achieve one million Block Erase Cycles by providing wear-leveling algorithms and graceful block retirement. These techniques have already been employed in many flash file systems and hard disk drive designs.

All operations are started by a sequence of Write commands to the device. Three types of Status Registers (described in detail later in this datasheet) and a RY/BY# output pin provide information on the progress of the requested operation.

The following Status Registers are used to provide device and WSM information to the user:

- A Compatible Status Register (CSR) which is 100% compatible with the 28F008SA FlashFile memory Status Register. The CSR, when used alone, provides a straightforward upgrade capability to the 28F016XD from a 28F008SA-based design.
- A Global Status Register (GSR) which also informs the system of overall Write State Machine (WSM) status.
- 32 Block Status Registers (BSRs) which provide block-specific status information such as the block lock-bit status.

The 28F016XD incorporates an open drain RY/BY# output pin. This feature allows the user to OR-tie many RY/BY# pins together in a multiple memory configuration such as a Resident Flash Array.

The 28F016XD is specified for a maximum fast page mode cycle time of 65 ns ($t_{PC,R}$) at 5.0V operation (4.75V to 5.25V) over the commercial temperature range (0°C to +70°C). A corresponding maximum fast page mode cycle time of 75 ns at 3.3V (3.0V to 3.6V and 0°C to +70°C) is achieved for reduced power consumption applications.

The 28F016XD incorporates an Automatic Power Saving (APS) feature, which substantially reduces the active current when the device is in static mode of operation (addresses not switching). In APS mode, the typical I_{CC} current is 1 mA at 5.0V (3.0 mA at 3.3V).

A deep power-down mode of operation is invoked when the RP# (called PWD# on the 28F008SA) pin transitions low. This mode brings the device power consumption to less than 2.0 μA, typically, and provides additional write protection by acting as a device reset pin during power transitions. A reset time of 300 ns (5.0V V_{CC} operation) is required from RP# switching high until dropping RAS#. In the deep power-down state, the WSM is reset (any current operation will abort) and the CSR, GSR and BSR registers are cleared.

A CMOS standby mode of operation is enabled when RAS# and CAS# transition high and RP# stays high with all input control pins at CMOS levels. In this mode, the device typically draws an I_{CC} standby current of 70 μA at 5V V_{CC}.

The 28F016XD is available in a 56-Lead, 1.2 mm thick, 14 mm x 20 mm TSOP Type I package. This form factor and pinout allow for very high board layout densities.

ADDITIONAL INFORMATION

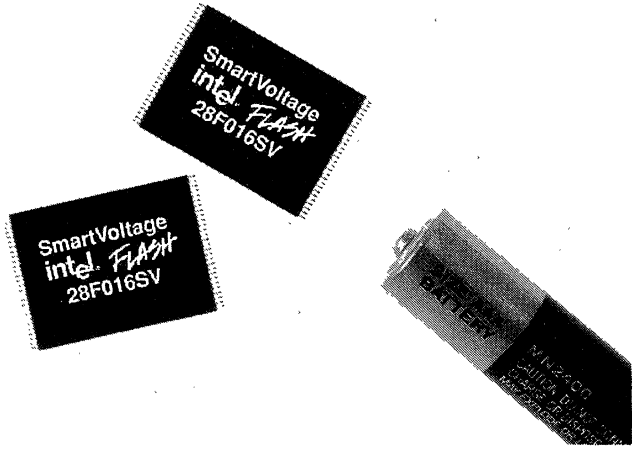
Order Number	Document/Tool
297372	16-Mbit Flash Product Family User's Manual
292152	AB-58, "28F016XD-Based SIMM Designs"
292165	AB-62, "Compiled Code Optimizations for Flash Memories"
292092	AP-357, "Power Supply Solutions for Flash Memory"
292123	AP-374, "Flash Memory Write Protection Techniques"
292126	AP-377, "16-Mbit Flash Product Family Software Drivers, 28F016SA/SV/XD/XS"
292131	AP-384, "Designing with the 28F016XD"
292163	AP-610, "Flash Memory In-System Code and Data Update Techniques"
292168	AP-614, "Adapting DRAM Based Designs for the 28F016XD"
294016	ER-33, "ETOX™ Flash Memory Technology—Insight to Intel's Fourth Generation Process Innovation"
297508	FLASHBuilder Utility
Contact Intel/Distribution Sales Office	28F016XD Benchmark Utility
Contact Intel/Distribution Sales Office	Flash Cycling Utility
Contact Intel/Distribution Sales Office	28F016XD iBIS Models
Contact Intel/Distribution Sales Office	28F016XD VHDL Model
Contact Intel/Distribution Sales Office	28F016XD Timing Designer Library Files
Contact Intel/Distribution Sales Office	28F016XD Orcad and ViewLogic Schematic Symbols

28F016SV
16-MBIT (1 MBIT x 16, 2 MBIT x 8)
FlashFile™ MEMORY

- **SmartVoltage Technology**
  - **— User-Selectable 3.3V or 5V V_{CC}**
  - **— User-Selectable 5V or 12V V_{PP}**
- **65 ns Access Time**
- **1 Million Erase Cycles per Block**
- **30.8 MB/sec Burst Write Transfer Rate**
- **0.48 MB/sec Sustainable Write Transfer Rate**
- **Configurable x8 or x16 Operation**
- **56-Lead TSOP and SSOP Type I Packages**
- **Backwards-Compatible with 28F016SA, 28F008SA Command Set**
- **Revolutionary Architecture**
  - **— Multiple Command Execution**
  - **— Write during Erase**
  - **— Command Super-Set of the Intel 28F008SA**
  - **— Page Buffer Write**
- **2 μA Typical Deep Power-Down**
- **32 Independently Lockable Blocks**
- **State-of-the-Art 0.6 μm ETOX™ IV Flash Technology**

Intel's 28F016SV 16-Mbit FlashFile™ memory is a revolutionary architecture which is the ideal choice for designing embedded direct-execute code and mass storage data/file flash memory systems. With innovative capabilities, low-power operation, user-selectable V_{PP} voltage and high read/write performance, the 28F016SV enables the design of truly mobile, high-performance personal computing and communications products.

The 28F016SV is the highest density, highest performance nonvolatile read/write solution for solid-state storage applications. Its symmetrically blocked architecture (100% compatible with the 28F008SA 8-Mbit and 28F016SA 16-Mbit FlashFile memories), extended cycling, flexible V_{CC} and V_{PP} voltage (SmartVoltage technology), fast write and read performance and selective block locking provide a highly flexible memory component suitable for Resident Flash Arrays, high-density memory cards and PCMCIA-ATA flash drives. The 28F016SV's dual read voltage enables the design of memory cards which can interchangeably be read/written in 3.3V and 5.0V systems. Its x8/x16 architecture allows optimization of the memory-to-processor interface. The flexible block locking option enables bundling of executable application software in a Resident Flash Array or memory card. The 28F016SV is manufactured on Intel's 0.6 μm ETOX IV process technology.



290575-1

The complete datasheet can be ordered by calling 1-800-548-4725. Refer to order number 290528.

November 1995
Order Number: 290575-001

1.0 INTRODUCTION

The documentation of the Intel 28F016SV memory device includes this datasheet, a detailed user's manual, and a number of application notes and design tools, all of which are referenced at the end of this datasheet.

The datasheet is intended to give an overview of the chip feature-set and of the operating AC/DC specifications. The 16-Mbit Flash Product Family User's Manual provides complete descriptions of the user modes, system interface examples and detailed descriptions of all principles of operation. It also contains the full list of software algorithm flowcharts, and a brief section on compatibility with the Intel 28F008SA.

A significant 28F016SV change occurred between datasheet revisions 290528-003 and 290528-004. This change centers around the addition of a 3/5# pin to the device's pinout configuration. Intel recommends that all customers obtain the latest revisions of 28F016SV documentation.

1.1 Enhanced Features

The 28F016SV is backwards compatible with the 28F016SA and offers the following enhancements:

- SmartVoltage Technology
  - — Selectable 5.0V or 12.0V V_{PP}
- V_{PP} Level Bit in Block Status Register
- Additional RY/BY# Configuration
  - — Pulse-On-Write/Erase
- Additional Upload Device Information Command Feedback
  - — Device Proliferation Code
  - — Device Configuration Code

1.2 Product Overview

The 28F016SV is a high-performance, 16-Mbit (16,777,216-bit) block erasable, nonvolatile random access memory, organized as either 1 Mword x 16 or 2 Mbyte x 8. The 28F016SV includes thirty-two 64-KB (65,536 byte) blocks or thirty-two 32-KW (32,768 word) blocks.

The implementation of a new architecture, with many enhanced features, will improve the device operating characteristics and result in greater product reliability and ease-of-use.

The 28F016SV incorporates SmartVoltage technology, providing V_{CC} operation at both 3.3V and 5.0V and program and erase capability at V_{PP} = 12.0V or 5.0V. Operating at V_{CC} = 3.3V, the 28F016SV consumes approximately one half the power consumption at 5.0V V_{CC}, while 5.0V V_{CC} provides the highest read performance capability. V_{PP} = 5.0V operation eliminates the need for a separate 12.0V converter, while V_{PP} = 12.0V maximizes write/erase performance. In addition to the flexible program and erase voltages, the dedicated V_{PP} gives complete code protection with $V_{PP} \leq V_{PPLK}$.

A 3/5# input pin configures the device's internal circuitry for optimal 3.3V or 5.0V read/write operation.

A Command User Interface (CUI) serves as the system interface between the microprocessor or microcontroller and the internal memory operation.

Internal Algorithm Automation allows byte/word writes and block erase operations to be executed using a Two-Write command sequence to the CUI in the same way as the 28F008SA 8-Mbit FlashFile memory.

A super-set of commands has been added to the basic 28F008SA command-set to achieve higher write performance and provide additional capabilities. These new commands and features include:

- Page Buffer Writes to Flash
- Command Queuing Capability
- Automatic Data Writes during Erase
- Software Locking of Memory Blocks
- Two-Byte Successive Writes in 8-bit Systems
- Erase All Unlocked Blocks

Writing of memory data is performed in either byte or word increments typically within 6 µs (12.0V V_{PP})—a 33% improvement over the 28F008SA. A block erase operation erases one of the 32 blocks in typically 0.6 sec (12.0V V_{PP}), independent of the other blocks, which is about a 65% improvement over the 28F008SA.

Each block can be written and erased a minimum of 100,000 cycles. Systems can achieve one million Block Erase Cycles by providing wear-leveling algorithms and graceful block retirement. These techniques have already been employed in many flash file systems and hard disk drive designs.

The 28F016SV incorporates two Page Buffers of 256 bytes (128 words) each to allow page data writes. This feature can improve a system write performance by up to 4.8 times over previous flash memory devices, which have no Page Buffers.

All operations are started by a sequence of Write commands to the device. Three Status Registers (described in detail later in this datasheet) and a RY/BY# output pin provide information on the progress of the requested operation.

While the 28F008SA requires an operation to complete before the next operation can be requested, the 28F016SV allows queuing of the next operation while the memory executes the current operation. This eliminates system overhead when writing several bytes in a row to the array or erasing several blocks at the same time. The 28F016SV can also perform write operations to one block of memory while performing erase of another block.

The 28F016SV provides selectable block locking to protect code or data such as Device Drivers, PCMCIA card information, ROM-Executable O/S or Application Code. Each block has an associated nonvolatile lock-bit which determines the lock status of the block. In addition, the 28F016SV has a master Write Protect pin (WP#) which prevents any modifications to memory blocks whose lock-bits are set.

The 28F016SV contains three types of Status Registers to accomplish various functions:

- A Compatible Status Register (CSR) which is 100% compatible with the 28F008SA FlashFile memory Status Register. The CSR, when used alone, provides a straightforward upgrade capability to the 28F016SV from a 28F008SA-based design.
- A Global Status Register (GSR) which informs the system of command Queue status, Page Buffer status, and overall Write State Machine (WSM) status.
- 32 Block Status Registers (BSRs) which provide block-specific status information such as the block lock-bit status.

The 28F016SV incorporates an open drain RY/BY# output pin. This feature allows the user to OR-tie many RY/BY# pins together in a multiple memory configuration such as a Resident Flash Array.

Other configurations of the RY/BY# pin are enabled via special CUI commands and are described in detail in the 16-Mbit Flash Product Family User's Manual.

The 28F016SV's enhanced Upload Device Information command provides access to additional information that the 28F016SA previously did not offer. This command uploads the Device Revision Number, Device Proliferation Code and Device Configuration Code to the page buffer. The Device Proliferation Code for the 28F016SV is 01H, and the Device Configuration Code identifies the current RY/BY# configuration. A subsequent Page Buffer Swap and Page Buffer Read command sequence is necessary to read the correct device information.

The 28F016SV also incorporates a dual chip-enable function with two input pins, CE_0# and CE_1#. These pins have exactly the same functionality as the regular chip-enable pin, CE#, on the 28F008SA. For minimum chip designs, CE_1# may be tied to ground and system logic may use CE_0# as the chip enable input. The 28F016SV uses the logical combination of these two signals to enable or disable the entire chip. Both CE_0# and CE_1# must be active low to enable the device. If either one becomes inactive, the chip will be disabled. This feature, along with the open drain RY/BY# pin, allows the system designer to reduce the number of control pins used in a large array of 16-Mbit devices.

The BYTE# pin allows either x8 or x16 read/writes to the 28F016SV. BYTE# at logic low selects 8-bit mode with address A_0 selecting between the low byte and high byte. On the other hand, BYTE# at logic high enables 16-bit operation with address A_1 becoming the lowest order address and address A_0 is not used (don't care).

The 28F016SV is specified for a maximum access time of 65 ns (t_{ACC}) at 5.0V operation (4.75V to 5.25V) over the commercial temperature range (0°C to +70°C). A corresponding maximum access time of 75 ns at 3.3V (3.0V to 3.6V and 0°C to +70°C) is achieved for reduced power consumption applications.

The 28F016SV incorporates an Automatic Power Saving (APS) feature, which substantially reduces the active current when the device is in static mode of operation (addresses not switching). In APS mode, the typical I_{CC} current is 1 mA at 5.0V (3.0 mA at 3.3V).

A deep power-down mode of operation is invoked when the RP# (called PWD# on the 28F008SA) pin transitions low. This mode brings the device power consumption to less than 2.0 μA, typically, and provides additional write protection by acting as a device reset pin during power transitions. A reset time of 400 ns (5.0V V_{CC} operation) is required from RP# switching high until outputs are again valid. In the Deep Power-Down state, the WSM is reset (any current operation will abort) and the CSR, GSR and BSR registers are cleared.

A CMOS standby mode of operation is enabled when either CE_0# or CE_1# transitions high and RP# stays high with all input control pins at CMOS levels. In this mode, the device typically draws an I_{CC} standby current of 70 μA at 5V V_{CC}.

The 28F016SV will be available in 56-lead, 1.2 mm thick, 14 mm x 20 mm TSOP and 56-lead, 1.8 mm thick, 16 mm x 23.7 SSOP Type I packages. The form factor and pinout of these two packages allow for very high board layout densities.

ADDITIONAL INFORMATION

Order Number	Document/Tool
297372	16-Mbit Flash Product Family User's Manual
290435	28F008SA Datasheet
290490	DD28F032SA Datasheet
292092	AP-357 "Power Supply Solutions for Flash Memory"
292123	AP-374 "Flash Memory Write Protection Techniques"
292124	AP-375 "Upgrade Considerations from the 28F008SA to the 28F016SA"
292126	AP-377 "16-Mbit Flash Product Family Software Drivers, 28F016SA/ 28F016SV/28F016XS/28F016XD"
292127	AP-378 "System Optimization Using the Enhanced Features of the 28F016SA"
292144	AP-393 "28F016SV Compatibility with 28F016SA"
292159	AP-607 "Multi-Site Layout Planning with Intel's FlashFile Components, including ROM Capability
292163	AP-610 "Flash Memory In-System Code and Data Update Techniques"
292165	AB-62 "Compiled Code Optimizations for Flash Memories"
294016	ER-33 "ETOX Flash Memory Technology—Insight to Intel's Fourth Generation Process Innovation"
297508	FLASHBuilder Utility
Contact Intel/Distribution Sales Office	Flash Cycling Utility
Contact Intel/Distribution Sales Office	28F016SV iBIS Model
Contact Intel/Distribution Sales Office	28F016SV VHDL Model
Contact Intel/Distribution Sales Office	28F016SV Timing Designer Library Files
Contact Intel/Distribution Sales Office	28F016SV Orcad and ViewLogic Schematic Symbols

intel®

AB-60

APPLICATION BRIEF

2/4/8-Mbit SmartVoltage Boot Block Flash Memory Family Overview

5

COLLIN K. ONG
TECHNICAL MARKETING ENGINEER

October 1995

Order Number: 292154-002

1.0 INTRODUCTION

This document includes a feature overview, pinouts, and memory maps for Intel's SmartVoltage boot block family, including 2/4/8-Mbit densities. These products offer feature and function compatibility with the boot block architecture, along with the SmartVoltage technology (SVT) outlined below. Follow the design steps in Section 5.0 to upgrade 12V V_{PP} designs to SVT.

2.0 BOOT BLOCK ARCHITECTURE

Intel's boot block architecture products offer the familiar features that optimize it for updateable firmware storage. These features include:

- Hardware-lockable boot block for secure kernel code storage
- Parameter blocks for parameter storage
- Main blocks for modular code updates, facilitating updateable firmware
- x8 or x16 user-selectable I/O operation
- RP# for reset and write protection
- PSOP and TSOP packages

Intel has integrated its SmartVoltage technology into the boot block family in order to increase the voltage flexibility of these components.

3.0 PINOUT COMPATIBLE DENSITY UPGRADES

In addition, Intel is providing density upgrades with pinout compatibility for the 2-Mbit, 4-Mbit, and 8-Mbit densities. The pinouts in Figures 2, 3, and 4 illustrate these compatible upgrade paths.

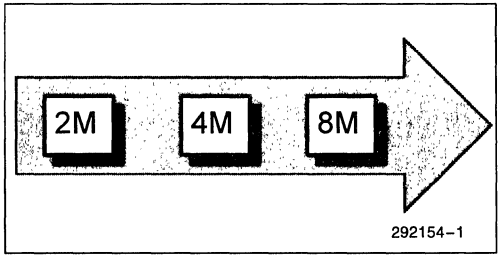


Figure 1. The SmartVoltage Technology Boot Block Line Features a Pinout-Compatible Upgrade Path

4.0 NEW SmartVoltage TECHNOLOGY FEATURES

SmartVoltage offers the following new features:

1. Voltage Flexibility
   - V_{CC} = 2.7–3.6V, 3.3V ±0.3V or 5V ±10% with enhanced circuits to optimize low-voltage performance when low power consumption is critical.
   - Program/erase operation with V_{PP} = 5V for convenient in-system writes without a DC–DC converter or V_{PP} = 12V when write/erase performance is a concern, such as during production.
2. Write Protection
   - WP# pin replaces a DU pin and is used in conjunction with the V_{PP} and RP# pins, as detailed in the table below, to control write protection of the boot block. (WP# pin not available on 8-Mbit 44-lead PSOP. In this package, treat as if the WP# pin is internally tied low, effectively eliminating the last row of the table below.)

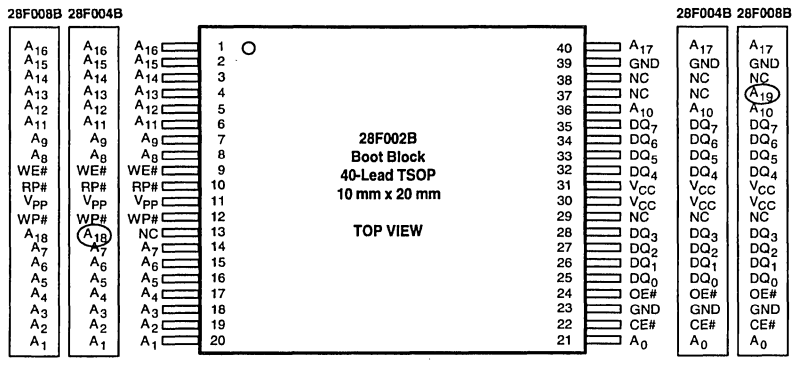
V_{PP}	RP#	WP#	Write Protection
V_{IL}	X	X	All Blocks Locked
≥ V_{PPLK}	V_{IL}	X	All Blocks Locked (Reset)
≥ V_{PPLK}	V_{HH}	X	All Blocks Unlocked
≥ V_{PPLK}	V_{IH}	V_{IL}	Boot Block Locked
≥ V_{PPLK}	V_{IH}	V_{IH}	All Blocks Unlocked

5.0 UPGRADING FROM 12V TO SVT

If you have designs using 12V V_{PP} boot block products, you must adhere to the following design steps to ensure you can upgrade to SVT:

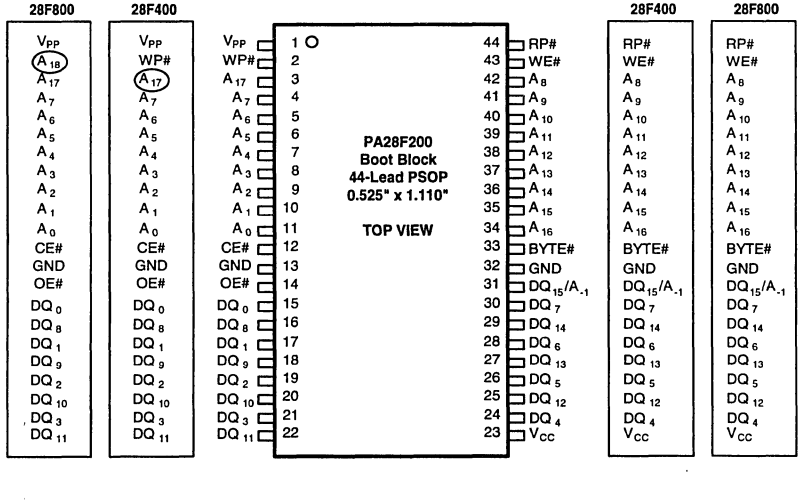
1. If using 5V program/erase, allow for connecting V_{PP} to 5V and disconnecting V_{PP} from 12V.
2. If adding a switch on V_{PP} for write protection, switch to GND instead of V_{CC}.
3. Connect WP# (DU on existing products) to V_{CC}, GND, or a control signal. This pin should not be left floating. The DU pin on BX/BL products can be driven to a logic-level in order to provide upgrade compatibility.

6.0 PACKAGE PINOUTS



NOTE:
The 28F008B pinout shown is for the 8-Mbit boot block, and not the 28F008SA FlashFile™ memory.

Figure 2. The 40-Lead TSOP Offers the Smallest Form Factor for Space-Constrained Applications



NOTE:
For the 8-Mbit device, pin 2 has been changed to A_{18} (WP# on 2/4-Mbit). Designers planning to upgrade to the 8-Mbit density from the 2/4-Mbit density in this package should design pin 2 to control WP# functionality at the 2/4-Mbit level and allow for pin 2 to control A_{18} after upgrading to the 8-Mbit density.

Figure 3. The 44-Lead PSOP Offers a Convenient Upgrade from JEDEC ROM Standards

5

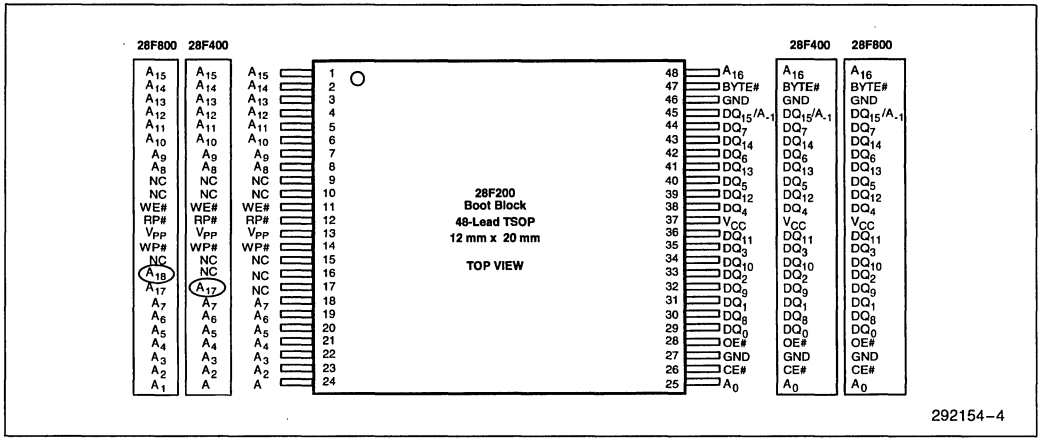


Figure 4. The 48-Lead TSOP Offers the Smallest Form Factor for x16 Operation

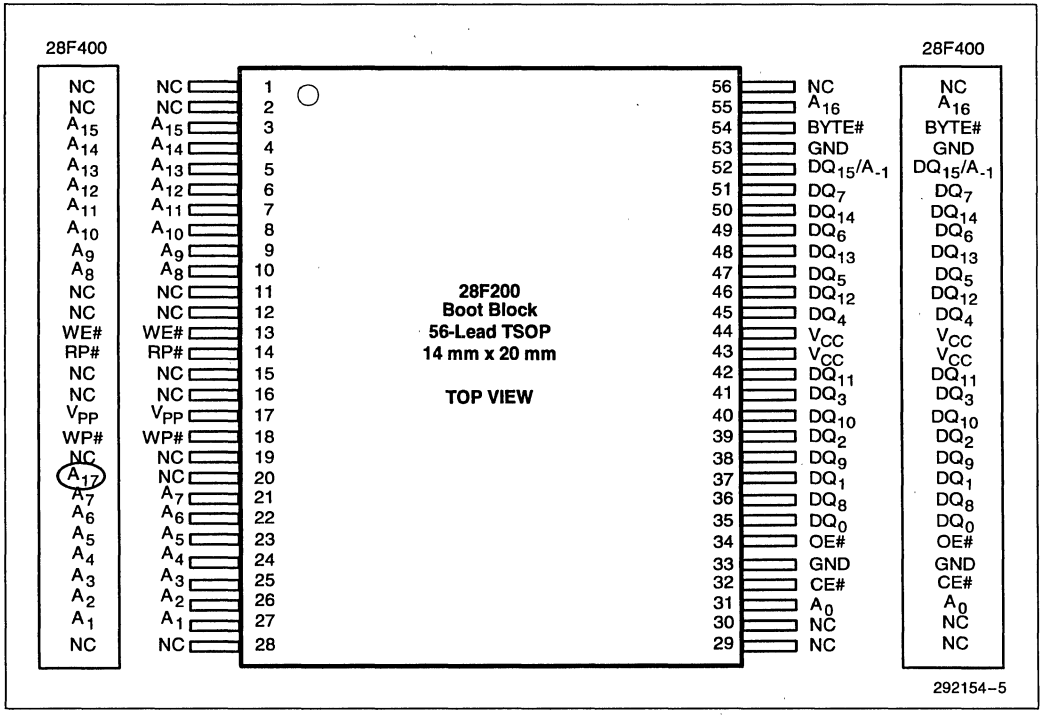
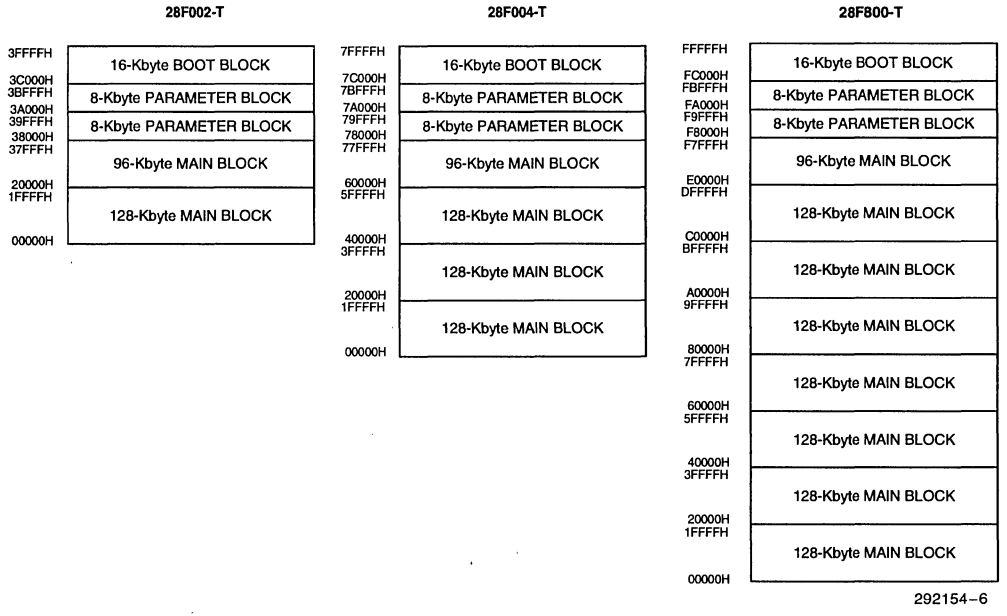


Figure 5. The 56-Lead TSOP Offers Compatibility between 2 and 4 Mbits

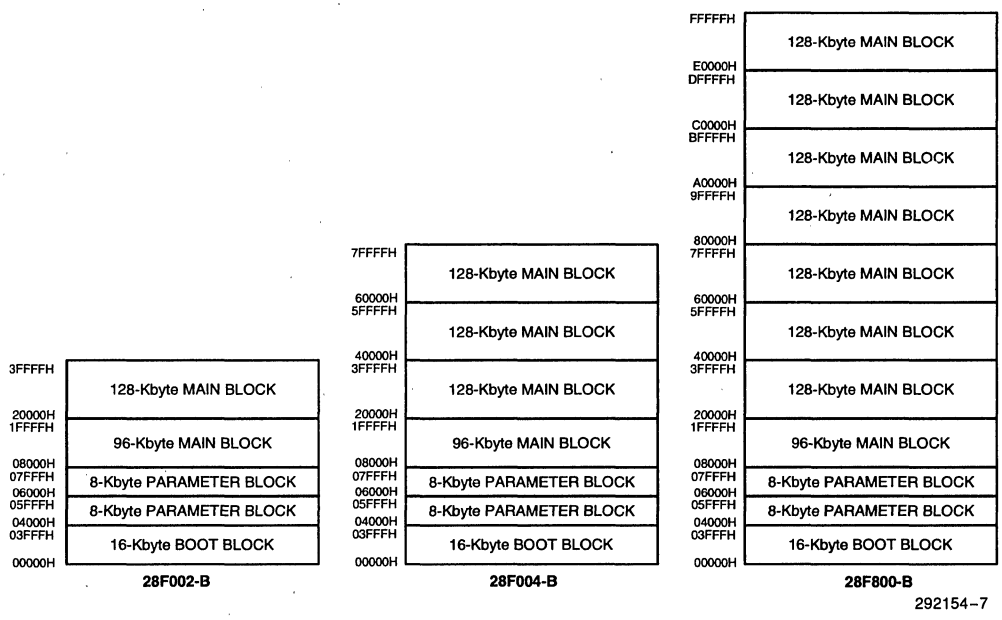
7.0 MEMORY MAPS



NOTE:
These memory maps apply to the 28F002/004/008-T components, or the 28F200/400/800-T components in byte-wide (x8) mode.

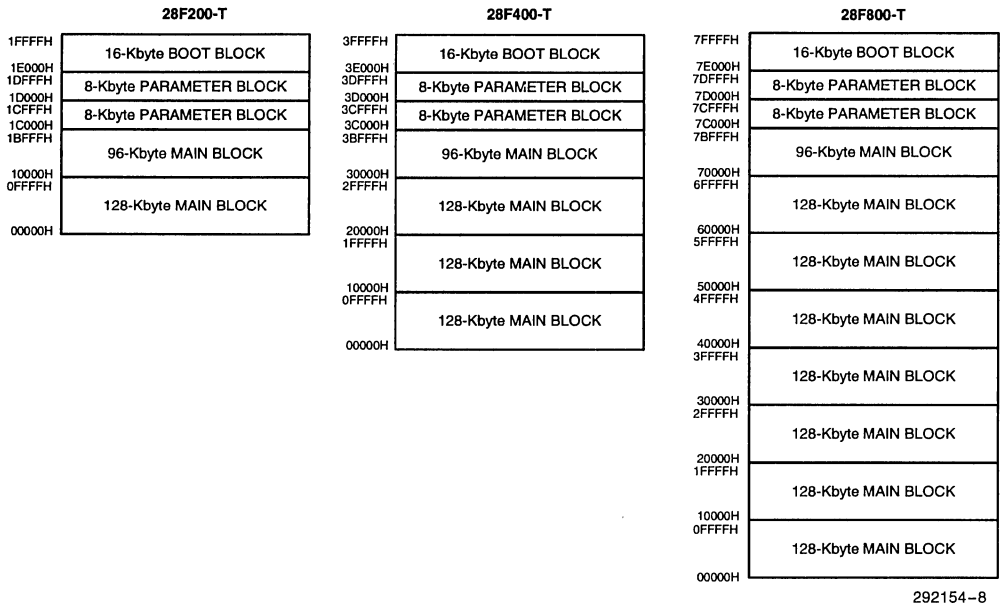
Figure 6. Byte-Wide x8-Mode Memory Maps (Top Boot)

5



NOTE:
These memory maps apply to the 28F002/004/008-B components, or the 28F200/400/800-B components in byte-wide (x8) mode.

Figure 7. Byte-Wide x8-Mode Memory Maps (Bottom Boot)

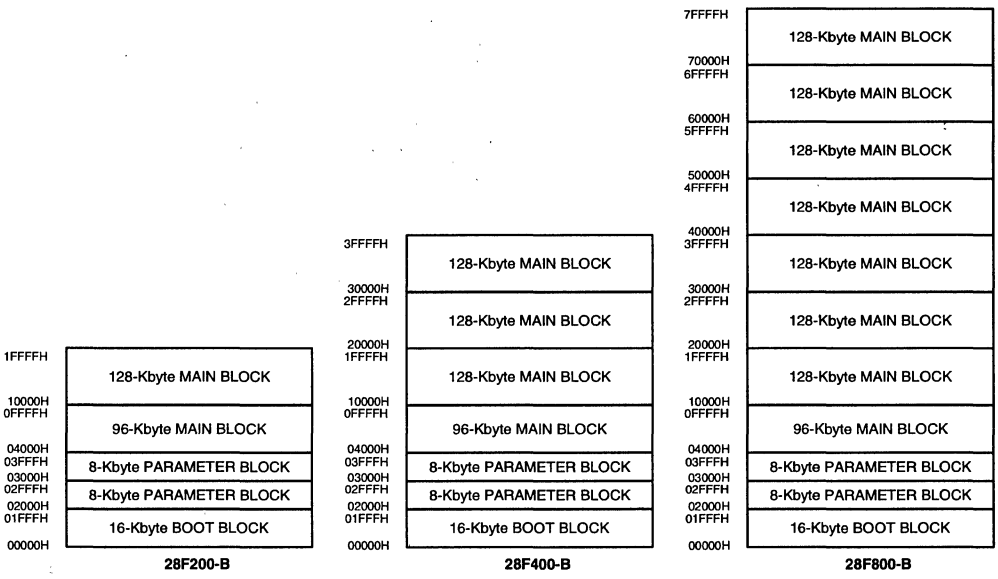


(Word addresses shown)

NOTE:
These memory maps apply to the denoted 28F200/400/800-T components in word-wide (x16) mode.

Figure 8. Word-Wide x16-Mode Memory Maps (Top Boot)

5



(Word addresses shown)

NOTE:
These memory maps apply to the denoted 28F200/400/800-B components in word-wide (x16) mode.

Figure 9. Word-Wide x16-Mode Memory Maps (Bottom Boot)

8.0 PRODUCT OFFERINGS

12V Program/Erase Boot Block Products

Product	Density x Org.	Pkg.	Speed (ns) (V_{CC} = 5V)	Speed (ns) (V_{CC} = 3.3V)	Extended Temperature
28F001BX	1 Mb, 128Kx8	P, N, E	70, 90, 120, 150		✓
28F200BX	2 Mb, 256Kx8/128Kx16	PA, E	60, 80, 120		✓
28F002BX	2 Mb, 256Kx8	E	60, 80, 120		✓
28F200BL	2 Mb, 256Kx8/128Kx16	PA, E	(1)	150	✓(2)
28F002BL	2 Mb, 256Kx8	E	(1)	150	✓(2)
28F400BX	4 Mb, 512Kx8/256Kx16	PA, E	60, 80, 120		✓
28F004BX	4 Mb, 512Kx8	E	60, 80, 120		✓
28F400BL	4 Mb, 512Kx8/256Kx16	PA, E	(1)	150	✓(2)
28F004BL	4 Mb, 512Kx8	E	(1)	150	✓(2)

NOTES:
1. The BL products also operate at 5V V_{CC} for programmer compatibility.
2. −20°C to +70°C operating range for Read; −0°C to +70°C for program and erase.

SmartVoltage Boot Block Products

Product	Density x Org.	Pkg.	Speed (ns) (V_{CC} = 5V)	Speed (ns) (V_{CC} = 3.3V)	Speed (ns) (V_{CC} = 2.7V)	Extended Temp.
28F200BV	2 Mb, 256Kx8/128Kx16	44, 56	60, 80, 120	110, 150, 180		✓
28F200CV	2 Mb, 256Kx8/128Kx16	48	60, 80, 120	110, 150, 180		✓
28F002BV	2 Mb, 256Kx8	40	60, 80, 120	110, 150, 180		✓
28F400BV	4 Mb, 512Kx8/256Kx16	44, 56	60, 80, 120	110, 150, 180		✓
28F400CV	4 Mb, 512Kx8/256Kx16	48	60, 80, 120	110, 150, 180		✓
28F004BV	4 Mb, 512Kx8	40	60, 80, 120	110, 150, 180		✓
28F800BV	8 Mb, 1024Kx8/512Kx16	44	70,120	120,150		✓
28F800CV	8 Mb, 1024Kx8/512Kx16	48	70,120	120,150		✓
28F008BV	8 Mb, 1024Kx8	40	70,120	120,150		✓
28F800CE	8 Mb, 1024Kx8/512Kx16	48	90		120	✓
28F008BE	8 Mb, 1024Kx8/512Kx16	40	90		120	✓

NOTES:
1. BV products also operate at V_{CC} = 5V for high-performance.
2. BE/CE products operate over the V_{CC} ranges 2.7–3.6V and 5V ±10%.

9.0 ORDERING INFORMATION

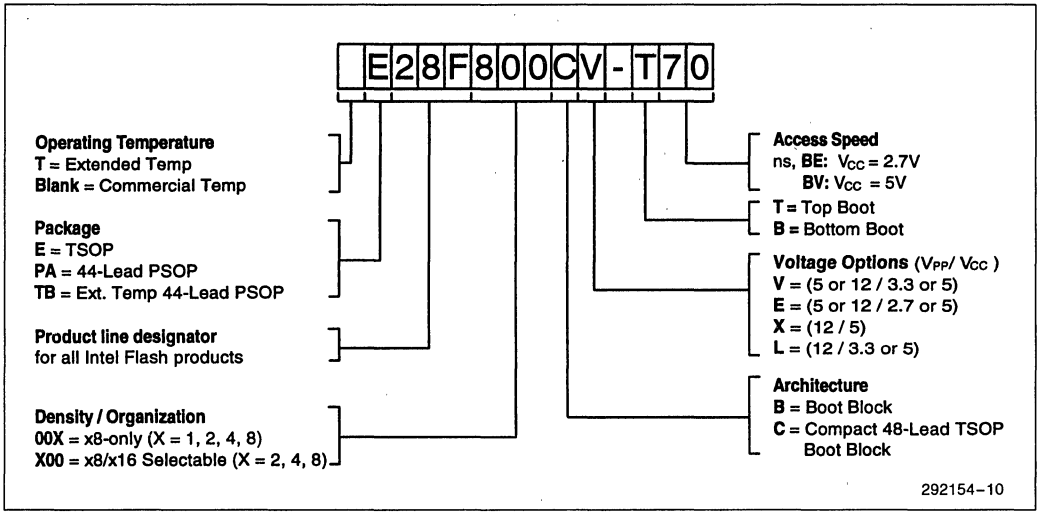


Figure 10. Decoding the Product Names

10.0 ADDITIONAL INFORMATION

10.1 Revision History

Number	Description
-001	Original Version
-002	Text updated. DU and WP# pin usage clarified. Note and "B" suffixes added to Figure 2. SmartVoltage Boot Block Products table: Speeds added to 28F800 and 28F008, 2.7V products added. Figure 10 updated.

intel®

TECHNICAL PAPER

Interfacing the 28F016XS to the Intel486™ Microprocessor Family

5

KEN MCKEE
TECHNICAL MARKETING ENGINEER

PHILIP BRACE
APPLICATIONS ENGINEER

December 1995

PRODUCT PREVIEW
Order Number: 297504-003

INTERFACING THE 28F016XS TO THE Intel486™ MICROPROCESSOR FAMILY

CONTENTS

CONTENTS

1.0 INTRODUCTION

This technical paper describes designs interfacing the high performance 28F016XS flash memory to the Intel486™ microprocessor. These designs are based on preliminary 28F016XS specifications. Please contact your Intel or distribution sales office for up-to-date information.

The 28F016XS is a 16-Mbit flash memory with a synchronous pipelined read interface. This optimized flash memory interface delivers equivalent or better read performance compared to DRAM. The 28F016XS combines ROM-like nonvolatility, DRAM-like read performance and in-system updateability into one memory technology. These inherent capabilities will improve performance and lower the over-all system cost.

The 28F016XS delivers optimal performance when interfacing to a burst processor, such as the Intel486 microprocessor. The Intel486 microprocessor sees widespread use in a variety of applications ranging from the PC to numerous embedded products, while providing code compatibility with thousands of commercially available software packages and the performance necessary for today's leading-edge systems. The Intel486 microprocessor's bus interface provides a burst transfer mechanism whereby four consecutive data items are fetched in one access sequence. The 28F016XS's synchronous pipelined read interface makes special use of the burst transfer mechanism to achieve extremely high read performance.

When interfacing the 28F016XS to a processor that executes an Intel or linear burst cycle, up to three simultaneous read accesses can be pipelined into the 28F016XS, sustaining a high read transfer rate. At 33 MHz, the 28F016XS-15 delivers zero wait-state performance after the initial pipeline fill. This enhanced read performance eliminates the costly expense of shadowing code from slow nonvolatile memory (ROM, hard disk drive, etc.) to fast DRAM for increased system performance. The 28F016XS enables direct code execution out of the flash memory array, eliminating unnecessary software and hardware overhead involved in shadowing code.

In an Intel486 microprocessor-based environment, BAPCo benchmarking analysis revealed a *13%* system performance improvement using the 28F016XS-15 over 70 ns DRAM.

In addition to the increased read performance, the 28F016XS offers an Intel486 microprocessor-based system a low power, nonvolatile memory that is electrically updateable via local processor control. The 28F016XS's low power consumption reduces system power dissipation and heat emission, and its updateability increases code flexibility and system reliability. Combined, the 28F016XS and the Intel486 microprocessor deliver a high performance, low power and cost-effective system solution.

The Intel486 microprocessor interface to the 28F016XS requires minimal logic while offering significant system enhancements. One programmable logic device (PLD), a 22V10-15, generates and monitors all 28F016XS and Intel486 microprocessor control signals. This technical paper explores the interface between the 28F016XS-15 and the Intel486 SX-33 microprocessor, describing the interface circuitry, explaining the read and write cycles and providing the interfacing PLD equations. It also provides detailed design suggestions for interfacing the 28F016XS to other Intel486 microprocessors.

2.0 OPTIMIZED 28F016XS/INTEL486™ SX MICROPROCESSOR INTERFACE

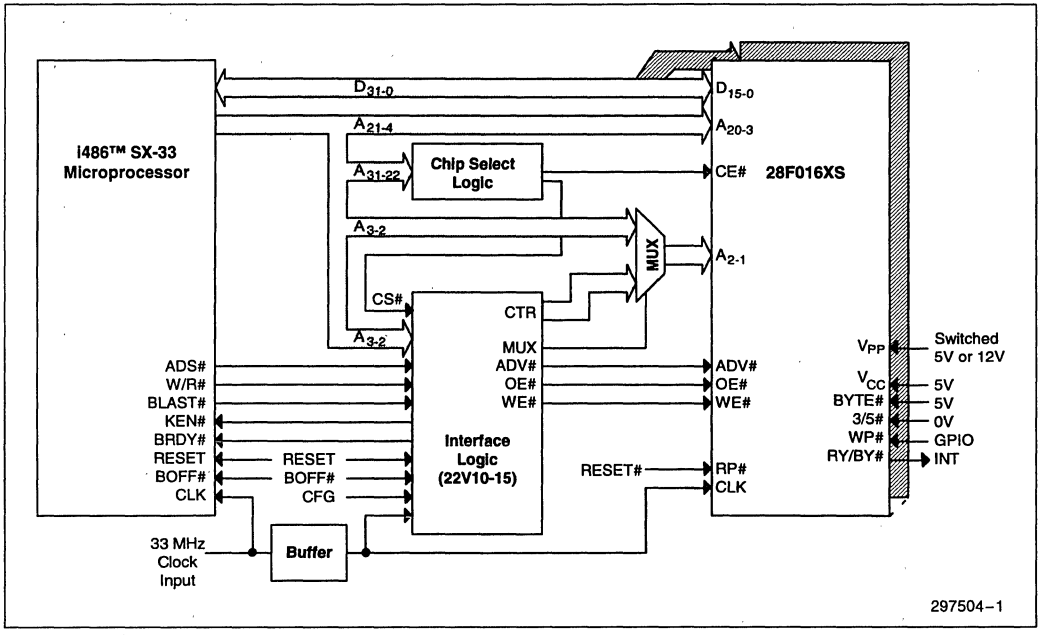


Figure 1. Optimized 28F016XS Interface to the Intel486 Microprocessor with Wait-State Profile of 2-0-0-0 up to 33 MHz

The 28F016XS-15 interface to the Intel486 SX-33 microprocessor, illustrated in Figure 1, delivers 2-0-0-0 wait-state read performance. Consult your Intel or distribution sales office for schematic and PLD files for this design.

See Section 3.0 for an alternative design.

2.1 Circuitry Description

This section will describe the circuitry involved in this design.

Memory Configuration

This design uses two 28F016XS-15s, each configured in x16 mode and arranged in parallel to match the Intel486 SX microprocessor's 32-bit data bus. This memory configuration provides 4 Mbytes of flash memory for system usage. Signals A_{21-4} from the Intel486 SX microprocessor and CTR_{1-0} from the PLD select locations within the 28F016XS memory space. The two-bit counter implemented in the PLD supplies burst addresses to the 28F016XSs.

Reset

The Intel486 SX microprocessor requires an active high reset signal, while the 28F016XSs use an active low RESET#. Figure 2 illustrates a suggested logic configuration for generating both an active high and low reset signal. The active high RESET controls the Intel486 SX microprocessor and PLD reset inputs, while the active low RESET# drives the 28F016XS RP# input.

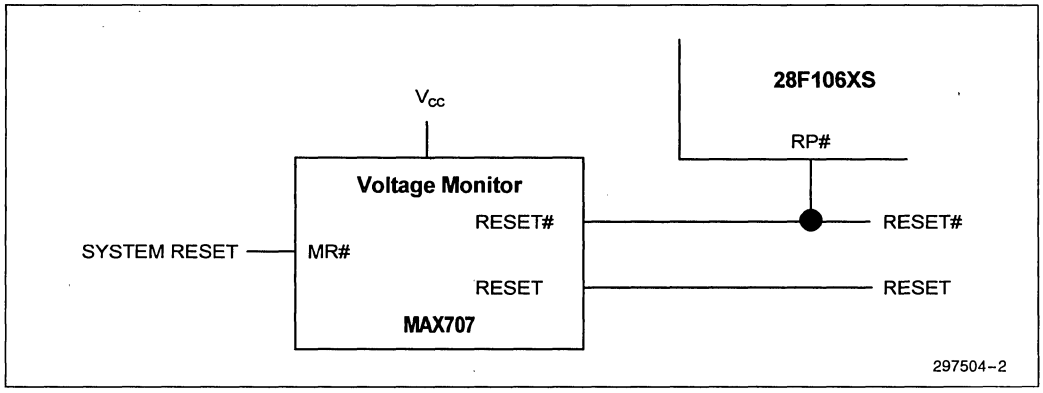


Figure 2. RESET Generation Method

Chip Select Logic

Chip select decode logic may use A_{31-22} to generate active low chip select signals, CE_X#, for the 28F016XS memory space and other system peripherals. The chip select addressing the 28F016XS memory space drives CE_0# on each 28F016XS-15 and a control input to the PLD. The 28F016XS-15s' CE_1# inputs are grounded. For many systems, using the upper address bits in a linear selection scheme may provide a sufficient number of chip select signals, thus eliminating chip select decode logic. (See Figure 3 for an example of using linear selection for chip selects.) When using a linear chip select scheme however, software must avoid using addresses that may select more than one device, which could result in bus contention. For example, addresses 01000000H through 010FFFFFH drive both A_{22} and A_{23} to a logic "0," which inadvertently selects two peripheral devices.

CLK Option

A 33 MHz CLK drives the Intel486 SX microprocessor. The buffer in Figure 1 delays this processor CLK input and drives the PLD and 28F016XS-15s. The buffer introduces an intentional system clock skew. This skew provides additional time for the processor to meet the 28F016XSs' address setup time.

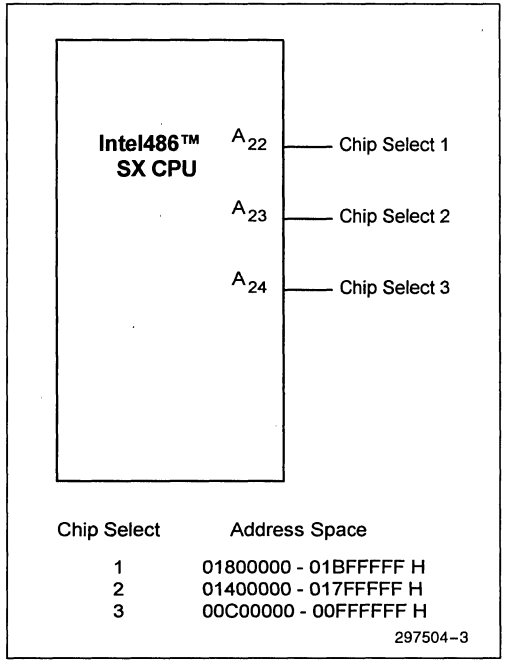


Figure 3. Example of Using Linear Chip Selection with Active Low Chip Select Signals

5

Multiplexer (MUX)

To achieve this type of wait-state profile, the Intel486 SX microprocessor directly loads the 28F016XSs with the initial address. The interface logic enables the MUX to permit the processor's lower address lines A_{3-2} access to the lower flash memory address lines. Next, the interface logic switches the data flow path through the MUX in anticipation of a burst transaction. The two-bit counter integrated into the control logic takes over driving the 28F016XSs' A_{2-1}. The counter supplies the flash memory with consecutive burst addresses for the remaining duration of the transaction.

Interface Control Signals

The interfacing state machine monitors the Intel486 SX microprocessor's external bus signals to identify the type, start and end of a bus cycle. At the beginning of a burst cycle, the interface logic loads a two-bit counter generates the flash memory control signals. The state machine also generates KEN# and BRDY# signals, informing the Intel486 SX microprocessor of the nature of the bus cycle.

Configuration Signal

A general purpose input/output (GPIO) generates the configuration signal input (CFG) to the state machine. CFG must reset to logic "0" on power-up and system reset to ensure coherency between the state machine and 28F016XS-15s. After optimizing the 28F016XSs' SFI Configuration, CFG must switch to logic "1" in order to take advantage of the optimized flash memory state. See Section 2.3 for more information regarding the configuration signal.

Additional 28F016XS Control Signals

The BYTE# input to the 28F016XS-15s is tied to 5.0V to configure the 28F016XS-15s for x16 mode, and 3/5# and A_0 are tied to GND (A_0 is only used for byte addressing). A GPIO controls the write protect input, WP#, to the 28F016XS-15s. The 28F016XS is compatible with either a 5.0V or a 12.0V V_{PP} voltage and is completely protected from data alteration when V_{PP} is switched to GND. With $V_{PP} \leq V_{PPLK}$, the 28F016XS will not successfully complete data write and erase operations, resulting in absolute flash memory data protection. Figure 1 also illustrates the 28F016XS-15's RY/BY# output connecting directly to a system interrupt, which enables background write/erase operations. RY/BY#, WP#, and V_{PP} implementations are application dependent. Consult the Additional Information section of this technical paper for documentation covering these topics in more detail.

2.2 Interfacing Signal Definitions

The interface logic that controls the 28F016XS-15 interface to the Intel486 SX-33 microprocessor monitors and regulates specific system signals. The next two sections describe these signals in detail.

2.2.1 28F016XS Signal Descriptions

This section describes the 28F016XS signals that are pertinent to this design.

ADV#—Address Valid (Input)

This active low signal informs the 28F016XS that a valid address is present on its address pins. ADV#, in conjunction with a rising CLK edge, initiates a read access to the 28F016XS. This signal is ignored during write operations.

CLK—Clock (Input)

CLK provides the fundamental timing and internal operating read frequency for the 28F016XS. CLK initiates read accesses (in conjunction with ADV#), times out the SFI Configuration, and synchronizes device outputs. CLK can be slowed or stopped with no loss of data synchronization. This signal, like ADV#, is ignored during write operations.

OE#—Output Enable (Input)

This active low signal activates the 28F016XS's output buffers when OE# equals "0." The outputs tri-state when OE# is driven to "1."

WE#—Write Enable (Input)

This active low signal controls access to the Control User Interface (CUI). Addresses (command or array) and data are latched on the rising edge of WE# during write cycles.

2.2.2 INTEL486 SX Microprocessor Signal Descriptions

This section describes the Intel486 SX microprocessor signals that are relevant to this interface. This interface assumes processor inputs are driven by only one controlling device (the PLD). If more than one device drives a processor input, the PLD output should be configured as an open drain to avoid signal contention. Many PLDs, FPGAs and ASICs provide output configuration capability.

ADS#—Address Status (Output)

This active low output signal from the Intel486 SX microprocessor indicates the presence of valid bus cycle and address signals on the bus. ADS# is driven in the same clock as the address signals. Typically, external circuitry uses ADS# to indicate the beginning of a bus cycle.

KEN#—Cache Enable (Input)

This active low input to the Intel486 SX microprocessor determines whether data being returned in the current bus cycle will be cached. In order for the current data to be cached, KEN# must be returned active in the clock prior to the first RDY# or BRDY# of the cycle and must also be returned active in the last clock of the data transfer.

BRDY#—Burst Ready (Input)

This active low input to the microprocessor performs the same function during a burst cycle as RDY# performs during a non-burst cycle. During a burst cycle, BRDY# is sampled on the rising edge of every clock. Upon sampling BRDY# active, the data on the data bus will be latched into the microprocessor (for burst reads). ADS# will be negated during the second transfer of the burst cycle; however, the lower address lines and byte enables may change to indicate the next data item requested by the processor.

BLAST#—Burst Last (Output)

This active low output from the microprocessor signals the final transfer in a burst cycle. The next time BRDY# is returned, it will be treated the same as RDY# and thus terminate any multiple cycle transfers.

2.3 System Interface Requirement

The system logic controlling the 28F016XS-15 interface to the Intel486 SX microprocessor incorporates an initial and an optimized read configuration, which correlates to specific SFI Configuration values. The interface read configuration is dependent upon the value of CFG (PLD input). CFG informs the interface of the SFI Configuration status. Note, the SFI Configuration status does not affect write operations.

Initial Read Configuration

Upon power-up/reset, the 28F016XS-15 defaults to a SFI Configuration value of 4, and the interface logic supports burst read accesses to the flash memory space. The interface returns BRDY# to inform the processor that the interface supports burst read transaction. A general purpose input/output (GPIO) informs the system interface of the status of the SFI Configuration.

The GPIO entitled CFG is set to logic "0" on power-up/reset. With CFG driven low, the state machine correctly matches the 28F016XS-15s' default configuration.

Optimized Read Configuration

At 33 MHz, the 28F016XS-15 operates at highest performance with a SFI Configuration value set to 2. To reconfigure the 28F016XS-15, program control should jump to an area of RAM to execute the configuration sequence. After reconfiguring the 28F016XS-15, the GPIO value must change to logic "1," in order to take advantage of the 28F016XS-15's optimized configuration. A pseudocode flow for this configuration sequence is shown below.

Execute Device Configuration command sequence
Activate CFG signal
End

In the optimized read configuration, the system logic supports burst cycles by generating BRDY#, which informs the microprocessor that the memory subsystem is capable of handling a burst transfer. The 28F016XS-15 memory array, after the initial pipeline fill delay from the first access, transfers data to the processor at a rate of 133 Mbytes/sec.

2.4 Read Control for Burst Transactions

The interface logic controlling the handshaking between the 28F016XS and processor performs one of two different read cycles, depending upon the value of CFG.

Read Abort Condition

A read cycle will abort only when an external system bus master asserts BOFF#, which forces the processor to give immediate bus control to the requester. When this situation occurs, the Intel486 SX microprocessor floats the address bus, which will cause the address decode logic to de-select the 28F016XS memory space. Detecting BOFF# active, the interfacing logic will transition to an idle state and wait for the processor to re-initiate the interrupted bus cycle after the bus master has relinquished the bus to the processor. OE# is immediately driven high, deactivating the 28F016XS-15's output buffers, upon detecting BOFF# driven active. This BOFF# condition can occur in both the initial and optimized configurations described in the paragraphs that follow.

Initial Configuration

Refer to Figures 4 and 5 for the following read cycle discussion.

With CFG set to logic "0," the interfacing read state machine executes cacheable burst read cycles. This configuration occurs upon power-up and reset.

Initially, the interface logic drives ADV# and MUX active while waiting for the Intel486 SX processor to initiate a bus cycle targeting the 28F016XS. With the MUX active, the processor's A_{3-2} drive the lower flash memory address lines in anticipation of a flash memory access. During this anticipation state, CE# is active to prevent a t_{ELCH} violation on the first access initiated by the processor. The delayed CLK prevents a possible timing violation. It provides the processor with sufficient time to meet the 28F016XSs' t_{AVCH} specification when initiating the first access.

When the microprocessor initiates a read access to the 28F016XS memory space, it provides an address and drives W/R# and ADS# low. Monitoring these signals, the state machine transitions into read control.

If ADS# = 0 and W/R# = 0 then READ CONTROL

At this point, CFG and CS# are examined to determine the configuration status of the 28F016XS-15s, and whether or not the current address targets the 28F016XS memory space. If CS# = "1," the state machine returns to an idle state waiting for a new access. Otherwise, the state machine will continue the read access, regulating ADV#, BRDY# and OE#.

At N = 1 (Figure 5), the interfacing read state machine loads and increments the two-bit counter. The counter is incremented because the processor supplies the flash memory with the initial address. The counter then provides the flash memory with the subsequent burst addresses throughout the remaining duration of the bus transaction.

With ADV# at logic "0," the interface initiates a read access to the 28F016XS-15s at N = 1. Next, ADV# immediately switches to a logic "1" at N = 1 and then toggles active on every other clock edge until N = 8. After this time, ADV# will remain inactive.

In the default SFI Configuration (SFI Configuration = 4), the first data will be accessible to the processor at N = 6. The remaining data will be available for the processor to retrieve at N = 8, 10 and 12. The 28F016XS-15's output buffers are enabled at N = 3 and BRDY# is driven low at N = 5. The processor will sample BRDY# active and latch the information residing on the data bus at N = 6. If the processor drives BLAST# inactive, indicating a burst transaction is in process, the interface logic will drive BRDY# active on every other clock edge until BLAST# is sampled active by the interface logic. The state machine will transition to an idle state where it deactivates OE# and waits for the processor to initiate a new bus cycle.

Initial Configuration Timing Consideration

In this initial read configuration, there are important timing considerations that need examination.

First, the buffer delay can cause possible timing violations if not chosen correctly. The purpose of the buffer is to provide time for the processor to load the 28F016XSs with the initial address during read transactions. Therefore, the buffer must have a minimum delay which satisfies the flash memory's t_{AVCH}.

$$t_{AVCH} + t_6 - 1/33\ MHz = 1\ ns$$

The buffer can also affect the processor's data setup time. Hence, the buffer must have a maximum delay of no greater than:

$$1/33\ \text{MHz} - t_{CHQV} - t_{22} = 5\ \text{ns}$$

Another important timing parameter is the Intel486 SX microprocessor's data hold time. Since the 28F016XS specifies a 0 ns guaranteed data hold time from CE# or OE# high, these two signals must be driven active until the processor's hold time is satisfied. CE# hold delay will not be concern because CE# is held active during the state machine's idle state. OE# has only 0.5 ns of margin to the processor's specification for the buffer used in this design. OE# hold time equals:

$$t_{PZX(min)} + t_{PHL(min)} = 3.5\ \text{ns}$$

Consult the appropriate datasheets for full timing information.

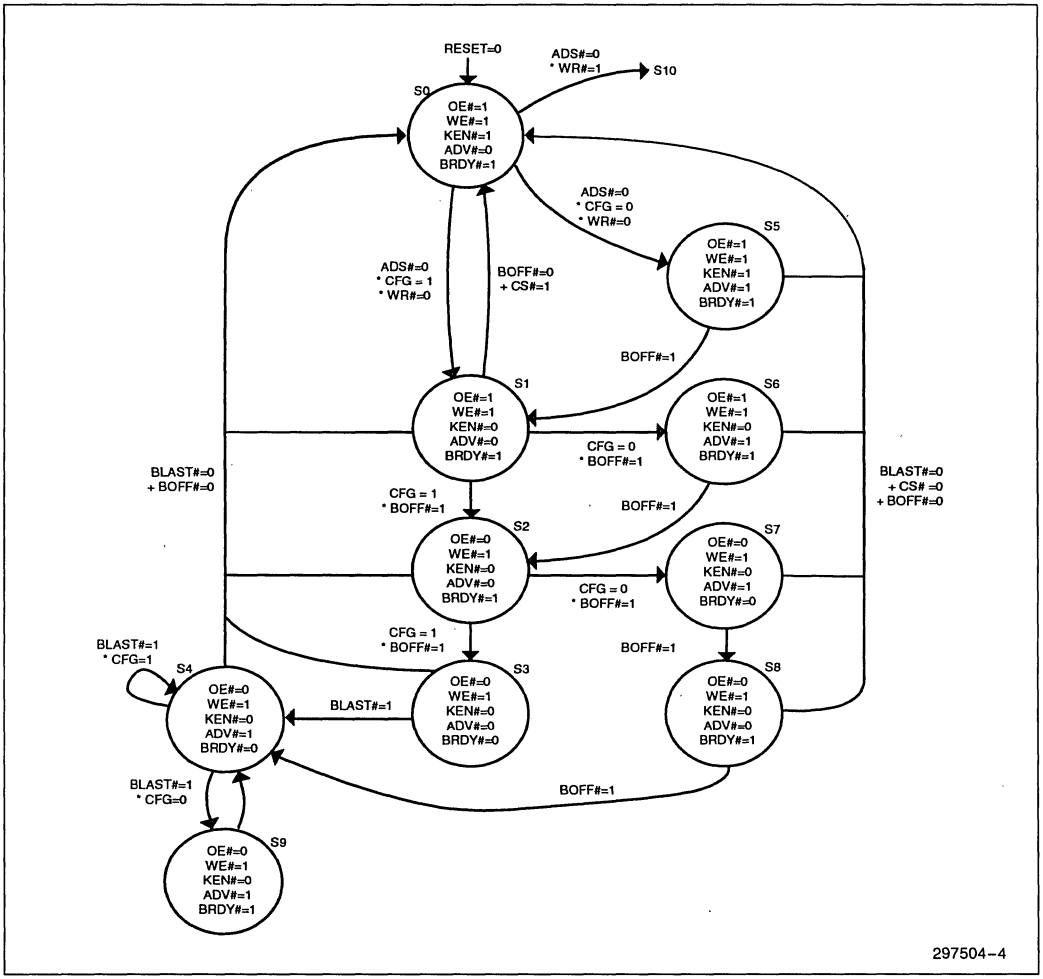


Figure 4. Optimized Read State Diagram for Burst Read Control (Interface Shown in Figure 1)

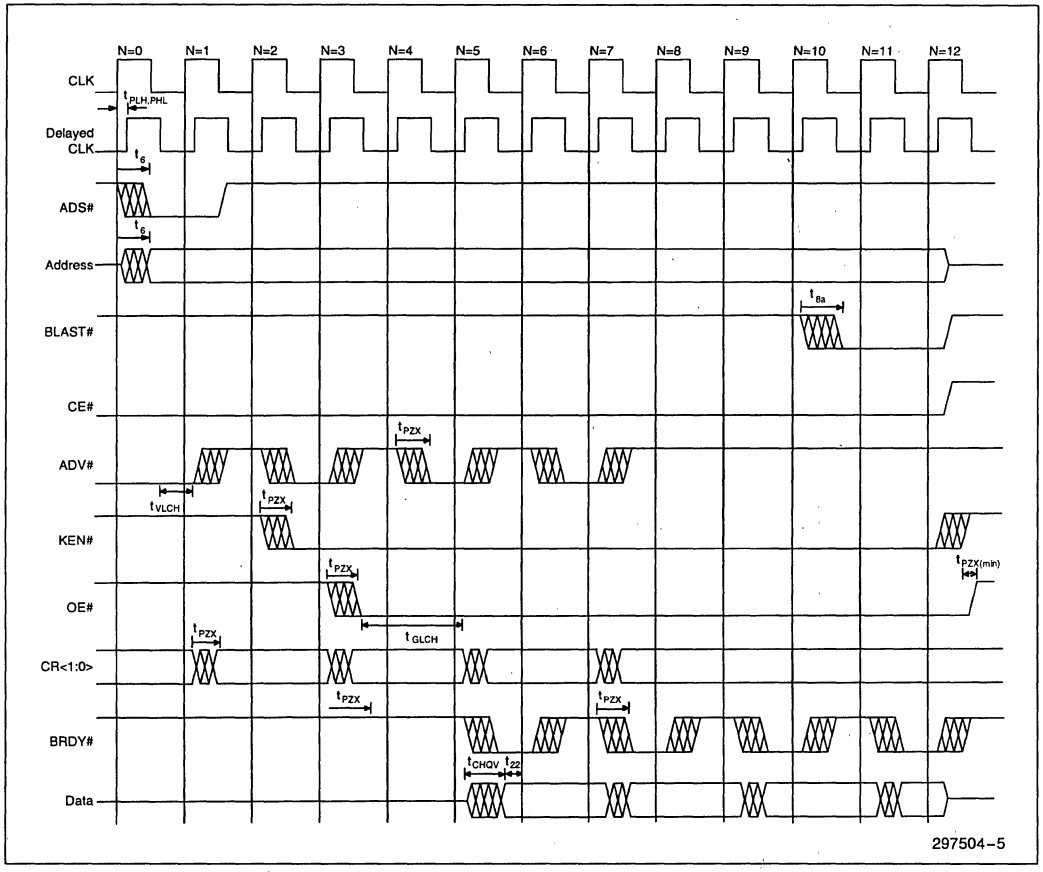


Figure 5. Example Initial Burst Read Cycle Showing Key Timing Specifications Requiring Consideration

Table 1. Example Optimized Read Cycle Specifications at 5V V_{CC}

Symbol	Description	Min	Max	Unit
$t_{PHL,PLH}$	Buffer Delay	1.5	5	ns
t_6	ADS# Delay (Intel486 SX-33 Microprocessor)	3	16	ns
t_{8a}	BLAST# Delay (Intel486 SX-33 Microprocessor)	3	20	ns
t_{22}	D_{31-0} Setup Time (Intel486 SX-33 Microprocessor)	5		ns
t_{ELCH}	CE_X# Setup Time to CLK (28F016XS-15)	25		ns
t_{VLCH}	ADV# Setup Time to CLK (28F016XS-15)	15		ns
t_{GLCH}	OE# Setup Time to CLK (28F016XS-15)	15		ns
t_{CHQV}	CLK to Data Delay (28F016XS-15)		20	ns
t_{PZX}	CLK Output Delay (22V10-15)	2	8	ns

NOTE:
Consult appropriate datasheets for up-to-date specifications.

Optimized Configuration

Refer to Figures 4 and 6 for the following discussion.

With the 28F016XS-15s in the optimized configuration (SFI Configuration = 2 at 33 MHz), and CFG set to a logic "1" value, the system interface executes cacheable burst cycles.

Like the initial configuration, the connecting logic initially drives ADV# and MUX active while waiting for the Intel486 SX processor to initiate a bus cycle targeting the 28F016XS. With the MUX active, the processor's A_{3-2} drive the lower flash memory address lines in anticipation of a flash memory access. During this anticipation state, CE# is active to prevent a t_{GLCH} violation on the first access initiated by the processor. The delayed CLK also inhibits also possible timing violations from occurring. It provides the processor with sufficient time to meet the 28F016XSs' t_{AVCH} specification when initiating the first access.

When the processor drives ADS# low, it notifies the interface logic that a valid address is on the address bus. Monitoring the external bus of the Intel486 SX microprocessor, the state machine then transitions into read control.

If ADS# = 0 and W/R# = 0 then READ CONTROL

In optimized read control, the state machine controls OE#, KEN# and BRDY#. If CS# = "0," the state machine at N = 1 loads and increments the two-bit counter, switches the data flow path through the MUX and holds ADV# active for the next three consecutive clock periods. While ADV# is driven low, the counter increments through the Intel burst order (Table 2), supplying the 28F016XS-15s with a new address at N = 2, 3 and 4. If CS# = "1," the state machine returns to an idle state waiting for a new memory access.

Table 2. Intel Burst Order (A_{3-2})

First Address	Second Address	Third Address	Fourth Address
0	4	8	C
4	0	C	8
8	C	0	4
C	8	4	0

At N = 2, the state machine drives KEN# active and holds it active until the end of the burst cycle, thereby executing a cache line fill.

The state machine activates the 28F016XS-15 output buffers (OE# driven to a logic "0" value) at N = 2 and holds them active throughout the burst read cycle.

With the SFI Configuration value set to 2, data will be available at N = 4, 5, 6 and 7. Driving BRDY# low at N = 3, the Intel486 SX microprocessor will sample BRDY# active at N = 4, which informs the processor of valid information on data pins D_{31-0} and that the 28F016XS memory space supports a burst read transfer. BRDY# is held low until the end of the burst cycle while the processor retrieves data on every rising clock edge. BRDY# is driven high upon sampling BLAST# low, marking the end of the burst cycle. Then the state machine will transition to and idle state where it deactivates OE# and waits for the processor to initiate a new bus cycle.

Optimized Configuration Timing Considerations

In the optimized configuration, the same timing consideration regarding the buffer propagation delay and OE# hold time require attention. For information regarding these concerns, see Section 2.4 Initial Configuration Timing Considerations.

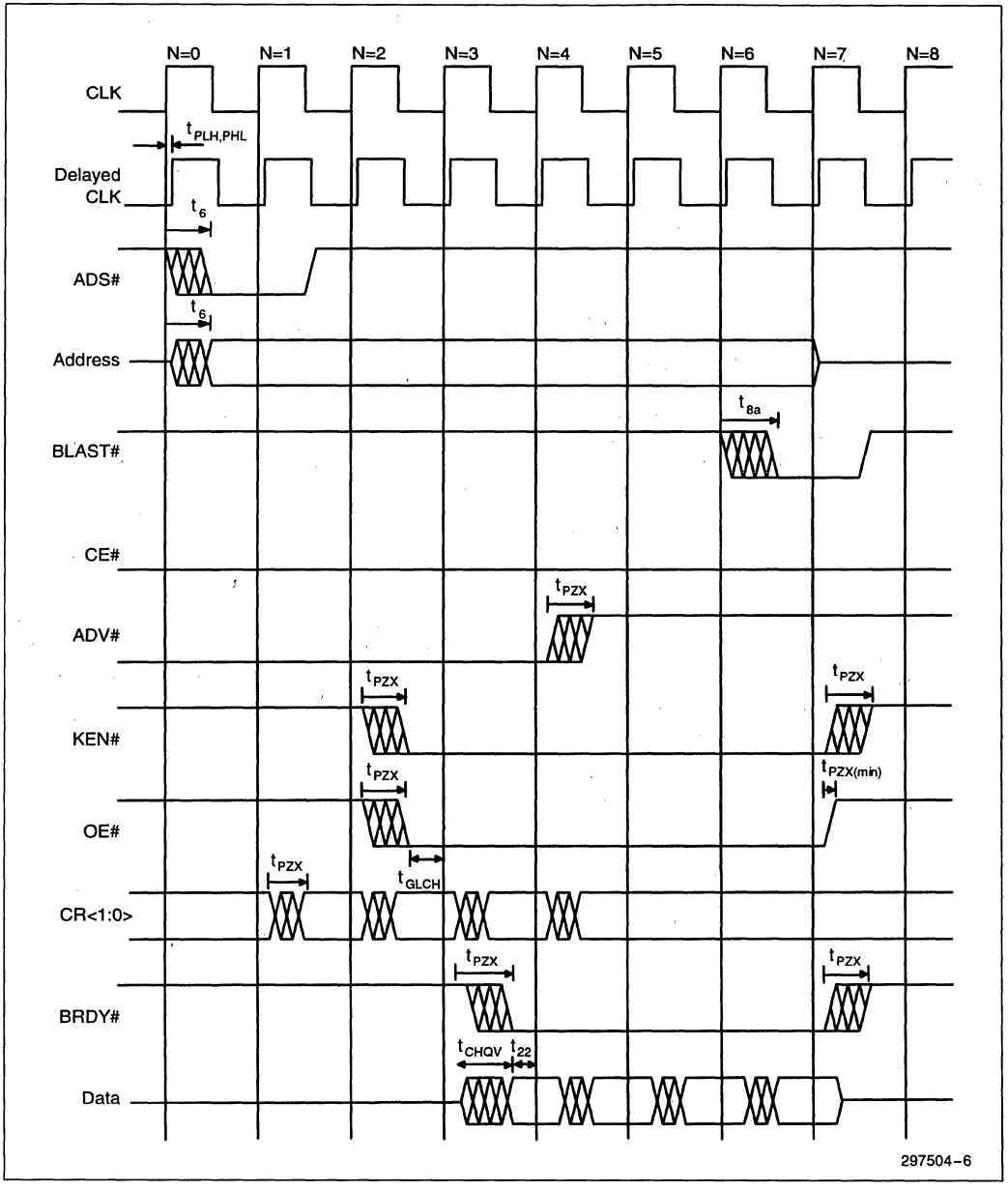


Figure 6. Example Burst Read Timing Waveform Illustrating Key Timing Specifications Requiring Consideration

Table 3. Example Optimized Read Cycle Specifications at 5V V_{CC}

Symbol	Description	Min	Max	Unit
$t_{PHL,PLH}$	Buffer Delay	1.5	5	ns
t_6	ADS# Delay (Intel486 SX-33 Microprocessor)	3	16	ns
t_{8a}	BLAST# Delay (Intel486 SX-33 Microprocessor)	3	20	ns
t_{22}	D_{31-0} Setup Time (Intel486 SX-33 Microprocessor)	5		ns
t_{ELCH}	CE_X# Setup Time to CLK (28F016XS-15)	25		ns
t_{VLCH}	ADV# Setup Time to CLK (28F016XS-15)	15		ns
t_{GLCH}	OE# Setup Time to CLK (28F016XS-15)	15		ns
t_{CHQV}	CLK to Data Delay (28F016XS-15)		20	ns
t_{PZX}	CLK Output Delay (22V10-15)	2	8	ns

NOTE:
Consult appropriate datasheets for up-to-date specifications.

2.5 Write Cycle Control

Refer to Figures 7 and 8 for the following write cycle discussion.

Write Abort Condition

A write cycle will abort only when an external system bus master asserts BOFF#, which forces the processor to give immediate bus control to the requester. When this situation occurs, the Intel486 SX microprocessor will float the address bus, causing the address decode logic to de-select the 28F016XS memory space. The interfacing logic, monitoring BOFF#, will transition to an idle state where it will wait for the processor to re-initiate the interrupted bus cycle after the bus master has relinquished the bus to the processor. WE# is immediately deactivated upon sensing BOFF# low.

Write Cycle Description

The 28F016XS-15 executes asynchronous write cycles like traditional flash memory components such as the 28F016SA/SV. The SFI Configuration does not influence write operations; therefore, the interfacing state machine does not examine CFG once detecting a write cycle.

During the first clock period, the Intel486 SX microprocessor drives ADS# low and W/R# high. The state machine, upon detecting a write cycle, immediately switches the data flow path through the MUX. The processor does not drive the flash memory's lower address lines during write cycles. The counter loads and supplies the address to the 28F016XSs' lower two address lines, A_{2-1}. The state machine then transitions to write control at N = 1. The counter only supplies the 28F016XS-15 with one address throughout the entire write operation. A write transaction must compete fully before issuing a second write operation.

If ADS# = 0 and W/R# = 1 then WRITE CONTROL

In write control, the state machine performs WE#-Controlled Command Write operations to the 28F016XS-15s. Data is written to the 28F016XS memory space via processor control. The interface only supports double word writes.

For the next two clock periods the state machine holds WE# low to satisfy the 28F016XS-15s' WE# active requirement. At N = 4, WE# transitions to a logic "1," which latches the address and data into the 28F016XS-15s.

5

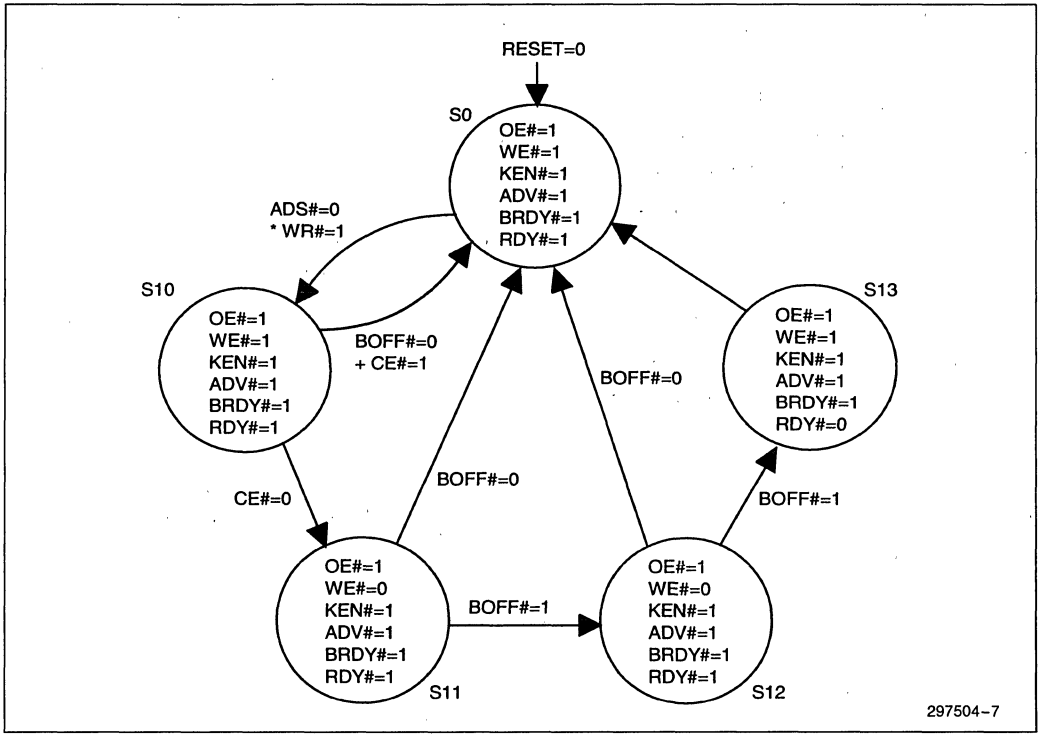


Figure 7. Non-Burst Write State Diagram Controlling the Interface Shown in Figure 1

BRDY# is not returned to the processor at N = 4 because the Intel486 SX microprocessor will only hold an address 3 ns after sampling BRDY# low. Instead, the interface activates BRDY# after N = 4, causing the processor to hold the address valid for an additional clock cycle, which satisfies the 28F016XS-15's address hold specification (t_{WHAX}). The state machine then returns to an inactive state at N = 5, waiting for a new memory access.

Write Timing Consideration

When performing a write operation, CS# is a critical system timing parameter, which must satisfy the interface logic's required setup time. The 22V10-15 requires a 9 ns setup time to CLK. Therefore, the system decode logic must generate a valid CS# to the interface within:

$$2 \times 1/33\ \text{MHz} - t_6 - t_{SU} = 35\ \text{ns}$$

Consult the appropriate datasheets for full timing information.

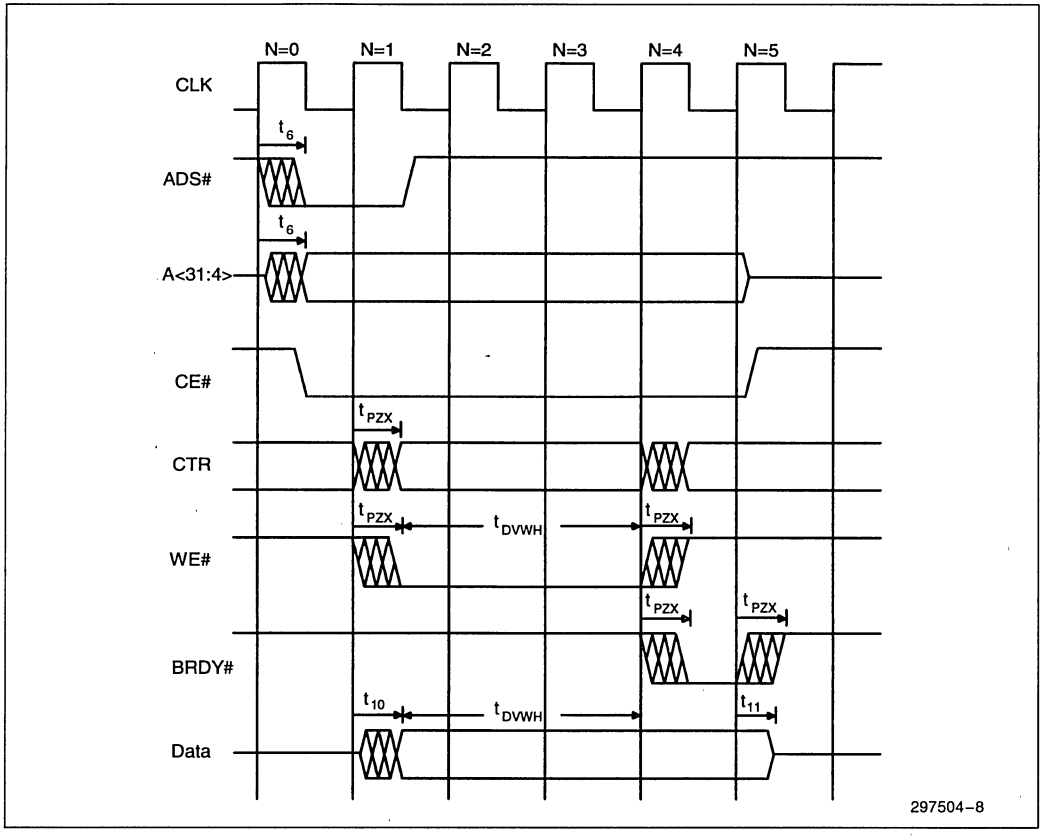


Figure 8. Example Write Cycle Showing Key Timing Specifications Requiring Consideration

Table 4. Example Write Cycle Timing Specifications at 5V V_{CC}

Symbol	Description	Min	Max	Unit
t_6	ADS# Delay(Intel486 SX-33 Microprocessor)	3	16	ns
t_{10}	Data Write Valid Delay (Intel486 SX-33 Microprocessor)	3	18	ns
t_{11}	Data Write Float Delay (Intel486 SX-33 Microprocessor)		20	ns
t_{WLWH}	WE# Pulse Width (28F016XS-15)		50	ns
t_{DVWH}	Data Setup to WE# Going High (28F016XS-15)		50	ns
t_{PZX}	CLK Output Delay (22V10-15)	2	8	ns

NOTE:
Consult appropriate datasheets for up-to-date specifications.

3.0 STANDARD 28F016XS/INTEL486 SX MICROPROCESSOR INTERFACE

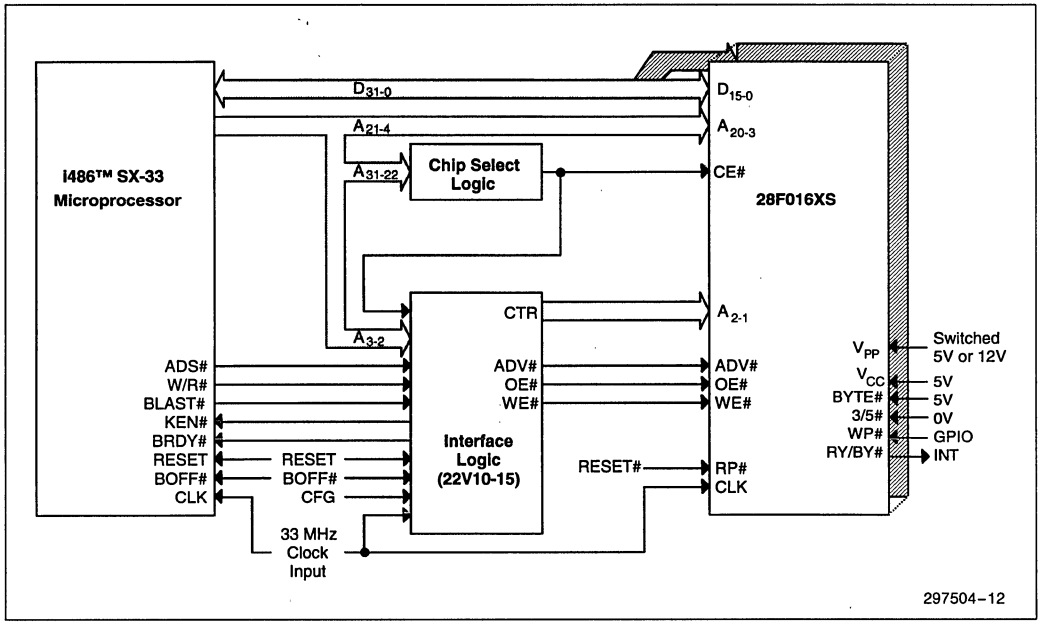


Figure 9. Minimal Glue Logic in Interfacing the 28F016XS-15 to the Intel486 SX-33 Microprocessor with a Wait-State Profile of 3-0-0-0 Up to 33 MHz

The 28F016XS-15 interface to the Intel486 SX-33 microprocessor illustrated in Figure 9 delivers 3-0-0-0 wait-state read performance. The design requires only one 22V10 to handle all interfacing requirements. Consult your Intel or distribution sales office for schematic and PLD equations for the interface documented in this section.

See Section 2.0 for an alternative design.

3.1 Interface Circuitry Description

This interface is extremely similar to the optimized design described in Section 2. The circuitry elements involved in the design are exactly the same except for the elimination of the buffer and multiplexer. For specific circuitry information about the individual aspects of this design, refer to Section 2.

CLK Option

Unlike the optimized 28F016XS/Intel486 SX microprocessor interface, a buffer is not implemented in this design. The processor's 33 MHz CLK input drives both the PLD and flash memory. To reduce system clock skew, position the PLD and 28F016XSs within close proximity to the microprocessor.

3.2 Read Control for Burst Transactions

Similar to the optimized design described in Section 2, the read state machine will perform one of two different read cycles, depending upon the CFG input value. This section will concentrate on the differences between the two read configurations.

Initial Configuration

Refer to Figures 10 and 11 for the following read cycle discussion.

With CFG set to logic "0," the interfacing read state machine executes cacheable burst read cycles. This configuration will occur upon power-up and reset.

The microprocessor initiates a read access to the 28F016XS memory space by providing an address, driving W/R# low and activating ADS#. Monitoring the external bus of the Intel486 SX microprocessor, the state machine transitions into read control.

If ADS# = 0 and W/R# = 0 then READ CONTROL

At this point, CFG and CE# are examined to determine the configuration status of the 28F016XS-15s, and whether or not the current address targets the 28F016XS memory space. If CE# = "1," the state machine returns to an idle state waiting for a new access. Otherwise, the state machine will continue read access. In the initial configuration (CFG = "0"), read control will regulate ADV#, BRDY# and OE#.

At N = 1 (Figure 5), the counter loads address bits A_{3-2} and transitions ADV# low. With ADV# at logic "0," the interface initiates a read access to the 28F016XS-15s at N = 2. After initiating the read access, ADV# immediately switches to a logic "1" at N = 2 and then toggles active on every other clock edge until N = 8. After which, ADV# will remain inactive.

In the default SFI Configuration (SFI Configuration = 4), the first data will be accessible to the processor at N = 7. The rest of the data will be available for the processor to retrieve at N = 9, 11 and 13. The 28F016XS-15's output buffers are enabled at N = 4 and BRDY# is driven low at N = 6. The processor will sample BRDY# active and latch the information residing on the data bus at N = 7. If the processor drives BLAST# inactive indicating a burst transaction is in process, the interface logic will drive BRDY# active on every other clock edge until BLAST# is sampled active by the interface logic.

The Intel486 SX microprocessor requires a 3 ns hold time after sampling BRDY# active, therefore, the state machine will hold OE# active for 15 ns after the processor reads the last double-word. Then, the state machine will transition to an idle state where it deactivates OE# and waits for the Intel486 SX microprocessor to initiate a new bus cycle targeting the 28F016XS memory space.

Initial Configuration Timing Consideration

In the initial read configuration, CE# setup is a key system timing parameter.

To satisfy the 28F016XS-15 setup requirement, CE# must be valid 25 ns prior to the first rising CLK edge with ADV# = "0." Therefore, the maximum time allotted for the address decoding logic to generate CE# equals:

$$2 \times 1/33\ \text{MHz} - t_6 - t_{ELCH} = 19\ \text{ns}$$

Consult the appropriate datasheets for full timing information.

5

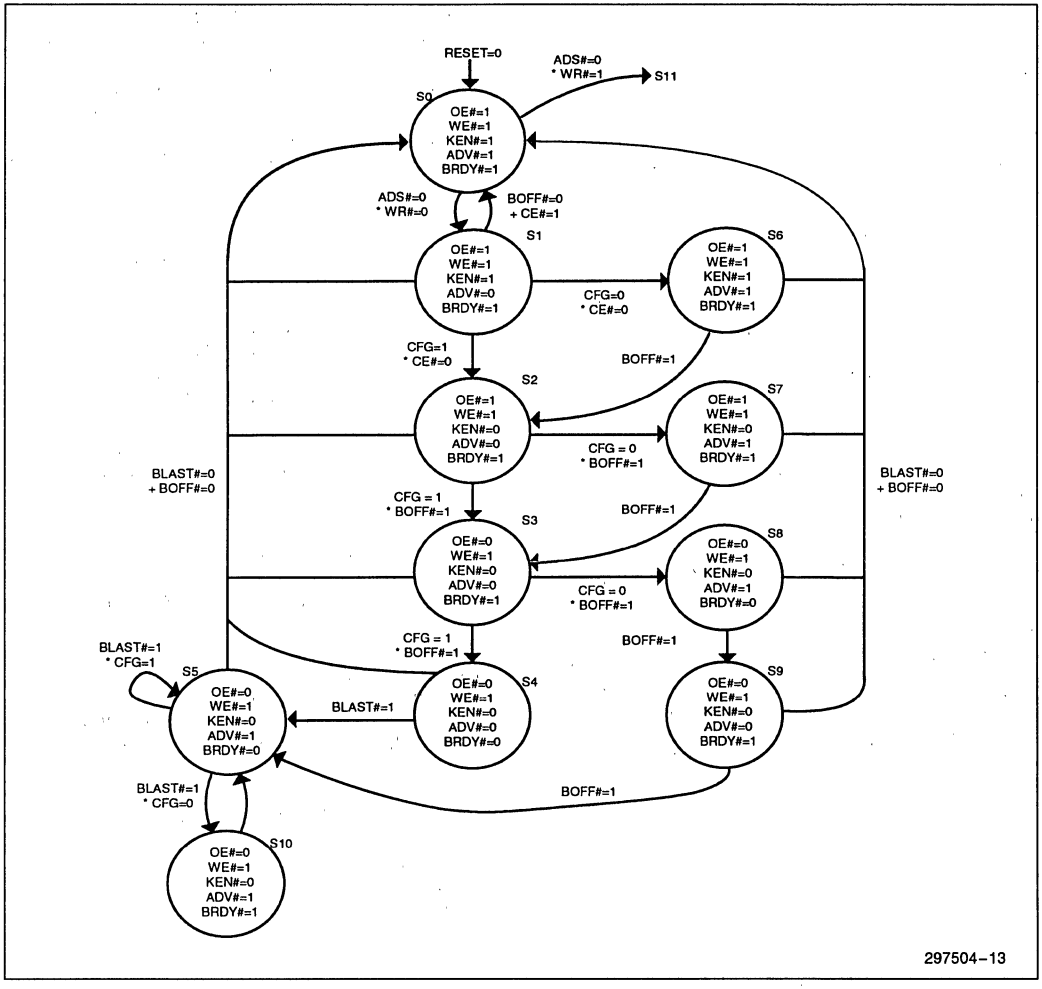


Figure 10. State Diagram of Single and Burst Read Control (Interface Shown in Figure 9)

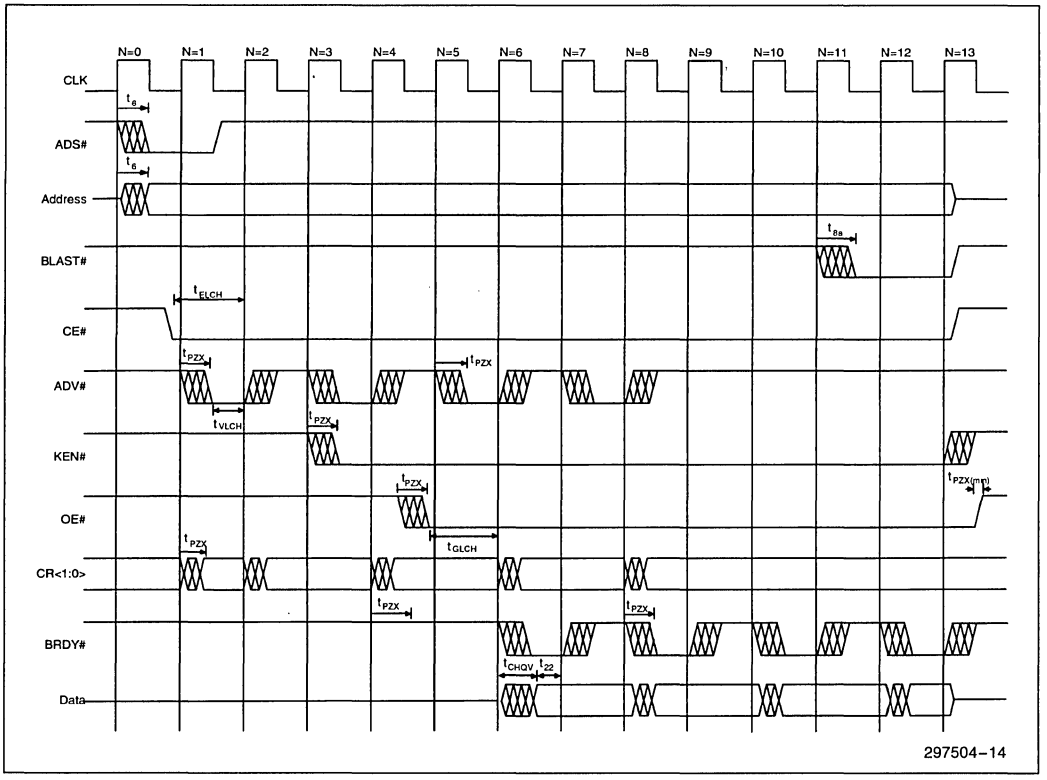


Figure 11. Example Initial Burst Read Cycle Showing Key Timing Specifications Requiring Consideration

Table 5. Example Initial Read Cycle Timing Specifications at 5V V_{CC}

Symbol	Description	Min	Max	Unit
t_6	ADS# Delay (Intel486 SX-33 microprocessor)	3	16	ns
t_{16}	RDY# Setup Time (Intel486 SX-33 microprocessor)	5		ns
t_{22}	D_{31-0} Setup Time (Intel486 SX-33 microprocessor)	5		ns
t_{ELCH}	CE_X# Setup Time to CLK (28F016XS-15)	25		ns
t_{VLCH}	ADV# Setup Time to CLK (28F016XS-15)	15		ns
t_{GLCH}	OE# Setup Time to CLK (28F016XS-15)	15		ns
t_{CHQV}	CLK to Data Delay (28F016XS-15)		20	ns
t_{PZX}	CLK Output Delay (22V10-15)	2	8	ns

NOTE:
Consult appropriate datasheets for up-to-date specifications.

Optimized Configuration

Refer to Figures 10 and 12 for the following discussion.

With the 28F016XS-15s in the optimized configuration (SFI Configuration = 2 at 33 MHz), and CFG set to a logic "1" value, the system interface executes cacheable burst cycles.

The Intel486 SX processor drives ADS# low, notifying the interface logic that a valid address is on the address bus. Monitoring the external bus of the Intel486 SX microprocessor, the state machine then transitions into read control.

If ADS# = 0 and W/R# = 0 then READ CONTROL

In optimized read control, the state machine controls OE#, KEN# and BRDY#. If CE# = "0," the state machine at N = 1 loads the two-bit counter (A_{3-2}) and activates ADV# (ADV# = "0") for the next four consecutive clock periods (N = 1 through 5). While ADV# is driven low, the counter increments through the Intel burst order (Table 2), supplying the 28F016XS-15s with a new address at N = 2, 3, 4 and 5. If CE# = "1," the state machine returns to an idle state waiting for a new memory access.

With the SFI Configuration value set to 2, new data will be available at N = 5, 6, 7 and 8. Driving BRDY# low at N = 4, the Intel486 SX microprocessor will sample BRDY# active at N = 5, which informs the processor of valid information on data pins D_{31-0} and that the 28F016XS memory space supports a burst read transfer. BRDY# is held low until the end of the burst cycle while the processor retrieves data on every rising clock edge. BRDY# is driven high upon sampling BLAST# low, marking the end of the burst cycle.

The Intel486 SX microprocessor requires a 3 ns hold time after sampling the last BRDY# active in a burst cycle. Therefore, the state machine will hold OE# active for 15 ns after the microprocessor samples the last double-word. At N = 9, the state machine transitions to an idle state, where it deactivates OE# and waits for the Intel486 SX microprocessor to initiate a new bus cycle targeting the 28F016XS memory space.

Optimized Configuration Timing Considerations

In the optimized configuration, CE# setup time is again a key system timing parameter. For information regarding the CE# setup time requirement, see the Initial Configuration Timing Considerations Timing section.

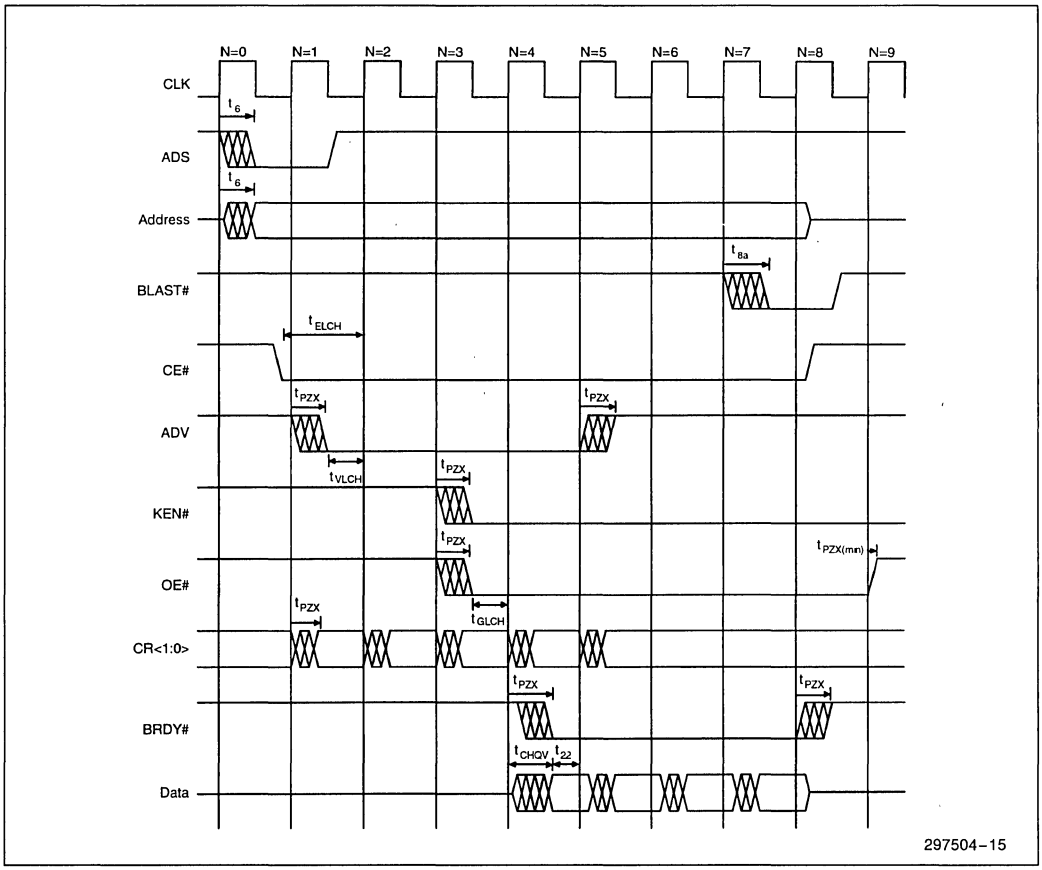


Figure 12. Example Burst Read Cycle Showing Key Timing Specifications Requiring Consideration

Table 6. Example Optimized Read Cycle Specifications at 5V V_{CC}

Symbol	Description	Min	Max	Unit
t_6	ADS# Delay (Intel486 SX-33 Microprocessor)	3	16	ns
t_{8a}	BLAST# Delay (Intel486 SX-33 Microprocessor)	3	20	ns
t_{22}	D_{31-0} Setup Time (Intel486 SX-33 Microprocessor)	5		ns
t_{ELCH}	CE_X# Setup Time to CLK (28F016XS-15)	25		ns
t_{VLCH}	ADV# Setup Time to CLK (28F016XS-15)	15		ns
t_{GLCH}	OE# Setup Time to CLK (28F016XS-15)	15		ns
t_{CHQV}	CLK to Data Delay (28F016XS-15)		20	ns
t_{PZX}	CLK Output Delay (22V10-15)	2	8	ns

NOTE:
Consult appropriate datasheets for up-to-date specifications.

3.3 Write Cycle Control

The write interface in for this design functionally behaves like the optimized design's write interface. The only difference between the two designs is the absence of the MUX in the standard interface. Therefore, the interface logic for this design does not have to concern itself with changing the data flow through the MUX. Instead, the interface simply loads and holds the address for the duration of the write operation.

For further detailed information about this cycle and write timing waveform, refer to Section 2.5.

4.0 INTERFACING TO OTHER INTEL486 MICROPROCESSORS

The Intel486 microprocessor family provides designers a large and diverse selection of CPUs, which offers designers different performance points to meet different market segment needs. Throughout the product family, the external bus architecture has remained consistent, which makes the 28F016XS interface to the entire Intel486 microprocessor family similar, if not identical, to the Intel486 SX microprocessor interface described in Sections 2 and 3. The 28F016XS-15 interface to the Intel486 SX-33 microprocessor works equally well for the following microprocessors at 5.0V V_{CC}.

- Intel486 SX-20, -25 processors
- Intel486 SX2-50 processor
- Intel486 DX-25, -33 processors
- Intel486 DX2-40, -50, -66 processors
- IntelDX4™-75, -100 processors (I/O buffers configured for 5.0V, V_{CC5} = 5.0V)

The 28F016XS-15 interface to the Intel486 SX-33 microprocessor also works well for the following Intel486 microprocessors at 3.3V V_{CC}. The 3.3V V_{CC} design utilizes a 22V10-15 low voltage PLD to control the interface between the 28F016XS-15 (operating at 3.3V V_{CC}) and the processor.

- Intel486 SX-20, -25 processors
- Intel486 DX-25 processor
- Intel486 DX2-40, -50 processors
- IntelDX4-75 processors (I/O buffers configured for 3.3V, V_{CC5} = 3.3V)

When the external bus frequency falls outside the 16.7 MHz through 33 MHz frequency range at 5.0V V_{CC} (12.5 MHz through 25 MHz at 3.3V V_{CC}), the optimized SFI Configuration value for the 28F016XS-15 differs in respect to the Intel486 SX-33 microprocessor design documented earlier. The state machine, therefore, requires slight modifications to accommodate the different SFI Configuration. Note, the initial read configuration and write control state machine remains consistent throughout all designs because they are not affected by the optimized SFI Configuration.

Intel486 SX-16 Microprocessor Interface at 5.0V V_{CC}

The 28F016XS-15's optimized SFI Configuration at 16 MHz equals 1. Therefore, 28F016XS-15 will begin driving data one CLK period after initiating the first read access. The optimized state machine must drive BRDY# and OE# active upon initiating the first access to the 28F016XS-15. BRDY# and OE# remain low throughout the burst cycle. The optimized and standard 28F016XS-15 interface to the Intel486 SX-16 microprocessor at this specific frequency deliver 1-0-0-0 and 2-0-0-0 wait-state read performance respectively.

Intel486 DX-50 Microprocessor Interface at 5V V_{CC}

Operating at 50 MHz, the 28F016XS-15's optimized SFI Configuration equals 3. The interface loads the two-bit counter and drives ADV# active at the first rising CLK edge after the processor initiates the read access. The optimized read state machine increments the two-bit counter and drives ADV# low every other CLK, thereby adhering to the *Alternating-A_1* and *Same-A_1* access rules (see Additional Information). The optimized and standard 28F016XS-15 interface to the Intel486 DX-50 microprocessor at this given frequency deliver 4-1-1-1 and 5-1-1-1 wait-state read performance respectively.

Intel486 DX-33 and IntelDX4-100 Microprocessor Interface at 3.3V V_{CC}

Operating at 33 MHz with 3.3V V_{CC}, the 28F016XS-15's optimized SFI Configuration equals 3. The interface loads the two-bit counter and drives ADV# active at the first rising CLK edge after the processor initiates the read access. The optimized read state machine increments the two-bit counter and drives ADV# low for two CLK periods and then strobes ADV# high for one CLK period. ADV# is again driven low for two CLK periods finishing the burst cycle. Refer to the *Alternating-A_1* and *Same-A_1* access rules (see Additional Information) for further information on consecutive accesses. The optimized and standard 28F016XS-15 interface to the Intel486 microprocessor at this frequency deliver 3-0-1-0 and 4-0-1-0 wait-state read performance respectively.

5.0 CONCLUSION

This technical paper has described the interface between the 28F016XS 16-Mbit flash memory component and the Intel486 microprocessor. This simple design has been implemented with a minimal number of components and achieves exceptional read performance. The 28F016XS provides the microprocessor with the nonvolatility and updateability of flash memory and the performance of DRAM. For further information about 28F016XS-15, consult reference documentation for a more comprehensive understanding of device capabilities and design techniques. Please contact your local Intel or distribution sales office for more information on Intel's flash memory products.

ADDITIONAL INFORMATION

Order Number	Document/Tools
290532	28F016XS 16-Mbit (1 Mbit x 16, 2 Mbit x 8) Synchronous Flash Memory Datasheet
297500	"Interfacing the 28F016XS to the i960® Microprocessor Family" (Technical Paper)
292147	AP-398, "Designing with the 28F016XS"
292146	AP-600, "Performance Benefits and Power/Energy Savings of 28F016XS-Based System Designs"
292163	AP-610, "Flash Memory In-System Code and Data Update Techniques"
292165	AB-62, "Compiled Code Optimizations for Flash Memories"
297372	16-Mbit Flash Product Family User's Manual
297508	FLASHBuilder Utility
Contact Intel/Distribution Sales Office	28F016XS Benchmark Utility
Contact Intel/Distribution Sales Office	28F016XS iBIS Model
Contact Intel/Distribution Sales Office	28F016XS VHDL Model
Contact Intel/Distribution Sales Office	28F016XS Timing Designer Library Files
Contact Intel/Distribution Sales Office	28F016XS Orcad and ViewLogic Schematic Symbols

REVISION HISTORY

Number	Description
001	Original Version
002	Incorporated initial read burst configuration, replacing the single read cycle. Added optimized design, improving system read performance
003	Added 3/5# pin reference to text and block diagrams (Figures 1 and 9) Revised state diagram numbering (Figures 4 and 7) General cosmetic changes.

APPENDIX A
PLD FILE FOR THE 28F016XS INTEL486™ MICROPROCESSOR INTERFACE

PLD file for the optimized interface described in Section 2.

```
TITLE                   Optimized 28F016XS / 486 Interface
PATTERN                 PDS
REVISION                1
AUTHOR                  Example
COMPANY NAME            Intel
DATE                    2/6/95

CHIP OPTIMIZED_28F016XS_486_INTERFACE 85C22V10

; input pins
PIN   1    CLK          ; clk frequency 33mhz
PIN        ADS          ; address strobe from 486
PIN        WR           ; multiplexed read/write strobe
PIN        BLAST        ; BLAST from the 486
PIN        CE           ; CE from the address decoding logic
PIN        CFG          ; informs interface to changes to the SFI Configuration
PIN        A2           ; lower address lines from the 486 used
PIN        A3           ; in loading the counter
PIN        RESET        ; system reset
PIN   25   GLOBAL
; output pins
PIN        ADV          ; address valid input
PIN        /KEN         ; cache control
PIN        /OE          ; output enable input
PIN        /WE          ; write enable input
PIN        /BRDY        ; initiating a burst cycle, 486 input
PIN        SWITCH       ; control MUX switching
PIN        Q0           ; state variable
PIN        CTR0         ; lower bit of the 2bit counter
PIN        CTR1         ; higher bit of the 2bit counter

STRING LD '(/ADS)'      ; load
STRING INC '(/ADV)'     ; increment

STATE MOORE_MACHINE
DEFAULT_BRANCH S0

; state assignments
S0 = /ADV * /BRDY * /OE * /WE * /KEN * /SWITCH * /Q0
S1 = /ADV * /BRDY * /OE * /WE * /KEN *  SWITCH * /Q0
S2 = /ADV * /BRDY *  OE * /WE *  KEN *  SWITCH * /Q0
S3 = /ADV *  BRDY *  OE * /WE *  KEN *  SWITCH * /Q0
S4 =  ADV *  BRDY *  OE * /WE *  KEN *  SWITCH * /Q0
S5 =  ADV * /BRDY *  OE * /WE *  KEN *  SWITCH * /Q0
```

297504-16

```
S6 = ADV * /BRDY * /OE * /WE * /KEN * SWITCH * Q0
S7 = ADV * /BRDY * /OE * /WE * KEN * SWITCH * Q0
S8 = ADV * BRDY * OE * /WE * KEN * SWITCH * Q0
S9 = /ADV * /BRDY * OE * /WE * KEN * SWITCH * Q0
S10 = ADV * /BRDY * /OE * /WE * /KEN * SWITCH * /Q0
S11 = ADV * /BRDY * /OE * WE * /KEN * SWITCH * /Q0
S12 = ADV * /BRDY * /OE * WE * /KEN * SWITCH * Q0
S13 = ADV * BRDY * /OE * /WE * /KEN * SWITCH * /Q0

; state transitions
S0 := (/ADS * /WR * /CFG)          -> S6  ; initial config READ cycle
   + (/ADS * /WR * CFG)            -> S1  ; optimized config READ cycle
   + (/ADS * WR)                   -> S10
                                  +-> S0
S1 := (/CFG * /CE * BOFF)          -> S7
   + ( CFG * /CE * BOFF)           -> S2
   + CE * /BOFF                    -> S0
S2 := /CFG * BOFF                  -> S8
   + CFG * BOFF                    -> S3
                                  +-> S0
S3 := /BLAST + /BOFF               -> S0
   + BLAST                         -> S4
S4 := /BLAST + /BOFF               -> S0
   + ( BLAST * /CFG)               -> S5
   + ( BLAST * CFG)                -> S4
S5 := BOFF                         -> S4
   + /BOFF                         -> S0
S6 := BOFF                         -> S1
   + /BOFF                         -> S0
S7 := BOFF                         -> S2
   + /BOFF                         -> S0
S8 := /BLAST + /BOFF               -> S0
   + BLAST                         -> S9
S9 := BOFF                         -> S4
   + /BOFF                         -> S0
S10:= /CE                          -> S11 ; write cycle
   + CE + /BOFF                    -> S0
S11:= BOFF                         -> S12 ; WE low for two clocks
   + /BOFF                         -> S0
S12:= BOFF                         -> S13
   + /BOFF                         -> S0
S13:= VCC                          -> S0  ; ready

EQUATIONS
 ; implement RESET
  GLOBAL.RSTF = /RESET
 ; implement 2-bit burst counter
  CTR1 := (LD * A3) + (/LD * INC * /CTR1 * S1)
      + (/LD * INC * CTR1 * /S1) + (/LD * /INC * CTR1 * /S1)
  CTR0 := (/WR * LD * /A2) + (WR * LD * A2)
      + (/LD * INC * /CTR0) + (/LD * /INC * CTR0)
```

297504-17

PLD file for the optimized interface described in Section 3.

```
Title           Standard 28F016XS / Intel486™ Interface
Pattern         PDS
Revision        1
Author          Example
Company Name    Intel
Date            2/14/94

CHIP   28F016XS_486_Interface   85C22V10   ; 85C22V10-15

; input pins
PIN    1     CLK          ; clk frequency 33mhz
PIN          ADS          ; address strobe from 486
PIN          WR           ; multiplexed read/write strobe
PIN          BLAST        ; BLAST from the 486
PIN          CE           ; CE from the address decoding logic
PIN          CFG          ; informs interface to changes to the SFI Configuration
PIN          BOFF         ; BOFF input to processor
PIN          A2           ; lower address lines from the 486 used
PIN          A3           ; in loading the counter
PIN          RESET        ; system reset
PIN    25    GLOBAL
;output pins
PIN          /ADV         ; address valid input
PIN          /KEN         ; cache control
PIN          /OE          ; output enable input
PIN          /WE          ; write enable input
PIN          /BRDY        ; initiating a burst cycle, 486 input
PIN          Q0           ; state variable
PIN          Q1           ; state variable
PIN          CONT0        ; lower bit of the 2bit counter
PIN          CONT1        ; higher bit of the 2bit counter

STRING  LD '(/ADS)'       ; load
STRING  INC '(/ADV)'

STATE MOORE_MACHINE
DEFAULT_BRANCH  S0
; state assignments
S0  = /ADV * /KEN * /WE * /BRDY * /Q0 * /Q1
S1  =  ADV * /KEN * /WE * /BRDY * /Q0 * /Q1
S2  =  ADV *  KEN * /WE * /BRDY * /Q0 * /Q1
S3  =  ADV *  KEN * /WE * /BRDY * /Q0 *  Q1
S4  =  ADV *  KEN * /WE *  BRDY * /Q0 * /Q1
S5  = /ADV *  KEN * /WE *  BRDY * /Q0 * /Q1
S6  = /ADV * /KEN * /WE * /BRDY * /Q0 *  Q1
S7  = /ADV *  KEN * /WE * /BRDY * /Q0 * /Q1
S8  = /ADV *  KEN * /WE * /BRDY * /Q0 *  Q1
S9  =  ADV *  KEN * /WE *  BRDY * /Q0 *  Q1
S10 = /ADV *  KEN * /WE * /BRDY *  Q0 *  Q1
S11 = /ADV * /KEN * /WE * /BRDY *  Q0 * /Q1
```

297504-18

5

```
S12 = /ADV * /KEN *  WE * /BRDY * /Q0 * /Q1
S13 = /ADV * /KEN *  WE * /BRDY * /Q0 *  Q1
S14 = /ADV * /KEN * /WE *  BRDY *  Q0 * /Q1

; state transitions
S0  :=  ADS * /BOFF          -> S0      ; start of an access
    + /ADS * /WR * /BOFF     -> S1
    + /ADS *  WR * /BOFF     -> S11
S1  := /CFG * /CE            -> S6      ; not reconfigured
    +  CFG * /CE             -> S2      ; reconfig active low
    +  CE                    -> S0
S2  := /CFG *  BOFF          -> S7      ; if chip enable is de-asserted,
    +  CFG *  BOFF           -> S3      ; quit the access and return
    + /BOFF                  -> S0
S3  := /CFG *  BOFF          -> S8
    +  CFG *  BOFF           -> S4
    + /BOFF                  -> S0
S4  := /BLAST + /BOFF        -> S0      ; situations will only ocur during
    +  BLAST                 -> S5      ; a BOFF.
S5  := /BLAST + /BOFF        -> S0      ; continous cycling until BLAST is
    + /CFG *  BLAST          -> S10     ; presented - end the burst cycle
    +  CFG *  BLAST          -> S5
S6  := /BOFF                 -> S0
    +  BOFF                  -> S2
S7  := /BOFF                 -> S0
    +  BOFF                  -> S3
S8  := /BOFF + /BLAST        -> S0
    +  BOFF                  -> S9
S9  := /BOFF                 -> S0
    +  BOFF                  -> S5
S10 := /BOFF                 -> S0
    +  BOFF                  -> S5
S11 := /CE                   -> S12     ; situation will only occur during
    +  CE + /BOFF            -> S0      ; during a BOFF.
S12 := /BOFF                 -> S0
    +  BOFF                  -> S13
S13 := /BOFF                 -> S0
    +  BOFF                  -> S14
S14 :=  VCC                  -> S0

EQUATIONS
 ; implement RESET
  GLOBAL.RSTF = /RESET
 ; implement 2-bit burst counter
  CTR1 := (LD * A3) + (/LD * INC * /CTR1 * S1)
      + (/LD * INC * CTR1 * /S2) + (/LD * /INC * CTR1 * /S2)
   CTR0 := (/WR * LD * A2) + (/LD * INC * /CTR0) + (/LD * /INC * CTR0)
 ; output enable control, triggered on falling clock edge
  OE := S2 + S3 + S4 + S5 + S7 + S8 + S9 + S10
  OE.CLKF = /CLK
```

297504-19

VALUE SERIES 100 FLASH MEMORY CARD 2-,4-,8-MEGABYTE

iMC002FLSC, iMC004FLSC, iMC008FLSC

- **Low-Cost Linear Flash Card**
- **Single Supply: 5 Volt Operation**
- **FAST Read Performance — 100 ns Maximum Access Time**
- **x16 Data Interface**
- **High-Performance Random Writes — 10 μs Typical Word Write**
- **50 μA Typical Deep Power-Down**
- **Automated Write and Erase Algorithms — 28F008SA Command Set**
- **State-of-the-Art 0.6 μm ETOX™ IV Flash Technology**
- **100,000 Erase Cycles per Block**
- **64-KWord Blocks**
- **PC Card Standard Type 1 Form Factor**

Intel's Value Series 100 card offers the lowest cost removable solid-state storage solution for code and data storage, high-performance disk emulation, and applications in mobile PCs and dedicated embedded applications. Manufactured with Intel's FlashFile™ memory, this card takes advantage of a revolutionary architecture that provides innovative capabilities, automated write/erase algorithms, reliable operation and very high read/write performance.

The flash memory card provides the lowest cost, highest performance nonvolatile read/write solution for solid-state storage applications. These applications are enhanced further with this product's symmetrically-blocked architecture, extended MTBF, and 5 Volt operation.

The flash memory card can be used as a simple x16 linear array of flash devices. The PC Card form factor offers an industry-standard pinout, removable linear flash memory, and the ability to upgrade system memory software without changing the board layout.

5

*Other brands and names are the property of their respective owners.

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

November 1995
Order Number: 290546-001

SERIES 2+ FLASH MEMORY CARDS
4-, 8-, 20- AND 40-MEGABYTE

iMC004FLSP, iMC008FLSP, iMC020FLSP, iMC040FLSP

- **Single Power Supply**
- **Automatically Reconfigure for 3.3V and 5V Systems**
- **150 ns Maximum Access Time with 5V Power Supply**
- **250 ns Maximum Access Time with 3.3V Power Supply**
- **High-Performance Random Writes**
  - **— 0.85 MB/S Sustained Throughput**
  - **— 1 KB Burst Write at 10 MB/S**
- **12 μA Typical Deep Power-Down**
- **Revolutionary Architecture**
  - **— Pipelined Command Execution**
  - **— Write during Erase**
  - **— Series 2 Command Super-Set**
- **State-of-the-Art 0.6 μm ETOX™ IV Flash Technology**
- **1 Million Erase Cycles per Block**
- **Up to 640 Independent Lockable Blocks**
- **PCMCIA 2.1/JEIDA 4.1-Compatible**
- **PCMCIA Type 1 Form Factor**
- **Series 2+ User's Manual**

Intel's Series 2+ Flash Memory Card sets the new record for high-performance disk emulation and eXecute-In-Place (XIP) applications in mobile PCs and dedicated equipment. Manufactured with Intel's 28F016SA 16-Mbit (DD28F032SA 32-Mbit) FlashFile™ Memory, this card takes advantage of a revolutionary architecture that provides innovative capabilities, low-power operation and very high read/write performance.

The Series 2+ Card provides today's highest density, highest performance nonvolatile read/write solution for solid-state storage applications. These applications are enhanced further with this product's symmetrically blocked architecture, extended MTBF, low-power 3.3V operation, built-in V_{PP} generator, and multiple block-locking methods. The Series 2+ Card's dual read and write voltages allow interchange between 3.3V and 5.0V systems.

290491-8

ETOX and FlashFile™ are trademarks of Intel Corporation.

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

November 1994
Order Number: 290491-006

SERIES 2 FLASH MEMORY CARDS
iMC002FLSA, iMC004FLSA, iMC010FLSA, iMC020FLSA

Extended Temperature Specifications Included

- **2, 4, 10 and 20 Megabyte Capacities**
- **PCMCIA 2.1/JEIDA 4.1 68-Pin Standard**
  - **— Hardwired Card Information Structure**
  - **— Byte- or Word-Wide Selectable**
- **Component Management Registers for Card Status/Control and Flexible System Interface**
- **Automatic Erase/Write**
  - **— Monitored with Ready/Busy Output**
- **Card Power-Down Modes**
  - **— Deep-Sleep for Low Power Applications**
- **Mechanical Write Protect Switch**
- **Solid-State Reliability**
- **Intel FlashFile™ Architecture**
- **High-Performance Read Access**
  - **— 150 ns Maximum**
- **High-Performance Random Writes**
  - **— 10 μs Typical Word Write**
- **Erase Suspend to Read Command**
  - **— Keeps Erase as Background Task**
- **Nonvolatility (Zero Retention Power)**
  - **— No Batteries Required for Back-up**
- **ETOX™ III 0.8μ Flash Memory Technology**
  - **— 5V Read, 12V Erase/Write**
  - **— High-Volume Manufacturing Experience**
- **Extended Temperature Version**
  - **— −40°C to +85°C**

Intel's Series 2 Flash Memory Card facilitates high-performance disk emulation in mobile PCs and dedicated equipment. Manufactured with Intel's ETOX™ III 0.8μ, FlashFile Memory devices, the Series 2 Card allows code and data retention while erasing and/or writing other blocks. Additionally, the Series 2 Flash Memory Card features low power modes, flexible system interfacing and a 150 ns read access time. When coupled with Intel's low-power microprocessors, these cards enable high-performance implementations of mobile computers and systems.

Series 2 Cards conform to the Personal Computer Memory Card International Association (PCMCIA 2.1)/Japanese Electronics Industry Development Association (JEIDA 4.1) 68-pin standard, providing electrical and physical compatibility.

Data file management software, Flash Translation Layer (FTL), provides data file storage and memory management, much like a disk operating system. Intel's Series 2 Flash Memory Cards, coupled with flash file management software, effectively provide a removable, all-silicon mass storage solution with higher performance and reliability than disk-based memory architectures.

Designing with Intel's FlashFile Architecture enables OEM system manufacturers to design and manufacture a new generation of mobile PCs and dedicated equipment where high performance, ruggedness, long battery life and lighter weight are a requirement. For large user groups in workstation environments, the Series 2 Cards provide a means to securely store user data and backup system configuration/status information.

The complete document for this product is available from Intel's Literature Center at 1-800-548-4725.

November 1995
Order Number: 290434-005

NORTH AMERICAN DISTRIBUTORS

ALABAMA

Anthem Electronics
600 Boulevard South
Suite 104F & H
Huntsville 35802
Tel: (205) 890-0302

Arrow/Schweber Electronics
1015 Henderson Road
Huntsville 35805
Tel: (205) 837-6955
FAX: (205) 721-1581

Hall-Mark Computer
4890 University Square
Huntsville 35816
Tel: (800) 409-1483

Hamilton Hallmark
4890 University Square
Suite 1
Huntsville 35816
Tel: (205) 837-8706
FAX: (205) 830-2565

MTI Systems Sales
4950 Corporate Drive
Suite 120
Huntsville 35805
Tel: (205) 830-9526
FAX: (205) 830-9557

Pioneer Technologies Group
4835 University Square
Suite 5
Huntsville 35805
Tel: (205) 837-9300
FAX: (205) 837-9358

Wyle Electronics
7800 Governers Dr., W.
Tower Building, 2nd Floor
Huntsville 35807
Tel: (205) 830-1119
FAX: (205) 830-1520

ARIZONA

Alliance Electronics
7550 East Redfield Rd
Scottsdale 85260
Tel: (602) 261-7988

Anthem Electronics
1555 West 10th Place
Suite 101
Tempe 85281
Tel: (602) 966-6600
FAX: (602) 966-4826

Arrow/Schweber Electronics
2415 West Erie Drive
Tempe 85282
Tel: (602) 431-0030
FAX: (602) 431-9555

Avnet Computer
1626 South Edwards Dr
Tempe 85281
Tel: (800) 426-7999

Hall-Mark Computer
4637 South 37th Place
Phoenix 85040
Tel: (800) 409-1483

Pioneer Standard
1438 West Broadway
Suite B-140
Tempe 85282
Tel: (602) 350-9335

Hamilton Hallmark
4637 South 36th Place
Phoenix 85040
Tel: (602) 437-1200
FAX: (602) 437-2348

Wyle Electronics
4141 East Raymond
Phoenix 85040
Tel: (602) 437-2088
FAX: (602) 437-2124

CALIFORNIA

Anthem Electronics
9131 Oakdale Avenue
Chatsworth 91311
Tel: (818) 775-1333
FAX: (818) 775-1302

Anthem Electronics
1 Okdfield Drive
Irvine 92718-2809
Tel: (714) 768-4444
FAX: (714) 768-6456

Anthem Electronics
580 Menlo Drive
Suite 8
Rocklin 95677
Tel: (916) 624-9744
FAX: (916) 624-9750

Anthem Electronics
9369 Carroll Park Drive
San Diego 92121
Tel: (619) 453-9005
FAX: (619) 546-7893

Anthem Electronics
1160 Ridder Park Drive
San Jose 95131
Tel: (408) 453-1200
FAX: (408) 441-4504

Arrow/Schweber Electronics
26707 West Agoura Road
Calabasas 91302
Tel: (818) 880-9686
FAX: (818) 772-8930

Arrow/Schweber Electronics
48834 Kato Road
Suite 103
Fremont 94538
Tel: (510) 490-9477
FAX: (510) 490-1084

Arrow/Schweber Electronics
6 Cromwell
Suite 100
Irvine 92718
Tel: (714) 581-4622
FAX: (714) 454-4206

Arrow/Schweber Electronics
9511 Ridgehaven Court
San Diego 92123
Tel: (619) 565-4800
FAX: (619) 279-8062

Arrow/Schweber Electronics
1180 Murphy Avenue
San Jose 95131
Tel: (408) 441-9700
FAX: (408) 453-4810

Avnet Computer
1 Mauchley
Irvine 92718
Tel: (800) 426-7999

Avnet Computer
371 Van Ness Way
Torrance 90501
Tel: (800) 426-7999

Avnet Computer
15950 Bernardo Ctr Dr
Suite 6
San Diego 92127
Tel: (800) 426-7999

Avnet Computer
1175 Bordeaux Drive
Suite A
Sunnyvale 94089
Tel: (800) 426-7999

Hall-Mark Computer
21150 Califa Street
Woodland Hills 91367
Tel: (800) 409-1483

Hall-Mark Computer
15950 Bernardo Ctr Dr
Suite C
San Diego 92127
Tel: (800) 409-1483

Hall-Mark Computer
1175 Bordeaux Drive
Sunnyvale 94089
Tel: (800) 409-1483

Hall-Mark Computer
1 Mauchly
Irvine 92718
Tel: (800) 409-1483

Hall-Mark Computer
580 Menlo Drive
Suite 2
Rocklin 95765
Tel: (800) 409-1483

Hamilton Hallmark
3170 Pullman Street
Costa Mesa 92626
Tel: (714) 641-4100
FAX: (714) 641-4122

Hamilton Hallmark
2105 Lundy Avenue
San Jose 95131
Tel: (408) 435-3500
FAX: (408) 435-3720

Hamilton Hallmark
4545 Viewridge Avenue
San Diego 92123
Tel: (619) 571-7540
FAX: (619) 277-6136

Hamilton Hallmark
21150 Califa Street
Woodland Hills 91367
Tel: (818) 594-0404
FAX: (818) 594-8234

Hamilton Hallmark
580 Menlo Drive
Suite 2
Rocklin 95762
Tel: (916) 624-9781
FAX: (916) 961-0922

Pioneer Standard
5126 Clareton Drive
Suite 106
Agoura Hills 91301
Tel: (818) 865-5800

Pioneer Standard
217 Technology Drive
Suite 110
Irvine 92718
Tel: (714) 753-5090

Pioneer Technologies Group
134 Rio Robles
San Jose 95134
Tel: (408) 954-9100
FAX: (408) 954-9113

Pioneer Standard
4370 La Jolla Village Drive
San Diego 92122
Tel: (619) 546-4906

Wyle Electronics
15370 Barranca Pkwy
Irvine 92713
Tel: (714) 753-9953
FAX: (714) 753-9877

Wyle Electronics
15360 Barranca Pkwy
Suite 200
Irvine 92713
Tel: (714) 753-9953
FAX: (714) 753-9877

Wyle Electronics
2951 Sunrise Blvd.
Suite 175
Rancho Cordova 95742
Tel: (916) 638-5282
FAX: (916) 638-1491

Wyle Electronics
9525 Chesapeake Dr.
San Diego 92123
Tel: (619) 565-9171
FAX: (619) 365-0512

Wyle Electronics
3000 Bowers Avenue
Santa Clara 95051
Tel: (408) 727-2500
FAX: (408) 727-5896

Wyle Electronics
17872 Cowan Avenue
Irvine 92714
Tel: (714) 863-9953
FAX: (714) 263-0473

Wyle Electronics
26010 Mureau Road
Suite 150
Calabasas 91302
Tel: (818) 880-9000
FAX: (818) 880-5510

Zeus Arrow Electronics
6276 San Ignacio Avenue
Suite E
San Jose 95119
Tel: (408) 629-4689
FAX: (408) 629-4792

Zeus Arrow Electronics
6 Cromwell Street
Suite 100
Irvine 92718
Tel: (714) 581-4622
FAX: (714) 454-4355

COLORADO

Anthem Electronics
373 Inverness Dr. S.
Englewood 80112
Tel: (303) 790-4500
FAX: (303) 790-4532

Arrow/Schweber Electronics
61 Inverness Dr East
Suite 105
Englewood 80112
Tel: (303) 799-0258
FAX: (303) 799-0730

Avnet Computer
9605 Maroon Circle
Englewood 80111
Tel: (800) 426-7999

Hall-Mark Computer
9605 Maroon Circle
Englewood 80111
Tel: (800) 409-1483

Hamilton Hallmark
12503 East Euclid Dr
Suite 20
Englewood 80111
Tel: (303) 790-1662
FAX: (303) 790-4991

Hamilton Hallmark
710 Wooten Road
Suite 28
Colorado Springs 80915
Tel: (719) 637-0055
FAX: (719) 637-0088

Pioneer Technologies
5600 Greenwood Plaza Blvd.
Suite 201
Englewood 80111
Tel: (303) 773-8090

Wyle Electronics
451 East 124th Avenue
Thornton 80241
Tel: (303) 457-9953
FAX: (303) 457-4831

CONNECTICUT

Anthem Electronics
61 Mattatuck Heights Road
Waterburg 06705
Tel: (203) 575-1575
FAX: (203) 596-3232

Arrow/Schweber Electronics
860 N. Main St. Ext.
Wallingford 06492
Tel: (203) 265-7741
FAX: (203) 265-7988

Hall-Mark Computer
Still River Corporate Ctr
55 Federal Road
Danbury 06810
Tel: (800) 409-1483

Hamilton Hallmark
125 Commerce Court, Unit 6
Cheshire 06410
Tel: (203) 271-2844
FAX: (203) 272-1704

Pioneer Standard
2 Trap Falls Road
Shelton 06484
Tel: (203) 929-5600

FLORIDA

Anthem Electronics
5200 NW 3rd Avenue
Suite 206
Ft. Lauderdale 33309
Tel: (305) 484-0990

Anthem Electronics
598 S. Northlake Blvd.
Suite 1024
Altamonte Sprgs 32701
Tel: (813) 797-2900
FAX: (813) 796-4880

Arrow/Schweber Electronics
400 Fairway Drive
Suite 102
Deerfield Beach 33441
Tel: (305) 429-8200
FAX: (305) 428-3991

Arrow/Schweber Electronics
37 Skyline Drive
Suite 3101
Lake Mary 32746
Tel: (407) 333-9300
FAX: (407) 333-9320

Arrow/Schweber Electronics
4010 Boy Scout Dr.
Suite 295
Tampa 33607
Tel: (813) 873-1030
FAX: (813) 873-0077

Avnet Computer
541 S. Orlando Ave.
Suite 203
Maitland 32751
Tel: (800) 426-7999

Hall-Mark Computer
10491 72nd St. North
Largo 34647
Tel: (800) 409-1483

Hall-Mark Computer
13700 58th St. North
Suite 206
Clearwater 34620
Tel: (800) 409-1483

Hamilton Hallmark
3350 N.W. 53rd Street
Suite 105-107
Ft. Lauderdale 33309
Tel: (305) 484-5482
FAX: (305) 484-2995

NORTH AMERICAN DISTRIBUTORS (Cont'd)

Hamilton Hallmark
10491 72nd St. North
Largo 34647
Tel: (813) 541-7440
FAX: (813) 544-4394

Hamilton Hallmark
7079 University Blvd.
Winter Park 32792
Tel: (407) 657-3300
FAX: (407) 678-4414

Pioneer Technologies Group
337 Northlake Blvd
Suite 1000
Alta Monte Spgs 32701
Tel: (407) 834-9090
FAX: (407) 834-0865

Pioneer Technologies Group
674 South Military Trail
Deerfield Beach 33442
Tel: (305) 428-8877
FAX: (305) 481-2950

Wyle Electronics
1000 112th Circle North
St. Petersburg 33716
Suite 800
Tel: (813) 579-1518
FAX: (813) 579-1518

Zeus Arrow Electronics
37 Skyline Drive
Bldg D., Suite 3101
Lake Mary 32746
Suite 800
Tel: (407) 333-3055
FAX: (407) 333-9681

GEORGIA

Anthem Electronics
2400 Pleasant Hill Rd
Suites 9 & 10
Duluth 30136
Tel: (404) 931-3900
FAX: (404) 931-3902

Arrow/Schweber Electronics
4250 E Rivergreen Pkwy
Suite E
Duluth 30136
Tel: (404) 497-1300
FAX: (404) 476-1493

Avnet Computer
3425 Corporate Way
Suite G
Duluth 30136
Tel: (800) 426-7999

Hall-Mark Computer
3425 Corporate Way
Suite G
Duluth 30136
Tel: (800) 409-1483

Hamilton Hallmark
3425 Corporate Way
Suite G & A
Duluth 30136
Tel: (404) 623-5475
FAX: (404) 623-5490

Pioneer Technologies Group
4250C Rivergreen Pkwy
Duluth 30136
Tel: (404) 623-1003
FAX: (404) 623-0665

Wyle Electronics
6025 The Corners Pkwy
Suite 111
Norcross 30092
Tel: (404) 441-9045
FAX: (404) 441-9086

ILLINOIS

Anthem Electronics
1300 Remington Road
Suite A
Schaumberg 60173
Tel: (708) 884-0200
FAX: (708) 885-0480

Arrow/Schweber Electronics
1140 W Thorndale Rd
Itasca 60143
Tel: (708) 250-0500

Avnet Computer
1124 Thorndale Ave
Bensenville 60106
Tel: (800) 426-7999

Hall-Mark Computer
1124 Thorndale Ave
Bensenville 60106
Tel: (800) 409-1483

Hamilton Hallmark
1130 Thorndale Ave
Bensenville 60106
Tel: (800) 426-7999

MTI Systems Sales
1140 West Thorndale Avenue
Itasca 60143
Tel: (708) 250-8222
FAX: (708) 250-8275

Pioneer Standard
2171 Executive Drive
Suite 200
Addison 60101
Tel: (708) 495-9680
FAX: (708) 495-9831

Wyle Electronics
2055 Army Trail Road
Suite 140
Addison 60101
Tel: (800) 853-9953

Zeus Arrow Electronics
1140 W Thorndale Ave
Itasca 60143
Tel: (708) 250-0500

INDIANA

Arrow/Schweber Electronics
7108 Lakeview
Parkway West Drive
Indianapolis 46268
Tel: (317) 299-2071
FAX: (317) 299-2379

Avnet Computer
655 West Carmel Drive
Suite 160
Carmel 46032
Tel: (800) 426-7999

Hall-Mark Computer
655 West Carmel Drive
Suite 160
Carmel 46032
Tel: (800) 409-1483

Hamilton Hallmark
655 West Carmel Drive
Suite 160
Carmel 46032
Tel: (317) 575-3500
FAX: (317) 575-3535

Pioneer Standard
9350 Priority Way W Dr
Indianapolis 46250
Tel: (317) 573-0880
FAX: (317) 573-0979

KANSAS

Arrow/Schweber Electronics
9801 Legler Road
Lenexa 66219
Tel: (913) 541-9542
FAX: (913) 541-0328

Hall-Mark Computer
10809 Lakeview Ave
Lenexa 66219
Tel: (800) 409-1483

Hamilton Hallmark
10809 Lakeview Avenue
Lenexa 66215
Tel: (913) 888-4747
FAX: (913) 888-0523

MARYLAND

Anthem Electronics
7168A Columbia
Gateway Drive
Columbia 21046
Tel: (800) 239-6039

Arrow/Schweber Electronics
9800J Patuxent Woods Drive
Columbia 21046
Tel: (301) 596-7800
FAX: (301) 596-7821

Avnet Computer
7172 Columbia
Gateway Drive
Suite G
Columbia 21045
Tel: (800) 426-7999

Hall-Mark Computer
7172 Columbia
Gateway Drive
Suite G
Columbia 21046
Tel: (800) 409-1483

Hamilton Hallmark
10240 Old Columbia Road
Columbia 21046
Tel: (410) 988-9800
FAX: (410) 381-2036

North Atlantic Industries
Systems Division
7125 River Wood Drive
Columbia 21046
Tel: (301) 312-5800
FAX: (301) 312-5850

Pioneer Technologies Group
15810 Gaither Road
Gaithersburg 20877
Tel: (301) 921-0660
FAX: (301) 670-6746

Wyle Electronics
9101 Guilford Road
Suite 120
Columbia 21046
Tel: (301) 490-2170
FAX: (301) 490-2190

MASSACHUSETTS

Anthem Electronics
200 Research Drive
Wilmington 01887
Tel: (508) 657-5170
FAX: (508) 657-6008

Arrow/Schweber Electronics
25 Upton Drive
Wilmington 01887
Tel: (508) 658-0900
FAX: (508) 694-1754

Avnet Computer
10 D Centennial Drive
Peabody 01960
Tel: (800) 426-7999

Hall-Mark Computer
10 D Centennial Drive
Peabody 01960
Tel: (800) 409-1483

Hamilton Hallmark
10 D Centennial Drive
Peabody 01960
Tel: (508) 531-7430
FAX: (508) 532-9802

Pioneer Standard
44 Hartwell Avenue
Lexington 02173
Tel: (617) 861-9200
FAX: (617) 863-1547

Wyle Electronics
5 Oak Park Drive
Bedford 01803
Tel: (617) 271-9953
FAX: (617) 275-3809

Zeus Arrow Electronics
25 Upton Drive
Wilmington 01887
Tel: (508) 658-4776
FAX: (508) 694-2199

MICHIGAN

Arrow/Schweber Electronics
44720 Helm Street
Plymouth 48170
Tel: (313) 462-2290
FAX: (313) 462-2686

Avnet Computer
41650 Garden Brk Rd
Suite 120
Novi 48375
Tel: (800) 426-7999

Hall-Mark Computer
41650 Garden Brk Rd
Suite 120
Novi 48375
Tel: (800) 409-1483

Hamilton Hallmark
44191 Plymouth Oaks Blvd.
Suite 1300
Plymouth 48170
Tel: (313) 416-5806
FAX: (313) 416-5811

Hamilton Hallmark
41650 Garden Brk Rd
Suite 100
Novi 49418
Tel: (313) 347-4271
FAX: (313) 347-4021

Pioneer Standard
4505 Broadmoor S.E.
Grand Rapids 49512
Tel: (616) 698-1800
FAX: (616) 698-1831

Pioneer Standard
44190 Plymouth Oaks Blvd.
Plymouth 48170
Tel: (313) 525-1800
FAX: (313) 427-3720

MINNESOTA

Anthem Electronics
7646 Golden Triangle Drive
Eden Prairie 55344
Tel: (612) 944-5454
FAX: (612) 944-3045

Arrow/Schweber Electronics
10100 Viking Drive
Suite 100
Eden Prairie 55344
Tel: (612) 941-5280
FAX: (612) 942-7803

Avnet Computer
9800 Bren Roaqd East
Suite 410
Minnetonka 55343
Tel: (800) 426-7999

Hall-Mark Computer
9800 Bren Road East
Suite 410
Minnetonka 55343
Tel: (800) 409-1483

Hamilton Hallmark
9401 James Ave South
Suite 140
Bloomington 55431
Tel: (612) 881-2600
FAX: (612) 881-9461

Pioneer Standard
7625 Golden Triangle Drive
Suite G
Eden Prairie 55344
Tel: (612) 944-3355
FAX: (612) 944-3794

Wyle Electronics
1325 East 79th Street
Suite 1
Bloomington 55425
Tel: (612) 853-2280
FAX: (612) 853-2298

MISSOURI

Arrow/Schweber Electronics
2380 Schuetz Road
St. Louis 63141
Tel: (314) 567-6888
FAX: (314) 567-1164

Avnet Computer
3783 Rider Trail South
Earth City 63045
Tel: (800) 426-7999

Hall-Mark Computer
3783 Rider Trail South
Earth City 63045
Tel: (800) 409-1483

Hamilton Hallmark
3783 Rider Trail South
Earth City 63045
Tel: (314) 291-5350
FAX: (314) 291-0362

NEW HAMPSHIRE

Avnet Computer
2 Executive Park Drive
Bedford 03102
Tel: (800) 426-7999

NEW JERSEY

Anthem Electronics
26 Chapin Road, Unit K
Pine Brook 07058
Tel: (201) 227-7960
FAX: (201) 227-9246

Arrow/Schweber Electronics
4 East Stow Road
Unit 11
Marlton 08053
Tel: (609) 596-8000
FAX: (609) 596-9632

Arrow/Schweber Electronics
43 Route 46 East
Pine Brook 07058
Tel: (201) 227-7880
FAX: (201) 227-2064

Avnet Computer
1-B Keystone Avenue
Building 36
Cherry Hill 08003
Tel: (800) 426-7999

Hall-Mark Computer
1-B Keystone Avenue
Building 36
Cherry Hill 08003
Tel: (800) 409-1483

Hall-Mark Computer
10 Lanidex Plaza West
Parsippany 07054
Tel: (800) 409-1483

Hamilton Hallmark
1 Keystone Avenue
Building 36
Cherry Hill 08003
Tel: (609) 424-0110
FAX: (609) 751-2552

Hamilton Hallmark
10 Lanidex Plaza West
Parsippany 07054
Tel: (201) 515-5300
FAX: (201) 515-1601

MTI Systems Sales
43 Route 46 East
Pinebrook 07058
Tel: (201) 882-8780
FAX: (201) 539-6430

NORTH AMERICAN DISTRIBUTORS (Cont'd)

PioneerStandard
14-A Madison Road
Fairfield 07006
Tel: (201) 575-3510
FAX: (201) 575-3454

Wyle Electronics
115 Route 46, Bldg F
Mountain Lakes 07046
Tel: (201) 402-4970

NEW MEXICO

Alliance Electronics, Inc.
3411 Bryn Mawr N.E.
Albuquerque 87101
Tel: (505) 292-3360
FAX: (505) 275-6392

Avnet Computer
7801 Academy Road
Building 1, Suite 204
Albuquerque 87109
Tel: (800) 426-7999

NEW YORK

Anthem Electronics
47 Mall Drive
Commack 11725
Tel: (516) 864-6600
FAX: (516) 493-2244

Arrow/Schweber Electronics
3375 Brighton Henrietta Townline Road
Rochester 14623
Tel: (716) 427-0300
FAX: (716) 427-0735

Arrow/Schweber Electronics
20 Oser Avenue
Hauppauge 11788
Tel: (516) 231-1000
FAX: (516) 231-1072

Avnet Computer
2 Penn Plaza
Suite 1245
New York 10121
Tel: (800) 426-7999

Avnet Computer
1057 E. Henrietta Road
Rochester 14623
Tel: (800) 426-7999

Hall-Mark Computer
2 Penn Plaza
New York 10121
Tel: (800) 409-1483

Hall-Mark Computer
1057 E Henrietta Road
Rochester 14623
Tel: (800) 409-1483

Hamilton Hallmark
933 Motor Parkway
Hauppauge 11788
Tel: (516) 434-7470
FAX: (516) 434-7491

Hamilton Hallmark
1057 E Henrietta Road
Rochester 14623
Tel: (716) 475-9130
FAX: (716) 475-9119

Hamilton Hallmark
3075 Veterans
Memorial Hwy.
Ronkonkoma 11779
Tel: (516) 737-0600
FAX: (516) 737-0838

MTI Systems Sales
1 Penn Plaza
250 West 34th Street
New York 10119
Tel: (212) 643-1280
FAX: (212) 643-1288

Pioneer Standard
68 Corporate Drive
Binghamton 13904
Tel: (607) 722-9300
FAX: (607) 722-9562

Pioneer Standard
60 Crossway Pk West
Woodbury, Long Island
11797
Tel: (516) 921-8700
FAX: (516) 921-2143

Pioneer Standard
840 Fairport Park
Fairport 14450
Tel: (716) 381-7070
FAX: (716) 381-5955

Zeus Arrow Electronics
100 Midland Avenue
Port Chester 10573
Tel: (914) 937-7400
FAX: (914) 937-2553

NORTH CAROLINA

Anthem Electronics
4805 Greenwood
Suite 100
Raleigh 27604
Tel: (919) 782-3550

Arrow/Schweber Electronics
5240 Greensdairy
Road
Raleigh 27604
Tel: (919) 876-3132
FAX: (919) 878-9517

Avnet Computer
4421 Stuart Andrew
Boulevard
Suite 600
Charlotte 28217
Tel: (800) 426-7999

Hall-Mark Computer
3510 Spring Forest Rd
Suite B
Raleigh 27604
Tel: (800) 409-1483

Hamilton Hallmark
3510 Spring Forest Rd
Suite B
Raleigh 27604
Tel: (800) 409-1483

Hamilton Hallmark
5234 Greens Dairy Rd
Raleigh 27604
Tel: (919) 878-0819

Pioneer Technologies Group
2200 Gateway Ctr. Blvd
Suite 215
Morrisville 27560
Tel: (919) 460-1530

OHIO

Arrow/Schweber Electronics
6573 Cochran Road
Suite E
Solon 44139
Tel: (216) 248-3990
FAX: (216) 248-1106

Arrow/Schweber Electronics
8200 Washington
Village Drive
Centerville 45458
Tel: (513) 435-5563
FAX: (513) 435-2049

Avnet Computer
7764 Washington
Village Drive
Dayton 45459
Tel: (800) 426-7999

Avnet Computer
2 Summit Park Drive
Suite 520
Independence 44131
Tel: (800) 426-7999

Hall-Mark Computer
5821 Harper Road
Solon 44139
Tel: (800) 409-1483

Hall-Mark Computer
777 Dearborn Pk Lane
Suite L
Worthington 43085
Tel: (800) 409-1483

Hamilton Hallmark
5821 Harper Road
Solon 44139
Tel: (216) 498-1100
FAX: (216) 248-4803

Hamilton Hallmark
777 Dearborn Pk Lane
Suite L
Worthington 43085
Tel: (614) 888-3313
FAX: (614) 888-0767

MTI Systems Sales
23404 Commerce Pk
Road
Beachwood 44122
Tel: (216) 464-6688
FAX: (216) 464-3564

Pioneer Standard
4433 Interpoint Blvd
Dayton 45424
Tel: (513) 236-9900
FAX: (513) 236-8133

Pioneer Standard
4800 East 131st Street
Cleveland 44105
Tel: (216) 587-3600
FAX: (216) 663-1004

Wyle Electronics
6835 Cochran Rd.
Solon 44139
Tel: (216) 248-9996

OKLAHOMA

Arrow/Schweber Electronics
12101 East 51st Street
Suite 106
Tulsa 74146
Tel: (918) 252-7537
FAX: (918) 254-0917

Hamilton Hallmark
5411 S. 125th E. Ave
Suite 305
Tulsa 74146
Tel: (918) 254-6110
FAX: (918) 254-6207

Pioneer Standard
9717 East 42nd Street
Suite 105
Tulsa 74146
Tel: (918) 665-7840
FAX: (918) 665-1891

OREGON

Almac Arrow Electronics
9500 S.W. Nimbus Ave
Suite E
Beaverton 97008
Tel: (503) 629-8090
FAX: (503) 645-0611

Anthem Electronics
9090 SW Gemini Drive
Beaverton 97005
Tel: (503) 643-1114
FAX: (503) 626-7928

Avnet Computer
9750 SW Nimbus Ave.
Beaverton 97005
Tel: (800) 426-7999

Hall-Mark Computer
9750 SW Nimbus Ave.
Beaverton 97005
Tel: (800) 409-1483

Hamilton Hallmark
9750 SW Nimbus Ave.
Beaverton 97005
Tel: (503) 526-6200
FAX: (503) 641-5939

Pioneer Technologies
8905 Southwest
Nimbus Ave.
Suite 160
Beaverton 97005
Tel: (503) 626-7300
FAX: (503) 626-5300

Wyle Electronics
9640 Sunshine Court
Building G
Suite 200
Beaverton 97005
Tel: (503) 643-7900
FAX: (503) 646-5466

PENNSYLVANIA

Anthem Electronics
355 Business Ctr Drive
Horsham 19044
Tel: (215) 443-5150
FAX: (215) 675-9875

Avnet Computer
213 Executive Drive
Suite 320
Mars 16046
Tel: (800) 426-7999

Arrow/Schweber Electronics
2681 Mosside Blvd
Suite 204
Monroeville 15146
Tel: (412) 856-9490

Pioneer Technologies Group
259 Kappa Drive
Pittsburgh 15238
Tel: (412) 782-2300
FAX: (412) 963-8255

Pioneer Technologies Group
500 Enterprise Road
Keith Valley Bus.Ctr
Horsham 19044
Tel: (215) 674-4000

Wyle Electronics
1 Eves Drive
Suite 111
Marlton 08053-3185
Tel: (609) 985-7953
FAX: (609) 985-8757

TEXAS

Anthem Electronics
651 N. Plano Road
Suite 401
Richardson 75081
Tel: (214) 238-7100
FAX: (214) 238-0237

Anthem Electronics
14050 Summit Drive
Suite 119
Tel: (512) 388-0049
FAX: (512) 388-0271

Arrow/Schweber Electronics
Brake Ctr III, Bldg M1
11500 Metric Boulevard
Suite 160
Austin 78758
Tel: (512) 835-4180
FAX: (512) 832-5921

Arrow/Schweber Electronics
3220 Commander Drive
Carrollton 75006
Tel: (214) 380-6464
FAX: (214) 248-7208

Arrow/Schweber Electronics
19416 Park Row
Suite 190
Houston 77084
Tel: (713) 647-6868
FAX: (713) 492-8722

Avnet Computer
4004 Beltline
Suite 200
Dallas 75244
Tel: (800) 426-799

Avnet Computer
1235 North Loop West
Suite 525
Houston 77008
Tel: (800) 426-7999

Hall-Mark Computer
12211 Technology Blvd
Austin 78727
Tel: (800) 409-1483

Hall-Mark Computer
4004 Beltline Road
Suite 200
Dallas 75244
Tel: (800) 409-1483

Hall-Mark Computer
1235 North Loop West
Houston 77008
Tel: (800) 409-1483

Hamilton Hallmark
12211 Technology
Boulevard
Austin 78727
Tel: (512) 258-8848
FAX: (512) 258-3777

Hamilton Hallmark
11420 Page Mill Road
Dallas 75243
Tel: (214) 553-4300
FAX: (214) 553-4395

Hamilton Hallmark
8000 Westglen
Houston 77063
Tel: (713) 781-6100
FAX: (713) 953-8420

Pioneer Standard
1826D Kramer Lane
Austin 78758
Tel: (512) 835-4000
FAX: (512) 835-9829

Pioneer Standard
13765 Beta Road
Dallas 75244
Tel: (214) 263-3168
FAX: (214) 490-6419

Pioneer Standard
10530 Rockley Road
Suite 100
Houston 77099
Tel: (713) 495-4700
FAX: (713) 495-5642

Wyle Electronics
1810 Greenville Ave
Richardson 75081
Tel: (214) 235-9953
FAX: (214) 644-5064

Wyle Electronics
9208 Waterford Center
Blvd
Suite 150
Austin 78750
Tel: (512) 345-8853
FAX: (512) 345-9330

Wyle Electronics
2901 Wilcrest
Suite 120
Houston 77099
Tel: (713) 879-9953
FAX: (713) 879-9953

Zeus Arrow Electronics
3220 Commander Dr
Carrollton 75006
Tel: (214) 380-4330
FAX: (214) 447-2222

UTAH

Anthem Electronics
1279 West 2200 South
Salt Lake City 84119
Tel: (801) 973-8555
FAX: (801) 973-8909

Arrow/Schweber Electronics
1946 West Parkway
Boulevard
Salt Lake City 84119
Tel: (801) 973-6913
FAX: (801) 972-0200

NORTH AMERICAN DISTRIBUTORS (Cont'd)

Avnet Computer
1100 East 6600 South
Suite 150
Salt Lake City 84121
Tel: (800) 426-7999

Hall-Mark Computer
1100 East 6600 South
Suite 150
Salt Lake City
Tel: (800) 409-1483

Hamilton Hallmark
1100 East 6600 South
Suite 120
Salt Lake City 84121
Tel: (801) 266-2022
FAX: (801) 263-0104

Wyle Electronics
1325 West 2200 South
Suite E
West Valley 84119
Tel: (801) 974-9953
FAX: (801) 972-2524

WASHINGTON

Almac Arrow Electronics
14360 S.E. Eastgate Way
Bellevue 98007
Tel: (206) 643-9992
FAX: (206) 643-9709

Anthem Electronics
19017 120th Ave N.E.
Suite 102
Bothell 98011
Tel: (206) 483-1700
FAX: (206) 486-0571

Avnet Computer
8630 154th Ave. NE
Redmond 98052
Tel: (800) 426-7999

Hamilton Hallmark
8630 154th Avenue
Redmond 98052
Tel: (206) 881-6697
FAX: (206) 867-0159

Pioneer Technologies
2800 156th Ave S.E.
Suite 100
Bellevue 98007
Tel: (206) 644-7500

Wyle Electronics
15385 NE 90th St
Redmond 98052
Tel: (206) 881-1150
FAX: (206) 881-1567

WISCONSIN

Arrow/Schweber Electronics
200 N. Patrick
Suite 100
Brookfield 53045
Tel: (414) 792-0150
FAX: (414) 792-0156

Avnet Computer
2440 South 179th St
New Berlin 53416
Tel: (800) 426-7999

Hall-Mark Computer
2440 South 179th St
New Berlin 53146
Tel: (800) 409-1483

Hamilton Hallmark
2440 South 179th St
New Berlin 53146
Tel: (414) 797-7844
FAX: (414) 797-9259

Pioneer Standard
120 Bishops Way
Suite 163
Brookfield 53005
Tel: (414) 780-3600
FAX: (414) 780-3613

Wyle Electronics
150 North Patrick
Building 7, Suite 150
Brookfield 53045
Tel: (414) 879-0434
FAX: (414) 879-0474

ALASKA

Avnet Computer
1400 W Benson Blvd
Suite 400
Anchorage 99503
Tel: (800) 426-7999

CANADA

ALBERTA

Avnet Computer
1144 29th Avenue NE
Suite 108
Calgary T2E 7P1
Tel: (800) 387-3406

Pioneer/Pioneer
560, 1212-31 Ave. NE
Calgary T2E 7S8
Tel: (403) 291-1988
FAX: (403) 295-8714

BRITISH COLUMBIA

Almac Arrow Electronics
8544 Baxter Place
Burnaby V5A 4T8
Tel: (604) 421-2333
FAX: (604) 421-5030

Hamilton Hallmark
8610 Commerce Court
Burnaby V5A 4N6
Tel: (604) 420-4101
FAX: (604) 420-5376

Pioneer/Pioneer
4455 North 6 Road
Rochmond V6V 1P6
Tel: (604) 273-5575
FAX: (604) 273-2413

MANITOBA

Pioneer/Pioneer
540 Marjorie Street
Winnipeg R3H 0S9

ONTARIO

Arrow/Schweber Electronics
36 Antares Drive
Unit 100
Nepean K2E 7W5
Tel: (613) 226-6903
FAX: (613) 723-2018

Arrow/Schweber Electronics
1093 Meyerside, Unit 2
Mississauga L5T 1M4
Tel: (416) 670-2010
FAX: (416) 670-5863

Avnet Computer
Canada System Engineering Group
151 Superior Blvd.
Mississauga L5T 2L1
Tel: (800) 387-3406

Avnet Computer
190 Colonade Road
Nepean K2E 7J5
Tel: (800) 387-3406

Canada System Engineering Group
151 Superior Boulevard
Mississauga L5T 2L1
Tel: (800) 387-3406

Hamilton Hallmark
151 Superior Blvd.,
Unit 1-6
Mississauga L5T 2L1
Tel: (416) 564-6060
FAX: (416) 564-6033

Hamilton Hallmark
190 Colonade Road
Nepean K2E 7J5
Tel: (613) 226-1700
FAX: (613) 226-1184

Pioneer/Pioneer
3415 American Drive
Mississauga L4V 1T6
Tel: (416) 507-2600
FAX: (416) 507-2831

Pioneer/Pioneer
155 Colonnade Rd., S.
Suite 17
Nepean K2E 7K3
Tel: (613) 226-8840
FAX: (613) 226-6352

QUEBEC

Arrow/Schweber Electronics
1100 Street Regis Blvd
Dorval H9P 2T5
Tel: (514) 421-7411
FAX: (514) 421-7430

Gates Arrow Electronics
500 Boul.
St-Jean-Baptiste Ave
Quebec H2E 5R9
Tel: (418) 871-7500
FAX: (418) 871-6816

Avnet Computer
7575 Trans Canada
Suite 601
St. Laurent H4T 1V6
Tel: (800) 265-1135

Hamilton Hallmark
7575 Transcanada Hwy
Suite 600
Street Laurent H4T 2V6
Tel: (514) 335-1000
FAX: (514) 335-2481

Pioneer/Pioneer
520 McCaffrey
Street Laurent H4T 1N1
Tel: (514) 737-9700
FAX: (514) 737-5212

Embedded Microprocessors

For over 10 years, the industry-accepted 186 family has continued to provide 16-bit, high-performance solutions for embedded microprocessor designs.

Later, with the introduction of the Intel386™ embedded processor family, embedded designers have a 32-bit performance solution. The Intel386 processors provide easy upgradability for existing 16-bit designs while providing new, high-performance features such as protected mode programming and DOS compatibility. With the introduction of the Intel386 EX embedded processor, designers have a highly-integrated solution for embedded systems.

In 1995, Intel introduced embedded designers to the popular Intel486™ processor family used in desktop computers. Now, embedded designers have a 32-bit solution that includes an internal cache and a RISC-technology core for added performance. For added performance, the embedded IntelDX2™ processor's internal-core clock frequency is twice as fast as the external clock frequency. The highest-performing member of the family, the IntelDX4™ processor, operates at three-times the external clock frequency.

Recently, Intel introduced two Ultra-Low Power members of the Intel486 processor family. Now, battery-operated, embedded designs can have the advantages of the Intel architecture.

ISBN 1-55512-249-3

Order Number 272396-003
Printed in USA/0196/25k/RRD/DS
Semiconductor Products

Printed on Recycled Material