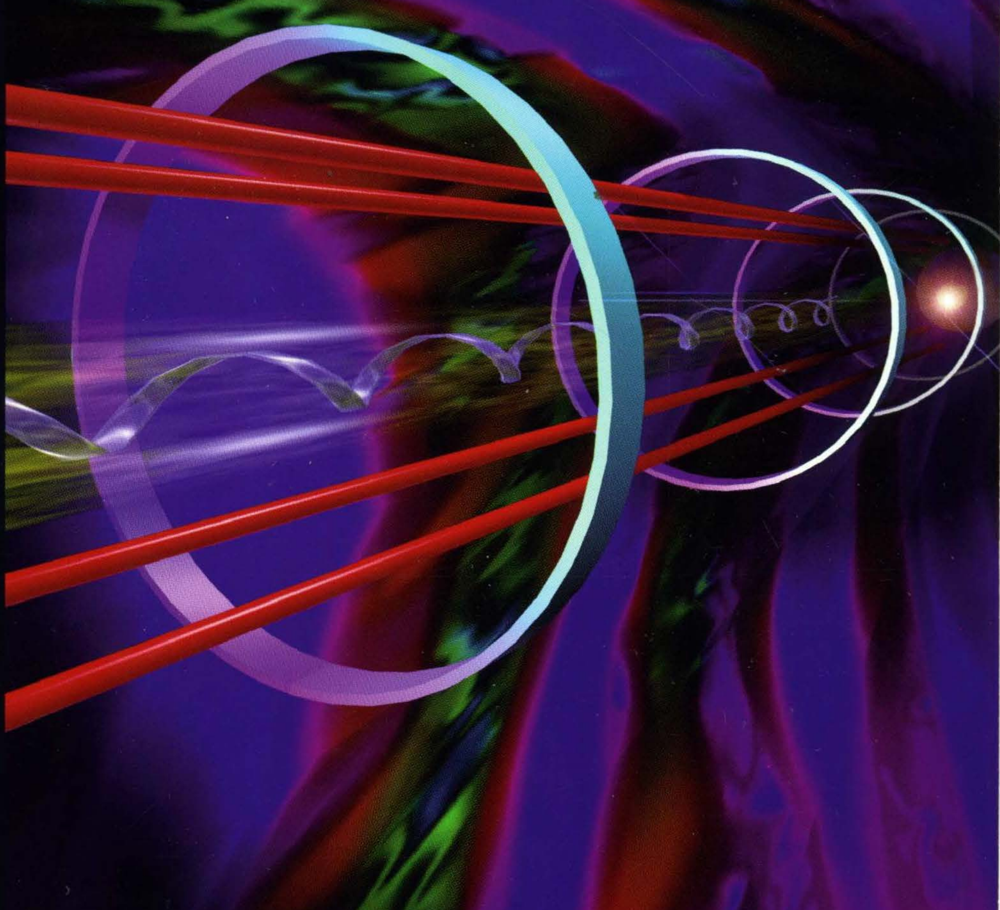
Intel486™ Microprocessors and Related Products

Microprocessors

PCIsets

Peripheral Components



intel®

intel® Intel486™ Microprocessors and Related Products 1995

intel®

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
P.O. Box 7641
Mt. Prospect, Il 60056-7641

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

CURRENT DATABOOKS

Product line databooks contain datasheets, application notes, article reprints, and other design information. 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 international sales office.

Title	Intel Order Number	ISBN
Automotive Products	231792	N/A
Embedded Applications (2 vol. set)	270648	1-55512-242-6
Embedded Microcontrollers	270646	1-55512-230-2
Embedded Microprocessors	272396	1-55512-231-0
Flash Memory (2 vol. set)	210830	1-55512-232-9
Intel486™ Microprocessors and Related Products	241731	1-55512-235-3
i960® Processors and Related Products	272084	1-55512-234-5
Military and Special Products	210461	N/A
Networking	297360	1-55512-236-1
OEM Boards, Systems and Software	280407	1-55512-237-X
Packaging	240800	1-55512-238-8
Pentium™ Processors and Related Products	241732	1-55512-239-6
Peripheral Components	296467	1-55512-240-X

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.

intel®

24-HOUR AUTOMATED TECHNICAL SUPPORT*

Intel's Application Bulletin Board System (BBS) and FaxBack System are at your service, 24-hours a day, at no charge, and the information is updated frequently.

FaxBack SYSTEM

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 is also available:

- Microcontroller and Flash
- OEM Branded Systems
- Multibus/BBS Listing
- Multimedia
- Development Tools
- Quality and Reliability/Change Notification
- Microprocessor/PCI/Peripheral
- Intel Architecture Lab

To use FaxBack for Intel components and systems, dial **(800) 628-2283** or (916) 356-3105 (U.S. and Canada) *or +44{0} 1793-496646 (Europe)* and follow the automated voice-prompt menu. Document orders will be faxed to the fax number you specify. Catalogs are updated twice a month, so call for the latest information!

BULLETIN BOARD SYSTEM

Intel's Application Bulletin Board System (BBS) enables file retrieval 24-hours a day. The following can be located on the BBS:

- Software Drivers
- Tool Information
- Software/Application Utilities
- Product/Technical Documentation
- Firmware Upgrades
- Quality and Reliability Data

To use the Intel Application BBS (components and systems), dial **(916) 356-3600** for download access (U.S. and Canada) *or +44{0} 1793-496340 (Europe).* The BBS will support 1200–19200 baud rate modem. *Typical modem configuration: 9600 baud rate, No Parity, 8 Data Bits, 1 Stop Bit.* A directory listing of BBS files is also available through FaxBack or our 800 BBS (800-897-2536).

Retail Products

Information on Intel's retail products (Coprocessors and wireless, video, personal conferencing and network products) is available through the following services:

Internet : ftp.intel.com (143.185.65.2)
CompuServe : GO INTELFORUM (modem settings: E-7-1, up to 14.4 Kbps)

Country	BBS (N-8-1, up to 14.4 Kbps)	FaxBack
North America	(503) 264-7999	(800) 525-3019 or (503) 264-6835
Europe	+44 1 793-432955	+44 1 793-432509
Australia	+61 2 975-3066	+61 2 975-3922
Taiwan	+886 2 718-6422	+886 2 514-0815
Singapore	+65 256-4776	+65 256-5350
Hong Kong	+852 530-4116	+852 844-4448
Korea	+822 784-3430	+822 767-2594

*Support services provided courtesy of Intel Application Support

intel®

Intel486™ Microprocessor and Related Products

Microprocessors, PCIsets, Peripheral Components

1995

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

Intel Corporation
Literature Sales
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.

Table of Contents

Table of Contents (Continued)

CHAPTER 5
Flash Memory Components

CHAPTER 6
Intel486™ Microprocessor SmartDie™ Products

Alphanumeric Index

1

Overview

Intel486™ MICROPROCESSOR

INTRODUCTION

Intel microprocessors and peripherals provide a broad range of time-saving, energy-efficient, high-performance solutions to designers of both mobile and desktop microprocessor-based systems. Intel's microprocessor/peripheral interface delivers *time* and *performance* advantages to the designers of microprocessor-based systems, meeting their demand for greater performance, lower power consumption and a wider variety of built-in features for their customers.

HIGH-PERFORMANCE ENTRY-LEVEL SYSTEMS

Intel offers an entire product line of entry-level microprocessors for desktop and mobile systems, ranging from the 25 MHz version of the Intel486™ SX processor to the high performance 100 MHz IntelDX4 processor. The IntelDX4 processor is the world's fastest 486, providing a new level of affordable computing power to the business desktop and unparalled performance to mobile computers. Intel couples superior performance with sophisticated energy-efficient SL technology to meet the requirements of the EPA's Energy Star guidelines.

REDUCED TIME TO MARKET

Intel's universal motherboard for Intel486 microprocessor-based desktop systems and scalable architecture for mobile systems reduces the development effort required to produce an entire product line of high-performance entry-level systems, thereby reducing time to market.

Intel offers a wide variety of off-the-shelf components to fulfill the requirements of system designers while simplifying the implementation of their designs. Off-the-shelf system solutions greatly decrease the potential risk of costly project delays due to component incompatibility. These system solutions greatly reduce the amount of time required to design, debug, manufacture and test microprocessor-based systems.

INCREASED RELIABILITY

High reliability is a tangible goal that translates to higher reliability for your product, reduced downtime, and reduced repair costs. As more and more functions are integrated into fewer components, the resulting system requires less power, produces less heat, and requires fewer mechanical connections—again resulting in greater system reliability.

LOWER COSTS

Using proven, reliable off-the-shelf components will reduce design costs, manufacturing costs, and time to market while increasing project viability and product reliability.

2

Intel486™ Microprocessor

INTEL486™ PROCESSOR FAMILY

- 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

*Other brands and names are the property of their respective owners.

December 1994
Order Number: 242202-001

DATA SHEET DESIGNATIONS

Intel uses various data sheet markings to designate each phase of the document as it relates to the product. The marking appears in the lower, inside corner of the data sheet. The following is the definition of these markings:

Data Sheet 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 data sheets 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 PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS PAGE

CONTENTS PAGE

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. 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 IntelDX2 processor is 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 mA–100 mA) and a "slow wake-up" Stop Clock state with CLK frequency at 0 MHz (100 μA–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 mA–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 IntelDX2 Processor only):

- *Write Back Internal Cache*—The Write-Back Enhanced IntelDX2 processor adds write-back support to the 8-Kbyte 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).

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						✓	✓	✓	✓	
2X Clock											
Intel486 SX Processor(2)	3.3V	✓	✓							✓	
	5V	✓	✓								✓
Intel486 DX 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 data sheet 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 IntelDX2 and information previously published for the introduction of the SL Enhanced Intel486 processors, this data sheet encompasses the entire current Intel486 processor family.

This data sheet describes the Intel486 processor architecture, features and technical details. Unless otherwise stated, any description for the Intel486 processor listed in this data sheet 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 data sheet does not detail the Intel486SL processor, the Low Power Intel486 SX or Low Power Intel486 DX directly. The Low Power Intel486 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 on-chip floating point 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.

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 data sheet.

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 and 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 containing 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 IntelDX2 processor. The specifics for the Write-Back Enhanced IntelDX2 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 IntelDX2 processor 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 Write-Back Enhanced IntelDX2 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 data sheet.

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 Data Sheet

This Data Sheet contains all of the latest information for the Intel486 processor family and replaces the following documentation:

SL Enhanced Intel486™ Microprocessor Data Sheet Addendum, Order No. 241696

Intel486™ SX Microprocessor Data Book, Order No. 240950

IntelSX2™ Microprocessor Data Sheet, 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 Data Sheet, Order No. 241199

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 the existing Intel486 processor products and the Intel486 processor products.

168-Pin PGA—Pin Grid Array

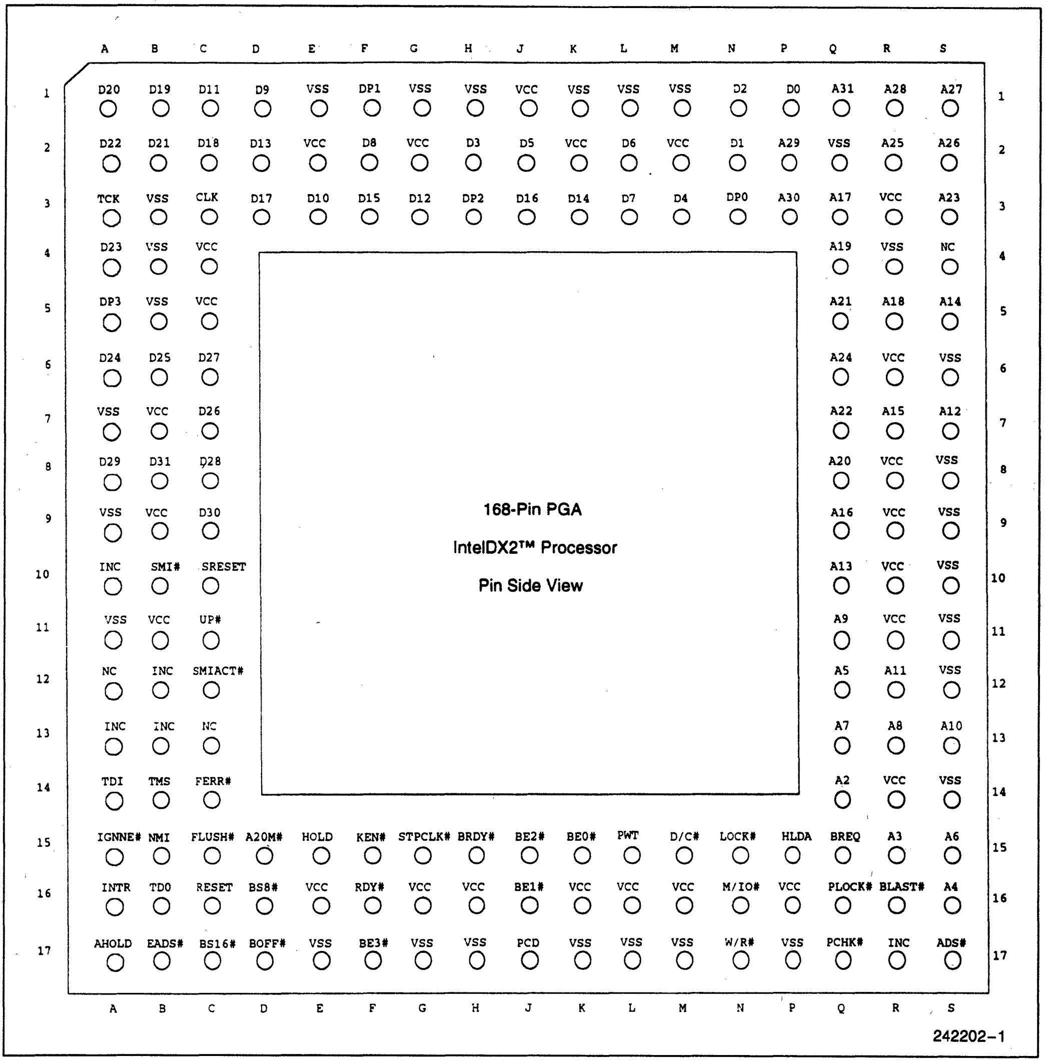
- Package Diagram
- Pin Assignment Difference Table
- Pin Cross Reference by Pin Name

208-Lead SQFP—Quad Flat Pack

- Package Diagram
- Pin Assignment Difference Table
- Pin Assignment Table in numerical order

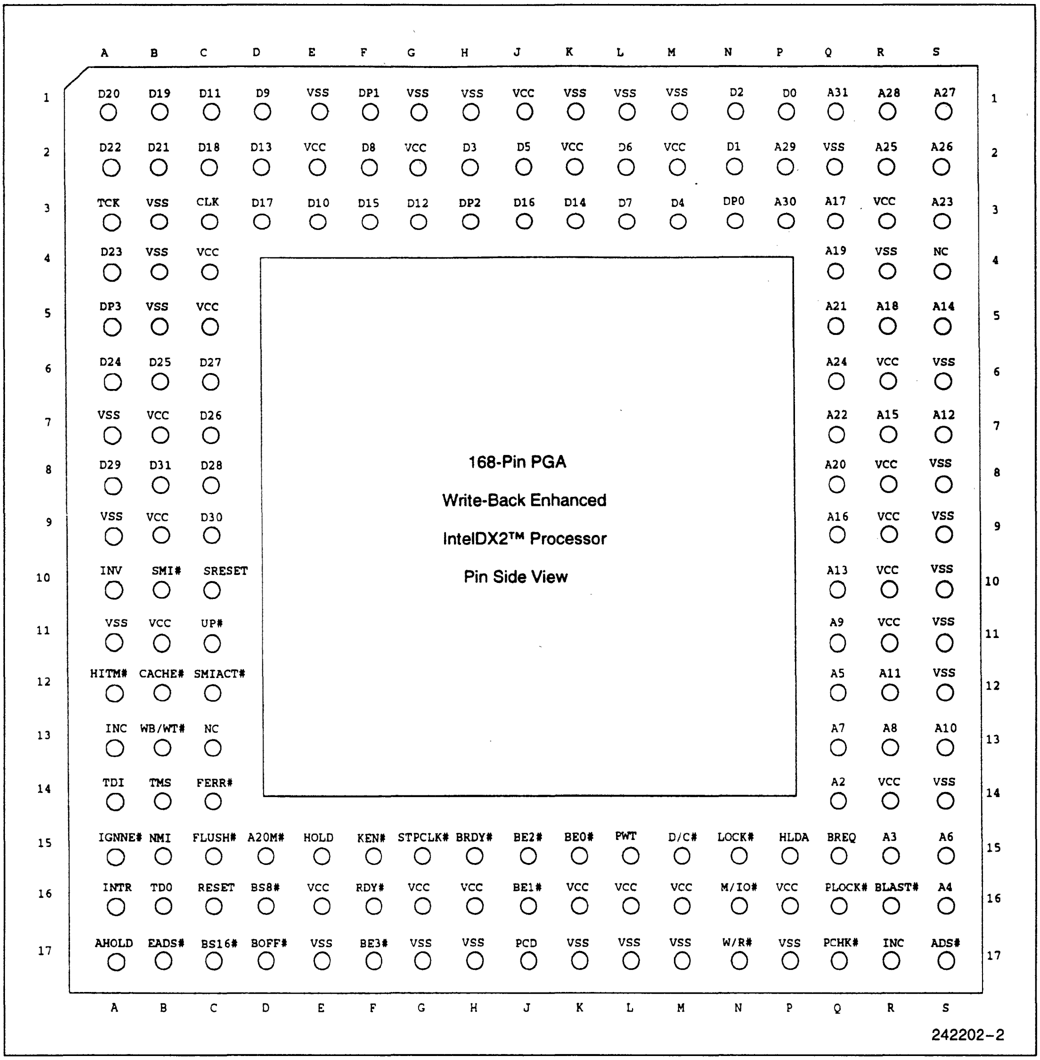
196-Lead PQFP—Plastic Quad Flat Pack

- Package Diagram
- Pin Assignment Difference Table
- Pin Assignment Table in numerical order



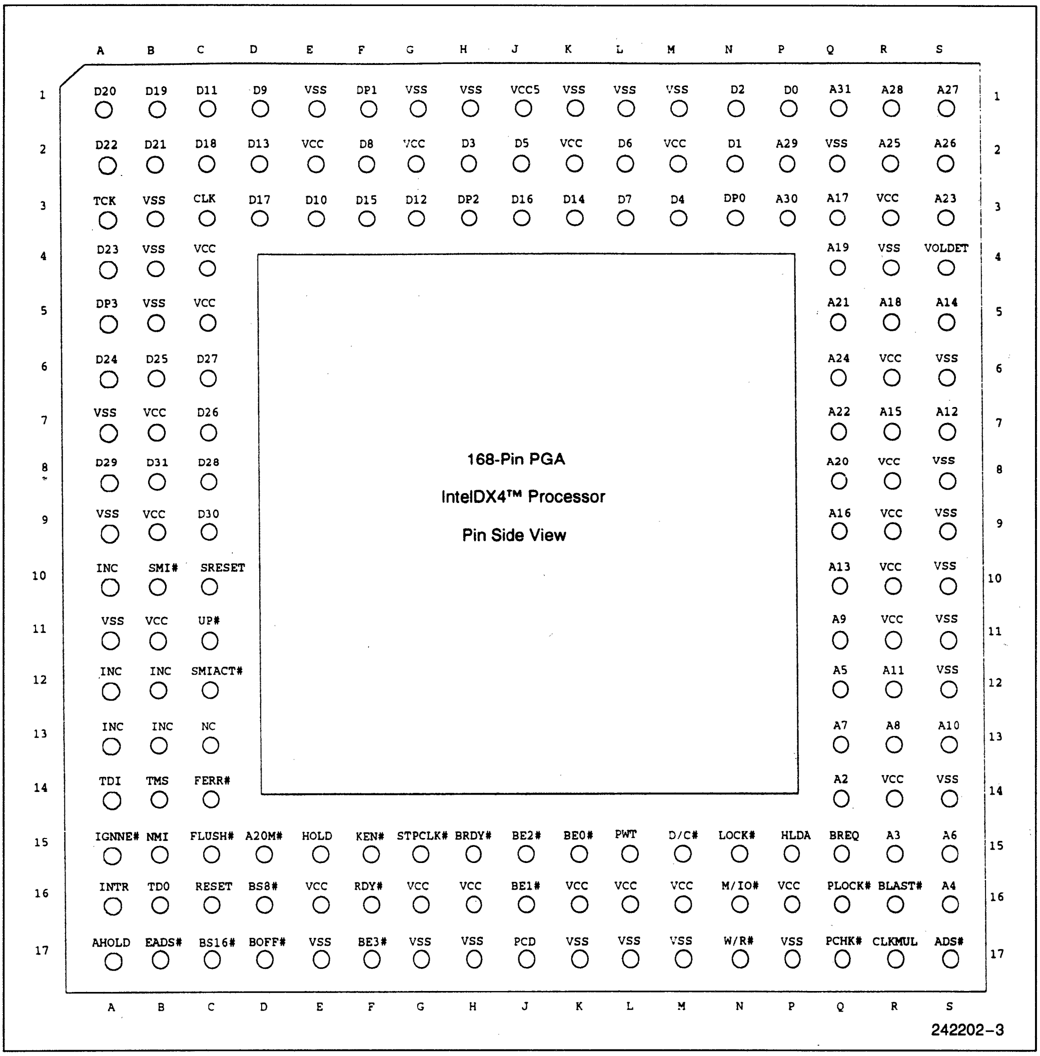
	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	
1	D20	D19	D11	D9	VSS	DP1	VSS	VSS	VCC	VSS	VSS	VSS	D2	D0	A31	A28	A27	1
2	D22	D21	D18	D13	VCC	D8	VCC	D3	D5	VCC	D6	VCC	D1	A29	VSS	A25	A26	2
3	TCK	VSS	CLK	D17	D10	D15	D12	DP2	D16	D14	D7	D4	DP0	A30	A17	VCC	A23	3
4	D23	VSS	VCC												A19	VSS	NC	4
5	DP3	VSS	VCC												A21	A18	A14	5
6	D24	D25	D27												A24	VCC	VSS	6
7	VSS	VCC	D26												A22	A15	A12	7
8	D29	D31	D28												A20	VCC	VSS	8
9	VSS	VCC	D30												A16	VCC	VSS	9
10	INC	SMI#	SRESET												A13	VCC	VSS	10
11	VSS	VCC	UP#												A9	VCC	VSS	11
12	NC	INC	SMIACT#												A5	A11	VSS	12
13	INC	INC	NC												A7	A8	A10	13
14	TDI	TMS	FERR#												A2	VCC	VSS	14
15	IGNNE#	NMI	FLUSH#	A20M#	HOLD	KEN#	STPCLK#	BRDY#	BE2#	BE0#	PWT	D/C#	LOCK#	HLDA	BREQ	A3	A6	15
16	INTR	TDO	RESET	BS8#	VCC	RDY#	VCC	VCC	BE1#	VCC	VCC	VCC	M/IO#	VCC	PLOCK#	BLAST#	A4	16
17	AHOLD	EADS#	BS16#	BOFF#	VSS	BE3#	VSS	VSS	PCD	VSS	VSS	VSS	W/R#	VSS	PCHK#	INC	ADS#	17
	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	

Figure 3-1. Package Diagram for 168-Pin PGA Package of the IntelDX2™ Processor



	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S
1	D20	D19	D11	D9	VSS	DP1	VSS	VSS	VCC	VSS	VSS	VSS	D2	D0	A31	A28	A27
2	D22	D21	D18	D13	VCC	D8	VCC	D3	D5	VCC	D6	VCC	D1	A29	VSS	A25	A26
3	TCK	VSS	CLK	D17	D10	D15	D12	DP2	D16	D14	D7	D4	DP0	A30	A17	VCC	A23
4	D23	VSS	VCC												A19	VSS	NC
5	DP3	VSS	VCC												A21	A18	A14
6	D24	D25	D27												A24	VCC	VSS
7	VSS	VCC	D26												A22	A15	A12
8	D29	D31	D28												A20	VCC	VSS
9	VSS	VCC	D30												A16	VCC	VSS
10	INV	SMI#	SRESET												A13	VCC	VSS
11	VSS	VCC	UP#												A9	VCC	VSS
12	HITM#	CACHE#	SMIACT#												A5	A11	VSS
13	INC	WB/WT#	NC												A7	A8	A10
14	TDI	TMS	FERR#												A2	VCC	VSS
15	IGNNE#	NMI	FLUSH#	A20M#	HOLD	KEN#	STPCLK#	BRDY#	BE2#	BE0#	PWT	D/C#	LOCK#	HLDA	BREQ	A3	A6
16	INTR	TDO	RESET	BS8#	VCC	RDY#	VCC	VCC	BE1#	VCC	VCC	VCC	M/IO#	VCC	PLOCK#	BLAST#	A4
17	AHOLD	EADS#	BS16#	BOFF#	VSS	BE3#	VSS	VSS	PCD	VSS	VSS	VSS	W/R#	VSS	PCHK#	INC	ADS#

Figure 3-2. Package Diagram for 168-Pin PGA Package of the Write-Back Enhanced IntelDX2™ Processor



	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	
1	D20	D19	D11	D9	VSS	DP1	VSS	VSS	VCC5	VSS	VSS	VSS	D2	D0	A31	A28	A27	1
2	D22	D21	D18	D13	VCC	D8	VCC	D3	D5	VCC	D6	VCC	D1	A29	VSS	A25	A26	2
3	TCK	VSS	CLK	D17	D10	D15	D12	DP2	D16	D14	D7	D4	DP0	A30	A17	VCC	A23	3
4	D23	VSS	VCC												A19	VSS	VOLDET	4
5	DP3	VSS	VCC												A21	A18	A14	5
6	D24	D25	D27												A24	VCC	VSS	6
7	VSS	VCC	D26												A22	A15	A12	7
8	D29	D31	D28												A20	VCC	VSS	8
9	VSS	VCC	D30												A16	VCC	VSS	9
10	INC	SMI#	SRESET												A13	VCC	VSS	10
11	VSS	VCC	UP#												A9	VCC	VSS	11
12	INC	INC	SMIACT#												A5	A11	VSS	12
13	INC	INC	NC												A7	A8	A10	13
14	TDI	TMS	FERR#												A2	VCC	VSS	14
15	IGNNE#	NMI	FLUSH#	A20M#	HOLD	KEN#	STPCLK#	BRDY#	BE2#	BE0#	PWT	D/C#	LOCK#	HLDA	BREQ	A3	A6	15
16	INTR	TDO	RESET	BS8#	VCC	RDY#	VCC	VCC	BE1#	VCC	VCC	VCC	M/IO#	VCC	PLOCK#	BLAST#	A4	16
17	AHOLD	EADS#	BS16#	BOFF#	VSS	BE3#	VSS	VSS	PCD	VSS	VSS	VSS	W/R#	VSS	PCHK#	CLKMUL	ADS#	17
	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	

Figure 3-3. 168-Pin PGA Pinout Diagram (Pin Side) for the 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
A12	INC	INC	INC	INC	INC	INC	INC HITM#(6)	INC
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
B13	INC	INC	INC	INC	INC	INC	INC WB/WT#(6)	INC
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:

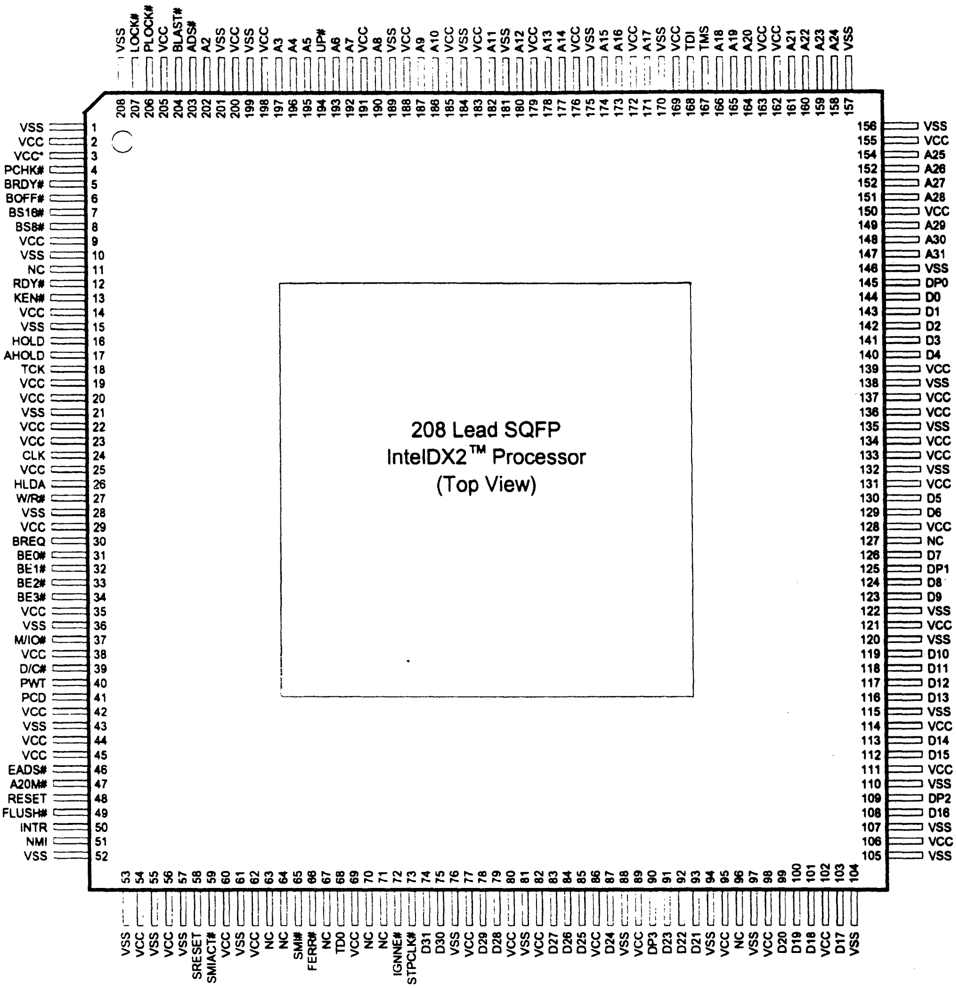
1. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Intel486 processors.
2. INC. Internal No Connect. These pins are not connected to any 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 IntelDX2 processor 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		E16	E1
A9 .....Q11	D7.......L3	BLAST# .R16	**NC (2)**	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	H17
A13 ....Q10	D11 .....C1	BS8#....D16		K2	H1
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)			

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.



242202-4

* Pin 3. See Note 1 for Table 3-3.

Figure 3-4. Package Diagram for 208-Lead SQFP of the IntelDX2™ Processor

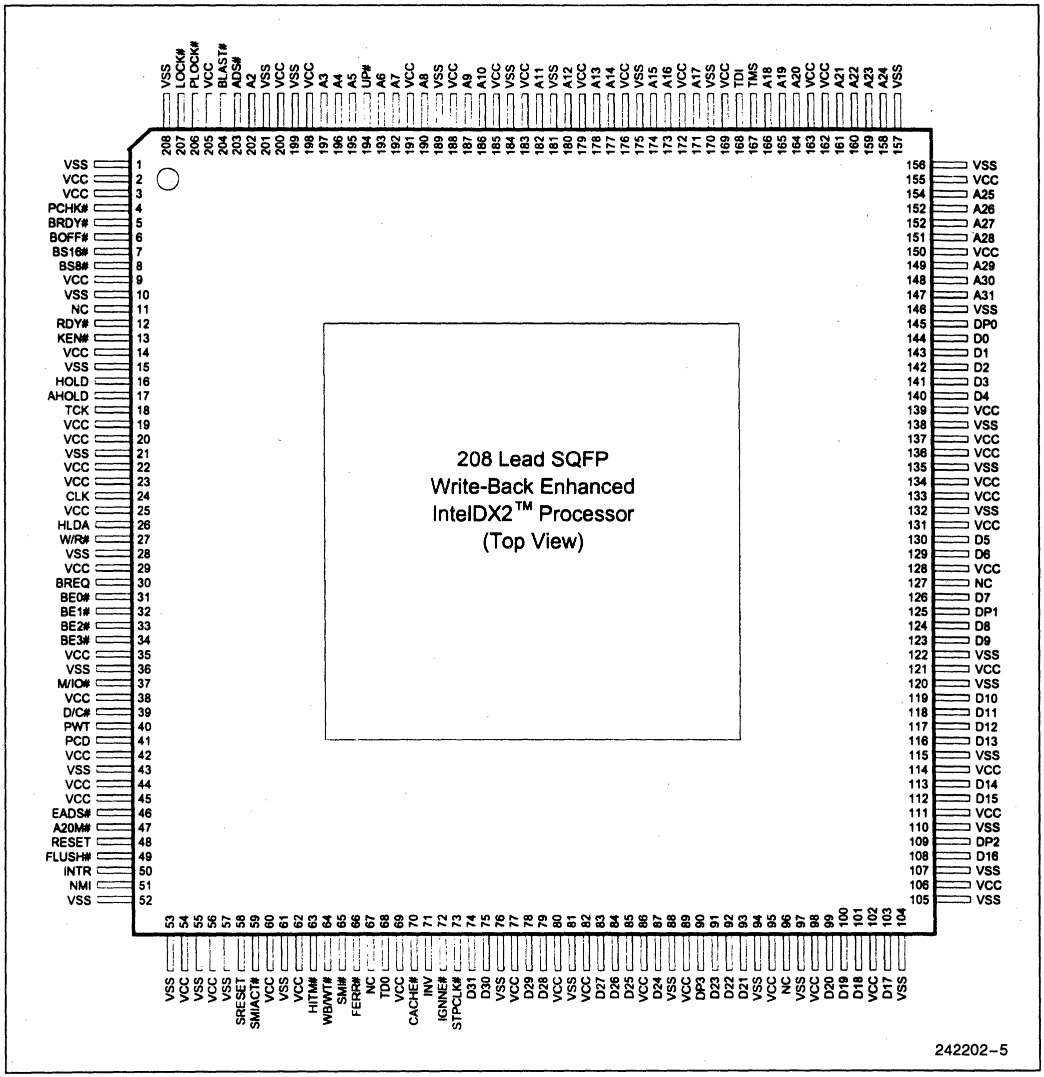


Figure 3-5. Package Diagram for 208-Lead SQFP of the Write-Back Enhanced IntelDX2™ Processor

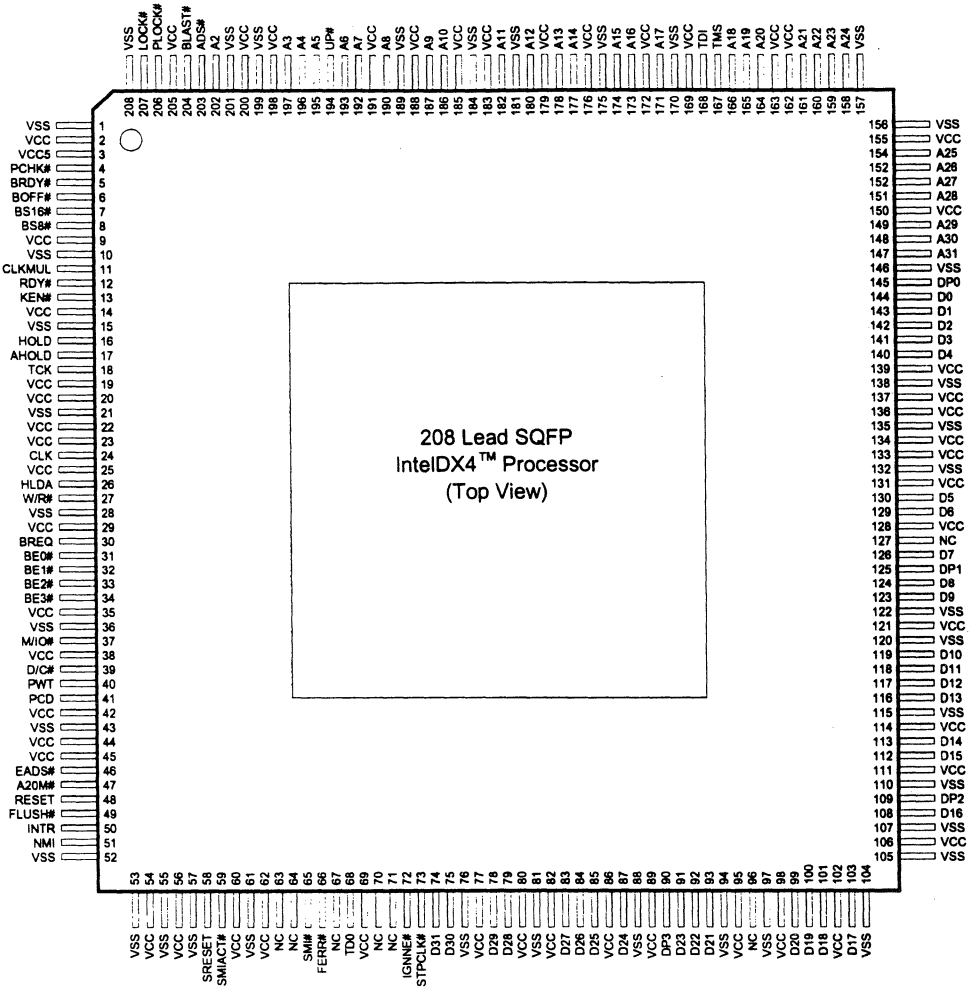


Figure 3-6. Package Diagram for 208-Lead SQFP of the IntelDX4™ Processor

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
3	V_{CC}(1)	V_{CC}	V_{CC}	V_{CC}	V_{CC5}
11	INC(2)	INC	INC	INC	CLKMUL
63	INC	INC	INC	HITM#	INC
64	INC	INC	INC	WB/WT#	INC
66	INC	FERR#	FERR#	FERR#	FERR#
70	INC	INC	INC	CACHE#	INC
71	INC	INC	INC	INV	INC
72	INC	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 (Continued)

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.

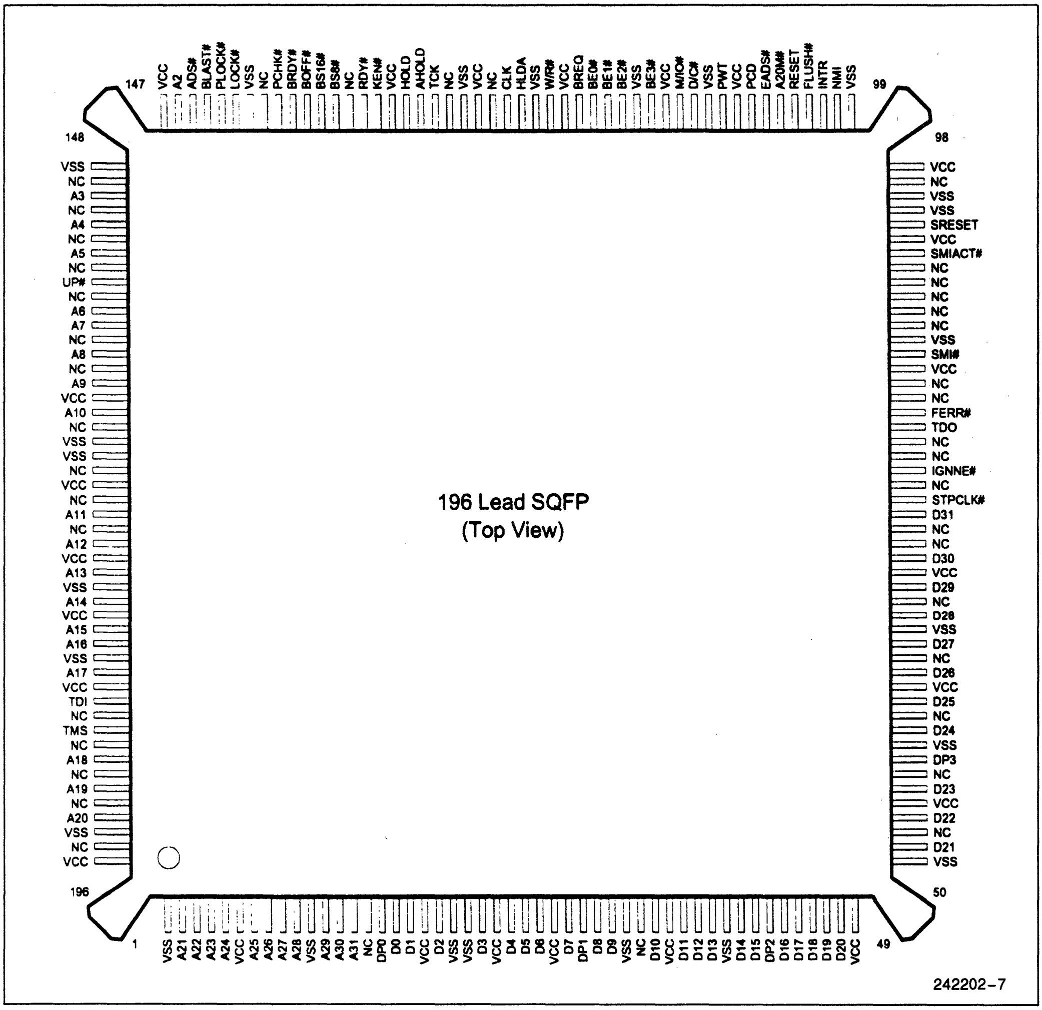


Figure 3-7. Package Diagram for 196-Lead PQFP Package of the Intel486™ DX Processor

Table 3-5. Pinout Differences for 196-Lead PQFP Package

Pin #	Previous Intel486TM 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 Intel486TM processors and OverDriveTM 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 data sheet.

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)

<table>
<tr><th>Symbol</th><th>Type</th><th>Name and Function</th></tr>
<tr><td colspan="3"><b>BUS CYCLE DEFINITION</b></td></tr>
<tr><td><b>M/IO#</b>
<b>D/C#</b>
<b>W/R#</b></td><td>O
O
O</td><td>The <b>memory/input-output, data/control</b> and <b>write/read</b> lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted.
<table>
<tr><th>M/IO#</th><th>D/C#</th><th>W/R#</th><th>Bus Cycle Initiated</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Halt/Special Cycle</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>I/O Read</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>I/O Write</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Code Read</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Memory Read</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Memory Write</td></tr>
</table>
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.</td></tr>
<tr><td><b>LOCK#</b></td><td>O</td><td>The <b>Bus Lock</b> 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.</td></tr>
<tr><td><b>PLOCK#</b></td><td>O</td><td>The <b>Pseudo-Lock</b> 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 FPU, 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 FPU), 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.</td></tr>
</table>

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 INTELDX2 PROCESSOR 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 INTELDX2 PROCESSOR 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 IntelDX2 processor 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 IntelDX2 processor 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.

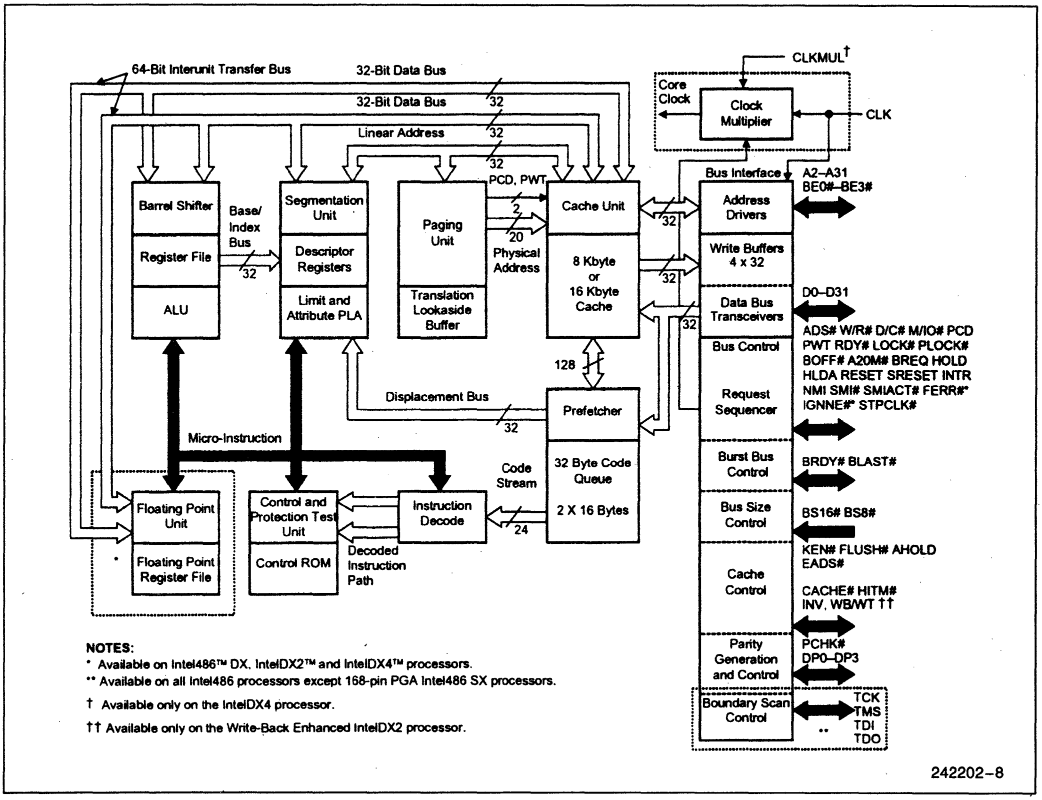
The Intel486 processor has two modes of operation: Real Address Mode (Real Mode) and Protected Mode Virtual Address Mode (Protected Mode). In Real Mode the Intel486 processor operates as a very fast 8086. Real Mode is required primarily to set up the Intel486 processor for Protected Mode operation. Protected 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 IntelDX2 processor 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 IntelDX2 processor.

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."

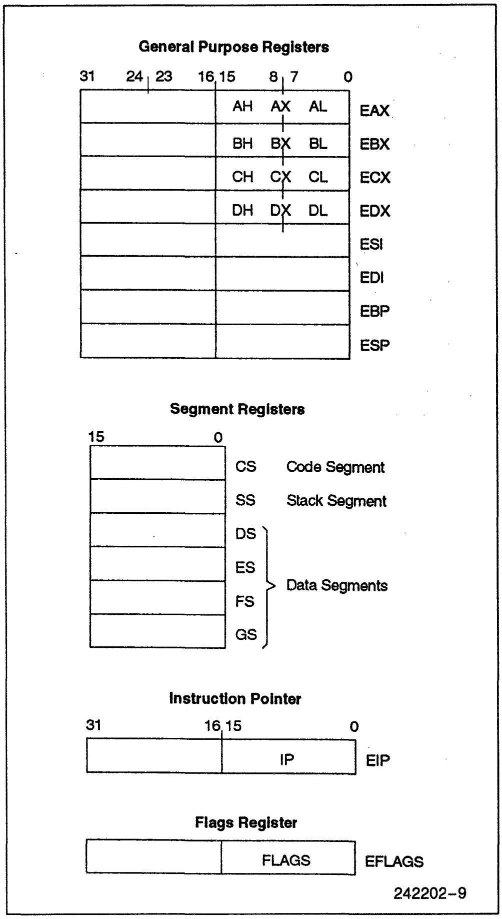


Figure 4-2. Base Architecture Registers

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.

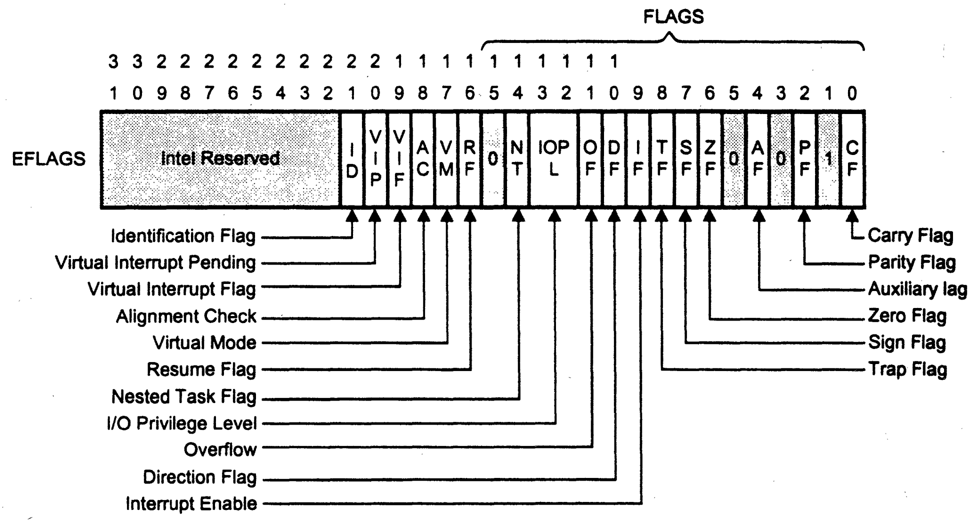
4.2.2.2 Instruction Pointer

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.



▨ indicates Intel Reserved; Do not define.

242202–10

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.

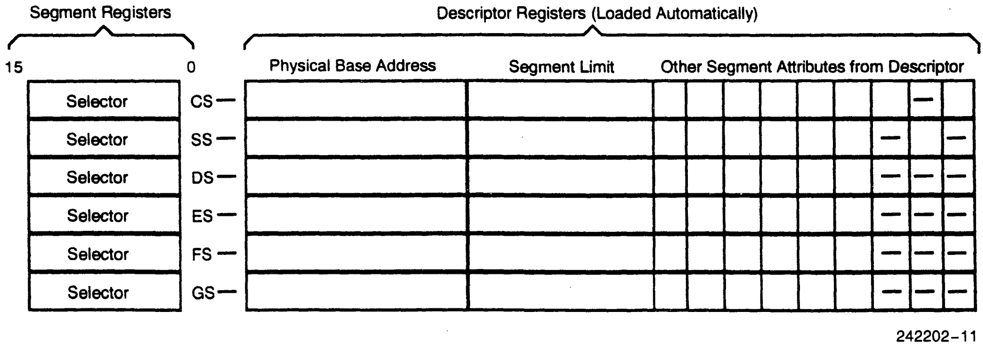


Figure 4-4. Intel486™ Processor Segment Registers and Associated Descriptor Cache Registers

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).

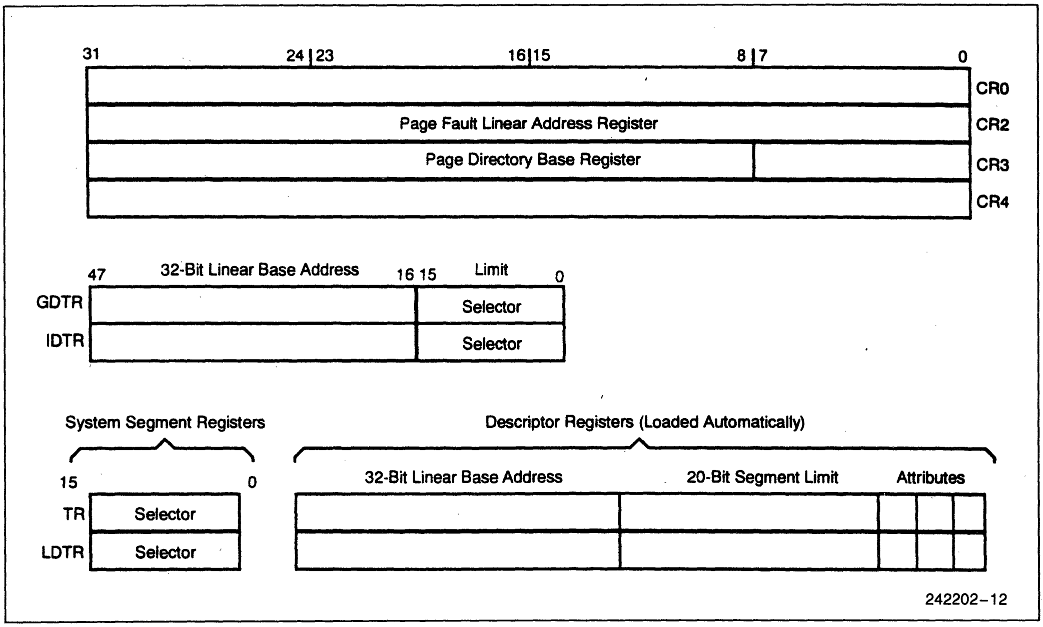
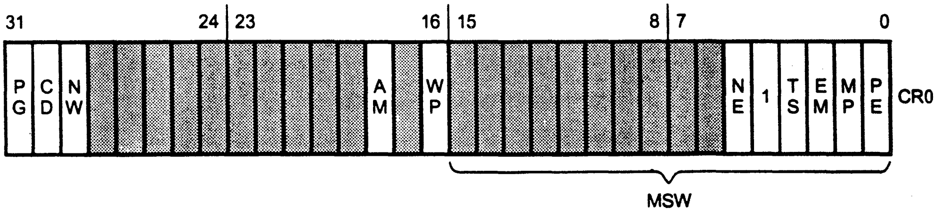

31
24
23
16
15
8
7
0
CR0
Page Fault Linear Address Register
CR2
Page Directory Base Register
CR3
CR4
47
32-Bit Linear Base Address
16 15
Limit
0
GDTR
Selector
IDTR
Selector
System Segment Registers
Descriptor Registers (Loaded Automatically)
15
0
32-Bit Linear Base Address
20-Bit Segment Limit
Attributes
TR
Selector
LDTR
Selector
242202-12

Figure 4-5. System Level Registers



▨ indicates Intel Reserved; Do not define.

242202-13

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 ignores 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 are 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

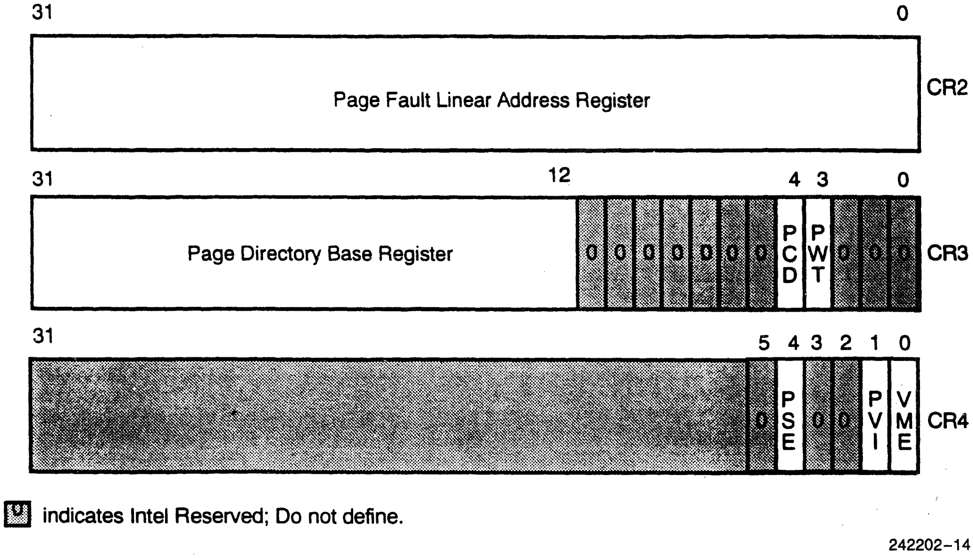
Table 4-5. Recommended Values of the Floating Point Related Bits for All Intel486™ Processors

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.


31
0
Page Fault Linear Address Register
CR2
31
12
4 3
0
Page Directory Base Register
0 0 0 0 0 0 0
PCD
PWT
0 0 0
CR3
31
5 4 3 2 1 0
0
PSE
0 0
PVI
VME
CR4
0 indicates Intel Reserved; Do not define.
242202-14

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.

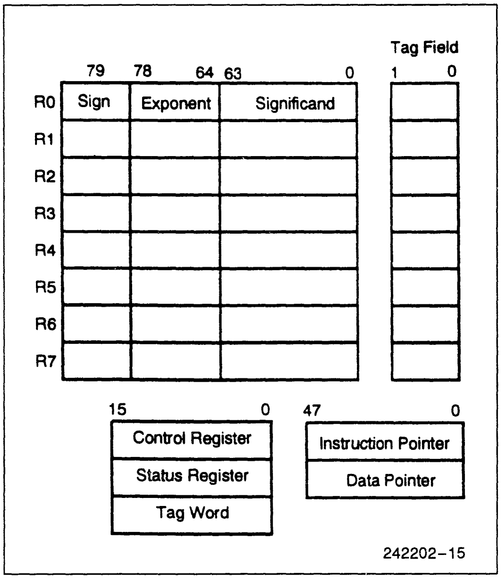


Figure 4-8. Floating Point Registers

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.

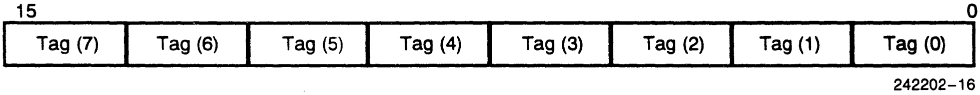
Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to use. 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

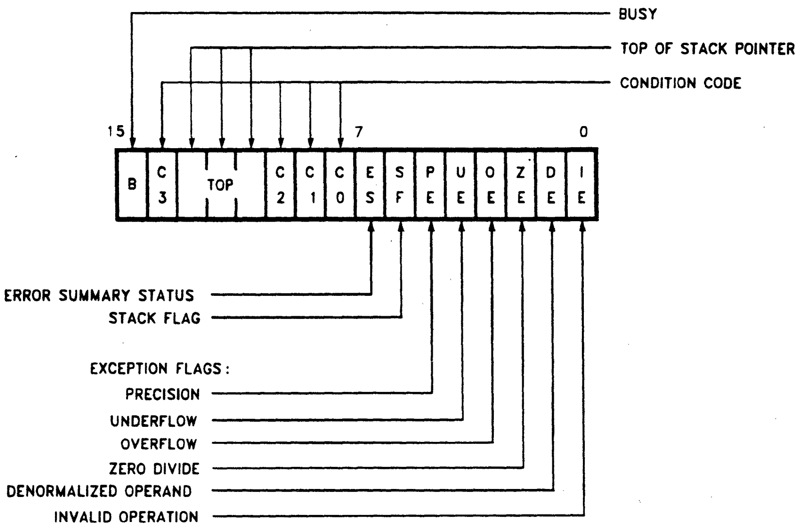
The 16-bit status word reflects the overall state of the FPU. The status word is shown in Figure 4-10 and is located in the status register.



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.) 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 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	
0	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	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

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.

32-Bit Protected Mode Format

31	23	15	7	0	
	Intel Reserved		Control Word		0
	Intel Reserved		Status Word		4
	Intel Reserved		Tag Word		8
		IP Offset			c
0000	OPCODE 10..0		CS Selector		10
		Data Operand Offset			14
	Intel Reserved		Operand Selector		18

242202–18

Figure 4-11. Protected Mode FPU Instructions and Data Pointer Image in Memory, 32-Bit Format

32-Bit Real Address Mode Format

31	23	15	7	0	
	Intel Reserved		Control Word		0
	Intel Reserved		Status Word		4
	Intel Reserved		Tag Word		8
	Intel Reserved		Instruction Pointer 15..0		c
0000	Instruction Pointer 31..16	0	OPCODE 10..0		10
	Intel Reserved		Operand Pointer 15..0		14
0000	Operand 31..16	0000	00000000		18

242202–19

Figure 4-12. Real Mode FPU Instruction and Data Pointer Image in Memory, 32-Bit Format

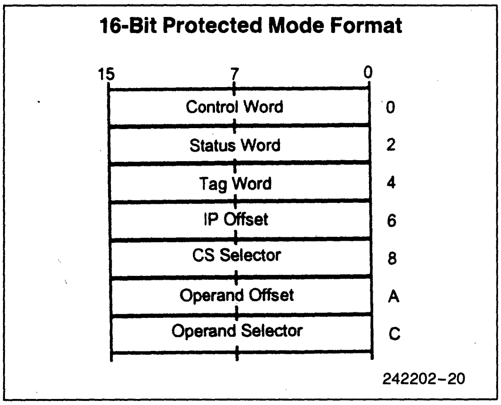


Figure 4-13. Protected Mode FPU Instruction and Data Pointer Image in Memory, 16-Bit Format

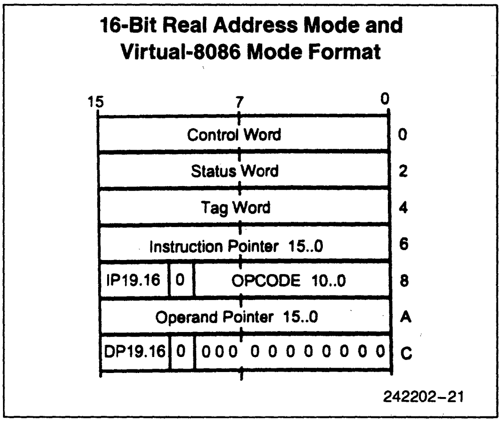
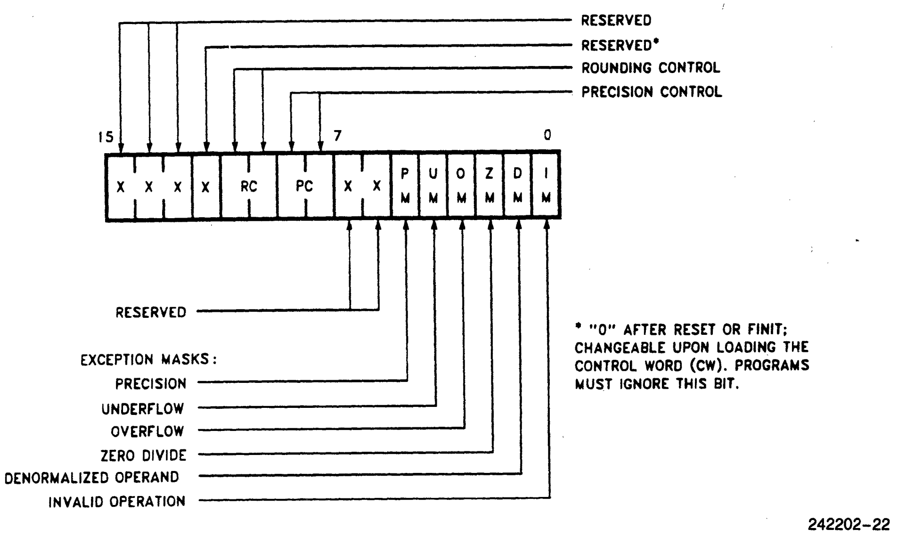


Figure 4-14. Real Mode FPU Instruction and Data Pointer Image in Memory, 16-Bit Format



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."

Debug Registers

Linear Breakpoint Address 0	DR0
Linear Breakpoint Address 1	DR1
Linear Breakpoint Address 2	DR2
Linear Breakpoint Address 3	DR3
Intel Reserved, Do Not Define	DR4
Intel Reserved, Do Not Define	DR5
Breakpoint Status	DR6
Breakpoint Control	DR7

Test Registers

Cache Test Data	TR3
Cache Test Status	TR4
Cache Test Control	TR5
TLB Test Control	TR6
TLB Test Status	TR7

TLB = Translation Lookaside Buffer

242202–23

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.

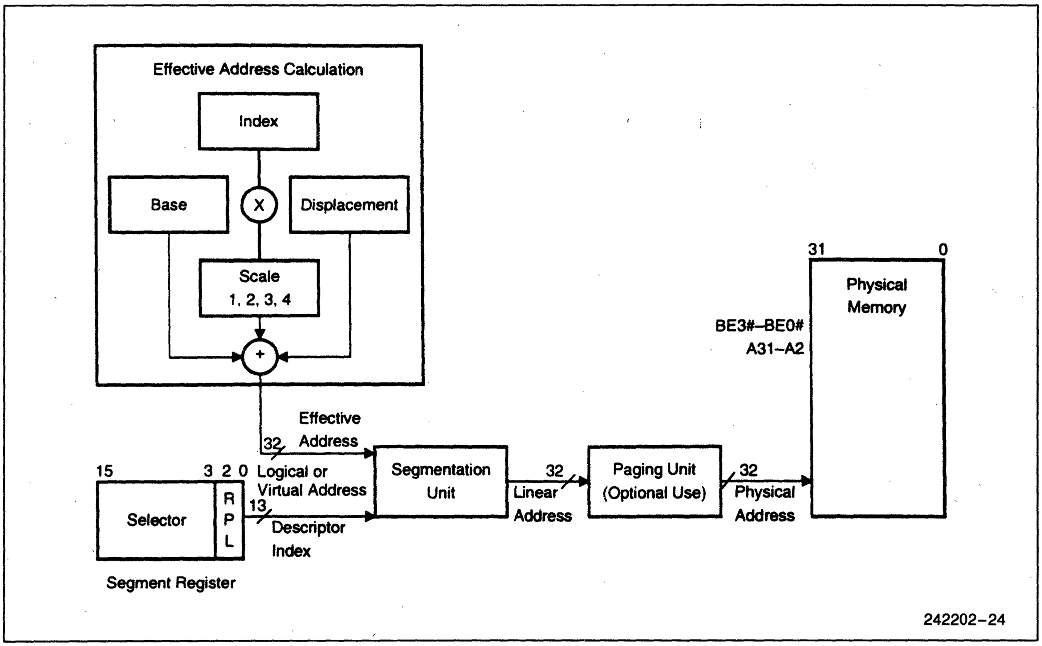


Figure 4-17. Address Translation

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:		All
[EAX]	DS	
[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 4 components make up the 9 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 is added to a DISPLACEMENT to form the operand's offset.

Example: MOV ECX, [EAX+24]

Index Mode: An INDEX register's contents is added to a DISPLACEMENT to form the operand's 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 operand's 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 operand's offset.

Example: MOV ECX, [EDX*8] [EAX]

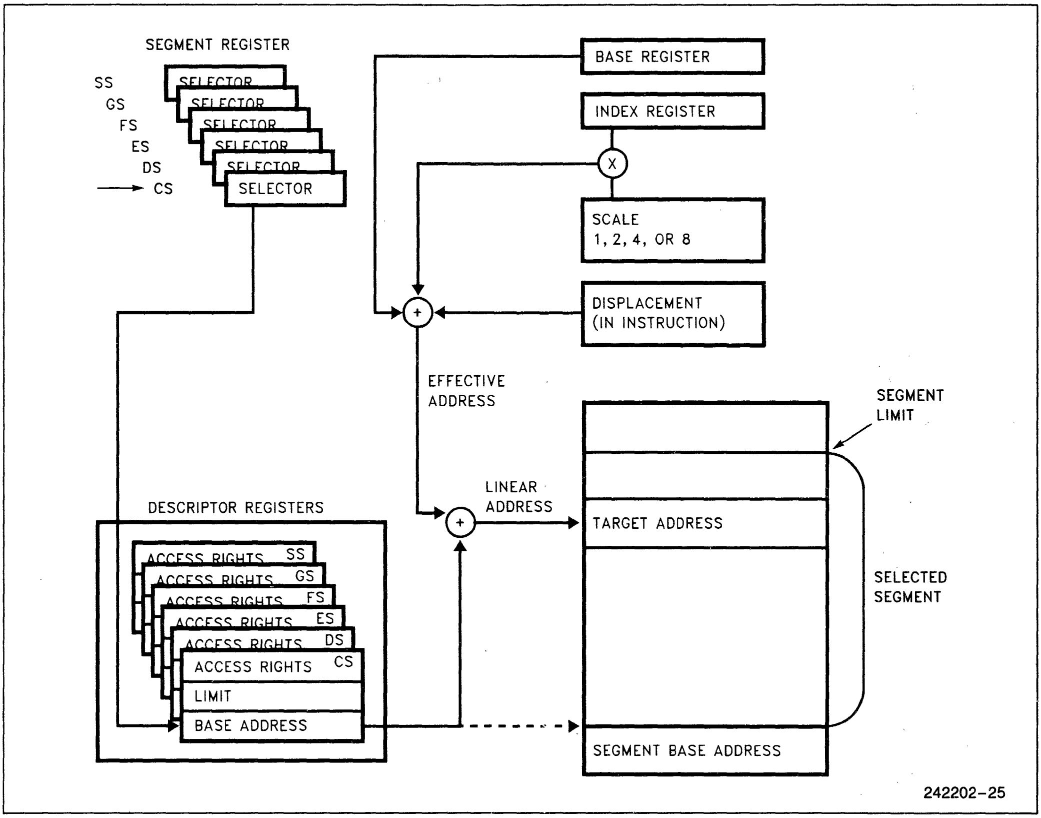


Figure 4-18. Addressing Mode Calculations

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.

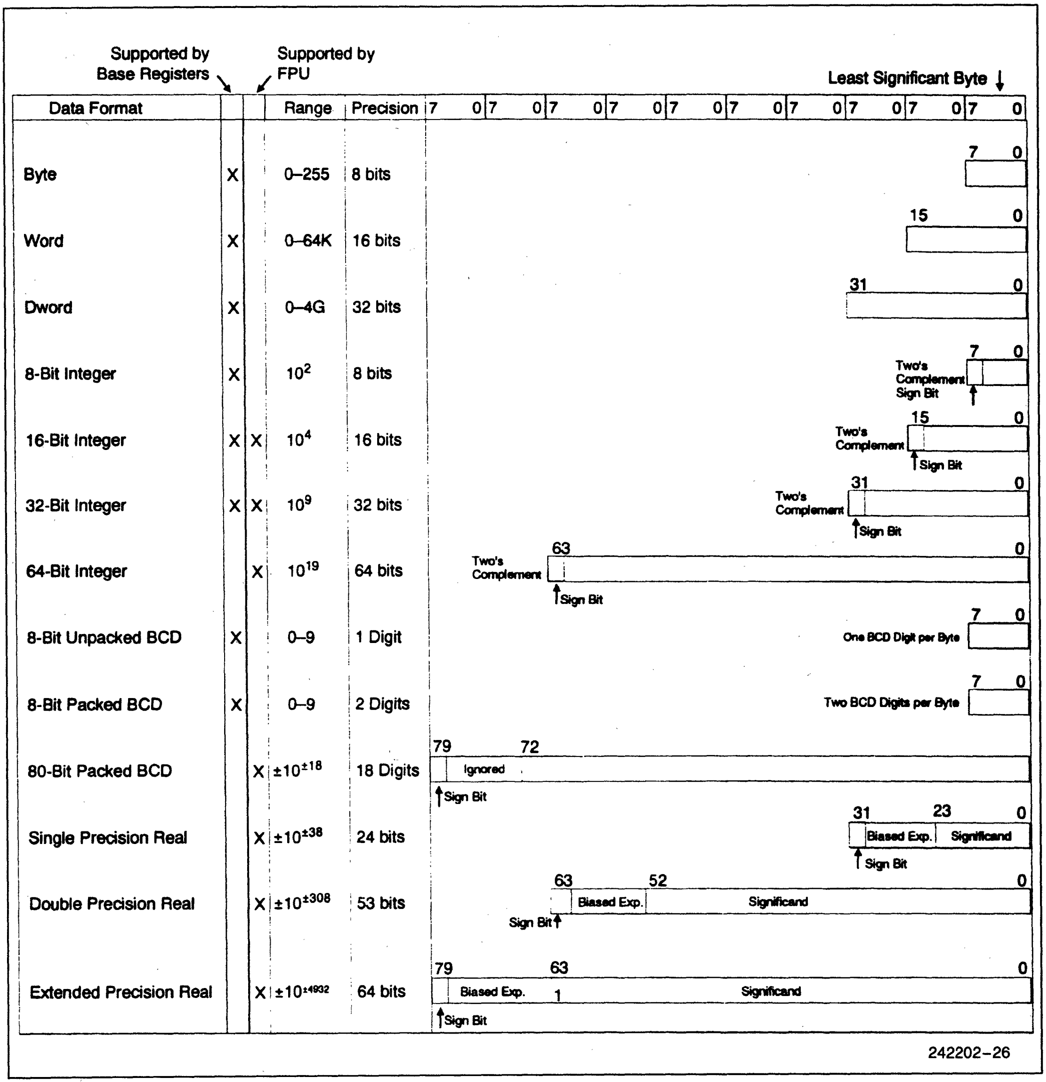
Word String: Contiguous sequence of words.

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.)



Data Format	Supported by Base Registers	Supported by FPU	Range	Precision
Byte	X		0–255	8 bits
Word	X		0–64K	16 bits
Dword	X		0–4G	32 bits
8-Bit Integer	X		10^2	8 bits
16-Bit Integer	X	X	10^4	16 bits
32-Bit Integer	X	X	10^9	32 bits
64-Bit Integer		X	10^{19}	64 bits
8-Bit Unpacked BCD	X		0–9	1 Digit
8-Bit Packed BCD	X		0–9	2 Digits
80-Bit Packed BCD		X	$\pm10^{\pm18}$	18 Digits
Single Precision Real		X	$\pm10^{\pm38}$	24 bits
Double Precision Real		X	$\pm10^{\pm308}$	53 bits
Extended Precision Real		X	$\pm10^{\pm4932}$	64 bits

242202-26

Figure 4-19. Intel486™ Processor Data Types

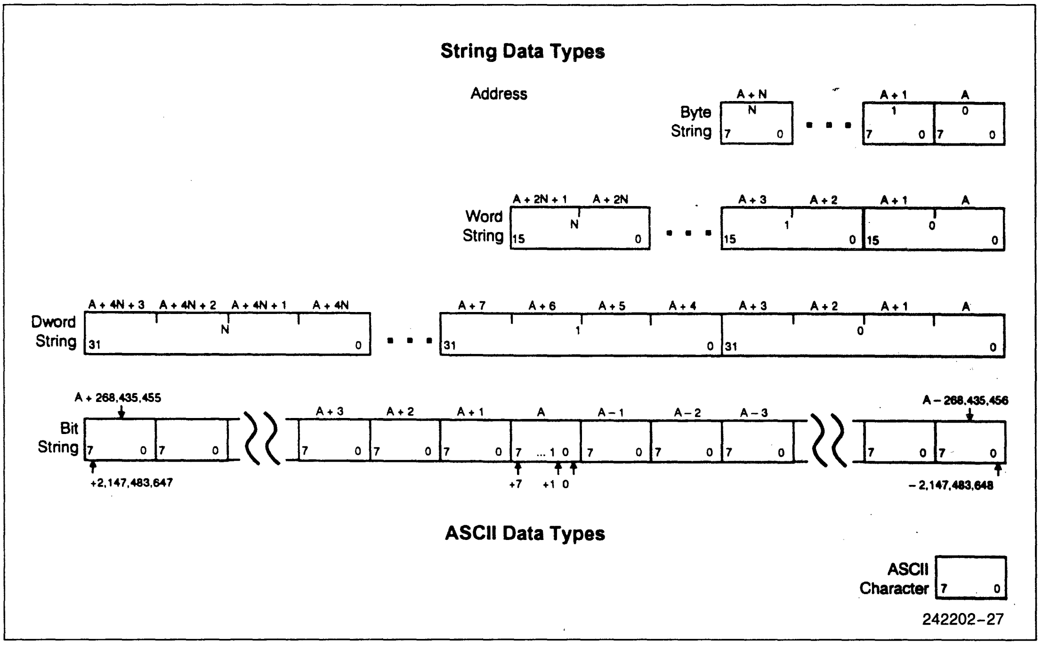


Figure 4-20. String and ASCII Data Types

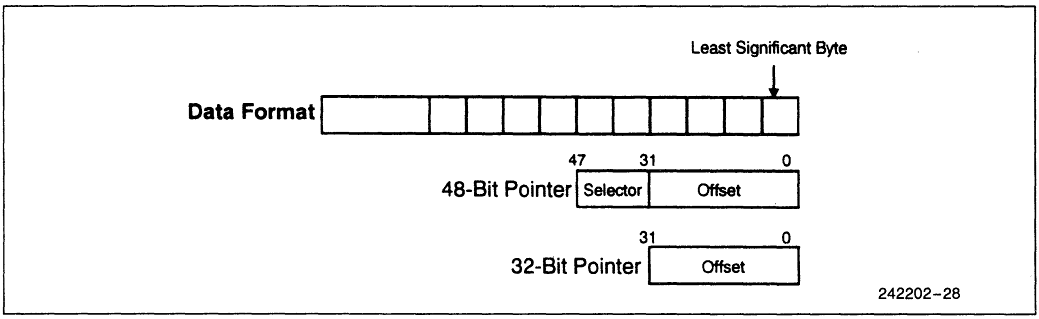


Figure 4-21. Pointer Data Types

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

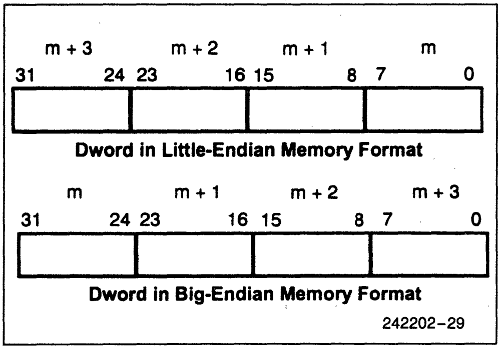


Figure 4-22. Big vs. Little Endian Memory Format

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 IntelDX2 Processor

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 IntelDX2 processor, 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 IntelDX2 Processor

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".)

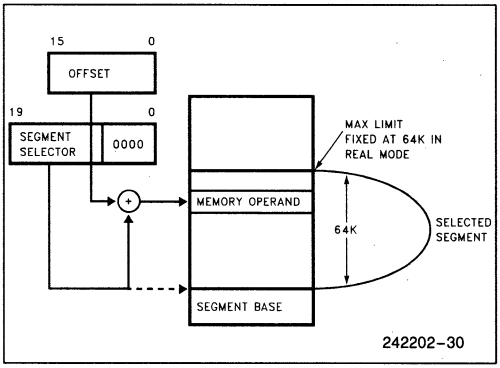


Figure 5-1. Real Address Mode Addressing

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 physical 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.

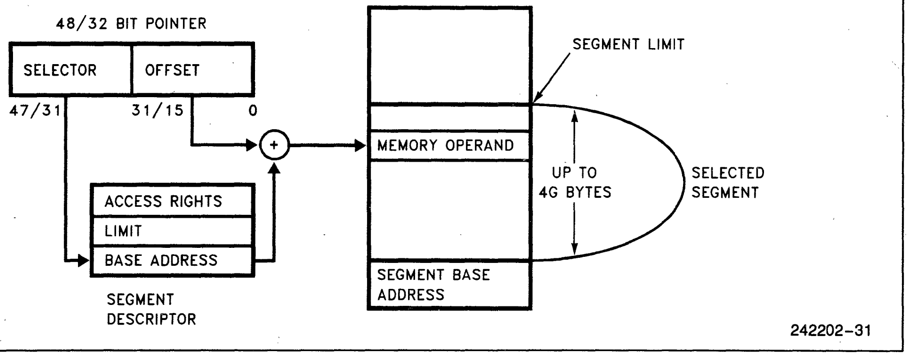


Figure 6-1. Protected Mode Addressing

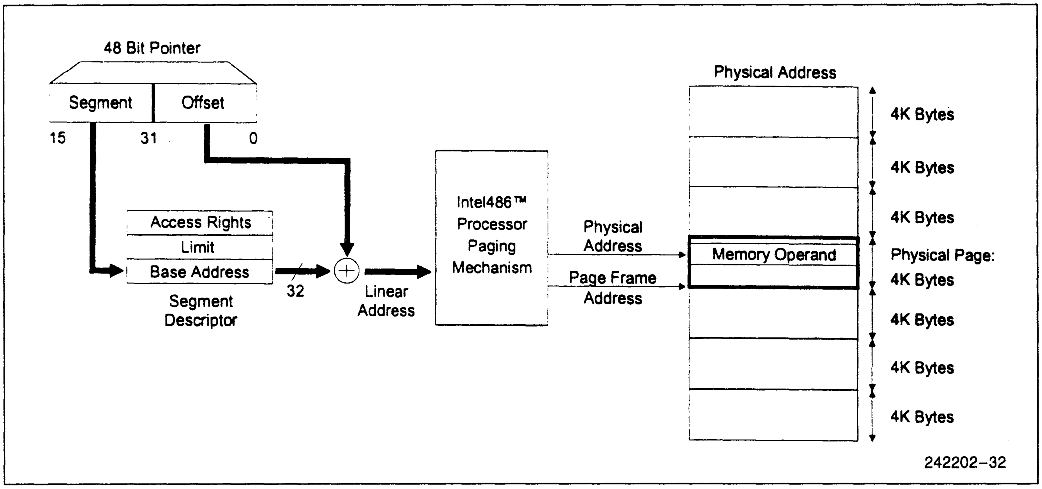


Figure 6-2. Paging and Segmentation

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.

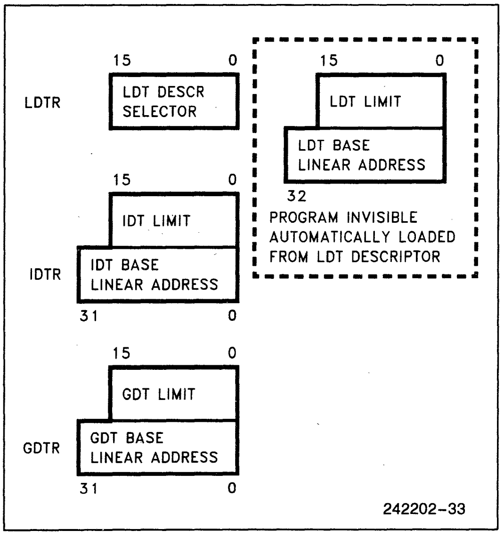


Figure 6-3. Descriptor Table Registers

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 Global Descriptor Table corresponds to the null selector and is not used. The null selector defines a null pointer value.

6.2.3.3 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.

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.")

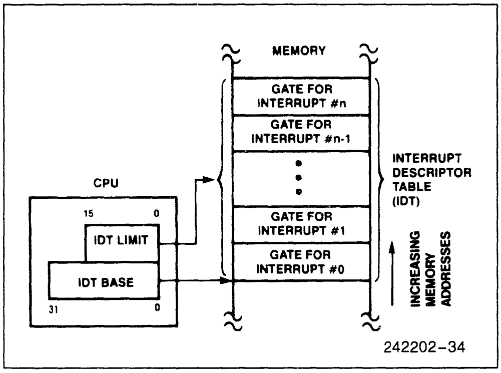


Figure 6-4. Interrupt Descriptor Table Register Use

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.

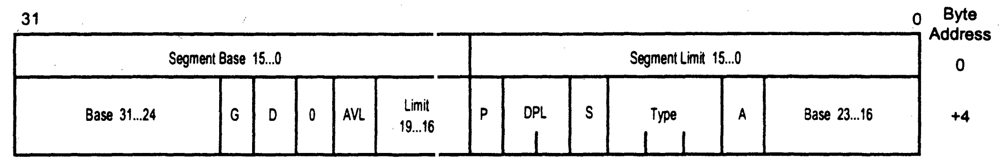
The executable **E** bit tells if a segment is a code or data segment. A code segment (E=1, S=1) may be execute-only or execute/read as determined by the Read **R** bit. Code segments are execute only if R=0, and execute/read if R=1. Code segments may never be written into.

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.")



242202-35

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.

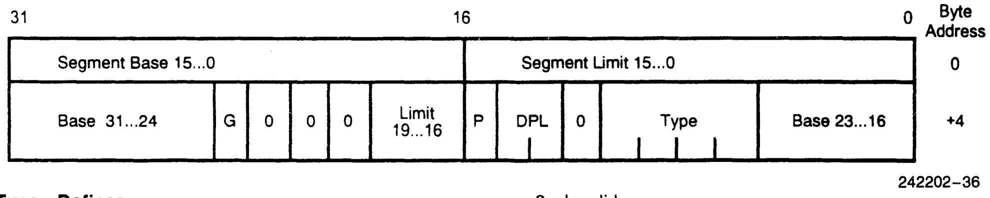
The **B** bit controls the size of the stack pointer register. If B = 1, then PUSHes, POPs, and CALLs all use the 32-bit ESP register for stack references and assume an upper limit of FFFFFFFFH. If B = 0, stack instructions all use the 16-bit SP register and assume an upper limit of FFFFH.

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. 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.

6.2.4.4 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. 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 Global Descriptor Table (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

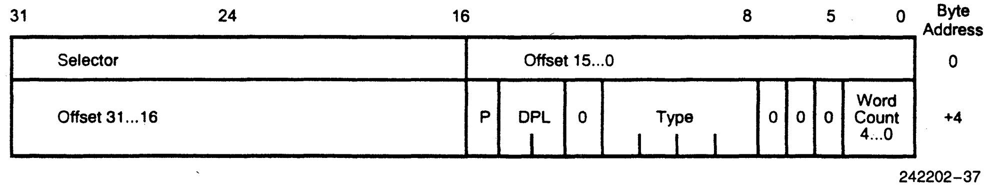
6.2.4.5 TSS Descriptors (S=0, TYPE=1, 3, 9, B)

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 16-bit Selector to the target code segment
SELECTOR selector or
Selector to the target task state segment for task gate
DESTINATION offset Entry point within the target code segment
OFFSET 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.

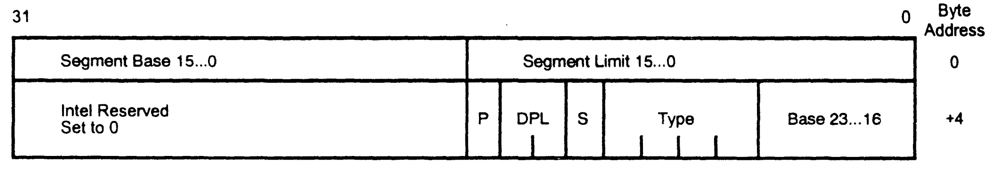
The access byte format is the same for all gate descriptors. P=1 indicates that the gate contents are valid. P=0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the descriptor privilege level and specifies when this descriptor may be used by a task. (See section 6.3, "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.



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 System Descriptor: 0=System, 1=User
TYPE Type of Segment

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.

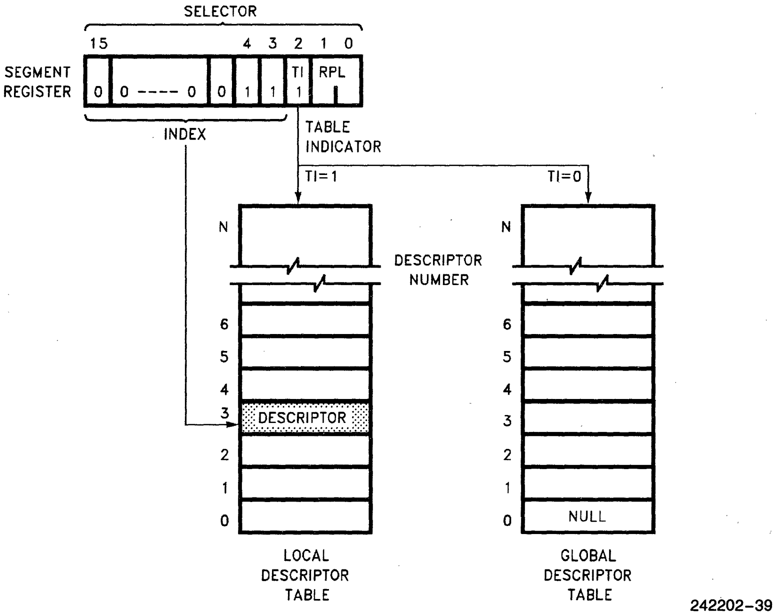
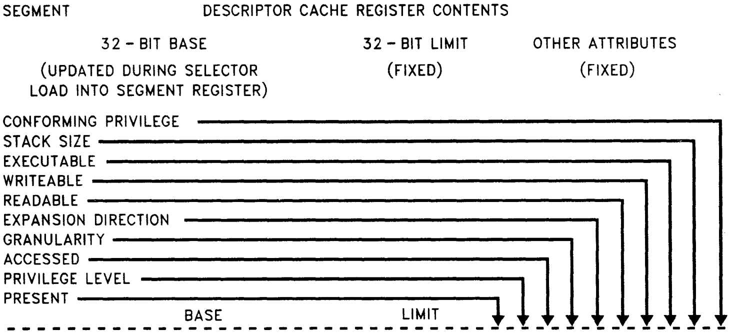


Figure 6-9. Example Descriptor Selection



SEGMENT	BASE	LIMIT	PRESENT	PRIVILEGE LEVEL	ACCESSED	GRANULARITY	EXPANSION DIRECTION	READABLE	WRITEABLE	EXECUTABLE	STACK SIZE	CONFORMING PRIVILEGE
CS	16X CURRENT CS SELECTOR*	0000FFFFH	Y	0	Y	B	U	Y	Y	Y	–	N
SS	16X CURRENT SS SELECTOR	0000FFFFH	Y	0	Y	B	U	Y	Y	N	W	–
DS	16X CURRENT DS SELECTOR	0000FFFFH	Y	0	Y	B	U	Y	Y	N	–	–
ES	16X CURRENT ES SELECTOR	0000FFFFH	Y	0	Y	B	U	Y	Y	N	–	–
FS	16X CURRENT FS SELECTOR	0000FFFFH	Y	0	Y	B	U	Y	Y	N	–	–
GS	16X CURRENT GS SELECTOR	0000FFFFH	Y	0	Y	B	U	Y	Y	N	–	–

242202-40

*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.)

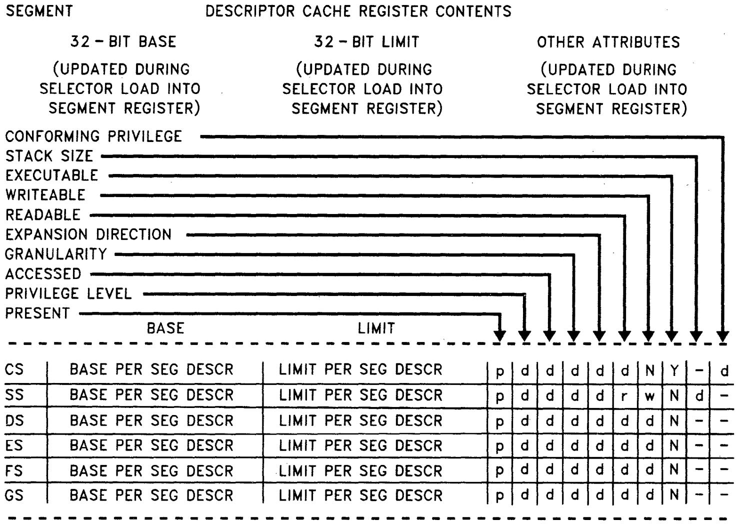
Key:

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.


SEGMENT
DESCRIPTOR CACHE REGISTER CONTENTS
32 - BIT BASE
(UPDATED DURING SELECTOR LOAD INTO SEGMENT REGISTER)
32 - BIT LIMIT
(UPDATED DURING SELECTOR LOAD INTO SEGMENT REGISTER)
OTHER ATTRIBUTES
(UPDATED DURING SELECTOR LOAD INTO SEGMENT REGISTER)
CONFORMING PRIVILEGE
STACK SIZE
EXECUTABLE
WRITEABLE
READABLE
EXPANSION DIRECTION
GRANULARITY
ACCESSED
PRIVILEGE LEVEL
PRESENT
BASE
LIMIT

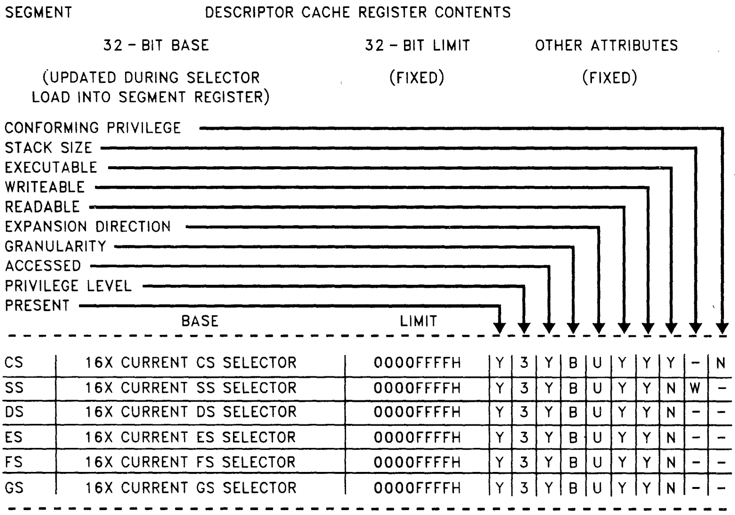
SEGMENT	BASE	LIMIT	PRESENT	PRIVILEGE LEVEL	ACCESSED	GRANULARITY	EXPANSION DIRECTION	READABLE	WRITEABLE	EXECUTABLE	STACK SIZE	CONFORMING PRIVILEGE
CS	BASE PER SEG DESCR	LIMIT PER SEG DESCR	p	d	d	d	d	d	N	Y	-	d
SS	BASE PER SEG DESCR	LIMIT PER SEG DESCR	p	d	d	d	d	r	w	N	d	-
DS	BASE PER SEG DESCR	LIMIT PER SEG DESCR	p	d	d	d	d	d	d	N	-	-
ES	BASE PER SEG DESCR	LIMIT PER SEG DESCR	p	d	d	d	d	d	d	N	-	-
FS	BASE PER SEG DESCR	LIMIT PER SEG DESCR	p	d	d	d	d	d	d	N	-	-
GS	BASE PER SEG DESCR	LIMIT PER SEG DESCR	p	d	d	d	d	d	d	N	-	-

242202–41

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)



SEGMENT	BASE	LIMIT	PRESENT	PRIVILEGE LEVEL	ACCESSED	GRANULARITY	EXPANSION DIRECTION	READABLE	WRITEABLE	EXECUTABLE	STACK SIZE	CONFORMING PRIVILEGE
CS	16X CURRENT CS SELECTOR	0000FFFFH	Y	3	Y	B	U	Y	Y	Y	-	N
SS	16X CURRENT SS SELECTOR	0000FFFFH	Y	3	Y	B	U	Y	Y	N	W	-
DS	16X CURRENT DS SELECTOR	0000FFFFH	Y	3	Y	B	U	Y	Y	N	-	-
ES	16X CURRENT ES SELECTOR	0000FFFFH	Y	3	Y	B	U	Y	Y	N	-	-
FS	16X CURRENT FS SELECTOR	0000FFFFH	Y	3	Y	B	U	Y	Y	N	-	-
GS	16X CURRENT GS SELECTOR	0000FFFFH	Y	3	Y	B	U	Y	Y	N	-	-

242202-42

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 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 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.

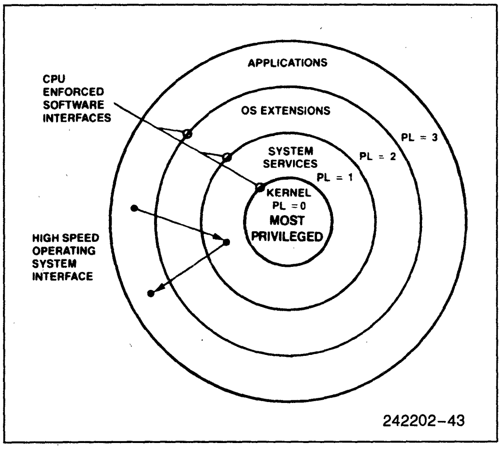


Figure 6-13. Four-Level Hierarchical Protection

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.

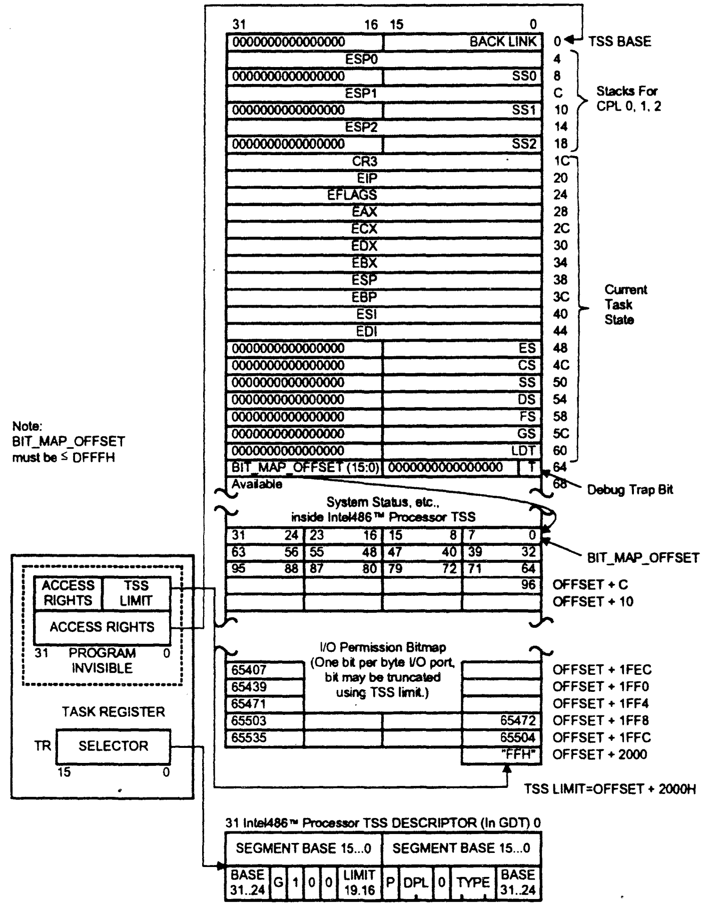
6.3.3.2 Selector Privilege (RPL)

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."

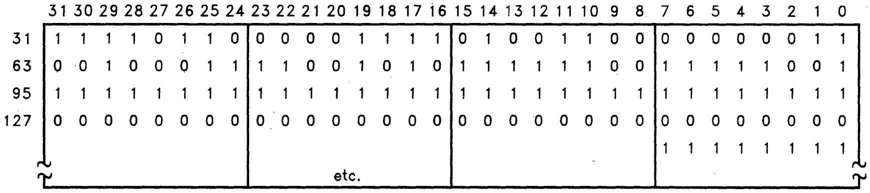
The I/O privilege level (IOPL) also affects whether several other instructions can be executed or cause an exception 13 fault instead. These instructions are called "IOPL-sensitive" instructions and they are CLI and STI. (Note that the LOCK prefix is *not* IOPL-sensitive on the Intel486 processor.)



Type=9: Available Intel486™ Processor TSS.
Type=B: Busy Intel486™ Processor TSS.

242202-44

Figure 6-14. Intel486™ Processor TSS and TSS Registers



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	1	1	1	1	0	1	1	0	0	0	0	0	1	1	1	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	1	1
63	0	0	1	0	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1
95	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																									1	1	1	1	1	1	1	1

etc.

242202-45

I/O Ports Accessible: 2 → 9, 12, 13, 15, 20 → 24, 27, 33, 34, 40, 41, 48, 50, 52, 53, 58 → 60, 62, 63, 96 → 127

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 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 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

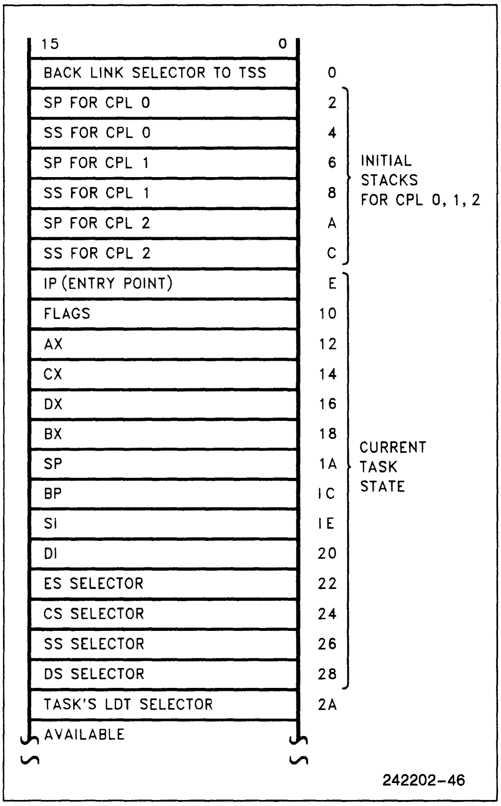
The procedure is identical for 80286 Call gates, except that 16-bit parameters are copied and 16-bit registers are pushed.

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 intersegment 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.



15 0		
BACK LINK SELECTOR TO TSS	0	
SP FOR CPL 0	2	INITIAL STACKS FOR CPL 0, 1, 2
SS FOR CPL 0	4	
SP FOR CPL 1	6	
SS FOR CPL 1	8	
SP FOR CPL 2	A	
SS FOR CPL 2	C	
IP (ENTRY POINT)	E	CURRENT TASK STATE
FLAGS	10	
AX	12	
CX	14	
DX	16	
BX	18	
SP	1A	
BP	1C	
SI	1E	
DI	20	
ES SELECTOR	22	
CS SELECTOR	24	
SS SELECTOR	26	
DS SELECTOR	28	
TASK'S LDT SELECTOR	2A	
AVAILABLE		

242202–46

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 causes 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 the descriptors needed for a simple Protected Mode Intel486 processor system. It has a single code and single data/stack segment each four-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.

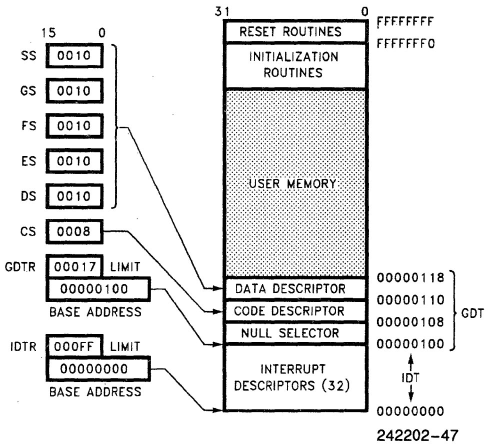


Figure 6-17. Simple Protected System

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 Task State Segment Register 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.

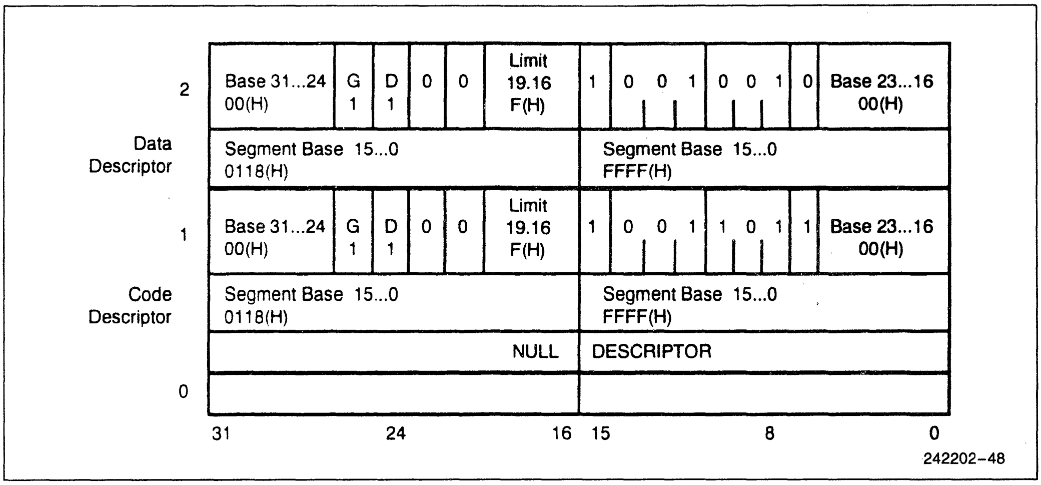


Figure 6-18. GDT Descriptors for Simple System

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.

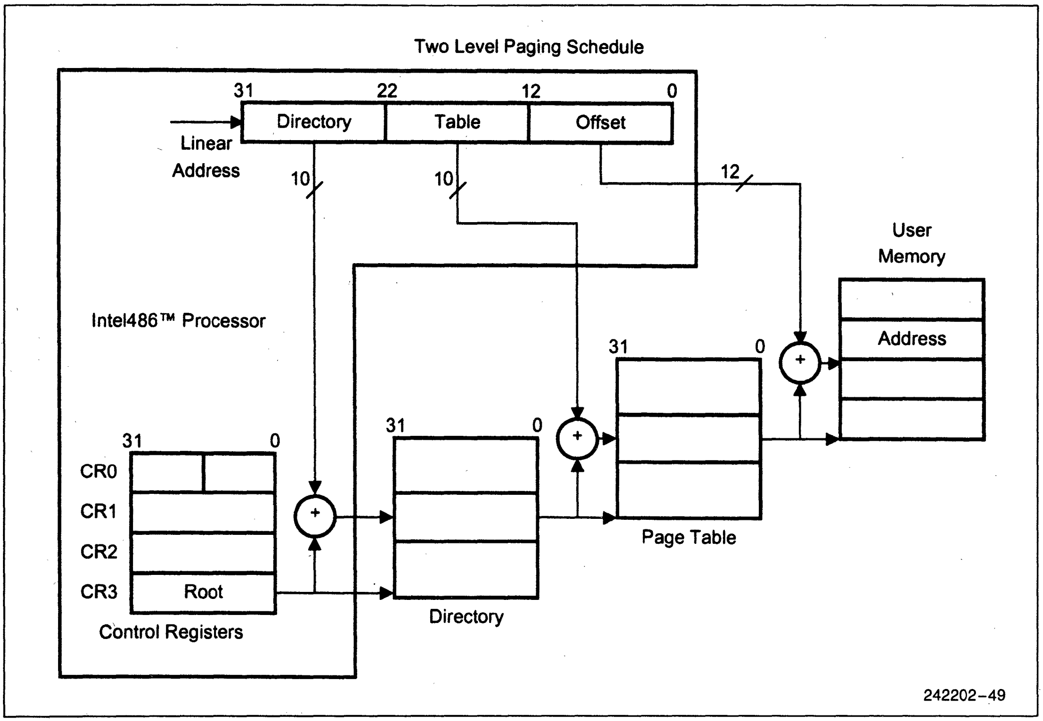


Figure 6-19. Paging Mechanism

31 … 12	11 10 9	8	7	6	5	4	3	2	1	0
PAGE TABLE ADDRESS 31 . . . 12	OS RESERVED	0	0	D	A	P C D	P W T	U — S	R — W	P

Figure 6-20. 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	P C D	P W T	U — S	R — W	P

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 affect 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.

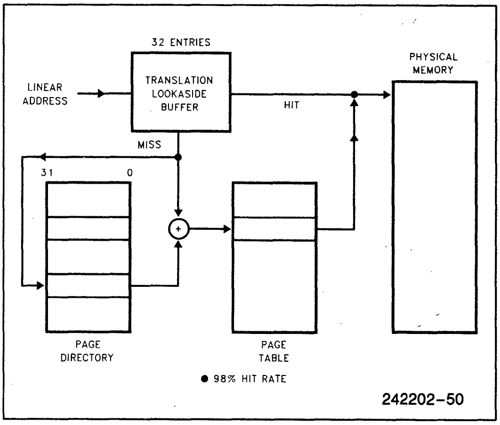


Figure 6-22. Translation Lookaside Buffer

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 are 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.

15													3	2	1	0
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U S	W R	P

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.

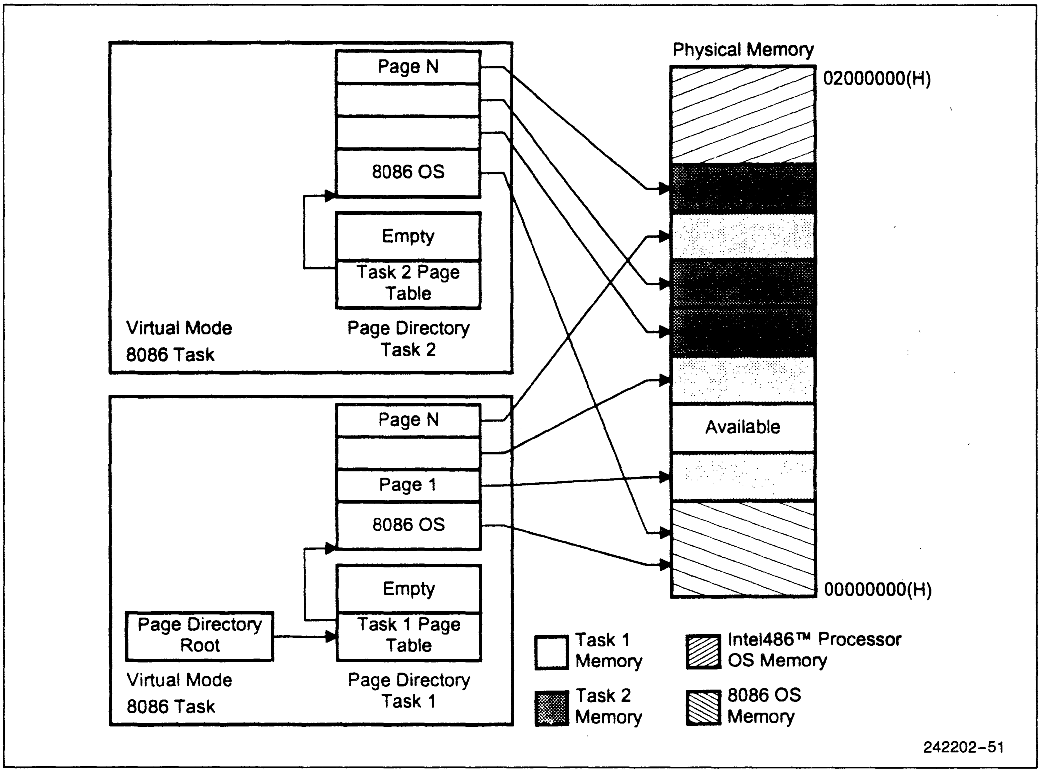


Figure 6-24. Virtual 8086 Environment Memory Management

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.

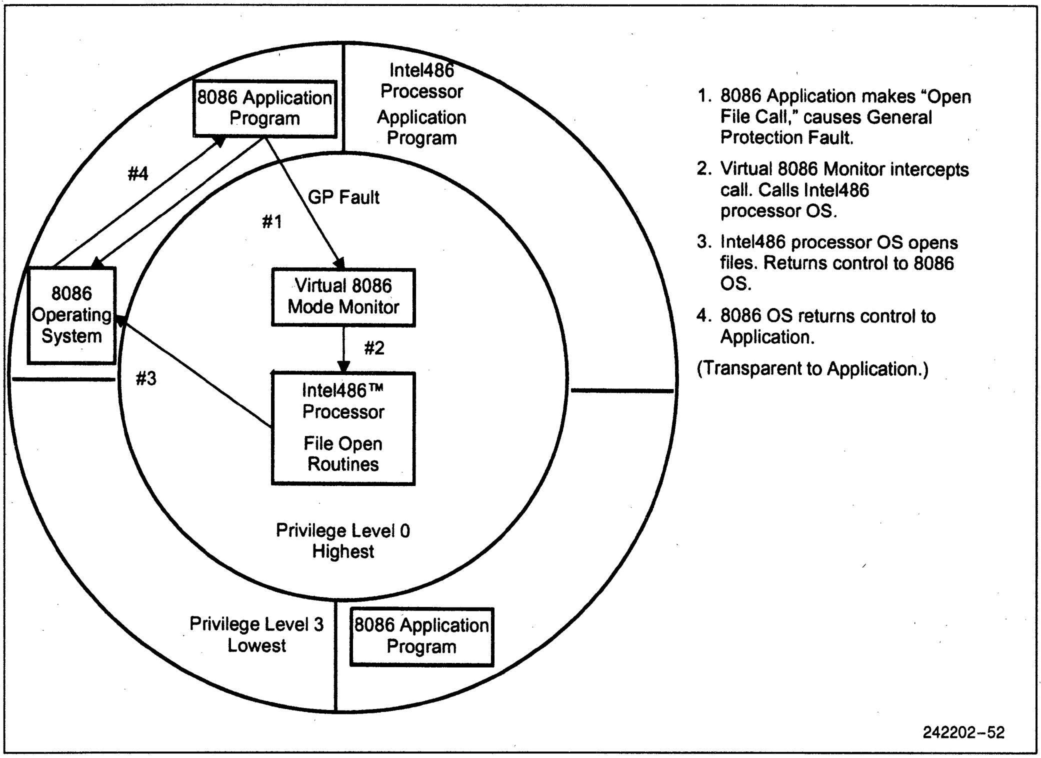
Because the matching IRET must occur from level 0, if an Interrupt or Trap Gate is used to field an interrupt or exception out of Virtual 8086 mode, the Gate must perform an inter-level interrupt only to level 0. Interrupt or Trap Gates through conforming segments, or through segments with DPL > 0, will 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.


Intel486 Processor Application Program
8086 Application Program
#4
GP Fault
#1
8086 Operating System
Virtual 8086 Mode Monitor
#2
#3
Intel486™ Processor File Open Routines
Privilege Level 0 Highest
Privilege Level 3 Lowest
8086 Application Program
242202–52

1. 8086 Application makes "Open File Call," causes General Protection Fault.
2. Virtual 8086 Monitor intercepts call. Calls Intel486 processor OS.
3. Intel486 processor OS opens files. Returns control to 8086 OS.
4. 8086 OS returns control to Application.

(Transparent to Application.)

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 intertask 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. 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. 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 Intel486 processor resumes the interrupted routine in Protected mode of 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 7.1.2, "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 7.1.2 for IntelDX4 processor details).

The cache organization is 4-way set associative and each line is 16-bytes wide. The eight Kbytes of cache memory are logically organized as 128 sets, each containing four 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.

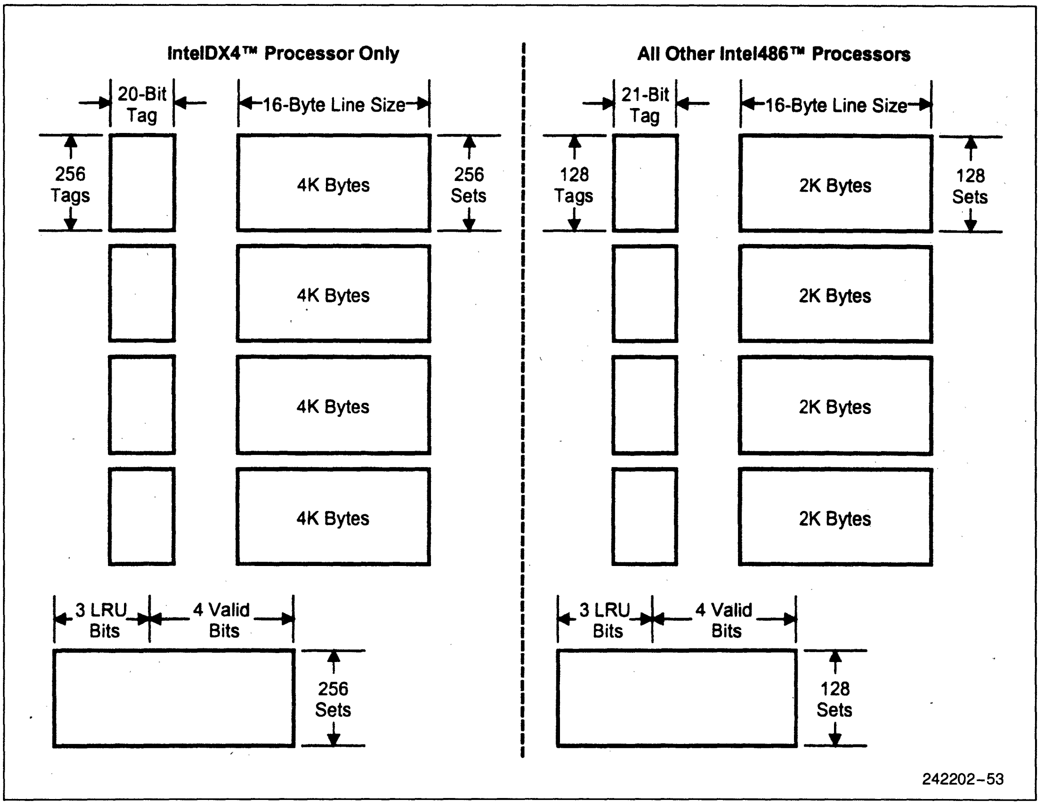


Figure 7-1. On-Chip Cache Physical Organization

For all Intel486 processors except the Write-Back Enhanced IntelDX2, 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 IntelDX2 processor supports 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 IntelDX2 is the same as the write-through cache for all other Intel486 processors. (See section 7.1.1, "Write-Back Enhanced IntelDX2 Processor Cache" and other write-back enhanced sections below for write-back cache information.)

7.1.1 WRITE-BACK ENHANCED INTELDX2 PROCESSOR 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 IntelDX2 processor 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 IntelDX2 processor 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 IntelDX2 in standard Bus Mode (Write-Through Cache) and all other Intel486 processors.

7.1.2 INTELDX4 PROCESSOR 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.

7.2 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 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 was 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 INTELDX2 PROCESSOR CACHE CONTROL AND OPERATING MODES

The Write-Back Enhanced IntelDX2 processor retains 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 IntelDX2 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 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 INTELDX2 PROCESSOR SNOOP CYCLES AND WRITE-BACK MODE INVALIDATION

In Enhanced bus mode, the Write-Back Enhanced IntelDX2 processor 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 IntelDX2 processor 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 IntelDX2 processor 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 IntelDX2 processor when the other bus master initiates a memory write cycle to determine if the Write-Back Enhanced IntelDX2 processor 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 IntelDX2 processor 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 IntelDX2 processor 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 l0, l1, l2, and l3. The order in which the valid bits are checked during an invalidation is l0, l1, l2 and l3. 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 IntelDX2 processor 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 least recently used (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 is set 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.

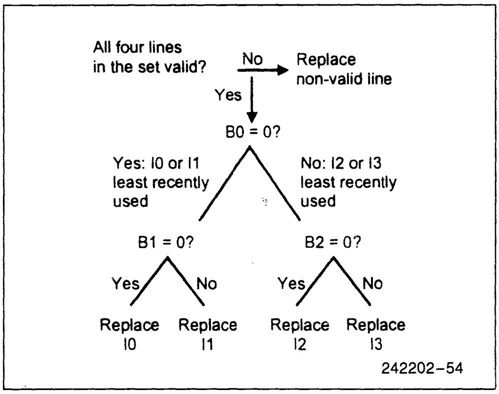


Figure 7-2. On-Chip Cache Replacement Strategy

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 are 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.

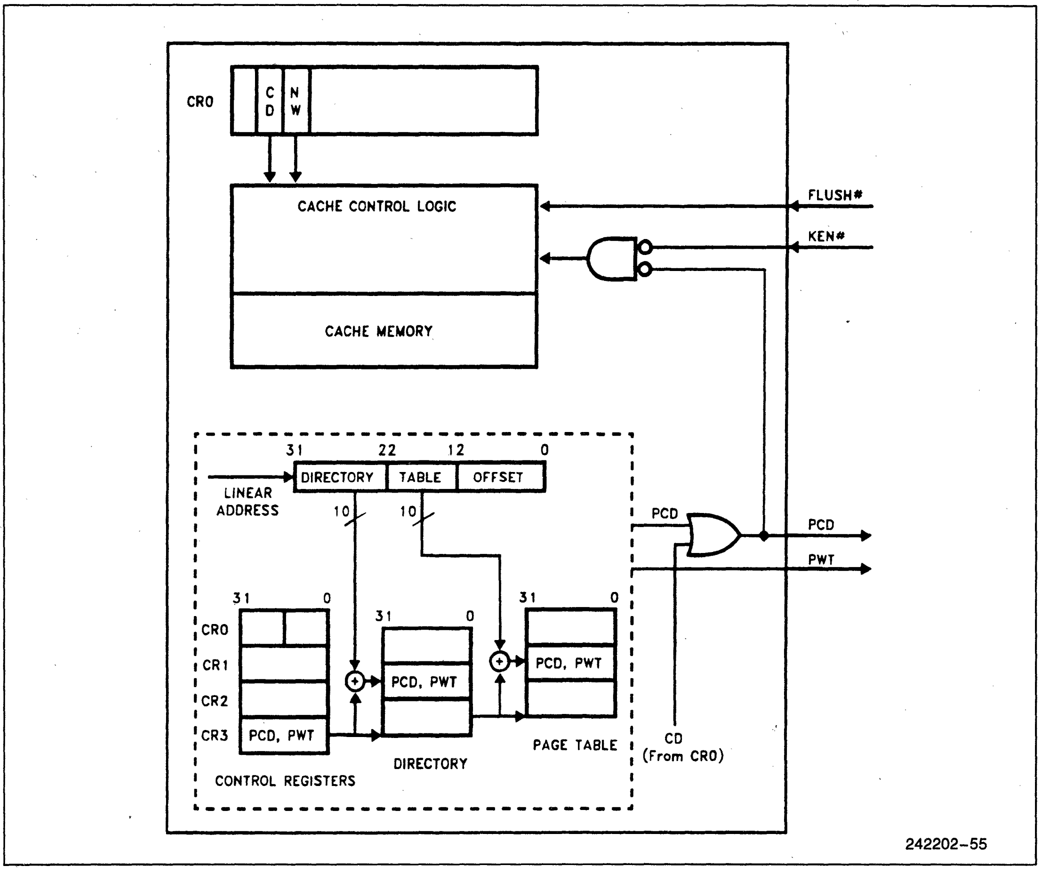


Figure 7-3. Page Cacheability

7.6.1 Write-Back Enhanced IntelDX2 PROCESSOR PAGE CACHEABILITY

In Write-Back Enhanced IntelDX2 processor-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 IntelDX2 processor. 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 INTELDX2 PROCESSOR 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 IntelDX2 processor 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 instruc-tion, 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 IntelDX2 processor 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 IntelDX2 processor 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 IntelDX2 Processor Write-Back Cache Architecture

This section describes additional features pertaining to the write-back mode of the Write-Back Enhanced IntelDX2 processor.

7.8.1 WRITE-BACK CACHE COHERENCY PROTOCOL

The Write-Back Enhanced IntelDX2 processor 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 re-ferred 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 IntelDX2 processor 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 IntelDX2 processor to execute a write-through cycle on the bus.

Every line in the Write-Back Enhanced IntelDX2 processor cache is assigned a state dependent on both Write-Back Enhanced IntelDX2 processor generated activities and activities generated by the system hardware. As the Write-Back Enhanced IntelDX2 processor is targeted for uniprocessor systems, a subset of MESI protocol, namely MEI, is used in the Write-Back Enhanced IntelDX2 processor 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 transi-tions 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 IntelDX2 processor also drives 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 IntelDX2 processor-generated write cycles are described in Table 7-6.

Table 7-5. Cache State Transitions for Write-Back Enhanced IntelDX2™ Processor 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 pro-vided to processor core by cache. No bus cycle is generated.
S	n/a	S	Read hit; Data is pro-vided 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 gen-erated 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 IntelDX2 processor.

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 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 con-tained within the cache of the Write-Back Enhanced IntelDX2 processor. 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 INTELDX2 PROCESSOR

The write-back policy of the on-chip cache of the Write-Back Enhanced IntelDX2 processor 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 IntelDX2 processor) 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 7. When the Family ID is 4, and the model ID is 7, the processor supports on-chip write-back caching. If the Family ID does not match 7, the processor only supports on-chip write-through caching.

The hardware mechanism involves using the HITM# signal. For the Write-Back Enhanced IntelDX2 processor, 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).

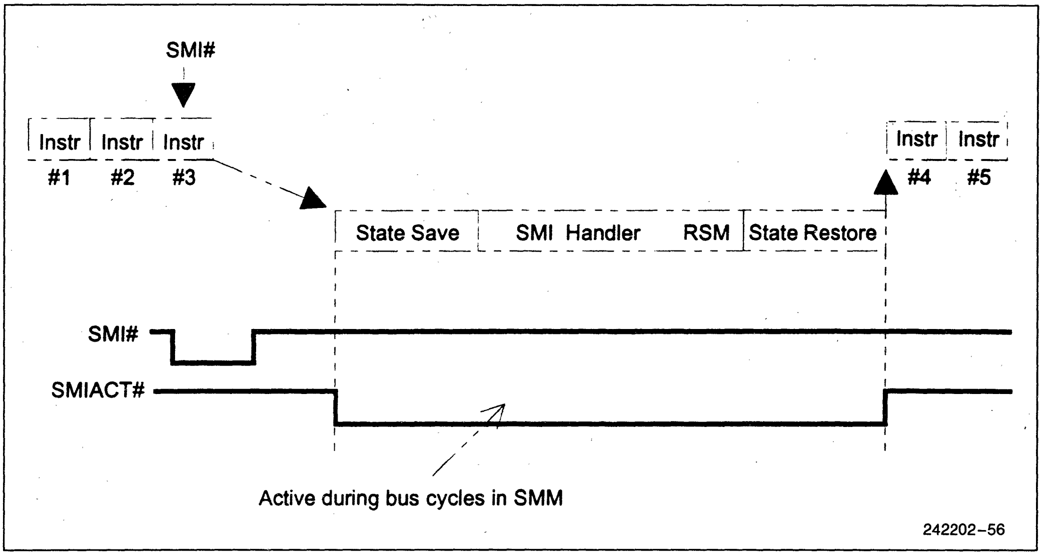


Figure 8-1. Basic SMI# Interrupt Service

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 processors 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.

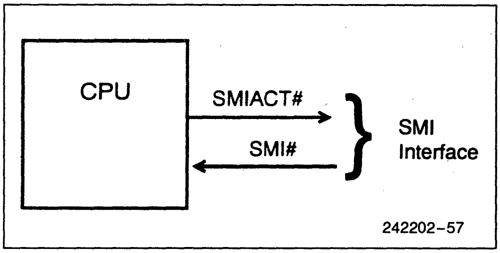


Figure 8-2. Basic SMI# Hardware Interface

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.

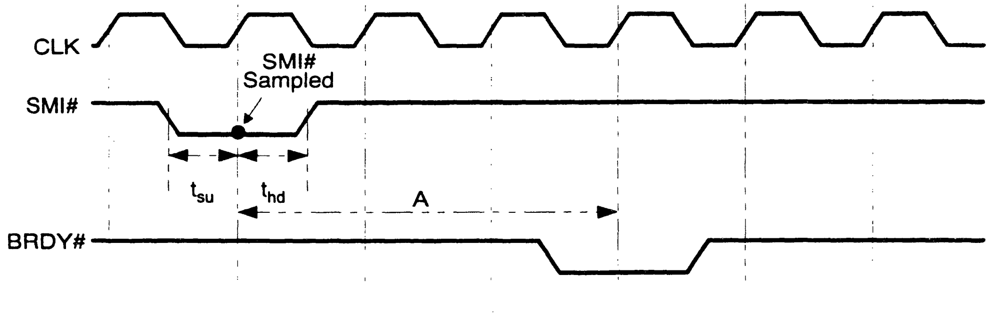
After the SMI# interrupt is recognized, the SMI# signal will be masked internally until the RSM instruction is executed and the interrupt service routine is complete. Masking the SMI# prevents recursive SMI# calls. SMI# must be de-asserted for at least 4 clocks to reset the edge triggered logic. If another 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.



A: Setup time for recognition on I/O instruction boundary

242202-58

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

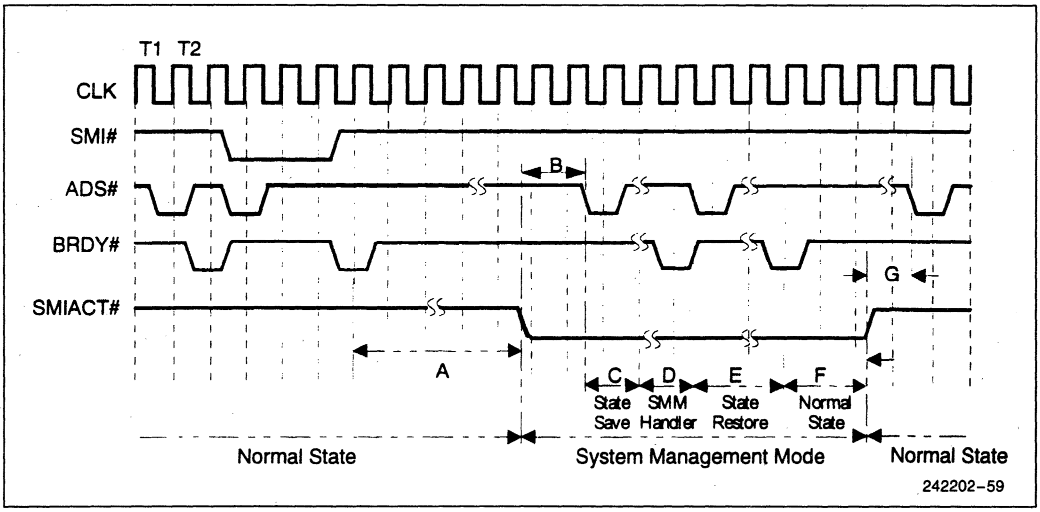


Figure 8-4. Intel486™ Processor SMIACT# Timing

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.

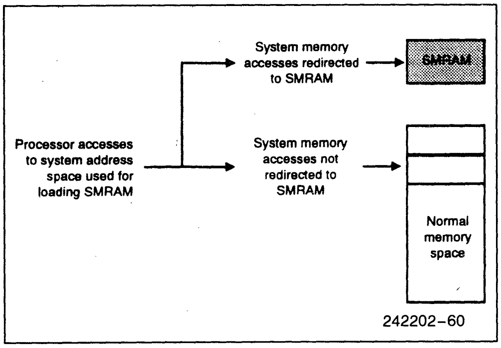

System memory accesses redirected to SMRAM
SMRAM
Processor accesses to system address space used for loading SMRAM
System memory accesses not redirected to SMRAM
Normal memory space
242202-60

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.

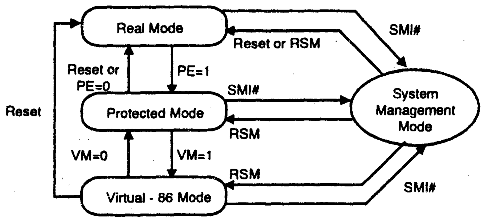
NOTE:

> 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 IntelDX2 Processor Environment

When the Write-Back Enhanced IntelDX2 processor is 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 processors 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.)

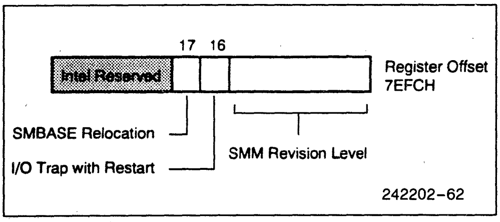


Figure 8-7. SMM Revision Identifier

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.)

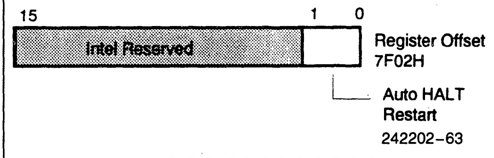


Figure 8-8. Auto HALT Restart

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.)

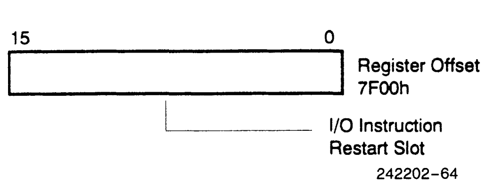


Figure 8-9. I/O Instruction Restart

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.)

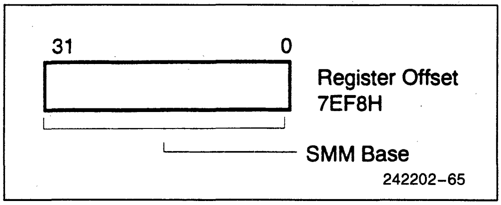


Figure 8-10. SMM Base Location

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:

$$\text{SMBASE} + 8000\text{H}$$

While the starting address for the SMRAM state save area is calculated by:

$$\text{SMM Base} + [8000\text{H} + 7\text{FFFH}]$$

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.

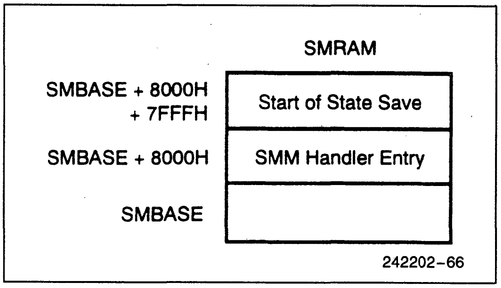


Figure 8-11. SMRAM Usage

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.

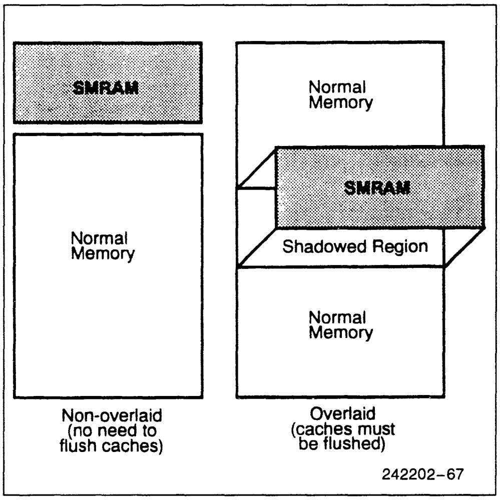


Figure 8-12. SMRAM Location

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#.

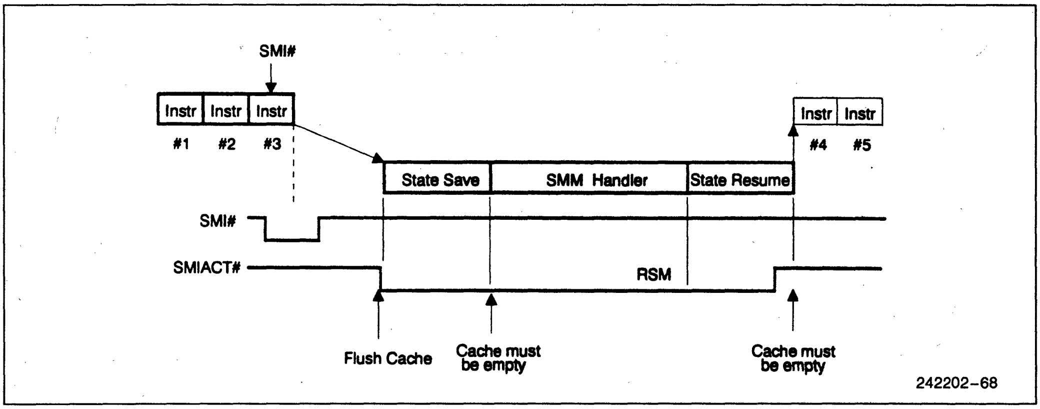


Figure 8-13. FLUSH# Mechanism during SMM

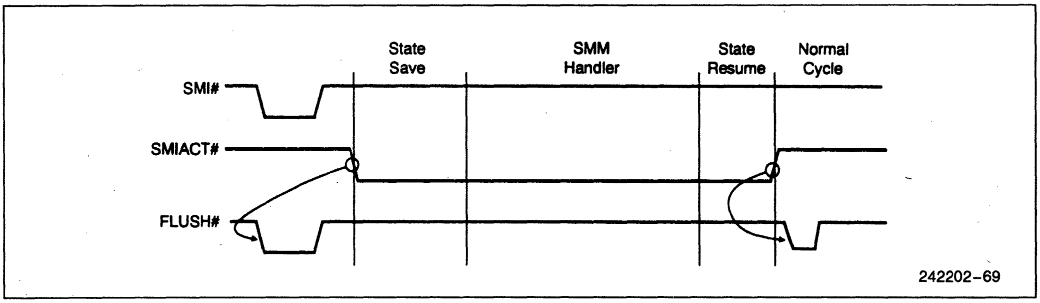


Figure 8-14. Cached SMM

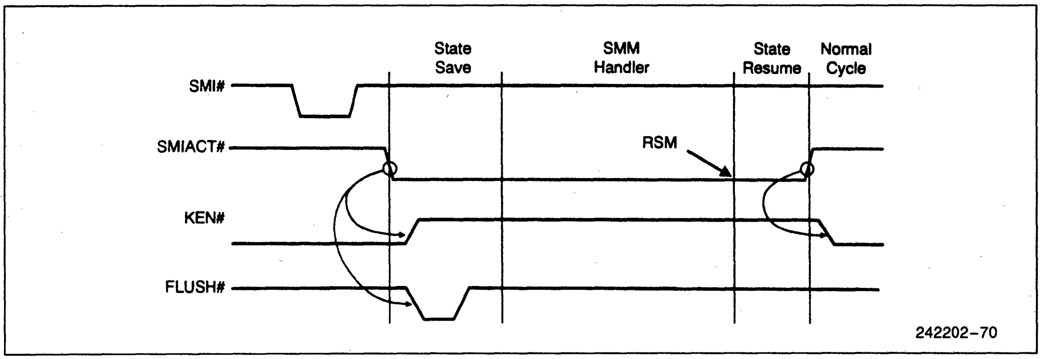


Figure 8-15. Non-Cached SMM

8.6.2.1 Write-Back Enhanced 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 IntelDX2 processor guarantees 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 IntelDX2 processor 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.

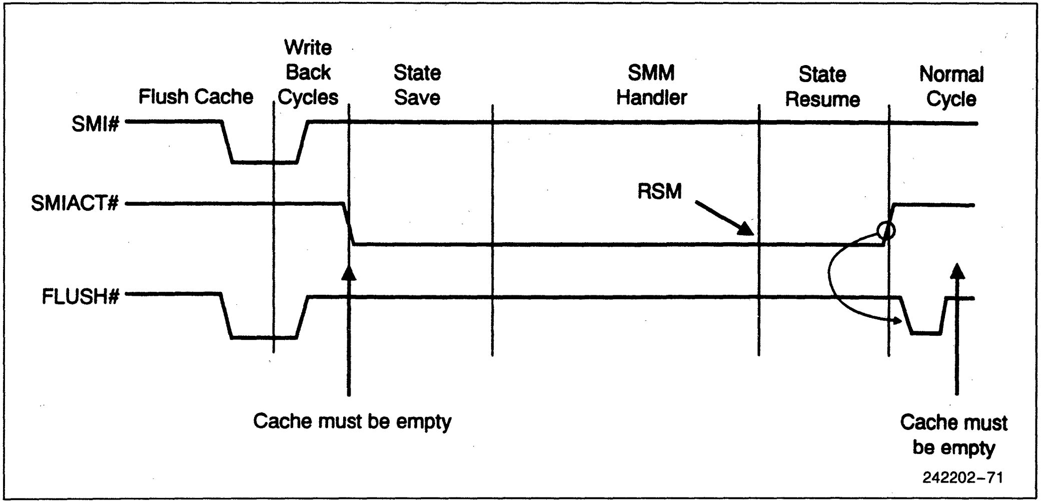


Figure 8-16. Write-Back Enhanced IntelDX2™ Processor Cache Flushing for Overlaid SMRAM upon Entry and Exit of Cached SMM

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]
```

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

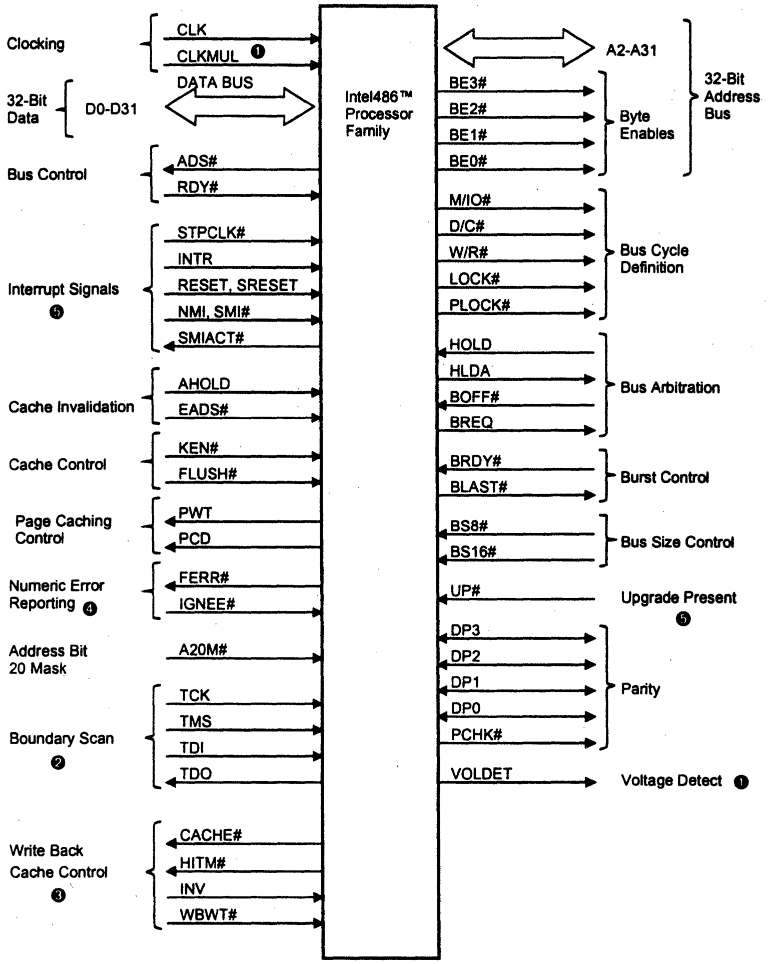
9.2.1 CLOCK (CLK)

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.

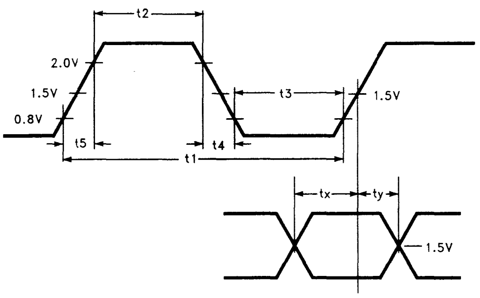
❸ Pins on Write-Back Enhanced IntelDX2 when in Enhanced Bus Mode/write back cache mode.

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



tx = input setup times
ty = input hold times, output float, valid and hold times

Figure 9-2. CLK Waveform

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

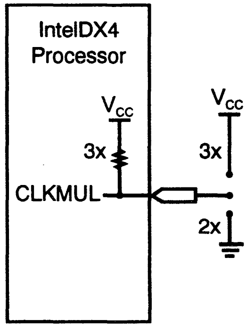


Figure 9-3. Voltage Detect (VOLDET) Sense Pin

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 are 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 insure 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 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# are returned.

A pseudo-locked transfer is meaningful only if the memory operand is aligned and if its completely contained within a single cache line.

Because PLOCK# is a function of the bus size and KEN# inputs, PLOCK# should be sampled only in 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 IntelDX2 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-bursted 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 tristate 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 IntelDX2 processor. (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).

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 de-asserted, the processor will freeze on a non-control floating point instruction if a previous instruction caused an error. 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 insure 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# are 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 2 clocks prior to RDY being returned for the I/O write. This insures 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 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. Write-Back Enhanced IntelDX2™ Processor 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™ 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 can not 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 on-going 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™ 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 IntelDX2 processor operates 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.

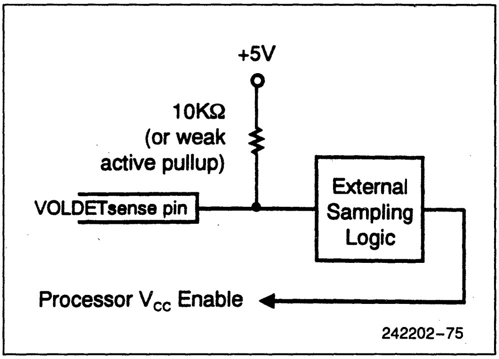


Figure 9-4. Voltage Detect (VOLDET) Sense Pin

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.

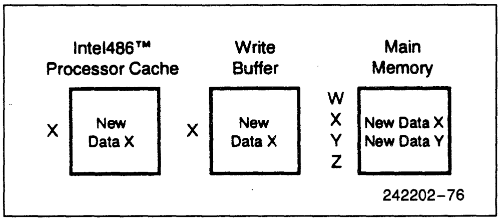


Figure 9-5. Reordering of a Reads with Write Buffers

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).

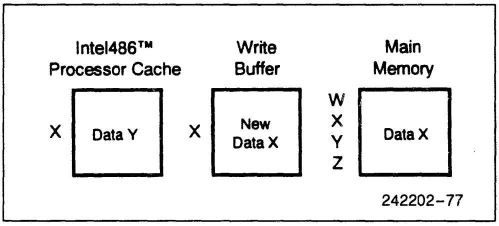


Figure 9-6. Reordering of a Reads with Write Buffers

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 insures 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 noncacheable 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 1 clock before and 1 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 was 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	04	7x
IntelDX2 Processor		
IntelDX4™ Processor	04	8x

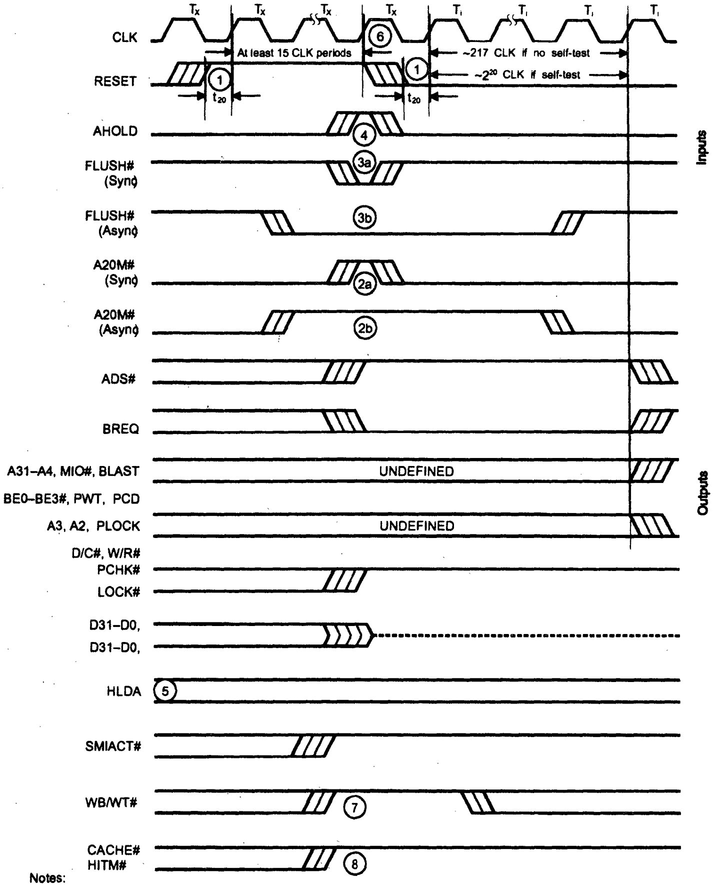
Table 9-10. Floating Point Values after Reset

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:

1. 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 period prior to and two CLKs after the falling edge of RESET to invoke the 3-state Output Test Mode. All outputs are guaranteed 3-stated within 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.

242202–78

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 regards to allowable CLK input during the power on sequence have never been specified. Clocking the processor before Vcc 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 Vcc 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 Vcc 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 floating point unit status word. The ES bit is initialized whenever the floating point unit state is initialized. The floating point unit'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 (07) will be inactive irrespective of the Built-In Self-Test (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 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 statethe 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 section 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 INTELDX2 PIN STATE DURING STOP GRANT SPECIFICS

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 IntelDX2 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 the state they were in prior to the HOLD/HLDA sequence.

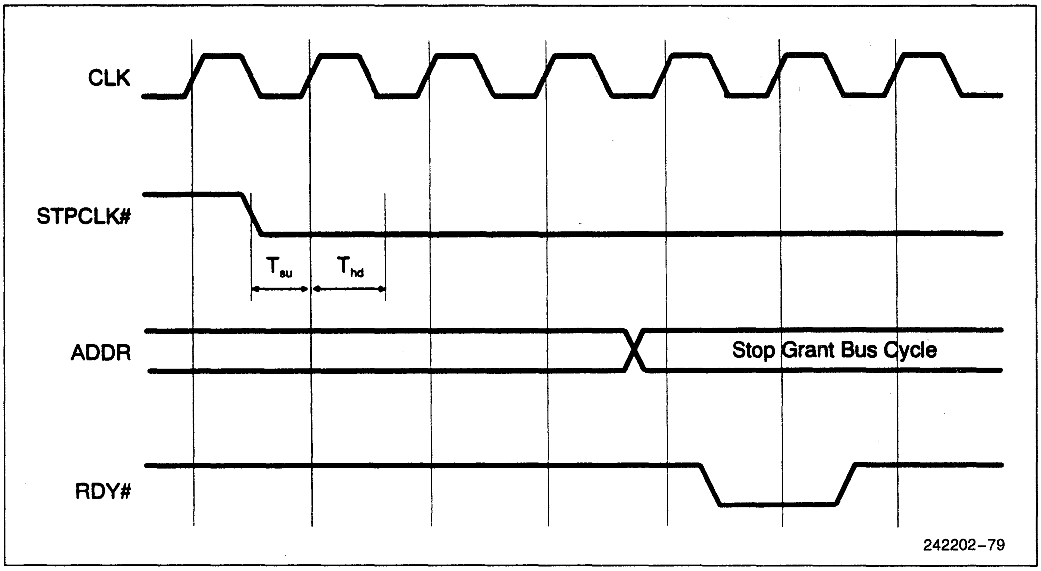


Figure 9-8. Stop Clock Protocol

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 IntelDX2 processor 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 resisters).

Table 9-13. 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 IntelDX2 processor Clock State specifics.

9.6.4.1 Normal State

This is the normal operating state of the processor.

9.6.4.2 Stop Grant State

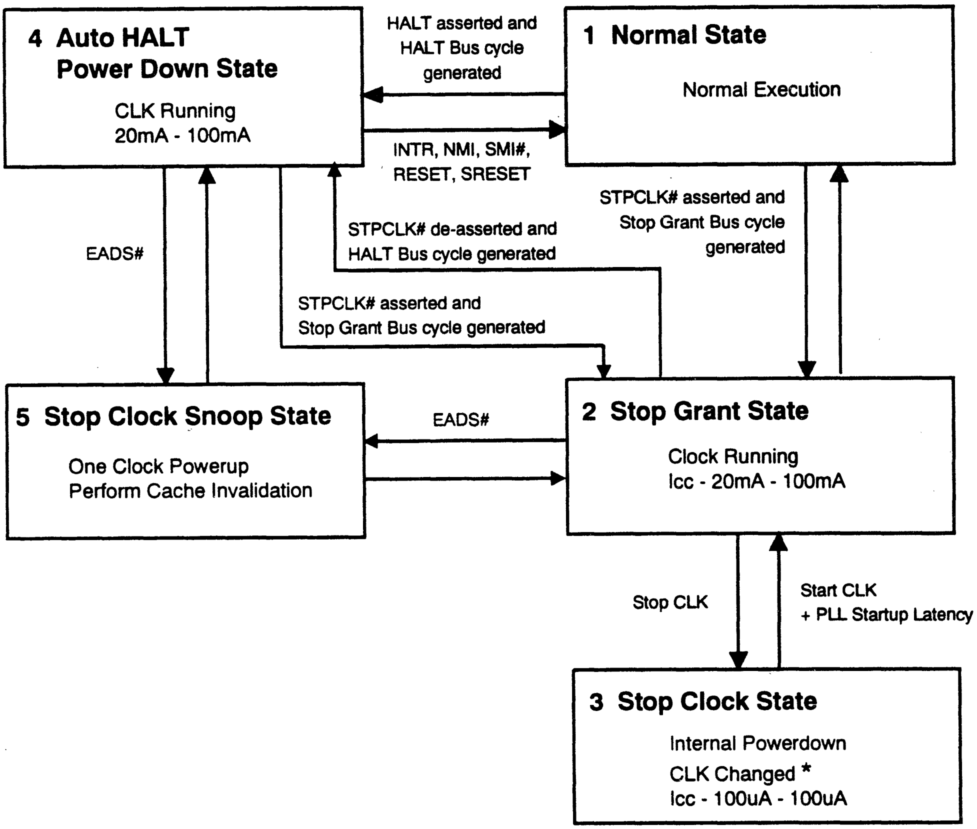
The **Stop Grant** state provides a fast wake-up state that can be entered by simply asserting the external STPCLK# interrupt pin. Once the Stop Grant bus cycle has been placed on the bus, and either RDY# or BRDY# is returned, the processor is in this state (depending on the CLK input frequency). The 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.


4 Auto HALT Power Down State
CLK Running
20mA - 100mA
HALT asserted and HALT Bus cycle generated
INTR, NMI, SMI#, RESET, SRESET
1 Normal State
Normal Execution
EADS#
STPCLK# de-asserted and HALT Bus cycle generated
STPCLK# asserted and Stop Grant Bus cycle generated
STPCLK# asserted and Stop Grant Bus cycle generated
5 Stop Clock Snoop State
One Clock Powerup
Perform Cache Invalidation
EADS#
2 Stop Grant State
Clock Running
Icc - 20mA - 100mA
Stop CLK
Start CLK + PLL Startup Latency
3 Stop Clock State
Internal Powerdown
CLK Changed *
Icc - 100uA - 100uA

242202–80

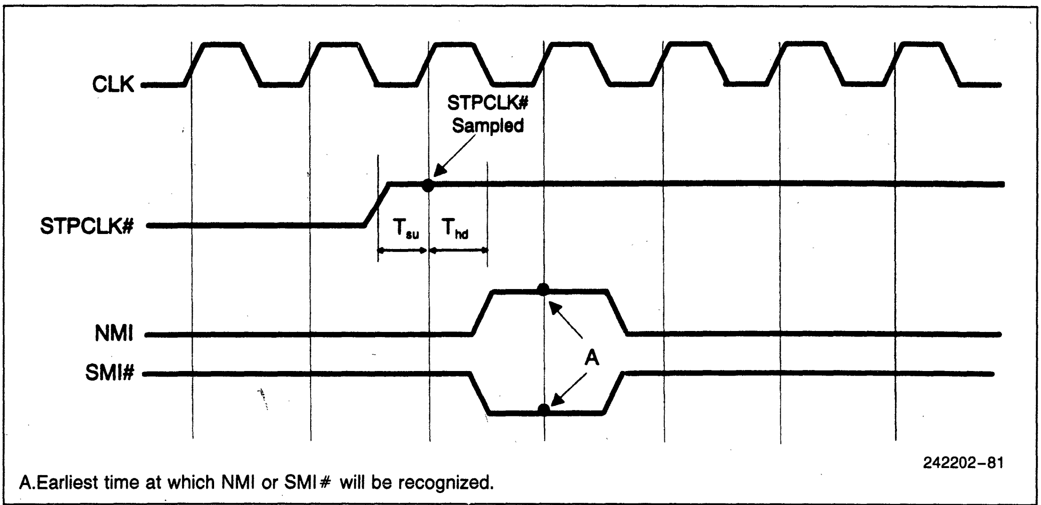
* The system can change the input frequency within the specified range or completely stop the CLK input frequency (input to PLL)

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 INTELDX2 PROCESSOR CLOCK CONTROL STATE DIAGRAM

Figure 9-11 (state diagram) shows the state transitions during Stop Clock for the Write-Back Enhanced IntelDX2 processor.

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".)

Normal StateThis 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 it's normal state.

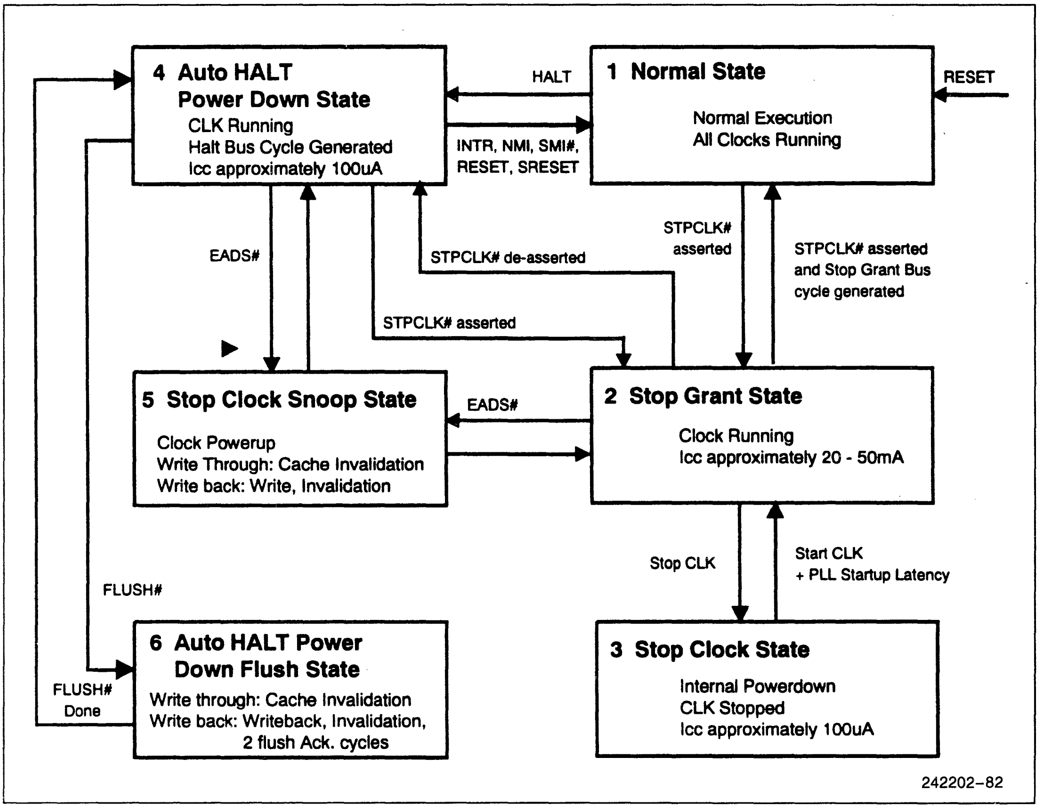

4 Auto HALT Power Down State
CLK Running
Halt Bus Cycle Generated
Icc approximately 100uA
HALT
INTR, NMI, SMI#, RESET, SRESET
1 Normal State
Normal Execution
All Clocks Running
RESET
EADS#
STPCLK# de-asserted
STPCLK# asserted
STPCLK# asserted
STPCLK# asserted and Stop Grant Bus cycle generated
5 Stop Clock Snoop State
Clock Powerup
Write Through: Cache Invalidation
Write back: Write, Invalidation
EADS#
2 Stop Grant State
Clock Running
Icc approximately 20 - 50mA
Stop CLK
Start CLK + PLL Startup Latency
FLUSH#
6 Auto HALT Power Down Flush State
Write through: Cache Invalidation
Write back: Writeback, Invalidation, 2 flush Ack. cycles
FLUSH# Done
3 Stop Clock State
Internal Powerdown
CLK Stopped
Icc approximately 100uA
242202–82

Figure 9-11. 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 IntelDX2 processor requires 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 IntelDX2 processor 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 StateStop 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 startup 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 IntelDX2

If the 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.

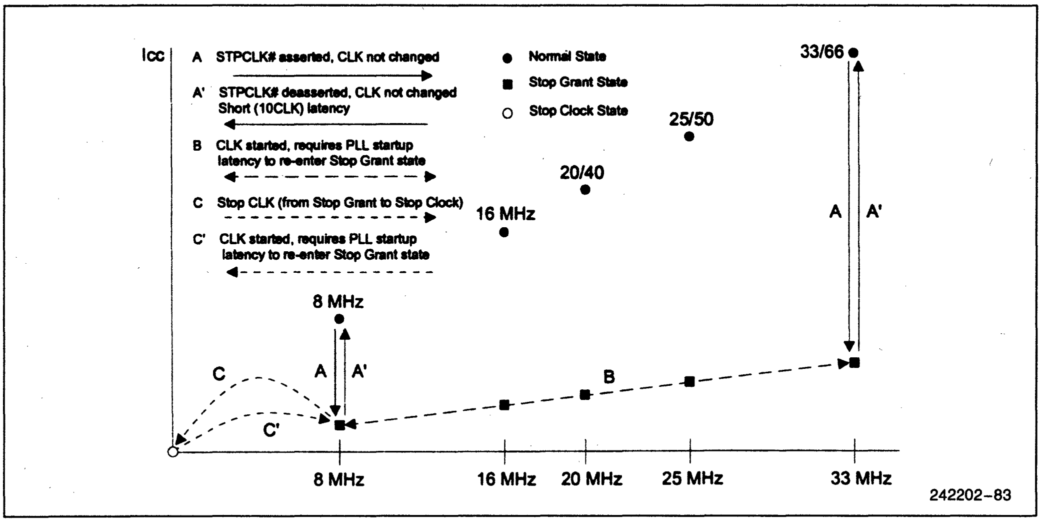

Icc
A STPCLK# asserted, CLK not changed
A' STPCLK# deasserted, CLK not changed
Short (10CLK) latency
B CLK started, requires PLL startup
latency to re-enter Stop Grant state
C Stop CLK (from Stop Grant to Stop Clock)
C' CLK started, requires PLL startup
latency to re-enter Stop Grant state
● Normal State
■ Stop Grant State
○ Stop Clock State
33/66
25/50
20/40
16 MHz
8 MHz
A
A'
B
C
C'
8 MHz
16 MHz
20 MHz
25 MHz
33 MHz
242202-83

Figure 9-12. Supply Current Model for Stop Clock Modes and Transitions for the Intel486™ Processor

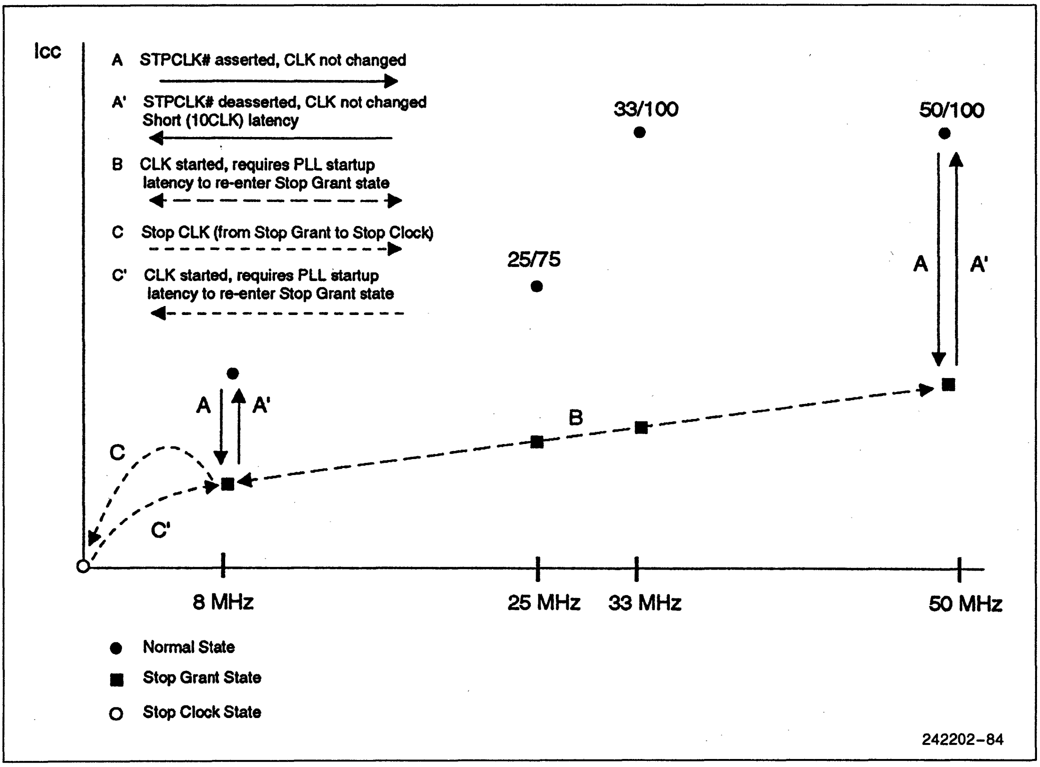

Icc
A STPCLK# asserted, CLK not changed
A' STPCLK# deasserted, CLK not changed
Short (10CLK) latency
B CLK started, requires PLL startup
latency to re-enter Stop Grant state
C Stop CLK (from Stop Grant to Stop Clock)
C' CLK started, requires PLL startup
latency to re-enter Stop Grant state
33/100
50/100
25/75
A
A'
B
C
C'
8 MHz
25 MHz
33 MHz
50 MHz
● Normal State
■ Stop Grant State
○ Stop Clock State
242202-84

Figure 9-13. Supply Current Model for Stop Clock Modes and Transitions for the IntelDX4™ Processor

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 IntelDX2 processor 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

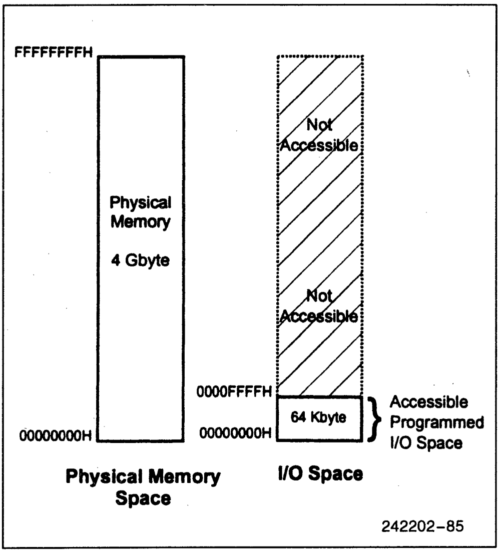


Figure 10-1. Physical Memory and I/O Spaces

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.

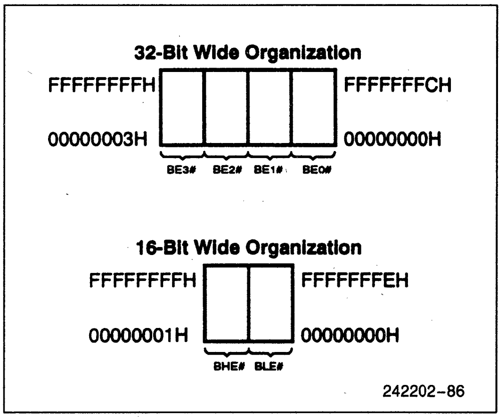


Figure 10-2. Physical Memory and I/O Space Organization

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#.

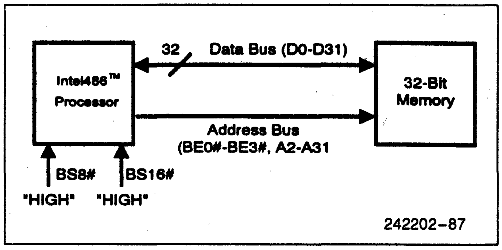


Figure 10-3. Intel486™ Processor with 32-Bit Memory

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.

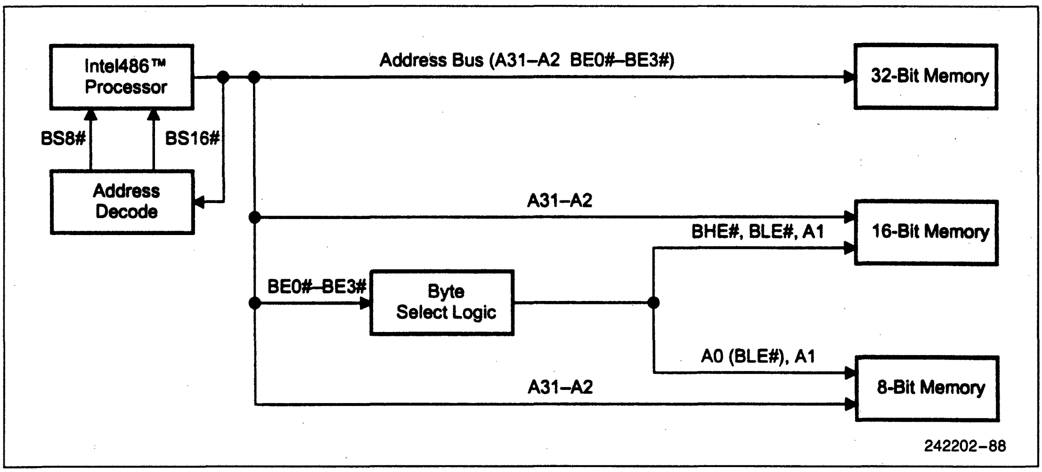


Figure 10-4. Addressing 16- and 8-Bit Memories

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

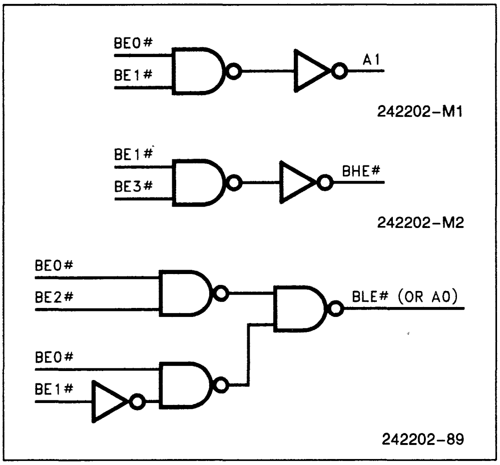


Figure 10-5. Logic to Generate A1, BHE# and BLE# for 16-Bit Buses

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).

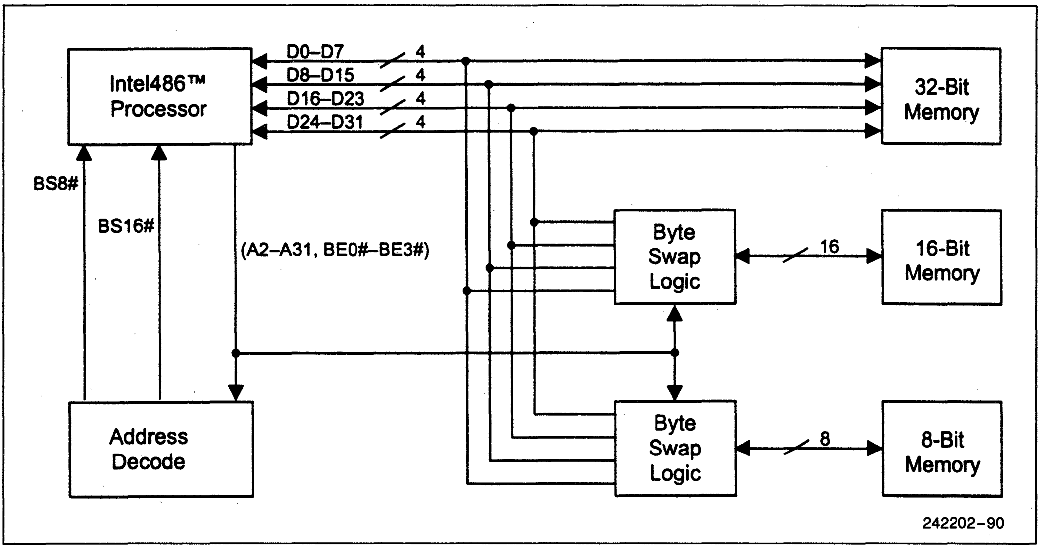


Figure 10-6. Data Bus Interface to 16- and 8-Bit Memories

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

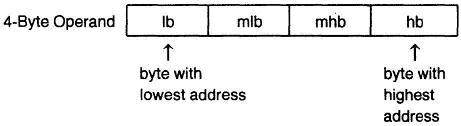
				First Cache Fill Cycle			Any Other Cycle		
BE3#	BE2#	BE1#	BE0#	A0	A1	BHE#	A0	A1	BHE#
1	1	1	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
*0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	1	0	0
1	0	0	1	0	0	0	1	0	0
*0	0	0	1	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	1
*0	0	1	1	0	0	0	0	1	0
*0	1	1	1	0	0	0	1	1	0

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	h = high-order portion
w = 2-byte transfer	l = low-order portion
3 = 3-byte transfer	m = mid-order portion
d = 4-byte transfer	



The function of unaligned transfers with dynamic bus sizing is not obvious. When the external systems 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 is available on the cycle definition lines and address bus.

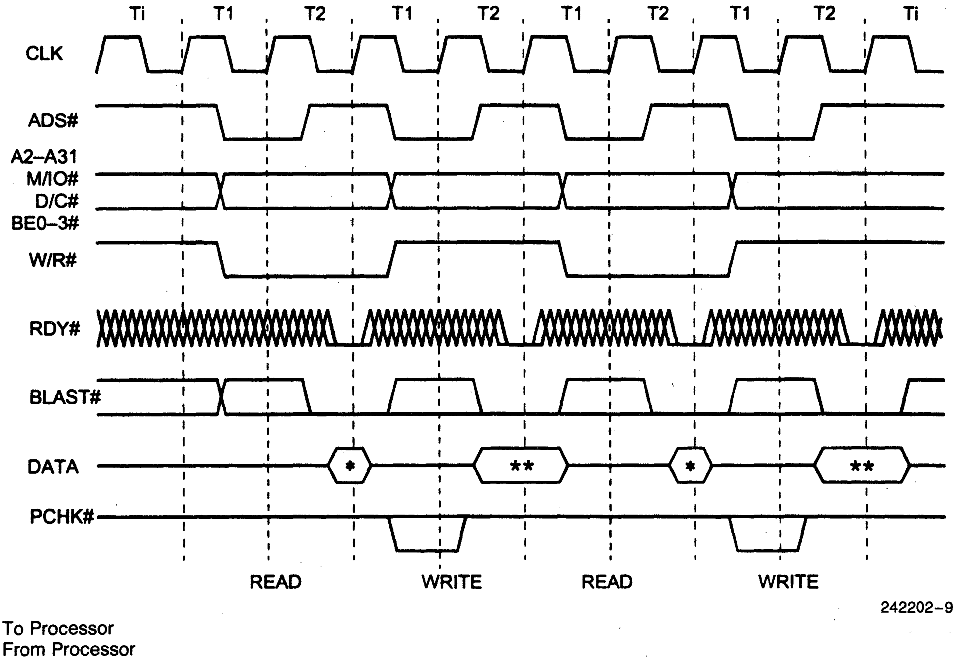


Figure 10-7. Basic 2-2 Bus Cycle

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.

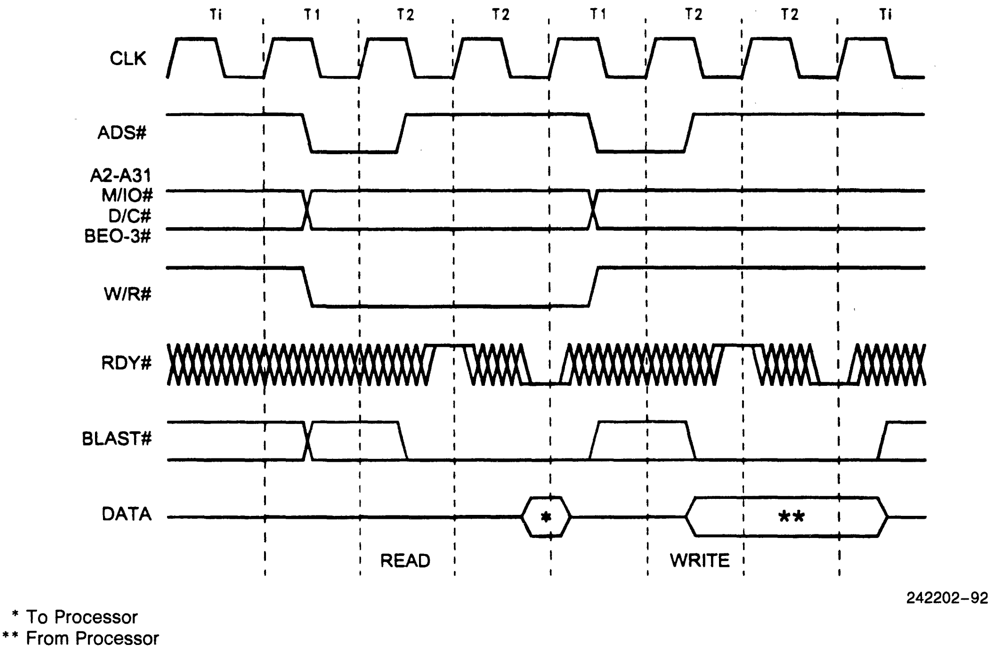


Figure 10-8. Basic 3-3 Bus Cycle

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#.

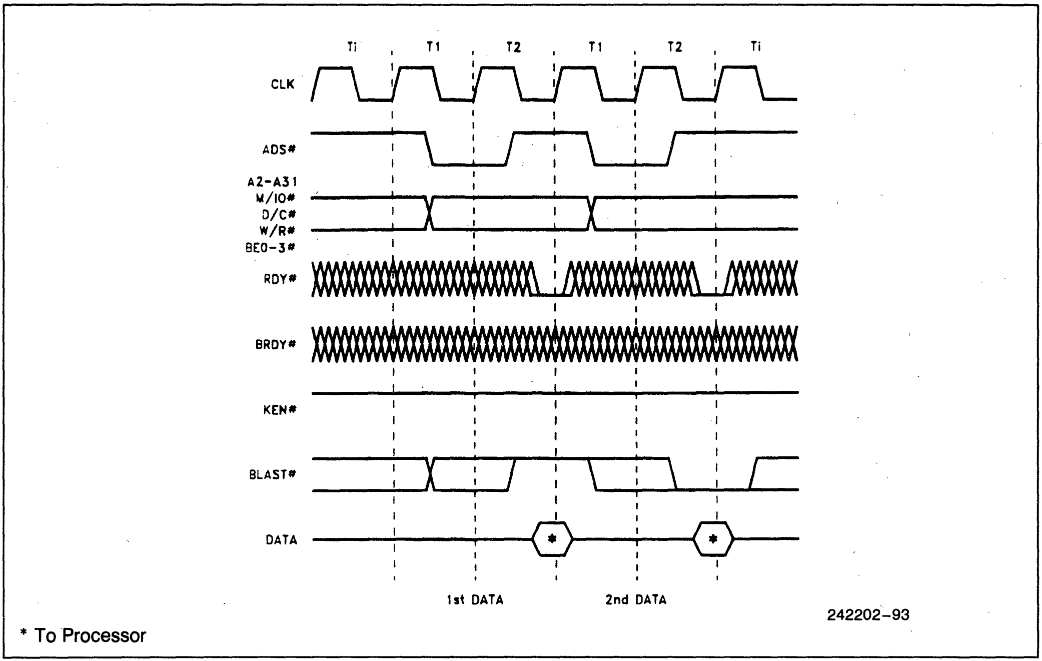


Figure 10-9. Non-Cacheable, Non-Burst, Multiple-Cycle Transfers

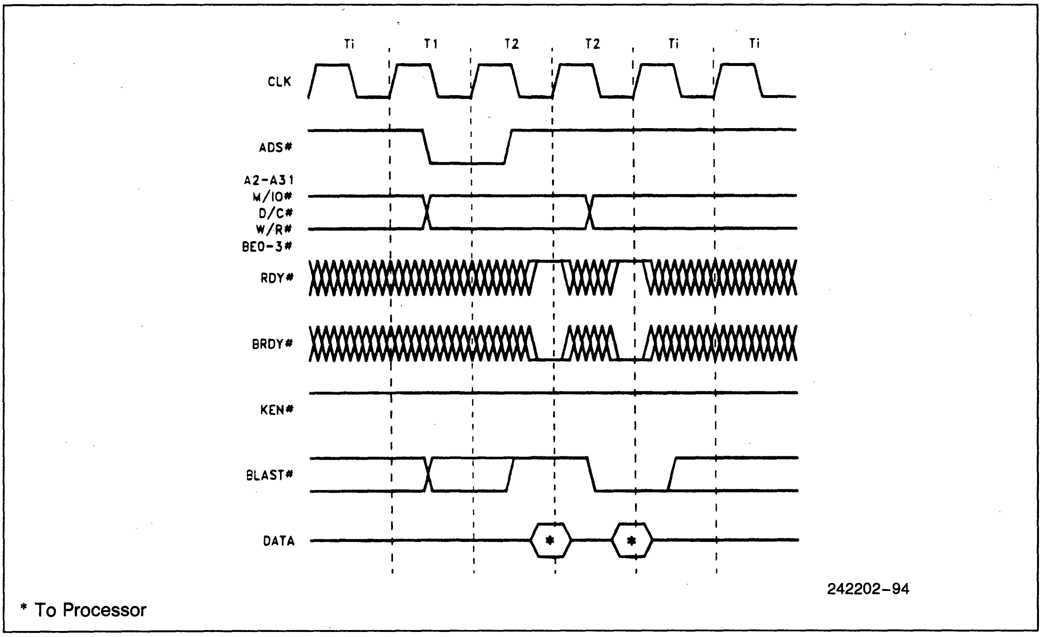


Figure 10-10. Non-Cacheable Burst Cycle

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."

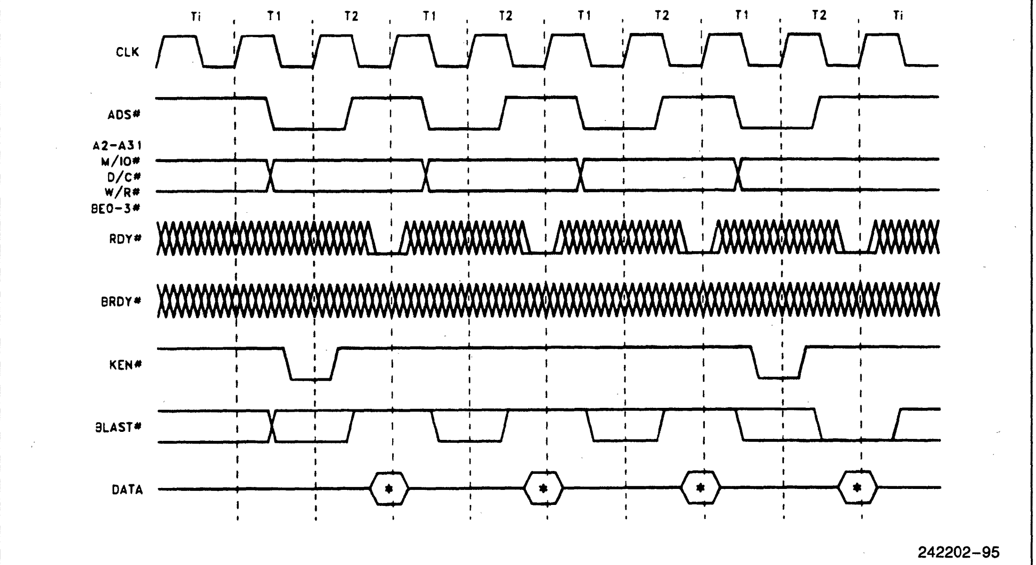
10.2.3.2 Non-Burst Cacheable Cycles

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 3 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') .



* To Processor

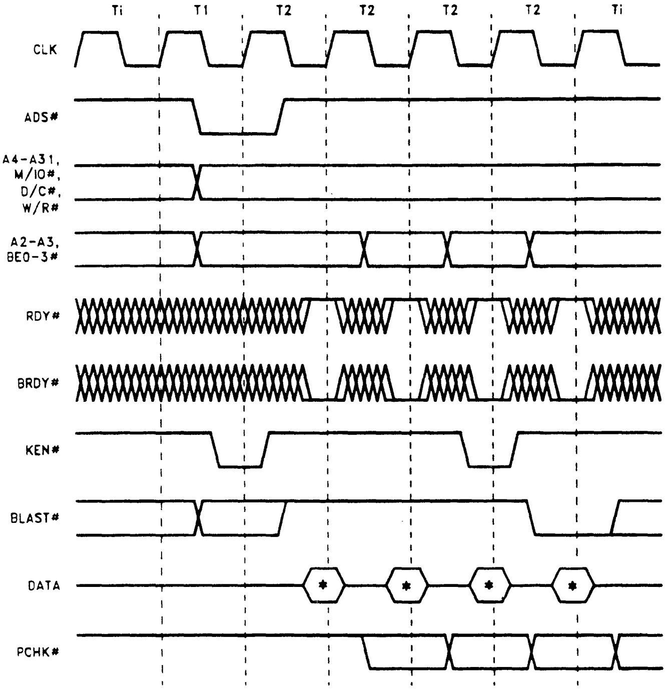
Figure 10-11. Non-Burst, Cacheable Cycles

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.



242202-96

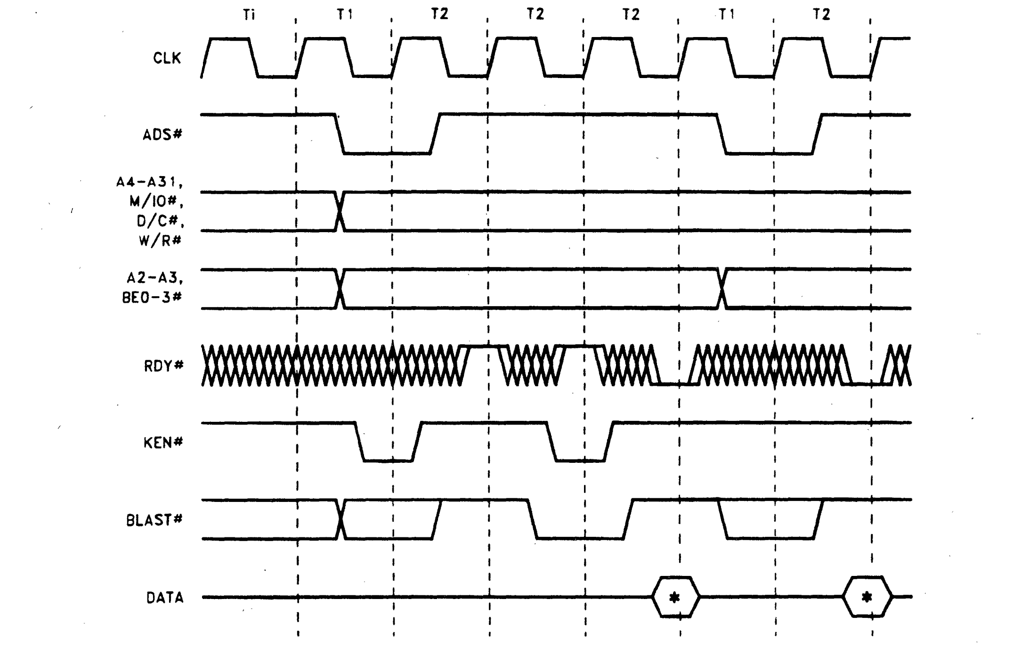
* To Processor

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.



242202-97

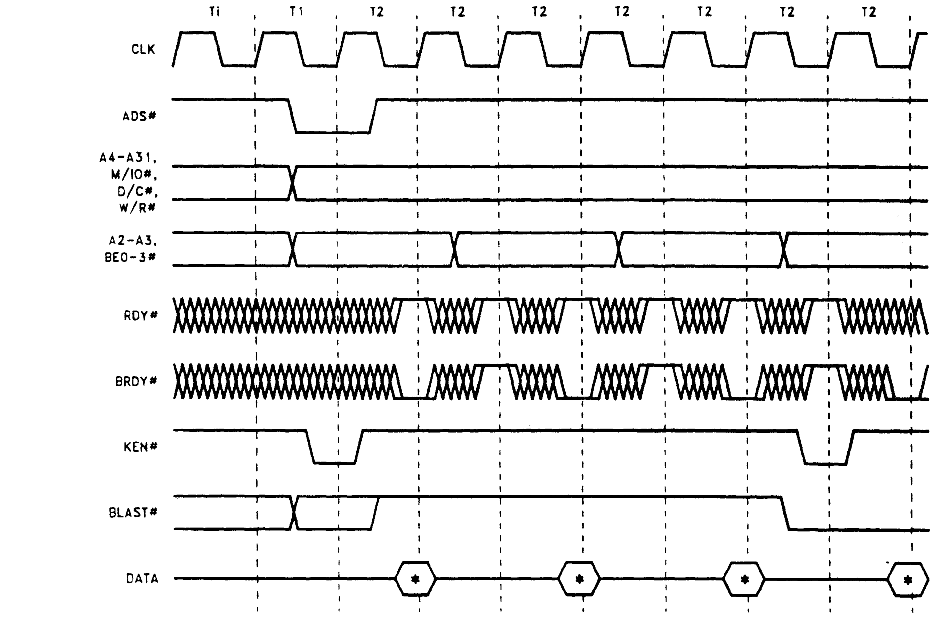
* To Processor

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# are 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.



242202-98

* To Processor

Figure 10-14. Slow Burst Cycle

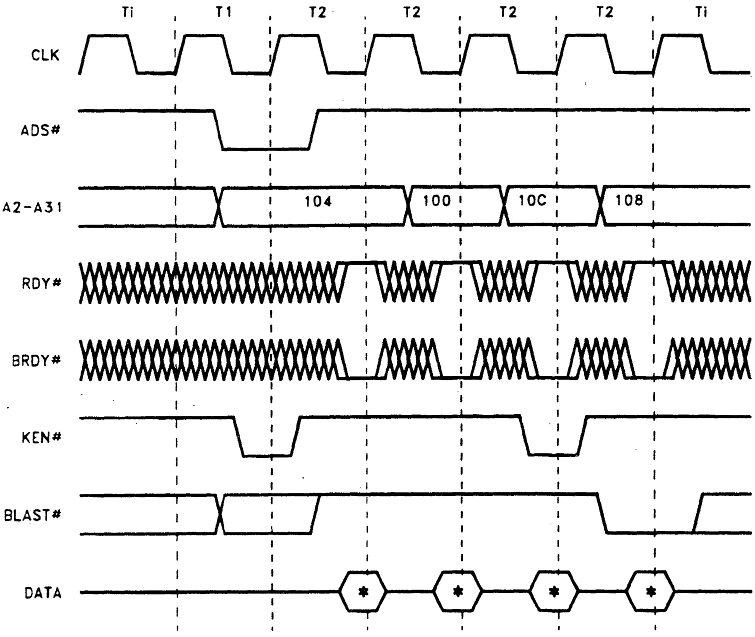
10.2.4.2 Burst and Cache Line Fill Order

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



242202-99

* To Processor

Figure 10-15. Burst Cycle Showing Order of Addresses

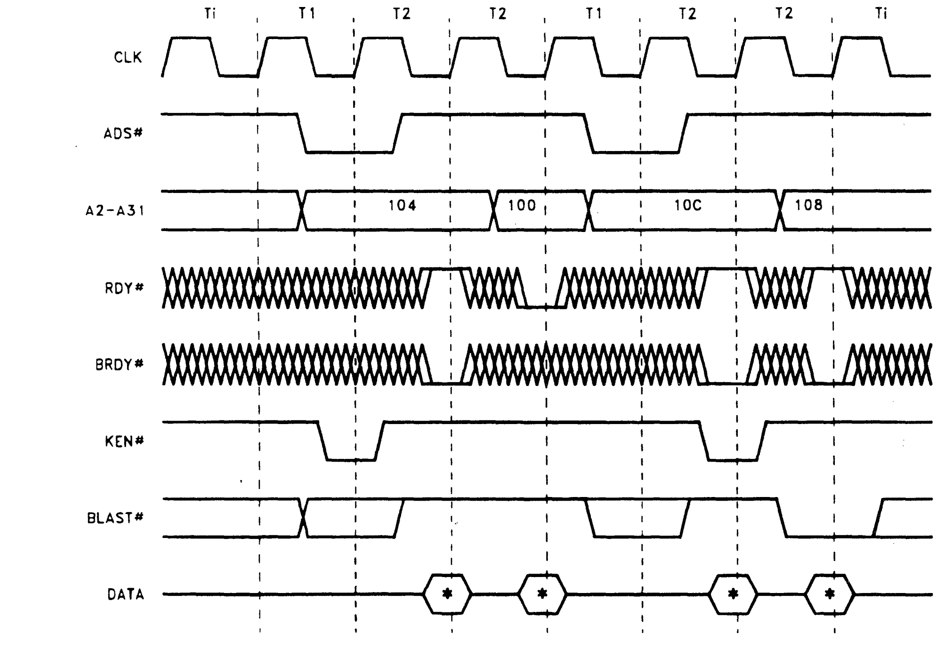
The sequences shown in Table 10-8 accommodate systems with 64-bit buses as well as systems with 32-bit data buses. The sequence applies to all bursts, regardless of whether the purpose of the burst is to fill a cache line, do a 64-bit read, or do a pre-fetch. If either BS8# or BS16# is returned active, 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# driven inactive one clock after ADS# begins the second bus cycle indicating that the transfer is not complete.



242202-A0

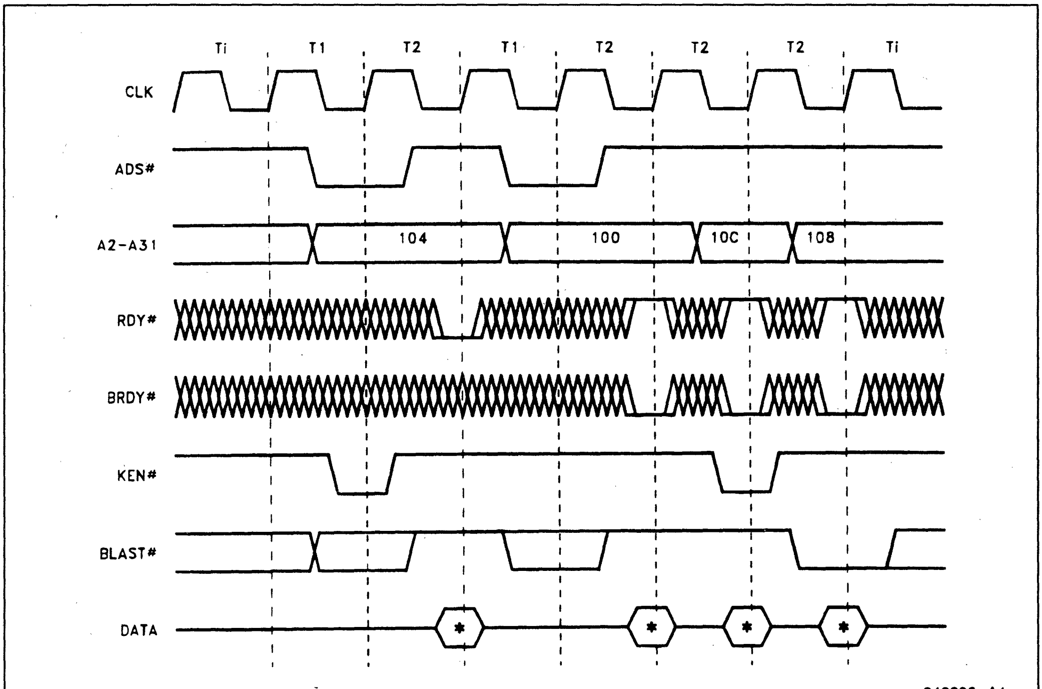
* To Processor

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 two 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#.



* To Processor

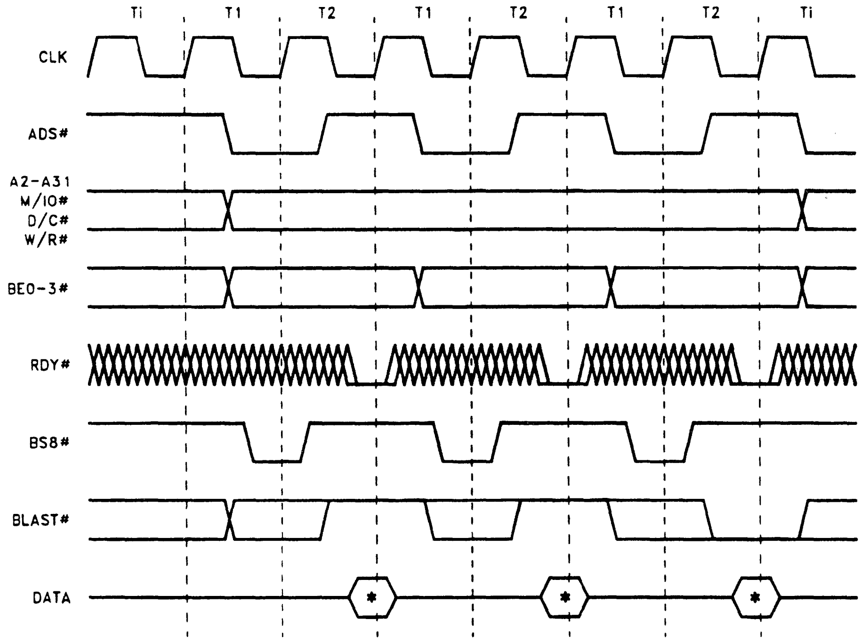
Figure 10-17. Interrupted Burst Cycle with Unobvious Order of Addresses

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.



242202-A2

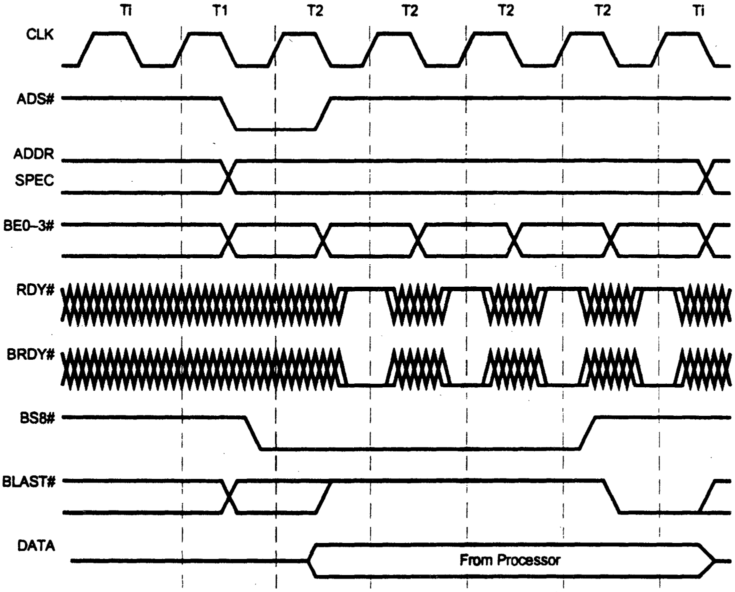
* To Processor

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# are 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.



242202–A3

Figure 10-19. Burst Write as a Result of BS8# or BS16#

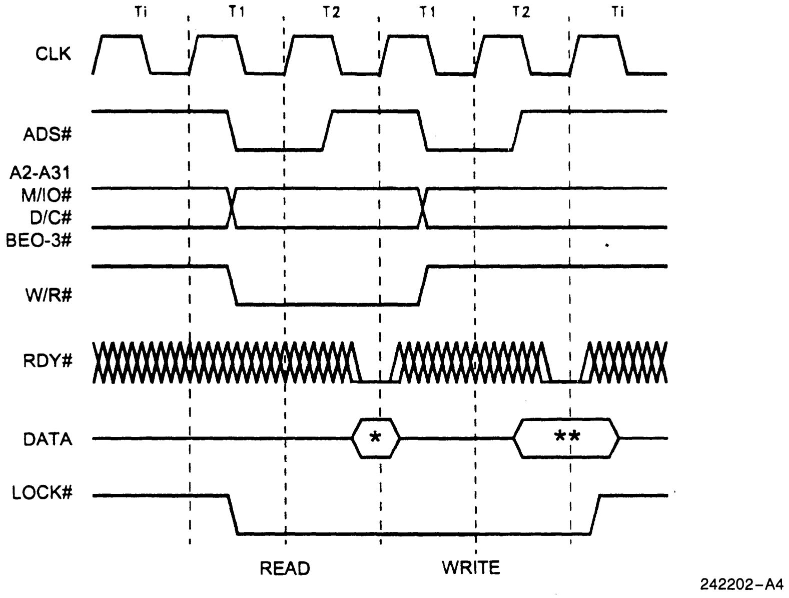
10.2.6 LOCKED CYCLES

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 bits 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#.



* To Processor
** From Processor

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.

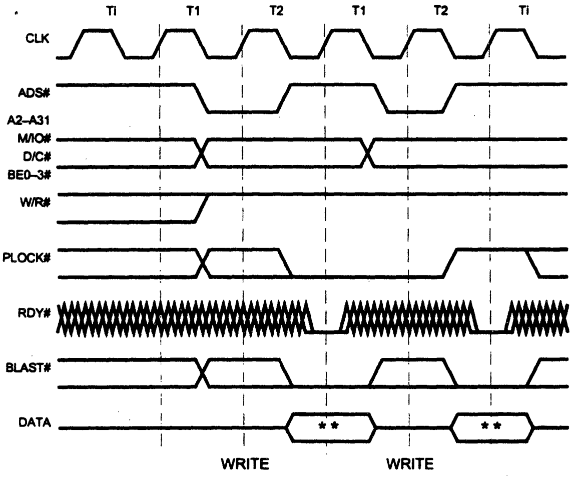
The first cycle of a 64-bit floating point write is the only case in which both PLOCK# and BLAST# are asserted. Normally PLOCK# and BLAST# are the inverse of each other.

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.



** From Processor

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.

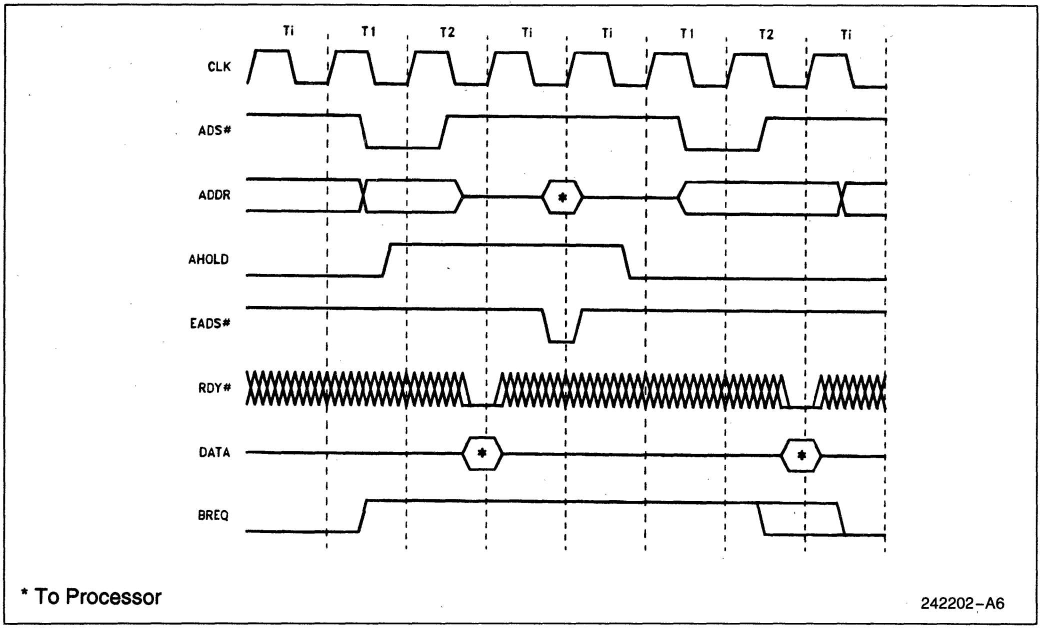
The Intel486 processor immediately relinquishes its address bus in the next clock upon assertion of AHOLD. For example, the bus could be 3 wait states into a read cycle. If AHOLD is activated, the Intel486 processor will immediately float its address bus before ready is returned terminating the bus cycle.

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.

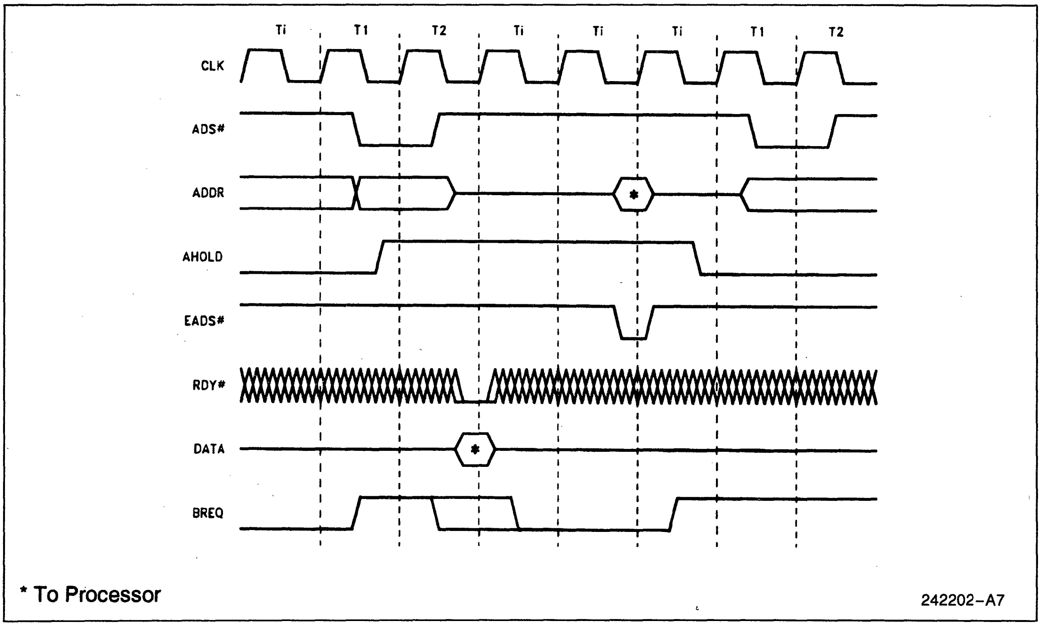
Running an invalidate cycle prevents the Intel486 processor cache from satisfying other internal requests, so invalidations should be run only when necessary. The fastest possible invalidate cycle is shown in Figure 10-22, while a more realistic invalidation cycle is shown in Figure 10-23. Both of the examples take one clock of cache access from the Intel486 processor.



* To Processor

242202-A6

Figure 10-22. Fast Internal Cache Invalidation Cycle



* To Processor

242202-A7

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.

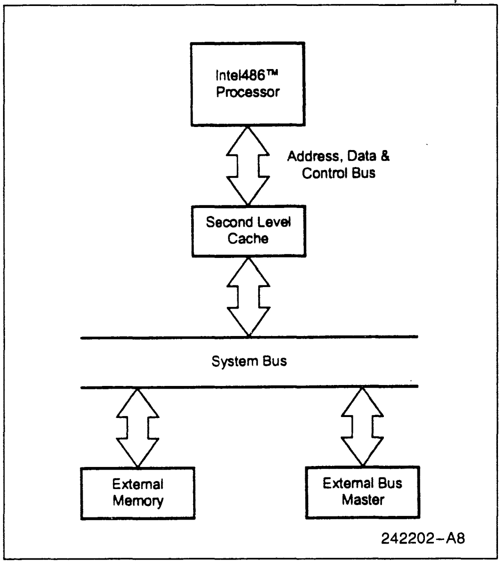


Figure 10-24. System with Second Level Cache

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.

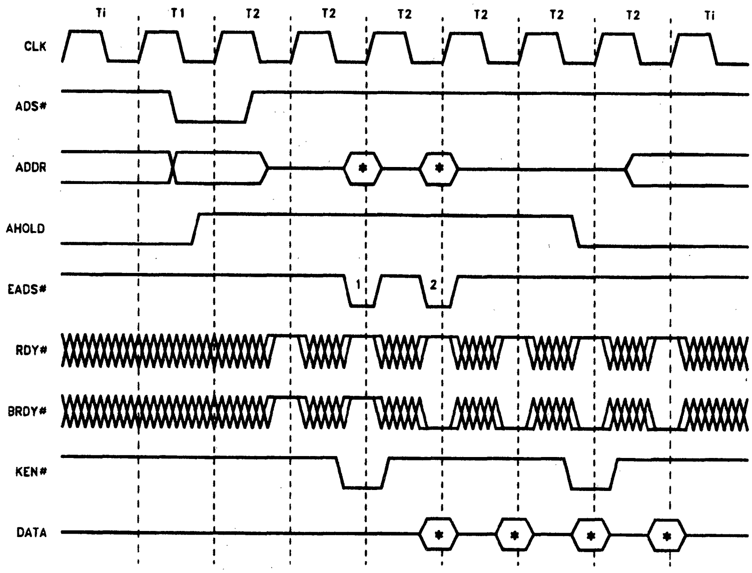
If the system asserts EADS# before the first data in the line fill is returned to the Intel486 processor, the system must return data consistent with the new data in the external memory upon resumption of the line fill after the invalidation cycle. This is illustrated by the asserted EADS# signal labeled 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.

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 un-aligned writes (less than or equal to 32-bits) with BLAST# being active for each write. For greater than 32-bit or un-aligned 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.



* To Processor

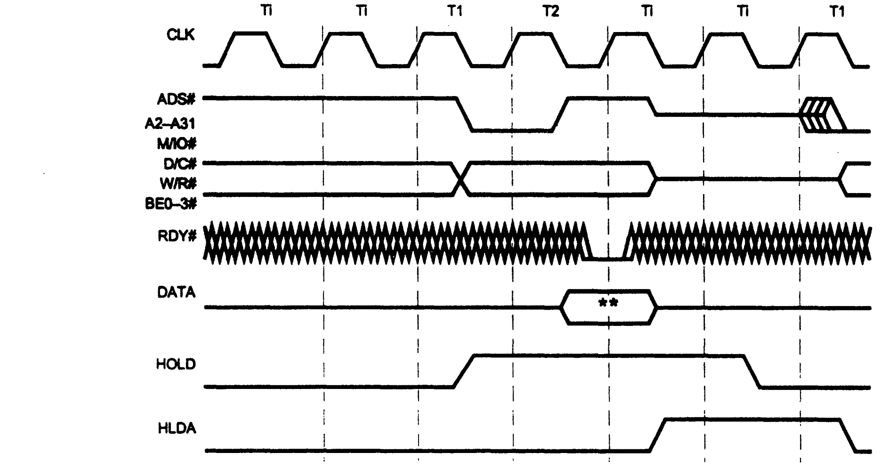
242202-A9

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

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.



** From Processor

Figure 10-26. HOLD/HLDA Cycles

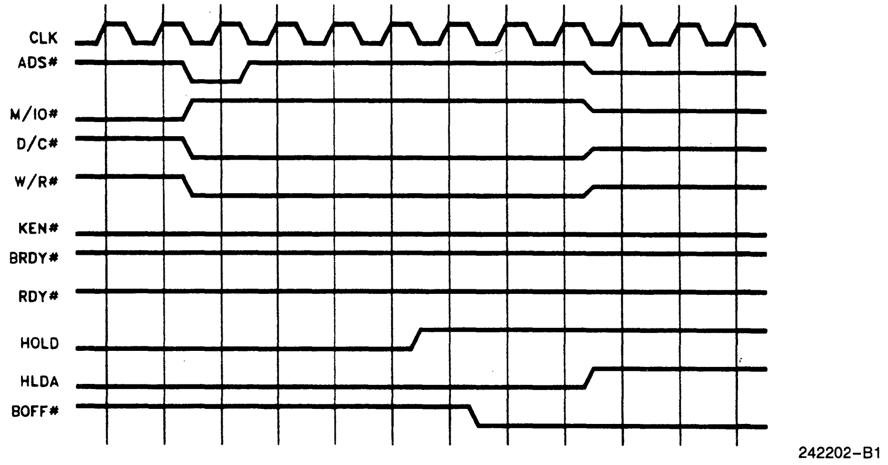


Figure 10-27. HOLD Request Acknowledged during BOFF#

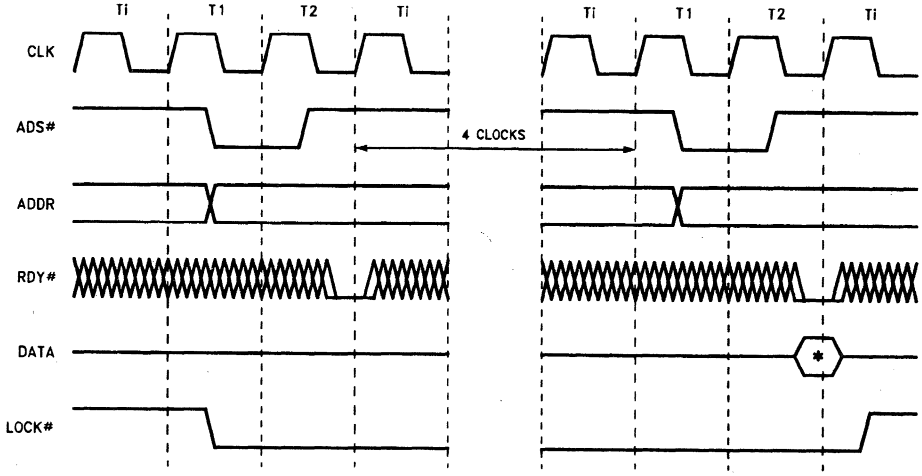
10.2.10 INTERRUPT ACKNOWLEDGE

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.



* To Processor

242202–B2

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.

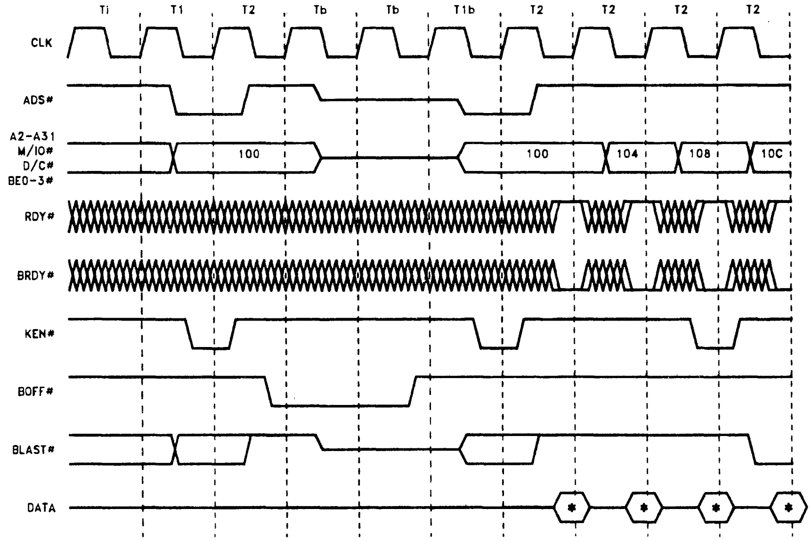
The external hardware must acknowledge these special bus cycles by returning RDY# or BRDY#.

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 IntelDX2 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.



* To Processor

242202-B3

Figure 10-29. Restarted Read 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.

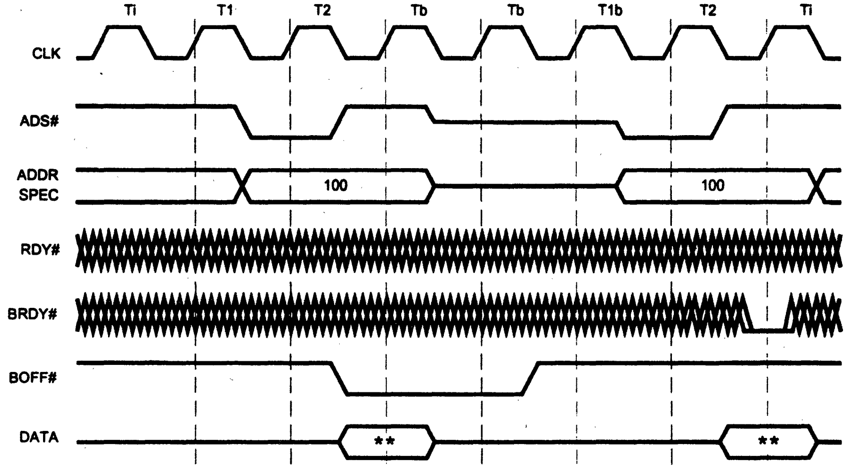
10.2.11.3 Stop Grant Indication Cycle

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.



242202–B4

** From Processor

Figure 10-30. Restarted Write Cycle

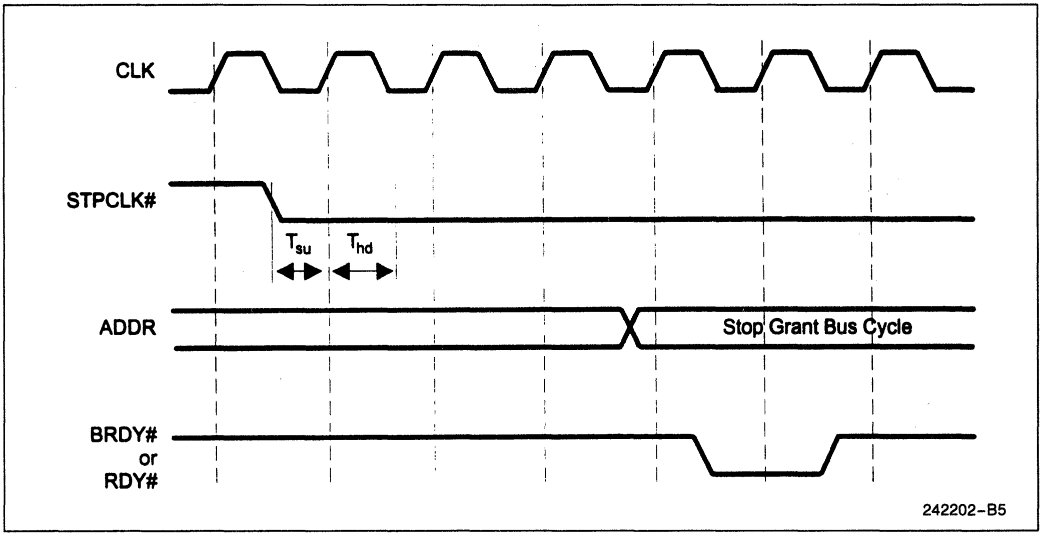


Figure 10-31. Stop Grant Bus Cycle

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 10-29 and 10-34). 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# are 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 insure 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.

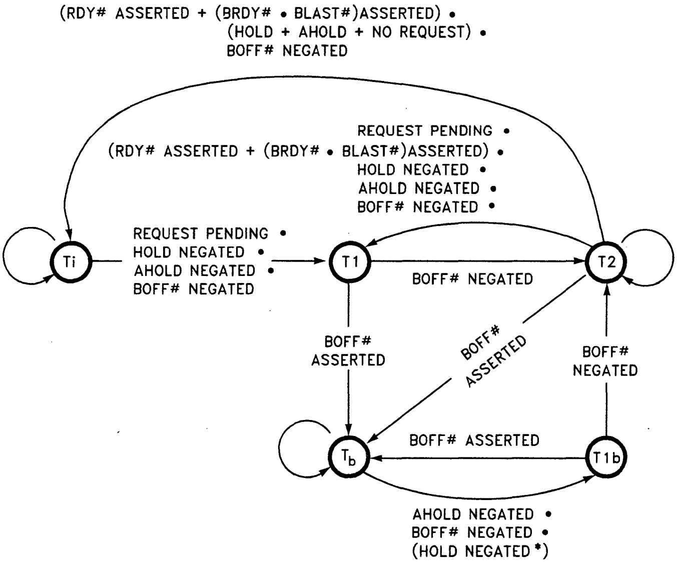
The bus remains in the high impedance state until BOFF# is negated. Upon negation, the Intel486 processor restarts its bus cycle by driving out the address and status and asserting ADS#. The bus cycle then continues as usual.

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.



* HOLD is only factored into this state transition if T_b was entered while a non-cacheable, non-bursted, code prefetch was in progress. Otherwise, ignore HOLD.

242202-B6

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.

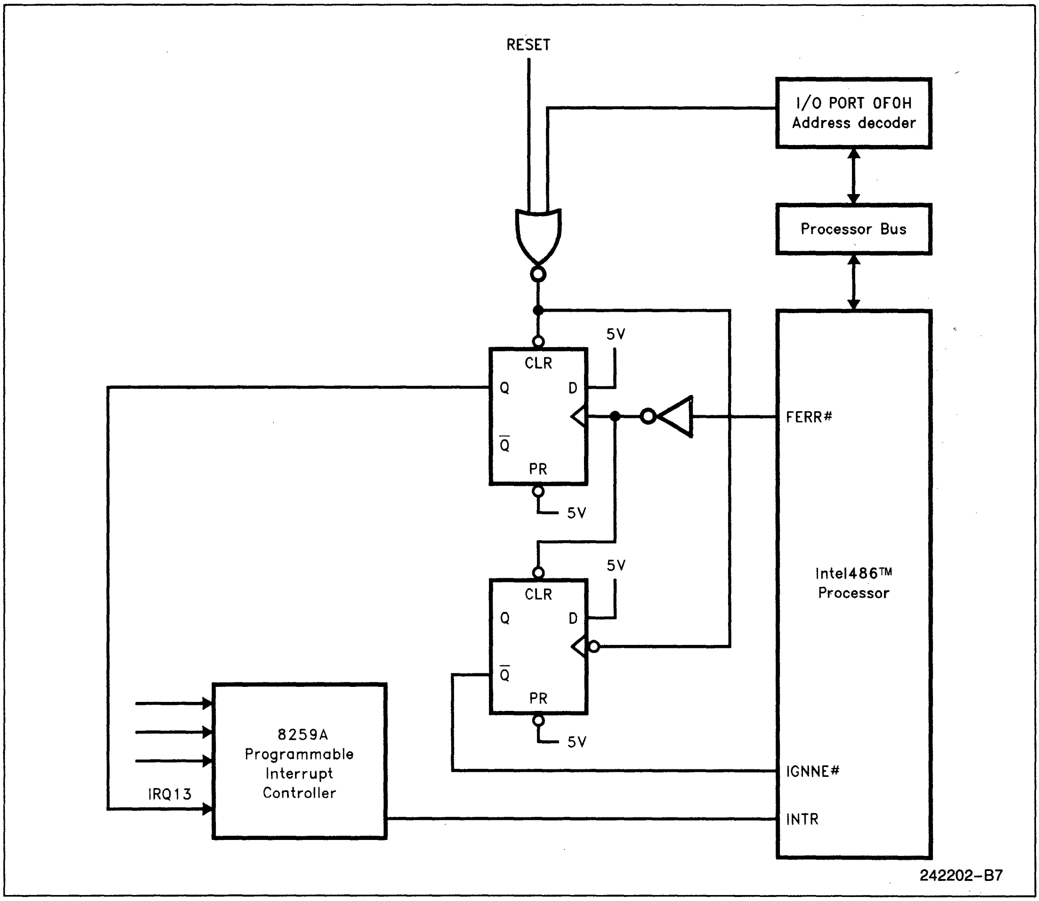


Figure 10-33. DOS-Compatible Numerics Error Circuit

10.3 Enhanced Bus Mode Operation (Write-Back Mode) for the Write-Back Enhanced IntelDX2 Processor

This section describes how the 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 doubleword 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 IntelDX2 processor. In the Enhanced Bus mode, both PCD and CACHE# are asserted if the cycle is internally cacheable. The Write-Back Enhanced IntelDX2 processor samples 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.

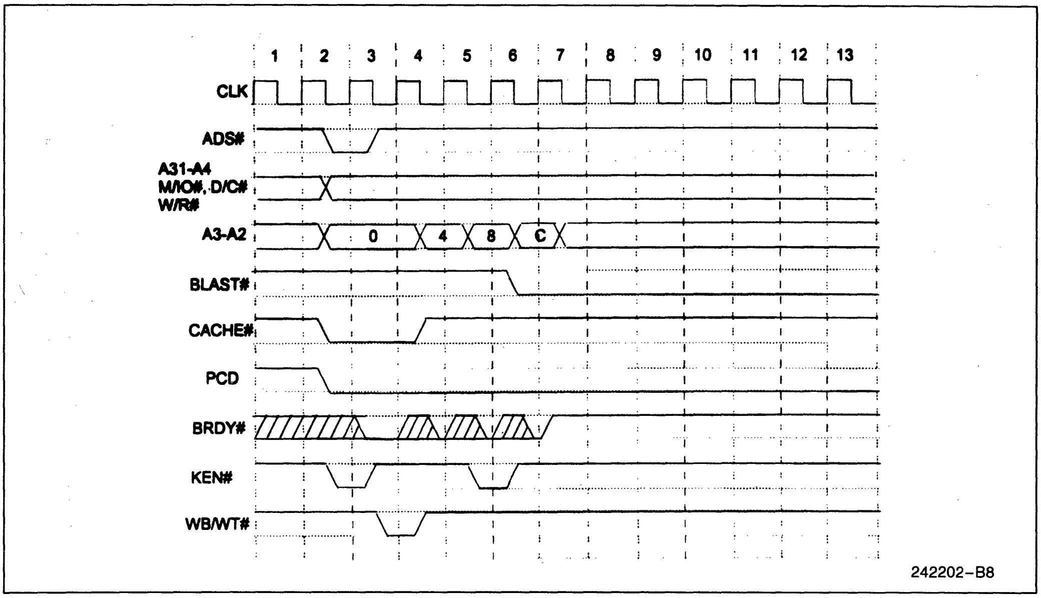


Figure 10-34. Basic Burst Read Cycle

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 IntelDX2 processor 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 noncacheable.

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 IntelDX2 processor 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 IntelDX2 processor is configured in Enhanced Bus mode, the system must drive INV high to invalidate a particular cache line. The Write-Back Enhanced IntelDX2 processor does 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 IntelDX2 processor 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 IntelDX2 processor asserts 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 IntelDX2 processor can accept EADS# in every clock period while in Standard Bus mode. In Enhanced Bus mode, the Write-Back Enhanced IntelDX2 processor can accept EADS# every other clock period or until a snoop hits an M-state line.** The Write-Back Enhanced IntelDX2 processor 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 IntelDX2 processor 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.			

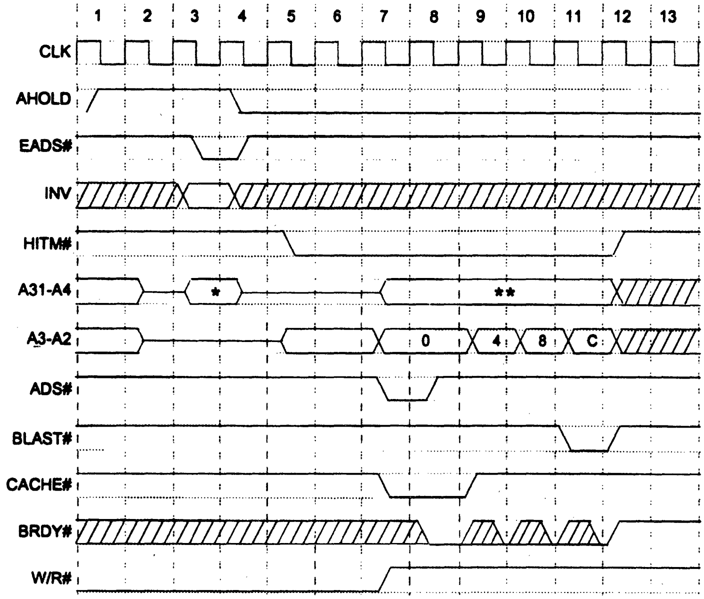
10.3.3.2 Snoop under AHOLD

Snooping under AHOLD begins by asserting AHOLD to force the Write-Back Enhanced IntelDX2 processor 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 driven next ADS# for a new cycle can occur immediately after the last RDY# or BRDY# of the write-back cycle. The Write-Back Enhanced IntelDX2 processor does 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 doubleword 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.



242202-B9

* To Processor
** Write-back from Processor

Figure 10-35. Snoop Cycle Invalidating a Modified Line

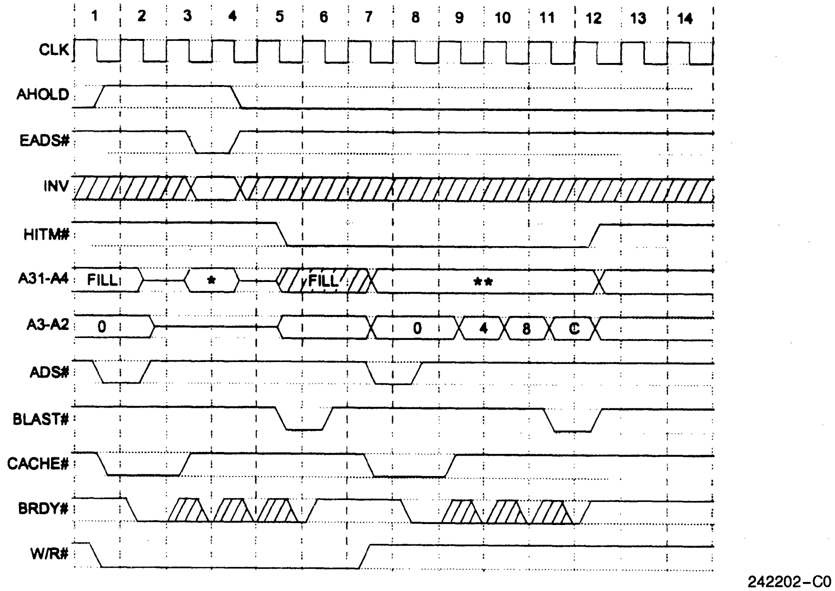
Snoop under AHOLD Overlaying a Line-Fill Cycle

The assertion of AHOLD during a line fill is allowed on the Write-Back Enhanced IntelDX2 processor. In this case, when a snoop cycle is overlaid by an ongoing 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 IntelDX2 processor 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 IntelDX2 processor 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 IntelDX2 processor. 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 IntelDX2 processor always runs 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.



* To Processor
** Write-back from Processor

Figure 10-36. Snoop Cycle Overlaying a Line-Fill Cycle

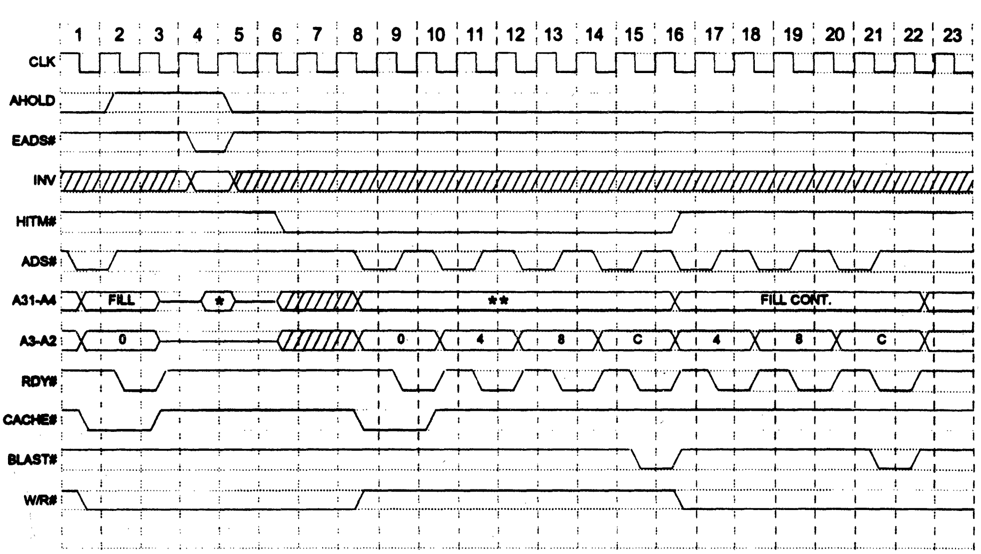
AHOLD Snoop Overlaying a Non-Burst Cycle

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 on-going 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.



242202-C1

* To Processor

** Write-back from Processor

Figure 10-37. Snoop Cycle Overlaying a Non-Burst Cycle

Snoop during Replacement Write-Back

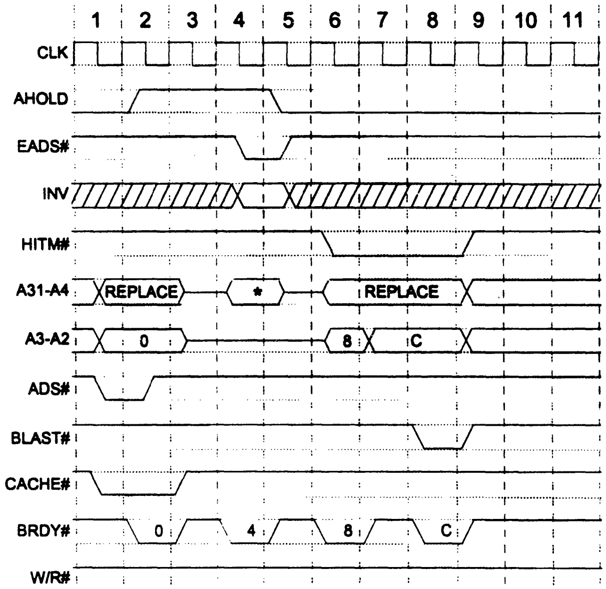
If the cache contains valid data during a line fill, one of the cache lines may be replaced as determined by the LRU algorithm. If the line being replaced is modified, this line will be written back to maintain cache coherency. When a replacement write-back cycle is in progress, it might be necessary 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.



* To Processor

Figure 10-38. Snoop to the Line That Is Being Replaced

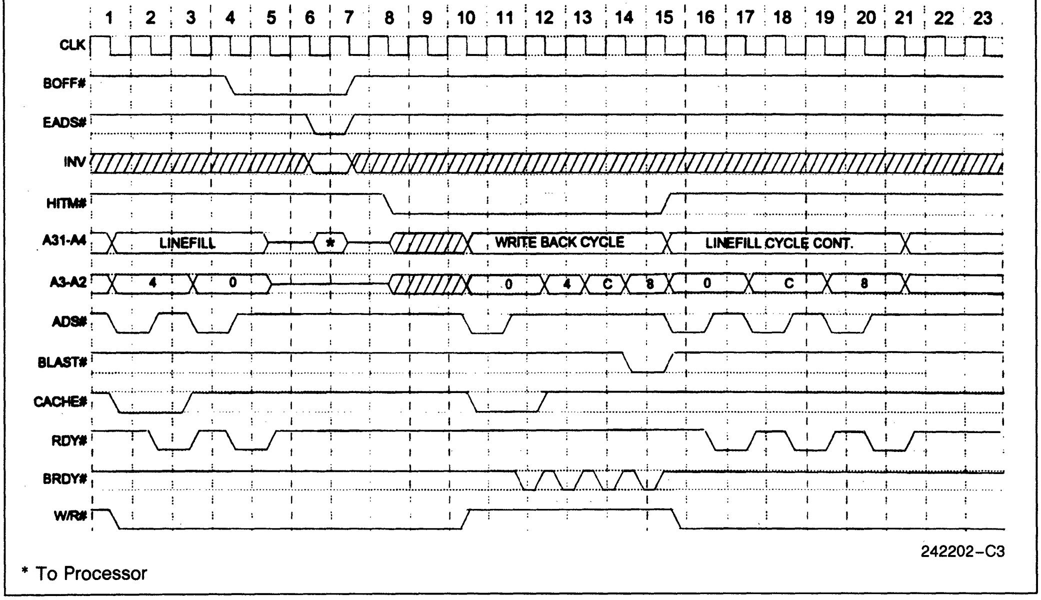
10.3.3.3 Snoop under BOFF#

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 IntelDX2 processor 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.



* To Processor

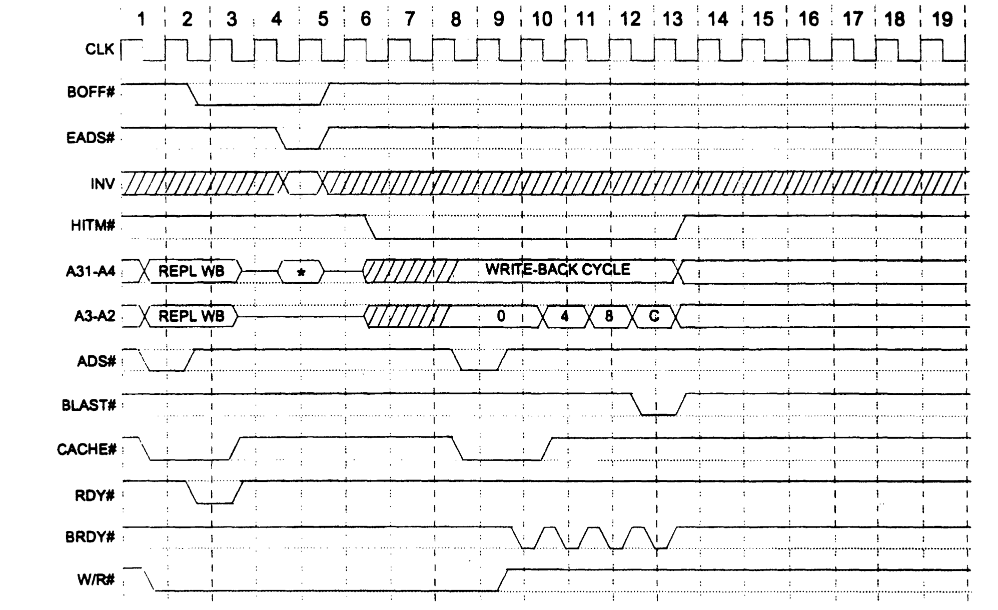
Figure 10-39. Snoop under BOFF# during a Cache Line-Fill Cycle

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 IntelDX2 processor accepts 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 IntelDX2 processor 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).



* To Processor

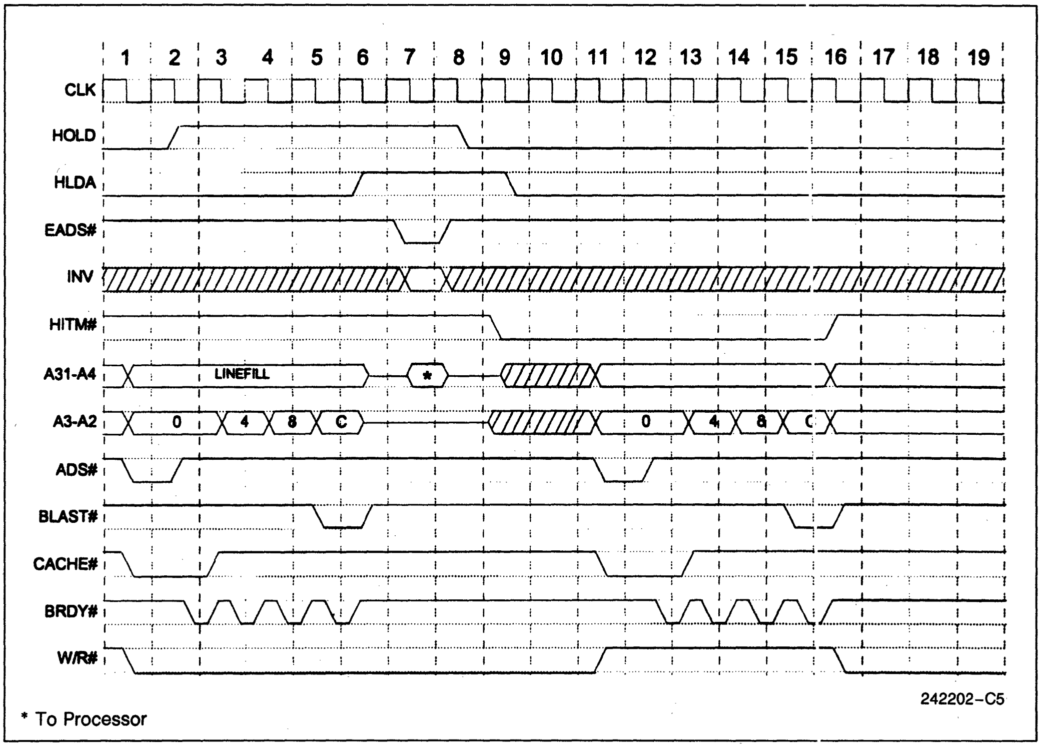
Figure 10-40. Snoop under BOFF# to the Line that is Being Replaced

10.3.3.4 Snoop under HOLD

HOLD can only fracture a non-cacheable, non-bursted code prefetch cycle. For all other cycles, the Write-Back Enhanced IntelDX2 processor will not assert HLDA until the entire current cycle is completed. 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 reordered 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-asserted. Consequently, multiple snoop cycles are permitted under a continuously asserted HLDA only up to the first asserted HITM#.

Snoop under HOLD durir g Cache Line Fill

As shown in Figure 10-41, HOLD (asserted in clock two) does not fracture the ɔursted cache line-fill cycle until the line fill is completed (in clock five). Upon completing the line fill in c ock five, the Write-Back Enhanced IntelDX2 proce:ssor asserts HLDA and the system begins snoopinȝ 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 snooɔ write-back cycle. The snoop write-back cycle be:ɉins upon the de-assertion of HOLD, and HITM# is asserted throughout the duration of the snoop ʍrite-back cycle.



* To Processor

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 IntelDX2 processor 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.

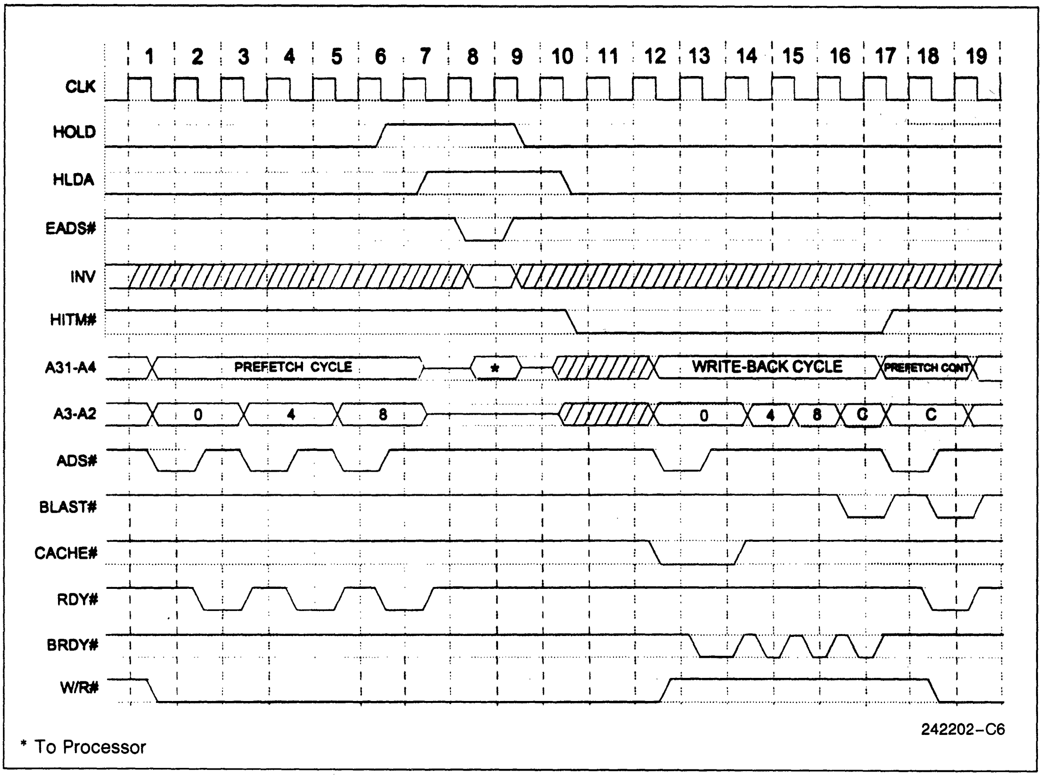


Figure 10-42. Snoop using HOLD during a Non-Cacheable, Non-Burstable Code Prefetch

Snoop under HOLD during Replacement Write-Back

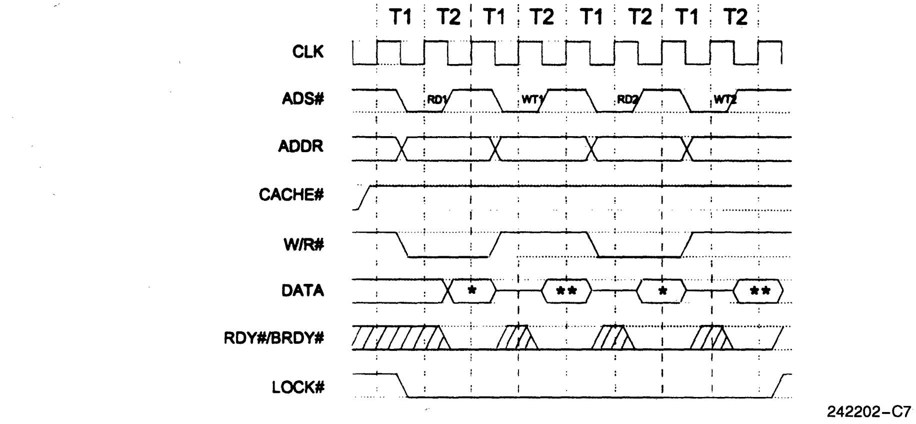
Collision of snoop cycles under a HOLD during the replacement write-back cycle can never occur, because HLDA is asserted only after the replacement write-back cycle (bursted or non-bursted) is completed.

10.3.4 LOCKED CYCLES

In both Standard and Enhanced Bus modes, the Write-Back Enhanced IntelDX2 processor 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 IntelDX2 processor does 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.



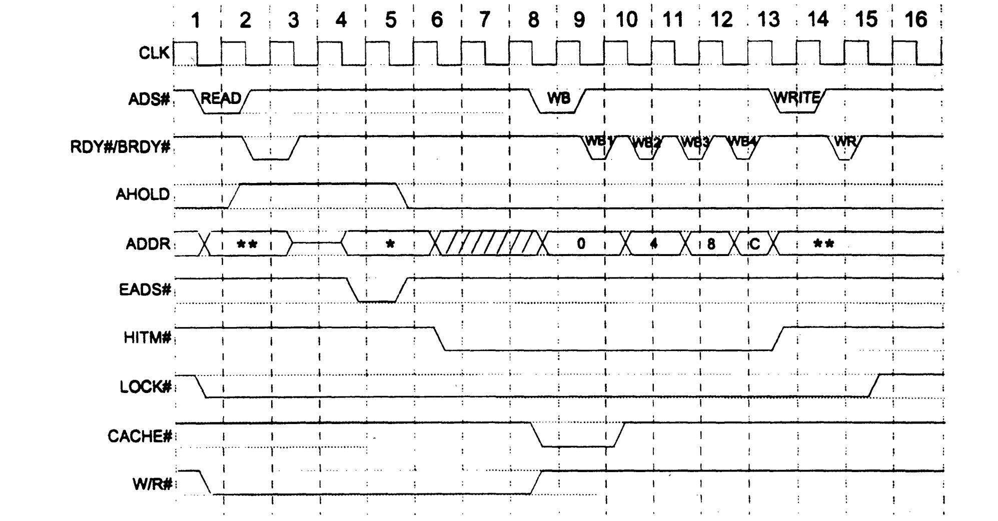
* To Processor
** From Processor

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.



242202-C8

* To Processor
** From Processor

Figure 10-44. Snoop Cycle Overlaying a Locked Cycle

10.3.5 FLUSH OPERATION

The Write-Back Enhanced IntelDX2 processor executes 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 IntelDX2 processor, 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.

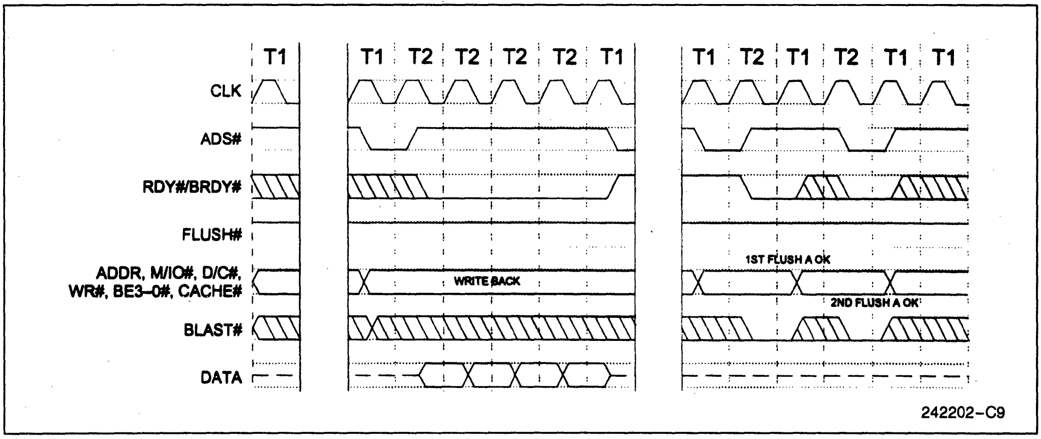


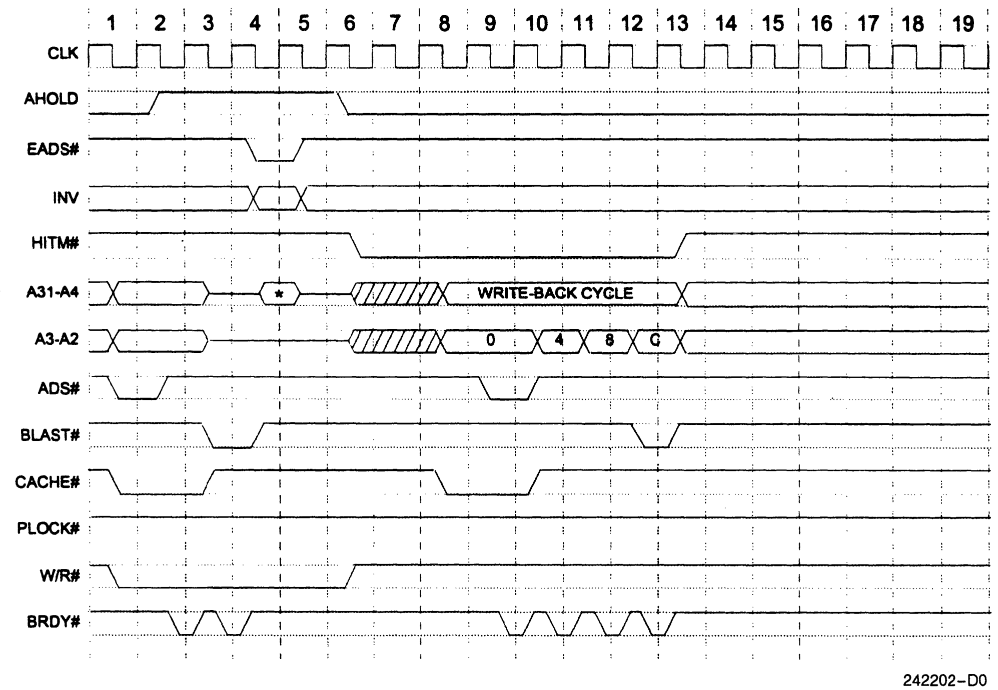
Figure 10-45. Flush Cycle

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.

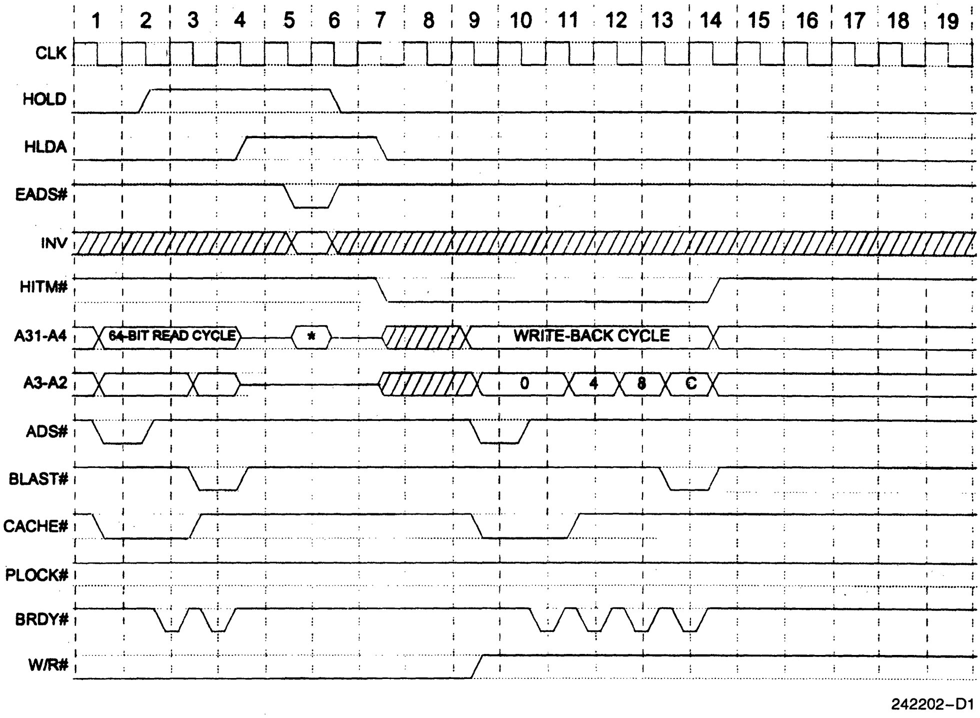


* To Processor

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 IntelDX2 processor will not issue HLDA until clock four. After the 64-bit transfer is completed, the Write-Back Enhanced IntelDX2 processor 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 IntelDX2 processor can issue HLDA in between bus cycles for a 64-bit transfer.



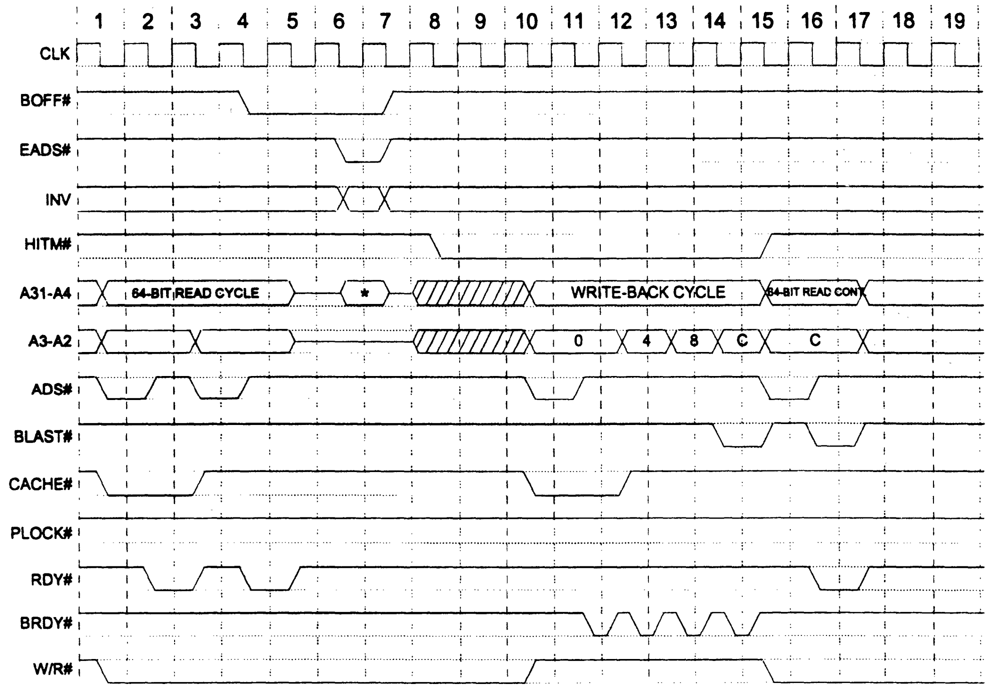
242202-D1

* To Processor

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.



* To Processor

242202-D2

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

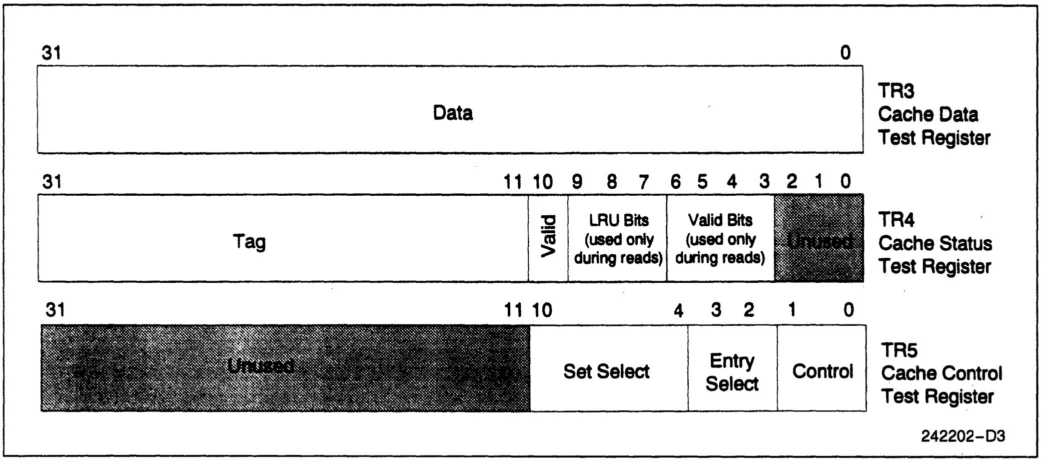


Figure 11-1. Cache Test Registers (All Intel486™ Processors Except the IntelDX4™ Processor)

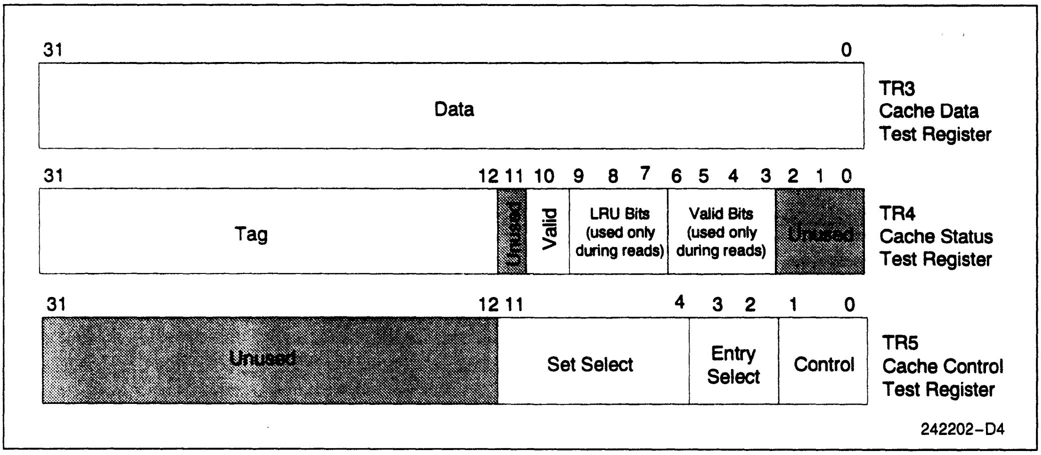


Figure 11-2. IntelDX4™ Processor Cache Test Registers

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 is 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 a seven-bit select field. All other Intel486 processors have an eight-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 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 (Writeback and Invalidate Data Cache) instruction.

11.2.6 ADDITIONAL CACHE TESTING FEATURES FOR ENHANCED BUS (WRITE-BACK) MODE

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 IntelDX2 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.

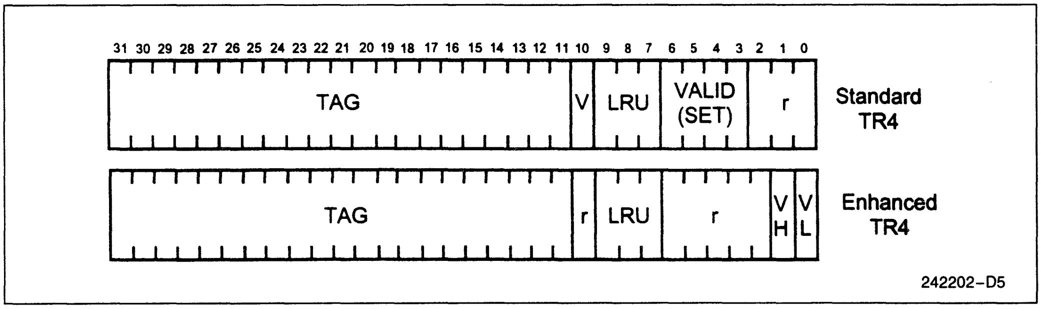


Figure 11-3. TR4 Definition for Standard and Enhanced Bus Modes for the Write-Back Enhanced IntelDX2™Processor

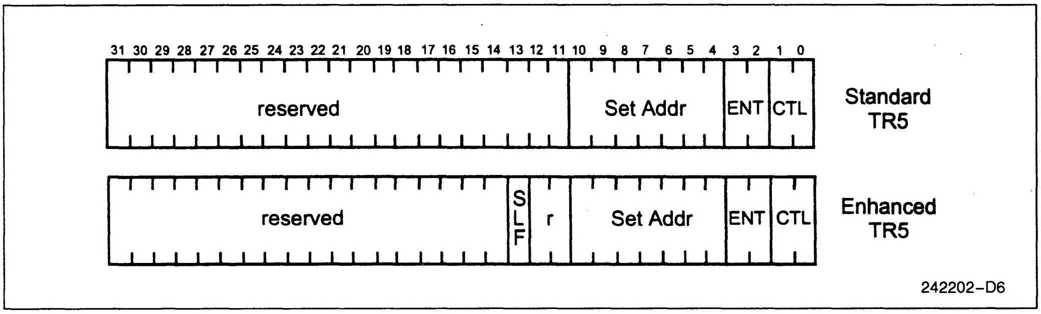


Figure 11-4. TR5 Definition for Standard and Enhanced Bus Modes for the Write-Back Enhanced IntelDX2™ Processor

11.3 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 11-5.

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.

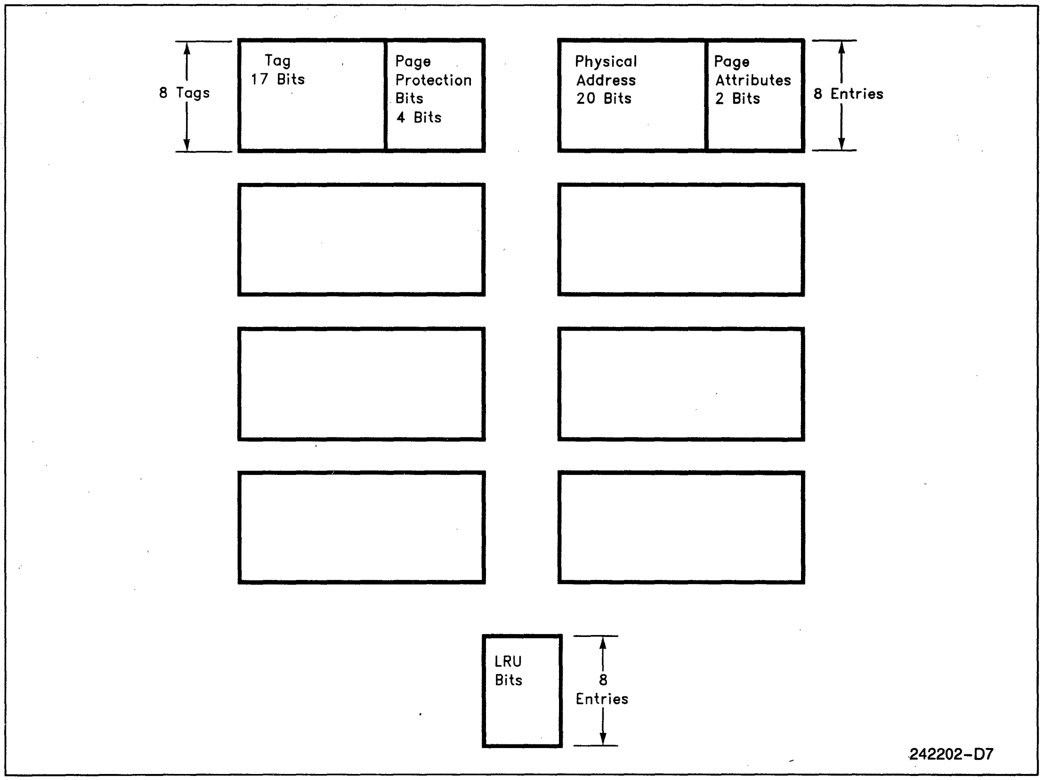


Figure 11-5. TLB Organization

11.3.2 TLB TEST REGISTERS TR6 AND TR7

The two TLB test registers are shown in Figure 11-6. 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.

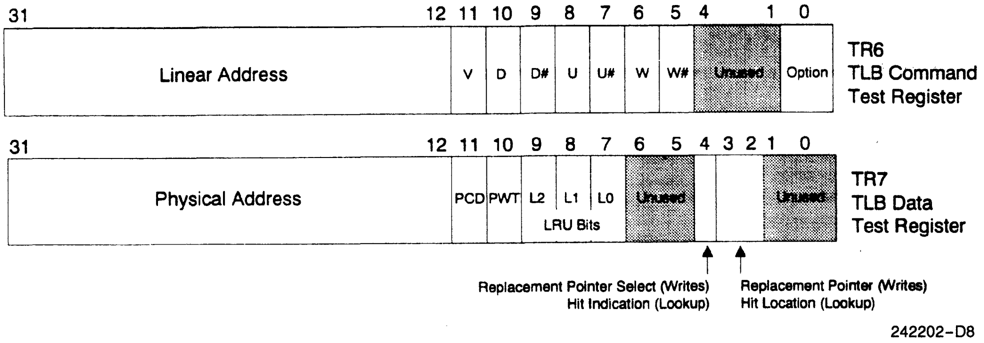


Figure 11-6. TLB Test Registers

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 11-4.

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 11-5.

Table 11-4. 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 11-5. 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 11-6 and Table 11-7. Finally, TR7 contains two bits (bits 2–3) to specify a TLB replacement pointer or the location of a TLB hit.

Table 11-6. 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 11-6.

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 11-7. 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 11-4.

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 11-7).

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 11-7 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."

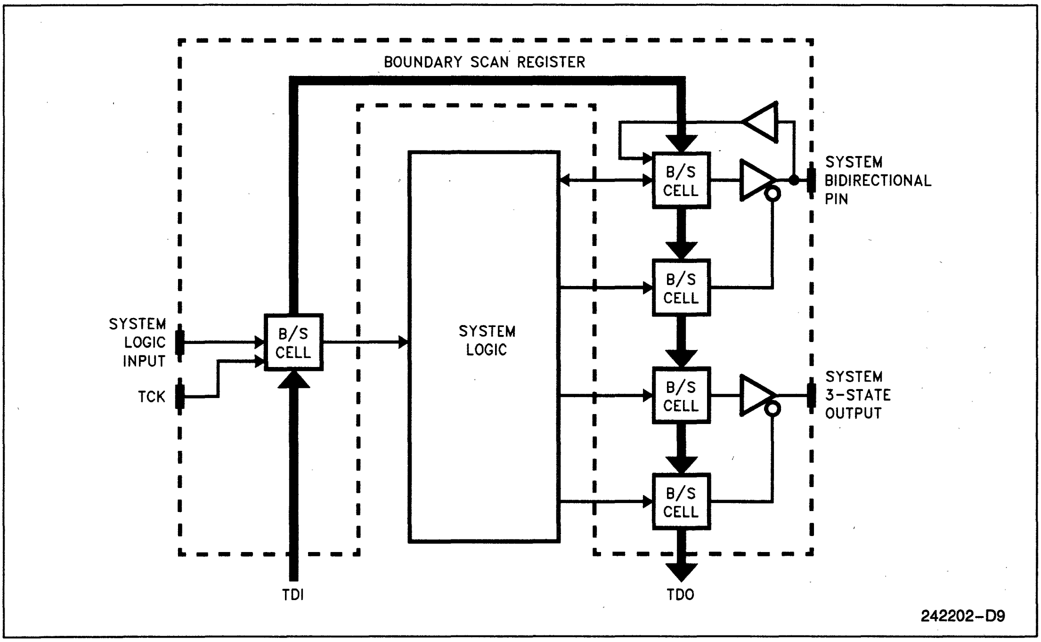


Figure 11-7. Logical Structure of Boundary Scan Register

11.5.2.1 Device Identification Register

The Device Identification Register contains the manufacturer's identification code, part number code, and version code. Table 11-8 lists the codes corresponding to the Intel486 processor.

11.5.2.2 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 11-9 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 11-8. 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

NOTE:
*Contact Intel for details

Table 11-9. 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.

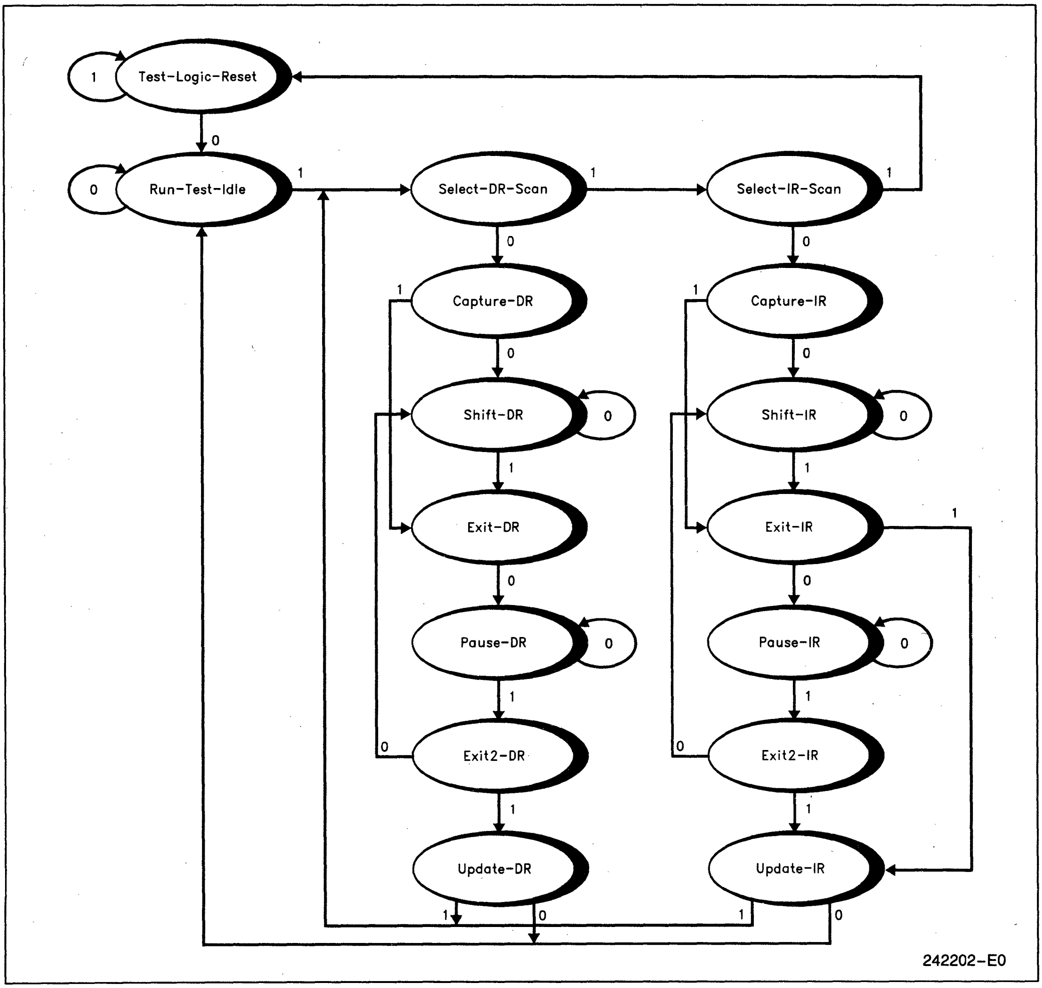


Figure 11-8. TAP Controller State Diagram

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 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

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.

11.5.6 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.

11.5.7 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.

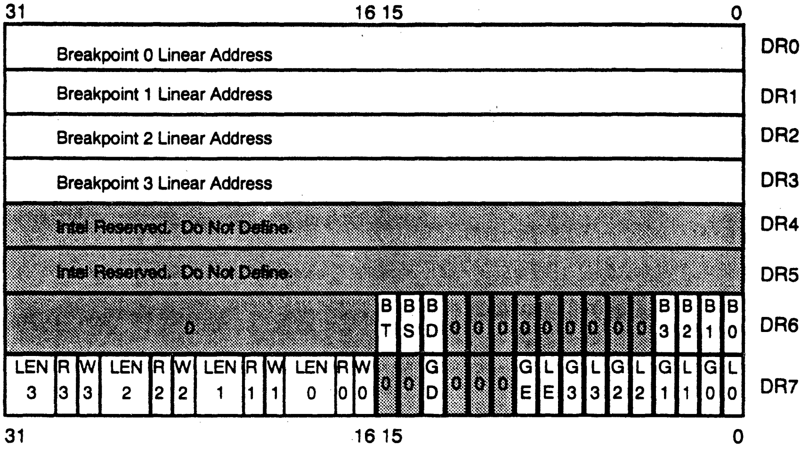
The Intel486 processor contains six Debug Registers, providing the ability to specify up to four distinct breakpoints addresses, breakpoint control options, and read breakpoint status. Initially after reset, breakpoints are in the disabled state. Therefore, no breakpoints will occur unless the debug 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:


31
16 15
0
Breakpoint 0 Linear Address
DR0
Breakpoint 1 Linear Address
DR1
Breakpoint 2 Linear Address
DR2
Breakpoint 3 Linear Address
DR3
Intel Reserved. Do Not Define.
DR4
Intel Reserved. Do Not Define.
DR5
0
BT BS BD 0 0 0 0 0 0 0 0 0 B3 B2 B1 B0
DR6
LEN 3 R3 W3 LEN 2 R2 W2 LEN 1 R1 W1 LEN 0 R0 W0 0 0 GD 0 0 0 GE LE G3 L3 G2 L2 G1 L1 G0 L0
DR7
31
16 15
0

0 indicates Intel Reserved; Do not define.

NOTE:
See section 4.2.7

Figure 12-1. Debug Registers

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.

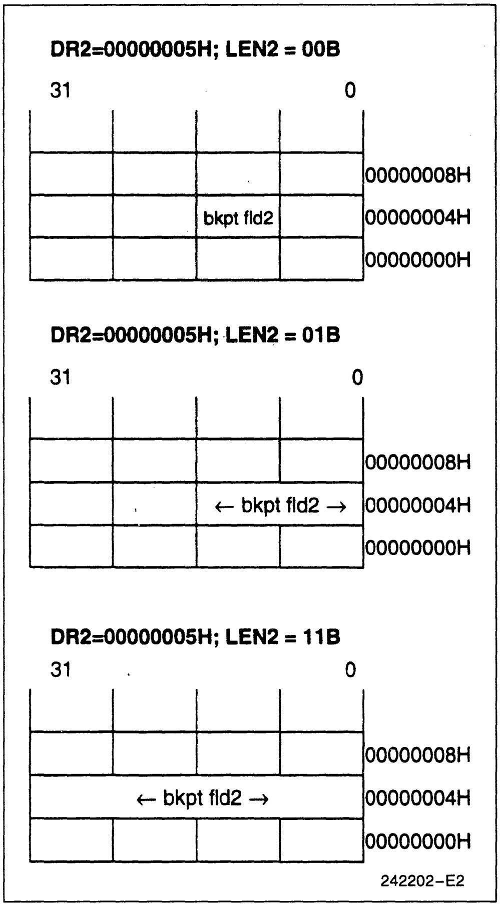


Figure 12-2. Size Breakpoint Fields

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.

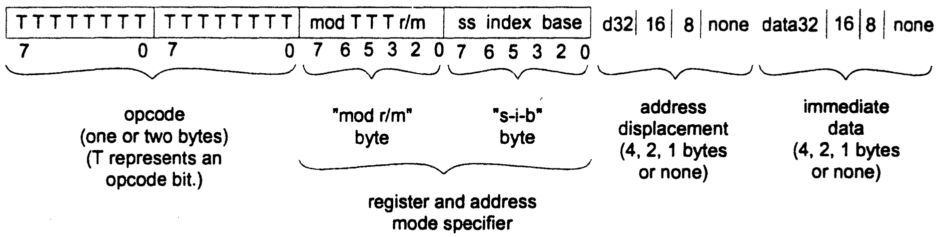

TTTTTTTT
TTTTTTTT
mod TTT r/m
ss index base
d32 | 16 | 8 | none
data32 | 16 | 8 | none
7 0 7 0 7 6 5 3 2 0 7 6 5 3 2 0
opcode (one or two bytes) (T represents an opcode bit.)
"mod r/m" byte
"s-i-b" byte
register and address mode specifier
address displacement (4, 2, 1 bytes or none)
immediate data (4, 2, 1 bytes or none)
242202-E3

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 immediately ahead.

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	mod r/m	Effective Address
00 000	DS:[BX+SI]	10 000	DS:[BX+SI+d16]
00 001	DS:[BX+DI]	10 001	DS:[BX+DI+d16]
00 010	SS:[BP+SI]	10 010	SS:[BP+SI+d16]
00 011	SS:[BP+DI]	10 011	SS:[BP+DI+d16]
00 100	DS:[SI]	10 100	DS:[SI+d16]
00 101	DS:[DI]	10 101	DS:[DI+d16]
00 110	DS:d16	10 110	SS:[BP+d16]
00 111	DS:[BX]	10 111	DS:[BX+d16]
01 000	DS:[BX+SI+d8]	11 000	register—see below
01 001	DS:[BX+DI+d8]	11 001	register—see below
01 010	SS:[BP+SI+d8]	11 010	register—see below
01 011	SS:[BP+DI+d8]	11 011	register—see below
01 100	DS:[SI+d8]	11 100	register—see below
01 101	DS:[DI+d8]	11 101	register—see below
01 110	SS:[BP+d8]	11 110	register—see below
01 111	DS:[BX+d8]	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	mod r/m	Effective Address
00 000	DS:[EAX]	10 000	DS:[EAX+d32]
00 001	DS:[ECX]	10 001	DS:[ECX+d32]
00 010	DS:[EDX]	10 010	DS:[EDX+d32]
00 011	DS:[EBX]	10 011	DS:[EBX+d32]
00 100	s-i-b is present	10 100	s-i-b is present
00 101	DS:d32	10 101	SS:[EBP+d32]
00 110	DS:[ESI]	10 110	DS:[ESI+d32]
00 111	DS:[EDI]	10 111	DS:[EDI+d32]
01 000	DS:[EAX+d8]	11 000	register—see below
01 001	DS:[ECX+d8]	11 001	register—see below
01 010	DS:[EDX+d8]	11 010	register—see below
01 011	DS:[EBX+d8]	11 011	register—see below
01 100	s-i-b is present	11 100	register—see below
01 101	SS:[EBP+d8]	11 101	register—see below
01 110	DS:[ESI+d8]	11 110	register—see below
01 111	DS:[EDI+d8]	11 111	register—see below

Register Specified by reg or r/m during 16-Bit Data Operations:			Register Specified by reg or r/m during 32-Bit Data Operations:		
mod r/m	Function of w Field		mod r/m	Function of w Field	
	(when w=0)	(when w=1)		(when w=0)	(when w = 1)
11 000	AL	AX	11 000	AL	EAX
11 001	CL	CX	11 001	CL	ECX
11 010	DL	DX	11 010	DL	EDX
11 011	BL	BX	11 011	BL	EBX
11 100	AH	SP	11 100	AH	ESP
11 101	CH	BP	11 101	CH	EBP
11 110	DH	SI	11 110	DH	ESI
11 111	BH	DI	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	

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				
Jcccc = 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				
SETcccc = 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

Instruction	Format	Cache Hit	Penalty if Cache Miss	Notes
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	

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

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 (EFLAGS.NT=1)		TS+32	4	9,10
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			

Table 13-19. Floating Point Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
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			

2

Table 13-19. Floating Point Clock Count Summary (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
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)	

Table 13-19. **Floating Point Clock Count Summary** (Continued)

Instruction	Format	Cache Hit	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
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	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
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	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
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	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
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	Penalty if Cache Miss	Concurrent Execution	Notes
		Avg (Lower Range . . . Upper Range)		Avg (Lower Range . . . Upper Range)	
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/				
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 $n = (\text{operand}/(\pi/4))$.

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 Processor
75	NC	STPCLK#	NC	STPCLK#
77	NC	NC	IGNNE#	IGNNE#
81	NC	NC	FERR#	FERR#
85	NC	SMI#	NC	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.

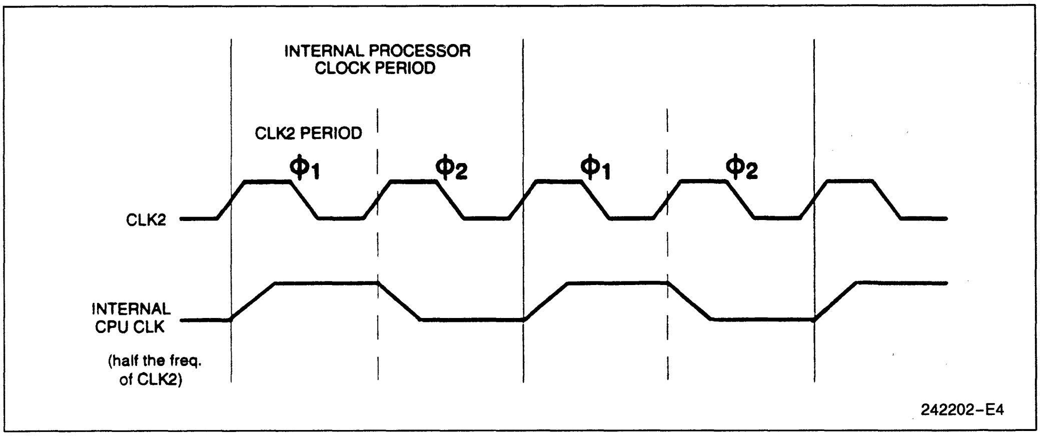


Figure 15-1. CLK2 Signal and Internal Processor Clock

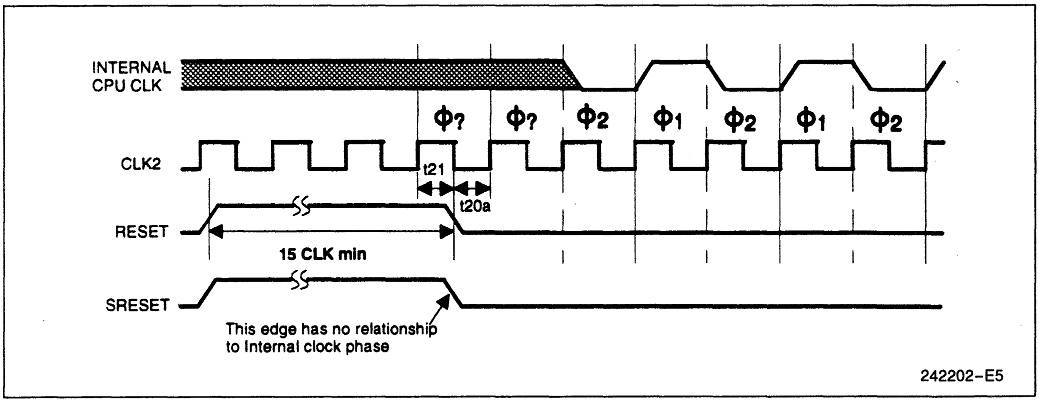


Figure 15- 2. CLK2 and Internal Processor CLK vs. SRESET and RESET Timings

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.

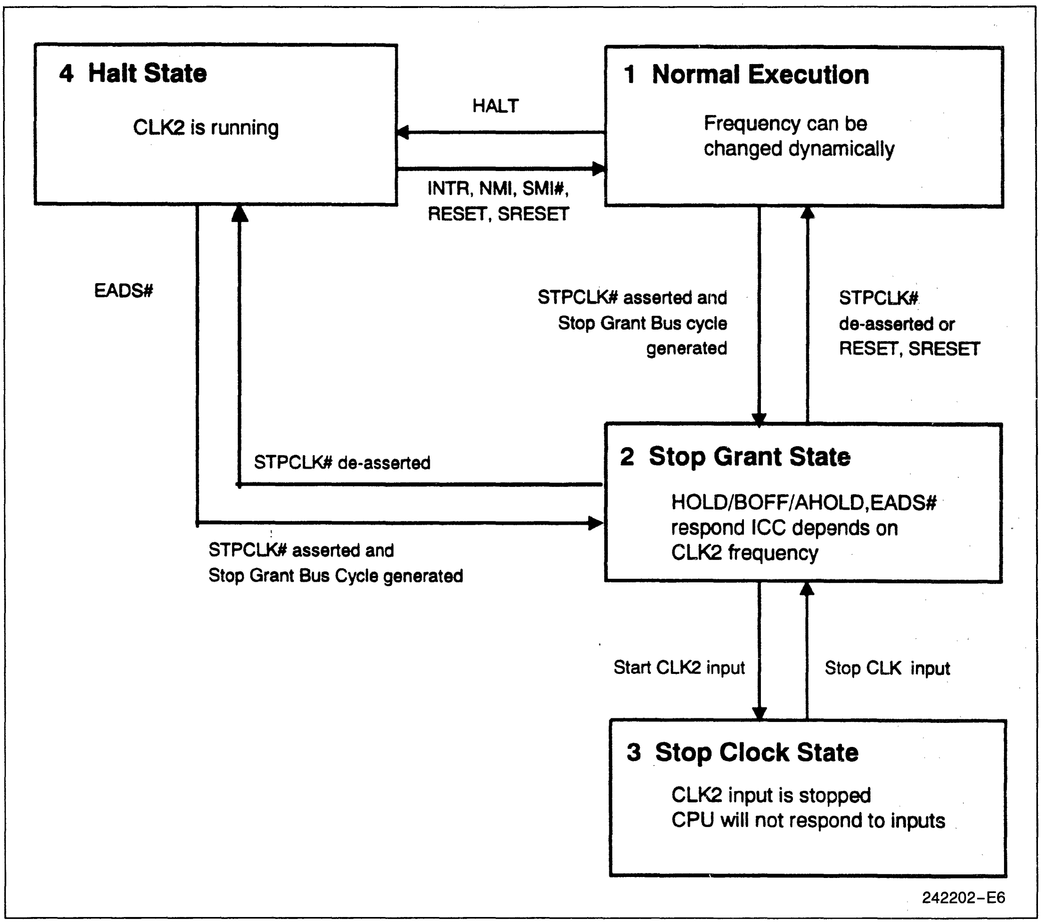

4 Halt State
CLK2 is running
1 Normal Execution
Frequency can be changed dynamically
HALT
INTR, NMI, SMI#, RESET, SRESET
EADS#
STPCLK# asserted and Stop Grant Bus cycle generated
STPCLK# de-asserted or RESET, SRESET
STPCLK# de-asserted
STPCLK# asserted and Stop Grant Bus Cycle generated
2 Stop Grant State
HOLD/BOFF/AHOLD,EADS# respond ICC depends on CLK2 frequency
Start CLK2 input
Stop CLK input
3 Stop Clock State
CLK2 input is stopped
CPU will not respond to inputs
242202-E6

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.

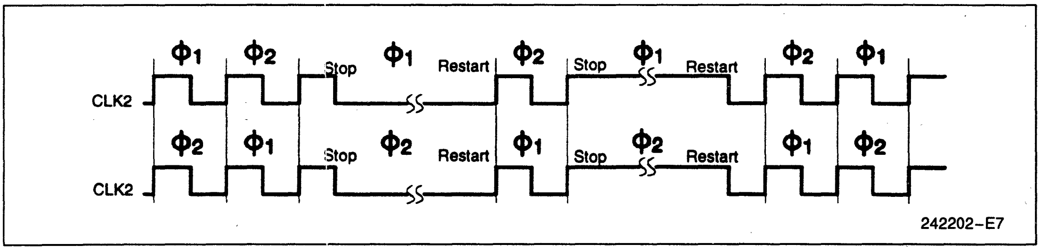


Figure 15-4. CLK2 Phase Coherence in CLK2 Stop and Restart

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.

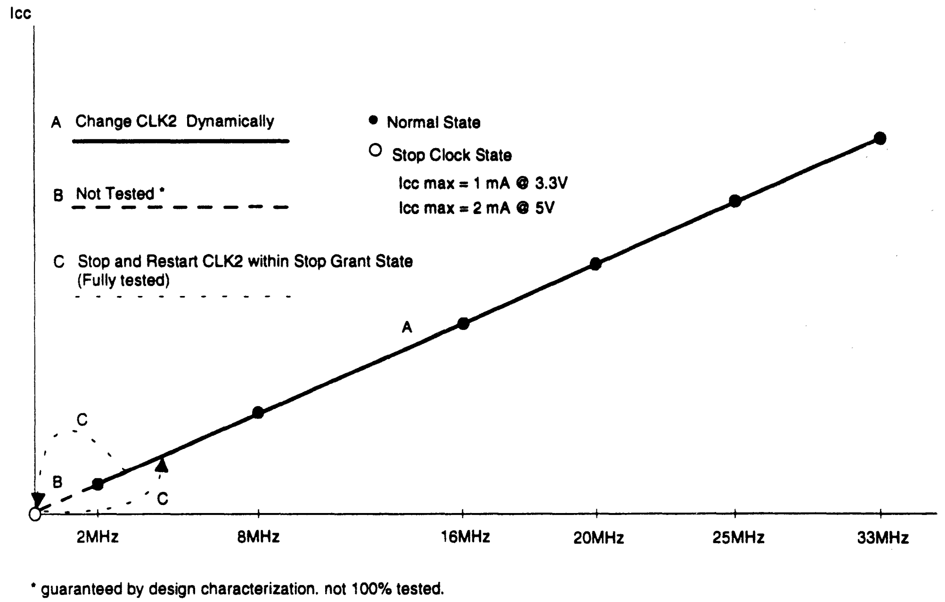


Figure 15-5. Supply Current Model for Stop Clock Modes and Transitions

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.)

15.1.15.1 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	1
	CLK2 Frequency		**50**		**66**	MHz	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	3
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA Float Delay		28		21	ns	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	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	2
t_{12}	EADS# Setup Time	9		6		ns	

Table 15-3. 3.3V AC Characteristics (2X Clock) (Continued)

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	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	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	1
	CLK2 Frequency		**50**		**66**	MHz	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	3
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, BREQ, HLDA Float Delay		28		21	ns	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	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	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	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	3
t_{22}	D0-D31, DP0-DP3, A4-A31 Read Setup Time	6		6		ns	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.

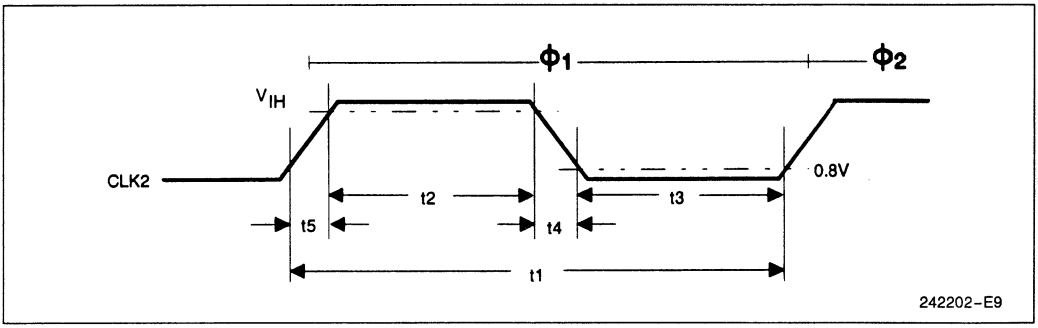
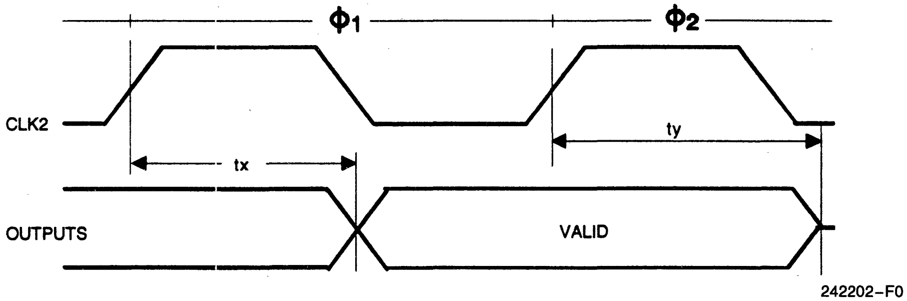
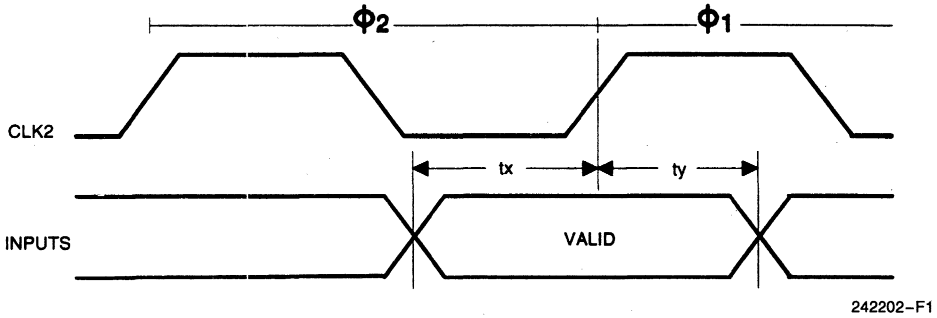


Figure 15-6. CLK2 Waveform



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 IntelSX2 and IntelDX2 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 in the Write-Back Enhanced IntelDX2 processor. 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 math coprocessor 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 Math Coprocessor 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 IntelSX2 and IntelDX2 OverDrive

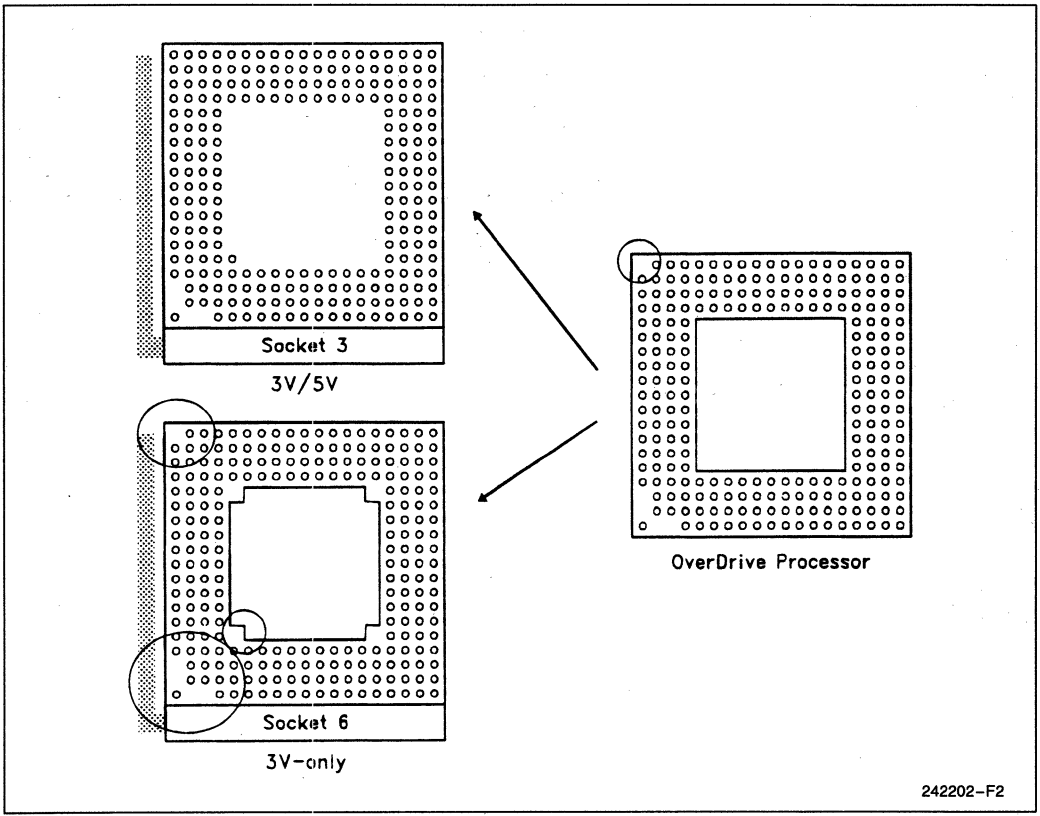


Figure 16-1. OverDrive™ Processor Sockets for Intel486™ Processor-Based Systems

processors. 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 IntelSX2 and IntelDX2 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,". Because the OverDrive processor must function in the OverDrive processor socket, the OverDrive processor socket heat dissipation specifications must be implemented. 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. Sections and 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, the pin C14 defines FERR# and the pin A13 is INC, while on the OverDrive processor socket the pin D15 is INC and the pin B14 defines FERR#.
- On Intel486 processors the 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 the "pin 1" (notched corner) oriented towards the "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, the pins D15 and B14 of the OverDrive processor socket should be connected together to the FERR# signal line, and the pins F19 (INIT) and D11 should be connected together to SRESET. When the OverDrive processor is installed, the pin C15 (UP#) must be isolated from TMS (see Figure 16-2).

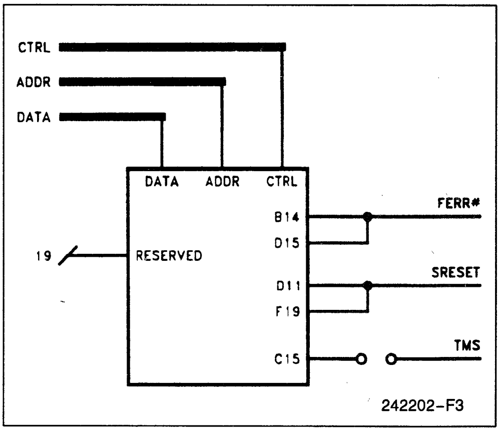


Figure 16-2. OverDrive™ Processor Socket Circuit Diagram-Single Socket Design

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.

The Pentium OverDrive processor socket has a fourth row of contacts around the outside of the 169 contacts defined for the IntelSX2 and IntelDX2 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 four corner pins removed. 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 IntelSX2 and IntelDX2 OverDrive processors.

16.3.2 DESIGN RECOMMENDATIONS

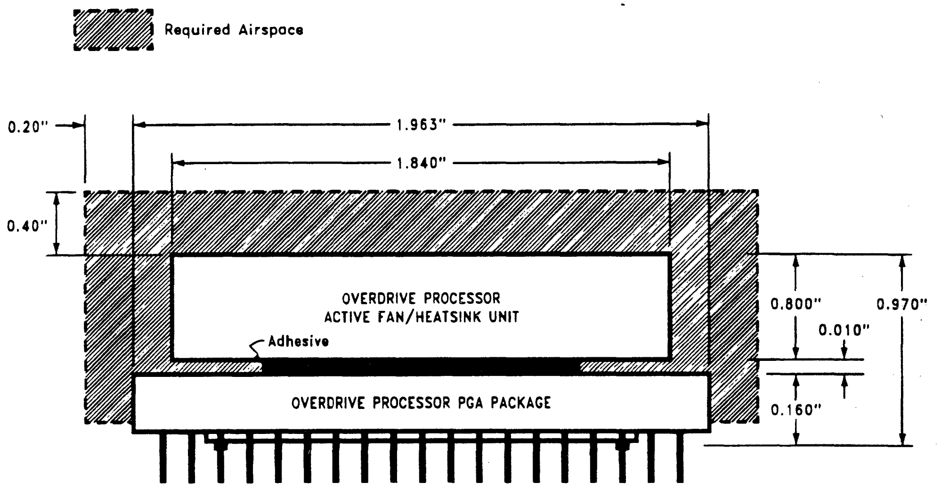
PC buyers value easy and safe upgrade installation. PC manufacturers can make upgrade component installation in the OverDrive processor socket simple and foolproof for the end user and reseller by implementing the suggestions listed in Table 16-2.

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
Heatsink Unit	1.830	1.850	N/A	N/A
Heatsink	N/A	N/A	0.790	0.810
Req'd Free Space	N/A	N/A	0.400	0.400
External Total	1.950	1.975	1.338	1.402
Space From Package	0.200	0.200	N/A	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 insure 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 insures 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.



242202-F4

Figure 16-3. Pentium™ OverDrive™ Processor PGA Package with Heatsink Attached

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 IntelSX2 and IntelDX2 OverDrive processors. The thermal efficiency of passive heatsink processors is dependent on system airflow. Please refer to the individual OverDrive processor Data Sheet 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	xxh
Pentium OverDrive processor (5V)	15h	3xh
IntelDX2™ OverDrive processor	04h	3xh
IntelSX2™ OverDrive processor	04h	5xh

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. This OverDrive processor also will 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 16.3, "Socket Layout,".

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. Some of these pins will be used to implement an on-chip writeback cache protocol, and the remaining pins will provide other OverDrive specific functions. 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

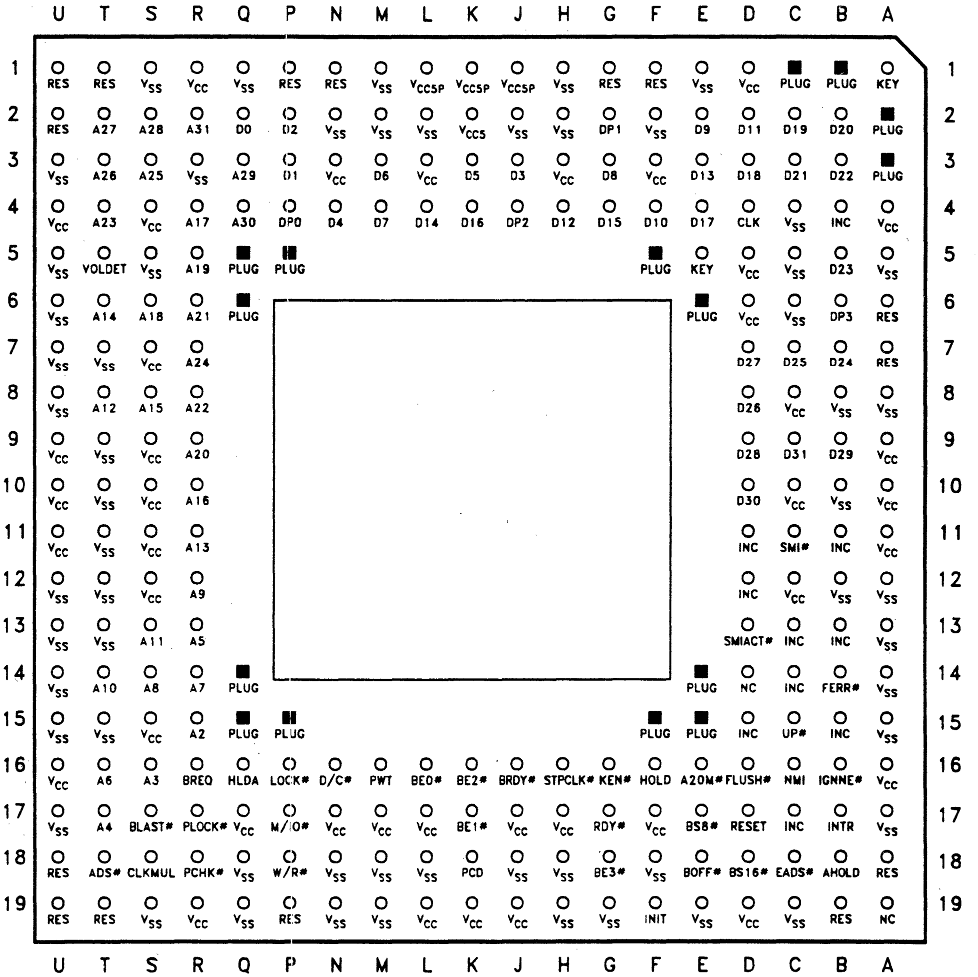
INC Pins are defined as Internal No-Connects. This means that the pin is not connected to the processor internally. Since the pin is inert and floating, it may be used in any manner seen fit, but must meet the **INC** pin specifications. In general, all **INC** pins have an intended use to implement 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

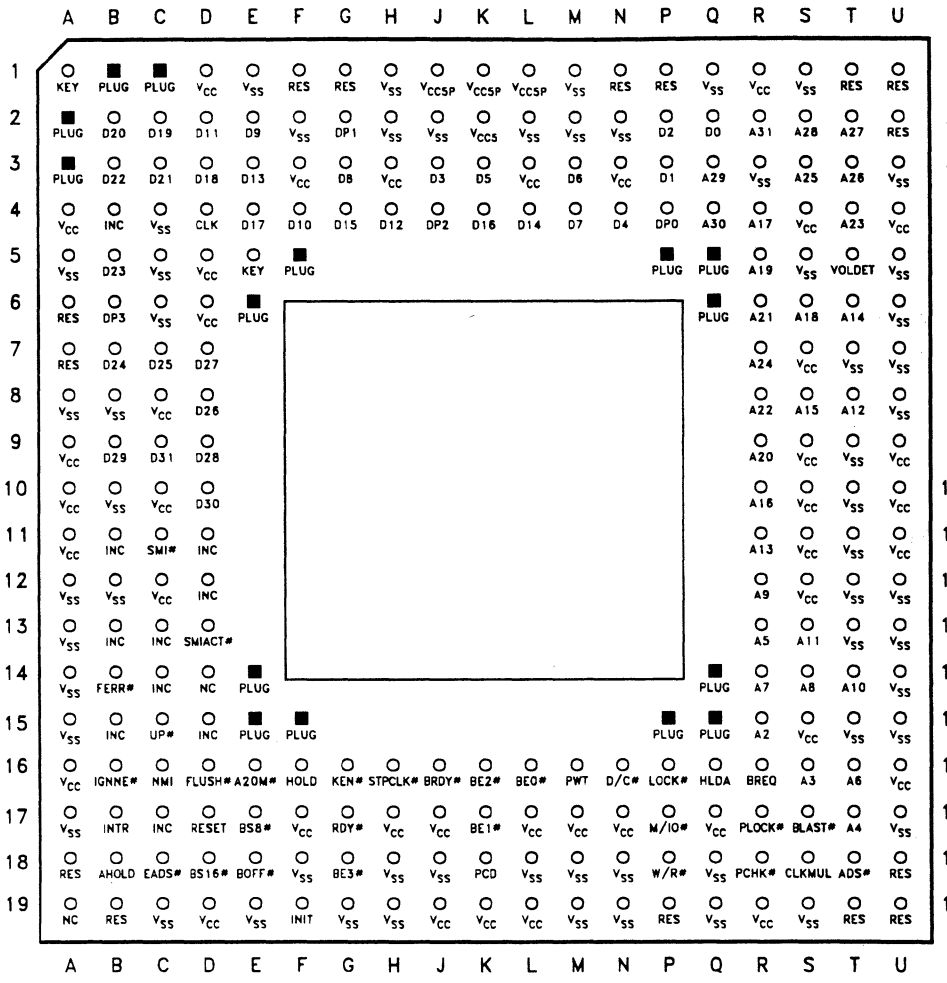


	U	T	S	R	Q	P	N	M	L	K	J	H	G	F	E	D	C	B	A	
1	RES	RES	V_{SS}	V_{CC}	V_{SS}	RES	RES	V_{SS}	V_{CC5P}	V_{CC5P}	V_{CC5P}	V_{SS}	RES	RES	V_{SS}	V_{CC}	PLUG	PLUG	KEY	1
2	RES	A27	A28	A31	D0	D2	V_{SS}	V_{SS}	V_{SS}	V_{CC5}	V_{SS}	V_{SS}	DP1	V_{SS}	D9	D11	D19	D20	PLUG	2
3	V_{SS}	A26	A25	V_{SS}	A29	D1	V_{CC}	D6	V_{CC}	D5	D3	V_{CC}	D8	V_{CC}	D13	D18	D21	D22	PLUG	3
4	V_{CC}	A23	V_{CC}	A17	A30	DP0	D4	D7	D14	D16	DP2	D12	D15	D10	D17	CLK	V_{SS}	INC	V_{CC}	4
5	V_{SS}	VOLDET	V_{SS}	A19	PLUG	PLUG								PLUG	KEY	V_{CC}	V_{SS}	D23	V_{SS}	5
6	V_{SS}	A14	A18	A21	PLUG										PLUG	V_{CC}	V_{SS}	DP3	RES	6
7	V_{SS}	V_{SS}	V_{CC}	A24												D27	D25	D24	RES	7
8	V_{SS}	A12	A15	A22												D26	V_{CC}	V_{SS}	V_{SS}	8
9	V_{CC}	V_{SS}	V_{CC}	A20												D28	D31	D29	V_{CC}	9
10	V_{CC}	V_{SS}	V_{CC}	A16												D30	V_{CC}	V_{SS}	V_{CC}	10
11	V_{CC}	V_{SS}	V_{CC}	A13												INC	SMI#	INC	V_{CC}	11
12	V_{SS}	V_{SS}	V_{CC}	A9												INC	V_{CC}	V_{SS}	V_{SS}	12
13	V_{SS}	V_{SS}	A11	A5												SMIACT#	INC	INC	V_{SS}	13
14	V_{SS}	A10	A8	A7	PLUG										PLUG	NC	INC	FERR#	V_{SS}	14
15	V_{SS}	V_{SS}	V_{CC}	A2	PLUG	PLUG								PLUG	PLUG	INC	UP#	INC	V_{SS}	15
16	V_{CC}	A6	A3	BREQ	HLDA	LOCK#	D/C#	PWT	BE0#	BE2#	BRDY#	STPCLK#	KEN#	HOLD	A20M#	FLUSH#	NMI	IGNNE#	V_{CC}	16
17	V_{SS}	A4	BLAST#	PLOCK#	V_{CC}	M/IO#	V_{CC}	V_{CC}	V_{CC}	BE1#	V_{CC}	V_{CC}	RDY#	V_{CC}	BS8#	RESET	INC	INTR	V_{SS}	17
18	RES	ADS#	CLKMUL	PCHK#	V_{SS}	W/R#	V_{SS}	V_{SS}	V_{SS}	PCD	V_{SS}	V_{SS}	BE3#	V_{SS}	BOFF#	BS16#	EADS#	AHOLD	RES	18
19	RES	RES	V_{SS}	V_{CC}	V_{SS}	RES	V_{SS}	V_{SS}	V_{CC}	V_{CC}	V_{CC}	V_{SS}	V_{SS}	INIT	V_{SS}	V_{CC}	V_{SS}	RES	NC	19
	U	T	S	R	Q	P	N	M	L	K	J	H	G	F	E	D	C	B	A	

242202–F5

NOTE:
All NC and RES pins must remain unconnected

Figure 16-4. OverDrive™ Processor Socket 3 Pinout (Top Side View)



	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	T	U
1	KEY	PLUG	PLUG	V_{CC}	V_{SS}	RES	RES	V_{SS}	V_{CC5P}	V_{CC5P}	V_{CC5P}	V_{SS}	RES	RES	V_{SS}	V_{CC}	V_{SS}	RES	RES
2	PLUG	D20	D19	D11	D9	V_{SS}	DP1	V_{SS}	V_{SS}	V_{CC5}	V_{SS}	V_{SS}	V_{SS}	D2	D0	A31	A28	A27	RES
3	PLUG	D22	D21	D18	D13	V_{CC}	D8	V_{CC}	D3	D5	V_{CC}	D6	V_{CC}	D1	A29	V_{SS}	A25	A26	V_{SS}
4	V_{CC}	INC	V_{SS}	CLK	D17	D10	D15	D12	DP2	D16	D14	D7	D4	DP0	A30	A17	V_{CC}	A23	V_{CC}
5	V_{SS}	D23	V_{SS}	V_{CC}	KEY	PLUG								PLUG	PLUG	A19	V_{SS}	VOLDET	V_{SS}
6	RES	DP3	V_{SS}	V_{CC}	PLUG										PLUG	A21	A18	A14	V_{SS}
7	RES	D24	D25	D27												A24	V_{CC}	V_{SS}	V_{SS}
8	V_{SS}	V_{SS}	V_{CC}	D26												A22	A15	A12	V_{SS}
9	V_{CC}	D29	D31	D28												A20	V_{CC}	V_{SS}	V_{CC}
10	V_{CC}	V_{SS}	V_{CC}	D30												A16	V_{CC}	V_{SS}	V_{CC}
11	V_{CC}	INC	SMI#	INC												A13	V_{CC}	V_{SS}	V_{CC}
12	V_{SS}	V_{SS}	V_{CC}	INC												A9	V_{CC}	V_{SS}	V_{SS}
13	V_{SS}	INC	INC	SMIACT#												A5	A11	V_{SS}	V_{SS}
14	V_{SS}	FERR#	INC	NC	PLUG										PLUG	A7	A8	A10	V_{SS}
15	V_{SS}	INC	UP#	INC	PLUG	PLUG								PLUG	PLUG	A2	V_{CC}	V_{SS}	V_{SS}
16	V_{CC}	IGNNE#	NMI	FLUSH#	A20M#	HOLD	KEN#	STPCLK#	BRDY#	BE2#	BE0#	PWT	D/C#	LOCK#	HLDA	BREQ	A3	A6	V_{CC}
17	V_{SS}	INTR	INC	RESET	BS8#	V_{CC}	RDY#	V_{CC}	V_{CC}	BE1#	V_{CC}	V_{CC}	V_{CC}	M/IO#	V_{CC}	PLOCK#	BLAST#	A4	V_{SS}
18	RES	AHOLD	EADS#	BS16#	BOFF#	V_{SS}	BE3#	V_{SS}	V_{SS}	PCD	V_{SS}	V_{SS}	V_{SS}	W/R#	V_{SS}	PCHK#	CLKMUL	ADS#	RES
19	NC	RES	V_{SS}	V_{CC}	V_{SS}	INIT	V_{SS}	V_{SS}	V_{CC}	V_{CC}	V_{CC}	V_{SS}	V_{SS}	RES	V_{SS}	V_{CC}	V_{SS}	RES	RES

242202–F6

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	N3	A5	M18
A3	S16	D1	P3	ADS#	T18	PCHK#	R18	A7	A9	N17	A8	M19
A4	T17	D2	P2	AHOLD	B18	PLOCK#	R17	A18	A10	Q17	A12	N2
A5	R13	D3	J3	BE0#	L16	PWT	M16	A19	A11	R1	A13	N18
A6	T16	D4	N4	BE1#	K17	RDY#	G17	B4	A16	R19	A14	N19
A7	R14	D5	K3	BE2#	K16	RESET	D17	B15	C8	S4	A15	Q1
A8	S14	D6	M3	BE3#	G18	SMI#	C11	B19	C10	S7	A17	Q18
A9	R12	D7	M4	BLAST#	S17	SMIACT#	D13	C17	C12	S9	B8	Q19
A10	T14	D8	G3	BOFF#	E18	STPCLK#	H16	F1	D1	S10	B10	R3
A11	S13	D9	E2	BRDY#	J16	UP#	C15	G1	D5	S11	B12	S1
A12	T8	D10	F4	BREQ	R16	VOLDET[2]	T5	N1	D6	S12	C4	S5
A13	R11	D11	D2	BS8#	E17	W/R#	P18	P1	D19	S15	C5	S19
A14	T6	D12	H4	BS16#	D18	**Position**		P19	F3	U4	C6	T7
A15	S8	D13	E3	CLK	D4	KEY	E5	T1	F17	U9	C19	T9
A16	R10	D14	L4	CLKMUL[2]	S18	KEY	A1	T19	H3	U10	E1	T10
A17	R4	D15	G4	D/C#	N16	PLUG	A2	U1	H17	U11	E19	T11
A18	S6	D16	K4	DP0	P4	PLUG	A3	U2	J17	U16	F2	T12
A19	R5	D17	E4	DP1	G2	PLUG	B1	U18	J19	**V_{CC5P}[2]**	F18	T13
A20	R9	D18	D3	DP2	J4	PLUG	C1	U19	K19	J1	G19	T15
A21	R6	D19	C2	DP3	B6	PLUG	E6	**N/C[1]**	L3	K1	H1	U3
A22	R8	D20	B2	EADS#	C18	PLUG	E14	A19	L17	L1	H2	U5
A23	T4	D21	C3	FERR#	B14	PLUG	E15	D14	L19	**V_{CC5}**	H18	U6
A24	R7	D22	B3	FLUSH#	D16	PLUG	F5	**INC**	M17	K2	H19	U7
A25	S3	D23	B5	HLDA	Q16	PLUG	F15	B11			J2	U8
A26	T3	D24	B7	HOLD	F16	PLUG	P5	B13			J18	U12
A27	T2	D25	C7	IGNNE#	B16	PLUG	P15	C13			L2	U13
A28	S2	D26	D8	INIT	F19	PLUG	Q5	C14			L18	U14
A29	Q3	D27	D7	INTR	B17	PLUG	Q6	D11			M1	U15
A30	Q4	D28	D9	KEN#	G16	PLUG	Q14	D12			M2	U17
A31	R2	D29	B9	LOCK#	P16	PLUG	Q15	D15				
		D30	D10	M/IO#	P17							
		D31	C9	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 only for the future Pentium™ OverDrive processor (3.3V).

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 ±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 Inte 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

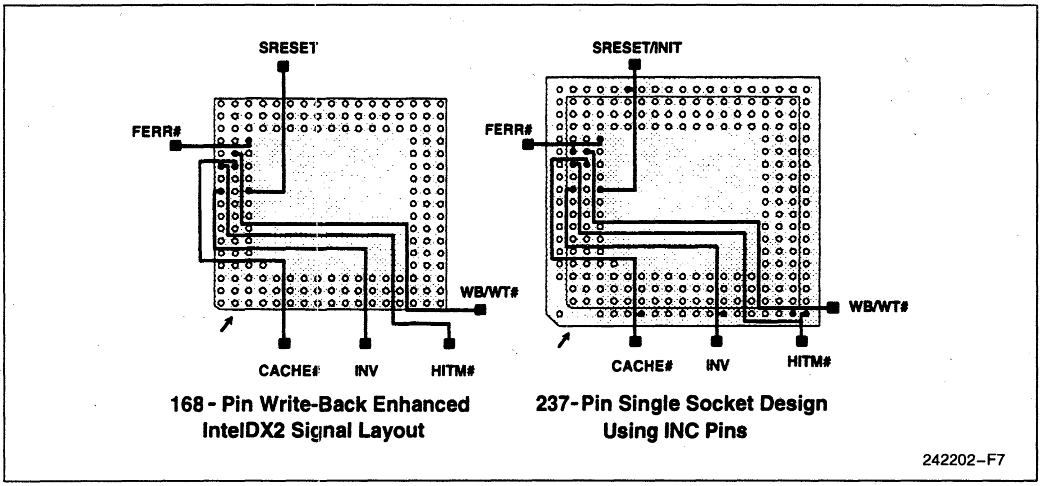

SRESET
FERR#
WB/WT#
CACHE#
INV
HITM#
168 - Pin Write-Back Enhanced IntelDX2 Signal Layout
SRESET/INIT
FERR#
WB/WT#
CACHE#
INV
HITM#
237-Pin Single Socket Design Using INC Pins
242202-F7

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.

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.

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.

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").

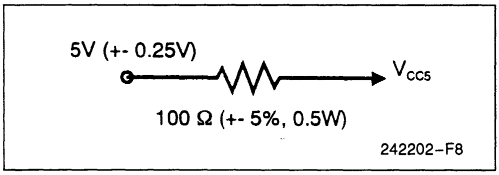


Figure 17-1. IntelDX4™ Processor V_{CC5} Current Limiting Resistor

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, and 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 0.45	 V V V	 7
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 ± 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 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		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

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 ≤ V_{IN} ≤ 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} ≥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} ≤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 ±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 ±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 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		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 INTELDX2 PROCESSOR

The data bus and data parity pins of the Write-Back Enhanced IntelDX2 processor 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 processor. 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 are 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 IntelDX2 processor is 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™ Processor 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 IntelDX2™ Processor 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	

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 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.
6. CACHE#, WB/WT#, HITM#, and INV are present only in the Write-Back Enhanced IntelDX2 processor.

Table 17-18. 3.3V AC Characteristics for the 75/25-MHz IntelDX4™ Processor

$V_{CC} = 3.3V \pm 0.3V$; $V_{CC5} = 5V \pm 0.25V$ (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 Valid Delay	3	19	ns	17-6	
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK# Float Delay		28	ns	17-7	3
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 (Continued)
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# Setup Time	8		ns	17-3	
t_{13}	EADS# Hold Time	3		ns	17-3	
t_{14}	KEN#, BS16#, BS8# Setup Time	8		ns	17-3	
t_{15}	KEN#, BS16#, BS8# Hold Time	3		ns	17-3	
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 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.
6. For adjacent clocks, assumes frequency of operation is constant. STPCLK# input should be used to change frequency of operation.

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 Valid Delay	3	14	ns	17-6	
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK# Float Delay		20	ns	17-7	3
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# Setup Time	5		ns	17-3	
t_{13}	EADS# Hold Time	3		ns	17-3	
t_{14}	KEN#, BS16#, BS8# Setup Time	5		ns	17-3	
t_{15}	KEN#, BS16#, BS8# Hold Time	3		ns	17-3	
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 (Continued)
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 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.
6. For adjacent clocks, assumes frequency of operation is constant. STPCLK# input should be used to change frequency of operation.

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 Valid Delay	2	12	ns	17-6	
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# Float Delay		18	ns	17-7	3
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# 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, 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 (Continued)
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 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.
6. For adjacent clocks, assumes frequency of operation is constant. STPCLK# input should be used to change frequency of operation.

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 2V
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 17-18 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 (Continued)

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 (Continued)

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 2V
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 17-18 for signal waveforms.

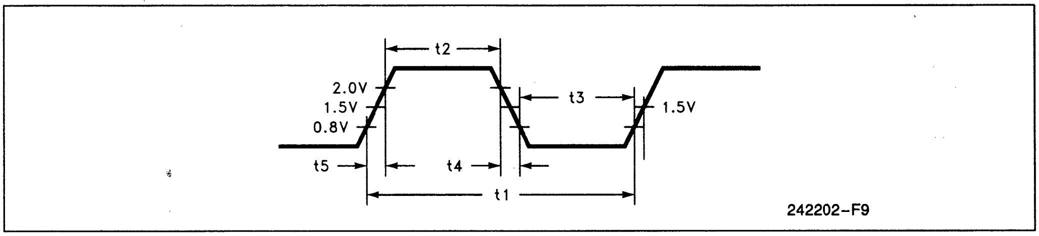


Figure 17-2. CLK Waveforms

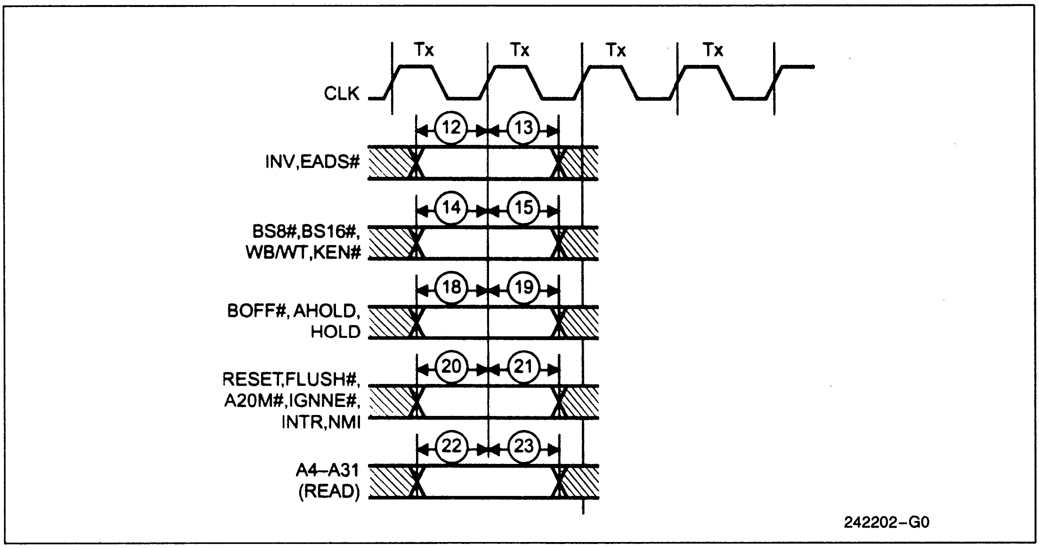


Figure 17-3. Input Setup and Hold Timing

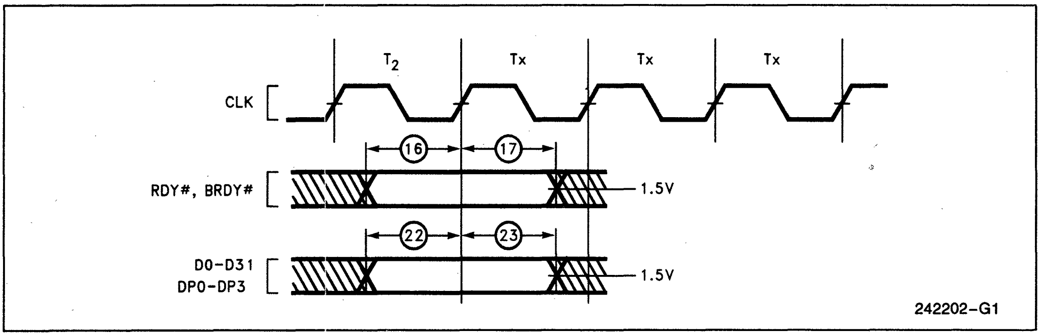


Figure 17-4. Input Setup and Hold Timing

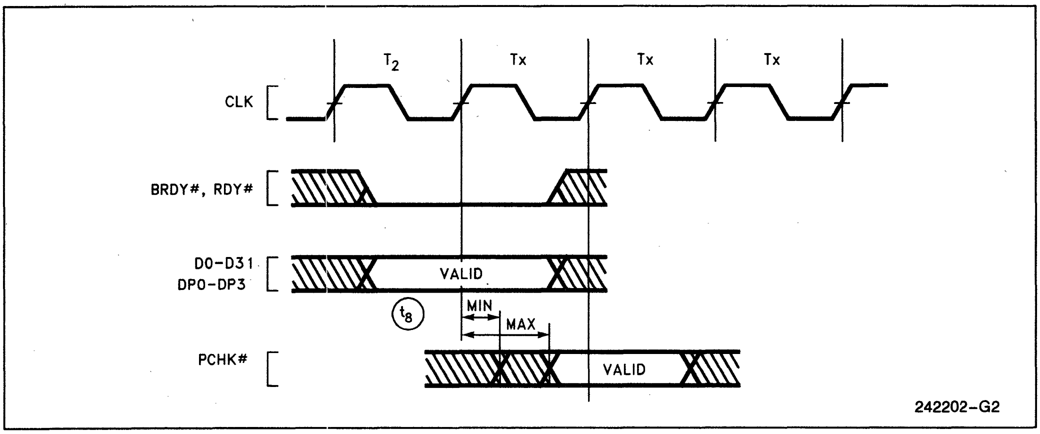


Figure 17-5. PCHK# Valid Delay Timing

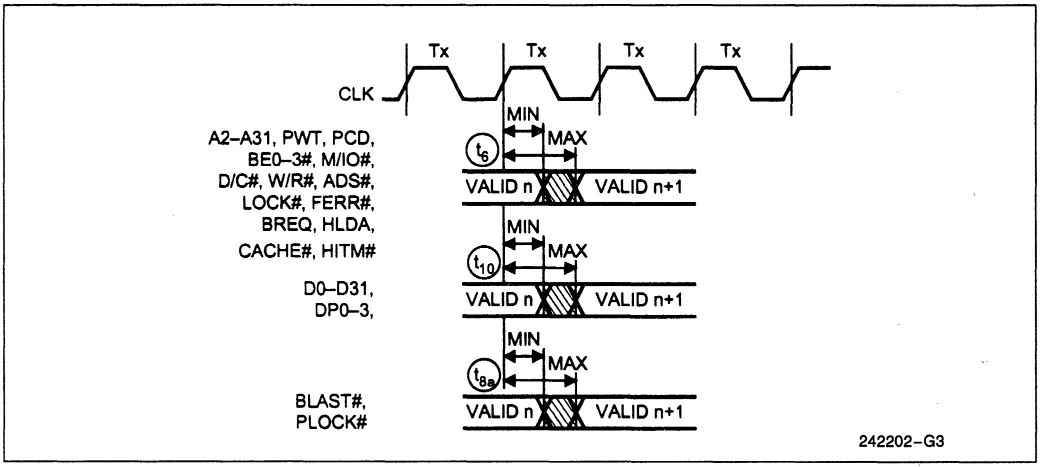


Figure 17-6. Output Valid Delay Timing

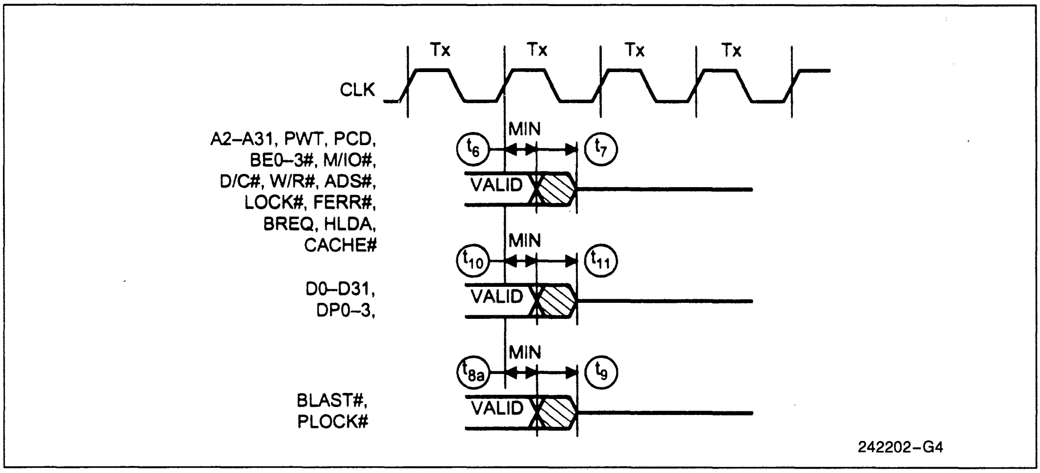


Figure 17-7. Maximum Float Delay Timing

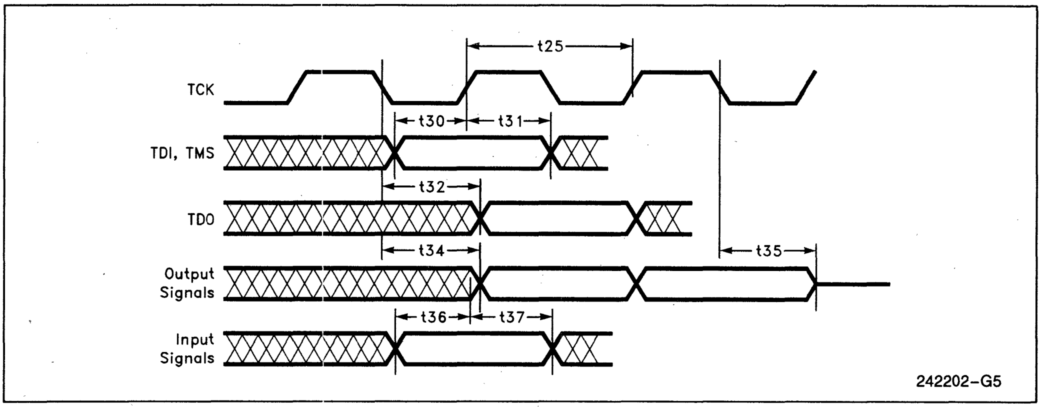
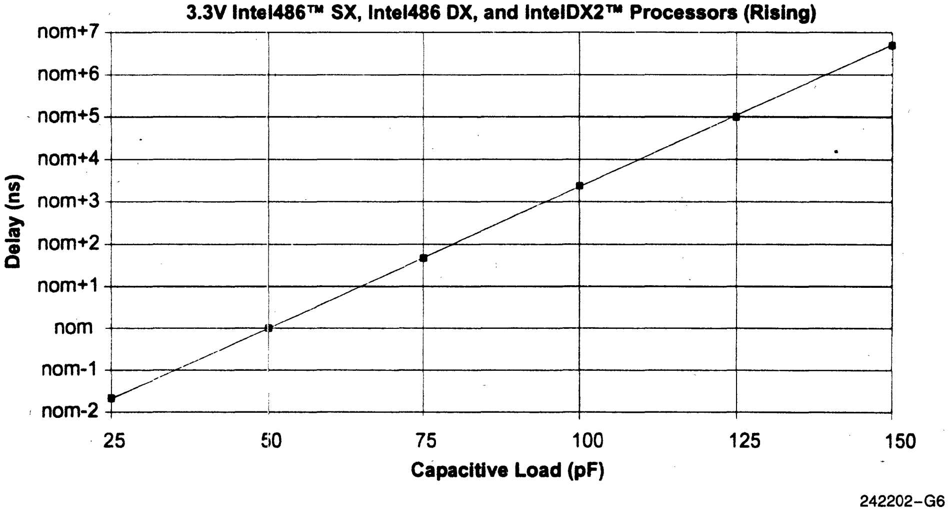


Figure 17-8. Test Signal Timing Diagram

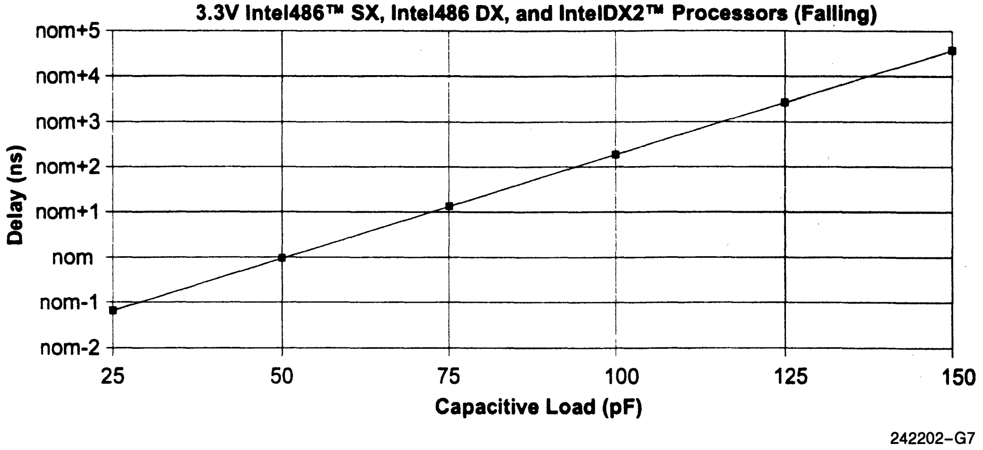
17.5 Capacitive Derating Curves

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. The figures apply to all frequencies specified for each corresponding product. Refer to Appendix C for bus frequencies above 33 MHz for Intel486 processors.



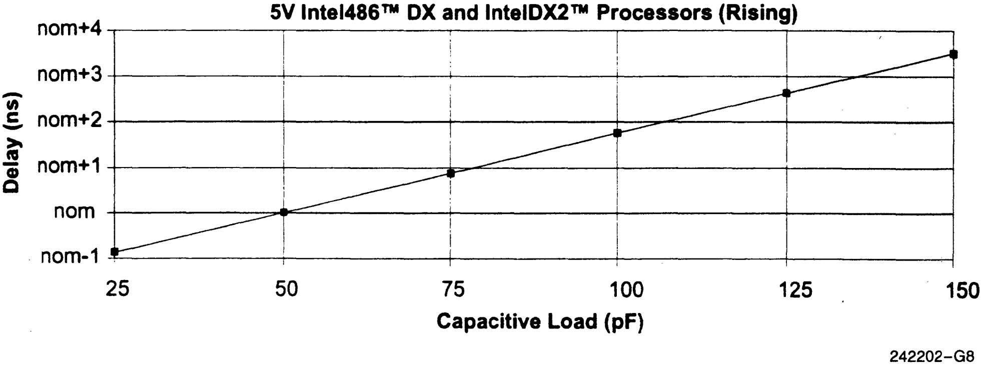
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

Figure 17-9. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



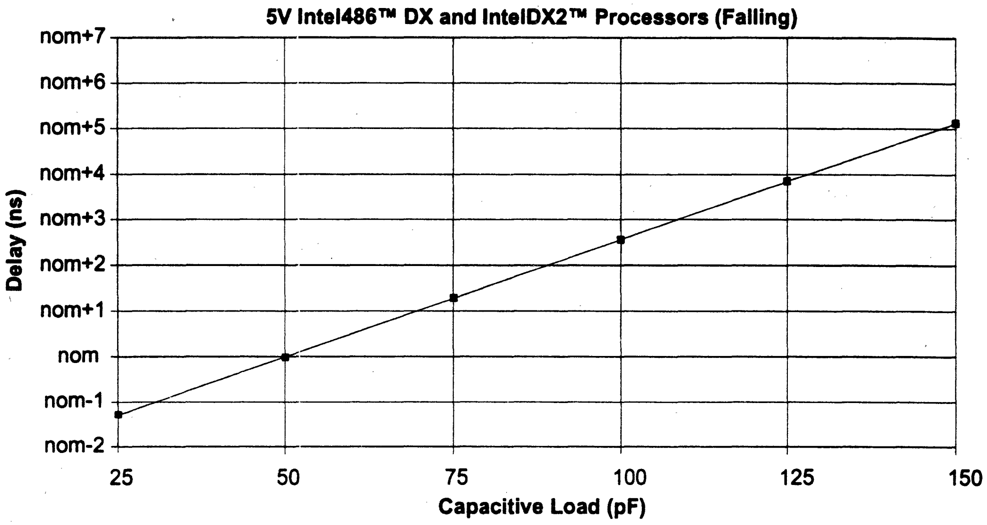
NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

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.

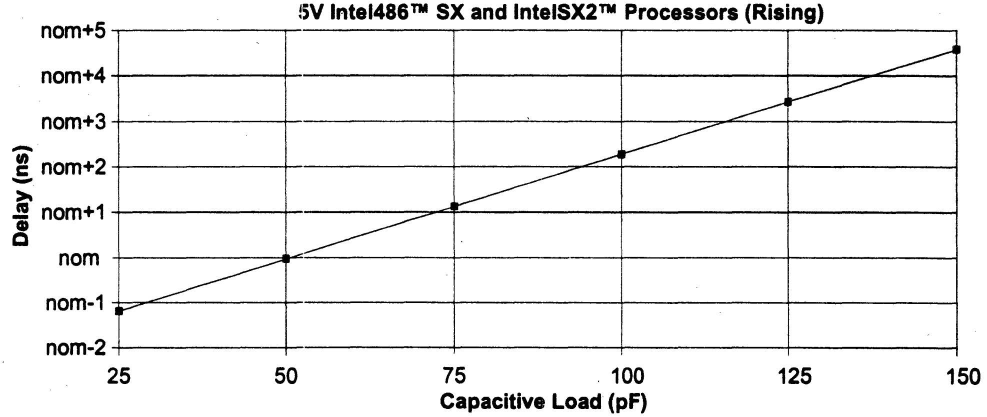
Figure 17-11. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



242202-G9

NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

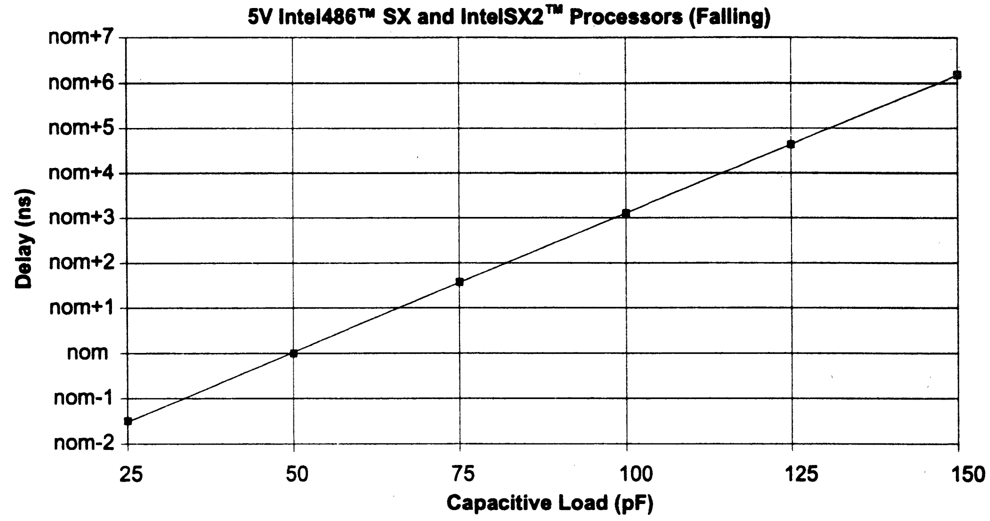
Figure 17-12. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a High-to-Low Transition



242202-H0

NOTE:
This graph will not be linear outside of the capacitive range shown.
nom = nominal value from the AC Characteristics table.

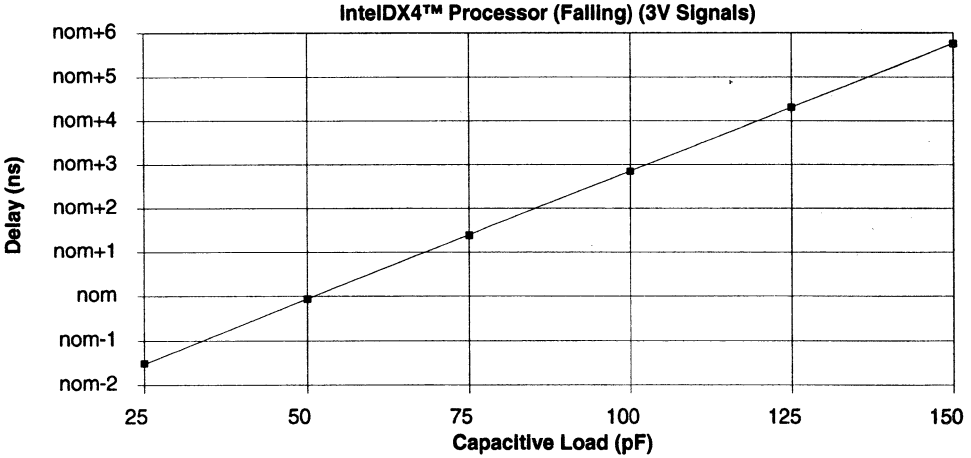
Figure 17-13. Typical Loading Delay versus Load Capacitance under Worst-Case Conditions for a Low-to-High Transition



242202–H1

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



242202–H2

Figure 17-15. IntelDX4™ Processor Capacitive Derating Curve for High-to-Low Transitions (3V Signals)

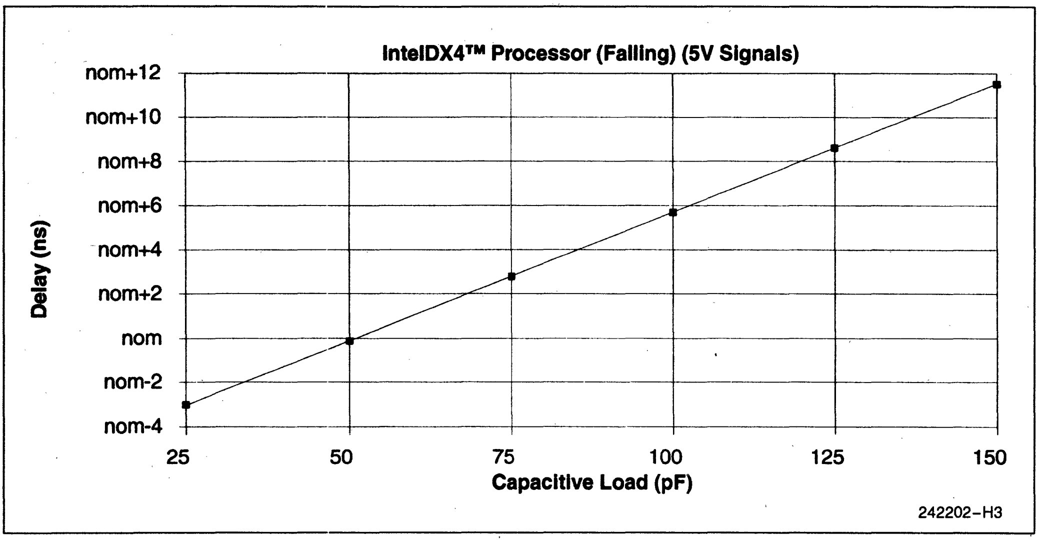


Figure 17-16. IntelDX4™ Processor Capacitive Derating Curve for Low-to-High Transitions (5V Signals)

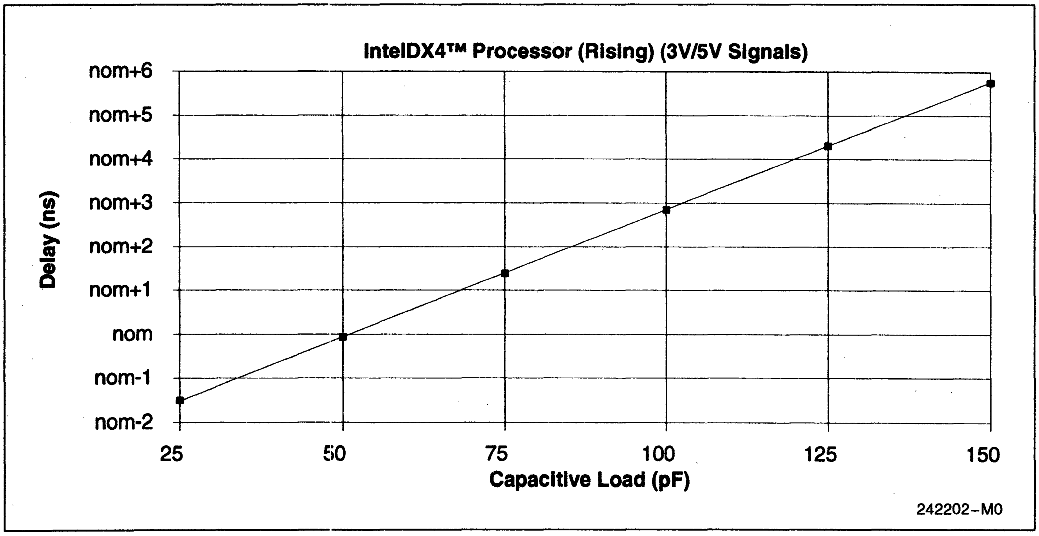


Figure 17-17. IntelDX4™ Processor Capacitive Derating Curve for Low-to-High Transitions (3V/5V Signals)

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

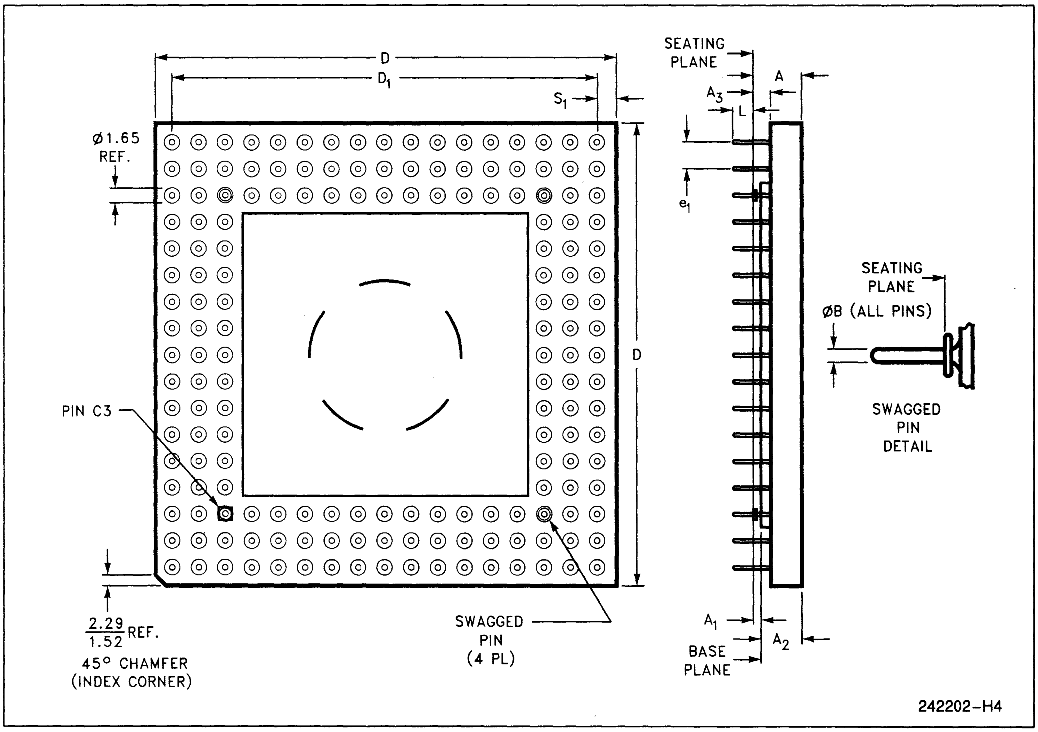


Figure 18-1. 168-Pin Ceramic PGA Package Dimensions

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

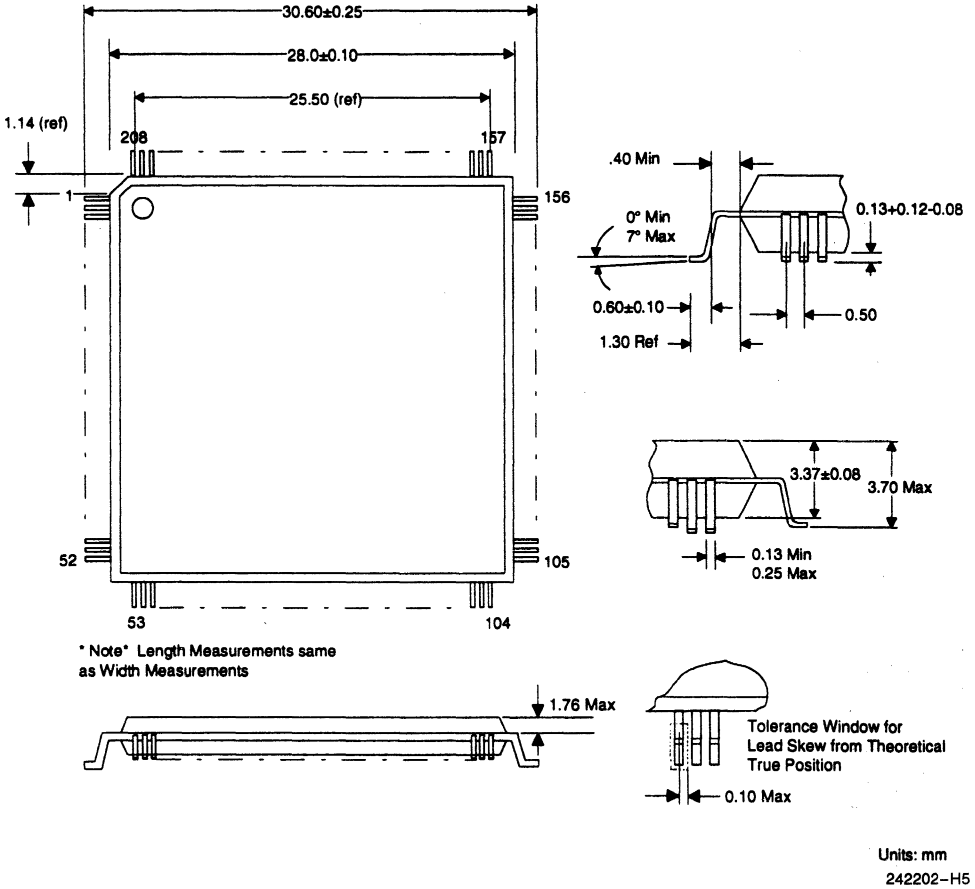
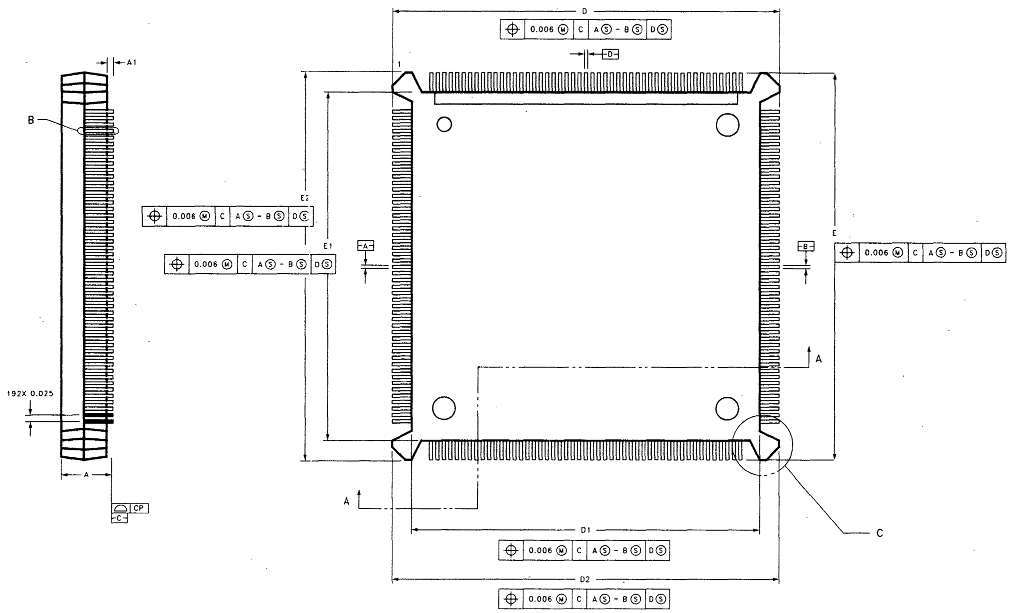


Figure 18-2. 208-Lead SQFP Package Dimensions

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.

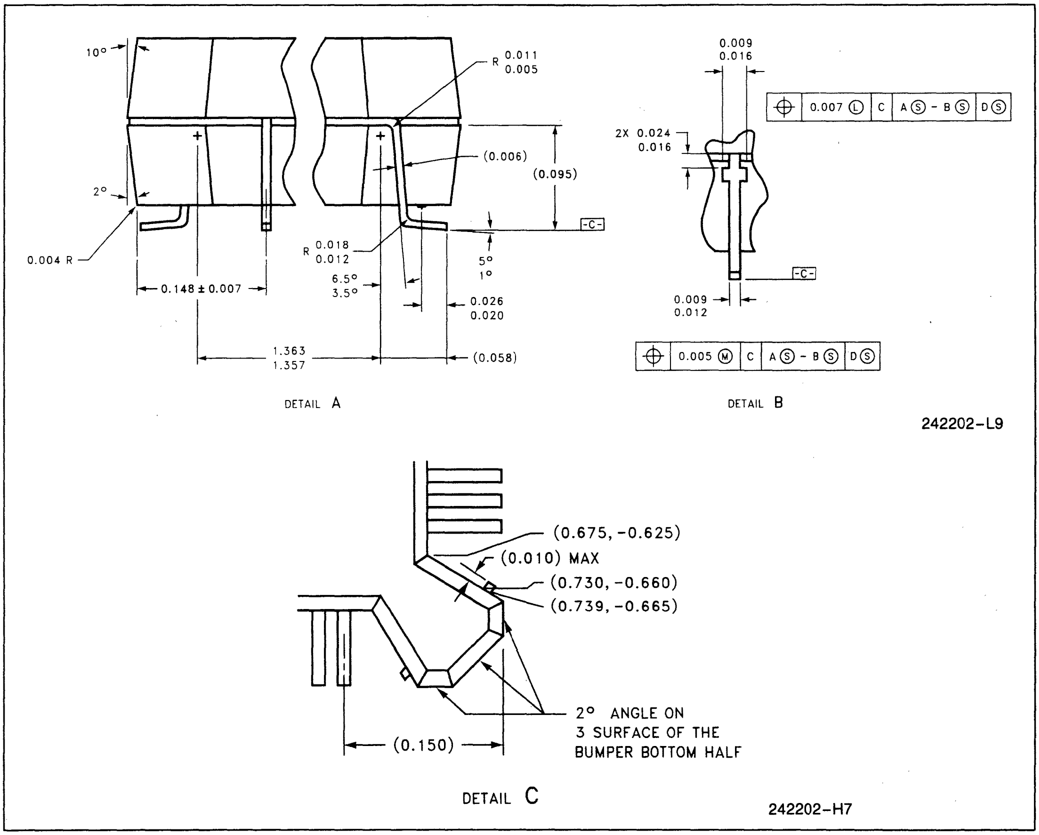


Figure 18-4. Typical Lead

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:

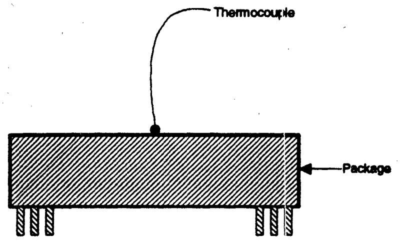
T_J, T_A, T_C = Junction, Ambient and Case Temperature, respectively.

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.

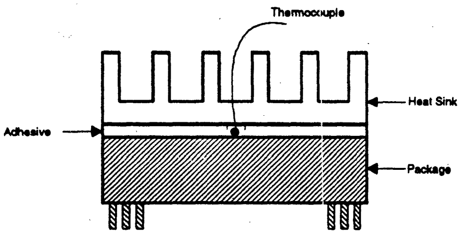
Note that T_A is greatly improved by attaching "fins" or a "heat sink" to the package. P (the maximum power consumption) is calculated by using the maximum I_{CC} at nominal V_{CC} (either 3.3V or 5V) as tabulated in the DC Charisteristics in section 17.



242202-H8

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



242202-H9

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

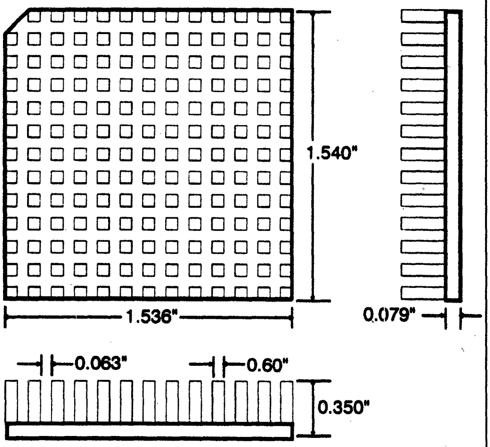
Refer to section 16, "OverDrive Processor Socket," for OverDrive processor thermal specifications.

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.



242202-I0

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 IntelDX4™ 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*	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 IntelDX4™ Processor

		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 the IntelDX4 processor 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

		Airflow—ft/min. (m/sec)			
	Freq. (MHz)	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

		Airflow—ft/min. (m/sec)			
	Freq. (MHz)	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 IntelDX4 Processor

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 2 power plane layers and 2 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™ Processor

		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_{JA} vs Airflow—ft/min (m/sec)			
	θ_{JC}	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

		Airflow—ft/min (m/sec)			
	Freq. (MHz)	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™ Processor 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 240486-002.

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

   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

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 a sample I/O buffer model parameters for the IntelDX4 processor. The first section presents an overview of signal buffer type catagorization. The second section presents a graphical representation of IBIS (I/O Buffer Information Sheet) data for each of the input, input/output, and output buffers types on the processor. The third section provides a text listing of the data 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 IntelDX4 and other members of the Intel486 processor family.

I/O Buffer Models for IntelDX4 Processor

Each valid delay for the 50-MHz bus is specified for a 0 pF load. The system designer should use I/O buffer modeling to account for signal delays due to loading. Table C-1 lists the buffer type to be used for each signal in the external interface.

Table C-1. External Interface Signal Buffer Assignment

Device	Signals	Type	Drive Buffer TYPE	Receiver Buffer Type
IntelDX4™ Processor	A20M#, AHOLD, BOFF#, BRDY#, BS8# BS16#, FLUSH#, HOLD, IGNNE#, INTR, KEN#, NMI, RDY#, RESET#, EADS#, SMI#, STPCLK#, SRESET, CLKMUL	I	N/A	IN1
	CLK	I	N/A	CLK
	D16–D0, DP2–DP0	I/O	I/O1	IN1
	D31–D17, DP3	I/O	I/O2	IN1
	A31–A4	I/O	I/O3	IN1
	ADS#, BLAST#, LOCK#, PLOCK#, SMIACT#, A3–A2, FERR#, HLDA	O	O1	N/A
	BE3#–BE0#, BREQ#, D/C#, M/IO#, PCD, PWT, PCHK#, W/R#	O	O2	N/A

Sample IBIS Files for IntelDX4 Processor

The following pages present sample IBIS file outputs for the IntelDX4 processor.

IBIS

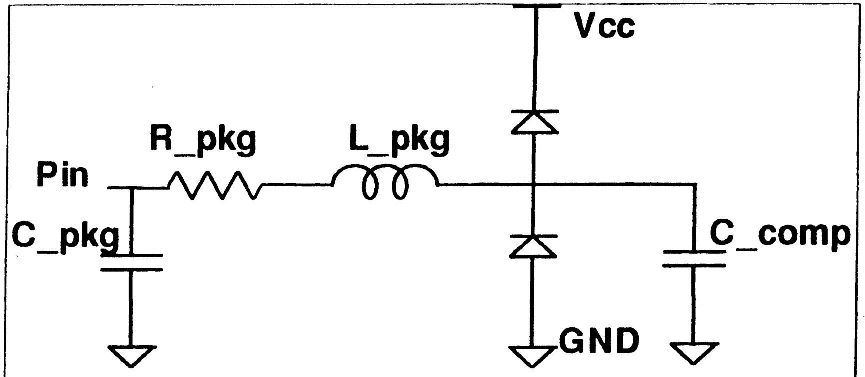
I/O Buffer Information Sheet

Component: IntelDX4(TM) Processor A-3 33/ 50MHz Bus

Signals: CLK

Buffer Type: CLOCK BUF

Revision:



Simplified electrical input model

Beyond the Rail Info			
Diode to GND		Diode to Vcc*	
V	I (mA)	V	I (mA)
0.0	0.0	vcc	0.0
-0.4	0.0	vcc+0.4	0.0
-0.5	0.2	vcc+0.5	0.0
-0.6	1.1	vcc+0.6	0.0
-0.7	3.0	vcc+0.7	0.1
-0.8	6.0	vcc+0.8	1.0
-0.9	11.0	vcc+0.9	8.0
-1.0	30.0	vcc+1.0	14.0
(-vcc)		vcc*2	

Intel does not guarantee diode operation for purposes other than ESD protection

*NOTE: VCC = voltage at pin J1

Packaging Characteristics			
	min	max	unit
R_pkg	140	325	mOhm
L_pkg	17.7	17.7	nH
C_pkg	9.1	9.1	pF
C_comp	2.0	2.0	pF

This information is for modeling purposes only and is not guaranteed.

242202-I1

IBIS

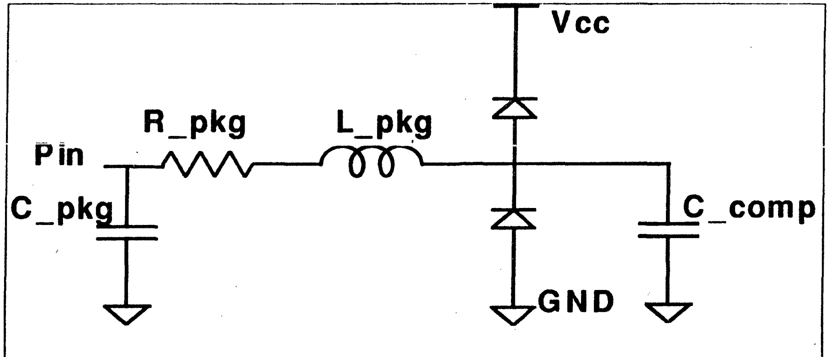
I/O Buffer Information Sheet

Component: IntelDX4(TM) Processor A-3

Signals: A20M#, AHOLD, BOFF#, BRDY#, BS16#, BS8#, EADS#, FLUSH#, HOLD, IGNNE#, INTR
KEN#, NMI, RDY#, RESET, SRESET, SMI#, STPCLK#

Buffer Type: INPUT #1

Revision:



Simplified electrical input model

Beyond the Rail Info			
Diode to GND		Diode to Vcc*	
V	I (mA)	V	I (mA)
0.0	0.0	vcc	0.0
-0.4	0.0	vcc+0.4	0.0
-0.5	0.2	vcc+0.5	0.0
-0.6	1.1	vcc+0.6	0.0
-0.7	3.0	vcc+0.7	0.1
-0.8	6.0	vcc+0.8	1.0
-0.9	11.0	vcc+0.9	8.0
-1.0	30.0	vcc+1.0	14.0
(-vcc)		vcc*2	
Intel does not guarantee diode operation for purposes other than ESD protection			

*NOTE: VCC = voltage at pin J1

Packaging Characteristics			
	min	max	unit
R_pkg	140	325	mOhm
L_pkg	9.7	19.5	nH
C_pkg	5.4	9.5	pF
C_comp	2.0	2.0	pF

This information is for modeling purposes only and is not guaranteed.

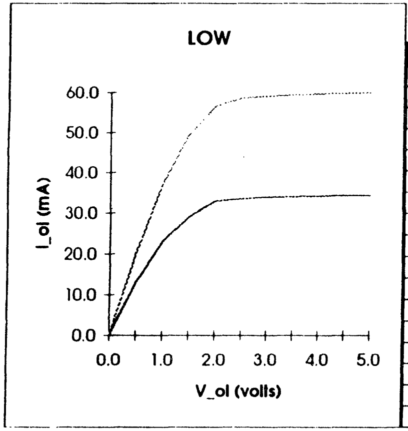
242202-12

IBIS

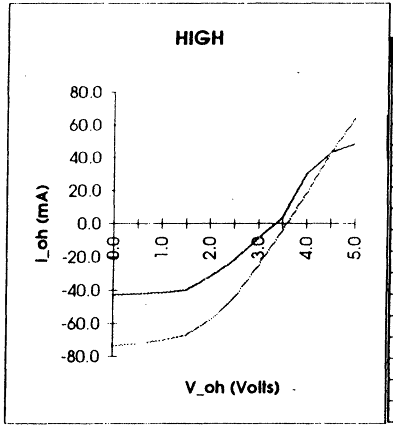
I/O Buffer Information Sheet

Component: IntelDX4(TM) Processor A-3 33/50MHz Bus
Signals: ADS#, BLAST#, SMIACT#, A<3:2>, FERR#, HLDA, PLOCK#, LOCK#

Buffer Type: OUT GROUP #1
Revision:



V_ol	I_ol min	I_ol max	I_ol typ
-5.0	-1000	-1830	-1380
2.0	-279.0	-472.0	-370.0
-1.0	-46.7	-57.1	-51.0
-0.5	-8.9	20.7	-13.5
0.0	0.0	0.0	0.0
0.5	13.0	20.1	12.9
1.0	23.0	37.2	23.7
1.5	29.0	49.7	31.5
2.0	33.0	56.6	35.4
2.5	33.5	58.6	36.3
3.0	34.0	59.0	36.6
3.5	34.1	59.4	36.9
4.0	34.2	59.7	37.0
4.5	34.4	59.8	37.1
5.0	34.4	60.0	37.2
6.0	34.4	60.0	37.2
10.0	34.4	60.0	37.2



V_oh	I_oh min	I_oh max	I_oh typ
-5.0	-1050.0	-1880.0	-1410.0
-1.0	-66.7	-95.2	-79.8
0.0	-43.0	-73.5	-47.1
0.5	-42.0	-72.1	-46.1
1.0	-41.5	-70.2	44.8
1.5	-40.0	-67.1	42.1
2.0	-32.0	-58.5	-35.1
2.5	-22.0	-44.9	-24.1
3.0	-9.0	-25.2	-9.5
3.5	3.4	-4.6	6.5
4.0	30.0	18.1	24.3
4.5	43.0	43.0	41.7
5.0	48.5	64.0	56.4
5.5	55.4	80.9	67.3
6.0	59.3	93.4	74.6
7.0	62.2	104.0	79.8
10.0	63.7	109.0	82.4

Beyond The Rail Info

Diode to GND		Diode to Vcc*	
V	I (mA)	V	I (mA)
0.0	0.0	vcc	0.0
-0.4	0.0	vcc+0.4	0.0
-0.5	0.2	vcc+0.5	0.0
-0.6	1.1	vcc+0.6	0.0
-0.7	3.0	vcc+0.7	0.1
-0.8	6.0	vcc+0.8	1.0
-0.9	11.0	vcc+0.9	8.0
-1.0	30.0	vcc+1.0	14.0
(-vcc)		vcc*2	

Intel does not guarantee diode operation for purposes other than ESD protection

*NOTE: VCC = voltage at pin J1

Packaging Characteristics

	min	max	unit
R_pkg	140.0	325.0	mOhm
L_pkg	9.5	15.1	nH
C_pkg	5.6	9.7	pF
C_comp	2.9	3.2	pF

Ramp Rate (into 0pF, no pkg)

	min	max	typ	unit
Rise	1.51	1.91	1.7	volts/ns
Fall	1.72	2.43	2.07	volts/ns

Simplified Output Resistance

	min	max	unit
Low	27.4	113.6	ohms
High	25.2	63.1	ohms

Simplified Output Resistance calculated using a 50 ohm load line

intel®

242202-13

This information is for modeling purposes only and is not guaranteed.

IBIS

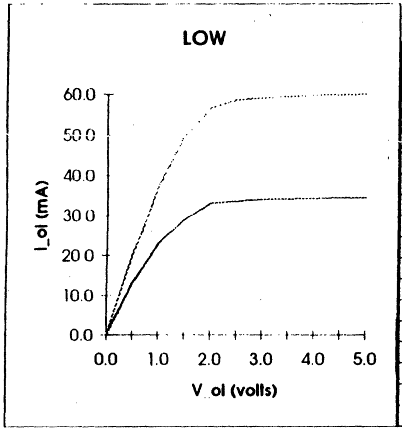
I/O Buffer Information Sheet

Component: IntelDX4(TM) Processor A-3 33/ 50MHz Bus

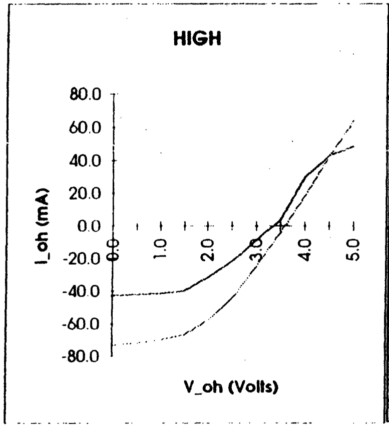
Signals: BE<3:0>#, BREQ, D/C#, M/IO#, PWT, PCD, PCHK#, W/R#

Buffer Type: OUT GROUP #2

Revision:



V_ol	I_ol min	I_ol max	I_ol typ
-5.0	-1000	-1830	-1380
-2.0	-279.0	-472.0	-370.0
-1.0	-46.7	57.1	-51.0
-0.5	-8.9	20.7	13.5
0.0	0.0	0.0	0.0
0.5	13.0	20.1	12.9
1.0	23.0	37.2	23.7
1.5	29.0	49.7	31.5
2.0	33.0	56.6	35.4
2.5	33.5	58.6	36.3
3.0	34.0	59.0	36.6
3.5	34.1	59.4	36.9
4.0	34.2	59.7	37.0
4.5	34.4	59.8	37.1
5.0	34.4	60.0	37.2
6.0	34.4	60.0	37.2
10.0	34.4	60.0	37.2



V_oh	I_oh min	I_oh max	I_oh typ
-5.0	-1050.0	-1880.0	-1410.0
-1.0	-66.7	-95.2	-79.8
0.0	-43.0	-73.5	-47.1
0.5	42.0	-72.1	-46.1
1.0	-41.5	-70.2	-44.8
1.5	-40.0	-67.1	-42.1
2.0	32.0	-58.5	35.1
2.5	22.0	44.9	24.1
3.0	-9.0	-25.2	-9.5
3.5	3.4	-4.6	6.5
4.0	30.0	18.1	24.3
4.5	43.0	43.0	41.7
5.0	48.5	64.0	56.4
5.5	55.4	80.9	67.3
6.0	59.3	93.4	74.6
7.0	62.2	104.0	79.8
10.0	63.7	109.0	82.4

Beyond The Rail Info

Diode to GND		Diode to Vcc*	
V	I (mA)	V	I (mA)
0.0	0.0	vcc	0.0
-0.4	0.0	vcc+0.4	0.0
-0.5	0.2	vcc+0.5	0.0
-0.6	1.1	vcc+0.6	0.0
-0.7	3.0	vcc+0.7	0.1
-0.8	6.0	vcc+0.8	1.0
-0.9	11.0	vcc+0.9	8.0
-1.0	30.0	vcc+1.0	14.0
(-vcc)		vcc*2	

Intel does not guarantee diode operation for purposes other than ESD protection

*NOTE: VCC = voltage at pin J1

Packaging Characteristics

	min	max	unit
R_pkg	140	325	mOhm
L_pkg	8.5	20.4	nH
C_pkg	5.2	11.0	pF
C_comp	2.7	2.7	pF

Ramp Rate (into 0pF, no pkg)

	min	max	typ	unit
Rise	1.51	1.91	1.7	volts/ns
Fall	1.72	2.43	2.07	volts/ns

Simplified Output Resistance

	min	max	unit
Low	27.4	113.6	ohms
High	25.2	63.1	ohms

Simplified Output Resistance calculated using a 50 ohm load line

242202-14

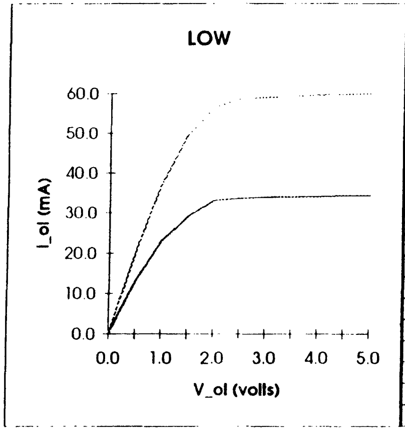
This information is for modeling purposes only and is not guaranteed.

IBIS

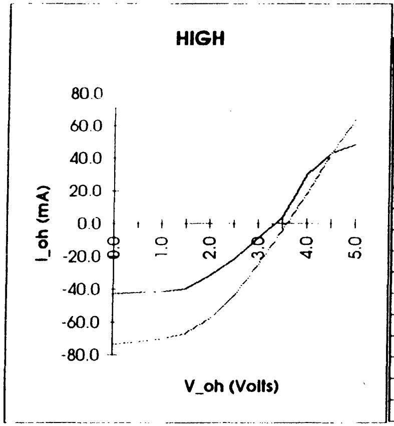
I/O Buffer Information Sheet

Component: IntelDX4(TM) Processor A-3 step 33/50MHz
Signals: DBUS<16:0>, DP<2:0>

Buffer Type: I/O GROUP #1
Revision:



V_ol	I_ol min	I_ol max	I_ol typ
-5.0	-1000	-1830	-1380
-2 0	279 0	472 0	370.0
1 0	-46 7	57.1	-51 0
0.5	-8 9	20.7	13 5
0 0	0.0	0.0	0.0
0.5	13.0	20.1	12.9
1.0	23 0	37 2	23 7
1 5	29 0	49.7	31 5
2.0	33.0	56 6	35.4
2.5	33 5	58.6	36 3
3 0	34 0	59.0	36 6
3 5	34 1	59 4	36 9
4 0	34.2	59.7	37.0
4.5	34 4	59.8	37.1
5.0	34 4	60.0	37.2
6.0	34 4	60.0	37.2
10 0	34.4	60.0	37 2



V_oh	I_oh min	I_oh max	I_oh typ
-5.0	-1050.0	-1880.0	-1410.0
-1.0	-66 7	-95.2	-79 8
0 0	-43.0	-73 5	47 1
0 5	-42 0	-72 1	46 1
1 0	-41 5	-70.2	44 8
1.5	-40 0	-67 1	-42 1
2.0	-32 0	58 5	-35 1
2 5	-22 0	44 9	-24 1
3 0	-9.0	25 2	-9.5
3.5	3 4	-4 6	6.5
4 0	30.0	18 1	24 3
4 5	43 0	43 0	41 7
5.0	48 5	64.0	56 4
5.5	55.4	80.9	67.3
6.0	59 3	93.4	74 6
7.0	62.2	104 0	79 8
10 0	63.7	109 0	82.4

Beyond The Rail Info

Diode to GND		Diode to Vcc*	
V	I (mA)	V	I (mA)
0 0	0.0	vcc	0 0
-0 4	0.0	vcc+0.4	0 0
-0.5	0.2	vcc+0.5	0 0
-0 6	1.1	vcc+0.6	0.0
-0.7	3.0	vcc+0.7	0 1
-0.8	6 0	vcc+0.8	1 0
-0.9	11.0	vcc+0.9	8 0
-1 0	30.0	vcc+1.0	14.0
(-vcc)		vcc*2	

Intel does not guarantee diode operation for purposes other than ESD protection

*NOTE: VCC = voltage at pin J1

Packaging Characteristics

	min	max	unit
R_pkg	140 0	325.0	mOhm
L_pkg	9.1	15.0	nH
C_pkg	4 8	6.5	pF
C_comp	2 7	2.7	pF

Ramp Rate (into 0pF, no pkg)

	min	max	typ	unit
Rise	1.51	1.91	1.7	volts/ns
Fall	1.72	2 43	2 07	volts/ns

Simplified Output Resistance

	min	max	unit
Low	27 4	113 6	ohms
High	25.2	63.1	ohms

Simplified Output Resistance calculated using a 50 ohm load line

This information is for modeling purposes only and is not guaranteed.

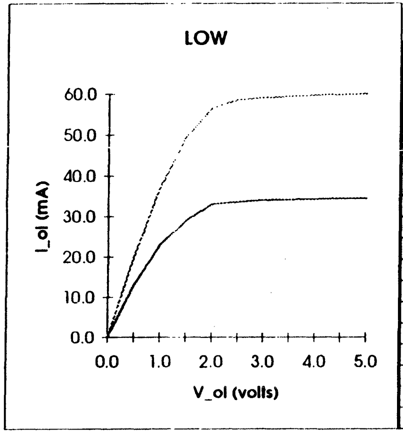
242202-15

IBIS

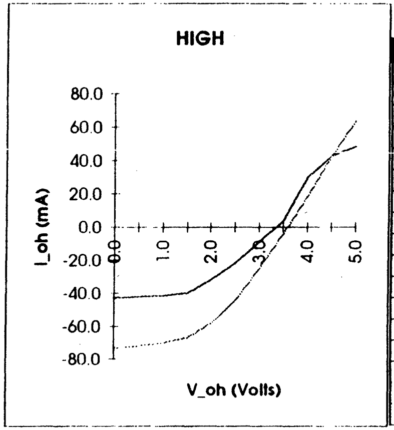
I/O Buffer Information Sheet

Component: IntelDX4(TM) Processor A-3 33/ 50MHz Bus
Signals: DBUS<31:17>, DP<3>

Buffer Type: I/O GROUP #1
Revision:



V_ol	I_ol min	I_ol max	I_ol typ
-5.0	-1000	-1830	-1380
-2.0	-279.0	-472.0	-370.0
-1.0	-46.7	-57.1	-51.0
-0.5	-8.9	-20.7	-13.5
0.0	0.0	0.0	0.0
0.5	13.0	20.1	12.9
1.0	23.0	37.2	23.7
1.5	29.0	49.7	31.5
2.0	33.0	56 6	35.4
2 5	33.5	58.6	36.3
3 0	34.0	59.0	36.6
3.5	34.1	59.4	36.9
4.0	34.2	59.7	37.0
4.5	34.4	59.8	37.1
5 0	34 4	60.0	37 2
6 0	34 4	60 0	37 2
10.0	34 4	60.0	37.2



V_oh	I_oh min	I_oh max	I_oh typ
-5.0	-1050.0	-1880.0	-1410.0
-1.0	-66.7	-95.2	-79 8
0 0	-43.0	-73 5	-47 1
0 5	-42.0	-72.1	-46 1
1 0	-41 5	-70.2	-44 8
1.5	-40.0	-67.1	-42 1
2.0	-32.0	-58.5	-35 1
2 5	-22.0	-44.9	-24.1
3 0	-9.0	25.2	-9 5
3 5	3.4	-4.6	6 5
4 0	30.0	18.1	24 3
4 5	43.0	43.0	41 7
5 0	48 5	64.0	56 4
5 5	55 4	80.9	67.3
6 0	59 3	93 4	74 6
7 0	62.2	104 0	79 8
10 0	63.7	109.0	82 4

Beyond The Rail Info

Diode to GND		Diode to Vcc*	
V	I (mA)	V	I (mA)
0.0	0.0	vcc	0.0
-0.4	0.0	vcc+0 4	0.0
-0 5	0.2	vcc+0 5	0 0
-0 6	1.1	vcc+0.6	0.0
-0.7	3.0	vcc+0.7	0.1
-0.8	6.0	vcc+0.8	1.0
-0.9	11.0	vcc+0.9	8.0
-1.0	30.0	vcc+1.0	14.0
(-vcc)		vcc*2	

Intel does not guarantee diode operation for purposes other than ESD protection

*NOTE: VCC = voltage at pin J1

Packaging Characteristics

	min	max	unit
R_pkg	140.0	325.0	mOhm
L_pkg	8.5	13.7	nH
C_pkg	4.0	6.3	pF
C_comp	2.9	3.2	pF

Ramp Rate (into 0pF, no pkg)

	min	max	typ	unit
Rise	1.51	1.91	1.7	volts/ns
Fall	1.72	2.43	2.07	volts/ns

Simplified Output Resistance

	min	max	unit
Low	27 4	113.6	ohms
High	25.2	63.1	ohms

Simplified Output Resistance calculated using a 60 ohm load line

This information is for modeling purposes only and is not guaranteed.

242202-16

IBIS

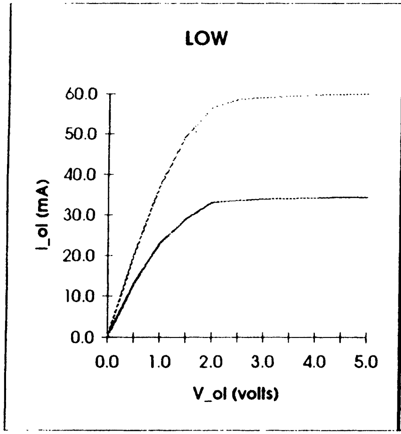
I/O Buffer Information Sheet

Component: IntelDX4(TM) Processor A-3 33/ 50MHz Bus

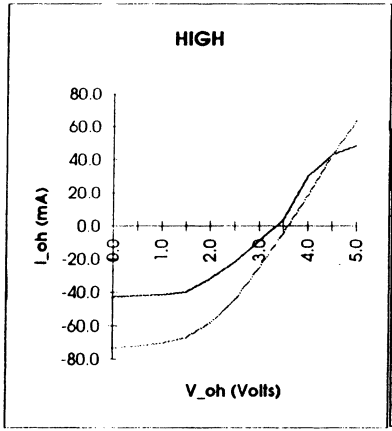
Signals: ABUS<31:4>

Buffer Type: I/O GROUP #3

Revision:



V_ol	I_ol min	I_ol max	I_ol typ
-5.0	-1000	-1830	-1380
-2 0	-279 0	-472.0	370.u
-1.0	-46.7	-57.1	-51.0
0.5	-8.9	-20 7	-13 5
0.0	0.0	0.0	0 0
0 5	13 0	20.1	12.9
1.0	23.0	37.2	23.7
1.5	29.0	49.7	31.5
2 0	33.0	56.6	35 4
2.5	33.5	58.6	36.3
3.0	34.0	59.0	36.6
3.5	34.1	59.4	36.9
4.0	34.2	59.7	37.0
4.5	34.4	59.8	37.1
5.0	34.4	60.0	37.2
6 0	34.4	60.0	37.2
10.0	34.4	60 0	37 2



V_oh	I_oh min	I_oh max	I_oh typ
-5 0	-1050.0	-1880.0	-1410.0
-1.0	-66 7	-95.2	-79 8
0.0	-43 0	-73.5	47 1
0.5	-42 0	-72 1	-46 1
1.0	-41 5	-70.2	44 8
1 5	-40 0	-67.1	-42 1
2.0	-32 0	-58.5	-35 1
2.5	-22 0	-44.9	-24.1
3.0	-9.0	-25.2	-9.5
3.5	3.4	-4.6	6.5
4.0	30.0	18.1	24.3
4.5	43.0	43.0	41.7
5.0	48 5	64.0	56.4
5.5	55.4	80.9	67.3
6.0	59 3	93.4	74 6
7.0	62 2	104.0	79 8
10.0	63 7	109.0	82 4

Beyond The Rail Info

Diode to GND		Diode to Vcc*	
V	I (mA)	V	I (mA)
0 0	0.0	vcc	0.0
0 4	0.0	vcc+0.4	0 0
-0 5	0.2	vcc+0.5	0 0
-0.6	1.1	vcc+0.6	0 0
-0 7	3 0	vcc+0 7	0 1
-0.8	6.0	vcc+0.8	1.0
-0.9	11.0	vcc+0.9	8.0
-1.0	30.0	vcc+1.0	14.0
(-VCC)		vcc*2	

Intel does not guarantee diode operation for purposes other than ESD protection

*NOTE: VCC = voltage at pin J1

Packaging Characteristics

	min	max	unit
R_pkg	140	325	mOhm
L_pkg	7.9	14.7	nH
C_pkg	4 4	7 4	pF
C_comp	2.9	2.9	pF

Ramp Rate (into 0pF, no pkg)

	min	max	typ	unit
Rise	1.51	1.91	1 7	volts/ns
Fall	1.72	2.43	2 07	volts/ns

Simplified Output Resistance

	min	max	unit
Low	27.4	113 6	ohms
High	25.2	63.1	ohms

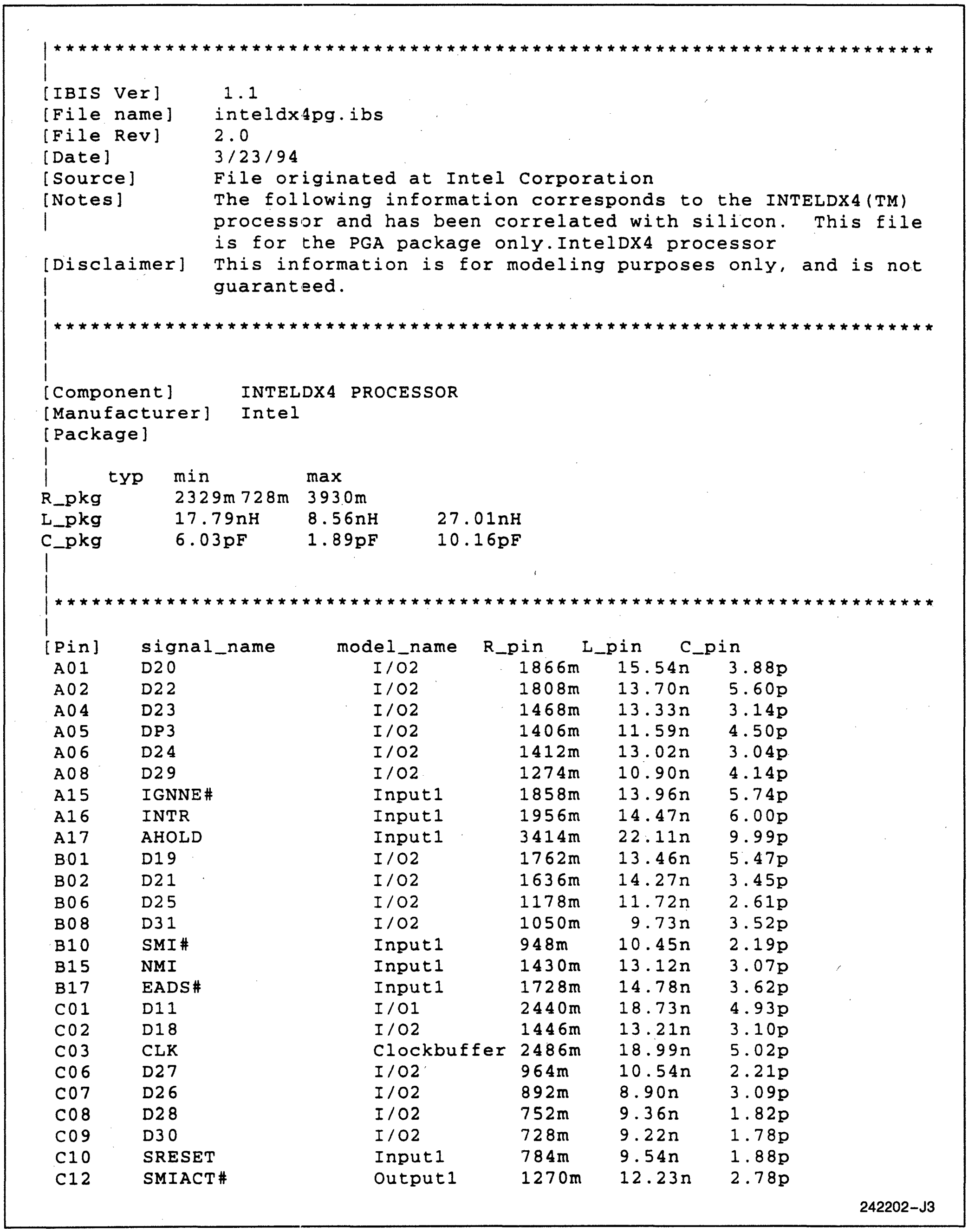
Simplified Output Resistance calculated using a 50 ohm load line

intel®

242202-17

This information is for modeling purposes only and is not guaranteed.

Sample Text Listing of IBIS Files for 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
```

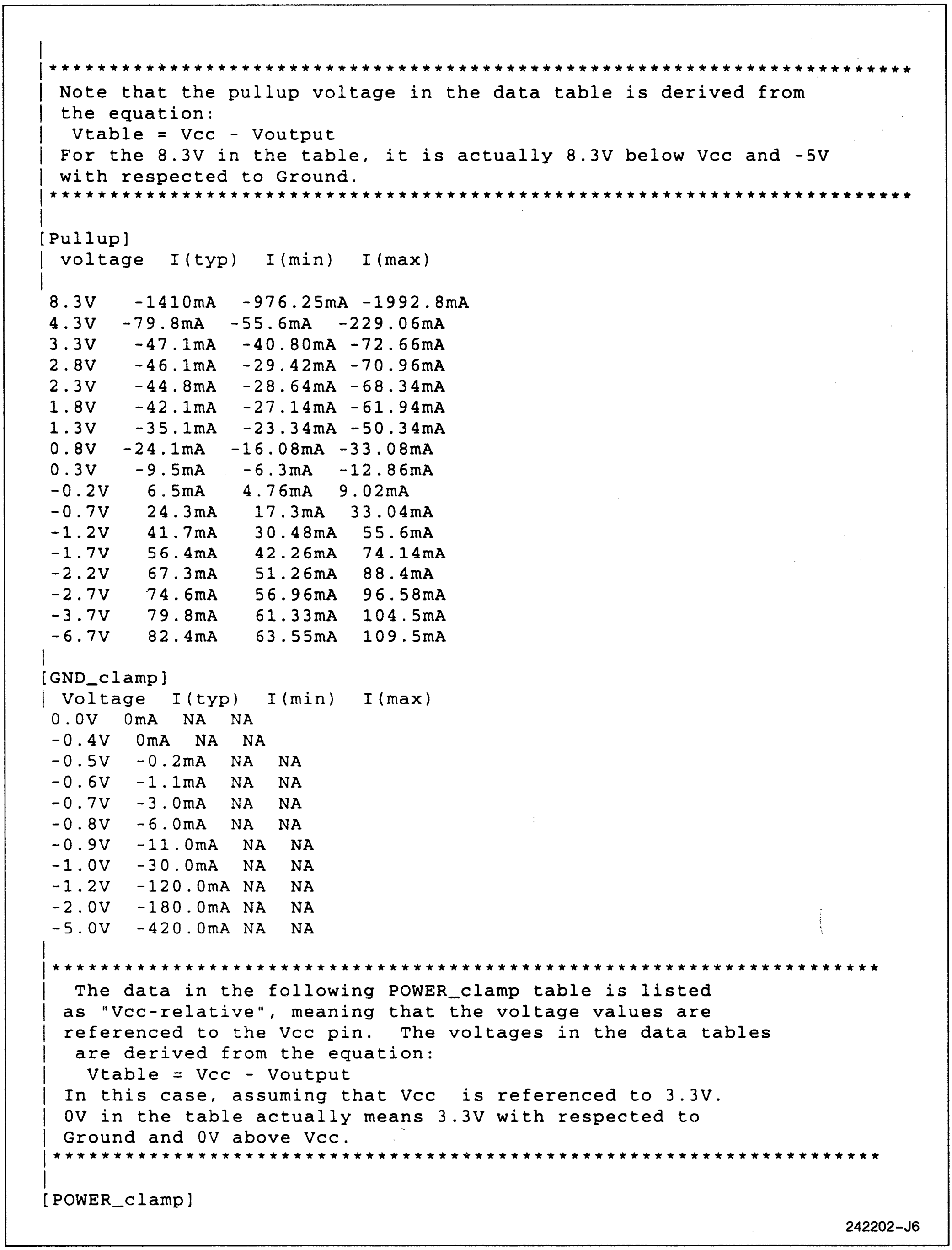
242202-J3

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

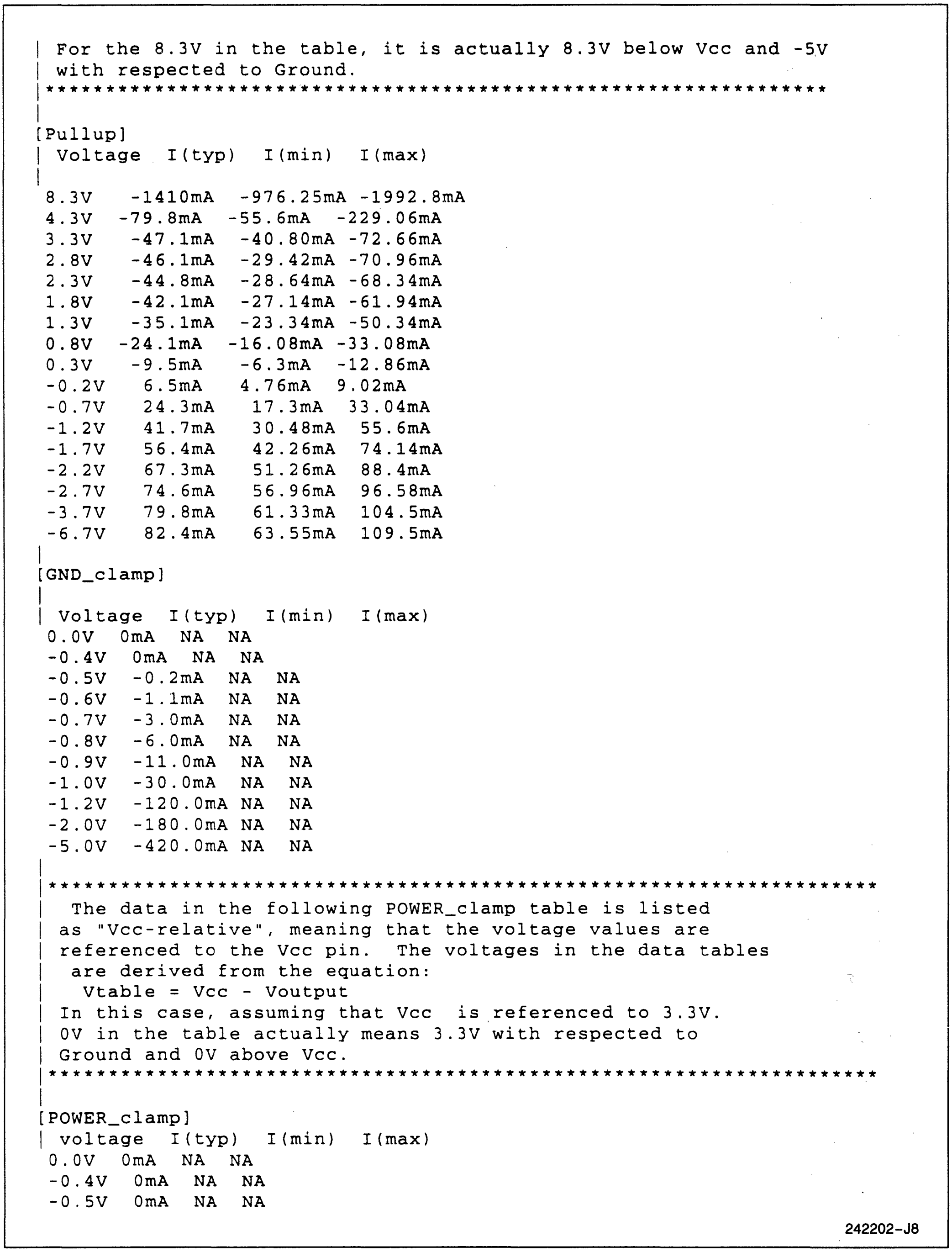


```
|
|**********************************************************************
| Note that the pullup voltage in the data table is derived from
| the equation:
|  Vtable = Vcc - Voutput
| For the 8.3V in the table, it is actually 8.3V below Vcc and -5V
| with respected to Ground.
|**********************************************************************
|
[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

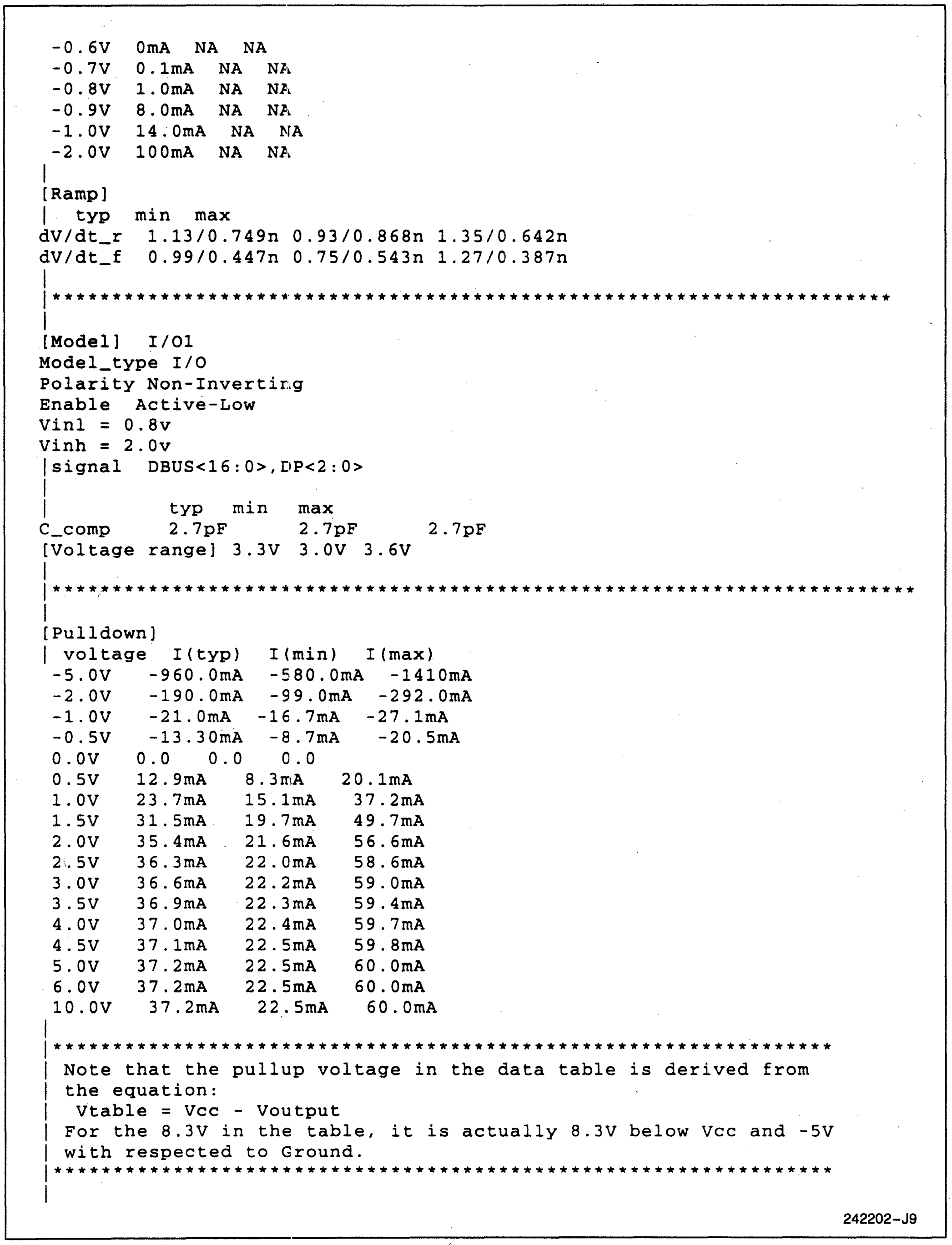
```
| voltage   I(typ)   I(min)   I(max)
 0.0V   0mA  NA  NA
 -0.4V  0mA  NA  NA
 -0.5V  0mA  NA  NA
 -0.6V  0mA  NA  NA
 -0.7V  0.1mA  NA  NA
 -0.8V  1.0mA  NA  NA
 -0.9V  8.0mA  NA  NA
 -1.0V  14.0mA  NA  NA
 -2.0V  100mA  NA  NA
|
|***********************************************************************
|
[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



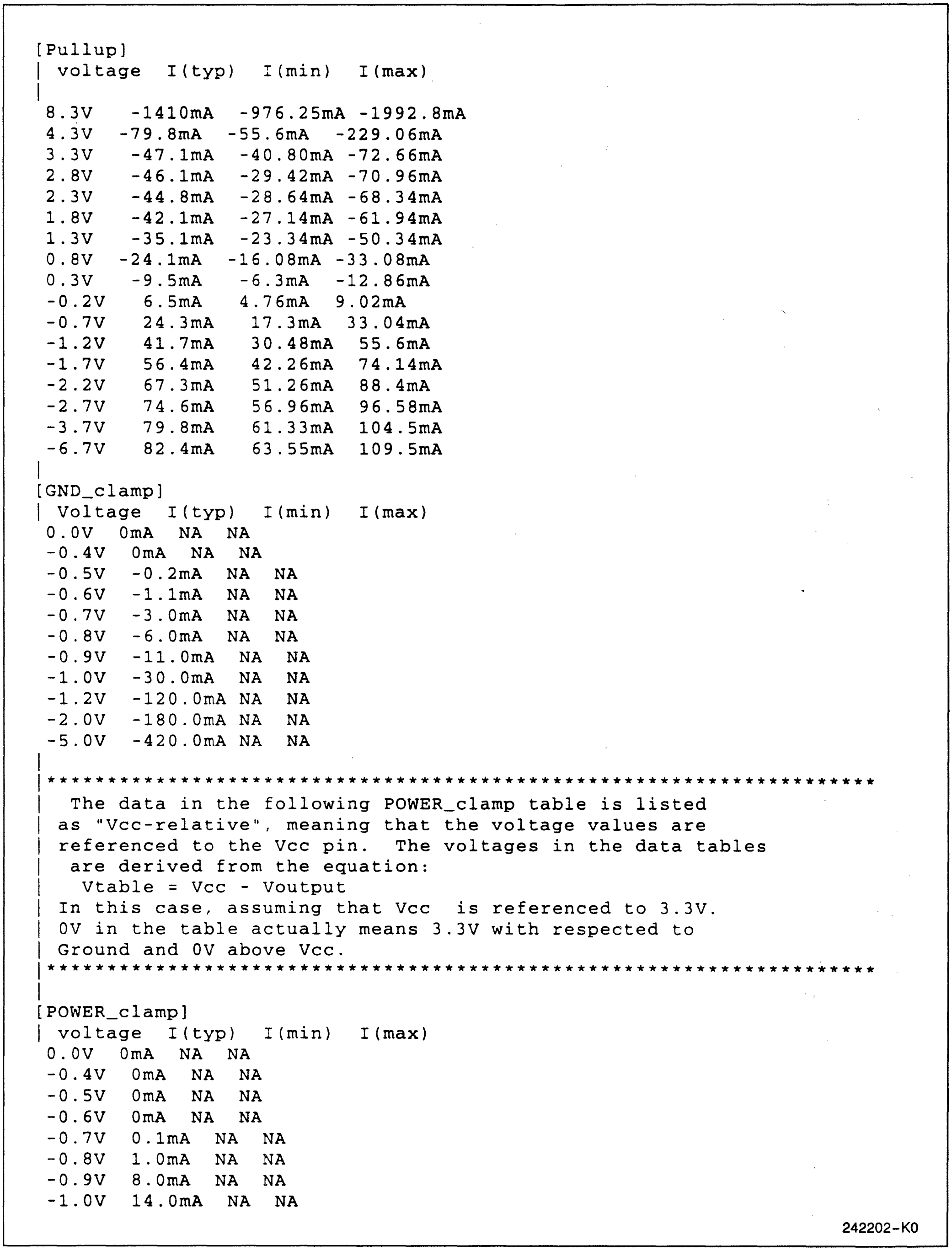
```
| For the 8.3V in the table, it is actually 8.3V below Vcc and -5V
| with respected to Ground.
|*******************************************************************
|
[Pullup]
| Voltage  I(typ)  I(min)  I(max)
|
 8.3V   -1410mA  -976.25mA -1992.8mA
 4.3V  -79.8mA  -55.6mA  -229.06mA
 3.3V   -47.1mA  -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



```
  -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

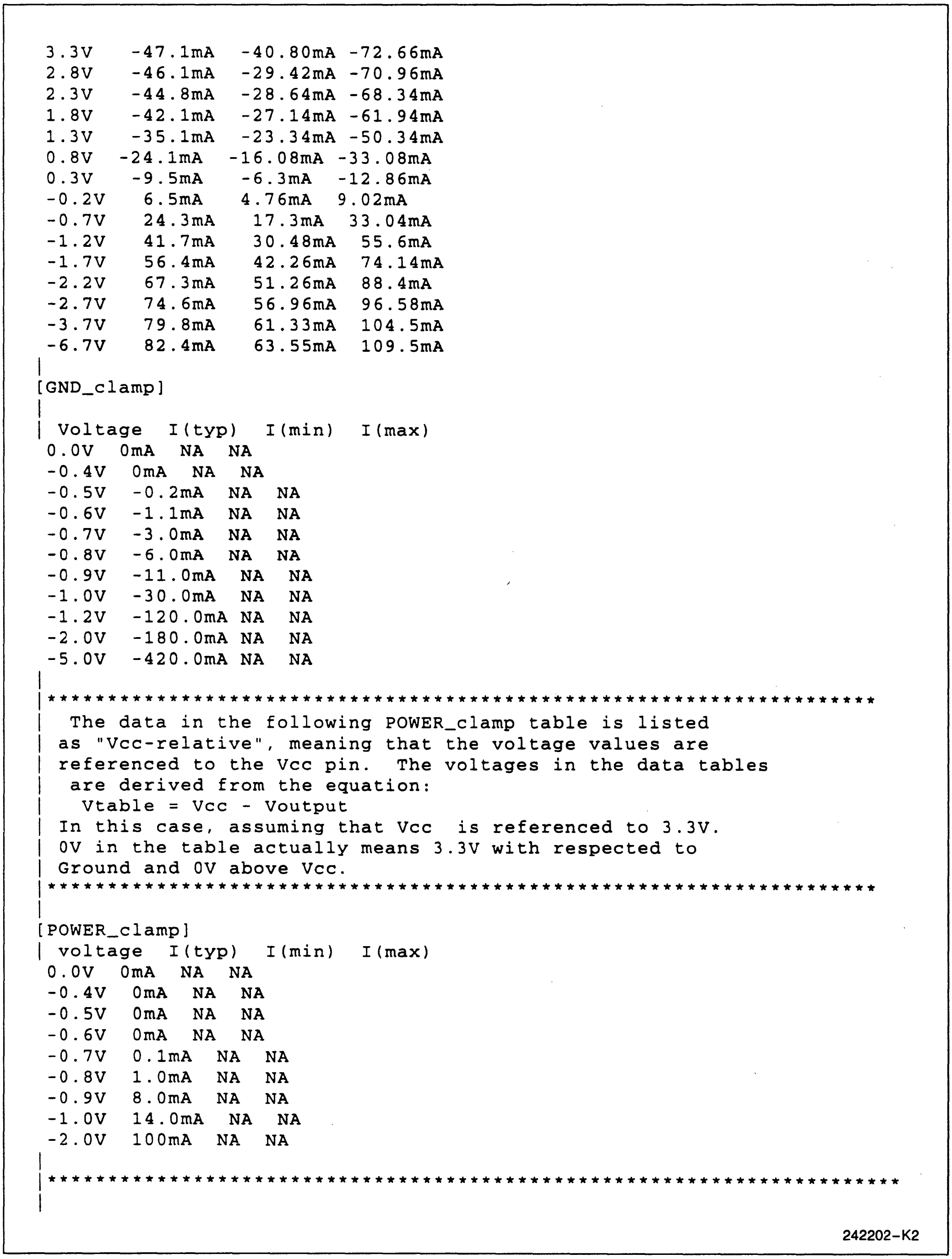


```
[Pullup]
| voltage   I(typ)   I(min)   I(max)
|
 8.3V   -1410mA  -976.25mA -1992.8mA
 4.3V  -79.8mA  -55.6mA   -229.06mA
 3.3V   -47.1mA  -40.80mA -72.66mA
 2.8V   -46.1mA  -29.42mA -70.96mA
 2.3V   -44.8mA  -28.64mA -68.34mA
 1.8V   -42.1mA  -27.14mA -61.94mA
 1.3V   -35.1mA  -23.34mA -50.34mA
 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

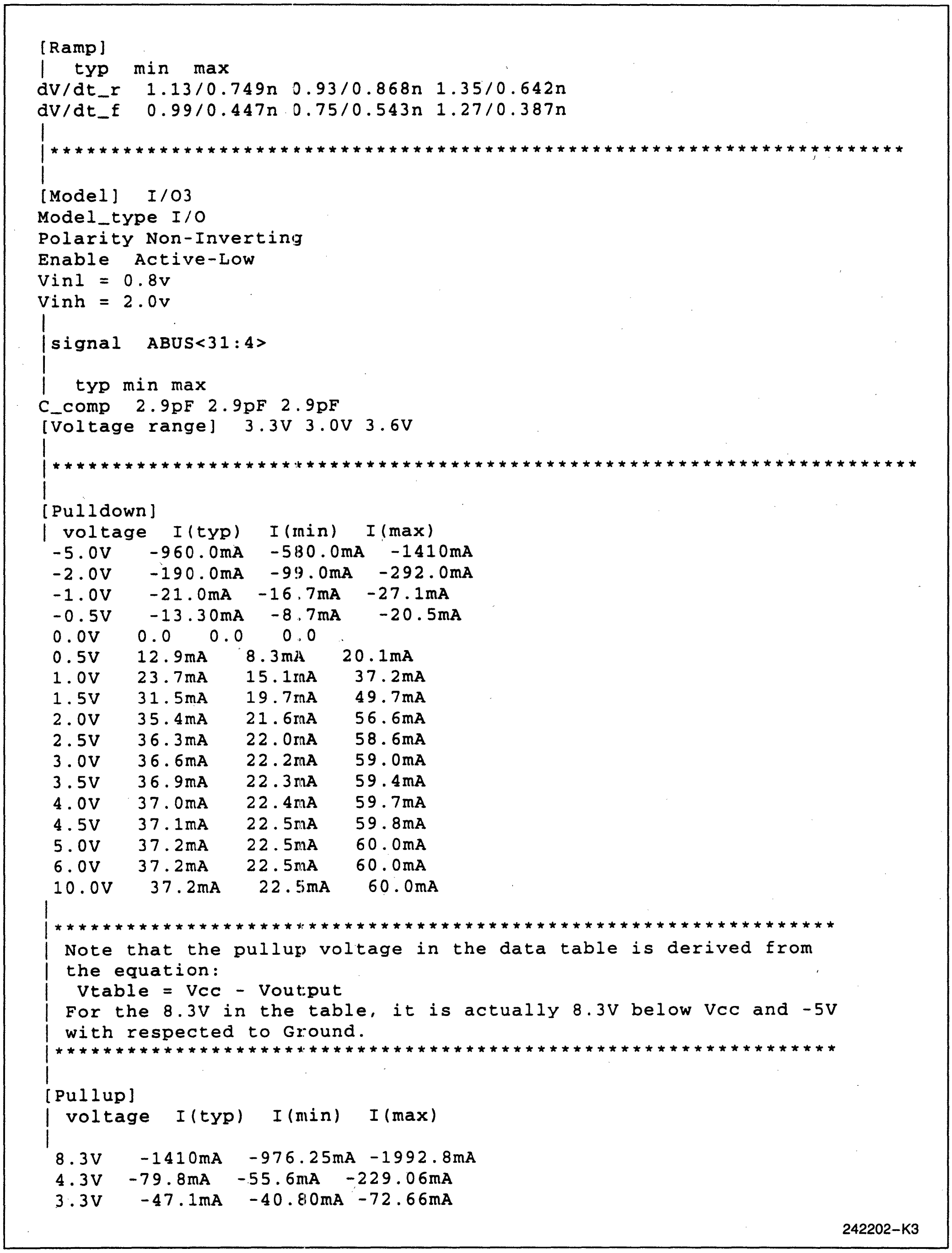
```
 -2.0V  100mA  NA  NA
|
[Ramp]
|  typ  min  max
dV/dt_r  1.13/0.749n 0.93/0.868n 1.35/0.642n
dV/dt_f  0.99/0.447n 0.75/0.543n 1.27/0.387n
|
|***********************************************************************
|
[Model]  I/O2
Model_type I/O
Polarity Non-Inverting
Enable  Active-Low
Vinl = 3.3v
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



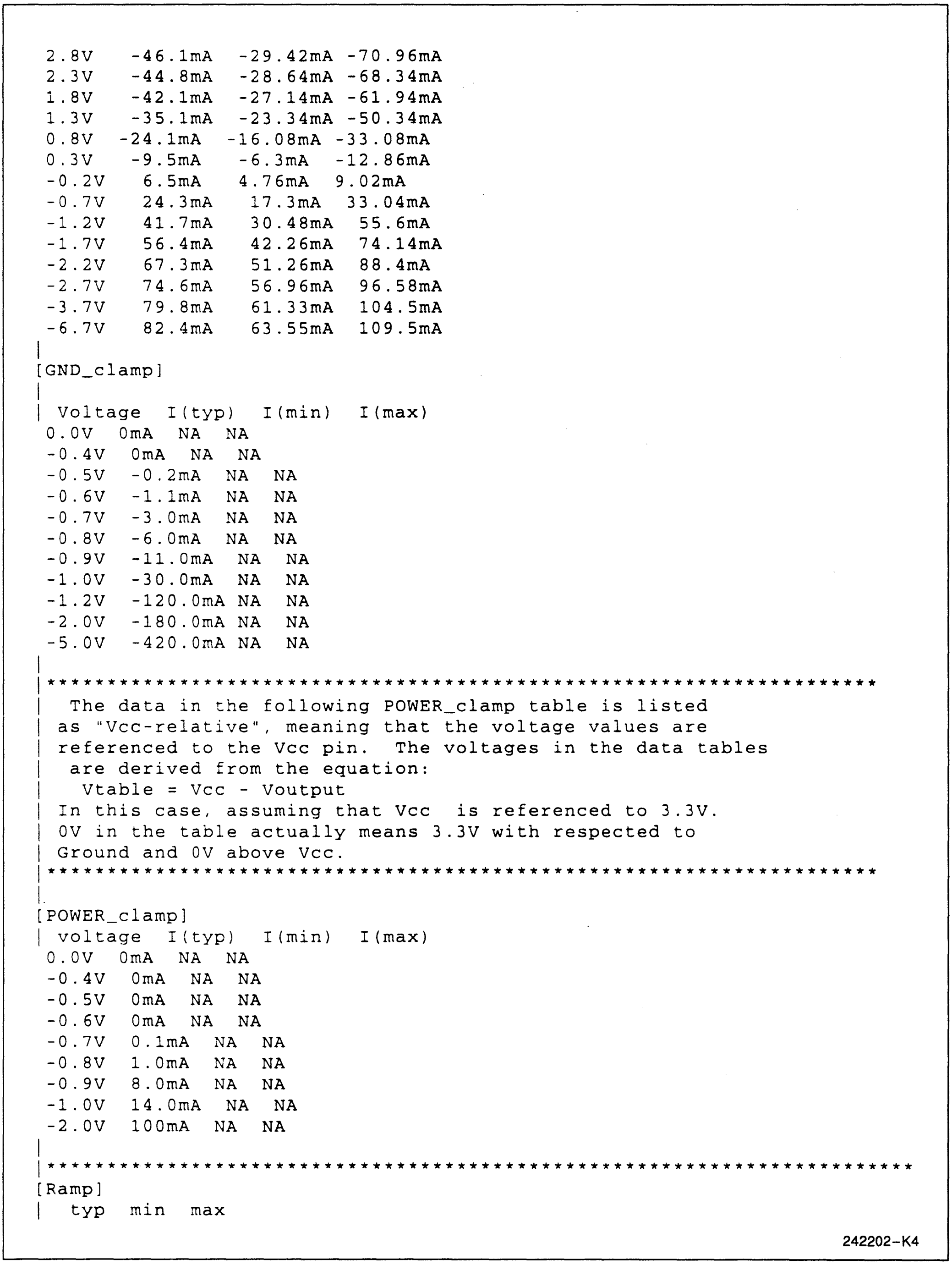
```
 3.3V    -47.1mA   -40.80mA -72.66mA
 2.8V    -46.1mA   -29.42mA -70.96mA
 2.3V    -44.8mA   -28.64mA -68.34mA
 1.8V    -42.1mA   -27.14mA -61.94mA
 1.3V    -35.1mA   -23.34mA -50.34mA
 0.8V   -24.1mA   -16.08mA -33.08mA
 0.3V    -9.5mA    -6.3mA   -12.86mA
 -0.2V    6.5mA    4.76mA   9.02mA
 -0.7V    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



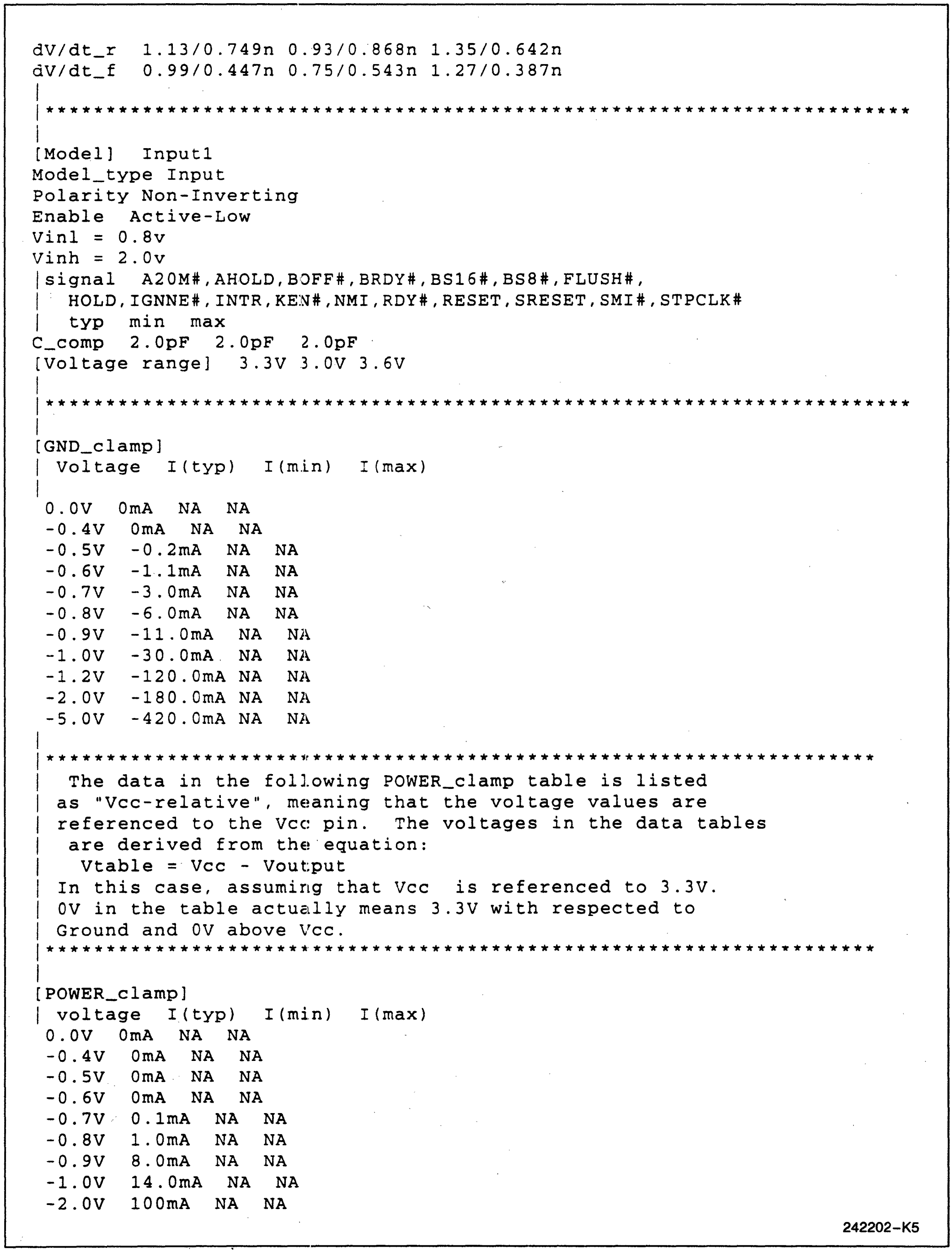
```
[Ramp]
|  typ  min  max
dV/dt_r  1.13/0.749n 0.93/0.868n 1.35/0.642n
dV/dt_f  0.99/0.447n 0.75/0.543n 1.27/0.387n
|
|**************************************************************************
|
[Model]  I/O3
Model_type I/O
Polarity Non-Inverting
Enable  Active-Low
Vinl = 0.8v
Vinh = 2.0v
|
|signal  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



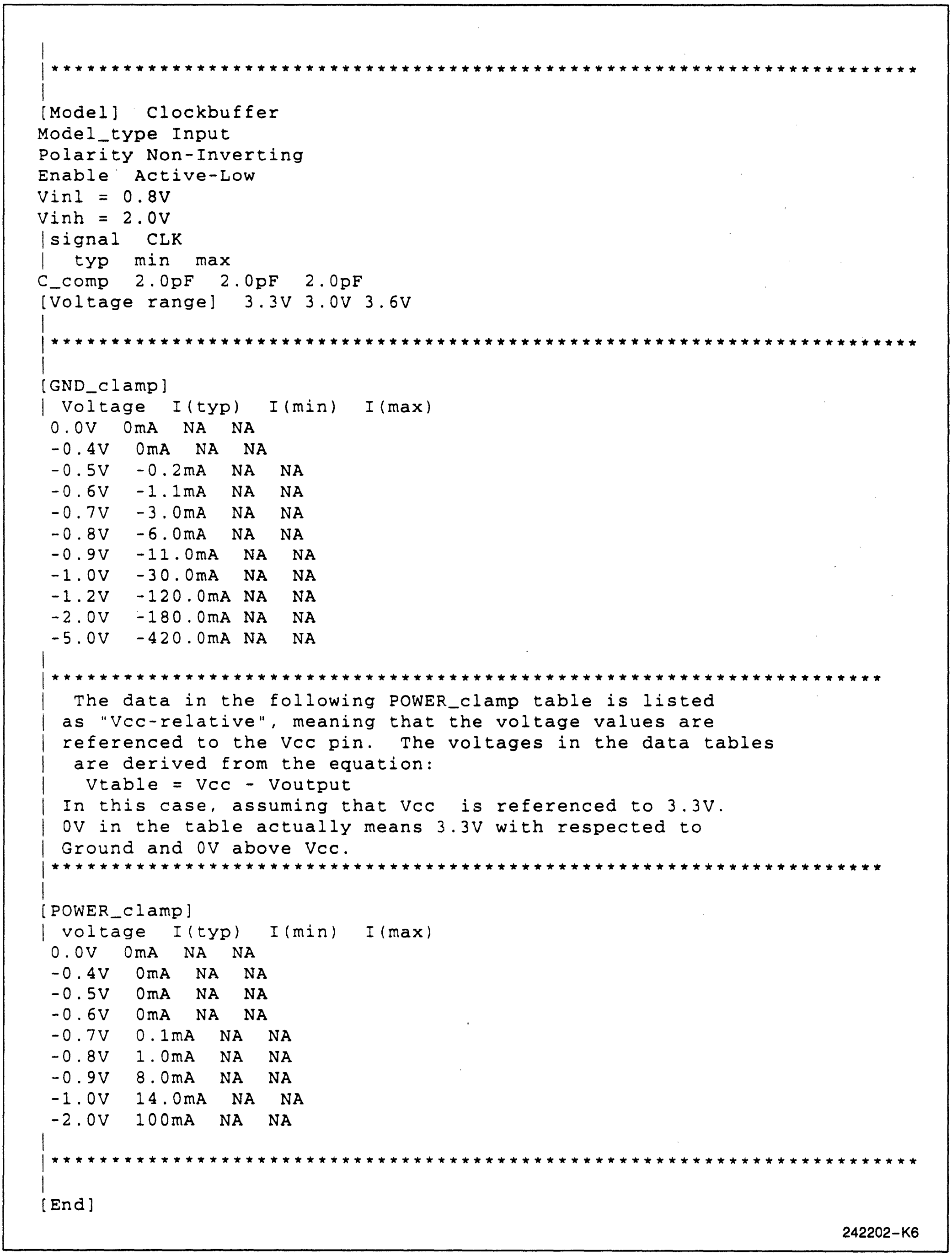
```
 2.8V   -46.1mA  -29.42mA -70.96mA
 2.3V   -44.8mA  -28.64mA -68.34mA
 1.8V   -42.1mA  -27.14mA -61.94mA
 1.3V   -35.1mA  -23.34mA -50.34mA
 0.8V  -24.1mA  -16.08mA -33.08mA
 0.3V   -9.5mA   -6.3mA  -12.86mA
 -0.2V   6.5mA   4.76mA  9.02mA
 -0.7V   24.3mA   17.3mA  33.04mA
 -1.2V   41.7mA   30.48mA  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



```
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]
```

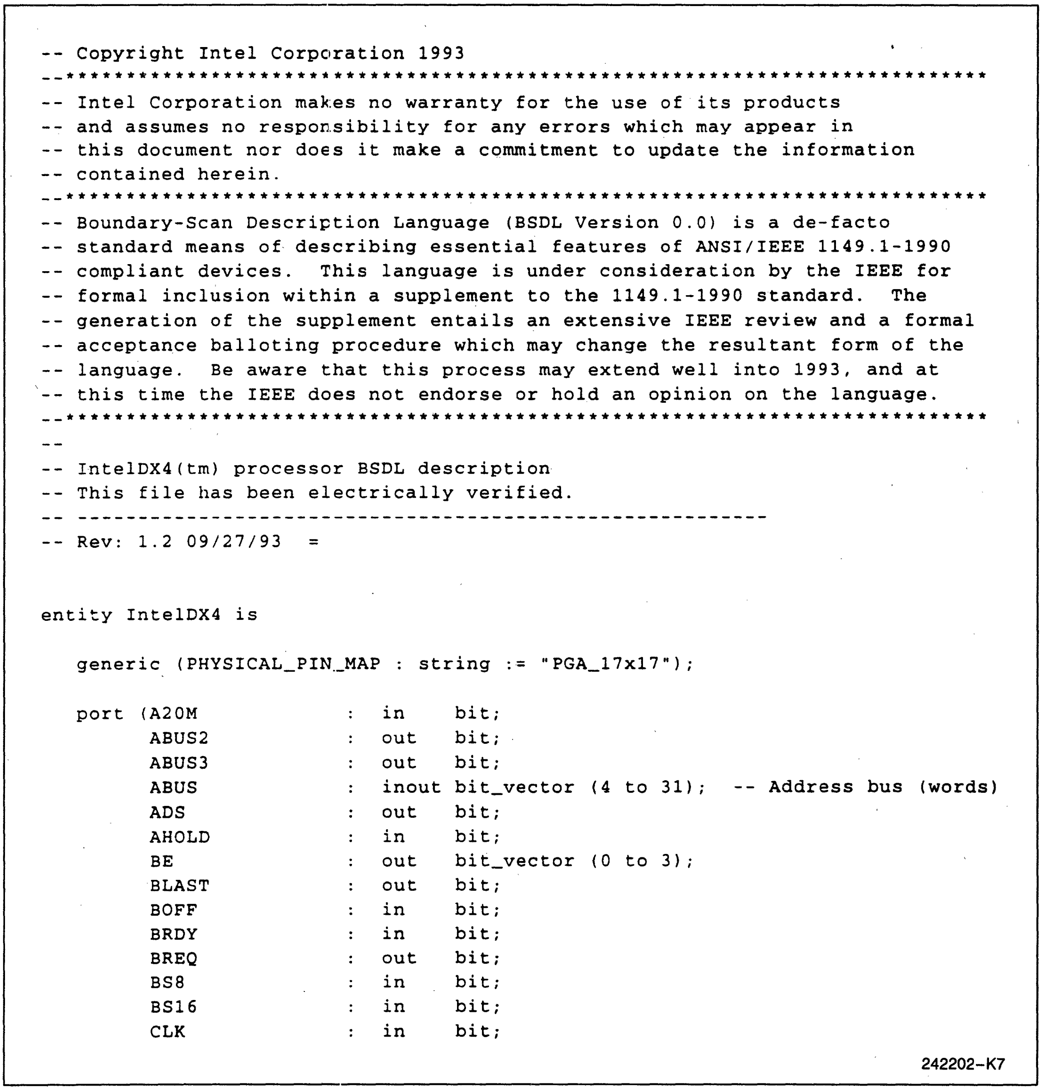
242202-K6

APPENDIX D
BSDL LISTINGS

Below is a listing of a boundary scan description language (BSDL) file for the 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 Boubdary Scan," for a complete description of BSDL instructions and usage.

IntelDX4 Processor Listing

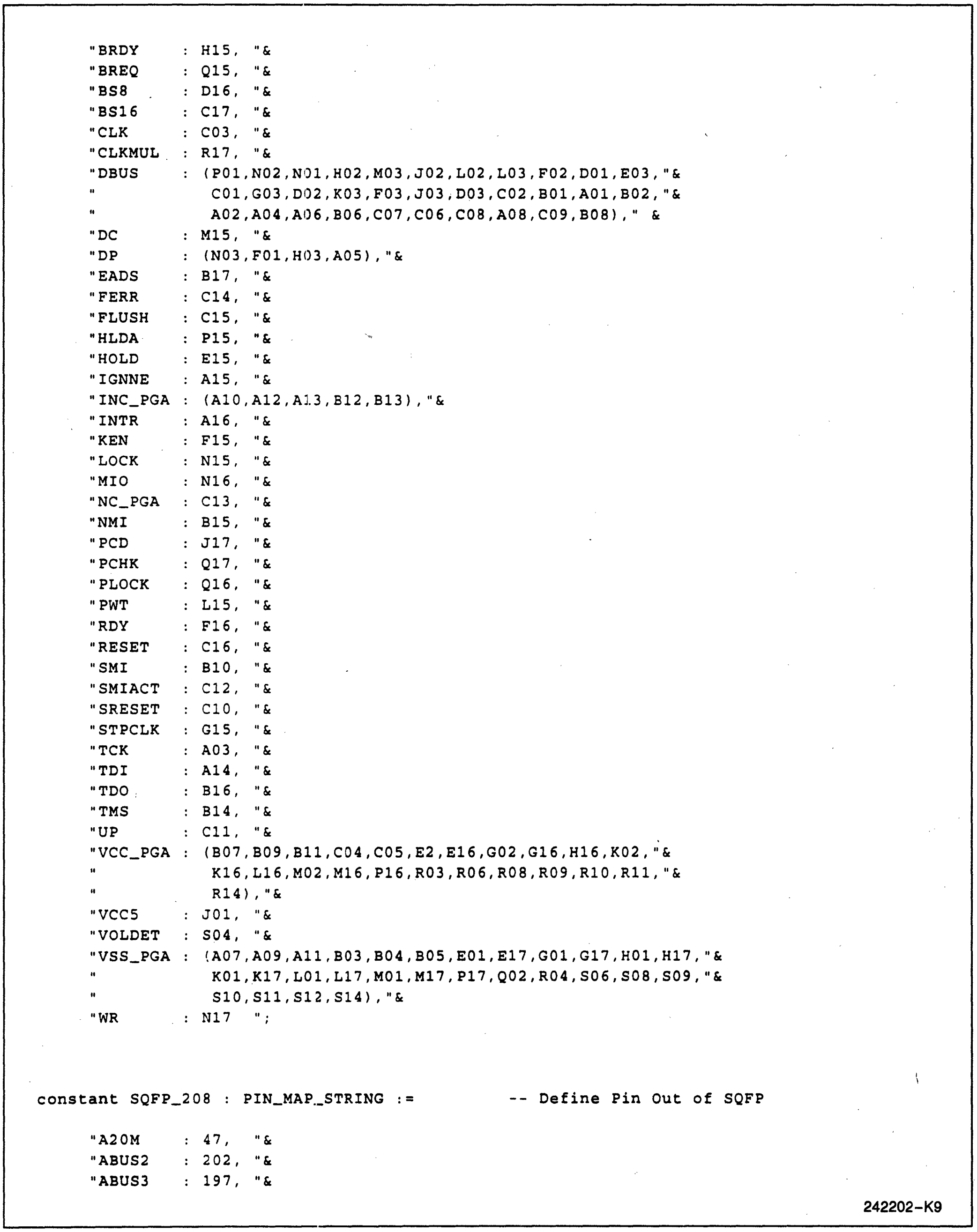


```
-- Copyright Intel Corporation 1993
--*************************************************************************
-- Intel Corporation makes no warranty for the use of its products
-- and assumes no responsibility for any errors which may appear in
-- this document nor does it make a commitment to 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.
--*************************************************************************
--
-- IntelDX4(tm) processor BSDL description
-- This file has been electrically verified.
-- ---------------------------------------------------------
-- Rev: 1.2 09/27/93  =


entity 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;
```

242202–K7

```
         CLKMUL           :  in     bit;
         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_vector (1 to 5);  -- Internal NC PGA
         INTR             :  in     bit;
         KEN              :  in     bit;
         LOCK             :  out    bit;
         MIO              :  out    bit;
         NC_PGA           :  linkage bit;  -- No Connect for PGA
         NC_SQFP          :  linkage bit_vector (1 to 7);  -- NC SQFP
         NMI              :  in     bit;
         PCD              :  out    bit;
         PCHK             :  out    bit;
         PLOCK            :  out    bit;
         PWT              :  out    bit;
         RDY              :  in     bit;
         RESET            :  in     bit;
         SMI              :  in     bit;
         SMIACT           :  out    bit;
         SRESET           :  in     bit;
         STPCLK           :  in     bit;
         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
         VCC_SQFP         :  linkage bit_vector (1 to 53);  -- VCC
         VCC5             :  linkage bit;  -- Reference Voltage
         VOLDET           :  linkage bit;  -- Voltage Detect Pin, PGA only
         VSS_PGA          :  linkage bit_vector (1 to 28);  -- VSS
         VSS_SQFP         :  linkage bit_vector (1 to 38);  -- VSS
         WR               :  out    bit);

   use STD_1149_1_1990.all;

   attribute PIN_MAP of 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,"&
         "           S07,Q10,S05,R07,Q09,Q03,R05,Q04,Q08,Q05,"&
         "           Q07,S03,Q06,R02,S02,S01,R01,P02,P03,Q01),"&
         "ADS     : S17, "&
         "AHOLD   : A17, "&
         "BE      : (K15,J16,J15,F17),"&
         "BLAST   : R16, "&
         "BOFF    : D17, "&
```

242202-K8



```
     "BRDY     : H15,  "&
     "BREQ     : Q15,  "&
     "BS8      : D16,  "&
     "BS16     : C17,  "&
     "CLK      : C03,  "&
     "CLKMUL   : R17,  "&
     "DBUS     : (P01,N02,N01,H02,M03,J02,L02,L03,F02,D01,E03,"&
     "            C01,G03,D02,K03,F03,J03,D03,C02,B01,A01,B02,"&
     "            A02,A04,A06,B06,C07,C06,C08,A08,C09,B08)," &
     "DC       : M15,  "&
     "DP       : (N03,F01,H03,A05),"&
     "EADS     : B17,  "&
     "FERR     : C14,  "&
     "FLUSH    : C15,  "&
     "HLDA     : P15,  "&
     "HOLD     : E15,  "&
     "IGNNE    : A15,  "&
     "INC_PGA  : (A10,A12,A13,B12,B13),"&
     "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      : A03,  "&
     "TDI      : A14,  "&
     "TDO      : B16,  "&
     "TMS      : B14,  "&
     "UP       : C11,  "&
     "VCC_PGA  : (B07,B09,B11,C04,C05,E2,E16,G02,G16,H16,K02,"&
     "            K16,L16,M02,M16,P16,R03,R06,R08,R09,R10,R11,"&
     "            R14),"&
     "VCC5     : J01,  "&
     "VOLDET   : S04,  "&
     "VSS_PGA  : (A07,A09,A11,B03,B04,B05,E01,E17,G01,G17,H01,H17,"&
     "            K01,K17,L01,L17,M01,M17,P17,Q02,R04,S06,S08,S09,"&
     "            S10,S11,S12,S14),"&
     "WR       : N17   ";


constant SQFP_208 : PIN_MAP_STRING :=         -- Define Pin Out of SQFP

     "A20M     : 47,   "&
     "ABUS2    : 202,  "&
     "ABUS3    : 197,  "&
```

242202-K9

```
"ABUS     : (196,195,193,192,190,187,186,182,180,178,"&
"            177,174,173,171,166,165,164,161,160,159,"&
"            158,154,153,152,151,149,148,147)," &
"ADS      : 203, "&
"AHOLD    : 17,  "&
"BE       : (31,32,33,34),"&
"BLAST    : 204, "&
"BOFF     : 6,   "&
"BRDY     : 5,   "&
"BREQ     : 30,  "&
"BS8      : 8,   "&
"BS16     : 7,   "&
"CLK      : 24,  "&
"CLKMUL   : 11,  "&
"DBUS     : (144,143,142,141,140,130,129,126,124,123,119,"&
"            118,117,116,113,112,108,103,101,100,99,93,"&
"            92,91,87,85,84,83,79,78,75,74),"&
"DC       : 39,  "&
"DP       : (145,125,109,90),"&
"EADS     : 46,  "&
"FERR     : 66,  "&
"FLUSH    : 49,  "&
"HLDA     : 26,  "&
"HOLD     : 16,  "&
"IGNNE    : 72,  "&
"INTR     : 50,  "&
"KEN      : 13,  "&
"LOCK     : 207, "&
"MIO      : 37,  "&
"NC_SQFP  : (63,64,67,70,71,96,127),"&
"NMI      : 51,  "&
"PCD      : 41,  "&
"PCHK     : 4,   "&
"PLOCK    : 206, "&
"PWT      : 40,  "&
"RDY      : 12,  "&
"RESET    : 48,  "&
"SMI      : 65,  "&
"SMIACT   : 59,  "&
"SRESET   : 58,  "&
"STPCLK   : 73,  "&
"TCK      : 18,  "&
"TDI      : 168, "&
"TDO      : 68,  "&
"TMS      : 167, "&
"UP       : 194, "&
"VCC_SQFP : (2,9,14,19,20,22,23,25,29,35,38,42,44,45,54,"&
"            56,60,62,69,77,80,82,86,89,95,98,102,106,111,"&
"            114,121,128,131,133,134,136,137,139,150,155,"&
"            162,163,169,172,176,179,183,185,188,191,198,"&
"            200,205),"&
"VCC5     : 3,   "&
"VSS_SQFP : (1,10,15,21,28,36,43,52,53,55,57,61,76,81,88,94,"&
"            97,104,105,107,110,115,120,122,132,135,138,146,"&
"            156,157,170,175,181,184,189,199,201,208),"&
"WR       : 27 ";
```

242202-L0

```
   attribute Tap_Scan_In of    TDI : signal is true;
   attribute Tap_Scan_Out of   TDO : signal is true;
   attribute Tap_Scan_Mode of  TMS : signal is true;

   attribute Tap_Scan_Clock of TCK : signal is (25.0e6, BOTH);

   attribute Instruction_Length of IntelDX4 : entity is 4;

   attribute Instruction_Opcode of 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 IntelDX4 : entity is "0001";

   -- there is no Instruction_Disable attribute for IntelDX4

   attribute Instruction_Private of IntelDX4 : entity is "private";

   attribute Idcode_Register of IntelDX4: entity is
-- ********************************************
      "0000"                    &  --version
      "1000001010001000"&  --new part number
      "00000001001"     &  --manufacturers identity
      "1";                         --required by the standard

   attribute Instruction_Usage of IntelDX4 : entity is
      "RUNBIST (registers BIST; "&
      "result 0;"         &
      "clock CLK in Run_Test_Idle;"&
      "length 1600000)";

   attribute Register_Access of IntelDX4 : entity is
      "BIST[1] (RUNBIST)";

--{*************************************************************}
--{  The first cell is closest to TDO                           }
--{*************************************************************}

   attribute Boundary_Length of IntelDX4 : entity is 109;
   attribute Boundary_Cells of IntelDX4 : entity is "BC_2, BC_1, BC_6";

   attribute Boundary_Register of IntelDX4 : entity is
      "0        (BC_2,  ABUS2,          output3,        X,  107,  1,  Z),"&
      "1        (BC_2,  ABUS3,          output3,        X,  107,  1,  Z),"&
      "2        (BC_6,  ABUS(4),        bidir,          X,  107,  1,  Z),"&
      "3        (BC_6,  ABUS(5),        bidir,          X,  107,  1,  Z),"&
      "4        (BC_1,  UP,             input,          X),"          &
      "5        (BC_6,  ABUS(6),        bidir,          X,  107,  1,  Z),"&
      "6        (BC_6,  ABUS(7),        bidir,          X,  107,  1,  Z),"&
      "7        (BC_6,  ABUS(8),        bidir,          X,  107,  1,  Z),"&
```

242202–L1

```
"8       (BC_6,  ABUS(9),      bidir,      X,  107,  1,  Z),"&
"9       (BC_6,  ABUS(10),     bidir,      X,  107,  1,  Z),"&
"10      (BC_6,  ABUS(11),     bidir,      X,  107,  1,  Z),"&
"11      (BC_6,  ABUS(12),     bidir,      X,  107,  1,  Z),"&
"12      (BC_6,  ABUS(13),     bidir,      X,  107,  1,  Z),"&
"13      (BC_6,  ABUS(14),     bidir,      X,  107,  1,  Z),"&
"14      (BC_6,  ABUS(15),     bidir,      X,  107,  1,  Z),"&
"15      (BC_6,  ABUS(16),     bidir,      X,  107,  1,  Z),"&
"16      (BC_6,  ABUS(17),     bidir,      X,  107,  1,  Z),"&
"17      (BC_6,  ABUS(18),     bidir,      X,  107,  1,  Z),"&
"18      (BC_6,  ABUS(19),     bidir,      X,  107,  1,  Z),"&
"19      (BC_6,  ABUS(20),     bidir,      X,  107,  1,  Z),"&
"20      (BC_6,  ABUS(21),     bidir,      X,  107,  1,  Z),"&
"21      (BC_6,  ABUS(22),     bidir,      X,  107,  1,  Z),"&
"22      (BC_6,  ABUS(23),     bidir,      X,  107,  1,  Z),"&
"23      (BC_6,  ABUS(24),     bidir,      X,  107,  1,  Z),"&
"24      (BC_6,  ABUS(25),     bidir,      X,  107,  1,  Z),"&
"25      (BC_6,  ABUS(26),     bidir,      X,  107,  1,  Z),"&
"26      (BC_6,  ABUS(27),     bidir,      X,  107,  1,  Z),"&
"27      (BC_6,  ABUS(28),     bidir,      X,  107,  1,  Z),"&
"28      (BC_6,  ABUS(29),     bidir,      X,  107,  1,  Z),"&
"29      (BC_6,  ABUS(30),     bidir,      X,  107,  1,  Z),"&
"30      (BC_6,  ABUS(31),     bidir,      X,  107,  1,  Z),"&
"31      (BC_6,  DP(0),        bidir,      X,  108,  1,  Z),"&
"32      (BC_6,  DBUS(0),      bidir,      X,  108,  1,  Z),"&
"33      (BC_6,  DBUS(1),      bidir,      X,  108,  1,  Z),"&
"34      (BC_6,  DBUS(2),      bidir,      X,  108,  1,  Z),"&
"35      (BC_6,  DBUS(3),      bidir,      X,  108,  1,  Z),"&
"36      (BC_6,  DBUS(4),      bidir,      X,  108,  1,  Z),"&
"37      (BC_6,  DBUS(5),      bidir,      X,  108,  1,  Z),"&
"38      (BC_6,  DBUS(6),      bidir,      X,  108,  1,  Z),"&
"39      (BC_6,  DBUS(7),      bidir,      X,  108,  1,  Z),"&
"40      (BC_6,  DP(1),        bidir,      X,  108,  1,  Z),"&
"41      (BC_6,  DBUS(8),      bidir,      X,  108,  1,  Z),"&
"42      (BC_6,  DBUS(9),      bidir,      X,  108,  1,  Z),"&
"43      (BC_6,  DBUS(10),     bidir,      X,  108,  1,  Z),"&
"44      (BC_6,  DBUS(11),     bidir,      X,  108,  1,  Z),"&
"45      (BC_6,  DBUS(12),     bidir,      X,  108,  1,  Z),"&
"46      (BC_6,  DBUS(13),     bidir,      X,  108,  1,  Z),"&
"47      (BC_6,  DBUS(14),     bidir,      X,  108,  1,  Z),"&
"48      (BC_6,  DBUS(15),     bidir,      X,  108,  1,  Z),"&
"49      (BC_6,  DP(2),        bidir,      X,  108,  1,  Z),"&
"50      (BC_6,  DBUS(16),     bidir,      X,  108,  1,  Z),"&
"51      (BC_6,  DBUS(17),     bidir,      X,  108,  1,  Z),"&
"52      (BC_6,  DBUS(18),     bidir,      X,  108,  1,  Z),"&
"53      (BC_6,  DBUS(19),     bidir,      X,  108,  1,  Z),"&
"54      (BC_6,  DBUS(20),     bidir,      X,  108,  1,  Z),"&
"55      (BC_6,  DBUS(21),     bidir,      X,  108,  1,  Z),"&
"56      (BC_6,  DBUS(22),     bidir,      X,  108,  1,  Z),"&
"57      (BC_6,  DBUS(23),     bidir,      X,  108,  1,  Z),"&
"58      (BC_6,  DP(3),        bidir,      X,  108,  1,  Z),"&
"59      (BC_6,  DBUS(24),     bidir,      X,  108,  1,  Z),"&
"60      (BC_6,  DBUS(25),     bidir,      X,  108,  1,  Z),"&
"61      (BC_6,  DBUS(26),     bidir,      X,  108,  1,  Z),"&
"62      (BC_6,  DBUS(27),     bidir,      X,  108,  1,  Z),"&
"63      (BC_6,  DBUS(28),     bidir,      X,  108,  1,  Z),"&
```

242202-L2

```
      "64       (BC_6,  DBUS(29),       bidir,          X,  108,  1,  Z),"&
      "65       (BC_6,  DBUS(30),       bidir,          X,  108,  1,  Z),"&
      "66       (BC_6,  DBUS(31),       bidir,          X,  108,  1,  Z),"&
      "67       (BC_2,  STPCLK,         input,          X),"&
      "68       (BC_1,  IGNNE,          input,          X),"&
      "69       (BC_2,  FERR,           output3,        X,  105,  1,  Z),"&
      "70       (BC_1,  SMI,            input,          X),"&
      "71       (BC_2,  SMIACT,         output3,        X,  106,  1,  Z),"&
      "72       (BC_1,  SRESET,         input,          X),"&
      "73       (BC_1,  NMI,            input,          X),"&
      "74       (BC_1,  INTR,           input,          X),"&
      "75       (BC_1,  FLUSH,          input,          X),"&
      "76       (BC_1,  RESET,          input,          X),"&
      "77       (BC_1,  A20M,           input,          X),"&
      "78       (BC_1,  EADS,           input,          X),"&
      "79       (BC_2,  PCD,            output3,        X,  106,  1,  Z),"&
      "80       (BC_2,  PWT,            output3,        X,  106,  1,  Z),"&
      "81       (BC_2,  DC,             output3,        X,  106,  1,  Z),"&
      "82       (BC_2,  MIO,            output3,        X,  106,  1,  Z),"&
      "83       (BC_2,  BE(3),          output3,        X,  106,  1,  Z),"&
      "84       (BC_2,  BE(2),          output3,        X,  106,  1,  Z),"&
      "85       (BC_2,  BE(1),          output3,        X,  106,  1,  Z),"&
      "86       (BC_2,  BE(0),          output3,        X,  106,  1,  Z),"&
      "87       (BC_2,  BREQ,           output3,        X,  105,  1,  Z),"&
      "88       (BC_2,  WR,             output3,        X,  106,  1,  Z),"&
      "89       (BC_2,  HLDA,           output3,        X,  105,  1,  Z),"&
      "90       (BC_1,  CLK,            input,          X),"&
      "91       (BC_1,  AHOLD,          input,          X),"&
      "92       (BC_1,  HOLD,           input,          X),"&
      "93       (BC_1,  KEN,            input,          X),"&
      "94       (BC_1,  RDY,            input,          X),"&
      "95       (BC_1,  CLKMUL,         input,          X),"&
      "96       (BC_1,  BS8,            input,          X),"&
      "97       (BC_1,  BS16,           input,          X),"&
      "98       (BC_1,  BOFF,           input,          X),"&
      "99       (BC_1,  BRDY,           input,          X),"&
      "100      (BC_2,  PCHK,           output3,        X,  105,  1,  Z),"&
      "101      (BC_2,  LOCK,           output3,        X,  106,  1,  Z),"&
      "102      (BC_2,  PLOCK,          output3,        X,  106,  1,  Z),"&
      "103      (BC_2,  BLAST,          output3,        X,  106,  1,  Z),"&
      "104      (BC_2,  ADS,            output3,        X,  106,  1,  Z),"&
      "105      (BC_2,  *,      control, 1),"&                 -- DISMISC
      "106      (BC_2,  *,      control, 1),"&                 -- DISBUS
      "107      (BC_2,  *,      control, 1),"&                 -- DISABUS
      "108      (BC_2,  *,      control, 1)";                  -- DISWR

  end IntelDX4;
```

242202-L3

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.

Example E-1. Initialization of Segment Registers within SMM

```
;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-L4

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.

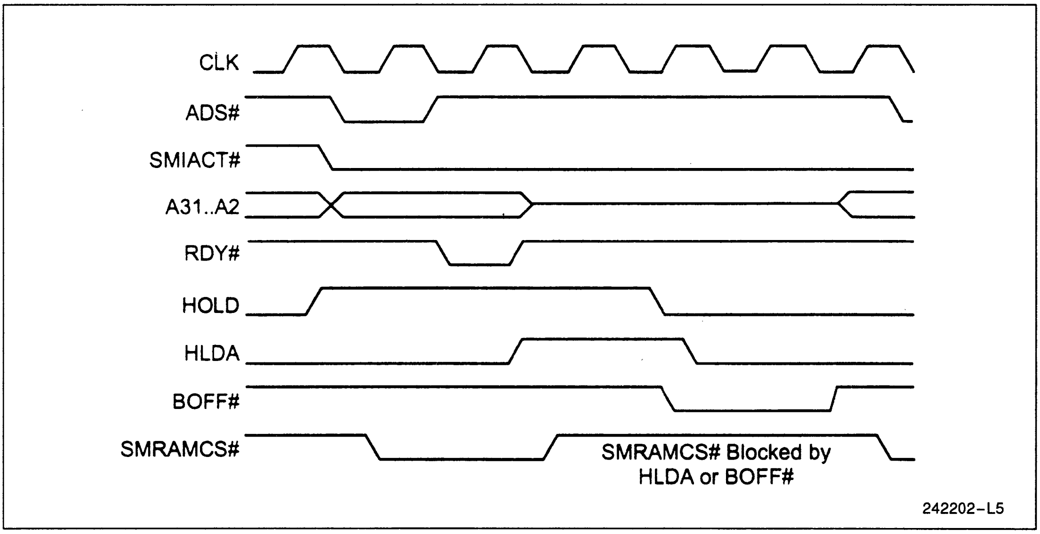


Figure E-1. Blocking Other Bus Masters from Accessing SMRAM

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.

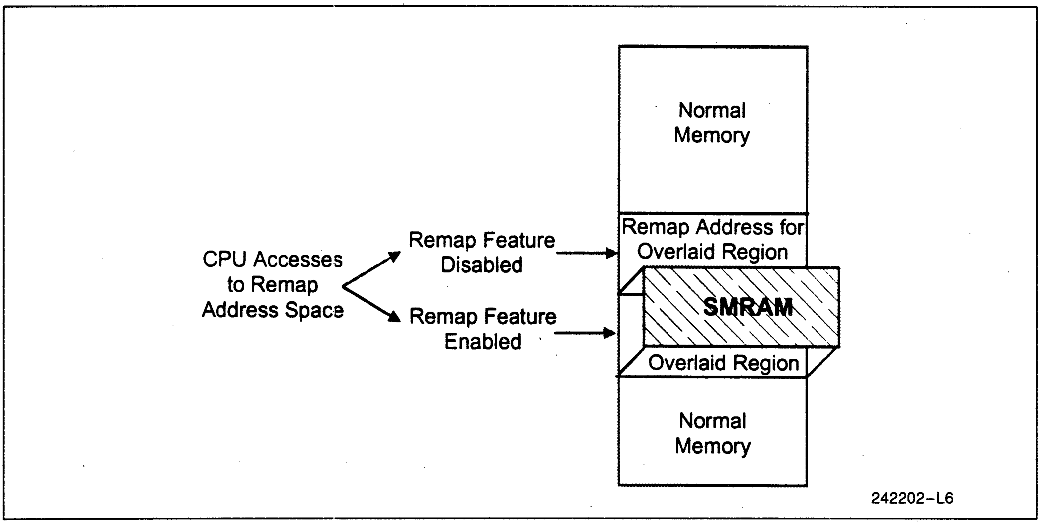


Figure E-2. Remapping Memory That Is Overlaid by SMRAM

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 64 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.

Example E-2. Saving the FPU State in 32-Bit Protected Mode

```
;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–L7

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.

Example E-3. Restoring the FPU State from a 32-Bit Protected Mode Save

```
;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-L8

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.

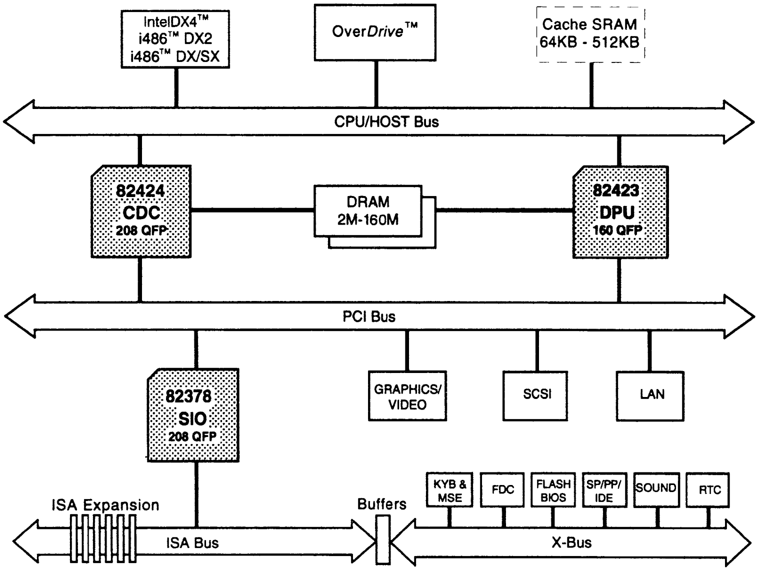
82420 PCIset

Intel's 82420 PCIset enables workstation level of performance for Intel486™ CPU desktop systems. The Peripheral Component Interconnect Bus (PCI) is driving a new architecture for PC's—eliminating the I/O bottleneck of standard expansion busses. PCI provides a glueless interface for high performance peripherals such as graphics, SCSI, LAN and video to be placed onto a fast local bus. By utilizing this technology and incorporating read/write bursts along with write buffers into the 82420 PCIset, a new level of performance is now possible for today's Intel486 CPU desktop systems.

The Intel 82420 PCIset is comprised of three components: the 82424ZX Cache DRAM Controller (CDC), the 82423TX Data Path Unit (DPU), and the 82378ZB System I/O (SIO). The CDC and DPU provide the core system architecture while the SIO is a PCI master/slave agent which bridges the core architecture to the ISA standard expansion bus. Intel also offers two components, the 82374EB (ESC) and 82375EB (PCEB), that work in conjunction to bridge the PCI bus to the EISA expansion bus. Refer to the ESC and PCEB data sheets for information regarding the EISA bridge components.

The chip set supports the Intel486 CPU family as well as Intel's future OverDrive™ processor for the Intel486 DX2 CPU. The high performance memory subsystem supports concurrent operation between PCI bus masters while the CPU accesses memory. An integrated second level cache can be programmed for write-through or write-back operation.

82420 PCIset



290467–1

Product Highlights

82424ZX—Cache DRAM Controller (CDC)

- Concurrent Linefill during Copyback Cycles
- Supports Intel486 CPU Family and OverDrive Processors
- Supports OverDrive Upgrade
- 64K–512K Level 2 Cache Support
- Level 2 Cache Configurable as Write-Back or Write-Through
- 208-Pin QFP Package

82423TX—Data Path Unit (DPU)

- Highly Integrated
- Four Dword Write Buffers
- Zero Wait-States for CPU Write Cycles
- PCI Burst Write Capability
- 160-Pin QFP Package

82378ZB—System I/O Component (SIO)

- Supports Fast DMA Type A, B, or F Cycles
- Supports DMA Scatter/Gather
- Arbitration Logic for Four PCI Masters
- Reusable across Multiple Platforms
- Directly Drives Six External ISA Slots
- Integrates Many of Today's Common I/O Functions
- 208-Pin QFP Package

Product Description

The 82424ZX Cache DRAM Controller (CDC) is a single-chip bridge from the CPU to the PCI bus. It provides the integrated functionality of a second level cache controller, a DRAM controller, and a PCI bus controller. It also features an optimized memory subsystem. The CDC is a dual ported device with one port as the host port and the other as the PCI port.

The 82423TX Data Path Unit (DPU) integrates the host data, memory data, and PCI data interface, DPU control/parity and four deep posted write buffers. With glue and buffers integrated directly into the DPU, the Intel 82420 PCIset reduces board space requirements. The DPU's posted write buffers allow CPU write cycles to be executed as 0 wait-states.

The 82378ZB System I/O (SIO) is a dual ported device which acts as a bridge between the PCI and standard ISA I/O bus. The SIO integrates the functionality of an ISA controller, PCI controller, fast 32-bit DMA controller, and standard system I/O functions.

ADVANCE INFORMATION

82420EX PCISET DATA SHEET 82425EX PCI SYSTEM CONTROLLER (PSC) AND 82426EX ISA BRIDGE (IB)

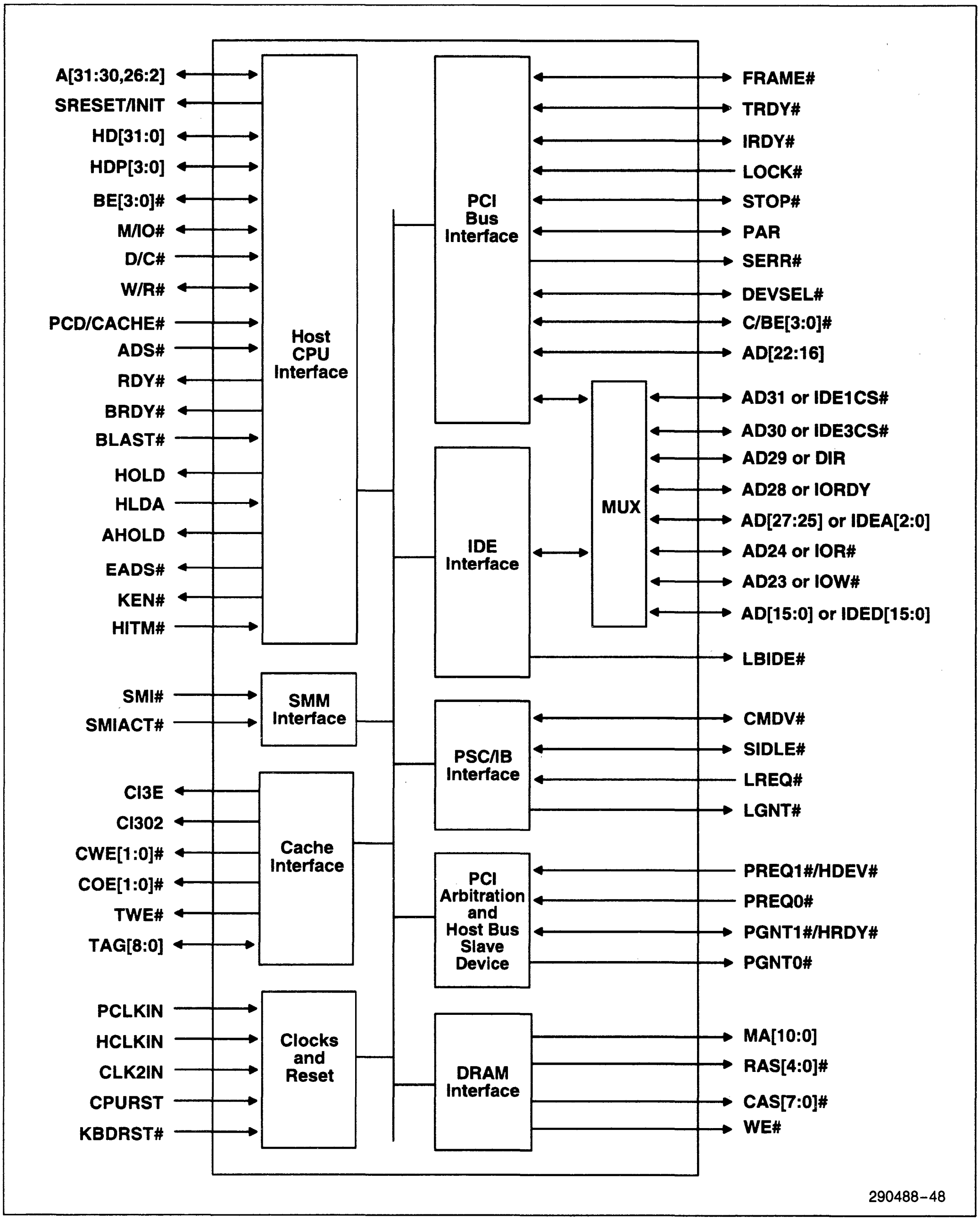
- Host CPU
  - 25–33 MHz Intel486™ and OverDrive™ Processors
  - L1 Write-Back Support
- Integrated DRAM Controller
  - 1 to 128 MByte Main Memory
  - 70 ns Fast Page Mode DRAM SIMMs Supported
  - Supports 256 KByte, 1 MByte, and 4 MByte Double and Single Sided SIMMs
  - Read Page Hit Timing of 3-2-2-2 at 33 MHz
  - Burst Mode PCI Master Accesses
  - Decoupled Refresh Reduces DRAM Latency
  - Five RAS Lines
- Integrated L2 Cache Controller
  - Write-Back and Write-Through Cache Policies
  - Direct Mapped Organization
  - 64, 128, 256 or 512 KByte Cache Sizes
  - Programmable Zero Wait-State L2 Cache Read and Write Accesses
  - Two Banks Interleaved or a Single Bank Non-Interleaved Operation
  - No VALID Bit Required
- 25/33 MHz PCI Bus Interface
  - Two Bus Masters
  - PCI Auto Configuration Support
- Host/PCI Bridge
  - Converts Back-to-Back Sequential Memory Writes to PCI Burst Writes
  - CPU Memory Write Posting to PCI
- PCI Local Bus IDE Interface
  - Supports Mode 3 Timing
- Programmable Attribute Map for First 1 MByte of Main Memory
- 100% ISA Compatible
  - Directly Drives 5 ISA Slots
- Two 8237 DMA Controllers
  - 7 DMA Channels
  - 27-bit Addressability
  - Compatible DMA Transfers
- One 82C54 Timer/Counter
  - System Timer
  - Refresh Request
  - Speaker Tone
- Two 82C59 Interrupt Controllers
  - 14 Interrupts
  - Edge/Level Sense is Programmable per Channel
  - PCI Interrupt Steering for Plug and Play Compatibility
- X-Bus Peripheral Support
  - RTC, KBC, BIOS Chip Selects
  - Control for Lower X-Bus Transceiver
  - Integrates Mouse Interrupt
  - Coprocessor Error Reporting
- Non-Maskable Interrupts (NMI)
  - PCI System Errors
  - Main Memory Parity Errors
  - ISA Parity Errors
- System Power Management (Intel SMM Support)
  - Programmable System Management Interrupt (SMI)—Hardware Events, Software Events, EXTSMI#
  - Programmable CPU Clock Control
  - Fast On/Off Mode
- Generates System Clocks
- 160-Pin QFP Package for IB
- 208-Pin QFP Package for PSC

December 1994
Order Number: 290488-003

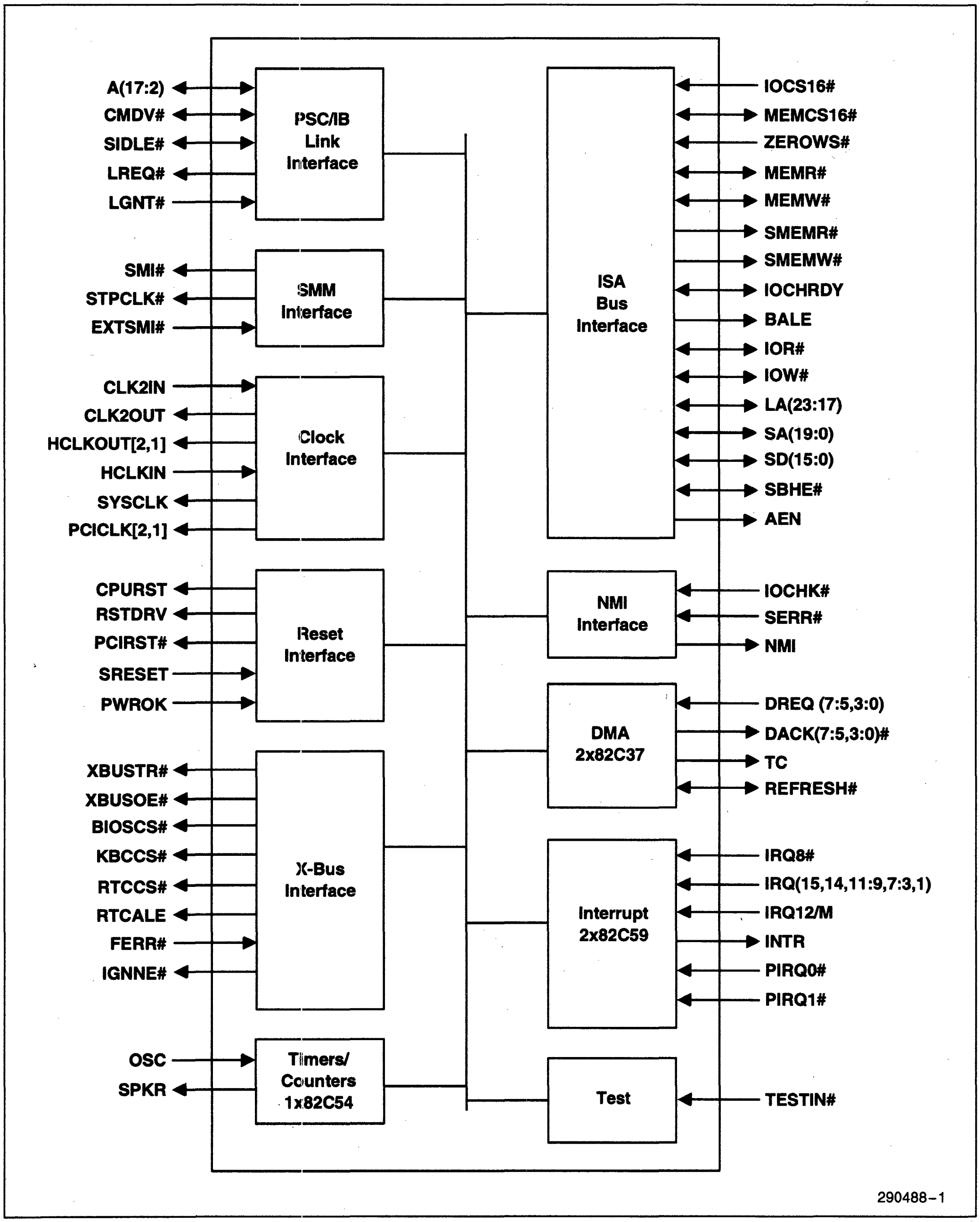
The 82420EX PCIset is the foundation for the **Value Flexible Motherboard** solution for entry-level Intel486™ processor-based PCI systems. The Value Flexible Motherboard solution, including 82420EX, Intel486 processor, 82091AA Advanced Integrated Peripherals, 82C42 Keyboard Controller, Flash BIOS, and Plug & Play software, drives PCI into the mainstream. The 82420EX PCIset is a highly integrated solution enabling low cost, small form factor motherboard designs. All Intel486 processors and upgrades are supported, including L1 write-back and Intel SMM power management. PCI Local Bus IDE is incorporated for higher performance IDE at no additional cost.

The 82420EX was designed from the ground up for PCI performance. It consists of two components—the 82425EX PCI System Controller (PSC) and the 82426EX ISA Bridge (IB). The PSC integrates the L2 cache controller and the DRAM controller. The cache controller supports both write-through and write-back cache policies and cache sizes from 64 KBytes to 512 KBytes in an interleaved or non-interleaved configuration. The DRAM controller interfaces main memory to the Host Bus and the PCI Bus. The PSC supports a two-way interleaved DRAM organization for optimum performance. Up to ten single-sided SIMMs or four double-sided and two single-sided SIMMs provide a maximum of 128 MBytes of main memory. The PSC provides memory write posting to PCI for enhanced CPU-to-PCI memory write performance. In addition, the PSC provides a high performance PCI Local Bus IDE interface.

The IB is the bridge between the ISA Bus and Host Bus, and integrates the common I/O functions found in today's ISA-based PC systems—a seven channel DMA controller, two 82C59 interrupt controllers, an 8254 timer/counter, Intel SMM power management support, and control logic for NMI generation. The IB also provides the decode for external BIOS, real time clock, and keyboard controller. Edge/Level interrupts and interrupt steering are supported for PCI plug and play compatibility. The IB integrates the ISA address and data path, reducing TTL and system cost. In addition, the integration of system clock generation logic eliminates the need for external host and PCI clock drivers.



82425EX PCI System Controller (PSC) Block Diagram



82426EX ISA Bridge (IB) Block Diagram

intel®

AP-469

APPLICATION NOTE

Cache and Memory Design Considerations for the Intel486™ DX2 Microprocessor

TAUFIK T. MA
INTEL TECHNICAL MARKETING

May 1992

PRELIMINARY

Order Number: 241261-001

Cache and Memory Design Considerations for the Intel486™ DX2 Microprocessor

CONTENTS PAGE

CONTENTS PAGE

FIGURES

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

TABLES

1.0 INTRODUCTION

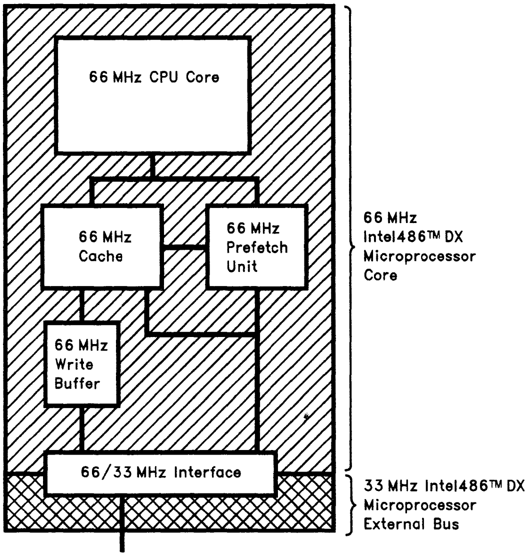
This section discusses CPU performance optimization techniques for the Intel486™ DX2 microprocessor. The reader should be familiar with the Intel486™ DX microprocessor as well as knowledgeable about memory systems and cache architectures. For further reference, the reader is directed to the following documents and application notes (corresponding Intel order number is shown in parentheses):

- Intel486™ DX2 Microprocessor Data Book (241245-001)
- Intel486™ DX Microprocessor Data Book (240440-004)
- Intel486™ DX Microprocessor Hardware Reference Manual (240552-001)
- Cache Tutorial 1991 (296543-002)
- AP447: A Memory Subsystem for the Intel486™ Family of Microprocessors including Second Level Cache (240799-002)
- 485TurboCache Module Intel486™ DX Microprocessor Cache Upgrade (240722-002)

2.0 The High-Performance Intel486™ DX2 Microprocessor

The Intel486 DX2 Microprocessor is functionally equivalent to the now-familiar Intel486™ DX Microprocessor. However, the Intel486 DX2 CPU's internal core runs at twice the frequency of its external bus. This architecture enables a very high level of performance while, at the same time, maintaining straightforward system design.

The Intel486 DX2 CPU is partitioned such that the cache and write buffers operate at the full core speed as illustrated in Figure 2.1. As such, the processor is only slowed to the external bus speed on cache misses and when the write-buffers are full. The Intel486 DX2 CPU's external bus interface is identical to its predecessor, i.e. all system cycles emanating from the CPU look exactly as if they would from the Intel486 DX CPU. The Intel486 DX2 microprocessor includes additional features such as JTAG boundary scan and power-down capability that are not covered in this section.


66 MHz CPU Core
66 MHz Cache
66 MHz Prefetch Unit
66 MHz Write Buffer
66/33 MHz Interface
66 MHz Intel486™ DX Microprocessor Core
33 MHz Intel486™ DX Microprocessor External Bus

241261-1

Figure 2.1. The 66MHz Intel486™ DX2 CPU Internal Architecture

Performance optimization for the Intel486 DX2 CPU is subtly different than for the Intel486 DX CPU due to the difference in the internal architecture. This is evident if you consider that external memory latencies now affect the full speed core by twice as many CPU clocks (refer to Figure 2.2). In other words, the memory system should really be designed to satisfy the data throughput demands of a 50/66MHz CPU. The next few sections examine the situation in detail so that educated trade-offs can be made during system design. The discussions will focus on CPU-cache memory performance; I/O performance and other architectural issues are not addressed in this applications note.

In an ideal system, all CPU cycles operate at zero-wait states and the theoretical maximum performance of the CPU is achieved. However, short of spending a lot of money on SRAMs, a real system always falls short of the ideal zero wait-states. There are many cache-memory designs that differ in both architecture and implementation. However, it may be impossible to design a system that performs better than all others across all applications; different applications generate different types of CPU bus activity and the cache-memory system will perform differently in each case. A range of statistical parameters may be used to illustrate this point:

- Internal Cache (L1) Hit Rate
- Number of prefetches
- Number of operand reads
- Number of operand writes
- Bus Utilization (amount of time spent using the CPU bus).

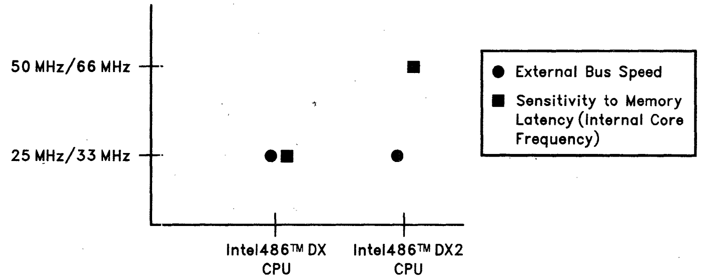
These parameters are examined next and will be useful information for cache-memory design trade-offs.

2.1 Internal Cache Hit Rates

The Intel486 DX2 CPU maintains the same unified code/data, four-way interleaved, 8K-byte internal cache as the Intel486 DX CPU. The internal cache (L1) hit rate is shown in Figure 2.3 for some different operating systems and applications. These hit rates were obtained from instruction traces captured from specific applications. Note that the L1 hit rate for the Intel486 DX2 CPU will be almost identical to that for the Intel486 DX CPU; the 2X-internal frequency does not significantly alter the cache miss statistics.

The DOS applications included Auto Cad, Lotus123, Excel, Turbo C, and Flight Simulator. Lotus123 had the highest hit rate at 99% while Auto Cad was the lowest at 89.6%. The Windows 3.0 results included instructions executed while clipping an image, drawing a dialog box, executing Excel and while executing Pagemaker. The category UNIX-iSPEC refers to applications within the UNIX SPEC benchmark suite that are mostly integer intensive, whereas UNIX-fSPEC refers to those which are floating point intensive. The last category UNIX-TP1 refers to the hit rates typical while running the TP1 transaction processing benchmark.

These results illustrate the different nature of different software on the bus characteristics. Single-threaded DOS applications have a typically higher hit rate compared to the multi-tasking nature of UNIX and Windows benchmarks. The UNIX floating point benchmarks show the lowest hit rates; this is partly due to the large data structures typical of floating point applications that do not fit well into the L1 cache. It is also due to the nature of the operations performed on those data structures; i.e. the application does not exhibit very much temporal or spatial locality.



241261-2

Figure 2.2. The Intel486™ DX2 Microprocessor is More Sensitive to Memory Latency than Its Predecessor

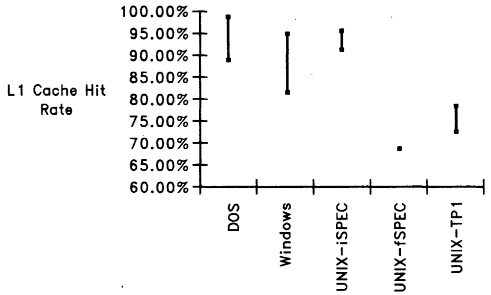


Figure 2.3. There will be a Range of L1 Hit Rates for Different Applications/Operating Systems

TP1 is a UNIX multiuser multithreaded application with heavy amounts of disk and I/O as well as computation. The L1 cache hit rate is low since it has many active contexts due to multiple requests per user.

High L1 hit rates are typical of many prevalent and commonly used DOS benchmarks - some have hit rates approaching 100%. This is unfortunate since they fail to properly account for the more realistic external cache and memory demands of most DOS and Windows applications. These demands will continue to be true with the advent of newer graphical-user-interface-based multi-tasking operating systems and applications. With these benchmarks, there is a danger of misrepresenting system performance for the Intel486 DX microprocessor. For the Intel486 DX2 microprocessor, this misrepresentation can be even more damaging since L1 cache misses incur a penalty that is twice the number of external clock cycles - since the CPU core now runs twice as fast internally. In other words, the Intel486 DX2 CPU is twice as sensitive to wait states compared to the Intel486 DX CPU. To properly gauge the external cache and memory performance, more demanding benchmarks (e.g. UNIX SPECmarks) or real application benchmarks should be used.

2.2 Bus Cycle Mix

Different applications cause the CPU to generate a different number of reads, writes and instruction prefetches. The reads and prefetches are filtered by the internal L1 cache before reaching the external CPU bus. All writes propagate through to the external bus since the internal cache follows a write-through protocol. This is shown in Figure 2.4 for an instruction trace captured from an integer SPEC benchmark.

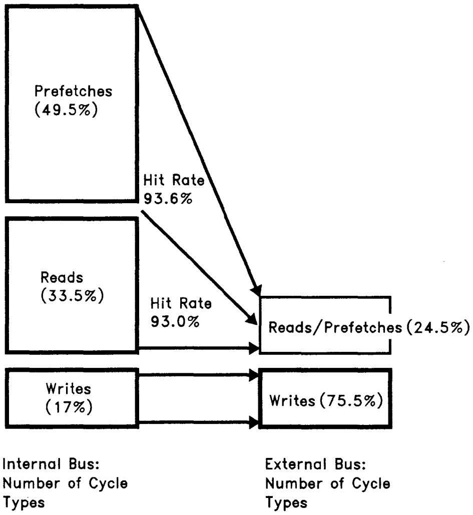


Figure 2.4. The Effect of L1 Cache Hits on the External Bus Cycle Mix for an Integer SPEC Benchmark

As with the Intel486 DX CPU, the bus cycle mix for the Intel486 DX2 CPU consists of mostly write cycles. However, the exact ratio of reads to writes is again application dependent. For example, for Lotus123 that has a L1 hit rate of 99%, writes make up 99.5% of external bus cycles.

2.3 Bus Utilization

Bus utilization refers to the amount of time that the CPU spends executing bus cycles for a given application. It is a measure of the amount of bus traffic generated by a particular application on the CPU's external bus. This metric is illustrated in Figure 2.5 where the CPU bus is busy for 75% of the twelve external clock cycles shown.

Bus utilization is dependent on the application and the external cache/memory system. Different applications generate different amounts of bus traffic. For example, some applications may have small data structures that fit easily within the internal cache and therefore smaller amounts of external bus cycles are generated.

The Intel486 DX2 CPU will have a larger percentage of bus utilization for the same application as compared to the Intel486 DX CPU. This is due to the faster CPU core that can now operate twice as fast and that will try to generate twice as many bus cycles in the same amount of time. However, since the external CPU bus remains at a 1X-frequency, it experiences heavy amounts of bus traffic as it trys to keep up with the 2X-internal core. In other words, with faster internal core execution, less time is spent idling on the external bus.

Different external cache/memory systems also affect the bus utilization; faster cache/memory systems allow CPU cycles to complete faster and therefore free up the bus more.

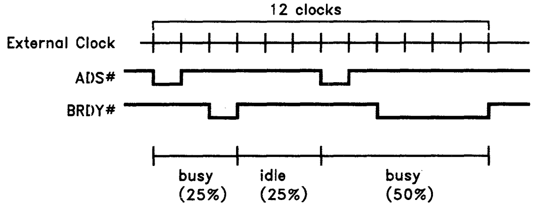


Figure 2.5. Definition of Bus Utilization

2.4 Profiles of Some Applications

The Intel486 DX2 CPU bus characteristics of some different applications and operating systems are shown in Tables 2.1 through 2.3. These results are derived from traces captured from the actual applications. These traces were subsequently used in a CPU-cache-memory simulator to extract the desired information. The results shown here assume an ideal zero wait-state memory system; i.e. all bus cycles complete in zero wait-states.

Table 2.1. Bus Profiles of UNIX Applications with a Zero Wait-State Memory System

UNIX Applications	SPEC1	GCC	UNIXMIX1
Total Number of CPU clocks simulated	12.14M	12.54M	12.44M
Overall L1 hit rate	91.4%	90.5%	94.3%
Prefetch hit rate	93.6%	91.7%	94.8
Read hit rate	93.0%	90.3%	94.2%
Write hit rate	82.0%	85.7%	93.5%
Number of external bus cycles	1.12M	0.882M	1.18M
% bus code prefetches	14.5%	22.6%	10.2%
% bus data reads	10.0%	20.0%	8%
% bus data writes	75.5%	57.4%	81.8%
Bus Utilization	51.6%	46.4%	51.4%

Table 2.2. Bus Profiles of Windows Applications with a Zero Wait-State Memory System

Windows Applications	Word	Excel—Calc	Pagemaker
Total Number of CPU clocks simulated	27.02M	7.39M	30.06M
Overall L1 hit rate	95.2%	78.4%	88.1%
Prefetch hit rate	96.9%	76.0%	81.8%
Read hit rate	98.0%	85.7%	95.2%
Write hit rate	87.7%	69.7%	83.5%
Number of external bus cycles	2.43M	0.654M	3.04M
% bus code prefetches	4.2%	27.9%	19.3%
% bus data reads	3.4%	15.1%	6.7%
% bus data writes	92.4%	57.0%	74%
Bus Utilization	40.9%	60.1%	56.0%

Table 2.3. Bus Profiles of DOS Applications with a Zero Wait-State Memory System

DOS Applications	Excel	Turbo C	Auto Cad
Total Number of CPU clocks simulated	11.1M	13.9M	16.1M
Overall L1 hit rate	98.2%	95.6%	89.3%
Prefetch hit rate	98.8%	94.2%	87.7%
Read hit rate	97.9%	98.1%	96.5%
Write hit rate	97.4%	93.8%	81.3%
Number of external bus cycles	1.06M	1.18M	1.77M
% bus code prefetches	2.1%	11.2%	11.8%
% bus data reads	3.1%	3.0%	4.0%
% bus data writes	94.8%	85.8%	84.2%
Bus Utilization	41.0%	40.6%	55.5%

The UNIX applications are described as:

- SPEC1: A mixture of integer SPEC benchmark suite programs running concurrently.
- GCC: SPEC benchmark suite GNU C compiler, compiling itself.
- UNIXMIX1: A mixture of UNIX utility programs like awk and grep, running concurrently.

The Microsoft Windows 3.0 applications are described as:

- Word: Microsoft Word for Windows converting a document for import (no VGA activity, includes kernel calls).
- Excel—Calc: Microsoft Excel for Windows running a calculation.
- Pagemaker: Pagemaker for Windows formatting a document (no VGA activity, includes kernel calls)

The three DOS applications shown are described as:

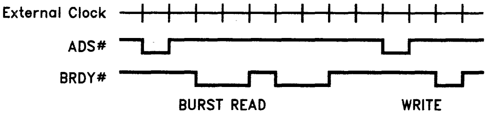
- Excel: Microsoft Excel (DOS version) recalculating a spreadsheet.
- Turbo C: Borland's Turbo C compiler compiling a large C program.
- Auto Cad: Auto Desk's Auto Cad program computing and displaying a drawing (reason for low hit rate)

Note that most DOS applications will typically use the external bus much less than UNIX or Windows applications. On the other hand, heavy duty DOS applications such as Auto Cad will actually exhibit a larger demand for the external bus similar to the demands of UNIX and Windows. Also, recall that the bus utilization numbers shown assume a zero wait-state memory system; more realistic external cache/memory systems with a finite number of wait-states will actually experience a larger percentage of bus utilization. Finally, note that since the L1 hit rate for the DOS applications is high, the external bus mix consists of mostly writes.

2.5 Wait States Explained

Now that we have considered the characteristics of L1 hit rates, bus cycle mix and utilization, we can examine the impact that the cache-memory system has on overall Intel486 DX2 CPU performance. To examine these effects, traces were captured from three applications (one from each operating environment) and were used in a CPU-cache-memory simulator to measure the CPU performance under different conditions. The traces used were SPEC1 (UNIX), Pagemaker (Windows) and Turbo C (DOS). The simulator used is an accurate and convenient method of comparing performance by varying different parameters independently. This allows us to develop some heuristic rules to guide system design. Before continuing, the following convention is defined to denote memory performance:

Notation Convention:
A memory system's performance is abbreviated as:
Lead-off clocks - Burst 2 - Burst 3 - Burst 4, Write clocks
e.g. 3-1-2-1, 3 would look like:



241261-6

A zero-wait-state case corresponds to a 2-1-1-1, 2 memory system.

Figure 2.6. Memory Performance Notation

2.5.1 THE IDEAL ZERO WAIT STATE MEMORY SYSTEM

As a starting point, the ideal zero-wait-state memory system was characterized for three applications. For I/O cycles, it was assumed that a constant 8 wait-states were required. Since the I/O instructions were a small portion of the instruction traces used, any inaccuracy due to this assumption will be insignificant.

The total execution times reported are as follows:

Table 2.4. Number of Internal Clocks (millions) Needed to Complete Application Trace

	SPEC1	Pagemaker	Turbo C
Intel486 DX CPU	10.76	26.68	13.34
Intel486 DX2 CPU	12.14	30.06	13.90

As an example, we can now compare the actual time required to complete the applications between a 66MHz Intel486 DX2 CPU and a 33MHz Intel486 DX CPU. The number of clock cycles is multiplied by 15ns for the Intel486 DX2 CPU and by 30ns for the Intel486 DX CPU.

Table 2.5 Total Execution Time in Seconds

	SPEC1	Pagemaker	Turbo C
Intel486 DX CPU	323 ms	800 ms	400 ms
Intel486 DX2 CPU	182 ms	451 ms	209 ms
Performance Increase	+77%	+77%	+91.4%

Note that although the Intel486 DX2 CPU's internal clock rate is twice that of the corresponding Intel486 DX CPU, the relative improvement is less than 100%. This is due to cache miss reads and write cycles that run at the external bus speed. Note that the improvement is much greater for Turbo C which has a lower cache miss rate and low bus utilization.

2.5.2 ADDING WAIT STATES

As wait states are added to the ideal zero wait state memory system, performance degrades. Three memory parameters are of interest in characterizing the memory performance. They are:

- Number of wait states added to the first ready of a read (a.k.a. the lead-off cycle). e.g. One wait-state with zero wait-state burst = 3-1-1-1
- Number of wait states during the remainder of the burst cycle. e.g. a zero wait-state lead-off with a one wait-state burst = 2-2-2-2
- Number of wait states on a write cycle. e.g. a one wait-state write takes three clocks.

To examine the impact of adding wait-states to each of the parameters above, three series of simulations are done where wait states are added to each memory parameter separately while the other memory parameters are held constant:

	Zero wait-states	1 wait-state	2 wait-states	3 wait-states
• Lead-off Series:	2-1-1-1, 2	3-1-1-1, 2	4-1-1-1, 2	5-1-1-1, 2...
• Burst Series:	2-1-1-1, 2	2-2-2-2, 2	2-3-3-3, 2	2-4-4-4, 2
• Write Series:	2-1-1-1, 2	2-1-1-1, 3	2-1-1-1, 4	2-1-1-1, 5...

As the number of wait states increases, the number of clocks needed to complete the application increases (and the CPU performance decreases). This series of measurements is used to separate out the dependency of the CPU performance on the different memory parameters. This information is useful for subsequent external cache/memory design trade-offs.

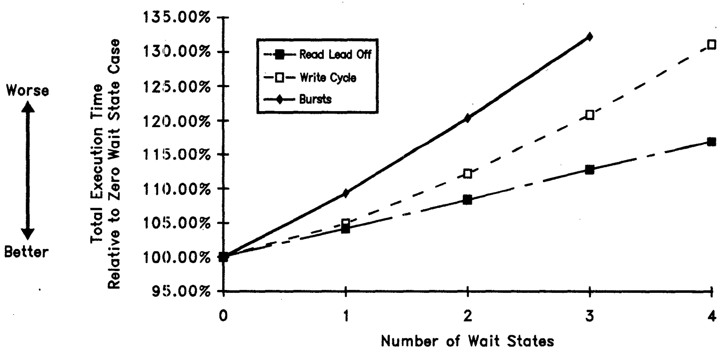
The number of clocks needed to complete the application relative to the zero wait-state case is referred to as the relative total execution time. This metric will be used in the following graphs instead of the reciprocal of total execution time which would be CPU performance; this is so that any inherent linear relationships between wait-states and execution time can be more easily recognized. To translate the total execution time back to CPU performance, the following table is provided for convenience (100% refers to the zero wait-state case):

Table 2.6. CPU Performance versus Total Execution Time

Total Execution Time	100.00%	110.00%	120.00%	130.00%	140.00%	150.00%	160.00%
CPU Performance	100.00%	90.91%	83.33%	76.92%	71.43%	66.67%	62.50%

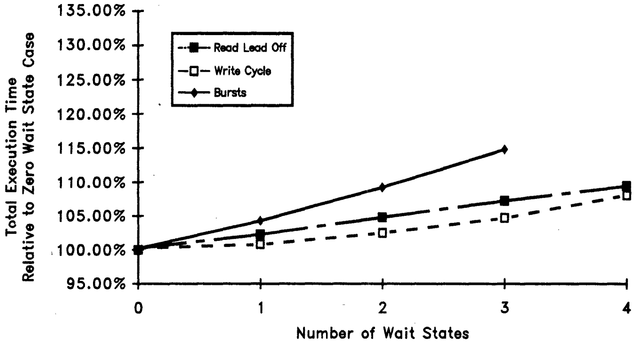
Figures 2.7 and 2.8 show the total execution time as wait states are added for the SPEC1 trace described earlier.

As would be expected, as wait states are added, the relative total execution time for the Intel486 DX2 CPU increases faster than the Intel486 DX CPU. (however, note that the absolute total execution time for a Intel486 DX2 CPU will never be greater than a Intel486 DX CPU of the same external bus speed). Also note that the order of importance of the memory parameters changes between the Intel486 DX and the Intel486 DX2 CPU. For the Intel486 DX CPU, burst performance is the most important, with read-lead-off performance being slightly more important than writes. This conclusion was presented in Chapter 4 of the original Intel486 DX Microprocessor Hardware Reference Manual (Order Number 240552-001). However for the Intel486 DX2 CPU, while burst performance is still the most important, write performance becomes more important than read-lead-off performance.



241261-7

Figure 2.7. Intel486 DX2 CPU Performance Degradation as Wait States are Added - for the SPEC1 Application Trace (UNIX)



241261-8

Figure 2.8. Intel486 DX CPU Performance Degradation as Wait States are Added - for the SPEC1 Application Trace (UNIX)

Figures 2.9 and 2.10 show the total execution time as wait states are added for the GCC trace described earlier.

Compared to the results for the SPEC1 application trace, the GCC results with the Intel486 DX2 CPU still show the burst cycles as being the most sensitive to wait states. However, note that the lead-off cycle is now more important than write cycles. This is due to the small percentage of bus writes (57.4%) while running the GCC application as compared with the SPEC1 trace.

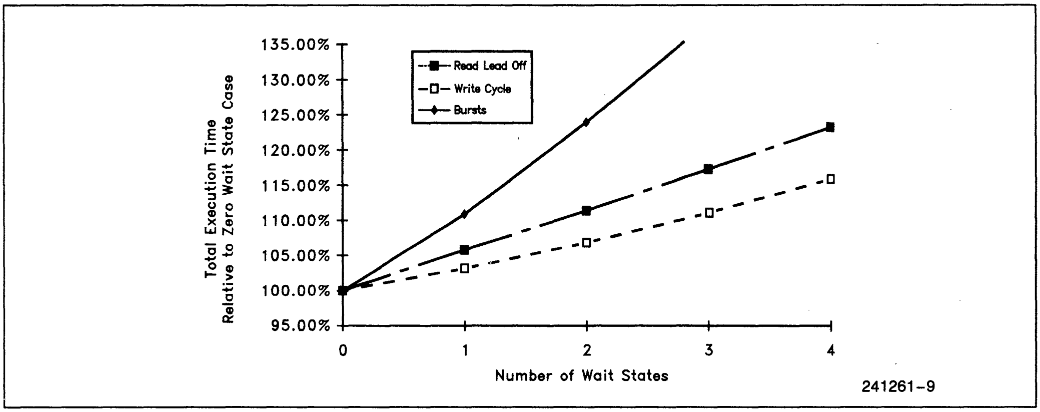


Figure 2.9. Intel486 DX2 CPU Performance Degradation as Wait States are Added - for the GCC Application Trace (UNIX)

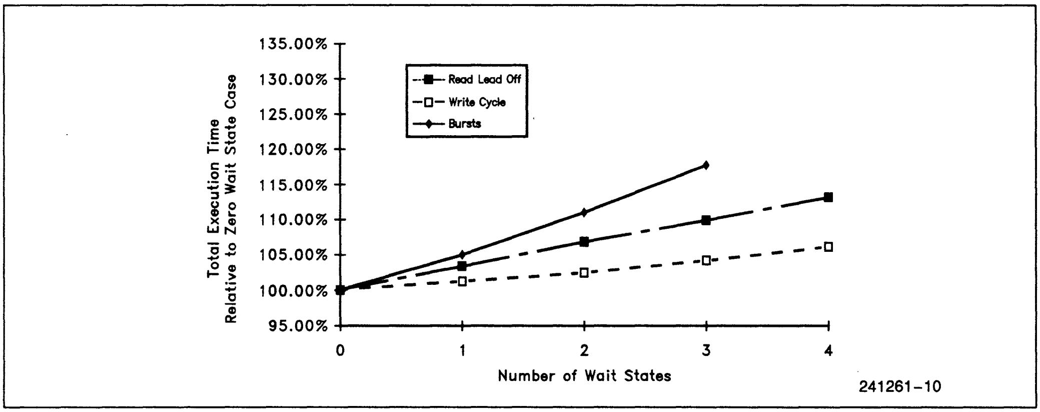


Figure 2.10. Intel486 DX CPU Performance Degradation as Wait States are Added - for the GCC Application Trace (UNIX)

The results for the Pagemaker Application under Windows are shown in Figures 2.11 and 2.12.

The same conclusions can be made for the Pagemaker application under Windows as for the SPEC1 results earlier. Basically, the degradation of the Intel486 DX2 CPU's relative execution time increases faster as wait-states are added as compared to the Intel486 DX CPU.

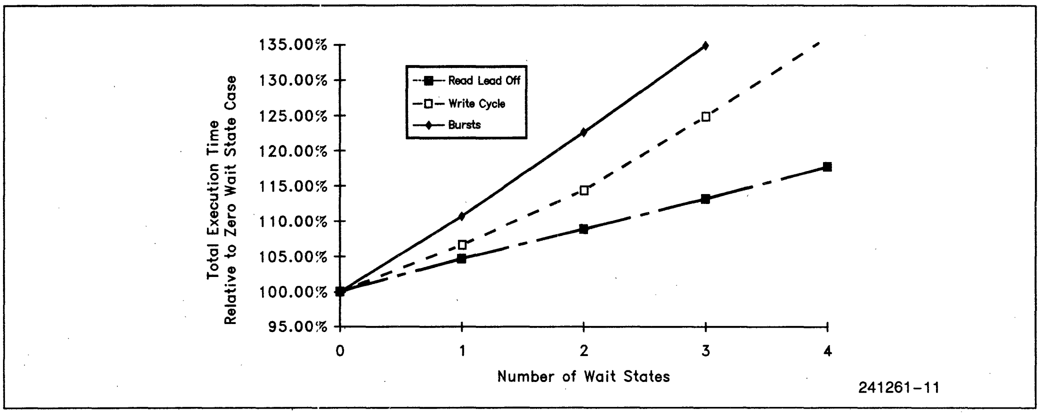


Figure 2.11. Intel486 DX2 CPU Performance Degradation as Wait States are Added - for the Pagemaker Application Trace (Windows)

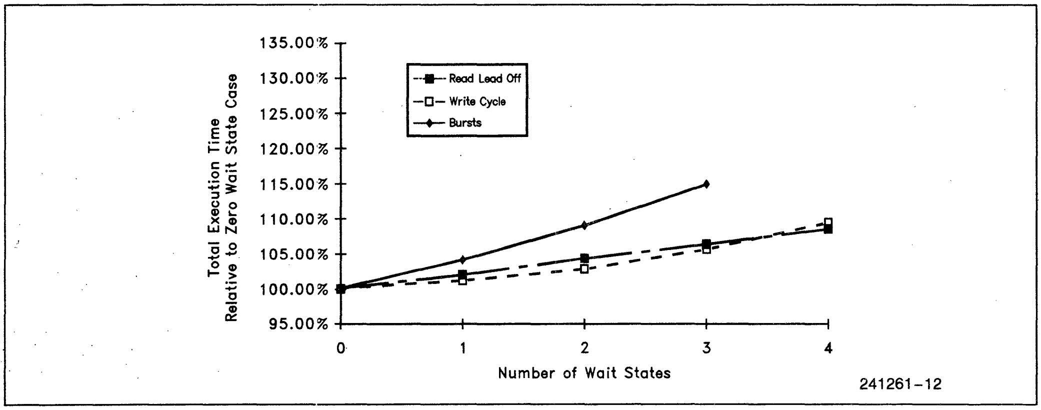


Figure 2.12. Intel486 DX CPU Performance Degradation as Wait States are Added - for the Pagemaker Application Trace (Windows)

Finally, the results for the Turbo C application under DOS are shown in Figures 2.13 and 2.14.

The rate of performance degradation for the Intel486 DX2 CPU with the Turbo C application is less than the UNIX and Windows examples. This is due to the lower external bus utilization of the application. However, the degradation is still about twice what it is for the Intel486 DX CPU. Note that the write importance is about equal to the burst importance in this case. This can be attributed to the greater percentage of writes in the bus cycle mix for this application.

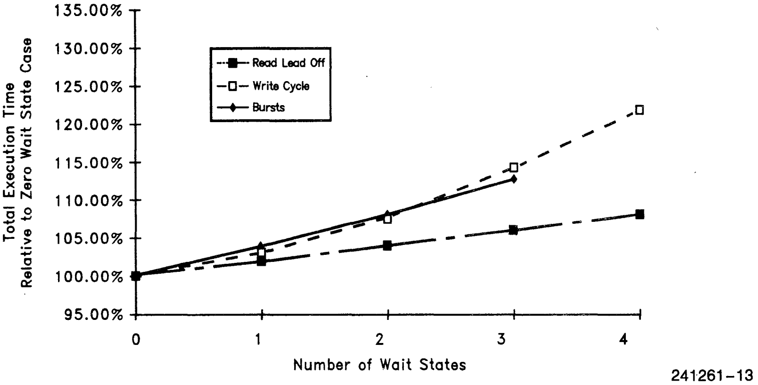


Figure 2.13. Intel486 DX2 CPU Performance Degradation as Wait States are Added - for the Turbo C Application Trace (DOS)

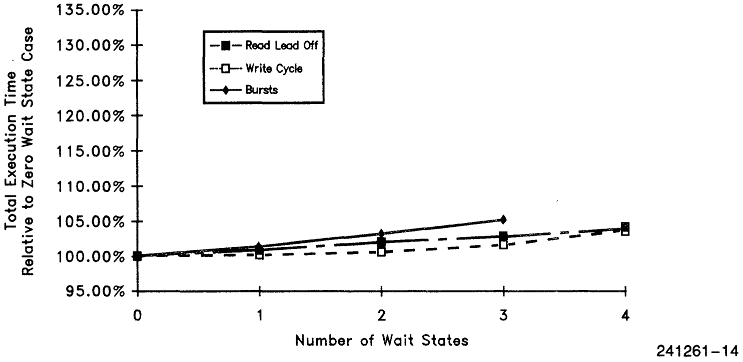


Figure 2.14. Intel486 DX CPU Performance Degradation as Wait States are Added - for the Turbo C Application Trace (DOS)

2.5.3 WAIT STATES AND CPU STALLS

The main reason why the relative performance of the Intel486 DX2 CPU degrades faster than the Intel486 DX CPU is that for every external wait state, two internal clock delays are caused. In fact, a zero wait state cycle on the external bus of the Intel486 DX2 CPU is already a two-wait state cycle as experienced by the 2X-clock internal core as shown in Fig. 2.15. The imaginary 2X-clock versions of the signals ADS# and BRDY# illustrate what the cycle might have looked like if the internal bus frequen cy was equal to the external.

Extending this fact, this means that a one wait-state cycle for the Intel486 DX2 CPU is actually equivalent to a four wait-state cycle for the 2X-clock internal CPU core.

The internal cycle start indication conditions may also have a one internal clock cycle synchronization penalty if it is active in the wrong phase of the external clock (also shown in Figure 2.15).

As the effective number of wait states increases, the CPU will stall program execution differently for each of the three memory parameters described above. The stall conditions for each memory parameter are elaborated below.

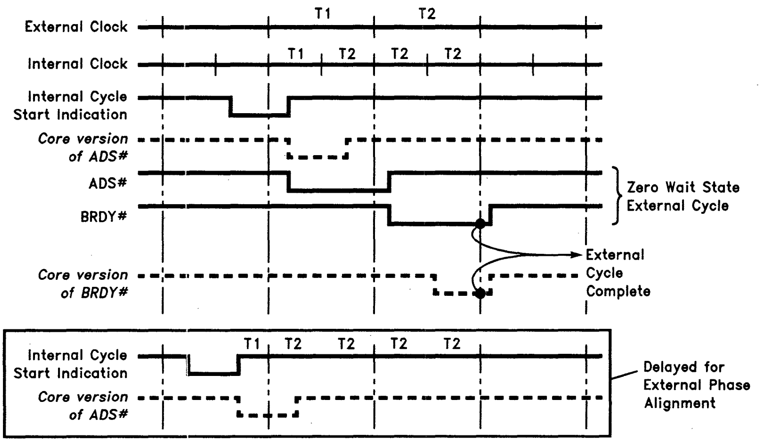
2.5.3.1 Delay Till First Ready of a Read

Wait states incurred on the first ready of a external read (the lead-off cycle) affect both data reads and code prefetches. For data reads, the CPU's execution is stalled under most conditions; no other operation can happen in parallel until the first ready of the line fill is received. For code prefetches, execution is stalled if the processor is fetching code as a result of a code branch (therefore flushing the prefetch buffers).

For most applications, the read lead-off delay increases the execution time the least compared to the other parameters. This is because writes cycles usually make up the dominant share of the bus cycles. However, there are exceptions to this case; for example, with the GCC trace, only 57.4% of the bus cycles were writes.

2.5.3.2 Wait states on Bursts

Adding wait states to the burst cycle increases the execution time the most. The burst is usually the result of a cache line fill or a code prefetch. Adding wait states to this parameter ties up the bus for the longest periods of time compared to adding the same number of wait states to the other parameters. As a result, all subsequent external bus requests are stalled as the CPU waits for the burst cycle to complete. These include stalls while the CPU waits to do a read cycle. A longer burst cycle also delays the rate at which the internal write buffers can be emptied since the write buffers must also wait for the external bus to free up. This causes stalls as described below for write cycles.



241261–15

Figure 2.15. A Zero-Wait State Write for the Intel486™ DX2 CPU

Finally, longer code prefetch bursts will slow down CPU execution if the prefetch was a result of the prefetch queue being flushed. This is especially so if the instruction required extends beyond the first dword of the burst and therefore the CPU must wait for subsequent dwords before execution can start.

2.5.3.3 Write Wait States

There are three conditions under which a longer write cycle will stall CPU execution as additional wait states are added. These conditions are:

1. The write buffers are full and cannot accept any more writes.
2. A read cannot bypass the write buffers and must wait for them to be flushed.
3. A read bypasses the write buffers but must wait for an existing write cycle to complete.

Before these effects are elaborated, it is worthwhile to reexamine the operation of the internal write buffers.

The Intel486 DX2 CPU uses the same four-deep write buffers as the Intel486 DX CPU. The write buffers can accept data writes from the execution core as fast as one per clock. Once a write request is buffered, the internal unit that generated the request is free to continue processing. When all write buffers are full, any subsequent write transfer will stall inside the processor until a write buffer becomes available.

The bus interface unit can re-order pending reads in front of buffered writes. This is done because pending reads can prevent an internal unit from continuing, whereas buffered writes need not have a detrimental effect on processing speed. Writes are propagated to the external bus in the same first-in-first-out order in which they are received from the internal unit. However, a subsequently generated read request (data or instruction) may be reordered in front of buffered writes. As a protection against reading invalid data (reading stale data from a location in main memory when the location has been modified in the write buffers), this reordering of reads will only occur if all buffered writes are internal cache hits. Because an external read will only be generated for a cache miss, and will only be reordered in from of buffered writes if all such writes are internal cache hits, any read generated on the external bus will never read a location that is about to be written by a buffered write.

This reordering can only happen once for a given set of buffered writes, because the data returned by the read cycle could otherwise replace data about to be written from the write buffers.

The first condition that causes CPU stalls is when the write buffer is full. Write wait states decrease the rate at which the write buffer can be emptied. Since the Intel486 DX2 runs at a 2X-internal frequency, the likelihood of filling up the write buffers increases when compared to the Intel486 DX CPU as shown in Figure 2.16.

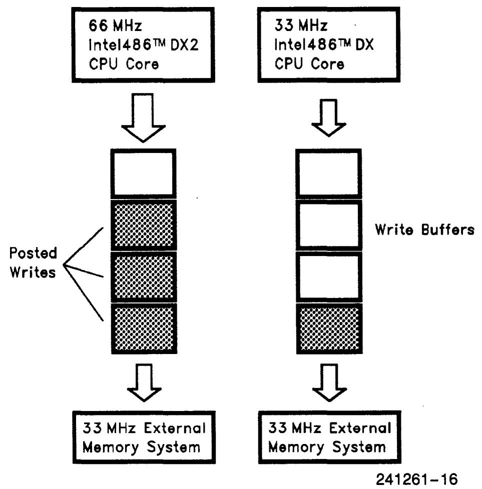


Figure 2.16. The Intel486 DX2 CPU's Write Buffers are More Heavily Used than the Intel486 DX CPU's

The second situation that degrades performance is during reads which cannot bypass the write buffers - either because the buffered writes were cache misses or because a read reordering had already occurred. These reads will be stalled until the write buffers are emptied. The more wait states required for writes on the external bus, the longer these stalls will last.

Finally, reads which can bypass the write buffers may be stalled by a write already in progress on the external bus. This condition is illustrated in Fig 2.17 for both the Intel486 DX and Intel486 DX2 CPUs. Note that for this example, although both the Intel486 DX2 and Intel486 DX CPUs take the same amount of time to complete the instruction stream, the Intel486 DX2 CPU is stalled longer (waiting for the write to complete) relative to its own internal 2X clock.

Out of the three conditions described above, stalls on write buffers full and stalls because of reads on busy writes dominate the increase in execution time as wait states are added to write cycles as shown in Table 2.7 for the SPEC1 trace. (The results shown in Figure 2.7 assume that the read and burst cycles complete in zero wait-states). These effects will be discussed again later when the addition of external write buffers is considered.

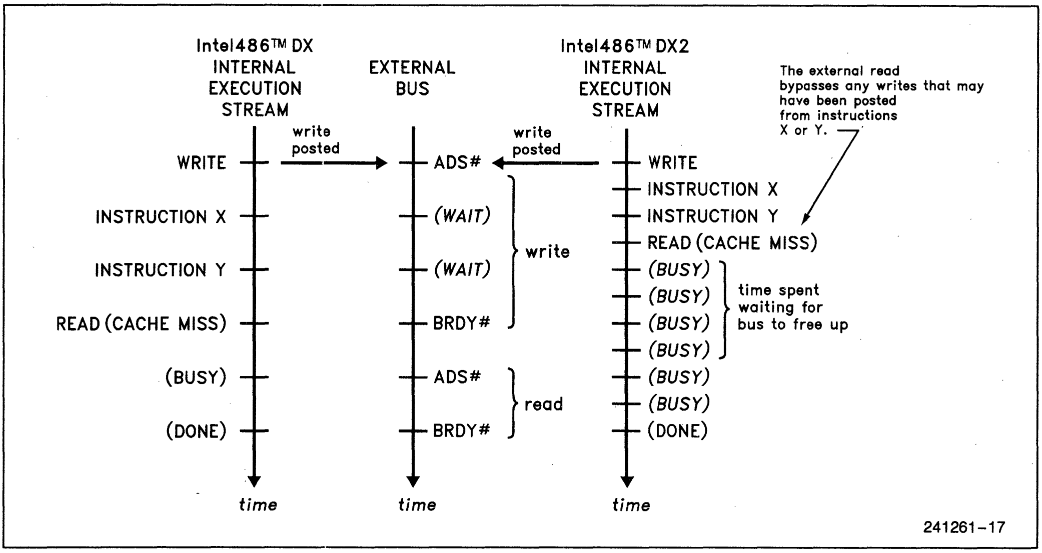


Figure 2.17. Reads Stalled as a Result of a Write Already In Progress

Table 2.7 Percentage of Total Execution Time Stalled under the Three Different Write Stall Conditions

	Intel486 DX CPU			Intel486 DX2 CPU		
Write wait states	**0**	**1**	**2**	**0**	**1**	**2**
Stalls on write buffers full	0.0%	0.1%	0.6%	1.0%	3.1%	6.5%
Reads cannot overtake writes	0.1%	0.1%	0.3%	0.3%	0.5%	0.5%
Stalls because of reads on busy writes	0.6%	1.3%	2.2%	2.4%	4.8%	7.5%

3.0 Memory Design Optimization

Some Intel486 DX2 CPU-based designs will include a memory system without an external cache. This section covers the design of such a cacheless memory system. Different memory architectures are discussed and the benefits of improving write performance through the addition of external write buffes will also be considered (see Figure 3.1).

Main memory performance will be important for external cache-based designs also, especially for applications with low external cache hit rates. It is recommended that the performance impacts of design choices in a cacheless memory design are understood even if you have already specified an external cache in your design.

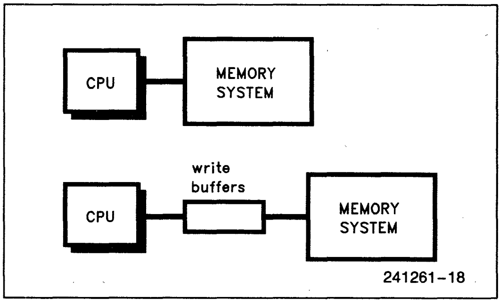


Figure 3.1 The Two Cacheless Intel486 DX2 CPU-Based Systems Considered

As discussed in the previous sections, the Intel486 DX2 Microprocessor requires a fast memory system for optimum performance. Memory systems that may have been adequate for Intel486 DX CPU designs running DOS applications may be suboptimal for the Intel486 DX2 CPU, especially running today's more demanding operating systems and applications.

Main memory page-mode operation and interleaving techniques are important for Intel486 DX2 CPU performance. These are commonly used in existing, well-designed Intel486 DX CPU memory systems. However, some systems still use non-interleaved memory designs borrowed from Intel386 DX systems. These will be less than optimal for a high performance Intel486 DX2 CPU workstations design.

3.1 Page Mode DRAM

Page-mode main memory controllers can be implemented in several fashions. Typical memory systems utilize paging for all accesses - during the beginning of a read, during read bursts and during write cycles; i.e. the RAS# line is held active after all accesses and only returned inactive during a page miss. Alternatively, paging may be used only for the burst portion of a read cycle; the RAS# line always returns inactive after the read or write cycle has been completed. This method is more commonly used in conjunction with a write-back external cache as discussed in Section 4.4.

For read burst cycles, the 16-byte linefill of data or code will always lie within a DRAM page, thereby allowing the data or code to be strobed out of memory with a series of back-to-back CAS# pulses. Paging allows for a much faster burst cycle compared to the case where a full RAS#-CAS# cycle is required for each dword. A page mode burst read access to a single bank of memory is shown in Fig. 3.2.

A paged memory system also allows for faster back-to-back write cycles. As was true for the Intel486 DX CPU, the Intel486 DX2 CPU generates writes in strings of two, about 60%-70% of the time, and writes in strings of three about 40%-50% of the time. This bus characteristic accounts for a large page hit rate for writes; therefore, it is faster to perform the back-to-back write cycles in a fast page mode rather than performing a full RAS#-CAS# cycle for each write.

At this point, it is worthwhile to examine the page hit/miss ratio for the three applications considered in the previous section. These results are shown in Table 3.1 and assume no external cache and a page size of 8192 bytes.

Table 3.1 Page Hit Ratios

	SPEC1	Pagemaker	Turbo C
Read Page Hit	31.2%	26.4%	25.4%
Read Page Miss	68.8%	73.2%	74.6%
Write Page Hit	65.5%	68.8%	68.9%
Write Page Miss	34.5%	31.2%	31.1%

Note the low page hit ratio for CPU reads. This is due to the internal cache of the Intel486 DX2 CPU that filters read requests and tends to make the cache miss reads more randomly distributed throughout main memory.

3.2 Interleaving

Interleaving involves the use of more than one bank of memory; different banks are controlled separately. As an access is occurring, the other banks are being readied for the next access. Interleaving can be implemented in several ways. Horizontally interleaved banks generate accesses for consecutive locations in memory. For example, one bank can be designated as an odd dword and another for the even dword. Vertically interleaved banks separate large contiguous regions of memory between banks; i.e. multiple DRAM pages are open. Memory controllers often combine both methods.

Horizontal interleaving can be combined with paging to generate very quick burst reads. Two 32-bit banks can generate a zero wait-state burst as detailed in the Intel Applications Note AP447 "A Memory Subsystem for the Intel486TM DX Family of Microprocessors including Second Level Cache." Fig. 3.3 illustrates a burst read cycle from a paged-interleaved memory system. The signals CAS0# and CAS1# drive each of the two 32-bit banks of memory in this example.

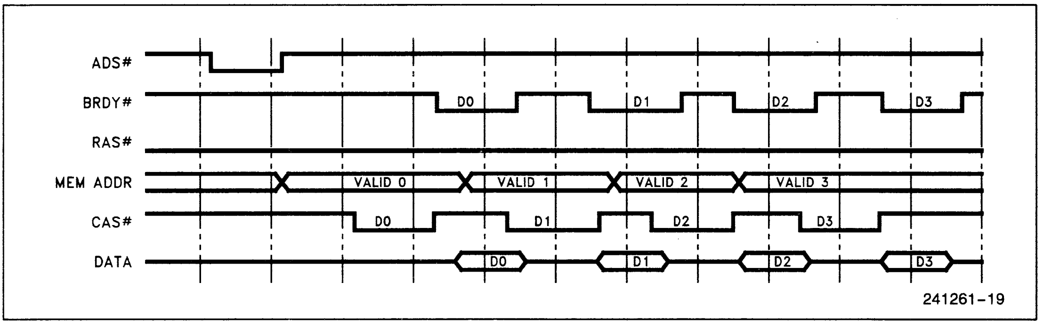


Figure 3.2. A Page Mode Burst Read (Page Hit on Lead-Off Cycle)

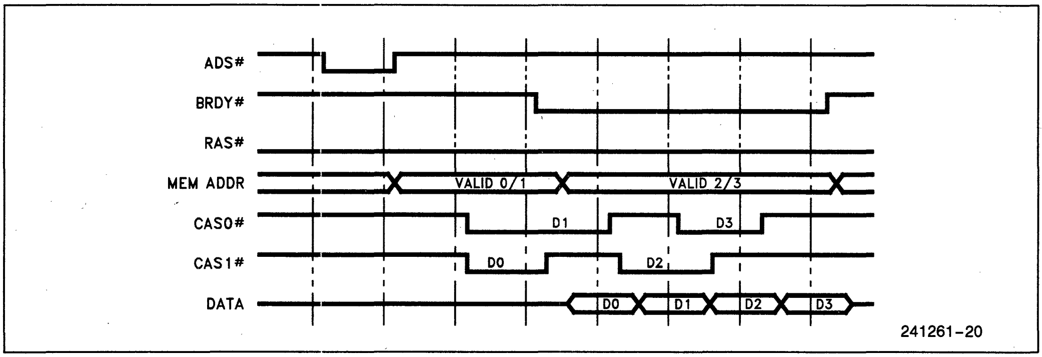


Figure 3.3. Burst Read Cycle from a Paged-Interleaved Memory System

Some implementations of a two-bank interleaved memory may be limited to a 1-2-1 burst cycle for the last three dwords. This is mainly limited by the amount of time it takes to invert the A3 address line between the second and third dwords.

3.3 Memory Read Performance Considerations

Seven memory systems with different read performance parameters are examined; write performance is kept constant during these simulations. The memory parameters are as follows:

Systems A through D represent the performance of some typical page mode memory controllers while the performance of systems E through G would require a paged-interleaved memory controller.

Table 3.2. Memory Systems used for the No-Cache System Test

	Read Page Hit	Read Page Miss	Write Page Hit	Write Page Miss
System A	4-3-3-3	8-3-3-3	3	6
System B	4-2-2-2	8-2-2-2	3	6
System C	4-2-2-2	7-2-2-2	3	6
System D	3-2-2-2	8-2-2-2	3	6
System E	3-1-2-1	7-1-2-1	3	6
System F	3-1-2-1	6-1-2-1	3	6
System G	3-1-1-1	6-1-1-1	3	6

The results for the Intel486 DX2 CPU with these memory systems is shown in Figure 3.4 through Figure 3.6 for the SPEC1, Pagemaker and Turbo C traces used previously. The graphs show the total execution time relative to the ideal zero-wait state memory system.

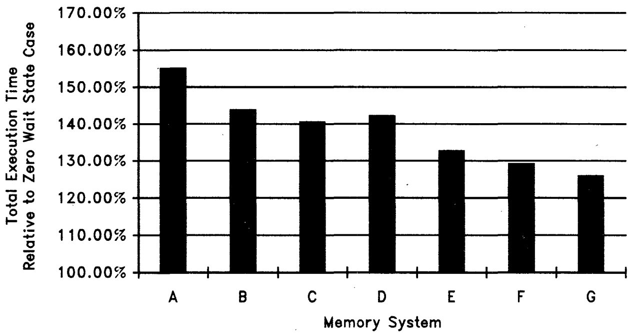
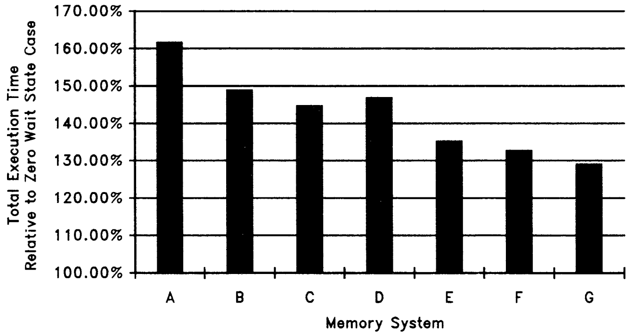
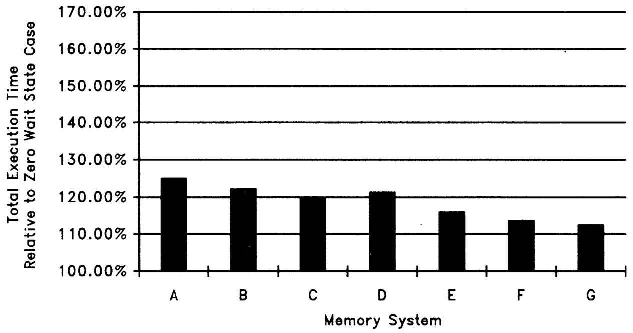


Figure 3.4. Total Intel486 DX2 CPU Execution Time versus Memory Read Performance - for SPEC1 (UNIX)



241261-22

Figure 3.5. Total Intel486 DX2 CPU Execution Time versus Memory Read Performance - for Pagemaker (Windows)



241261-23

Figure 3.6. Total Intel486 DX2 CPU Execution Time versus Memory Read Performance - for Turbo C (DOS)

The interesting points to note here are:

- As expected, the DOS application suffers the least from slow memory performance.
- Burst performance is very important. Note the improvement from system A to B, system D to E and even from system F to G (where one clock was removed from the third burst).
- Since the read page hit ratio is lower than 50%, improving the read page miss lead-off cycle is more important than the read page hit lead-off cycle. Note the improvement from system B to C versus the improvement from system B to D.

3.4 Memory Write Performance Considerations

There are several methods of improving the write performance of the memory system. These methods are first described; the benefits of the different methods are discussed later.

The most common method for improving write performance is to employ page mode accesses to DRAM. As shown in Table 3.1, the page hit ratios for write cycles favor the use of page mode accesses. An example of a DRAM write cycle is shown in Fig. 3.7. On page hits, back-to-back three clock write cycles can be maintained. A page miss write would of course take an additional number of clocks to allow for the RAS# precharge time.

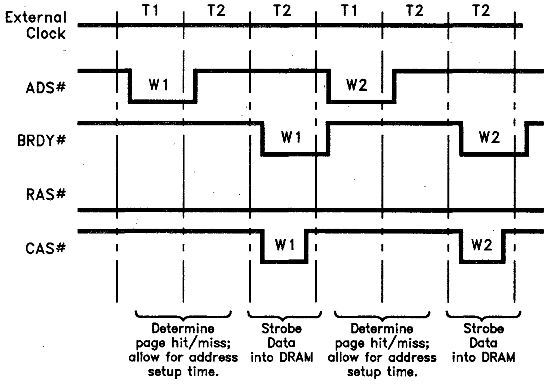

External Clock
T1
T2
T2
T1
T2
T2
ADS#
W1
W2
BRDY#
W1
W2
RAS#
CAS#
W1
W2
Determine page hit/miss; allow for address setup time.
Strobe Data into DRAM
Determine page hit/miss; allow for address setup time.
Strobe Data into DRAM
241261-24

Figure 3.7. Page Mode DRAM Allow for Fast Back-to-back Writes

Memory write performance can be improved further by two related methods: write buffering and pipelining. If one external write buffer is added to the memory system shown in Fig. 3.7, the write cycles in Fig. 3.8 may be observed:

In the example shown, the one level of write buffering allows the first ready signal to be returned one clock earlier. The first write cycle finishes in two clocks (zero wait states); however, if the CPU puts out many back-to-back writes (as is typical), the memory system will still be limited to a throughput of three clock writes subsequent to the first write cycle. If the CPU write is not followed immediately by any bus traffic, the one write buffer does relieve the CPU quickly, especially if the write was a page miss.

More than one level of write buffering is sometimes employed. This would allow multiple writes to be accepted at zero wait states before wait states of the main memory system affect the CPU bus. To get the maximum benefit from multiple write buffering, CPU reads that occur when there are more than one writes pending in the external write buffers should be allowed to bypass the writes and be executed as soon as the existing memory write is complete. This is similar or course to the internal write buffers of the Intel486 CPU. If the read cycle's address corresponds to an address already in the write buffers, the read must wait until the corresponding write completes so that the read does not fetch stale data from memory. In other words, care must be taken to ensure data consistency when using external write buffers.

Write pipelining extends the use of buffering by overlapping the memory controller operations during the write cycle. An example is shown in Fig. 3.9 below. The data phase of the last write cycle (CAS# pulse) is overlapped in time with the address phase of the next memory write cycle (Page hit/miss decoding, etc.).

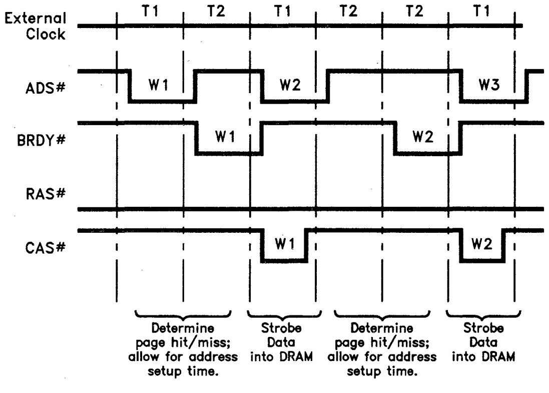
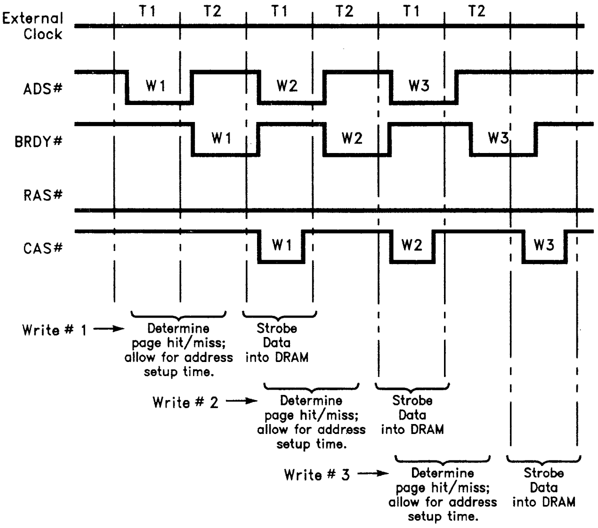

External Clock
T1
T2
T1
T2
T2
T1
ADS#
W1
W2
W3
BRDY#
W1
W2
RAS#
CAS#
W1
W2
Determine page hit/miss; allow for address setup time.
Strobe Data into DRAM
Determine page hit/miss; allow for address setup time.
Strobe Data into DRAM
241261-25

Figure 3.8. Adding One Write Buffer to the Memory System



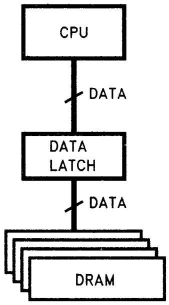
External Clock
T1
T2
T1
T2
T1
T2
ADS#
W1
W2
W3
BRDY#
W1
W2
W3
RAS#
CAS#
W1
W2
W3
Write # 1
Determine page hit/miss; allow for address setup time.
Strobe Data into DRAM
Write # 2
Determine page hit/miss; allow for address setup time.
Strobe Data into DRAM
Write # 3
Determine page hit/miss; allow for address setup time.
Strobe Data into DRAM

241261-26

Figure 3.9. Pipelining the Writes to Memory

With pipelining, it is possible to achieve a throughput of many two-clock back-to-back page hit writes. (An example of write pipelining can be found in Intel Applications Note AP447) Note that pipelining may affect a subsequent read cycle; if the CPU read occurs immediately after the write, the beginning of the read will be delayed until the DRAM write cycle has completed. This is especially true for page miss writes where the write may take several clocks to complete. Note also that pipelined memory write systems can be combined with write buffering; this helps for the case where many back-to-back page miss writes occur.

Note that both write buffering and/or pipelining require a data path device between the CPU and main memory (see Fig. 3.10). This is needed to capture the data from the CPU before RDY# or BRDY# is returned, after which point the data will become invalid.



241261-27

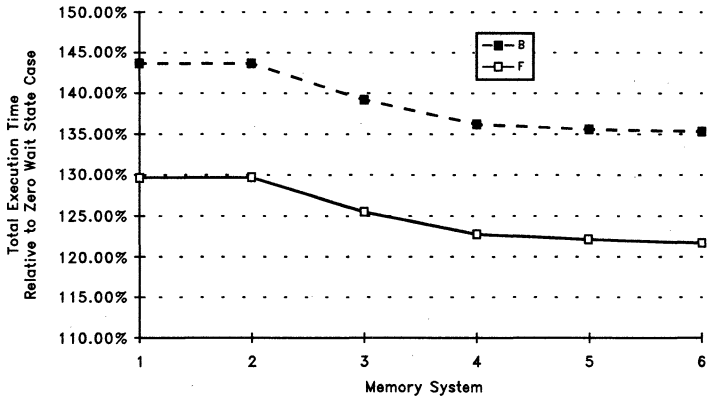
Figure 3.10. A Data Latch is Required for Write Buffering or Pipelining

To understand the performance benefits of the various methods described, systems B and F from the earlier simulations for read performance are repeated with different write performance parameters (refer to Table 3.3). Systems B and F were chosen as typical representations of paged and paged-interleaved memory controllers respectively.

Table 3.3. Memory Systems with Different Write Performances

	Read Pg Hit	Read Pg Miss	Write Pg Hit	Write Pg Miss	Write Method
System B1	4-2-2-2	8-2-2-2	3	6	Normal
System B2	4-2-2-2	8-2-2-2	3	6	One buffer
System B3	4-2-2-2	8-2-2-2	2	6	Pipelined
System B4	4-2-2-2	8-2-2-2	2	5	Pipelined
System B5	4-2-2-2	8-2-2-2	2	5	Pipelined with two buffers
System B6	4-2-2-2	8-2-2-2	2	5	Pipelined with four buffers
System F1	3-1-2-1	6-1-2-1	3	6	Normal
System F2	3-1-2-1	6-1-2-1	3	6	One buffer
System F3	3-1-2-1	6-1-2-1	2	6	Pipelined
System F4	3-1-2-1	6-1-2-1	2	5	Pipelined
System F5	3-1-2-1	6-1-2-1	2	5	Pipelined with two buffers
System F6	3-1-2-1	6-1-2-1	2	5	Pipelined with four buffers

The memory systems above were simulated again for the Intel486 DX2 CPU with the three applications. The results are shown in Fig. 3.11 through Fig. 3.13 below:



241261-28

Figure 3.11. Total Execution Time versus Write Performance- for SPEC1 (UNIX)

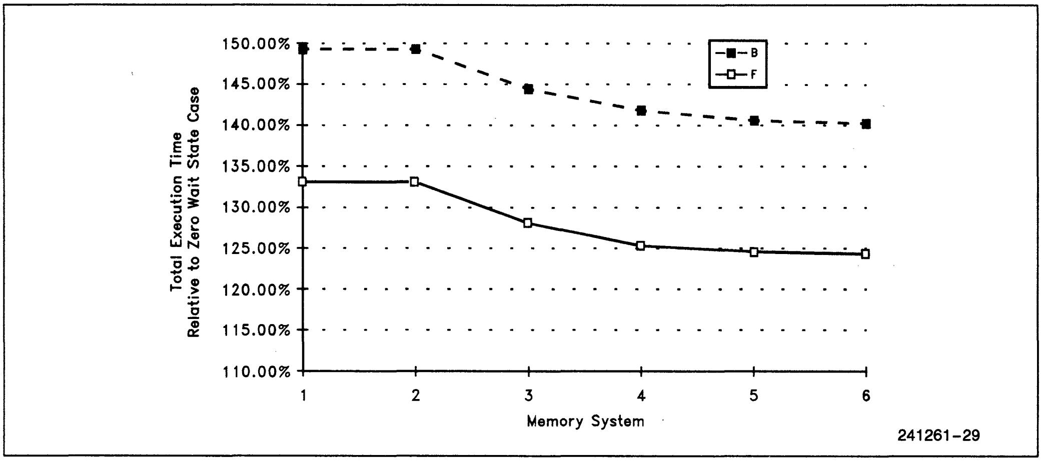


Figure 3.12. Total Execution Time versus Write Performance - for Pagemaker (Windows)

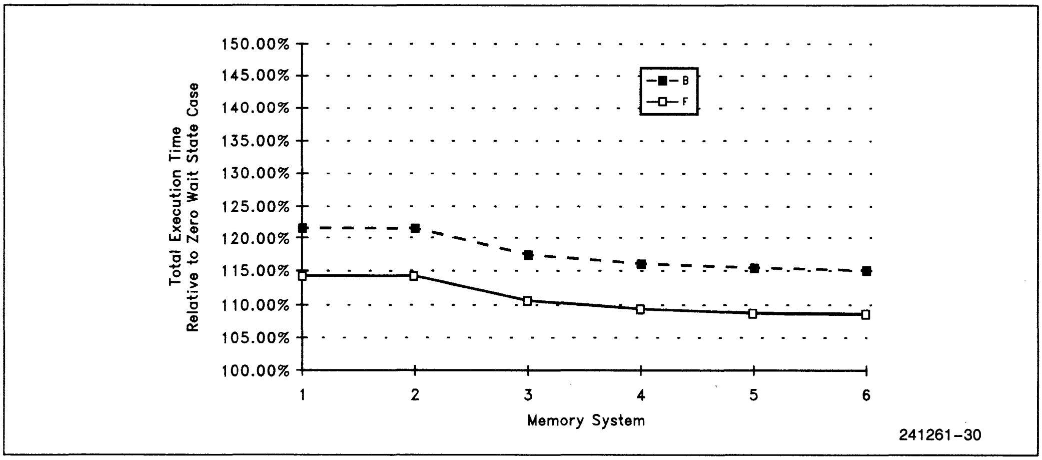


Figure 3.13. Total Execution Time versus Write Performance - for Turbo C (DOS)

From the results shown, the following significantly improved CPU performance:

- Reducing the number of clocks for page hit writes (system B2 to B3 and F2 to F3)
- Reducing the number of clocks for page miss writes (system B3 to B4 and F3 to F4)

The following caused marginal improvement in the execution time:

- Adding one level of buffering (from memory systems B1 to B2 and F1 to F2)
- Adding more than two write buffers (from B4 to B5 to B6 and F4 to F5 to F6) resulting in only a 1-2% improvement.

These results may be somewhat surprising considering the earlier graph showing performance degradation as the number of write wait states increases (Fig. 3.7). One would expect that adding write buffers would compensate for the slower memory write system more than they do. In order to understand the results, consider the statistics in Table 3.4 for processor execution stalls as a result of write activity as described in Section 2.5.3. The statistics are shown for the SPEC1 trace.

Note that the Stalls Because of Reads On Busy Writes dominates the increase in execution time for memory system F1 compared to the zero wait state case. Adding one write buffer (from system F1 to F2) - in an attempt to improve performance - decreases the percentage of stalls on a full write buffer from 5.3% to 4.5%. However, while the number of Stalls Because of Reads on Busy Writes did decrease, the wait states were simply transferred to stalls while waiting for the first ready of a read and no net imp rovement is observed. One example of this situation is illustrated in Fig. 3.14.

Table 3.4. Stall Statistics for the Write Buffers for the SPEC1 (UNIX) Trace

	Percentage of Total Execution Time Stalled:			
	On Write Buffers Full	Because of reads on Busy Writes	Because a Read Cannot Overtake a Write	Waiting for First Ready
Zero Wait State Case	1.0%	2.4%	0.3%	8.8%
Memory System F1	5.3%	8.0%	0.5%	17.1%
System F1 plus one write buffer	4.5%	4.9%	0.4%	20.9%

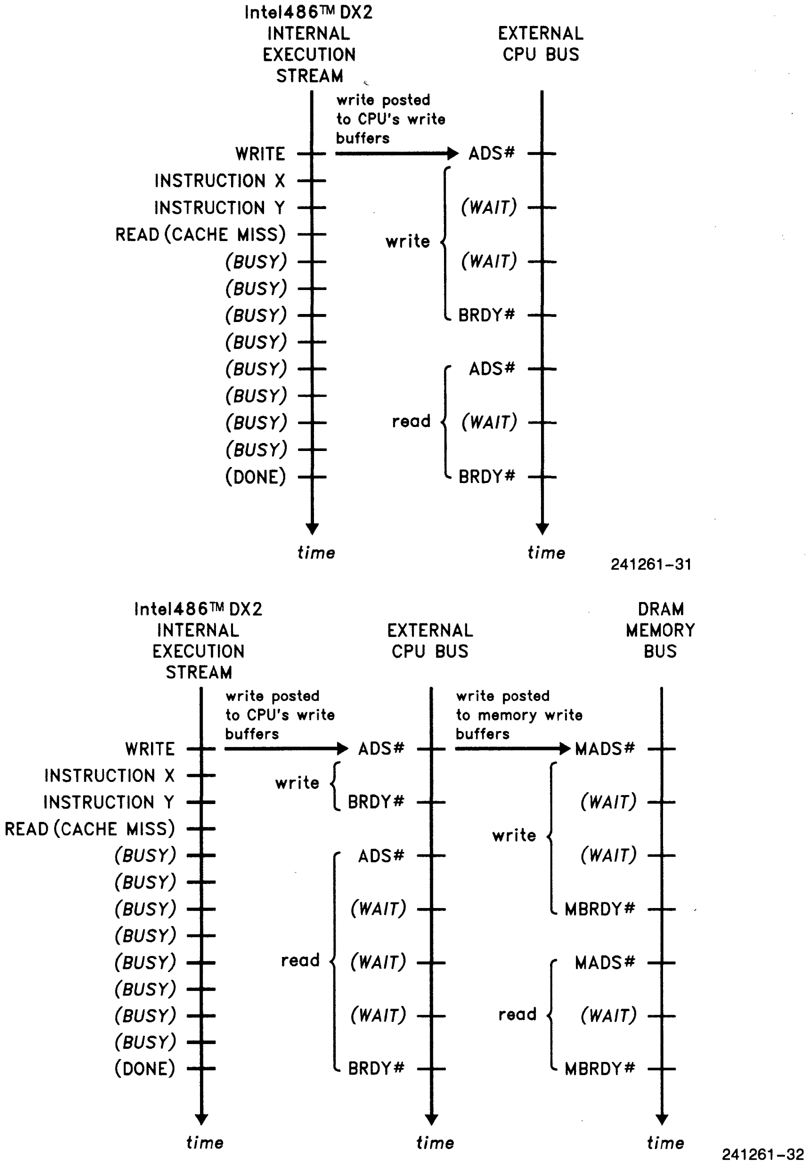


Figure 3.14. Adding Write Buffers Does Not Improve Stalls Because of Reads on Busy Writes

In the example shown, without external memory write buffers, the cache miss read shown stalls for four clock cycles while waiting for the write cycle to complete. With memory write buffers, the CPU need not wait to start the read cycle since the write completed in zero wait states. However, since main memory is still occupied with the original write cycle, the read is still delayed externally while the write completes. The net effect is that the read-stalls because of write traffic does not decrease; th e write traffic has simply been transferred from the CPU bus to the memory bus where it has to contend with the next read cycle.

Note that there will be instances where the addition of external write buffers to a cacheless memory system does benefit a sequence of bus cycles. This would be the case for applications with very low external read traffic and large amounts of write traffic. In this case, the write buffers do benefit the heavy write traffic while the reads on busy writes will be a lower percentage of total stalls.

3.5 Viability of Intel486 DX2 System without an External Cache

As shown in this section, the CPU performance of a cacheless, main-memory-only Intel486 DX2 CPU based system will range from good to fair depending on the application. The correct cost-performance point will dictate the viability of such a product. For the Windows and UNIX applications tested, with a good memory design, the Intel486 DX2 CPU will get to about 120% of the execution time of the ideal zero wait state case with the examples shown. The reciprocal of total execution time is CPU performance; wh ich works out to 83% of maximum in this case. Of course, other system design factors will come into play, such as refresh requirements, other bus master memory requirements, etc.

More exotic memory architectures may improve the performance of the cacheless Intel486 DX2 CPU system design over what has been discussed here. However, the next section will address the more straightforward method of increasing CPU performance further: adding an external cache.

4.0 CACHE DESIGN OPTIMIZATION

An external cache will supplement the on-chip 8K cache of the Intel486 DX2 CPU. The requirement for an external cache is more important for the Intel486 DX2 CPU than it was for the Intel486 DX CPU for all the reasons discussed in the previous sections.

Many cache architectures have been implemented with the Intel486 DX CPU. Caches differ depending on their size, associativity, serial vs. parallel implementations, write-through vs. write-back policies, etc. This section will focus on optimizing the performance of a uniprocessing system, i.e. the CPU is the major consumer of main memory bandwidth. The architectures discussed are shown in Figure 4.1.

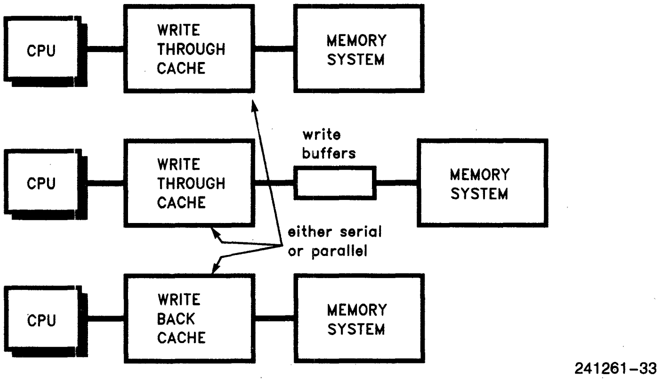
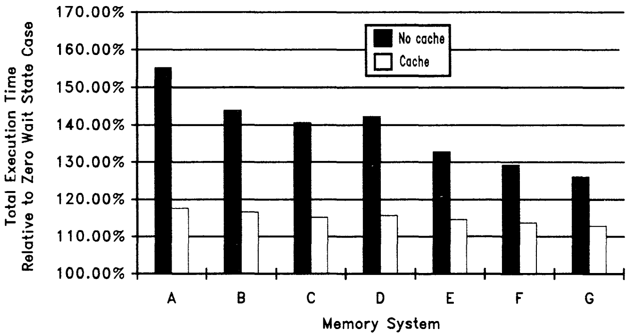


Figure 4.1. Different Cache Architectures Discussed

4.1 Overall Effect of an External Cache on CPU Performance

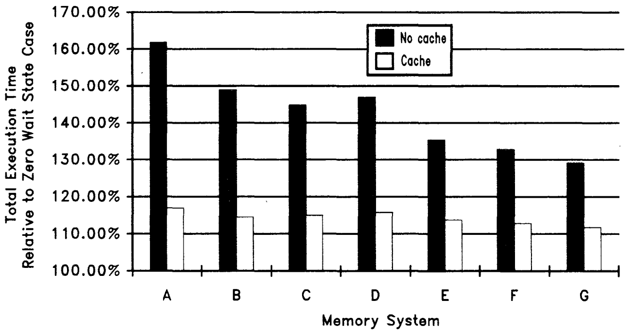
The same memory systems tested in the previous section are tested again with a 128K 2-way associative write-through parallel cache. This will yield the improvement achieved by the decrease in the effective number of read and burst wait states. The results are shown in Figures 4.2 through 4.4 for the three applications tested earlier.

The addition of an external write-through cache reduces the performance degradation caused by main-memory wait states for the lead-off cycle of a read and for wait states during the remainder of a burst. The impact of these wait states was discussed in Section 2.5.3.



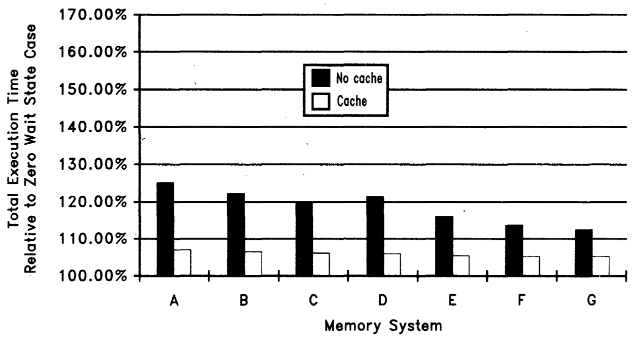
241261-34

Figure 4.2. Adding an External Cache Decreases Execution Time - for SPEC1 (UNIX)



241261-35

Figure 4.3. Adding an External Cache Decreases Execution Time - for Pagemaker (Windows)



241261-36

Figure 4.4. Adding an External Cache Decreases Execution Time - for Turbo C (DOS)

For the UNIX and Windows applications, the addition of the external cache improved the execution time by 15%-35% depending on the memory design. The cache used in this case - a 128K two-way set associative cache - does an excellent job of buffering the CPU performance from the memory system performance. However, note that even with the external cache, the execution time is still 12%-18% above the zero wait state case for these two applications. This is due to the write performance of the memory system s ince the cache policy in this example is write-through. Further improvement on the write performance is investigated later in this section.

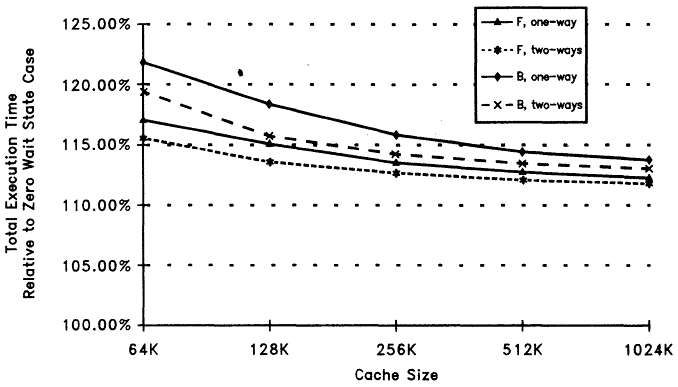
For the DOS application, the addition of the cache brings the performance of the Intel486 DX2 within 5% of the ideal zero wait state case. This is of course due to the lower miss rate of the CPU's internal cache and the application's low bus utilization.

4.2 Effect of Cache Size and Associativity

A 128K two-way-associative, write-through, parallel, external cache was used in the previous section. As the size and associativity of the cache are varied, the CPU performance varies. This is shown in Fig 4.5 for the memory systems B and F as used earlier (see Table 4.1). Both one-way (direct mapped) and two-way set associative caches are tested with the SPEC1 application trace.

Table 4.1. Memory Systems for the Write-Through Cache Test

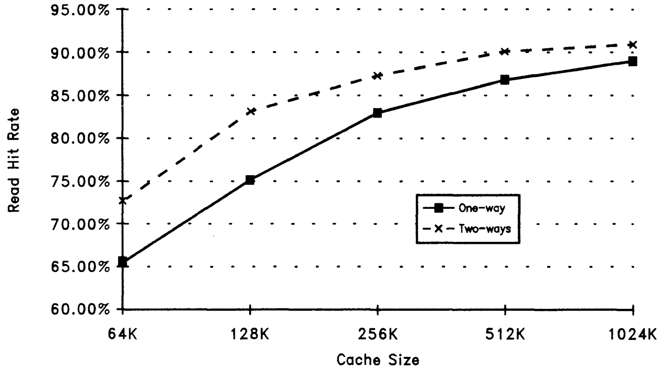
	Read Page Hit	Read Page Miss	Write Page Hit	Write Page Miss
System B	4-2-2-2	8-2-2-2	3	6
System F	3-1-2-1	6-1-2-1	3	6



241261-37

Figure 4.5. Total Execution Time for the Intel486 DX2 CPU as a Function of Cache Size and Associativity - for SPEC1

Fig 4.6 illustrates the external cache hit rates for CPU read cycles. The hit rates are directly related to the total execution time; higher hit rates result in shorter execution times.



241261-38

Figure 4.6. L2 Read Hit Rate as a Function of Cache Size and Associativity (UNIX)

4.3 Improving the Performance of a Write-Through Cache

With a write-through cache, good memory write performance is necessary to achieve the best possible performance with the Intel486 DX2 microprocessor. All of the methods for improving the write performance for a cacheless system, discussed in section 3.4, also apply for the write-through cache-based system.

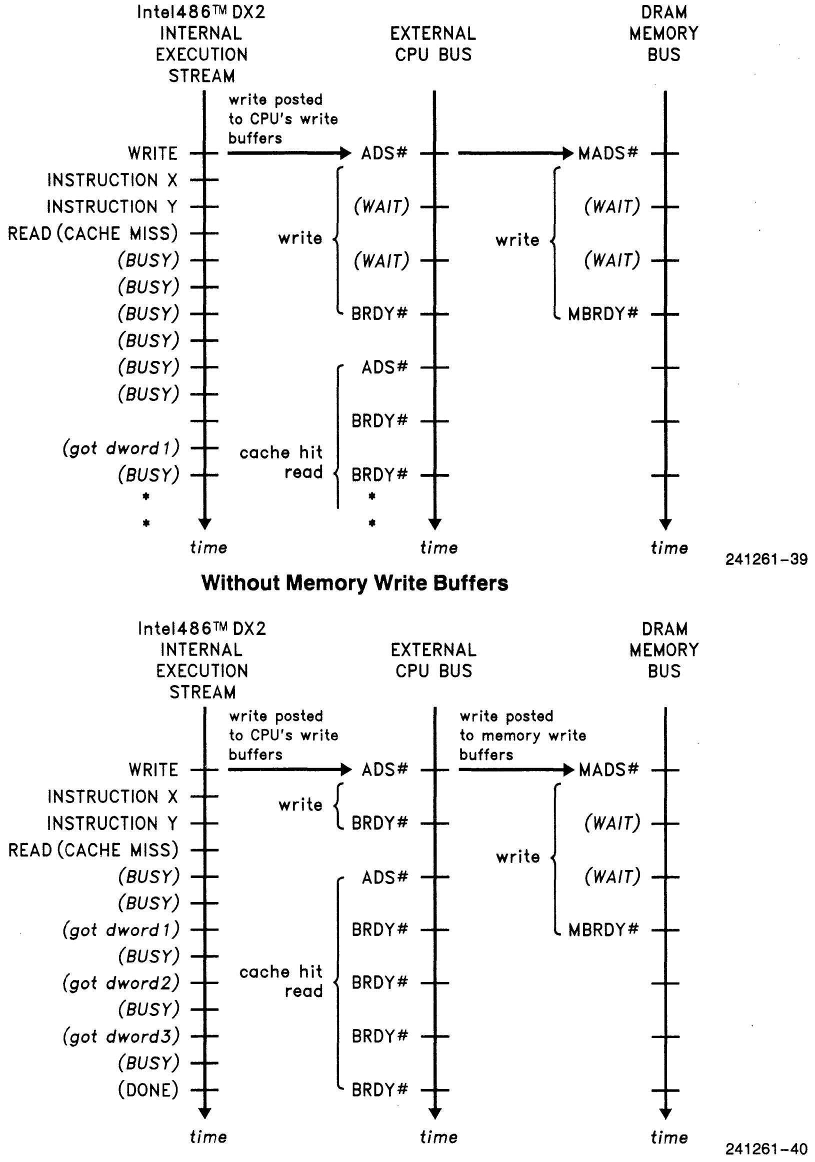
4.3.1 MEMORY WRITE PIPELINING

The previous results for a write-through cache assumed a non-pipelined memory system with a page-hit write performance of three clocks and a page-miss performance of six. The most effective method of increasing the memory write performance further is the use of memory write pipelining. The write performance can be improved so that continuous back-to-back page-hit write cycles can complete in zero wait-states. Pipelining can also reduce the number clocks required for a page-miss write cycle. As the write performance improves using this technique, the write-through cache system can come close to that of the ideal zero wait state system. These results are shown in Section 4.3.3 to follow.

4.3.2 EXTERNAL WRITE BUFFERS

Adding one or more write buffers to a external write-through, cache-based system improves performance by a larger amount compared to the cacheless case. Figure 4.7 illustrates why.

The addition of external write buffers allows the memory write cycle to be "hidden" from the CPU bus if the next CPU cycle happens to be a external cache hit read. And since the external cache read hit ratio is high, most of the delays which were present in a cacheless system under these circumstances are removed. In essence, the on-chip cache/write-buffers have been duplicated externally to provide a multiple level architecture (see Fig. 4.8).



Without Memory Write Buffers

With Memory Write Buffers

Figure 4.7. Adding External Write Buffers to an External Cache Reduces Execution Time

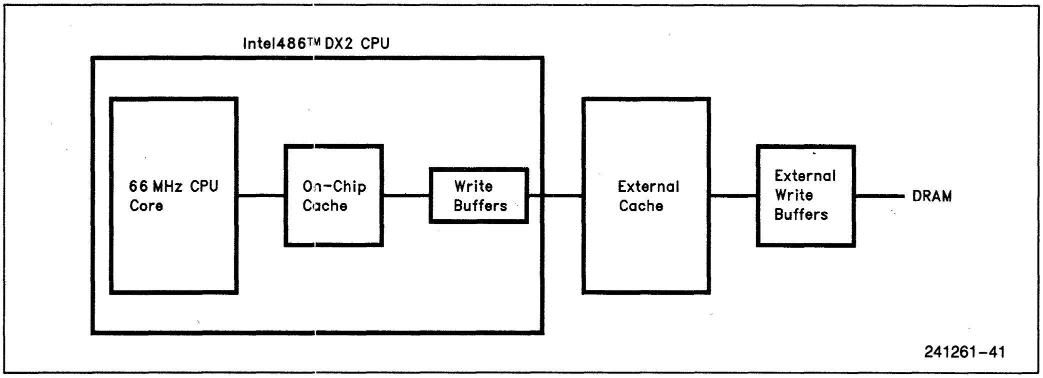


Figure 4.8. A Hierarchy of Caches and Write Buffers

4.3.3 PERFORMANCE WITH AN EXTERNAL WRITE-THROUGH CACHE

To quantify the benefits of improving the write performance, the systems in Table 4.2 were tested.

Table 4.2. Memory Systems Used for Write-Through Cache Test

	Read Pg Hit	Read Pg Miss	Write Pg Hit	Write Pg Miss	Write Method
System B1	4-2-2-2	8-2-2-2	3	6	Normal
System B2	4-2-2-2	8-2-2-2	3	6	One buffer
System B3	4-2-2-2	8-2-2-2	2	6	Pipelined
System B4	4-2-2-2	8-2-2-2	2	5	Pipelined
System F1	3-1-2-1	6-1-2-1	3	6	Normal
System F2	3-1-2-1	6-1-2-1	3	6	One buffer
System F3	3-1-2-1	6-1-2-1	2	6	Pipelined
System F4	3-1-2-1	6-1-2-1	2	5	Pipelined

Fig. 4.9 shows the results of the memory systems tested with a 128K, two-way associative, write-through, parallel cache and the Intel486 DX2 CPU using the SPEC1 application trace.

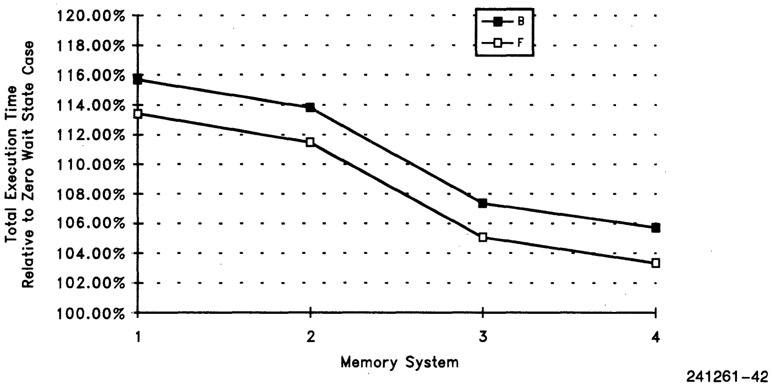


Figure 4.9. Improving the Write Performance Benefits a Write Through Cache - for SPEC1

As the write performance increases, the CPU performance approaches that of the zero wait state case. The improvement from systems B1 to B2 and from F1 to F2 illustrate the benefit of write buffering with an external cache. The improvement from systems B2 to B3 and from F2 to F3 reflect the benefit of memory write pipelining. Finally, the improvement from systems B3 to B4 and from F3 to F4 show how reducing the page-miss write performance also increases performance.

4.4 Write-Back Caches

If correctly implemented, a write-back external cache can provide good performance for a uniprocessing Intel486 DX2 CPU based system. Serial write-back caches have typically been used to reduce bus utilization for multiprocessing systems. The design complexity of a write-back cache controller is typically an order of magnitude higher than for a write-through cache controller. However, correct implementation is absolutely necessary if significant performance gains are to be realized with the Intel486 DX2 CPU.

A write-back cache is different from a write-through cache in that it allows cache write hits to modify the cache line without updating main memory. The cache has tags that include a bit called the modified (dirty) bit. This bit is set if the cache location has been written with new information and therefore contains information that is more recent than the corresponding information in main memory. If a subsequent read miss occurs and the line being fetched needs to fill the cache location that is currently being occupied by the modified line, the cache controller must then write the modified cache line back to main memory; hence coherency is maintained.

If a CPU write is not a cache hit, the cache controller has the option of allowing the write to propagate through to memory or to fetch the cache line from memory to be merged with the new write data. The cache line fill in the second option is called a write-allocation. In the following discussions, it is assumed that no write-allocations are being performed.

4.4.1 MAIN MEMORY CONTROLLER CONSIDERATIONS

The addition of an external write-back cache changes the characteristics of the main memory bus traffic. Since the cache effectively filters all CPU requests, the cycles that do propagate to main memory tend to be more distributed in their locations. This decrease in temporal and spatial locality will reduce the DRAM page hit rate as shown in Table 4.3 for a 128K, two-way associative, write-back cache with the SPEC1 application trace. Compare these results to the prior results in Table 2.1 for a cacheless system.

Table 4.3. Page Hit Ratios for a Write-Back Cache - for SPEC1

MEMORY CYCLES (100%)		SPEC1	PGMK	TURBOC
Reads:	Page Hits	17.1%	25.6%	24.7%
	Page Misses	13.8%	13.9%	12.9%
Writes:	Page Hits	58.9%	55.8%	40.8%
	Page Misses	10.2%	4.7%	21.6%

Therefore, it is less beneficial with a write-back cache to implement a page-mode main memory controller.

Of course, page mode DRAM accesses within the burst cycle are still important to retrieve the four words of a cache line quickly. This is also true for the write-back cycle where four dwords of the cache line must be written to memory. Memory controllers should be designed to support a burst write cycle instead of having to write each dword separately.

4.4.2 WRITE-BACK CYCLE

The write-back cycle is the sequence where a cache line fill from main memory has to displace a modified line that was already in the cache. The method in which the modified line is written back to main memory has an impact on overall CPU performance. Before analyzing the write-back cycle, consider first the architectures shown in Fig. 4.10.

In the simplest implementation, a write-back cache will share the data bus with the CPU and main memory as shown in configuration X. If this is the case, then during a write-back cycle, the modified line must be written back to main memory before the cache linefill can commence. This has a detrimental effect on performance since the CPU must wait while the write-back occurs. This sequence is shown in Figure 4.11.

With a data path device between the CPU-Cache bus and main memory as shown in configuration Y, the cache controller is able to defer the write-back of the modified data till after the linefill has completed. The CPU can continue execution after the linefill as long as subsequent cycles are all cache hits.

In configuration Z, a wider cache bus exists between the SRAM and the data path devices. This allows the modified data to be transferred more quickly from the SRAM to the data path device, thereby allowing the cache linefill to commence even sooner.

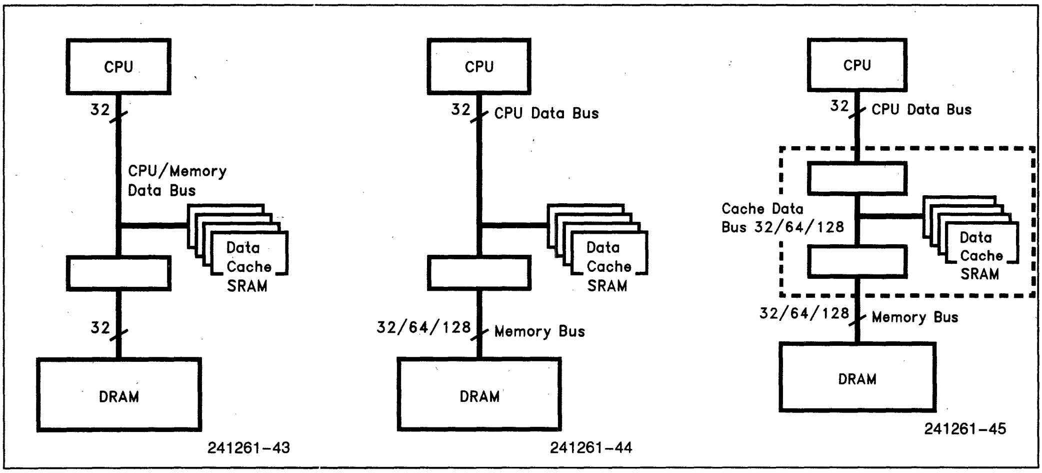


Figure 4.10. Different Architectures will Effect CPU Performance with a Write-Back Cache

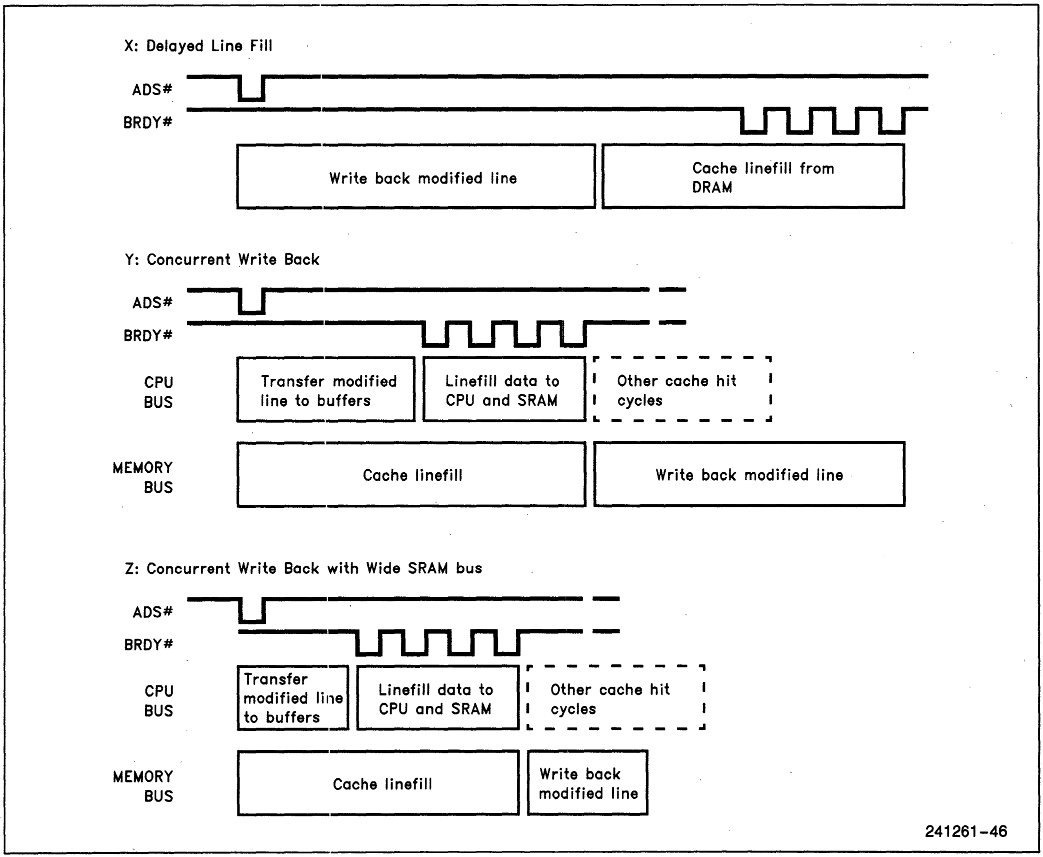


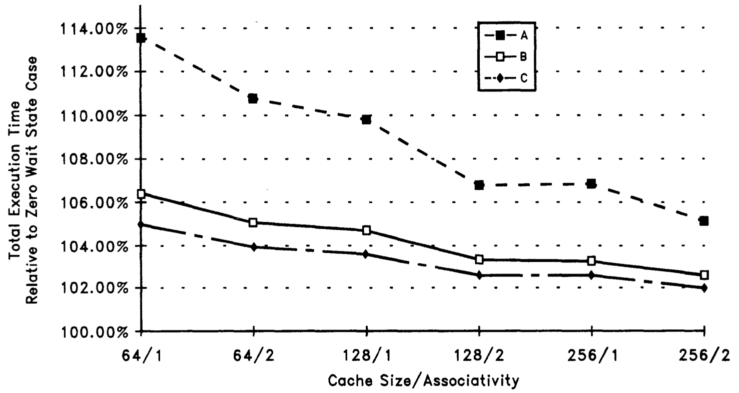
Figure 4.11. Different Implementations of the Write-Back Cycle

The following systems are used to demonstrate Intel486 DX2 microprocessor performance with different cache sizes and associativities.

The results are shown in Fig. 4.12 for the Intel486 DX2 CPU running the SPEC1 trace.

Table 4.4. Memory Systems used for Write-Back Cache Test

	Reads	Writes	Write-Back Method (described above)
System A	5-1-1-1	4-1-1-1 (burst)	Concurrent Write Back
System B	6-3-3-3	4-4-4-4 (non-burst)	Concurrent Write Back
System C	6-3-3-3	4-4-4-4 (non-burst)	Delayed Line Fill



241261-47

Figure 4.12. Intel486™ DX2 CPU Total Execution Time with Different Cache Size, Associativity, Memory Speed and Write-Back Method - for SPEC1

The addition of a write-back cache does an excellent job of decoupling the CPU performance from the main memory performance as shown with memory systems A and B. However, note that memory system B (with the delayed line fill) performs poorly - even worse than a good write-through cache - unless a significant amount of cache memory is added to reduce the miss rate.

5.0 CONCLUSION

This document has shown that good memory performance is especially important for the Intel486 DX2 microprocessor. Business workstation designs will require excellent CPU performance and will consequently have to incorporate well-designed, high-performance cache and memory systems.

In optimizing memory performance, an external cache is essential for hiding slow main memory access times. Write-through external caches offer good performance if coupled with good memory write performance. Write-back external caches can also offer excellent performance if designed correctly. Parallel write-back caches that cannot defer the write-back cycle till after a cache line fill will perform worse than a good write-through cache design.

intel®

AP-485

APPLICATION NOTE

Intel Processor Identification with the CPUID Instruction

December 1994

Order Number: 241618-003

INTEL PROCESSOR IDENTIFICATION WITH THE CPUID INSTRUCTION

CONTENTS

PAGE

CONTENTS

PAGE

Examples

Figures

Tables

1.0 INTRODUCTION

As the Intel Architecture evolves, with the addition of new generations and models of processors (8086, 8088, Intel 286, Intel386™, Intel486™, and Pentium™ processors), it is essential that Intel provides an increasingly sophisticated means with which software can identify the features available on each processor. This identification mechanism has evolved in conjunction with the Intel Architecture as follows:

- Originally, Intel published code sequences that could detect minor implementation differences to identify processor generations.
- Later, with the advent of the Intel386 processor, Intel implemented processor signature identification, which provided the processor family, model, and stepping numbers to software at reset.
- As the Intel Architecture evolved, Intel extended the processor signature identification into the CPUID instruction. The CPUID instruction not only provides the processor signature, but also provides information about the features supported by and implemented on the Intel processor.

The evolution of processor identification was necessary because, as the Intel Architecture proliferates, the computing market must be able to tune processor functionality across processor generations and models that have differing sets of features. Anticipating that this trend will continue with future processor generations, the Intel Architecture implementation of the CPUID instruction is extensible.

This Application Note explains how to use the CPUID instruction in software applications, BIOS implementations, and tools. By taking advantage of the CPUID instruction, software developers can create software applications and tools that can execute compatibly across the widest range of Intel processor generations and models, past, present, and future.

1.1 Update Support

New Intel processor signature and feature bits information can be obtained from the user's manual, programmer's reference manual or appropriate documentation for a processor. In addition, Intel can provide you with updated versions of the programming examples included in this application note; contact your Intel representative for more information.

2.0 DETECTING THE CPUID INSTRUCTION

Intel has provided a straightforward method for detecting whether the CPUID instruction is available. This method uses the ID flag in bit 21 of the EFLAGS register. If software can change the value of this flag, the CPUID instruction is available. The program examples at the end of this Application Note show how to use the PUSHFD instruction to change the value of the ID flag.

3.0 OUTPUTS OF THE CPUID INSTRUCTION

Figure 1 summarizes the outputs of the CPUID instruction.

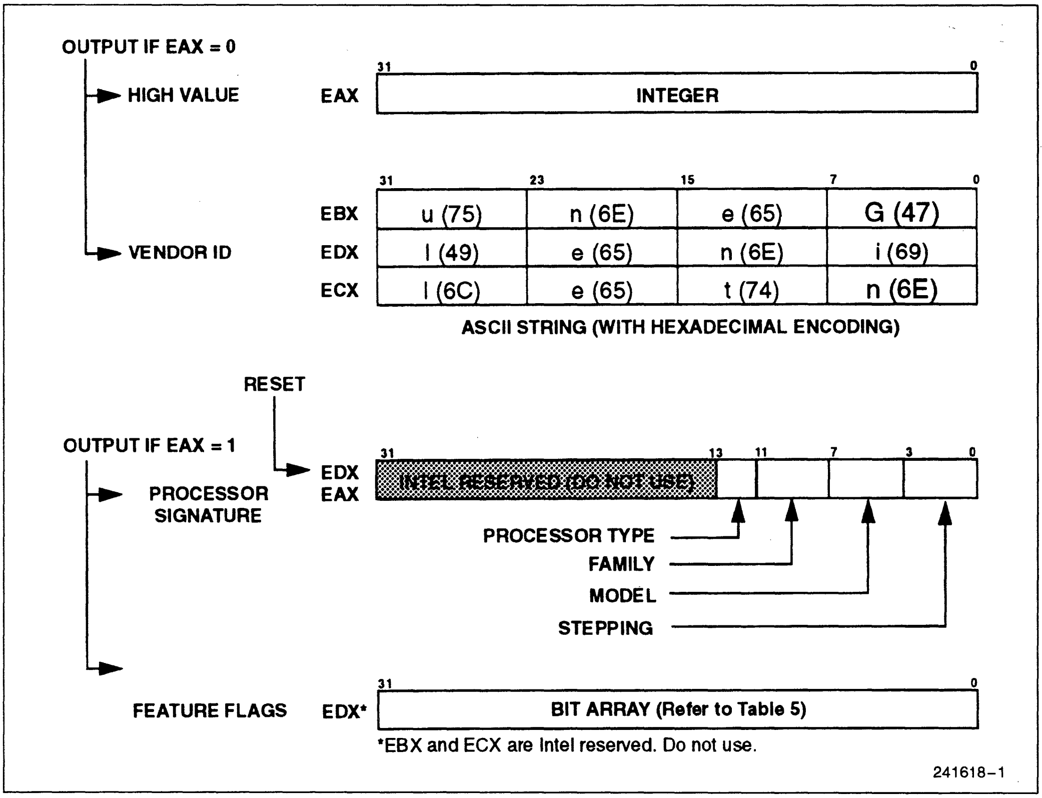
The CPUID instruction can be executed multiple times, each time with a different parameter value in the EAX register. The output depends on the value in the EAX register, as specified in Table 1. To determine the highest acceptable value in the EAX register, the program should set the EAX register parameter value to 0. In this case, the CPUID instruction returns the highest value that can be recognized in the EAX register. CPUID instruction execution should always use a parameter value that is less than or equal to this highest returned value. Currently, the highest value recognized by the CPUID instruction is 1. Future processors might recognize higher values.

The processor type, specified in bits 12 and 13, indicate whether the processor is an original OEM processor, an OverDrive™ processor, or is a dual processor (capable of being used in a dual processor system). Table 2 shows the processor type values that can be returned in bits 12 and 13 of the EAX register.

While any imitator of the Intel Architecture can provide the CPUID instruction, no imitator can legitimately claim that its part is a genuine Intel part. Therefore, the presence of the GenuineIntel string is an assurance that the CPUID instruction and the processor signature are implemented as described in this document.

3.1 Vendor-ID String

If the EAX register contains a value of 0, the vendor identification string is returned in the EBX, EDX, and ECX registers. These registers contain the ASCII string GenuineIntel.



	31	23	15	7
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)

Figure 1. CPUID Instruction Outputs

Table 1. Effects of EAX Contents on CPUID Instruction Output

Parameter	Outputs of CPUID
EAX = 0	EAX ← Highest value recognized
	EBX:EDX:ECX ← Vendor identification string
EAX = 1	EAX ← Processor signature
	EDX ← Feature flags
	EBX:ECX ← Intel reserved (Do not use.)
1 < EAX ≤ highest value	Currently undefined
EAX > highest value	EAX:EBX:ECX:EDX ← Undefined (Do not use.)

Table 2. Processor Type

Bit Position	Value	Description
13,12	00	Original OEM Processor
	01	OverDrive™ Processor
	10	Dual Processor[1]
	11	Intel reserved (Do not use.)

NOTE:

[1] Not applicable to Intel 386 and Intel486 processors.

3.2 Processor Signature

Beginning with the Intel386 processor family, the processor signature has been available at reset. With processors that implement the CPUID instruction, the processor signature is available both upon reset and upon execution of the CPUID instruction. Figure 1 shows the format of the signature for the Intel486 and Pentium processor families. Table 3 shows the values that are currently defined. (The high-order 18 bits are undefined and reserved.)

Older versions of Intel486 SX, Intel486 DX and IntelDX2 processors do not support the CPUID instruction. Therefore, the processor signature is only available upon reset for these processors. Refer to the programming examples at the end of this Application Note to determine which processors support the CPUID instruction.

On Intel386 processors, the format of the processor signature is somewhat different, as Figure 2 shows. Table 4 gives the current values.

Table 3. Intel486™ and Pentium™ 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
0101	0001	xxxx	Pentium™ Processors (510\60, 567\66)
0101	0010	xxxx	Pentium Processors (735\90, 815\100)
0101	0011	xxxx	Pentium OverDrive Processors
0101	0101	xxxx	Reserved for Pentium OverDrive Processor for IntelDX4 Processor
0101	0010	xxxx	Reserved for Pentium OverDrive Processor for Pentium Processor (510\60, 567\66)
0101	0100	xxxx	Reserved for Pentium OverDrive Processor for Pentium Processor (735\90, 815\100)

NOTES:
1. Intel releases information about stepping numbers as needed.
2. This processor does not implement the CPUID instruction.

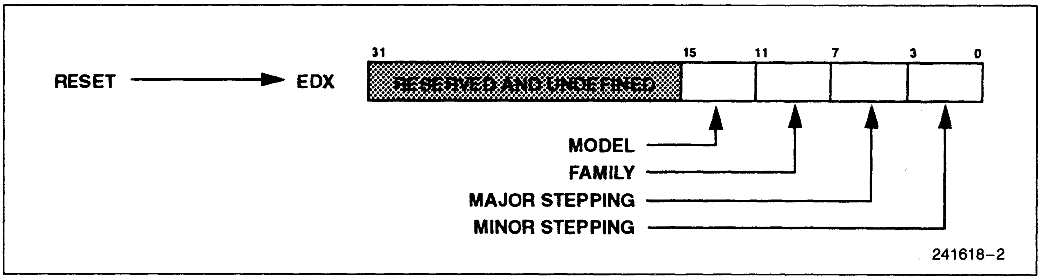


Figure 2. Processor Signature Format on Intel386™ Processors

Table 4. Intel386™ Processor Signatures

Model	Family	Major Stepping	Minor Stepping[1]	Description
0000	0011	0000	xxxx	Intel386™ DX Processor
0010	0011	0000	xxxx	Intel386 SX Processor
0010	0011	0000	xxxx	Intel386 CX Processor
0010	0011	0000	xxxx	Intel386 EX Processor
0100	0011	0000 and 0001	xxxx	Intel386 SL Processor
0000	0011	0100	xxxx	RAPIDCAD™ Processor

NOTE:
1. Intel releases information about minor stepping numbers as needed.

3.3 Feature Flags

When a value of 1 is placed in the EAX register, the CPUID instruction loads the EDX register with the feature flags. The feature flags indicate which features the processor supports. A value of 1 in a feature flag can indicate that a feature is either supported or not supported, depending on the implementation of the CPUID instruction for a specific processor. Table 5 lists the currently defined feature flag values. For future processors, refer to the programmer's reference manual, user's manual, or the appropriate documentation for the latest feature flag values.

Developers should use the feature flags in applications to determine which processor features are supported. By using the CPUID feature flags to predetermine processor features, software can detect and avoid incompatibilities that could result if the features are not present.

Table 5. Feature Flag Values

Bit	Name	Description When Flag = 1	Comments
0	FPU	Floating-Point Unit On-Chip	The processor contains an FPU that supports the Intel387 floating-point instruction set.
1	VME	Virtual Mode Extension	The processor supports extensions to virtual-8086 mode.
2[1]			(See note)
3	PSE	Page Size Extension	The processor supports 4-Mbyte pages.
4–6[1]			(See note)
7	MCE	Machine Check	Exception 18 is defined for Pentium processor style machine checks, including CR4.MCE for controlling the feature. This feature does not define the model-specific implementation of the machine-check error logging reporting and processor shutdowns. Machine-check exception handlers may have to depend on processor version to do model-specific processing of the exception or test for the presence of the standard machine-check feature.
8	CX8	CMPXCHG8B	The 8-byte (64-bit) compare and exchange instructions is supported (implicitly locked and atomic).
9	APIC	On-Chip APIC	Indicates that an integrated APIC is present and hardware enabled. (Software disabling does not affect this bit.)
10–31[1]			(See note)

NOTE:

[1] Some non-essential information regarding Intel486 and Pentium processors is considered Intel confidential and proprietary and is not documented in this publication. This information is provided in the *Supplement to the Pentium™ Processor User's Manual* and is available with the appropriate non-disclosure agreements in place. Contact Intel Corporation for details.

4.0 USAGE GUIDELINES

This document presents Intel-recommended feature-detection methods. Software should not try to identify features by exploiting programming tricks, undocumented features, or otherwise deviating from the guidelines presented in this Application Note. The following is a list of guidelines that can help programmers maintain the widest range of compatibility for their software.

- Do not depend on the absence of an invalid opcode trap on the CPUID opcode to detect CPUID. Do not depend on the absence of an invalid opcode trap on the PUSHFD opcode to detect a 32-bit processor. Test the ID flag, as described in Section 2.0 and shown in Section 6.0.
- Do not assume that a given family or model has any specific feature. For example, do not assume that, because the family value is 5 (Pentium processor), there must be a floating-point unit on-chip. Use the feature flags for this determination.
- Do not assume that the features in the OverDrive processors are the same as those in the OEM version of the processor. Internal caches and instruction execution might vary.
- Do not use undocumented features of a processor to identify steppings or features. For example, the Intel386 processor A-step had bit instructions that were withdrawn with B-step. Some software attempted to execute these instructions and depended on the invalid-opcode exception as a signal that it was not running on the A-step part. This software failed to word correctly when the Intel486 processor used the same opcodes for different instructions. That software should have used the stepping information in the processor signature.
- Do not assume that a value of 1 in a feature flag ind icates that a given feature is present, even though that is the case in the first models of the Pentium processor in which the CPUID instruction is implemented. For some feature flags that might be defined in the future, a value of 1 can indicate that the corresponding feature is not present.
- Programmers should test feature flags individually and not make assumptions about undefined bits. It would be a mistake, for example, to test the FPU bit by comparing the feature register to a binary 1 with a compare instruction.
- Do not assume that the clock of a given family or model runs at a specific frequency and do not write clock-dependent code, such as timing loops. For instance, an OverDrive Processor could operate at a higher internal frequency and still report the same family and/or model. Instead, use the system's timers to measure elapsed time.
- Processor model-specific registers may differ among processors, including in various models of the Pentium processor. Do not use these registers unless identified for the installed processor.

5.0 BIOS RECOGNITION FOR INTEL OVERDRIVE™ PROCESSORS

A system's BIOS will typically identify the processor in the system and initialize the hardware accordingly. In many cases, the BIOS identifies the processor by reading the processor signature, comparing it to known signatures, and, upon finding a match, executing the corresponding hardware initialization code.

The Pentium OverDrive processor is designed to be an upgrade to any Intel486 family processor. Because there are significant operational differences between these two processor families, processor misidentification can cause system failures or diminished performance. Major differences between the Intel486 processor and the Pentium OverDrive processor include the type of on-chip cache supported (write-back or write-through), cache organization and cache size. The OverDrive processor also has an enhanced floating point unit and System Management Mode (SMM) that may not exist in the OEM processor. Inability to recognize these features causes problems like those described below.

In many BIOS implementations, the BIOS reads the processor signature at reset and compares it to known values. If the OverDrive processor's signature is not among the known values, a match will not occur and the OverDrive processor will not be identified. Often the BIOS will drop out of the search and initialize the hardware based on a default case such as initializing the chipset for an Intel486 SX processor. Following are two common examples of system failures and how to avoid them.

Example 1

If (for the Pentium OverDrive processor) the system's hardware is configured to enable the write-back cache but the BIOS fails to detect the Pentium OverDrive processor signature, the BIOS may incorrectly cause the chipset to support a write-through processor cache. This results in a data incoherency problem with the bus masters. When a bus master accesses a memory location (which was also in the processor's cache in a modified state), the processor will alert the chipset to allow it to update this data in memory. But the chipset is not programmed for such an event and the bus master instead receives stale data. This usually results in a system failure.

Example 2

If the BIOS does not recognize the OverDrive processor's signature and defaults to an Intel486 SX processor, the BIOS can incorrectly program the chipset to ignore, or improperly route, the assertion of the floating point error signaled by the processor. The result is that floating point errors will be improperly handled by the Pentium OverDrive processor. The BIOS may also completely disable math exception handling in the OverDrive processor. This can cause installation errors in applications that require hardware support for floating point instructions.

Hence, when programming or modifying a BIOS, be aware of the impact of future OverDrive processors. Intel recommends that you include processor signatures for the OverDrive processors in BIOS identification routines to eliminate diminished performance or system failures. The recommendations in this application note can help a BIOS maintain compatibility across a wide range of processor generations and models.

6.0 PROPER IDENTIFICATION SEQUENCE

The `cpuid3a.asm` program example demonstrates the correct use of the CPUID instruction. (See Example 1.) It also shows how to identify earlier processor generations that do not implement the processor signature or CPUID instruction. This program example contains the following two procedures:

- `get_cpu_type` identifies the processor type. Figure 3 illustrates the flow of this procedure.
- `get_fpu_type` determines the type of floating-point unit (FPU) or math coprocessor (MCP).

This procedure has been tested with 8086, 80286, Intel386, Intel486, and Pentium processors. This program example is written in assembly language and is suitable for inclusion in a run-time library, or as system calls in operating systems.

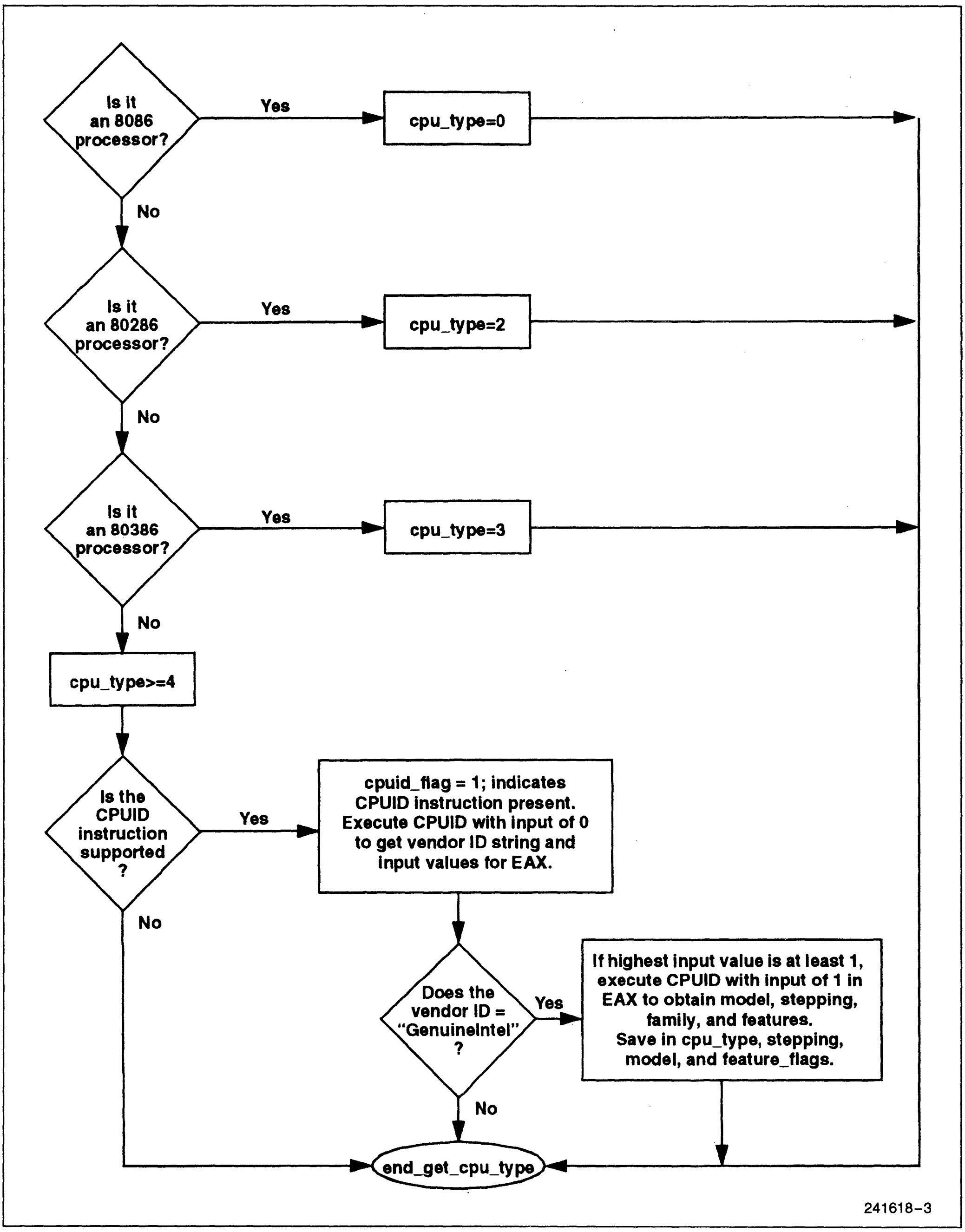
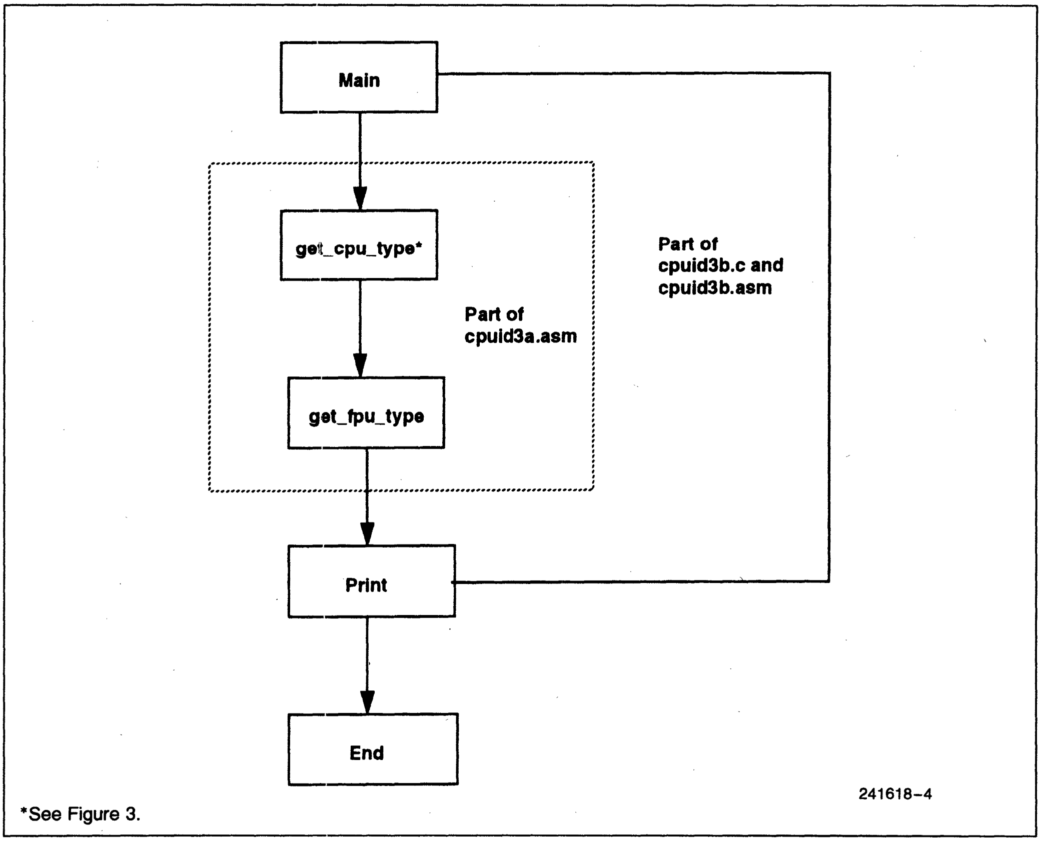

Is it an 8086 processor?
Yes
cpu_type=0
No
Is it an 80286 processor?
Yes
cpu_type=2
No
Is it an 80386 processor?
Yes
cpu_type=3
No
cpu_type>=4
Is the CPUID instruction supported ?
Yes
cpuid_flag = 1; indicates CPUID instruction present. Execute CPUID with input of 0 to get vendor ID string and input values for EAX.
No
Does the vendor ID = "GenuineIntel" ?
Yes
If highest input value is at least 1, execute CPUID with input of 1 in EAX to obtain model, stepping, family, and features. Save in cpu_type, stepping, model, and feature_flags.
No
end_get_cpu_type
241618-3

Figure 3. Flow of Processor get__cpu__type Procedure

7.0 USAGE PROGRAM EXAMPLE

The cpuid3b.asm and cpuid3b.c program examples demonstrate applications that call get_cpu_type and get_fpu_type procedures and interpret the returned information. The results, which are displayed on the monitor, identify the installed processor and features. The cpuid3b.asm example is written in assembly language and demonstrates an application that displays the returned information in the DOS environment. The cpuid3b.c example is written in the C language. (See Examples 2 and 3.)

Figure 4 presents an overview of the relationship between the three program examples.



*See Figure 3.

Figure 4. Flow of Processor Identification Extraction Procedures

Example 1. Processor Identification Extraction Procedure

```
;       Filename:       cpuid3a.asm
;       Copyright 1993, 1994 by Intel Corp.
;
;       This program has been developed by Intel Corporation.  You
;       have Intel's permission to incorporate this source code into
;       your product, royalty free.  Intel has intellectual property
;       rights which it may assert if another manufacturer's processor
;       mis-identifies itself as being "GenuineIntel" when the CPUID
;       instruction is executed.
;
;       Intel specifically disclaims all warranties, express or
;       implied, and all liability, including consequential and other
;       indirect damages, for the use of this code, including
;       liability for infringement of any proprietary rights, and
;       including the warranties of merchantability and fitness for a
;       particular purpose.  Intel does not assume any responsibility
;       for any errors which may appear in this code nor any
;       responsibility to update it.
;
;       This code contains two procedures:
;       _get_cpu_type: Identifies processor type in _cpu_type:
;               0=8086/8088 processor
;               2=Intel 286 processor
;               3=Intel386(TM) family processor
;               4=Intel486(TM) family processor
;               5=Pentium(TM) family processor
;
;       _get_fpu_type: Identifies FPU type in _fpu_type:
;               0=FPU not present
;               1=FPU present
;               2=287 present (only if _cpu_type=3)
;               3=387 present (only if _cpu_type=3)
;
;       This program has been tested with the MASM assembler.
;       This code correctly detects the current Intel 8086/8088,
;       80286, 80386, 80486, and Pentium(tm) processors in the
;       real-address mode.
;
;       To assemble this code with TASM, add the JUMPS directive.
;       jumps                   ; Uncomment this line for TASM


        TITLE   cpuid3a.asm
        DOSSEG
        .model  small

CPU_ID  MACRO
```

241618–5

```
        db      0fh                 ; Hardcoded CPUID instruction
        db      0a2h
ENDM

        .data
        public  _cpu_type
        public  _fpu_type
        public  _cpuid_flag
        public  _intel_CPU
        public  _vendor_id
        public  _cpu_signature
        public  _features_ecx
        public  _features_edx
        public  _features_ebx
_cpu_type       db      0
_fpu_type       db      0
_cpuid_flag     db      0
_intel_CPU      db      0
_vendor_id      db      "------------"
intel_id        db      "GenuineIntel"
_cpu_signature  dd      0
_features_ecx   dd      0
_features_edx   dd      0
_features_ebx   dd      0
fp_status       dw      0

        .code
        .8086

;*********************************************************************

        public  _get_cpu_type
_get_cpu_type   proc

;       This procedure determines the type of processor in a system
;       and sets the _cpu_type variable with the appropriate
;       value.  If the CPUID instruction is available, it is used
;       to determine more specific details about the processor.
;       All registers are used by this procedure, none are preserved.
;       To avoid AC faults, the AM bit in CR0 must not be set.

;       Intel 8086 processor check
;       Bits 12-15 of the FLAGS register are always set on the
;       8086 processor.

check_8086:
        pushf                       ; push original FLAGS
        pop     ax                  ; get original FLAGS
        mov     cx, ax              ; save original FLAGS
        and     ax, 0fffh           ; clear bits 12-15 in FLAGS
```

241618-6

```
        push    ax                  ; save new FLAGS value on stack
        popf                        ; replace current FLAGS value
        pushf                       ; get new FLAGS
        pop     ax                  ; store new FLAGS in AX
        and     ax, 0f000h          ; if bits 12-15 are set, then
        cmp     ax, 0f000h          ;   processor is an 8086/8088
        mov     _cpu_type, 0        ; turn on 8086/8088 flag
        je      end_cpu_type        ; jump if processor is 8086/8088

;       Intel 286 processor check
;       Bits 12-15 of the FLAGS register are always clear on the
;       Intel 286 processor in real-address mode.

        .286
check_80286:
        or      cx, 0f000h          ; try to set bits 12-15
        push    cx                  ; save new FLAGS value on stack
        popf                        ; replace current FLAGS value
        pushf                       ; get new FLAGS
        pop     ax                  ; store new FLAGS in AX
        and     ax, 0f000h          ; if bits 12-15 are clear
        mov     _cpu_type, 2        ; processor=80286, turn on 80286 flag
        jz      end_cpu_type        ; if no bits set, processor is 80286

;       Intel386 processor check
;       The AC bit, bit #18, is a new bit introduced in the EFLAGS
;       register on the Intel486 processor to generate alignment
;       faults.
;       This bit cannot be set on the Intel386 processor.

        .386                        ; it is safe to use 386 instructions
check_80386:
        pushfd                      ; push original EFLAGS
        pop     eax                 ; get original EFLAGS
        mov     ecx, eax            ; save original EFLAGS
        xor     eax, 40000h         ; flip AC bit in EFLAGS
        push    eax                 ; save new EFLAGS value on stack
        popfd                       ; replace current EFLAGS value
        pushfd                      ; get new EFLAGS
        pop     eax                 ; store new EFLAGS in EAX
        xor     eax, ecx            ; can't toggle AC bit, processor=80386
        mov     _cpu_type, 3        ; turn on 80386 processor flag
        jz      end_cpu_type        ; jump if 80386 processor

        push    ecx
        popfd                       ; restore AC bit in EFLAGS first

;       Intel486 processor check
;       Checking for ability to set/clear ID flag (Bit 21) in EFLAGS
;       which indicates the presence of a processor with the CPUID
```

241618-7

```
;       instruction.

        .486
check_80486:
        mov     _cpu_type, 4      ; turn on 80486 processor flag
        mov     eax, ecx          ; get original EFLAGS
        xor     eax, 200000h      ; flip ID bit in EFLAGS
        push    eax               ; save new EFLAGS value on stack
        popfd                     ; replace current EFLAGS value
        pushfd                    ; get new EFLAGS
        pop     eax               ; store new EFLAGS in EAX
        xor     eax, ecx          ; can't toggle ID bit,
        je      end_cpu_type      ; processor=80486

;       Execute CPUID instruction to determine vendor, family,
;       model, stepping and features.  For the purpose of this
;       code, only the initial set of CPUID information is saved.

        mov     _cpuid_flag, 1    ; flag indicating use of CPUID inst.
        push    ebx               ; save registers
        push    esi
        push    edi
        mov     eax, 0            ; set up for CPUID instruction
        CPU_ID                    ; get and save vendor ID

        mov     dword ptr _vendor_id, ebx
        mov     dword ptr _vendor_id[+4], edx
        mov     dword ptr _vendor_id[+8], ecx

        mov     si, ds
        mov     es, si

        mov     si, offset _vendor_id
        mov     di, offset intel_id
        mov     cx, 12            ; should be length intel_id
        cld                       ; set direction flag
        repe    cmpsb             ; compare vendor ID to "GenuineIntel"
        jne     end_cpuid_type    ; if not equal, not an Intel processor

        mov     _intel_CPU, 1     ; indicate an Intel processor
        cmp     eax, 1            ; make sure 1 is valid input for CPUID
        jl      end_cpuid_type    ; if not, jump to end
        mov     eax, 1
        CPU_ID                    ; get family/model/stepping/features
        mov     _cpu_signature, eax
        mov     _features_ebx, ebx
        mov     _features_edx, edx
        mov     _features_ecx, ecx

        shr     eax, 8            ; isolate family
```

241618-8

```
        and     eax, 0fh
        mov     _cpu_type, al    ; set _cpu_type with family

end_cpuid_type:
        pop     edi              ; restore registers
        pop     esi
        pop     ebx

        .8086
end_cpu_type:
        ret
_get_cpu_type   endp

;**********************************************************************

        public  _get_fpu_type
_get_fpu_type   proc

;       This procedure determines the type of FPU in a system
;       and sets the _fpu_type variable with the appropriate value.
;       All registers are used by this procedure, none are preserved.

;       Coprocessor check
;       The algorithm is to determine whether the floating-point
;       status and control words are present.  If not, no
;       coprocessor exists.  If the status and control words can
;       be saved, the correct coprocessor is then determined
;       depending on the processor type.  The Intel386 processor can
;       work with either an Intel287 NDP or an Intel387 NDP.
;       The infinity of the coprocessor must be checked to determine
;       the correct coprocessor type.

        fninit                   ; reset FP status word
        mov     fp_status, 5a5ah; initialize temp word to non-zero
        fnstsw  fp_status        ; save FP status word
        mov     ax, fp_status    ; check FP status word
        cmp     al, 0            ; was correct status written
        mov     _fpu_type, 0     ; no FPU present
        jne     end_fpu_type

check_control_word:
        fnstcw  fp_status        ; save FP control word
        mov     ax, fp_status    ; check FP control word
        and     ax, 103fh        ; selected parts to examine
        cmp     ax, 3fh          ; was control word correct
        mov     _fpu_type, 0
        jne     end_fpu_type     ; incorrect control word, no FPU
        mov     _fpu_type, 1

;       80287/80387 check for the Intel386 processor
```

241618–9

2

```
check_infinity:
        cmp     _cpu_type, 3
        jne     end_fpu_type
        fld1                    ; must use default control from FNINIT
        fldz                    ; form infinity
        fdiv                    ; 8087/Intel287 NDP say +inf = -inf
        fld     st              ; form negative infinity
        fchs                    ; Intel387 NDP says +inf <> -inf
        fcompp                  ; see if they are the same
        fstsw   fp_status       ; look at status from FCOMPP
        mov     ax, fp_status
        mov     _fpu_type, 2    ; store Intel287 NDP for FPU type
        sahf                    ; see if infinities matched
        jz      end_fpu_type    ; jump if 8087 or Intel287 is present
        mov     _fpu_type, 3    ; store Intel387 NDP for FPU type
end_fpu_type:
        ret
_get_fpu_type   endp

        end
```

241618-10

Example 2. Processor Identification Procedure in Assembly Language

```
;       Filename:       cpuid3b.asm
;       Copyright 1993, 1994 by Intel Corp.
;
;       This program has been developed by Intel Corporation.  You
;       have Intel's permission to incorporate this source code into
;       your product, royalty free.  Intel has intellectual property
;       rights which it may assert if another manufacturer's processor
;       mis-identifies itself as being "GenuineIntel" when the CPUID
;       instruction is executed.
;
;       Intel specifically disclaims all warranties, express or
;       implied, and all liability, including consequential and other
;       indirect damages, for the use of this code, including
;       liability for infringement of any proprietary rights, and
;       including the warranties of merchantability and fitness for a
;       particular purpose.  Intel does not assume any responsibility
;       for any errors which may appear in this code nor any
;       responsibility to update it.
;
;       This program contains three parts:
;       Part 1: Identifies processor type in the variable _cpu_type:
;
;       Part 2: Identifies FPU type in the variable _fpu_type:
;
;       Part 3: Prints out the appropriate message.  This part is
;               specific to the DOS environment and uses the DOS
;               system calls to print out the messages.
;
;       This program has been tested with the MASM assembler.
;       If this code is assembled with no options specified and linked
;       with the cpuid3a.asm module, it correctly identifies the
;       current Intel 8086/8088, 80286, 80386, 80486, and Pentium(tm)
;       processors in the real-address mode.
;
;       To assemble this code with TASM, add the JUMPS directive.
;       jumps                   ; Uncomment this line for TASM


        TITLE   cpuid3b.asm
        DOSSEG
        .model  small
        .stack  100h

        .data
        extrn   _cpu_type: byte
        extrn   _fpu_type: byte
        extrn   _cpuid_flag: byte
```

241618-11

```
        extrn   _intel_CPU: byte
        extrn   _vendor_id: byte
        extrn   _cpu_signature: dword
        extrn   _features_ecx: dword
        extrn   _features_edx: dword
        extrn   _features_ebx: dword

;       The purpose of this code is to identify the processor and
;       coprocessor that is currently in the system.  The program
;       first determines the processor type.  Then it determines
;       whether a coprocessor exists in the system.  If a
;       coprocessor or integrated coprocessor exists, the program
;       identifies the coprocessor type.  The program then prints
;       the processor and floating point processors present and type.

        .code
        .8086
start:  mov     ax, @data
        mov     ds, ax          ; set segment register
        mov     es, ax          ; set segment register
        and     sp, not 3       ; align stack to avoid AC fault
        call    _get_cpu_type   ; determine processor type
        call    _get_fpu_type
        call    print
        mov     ax, 4c00h       ; terminate program
        int     21h

;**********************************************************************

        extrn   _get_cpu_type: proc

;**********************************************************************

        extrn   _get_fpu_type: proc

;**********************************************************************


FPU_FLAG        equ     0001h
VME_FLAG        equ     0002h
PSE_FLAG        equ     0008h
MCE_FLAG        equ     0080h
CMPXCHG8B_FLAG  equ     0100h
APIC_FLAG       equ     0200h

        .data
id_msg          db      "This system has a$"
cp_error        db      "n unknown processor$"
cp_8086         db      "n 8086/8088 processor$"
cp_286          db      "n 80286 processor$"
```

241618-12

```
cp_386          db      "n 80386 processor$"

cp_486          db      "n 80486DX, 80486DX2 processor or"
                db      " 80487SX math coprocessor$"
cp_486sx        db      "n 80486SX processor$"

fp_8087         db      " and an 8087 math coprocessor$"
fp_287          db      " and an 80287 math coprocessor$"
fp_387          db      " and an 80387 math coprocessor$"

intel486_msg    db      " Genuine Intel486(TM) processor$"
intel486dx_msg  db      " Genuine Intel486(TM) DX processor$"
intel486sx_msg  db      " Genuine Intel486(TM) SX processor$"
inteldx2_msg    db      " Genuine IntelDX2(TM) processor$"
intelsx2_msg    db      " Genuine IntelSX2(TM) processor$"
inteldx4_msg    db      " Genuine IntelDX4(TM) processor$"
inteldx2wb_msg  db      " Genuine Write-Back Enhanced"
                db      " IntelDX2(TM) processor$"
pentium_msg     db      " Genuine Intel Pentium(TM) processor$"
unknown_msg     db      "n unknown Genuine Intel processor$"

; The following 16 entries must stay intact as an array
intel_486_0     dw      offset intel486dx_msg
intel_486_1     dw      offset intel486dx_msg
intel_486_2     dw      offset intel486sx_msg
intel_486_3     dw      offset inteldx2_msg
intel_486_4     dw      offset intel486_msg
intel_486_5     dw      offset intelsx2_msg
intel_486_6     dw      offset intel486_msg
intel_486_7     dw      offset inteldx2wb_msg
intel_486_8     dw      offset inteldx4_msg
intel_486_9     dw      offset intel486_msg
intel_486_a     dw      offset intel486_msg
intel_486_b     dw      offset intel486_msg
intel_486_c     dw      offset intel486_msg
intel_486_d     dw      offset intel486_msg
intel_486_e     dw      offset intel486_msg
intel_486_f     dw      offset intel486_msg
; end of array

family_msg      db      13,10,"Processor Family:  $"
model_msg       db      13,10,"Model:             $"
stepping_msg    db      13,10,"Stepping:          "
cr_lf           db      13,10,"$"

turbo_msg       db      13,10,"The processor is an OverDrive(TM)"
                db      " processor$"
dp_msg          db      13,10,"The processor is the upgrade processor"
                db      " in a dual processor system$"
fpu_msg         db      13,10,"The processor contains an on-chip FPU$"
```

241618-13

2

```
mce_msg         db      13,10,"The processor supports Machine Check"
                db      " Exceptions$"
cmp_msg         db      13,10,"The processor supports the CMPXCHG8B"
                db      " instruction$"
vme_msg         db      13,10,"The processor supports Virtual Mode"
                db      " Extensions$"
pse_msg         db      13,10,"The processor supports Page Size"
                db      " Extensions$"
apic_msg        db      13,10,"The processor contains an on-chip"
                db      " APIC$"

not_intel       db      "t least an 80486 processor."
                db      13,10,"It does not contain a Genuine Intel"
                db      " part and as a result, the",13,10,"CPUID"
                db      " detection information cannot be determined"
                db      " at this time.$"

ASC_MSG MACRO   msg
        LOCAL   ascii_done              ; local label
        add     al, 30h
        cmp     al, 39h                 ; is it 0-9?
        jle     ascii_done
        add     al, 07h
ascii_done:
        mov     byte ptr msg[20], al
        mov     dx, offset msg
        mov     ah, 9h
        int     21h
ENDM

        .code
        .8086
print   proc

;       This procedure prints the appropriate cpuid string and
;       numeric processor presence status.  If the CPUID instruction
;       was used, this procedure prints out the CPUID info.
;       All registers are used by this procedure, none are preserved.

        mov     dx, offset id_msg       ; print initial message
        mov     ah, 9h
        int     21h

        cmp     _cpuid_flag, 1          ; if set to 1, processor
                                        ;   supports CPUID instruction
        je      print_cpuid_data        ; print detailed CPUID info

print_86:
        cmp     _cpu_type, 0
        jne     print_286
```

241618–14

```
        mov     dx, offset cp_8086
        mov     ah, 9h
        int     21h
        cmp     _fpu_type, 0
        je      end_print
        mov     dx, offset fp_8087
        mov     ah, 9h
        int     21h
        jmp     end_print

print_286:
        cmp     _cpu_type, 2
        jne     print_386
        mov     dx, offset cp_286
        mov     ah, 9h
        int     21h
        cmp     _fpu_type, 0
        je      end_print
print_287:
        mov     dx, offset fp_287
        mov     ah, 9h
        int     21h
        jmp     end_print

print_386:
        cmp     _cpu_type, 3
        jne     print_486
        mov     dx, offset cp_386
        mov     ah, 9h
        int     21h
        cmp     _fpu_type, 0
        je      end_print
        cmp     _fpu_type, 2
        je      print_287
        mov     dx, offset fp_387
        mov     ah, 9h
        int     21h
        jmp     end_print

print_486:
        cmp     _cpu_type, 4
        jne     print_unknown           ; Intel processors will have
        mov     dx, offset cp_486sx     ; CPUID instruction
        cmp     _fpu_type, 0
        je      print_486sx
        mov     dx, offset cp_486
print_486sx:
        mov     ah, 9h
        int     21h
        jmp     end_print
```

241618–15

```
print_unknown:
        mov     dx, offset cp_error
        jmp     print_486sx

print_cpuid_data:
        .486
        cmp     _intel_CPU, 1           ; check for genuine Intel
        jne     not_GenuineIntel        ;   processor
print_486_type:
        cmp     _cpu_type, 4            ; if 4, print 80486 processor
        jne     print_pentium_type
        mov     ax, word ptr _cpu_signature
        shr     ax, 4
        and     eax, 0fh                ; isolate model
        mov     dx, intel_486_0[eax*2]
        jmp     print_common
print_pentium_type:
        cmp     _cpu_type, 5            ; if 5, print Pentium processor
        jne     print_unknown_type
        mov     dx, offset pentium_msg
        jmp     print_common
print_unknown_type:
        mov     dx, offset unknown_msg  ; if neither, print unknown

print_common:
        mov     ah, 9h
        int     21h

; print family, model, and stepping

print_family:
        mov     al, _cpu_type
        ASC_MSG family_msg              ; print family msg

print_model:
        mov     ax, word ptr _cpu_signature
        shr     ax, 4
        and     al, 0fh
        ASC_MSG model_msg               ; print model msg

print_stepping:
        mov     ax, word ptr _cpu_signature
        and     al, 0fh
        ASC_MSG stepping_msg            ; print stepping msg

print_upgrade:
        mov     ax, word ptr _cpu_signature
        test    ax, 1000h               ; check for turbo upgrade
        jz      check_dp
```

241618-16

```
            mov     dx, offset turbo_msg
            mov     ah, 9h
            int     21h
            jmp     print_features

    check_dp:
            test    ax, 2000h                   ; check for dual processor
            jz      print_features
            mov     dx, offset dp_msg
            mov     ah, 9h
            int     21h

    print_features:
            mov     ax, word ptr _features_edx
            and     ax, FPU_FLAG                ; check for FPU
            jz      check_MCE
            mov     dx, offset fpu_msg
            mov     ah, 9h
            int     21h

    check_MCE:
            mov     ax, word ptr _features_edx
            and     ax, MCE_FLAG                ; check for MCE
            jz      check_CMPXCHG8B
            mov     dx, offset mce_msg
            mov     ah, 9h
            int     21h

    check_CMPXCHG8B:
            mov     ax, word ptr _features_edx
            and     ax, CMPXCHG8B_FLAG          ; check for CMPXCHG8B
            jz      check_VME
            mov     dx, offset cmp_msg
            mov     ah, 9h
            int     21h

    check_VME:
            mov     ax, word ptr _features_edx
            and     ax, VME_FLAG                ; check for VME
            jz      check_PSE
            mov     dx, offset vme_msg
            mov     ah, 9h
            int     21h

    check_PSE:
            mov     ax, word ptr _features_edx
            and     ax, PSE_FLAG                ; check for PSE
            jz      check_APIC
            mov     dx, offset pse_msg
            mov     ah, 9h
```

241618-17

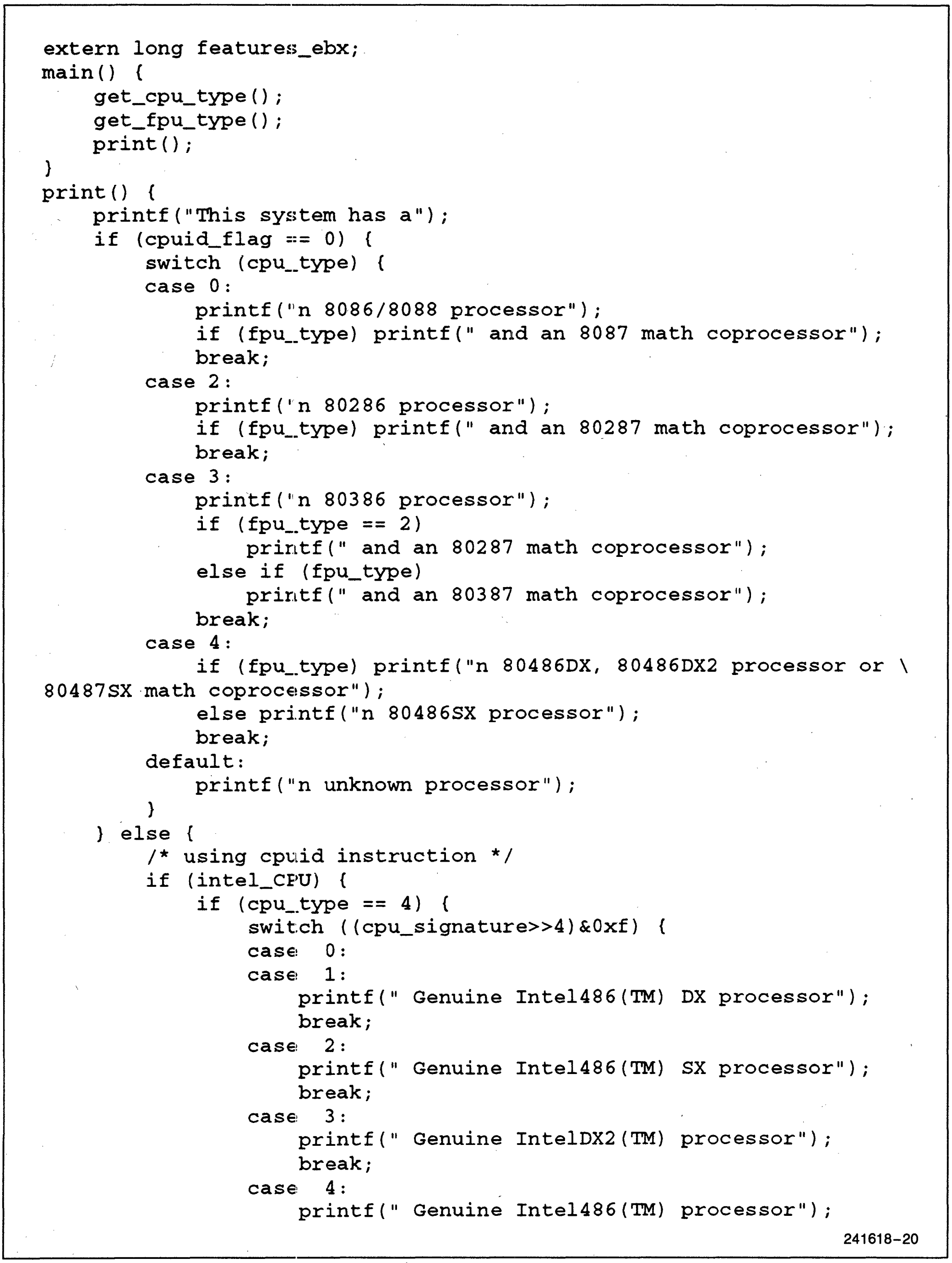
```
        int     21h

check_APIC:
        mov     ax, word ptr _features_edx
        and     ax, APIC_FLAG           ; check for APIC
        jz      end_print
        mov     dx, offset apic_msg
        mov     ah, 9h
        int     21h

        jmp     end_print

not_GenuineIntel:
        mov     dx, offset not_intel
        mov     ah, 9h
        int     21h

end_print:
        mov     dx, offset cr_lf
        mov     ah, 9h
        int     21h
        ret
print   endp

        end     start
```

241618-18

Example 3. Processor Identification Procedure in the C Language

```
/* Filename:    cpuid3b.c                                             */
/* Copyright 1994 by Intel Corp.                                      */
/*                                                                    */
/* This program has been developed by Intel Corporation.  You        */
/* have Intel's permission to incorporate this source code into      */
/* your product, royalty free. Intel has intellectual property       */
/* rights which it may assert if another manufacturer's processor*/
/* mis-identifies itself as being "GenuineIntel" when the CPUID      */
/* instruction is executed.                                           */
/*                                                                    */
/* Intel specifically disclaims all warranties, express or           */
/* implied, and all liability, including consequential and other */
/* indirect damages, for the use of this code, including             */
/* liability for infringement of any proprietary rights, and         */
/* including the warranties of merchantability and fitness for a */
/* particular purpose.  Intel does not assume any responsibility */
/* for any errors which may appear in this code nor any              */
/* responsibility to update it.                                       */
/*                                                                    */
/* This program contains three parts:                                */
/* Part 1: Identifies CPU type in the variable _cpu_type:            */
/*                                                                    */
/* Part 2: Identifies FPU type in the variable _fpu_type:            */
/*                                                                    */
/* Part 3: Prints out the appropriate message.                       */
/*                                                                    */
/* This program has been tested with the Microsoft C compiler.      */
/* If this code is compiled with no options specified and linked */
/* with the cpuid3a.asm module, it correctly identifies the          */
/* current Intel 8086/8088, 80286, 80386, 80486, and                 */
/* Pentium(tm) processors in the real-address mode.                  */

#define FPU_FLAG        0x0001
#define VME_FLAG        0x0002
#define PSE_FLAG        0x0008
#define MCE_FLAG        0x0080
#define CMPXCHG8B_FLAG  0x0100
#define APIC_FLAG       0x0200

extern char cpu_type;
extern char fpu_type;
extern char cpuid_flag;
extern char intel_CPU;
extern char vendor_id[12];
extern long cpu_signature;
extern long features_ecx;
extern long features_edx;
```

241618-19



```
extern long features_ebx;
main() {
    get_cpu_type();
    get_fpu_type();
    print();
}
print() {
    printf("This system has a");
    if (cpuid_flag == 0) {
        switch (cpu_type) {
        case 0:
            printf("n 8086/8088 processor");
            if (fpu_type) printf(" and an 8087 math coprocessor");
            break;
        case 2:
            printf("n 80286 processor");
            if (fpu_type) printf(" and an 80287 math coprocessor");
            break;
        case 3:
            printf("n 80386 processor");
            if (fpu_type == 2)
                printf(" and an 80287 math coprocessor");
            else if (fpu_type)
                printf(" and an 80387 math coprocessor");
            break;
        case 4:
            if (fpu_type) printf("n 80486DX, 80486DX2 processor or \
80487SX math coprocessor");
            else printf("n 80486SX processor");
            break;
        default:
            printf("n unknown processor");
        }
    } else {
        /* using cpuid instruction */
        if (intel_CPU) {
            if (cpu_type == 4) {
                switch ((cpu_signature>>4)&0xf) {
                case  0:
                case  1:
                    printf(" Genuine Intel486(TM) DX processor");
                    break;
                case  2:
                    printf(" Genuine Intel486(TM) SX processor");
                    break;
                case  3:
                    printf(" Genuine IntelDX2(TM) processor");
                    break;
                case  4:
                    printf(" Genuine Intel486(TM) processor");
```

241618–20

```
                    break;
                case  5:
                    printf(" Genuine IntelSX2(TM) processor");
                    break;
                case  7:
                    printf(" Genuine Write-Back Enhanced \
IntelDX2(TM) processor");
                    break;
                case  8:
                    printf(" Genuine IntelDX4(TM) processor");
                    break;
                default:
                    printf(" Genuine Intel486(TM) processor");
                }
            } else if (cpu_type == 5)
                printf(" Genuine Intel Pentium(TM) processor");
            else
                printf("n unknown Genuine Intel processor");
            printf("\nProcessor Family: %X", cpu_type);
            printf("\nModel:           %X", (cpu_signature>>4)&0xf);
            printf("\nStepping:        %X\n", cpu_signature&0xf);
            if (cpu_signature & 0x1000)
                printf("\nThe processor is an OverDrive(TM) upgrade
           \processor");
            else if (cpu_signature & 0x2000)
                printf("\nThe processor is the upgrade processor \
in a dual processor system");
            if (features_edx & FPU_FLAG)
                printf("\nThe processor contains an on-chip FPU");
            if (features_edx & MCE_FLAG)
                printf("\nThe processor supports Machine Check \
Exceptions");
            if (features_edx & CMPXCHG8B_FLAG)
                printf("\nThe processor supports the CMPXCHG8B \
instruction");
            if (features_edx & VME_FLAG)
                printf("\nThe processor supports Virtual Mode \
Extensions");
            if (features_edx & PSE_FLAG)
                printf("\nThe processor supports Page Size \
Extensions");
            if (features_edx & APIC_FLAG)
                printf("\nThe processor contains an on-chip APIC");
        } else {
            printf("t least an 80486 processor.\nIt does not \
contain a Genuine Intel part and as a result, the\nCPUID detection \
information cannot be determined at this time.");
        }
    }
    printf("\n");
}
```

241618-21

Revision	Revision History	Date
-001	Original Issue.	05/93
-002	Modified Table 2. Intel486 and Pentium Processor Signatures.	10/93
-003	Updated to accommodate new processor versions. Program examples modified for ease of use, section added discussing BIOS recognition for OverDrive processors, and feature flag information updated.	09/94

intel®

AP-496

APPLICATION NOTE

Migrating from the Intel486™ SL Microprocessor to the SL Enhanced Intel486™ Microprocessor

DESMOND YUEN
MCG TECHNICAL MARKETING

October 1993

Migrating from the Intel486™ SL Microprocessor to the SL Enhanced Intel486™ Microprocessor

CONTENTS PAGE

INTRODUCTION

Since the introduction of the Intel386™ SL microprocessor and the subsequent introduction of the Intel486™ SL microprocessor, the SL Architecture has become the *de facto* standard for mobile computers. Given the industry acceptance of SL Architecture, Intel is extending the SL Architecture to the SL Enhanced Intel486 microprocessor family. This application note describes how the same features of the SL Architecture can be implemented on the SL Enhanced Intel486 CPUs. Although this application note is written for people with experience designing Intel486 SL CPU-based mobile computers, the information provided in this document will also be useful for anyone interested in learning more about the SL Enhanced Intel486 CPUs.

The first section of this document highlights the differences between the Intel486 SL CPU and the SL Enhanced Intel486 CPU. Section two describes the architectural differences in System Management Mode (SMM). Section three discusses power management features of the Intel486 SL CPU and the SL Enhanced Intel486 CPU. Section four explains reset implementation of the Intel486 SL CPU and the SL Enhanced Intel486 CPU.

1.0 COMPARISON OF THE SL ENHANCED Intel486™ CPU AND Intel486 SL CPU

The SL Enhanced Intel486 CPU supports many of the features available in the SL Architecture. The major difference between the SL Enhanced Intel486 CPU and the Intel486 SL CPU is level of integration. The Intel486 SL CPU is a highly integrated CPU with memory controller, ISA/PI-bus controller, and power management built into it. The SL Enhanced Intel486 CPU has retained all the SMM and power management features from the SL Architecture. Features not supported by the SL Enhanced Intel486 CPU can easily be implemented by external hardware. Table 1 highlights the differences between the SL Enhanced Intel486 CPU and the Intel486 SL CPU.

Table 1. Feature Comparison of the SL Enhanced Intel486 CPU and the Intel486 SL CPU

Features	SL Enhanced Intel486 CPU	Intel486 SL CPU
System Management Mode	Yes	Yes
SMBASE Relocation	Yes	No
Stop Clock	Yes	Yes
Upgrade Power-Down Mode	Yes	No
Package Options	168 lead PGA, 196 lead PQFP, 208 lead SQFP	196 lead PQFP, 208 lead SQFP
3.3V Operation	Yes	Yes
Clocking Options	1X clock input or 2X clock input	2X clock input
CPU Frequency	Intel486 SX CPU: 25 MHz, 33 MHz Intel486 DX CPU: 33 MHz, 50 MHz Intel486 DX2 CPU: 40 MHz, 50 MHz, 66 MHz	25 MHz, 33 MHz

2.0 SYSTEM MANAGEMENT MODE IMPLEMENTATION

System Management Mode (SMM), first introduced in the SL Architecture for notebook computers, provides a unique environment for software to perform power management functions much more efficiently. Since then, SMM has found its way into many new applications. The SMM hardware interface on the SL Enhanced Intel486 CPU is similar to that of the Intel486 SL CPU except for the handshaking protocol. The SL Enhanced CPUs handshake through the SMI and SMIACT# signals (see Figure 1), and the Intel486 SL CPUs handshake through the SMI and SMRAMCS# signals.

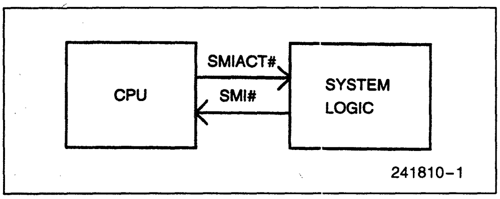


Figure 1. Basic SMI# Hardware Interface

2.1 System Management Interrupt

The system interrupts the normal program execution and invokes SMM by generating a System Management Interrupt (SMI#) to the CPU. On the Intel486 SL CPU, the SMI# input is held low as long as the CPU is in SMM. With the SL Enhanced Intel486 CPU, SMI# input only needs to remain active for a single clock provided the SMI setup and hold times, t20 and t21, are met. SMI# will also work correctly if it is held active for an arbitrary number of clocks.

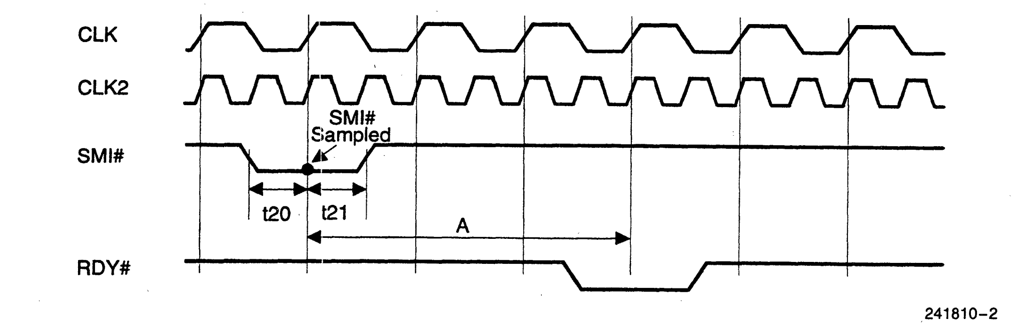
2.1.1 GENERAL DESIGN CONSIDERATIONS

For Intel486 SL processor-based systems, the 82360SL I/O generates the SMI request. For any SMI to be recognized by the SL Enhanced Intel486 CPU, the system logic must ensure all of the required timings are met. The following sections discuss the timing requirements that must be observed by the SMI generation logic interfacing to the SL Enhanced Intel486 CPU.

2.1.1.1 I/O Trapping

I/O trapping has proven to be very useful in the SL Architecture for device power management. Trapping the last I/O access prior to entering SMM prevents the CPU from accessing a powered-down device. With the exception of the SMFILO (System Management FILO), the SL Enhanced CPU supports the same I/O Instruction Restart option under SMM featured in the Intel486 SL CPU.

The I/O Instruction Restart feature of the SL Enhanced Intel486 CPU is used the same way as the I/O Instruction Restart feature of the Intel486 SL CPU. When the I/O Instruction Restart option is enabled (by setting offset 0FF00H in the SMRAM to 0FFH), the RSM instruction microcode modifies the restored EIP to point to the instruction immediately preceding the SMI# request, so that the I/O instruction can be re-executed. *For the CPU to trap the last I/O access correctly, the external hardware must ensure the SMI# signal is asserted at least three CPU clock periods prior to asserting the RDY# signal (see Figure 2).*



NOTE:
A: Setup time for recognition on I/O instruction boundary

Figure 2. SMI# Timing when Servicing an I/O Trap

2.1.1.2 Back-to-Back SMIs

For back-to-back SMIs, the SMI# input must be held inactive for at least four clocks after it is de-asserted to reset the edge-triggered SMI detection logic. Otherwise, the second SMI# request may not be recognized (see Figure 3).

2.2 SMI Active (SMIACT#)

A new pin called SMIACT# (SMI ACTive) which indicates that the CPU is operating in SMM has been added to the SL Enhanced Intel486 CPU. The CPU asserts SMIACT# in response to an SMI interrupt request on the SMI# pin. SMIACT# is driven active by the CPU before accessing the SMRAM. SMIACT# remains active until the last access to SMRAM when the CPU restores (reads) its state from SMRAM. After the RSM instruction is executed, the CPU de-asserts the SMIACT# signal.

The SMIACT# signal is equivalent to the SMRAMCS# signal on the Intel486 SL microprocessor except that SMIACT# on the Intel486 SL CPU is active all the time and cannot be used as a chip select signal for external SMRAM. On the Intel486 SL microprocessor, the SMRAM is enabled automatically whenever the CPU is switched into SMM. A similar mechanism can also be implemented by using the SMIACT# signal. Whenever the SMIACT# signal is active, the SMRAM will be enabled by the system logic.

2.2.1 GENERAL DESIGN CONSIDERATIONS

As previously mentioned, one of the many uses of the SMIACT# output is to enable SMRAM when the CPU is operating in SMM. Most importantly, the SMIACT# output is used by the system logic to maintain system integrity while the CPU is in SMM.

If part of the system memory is overlaid by the SMRAM while the CPU is in SMM, the system logic should ensure that only the CPU and SMI handler have access to the SMRAM area. Accesses to addresses overlaid by a bus master or DMA controller when SMIACT# is active should be re-directed to the system memory underneath the SMRAM and not the SMRAM itself.

While inside SMM, the CPU should be protected from system activities such as CPU RESET, interrupt requests, and NMI, etc. The SMIACT# can be used by the system logic to block off these system activities while the CPU is in SMM.

2.3 SMRAM Interfacing

SMRAM resides in a unique address space so that the software operating under SMM is transparent to the normal address space. On the Intel486 SL microprocessor, the size of SMRAM can be either 32 Kbytes or 64 Kbytes. Depending on the size of the SMRAM, the SMRAM area can be located in either 38000H–3FFFFH or 30000H–3FFFFH.

On the SL Enhanced Intel486 CPU, the size of the SMRAM can be between 32 Kbytes and 4 Gbytes. The location of the SMRAM is determined by the SMBASE (SMRAM BASE ADDRESS) register, and defaults to SMBASE + 8000H, which is 38000H after CPU RESET. The first SMI after a CPU RESET always begins executing instructions at 38000H.

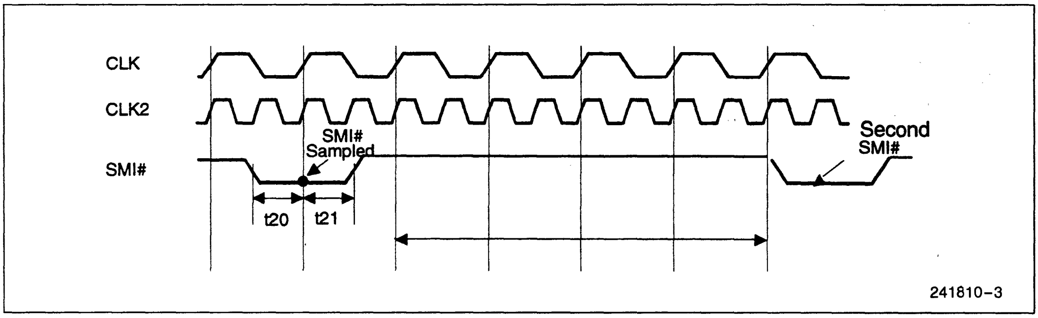


Figure 3. Back-To-Back SMI# Timing

2.3.1 SMRAM INITIALIZATION

The SL Enhanced Intel486 CPU family provides a new control register, SMBASE for changing the SMRAM base address (see Figure 4). The SMRAM base address can be changed after CPU RESET by invoking a dummy SMI call to change the SMBASE register.

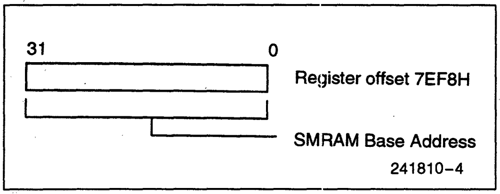


Figure 4. SMBASE Register

In the SL Enhanced Intel486 CPU, a new slot is added to the CPU dump area inside the SMRAM at offset 7EF8H for changing the SMRAM base address. During the execution of the RSM instruction, if the relocation bit is set, the CPU will read this slot and initialize the CPU to use the new SMBASE during the next SMI. From then on, the CPU 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 execute the new jump vector based on the current SMBASE.

SMBASE must start at a 32K aligned boundary. **Programming the SMBASE register to values that are not 32K aligned will cause the CPU to enter the shutdown state when executing the RSM instruction.** After the SMRAM base address is changed, the new starting address for the SMI jump vector is calculated by adding 8000H to the new SMRAM base address. The starting address for CPU state dump area will be remapped to the new SMRAM base address plus 0FFFFH.

A new bit (bit 17) has been added to the SMM Revision Identifier on the SL Enhanced Intel486 CPU to indicate whether the processor supports relocation of the SMRAM base address. With the SL Enhanced Intel486 CPU, the SMBASE relocation bit is always set to one to indicate the processor supports SMBASE relocation.

2.3.2 GENERAL DESIGN CONSIDERATIONS

Since the memory controller is embedded inside the Intel486 SL CPU, many design issues with SMRAM interface are handled internally by the CPU. For the SL Enhanced Intel486 CPU, these issues must be handled by the external system logic interfacing to the CPU.

2.3.2.1 Accessing SMRAM

Before the CPU can execute code inside SMM, the SMRAM must be loaded with valid SMM code. If the SMRAM is not initialized with code prior to entering SMM, executing invalid code out of the SMRAM can place the CPU in an unknown state. Thus, the external memory controller must provide a mechanism to bring the SMRAM into system address space without invoking SMM. This will allow software such as BIOS to load the SMM code into SMRAM.

The Intel486 SL CPU provides a hardware mechanism to access memory overlaid by SMRAM. Although system logic is not required to provide a mechanism to access memory located underneath SMRAM, it may be much easier to implement a suspend state (0-volt suspend or 5-volt suspend) if such a mechanism is provided.

2.3.2.2 Cache Coherency

Since the Intel486 SL CPU does not support a second level cache, cache coherency with SMRAM is handled completely inside the CPU. The CPU's internal cache is automatically emptied before entering SMM and after exiting SMM.

The SL Enhanced Intel486 CPU does not flush its cache before entering SMM or after leaving SMM. Cache flushing is not required if the SMRAM is located in a non-cacheable area in the memory address space or in an external address space which is not visible to the system. If the SMRAM is located in a cacheable area that overlays system memory, both the CPU internal cache and any second level caches must be flushed before entering SMM. If SMRAM is cacheable, the CPU internal cache and any second level caches must also be flushed when exiting SMM. The following steps must be taken by the system logic to maintain cache coherency when SMM overlays normal system memory:

1. Before entering SMM, the FLUSH# pin should be asserted when SMIACT# is driven active to empty the CPU cache.
2. The KEN# pin must be driven inactive to stop accesses to the SMRAM area from filling in the cache line if SMRAM is not cacheable.

3. Upon leaving SMM, if SMRAM is cacheable, the CPU cache is emptied by asserting the FLUSH# pin within one CPU CLK after the SMIACT# pin is de-asserted.

It is the responsibility of the system logic to ensure that the setup and hold times for FLUSH# and SMIACT# signals are met.

2.3.2.3 External Write Buffers

Like the Intel486 SL processor, the SL Enhanced Intel486 CPU empties its internal write buffers before entering SMM to prevent data in the write buffers from being written to SMRAM space. If a system supports a second level cache, the second level write buffers must also be emptied before the CPU enters SMM. It is possible that the CPU is in SMM before the second level write buffers are completely emptied by the memory controller. In case the second level write buffer is not completely emptied, the SMIACT# signal can be used to direct the memory write cycles to either SMM space or memory space.

2.3.2.4 A20M# Pin

The A20M# pin on the Intel486 CPU is provided to emulate the address wraparound at the 1 Mbyte boundary which occurs on the 8086 microprocessor (see Figure 5). The SMRAM space on the Intel486 SL CPU is always below 1 Mbyte memory address space. Memory above 1 Mbyte can either be accessed through the ISAWINDOW register or the MCWINDOW register. The A20M# signal is automatically driven low whenever the CPU is in SMM. When A20M# is active, all external bus cycles will drive A20 low, and all internal cache accesses will be performed with A20 low.

The SL Enhanced Intel486 CPU does not provide any memory mapping mechanism to access memory above 1 Mbyte. To access memory above 1 Mbyte inside SMM, the software has to disable the A20M# manually through the keyboard controller. Also, if the SMRAM is located above 1 Mbyte and A20M# is not enabled before entering SMM, the system will crash. In this case, the CPU will attempt to access SMRAM at the relocated address with A20 low, and will thus fetch invalid code.

For these reasons, the A20M# should be driven inactive prior to entering SMM and remain inactive as long as the CPU is in SMM. This can be accomplished by blocking the assertion of A20M# whenever SMIACT# is active. The state of the A20M# should be saved upon entry to SMM and restored to its original state after leaving SMM.

2.4 SMM Environment Initialization

When the CPU is running in SMM, the processor is in a pseudo "real mode" environment, but without the 64 Kbyte limit. After the SMRAM base address register has been relocated, the CPU segment registers will have values shown in Table 2 when an SMI# occurs. *The CS selector register still contains the value 3000H, not the value corresponding to the new SMBASE. The rest of the registers are still initialized to zero.*

If the SMRAM base address has not been relocated, the segment registers can be initialized in the same way as with the Intel486 SL processor, i.e., using the CS register which defaults to 3000H. Otherwise, the segment registers must be initialized correctly to point to the new SMRAM memory space.

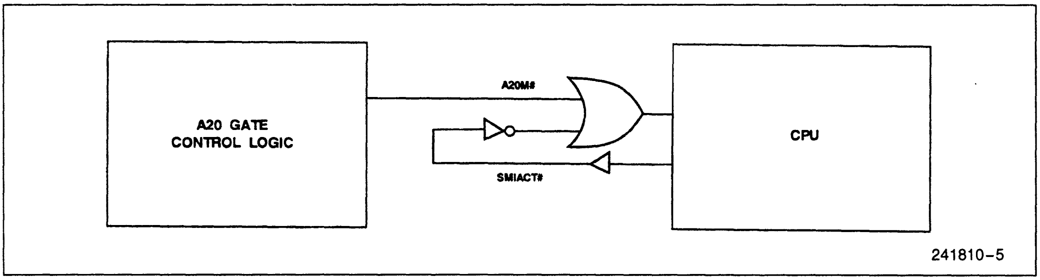


Figure 5. A20M# Interface Logic

Normally, the data segment registers are initialized to point to the SMRAM base address. Upon entering SMM, the CS BASE segment register is initialized to point to the SMRAM base address. The location of the SMRAM base address can be determined by reading the SMBASE register in the SMRAM at offset 0FEF8H *(the location of the SMRAM base address can also be stored in another memory location such as CMOS RAM by the BIOS which can be retrieved by the SMM program).*

The SMBASE contains a 32-bit address and has to be shifted to the right by four bits to generate a 16-bit segment address before it can be placed in the data segment selector registers. The CS selector register cannot be initialized by writing directly to it. It has to be initialized by executing a far jump instruction to an address within the SMRAM to force the CS selector register to point to the SMRAM base address.

When the CPU is in SMM, the operand size and the address size are still 16 bits but there are no limits to segment size. The physical address of an instruction is obtained by adding the value in CS segment base register to the value in EIP register, rather than the IP register. To access data anywhere within the four Gbyte logical address space, operand-size override (opcode 66H) and address-size override (opcode 67H) prefixes can be used as needed. Alternatively, SMRAM data located above 1 Mbyte can also be accessed by using 32-bit displacement registers.

Table 2. Register Values after SMI#

Segment Register	Selector	Base	Limit
CS	3000H	SMBASE	4 Gbytes
DS	0H	0H	4 Gbytes
ES	0H	0H	4 Gbytes
FS	0H	0H	4 Gbytes
GS	0H	0H	4 Gbytes
SS	0H	0H	4 Gbytes

3.0 POWER MANAGEMENT

One of the most important power management features on the Intel486 SL processor is CPU clock control. The clock control scheme on the SL Enhanced Intel486 CPU is similar to the Intel486 SL CPU, with the Intel486 SL CPU being driven by a 2X clock, and the SL Enhanced Intel486 CPU available with both the IX and 2X clocking options.

The 2X clock input is twice the internal frequency of the CPU, whereas the IX clock frequency is the same internal frequency of the CPU. With the IX clock, the two internal clock phases, "phase one" and "phase two", are generated by an internal Phase Lock Loop (PLL). The CPU clock input for a IX clock cannot be changed dynamically because the PLL requires a constant frequency CLK input (to within 0.1 %).

3.1 STPCLK# Interrupt

As with the Intel486 SL CPU, the SL Enhanced Intel486 CPU provides an interrupt mechanism, STPCLK#, which allows system hardware to control the power consumption of the CPU by stopping the internal clock to the CPU. Unlike the normal interrupts, INTR and NMI, the STPCLK# interrupt does not initiate interrupt acknowledge cycles or interrupt table reads.

The Stop Clock feature on the SL Enhanced Intel486 CPU has been improved, allowing the input to the STPCLK# to be driven asynchronously as well as synchronously. The major difference between asynchronous and synchronous control is that the STPCLK# interrupt latency is much shorter with asynchronous control.

With the Intel486 SL CPU, the STPCLK# input is controlled asynchronously through software. The STPCLK# is asserted after doing a dummy I/O read to the STPCLK register in the 82360SL or executing an HLT instruction. The STPCLK# signal will remain asserted until a system event wakes up the CPU. If the STPCLK# input is driven asynchronously, both setup and hold times t_{20} and t_{21} must be met for the STPCLK# interrupt request to be recognized.

After a STPCLK# interrupt request is recognized by the CPU, the processor will stop execution on the next instruction boundary, stop the pre-fetch unit, and then empty all internal pipelines and the write buffers. Finally, a special Stop Grant bus cycle is generated. The pin state during a Stop Grant cycle is shown in Table 3.

The interrupt acknowledge cycle is terminated when the system logic returns RDY# or BRDY#. At this point the CPU is in the Stop Grant state and the internal clock is stopped. The Stop Grant cycle is similar to the HALT cycle except that the address bus has the value 04H instead of 00H.

Table 3. Pin State During Stop Grant Cycle

Signals	State
M/IO#	0
D/C#	0
W/R#	1
Address Bus	0000 0010H ($A_4 = 1$)
BE3#-BE0#	1011 (same as HALT)
Data bus	Floated

Using the STPCLK# input, the SL Enhanced Intel486 CPU can be put into low power states similar to the Global Standby and Suspend states as with the Intel486 SL microprocessor.

3.2 Global Standby Implementation

In an Intel486 SL processor-based system, the 82360SL puts the CPU in a low power standby state (CPU I_{CC} ~ 20 mA–50 mA) when the system is in Global Standby. A similar state called **Stop Grant State** is provided by the SL Enhanced Intel486 CPU. The Stop Grant state can be entered by simply asserting the external STPCLK# interrupt pin. Once the STPCLK# interrupt is acknowledged by the CPU (i.e., after the Stop Grant cycle is placed on the bus), the CPU is in the Stop Grant state.

While in the Stop Grant state, the CPU still responds to RESET or SRESET and requests a cache invalidation (i.e., HOLD, AHOLD, BOFF# and EADS#). However, the CPU does not recognize any other inputs while in the Stop Grant state. Input signals to the CPU will not be recognized until one CPU clock cycle after STPCLK# is deasserted.

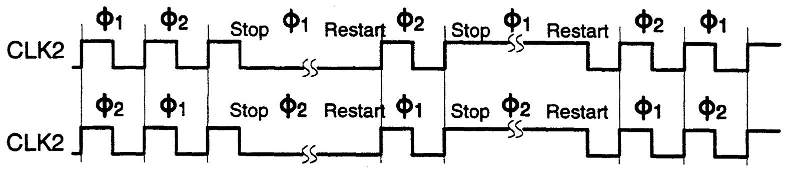
To emulate Global Standby, the stop clock control logic must be able to de-assert the STPCLK# signal whenever there is system activity (i.e., INTR, IRQ, NMI, and SMI#). Typically, the stop clock de-assertion logic is implemented by logic which latches the incoming interrupt requests from the system. The CPU returns to its normal state within 10–20 CPU clock cycles after exiting the Stop Grant state.

As mentioned before, the CPU does not recognize any interrupt request while the STPCLK# input is active. To prevent the interrupt request from getting lost, the interrupt request logic must ensure the interrupt signal is held active for at least one CPU clock after the STPCLK# input is de-asserted.

3.2.1 SUSPEND IMPLEMENTATION

From the Stop Grant state, the CPU can go into a lower power state similar to the suspend state offered in the Intel486 SL processor. After the CPU is in the Stop Grant state, the CPU can enter the lowest power **Stop Clock state (~100 μA–200 μA)** by stopping the CPU clock input. The CPU clock input can be driven to either logic high or logic low during Stop Clock state. The CPU will not generate any acknowledge cycle when entering stop clock state.

For a 2X clock input, the clock input to CLK2 can be stopped on either a logic high or logic low independent of the clock phase. The CPU will go back to Stop Grant state as soon as the CPU clock is re-started. Upon exit from Stop Clock State, the CPU clock input must be re-started in the same state when it was stopped (see Figure 6).



241810-6

Figure 6. CLK2 Phase Coherence in CLK2 Stop and Restart

For a CPU with a 1X clock input, the CPU clock can be stopped in the same manner as a CPU with a 2X clock input. Because of the phase lock loop, the CPU will not return to the Stop Grant state right after the CPU clock input has been re-started. To allow time for the PLL to stabilize, the CPU clock input must be held at a constant frequency for a period of time equal to the PLL startup latency (as specified in the data book) before the CPU will return to the Stop Grant state.

As long as the CPU clock input is stopped, the system logic must keep all the CPU input signals in the same state before the clock was stopped. Any change in state on an input signal (except for INTR) before the CPU has returned to the Stop Grant state will result in unpredictable behavior. The CPU will not be able to recognize any interrupt request while the CPU clock is stopped.

3.2.1.1 Dynamic Clock Switching

For a CPU clock with a 1X clock input, the CPU clock cannot be changed "on-the-fly". For power management as well as implementation of features such as deturbo mode, it is advantageous to run the CPU clock at a lower frequency. This can be accomplished by putting the CPU into Stop Clock state and change the CPU clock to a lower speed. After the CPU clock input frequency is changed, the clock control logic must ensure that the clock input has been running at a constant frequency for the time period necessary for the PLL to stabilize before de-asserting the STPCLK# signal. The lowest CPU clock rate for a 1X part is 8 MHz (see Figure 7).

3.2.1.2 Power Consumption

The Stop Grant and Stop Clock states are designed to save power. While the processor is in Stop Grant state, the input/output signals on the CPU remain at the same state when entering the Stop Grant state, floated (data and parity signals), or driven to a different state. If some of the signals are driven improperly, the system can end up consuming more power.

To achieve the lowest power consumption, all the possible current leakage must be eliminated. The system logic should never drive the input signals with pull-up resistors LOW and input signals with pull-down resistors HIGH. 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 CPU entered the Stop Grant state. For minimum CPU power consumption, all other input pins should be driven to their inactive level while the CPU is in the Stop Grant state.

3.2.2 GENERAL DESIGN CONSIDERATIONS

The STPCLK# input is an asynchronous signal. The system can crash if the interface to the STPCLK# is not designed properly. Special care must be taken to ensure that all the timing requirements are met and the proper protocol is used. Listed below are some design considerations that should be considered when designing the STPCLK# interface.

- The CPU cannot empty the write buffer during an HLDA cycle. Therefore, the CPU will not acknowledge any STPCLK# request during an HLDA cycle.

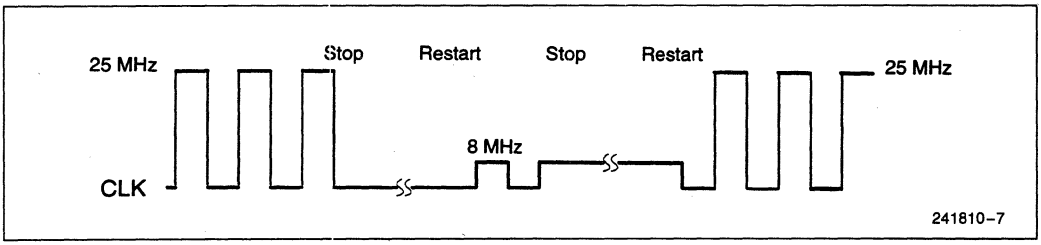


Figure 7. Clock Switching on IX Clock Input

- After the STPCLK# is asserted, the CPU does not generate a Stop Grant cycle until it completes the current instruction. The latency between a STPCLK# request and the Stop Grant bus cycle depends on the current instruction, the amount of data in the CPU write buffers, and the system memory performance.
- The CPU will not enter the Stop Grant state until either RDY# or BRDY# has been returned.
- In response to HOLD being driven active during the Stop Grant state (when the CLK input is running), the CPU will generate HLDA and three-state all output and input/output signals that are three-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.
- When the CPU enters the Stop Grant state, the internal pull-up resistor is disabled so that the CPU power consumption is reduced. The STPCLK# input must be driven high (not floated) in order to exit the Stop Grant state.
- It is the responsibility of the system designer to ensure that the CPU is in the correct state prior to asserting cache invalidation or interrupt signals to the CPU.

4.0 RESET IMPLEMENTATION

On a standard PC, the CPU can be reset by either hardware or software. On the SL Enhanced Intel486 CPU, asserting the RESET input to the CPU will also set the SMBASE register to the default value of 30000H. In other words, the SMRAM base address will reset to 30000H whenever the operating system asserts the CPU RESET signal. For some older software, a CPU RESET is generated by the software to return the CPU to real mode from protected mode.

Normally, this is not a problem if SMBASE relocation is not used. If the SMRAM base address has been relocated, the CPU could be executing invalid SMM code from the address. The SRESET pin has the same functions as RESET except that it does not reset the SMBASE register.

For a system which uses SMBASE relocation, the logic which generates the software CPU RESET must be tied to the SRESET input and not the RESET input on the CPU. All hardware resets should be implemented through the RESET pin (see Figure 8), and all software resets should be implemented through the SRESET pin.

While inside SMM, the CPU should be protected from being reset by a software CPU RESET. SRESET should be blocked whenever the SMIACT# is active. Any request for a CPU RESET when the CPU is in SMM should be latched so it can be serviced after the CPU exits SMM. To ensure the execution of the RSM instruction does not get interrupted by the SRESET, the SRESET must be blocked until at least 20 CPU clock cycles after SMIACT# has been driven inactive.

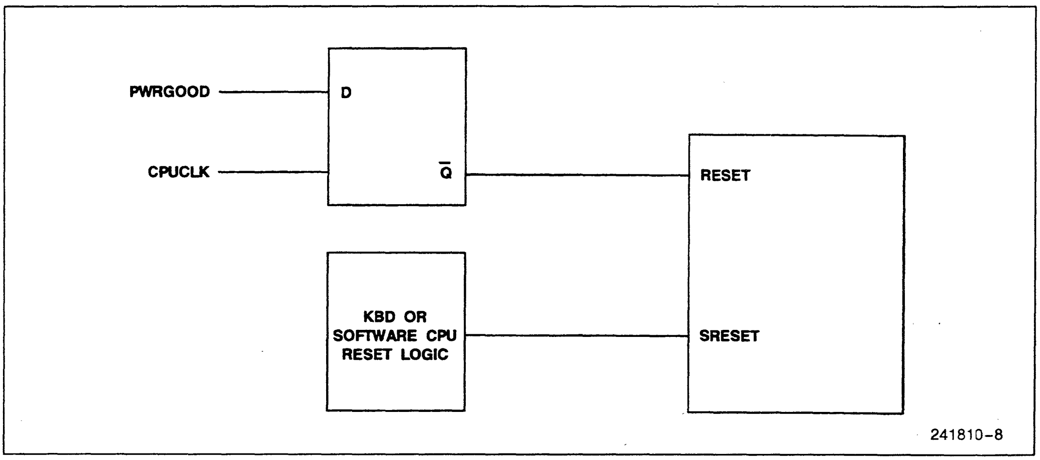


Figure 8. SRESET Interface Logic

4.1 General Design Considerations

The system designer should consider the following restrictions while implementing the CPU Reset logic:

- For SRESET to be recognized by the CPU, the SRESET input must be driven active (high) for at least 15 CPU clock cycles.
- The SRESET is not intended to be used for flushing the on-chip cache. For compatibility with future generation Intel CPUs, the SRESET input pin should not be used for flushing the on-chip cache.
- Hardware resets should not be blocked when the CPU is in SMM so that the system can recover from a fatal system failure.

CONCLUSION

While taking advantage of the benefits of the SL Architecture, the SL Enhanced Intel486 CPU family provides a whole new world of opportunity for system designers to develop innovative, energy-efficient mobile and desktop designs. The SL Enhanced Intel486 microprocessor family combines power management, compatibility and performance, allowing system designers to build a wide variety of machines to meet the diverse needs of a broad range of users.

References

Intel's SL Architecture: Designing Portable Applications, 1993, McGraw-Hill.

Intel486 Microprocessor Family Data Book Addendum: SL Enhanced Intel486 Microprocessor Family, 1993, Intel Corporation.

intel®

AP-497

APPLICATION NOTE

Managing Power with the SL Enhanced Intel486™ Microprocessor

2

CHENG XIE
MCG TECHNICAL MARKETING

October 1993

Managing Power with the SL Enhanced Intel486™ Microprocessor

CONTENTS PAGE

1.0 INTRODUCTION

Intel's System Management Mode (SMM), introduced as part of the Intel SL technology, has become an industry standard for portable computing. Through the utilization of SMM, system designers have a new method of adding software controlled features that operate transparently to the operating system and applications software. In portable computer systems, SMM is often used to implement the power control on various system components to conserve power consumption. Flexible clock control has also become essential to the design of power-saving computers. The new SL Enhanced Intel486™ microprocessor family incorporates all of the best power management features which first appeared in Intel SL technology, bringing Intel486 CPU performance to portable computers and energy-efficient desktop systems. The purpose of this application note is to provide system designers with a better understanding of the theory and implementation of power management with the new SL Enhanced Intel486 CPUs.

2.0 POWER MANAGEMENT FEATURES

- **System Management Mode**—This mode is composed of a special purpose interrupt that serves as the hardware interface and a secure memory address space that stores processor status and special software routines. It can be used to implement power management for portable systems.
- **Flexible Clocking Options**—The clock input to the CPU can either be a 1X clock or a 2X clock. For 1X clock option, the internal clock of the CPU runs at the same speed as the input clock (CLK input). In the 2X clock case, the clock input (CLK2 input) needs to be twice the frequency of the internal clock.
- **Different Low Power States**—Different low power processor states are available for various operating conditions. This feature enables the conservation of processor power consumption without sacrificing performance.
- **Low Voltage Power Supply Option**—The SL Enhanced Intel486 CPUs can be powered by either a 5V or a 3.3V supply, with the 3.3V supply providing a 50% power savings.

2.1 System Management Mode

The System Management Interrupt (SMI#) input pin on the processor provides the hardware interface for the computer system to invoke SMM. An exclusive memory address space, SMRAM, is only available for the CPU to access while in SMM. The size of SMRAM can be between 32 Kbytes and 4 Gbytes. It is used to store processor state and SMI handlers. SMI handlers are special software routines that can be designed to control the power states of system components. Intel's SMM has a special instruction, RSM, that is responsible for exiting SMM. When executed, RSM instruction restores the processor state from SMRAM and returns control to the application that was interrupted by SMI#.

The servicing of SMI# is different from that of a regular interrupt. The system invokes SMM by generating an SMI# to the SL Enhanced Intel486 CPU. Normal program execution will be interrupted as a result and the CPU will respond to the interrupt by asserting SMIACT#. This signal is used by the system to enable SMRAM space. The CPU will then save its state into the SMRAM area, starting at address location 3FFFFH and proceeding downward in a stack-like fashion. After completing a state save, the CPU will be in a pseudo real-mode processor environment. The microcode will then direct the CPU to begin executing instructions at the absolute address of 38000H (SMRAM), at which point it will begin executing SMI handlers. SMI handlers can perform various system management functions, including system power control. The last instruction in an SMI handler is always the RSM instruction. This instruction will restore the CPU state from SMRAM, de-assert SMIACT# and return control of the system to the interrupted program.

The generation of an SMI# to the CPU can be initiated by hardware or software for the purpose of power management. The actual implementation depends on the specific chipset used and how the system is designed. Hardware can generate SMI# by pulling the SMI# pin low directly or through other chipset pins, i.e., battery level control. When the charging level of the main battery falls below a certain limit, the chipset monitoring the battery level can interrupt the normal program operation by pulling the SMI# pin low. While in SMM, the CPU can execute certain power-down SMI handlers to put the entire computer system in a low-power mode that can operate out of a different battery source. This will enable the system to maintain its current status while allowing the main battery to be changed. SMI# can also be initiated through software. Different chipsets have various ways of interfacing to the CPU in this aspect. Most of them have dedicated timers to detect an idled device. Once these timers are enabled, the timeout will automatically generate an SMI#.

Exiting SMM is accomplished by the execution of the RSM instruction. Besides restoring the CPU state, the RSM instruction can also perform three other functions. The first is called "Auto HALT Restart". The System Management Interrupt request can interrupt the HALT instruction. By setting the appropriate control slot in the SMRAM space, the RSM instruction can return control to the HALT instruction or the instruction immediately following the HALT instruction. The second special feature is "I/O Trap Restart". If SMI# interrupt is generated on an I/O access, the RSM instruction will re-execute that I/O instruction if its I/O Trap Restart slot in the SMRAM is set. The third function is the relocation of SMBASE, the starting address of SMRAM. This provides system designers with the flexibility of placing SMRAM space into an area in which its integrity is best ensured. The starting address is controlled by the SMBASE register, which has a power-on reset value of 30000H. The default SMRAM area starts at 38000H and ends at 3FFFFH. The SMRAM can be relocated to any 32K aligned area, either overlaid on top of the normal system address space or placed in a distinct address space.

2.2 Flexible Clock Control Options

The standard Intel486 SX and Intel486 DX CPUs are driven by 1X clock as opposed to the Intel386 CPUs which use a 2X clock input. The SL Enhanced Intel486 CPUs are available with either the 1X or the 2X clocking options.

The 1X clock allows simpler system design by cutting in half the clock speed required in the external system. The 1X clock relies on an internal Phase Lock Loop to generate the two internal clock phases, "phase one" (ph1) and "phase two" (ph2). The rising edge of the CLK input corresponds to the start of ph1. All external timings are specified with respect to the rising edge of the CLK input. The PLL requires a constant frequency CLK input (to within 0.1%), and therefore the CLK input cannot be changed dynamically.

The 1X CLK input option is essential for those processors with an on-chip clock doubler. The 1X CLK input provides the fundamental timing references for the bus interface unit. The CLK input is doubled internally so that the CPU core will operate at twice the CLK input frequency, and hence twice the bus frequency. The internal clock doubler enhances all operations using the internal cache and/or not blocked by external bus accesses. This mode also uses PLL and therefore the CPU CLK input must be maintained at a constant frequency.

The SL Enhanced Intel486 CPUs also offer a 2X clock option for systems that rely on dynamic frequency scaling for CPU power management. The frequency of the CLK2 input is twice the internal frequency of the CPU. The internal clock is comprised of two phases, "PH1" and "PH2". Each CLK2 period is a phase of the internal clock. All timings are referenced to the rising edge of the PH1 of the CLK2 input. It is therefore important to synchronize the external circuitry with the PH1 of the CLK2 input. Because the 2X clock option does not rely on the PLL to generate the internal phase clocks, the frequency of the CLK2 input can be changed dynamically or "on-the-fly".

2.3 Different Low Power States

2.3.1 POWER STATES IN 1X MODE

Several low power modes are available on the 1X microprocessors. These modes make it possible for a power-sensitive computer system to optimize power conservation. Some of the CPU power controls are realized through a specially provided interrupt mechanism. Each of the following low power states is described in detail.

- **Auto Idle Power Down State**

  This low power state is available in normal operation for the DX2 CPUs. DX2 CPUs have an internal clock doubler which doubles the 1X clock input and therefore enables the CPU core to operate at twice the speed of the input clock. When the SL Enhanced Intel486 CPU is known to be truly idle and waiting for a ready from a memory or an I/O bus cycle read or write, the DX2 CPU will reduce its core clock rate to 1X from the doubled DX2 clock. In this state, the CPU only consumes half of the normal power. More importantly, this function is transparent to software and external hardware and therefore does not cause any performance degradation.

- **Stop Grant State**

  The Stop Grant state is initiated by simply asserting the external STPCLK# interrupt pin. The Stop Grant state is used to transition to the Stop Clock state.

  The CPU enters the Stop Grant state through the following steps: When the CPU recognizes the STPCLK# request, it will stop the execution of the normal program on the next instruction boundary, stop the pre-fetch unit, empty all internal pipelines and write buffers, generate a Stop Grant bus cycle and then stop the internal clock.

  This state is exited when STPCLK# pin is pulled high. The rising edge of the STPCLK# will tell the CPU to return control to the interrupted program and start to execute the instruction following the last executed instruction of the interrupted program.

- **Stop Clock State**

  The Stop Clock state is entered from the Stop Grant state by completely stopping the clock input to the PLL. In this state, the CPU consumes only ~100 μA–200 μA of current.

- **Auto HALT Power Down State**

  When the HALT instruction is executed, the CPU will issue a normal HALT bus cycle. The SL Enhanced Intel486 CPU will automatically stop the internal CPU clock, therefore causing the CPU to enter a low power state with a current of ~20 mA–55 mA.

- **Stop Clock Snoop State**

  Cache snooping is necessary during Stop Grant and Auto HALT Power Down states in order to maintain memory coherency. When the system issues a request for cache snooping, the CPU will transparently enter the Stop Clock Snoop state and will power up for 1 full core clock to complete the cache snoop cycle. It will then re-freeze the clock to the CPU core and either return to the Stop Grant or the Auto HALT Power Down state.

2.3.2 Transition of Power States and Latency Associated with 1X Mode

It is important to understand how the different power states are interrelated and how one transitions to another. The latency is different for different state transitions. Figure 1 depicts which transitions are allowed. We shall describe how the transitions are made and how much latency is associated with each transition.

1. The Auto Idle Power Down state is entered whenever the CPU is detected idle and waiting for a RDY# from either a memory or I/O read/write. This state only applies to SL Enhanced Intel486 DX2 CPUs. Both the internal CPU core clock and the current drop to half of the doubled frequency. There is no latency associated with this transition. The CPU will go back to normal operation when a RDY# is detected.
2. The Stop Grant state is entered when the STPCLK# interrupt is asserted to the CPU by the system. In this state, the clock output of PLL (or the clock input to the internal core) is stopped. The speed of the clock input to the PLL can be maintained or changed. If the clock frequency is changed, the CPU requires the clock to be held at a constant frequency for a minimum of 1 ms before de-asserting STPCLK#. The 1 ms time period is necessary so that the PLL can stabilize the input clock period. De-asserting STPCLK# returns the CPU to normal operation. The CPU will also return to its normal state when RESET or SRESET is asserted.
3. The Stop Clock state can only be entered from Stop Grant state when the clock input to the PLL is stopped. The CPU must go through Stop Grant state to return to normal operation. Before the CPU can return to Stop Grant state, the clock input has to stabilize for the 1 ms required by the PLL.
4. The Auto HALT Power Down state is entered when HALT instruction is executed. The clock input to the internal CPU core is automatically stopped upon the execution of HALT instruction. The clock input to the PLL cannot be changed during this state. This state is exited back to normal operation whenever any of the following events happen: INTR, NMI, SMI#, RESET or SRESET. There is no latency associated with this state transition.

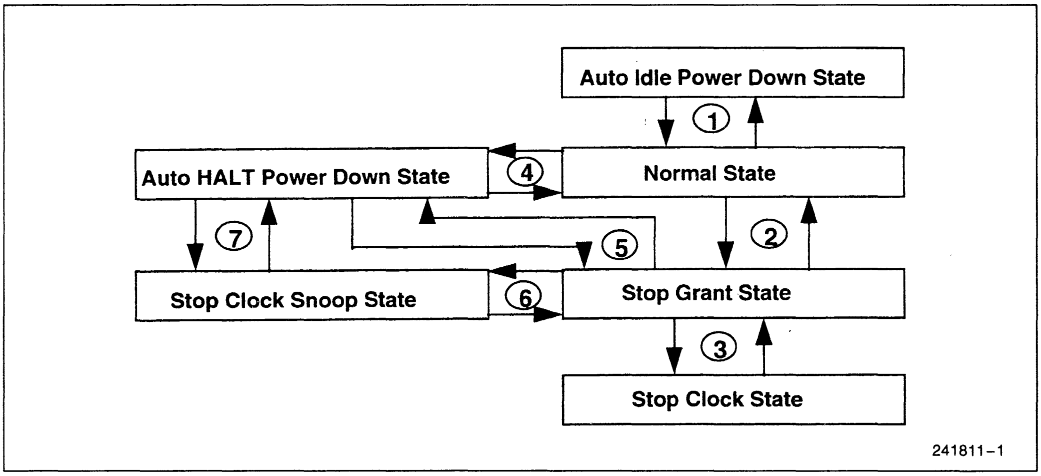


Figure 1. Power State Transitions for 1X CPUs

5. When the CPU is in Auto HALT Power Down state, the system can generate STPCLK# to the CPU to bring the CPU into a Stop Grant State. When the system de-asserts the STPCLK# request, the CPU will return to the Auto HALT Power Down state. There is no latency associated with this transition. HALT bus cycles will be launched whenever this transition occurs.

6, 7. Cache snooping can be performed when the CPU is either in Stop Grant state or in Auto HALT Power Down state. Cache snoop cycles begin when the CPU receives an EADS# from the system. The CPU will only wake up for 1 complete core clock to perform cache invalidation and then re-freeze the clock, i.e., either return to the Auto HALT Power Down state or to the Stop Grant state.

2.3.3 POWER STATES IN 2X MODE

There are five operating modes for the 2X SL Enhanced Intel486 CPU. The CPU power controls are realized through a specially provided interrupt mechanism, STPCLK#. In the following, we shall describe each of the power states in detail.

- **Normal State**

  2X CPUs do not have a PLL, and therefore do not require a stabilized clock period. This means the frequency of the input clock (CLK2) can be changed dynamically. Depending on the level of activity, CPUs do not always have to operate at full speed. Reducing the CPU speed saves power.

- **Stop Grant State**

  The Stop Grant state is initiated by simply asserting the external STPCLK# interrupt pin. The Stop Grant state is used to transition to the Stop Clock state.

  The CPU enters the Stop Grant state through the following steps: when the CPU recognizes the STPCLK# request, it will stop the execution of the normal program on the next instruction boundary, stop the pre-fetch unit, empty all internal pipelines and write buffers, generate a Stop Grant bus cycle and then stop the internal clock.

  This state is exited when STPCLK# pin is pulled high. The rising edge of the STPCLK# will tell the CPU to return control to the interrupted program and start to execute the instruction following the interrupted instruction.

- **Stop Clock State**

  The Stop Clock state is entered from the Stop Grant state by completely stopping the clock input (CLK2). In this state, the CPU consumes only ~100 μA–200 μA of current.

- **HALT State**

  When the HALT instruction is executed, the CPU will issue a normal HALT bus cycle. For 2X CPUs, there is no power savings in this state.

- **Stop Clock Snoop State**

  Cache snooping is necessary during Stop Grant state in order to maintain memory coherency. When the system issues a request for cache snooping, the CPU will transparently enter the Stop Clock Snoop state and will power up for 1 full core clock to complete the cache snoop cycle.

2.3.4 TRANSITION OF POWER STATES AND LATENCY ASSOCIATED WITH 2X MODE

Figure 2 shows the state transitions of a 2X microprocessor. We shall describe how the transitions are made and how much latency is associated with each transition.

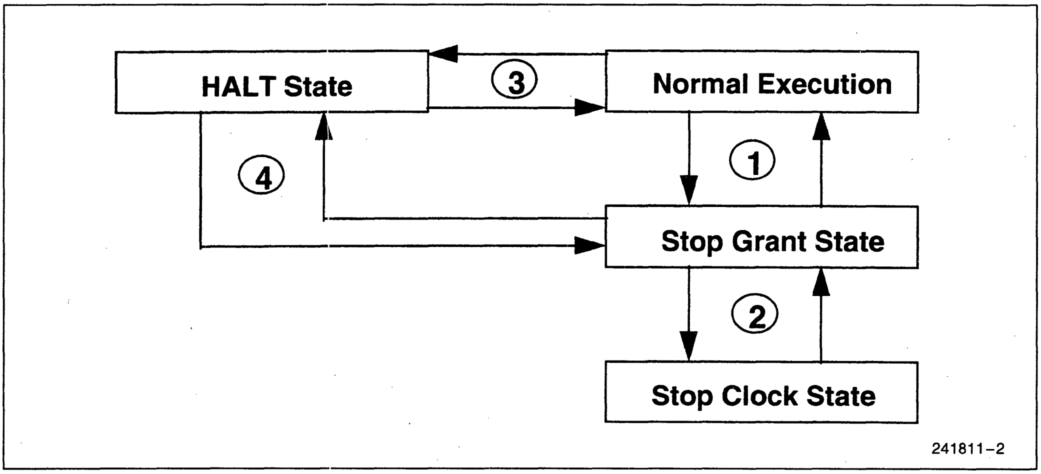


Figure 2. Power State Transitions for 2X CPUs

1. The Stop Grant state is entered when the STPCLK# interrupt is asserted to the CPU by the system. In this state, the speed of the external clock input (CLK2) can be maintained or changed. There is no latency associated with changing the CLK2 frequencies. De-asserting STPCLK# returns the CPU to normal operation. The CPU will also return to its normal state when RESET or SRESET is asserted.
2. The Stop Clock state is entered from Stop Grant state when the clock input (CLK2) is stopped. The CPU must go through Stop Grant state to return to normal operation. There is no additional delay associated with returning to normal operation.
3. This state is entered when the HALT instruction is executed. The HALT state consumes the same power as the normal state. The HALT state is exited back to normal operation whenever any of the following events happen: INTR, NMI, SMI#, RESET or SRESET. There is no latency associated with this state transition.
4. When the CPU is in HALT state, the system can generate STPCLK# to the CPU to bring the CPU into Stop Grant state. When the system de-asserts the STPCLK# request, the CPU will return to the HALT state. There is no latency associated with this transition. HALT bus cycles will be launched whenever this transition occurs.

3.0 IMPLEMENTATION OF POWER MANAGEMENT FEATURES

The means of implementing power management features depend on the specific chipset used. Most of the chipsets have both hardware and software power management. There are always a number of dedicated or user-definable timers that monitor the activity of certain device(s), such as CPU or peripheral components. In a software approach, the timeout of these timers can trigger a System Management Interrupt. Upon the detection of an SMI, the CPU will execute the power management SMI handlers in the BIOS which exercise device power control through detailed programming. For a hardware-based approach, the timers can automatically be enabled to perform power controls. If the status of the specific device or the entire system needs to be saved before changing its power state, the software approach must be used. In other words, if the original status of a device or the entire system is required to return the system to its operational state before the power state change, SMM must be invoked and power control will be accomplished by the SMI handlers. This section summarizes several power control schemes that are common conceptually to all major chipsets and explains how they interact with the power management features offered by the SL Enhanced Intel486 CPUs.

3.1 CPU Power Control

Controlling SL Enhanced Intel486 CPU power is achieved by reducing the CPU clock speed, stopping the CPU clock or shutting off the CPU power.

All chipsets have the option of pre-programming the CPU speed regardless of the level of system activity there is and the CPU clock speed can be divided down by 2 to 64. Once the CPU is selected to run in a reduced speed mode and the CPU clock division is selected, the CPU will always run at a divided speed until the CPU is switched into some other mode of operation. Speed reduction is done by BIOS through programming certain register bits immediately after booting.

The speed of the CPU can also be changed dynamically depending on the level of system activity. Most of the chipsets provide mechanisms of monitoring the level of system activity involving the CPU by detecting the toggling of certain CPU signal lines. The timers associated with these monitoring devices are responsible for determining when to reduce the CPU speed by timing out a programmable time period. The chipset reduces the speed of the CPU clock by dividing its clock input to the CPU after STPCLK# is asserted. For 2X SL Enhanced Intel486 CPUs, the speed can be changed on-the-fly. For 1X SL Enhanced Intel486 CPUs, clock input has to be stabilized for 1 ms before de-asserting STPCLK#. Stopping the CPU clock is accomplished in the same fashion.

The CPU clock control can be achieved either through hardware or software approach. If the status of the CPU needs to be saved in order to return the system to the original state, CPU clock control should be done through the SMI handlers in SMM mode.

Shutting off the power to CPU can only be done in suspend mode.

3.2 Controlling Power of Peripheral Components

The power control of peripheral components is accomplished through idle detectors and SMI generators.

Idle detectors monitor the access to the following devices: Keyboard, Video RAM, Floppy Disk Drive, Hard Disk Drive, Serial Port and Parallel Port. Idle detectors can also monitor access to a programmable address range. Some chipsets even provide additional pins to monitor other user-definable miscellaneous activities. There are timers associated with each of the idle detectors with programmable idle time period and the timers activate power control pins that are directly tied to the controlled devices. When a timer times out the programmed period, it will activate the power control pin to shut down the power to the specific device. For address range idle detectors, when there is no access to the monitored address range for a programmed timeout period, the timers will either reduce the clock speed or shut off the power of the device that has the same address bits. The idle detectors can operate outside of SMM mode and can be independent of CPU state.

SMI event generators generate SMI requests through a number of dedicated or user-defined events. Depending on the specific chipset used, these events can be the activity timeout of individual devices or a collection of devices. Upon the timeout, an SMI request will be generated to the CPU, which in turn invokes SMM. The SMI handlers in the SMRAM contain the software routine that controls the power state of the device(s) initiating the SMI request. By executing this routine, the CPU will access the power management control registers associated with the devices. After proper programming, these registers will activate the proper power control pins to shut down the power to the proper device(s). Controlling the power of peripheral devices through SMI handlers offers complete flexibility to either manage the power individually or collectively. This is very important for optimum power conservation.

3.3 Suspend and Resume

Suspend state is the deepest level of power conservation. There are two types of suspend: normal suspend and 0-volt suspend. In normal suspend, only the CPU, chipset and memory sub-system are powered. System status is saved into SMRAM. The rest of the system is shut down. DRAM is refreshed with a very slow clock (64 KHz or 32 KHz). In 0V suspend, the entire system, including the CPU, is shut down except the part of the system logic that is responsible for resume. The resume logic is always powered by an RTC battery. The system status is saved onto hard disk. Suspend is normally triggered by the suspend timer, and the timeout of the suspend timer is also programmable.

Because of the amount of BIOS support required by Suspend, SMM must be invoked. Hardware alone cannot accomplish the task.

4.0 SUMMARY

SL Enhanced Intel486 CPUs provided the best power management features that Intel SL technology offers. Intel's System Management Mode has become an industry standard for power-saving computing. Through Intel's SMM, the implementation of power management is very flexible, enabling the optimization of power conservation for different system designs. The various CPU clock control options available on SL Enhanced Intel486 processors provide the basis for runtime power management with no performance impact. All major chipsets support the Intel power management scheme with easy-to-design software and hardware interfaces.

intel®

AP-498

APPLICATION NOTE

2

Thermal Design for High Performance Notebooks

VLADIMIR ALEKSIEV
CHIA-PIN CHIU
WENDY LUI
ED WILSON
DAVID YUAN
MCG TECHNICAL MARKETING
MCG PACKAGING
THERMAL/MECHANICAL TOOLS AND ANALYSIS GROUP

October 1993

Thermal Design for High Performance Notebooks

CONTENTS PAGE

1.0 INTRODUCTION

Today's market for notebook computers demands desktop performance in smaller and smaller form factors. Along with the higher performance comes greater power consumption, which adds unique challenges for the mobile operating environment (battery operating time, thermal management, physical dimensions, etc.). Included in this application note is a basic description of the thermal forces at work in mobile applications, with mathematical models that can be used to project system thermal parameters and aid the designer in worst-case design.

The first section begins with a review of the basic thermal definitions which apply to notebook designs. Thermal data from a notebook experiment is presented to show a relationship between the temperature outside the notebook and the CPU case temperature. A model is given that helps the designer ensure the CPU thermal operating specifications are met.

Next, several power consumption profiles are provided, starting with the assumed worst-case model, along with more conservative power profiles based on the degree of power management implementation. Based on these models, designers may forecast the amount of additional thermal margin their applications need.

With these thermal models, power consumption profiles, and test measurements, the designers will have the necessary tools and techniques to design for the worst case, and understand that by applying simple design enhancements, they can improve the quality of their designs. The designer should ensure the measured CPU case temperature (T_{CASE}) complies with the T_{CASE} specifications published in the SL Enhanced Intel486 Microprocessor Data Sheet Addendum.

2.0 THERMAL BACKGROUND

2.1 Heat Transfer

Designing high performance CPU notebook systems requires some knowledge of the three processes by which heat is transferred from one point to another, namely: conduction, convection, and radiation, which are described in the following three sections. This knowledge will help the designer understand the subsequent methods of heat transfer and their value in maintaining the CPU within its specified T_{CASE} limits. The formulas describing heat transfer by conduction, convection and radiation can be shown analogous to Ohm's Law:

$$I = \frac{V}{R}.$$

The thermal current, temperature difference and thermal resistance are analogous to electrical current (I), voltage (V), and electrical resistance (R), respectively.

2.1.1 CONDUCTION

Conduction is a process by which heat flows from a region of higher temperature to one of lower temperature within a medium (solid, liquid, or gas) or between mediums in direct physical contact[1]. In a one-dimensional system, conductive heat transfer is governed by the following relation:

Conductive Heat Transfer		Ohm's Law
$q = \frac{\Delta T}{L/kA}$	$\rightarrow$	$I = \frac{V}{R}$

where:

q = Heat flow rate (W)

k = Material thermal conductivity (W/m°C)

A = Cross-sectional area (m^2)

$\frac{\Delta T}{L}$ = Temperature gradient (°C/m)

L = Distance of heat transfer

In the preceding equations, the thermal current, q, can be viewed analogous to electrical current; ΔT analogous to voltage; and L/kA analogous to thermal resistance. To improve thermal conduction in a notebook environment, copper and other highly-thermal conductive metals can be used in the package design.

2.1.2 CONVECTION

Convection is a process of energy transport by the combined action of heat conduction, energy storage, and mixing motion[2]. Convection is the predominant mechanism for transferring energy between a solid surface and a fluid. In the notebook environment, this is equivalent to the heat transfer between the case surface and the ambient environment (air). The basic relation that describes heat transfer by convection from a surface to a fluid presumes a linear dependence on the difference between the temperature at the surface and deep in the fluid, and is referred to as Newtonian cooling:

Convective Heat Transfer		Ohm's Law
$q_C = \frac{T_S - T_A}{1/h_C A}$	$\rightarrow$	$I = \frac{V}{R}$

NOTES:

[1] *Physics, Second Edition,* Paul A. Tiper. Worth Publishers, Inc., 1982, p. 531.

[2] *Physics, Second Edition,* Paul A. Tiper. Worth Publishers, Inc., 1982, p. 531.

where:

q_C = Convective heat flow rate from a surface to ambient (W)

A = Surface area (m^2)

T_S = Surface temperature (°C)

T_A = Ambient temperature (°C)

h_C = Average convective heat transfer coefficient ($W/m^2 °C$)

In the preceding equations, the thermal current, q_C, can be viewed analogous to electrical current; $T_S - T_A$ analogous to voltage; and $1/h_C A$ analogous to thermal resistance.

In forced convection, fluid flow is caused by an external factor such as a fan, while in free or natural convection, fluid motion is induced by density differences resulting from temperature gradients in the fluid (liquid or gas). Under the influence of gravity or other body forces, these density differences give rise to buoyancy forces that circulate the affected fluid and convect heat toward or away from surfaces wetted by the fluid. Although fans are often used to increase air convection inside desktop computers, they may not be a practical solution for notebook systems.

2.1.3 RADIATION

Thermal radiation is defined as radiant energy emitted by a medium by virtue of its temperature, without the aid of any intervening medium[3]. The amount of heat transferred by radiation between two bodies at temperatures T_1 and T_2 is governed by the following expression:

Radiative Heat Transfer		Ohm's Law
$q = \frac{T_1^4 - T_2^4}{1/\epsilon\sigma}$	→	$I = \frac{V}{R}$

where:

q = Amount of heat transferred by radiation (W)

ϵ = Emissivity $0 < \epsilon < 1$

σ = Stefan-Boltzmann constant, 5.67×10^{-8} ($W/m^2 °K^4$)

A = Area (m^2)

T_1, T_2 = Surface temperature (°K)

For radiation to be an effective method of heat transfer, compared to natural or forced convection mechanisms, a relatively large temperature difference must exist between T_1 and T_2. For most low-power electronic applications, these temperature differences are relatively small, and therefore, radiative effects are normally neglected. However, for high-power applications, heat transfer by radiation factors should be considered. Although heat radiation is a secondary thermal effect in a notebook system, some manufacturers are selecting coatings (i.e., black paint) for their notebook designs that are absorptive in the infrared to improve upon these thermal radiation effects.

2.2 Thermal Impedance

Thermal management of an electronic system encompasses all the thermal processes and technologies which must be used to remove and transport heat from individual components to the system thermal sink in a controlled manner. The primary heat transfer processes (conduction, convection and radiation) can be combined into a single linear model (see Section 5.1).

The junction-to-case (θ_{JC}) and junction-to-ambient (θ_{JA}) thermal resistance values are used as measures of IC package thermal performance. These parameters are defined by the following relations:

$$\theta_{JC} = \frac{T_J - T_C}{P}$$

$$\theta_{JA} = \frac{T_J - T_A}{P}$$

$$\theta_{JA} = \theta_{JC} + \theta_{CA}$$

where:

θ_{JA} = Junction-to-ambient thermal resistance (°C/W)

θ_{JC} = Junction-to-case thermal resistance (°C/W)

θ_{CA} = Case-to-ambient thermal resistance (°C/W)

T_J = Average die temperature (°C)

T_C = Case temperature at a predefined location (°C)

P = Device power dissipation (W)

T_A = Ambient temperature (°C)

θ_{JC} is a measure of package internal thermal resistance from silicon die to package exterior. This value is highly dependent upon packaging material, thermal conductivity, and package geometry. θ_{JA} measures the

NOTE:

[3] *Physics, Second Edition,* Paul A. Tiper. Worth Publishers, Inc., 1982, p. 535.

conductivity and convective thermal resistance from package exterior to the ambient, as well as package internal thermal resistance. θ_{JA} values depend on material, thermal conductivity, package geometry, and ambient conditions such as flow rates and coolant physical properties.

To improve CPU thermal characteristics in a notebook system, heat sinks are sometimes mounted on the top of the CPU using highly conductive adhesive materials. Adding a heat sink will not change θ_{JC}; however, it will improve heat conduction and convection due to the increase in surface area, resulting in a significant reduction in the case-to-ambient and, therefore, junction-to-ambient thermal resistance. Depending on the product and materials used, θ_{JA} can be reduced by 10 to 30 percent.

To guarantee component functionality and reliability, the maximum device operating temperature is defined and constrained by the package exterior temperature at a predefined location. The guidelines for ambient temperature specify that measurements should be taken at an undisturbed location at a certain distance away from the package—traditionally 12 inches horizontally from the center of the CPU. Measuring T_A in the traditional manner is not possible in a notebook system. The case temperature, however, is measured at the center surface of the package. Depending on the ambient temperature and board power in the system's environment, thermal enhancements such as heat fins or forced air cooling may be necessary to meet the case temperature requirements.

3.0 POWER MODELING

3.1 Power Consumption Model

In a linear model, power consumption is governed by the following equation:

$$P = V_{CC} \times I_{CC}$$

where:

P = Power consumed by the component
V_{CC} = Supply voltage
I_{CC} = Current through the component

The preceding equation indicates that power consumption is linearly proportional with both supply voltage and current flowing through the component. For example, a 3.3V microprocessor will consume less power than a 5V microprocessor running the same application under equivalent operating conditions.

A system's total power consumption is defined as either the sum of the power consumed by each individual module within the system, or the sum of the products of each module's voltage supply and its current. It is equivalent to:

$$P_{system} = P_{CPU} + P_{memory} + P_{display} + \text{etc}$$

3.2 Empirical Data

Several CPU I_{CC} measurements were taken using an SL Enhanced Intel486 CPU evaluation board with various CPU modules running under two different operating environments: Windows 3.1 and Indeo™ Video software (see Table 3-1). The modules used were SL Enhanced Intel486 DX-33 CPUs (1X SQFP, 1X PGA, and 2X SQFP), and SL Enhanced Intel486 DX2-50 CPUs (1X PGA). All of the CPU modules use a chipset with Power Management software.

When comparing the I_{CC} measurements between several application environments, the CPUs running Indeo Video software consume the most I_{CC} current by approximately 10%. Since the I_{CC} value represents the number of gates switching inside the CPU, and hence, the intensity of the CPU working condition, it is therefore concluded that running Indeo Video software will be close to the worst case for thermal measurement purposes.

Table 3-1. Case Analysis of Power Consumption for SL Enhanced Intel486 CPUs

							Windows 3.1			Indeo™ Video Software		
							I_{CC}			I_{CC}		
CLK	**CPU**	**V_{CC}**	**Freq**	**Package**	**θ_{JC}**	**θ_{JA}**	**Active**	**Stop Grant**	**STPCLK**	**Active**	**Stop Grant**	**STPCLK**
1X	DX	3.3	33	SQFP	3.5	25.0	0.279	0.008	0	0.290	0.008	0
	DX	5.0	33	PGA	1.5	17.0	0.491	0.041	0	0.512	0.041	0
	DX2	5.0	50	PGA	1.5	17.0	0.656	0.046	0	0.685	0.046	0
2X	DX	3.3	33	SQFP	3.5	25.0	0.283	N/A	0	0.294	N/A	0

NOTES:
1. All thermal values are measured at zero airflow.
2. Measurements taken on an SL Enhanced Intel486 CPU Evaluation Board (no heat spreader, no heat sink).
3. All measurements were taken using one module of each microprocessor.

3.3 Typical System Power Consumption Profiles

Table A-2 in the Appendix examines the power dissipated for four typical "system" profiles. These are systems in the sense that the power used is assumed controlled (except in the first case) by either the operating system or the system hardware, aside from the CPU. These cases, however, do **not** attempt to add the effects of other power dissipating components that would exist in a complete PC notebook system. The first case gives the most conservative power calculation: the maximum power that can be generated by the CPU. The second case gives the typical average power. The last two suggest power consumption possibilities that could occur in a given system, or even be guaranteed by power management. Many similar combinations would also be reasonable. V_{CC} in each case assumed as the standard value (5.0V or 3.3V).

In Case 1, the average power is calculated as the standard V_{CC} (3.3V or 5V) times the maximum current, I_{CC}(max), that can be drawn by a particular CPU. This calculation gives the maximum power that can be dissipated while continuously executing the most power consuming instruction sequence. This value should be used in a system design if no thermal power management is imposed, and the designer wants to minimize potential problems even under worst-case circumstances: a conservative design.

In Case 2, the average power is calculated as the standard V_{CC} (3.3V or 5V) times the TYPICAL current, as specified in Intel486 microprocessor data books. This calculates the average heat from the CPU that would be dissipated over time while executing a typical mix of software. The designer could use this power value in a less conservative design. However, if the CPU case temperature approaches its maximum specified value and no thermal power management is applied, the T_C(max) specification could be exceeded. (Section 5.5 discusses the time dependency issues in averaging the thermal power generated while executing a mix of software.)

In Case 3, the average power is calculated as the standard V_{CC} (3.3V or 5V) times the I_{CC}(max) for 10% of the time, and I_{CC}(typical) for 90% of the time. This is a more conservative assumption than Case 2, allowing for some intervals in which the CPU runs at full power, and an overall thermal guardband over Case 2.

Case 4 assumes that the current is distributed at the maximum for 10%, typical for 80%, and Stop Clock for 10% of the time. A mix of this sort is appropriate in a design where power management is applied to assert Stop Clock for at least 10% of the time. This mix could reduce the performance of the CPU. Suppose the system designer devises a theoretical mix of I_{CC}(Max, Typical and Stop Clock) which is exceeded by the system only 1% of the time when running all standard benchmarks and applications. The designer builds the system to that specification, with thermal power management responding only when the limit is exceeded. Then one can safely design the system assuming a significantly lower power than the absolute maximum, and experience performance degradation only 1% of the time.

The cases above illustrate that many less than worst-case power profiles are possible, depending on the software being run, and power management options being used. Intel recommends that systems be tested for thermal problems under the worst case power usage that the customer could contrive, and in an ambient temperature equal to the maximum specified for the design.

4.0 θ_{JA} BASED ON EXTERNAL T_A

This section provides some experimental thermal data which will help the designer better understand some of the system level issues affecting the thermal performance of a notebook. Section 4.1 describes in one experiment how test chamber θ_{JA} can be used as an approximate thermal performance criteria in the early stages of a notebook design. Section 4.2 shows how in another experiment θ_{JA} is affected by CPU location on the motherboard inside the notebook. Section 4.3 presents a model showing the relationship between power generated by other components on the motherboard and the CPU's θ_{JA} requirement.

4.1 Measurements from Commercial Notebooks

Since component location differs among notebook designs causing internal power densities to vary between notebooks there is no one place inside a notebook where T_A can be defined in order to obtain an accurate system thermal profile. This section presents experimental thermal data collected for four different Intel386 SL CPU notebooks running Indeo Video software and shows—as a rough estimate—that the ambient temperature (T_A) outside a notebook can be used with thermal resistance (θ_{JA}) to project CPU temperture (T_J, T_C).

In this notebook experiment, the CPU case temperature of four different Intel386 SL CPU notebooks (PQFP packages) were measured using K-type thermocouples a digital multimeter, and an Intel386 DX CPU-based system with data acquisition software. To simulate maximum I_{CC} consumption, each notebook continuously executed Indeo Video software for the duration of the experiment. Table 4-1 shows the junction temperature and θ_{JA} calculated by using the thermal impedance equations from Section 2.2. The CPU case temperature, the maximum power consumption of 2.5W running Indeo Video software, and the test chamber θ_{JC} of 6°C/W as specified in the Intel386™ SL Microprocessor Data Book for the 196L PQFP were used to calculate the junction temperature of each CPU. With this calculated CPU T_J and measured ambient room temperature of 25°C, the corresponding notebook's θ_{JA} was obtained. θ_{JA} can be calculated by combining the first two equations in Section 2.2 as follows:

$$\theta_{JA} = \theta_{JC} + \frac{T_C - T_A}{P}$$

Although the test chamber θ_{JA} for the 196L PQFP of 23°C/W was collected with only the CPU present on a test board, the value obtained approximates θ_{JA} of a notebook operating in an environment of 25°C. One possible explanation is that the conductive and radiative effects the other components have on the CPU inside the operating notebook, such as the PC board, connectors, floppy disk, shielding and plastic enclosure, actually cause the temperature inside the notebook to be lower than expected. The end result is a lower θ_{JA} for the CPU inside the operating notebook than the θ_{JA} of a lone CPU inside a test chamber where the only conductive and radiative path is to the surrounding air, as shown in Notebooks #1 and #3. Table 4-1 shows the large variation in notebook θ_{JA} caused by different system designs. Thus, the test chamber θ_{JA} can be used as a rough guideline in the early stages of a notebook design. For a more in-depth analysis of the conditions inside an operating notebook, see Section 4.3. For the final design, the thermal performance of the system should always be verified by measuring the CPU case temperature.

4.2 θ_{JA} and θ_{JC} Measurements in an Actual Environment

A test motherboard with the same form factor as that of the original Test Notebook #4 was fabricated in order to take experimental measurements of θ_{JA} and θ_{JC} (see Figure 4-1). The only differences between the two boards are the following:

1. The test board had two slots instead of the 70/80 pin connectors on the motherboard.
2. The test board only had six thermal test packages mounted on it (three on the component side and three on the solder side).

Measurements from the 6 thermal test units yielded a θ_{JA} range of 20°C/W–25°C/W in the test chamber and 22°C/W–28°C/W in the Test Notebook and a θ_{JC} range of 3°C/W–5°C/W in both the test chamber and the Test Notebook (see Table 4-2).

Table 4-1. θ_{JA} Calculations for Intel386 SL CPU Notebooks Running Indeo™ Video Software

Notebook #	T_{CASE}	T_J (calculated)	θ_{JA} (calculated)
1	48.9	63.9	15.5
2	69.0	84.0	23.6
3	52.1	67.1	16.8
4	71.5	86.5	24.6

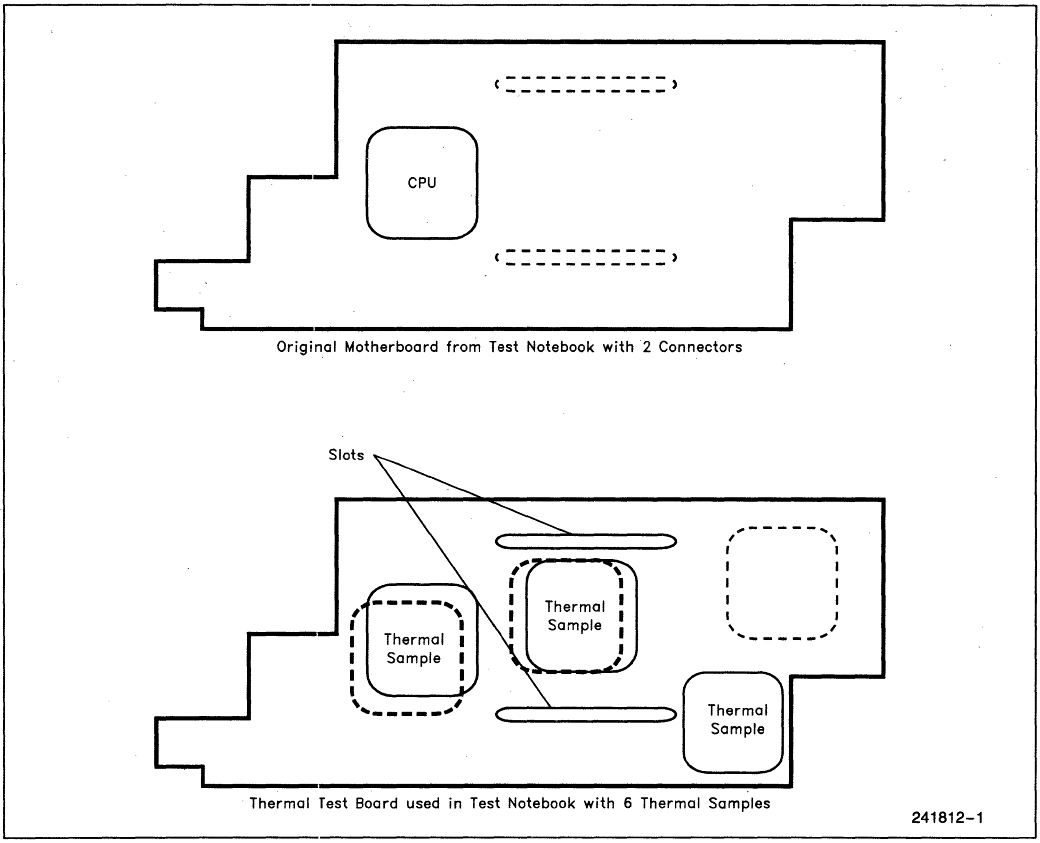

CPU
Original Motherboard from Test Notebook with 2 Connectors
Slots
Thermal Sample
Thermal Sample
Thermal Sample
Thermal Test Board used in Test Notebook with 6 Thermal Samples
241812-1

Figure 4-1. Test Notebook Boards

Table 4-2. Thermal Resistance θ_{JA} and θ_{JC} for the SQFP Package

	Test Motherboard in Test Chamber (°C/W)	Test Motherboard in Test Notebook (°C/W)
θ_{JA}	20–25	22–28
θ_{JC}	3–5	3–5

NOTES:

1. Test Notebook #4 was used to collect the experimental data.
2. 208L SQFP test package with heat spreader containing thermal test die was used in all experiments to vary and measure the power going into package as well as to measure the temperature of the die.
3. All measurements were made with zero airflow, simulating a typical notebook environment. Ambient temperature is defined as ambient temperature outside the notebook.

The worst CPU thermal location was on the center of the solder side with θ_{JA} = 28°C/W. The measured thermal resistance at this location was unfavorable because of the reduced CPU board area surrounding the CPU (due to the two slots) and the reduced convection cooling on the bottom side of the board. The best CPU thermal location was on the component side at one end of the test board with the most PCB area surrounding the package with θ_{JA} = 22°C/W. This shows how CPU location and system layout can impact the overall thermal performance of a notebook.

The θ_{JA} numbers should only be used as a first order estimate in preliminary notebook designs. Since the location of a CPU inside a notebook impacts thermal performance, T_J should always be verified in the final design by θ_{JC} and the CPU case temperature.

4.3 System Impact on CPU θ_{JA}

As notebooks evolve into smaller form factors with higher component density and smaller PCB sizes, the increasing power density inside the notebook has a large effect on CPU temperature. The power dissipated by components other than the CPU, and the layout of the components, as well as the thermal-mechanical characteristics of the enclosure must be taken into account in order to ensure a CPU junction temperature within specifications. In one model, such effects are approximated by the introduction of the factors R and P_b to the thermal impedence equations in Section 2.2:

$$\theta_{JA} = \frac{T_J - T_A - P_b \times R}{P_{CPU}}.$$

R is defined as the thermal coupling factor between the CPU and the other components on the PCB. This factor takes into account the effects of the power dissipation of the other components on the CPU case and junction temperatures. P_b is defined as PCB power dissipation. This P_b value is the power dissipated by all components (except the CPU) inside the notebook.

A detailed discussion of how R and P_b are used to calculate Thermal Headroom (thermal margin) for a Test Notebook is given in Section 5.3. Figure 4-2 gives a graphical representation of the effects of PCB power on the θ_{JA} requirements. The horizontal line represents the current method of a fixed θ_{JA} requirement of 23°C/W for this package, over the entire range of PCB power dissipated, P_b. The second line represents the model used with the factors P_b and R which takes into account the temperature rise inside the notebook and is obtained by substituting the example values T_J = 100°C, T_A = 30°C, P_{CPU} = 2W, and R = 3.9 into the preceding equation. This line shows as the board power (P_b) increases, the thermal margin inside the notebook becomes smaller due to the higher temperature environment. In both models, θ_{JA} must fall below the line to ensure a junction temperature below specification for the given conditions. For the Test Notebook #4 the cross-over point between the two lines is 6W. Beyond this point, the package thermal resistance of 23°C/W is too high and thermal enhancements are necessary to reduce that resistance. In summary, this example shows that for the current method, a θ_{JA} value obtained from the test chamber leaves margin for P_b less than 6W. However, for P_b greater than 6W, the junction temperature will be exceeded. Again, it is emphasized that the designer should always perform a thorough system thermal analysis to ensure the specified T_{CASE} (max) is not exceeded.

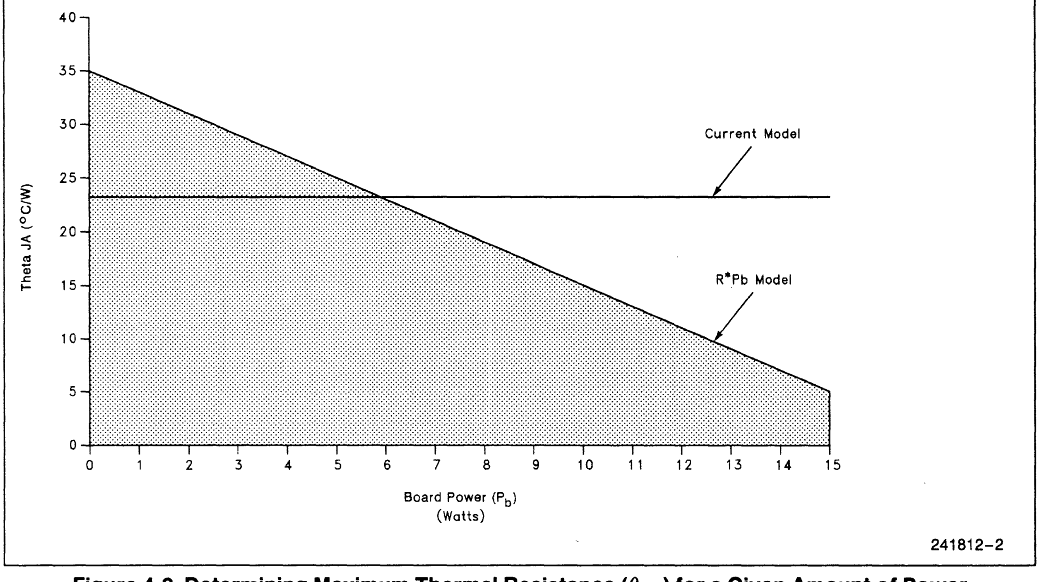


Figure 4-2. Determining Maximum Thermal Resistance (θ_{JA}) for a Given Amount of Power

5.0 CALCULATING THERMAL HEADROOM

Thermal Headroom is the temperature margin between the calculated T_A(max) and the T_A measured outside a given system, with a particular CPU and power. This section shows how to calculate Thermal Headroom and use it as simple model for a system's thermal properties. Then a term accounting for board power is added to the model, and the experimental measurements needed to implement this more sophisticated version are described. Finally, the significance of two secondary effects is analyzed.

5.1 Using Thermal Headroom Graphs

Figures A-1 and A-2 (in Appendix A) plot the calculated T_A(max) vs the power being used by the CPU and are intended to facilitate quick determination of thermal headroom. The lines on the graphs indicate the (estimated) maximum allowed ambient temperature (T_A in degrees Celsius) as a function of power dissipated (P in Watts). Maintaining T_A below or equal to T_A(max) indicates that the required T_C(max) is probably not exceeded. The graph lines are generated from the formula:

$$T_A(\text{max}) = T_C(\text{max}) - \theta_{CA} \times P$$

θ_{CA} is the thermal resistance to heat flow between the CPU case and the ambient environment, as specified in the Intel Packaging Handbook. As discussed in Section 4, experiments show that these parameters are approximately the same for a notebook PC with T_A measured in open air outside the notebook case. The tendency of the notebook case to *increase* θ_{CA} by adding extra layers of insulation is approximately offset by its action as a heat spreader, since it is thermally connected to the CPU board. Different types of CPU packages have different θ_{CA} values, and thus generate different lines on the graphs. For example, Figure A-1 shows the SQFP, PGA and PQFP packages for the SL Enhanced Intel486 SX CPU and Figure A-2 shows the SQFP and PGA packages for the SL Enhanced Intel486 DX2 CPU.

To determine thermal headroom for a given CPU, first determine the correct line for the CPU type. (Some of the CPUs are marked on the graphs. For a CPU type that is not, find its θ_{CA} in Table A-1, and match it to a CPU type that is marked on a graph line. Or use θ_{CA} as the slope, and T_C = 85°C as the temperature axis intercept to match directly to a graph line.)

Second, determine the power at which the CPU will operate. Various average power use scenarios could be appropriate for a given design; four of them are detailed in Table A-2 for each different CPU (discussed in detail in Section 3.3). One can also calculate a custom power usage profile for one's system using $P = I_{CC} \times V_{CC}$, and Table A-1, which gives I_{CC} under 3 different conditions (Active, Stop Grant and Stop Clock).

Third, draw an ordinate (vertical line) at the power value (determined above) to intersect the appropriate line for the CPU package type chosen. Draw an abscissa (horizontal line) from the intercept to the T_A axis, obtaining the maximum recommended ambient temperature for this system. If this $T_{A(max)}$ is greater than what is measured in the air outside the actual system, the design has a positive thermal headroom of $T_{A(max)} - T_A$(measured), as long as the effect of "board power", P_b, is neglected. If however the actual system is exceeding this T_A(max), the thermal headroom is negative even before considering P_b, and thus the thermal properties of the design will need improvement ("thermal mitigation"), (See Section 4 for P_b definition, and Section 5.3 for more information about Thermal Headroom).

There are numerous techniques for thermal mitigation, or improving the thermal properties of a design (i.e., a lower voltage version of the CPU or a CPU package with a lower θ_{CA} could be used). Depending on package size and material, various θ_{CA} values can be obtained. Various passive and active thermal management strategies are discussed in Section 6.

5.2 Example of Graph Use

Consider the 5V PQFP SL Enhanced Intel486 SX-33 CPU, which is likely to have special thermal needs because of its power requirements. The graph line for it is indicated by the label (33 MHz PQFP 5V) on the Intel486 SX CPU graph, or by the fact that its slope value from Table A-1 is 17.0 °C/W. The power shown for Figure A-1 is for Case 1: V_{CC} = 5.0V; I_{CC} = 0.685 mA (from Table A-1); Active Max for 100% of the time; P = 3.43W (from Table A-2, or calculation).

The ordinate is drawn from the P axis at 3.43W to intersect the intermediate slope line, and the abscissa from that point intersects the Temp. axis at about 27°C. If we assume 40°C is the lowest T_A that can be readily achieved, we get a negative thermal headroom of 13°C. Designing a portable PC with this CPU clearly will require thermal mitigation.

5.3 Adjusting Thermal Headroom for Board Power

Experiments have shown that the term $R \times P_b$ provides a good way to model the effects on the CPU T_C due to other heat sources inside a notebook. (Here R is an experimentally determined thermal coupling coefficient, and P_b is the board power, as defined in Section 4.). The term adds to T_C, or reduces the T_A(max) that is required to ensure that T_C does not exceed T_C(max):

$$T_A(\text{max}) = T_C(\text{max}) - \theta_{CA} \times P_{CPU} - R \times P_b$$

To calculate thermal headroom adjusted for the effect of P_b, subtract R × P_b from the thermal headroom calculated as above. This of course makes the headroom smaller (worse), but by how much? This depends on the size of P_b, but also on R, which is highly dependent on the particular design. In theory, the smallest value possible for R is zero: no thermal coupling between the CPU and other heat sources inside the notebook. From testing one system, the range measured experimentally (with no effort made to thermally isolate the CPU) has been 3.9 to 4.9. Values closer to zero are obtalned by positioning the other high-power devices away from the CPU, and thermally grounding them to the outer case. (Section 5.4 describes how to measure R for a given system.)

5.4 Experimental Measurements are Essential

The θ_{CA} values given by Intel can be used as a rough "rule of thumb" to estimate likely thermal margins. If the thermal headroom calculated from the graphs is positive for a given design even after correcting for the board power (R = 4 would be conservative for a real notebook design), the design is most likely satisfactory. But even then, Intel recommends that the CPU T_C be measured when the complete design can be run at full power, with T_A at the maximum allowed by the designer's specifications, to be really sure that T_C(max) will never be exceeded. If a conservative calculation of thermal headroom (including R × P_b) is negative, it is essential that the system be tested, and improvements in thermal mitigation be made until T_C(max) is never exceeded.

There are several levels of thermal experiments that can be used. The simplest is to just measure the CPU T_C, and make adjustments in the design until it never exceeds T_C(max). Then the equations and graphs can be ignored; if the design meets the T_C(max) specification, it does not matter if the (estimated) T_A(max) is violated, as far as Intel's CPU is concerned. (Consideration should be given, however, to other components, such as a disk drive, that might have trouble due to high temperatures inside the notebook.)

Suppose, however, the design is expected to be used for several variations over time, i.e., an Intel486 DX CPU now, and an Intel486 DX2 CPU later, with perhaps higher power peripherals which can also increase P_b in later versions. These later versions with more power will likely require more thermal mitigation efforts, but simply measuring T_C in the first version of the notebook will give little guidance about how much more thermal mitigation will be needed later. In this case, more detailed experiments on the first version, in order to build an accurate thermal model of the product line, can be very cost effective. This can be done using the equation from Section 5.3 and solving for T_C:

$$T_C = T_A + \theta_{CA} \times P_{CPU} + R \times P_b$$

In the preceding equation, T_A is the actual air temperature outside the notebook during the experiment; P_{CPU} is held fixed, and P_b is varied while the resulting T_C is measured. (T_A + θ_{CA} × P_{CPU}) is the intercept of the resulting straight line, and R is the slope. The easiest way to measure R is to disconnect the CPU (so P_{CPU} = 0) and measure T_C with V_{CC} at the upper and lower limits of its range (i.e., 4.5V and 5.5V). P_b is obtained for each V_{CC} value by V_{CC} × I_{CC}. A third data point requires no measurement; when P_b = 0, T_C = T_A.

When this semi-empirical model has been constructed for a given notebook design, it can be used to accurately determine thermal headroom for variations in both P_{CPU} (plugging in a higher frequency CPU) and P_b (adding higher power peripherals). Of course, if the thermal mechanical design is subsequently modified, R should be measured again for the new version.

If the designers expect the P_b to roughly track P_{CPU}, and R is small (as it should be in a good design), an approximation to the above model may make measurements easier: Assume P_b = C × P_{CPU}, where C is the coefficient relating board power to CPU power. The preceding equation for T_C then becomes:

$$\begin{aligned} T_C &= T_A + \theta_{CA} \times P_{CPU} + R \times P_b \\ &= T_A + \theta_{CA} \times P_{CPU} + R \times C \times P_{CPU} \\ &= T_A + (\theta_{CA} + R \times C)\, P_{CPU} \end{aligned}$$

Then (θ_{CA} + R × C) becomes a new coefficient, say θ_{CA}', which can be measured by varying P_{CPU} and P_b together by varying V_{CC} over its functional range.

5.5 Secondary Effects

Two kinds of secondary effects will be evaluated. The main model used in thermal analysis assumes a linear relationship between the temperature gradient and the rate of heat transfer, and also assumes a steady state (time independent) model. How valid are these assumptions?

Heat transfer by conduction is governed by a linear relationship between the temperature difference and the rate of heat transfer, and heat transfer by convection can be approximated by a linear relationship, as described in Section 2.1. However, heat transfer by radiation is proportional to the fourth power of the absolute temperatures involved. To demonstrate the contribution that heat transfer by radiation makes to cooling the

CPU, consider the largest allowed T_C (85°C) and the smallest T_A that most designs would find acceptable (40°C). One of the larger CPU packages is 4.4 cm square. Assume the largest emissivity (let $\epsilon = 1$). The temperatures must be converted to degrees Kelvin by adding 273°. Using the formula from Section 2.1, q, the radiative heat transfer in Watts is:

$$q = \epsilon\sigma A\,(T_C{}^4 - T_A{}^4) = 1 * (5.67\ 10^{-8}\ W/(m^2K^4))\ (0.044m)^2\ ((358\ K)^4 - (313\ K)^4) = 0.75W$$

To determine the significance of heat transfer by radiation in this case, one compares the 0.75W just calculated to the total heat transfer predicted by the linear approximation using the experimentally determined θ_{CA}. By rearranging the equation

$$\theta_{CA} = \frac{T_C - T_A}{P}$$ for P, one obtains

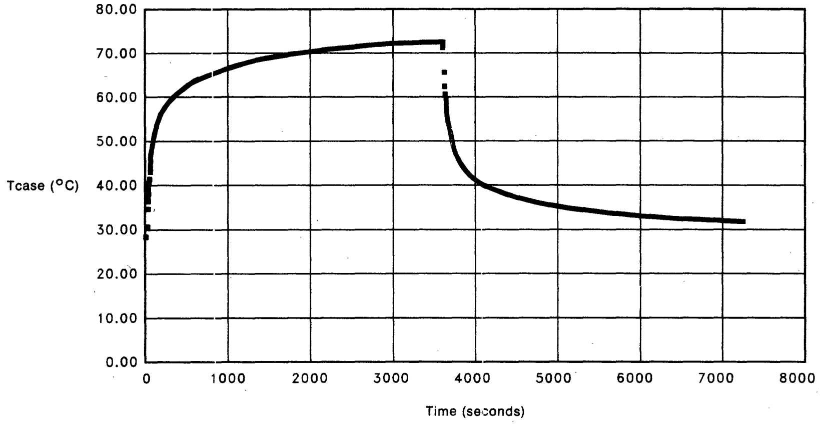
$$P = \frac{T_C - T_A}{\theta_{CA}}.$$

$T_C - T_A = 45°C$ in this case, and θ_{CA} ranges 15.5°C/W to 32.0°C/W for the CPUs covered in this article. These figures give a total heat power dissipation ranging from 1.41W to 2.90W. Thus, the radiative component varies between one fourth to one half of the total. This explains why the addition of black paint on a notebook case improved θ_{JA} measurably (see Section 6).

The experimental determination of θ_{CA} effectively incorporates the radiative component, along with the conductive and convective components, in a combined linear approximation, which will be accurate for temperatures near the values used for the measurement. The effect of the radiative, nonlinear component will be beneficial in that the actual power radiated away from the CPU, when temperatures are higher than those used in the θ_{CA} measurements, will be greater than predicted by the linear model. This means that T_C will not increase as much as predicted by the model, for a given increase in power.

The time independent assumption is fine if one is content to build a system to tolerate Case 1 maximum power, and with only passive thermal management (i.e., heat sinks and heat spreaders). However, if one assumes some power averaging over a typical mix of software, as in Cases 2, 3 and 4, one must consider the time dependent effects of bursts of maximum power, alternating with lower power periods. Also, if the design uses active thermal management, especially in a closed loop design with the system responding to a temperature sensor, the lag time between temperature sensing and response must be considered.

The experimental measurements taken on an Intel386 SL CPU notebook show how T_{CASE} varies with time at different CPU power levels, and allow an approximate determination of thermal time constants (see Figure 5-1). The time constant (approximately 3 minutes for this notebook) is defined here as the time elapsed from when power was switched from Full On to Standby, to when the temperature has declined toward its Standby asymptote by 1/e. This means that it is reasonable to average the CPU power over approximately one minute when calculating average power generated by a mix of software. This is a large interval in CPU cycles (billions of CPU clocks).



241812-3

Figure 5-1. T_{CASE} of Intel386 SL CPU (PQFP) Running Indeo™ Video Software

This time constant also indicates a significant lag between a temperature sensor reaching some action value (the action could be turning off the CPU clock for an interval), and a temperature response to that action. This means that the setpoint temperature (the temperature that triggers power reduction) should be somewhat below T_C(max). Note also that the temperature does not come to equilibrium until about an hour after a major power change. This shows that when testing T_C to assure that it does not exceed T_C(max), one should run the notebook under worst case conditions continually for at least an hour.

6.0 IMPROVING THERMAL HEADROOM

By using the thermal management theories that have been reviewed here and keeping in mind the notebook platform limitation, the designer can apply proven thermal management techniques in several areas, including increasing thermal conduction and convection, optimum system layout, and power management techniques.

6.1 Improving Thermal Convection, Conduction and Radiation

The most obvious method for improving thermal convection is by adding a fan to circulate the air. Unfortunately, fans are a compromise in mobile designs because of extra power and space requirements, and electromechanical noise. Intel's Thermal/Mechanical Tools and Analysis Group has collected data using more realistic techniques for reducing T_{CASE}. Data was collected from an actual Intel386 SL microprocessor notebook computer modified to measure T_{CASE}. The case temperature was measured by applying 3W to the CPU with all other devices/components off.

6.1.1 BLACK PAINT

The inside of the notebook case was painted Flat black using a paint that is highly absorptive in the infra-red as well as visible. Figure 6-1 shows that T_{CASE} was improved by 2.3°C, or 3%. Table 6-1 shows that heat transfer improvement by radiation reduces θ_{JA} by 0.5°C/W. The remaining experiments were performed with the inside of the notebook case painted black.

6.1.2 COPPER FOIL

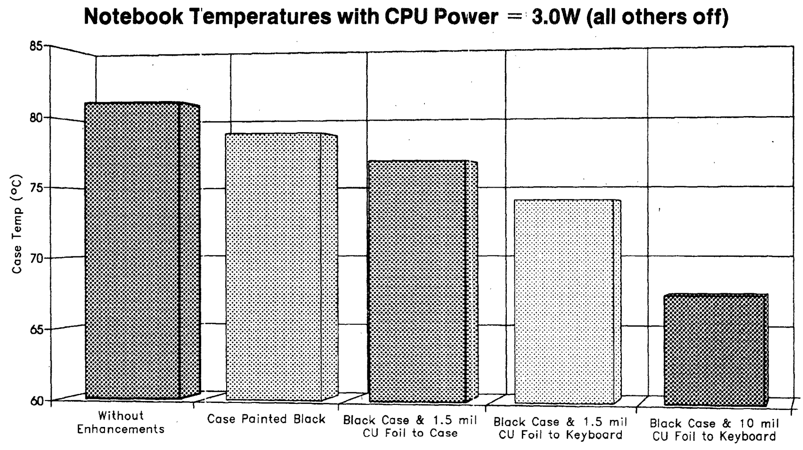
A copper foil (1" x 3" x 1.5 mil) was attached from the CPU to the bottom of the keyboard (constructed of aluminum material) and then to the plastic case using thermal grease. Figure 6-1 shows a 4.8°C (6%) and a 2.0°C (3%) improvement when the foil is connected from the CPU to the underside of keyboard and from the CPU to the case, respectively. After connecting the CPU to the bottom of the keyboard, θ_{JA} improved because of the higher thermal dispersion by the aluminum plate. A thicker copper foil (1" x 3" x 10 mil) was then connected from the CPU to the keyboard bottom which yielded an improvement of 11.4°C or 15%.

6.1.3 PERFLUOROCARBON FLUID AND SILICONE ELASTOMERS

Liquid heat sinks containing perfluorocarbon fluid can offer a reasonable substitute for standard heat sinks. The heat transfer coefficient for natural convection in a perfluorocarbon fluid is greater than that of natural air convection. Measurements were taken using a perfluorocarbon liquid heat sink connected between the CPU and the plastic case. Figure 6-2 shows a 7.5°C (10%) improvement in temperature.

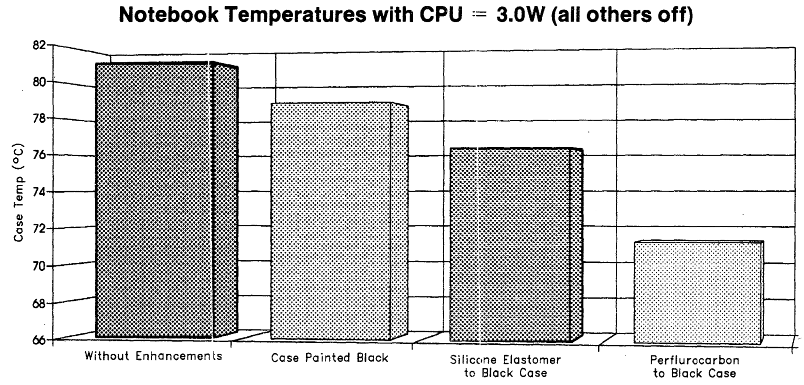
Table 6-1. Thermal Enhancements to θ_{JA}

	Black Paint	1.5 mil Copper Foil to Case	1.5 mil Copper Foil to Keyboard	10 mil Copper Foil to Keyboard
$\Delta\theta_{JA}$ (°C/W)	0.5	0.9	1.8	3.9



241812–8

Figure 6-1. Black Paint and Copper Foil Thermal Enhancements



241812–9

Figure 6-2. Silicone Elastomer and Perfluorocarbon Thermal Enhancements

A type of heat sink that helps blanket uneven surfaces is the silicone elastomers heat sink. These soft materials fill air gaps between hot components and the metal chassis. A piece of silicone elastomer was cut out to the same size as the CPU and placed between the CPU and the case, yielding a 1.5°C (2%) improvement (see Figure 6-2).

6.2 Optimizing System Layout

Power must be optimally distributed to ensure the lowest T_A. The objective is to reduce power density within the system to avoid hot spots at any particular device. Keeping the CPU away from batteries and power supplies is one challenge to the system designer. Thermally connecting the CPU case to the PC case can increase thermal area thus greatly improving thermal spreading. There are many opportunities for creativity in transferring heat away from the CPU.

6.3 Effective System Power Management

One technique which has become a standard in notebook designs is using Intel's System Management Mode (SMM) to effectively monitor system activity and shut off devices to slow or control clocks when low activity is detected. Another approach could be to monitor the CPU activity or temperature and slow or stop the CPU clock when a long period of system inactivity or a high temperature is detected.

7.0 CONCLUSION

System notebook designers need to pay close attention to the special thermal requirements for future notebook designs. These include demands for increasing computing power, and thus power consumption, in decreasing sizes. With proper thermal design, the CPU can be kept below its stated maximum case temperature, even under worst case conditions.

Experiments have shown that θ_{CA} as measured in open air in factory tests can be used for a rough estimate for an actual notebook PC design, where T_A is the air temperature outside the notebook. Using this θ_{CA} to estimate thermal headroom provides the simplest model. Experiments have also shown that adding a term to the thermal equation that includes the board power (P_b), with a coupling coefficient (R), provides a better model. This enhanced model takes into account the effect of other heat sources inside the notebook on the CPU T_C, and allows accurate prediction of the thermal effects of upgrades in a notebook design (adding a higher power CPU and/or peripherals on the board). Also, a number of suggestions have been made about how to improve the thermal characteristics of a design, by various passive and active techniques.

The information within this application note allows the designer to use initially a simple analytical model, as well as to build a semi-empirical model to correlate with the actual design. Techniques have been shown not only to increase the average thermal margin, but to eliminate thermal margin problems under worst case operating conditions. By designing for the worst case, and taking actual measurements to guarantee proper operation within spec limits, the notebook designer can provide the highest quality, most reliable products for their customers.

APPENDIX A

Table A-1. Power Consumption and Thermal Specifications for SL Enhanced intel486 CPUs

CLK	CPU	V_{CC}	Freq	Pkg	T_{CASE}	V_{CC} tol	I_{CC} Active		I_{CC} Stop Grant		I_{CC} Stop Clock		θ_{JC}	θ_{JA}	θ_{CA} (calc)
							Typ	Max	Typ	Max	Typ	Max			
1X	SX	3.3	25	SQFP	85	±0.30	0.250	0.315	0.020	0.040	0.0001	0.001	4.0	36.0	32.0
		3.3	33	SQFP	85	±0.30	0.300	0.385	0.025	0.050	0.0001	0.001	4.0	36.0	32.0
		5.0	25	PGA	85	±0.25	0.430	0.560	0.035	0.065	0.0002	0.002	1.5	17.0	15.5
		5.0	25	PQFP	85	±0.25	0.430	0.560	0.035	0.065	0.0002	0.002	3.5	20.5	17.0
		5.0	33	PGA	85	±0.25	0.590	0.685	0.040	0.080	0.0002	0.002	1.5	17.0	15.5
		5.0	*33*	*PQFP*	*85*	*±0.25*	*0.590*	*0.685*	*0.040*	*0.080*	*0.0002*	*0.002*	*3.5*	*20.5*	*17.0*
1X	DX	3.3	33	SQFP	85	±0.30	0.330	0.415	0.025	0.050	0.0001	0.001	3.5	25.0	21.5
		5.0	33	PGA	85	+0.25	0.500	0.630	0.040	0.080	0.0002	0.002	1.5	17.0	15.5
		5.0	33	PQFP	85	±0.25	0.500	0.630	0.040	0.080	0.0002	0.002	3.5	20.5	17.0
		5.0	50	PGA	85	±0.25	0.775	0.950	0.050	0.100	0.0002	0.002	1.5	17.0	15.5
1X	DX2	3.3	40	SQFP	85	±0.30	0.375	0.450	0.020	0.040	0.0001	0.001	3.5	24.0	20.5
		3.3	50	SQFP	85	±0.30	0.460	0.550	0.035	0.065	0.0001	0.001	3.5	24.0	20.5
		5.0	50	PGA	85	±0.25	0.775	0.950	0.023	0.050	0.0002	0.002	1.5	17.0	15.5
		5.0	66	PGA	85	±0.25	0.975	1.200	0.045	0.090	0.0002	0.002	1.5	17.0	15.5
2X	SX	3.3	25	SQFP	85	±0.30	0.250	0.315	N/A	N/A	0.0001	0.001	4.0	36.0	32.0
		3.3	33	SQFP	85	±0.30	0.330	0.415			0.0001	0.001	4.0	36.0	32.0
		5.0	25	PQFP	85	±0.25	0.430	0.560			0.0002	0.002	3.5	20.5	17.0
		5.0	33	PQFP	85	±0.25	0.590	0.685			0.0002	0.002	3.5	20.5	17.0
2X	DX	3.3	33	SQFP	85	±0.30	0.330	0.415	N/A	N/A	0.0001	0.001	3.5	25.0	21.5
		5.0	33	PQFP	85	±0.25	0.500	0.630			0.0002	0.002	3.5	20.5	17.0

Table A-2. Thermal Headroom based on Typical Power Consumption Profiles of SL Enhanced Intel486 CPUs

CLK	CPU	V_{CC}	Freq	Pkg	T_C	V_{CC} tol	Case 1			Case 2			Case 3			Case 4		
							I_{CC} AVG	Power AVG	Thermal Headrm	I_{CC} AVG	Power AVG	Thermal Headrm	I_{CC} AVG	Power AVG	Thermal Headrm	I_{CC} AVG	Power AVG	Thermal Headrm
1X	SX	3.3	25	SQFP	85	±0.30	0.32	1.04	11.7	0.25	0.83	18.6	0.26	0.85	17.9	0.23	0.76	20.5
		3.3	33	SQFP	85	±0.30	0.39	1.27	4.3	0.30	0.99	13.3	0.31	1.02	12.4	0.28	0.92	15.6
		5.0	25	PGA	85	±0.25	0.56	2.80	1.6	0.43	2.15	11.7	0.44	2.22	10.7	0.40	2.00	14.0
		5.0	25	PQFP	85	±0.25	0.56	2.80	(2.6)	0.43	2.15	8.5	0.44	2.22	7.3	0.40	2.00	11.0
		5.0	33	PGA	85	±0.25	0.69	3.43	(8.1)	0.59	2.95	(0.7)	0.60	3.00	(1.5)	0.54	2.70	3.1
		5.0	*33*	*PQFP*	*85*	*±0.25*	*0.69*	*3.43*	*(13.2)*	*0.59*	*2.95*	*(5.2)*	*0.60*	*3.00*	*(6.0)*	*0.54*	*2.70*	*(1.0)*
1X	DX	3.3	33	SQFP	85	±0.30	0.42	1.37	15.6	0.33	1.09	21.6	0.34	1.12	21.0	0.31	1.01	23.3
		5.0	33	PGA	85	±0.25	0.63	3.15	(3.8)	0.50	2.50	6.3	0.51	2.57	5.2	0.46	2.32	9.1
		5.0	33	PQFP	85	±0.25	0.63	3.15	(8.6)	0.50	2.50	2.5	0.51	2.57	1.4	0.46	2.32	5.6
		5.0	50	PGA	85	±0.25	0.95	4.75	(28.6)	0.78	3.88	(15.1)	0.79	3.96	(16.4)	0.72	3.58	(10.4)
1X	DX2	3.3	40	SQFP	85	±0.30	0.45	1.49	14.6	0.38	1.24	19.6	0.38	1.26	19.1	0.35	1.14	21.7
		3.3	50	SQFP	85	±0.30	0.55	1.82	7.8	0.46	1.52	13.9	0.47	1.55	13.3	0.42	1.40	16.4
		5.0	50	PGA	85	±0.25	0.95	4.75	(28.6)	0.78	3.88	(15.1)	0.79	3.96	(16.4)	0.72	3.58	(10.4)
		5.0	66	PGA	85	±0.25	1.20	6.00	(48.0)	0.98	4.88	(30.6)	1.00	4.99	(32.3)	0.90	4.50	(24.8)
2X	SX	3.3	25	SQFP	85	±0.30	0.32	1.04	11.7	0.25	0.83	18.6	0.26	0.85	17.9	0.23	0.76	20.5
		3.3	33	SQFP	85	±0.30	0.42	1.37	1.2	0.33	1.09	10.2	0.34	1.12	9.3	0.31	1.01	12.7
		5.0	25	PQFP	85	±0.25	0.56	2.80	(2.6)	0.43	2.15	8.5	0.44	2.22	7.3	0.40	2.00	11.0
		5.0	33	PQFP	85	±0.25	0.69	3.43	(13.2)	0.59	2.95	(5.2)	0.60	3.00	(6.0)	0.54	2.70	(1.0)
2X	DX	3.3	33	SQFP	85	±0.30	0.42	1.37	15.6	0.33	1.09	21.6	0.34	1.12	21.0	0.31	1.01	23.3
		5.0	33	PQFP	85	±0.25	0.63	3.15	(8.6)	0.50	2.54	2.5	0.51	2.57	1.4	0.46	2.32	5.6

NOTES:

Case temperature specifications assume a heat spreader and no heat sink.

CASE 1:	I_{CC} Active (max)	= 100%
CASE 2:	I_{CC} Active (typ)	= 100%
CASE 3:	I_{CC} Active (max)	= 10%
	I_{CC} Active (typ)	= 90%
CASE 4:	I_{CC} Active (max)	= 10%
	I_{CC} Active (typ)	= 80%
	I_{CC} Stop Clock (max)	= 10%

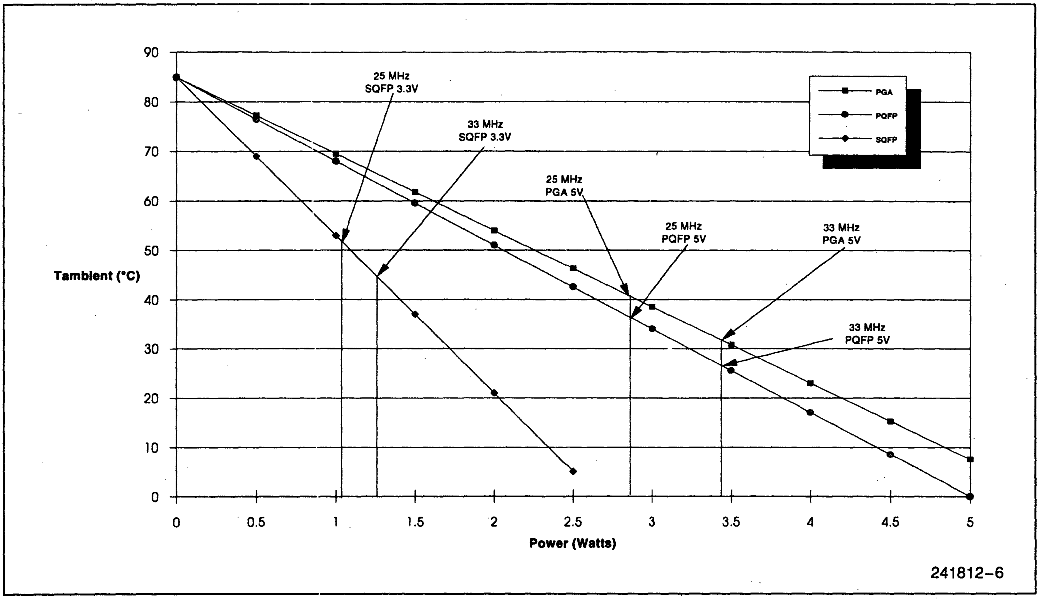


Figure A-1. Maximum Thermal Headroom for SL Enhanced Intel486 SX CPUs

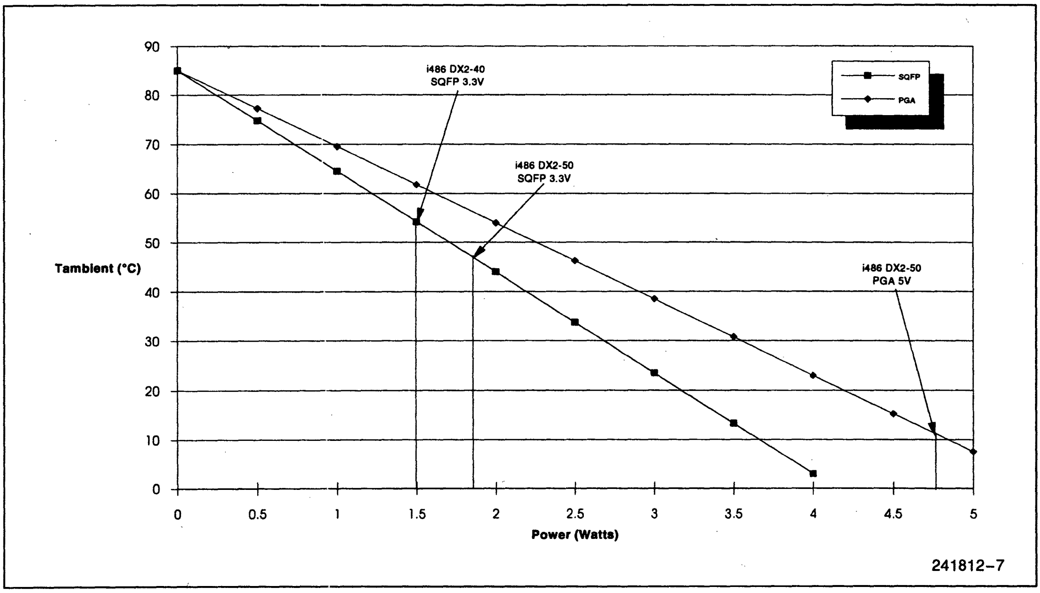


Figure A-2. Maximum Thermal Headroom for SL Enhanced Intel486 DX2 CPUs

APPENDIX B
THERMAL ENHANCEMENT VENDORS

Chomerics, Inc.

77 Dragon Court
Woburn, MA 01888-4014
(617) 935-4850
FAX: (617) 933-4318
Product ID: CHO-THERM A274
(Silicon Elastomer)

3M Corporation

Building 223-6S-04
3M Center
St. Paul, MN 55144-1000
(612) 733-3735 or (800) 833-5045
Product ID: Fluorinert Liquid FC-77
(Perfluorocarbon Fluid)

APPENDIX C BIBLIOGRAPHY

BIBLIOGRAPHY

SL Enhanced Intel486™ Microprocessor Data Sheet Addendum, Intel Corporation, 1993. Order Number 241696.

Intel386™ SL Microprocessor SuperSet Data Book, Intel Corporation, 1992. Order Number 240814.

1993 Packaging Handbook, Intel Corporation, 1993. Order Number 240800.

Physics, Second Edition, Paul A. Tiper. Worth Publishers, Inc., 1982, pp. 531, 535.

AP-504

APPLICATION
NOTE

Clock Throttling the SL Enhanced Intel486™ CPU in a Networked Environment

PHILIP BRACE
RICK BROWN
TODD ERDNER
JOSEPH MIDDLETON

May 1994

Clock Throttling the SL Enhanced Intel486™ CPU in a Networked Environment

CONTENTS PAGE

FIGURES

TABLES

1.0 INTRODUCTION

The SL Enhanced Intel486 microprocessors contain new features to enable simple, economical, and robust power management. The Energy Star program and a general world wide trend towards responsible energy consumption have generated interest in providing computer equipment which can enter a low power state when not being used. The SL Enhanced Intel486 microprocessors enable a simple design which is capable of power managing both the CPU itself, and the system as a whole.

The SL Enhanced Intel486 microprocessors can be placed in a low power state through software or hardware. Execution of the HALT instruction will cause the CPU to automatically enter a ~20 mA–55 mA state called the Auto HALT Power Down state. The CPU will issue a normal HALT bus cycle when entering this state.The CPU will transition to the Normal state on the occurrence of INTR, NMI, SMI#, RESET, or SRESET.

The STPCLK# interrupt allows system hardware to control the power consumption of the CPU by stopping the internal clock (output of the PLL) to the CPU core in a controlled manner. When STPCLK# is asserted the SL Enhanced Intel486 CPU enters a low power state, the Stop Grant state, of approximately 20 mA to 55 mA at the conclusion of the current bus cycle. Deasserting STPCLK# will enable the processor to resume functioning on the next CLK cycle. Periodically asserting and deasserting STPCLK# can result in significant power savings while still keeping a base level of CPU activity to ensure that interrupts are not missed, time of day lost, network connections dropped, etc. The process of rapidly asserting and deasserting STPCLK# to provide power management while maintaining a reduced level of system activity is referred to as clock throttling (see Figure 1). Clock throttling can be implemented with a variety of periods and duty cycles, or with an event driven period such as deassertion on INTR.

It has been suggested that clock throttling may cause problems and performance degradation with networked systems. This paper will discuss the power savings possible with clock throttling, and results of network testing. Included also is an overview of the power requirements of typical LAN cards.

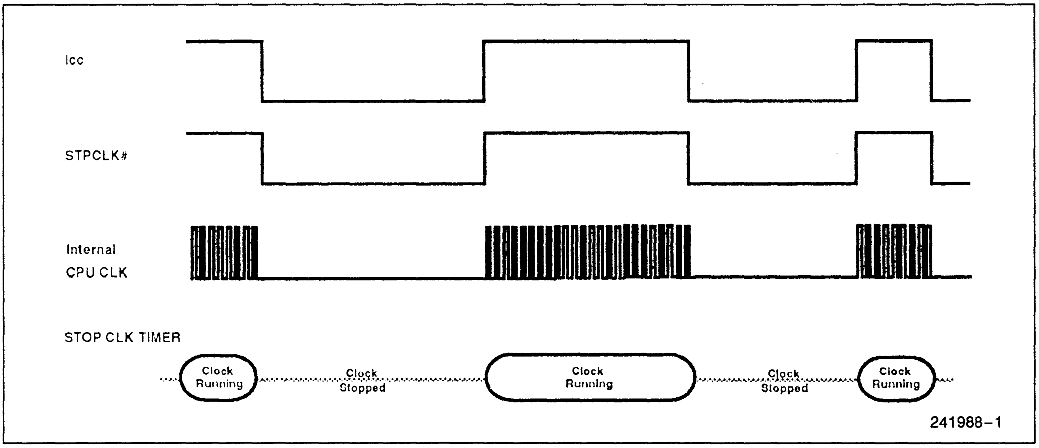


Figure 1. Clock Throttling

2.0 TEST HARDWARE

All testing was done with the use of a daughter card which was inserted into an Intel486 CPU PGA socket on a standard Intel486 CPU motherboard. This daughter card consisted of a socket for an SL Enhanced Intel486™ CPU, an input for a signal from a pulse generator from which STPCLK# was derived, and two PLDs to perform the following functions. Figure 2 shows the hardware setup.

- STPCLK# specification requires that if STPCLK# is asserted, it must be held at least until the Stop Grant Bus Cycle is returned (see Figure 3). The PLD circuitry ensured that this specification was met regardless of the period and duty cycle of the signal from the pulse generator.
- At the conclusion of the STPCLK# period, once STPCLK# is deasserted it must remain deasserted for a minimum of 5 clocks before being asserted again (see Figure 3). Again the PLD circuitry ensured that this specification was met.
- Deassertion of STPCLK# on INTR. The PLD circuitry was designed so that STPCLK# could be deasserted on INTR if desired. A jumper controlled whether or not this occurred (see Figure 4). The logic was designed such that INTR could only deassert STPCLK# after the Stop Grant Bus Cycle had been returned.

Appendix A contains a schematic of the daughter card and the PLD equations.

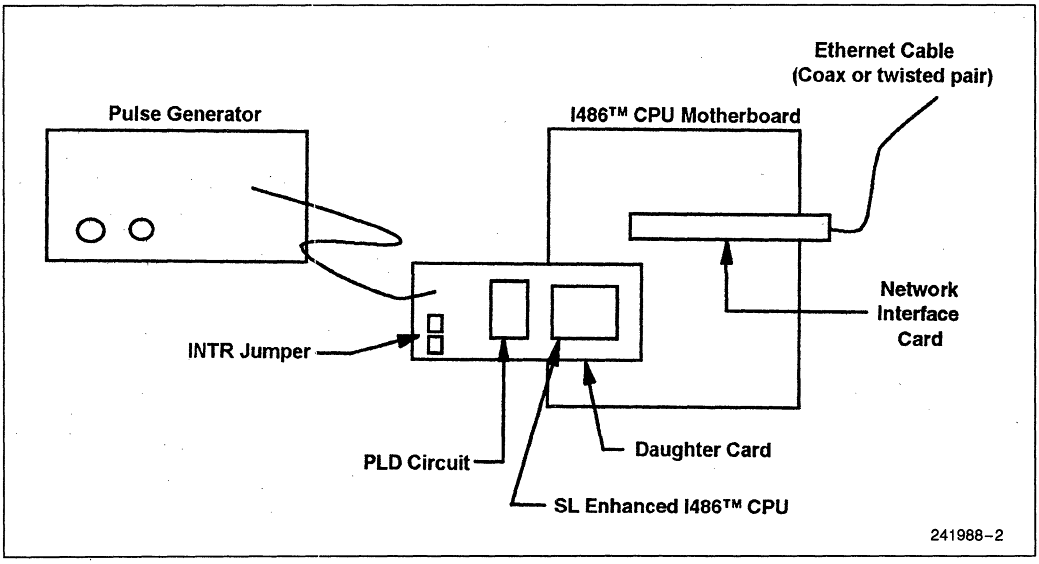


Figure 2. Test Hardware Setup

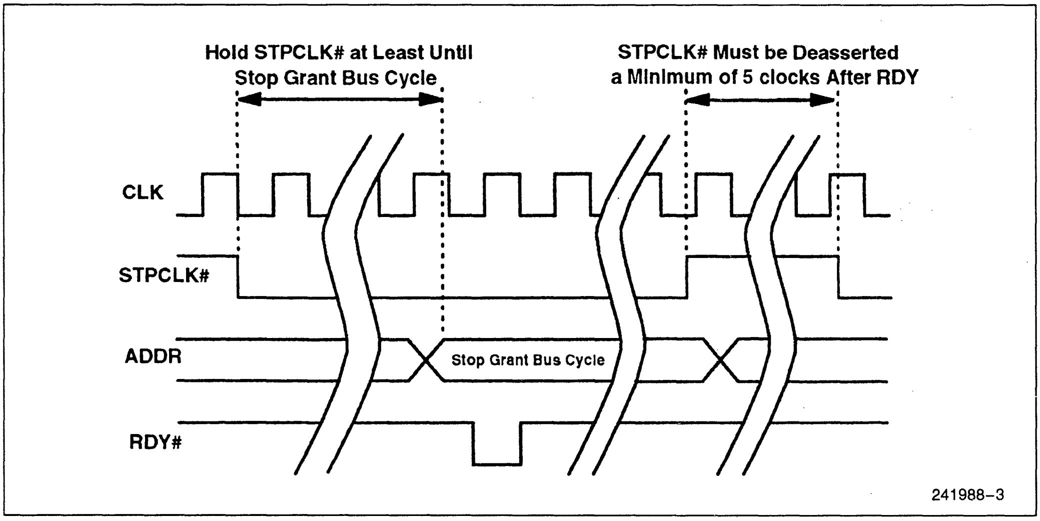

Hold STPCLK# at Least Until
Stop Grant Bus Cycle
STPCLK# Must be Deasserted
a Minimum of 5 clocks After RDY
CLK
STPCLK#
ADDR
Stop Grant Bus Cycle
RDY#
241988-3

Figure 3. Stop Grant Bus Cycle

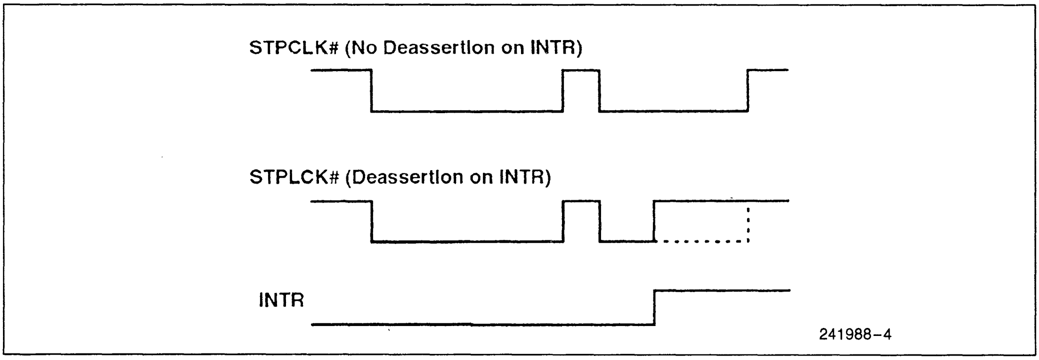

STPCLK# (No Deassertion on INTR)
STPLCK# (Deassertion on INTR)
INTR
241988-4

Figure 4. STPCLK# and INTR

3.0 POWER SAVINGS WITH CLOCK THROTTLING

Asserting STPCLK# saves power on the SL Enhanced Intel486 CPU motherboard by reducing the current to the CPU, and by reducing the current to DRAM and external cache memory. As the CPU is effectively halted no memory cycles will occur. In the case of clock throttling, as STPCLK# is asserted and deasserted, the motherboard will rapidly toggle between a low power and normal power state. The average of these two states will be the effective "sleeping state". It is obvious, therefore, that the *duty cycle* of STPCLK# will have a profound effect on the sleeping energy consumption.

In the implementation where STPCLK# is deasserted for the remainder of the current period with INTR (see Figure 4) then the *period* of STPCLK# will also have an effect on the sleeping energy consumption. An unused sleeping system (AT† type PC) will experience a real time clock interrupt every 55 ms. As the period increases, the effect of deasserting STPCLK# on INTR reduces the effective duty cycle (see Figure 5).

Power consumption data was collected for a variety of duty cycles and periods, and with/without deassertion of STPCLK# on INTR. Please see Figures 6 through 8. Data for charts is in Appendix B.

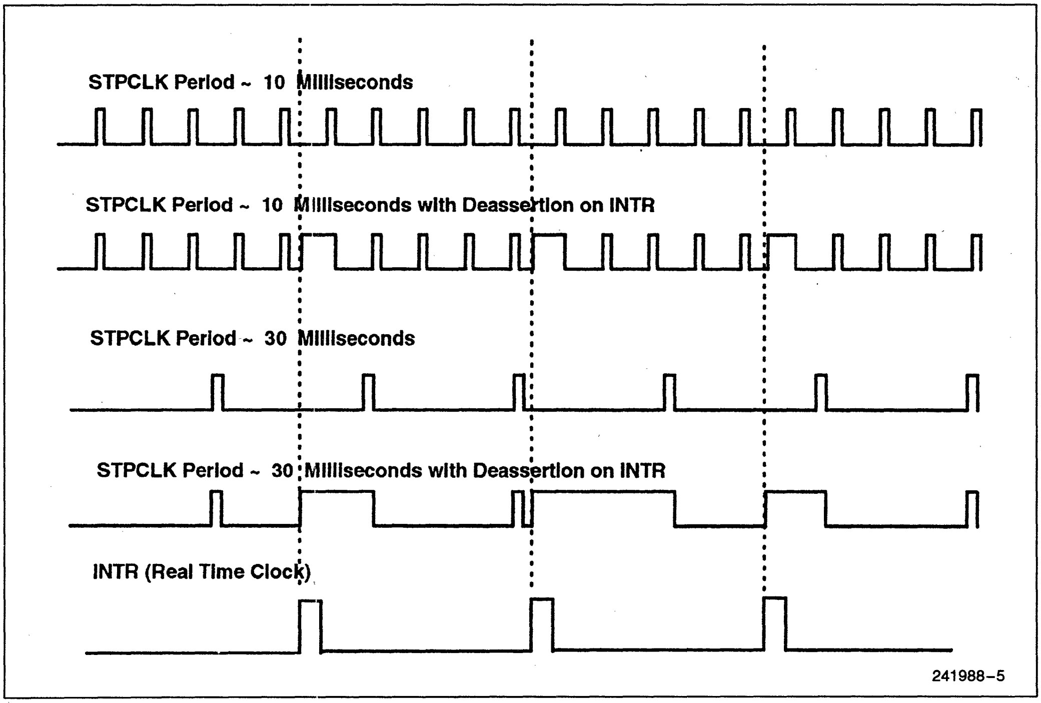

STPCLK Period ~ 10 Milliseconds
STPCLK Period ~ 10 Milliseconds with Deassertion on INTR
STPCLK Period ~ 30 Milliseconds
STPCLK Period ~ 30 Milliseconds with Deassertion on INTR
INTR (Real Time Clock)
241988-5

Figure 5. Effect of Deasserting STPCLK# on INTR

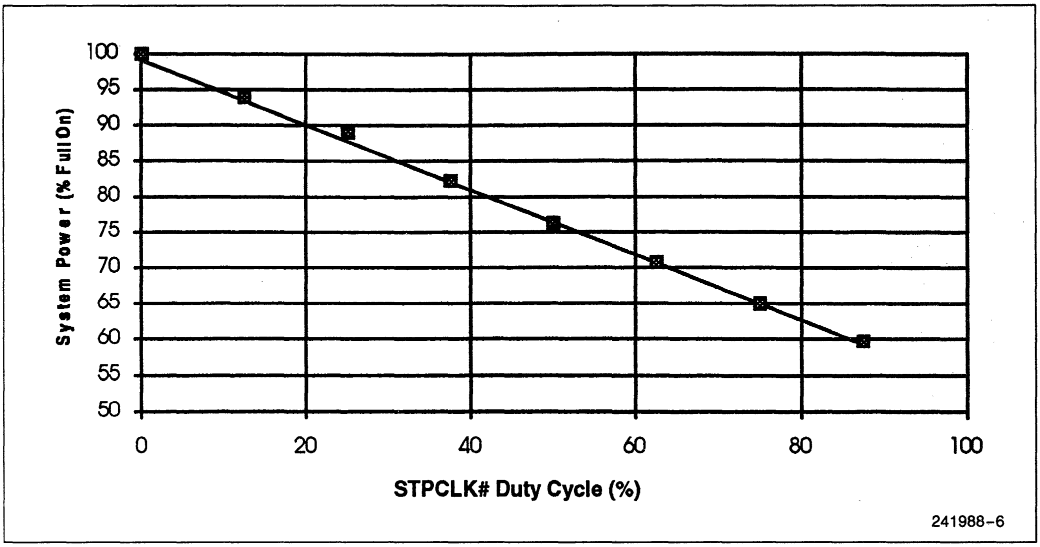


Figure 6. Power vs STPCLK Duty Cycle, Interrupts do not Deassert STPCLK#

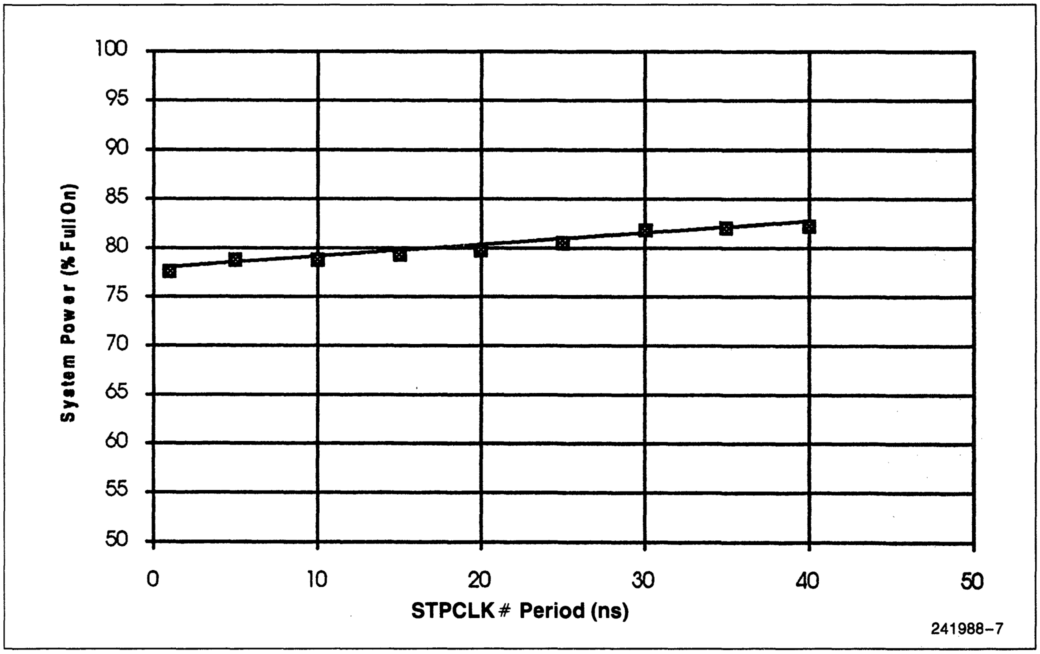


Figure 7. Power vs STPCLK Period, Duty Cycle = 50%, Interrupts Deassert STPCLK#

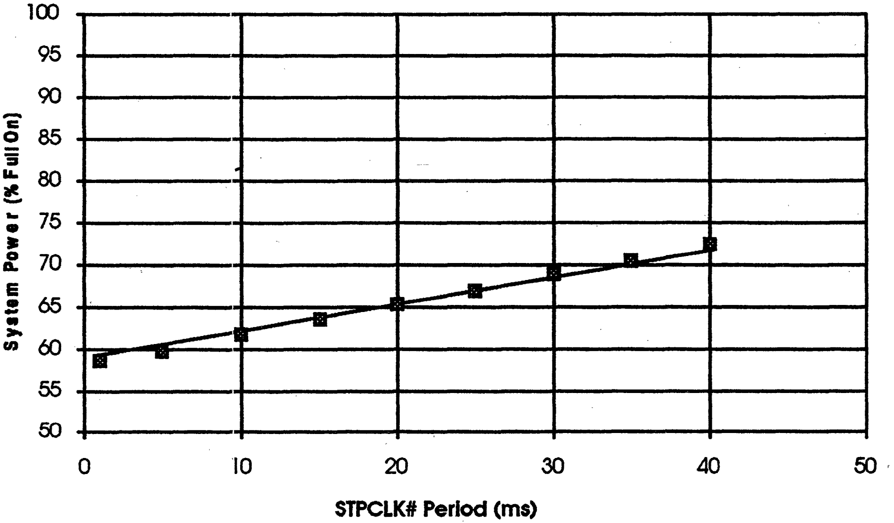


Figure 8. Power vs STPCLK Period, Duty Cycle = 90%, Interrupts Deassert STPCLK#

Looking at the graphs it is clear that the duty cycle of the STPCLK# pulse has the most effect on the overall system power. The period of the pulse also affects the power, however not as significantly as the duty cycle variation. The period of the pulse has the greatest effect when the duty cycle is set at 90%.

4.0 OVERVIEW OF NETWORK TEST PLAN

Given the almost infinite matrix of network operating systems, network topologies, network interface cards, host system bus architectures, and potential STPCLK# periods and duty cycles, an exhaustive test of all permutations is not feasible. Therefore, testing centered on the Novell network operating system, the Ethernet™ network topology, and the ISA bus architecture due to their overwhelming market share. An understanding of the test philosophy and test conditions can allow some extrapolation of results to other environments.

Maintaining the network connection while in a low power state (sleeping) is mandatory for any energy efficient computer. A variety of network cards and STPCLK# periods and duty cycles were tested in a PC connected to a Novell NetWare† 3.11 network. Section 5 discusses the procedures and results of the testing of the ability of clock throttled systems to maintain a network connection.

Peer-to-peer networks have an added complication for sleeping PCs in that any user's workstation may be configured as a server as well as a client. A user's sleeping PC may be accessed by another user. This access may or may not be considered a wake-up event, depending on the particular power management scheme. Section 6 covers the performance implications of transferring files from a sleeping PC in a peer-to-peer network (Novell Lite).

A similar problem to the previous one involving peer to peer networks, can occur in classical client-server networks where user's PCs are configured as network print servers. Section 7 discusses test results of printing to a remote sleeping print server in a Netware 3.11 environment.

Intel's Compatibility Validation Lab continually tests new Intel CPUs in a variety of machines to guarantee all new microprocessors are completely Intel compatible. These workstation tests are regularly performed over a network to facilitate the process. While an energy efficient computer would not normally be sleeping during these tests (as they emulate user activity), for research purposes these tests were completed with an aggressively clock-throttled machine. The details of these tests are in Section 7.

5.0 NETWARE 3.11 TEST SUMMARY

5.1 Test Environment

The test environment can be summarized with the following list:

- Novell 3.11 File Server running on an Intel486 DX2 66-MHz CPU
- Traffic Generating Station using Intel NetSight Professional
- One Traffic Monitoring Station using Intel LanSight
- Two dummy clients
- One test station with an SL Enhanced Intel486 DX2 66-MHz CPU
- Twisted Pair Ethernet Cable
- Hub

5.2 Test Description

This section of the documents describes the tests performed and provides explanations for some of the independent variables used in the test. As mentioned in section 4, the primary motive of the test suites was to demonstrate that the network connection is not lost when the processor enters into the power down or stop clock state. The variables initially considered to be a factor included: the network utilization, the period of the STPCLK# pulse, the duty cycle of the STPCLK# pulse, whether interrupts deasserted the STPCLK# signal or not, and several others. To perform an exhaustive test of all possible values of all possible parameters was not feasible for the scope of this paper, (and not particularly useful as shown later), so several parameters were fixed at chosen values.

Network Utilization

The network utilization was fixed at 22.8%. [The traffic generator was set at 25% but measured values demonstrate that the actual utilization was 22.8%.] This was fixed because preliminary tests demonstrated that network load did not have a perceptible effect on the outcome of the test. Only broadcast packets, or individually addressed packets are actually loaded into the network interface card's receive buffer. Given the assumption that the client is in a low power state because of minimal system activity, then it follows that only a minute percentage of the network traffic will correspond to the sleeping station. 22.8% was chosen as the utilization of a fairly busy network.

Interrupts will not Deassert STPCLK#

In the real world environment, this parameter will be implemented by the power management logic of the particular design. If interrupts deassert STPCLK#, then all interrupts (like the real time clock, and many others) will bring the processor out of low power mode. This scenario will provide improved performance over the case where interrupts do not deassert STPCLK#. For our tests, the worst case implementation was chosen. That is, interrupts will not deassert the STPCLK# signal. If the tests succeed in this environment, they will succeed if interrupts deassert STPCLK#.

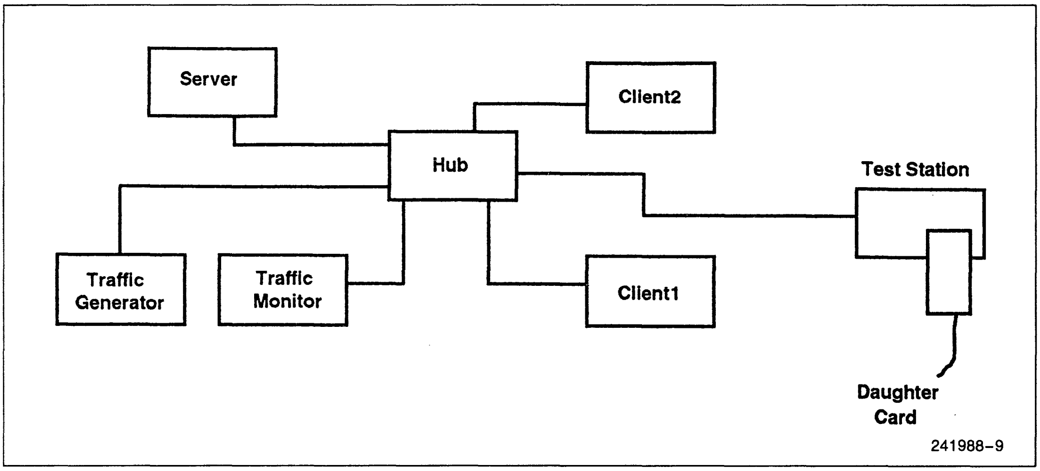


Figure 9. Test Environment

Test Station Idle

The test station is not performing any operations for the duration of the test. This parameter is based on our initial assumption that the client is asleep *because* there is no activity.

Time

To determine the length of time to test some of the cards, it was first necessary to determine the length of time that the server drops the connection when no activity is occurring. If the cable is actually disconnected from the network interface card, the server will drop the connection in less than fifteen minutes. A reasonable length of time to test would be 4x this value. So, each of the tests was run for *at least* one hour. As a sanity check, the STPCLK# pin was grounded (made active) and the length of time for the server to drop the connection was measured.

Table 1. Network Test Loading (Fixed 25% Loading, Interrupts Disabled)

Network Card	STPCLK# Period	Duty Cycle	Time	Pass/ Fail	Server Disconnect Time
3Com Etherlink II	1 ms	90%	1.1 hr	pass	11 min
	8 ms	90%	1.1 hr	pass	
	55 ms	90%	1.4 hr	pass	
Ansel 2100	1 ms	90%	1 hr	pass	13 min
	8 ms	90%	1.4 hr	pass	
	55 ms	90%	1 hr	pass	
Ansel 2200	1 ms	90%	1 hr	pass	12 min
	8 ms	90%	1 hr	pass	
	55 ms	90%	2.4 hr	pass	
Intel EtherExpress™ Card	1 ms	90%	1 hr	pass	13 min
	8 ms	90%	1 hr	pass	
	55 ms	90%	1 hr	pass	
Kingston KNE 2121	1 ms	90%	1.4 hr	pass	11 min
	8 ms	90%	1 hr	pass	
	55 ms	90%	1 hr	pass	
SMC 16	1 ms	90%	1 hr	pass	11 min
	8 ms	90%	1 hr	pass	
	55 ms	90%	2.4 hr	pass	

NOTES:
90% is percentage time that the STPCLK# is asserted (low).
Time is the time that each card was tested under 25% network load.
Pass/Fail: A card is considered to have passed if it maintains connection with the network for 1 hour or more.
Server Disconnect time is time that it takes for the network to drop a station that does not respond. This time was computed by grounding the STPCLK# (completely halting the CPU).

5.3 Test Results

All of the LAN cards tested passed all of the tests with the initial conditions as described in Section 5.1..As all the tests passed, it is interesting to consider some of the factors at work. One factor is the packet size. The default Novell packet size is approximately 1.5 Kbytes. On a card with a receive buffer of 32K, twenty-one full size packets can be put into the buffer without dropping any. When little or no network traffic is being generated by a particular station, the number of packets addressed to the station is small. Generally speaking, the server will send "hello?" packets every few minutes to ensure that the clients are still maintaining the connection on the network. It has been shown that the file server actually terminates the connection before the buffer would be filled with unopened messages. So, in the fifteen minutes that a server takes to confirm a lost transmission, twenty packets are not sent to the client. The client need respond to only one of these "hello" packets for the connection to be maintained. Another related factor is the protocol itself. Normal Ethernet protocol specifies that if a packet is sent without a response, the packet is resent. So, in the event the client receive buffer does fill up and a packet is lost because the receive buffer is full, the initiating station will resend the message. The chance of dropping a package is slim, and the chance of not recovering from a dropped package is even less.

One scenario that has not been mentioned is the scenario whereby the client goes to sleep in the midst of a large network operation. Even if this unlikely situation should occur, it will not affect the network connection. Protocol calls for a handshaking mechanism for all packets. A request is sent and a reply is received. In the event that the client enters into a low power state, the CPU will *request* the data less frequently, and thus receive data less frequently than if operating in a fully awake state. So as the effective frequency of the processor decreases so will the network bandwidth required by the operation in progress.

Another key consideration in the success of the tests was a simple matter of processor performance. Even when operating with a 90% duty cycle on STPCLK#, the SL Enhanced Intel486 processor compares very favorably with older processors still connected to many of today's networks.

In summary, it has been shown that the STPCLK features of Intel's SL Enhanced processors will not corrupt the Novell 3.11 Client Server LAN or lose network connections under normal circumstances.

6.0 NOVELL LITE TEST SUMMARY

6.1 Test Environment

The test environment can be summarized with the following list:

- Novell Lite 1.1 server and client software running on SL Enhanced Intel486 DX 33-MHz CPU
- Traffic generating station using Intel NetSight Professional
- One traffic monitoring station using Intel NetSight Professional
- One test station with SL Enhanced Intel486 DX2 66-MHz CPU configured as Novell Lite 1.1 server and client
- Coaxial cable connecting the four stations

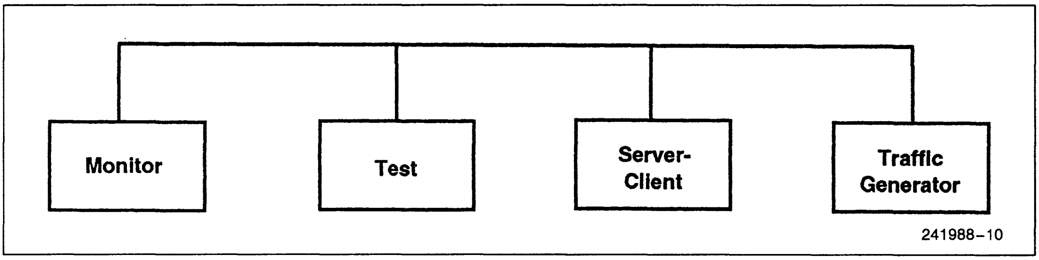


Figure 10. Test Environment

6.2 Test Description

This section describes the tests performed and provides explanations for some of the independent variables in the test. As mentioned in Section 4, the primary motive of the Novell Lite testing was to determine the performance implications of clock throttling PCs in a peer-to-peer networking environment.

The performance measurements were done by running two different tests while varying the period of STPCLK# assertion from 1 to 55 ms, the duty cycle of STPCLK# assertion from 50% to over 90%, and allowing and not allowing interrupts to deassert STPCLK#. The tests consisted of running two batch files which transferred files to and from the host machine. The first batch file transferred 2 files 10 times each. These files had files sizes of 41 Kbytes and 65 Kbytes. The second batch file transferred two large files having sizes of 130 Kbytes and 333 Kbytes twice to and from the host machine. Each of these tests were run on the seven Ethernet cards in the test suite. The results can be seen in Appendix C.

The range of the period from 1 to 55 ms was chosen because this is the minimum and maximum periods of typical implementations. The period was varied from 50% to 90% for maximum power savings. Data was also taken in the cases of allowing and not allowing interrupts to deassert STPCLK#, as obtaining performance results in both situations is necessary for a complete analysis.

Network Utilization

The network utilization was fixed at 22.8%. [The traffic generator was set at 25% but measured values demonstrate that the actual utilization was 22.8%.] The utilization was fixed because preliminary tests indicated that network utilization did not have a perceivable effect on network connection failures or on network performance. This percentage is considered a heavy traffic load but makes the test more representative of an actual network. The packets generated by the traffic generator were sent to random addresses, and had no effect on the performance of the individual cards other than using up network bandwidth. The bogus packets broadcasted on the network were 64 bytes long, and approximately 1,500 packets were sent per second.

Test Station Idle

Our test station was idle except when responding to the network requests generated by the test programs. This parameter is based on the initial assumption that the client is asleep because there is no activity.

6.3 Test Results

Performance

Performance slowdown is defined as the length of time the test took with clock throttling compared to the length of time the test took without clock throttling expressed as a percentage. A slowdown of 100% would mean that the tests took twice as long.

The highest performance levels were obtained by allowing interrupts to deassert STPCLK#. Using this method, the performance slowdown was never more than 60% of normal operating conditions. The best performance using STPCLK# at a 50% duty cycle was seen at a very short period (1 ms). The best performance seen in the 90% duty cycle occurs at the highest period tested, 50 ms. See Figures 11 and 12.

When interrupts were not allowed to deassert STPCLK#, the highest performance at a 90% duty cycle was on the shortest period tested. While the performance degradation (120%) is significant, this may be acceptable in environments where "sleeping" PCs are rarely accessed, or, of course, in non peer-to-peer networks where this situation will not occur.

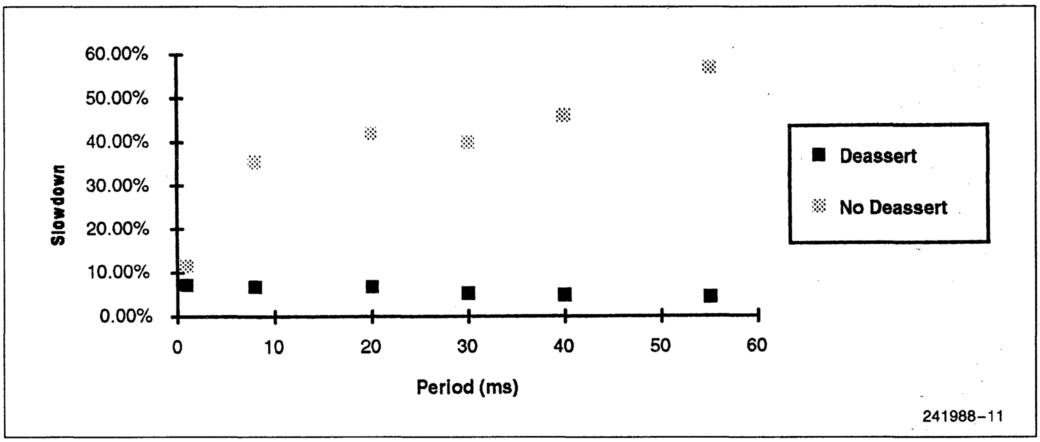


Figure 11. Performance Slowdown vs STPCLK# Period Relative to Peak Performance at 50% Duty Cycle (with and without STPCLK# Deassertion on INTR)

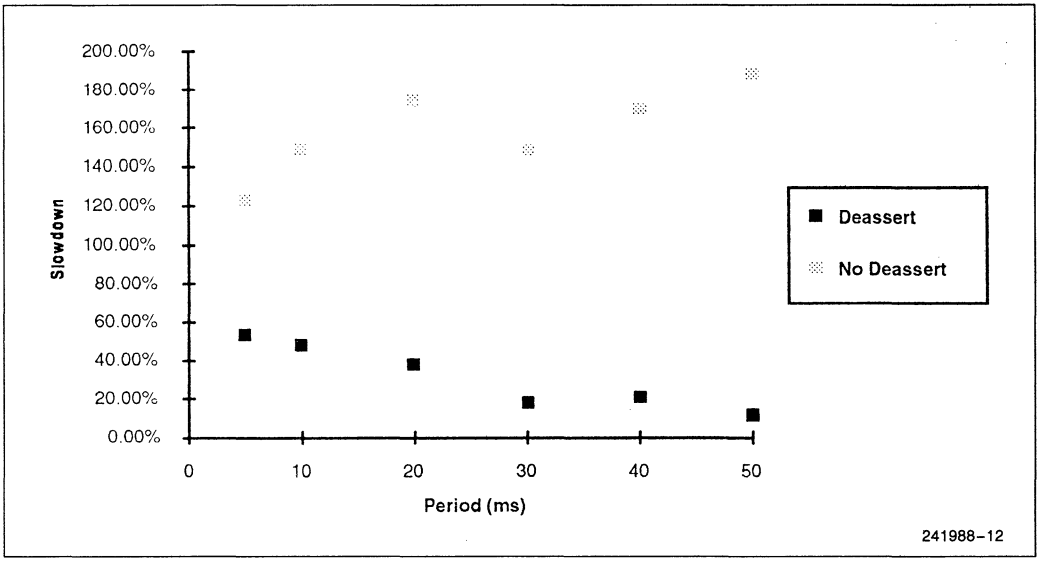


Figure 12. Performance Slowdown vs STPCLK# Period Relative to Peak Performance at 90% Duty Cycle (with and without STPCLK# Deassertion)

The knee in the graph at approximately 30 ms could be a function of the operating system's interaction with the I/O subsystem. As the period increases with the same duty cycle, the performance diminishes because the waiting time for the disk drive or LAN increases. For example many small periods would be more efficient than fewer bigger periods. At some point, however, the system reaches the point where it can complete an entire job (i.e. one file copy request) in the period (around 30 ms). At this point, the efficiency suddenly improves. Then once again, the efficiency diminishes as the period increases. The system can do 1.1 job, 1.2 job, etc. Theoretically the system would see another increase in performance at the two job interval (around 60 ms)

A connection test was also performed. The systems were all run for at least one hour with STPCLK# asserted over 92% of the time with a period of 55 ms. During this time, interrupts did not deassert STPCLK#. No systems failed and no connections were lost. A period of 55 ms was chosen because this is the maximum period that a system can be asleep without losing time from the real time clock. All the cards were tested in this environment and all the cards passed. An explanation of the Novell Lite protocol shows this to be a reasonable result.

If a Novell Lite machine attempts to access another machine and its first message is not acknowledged, it will retry 15 times at approximately 220 ms intervals. A retry is only counted towards the 15 if the request is successfully transmitted onto the network media (no collision occurred). If there is heavy traffic on the network, it may transmit at longer intervals than 220 ms due to collisions.

Novell Lite also uses a stop and wait protocol. The stop and wait protocol mandates that after one packet is transmitted, the next one will not be transmitted until an acknowledgment is received. If a packet is lost, the protocol will retransmit the packet (or request) after a time-out period. Standard Ethernet was never assumed to be an error-free transmission media, and procedures are already in place to handle lost packets. Therefore, even if the packet arrives successfully, and for some reason was lost in the destination machine (which did not appear to happen), the sending machine will retransmit the package since an acknowledgment was not received.

7.0 NOVELL PRINT SERVER TEST SUMMARY

7.1 Test Environment

The test environment can be summarized with the following list:

- Novell 3.11 File Server running on an Intel486 DX2 66-MHz CPU
- Traffic Generating Station using Intel Netsight Professional
- One Traffic Monitoring Station using Intel LanSight
- Two dummy clients
- One test station with an SL Enhanced Intel486 DX2 66-MHz CPU
  - — Configured as a Novell 3.1 network print server
  - — Intel EtherExpress LAN card
- Twisted Pair Ethernet Cable
- Hub
- HP DeskJet† 550C Printer

7.2 Test Description

The primary motive of this test is to show that there will be no significant loss in performance from a network workstation configured as a print server if it goes into power down mode. Testing was performed to compare normal network printing (STPCLK# disabled) with a worst case scenario in which STPCLK# is asserted and interrupts are disabled. The test consisted of printing a bitmap file and a text file from a client station to the test station configured as a print server. Testing was performed with the test station fully "awake" and "asleep". The "asleep" or worst case test consisted of a 25% network load (as did awake case), a STPCLK# duty cycle of 90%, and a STPCLK# period of 1 ms. The tests were performed using an Intel EtherExpress LAN card. The two file types used for testing were a 13 page 36K text file and a 308K bitmap file.

7.3 Test Results

Table 2. Test Results for Novell Print Server

File Type	STPCLK#	Duty Cycle	Period	File Print Time	Total Pages
36K text file	disabled	NA	NA	7.2 min	13
36K text file	enabled	90%	1ms	8.0 min	13
308K bit map	disabled	NA	NA	3.5 min	1
308K bit map	enabled	90%	1ms	3.8 min	1

NOTES:
STPCLK# disabled: CPU fully "awake".
STPCLK# enabled: CPU "asleep", 90% duty cycle, 1ms period, interrupts disabled.
Duty Cycle: 90% is the percentage time that the STPCLK# is asserted (low).
Period: STPCLK# period.
Network Load: actual network load as recorded by network monitor was 22.8%.
File Print Time: time to print entire file beginning at client.

7.4 Test Conclusions

As expected, there was no significant difference between file print times for a fully "awake" CPU and a "sleeping" CPU. The slight variations in time can be due to the Novell protocol itself. First, the file to be printed is transferred over the network to the network server; it is sent to the print server where it is stored in a print queue. From there it is spooled into the printer itself. Although a "sleeping" CPU transfers data much slower from the LAN card to memory, and from memory to the printer, it is sufficiently fast enough to keep up with the printer. The limiting factor here was not the CPU speed, but actually the printer speed. Also, note that this is a "worst" case, in which interrupts do NOT disable the STPCLK#. In summary, STPCLK# features do not have a significant impact on print server performance.

8.0 INTEL COMPATIBILITY VALIDATION LAB TEST SUMMARY

Intel's Compatibility Validation Lab is chartered with ensuring all Intel microprocessors are Intel compatible. To this end, they extensively test all new Intel CPUs in a variety of environments. Passing their tests is an indication that the device under test is Intel compatible. A PC equipped with the daughter card to enable clock throttling was submitted to the CV Lab for testing. In effect, the testing was to determine whether or not a clock throttled CPU was Intel compatible.

For logistic reasons, the CV Lab performs its workstation tests over a network. CV Lab's workstation tests consist of running various industry application packages. The test suites emulate typical user activity in applications such as Microsoft Excel† , Word† , etc. An Energy Star system would normally be in a full on state during such activity; however, for study purposes this testing was done with a clock throttled system to investigate any issues involved with interaction between a clock throttled CPU and heavy network traffic.

Nine standard workstation tests were performed on a PC equipped with the daughter card to enable clock throttling. The following "typical" STPCLK# implementation was chosen.

1. Period equal to 8 ms
2. Duty Cycle equal to 90%
3. STPCLK# was deasserted for remainder of period on INTR

All tests were run successfully, and no network failures were observed. In conclusion, the CV Lab testing did not indicate any incompatibility between the STPCLK# clock throttled PC and the network.

9.0 RECOMMENDATIONS FOR CLOCK THROTTLING

Clock throttling provides a clean solution for hardware power management of the CPU and memory subsystem. Clock throttling offers an easy way to design a system which can enter a low power state (less than 30W) during periods of inactivity, while still providing adequate functionality to maintain a network connection and even respond to printing or data requests. Hardware power management through clock throttling means that the power managed system will not have to depend on software drivers or a particular operating system to meet Energy Star compliance. This offers the OEM greater flexibility in the system configuration, and frees the end user from concerns over software upgrades.

It is important in an AT† type personal computer that the real time interrupt (once every 55 ms) not be missed. This can be guaranteed by either choosing a clock throttling period of less than 55 ms, or by momentarily deasserting STPCLK# on the real time interrupt.

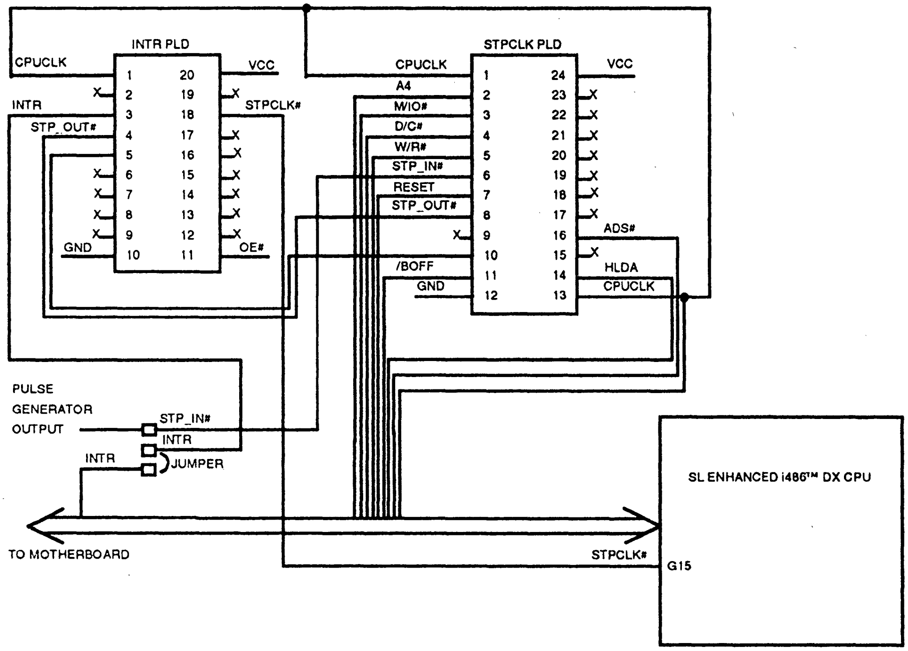
Clock throttling enables the system designer to choose the level of power savings necessary for a particular implementation. It has been demonstrated that increasing the STPCLK# duty cycle dramatically affects the power requirements of the system, with little impact on performance of "sleeping systems". If the performance of "sleeping systems" is a major concern, then momentarily waking for system interrupts can obviate this degradation to a great extent. If the system will be awakened for the remainder of the STPCLK# period, as was done in the test daughter card, then the period of STPCLK# clock throttling will affect energy savings.

An alternative method of STPCLK# clock throttling is with a variable period driven by external events. Basically STPCLK# is asserted until an external event (such as interrupt) occurs. At this time STPCLK# is deasserted for a programmable period (2 ms–4 ms), and subsequently reasserted. This dynamic clock throttling scenario comes very close to always operating the CPU in a low power state, except for when there is actually work to do (such as service the interrupt).

Finally, it should be noted that STPCLK# testing revealed no inconsistencies with clock throttling an Intel486 SL Enhanced CPU in a network environment, and therefore its proven network compatibility should be considered when judging the merits of using clock throttling as part of a power management program.

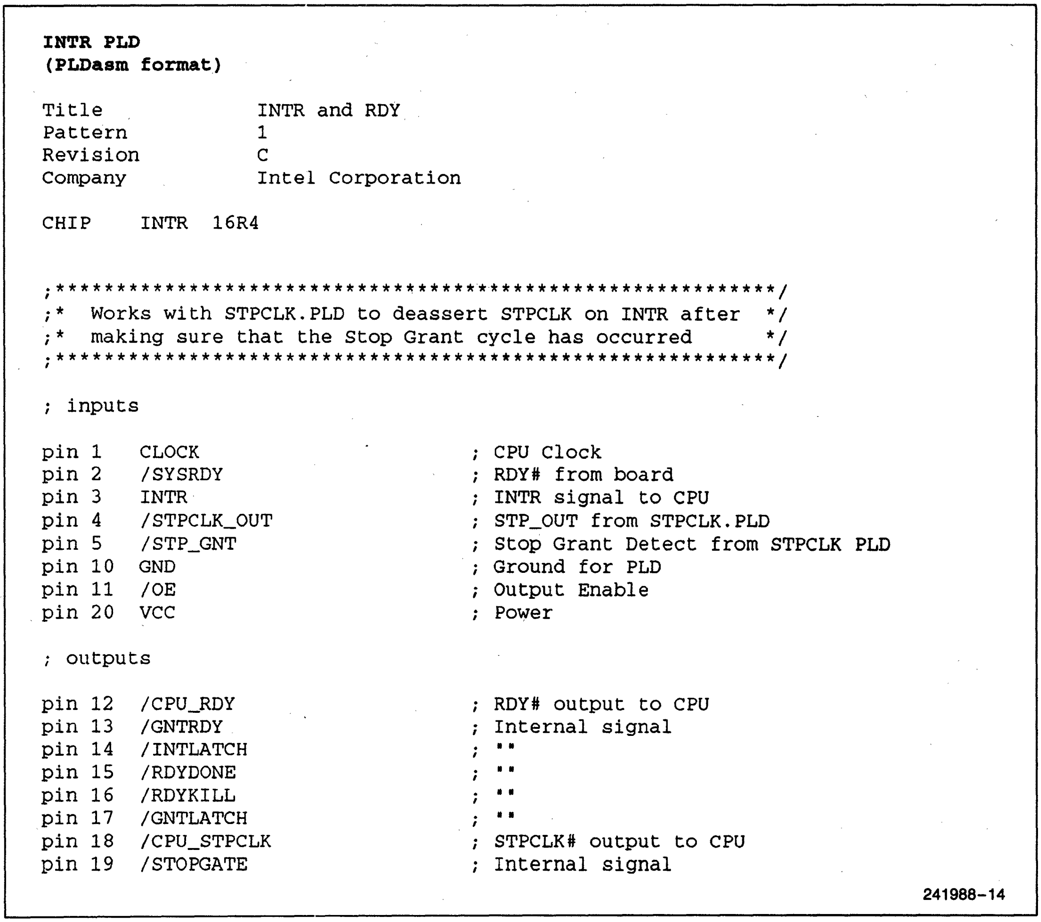
APPENDIX A
STPCLK# TEST CIRCUITRY

STPCLK# Daughter Card Schematic



241988-13

Figure A-1. STPCLK# Test Circuitry



INTR PLD
(PLDasm format)

```
Title           INTR and RDY
Pattern         1
Revision        C
Company         Intel Corporation

CHIP    INTR  16R4


;***********************************************************/
;*  Works with STPCLK.PLD to deassert STPCLK on INTR after  */
;*  making sure that the Stop Grant cycle has occurred      */
;***********************************************************/

; inputs

pin 1   CLOCK                        ; CPU Clock
pin 2   /SYSRDY                      ; RDY# from board
pin 3   INTR                         ; INTR signal to CPU
pin 4   /STPCLK_OUT                  ; STP_OUT from STPCLK.PLD
pin 5   /STP_GNT                     ; Stop Grant Detect from STPCLK PLD
pin 10  GND                          ; Ground for PLD
pin 11  /OE                          ; Output Enable
pin 20  VCC                          ; Power

; outputs

pin 12  /CPU_RDY                     ; RDY# output to CPU
pin 13  /GNTRDY                      ; Internal signal
pin 14  /INTLATCH                    ; ""
pin 15  /RDYDONE                     ; ""
pin 16  /RDYKILL                     ; ""
pin 17  /GNTLATCH                    ; ""
pin 18  /CPU_STPCLK                  ; STPCLK# output to CPU
pin 19  /STOPGATE                    ; Internal signal
```

241988-14

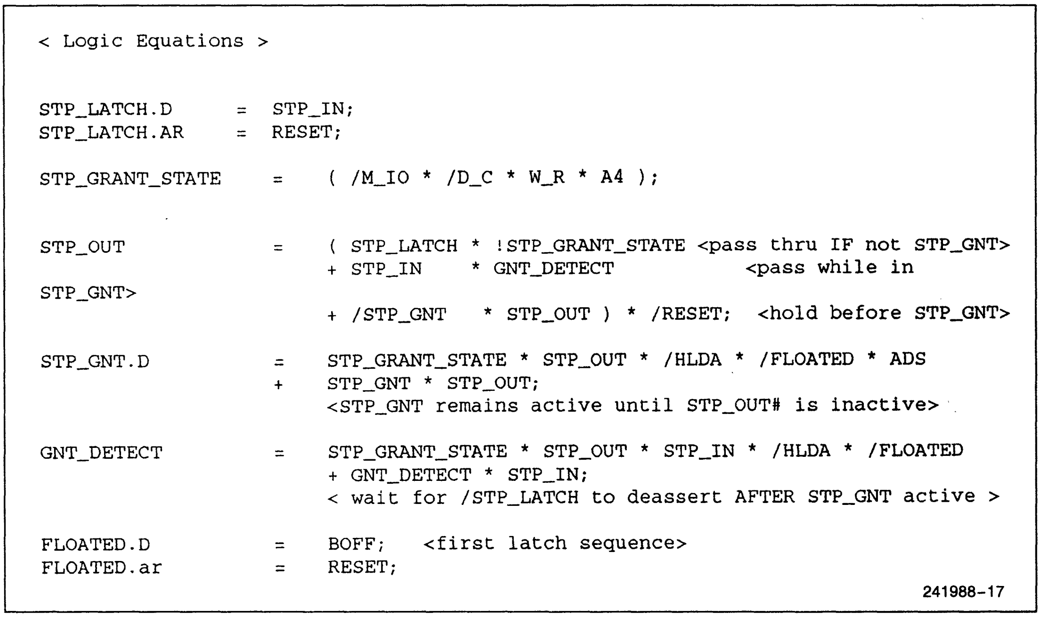
```
EQUATIONS

CPU_RDY.TRST        =    VCC

STOPGATE            :=   STPCLK_OUT*/INTLATCH + STOPGATE*STPCLK_OUT
                         + STOPGATE*/RDYDONE

GNTLATCH            :=   STP_GNT              ; GENERATE RDY AFTER STOP
RDYKILL             :=   GNTLATCH             ; GRANT FOR ONE BUS CYCLE

GNTRDY              :=   GNTLATCH*/RDYKILL

CPU_RDY             =    SYSRDY

RDYDONE             :=   RDYKILL + STOPGATE*RDYDONE   ; HOLD RDYDONE TILL
                                                      ; STPCLK GOES AWAY

INTLATCH            :=   INTR + STOPGATE*INTLATCH ; HOLD INTLATCH TILL
                                                  ; STPCLK GOES AWAY

CPU_STPCLK          :=   STOPGATE*/INTLATCH + STOPGATE*/RDYDONE

;end of equations
```

241988-15

STPCLK PLD
(CUPL Format)

```
Name         STPCLK.PLD;
Partno       85C060  ;
Revision     05;
Company      Intel Corporation, Inc.;
Assembly     CUPL;
Location     U00;
Device       EP600;         /* PLDXXXXX Intel 85C060 */

/*************************************************************/
/* Hardware implementation assert minimum STPCLK# duration    */
/* until CPU acknowledges with STOP GRANT CYCLE.              */
/*************************************************************/

< Inputs >
pin 1        = CPUCLK  ;   <pin #1 and #13 connected together>
pin 2        = A4      ;
pin 3        = M_IO    ;
pin 4        = D_C     ;
pin 5        = W_R     ;
pin 6        = /STP_IN ;   < asynchronous input from system >
pin 7        = RESET   ;
pin 11       = /BOFF   ;   < omit if not used >
pin 14       = HLDA    ;
pin 16       =  !ADS;

< Outputs >
pin 8        = /STP_OUT   ; < STPCLK# output to CPU >
pin 9        =  STP_LATCH ;
pin 10       = /STP_GNT   ;
pin 15       =  GNT_DETECT;
pin 19       =  FLOATED;
```

241988-16



```
< Logic Equations >


STP_LATCH.D       =  STP_IN;
STP_LATCH.AR      =  RESET;

STP_GRANT_STATE      =    ( /M_IO * /D_C * W_R * A4 );


STP_OUT              =    ( STP_LATCH * !STP_GRANT_STATE <pass thru IF not STP_GNT>
                          + STP_IN    * GNT_DETECT           <pass while in
STP_GNT>
                          + /STP_GNT   * STP_OUT ) * /RESET;  <hold before STP_GNT>

STP_GNT.D            =    STP_GRANT_STATE * STP_OUT * /HLDA * /FLOATED * ADS
                     +    STP_GNT * STP_OUT;
                          <STP_GNT remains active until STP_OUT# is inactive>

GNT_DETECT           =    STP_GRANT_STATE * STP_OUT * STP_IN * /HLDA * /FLOATED
                          + GNT_DETECT * STP_IN;
                          < wait for /STP_LATCH to deassert AFTER STP_GNT active >

FLOATED.D            =    BOFF;   <first latch sequence>
FLOATED.ar           =    RESET;
```

241988-17

APPENDIX B
SYSTEM POWER MEASUREMENTS WITH CLOCK THROTTLING

Period = 40 ms
STPCLK# Not Deasserted on Interrupts

STPCLK# Asserted (ms)	Duty Cycle	Current (mAmps)	Power (Watts)
0	0%	2.60	13.00
5	12.5%	2.44	12.20
10	25%	2.31	11.57
15	37.5%	2.14	10.68
20	50%	1.98	9.90
25	62.5%	1.84	9.20
30	75%	1.69	8.45
35	87.5%	1.55	7.75

50% Duty Cycle
STPCLK# Deasserted on Interrupts

Period (ms)	DC Current (mAmps)	Power (Watts)
1	2.02	10.09
5	2.05	10.23
10	2.05	10.25
15	2.06	10.31
20	2.07	10.37
25	2.09	10.45
30	2.13	10.64
35	2.13	10.65
40	2.14	10.68

90% STPCLK# Asserted
STPCLK# Deasserted on interrupts

Period (ms)	DC Current (mAmps)	Power (Watts)
1	1.52	7.61
5	1.56	7.78
10	1.60	8.01
15	1.65	8.25
20	1.70	8.49
25	1.74	8.70
30	1.79	8.95
35	1.83	9.15
40	1.88	9.41

APPENDIX C
NOVELL LITE
COMPLETE TEST RESULTS

Network Card: Ansel 2000
Type: AUI/BNC/TPI
Extended Test: Passed

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
	0.00%	N/A	N/A	N/A	12.74	7.41
1	50.00%	0.50	0.50	no	15.54	9.77
1	50.00%	0.50	0.50	yes	13.18	9.72
8	50.00%	4.00	4.00	no	16.31	10.21
8	50.00%	4.00	4.00	yes	13.12	7.68
20	50.00%	10.00	10.00	no	18.01	11.53
20	50.00%	10.00	10.00	yes	13.73	7.85
20	80.00%	16.00	4.00	no	28.72	19.88
20	80.00%	16.00	4.00	yes	13.45	7.96
30	50.00%	15.00	15.00	no	17.74	11.09
30	50.00%	15.00	15.00	yes	13.07	7.85
30	86.67%	26.00	4.00	no	46.57	32.46
30	86.67%	26.00	4.00	yes	13.45	8.45
30	53.33%	16.00	14.00	no	23.34	14.33
30	53.33%	16.00	14.00	yes	13.56	7.85
40	50.00%	20.00	20.00	no	18.34	11.58
40	50.00%	20.00	20.00	yes	13.4	7.96
40	90.00%	36.00	4.00	no	58.6	40.31
40	90.00%	36.00	4.00	yes	14.06	7.96
55	50.00%	27.50	27.50	no	18.56	11.47
55	50.00%	27.50	27.50	yes	13.4	7.63
55	92.73%	51.00	4.00	no	116.33	93.09
55	92.73%	51.00	4.00	yes	13.4	7.79

Network Card: **Ansel 2100**
Type: **AUI/BNC/TPI**
Extended Test: **Passed**

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
	0.00%	N/A	N/A	N/A	12.24	7.19
1	50.00%	0.50	0.50	no	13.12	7.85
1	50.00%	0.50	0.50	yes	13.01	7.52
8	50.00%	4.00	4.00	no	15.59	9.66
8	50.00%	4.00	4.00	yes	13.12	7.46
20	50.00%	10.00	10.00	no	16.42	12.027
20	50.00%	10.00	10.00	yes	13.01	7.3
20	80.00%	16.00	4.00	no	24.88	16.64
20	80.00%	16.00	4.00	yes	13.07	7.57
30	50.00%	15.00	15.00	no	17.13	10.32
30	50.00%	15.00	15.00	yes	13.18	7.36
30	86.67%	26.00	4.00	no	34.21	24.27
30	86.67%	26.00	4.00	yes	12.96	7.41
30	53.33%	16.00	14.00	no	18.29	10.76
30	53.33%	16.00	14.00	yes	12.96	7.36
40	50.00%	20.00	20.00	no	18.34	10.98
40	50.00%	20.00	20.00	yes	12.57	7.25
40	90.00%	36.00	4.00	no	45.64	31.25
40	90.00%	36.00	4.00	yes	12.79	7.57
55	50.00%	27.50	27.50	no	20.65	12.85
55	50.00%	27.50	27.50	yes	12.52	7.36
55	92.73%	51.00	4.00	no	77.06	52.06
55	92.73%	51.00	4.00	yes	12.85	7.46

Network Card: Eagle NE2000 Plus 3
Type: AUI/BNC/TPI
Extended Test: Passed Eagle

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
	0.00%	N/A	N/A	N/A	12.68	7.63
1	50.00%	0.50	0.50	no	13.23	8.07
1	50.00%	0.50	0.50	yes	12.63	8.40
8	50.00%	4.00	4.00	no	17.30	10.98
8	50.00%	4.00	4.00	yes	14.33	8.51
20	50.00%	10.00	10.00	no	15.65	9.80
20	50.00%	10.00	10.00	yes	13.56	8.18
20	80.00%	16.00	4.00	no	21.53	14.22
20	80.00%	16.00	4.00	yes	15.04	9.17
30	50.00%	15.00	15.00	no	16.86	10.71
30	50.00%	15.00	15.00	yes	13.45	8.01
30	86.67%	26.00	4.00	no	27.51	19.60
30	86.67%	26.00	4.00	yes	14.72	10.21
30	53.33%	16.00	14.00	no	16.75	10.76
30	53.33%	16.00	14.00	yes	13.51	8.01
40	50.00%	20.00	20.00	no	16.25	10.10
40	50.00%	20.00	20.00	yes	13.67	8.01
40	90.00%	36.00	4.00	no	27.90	19.11
40	90.00%	36.00	4.00	yes	14.55	9.28
55	50.00%	27.50	27.50	no	17.19	11.14
55	50.00%	27.50	27.50	yes	13.12	8.01
55	92.73%	51.00	4.00	no	34.27	24.38
55	92.73%	51.00	4.00	yes	15.37	9.22

Network Card: Intel EtherExpress 16C
Type: AUI/BNC/TPI
Extended Test: Passed

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
	0.00%	N/A	N/A	N/A	12.52	7.41
1	50.00%	0.50	0.50	no	13.4	8.12
1	50.00%	0.50	0.50	yes	12.9	7.85
8	50.00%	4.00	4.00	no	15.65	9.94
8	50.00%	4.00	4.00	yes	13.4	7.74
20	50.00%	10.00	10.00	no	17.13	10.87
20	50.00%	10.00	10.00	yes	13.4	7.74
20	80.00%	16.00	4.00	no	24.49	16.69
20	80.00%	16.00	4.00	yes	13.45	8.01
30	50.00%	15.00	15.00	no	17.13	10.49
30	50.00%	15.00	15.00	yes	13.07	7.74
30	86.67%	26.00	4.00	no	35.26	24.82
30	86.67%	26.00	4.00	yes	13.29	7.96
30	53.33%	16.00	14.00	no	17.63	10.82
30	53.33%	16.00	14.00	yes	13.23	7.85
40	50.00%	20.00	20.00	no	18.23	11.36
40	50.00%	20.00	20.00	yes	13.56	7.68
40	90.00%	36.00	4.00	no	45.86	32.07
40	90.00%	36.00	4.00	yes	13.45	7.96
55	50.00%	27.50	27.50	no	20.26	12.68
55	50.00%	27.50	27.50	yes	12.9	8.01
55	92.73%	51.00	4.00	no	89.58	68.54
55	92.73%	51.00	4.00	yes	13.12	8.07

Network Card: Kingston KNE2121
Type: BNC/TPI
Extended Test: Passed

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
	0.00%	N/A	N/A	N/A	12.24	7.14
1	50.00%	0.50	0.50	no	13.18	7.9
1	50.00%	0.50	0.50	yes	12.85	7.57
8	50.00%	4.00	4.00	no	16.25	10.16
8	50.00%	4.00	4.00	yes	13.01	7.74
20	50.00%	10.00	10.00	no	17.08	10.87
20	50.00%	10.00	10.00	yes	13.07	7.63
20	80.00%	16.00	4.00	no	26.03	18.45
20	80.00%	16.00	4.00	yes	13.18	7.74
30	50.00%	15.00	15.00	no	17.08	10.54
30	50.00%	15.00	15.00	yes	13.12	7.63
30	86.67%	26.00	4.00	no	36.47	24.93
30	86.67%	26.00	4.00	yes	12.96	7.52
30	53.33%	16.00	14.00	no	17.68	11.14
30	53.33%	16.00	14.00	yes	12.85	7.68
40	50.00%	20.00	20.00	no	17.63	11.04
40	50.00%	20.00	20.00	yes	12.79	7.57
40	90.00%	36.00	4.00	no	45.31	32.24
40	90.00%	36.00	4.00	yes	13.29	8.07
55	50.00%	27.50	27.50	no	19.44	12.19
55	50.00%	27.50	27.50	yes	12.85	7.46
55	92.73%	51.00	4.00	no	71.67	51.3
55	92.73%	51.00	4.00	yes	13.73	7.68

Netword Card: SMC Elite 16C Ultra
Type: AUI/BNC/TPI
Extended Test: Passed

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
1	50.00%	0.50	0.50	no	14.61	8.07
1	50.00%	0.50	0.50	yes	13.23	7.85
8	50.00%	4.00	4.00	no	16.14	10.43
8	50.00%	4.00	4.00	yes	13.51	7.96
20	50.00%	10.00	10.00	no	17.08	11.04
20	50.00%	10.00	10.00	yes	13.73	8.23
20	80.00%	16.00	4.00	no	23.06	19.6
20	80.00%	16.00	4.00	yes	14.44	8.51
30	50.00%	15.00	15.00	no	16.69	10.32
30	50.00%	15.00	15.00	yes	13.51	7.74
30	86.67%	26.00	4.00	no	37.45	26.47
30	86.67%	26.00	4.00	yes	14.33	8.78
30	53.33%	16.00	14.00	no	12.9	10.1
30	53.33%	16.00	14.00	yes	13.45	7.85
40	50.00%	20.00	20.00	no	18.01	11.36
40	50.00%	20.00	20.00	yes	13.34	7.9
40	90.00%	36.00	4.00	no	48.05	28.78
40	90.00%	36.00	4.00	yes	13.67	8.07
55	50.00%	27.50	27.50	no	19.88	12.24
55	50.00%	27.50	27.50	yes	13.34	8.29
55	92.73%	51.00	4.00	no	59.75	46.13
55	92.73%	51.00	4.00	yes	13.62	8.45

Netword Card: 3Com Etherlink III
Type: AUI/BNC/TPI
Extended Test: Passed

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
	0.00%	N/A	N/A	N/A	13.07	7.41
1	50.00%	0.50	0.50	no	14.17	8.23
1	50.00%	0.50	0.50	yes	13.62	7.96
8	50.00%	4.00	4.00	no	17.9	10.98
8	50.00%	4.00	4.00	yes	13.73	8.07
20	50.00%	10.00	10.00	no	17.24	10.93
20	50.00%	10.00	10.00	yes	13.45	8.07
20	80.00%	16.00	4.00	no	26.36	17.63
20	80.00%	16.00	4.00	yes	13.73	8.18
30	50.00%	15.00	15.00	no	17.41	10.71
30	50.00%	15.00	15.00	yes	13.18	7.79
30	86.67%	26.00	4.00	no	39.49	28.56
30	86.67%	26.00	4.00	yes	13.62	8.12
30	53.33%	16.00	14.00	no	17.57	10.82
30	53.33%	16.00	14.00	yes	13.45	7.9
40	50.00%	20.00	20.00	no	17.63	11.09
40	50.00%	20.00	20.00	yes	13.18	7.52
40	90.00%	36.00	4.00	no	56.29	37.84
40	90.00%	36.00	4.00	yes	13.18	7.57
55	50.00%	27.50	27.50	no	17.85	10.82
55	50.00%	27.50	27.50	yes	13.12	7.57
55	92.73%	51.00	4.00	no	56.79	37.84
55	92.73%	51.00	4.00	yes	13.34	7.74

Averages

Stop Clock Period (ms)	Duty Cycle	STPCLK Asserted (ms)	STPCLK Deasserted (ms)	Interrupts Enabled	Test 1	Test 2
	0.00%	N/A	N/A	N/A	12.61	7.36
1	50.00%	0.50	0.50	no	13.89	8.29
1	50.00%	0.50	0.50	yes	13.06	8.12
8	50.00%	4.00	4.00	no	16.45	10.34
8	50.00%	4.00	4.00	yes	13.46	7.88
20	50.00%	10.00	10.00	no	16.94	11.01
20	50.00%	10.00	10.00	yes	13.42	7.86
20	80.00%	16.00	4.00	no	25.01	17.59
20	80.00%	16.00	4.00	yes	13.77	8.16
30	50.00%	15.00	15.00	no	17.15	10.60
30	50.00%	15.00	15.00	yes	13.23	7.73
30	86.67%	26.00	4.00	no	36.71	25.87
30	86.67%	26.00	4.00	yes	13.62	8.35
30	53.33%	16.00	14.00	no	17.74	11.25
30	53.33%	16.00	14.00	yes	13.29	7.79
40	50.00%	20.00	20.00	no	17.78	11.07
40	50.00%	20.00	20.00	yes	13.22	7.70
40	90.00%	36.00	4.00	no	46.81	31.66
40	90.00%	36.00	4.00	yes	13.57	8.07
55	50.00%	27.50	27.50	no	19.12	11.91
55	50.00%	27.50	27.50	yes	13.04	7.76
55	92.73%	51.00	4.00	no	72.21	53.33
55	92.73%	51.00	4.00	yes	13.63	8.06

APPENDIX D
TYPICAL NETWORK INTERFACE CARD POWER REQUIREMENTS

Card (Drivers Loaded)	Current (Amps)	Power (Watts)
Ansel 2000	0.33	1.65
Ansel 2100	0.54	2.7
Eagle NE2000	0.13	0.65
Intel EtherExpress	0.48	2.4
Kingston KNE2121	0.41	2.05
SMC Elite16C	0.46	2.3
3Com Etherlink III	0.09	0.45
Intel 82595	0.099	0.495

NOTE:
It is important to properly understand the results in the above table. The network cards with a power consumption of approximately 2.5W represent a class of cards with older generation technology. The NICs with power consumption below 1W represent the newer integrated single chip lan controllers. For Energy Star compliant systems, 2.5W could represent as much as 17% of the power budget (depending on the efficiency of the power supply). So highly optimized systems with sufficient power budget could easily accommodate any network card. A system with a smaller marginal power budget may wish to consider a NIC with the newer technology.

intel®

AP-505

APPLICATION NOTE

Picking Up the Pace: Designing the IntelDX4™ Processor into Intel486™ Processor Based Desktop Systems

DAVID HARRIMAN
INTEL TECHNICAL MARKETING

July 1994

Order Number: 242034-001

Picking Up the Pace: Designing the IntelDX4™ Processor into Intel486™ Processor Based Desktop Systems

CONTENTS PAGE

1.0 INTRODUCTION

The IntelDX4™ processor is the newest and highest performing member of the Intel486™ processor family. At internal speeds of up to 100 MHz, the IntelDX4 Processor is the fastest 486, designed for users who want the best value in 486 desktop computing today. With its larger 16K internal cache size and improved core speed, the 100 MHz IntelDX4 processor outperforms the 66 MHz IntelDX2™ processor by as much as 50%. Intel's unique 5V tolerant input buffers make this performance improvement achievable with minimal modifications to existing IntelDX2 processor based desktop designs. This document provides a straightforward process for updating your IntelDX2 processor based desktop system design to match the potential of the IntelDX4 processor, while maintaining system design compatibility with previous generations of the Intel486 processor family.

The IntelDX4 processor is based on proven Intel486 technology and is compatible with the huge installed base of over 50,000 applications written for the Intel Architecture. To ensure end-user investment protection, IntelDX4 processor based systems should be verified for upgradability to a future Pentium™ OverDrive™ Processor. This application note provides clear guidance on how to prepare your system for Pentium OverDrive Processor verification.

Intel appreciates your interest in the IntelDX4 processor. This document has been developed to allow you to minimize your investment in development time and bring a reliable design to market quickly.

NOTE:
Important recommendations that should be carefully addressed for a reliable design are highlighted in bold. These recommendations must be followed precisely to help ensure that your design will be ready for manufacture with minimal redesign.

The checklist and flowchart in Appendices A and B will help you quickly confirm that you have incorporated all of the critical design recommendations. By using these tools you can be confident that your system is ready to meet the new standard for 486 computing set by the 100 MHz IntelDX4 processor.

2.0 HARDWARE RECOMMENDATIONS

The following design recommendations cover upgrading an existing IntelDX2 processor based desktop system design to support the IntelDX4 processor. It is assumed that the existing IntelDX2 processor based design supports both the processor and its corresponding Pentium OverDrive processor using a single socket. There are two categories of recommendations included: those covering features which are new to the IntelDX4 processor, and those which apply to the IntelDX2 processor as well as the IntelDX4 processor, but have renewed importance for the IntelDX4 processor.

Table 1. Hardware Differences between the IntelDX4™ Processor and the IntelDX2™ Processor and Their Respective Pentium™ OverDrive™ Processors

	IntelDX4™ Processor	IntelDX2™ Processor	Future Pentium™ OverDrive™ Processor for the IntelDX4™ Processor	Pentium™ OverDrive™ Processor for the IntelDX2™ Processor
Max. Internal Speed	100 MHz	66 MHz	*	*
Supply Voltage	3.3V	5.0V	3.3V	5.0V
Cache Size	16 Kbytes	8 Kbytes	*	*
168-Pin PGA Pin R17 (S18**)	CLKMUL	INC	CLKMUL	INC
168-Pin PGA Pin S4 (T5**)	VOLDET	NC	VOLDET	INC
168-Pin PGA Pin J1 (K2**)	V_{CC5}	V_{CC}	V_{CC5}	V_{CC}
Pentium OverDrive Processor Socket Pins J1, K1 and L1	N.A.	N.A.	5.0V for Fan/Heatsink	V_{CC}

NOTES:
* Contact your local Intel representative for details.
** Pentium OverDrive Processor socket pin number.

Table 1 summarizes the differences between the IntelDX4 processor and the IntelDX2 processor. Note that the operation of SMM and STPCLK are identical to the SL Enhanced IntelDX2 processor. Also note that the future Pentium OverDrive processor for the IntelDX4 processor is different from the Pentium OverDrive processor for the IntelDX2 processor.

2.1 Power Supply

The new features of the IntelDX4 processor bring with them some new requirements on the processor power supply. Since the IntelDX2 processor is a 5V part, existing IntelDX2 processor based designs typically run all system logic at 5V. The IntelDX4 processor is a 3.3V part, so it is necessary to modify the processor power supply to provide this voltage. Care was taken in the design of the IntelDX4 processor to ensure that a single system board could be designed which would function with any Intel486 processor, while getting maximum performance from the IntelDX4 processor.

2.1.1 PROVIDING 3.3V IN A 5V SYSTEM

In most system board designs, the 5V system power supply is routed to the components on the board through a dedicated board layer. With the requirement of a new 3.3V supply for the IntelDX4 processor, it is not necessary to add a completely new power supply layer to the circuit board, as it is possible to create a 3.3V "island" around the processor in the existing power supply plane. Figure 1 shows a recommended "island" layout. Note the connection from the 5V plane to the V_{CC5P} pins, which will power the integrated fan/heatsink in the future Pentium OverDrive processor. The IntelDX4 processor's 5V tolerant input buffers and TTL compatible outputs allow the processor to interface with existing TTL compatible external logic without requiring extra components Thus, the processor can run at 3.3V while the system logic runs at 5V. The "island" needs to be large enough to include the processor, the required power supply decoupling capacitance (see section 2.2), and the necessary connection to the 3.3V source. To minimize signal degradation, the gap between the 3.3V "island" and the 5V plane should be kept small. A typical gap size is about 0.02 inches. Minimize the number of traces routed across the power plane gap, since each crossing introduces signal degradation due to the impedance discontinuity that occurs at the gap. For traces that must cross the gap, route them on the side of the board next to the ground plane to reduce or eliminate the signal degradation caused by crossing the gap. If this is not possible, route the trace to cross the gap at a right angle (90 degrees).

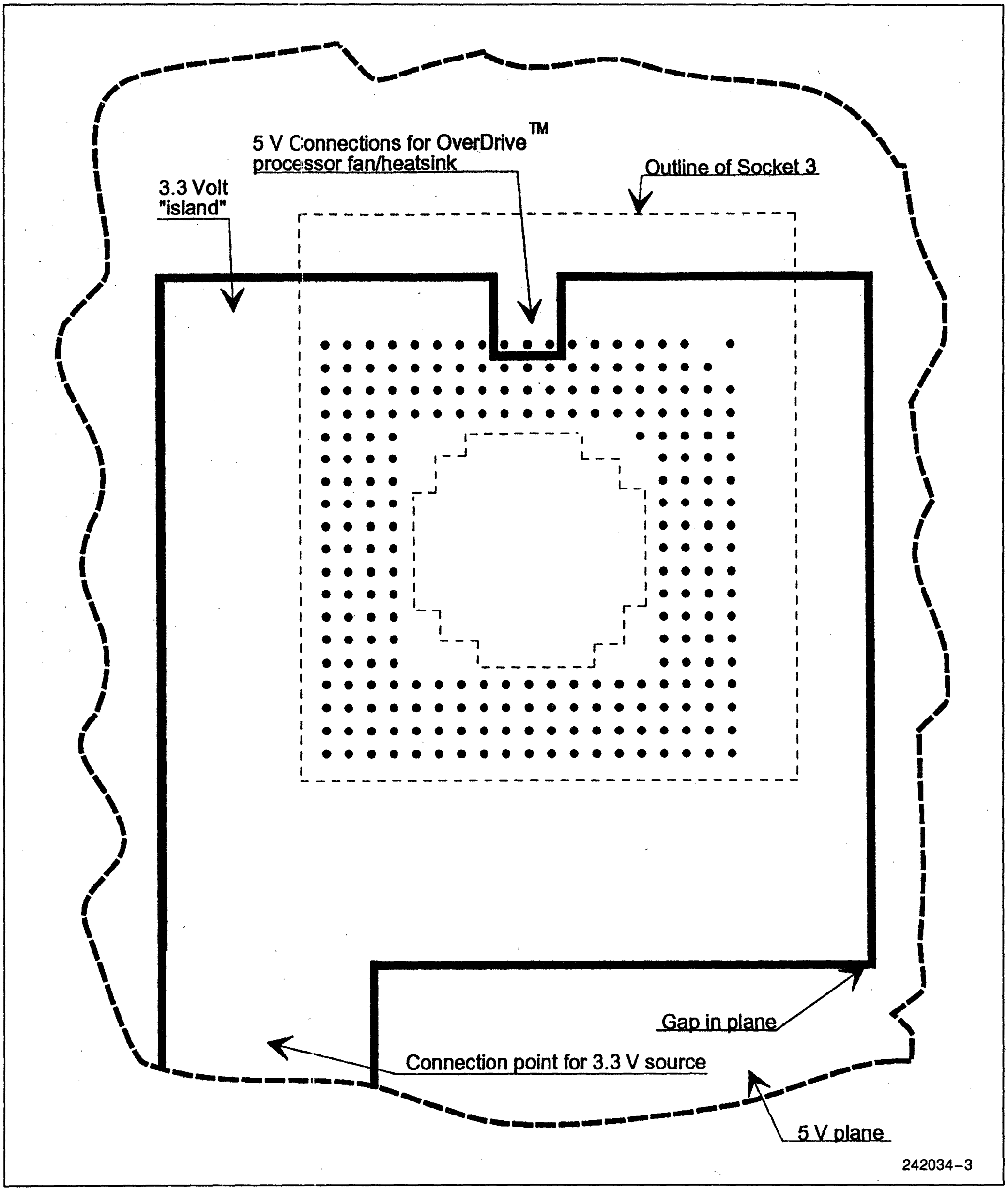
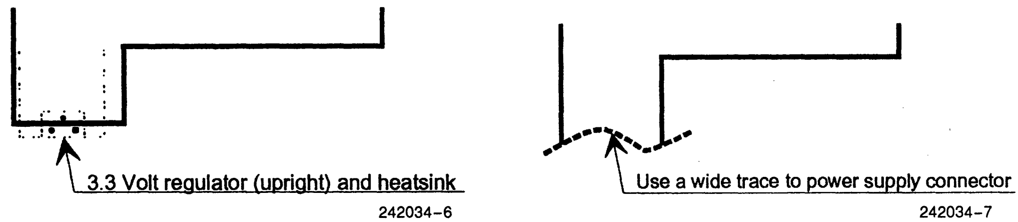

5 V Connections for OverDrive™ processor fan/heatsink
3.3 Volt "island"
Outline of Socket 3
Gap in plane
Connection point for 3.3 V source
5 V plane
242034-3

Figure 1. Creating a Power "Island"


3.3 Volt regulator (upright) and heatsink
242034-6
Use a wide trace to power supply connector
242034-7

3.3V Supply Using Linear Regulator

3.3V Supply Using System Power Supply

Figure 2. Recommended Power Supply Connection Layout

2.1.2 CHOOSING A POWER SOURCE

The three principle concerns which must be addressed when selecting a power source are maximum and minimum load current requirements, and response time. The processor power supply must be able to maintain correct voltage regulation at current levels below 0.5 mA for the IntelDX4 processor in the Stop Clock State, and up to the maximum current of 3.0A for the future Pentium OverDrive processor. The power saving technology in Intel SL Enhanced processors, including the IntelDX4 processor, will cause the processor to switch to very low power levels during normal operation, even if external power management is not used. For example, executing a HALT instruction will cause the IntelDX4 processor to enter the Auto HALT Power Down State, which will cause a significant reduction in the current consumption of the processor in as little as 100 ns. The transition from HALT to the Normal State will cause current consumption to return to the normal levels in a similarly short period of time. The processor power supply must be able to maintain correct voltage regulation during these transitions.

There are basically two options for supplying 3.3V to the processor, either adding a 3.3V tap to the primary system power supply or using on-board secondary regulation to derive 3.3V from the 5V system power supply.

For on-board secondary regulation, a linear voltage regulator will perform adequately for most desktop and server designs. If low heat or power dissipation is a design goal, the higher complexity and cost of a switching regulator may be warranted. Switching regulators offer better efficiency, thereby lowering regulator power consumption and heat. See section 2.4 for related thermal considerations.

Figure 2 shows recommended layouts for power supply or linear regulator connection to the 3.3V "island."

Appendix C includes a list of possible vendors for power supplies and voltage regulators.

2.1.3 POWER SUPPLY SELECTION FOR FLEXIBLE MOTHERBOARDS

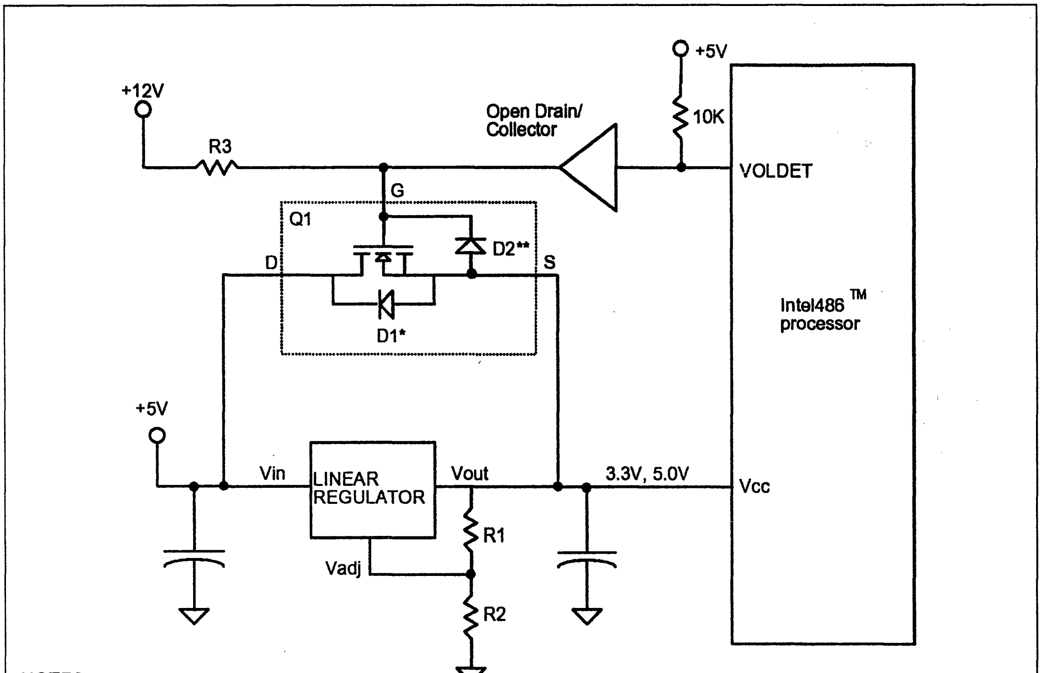
Using the voltage detect sense feature of the IntelDX4 processor, you may design a flexible system motherboard which will automatically use the proper processor voltage for an IntelDX4 processor or a different Intel486 processor. It is also possible to make the selection of processor voltage an option during system board assembly.

2.1.3.1 VOLDET Automatic Voltage Select Circuit Option

By sampling the VOLDET pin at power up, system boards can automatically select the processor power supply voltage, enabling a design that may use the IntelDX4 processor or a 5V Intel486 processor without jumpers or assembly time changes. The VOLDET pin is only present in the PGA package version of the IntelDX4 processor. This pin, which is an NC (No Connect) on previous Intel486 family processors, is connected internally to V_{SS} on the IntelDX4 processor. This pin should be left unconnected in designs that do not use the voltage detect feature. Figure 3 shows an example of the use of the VOLDET pin with a linear regulator circuit to automatically select the correct power supply voltage. If the VOLDET pin is not connected inside the processor, indicating a 5V part, the gate of MOSFET Q1 is pulled high, which causes it to bypass the 3.3V regulator, supplying 5V directly to the processor. Shorting the input of the regulator to the output in this way is harmless for most linear regulators, due to regulator feedback circuitry which shuts the regulator off (contact regulator manufacturers for specifics). Note that in this case, most regulators will require Q1 to handle all the processors current requirements, and so it should be a high-current, low on-state-resistance MOSFET. If the VOLDET pin is connected to V_{SS}, indicating a 3.3V part, the Q1 transistor is turned off, allowing the regulator to function normally. Figure 4 shows a suggested placement and layout for MOSFET Q1.

2.1.3.2 Other Voltage Selection Options

It is also possible to design a flexible system board where the processor supply voltage is selected by an assembly time option. There are several methods to achieve this; the key requirement being that the design must handle the maximum current of 3.0A for the future Pentium OverDrive processor. Note that normal jumpers may not be adequate, and it may be necessary to use several in parallel.



NOTES:
* D1 internal on all MOSFETs
** D2 internal on some MOSFETs

Figure 3. Example Voltage Auto-Select Circuit Topology (Courtesy of Linear Technology Corporation)

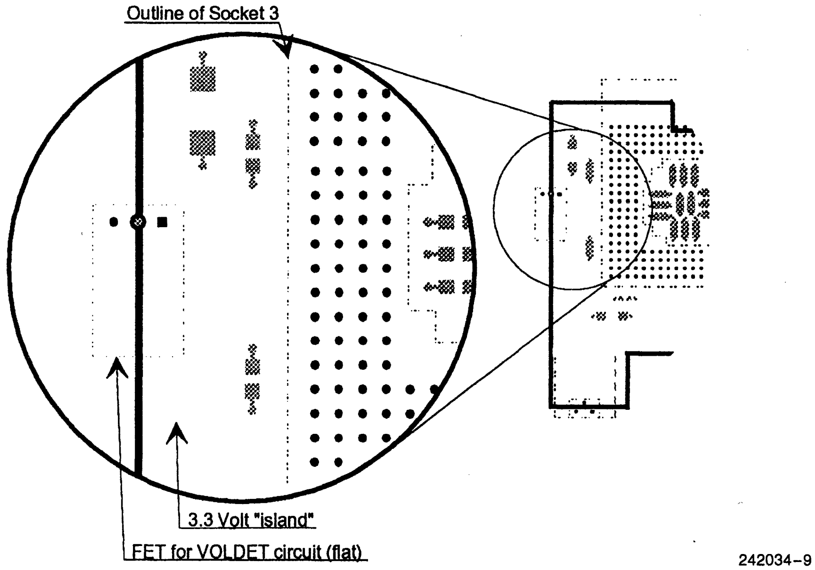


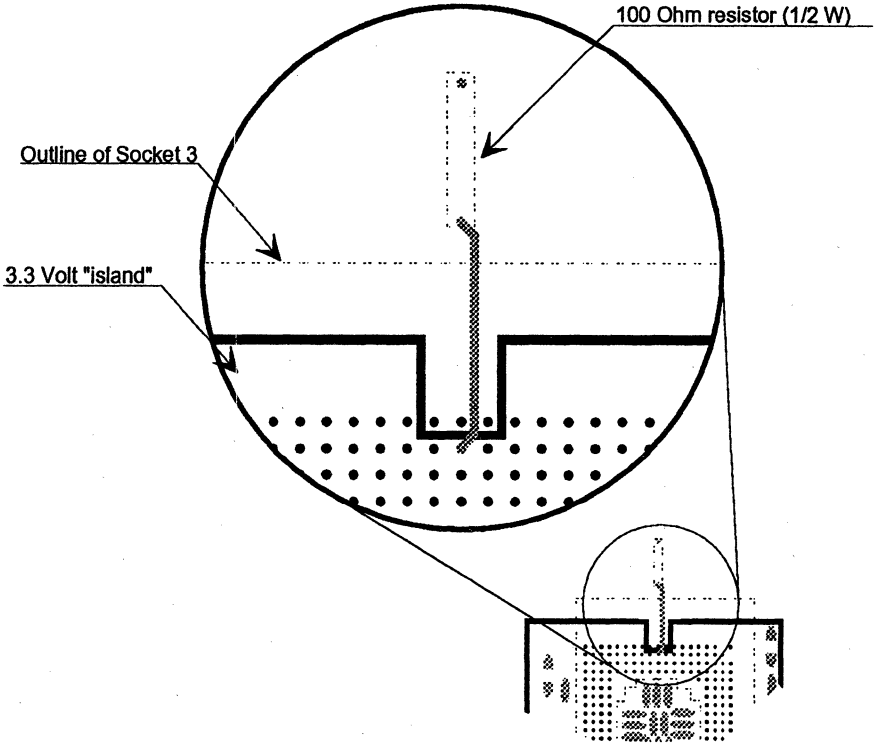
Figure 4. Suggested Placement and Layout for MOSFET Used in Optional Voltage Auto-Select Circuit

2.1.4 V_{CC5} PIN REQUIREMENT

For mixed voltage systems where the processor interfaces with 5V components, the V_{CC5} pin must be connected to 5V for proper 5V tolerant buffer operation. **The V_{CC5} input should not exceed V_{CC} by more than 2.25V during power-up, power-down or during operation.** If this requirement is not met, current flow through the pin may exceed 55 mA, and damage to the component may begin to occur.

To meet this requirement, one of two things must be done: either the power supply must be designed to turn on and off such that the difference between the V_{CC5} and V_{CC} voltages never exceeds 2.25V, or a 100Ω resistor must be put in series with the V_{CC5} pin to limit the current through this path (Figure 5 shows a possible layout for this connection). The 100Ω series resistor is required for power supplies which do not meet the voltage difference specification, and also provides protection in the case of a power supply failure (where the 5V supply remains on, but the 3.3V supply goes to zero).

Note that the V_{CC5P} pins for the future Pentium OverDrive processor fan/heatsink unit should be connected directly to the 5V supply, and not through a series resistor.



242034–10

NOTE:
The future Pentium™ OverDrive™ processor requires a 5V supply through the V_{CC5P} pins. The V_{CC5P} pins must be connected directly to the 5V supply, and not through the V_{CC5} pin protection resistor.

Figure 5. Possible Layout for V_{CC5P} Pin Connection

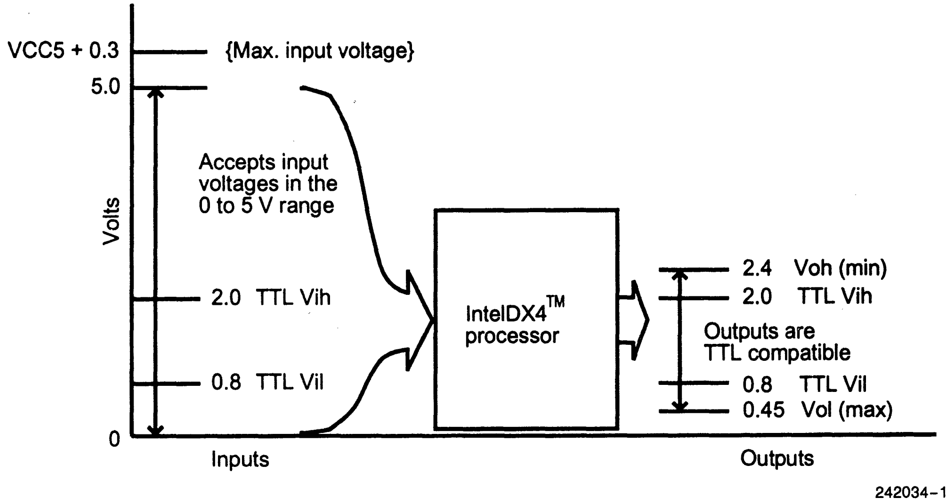


Figure 6. Voltage Relationships in an IntelDX4 Processor Based System with 5V External Logic

2.1.5 EXPLANATION OF 5V TOLERANT INPUTS AND TTL COMPATIBLE OUTPUTS

The IntelDX4 processor and future Pentium OverDrive processor include 5V tolerant input buffers and TTL compatible outputs. This feature enables the processor to interface with existing 5V logic even though the processor is running at 3.3V internally.

In a system with the 3.3V IntelDX4 processor interfaced to 5V components, the V_{CC5P} pin on the IntelDX4 processor must be attached to the 5V system power supply. The V_{CC5P} pin provides a voltage reference for the processor's input buffers. With V_{CC5P} connected to 5V, the processor can accept input signals up to 5.3V, making its inputs compatible with 5V TTL or CMOS level outputs.

Output voltages from the processor are guaranteed to be at or above 2.4V for a logic "1" and at or below 0.45V for a logic "0." This allows the IntelDX4 processor to drive TTL compatible input levels (2.0V and 0.8V), but not 5V CMOS levels ("rail to rail"). Figure 6 shows the input and output voltage relationships for the IntelDX4 processor in a 5V system.

In a 3.3V only system, the V_{CC5P} pin should be connected to the 3.3V supply, as 5V tolerant operation is not required.

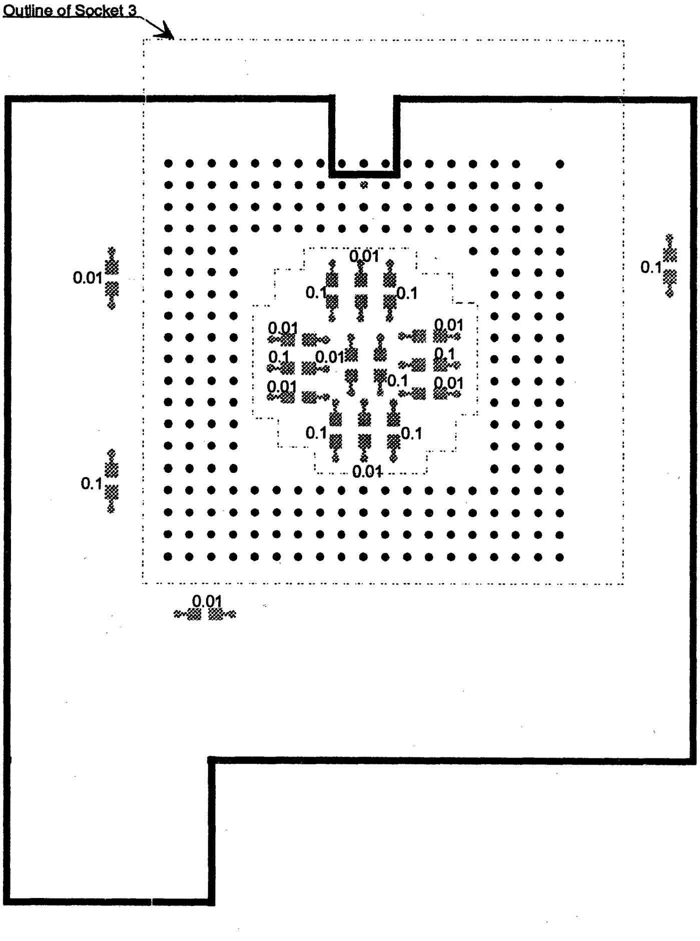
2.2 Processor Power Supply Decoupling

Processor power supply decoupling is critical for reliable operation. With the IntelDX4 processor, there are two areas of concern: high frequency decoupling, necessitated by the high speed operation of the processor, and low frequency decoupling, necessitated by the power saving features of the processor.

2.2.1 HIGH FREQUENCY POWER SUPPLY DECOUPLING

High frequency decoupling is critical on the IntelDX4 processor because of its high speed external bus, and because of its very fast 100 MHz internal operation. A **reliable design will include a minimum of nine 0.1 μF capacitors and nine 0.01 μF surface mount capacitors between power and ground, evenly distributed, close to the processor. The capacitors must be placed as close to the processor as possible, attached directly to the power and ground planes, or circuit board inductance will significantly reduce their effectiveness.** A typical failure mode caused by inadequate high frequency decoupling is unreliable or inconsistent program behavior. These failures are often intermittent, and are very hard to debug. Figure 7 shows a recommended layout for the high frequency capacitors, with values as shown.

Appendix C includes a list of possible vendors for power supply decoupling capacitors.



NOTE:
All values in microFarads

Figure 7. Recommended High Frequency Capacitor Values and Layout

2.2.2 BULK POWER SUPPLY DECOUPLING

Bulk, or low frequency, decoupling is needed on all SL Enhanced Intel486 processors, including the IntelDX4 processor, since the processor may switch between normal and low power states very quickly, causing large instantaneous current changes. To properly handle these instantaneous current changes, all designs must have adequate bulk decoupling. In 5V only systems, the processor can use the bulk decoupling capacitance all over the system board, but with the processor on a separate power plane "island," it is necessary to place adequate bulk capacitance on the processor "island." For bulk decoupling, multiple capacitors each in the range of 10 μF to 100 μF are typically used in parallel to achieve the required capacitance while maintaining a low effective series resistance (ESR). You can determine the amount of bulk decoupling required with the following formula:

$$C \approx (\Delta I * \Delta T) / \Delta V$$

where ΔI is the maximum change in current, ΔT is the time it takes the power supply to adjust to the current change, ΔV is the allowable voltage change to remain within specification.

The effective series resistance (ESR) must also be taken into account. You can find the maximum allowable ESR with this formula:

$$ESR \approx \Delta V / \Delta I$$

where ΔV and ΔI are the same as in the first equation.

For example, for the future Pentium OverDrive processor, the maximum change in current is about 2.8A. The response time of a linear regulator may be around 15 μs (contact regulator manufacturer for precise value). With no guard band, the maximum allowable supply voltage deviation from 3.3V is 0.3V, yielding the following:

$$C \approx (2.8A * 15\ \mu s) / 0.3V = 140\ \mu F$$

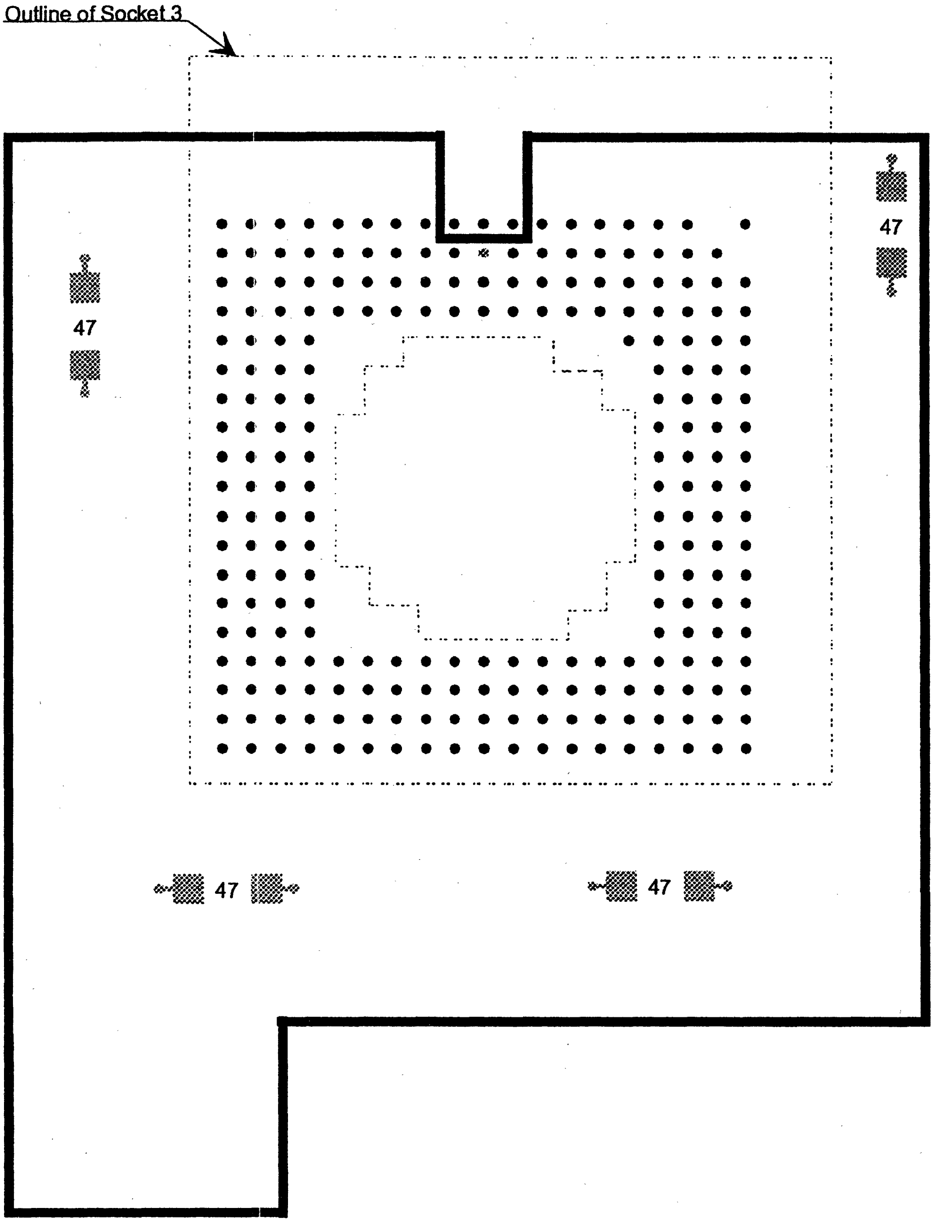
with a maximum allowable ESR:

$$ESR \approx 0.3V / 2.8A = 0.11\Omega$$

Placing four 47 μF tantalum surface mount capacitors in parallel, directly between the power and ground planes, will reduce the ESR below this limit and provide adequate capacitance. Figure 8 shows a recommended layout for this example.

For an example program which exercises the power saving features, Appendix D includes a program that alternates between HALT and normal operating modes with a keypress. This represents a typical load change for bulk capacitance testing with the IntelDX4 processor, but does not cover the future Pentium OverDrive processor. The Intel Verification Program provides a Voltage Regulator Transient Tester in the IVP Pretest Kit. This tool simulates worst case load changes with the future Pentium OverDrive processor. Another tool available from Intel for power transient testing is the Power Validator, which includes on board limit testers and failure indicators. To order the Power Validator or the IVP Pretest Kit, contact your local Intel representative.

Appendix C includes a list of possible vendors for power supply decoupling capacitors.



NOTE:
All values in microFarads

Figure 8. Recommended Bulk Decoupling Capacitor Values and Locations

2.2.3 WHY IS DECOUPLING NECESSARY?

CMOS logic only consumes significant current when switching. This allows great power savings by shutting down elements of a processor not being used. With the IntelDX4 processor, virtually the entire chip may be shut down and quickly restarted. Switching this amount of current on and off in very short periods of time may cause serious power supply voltage surges and droops in systems with inadequate bulk decoupling. Adequate power supply bulk decoupling capacitance, located near the processor, is necessary to filter these surges and droops (see Figure 8). The delayed response of voltage regulators to load increases is the principle cause of power supply droops, with the amount of droop depending on the response time of the regulator as well as power supply inductance. Adequate bulk capacitance is necessary to provide a current reservoir until the power supply or regulator can respond to the load increase. Surges are caused by inductance in the power supply, their severity being determined by the value of the inductance and the speed with which the current load drops. Most regulators only boost the regulated supply if it falls below the specified voltage, with the regulator turning off if the output voltage rises above the specified voltage. This means that a fast regulator will not lessen the effect of voltage surges. Another factor concerning the filtering ability of the bulk capacitance is the effective series resistance of the capacitor(s), which is an element of the non-ideal behavior of real components. The effect of this resistance must be low enough to not offset the desired filtering effect of the capacitance. Figure 9 shows an oscilloscope measurement of a surge in the power supply at the processor in an IntelDX4 processor based system with poor low frequency decoupling capacitance as the processor enters the Auto HALT Power Down State.

In addition, the increased internal speed of the IntelDX4 processor over the IntelDX2 processor means higher frequency noise components in the processor power supply. Traffic on the external bus causes high frequency power supply current spikes due to the large number of external outputs switching. High frequency (low inductance) capacitors, connected between the power and ground planes, near the processor, are required to filter these high frequency components of the noise. The inductive effects of circuit board traces and component leads become critical at these frequencies. For this reason, it is critical that the high frequency capacitors are placed as near as possible to the processor, using short traces to minimize inductance. Surface mount capacitors placed directly next to the processor and inside the socket cavity are recommended (see Figure 7). Figure 10 shows an oscilloscope measurement of the power supply at the processor in an IntelDX4 processor based system with poor high frequency decoupling capacitance.

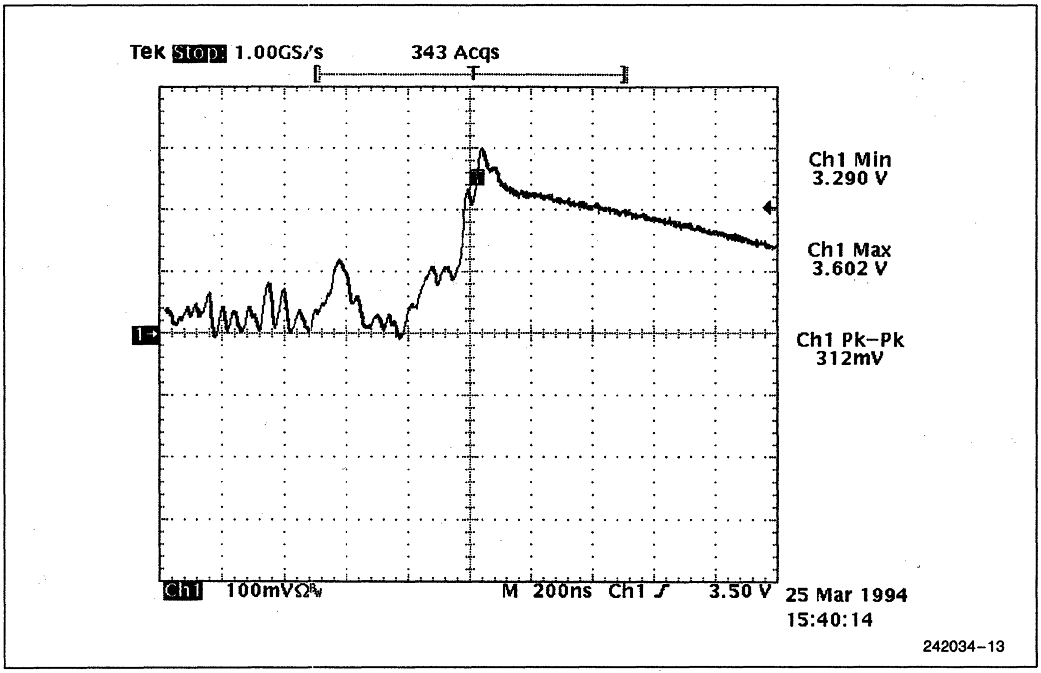


Figure 9. Oscilloscope Measurement Showing Power Supply Surge with Processor Entering HALT Mode in System with Poor Bulk Decoupling

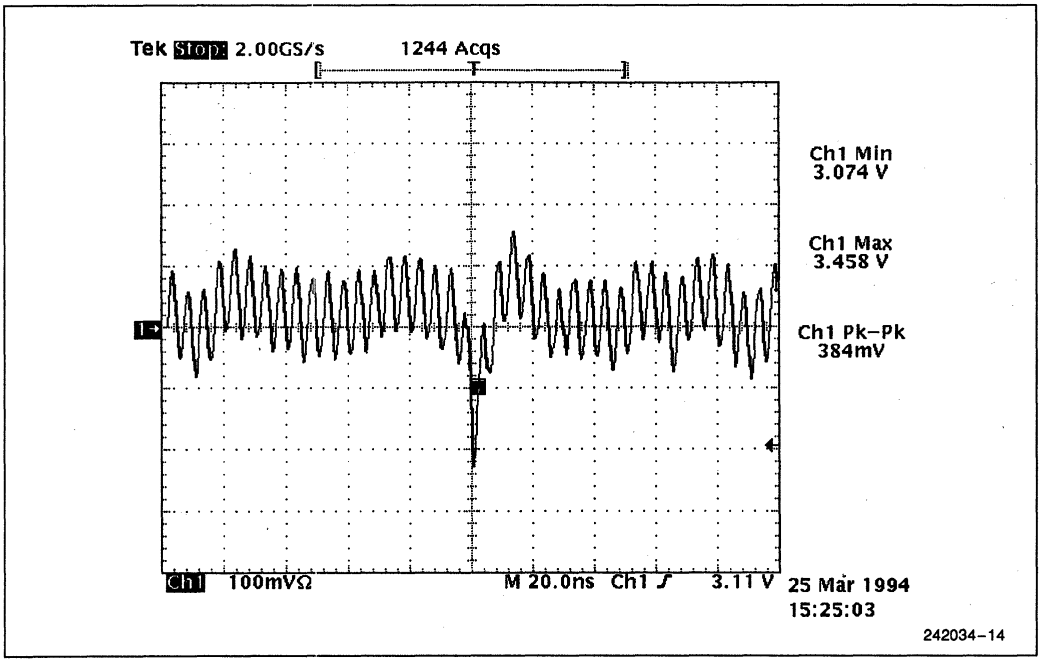


Figure 10. Oscilloscope Measurement Showing Power Supply Noise in System with Poor High Frequency Decoupling

2.3 Clock Multiplier Selection

The clock multiplier on the IntelDX4 processor may be selected in the system using the CLKMUL pin. This pin is an INC (Internal No Connect) on the previous Intel486 processors. The CLKMUL pin is sampled during cold (power on) processor resets. The clock multiplier cannot be changed during warm resets, and SRESET cannot be used to select the clock multiplier ratio. Typically, CLKMUL will be connected via a jumper to the proper setting. In systems with 33 MHz bus speeds, the IntelDX4 processor will operate with an internal clock frequency of 100 MHz if the CLKMUL pin is left unconnected, or is connected to V_{CC}. Existing IntelDX2 processor based systems running at 66 MHz internally with a 33 MHz bus speed require no modification to use the IntelDX4 processor at 100 MHz internally and 33 MHz externally if the CLKMUL pin is left unconnected. In 50 MHz bus systems, it is necessary to connect the CLKMUL pin to V_{SS} to achieve 100 MHz internal frequency operation. These relationships are shown in Table 2, and an example circuit is shown in Figure 11.

Table 2. Clock Multiplier Selection

CLKMUL at RESET	Clock Multiplier	External Clock Frequency	Internal Clock Frequency
V_{CC} or Not Driven	3	25 MHz 33 MHz	75 MHz 100 MHz
V_{SS}	2	50 MHz	100 MHz

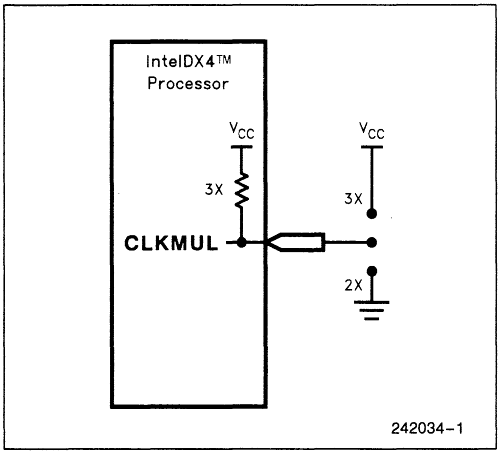


Figure 11. Jumpers for Clock Multiplier Selections

2.4 Thermal Considerations

In desktop systems, proper thermal management is critical to prevent system reliability problems caused by excess heat. The principle concern is with the 3.3V power supply when implemented with secondary onboard regulation.

2.4.1 Voltage Regulator Considerations

Special thermal consideration is required for systems implementing the 3.3V processor supply with a linear voltage regulator. The 10W dissipation of the future Pentium OverDrive processor requires a 3.0A power supply current. **A linear voltage regulator will dissipate approximately 5W of power to provide 3.0A. To keep the voltage regulator within thermal specification, a heatsink will be necessary to dissipate the heat generated by the regulator.** If the temperature of the regulator is not maintained within the manufacturer's specification, improper regulator operation may occur, jeopardizing regulator and processor reliability.

The size and performance of the voltage regulator heatsink are dependent upon the regulator specifications and system air flow. If the regulator is located adjacent to the upgrade processor socket, the ambient temperature should not exceed 55°C; the limit for the future Pentium OverDrive processor.

The following formula may be used to calculate the performance of a heatsink in a particular application:

$$\theta_{CA} = (T_J - T_A) / P_D - \theta_{JC}$$

where θ_{CA} is the maximum allowable thermal resistance of the heatsink and insulator, T_J is the maximum regulator junction temperature, T_A is the maximum allowable ambient temperature, P_D is the maximum power dissipated in the regulator, and θ_{JC} is the thermal resistance from the regulator junction to its case.

For example, for the Linear Technology LT1085CT regulator, the values would be 125°C for T_J, 55°C for T_A, 3°C/W for θ_{JC} so:

$$\theta_{CA} = (125°C - 55°C) / 5W - 3°C/W$$
$$= 11°C/W \text{ (for the LT1085CT)}$$

In this example, the thermal resistance of the heatsink and insulator (θ_{CA}) cannot exceed 11°C/W to meet specifications in still air.

Appendix C includes a list of possible vendors for heatsinks.

2.4.2 PROCESSOR CONSIDERATIONS

The power consumption of the IntelDX4 processor is comparable to that of the IntelDX2 processor, so existing thermal solutions that adhere to the published specifications should be adequate for the IntelDX4 processor.

The future Pentium OverDrive processor upgrade for the IntelDX4 processor will have an on-package fan/heatsink and is thermally equivalent to the Pentium OverDrive processor for IntelDX2 processor based systems. The on-package fan/heatsink unit is powered by the V_{CC5P} pins, which should be connected directly to the 5V system power supply.

2.5 Placement and Layout Suggestions

Figure 12 shows a complete suggested layout for the processor including the 3.3V power supply "island," adequate power supply decoupling, a linear voltage regulator, and a possible placement for the FET used with the VOLDET automatic voltage select circuit option. Note the placement of the capacitors close to the processor, and the wide connection from the voltage regulator to the "island."

Figure 13 shows the clearance required for the IntelDX4 processor and the future Pentium OverDrive processor. The future Pentium OverDrive processor is physically larger than the IntelDX4 processor, so it is not sufficient to merely provide clearance for the IntelDX4 processor if the system is to be upgradeable with the future Pentium OverDrive processor.

2.6 Intel Verification Program

The Intel Verification Program establishes minimum system design criteria for reliable and straightforward CPU upgradability with the Pentium OverDrive processors. The criteria encompass physical, functional, electrical, thermal, and installation attributes of the Pentium OverDrive processors. Upon successful system design verification by Intel, licensed OEMs will be able to advertise and promote their branded systems as "Intel Verified." An Intel published list of verified systems will promote consumer awareness providing greater confidence in both system and upgrade buying decisions.

Converting An "Intel Verified: For the Pentium OverDrive Processor" System Design

The Intel Verification program for IntelDX4 processor based system designs is similar to the recently introduced program for the IntelDX2 processor based systems. The criteria are basically the same with additional electrical and thermal tests. For IntelDX2 processor based designs that meet the criteria for the Pentium OverDrive processor, the additional criteria for the future Pentium OverDrive processor for the IntelDX4 processor pertain to the 3.3V supply specifications and voltage supply thermal requirements. This application note addresses both issues. Contact your local Intel representative for further information on the Intel Verification Program.

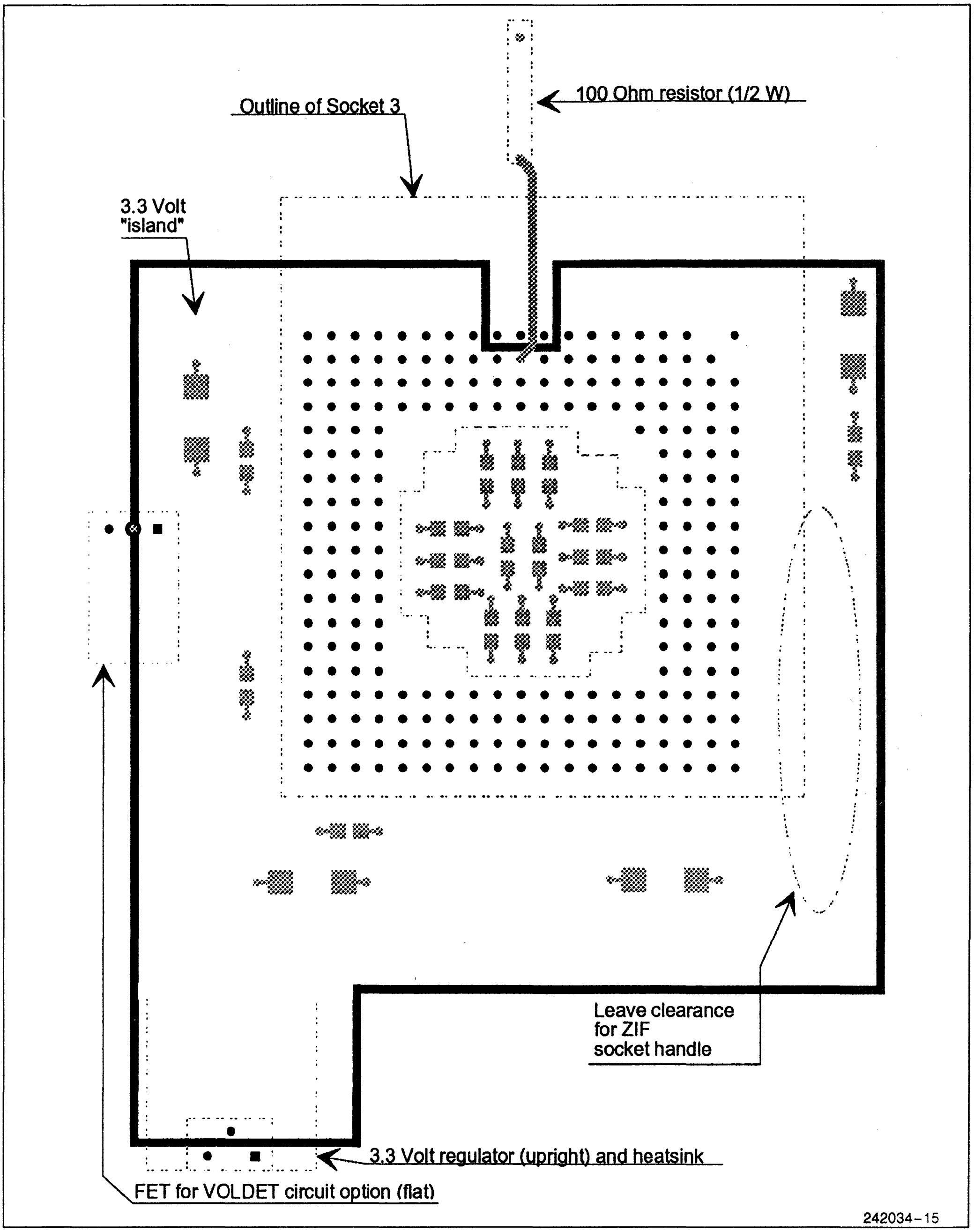

Outline of Socket 3
100 Ohm resistor (1/2 W)
3.3 Volt "island"
Leave clearance for ZIF socket handle
3.3 Volt regulator (upright) and heatsink
FET for VOLDET circuit option (flat)
242034-15

Figure 12. Complete Suggested Layout

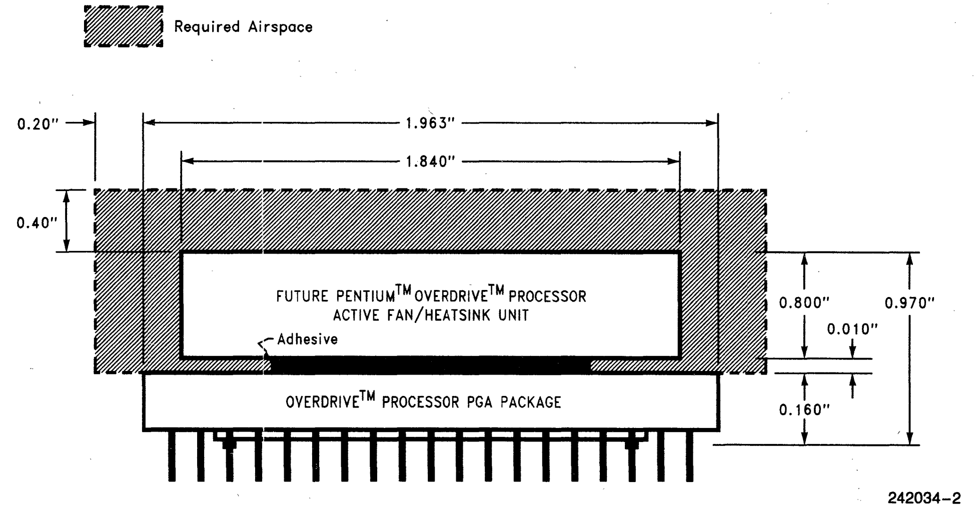


Figure 13. Clearance for Future Pentium™ OverDrive™ Processor

2.7 Cache and Memory Considerations

The IntelDX4 processor, because of its higher internal speed, will benefit even more from external performance enhancing features than the IntelDX2 processor. In particular, a second level cache can provide significant performance gains compared to systems without a second level cache. In addition, the future Pentium OverDrive processor for the IntelDX4 processor will allow write back operation of the processor's internal cache, which must be supported for maximum performance.

2.7.1 SECOND LEVEL CACHE

For maximum performance, a second level cache should be used. A simple 128 Kbyte second level cache will improve performance 10%–20% over systems without a second level cache. Increasing the cache size to 256 Kbytes will improve performance by roughly an additional 5%. Using a write-back algorithm will provide a 10%–15% improvement over a write-through system. If a write-back algorithm is used, a two-way cache architecture will yield about 10% performance gain over a direct-mapped cache, but this improvement is much less if a write-through algorithm is used. The greatest benefit is achieved when no wait states are required, but if wait states are necessary, it is better to have one wait state on the leadoff cycle and no wait states in the burst cycles, than to have no wait state on the leadoff and one wait state on the bursts. It may be necessary to use interleaving to achieve zero wait state burst access. Memory system performance also has a significant effect on system performance. The effect of these system design parameters will vary between applications; the performance figures given above are for typical applications. See application note AP-469: Cache and Memory Design Considerations for the Intel486™ DX2 Microprocessor (Order Number: 241261-001) for more information on cache and memory system design.

2.7.2 WRITE-BACK SUPPORT FOR THE FUTURE PENTIUM OVERDRIVE PROCESSOR

For optimal performance with the future Pentium OverDrive processor, your system design should be able to support processors with write-back internal caches. Contact your local Intel representative for more information.

3.0 SOFTWARE VISIBLE DIFFERENCES

Table 3 summarizes the system programmer visible differences between the IntelDX4 processor and the IntelDX2 processor and their respective Pentium OverDrive processors. Note that the future Pentium OverDrive processor for IntelDX4 processor based designs is different from the Pentium OverDrive processor for IntelDX2 processor based designs.

Table 3. Programmer Visible Differences between the IntelDX4™ Processor, the IntelDX2™ Processor and Their Corresponding Pentium™ OverDrive Processors

	IntelDX4™ Processor	IntelDX2™ Processor	Future Pentium™ OverDrive™ Processor for the IntelDX4™ Processor	Pentium™ OverDrive™ Processor for the IntelDX2™ Processor
Component Identifier in DX Register at Reset	048x	043x	155x	153x
CPUID Instruction	yes	yes*	yes	yes
Internal Cache	WT Unified 16 Kbytes, 4-way	WT Unified 8 Kbytes, 4-way	WT/WB **	WT/WB **

NOTES:
* Not available in older non-SL Enhanced versions
** Contact your local Intel representative for details

The following sections highlight the key points to consider when converting an IntelDX2 processor based design to an IntelDX4 processor based design, particularly processor identification and cache test register differences.

Information concerning the IntelDX4™ processor Virtual Mode Extensions is not provided in this document. This information is specifically targeted at writers of operating system kernels and virtual memory managers, and is available with the appropriate non-disclosure agreements in place. Please contact your local Intel representative for details.

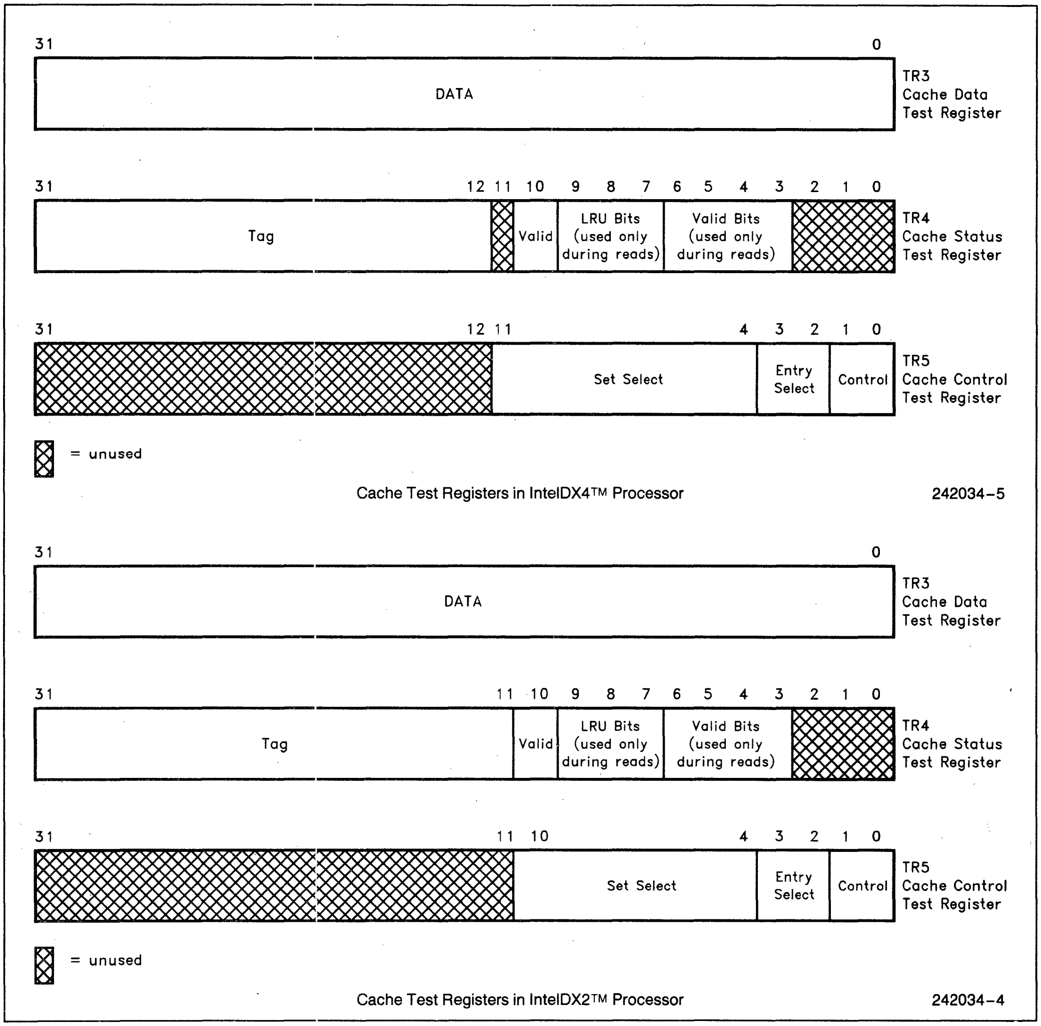
3.1 Processor Identification

Processor identification is needed principally for testing of the processor during the power-on-self-test (POST) typically performed by the BIOS on RESET. Improper processor identification may result in improper cache testing, or an attempt to use a non-implemented instruction. The correct way to identify the IntelDX4 processor is by examining the contents of the DX register immediately after RESET, or by executing the CPUID instruction. All Intel SL Enhanced processors implement the CPUID instruction. See application note AP-485: Intel Processor Identification with the CPUID Instruction (Order Number: 241618-001), for the proper use of the CPUID instruction. Alternately, the BIOS may read the DX register at startup and write the processor information into a location in memory for later use.

The IntelDX4 processor can be distinguished by its component identifier of 048x (hex), where the last nybble is a version number. Since the system processor may be a Pentium OverDrive processor, the BIOS should be written to correctly identify and be compatible with the Pentium OverDrive processor.

3.2 Cache Test Register Difference

The cache test registers of the IntelDX4 processor are slightly different from those of the IntelDX2 processor, as shown in Figure 14. To avoid false indications of failure, BIOS or other software routines which test the cache should be modified to account for this difference. It should also be remembered that the future Pentium OverDrive processor for the IntelDX4 processor may be different from the IntelDX4 processor and the IntelDX2 processor in this respect. Contact your local Intel representative for details.



Cache Test Registers in IntelDX4™ Processor

242034–5

Cache Test Registers in IntelDX2™ Processor

242034–4

Figure 14. Cache Test Register Difference between the IntelDX4™ Processor and the IntelDX2™ Processor

4.0 SUMMARY

As you plan your design using the IntelDX4 processor, please keep in mind the points addressed in this application note. Some of the key points are:

- **The IntelDX4 processor is a 3.3V component**
- **Adequate high frequency power supply decoupling must be used**
- **Adequate low frequency power supply decoupling must be used**
- **When using on-board regulation for processor power supply, a heatsink will be needed to dissipate regulator heat when supplying 3.0A for OverDrive processor**

Other points to remember are:

- Meet the voltage difference requirement on V_{CC5} pin, or use a 100Ω resistor in series with the 5V supply
- The IntelDX4 processor CLKMUL pin is used to select internal clock multiplier
- The IntelDX4 processor provides 5V tolerant inputs, and TTL compatible outputs
- The future Pentium OverDrive processor for IntelDX4 processor requires 5V supply through V_{CC5P} pins for fan/heatsink unit
- The VOLDET pin may be used to automatically select the power supply voltage
- Correct processor identification is necessary for correct testing by the BIOS

Intel appreciates your interest in the IntelDX4 processor. This document has been developed to allow you to minimize your investment in development time and bring a reliable design to market quickly. By keeping the points addressed in the application note in mind, you can be confident that your system will be ready to meet the new standard for 486 computing set by the 100 MHz IntelDX4 processor.

References

IntelDX4 Processor Data Book (Order Number: 241944-001)

i486 Microprocessor Hardware Reference Manual (Order Number: 240552-001)

AP-469: Cache and Memory Design Considerations for the Intel486™ DX2 Microprocessor (Order Number: 241261-001)

AP-485: Intel Processor Identification with the CPUID Instruction (Order Number: 241618-001)

APPENDIX A
DESIGN CHECKLIST

Reliable system designs must meet the requirements given in the IntelDX4 processor data book. The following design elements should be checked against the specifications in the IntelDX4 processor data book:

1) 3.3V processor power supply (Section 2.1)

Type:	System power supply or Regulator (linear or switching)
Tolerance:	±0.3V at processor V_{CC} pins
Max current:	At least 3.0A for OverDrive processor
VCC5:	Less than 2.25V difference between V_{CC} pins and V_{CC5} at all times (Use a 100Ω resistor in series with V_{CC5} pin if this requirement cannot be met)

2) High frequency decoupling capacitance on board (Section 2.2.1)

Type:	High quality (low inductance, low resistance)
Quantity:	Nine 0.01 μF, nine 0.1 μF
Location:	As near as possible to the processor
Test:	Unreliable or inconsistent operation may be caused by inadequate high frequency decoupling

3) Bulk decoupling capacitance on board (Section 2.2.2)

Value:	Use formulas given in Section 2.2.2
Test:	Using available test programs and tools, does the processor supply voltage stay in specification? (Note: the OverDrive™ processor may switch by 2.8A)

4) Clock multiplier selection (Section 2.3)

Note:	Use the CLKMUL pin to select the desired multiplier (May be left unconnected to achieve 100 MHz internal with 33 MHz external operation)

5) Thermal requirements (Section 2.4)

Test:	Is measured processor T_{case} within specification?
Check:	Does voltage regulator have an adequate heatsink to provide 3A for the OverDrive processor? (If using on-board regulation)

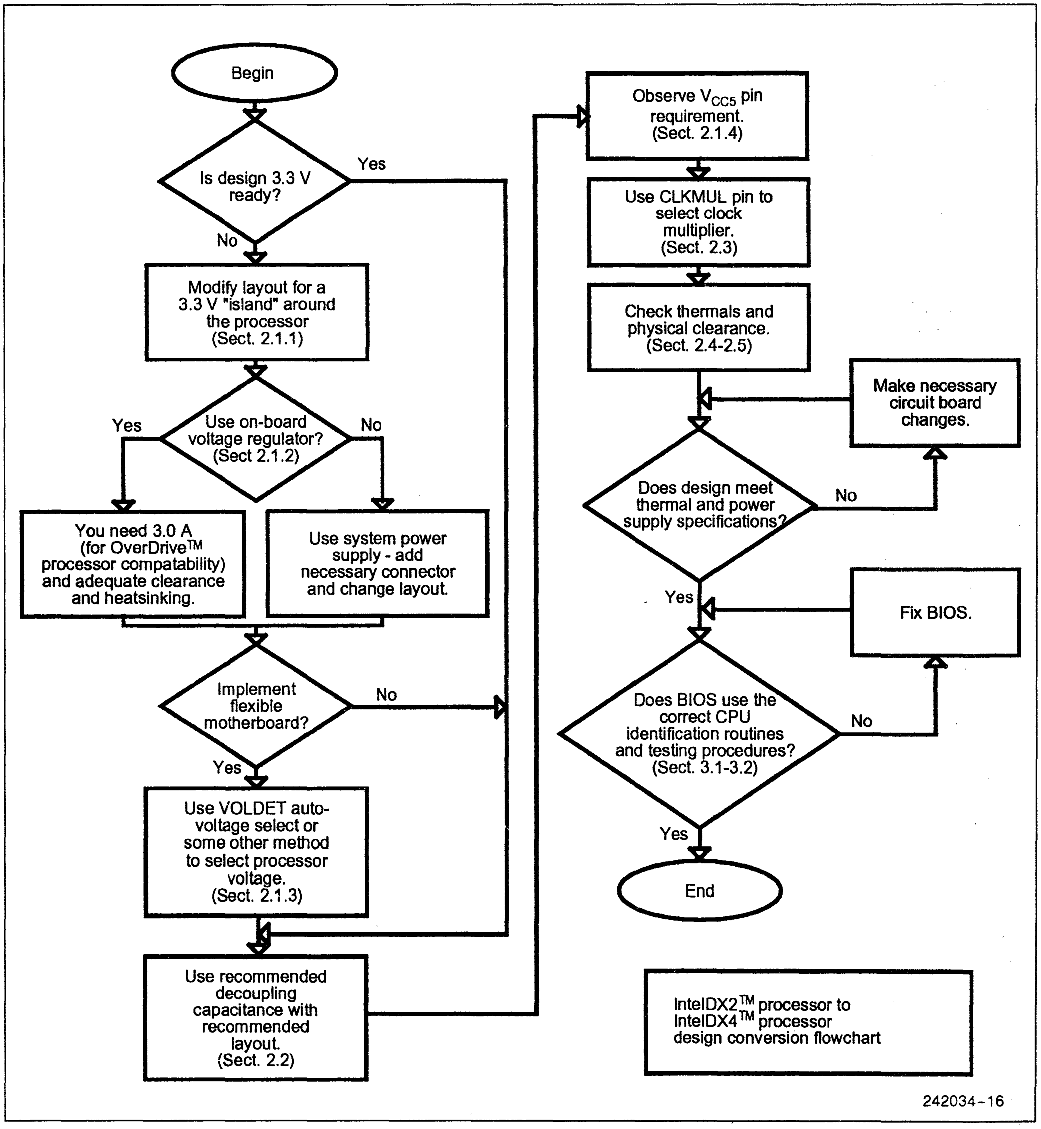
6) Physical requirements (Section 2.5)

Check:	Is clearance sufficient for OverDrive processor?

7) Software (Section 3.0)

Test:	Will the processor be properly identified (if applicable)?
Check:	Is the cache tested correctly? (If BIOS tests the cache)
Check:	Will an OverDrive processor be properly identified?

APPENDIX B
DESIGN FLOWCHART


Begin
Is design 3.3 V ready?
Yes
No
Modify layout for a 3.3 V "island" around the processor (Sect. 2.1.1)
Use on-board voltage regulator? (Sect 2.1.2)
Yes
No
You need 3.0 A (for OverDrive™ processor compatability) and adequate clearance and heatsinking.
Use system power supply - add necessary connector and change layout.
Implement flexible motherboard?
No
Yes
Use VOLDET auto-voltage select or some other method to select processor voltage. (Sect. 2.1.3)
Use recommended decoupling capacitance with recommended layout. (Sect. 2.2)
Observe V_{CC5} pin requirement. (Sect. 2.1.4)
Use CLKMUL pin to select clock multiplier. (Sect. 2.3)
Check thermals and physical clearance. (Sect. 2.4-2.5)
Make necessary circuit board changes.
Does design meet thermal and power supply specifications?
No
Yes
Fix BIOS.
Does BIOS use the correct CPU identification routines and testing procedures? (Sect. 3.1-3.2)
No
Yes
End
IntelDX2™ processor to IntelDX4™ processor design conversion flowchart
242034-16

APPENDIX C
SUPPORT COMPONENT VENDOR LIST

NOTE:
This list is meant to be representative only, and may not include all vendors of a particular type. Intel has not tested all of the components listed below and cannot guarantee that these components will meet every PC manufacturers specific requirements.

Voltage Regulators:

Linear Technology
1630 McCarthy Blvd
Milpitas, CA 95035
Tel. (408) 432-1900

Maxim Integrated Products
120 San Gabriel Drive
Sunnyvale, CA 94086
Tel. (408) 737-7600

Micrel Semiconductor
1849 Fortune Dr.
San Jose, CA 95131
Tel. (408) 944-0800

National Semiconductor Corp.
2900 Semiconductor Drive
Santa Clara, CA 95052
Tel. (800) 272-9959

3.3V System Power Supplies:

Astec America, Inc.
401 Jones Road
Oceanside, CA 92054
Tel. (619) 757-1880

Golden Systems, Inc.
2125B Madera Road
Simi Valley, CA 93065
Tel. (805) 582-4400

Teapo Electronic Corporation
No. 3 Lane 89. Sec. 3 Chung-Yang Rd
Tu-Cheng Hsiang, Taipei Hsein, Tiawan, R.O.C.
Tel. 886 2 260-4151
In USA: (404) 449-6220

Heatsinks for Linear Regulators:

Aavid Engineering, Inc.
P.O. Box 400
Laconia, NH 03247
Tel. (603) 528-3400

Thermalloy, Inc.
2021 W.Valley View Lane
Dallas, TX 75234
Tel. (214) 243-4321

Decoupling capacitors:

Kemet Electronics Corporation
P.O. Box 5928
Greenville, SC 29606
Tel. (803) 963-6348

Mallory/North American Capacitor Company
P.O. Box 1284
Indianapolis, IN 46206
Tel. (317) 273-0090

Sanyo Video Components (USA) Corp.
2001 Sanyo Ave.
San Diego, CA 92073
Tel. (619) 661-6835

Power MOSFETS:

Harris Semiconductor
1301 Woody Burke Road
Melbourne, FL 32902
Tel. (800) 442-7747

International Rectifier Corp.
233 Kansas St.
El Segundo, CA 90245
Tel. (310) 322-3331

National Semiconductor Corp.
2900 Semiconductor Drive
Santa Clara, CA 95052
Tel. (800) 272-9959

Siliconix
2201 Laurelwood Rd.
Santa Clara, CA 96056
Tel. (800) 554-5565

ZIF Sockets:

Note:
Contact your local Intel representative for a list of qualified sockets for the Intel Verification Program.

AMP Inc.
219 American Avenue
Greensboro, NC 27409-1806
Tel. (800) 522-6752

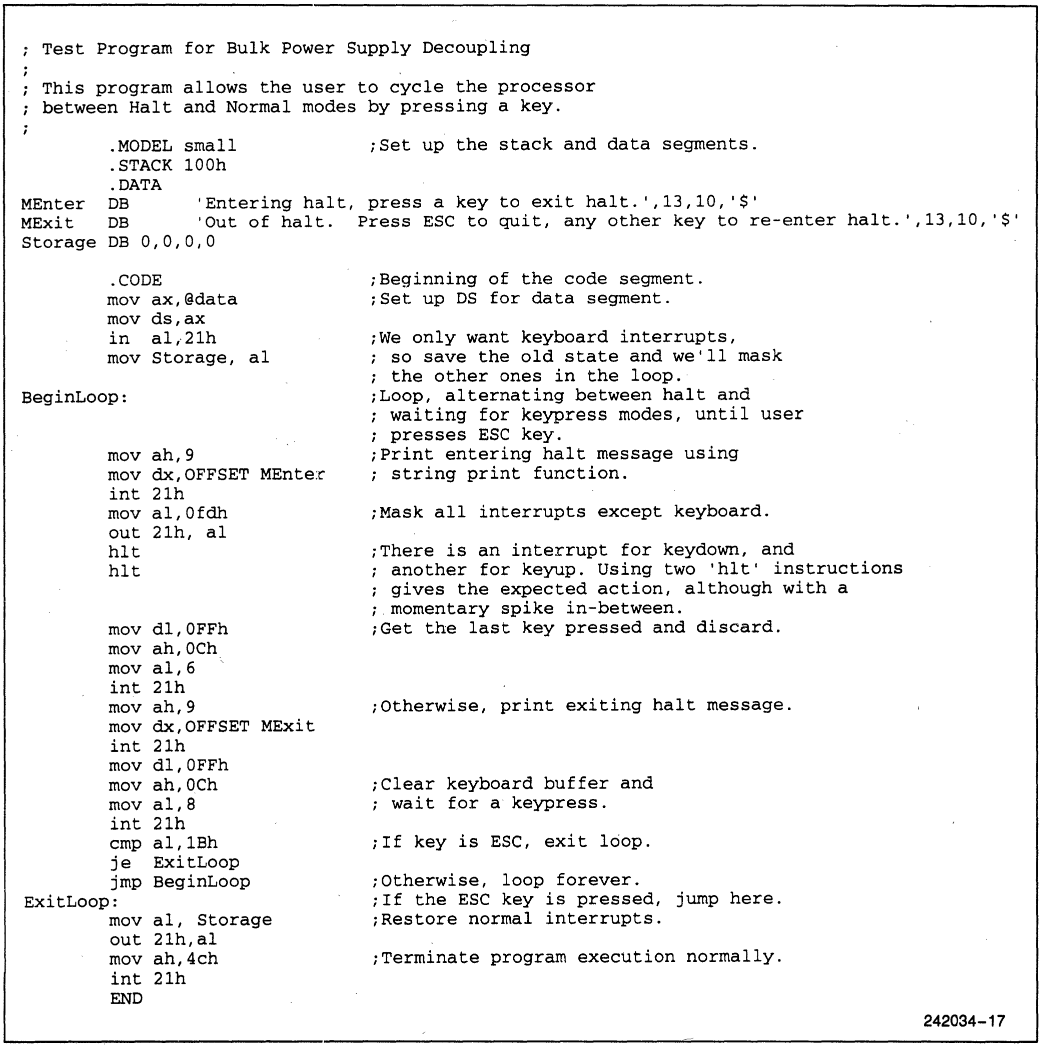
Augat Inc.
452 John Dietsch Blvd.
Attleboro Falls, MA 02763
Tel. (800) 999-7646

Foxconn / Hon Hai Precision Industry Co., Ltd.
2, Tzu Yu St., Tu-Chen
Taipei Hsien, Taiwan, R.O.C.
Tel. 02-268-3466
In USA: (800) 727-3699

Loranger International Corp.
817 Fourth Ave.
Warren, PA. 16365
Tel. (814) 723-2250

Yamaichi Electronics Co., Ltd.
3-28-7, Nakamagomi, Ohta-Ku
Tokyo 143
Japan
Tel. 03-3778-6160
In USA: (800) 769-0797

APPENDIX D
TEST PROGRAM FOR BULK POWER SUPPLY DECOUPLING



```
; Test Program for Bulk Power Supply Decoupling
;
; This program allows the user to cycle the processor
; between Halt and Normal modes by pressing a key.
;
        .MODEL small            ;Set up the stack and data segments.
        .STACK 100h
        .DATA
MEnter  DB      'Entering halt, press a key to exit halt.',13,10,'$'
MExit   DB      'Out of halt.  Press ESC to quit, any other key to re-enter halt.',13,10,'$'
Storage DB 0,0,0,0

        .CODE                   ;Beginning of the code segment.
        mov ax,@data            ;Set up DS for data segment.
        mov ds,ax
        in  al,21h              ;We only want keyboard interrupts,
        mov Storage, al         ; so save the old state and we'll mask
                                ; the other ones in the loop.
BeginLoop:                      ;Loop, alternating between halt and
                                ; waiting for keypress modes, until user
                                ; presses ESC key.
        mov ah,9                ;Print entering halt message using
        mov dx,OFFSET MEnter    ; string print function.
        int 21h
        mov al,0fdh             ;Mask all interrupts except keyboard.
        out 21h, al
        hlt                     ;There is an interrupt for keydown, and
        hlt                     ; another for keyup. Using two 'hlt' instructions
                                ; gives the expected action, although with a
                                ; momentary spike in-between.
        mov dl,0FFh             ;Get the last key pressed and discard.
        mov ah,0Ch
        mov al,6
        int 21h
        mov ah,9                ;Otherwise, print exiting halt message.
        mov dx,OFFSET MExit
        int 21h
        mov dl,0FFh
        mov ah,0Ch              ;Clear keyboard buffer and
        mov al,8                ; wait for a keypress.
        int 21h
        cmp al,1Bh              ;If key is ESC, exit loop.
        je  ExitLoop
        jmp BeginLoop           ;Otherwise, loop forever.
ExitLoop:                       ;If the ESC key is pressed, jump here.
        mov al, Storage         ;Restore normal interrupts.
        out 21h,al
        mov ah,4ch              ;Terminate program execution normally.
        int 21h
        END
```

242034-17

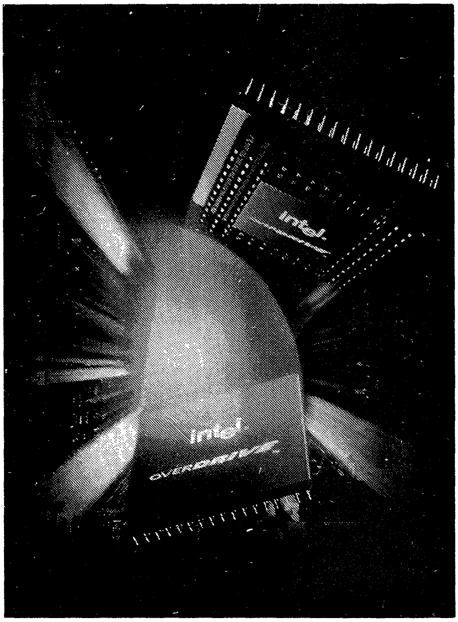
3

Intel OverDrive™ Processors

PRELIMINARY

INTEL OverDrive™ PROCESSORS

- Powerful Processor Upgrades for Intel486™ Microprocessor-Based Systems
  - Significantly Accelerates All Software Applications
- Intel OverDrive™ Processor Family Includes:
  - IntelSX2™ OverDrive™ Processor
  - IntelDX2™ OverDrive™ Processor
  - IntelDX4™ OverDrive™ Processor
- Will Upgrade Systems Based on:
  - Intel486™ SX Processors
  - Intel486™ DX Processors
- Large Installed Base of Thousands of Applications
- Based on Advanced Intel486™ Processor Technology for 100% Compatibility
- Backed by Intel's Lifetime Warranty
- Incorporates SMM Power Saving Features



290436–20

Intel OverDrive processors are powerful, processor upgrades for Intel486 microprocessor-based systems. The IntelSX2 OverDrive processor is the most affordable processor upgrade designed for Intel486 SX processor-based systems. It features the Intel speed doubling technology, thereby accelerating all non-floating point intensive software. This provides home and small office users increased performance across a wide variety of applications. The IntelDX2 OverDrive processor is a powerful processor upgrade designed for Intel486 SX and Intel486 DX processor-based systems. Utilizing the Intel speed doubling technology, the IntelDX2 OverDrive processor accelerates both integer and floating point software, providing business users significantly increased performance across a wide variety of applications. The IntelDX4 OverDrive processor is a high-end processor upgrade designed for Intel486 SX and Intel486 DX processor-based systems. Utilizing the Intel speed tripling technology, the IntelDX4 OverDrive processor accelerates both integer and floating point software, providing power users with significant software performance improvements across a wide variety of applications.

*Other brands and names are the property of their respective owners.

October 1994
Order Number: 290436-005

Intel OverDrive™ Processors

CONTENTS PAGE

CONTENTS PAGE

1.0 INTRODUCTION

This data sheet describes the Intel OverDrive processors, a family of CPU upgrades for Intel486 processor-based systems. This family includes the IntelSX2 OverDrive processor, the IntelDX2 OverDrive processor and the IntelDX4 OverDrive processor. These processor upgrades significantly accelerate all software applications run on your existing system, thereby increasing the overall performance of your PC.

It is important to note that this data sheet is intended to be used in conjunction with the data sheet for the original processor in the system—the Intel486 SX or Intel486 DX processor—which describes the Intel Family Architecture and functionality (Order # 241731-001). All enhancements or differences between the OverDrive processor and the original processor (i.e., IntelSX2 OverDrive vs. Intel486 SX2 processor, IntelDX2 or IntelDX4 OverDrive vs. Intel486 DX processor) are described in this data sheet.

Intel486 SX or Intel486 DX processor-based systems that are compatible to the Intel OverDrive processor(s) must be designed to both the original processor specifications and the Intel OverDrive processor(s) specifications.

1.1 Product Overview

The following sections provide an overview of each of the OverDrive processors. Refer to the specific product section(s) for more detailed information.

Figure 1-1 lists some of the key features of each OverDrive processor. Figure 1-2 describes the upgrade choices available for an existing Intel486 SX or DX system.

1.1.1 IntelSX2™ OverDrive™ PROCESSOR

The IntelSX2 OverDrive processor is the most affordable upgrade designed for Intel486 SX processor-based systems. It features the Intel speed doubling technology, thereby accelerating all non-floating point intensive software. This provides home and small office users increased performance across a wide variety of software applications.

Based on the Intel486 SX2 processor technology, the IntelSX2 OverDrive processor integrates an integer unit, a memory management unit and an 8 KByte cache on a single chip. The speed doubling technology allows the processor to operate internally at twice the speed of the system bus; up to a maximum of 50 MHz for a 25 MHz system.

IntelSX2™ OverDrive Processor	• For Up To 25 MHz Intel486 SX Processor-Based Systems • Based on Intel486 SX2 Speed Doubling Technology • Intel487SX Pinout • Hardware Floating Point Math Unit Not Supported • 8 KByte Integrated Instruction and Data Cache
IntelDX2™ OverDrive Processor	• For Up To 33 MHz Intel486 SX or DX Processor-Based Systems • Based on Intel486 DX2 Technology, which Includes Speed Doubling and Floating Point Math Unit • Intel486 DX and Intel487 Pinouts • 8 KByte Integrated Instruction and Data Cache
IntelDX4™ OverDrive Processor	• For Up To 33 MHz Intel486 SX or DX Processor-Based Systems • Based on Intel486 DX4 Speed Tripling Technology, which Includes Floating Point Math Unit • Intel486 DX and Intel487 Pinouts • 16 KByte Integrated Instruction and Data Cache

290436-27

Figure 1-1. Key Features

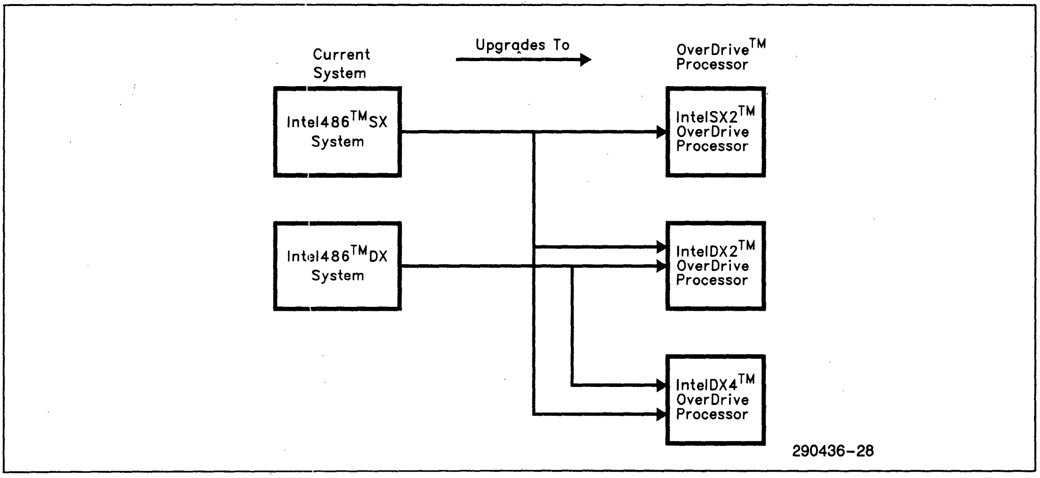


Figure 1-2. Upgrade Choices

The IntelSX2 OverDrive processor is designed to be installed into the OverDrive processor socket of Intel486 SX processor-based systems. It can also replace the existing Intel486 SX processor in single-socket systems.

1.1.2 IntelDX2™ OverDrive™ PROCESSOR

The IntelDX2 OverDrive processor is a powerful processor upgrade designed for Intel486 SX and Intel486 DX processor-based systems. Based on the Intel486 DX2 processor, it contains the Intel speed doubling technology. This accelerates both integer and floating point software, providing business users significantly increased performance across a wide variety of software applications.

The IntelDX2 OverDrive processor integrates an integer unit, a floating point math coprocessor unit, a memory management unit and an 8 KByte cache on a single chip. The speed doubling technology allows the processor to operate internally at twice the speed of the system bus; up to a maximum of 66 MHz for a 33 MHz system.

The IntelDX2 OverDrive processor comes in two package offerings; 168-lead Pin Grid Array (PGA) and 169-lead PGA. It is designed to be installed into the OverDrive processor socket of Intel486 SX and DX processor-based systems. It can also replace the existing processor in single-socket systems.

1.1.3 IntelDX4™ OverDrive™ PROCESSOR

The IntelDX4 OverDrive processor is a high-end processor upgrade designed for Intel486 SX and Intel486 DX processor-based systems. Utilizing the Intel speed tripling technology, the IntelDX4 OverDrive processor accelerates both integer and floating point software, providing power users with significant software performance improvements across a wide variety of applications.

The IntelDX4 OverDrive processor integrates an integer unit, a floating point math coprocessor unit, a memory management unit and a 16-KByte cache on a single chip. The speed tripling technology allows the processor to operate internally at three times the speed of the system bus; up to a maximum of 100 MHz for a 33 MHz system.

The IntelDX4 OverDrive processor comes in two package offerings; 168-lead Pin Grid Array (PGA) and 169-lead PGA. It is designed to be installed into the OverDrive processor socket of Intel486 SX and DX processor-based systems. It can also replace the existing processor in single-socket systems.

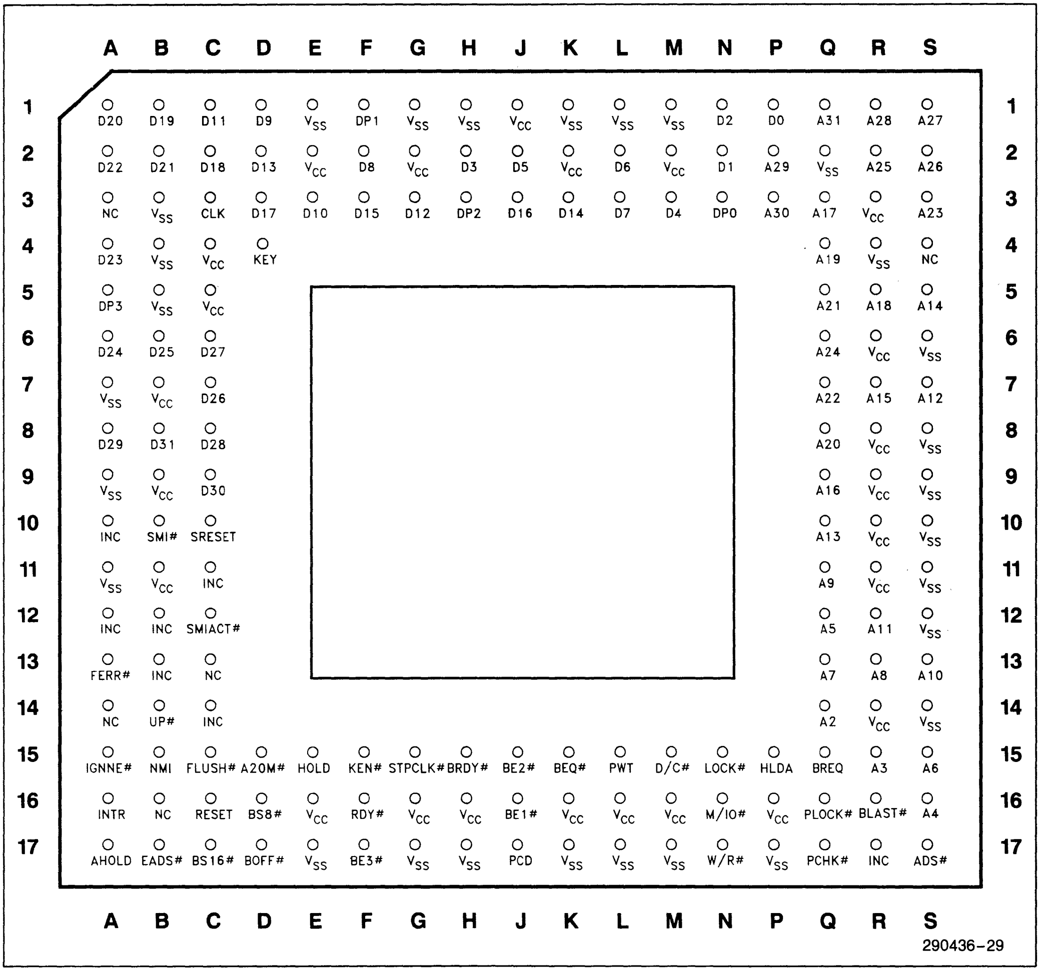
1.2 Pinouts

Refer to Figures 1-3 and 1-4 for an illustration of each of the two PGA packages. Figure 1-3 shows the 169-lead PGA package, while Figure 1-4 illustrates the 168-lead PGA package.

Table 1-1 cross-references the pin number to pin function for the 169-lead PGA package. Table 1-2 is a cross-reference for the 168-lead package.

Table 5-1 in Section 5 gives a brief description of the function of each pin.

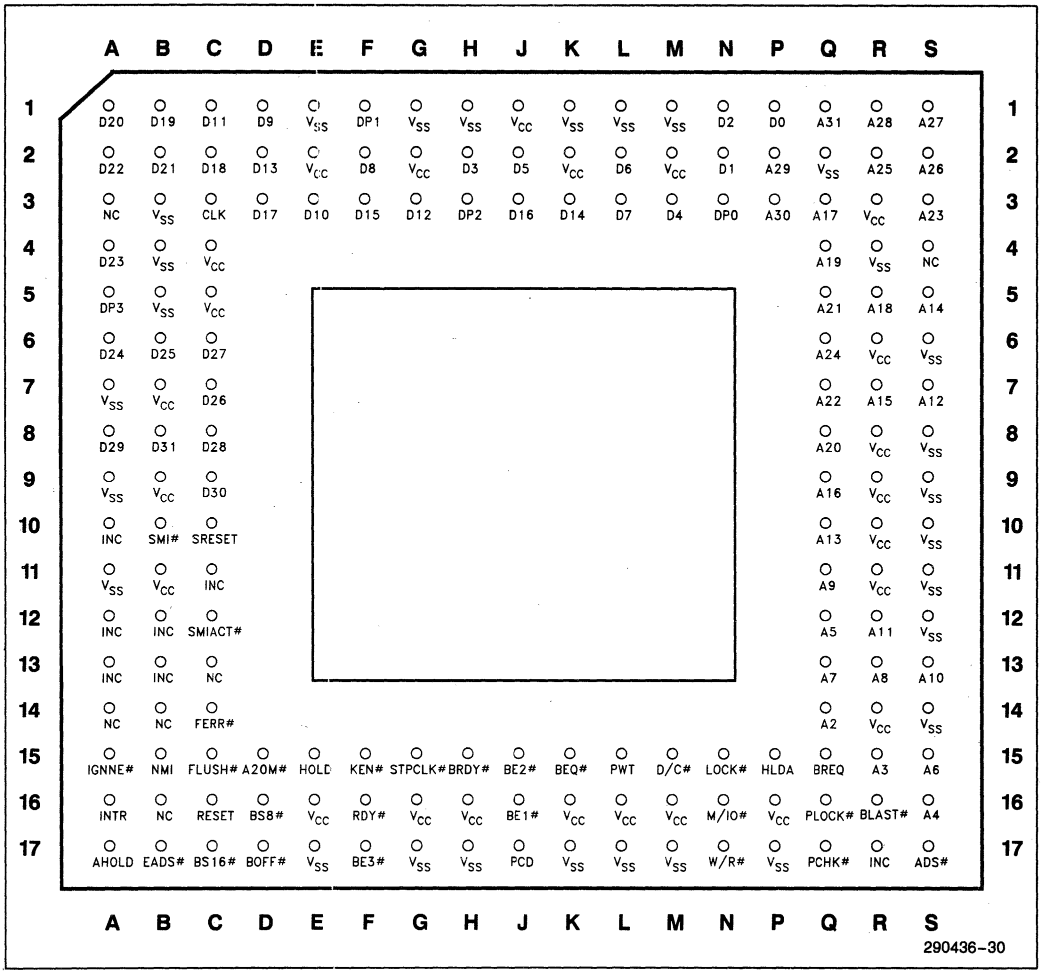
Refer to each specific OverDrive processor section for a description of any differences from the pinouts described in this section.



	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	
1	D20	D19	D11	D9	V_{SS}	DP1	V_{SS}	V_{SS}	V_{CC}	V_{SS}	V_{SS}	V_{SS}	D2	D0	A31	A28	A27	1
2	D22	D21	D18	D13	V_{CC}	D8	V_{CC}	D3	D5	V_{CC}	D6	V_{CC}	D1	A29	V_{SS}	A25	A26	2
3	NC	V_{SS}	CLK	D17	D10	D15	D12	DP2	D16	D14	D7	D4	DP0	A30	A17	V_{CC}	A23	3
4	D23	V_{SS}	V_{CC}	KEY											A19	V_{SS}	NC	4
5	DP3	V_{SS}	V_{CC}												A21	A18	A14	5
6	D24	D25	D27												A24	V_{CC}	V_{SS}	6
7	V_{SS}	V_{CC}	D26												A22	A15	A12	7
8	D29	D31	D28												A20	V_{CC}	V_{SS}	8
9	V_{SS}	V_{CC}	D30												A16	V_{CC}	V_{SS}	9
10	INC	SMI#	SRESET												A13	V_{CC}	V_{SS}	10
11	V_{SS}	V_{CC}	INC												A9	V_{CC}	V_{SS}	11
12	INC	INC	SMIACT#												A5	A11	V_{SS}	12
13	FERR#	INC	NC												A7	A8	A10	13
14	NC	UP#	INC												A2	V_{CC}	V_{SS}	14
15	IGNNE#	NMI	FLUSH#	A20M#	HOLD	KEN#	STPCLK#	BRDY#	BE2#	BEQ#	PWT	D/C#	LOCK#	HLDA	BREQ	A3	A6	15
16	INTR	NC	RESET	BS8#	V_{CC}	RDY#	V_{CC}	V_{CC}	BE1#	V_{CC}	V_{CC}	V_{CC}	M/IO#	V_{CC}	PLOCK#	BLAST#	A4	16
17	AHOLD	EADS#	BS16#	BOFF#	V_{SS}	BE3#	V_{SS}	V_{SS}	PCD	V_{SS}	V_{SS}	V_{SS}	W/R#	V_{SS}	PCHK#	INC	ADS#	17
	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	

290436-29

Figure 1-3. 169-Lead PGA Bottom View Pinout (ODP)



	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S
1	D20	D19	D11	D9	V_{SS}	DP1	V_{SS}	V_{SS}	V_{CC}	V_{SS}	V_{SS}	V_{SS}	D2	D0	A31	A28	A27
2	D22	D21	D18	D13	V_{CC}	D8	V_{CC}	D3	D5	V_{CC}	D6	V_{CC}	D1	A29	V_{SS}	A25	A26
3	NC	V_{SS}	CLK	D17	D10	D15	D12	DP2	D16	D14	D7	D4	DP0	A30	A17	V_{CC}	A23
4	D23	V_{SS}	V_{CC}												A19	V_{SS}	NC
5	DP3	V_{SS}	V_{CC}												A21	A18	A14
6	D24	D25	D27												A24	V_{CC}	V_{SS}
7	V_{SS}	V_{CC}	D26												A22	A15	A12
8	D29	D31	D28												A20	V_{CC}	V_{SS}
9	V_{SS}	V_{CC}	D30												A16	V_{CC}	V_{SS}
10	INC	SMI#	SRESET												A13	V_{CC}	V_{SS}
11	V_{SS}	V_{CC}	INC												A9	V_{CC}	V_{SS}
12	INC	INC	SMIACT#												A5	A11	V_{SS}
13	INC	INC	NC												A7	A8	A10
14	NC	NC	FERR#												A2	V_{CC}	V_{SS}
15	IGNNE#	NMI	FLUSH#	A20M#	HOLD	KEN#	STPCLK#	BRDY#	BE2#	BEQ#	PWT	D/C#	LOCK#	HLDA	BREQ	A3	A6
16	INTR	NC	RESET	BS8#	V_{CC}	RDY#	V_{CC}	V_{CC}	BE1#	V_{CC}	V_{CC}	V_{CC}	M/IO#	V_{CC}	PLOCK#	BLAST#	A4
17	AHOLD	EADS#	BS16#	BOFF#	V_{SS}	BE3#	V_{SS}	V_{SS}	PCD	V_{SS}	V_{SS}	V_{SS}	W/R#	V_{SS}	PCHK#	INC	ADS#

Figure 1-4. 168-Lead PGA Bottom View Pinout (ODPR)

Table 1-1. 169-Lead PGA Pin Cross Reference by Pin Name (ODP)

Address		Data		Control		Control		NC	V_{CC}	V_{SS}
A_2	Q14	D_0	P1	A20M#	D15	PLOCK#	Q16	A3	B7	A7
A_3	R15	D_1	N2	ADS#	S17	PWT	L15	A14	B9	A9
A_4	S16	D_2	N1	AHOLD	A17	RDY#	F16	B16	B11	A11
A_5	Q12	D_3	H2	BE0#	K15	RESET	C16	C13	C4	B3
A_6	S15	D_4	M3	BE1#	J16	SMI#	B10		C5	B4
A_7	Q13	D_5	J2	BE2#	J15	SMIACT#	C12		E2	B5
A_8	R13	D_6	L2	BE3#	F17	SRESET	C10		E16	E1
A_9	Q11	D_7	L3	BLAST#	R16	STPCLK#	G15		G2	E17
A_{10}	S13	D_8	F2	BOFF#	D17	UP#	B14		G16	G1
A_{11}	R12	D_9	D1	BRDY#	H15	W/R#	N17		H16	G17
A_{12}	S7	D_{10}	E3	BREQ#	Q15				J1	H1
A_{13}	Q10	D_{11}	C1	BS8#	D16				K2	H17
A_{14}	S5	D_{12}	G3	BS16#	C17				K16	K1
A_{15}	R7	D_{13}	D2	CLK	C3	**Position**			L16	K17
A_{16}	Q9	D_{14}	K3	D/C#	M15				M2	L1
A_{17}	Q3	D_{15}	F3	DP0	N3	KEY	D4		M16	L17
A_{18}	R5	D_{16}	J3	DP1	F1	PLUG	D5		P16	M1
A_{19}	Q4	D_{17}	D3	DP2	H3	PLUG	D13		R3	M17
A_{20}	Q8	D_{18}	C2	DP3	A5	PLUG	D14		R6	P17
A_{21}	Q5	D_{19}	B1	EADS#	B17	PLUG	E4	**INC**	R8	Q2
A_{22}	Q7	D_{20}	A1	FERR#	A13	PLUG	E14		R9	R4
A_{23}	S3	D_{21}	B2	FLUSH#	C15	PLUG	N4	A10	R10	S6
A_{24}	Q6	D_{22}	A2	HLDA	P15	PLUG	N14	A12	R11	S8
A_{25}	R2	D_{23}	A4	HOLD	E15	PLUG	P4	B12	R14	S9
A_{26}	S2	D_{24}	A6	IGNNE#	A15	PLUG	P5	B13		S10
A_{27}	S1	D_{25}	B6	INTR	A16	PLUG	P13	C11		S11
A_{28}	R1	D_{26}	C7	KEN#	F15	PLUG	P14	C14		S12
A_{29}	P2	D_{27}	C6	LOCK#	N15			R17		S14
A_{30}	P3	D_{28}	C8	M/IO#	N16			S4		
A_{31}	Q1	D_{29}	A8	NMI	B15					
		D_{30}	C9	PCD	J17					
		D_{31}	B8	PCHK#	Q17					

NOTES:

1. All NC pins must remain unconnected.
2. Refer to each specific OverDrive section for differences in pin functions.
3. NC = No Connection.
4. INC = Internal No Connect.

Table 1-2. 168-Lead PGA Pin Cross Reference by Pin Name (ODPR)

Address		Data		Control		Control		N/C	V_{CC}	V_{SS}
A_2	Q14	D_0	P1	A20M#	D15	PLOCK#	Q16	A3	B7	A7
A_3	R15	D_1	N2	ADS#	S17	PWT	L15	A14	B9	A9
A_4	S16	D_2	N1	AHOLD	A17	RDY#	F16	B14	B11	A11
A_5	Q12	D_3	H2	BE0#	K15	RESET	C16	B16	C4	B3
A_6	S15	D_4	M3	BE1#	J16	SMI#	B10	C13	C5	B4
A_7	Q13	D_5	J2	BE2#	J15	SMIACT#	C12		E2	B5
A_8	R13	D_6	L2	BE3#	F17	SRESET	C10		E16	E1
A_9	Q11	D_7	L3	BLAST#	R16	STPCLK#	G15		G2	E17
A_{10}	S13	D_8	F2	BOFF#	D17	UP#	C11		G16	G1
A_{11}	R12	D_9	D1	BRDY#	H15	W/R#	N17		H16	G17
A_{12}	S7	D_{10}	E3	BREQ#	Q15				J1	H1
A_{13}	Q10	D_{11}	C1	BS8#	D16				K2	H17
A_{14}	S5	D_{12}	G3	BS16#	C17				K16	K1
A_{15}	R7	D_{13}	D2	CLK	C3				L16	K17
A_{16}	Q9	D_{14}	K3	D/C#	M15				M2	L1
A_{17}	Q3	D_{15}	F3	DP0	N3				M16	L17
A_{18}	R5	D_{16}	J3	DP1	F1				P16	M1
A_{19}	Q4	D_{17}	D3	DP2	H3				R3	M17
A_{20}	Q8	D_{18}	C2	DP3	A5			**INC**	R6	P17
A_{21}	Q5	D_{19}	B1	EADS#	B17				R8	Q2
A_{22}	Q7	D_{20}	A1	FERR#	C14			A10	R9	R4
A_{23}	S3	D_{21}	B2	FLUSH#	C15			A12	R10	S6
A_{24}	Q6	D_{22}	A2	HLDA	P15			A13	R11	S8
A_{25}	R2	D_{23}	A4	HOLD	E15			B12	R14	S9
A_{26}	S2	D_{24}	A6	IGNNE#	A15			B13		S10
A_{27}	S1	D_{25}	B6	INTR	A16			R17		S11
A_{28}	R1	D_{26}	C7	KEN#	F15			S4		S12
A_{29}	P2	D_{27}	C6	LOCK#	N15					S14
A_{30}	P3	D_{28}	C8	M/IO#	N16					
A_{31}	Q1	D_{29}	A8	NMI	B15					
		D_{30}	C9	PCD	J17					
		D_{31}	B8	PCHK#	Q17					

NOTES:

1. All NC pins must remain unconnected.
2. Refer to each specific OverDrive section for differences in pin functions.
3. NC = No Connection.
4. INC = Internal No Connect.

2.0 IntelSX2™ OverDrive™ PROCESSOR FOR Intel486™ SX MICROPROCESSOR-BASED SYSTEMS

- **Processor Upgrade for Intel486™ SX Processor-Based Systems**
  - **Single-Chip Upgrade**
- **169-Lead Pin Grid Array Package**
  - **Pin Compatible With Intel487 SX Math CoProcessor**
  - **169th Alignment Pin Ensures Proper Chip Orientation**
- **High Integration Enables On-Chip**
  - **8 KByte Code and Data Cache**
  - **Paged, Virtual Memory Management**
- **Complete 32-Bit Architecture**
  - **Address and Data Busses**
  - **Registers**
  - **8-, 16-, 32-Bit Data Types**
- **Utilizes Intel486™ SX2 Speed-Doubling Technology**
  - **Processor Core Runs at Twice the Frequency of the System Bus**
  - **Compatible with 25 MHz, 20 MHz and 16 MHz Systems**
- **Binary Compatible with Large Installed Software and Operating System Base**
  - **MS-DOS, OS/2™, Windows**
  - **UNIX System V/386**
  - **IRMX, IRMK™ Kernals**
- **High Performance Design**
  - **Core Clock Speed up to 50 MHz**
  - **80 Mbyte/sec Burst Bus**
  - **CHMOS V Process Technology**
- **SL Enhanced Intel486™ Microprocessor Features Included On-Chip**

The IntelSX2 OverDrive processor for Intel486 SX microprocessor-based personal computers is a processor upgrade that offers excellent price/performance. Based on the new Intel486 SX2 technology, the IntelSX2 OverDrive processor integrates an integer unit, a memory management unit and an 8 KByte cache on a single chip.

Using the Intel486 SX2's speed doubling technology, the IntelSX2 OverDrive processor operates internally at twice the speed of the system bus. This allows users of Intel486 SX microprocessor-based systems to double the frequency of their computer's processor by adding a single chip, without upgrading or modifying any other system components. For example, adding an IntelSX2 OverDrive processor to an Intel486 SX 25 MHz system will double the processors internal operating speed to 50 MHz.

The IntelSX2 OverDrive processor is designed to either replace the existing processor or be installed into the Math CoProcessor socket of Intel486 SX microprocessor-based systems.

The IntelSX2 OverDrive processor is based on the Intel486 SX2 microprocessor technology. This technology doubles the clock speed of the internal processor core, while interfacing with the system at the same external clock speed. When installed in a 25 MHz Intel486 SX microprocessor-based system, the internal processor core, integer unit and cache operate at 50 MHz, while the speed of the external bus remains at 25 MHz. This provides increased processor performance while maintaining compatibility with the existing system design.

The IntelSX2 OverDrive processor is currently available in one product version. The 50 MHz IntelSX2 OverDrive processor is designed to upgrade 25 MHz, 20 MHz and 16 MHz Intel486 SX microprocessor-based systems. The speed doubling technology will thereby double the internal speed of the processor to 50 MHz, 40 MHz and 32 MHz, respectively.

This product comes with a (0.25″ high) heat sink attached to the standard 169-lead PGA package to aid in heat dissipation.

The IntelSX2 OverDrive processor includes the power management features of the SL Enhanced Intel486 SX2 microprocessor and is binary compatible with a large base of software based on DOS, OS/2, Windows and Unix operating systems.

For more detailed information about the operation of the IntelSX2 OverDrive processor, refer to the Intel SX2 microprocessor data book (Order #241731-001).

2.1 Socket Configurations

The IntelSX2 OverDrive processor can be used to upgrade a single-socket Intel486 SX microprocessor-based system by replacing the original processor with the OverDrive processor. In a two-socket system, the IntelSX2 OverDrive processor can simply be placed into the empty OverDrive processor socket, while the original processor remains in its socket. Figure 2-1 illustrates this.

Figure 1-3 shows the bottom-view (pin-side) pinout diagram of the 169-lead Pin Grid Array (PGA) package. Table 1-1 cross references the device's pin numbers to the pin names.

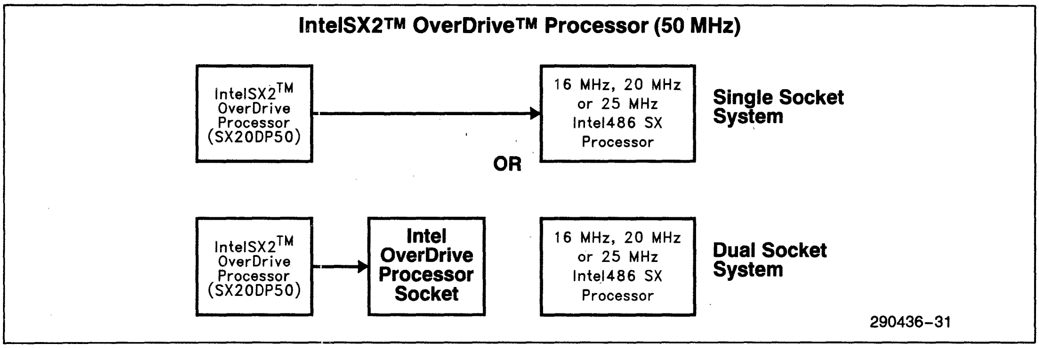

IntelSX2™ OverDrive™ Processor (50 MHz)
IntelSX2™ OverDrive Processor (SX20DP50)
16 MHz, 20 MHz or 25 MHz Intel486 SX Processor
Single Socket System
OR
IntelSX2™ OverDrive Processor (SX20DP50)
Intel OverDrive Processor Socket
16 MHz, 20 MHz or 25 MHz Intel486 SX Processor
Dual Socket System
290436-31

Figure 2-1. IntelSX2™ OverDrive™ Processor Sockets

2.2 Differences between the IntelSX2™ OverDrive™ Processor and the Intel486™ SX Processor

A complete pinout diagram for the 169-lead PGA package can be seen in Figure 1-3. This diagram shows the standard pinout for the package, which is utilized by all three of the OverDrive processors. Table 1-1 contains a complete pin description for the same package. Again, this is the standard pin description for the package which is used by all three OverDrive processors.

Table 2-1 identifies the differences between the standard ODPR pinout and the pinout used on the IntelSX2 OverDrive processor.

Table 2-1. IntelSX2™ Pinout Differences

Package (OverDrive Processor)	Pin A13	Pin C14	Pin A15
ODP (IntelDX2, IntelDX4)	FERR#	INC	IGNNE#
ODPR (IntelDX2, IntelDX4)	INC	FERR#	IGNNE#
ODP (IntelSX2)	NOFPU	INC	INC

3.0 IntelDX2™ OverDrive™ PROCESSOR FOR Intel486™ SX AND DX MICROPROCESSOR-BASED SYSTEMS

- **Processor Upgrade for Intel486™ SX and DX Processor-Based Systems**
  - **—Single-Chip Upgrade**
  - **—Increases Both Integer and Floating Point Performance**
- **Two Package Variations to Support Systems with and without an OverDrive™ Processor Socket**
- **169-Lead Pin Grid Array Package**
  - **—Pin Compatible with Intel487™ SX Math CoProcessor**
  - **—169th Alignment Pin Ensures Proper Chip Orientation**
- **168-Lead Pin Grid Array Package**
  - **—Pin Compatible with Intel486™ DX Processor**
- **Utilizes Intel486™ DX2 Speed-Doubling Technology**
  - **—Processor Core Runs at Twice the Frequency of the System Bus**
  - **—Compatible with 33, 25, 20 and 16 MHz Systems**
- **Floating Point Math Unit Included On-Chip**
- **High Integration Enables On-Chip**
  - **—8 KByte Code and Data Cache**
  - **—Paged, Virtual Memory Management**
- **Binary Compatible with Large Installed Software Base**
  - **—MS-DOS, OS/2™, Windows**
  - **—UNIX System V/386**
  - **—IRMX, IRMK™ Kernals**
- **High Performance Design**
  - **—Core Clock Speed up to 66 MHz**
  - **—106 Mbyte/sec Burst Bus**
  - **—CHMOS V Process Technology**
- **Complete 32-Bit Architecture**
  - **—Address and Data Busses**
  - **—Registers**
  - **—8-, 16-, 32-Bit Data Types**
- **Compatible with Intel SL Enhanced Features**

The Intel486 DX2 OverDrive processor for Intel486SX and DX microprocessor-based systems is a powerful, processor upgrade that offers excellent price/performance. Based on Intel's Intel486 DX2 technology, the IntelDX2 OverDrive processor integrates an integer unit, a floating point unit, a memory management unit, SL Enhanced features and an 8 KByte cache on a single chip.

Using the Intel486 DX2's speed doubling technology, the IntelDX2 OverDrive processor operates internally at twice the speed of the system bus. This allows users of Intel486 SX and DX microprocessor-based systems to double the frequency of their computer's processor by adding a single chip, without upgrading or modifying any other system components. For example, adding an IntelDX2 OverDrive processor to an Intel486 DX 33 MHz system will double the processors internal operating speed to 66 MHz.

The IntelDX2 OverDrive processor is based on the Intel 486 DX2 microprocessor technology. This technology doubles the clock speed of the internal processor core, while interfacing with the system at the same external clock speed. When installed in a 33 MHz Intel486 SX or DX microprocessor-based system, the internal processor core, integer unit, floating point unit and cache operate at 66 MHz, while the speed of the external bus remains at 33 MHz. This provides increased processor performance while maintaining compatibility with the existing system design.

The IntelDX2 OverDrive processor is currently available in four product versions, which consist of two speed options (50 MHz and 66 MHz) and two package options (168-lead Pin Grid Array (PGA) and 169-lead PGA).

The 50 MHz IntelDX2 OverDrive processor is designed to upgrade 25 MHz Intel486 DX microprocessor-based systems and 16 MHz, 20 MHz and 25 MHz Intel486 SX microprocessor-based systems. The 66 MHz IntelDX2 OverDrive processor is designed to upgrade 33 MHz Intel486 SX and DX microprocessor-based systems. Figures 3-1 and 3-2 illustrate this. The speed doubling technology doubles the internal speed of the processor to twice the bus speed of the existing system.

These products come with a (0.25″ high) heat sink attached to the standard 169-lead PGA or 168-lead PGA package to aid in heat dissipation. All IntelDX2 OverDrive processors are binary compatible with a large base of software based on DOS, OS/2, Windows and Unix operating systems.

For more detailed information about the operation of the IntelDX2 OverDrive processor, refer to the Intel486 DX2 microprocessor data book (Order #241731-001).

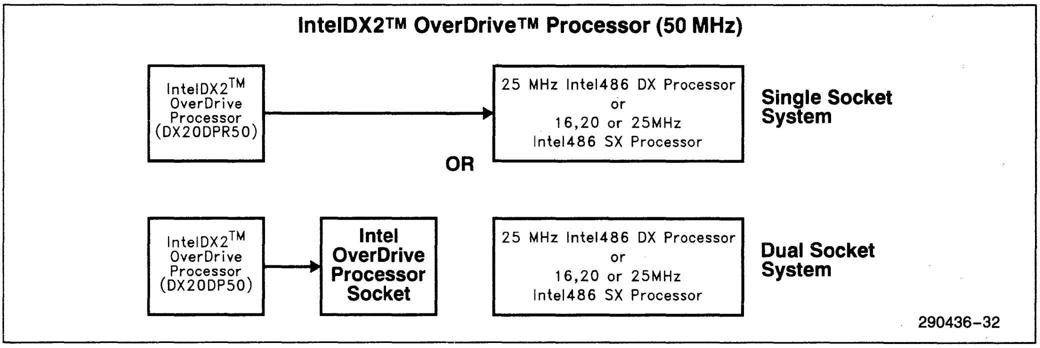

IntelDX2™ OverDrive™ Processor (50 MHz)
IntelDX2™ OverDrive Processor (DX2ODPR50)
25 MHz Intel486 DX Processor or 16,20 or 25MHz Intel486 SX Processor
Single Socket System
OR
IntelDX2™ OverDrive Processor (DX2ODP50)
Intel OverDrive Processor Socket
25 MHz Intel486 DX Processor or 16,20 or 25MHz Intel486 SX Processor
Dual Socket System
290436–32

Figure 3-1. 50 MHz IntelDX2™ OverDrive™ Processor Sockets

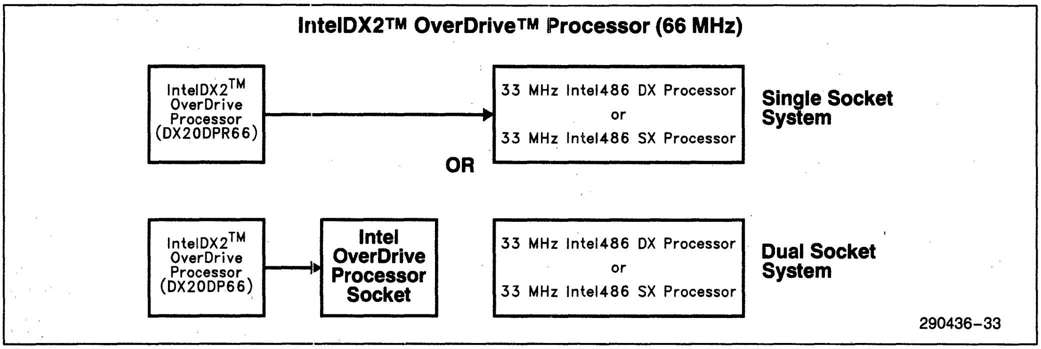

IntelDX2™ OverDrive™ Processor (66 MHz)
IntelDX2™ OverDrive Processor (DX2ODPR66)
33 MHz Intel486 DX Processor or 33 MHz Intel486 SX Processor
Single Socket System
OR
IntelDX2™ OverDrive Processor (DX2ODP66)
Intel OverDrive Processor Socket
33 MHz Intel486 DX Processor or 33 MHz Intel486 SX Processor
Dual Socket System
290436–33

Figure 3-2. 66 MHz IntelDX2™ OverDrive™ Processor Sockets

3.1 Socket Configurations

Both single-socket and two-socket system configurations can be upgraded with the IntelDX2 OverDrive processor. In a single-socket Intel486 microprocessor-based system, this is done by replacing the original processor with the OverDrive processor. In a two-socket system, the IntelDX2 OverDrive processor can simply be placed into the empty OverDrive processor socket. Figures 3-1 and 3-2 illustrate this.

3.2 169-Lead PGA Device (DX2ODP)

The 169-lead version of the IntelDX2 OverDrive processor is currently available in two speeds; 50 MHz (DX2ODP50) and 66 MHz (DX2ODP66). The 169-lead versions are designed to be used in two-socket systems and contain a key pin to assure proper orientation of the device (refer to Figures 3-1 and 3-2). The OverDrive processor is simply inserted into the OverDrive processor socket, while the original processor remains in its socket.

Figure 1-3 shows the bottom-view (pin-side) pinout diagram of the 169-lead Pin Grid Array (PGA) package. Table 1-1 cross references the device's pin numbers to the pin names.

3.3 168-Lead PGA Device (DX2ODPR)

The 168-lead version of the IntelDX2 OverDrive processor is currently available in two speeds; 50 MHz (DX2ODPR50) and 66 MHz (DX2ODPR66). The 168-lead versions are designed to be used in a single-socket system (refer to Figures 3-1 and 3-2). The existing processor is removed and the upgrade processor is simply inserted into the same socket.

Figure 1-4 shows the bottom-view (pin-side) pinout diagram of the 168-lead Pin Grid Array (PGA) package. Table 1-2 cross references the device's pin numbers to the pin names.

4.0 IntelDX4™ OverDrive™ PROCESSOR FOR Intel486™ SX AND DX MICROPROCESSOR-BASED SYSTEMS

- **Processor Upgrade for Intel486™ SX and DX Processor-Based Systems**
  - **Single-Chip Upgrade**
  - **Increases Both Integer and Floating Point Performance**
- **Two Package Variations to Support Systems with and without an OverDrive™ Processor Socket**
- **169-Lead Pin Grid Array Package**
  - **Pin Compatible with Intel487™ SX Math CoProcessor**
  - **169th Alignment Pin Ensures Proper Chip Orientation**
- **168-Lead Pin Grid Array Package**
  - **Pin Compatible with Intel486™ DX Processor**
- **High Integration Enables On-Chip**
  - **16 KByte Code and Data Cache**
  - **Paged, Virtual Memory Management**
- **Floating Point Math Unit Included On-Chip**
- **Utilizes Intel486™ DX4 Speed-Tripling Technology**
  - **Processor Core Runs at Three Times the Frequency of the System Bus**
  - **Compatible with 33, 25, 20 and 16 MHz Systems**
- **Binary Compatible with Large Installed Software and Operating System Base**
  - **MS-DOS, OS/2™, Windows**
  - **UNIX System V/386**
  - **IRMX, IRMK™ Kernals**
- **High Performance Design**
  - **Core Clock Speed up to 100 MHz**
  - **CHMOS V Process Technology**
- **Complete 32-Bit Architecture**
  - **Address and Data Busses**
  - **Registers**
  - **8-, 16-, 32-Bit Data Types**
- **SL Enhanced Intel486™ Microprocessor Features Included On-Chip**

The IntelDX4 OverDrive processor is a high performance, processor upgrade for Intel486 SX and DX microprocessor-based systems. It operates at a maximum internal core frequency of 100 MHz and is available in two package versions. When installed in a system, the IntelDX4 OverDrive processor significantly increases both the integer and floating point performance.

The IntelDX4 OverDrive processor offers several new features not found in the previous Intel OverDrive processors for Intel486 SX and DX microprocessor-based systems. It has 16 KByte on-chip cache and the internal core operates at 3x (speed tripled) the external clock frequency. The IntelDX4 OverDrive processor supports System Management Mode (SMM) and Stop Clock Mode. The SMM and Stop Clock Mode, identical to those implemented in SL Enhanced Intel486 SX and DX microprocessors, make the IntelDX4 OverDrive processor compatible with the advanced power management, system security and device emulation features of SL Enhanced systems.

The underlying technology behind the IntelDX4 OverDrive processor is the IntelDX4 microprocessor core with on-package voltage regulation. This allows the OverDrive processor to plug directly into existing 5V systems.

The IntelDX4 OverDrive processor is based on the Intel486 DX4 microprocessor technology. This technology triples the clock speed of the internal processor core, while interfacing with the system at the same external clock speed. When installed in a 33 MHz Intel486 SX or DX microprocessor-based system, the internal processor core, integer unit, floating point unit and cache operate at 100 MHz, while the speed of the external bus remains at 33 MHz. This provides increased processor performance while maintaining compatibility with the existing system design. In addition, the internal cache has been doubled to 16 KBytes.

The IntelDX4 OverDrive processor is currently available in four product versions, which consist of two speed options (75 MHz and 100 MHz) and two package options (168-lead Pin Grid Array (PGA) and 169-lead PGA).

The 100 MHz OverDrive processors are designed to upgrade 33 MHz Intel486 SX and DX microprocessor-based systems. The 75 MHz OverDrive processors are designed to upgrade 25 MHz Intel486 DX microprocessor-based systems and 16 MHz, 20 MHz and 25 MHz Intel486 SX microprocessor-based systems. Figures 3-1 and 3-2 illustrate this. The speed tripling technology will triple the internal speed of the processor to three times the bus speed of the existing system.

These products come with a (0.6″ high) heat sink attached to the standard 169-lead PGA or 168-lead PGA package to aid in heat dissipation. Refer to Sections 10.0 and 11.0 for clearance and thermal requirements. All IntelDX4 OverDrive processors are binary compatible with a large base of software based on DOS, OS/2, Windows and Unix operating systems.

For more detailed information about the operation of the IntelDX4 OverDrive processor, refer to the Intel DX4 microprocessor data book (Order #241944-001).

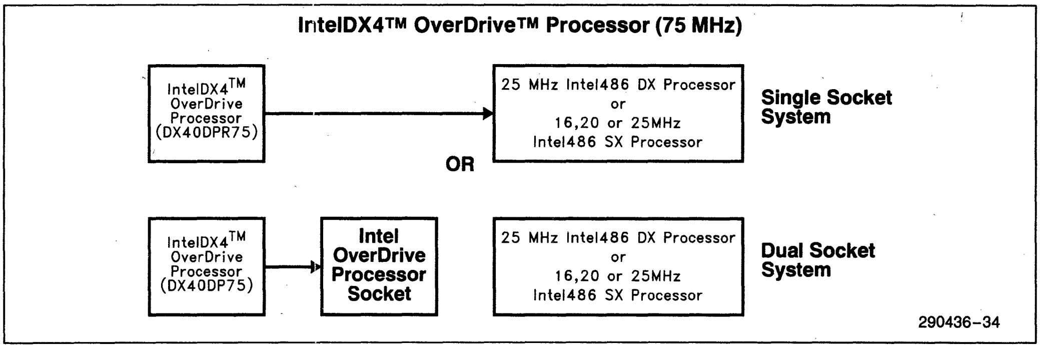

IntelDX4™ OverDrive™ Processor (75 MHz)
IntelDX4™ OverDrive Processor (DX4ODPR75)
25 MHz Intel486 DX Processor or 16,20 or 25MHz Intel486 SX Processor
Single Socket System
OR
IntelDX4™ OverDrive Processor (DX4ODP75)
Intel OverDrive Processor Socket
25 MHz Intel486 DX Processor or 16,20 or 25MHz Intel486 SX Processor
Dual Socket System
290436–34

Figure 4-1. 75 MHz IntelDX4™ OverDrive™ Processor Sockets

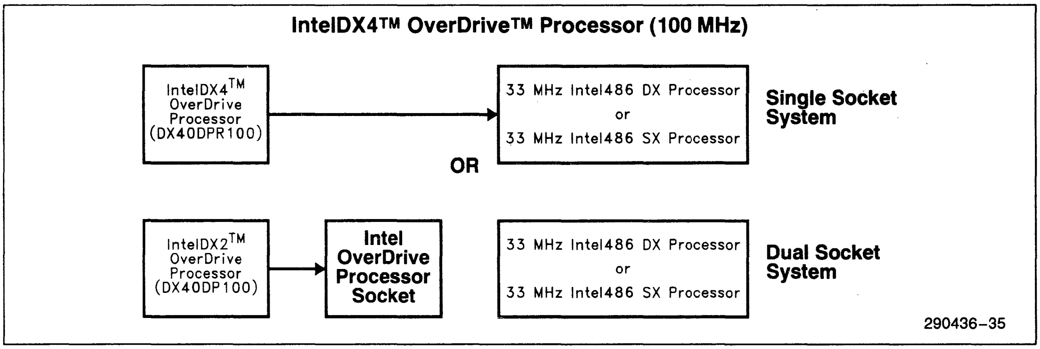

IntelDX4™ OverDrive™ Processor (100 MHz)
IntelDX4™ OverDrive Processor (DX4ODPR100)
33 MHz Intel486 DX Processor or 33 MHz Intel486 SX Processor
Single Socket System
OR
IntelDX2™ OverDrive Processor (DX4ODP100)
Intel OverDrive Processor Socket
33 MHz Intel486 DX Processor or 33 MHz Intel486 SX Processor
Dual Socket System
290436-35

Figure 4-2. 100 MHz IntelDX4™ OverDrive™ Processor Sockets

4.1 Socket Configurations

Both single-socket and two-socket system configurations can be upgraded with the IntelDX4 OverDrive processor. In a single-socket Intel486 microprocessor-based system, this is done by replacing the original processor with the OverDrive processor. In a two-socket system, the IntelDX4 OverDrive processor can simply be placed into the empty OverDrive processor socket. Figures 4-1 and 4-2 illustrate this.

4.2 169-Lead PGA Device (DX4ODP)

The 169-lead version of the IntelDX4 OverDrive processor is currently available in two speeds; 75 MHz (DX4ODP75) and 100 MHz (DX4ODP100). The 169-lead versions are designed to be used in a two-socket system and contain a key pin to assure proper orientation of the device (refer to Figures 4-1 and 4-2). The processor is simply inserted into the OverDrive processor socket, while the original processor remains in its socket.

Figure 1-3 shows the bottom-view (pin-side) pinout diagram of the 169-lead Pin Grid Array (PGA) package. Table 1-1 cross references the device's pin numbers to the pin names.

4.3 168-Lead PGA Device

The 168-lead version of the IntelDX4 OverDrive processor is currently available in two speeds; 75 MHz (DX4ODPR75) and 100 MHz (DX4ODPR100). The 168-lead versions are designed to be used in a single-socket system (refer to Figures 4-1 and 4-2). The existing processor is removed and the upgrade processor is simply inserted into the same socket.

Figure 1-4 shows the bottom-view (pin-side) pinout diagram of the 168-lead Pin Grid Array (PGA) package. Table 1-2 cross references the device's pin numbers to the pin names.

5.0 PIN DESCRIPTIONS

Tables 5-1 through 5-4 list pin descriptions of the signals present on the IntelSX2, Intel DX2 and IntelDX4 OverDrive processors.

Table 5-1. Pin Descriptions

Symbol	Type	Name and Function
CLK	I	*Clock* provides the fundamental timing for the bus interface unit and is multiplied by two (2x) for the IntelSX2 and IntelDX2 OverDrive Processors or three (3x) for the IntelDX4 OverDrive Processor to provide the internal frequency for the Intel OverDrive processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 A2–A3	I/O O	A31–A2 are the *address lines* of the processor. A31–A2, together with the byte enables 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	These are the *data lines* for the Intel OverDrive processor. 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 Intel OverDrive processor. Even parity information must be driven back into the microprocessor on the data parity pins with the same timing as read information to insure that the correct parity check status is indicated by the Intel OverDrive 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 pullup resistor in systems which 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 microprocessor. At all other times PCHK# is inactive (HIGH). PCHK# is never floated.

Table 5-1. Pin Descriptions (Continued)

<table>
<tr><th>Symbol</th><th>Type</th><th>Name and Function</th></tr>
<tr><td colspan="3"><b>BUS CYCLE DEFINITION</b></td></tr>
<tr><td>M/IO#
D/C#
W/R#</td><td>O
O
O</td><td>The <i>memory/input-output, data/control</i> and <i>write/read</i> lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted.
<table>
<tr><th>M/IO#</th><th>D/C#</th><th>W/R#</th><th>Bus Cycle Initiated</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Halt/Special Cycle</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>I/O Read</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>I/O Write</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Code Read</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Memory Read</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Memory Write</td></tr>
</table>
The bus definition signals are not driven during bus hold and follow the timing of the address bus. Refer to Section 7.2.11 for a description of the special bus cycles.</td></tr>
<tr><td>LOCK#</td><td>O</td><td>The <i>bus lock</i> pin indicates that the current bus cycle is locked. The Intel OverDrive 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 RDY# 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.</td></tr>
<tr><td>PLOCK#</td><td>O</td><td>The <i>pseudo-lock</i> pin indicates that the current bus transaction requires more than one bus cycle to complete. Examples of such operations are floating point long reads and writes (64 bits), segment table descriptor reads (64 bits), in addition to cache line fills (128 bits). The Intel OverDrive 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, 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 RDY# is returned. PLOCK# is active LOW and is not driven during bus hold.</td></tr>
<tr><td colspan="3"><b>BUS CONTROL</b></td></tr>
<tr><td>ADS#</td><td>O</td><td>The <i>address status</i> 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.</td></tr>
<tr><td>RDY#</td><td>I</td><td>The <i>non-burst ready</i> 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 Intel OverDrive 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 pullup resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.</td></tr>
</table>

Table 5-1. Pin Descriptions (Continued)

Symbol	Type	Name and Function
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 microprocessor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely interrupted BRDY# is active LOW and is provided with a small pullup 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 processor to begin execution at a known state. Reset is asynchronous, but must meet setup and hold times t20 and t21 for recognition in any specific clock. The processor cannot begin execution of instructions until at least 1 ms after V_{CC} and CLK have reached their proper AC and DC specifications. However, for soft resets, RESET should remain active for at least 15 CLK periods. The RESET pin should remain active during this time to ensure proper processor operation. RESET is active HIGH. RESET sets the SMBASE descriptor to a default address of 30000H. If the system uses SMBASE relocation, then the SRESET pin should be used for soft resets.
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# (I) is asserted, SRESET will not have an effect on the host microprocessor. 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.

Table 5-1. Pin Descriptions (Continued)

Symbol	Type	Name and Function
INTERRUPTS (Continued)		
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 asynchronous but setup and hold times t_{20} and t_{21} must be met to ensure recognition in any specific clock.
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 Intel OverDrive 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 pulldown 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 pulldown resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
BUS ARBITRATION		
BREQ	O	The *internal cycle pending* signal indicates that the Intel OverDrive processor has internally generated a bus request. BREQ is generated whether or not the Intel OverDrive 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 Intel OverDrive processor bus. In response to HOLD going active the Intel OverDrive 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 Intel OverDrive processor will remain in this state until HOLD is deasserted. HOLD is active high and is not provided with an internal pulldown 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 Intel OverDrive processor has given the bus to another local bus master. HLDA is driven active in the same clock that the Intel OverDrive processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.

Table 5-1. Pin Descriptions (Continued)

Symbol	Type	Name and Function
BUS ARBITRATION (Continued)		
BOFF#	I	The *backoff* input forces the Intel OverDrive processor to float its bus in the next clock. The microprocessor 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 microprocessor 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 Intel OverDrive processor's address bus for a cache invalidation cycle. The Intel OverDrive 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 pulldown 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 Intel OverDrive 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 pullup 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 Intel OverDrive processor generates a cycle that can be cached and KEN# is active, 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 pullup resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	The *cache flush* input forces the Intel OverDrive 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. FLUSH# being sampled low in the clock before the falling edge of RESET causes the Intel OverDrive processor to enter the tri-state test mode.
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 or page directory entry. If paging is disabled or for cycles that are not paged, PWT and PCD reflect the state of the PWT and PCD bits in control register 3. 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.
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.

Table 5-1. Pin Descriptions (Continued)

Symbol	Type	Name and Function
NUMERIC ERROR REPORTING (Continued)		
IGNNE#	I	When the *ignore numeric error* pin is asserted the Intel OverDrive processor will ignore a numeric error and continue executing non-control floating point instructions, but FERR# will still be activated by the Intel OverDrive processor. When IGNNE# is deasserted the Intel OverDrive 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 pullup resistor. IGNNE# is asynchronous but setup and hold times t_{20} and t_{21} must be met to insure recognition on any specific clock.
BUS SIZE CONTROL		
BS16# BS8#	I I	The *bus size 16* and *bus size 8* pins (bus sizing pins) cause the Intel OverDrive 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 Intel OverDrive processor to determine the bus size. These signals are active LOW and are provided with internal pullup 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 Intel OverDrive 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. 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.
i486 DX AND i486 SX PROCESSOR INTERFACE		
UP#(1,2)	O	The *upgrade present* pin is used to signal the Intel486 Microprocessor to float its outputs and get off the bus. It is active low and is never floated. UP# is driven low at power-up and remains active for the entire duration of the Upgrade Processor operation.
KEY PIN		
KEY(2)		The KEY pin is an electrically non-functional pin which is used to ensure correct Upgrade Processor orientation in a 169-pin socket.

NOTE:

1. The UP# pin was previously named the MP# pin in the i486 SX Microprocessor/i487 SX Math CoProcessor data book. The functionality is the same, only the name has changed.
2. The UP# input pin and KEY pin are not defined on the OverDrive processor for replacement of PGA Intel486 DX Microprocessor (ODPR).

Table 5-2. Output Pins

Name	Active Level	When Floated
BREQ	HIGH	
HLDA	HIGH	
BE0#-BE3#	LOW	Bus Hold
PWT, PCD	HIGH	Bus Hold
W/R#, D/C#, M/IO#	HIGH	Bus Hold
LOCK#	LOW	Bus Hold
PLOCK#	LOW	Bus Hold
ADS#	LOW	Bus Hold
BLAST#	LOW	Bus Hold
PCHK#	LOW	
FERR#	LOW	
SMIACT#	LOW	
UP#	LOW	
A2-A3	HIGH	Bus, Address Hold

Table 5-3. Input Pins

Name	Active Level	Synchronous/ Asynchronous
CLK		
RESET	HIGH	Asynchronous
HOLD	HIGH	Synchronous
AHOLD	HIGH	Synchronous
EADS#	LOW	Synchronous
BOFF#	LOW	Synchronous
FLUSH#	LOW	Asynchronous
A20M#	LOW	Asynchronous
BS16#, BS8#	LOW	Synchronous
KEN#	LOW	Synchronous
RDY#	LOW	Synchronous
BRDY#	LOW	Synchronous
INTR	HIGH	Asynchronous
NMI	HIGH	Asynchronous
SRESET	HIGH	Asynchronous
SMI#	LOW	Asynchronous
STPCLK#	LOW	Asynchronous
IGNNE#	LOW	Asynchronous

Table 5-4. Input/Output Pins

Name	Active Level	When Floated
D0-D31	HIGH	Bus Hold
DP0-DP3	HIGH	Bus Hold
A4-A31	HIGH	Bus, Address Hold

5.1 Architecture Block Diagram

Figure 5-1 shows a block diagram of the Intel OverDrive Processor Architecture. There are a few minor architectural differences between each of the OverDrive processors. These differences are summarized below, with respect to Figure 5-1.

The IntelSX2 OverDrive processor does not contain the floating point unit located in the lower left of the diagram.

The IntelDX4 OverDrive processor contains a clock tripling circuit, as opposed to the clock doubling circuit used in the IntelSX2 and IntelDX2 OverDrive processors. This is located in the upper left of the diagram.

The IntelDX4 OverDrive processor contains a 16 KByte cache, as opposed to the 8 KByte cache used in the IntelSX2 and IntelDX2 OverDrive processors. This is located in the center of the diagram.

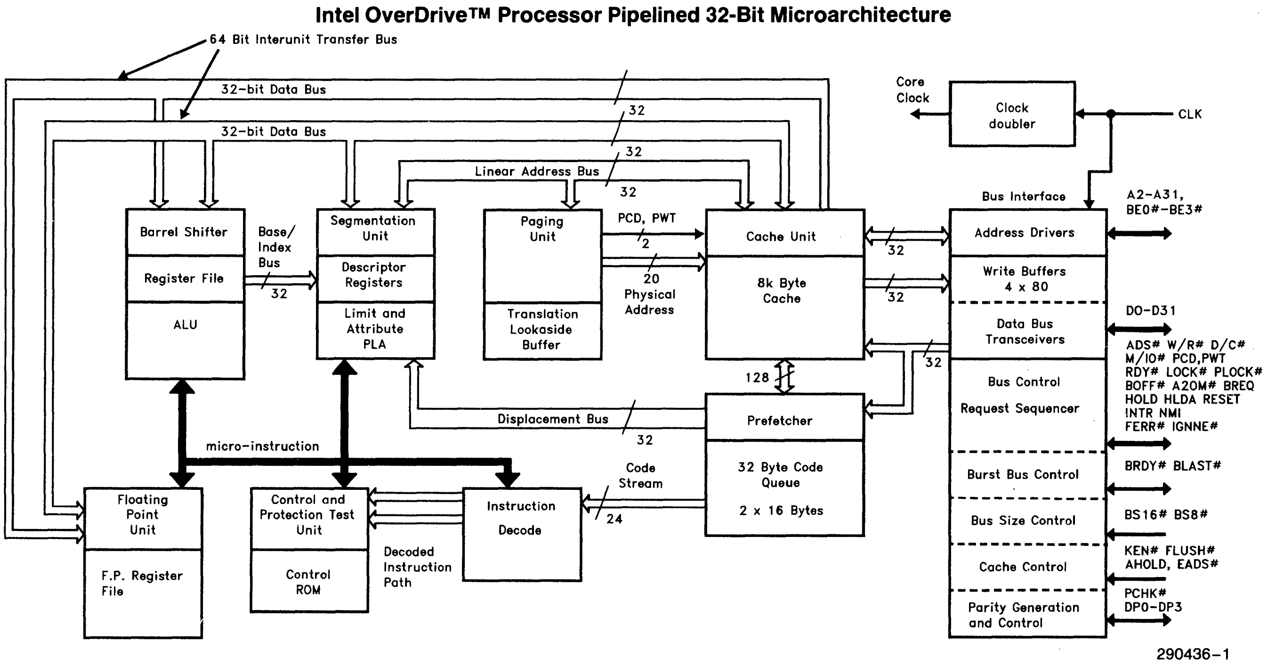


Figure 5-1. OverDrive™ Processor Architecture Block Diagram

6.0 DIFFERENCES IN FUNCTIONALITY BETWEEN THE OverDrive™ PROCESSOR FAMILY AND THE Intel486™ SX AND Intel486™ DX PROCESSORS

The Intel OverDrive processors are an enhanced family of Intel486 microprocessors. There are, however, four functional differences. First, the Intel OverDrive processors have an internal clock doubling (IntelSX2, IntelDX2) or clock tripling (IntelDX4) circuit which decreases the time required to execute instructions. Second, the Intel OverDrive processor family does not support the JTAG boundary scan test feature. Third, the Intel OverDrive processors have different processor revision identifications than the Intel486 SX or Intel486 DX processors. Finally, the IntelDX4 OverDrive processor contains a 16 KByte cache, as opposed to the 8 KByte cache on the IntelSX2 and IntelDX2 OverDrive processors. These four differences are described in the following sections, according to how they affect the processor functionality.

6.1 Hardware Interface

The bus of the Intel OverDrive processors has been designed to be identical to the Intel486 Microprocessor bus. Although the external clock is internally doubled or tripled, and data and instructions are manipulated in the processor core at twice or three times the external frequency, the external bus is functionally identical to that of the Intel486 processor.

The four boundary scan test signals (TCK, Test clock; TMS, Test Mode select; TDI, Test Data Input; TDO, Test Data Output), defined for some Intel486 processors, are not specified for the Intel486 DX2 OverDrive processor.

The UP# (Upgrade Present) signal, which is defined as an input for some Intel486 processors, is an output signal on the Intel OverDrive processor. The UP# pin on the Intel OverDrive processor provides a logical low output signal which can be used to enable logic to recognize and configure the system for the Intel OverDrive processor. This signal is identical to the MP# output defined for the Intel487 SX Math CoProcessor. Refer to Section 7 for examples of use of the UP# signal.

The DX register always contains the component identifier at the conclusion of RESET. The Intel OverDrive processor has a different revision identifier in the DL register than the Intel486 SX or Intel486 DX microprocessors (refer to Section 8.1). When the OverDrive processor is installed in a system the component identifier is supplied by the OverDrive processor, rather than the original processor. The stepping identification portion of the component identification will change with different revisions of the OverDrive processor. The designer should only assume that the component identification for the OverDrive processor will be 045x for the IntelSX2 OverDrive processor, 043x for the IntelDX2 OverDrive processor and 148x for the IntelDX4 OverDrive processor, where "x" is the stepping identifier.

6.2 Testability

As detailed in Section 6.1, the Intel OverDrive processor does not support the JTAG boundary scan testability feature.

6.3 Instruction Set Summary

The Intel OverDrive processor supports all Intel486 extensions to the 8086/80186/80286 instruction set. In general, instructions will run faster on the Intel OverDrive processors than on the Intel486 microprocessor. Specifically, an instruction that only uses memory from the on-chip cache executes at the full core clock rate while all bus accesses execute at the bus clock rate. To calculate the elapsed time of an instruction, the number of clock counts for that instruction must be multiplied by the clock period for the system. The instruction set clock count summary tables from the Intel486 SX and Intel486 DX Microprocessor Data Sheets can be used for the OverDrive processor with the following modifications:

— Clock counts for a cache hit: This value represents the number of internal processor core clocks for an instruction that requires no external bus accesses or the base core clocks for an instruction requiring external bus accesses.

— Penalty clock counts for a cache miss: This value represents the worst-case approximation of the additional number of external clock counts that are required for an instruction which must access the external bus for data (a cache miss). This number must be multiplied by 2 (for the IntelSX2 and IntelDX2 OverDrive processors) or 3 (for the IntelDX4 OverDrive processor) to convert it to an equal number of internal processor core clock counts and added to the base core clocks to compute the number of core clocks for this instruction.

The actual number of core clocks for an instruction with a cache miss may be less than the base clock counts (from the cache hit column) plus the penalty clock counts (2 times the cache miss column number for the IntelSX2 and IntelDX2, 3 times the cache miss column number for the IntelDX4). The clock counts in the cache miss penalty column can be a cumulative value of external bus clocks (for data reads) and internal clocks for manipulating the data which has been loaded from the external bus. The number of clocks which are related to external bus accesses are correctly represented in terms of internal core clocks by multiplying by two. However, the clock counts related to internal data manipulation should not be multiplied by two. Therefore the total number of processor core clock counts for an instruction with a cache miss represents a worst-case approximation.

To calculate the execution time for an OverDrive processor instruction, multiply the total processor core clock counts by the core clock period. For example, in a 25 MHz system upgraded with a 50 MHz IntelDX2 OverDrive processor, the core clock period is 20 ns (1/50 MHz).

Additionally, the assumptions specified below should be understood in order to estimate instruction execution time.

A cache miss will force the OverDrive processor to run an external bus cycle. The Intel486 microprocessor 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 is 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 fastest bus the OverDrive processor can support is 2−1−2 assuming 0 waits states. The clock counts in the cache miss penalty column assume a 2−1−2 bus. For slower busses 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 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 OverDrive 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.

7.0 INTEL OverDrive™ PROCESSOR CIRCUIT DESIGN

7.1 Upgrade Circuit for Intel486 Processor-Based Systems with UP#

Figure 7-1 shows the Intel OverDrive processor socket circuit for Intel486 processor-based systems using UP#. The Upgrade Present input, UP# pin, allows the Intel486 processor to directly recognize when the Intel OverDrive processor socket is populated. When the UP# pin is driven active to the Intel486 processor, the Intel486 processor tri-states all of its output pins and enters power-down mode.

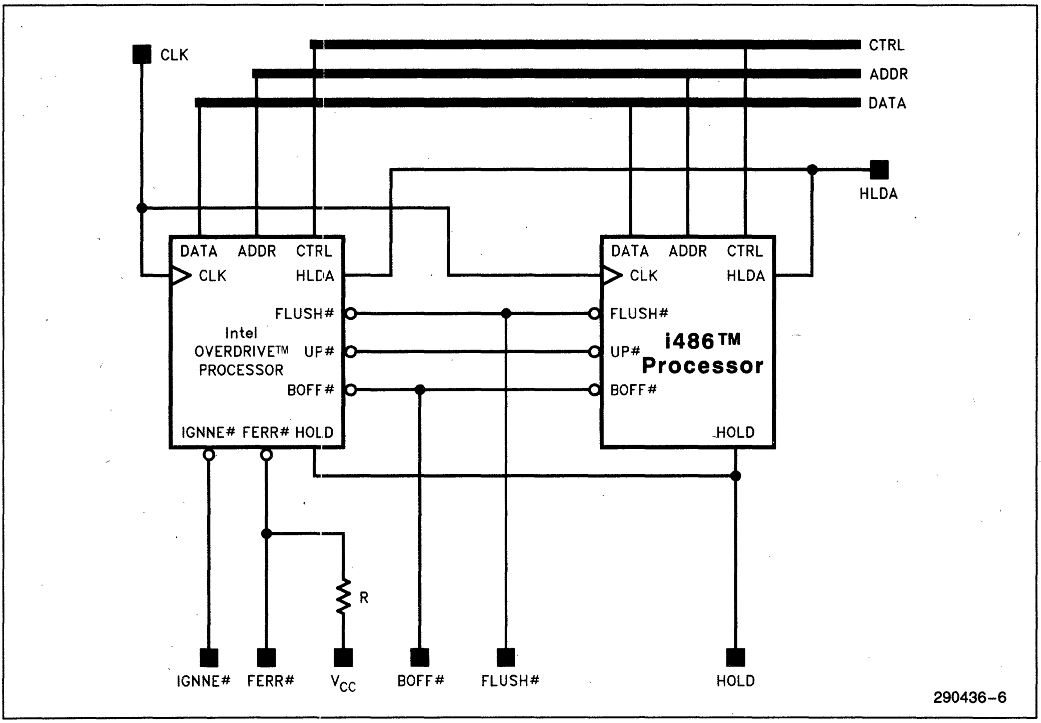


Figure 7-1. Intel OverDrive™ Socket Circuit Diagram for Systems Based on Intel486™ Processors That Have the UP# Input Pin

8.0 BIOS AND SOFTWARE

The following should be considered when designing a system for upgrade with an Intel OverDrive processor.

8.1 Intel OverDrive™ Processor Detection

The component identifier and the stepping/revision identifier for the Intel OverDrive processors is readable in the DH and DL registers, respectively, immediately after RESET. The value loaded into each register is defined in Table 8-1. The "x" value defines the device stepping.

Table 8-1. CPU ID Values

Processor	DH Reg.	DL Reg.
Intel486DX	04h	0xh, 1xh
Intel486SX	04h	2xh
IntelSX2 OverDrive	04h	5xh
IntelDX2 OverDrive	04h	3xh
IntelDX4 OverDrive	14h	8xh

As it is difficult to differentiate between Intel486 DX processor and some of the Intel OverDrive processors in software, it is recommended that the BIOS save the contents of the DX register immediately after RESET. This will allow the information to be used later, if required, to identify an Intel OverDrive processor in the system.

Alternately, for those OverDrive processors supporting it, the CPUID instruction can be used to identify the processor. Refer to the Intel486 Microprocessor Data Book for additional information on the CPUID instruction and its use.

NOTE:

Initialization routines for IntelSX2 OverDrive processor and Intel486 SX processor-based systems should check for the presence of a floating point unit and set the CR0 register accordingly (refer to the Intel486 SX Microprocessor Data Book for specific details). In addition, the BIOS should check for the presence of the 16 KByte cache in the IntelDX4 OverDrive processor.

8.2 Timing Dependent Loops

The Intel OverDrive processors execute instructions at two times (for the IntelSX2 and IntelDX2 OverDrive processors) or three times (for the IntelDX4 OverDrive processor) the frequency of the input clock. Thus, software (or instruction based) timing loops will execute faster on the Intel OverDrive processor than on the Intel486 DX or Intel486 SX processor (at the same input clock frequency). Instructions such as NOP, LOOP, and JMP $+2, have been used by 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 for systems intended to be upgradable with the Intel OverDrive processors.

In order to avoid any incompatibilities, it is recommended that timing requirements 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 routine 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 to the processor by keeping RDY# or BRDY# deasserted for the appropriate amount of time.

9.0 ELECTRICAL DATA

The following sections describe recommended electrical connections for the Intel OverDrive processor, and its electrical specifications.

9.1 Power and Grounding

9.1.1 POWER CONNECTIONS

Power and ground connections must be made to all external V_{CC} and GND pins of the Intel OverDrive 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.

9.1.2 POWER DECOUPLING RECOMMENDATIONS

Liberal decoupling capacitance should be placed near the Intel OverDrive processor. The Intel OverDrive processor driving its 32-bit parallel address and data busses 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 Intel OverDrive processor and decoupling capacitors as much as possible. Capacitors specifically for PGA packages are also commercially available.

9.1.3 OTHER CONNECTION RECOMMENDATIONS

N.C. pins should always remain unconnected.

For reliable operation, always connect unused inputs to an appropriate signal level. Active LOW inputs should be connected to V_{CC} through a pullup resistor. Pullups in the range of 20 KΩ are recommended. Active HIGH inputs should be connected to GND.

9.2 Maximum Ratings

Table 9-1 lists the absolute maximum ratings for each of the OverDrive processors. This table is a stress rating only, and functional operation at the maximums is not guaranteed. Functional operating conditions are given in Section 9.3, D.C. Specifications, and Section 9.4, A.C. Specifications.

Table 9-1. Absolute Maximum Ratings

	IntelSX2™ OverDrive™	IntelDX2™ OverDrive™	IntelDX4™ OverDrive™
Case Temperature under Bias	−65°C to +110°C	−65°C to +110°C	−30°C to +110°C
Storage Temperature	−65°C to +150°C	−65°C to +150°C	−30°C to +125°C
Voltage on any Pin with Respect to Ground	−0.5V to V_{CC} + 0.5V	−0.5V to V_{CC} + 0.5V	−0.5V to V_{CC} + 0.5V
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V	−0.5V to +6.5V	−0.5V to +6.5V

9.3 D.C. Specifications

The D.C. specifications for each of the OverDrive processors are contained in the tables in Sections 9.3.1, 9.3.2 and 9.3.3. For additional information, refer to the appropriate Intel microprocessor handbook.

9.3.1 IntelSX2™ OverDrive™ PROCESSOR D.C. SPECIFICATIONS

Table 9-2 details the D.C. Specifications of the IntelSX2 OverDrive processor.

Table 9-2. D.C. Specifications for the IntelSX2™ OverDrive™ Processor

Functional operating range: $V_{CC} = 5V \pm 5\%$; $T_{SINK} = 0°C$ to $+85°C$

Symbol	Parameter	Min	Typ	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_{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		15	µA	(Note 3)
I_{IH}	Input Leakage Current (all pins except SRESET) Input Leakage Current for SRESET			200 300	µA µA	(Note 4)
I_{CC}	Power Supply Current CLK = 25 MHz		700	950	mA	
I_{IL}	Input Leakage Current			−400	µA	(Note 5)
I_{LO}	Output Leakage Current	−15		15	µA	
C_{IN}	Input Capacitance PGA			20	pF	(Note 7)
C_{OUT}	Output or I/O Capacitance PGA			20	pF	(Note 7)
C_{CLK}	CLK Capacitance PGA			20	pF	(Note 7)

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 pullups or pulldowns and $0V \leq V_{IN} \leq V_{CC}$.
4. This parameter is for inputs with pulldowns and $V_{IH} = 2.4V$.
5. This parameter is for inputs with pullups 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.

9.3.2 IntelDX2™ OverDrive™ PROCESSOR D.C. SPECIFICATIONS

Table 9-3 details the D.C. Specifications of the IntelDX2 OverDrive processor.

Table 9-3. D.C. Specifications for the IntelDX2™ OverDrive™ Processor

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 2)
V_{OH}	Output High Voltage	2.4		V	(Note 3)
I_{CC}	Power Supply Current CLK = 33 MHz CLK = 25 MHz		 1200 950	mA	(Note 4)
I_{LI}	Input Leakage Current		±15	μA	(Note 5)
I_{IH}	Input Leakage Current		200	μA	(Note 6)
I_{IL}	Input Leakage Current		−400	μA	(Note 7)
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		13	pF	F_C = 1 MHz[8]
C_O	I/O or Output Capacitance		17	pF	F_C = 1 MHz[8]
C_{CLK}	CLK Capacitance		15	pF	F_C = 1 MHz[8]

NOTES:

1. The function operating temperature range is:
   OverDrive processor—25 MHz, T_{sink} = 0°C to +95°C
   OverDrive processor—33 MHz, T_{sink} = 0°C to +95°C
2. This parameter is measured at:
   Address, Data, BEn 4.0 mA
   Definition, Control 5.0 mA
3. This parameter is measured at:
   Address, Data, BEn −1.0 mA
   Definition, Control −0.9 mA
4. Typical supply current:
   775 mA @ CLK = 25 MHz
   975 mA @ CLK = 33 MHz
5. This parameter is for inputs without internal pullups or pulldowns and $0 \leq V_{IN} \leq V_{CC}$.
6. This parameter is for inputs with internal pulldowns and V_{IH} = 2.4V.
7. This parameter is for inputs with internal pullups and V_{IL} = 0.45V.
8. Not 100% tested.

9.3.3 IntelDX4™ OverDrive™ PROCESSOR D.C. SPECIFICATIONS

Table 9-4 details the D.C. Specifications of the IntelDX4 OverDrive processor.

Table 9-4. D.C. Specifications for the IntelDX4™ OverDrive™ Processor

Functional operating range: V_{CC} = 5V + 5%, T_{SINK} = 0°C to +95°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	I_{OH} = −2 mA
I_{CC}	Power Supply Current CLK = 25/75 MHz CLK = 33/100 MHz		 1200 1550	mA	(Note 2)
I_{CC} Stop Grant	Power Supply Current in Stop Grant State CLK = 25/75 MHz CLK = 33/100 MHz		 85 110	 mA mA	(Note 3)
I_{CC} Stop Clock	Power Supply Current in Stop Clock State		20	mA	(Note 4)
I_{LI}	Input Leakage Current		±15	μA	(Note 5)
I_{IH}	Input Leakage Current		200	μA	(Note 6)
I_{IL}	Input Leakage Current		−400	μA	(Note 7)
I_{LO}	Output Leakage Current		±15	μA	
C_{IN}	Input Capacitance		13	pF	F_C = MHz(8)
C_O	I/O or Output Capacitance		17	pF	F_C = MHz(8)
C_{CLK}	CLK Capacitance		15	pF	F_C = MHz(8)

NOTES:

1. This parameter is measured at:
   4.0 mA: Address, Data, BEn
   5.0 mA: Definition, Control
2. The maximum and typical values shown here are design estimates. Typical supply current:
   I_{CC} = 835 mA @ CLK = 25 MHz
   I_{CC} = 1085 mA @ CLK = 33 MHz
3. The I_{CC} Stop Grant specification refers to the I_{CC} value once the IntelDX4 OverDrive processor enters the Stop Grant or Halt Auto Powerdown State.
4. The I_{CC} Stop Clock specification refers to the I_{CC} value once the IntelDX4 OverDrive processor enters the Stop Clock State. V_{IH} and V_{IL} levels must be V_{CC} and 0V, respectively, in order to meet the I_{CC} Stop Clock specification.
5. This parameter is for inputs without pullups or pulldowns and $0 \leq V_{IN} \leq V_{CC}$.
6. This parameter is for inputs with pulldowns and V_{IH} = 2.4V.
7. This parameter is for inputs with pullups and V_{IL} = 0.45V.
8. Not 100% tested.

9.4 A.C. Specifications

The A.C. specifications for each of the OverDrive processors are contained in the tables in Sections 9.4.1, 9.4.2 and 9.4.3. These specifications consist of output delays, input setup requirements and input hold requirements. All A.C. specifications are relative to the rising edge of the CLK signal.

A.C. specification measurements are defined by Figures 9-1 through 9-6. All timings are referenced to 1.5V, unless otherwise specified. Inputs must be driven to the voltage levels indicated by Figure 9-3 when A.C. specifications are measured. Intel OverDrive processor output delays are specified with minimum and maximum limits, measured as shown. The minimum Intel OverDrive processor delay times are hold times provided to external circuitry. Intel OverDrive processor input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous signal must be stable for correct Intel OverDrive processor operation.

Table 9-5 defines the A.C. timing specifications for a 33 MHz system. Table 9.7 defines the A.C. timing specifications for a 25 MHz system. Table 9-8 defines the A.C. timing specifications for a 20 MHz system. Table 9-8 defines the A.C. timing specifications for a 16 MHz system.

Each Intel OverDrive processor meets the A.C. specifications for the processor it is upgrading. For example, a 100 MHz IntelDX4 OverDrive processor meets the system A.C. timing specifications for the 33 MHz processor it is upgrading.

Refer to Sections 9.4.1 through 9.4.3 for any timing differences from those specified in the following tables.

For additional information, refer to the appropriate Intel microprocessor handbook.

9.4.1 IntelSX2™ OverDrive™ PROCESSOR A.C. SPECIFICATIONS

The IntelSX2 OverDrive processor can be placed into an existing 16 MHz, 20 MHz or 25 MHz Intel486 system, doubling the internal processor speed to 32 MHz, 40 MHz or 50 MHz, respectively.

Tables 9-6 through 9-8 contain the A.C. timing specifications for the processors in those systems.

9.4.2 IntelDX2™ OverDrive™ PROCESSOR A.C. SPECIFICATIONS

The IntelDX2 OverDrive processor can be placed into an existing 16 MHz, 20 MHz, 25 MHz or 33 MHz Intel486 system, doubling the internal processor speed to 32 MHz, 40 MHz, 50 MHz or 66 MHz, respectively.

Tables 9-5 through 9-8 contain the A.C. timing specifications for the processors in those systems.

9.4.3 IntelDX4™ OverDrive™ PROCESSOR A.C. SPECIFICATIONS

The IntelDX4 OverDrive processor can be placed into an existing 16 MHz, 20 MHz, 25 MHz or 33 MHz Intel486 system, tripling the internal processor speed to 48 MHz, 60 MHz, 75 MHz or 100 MHz, respectively.

Tables 9-5 through 9-8 contain the A.C. timing specifications for the processors in those systems.

Table 9-5. 33 MHz Intel Processor Characteristics(1)

V_{CC} = 5V ±5%; T_{sink} = See Note 6; C_l = 50 pF unless otherwise specified(3)

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	33	MHz		1X Clock Driven to OverDrive processor
t_1	CLK Period	30	125	ns	5.1	
t_{1a}	CLK Period Stability		0.1%	Δ		Adjacent Clocks
t_2	CLK High Time	11		ns	5.1	at 2V
t_3	CLK Low Time	11		ns	5.1	at 0.8V
t_4	CLK Fall Time		3	ns	5.1	2V to 0.8V
t_5	CLK Rise Time		3	ns	5.1	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, SMIACT#, FERR#, BREQ, HLDA Valid Delay	3	14	ns	5.5	(Note 4)
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK# Float Delay		20	ns	5.6	(Note 2)
t_8	PCHK# Valid Delay	3	22	ns	5.4	(Note 4)
t_{8a}	BLAST#, PLOCK# Valid Delay	3	20	ns	5.5	(Note 4)
t_9	BLAST#, PLOCK# Float Delay		20	ns	5.6	(Note 2)
t_{10}	D0–D31, DP0–3 Write Data Valid Delay	3	18	ns	5.5	(Note 4)
t_{11}	D0–D31, DP0–3 Write Data Float Delay		20	ns	5.6	(Note 2)
t_{12}	EADS# Setup Time	5		ns	5.2	
t_{13}	EADS# Hold Time	3		ns	5.2	
t_{14}	KEN#, BS16#, BS8# Setup Time	5		ns	5.2	
t_{15}	KEN#, BS16#, BS8# Hold Time	3		ns	5.2	
t_{16}	RDY#, BRDY# Setup Time	5		ns	5.3	
t_{17}	RDY#, BRDY# Hold Time	3		ns	5.3	
t_{18}	HOLD, AHOLD Setup Time	6		ns	5.2	
t_{18a}	BOFF# Setup Time	7		ns	5.2	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		ns	5.2	
t_{20}	RESET, FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, IGNNE# Setup Time	5		ns	5.2	
t_{21}	RESET, FLUSH#, A20M#, NMI, INTR, SMI#, STPCLK#, SRESET, IGNNE# Hold Time	3		ns	5.2	
t_{22}	D0–D31, DP0–3, A4–A31 Read Setup Time	5		ns	5.2, 5.3	
t_{23}	D0–D31, DP0–3, A4–A31 Read Hold Time	3		ns	5.2, 5.3	

NOTES:

1. To be used for 66 MHz IntelDX2 and 100 MHz IntelDX4 OverDrive processors.
2. Not 100% tested. Guaranteed by design characterization.
3. All timing specifications assume C_L = 50 pF.
4. The minimum Intel OverDrive processor output valid delays are hold times provided to external circuitry.
5. A reset pulse width of 15 CLK cycles is required for warm resets. Power-up resets require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. T_{SINK} temperatures are:
   IntelSX2 OverDrive processor: 0°C to +85°C
   IntelDX2 OverDrive processor: 0°C to +95°C
   IntelDX4 OverDrive processor: 0°C to +95°C

Table 9-6. 25 MHz Intel Processor Characteristics[1]

V_{CC} = 5V ±5%; T_{sink} = See Note 6; C_l = 50 pF unless otherwise specified[3]

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	25	MHz		1X Clock Driven to OverDrive Processor
t_1	CLK Period	40	125	ns	5.1	
t_{1a}	CLK Period Stability		0.1%	Δ		Adjacent Clocks
t_2	CLK High Time	14		ns	5.1	at 2V
t_3	CLK Low Time	14		ns	5.1	at 0.8V
t_4	CLK Fall Time		4	ns	5.1	2V to 0.8V
t_5	CLK Rise Time		4	ns	5.1	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, FERR#, BREQ, HLDA, SMIACT#, Valid Delay	3	19	ns	5.5	(Note 4)
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK# Float Delay		28	ns	5.6	(Note 2)
t_8	PCHK# Valid Delay	3	24	ns	5.4	(Note 4)
t_{8a}	BLAST#, PLOCK# Valid Delay	3	24	ns	5.5	(Note 4)
t_9	BLAST#, PLOCK# Float Delay		28	ns	5.6	(Note 2)
t_{10}	D0–D31, DP0–3 Write Data Valid Delay	3	20	ns	5.5	(Note 4)
t_{11}	D0–D31, DP0–3 Write Data Float Delay		28	ns	5.6	(Note 2)
t_{12}	EADS# Setup Time	8		ns	5.2	
t_{13}	EADS# Hold Time	3		ns	5.2	
t_{14}	KEN#, BS16#, BS8# Setup Time	8		ns	5.2	
t_{15}	KEN#, BS16#, BS8# Hold Time	3		ns	5.2	
t_{16}	RDY#, BRDY# Setup Time	8		ns	5.3	
t_{17}	RDY#, BRDY# Hold Time	3		ns	5.3	
t_{18}	HOLD, AHOLD, BOFF# Setup Time	8		ns	5.2	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		ns	5.2	
t_{20}	RESET, FLUSH#, A20M#, NMI, SMI#, STPCLK#, SRESET, INTR, IGNNE# Setup Time	8		ns	5.2	
t_{21}	RESET, FLUSH#, A20M#, NMI, SMI#, STPCLK#, SRESET, INTR, IGNNE# Hold Time	3		ns	5.2	
t_{22}	D0–D31, DP0–3, A4--A31 Read Setup Time	5		ns	5.2, 5.3	
t_{23}	D0–D31, DP0–3, A4--A31 Read Hold Time	3		ns	5.2, 5.3	

NOTES:

1. To be used for 50 MHz IntelSX2, 50 MHz or 60 MHz IntelDX2 and 75 MHz or 100 MHz IntelDX4 OverDrive processors.
2. Not 100% tested. Guaranteed by design characterization.
3. All timing specifications assume C_L = 50 pF.
4. The minimum Intel OverDrive processor output valid delays are hold times provided to external circuitry.
5. A reset pulse width of 15 CLK cycles is required for warm resets. Power-up resets require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. T_{SINK} temperatures are:
   IntelSX2 OverDrive processor: 0°C to +85°C
   IntelDX2 OverDrive processor: 0°C to +95°C
   IntelDX4 OverDrive processor: 0°C to +95°C

Table 9-7. 20 MHz Intel Processor Characteristics[1]

V_{CC} = 5V ±5%; T_{SINK} = See Note 6; C_l = 50 pF unless otherwise specified[3]

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	20	MHz		1X Clock Driven to OverDrive Processor
t_1	CLK Period	50	125	ns	5.1	
t_{1a}	CLK Period Stability		0.1%	Δ		Adjacent Clocks
t_2	CLK High Time	16		ns	5.1	at 2V
t_3	CLK Low Time	16		ns	5.1	at 0.8V
t_4	CLK Fall Time		6	ns	5.1	2V to 0.8V
t_5	CLK Rise Time		6	ns	5.1	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, FERR#, BREQ, HLDA, SMIACT# Valid Delay	3	23	ns	5.5	(Note 4)
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK# Float Delay		37	ns	5.6	(Note 2)
t_8	PCHK# Valid Delay	3	28	ns	5.4	(Note 4)
t_{8a}	BLAST#, PLOCK# Valid Delay	3	28	ns	5.5	(Note 4)
t_9	BLAST#, PLOCK# Float Delay		37	ns	5.6	(Note 2)
t_{10}	D0–D31, DP0–3 Write Data Valid Delay	3	26	ns	5.5	(Note 4)
t_{11}	D0–D31, DP0–3 Write Data Float Delay		37	ns	5.6	(Note 2)
t_{12}	EADS# Setup Time	10		ns	5.2	
t_{13}	EADS# Hold Time	3		ns	5.2	
t_{14}	KEN#, BS16#, BS8# Setup Time	10		ns	5.2	
t_{15}	KEN#, BS16#, BS8# Hold Time	3		ns	5.2	
t_{16}	RDY#, BRDY# Setup Time	10		ns	5.3	
t_{17}	RDY#, BRDY# Hold Time	3		ns	5.3	
t_{18}	HOLD, AHOLD, Setup Time	12		ns	5.2	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	3		ns	5.2	
t_{20}	RESET, FLUSH#, A20M#, NMI, SMI#, STPCLK#, SRESET, INTR, IGNNE# Setup Time	12		ns	5.2	(Note 5)
t_{21}	RESET, FLUSH#, A20M#, NMI, SMI#, STPCLK#, SRESET, INTR, IGNNE# Hold Time	3		ns	5.2	(Note 5)
t_{22}	D0–D31, DP0–3, A4–A31 Read Setup Time	6		ns	5.2, 5.3	
t_{23}	D0–D31, DP0–3, A4–A31 Read Hold Time	3		ns	5.2, 5.3	

NOTES:

1. To be used for 50 MHz IntelSX2, 50 MHz or 60 MHz IntelDX2 and 75 MHz or 100 MHz IntelDX4 OverDrive processors.
2. Not 100% tested. Guaranteed by design characterization.
3. All timing specifications assume C_L = 50 pF.
4. The minimum Intel OverDrive processor output valid delays are hold times provided to external circuitry.
5. A reset pulse width of 15 CLK cycles is required for warm resets. Power-up resets require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. T_{SINK} temperatures are:
   IntelSX2 OverDrive processor: 0°C to +85°C
   IntelDX2 OverDrive processor: 0°C to +95°C
   IntelDX4 OverDrive processor: 0°C to +95°C

Table 9-8. 16 MHz Intel Processor Characteristics[1]

V_{CC} = 5V ±5%; T_{SINK} = See Note 6; C_l = 50 pF unless otherwise specified

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	16	MHz		1X Clock Driven to OverDrive Processor
t_1	CLK Period	62.5	125	ns	5.1	
t_{1a}	CLK Period Stability		0.1%	Δ		Adjacent Clocks
t_2	CLK High Time	20		ns	5.1	at 2V
t_3	CLK Low Time	20		ns	5.1	at 0.8V
t_4	CLK Fall Time		8	ns	5.1	2V to 0.8V
t_5	CLK Rise Time		8	ns	5.1	0.8V to 2V
t_6	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK#, FERR#, BREQ, HLDA, SMIACT# Valid Delay	3	26	ns	5.5	(Note 4)
t_7	A2–A31, PWT, PCD, BE0–3#, M/IO#, D/C#, W/R#, ADS#, LOCK# Float Delay		42	ns	5.6	(Note 2)
t_8	PCHK# Valid Delay	3	35	ns	5.4	(Note 4)
t_{8a}	BLAST#, PLOCK# Valid Delay	3	35	ns	5.5	(Note 4)
t_9	BLAST#, PLOCK# Float Delay		42	ns	5.6	(Note 2)
t_{10}	D0–D31, DP0–3 Write Data Valid Delay	3	30	ns	5.5	(Note 4)
t_{11}	D0–D31, DP0–3 Write Data Float Delay		42	ns	5.6	(Note 2)
t_{12}	EADS# Setup Time	12		ns	5.2	
t_{13}	EADS# Hold Time	4		ns	5.2	
t_{14}	KEN#, BS16#, BS8# Setup Time	12		ns	5.2	
t_{15}	KEN#, BS16#, BS8# Hold Time	4		ns	5.2	
t_{16}	RDY#, BRDY# Setup Time	12		ns	5.3	
t_{17}	RDY#, BRDY# Hold Time	4		ns	5.3	
t_{18}	HOLD, AHOLD, BOFF# Setup Time	12		ns	5.2	
t_{19}	HOLD, AHOLD, BOFF# Hold Time	4		ns	5.2	
t_{20}	RESET, FLUSH#, A20M#, NMI, SMI#, STPCLK#, SRESET, INTR, IGNNE# Setup Time	14		ns	5.2	(Note 5)
t_{21}	RESET, FLUSH#, A20M#, NMI, SMI#, STPCLK#, SRESET, INTR, IGNNE# Hold Time	4		ns	5.2	(Note 5)
t_{22}	D0–D31, DP0–3, A4–A31 Read Setup Time	10		ns	5.2, 5.3	
t_{23}	D0–D31, DP0–3, A4–A31 Read Hold Time	4		ns	5.2, 5.3	

NOTES:

1. To be used for 50 MHz IntelSX2, 50 MHz or 60 MHz IntelDX2 and 75 MHz or 100 MHz IntelDX4 OverDrive processors.
2. Not 100% tested. Guaranteed by design characterization.
3. All timing specifications assume C_L = 50 pF.
4. The minimum Intel OverDrive processor output valid delays are hold times provided to external circuitry.
5. A reset pulse width of 15 CLK cycles is required for warm resets. Power-up resets require RESET to be asserted for at least 1 ms after V_{CC} and CLK are stable.
6. T_{SINK} temperatures are:
   IntelSX2 OverDrive processor: 0°C to +85°C
   IntelDX2 OverDrive processor: 0°C to +95°C
   IntelDX4 OverDrive processor: 0°C to +95°C

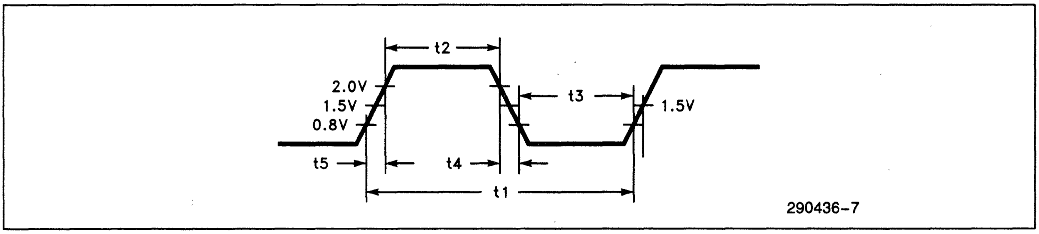


Figure 9-1. CLK Waveforms

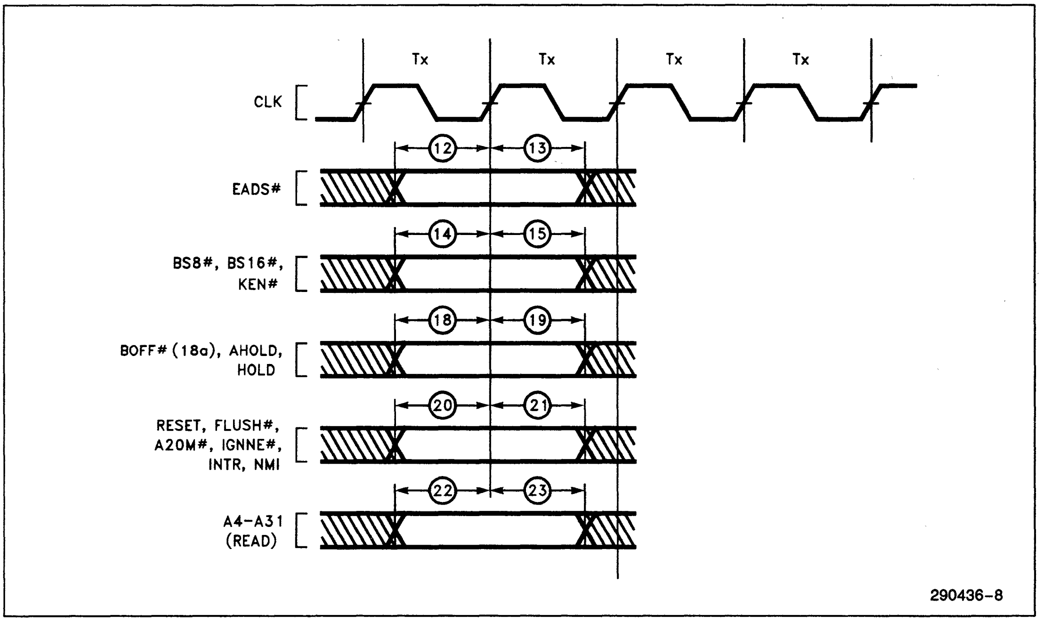


Figure 9-2. Input Setup and Hold Timing

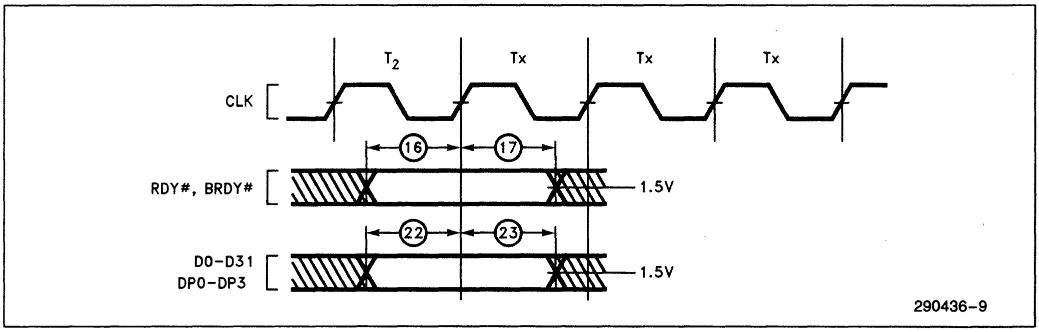


Figure 9-3. Input Setup and Hold Timing

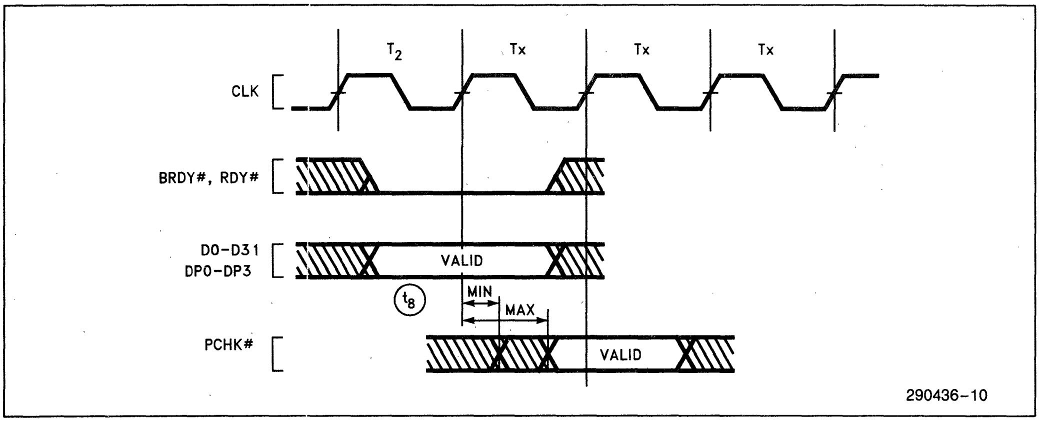


Figure 9-4. PCHK# Valid Delay Timing

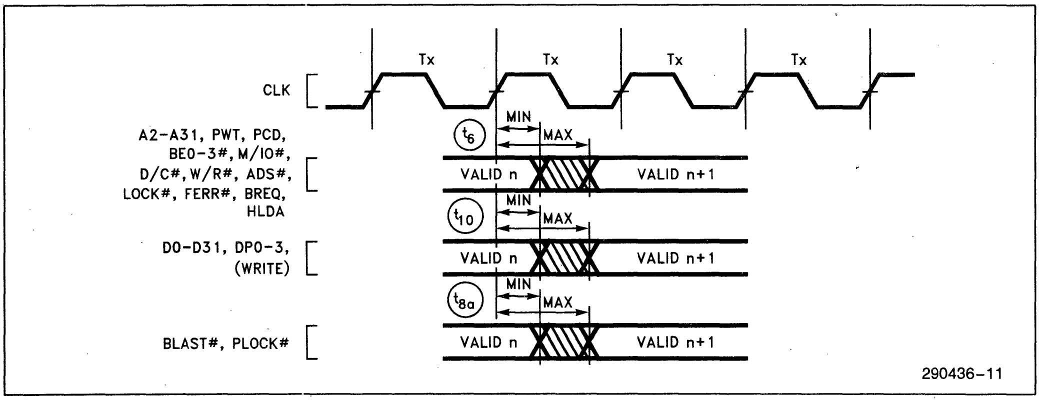


Figure 9-5. Output Valid Delay Timing

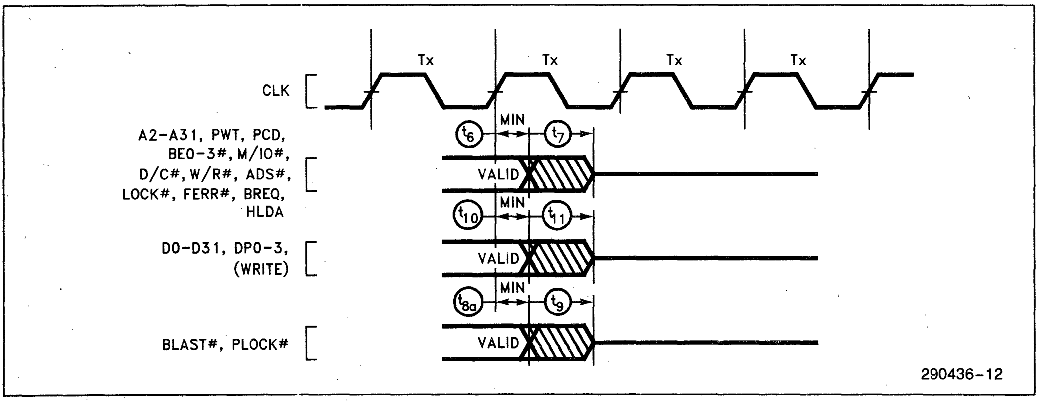


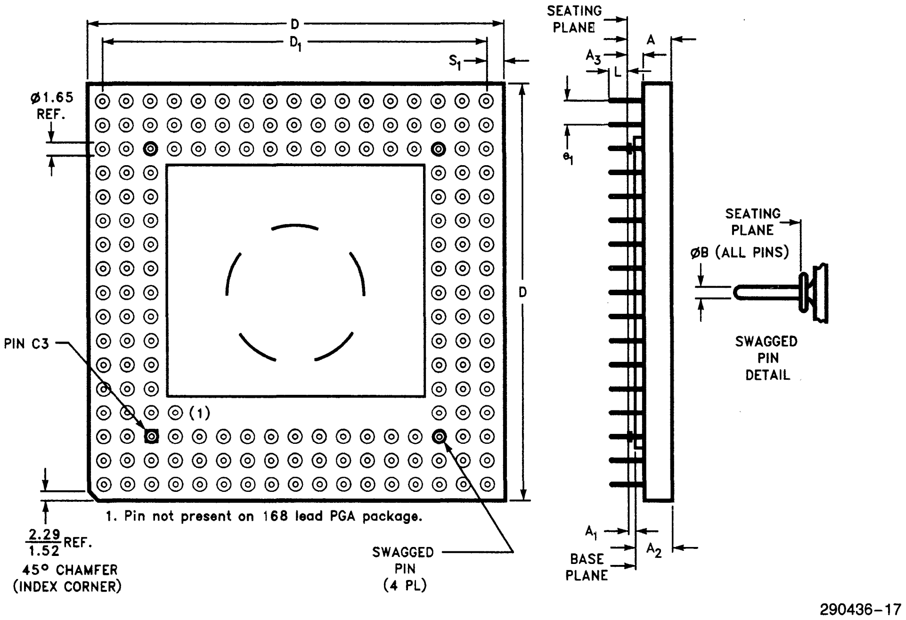
Figure 9-6. Maximum Float Delay Timing

10.0 MECHANICAL DATA

The following sections describe the physical dimensions of the OverDrive processor packages and heat sinks.

10.1 Package Dimensions

Figure 10-1 describes the physical dimensions of the PGA packages (168-lead PGA and 169-lead PGA) used with the IntelSX2, IntelDX2 and IntelDX4 OverDrive processors.



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, 169			168, 169		
S_1	1.52	2.54		0.060	0.100	
ISSUE	IWS REV X 7/15/88					

Figure 10-1. OverDrive™ Processor Package Dimensions

10.2 Heat Sink Dimensions

There are two different heat sinks used on the Intel OverDrive processors. The IntelSX2 and IntelDX2 OverDrive processors both use the 0.25″ heat sink. The IntelDX4 OverDrive processor uses the 0.6″ heat sink. Both heat sinks are described in the following sections.

10.2.1 0.25″ HEAT SINK

Figure 10-2 describes the physical dimensions of the 0.25″ heat sink used with the IntelSX2 and IntelDX2 OverDrive processors. Table 10-1 lists the physical dimensions.

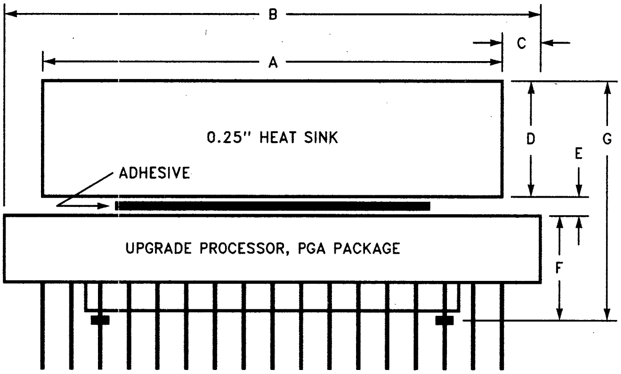


Figure 10-2. Dimensions, IntelSX2™/IntelDX2™ OverDrive™ Processor with 0.25″ Heat Sink

Table 10-1. 0.25″ Heat Sink Dimensions

Dimension (inches)	Minimum	Maximum
A. Heat Sink Width	1.520	1.550
B. PGA Package Width	1.735	1.765
C. Heat Sink Edge Gap	0.065	0.155
D. Heat Sink Height	0.212	0.260
E. Adhesive Thickness	0.008	0.012
F. Package Height from Stand-Offs	0.140	0.180
G. Total Height from Package Stand-Offs to Top of Heat Sink	0.360	0.452

10.2.2 0.6″ HEAT SINK

Figure 10-3 describes the physical dimensions of the 0.6″ heat sink used with the IntelDX4 OverDrive processors. The maximum and minimum dimensions for the PGA package with heat sink are shown in Table 10-2. As the table shows, the maximum height of the IntelDX4 OverDrive processor from the pin stand-offs to the top of the heat sink, including the adhesive thickness, is 0.780 inches. A minimum clearance of 0.25″ should be allowed above the top of the heat sink.

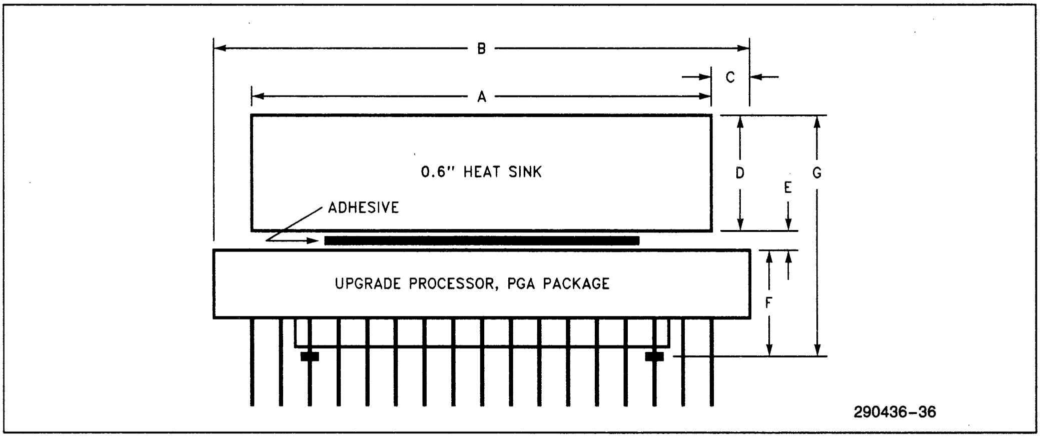


Figure 10-3. Dimensions, IntelDX4™ OverDrive™ Processor with 0.6″ Heat Sink

Table 10-2. 0.6″ Heat Sink Dimensions

Dimension (inches)	Minimum	Maximum
A. Heat Sink Width	1.520	1.550
B. PGA Package Width	1.735	1.765
C. Heat Sink Edge Gap	0.065	0.155
D. Heat Sink Height	0.580	0.600
E. Adhesive Thickness	0.006	0.012
F. Package Height from Stand-Offs	0.140	0.180
G. Total Height from Package Stand-Offs to Top of Heat Sink	0.720	0.780

11.0 THERMAL MANAGEMENT

The heat generated by the Intel OverDrive processor requires that heat dissipation be managed carefully. All OverDrive processors are supplied with a heat sink attached with adhesive to the package. System designs must, therefore, provide sufficient clearance (a minimum of 0.25" above the heat sink) for the processor and the attached heat sink.

Section 10 contains the physical dimensions for each of the heat sinks and packages used.

The standard product markings and logo for the Intel OverDrive processor with the attached heat sink will be included on a 1in^2 plate located on the top, center of the heat sink.

The heat sink is omni-directional, allowing air to flow from any direction in order to achieve adequate cooling. The thermal resistance values for the OverDrive processors with an attached heat sink are shown in Table 11-1 through Table 11-3.

Table 11-1. Thermal Resistance, IntelSX2™ OverDrive™ Processor with Attached Heat Sink

θ_{JS} = 1.5°C/W	Airflow (LFM)					
	0	200	400	600	800	1000
θ_{JA}(°C/W)	13.0	8.0	6.0	5.0	4.5	4.25

Table 11-2. Thermal Resistance, IntelDX2™ OverDrive™ Processor with Attached Heat Sink

θ_{JS} = 2.5°C/W	Airflow (LFM)				
	0	200	400	600	800
θ_{JA}(°C/W)	14.0	10.0	7.5	6.2	5.7

Table 11-3. Thermal Resistance, IntelDX4™ OverDrive™ Processor with Attached Heat Sink

θ_{JS} = 2.0°C/W	Airflow (LFM)			
	0	50	100	200
θ_{JA}(°C/W)	11.5	10.7	9.5	7.0

APPENDIX A
DESIGN CONSIDERATIONS

Intel has designed the family of OverDrive processors so that they can be installed by the end user. PC manufacturers can support this by implementing the design considerations listed in Table A-1.

Table A-1. Design Considerations

Design Consideration	Implementation
Visible OverDrive Processor Socket	The Intel OverDrive processor socket should be easily visible when the PC's cover is removed. Label the Intel OverDrive processor socket and the location of pin 1 by silk screening this information on the PC board.
Accessible OverDrive Processor Socket	Make the Intel OverDrive processor socket easily accessible to the end user (i.e., do not place the Intel OverDrive processor socket under a disk drive). If a Low Insertion Force (LIF) or screw machine socket is used, position the Intel OverDrive processor socket on the PC board such that there is ample clearance around the socket.
Foolproof Chip Orientation	Intel packages all Intel OverDrive processors in a 169-pin, PGA package. The 169th pin is called the "key" pin and insures that the Intel OverDrive processor fits into a 169-pin socket in only the correct orientation. Supplying a 169-pin socket as the Intel OverDrive processor socket eliminates the possibility of end users or resellers damaging the PC board or Intel OverDrive processor by powering up the system with the Intel OverDrive processor incorrectly oriented.
Zero Insertion Force Upgrade Socket	The high pin count of the Intel OverDrive processor makes the insertion force required for installation in a screw machine PGA socket excessive for end users or resellers. Even most Low Insertion Force (LIF) sockets often require more than 60 lbs. of insertion force. A Zero Insertion Force (ZIF) socket insures that the chip insertion force does not damage the PC board. If the ZIF socket has a handle, be sure to allow enough clearance for the socket handle. If a LIF or screw machine socket is used, additional PC board support is recommended.
"Plug and Play"	Jumper or switch changes should not be needed to electrically configure the system for the Intel OverDrive processor.
Thorough Documentation	Describe the Intel OverDrive processor's installation procedure in the PC's User's Manual.

APPENDIX B
ZIF AND LIF SOCKET VENDORS

The following list provides examples of sockets which can be used for Intel486 SX and Intel486 DX CPU-based systems.

NOTE:
This is not a comprehensive list. Intel has not tested all of the socket vendors' sockets listed below and cannot guarantee that these sockets will meet every PC manufacturer's specific requirements.

Zero Insertion Force (ZIF) and Low Insertion Force (LIF) Socket Vendors:

1. AMP Inc.
   219 American Avenue
   Greensboro, NC 27409-1803
   Tel: (800) 522-6752

2. Augat Inc.
   425 John Dietsch Blvd.
   Attleboro Falls, MA 02763
   Tel: (800) 999-7646

3. Aries Electronics
   P.O. Box 130
   Frenchtown, NJ 08825
   Tel: (908) 996-3891

8. Foxconn International Inc.
   930 West Maude Avenue
   Sunnyvale, CA 94086
   Tel: (800) 727-3699

4. JAE
   142 Technology Drive
   Irvine, CA 92718-2401
   Tel: (714) 753-2600

7. Loranger International Corp.
   817 Fourth Avenue
   Warren, PA 16365
   Tel: (814) 723-2250

5. Thomas and Betts
   200 Executive Center Drive
   P.O. Box 24901
   Greenville, SC 29616-2401
   Tel: (803) 676-2900

6. Yamaichi Electronics
   1420 Koll Circle, Suite B
   San Jose, CA 95112
   Tel: (800) 769-0797

4

Peripheral Components

82091AA
ADVANCED INTEGRATED PERIPHERAL (AIP)

- Single-Chip PC Compatible I/O Solution for Notebook and Desktop Platforms:
  - 82078 Floppy Disk Controller Core
  - Two 16550 Compatible UARTs
  - One Multi-Function Parallel Port
  - IDE Interface
  - Integrated Back Power Protection
  - Integrated Game Port Chip Select
  - 5V or 3.3V Supply Operation with 5V Tolerant Drive Interface
  - Full Power Management Support
  - Supports Type F DMA Transfers for Faster I/O Performance
  - No Wait-State Host I/O Interface
  - Programmable Interrupt Interfaces
  - Single Crystal/Oscillator Clock (24 MHz)
  - Software Detectable Device ID
  - Comprehensive Powerup Configuration
- The 82091AA is 100 Percent Compatible with EISA, ISA and AT
- Host Interface Features
  - 8-Bit Zero Wait-State ISA Bus Interface
  - DMA with Type F Transfers
  - Five Programmable ISA Interrupt Lines
  - Internal Address Decoder
- Parallel Port Features
  - All IEEE Standard 1284 Protocols Supported (Compatibility, Nibble, Byte, EPP, and ECP)
  - Peak Bi-Directional Transfer Rate of 2 MB/sec
  - Provides Interface for Low-Cost Engineless Laser Printer
  - 16-Byte FIFO for ECP
  - Interface Backpower Protection
- Floppy Disk Controller Features
  - 100 Percent Software Compatible with Industry Standard 82077SL and 82078
  - Integrated Analog Data Separator 250K, 300K, 500K, and 1 MBits/sec
  - Programmable Powerdown Command
  - Auto Powerdown and Wakeup Modes
  - Integrated Tape Drive Support
  - Perpendicular Recording Support for 4 MB Drives
  - Programmable Write Pre-Compensation Delays
  - 256 Track Direct Address, Unlimited Track Support
  - 16-Byte FIFO
  - Supports 2 or 4 Drives
- 16550 Compatible UART Features
  - Two Independent Serial Ports
  - Software Compatible with 8250 and 16450 UARTs
  - 16-Byte FIFO per Serial Port
  - Two UART Clock Sources, Supports MIDI Baud Rate
- IDE Interface Features
  - Generates Chip Selects for IDE Drives
  - Integrated Buffer Control Logic
  - Dual IDE Interface Support
- Power Management Features
  - Transparent to Operating Systems and Applications Programs
  - Independent Power Control for Each Integrated Device
- 100-Pin QFP Package

(See Packaging Spec. 240800)

The 82091AA Advanced Integrated Peripheral (AIP) is an integrated I/O solution containing a floppy disk controller, 2 serial ports, a multi-function parallel port, an IDE interface, and a game port on a single chip. The integration of these I/O devices results in a minimization of form factor, cost and power consumption. The floppy disk controller is the 82078 core. The serial ports are 16550 compatible. The parallel port supports all of the IEEE Standard 1284 protocols (ECP, EPP, Byte, Compatibility, and Nibble). The IDE interface supports 8- or 16-bit programmed I/O and 16-bit DMA. The Host Interface is an 8-bit ISA interface optimized for type "F" DMA and no wait-state I/O accesses. Improved throughput and performance, the 82091AA contains six 16-byte FIFOs-two for each serial port, one for the parallel port, and one for the floppy disk controller. The 82091AA also includes power management and 3.3V capability for power sensitive applications such as notebooks. The 82091AA supports both motherboard and add-in card configurations.

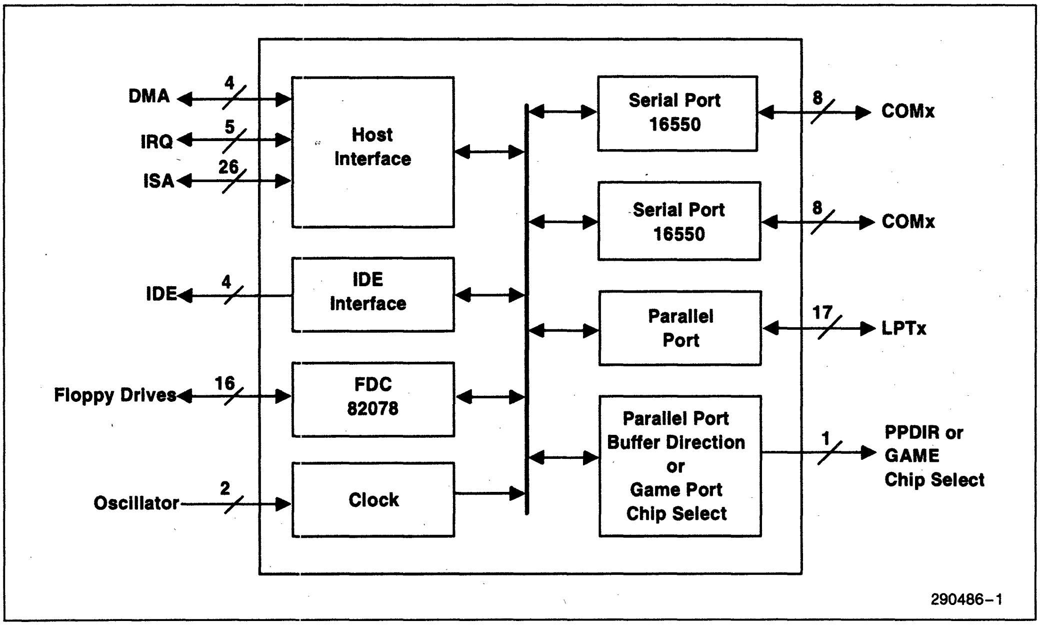


Figure 1. 82091AA Advanced Integrated Peripheral Block Diagram

82091AA
ADVANCED INTEGRATED PERIPHERAL (AIP)

CONTENTS

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

CONTENTS

PAGE

1.0. OVERVIEW

The major functions of the 82091AA are shown in Figure 1. A brief description of each of these functions is presented in this section.

Host Interface

The 82091AA host interface is an 8-bit direct-drive (24 mA) ISA Bus/X-Bus interface that permits the CPU to access its registers through read/write operations in I/O space. These registers may be accessed by programmed I/O and/or DMA bus cycles. With the exception of the IDE Interface, all functions on the 82091AA require only 8-bit data accesses. The 16-bit access required for the IDE Interface is supported through the appropriate chip selects and data buffer enables from the 82091AA.

Figure 2 shows an example system implementation with the 82091AA located on an ISA Bus add-in card. This add-in card could also be used in a PCI-based system as shown in Figure 3. For motherboard implementations, the 82091AA can be located on the X-Bus as shown in Figure 4.

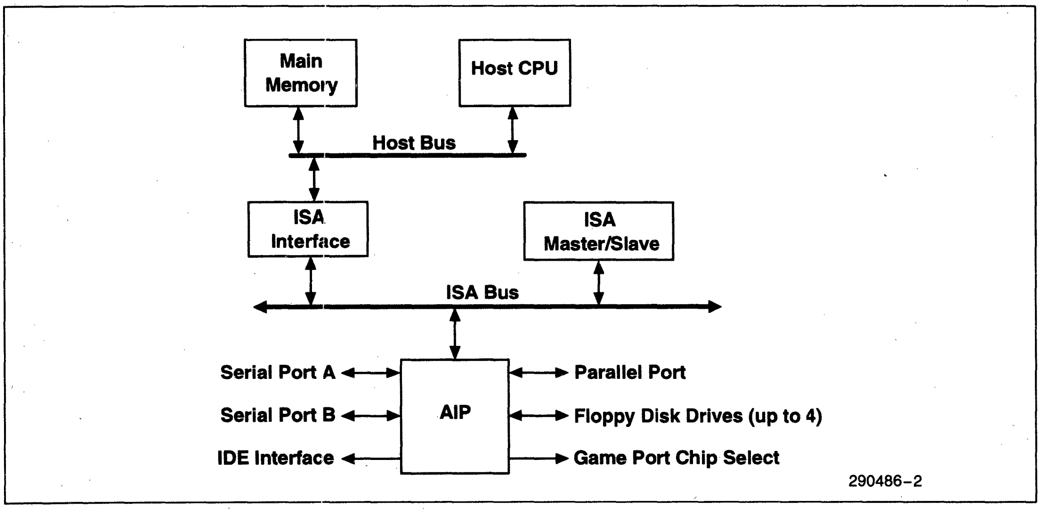


Figure 2. Block Diagram of the 82091AA on the ISA Bus

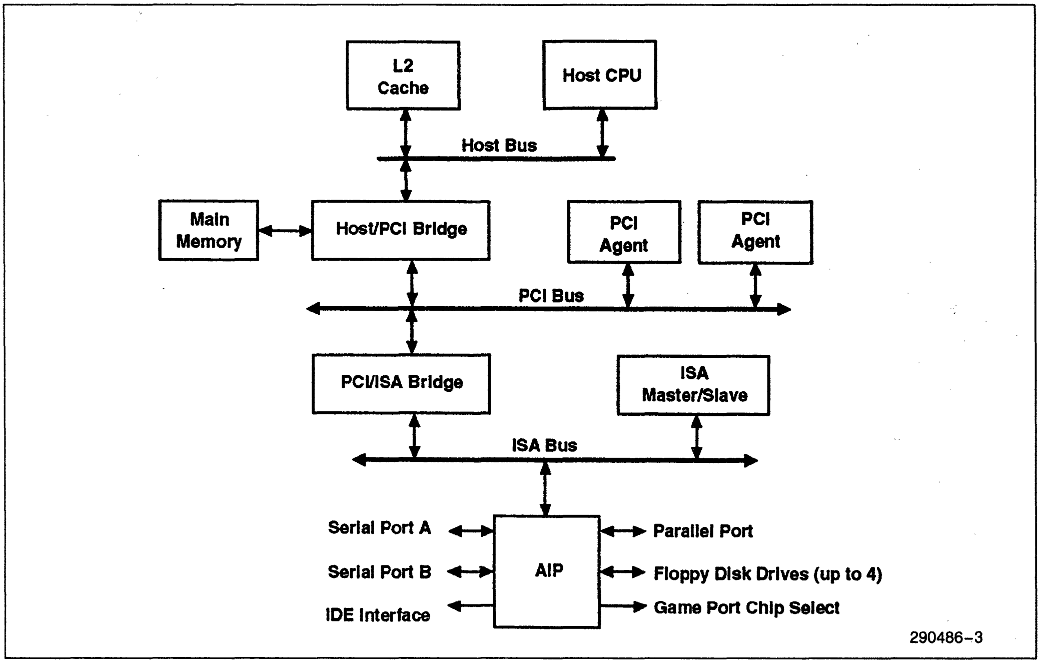


Figure 3. Block Diagram of the 82091AA in a PCI System

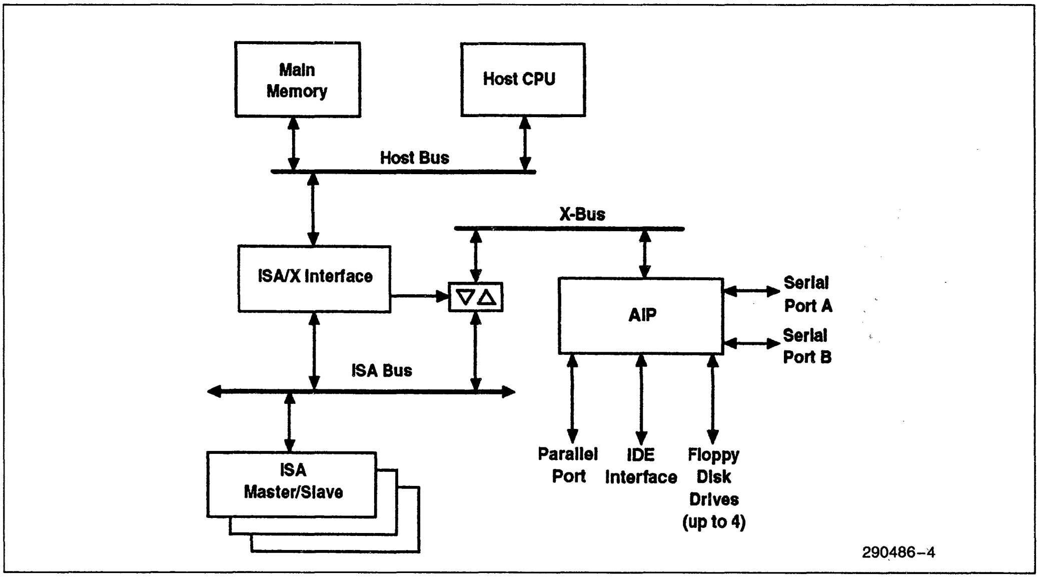


Figure 4. Block Diagram of the 82091AA on the X-Bus

Floppy Disk Controller

The 82091AA's enhanced floppy disk controller (FDC) incorporates several new features allowing for easy implementation in both the portable and desktop markets. It provides a low cost, small form factor solution targeted for 5.0V and 3.3V platforms. The FDC supports up to four drives.

The 82091AA's FDC implements these new features while remaining functionally compatible with 82078/82077SL/82077AA/8272A floppy disk controllers. Together, with a 24-MHz crystal, a resistor package and a device chip select, these devices allow for the most integrated solution available. The integrated analog PLL data separator has better performance than most board level discrete PLL implementations and can be operated at 1 Mbps/500 Kbps/300 Kbps/250 Kbps. A 16-byte FIFO substantially improves system performance and is ideal for multimaster systems (e.g., EISA).

Serial Ports

The 82091AA contains two independent serial ports that provide asynchronous communications that are equivalent to two 16550 UARTs. The serial ports have identical circuitry and provide the serial communication interface to a peripheral device or modem via Serial Port A and Serial Port B. Each serial port can be configured for one of eight address assignments. The standard PC/AT compatible logical address assignments for COM1, COM2, COM3, and COM4 are supported.

The serial ports perform serial-to-parallel conversion on data characters received from a peripheral device or modem, and parallel-to-serial conversion on data characters received from the host. The serial ports can operate in either FIFO mode or non-FIFO mode. In FIFO mode, a 16-byte transmit FIFO holds data from the host to be transmitted on the serial link and a 16-byte receive FIFO that buffers data from the serial link until read by the host.

The serial ports contain programmable baud rate generators that divide the internal reference clock by divisors of 1 to ($2^{16} - 1$), and produce a 16x clock for driving the transmitter and receiver logic. The internal reference clock can be programmed to support MIDI. The serial ports have complete modem-control capability and a prioritized interrupt system.

Parallel Port

The 82091AA provides a multi-function parallel port that transfers information between the host and peripheral device (e.g., printer). The parallel port interface contains nine control/status lines and an 8-bit data bus. The standard PC/AT compatible logical address assignments for LPT1, LPT2, and LPT3 are supported. The parallel port can be configured for one of four modes and supports the following IEEE Standard 1284 parallel interface protocol standards:

Parallel Port Mode	Parallel Interface Protocol
ISA-Compatible Mode	Compatibility, Nibble
PS/2-Compatible Mode	Byte
EPP Mode	EPP
ECP Mode	ECP

For ISA-Compatible and PS/2-Compatible modes, software controls the handshake signals on the parallel port interface to transfer data between the host and peripheral device. Status and Control registers permit software to monitor the state of the peripheral device and generate handshake sequences.

The EPP parallel port interface protocol increases throughput by specifying an automatic handshake sequence. In EPP mode, the 82091AA parallel port automatically generates this handshake sequence in hardware to transfer data between the host and peripheral device.

In addition to a hardware handshake on the parallel port interface, the ECP protocol specification also defines DMA and FIFO capability. To minimize processor overhead data transfer to/from a peripheral device, the 82091AA parallel port, in ECP mode, provides a 16-byte FIFO with DMA capability.

IDE Interface

The 82091AA supports the IDE (Integrated Drive Electronics) interface by providing chip selects and lower data byte control. Two chip selects are used to access registers on the IDE device. Separate lower and upper byte data control signals are provided. With these control signals, minimal external logic is needed to implement 16-bit IDE I/O and DMA interfaces.

Game Port

The 82091AA provides a game port chip select signal for use when the 82091AA is in an add-in card application. This function is assigned to I/O address location 201h. Note that when the 82091AA is located on the motherboard, this feature is not available.

Power Management

82091AA power management provides a mechanism for saving power when the device or a portion of the device is not being used. By programming the appropriate 82091AA registers, software can invoke power management to the entire 82091AA or selected modules within the 82091AA (e.g., floppy disk controller, serial port, or parallel port). There are two methods for applying power management—direct powerdown or auto powerdown. Direct powerdown turns off the clock to a particular module immediately placing that module into a powerdown state. This method removes the clock regardless of the activity or status of the module. When auto powerdown is invoked, the module enters a powerdown state (clock is turned off) after certain conditions are met and the module is in an idle state.

1.1. 3.3V/5V Operating Modes

The 82091AA can operate at a power supply of 3.3V, 5V or a mix of 3.3V and 5V. The mixed power supply mode provides 5V interfaces for the floppy disk controller and parallel port while all other 82091AA interfaces and internal logic (including the floppy disk controller and parallel port internal circuitry) operate at 3.3V. The mixed mode permits 5V floppy disk drives and parallel port peripherals to be used in a 3.3V system without external buffering.

NOTE:

3.3V operation is available only in the 82091AA.

2.0. SIGNAL DESCRIPTION

This section describes the 82091AA signals. The interface signals are shown in Figure 5 and described in the following tables. Signal descriptions are organized by functional group.

Note that the "#" symbol at the end of a signal name indicates the active, or asserted, state occurs when the signal is at a low voltage level. When "#" is not present after the signal name, the signal is asserted when at the high voltage level.

The terms assertion and negation are used extensively. This is done to avoid confusion when working with a mixture of "active-low" and "active-high" signals. The term **assert**, or **assertion**, indicates that a signal is active, independent of whether that level is represented by a high or low voltage. The term **negate**, or **negation**, indicates that a signal is inactive.

The following notations are used to describe pin types:

I	Input Pin
O	Output Pin
I/O	Bi-Directional Pin

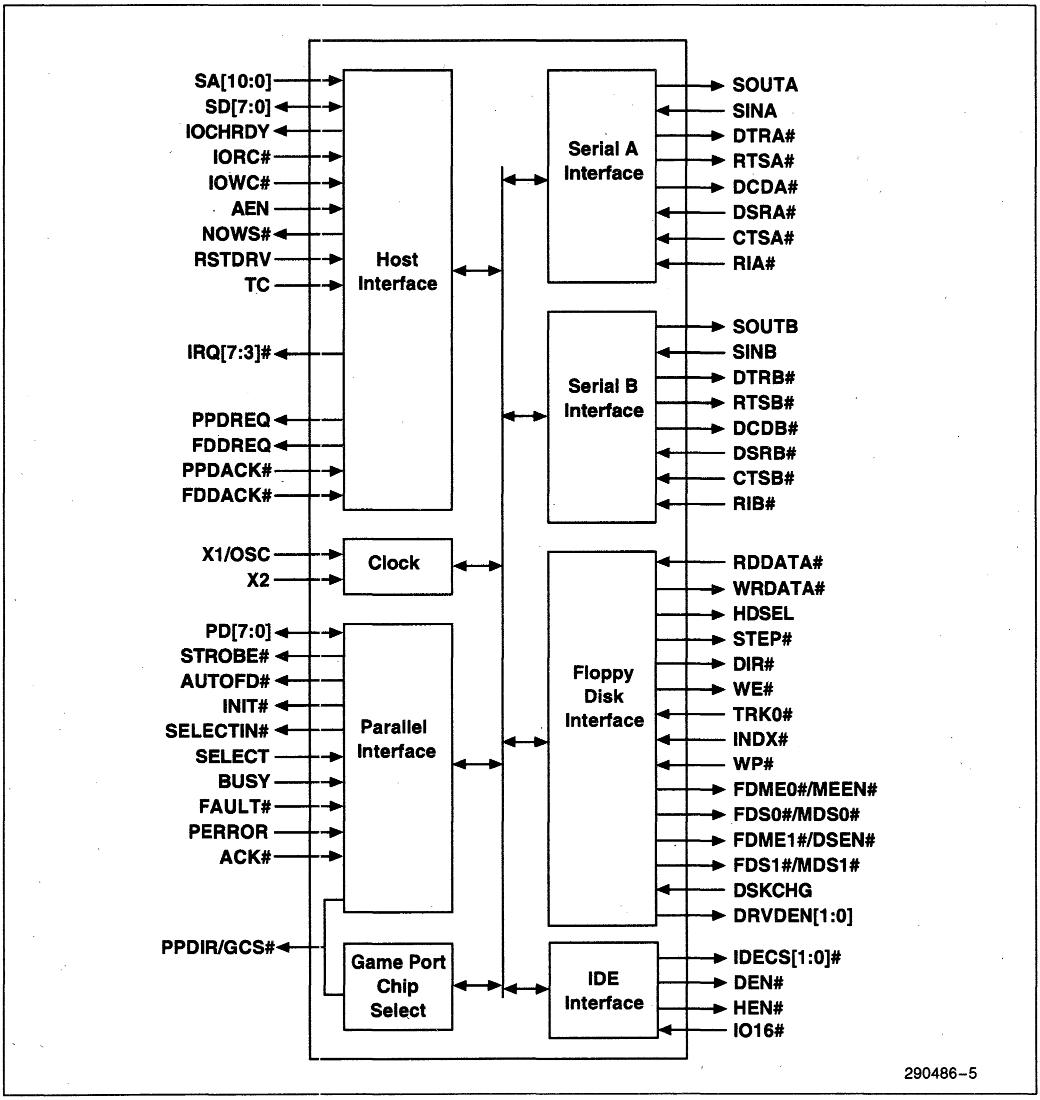


Figure 5. 82091AA Signals

2.1 Host Interface Signals

Signal Name	Type	Description
ISA SIGNALS		
SA[10:0]	I	**SYSTEM ADDRESS BUS:** The 82091AA decodes the standard ISA I/O address space using SA[9:0]. SA10 is used along with SA[9:0] to decode the extended register set of the ECP parallel port. SA[10:0] connects directly to the ISA system address bus.
SD[7:0]	I/O	**SYSTEM DATA BUS:** SD[7:0] is a bi-directional data bus. Data is written to and read from the 82091AA on these signal lines. SD[7:0] connect directly to the ISA system data bus.
IORC#	I	**I/O READ COMMAND STROBE:** IORC# is an I/O access read control signal. When a valid internal address is decoded by the 82091AA and IORC# is asserted, data at the decoded address location is driven onto the SD[7:0] signal lines.
IOWC#	I	**I/O WRITE COMMAND STROBE:** IOWC# is an I/O access write control signal. When a valid internal address is decoded by the 82091AA and IOWC# is asserted, data on the SD[7:0] signal lines is written into the decoded address location at the rising edge of IOWC#.
NOWS#	O	**NO WAIT-STATES:** End data transfer signal. The 82091AA asserts NOWS# when a valid internal address is decoded by the 82091AA and the IORC# or IOWC# signal is asserted. This reduces the total bus cycle time by eliminating the wait-states associated with the default 8-bit I/O cycles. NOWS# is not asserted for IDE accesses or DMA accesses. This is an open drain output pin.
IOCHRDY	O	**I/O CHANNEL READY:** The 82091AA uses this signal for parallel port data transfers when the parallel port is in EPP mode. In this case, the 82091AA negates IOCHRDY to extend the cycle to allow for completion of transfers to/from the peripheral attached to the parallel port. When the parallel port is in EPP mode, the 82091AA negates IOCHRDY to lengthen the ISA Bus cycle if the parallel port BUSY signal is asserted. The 82091AA also uses IOCHRDY during hardware configuration time (see Section 4.0, AIP Configuration). If IOWC#/IORC# is asserted to the 82091AA during hardware configuration time, the 82091AA negates IOCHRDY until hardware configuration time is completed. This is an open drain output pin.
AEN	I	**ADDRESS ENABLE:** AEN is used during DMA cycles to prevent the 82091AA from misinterpreting DMA cycles from valid I/O cycles. When negated, AEN indicates that the 82091AA may respond to address and I/O commands addressed to the 82091AA. When asserted, AEN informs the 82091AA that a DMA transfer is occurring. When AEN is asserted and a xDACK# signal is asserted, the 82091AA responds to the cycle as a DMA cycle.
RSTDRV	I	**RESET DRIVE:** RSTDRV forces the 82091AA to a known state. All 82091AA registers are set to their default state.
X1/OSC	I	**CRYSTAL1/OSCILLATOR:** Main clock input signal can be a 24 MHz crystal connected across X1 and X2 or a 24 MHz TTL level clock input connected to X1.
X2	I	**CRYSTAL2:** This signal pin is connected to one side of the crystal when a crystal oscillator is used to provide the main clock. If an external oscillator/clock is connected to X1, this pin is not used and left unconnected.

2.1 Host Interface Signals (Continued)

Signal Name	Type	Description
DMA SIGNALS		
FDDREQ	O	**FLOPPY DISK CONTROLLER DMA REQUEST:** The 82091AA asserts FDDREQ to request service from a DMA controller for the FDC module. This signal is enabled/disabled by bit 3 of the Digital Output Register (DOR). When disabled, FDDREQ is tri-stated.
FDDACK#	I	**FLOPPY DISK CONTROLLER DMA ACKNOWLEDGE:** The DMA controller asserts this signal to acknowledge the FDC DMA request. When asserted, the IORC# and IOWC# inputs are enabled during DMA transfers. This signal is enabled/disabled by bit 3 of the DOR.
PPDREQ	O	**PARALLEL PORT DMA REQUEST:** Parallel port DMA service request to the system DMA controller. This signal is only used when the parallel port is in ECP hardware mode and is always negated when the parallel port is not in this mode. In ECP hardware mode DMA requests are enabled/disabled by bit 3 of the ECP Extended Control Register (ECR). When disabled, PPDREQ is tri-stated.
PPDACK#	I	**PARALLEL PORT DMA ACKNOWLEDGE:** The DMA controller asserts this signal to acknowledge the parallel port DMA request. When asserted the IORC# and IOWC# inputs are enabled during DMA transfers. This signal is enabled/disabled by bit 3 of the ECR Register.
TC	I	**TERMINAL COUNT:** The system DMA controller asserts TC to indicate it has reached the last programmed data transfer. TC is accepted only when FDDACK# or PPDACK# is asserted.
INTERRUPT SIGNALS		
IRQ3, IRQ4	O	**INTERRUPT 3 AND 4:** IRQ3 and IRQ4 are associated with the serial ports and can be programmed (via the AIPCFG2 Register) to be either active high or active low. These signals can be configured for a particular serial channel via hardware configuration (at powerup) or by software configuration. **Under Hardware Configuration** IRQ3 is used as a serial port interrupt if the serial port is configured at address locations 2F8h–2FFh or 2E8h–2EFh. IRQ4 is used as a serial port interrupt if the serial port is configured at address locations 3F8h–3FFh or 3E8h–3EFh. **Under Software configuration** IRQ3 and IRQ4 are independently configured (i.e., the IRQ does not automatically track the communication port address assignment). These interrupts are enabled/disabled globally via bit 3 of the serial port Modem Control Register (MCR) and for specific conditions via the Interrupt Enable Register (IER). IRQ3 and IRQ4 are tri-stated when not enabled.
IRQ5, IRQ7	O	**INTERRUPT REQUEST 5:** IRQ5 and IRQ7 are associated with the parallel port and can be programmed (via AIPCFG2 Register) to be either active high or active low. Either IRQ5 or IRQ7 is enabled/disabled via PCFG1 Register to signal a parallel port interrupt. The interrupt not selected is disabled and tri-stated. During hardware configuration (see Section 4.0, AIP Configuration), IRQ5 is used if the parallel port is assigned to 278h–27Fh and IRQ7 is used if the parallel port interrupt is assigned to either 3BCh–3BFh or 378h–37Fh.

2.1 Host Interface Signals (Continued)

Signal Name	Type	Description
INTERRUPT SIGNALS (Continued)		
IRQ6	O	**INTERRUPT REQUEST 6:** IRQ6 is associated with the floppy disk controller and can be programmed (via the AIPCFG2 Register) to be either active high or active low. In non-DMA mode this signal is asserted to signal when a data transfer is ready. IRQ6 is also asserted to signal the completion of the execution phase for certain FDC commands. This signal is enabled/disabled by the DMAGATE bit in the Digital Output Register of the FDC. The signal is tri-stated when disabled.

2.2 Floppy Disk Controller Interface

Signal Name	Type	Description
RDDATA#	I	**READ DATA:** Serial data from the disk drive.
WRDATA#	O	**WRITE DATA:** MFM serial data to the disk drive. Precompensation value is selectable through software.
HDSEL	O	**HEAD SELECT:** Selects which side of a disk is to be accessed. When asserted (low), side 1 is selected. When negated (high), side 0 is selected.
STEP#	O	**STEP:** STEP# supplies step pulses (asserted) to the drive to move the head between the tracks during a seek operation.
DIR#	O	**DIRECTION:** Controls the direction the head moves when a step signal is present. The head moves toward the center when DIR# is asserted and away from the center when negated.
WE#	O	**WRITE ENABLE:** WE# is a disk drive control signal. When asserted, WE# enables the head to write to the disk.
TRK0#	I	**TRACK0:** The disk drive asserts this signal to indicate that the head is on track 0.
INDX#	I	**INDEX:** The disk drive asserts this signal to indicate the beginning of the track.
WP#	I	**WRITE PROTECT:** The disk drive asserts this signal to indicate that the disk drive is write-protected.
DSKCHG	I	**DISK CHANGE:** The disk drive asserts this signal to indicate that the drive door has been opened. The state of this signal input is available in the Digital Input Register (DIR#).
DRIVDEN0 DRIVDEN1	O	**DRIVE DENSITY:** These signals are used by the disk drive to configure the drive for the appropriate media density. These signals are controlled by the FDC's Drive Specification Command.

2.2 Floppy Disk Controller Interface (Continued)

Signal Name	Type	Description
FDME1#/ DSEN#[1]	O	**FLOPPY DRIVE MOTOR ENABLE 1, IDLE, OR DRIVE SELECT ENABLE:** This signal pin has two functions[1]. FDME1# is the motor enable for drive 1. FDME1# is directly controlled via the Digital Output Register (DOR) and is a function of the mapping based on the BOOTSEL bits in the Tape Drive Register (TDR). The Drive Select Enable (DSEN#) function is only used in a four floppy drive system (see Appendix A, FDC Four Drive Support).
FDS1#/ MDS1[1]	O	**FLOPPY DRIVE SELECT1, POWERDOWN, OR MOTOR DRIVE SELECT 1:** This signal pin has two functions[1]. FDS1# is the floppy drive select for drive 1. FDS1# is controlled by the select bits in the DOR and is a function of the mapping based on the BOOTSEL bits in the TDR. The Motor Drive Select 1 (MDS1) function is only used in a four floppy drive system (see Appendix A, FDC Four Drive Support).
FDME0#/ MEEN#[1]	O	**FLOPPY DRIVE MOTOR ENABLE 0 OR MOTOR ENABLE ENABLE:** This signal pin has two functions[1]. FDME0# is the motor enable for drive 0. FDME0# is directly controlled via the Digital Output Register (DOR) and is a function of the mapping based on the BOOTSEL bits in the Tape Drive Register (TDR). The Motor Enable Enable (MEEN#) function is only used in a four floppy drive system (see Appendix A, FDC Four Drive Support).
FDS0#/ MDS0[1]	O	**FLOPPY DRIVE SELECT 0 OR MOTOR DRIVE SELECT 0:** This signal pin has two functions[1]. FDS0# is the floppy drive select for drive 0. This output is controlled by the drive select bits in the DOR and is a function of the mapping based on BOOTSEL bits in the TDR. The Motor Drive Select 0 (MDS0) function is only used in a four floppy drive system (see Appendix A, FDC Four Drive Support).

NOTE:

1. The function selected for these pins is based on the FDDQTY bit in the FCFG1 Register as shown in the following table.

Signal Pin	2 Drive System (FDDQTY=0)	4 Drive System (FDDQTY=1)
FDME1#/DSEN#	FDME1#	DSEN#
FDS1#/MDS1#	FDS1#	MDS1
FDME0#/MEEN#	FDME0#	MEEN#
FDS0#/MDS0	FDS0#	MDS0

When FDDQTY=1, these signal pins are used to control an external decoder for a four floppy disk drive system as described in Appendix A, FDC Four Drive Support.

2.3 Serial Port Interface

Serial Port A signal names end in the letter A and Serial Port B signal names end in the letter B. Serial Port A and B signals have the same functionality.

Signal Name	Type	Description
CTSA#, CTSB#	I	**CLEAR TO SEND:** When asserted, this signal indicates that the modem or data set is ready to exchange data. The CTS# signal is a modem status input whose condition the CPU can determine by reading the CTS bit in Modem Status Register (MSR) for the appropriate serial port. The CTS bit is the compliment of the CTS# signal. The DCTS bit in the MSR indicates whether the CTS# input has changed state since the previous reading of the MSR. CTS# has no effect on the transmitter.
DCDA#, DCDB#	I	**DATA CARRIER DETECT:** When asserted, this signal indicates that the data carrier has been detected by the modem or data set. The DCD# signal is a modem status whose condition the CPU can determine by reading the DCD bit in the MSR for the appropriate serial port. The DCD bit is the compliment of the DCD# signal. The DDCD bit in the MSR indicates whether the DCD# input has changed state since the previous reading of the MSR. DCD# has no effect on the transmitter.
DSRA#, DSRB#	I	**DATA SET READY:** When asserted, this signal indicates that the modem or data set is ready to establish the communications link with the serial port module. The DSR# signal is a modem status whose condition the CPU can determine by reading the DSR bit in the MSR for the appropriate serial channel. The DSR bit is the compliment of the DSR# signal. The DSR bit in the MSR indicates whether the DSR# input has changed state since the previous reading of the MSR. DSR# has no effect on the transmitter.
DTRA#, DTRB#	I/O	**DATA TERMINAL READY:** DTRA#/DTRB# are outputs during normal system operations. When asserted, this signal indicates to the modem or data set that the serial port module is ready to establish a communications link. The DTR# signal can be asserted via the Modem Control Register (MCR). A hard reset negates this signal. **Hardware Configuration** These signals are only inputs during hardware configuration time (RSTDRV asserted and for a short time after RSTDRV is negated). (See Section 4.0, AIP Configuration.)
RIA#, RIB#	I	**RING INDICATOR:** When asserted, this signal indicates that a telephone ringing signal has been received by the modem or data set. The RI# signal is a modem status input whose condition the CPU can determine by reading the RI bit in the MSR for the appropriate serial channel. The RI bit is the compliment of the RI# signal. The TERI bit in the MSR indicates whether the RI# input has changed from low to high since the previous reading of the MSR.

2.3 Serial Port Interface (Continued)

Signal Name	Type	Description
RTSA#, RTSB#	I/O	**REQUEST TO SEND:** RTSA#/RTSB# are outputs during normal system operations. When asserted, this signal informs the modem or data set that the serial port module is ready to exchange data. The RTS# signal can be asserted via the RTS bit in the Modem Control Register. A hard reset negates this signal. **Hardware Configuration** These signals are only inputs during hardware configuration time (RSTDRV asserted and for a short time after RSTDRV is negated). (See Section 4.0, AIP Configuration.)
SINA, SINB	I	**SERIAL INPUT:** Serial data input from the communications link. (Peripheral device, modem, or data set.)
SOUTA, SOUTB	I/O	**SERIAL OUTPUT:** SOUTA/SOUTB are serial data outputs to the communications link during normal system operations. (Peripheral device, modem, or data set.) The SOUT signal is set to a marking state (logic 1) after a hard reset. **Test Mode** In test mode (selected via the SACFG2 or SBCFG2 Registers), the baudout from the baud rate generator is output on SOUTx. **Hardware Configuration** These signals are only inputs during hardware configuration time (RSTDRV asserted and for a short time after RSTDRV is negated). (See Section 4.0, AIP Configuration.)

2.4 IDE Interface

Signal Name	Type	Description
IO16#	I	**16-BIT I/O:** This signal is driven by I/O devices on the ISA Bus to indicate support for 16-bit I/O bus cycles. The IDE interface asserts this signal to the 82091AA to indicate support for 16-bit transfers. For IDE transfers, the 82091AA asserts HEN# when IO16# is asserted.
IDECS[1:0]#	I/O	**IDE CHIP SELECT:** IDECS[1:0]# are outputs during normal system operation and are chip selects for the IDE interface. IDECS[1:0]# select the Command Block Registers of the IDE device and are decoded from SA[9:3] and AEN. **Hardware Configuration** These signals are only inputs during hardware configuration time (RSTDRV asserted). (See Section 4.0, AIP Configuration.)

2.4 IDE Interface (Continued)

Signal Name	Type	Description
DEN#	I/O	**DATA ENABLE:** DEN# is an output during normal system operations and is a data enable for an external data buffer for all 82091AA and IDE accesses. The SD[7:0] signals can be connected directly to the ISA. In this case, the DEN# signal is not used. However, an external buffer can be used to isolate the SD[7:0] signals from the 240 pF loading of the ISA Bus. With an external buffer implementation, DEN# controls the external buffers for transfers to/from the ISA Bus. **Hardware Configuration** This signal is only an input during hardware configuration time (RSTDRV asserted). (See Section 4.0, AIP Configuration.)
HEN#	I/O	**IDE UPPER DATA TRANSCEIVER ENABLE:** HEN# is an output during normal system operations and is a high byte data transceiver enable signal for the IDE hard disk drive interface. HEN# is asserted for I/O accesses to the IDE data register when the drive asserts IO16#. **Hardware Configuration** This signal is only an input during hardware configuration time (RSTDRV asserted). (See Section 4.0, AIP Configuration.)

2.5 Parallel Port External Buffer Control/Game Port

Signal Name	Type	Description
PPDIR/GCS#	I/O	**PARALLEL PORT DIRECTION (PPDIR) or GAME PORT CHIP SELECT (GCS#):** This signal is an output during normal operations and provides the PPDIR and GCS# functions as follows: **PPDIR** This signal pin functions as a parallel port direction control output when the 82091AA is configured for software motherboard mode (SWMB). For configuration details, see Section 4.0, AIP Configuration. If external buffers are used on PD[7:0], PPDIR can be used to control the buffer direction. The 82091AA drives this signal low when PD[7:0] are outputs and the 82091AA drives this signal high when PD[7:0] are inputs. Note that if a configuration mode other than SWMB is selected, this signal pin is a game port chip select and does not track the PD[7:0] signal direction. **GCS#** This signal pin functions as a game port chip select output when 82091AA configuration is set for Software Add-In (SWAI), Hardware Basic (HWB), or Hardware Extended (HWE) modes. When the host accesses I/O address 201h, GCS# is asserted. **Hardware Configuration** This signal is only an input during hardware configuration time (RSTDRV asserted). (See Section 4.0, AIP Configuration.)

2.6 Parallel Port Interface

The 82091AA parallel port is a multi-function interface that can be configured for one of four hardware modes (see Section 4.0, AIP Configuration). The hardware modes are ISA-Compatible, PS/2-Compatible, EPP, and ECP modes. These parallel port modes support the compatibility, nibble, byte, EPP and ECP parallel interface protocols described in the IEEE 1284 standard. The operation and use of the interface signal pins are a function of the parallel port hardware mode selected and the protocol used.

Table 1 shows a matrix of the 82091AA parallel port signal names and corresponding signal names for each of the protocols. Sections 2.6.1–2.6.5 provide a signal description for the five interface protocols. Note that the 82091AA hardware operations are the same for Compatibility and Nibble protocols. The signals, however, are controlled and used differently via software and the peripheral device.

Table 1. Parallel Port Signal Name Cross Reference

82091AA Signal Names	Compatibility Protocol Signal Names	Nibble Protocol Signal Names	Byte Protocol Signal Names	EPP Protocol Signal Names	ECP Protocol Signal Names
STROBE#	Strobe#	—	HostCLK	Write#	HostClk
BUSY	Busy	PtrBusy	PtrBusy	Wait#	PeriphAck
ACK#	Ack#	PtrClk	PtrClk	Intr	PeriphClk#
SELECT	Select	Xflag	Xflag	Xflag	Xflag
PERROR	PError	AckDataReq	AckDataReq	AckDataReq	AckReverse#
FAULT#	Fault#	DataAvail#	DataAvail#	DataAvail#	PeriphRequest#
INIT#	Init#	—	—	Init#	ReverseRequest#
AUTOFD#	AutoFd#	HostBusy	HostBusy	DStrb#	HostAck
PD[7:0]	Data[8:1]	—	Data[8:1]	Data[8:1]	Data[8:1]
SELECTIN#	SelectIn#	—	—	AStrb#	ECP Mode

NOTE:
Not all parallel port signal pins are used for certain parallel port interface protocols. These signals are labeled "—".

2.6.1 COMPATIBILITY PROTOCOL SIGNAL DESCRIPTION

Except for the data bus, the 82091AA and compatibility protocol signal names are the same. For the data bus, the 82091AA signal names PD[7:0] corresponds to the compatibility protocol signal names Data[8:1].

82091AA Signal Name	Type	Compatibility Protocol Signal Name and Description
STROBE#	O	**STROBE:** The host asserts STROBE# to latch data into the peripheral device's input latch. This signal is controlled via the PCON Register.
BUSY	I	**BUSY:** BUSY is asserted by the peripheral to indicate that the peripheral device is not ready to receive data. The status of this signal line is reported in the PSTAT Register.
ACK#	I	**ACKNOWLEDGE:** The printer asserts this signal to indicate that it has received the data and is ready for new data. The status of this signal line is reported in the PSTAT Register.
SELECT	I	**SELECT:** SELECT is asserted by the peripheral device to indicate that the device is on line. The status of this signal line is reported in the PSTAT Register.
PERROR	I	**PAPER ERROR:** The peripheral device asserts PERROR to indicate that it has encountered an error in the paper path. The exact meaning varies from peripheral device to peripheral device. The status of this signal line is reported in the PSTAT Register.
FAULT#	I	**FAULT:** FAULT# is asserted by the peripheral device to indicate that an error has occurred. The status of this signal line is reported in the PSTAT Register.
INIT#	O	**INITIALIZE:** The host asserts INIT# to issue a hardware reset to the peripheral device. This signal is controlled via the PCON Register.
AUTOFD#	O	**AUTO FEED:** AUTOFD# is asserted by the host to put the peripheral device into auto-line feed mode. This means that when software asserts this signal, the printer is instructed to advance the paper one line for each carriage return encountered. This signal is controlled via the PCON Register.
PD[7:0]	O	**DATA:** Forward channel data.
SELECTIN#	O	**SELECT INPUT:** SELECTIN# is asserted by the host to select a peripheral device. This signal is controlled via the PCON Register.

2.6.2 NIBBLE PROTOCOL SIGNAL DESCRIPTION

The Nibble protocol assigns the following signal operation to the parallel port pins. The name in bold at the beginning of the signal description column is the Nibble protocol signal name. The terms **assert** and **negate** are used in accordance with the 82091AA signal name as described at the beginning of Section 2.0. For example, AUTOFD# (HostBusy) asserted refers to AUTOFD# (HostBusy) at a low level.

82091AA Signal Name	Type	Nibble Protocol Signal Name and Description
STROBE#	O	**STROBE:** The host controls this signal via the PCON Register and STROBE# should be held negated by the host.
BUSY	I	**PRINTER BUSY (PtrBusy):** The peripheral drives this signal to transfer data bits 3 and 7 sequentially. The status of this signal line is reported in the PSTAT Register.
ACK#	I	**PRINTER CLOCK (PtrClk):** The peripheral device asserts ACK# (PtrClk) to indicate to the host that data is available. The signal is subsequently asserted to qualify data being sent to the host. The status of this signal line is reported in the PSTAT Register. If interrupts are enabled via the PCON Register, the assertion of this signal causes a host interrupt to be generated.
SELECT	I	**XFLAG:** The peripheral device drives this signal to transfer data bits 1 and 5 sequentially. The status of this signal line is reported in the PSTAT Register.
PERROR	I	**ACKNOWLEDGE DATA REQUEST (AckDataReq):** This signal is initially high. The peripheral device drives this signal low to acknowledge HostBusy assertion. PERROR is subsequently used to transfer data bits 2 and 6 sequentially. The status of this signal line is reported in the PSTAT Register.
FAULT#	I	**DATA AVAILABLE (DataAvail):** The peripheral device asserts FAULT# (DataAvail) to indicate data availability. Subsequently used to transfer data bits 0 and 4 sequentially. The status of this signal line is reported in the PSTAT Register.
INIT#	O	**INITIALIZE:** The host controls this signal via the PCON Register.
AUTOFD#	O	**HOST BUSY (HostBusy):** The host negates AUTOFD# (HostBusy) in response to ACK# being asserted. This signal is subsequently driven low to enable the peripheral to transfer data to the host. AUTOFD# is then driven high to acknowledge receipt of byte data. This signal is controlled via the PCON Register.
PD[7:0]	O	**DATA:** This 8-bit output data path to the peripheral Host data is written to the peripheral attached to the parallel port interface on these signal lines.
SELECTIN#	O	**SELECT INPUT:** This signal is controlled by the PCON Register.

2.6.3 BYTE MODE SIGNAL DESCRIPTION

The Byte protocol assigns the following signal operation to the parallel port pins. The name in bold at the beginning of the signal description column is the Byte protocol signal name. The terms **assert** and **negate** are used in accordance with the 82091AA signal name as described at the beginning of Section 2.0. For example, STROBE# (HostClk) asserted refers to STROBE# (HostClk) at a low level.

82091AA Signal Name	Type	Byte Protocol Signal Name and Description
STROBE#	O	**HOST CLOCK (HostClk):** This signal is strobed low by the host to acknowledge receipt of data. Note that the peripheral must not interpret this as a latch strobe for forward channel data.
BUSY	I	**PRINTER BUSY (PtrBusy):** The peripheral device asserts BUSY (PtrBusy) to provide forward channel peripheral busy status. The status of this signal line is reported in the PSTAT Register.
ACK#	I	**PRINTER CLOCK (PtrClk):** The peripheral device asserts ACK# (PtrClk) to indicate to the host that data is available. The signal is subsequently asserted to qualify data being sent to the host. The status of this signal line is reported in the PSTAT Register. If interrupts are enabled via the PCON Register, the assertion of this signal causes a host interrupt to be generated.
SELECT	I	**XFLAG:** SELECT (XFLAG) is asserted by the peripheral device to indicate that the device is on line. The status of this signal line is reported in the PSTAT Register.
PERROR	I	**ACKNOWLEDGE DATA REQUEST (AckDataReq):** This signal is initially high. The peripheral device drives this signal low to acknowledge HostBusy assertion. The status of this signal line is reported in the PSTAT Register.
FAULT#	I	**DATA AVAILABILITY (DataAvail):** The peripheral device asserts FAULT# (DataAvail) to indicate data availability. The status of this signal line is reported in the PSTAT Register.
INIT#	O	**INITIALIZE:** The host controls this signal via the PCON Register and INIT# should be held in the negated state.
AUTOFD#	O	**HOST BUSY (HostBusy):** The host negates AUTOFD# (HostBusy) in response to ACK# being asserted.The signal is subsequently driven low to enable the peripheral to transfer data to the host. AUTOFD# is then driven high to acknowledge receipt of nibble data. This signal is controlled via the PCON Register.
PD[7:0]	O	**DATA:** This 8-bit data bus is used for bi-directional data transfer.
SELECTIN#	I/O	**SELECT INPUT:** This signal is controlled by the PCON Register.

2.6.4 ENHANCED PARALLEL PORT (EPP) PROTOCOL SIGNAL DESCRIPTION

EPP protocol assigns the following signal operation to the parallel port pins. The name in bold at the beginning of the signal description column is the EPP mode signal name. The terms **assert** and **negate** are used in accordance with the 82091AA signal name as described at the beginning of Section 2.0. For example, BUSY (Wait#) asserted refers to BUSY (Wait#) being high.

82091AA Signal Name	Type	EPP Protocol Signal Name and Description
STROBE#	O	**WRITE (Write#):** STROBE# (Write#) indicates an address or data read/write operation to the peripheral. The 82091AA drives this signal low for a write and high for a read.
BUSY	I	**WAIT (Wait#):** The peripheral sets BUSY (Wait#) low to indicate that the device is not ready. When BUSY signal is low, the 82091AA negates IOCHRDY on the ISA Bus to lengthen the I/O cycles. The peripheral device sets BUSY (Wait#) high to indicate that transfer of data or address is completed.
ACK#	I	**INTERRUPT REQUEST (Intr):** The peripheral asserts ACK# (Intr) to generate an interrupt the host. When this signal is low and interrupts are enabled via bit 4 of the PCON Register, the 82091AA generates an interrupt request (via either IRQ5 or IRQ7) to the host.
SELECT	I	**SELECT:** SELECT is asserted by the peripheral device to indicate that the device is on line. The status of this signal line is reported in the PSTAT Register.
PERROR	I	**PAPER ERROR:** The peripheral device asserts PERROR to indicate that it has encountered an error in the paper path. The exact meaning varies from peripheral device to peripheral device. The status of this signal line is reported in the PSTAT Register.
FAULT#	I	**FAULT:** FAULT# is asserted by the peripheral device to indicate that an error has occurred. The status of this signal line is reported in the PSTAT Register.
INIT#	O	**INITIALIZE:** The host asserts INIT# to issue a hardware reset to the peripheral device. This signal is controlled via the PCON Register.
AUTOFD#	O	**DATA STROBE (DStrb#):** The 82091AA asserts AUTOFD# (DStrb#) to indicate that valid data is present on PD[7:0] and is used by the peripheral to latch data during write cycles. For reads, the 82091AA reads in data from PD[7:0] when this signal is asserted.
PD[7:0]	I/O	**DATA:** This 8-bit bi-directional bus provides addresses or data during the write cycles and supplies addresses or data to the 82091AA during the read cycles.
SELECTIN#	O	**ADDRESS STROBE (AStrb#):** The 82091AA asserts SELECTIN# (AStrb#) to indicate that a valid address is present on PD[7:0] and is used by the peripheral to latch addresses during write cycles. For reads, the 82091AA reads in an address from PD[7:0] when this signal is asserted.

2.6.5 EXTENDED CAPABILITIES PORT (ECP) PROTOCOL SIGNAL DESCRIPTION

ECP protocol assigns the following signal operation to the parallel port pins. The name in bold at the beginning of the signal description column is the ECP protocol signal name. The terms **assert** and **negate** are used in accordance with the 82091AA signal name as described at the beginning of Section 2.0. For example, STROBE# (HostClk) asserted refers to STROBE# (HostClk) being low.

82091AA Signal Name	Type	ECP Protocol Signal Name and Description
STROBE#	O	**HOST CLOCK (HostClk):** In the forward direction, the 82091AA asserts STROBE# (HostClk) to instruct the peripheral to latch the data on PD[7:0]. During write operations, the peripheral should latch data on the rising edge of STROBE# (HostClk). STROBE# (HostClk) handshakes with BUSY (PeriphAck) during write operations and is negated after the 82091AA detects BUSY (PeriphAck) asserted. STROBE# (HostClk) is not asserted by the 82091AA again until BUSY (PeriphAck) is detected negated. For read operations (reverse direction), STROBE# (HostClk) is not used.
BUSY	I	**PERIPHERAL ACKNOWLEDGE (PeriphAck):** The peripheral device asserts this signal during a host write operation to acknowledge receipt of data. The peripheral device then negates the signal after STROBE# is detected high to terminate the transfer. For host write operations (forward direction), this signal handshakes with STROBE# (HostClk). During a host read operation (reverse direction), BUSY (PeriphAck) is normally low and is driven high by the peripheral to identify Run Length Encoded (RLE) data.
ACK#	I	**PERIPHERAL CLOCK (PeriphClk):** During a peripheral to host transfer (reverse direction), ACK# (PeriphClk) is asserted by the peripheral to indicate data is valid on the data bus and then negated after AUTOFD# is detected high. This signal handshakes with AUTOFD# to transfer data.
SELECT	I	**XFLAG (Xflag):** This signal is asserted by the peripheral to indicate that it is on-line. The status of this signal line is reported in the PSTAT Register.
PERROR	I	**ACKNOWLEDGE REVERSE (AckReverse#):** PERROR (AckReverse#) is driven low by the peripheral to acknowledge a reverse transfer request by the host. This signal handshakes with INIT# (ReverseRequest#). The status of this signal line is reported in the PSTAT Register.
FAULT#	I	**PERIPHERAL REQUEST (PeriphRequest#):** The peripheral asserts FAULT# (PeriphRequest#) to request a reverse transfer. The status of this signal line is reported in the PSTAT Register.
INIT#	O	**REVERSE REQUEST (ReverseRequest#):** The host controls this signal via the PCON Register to indicate the transfer direction. The host asserts this signal to request a reverse transfer direction and negates the signal for a forward transfer direction.
AUTOFD#	O	**HOST ACKNOWLEDGE (HostAck):** The 82091AA asserts AUTOFD# (HostAck) to request data from the peripheral (reverse direction). This signal handshakes with ACK# (PeriphClk). AUTOFD# (HostAck) is negated when the peripheral indicates valid state of the data bus (i.e., ACK# is detected asserted). In the forward direction, AUTOFD# (HostAck) indicates whether PD[7:0] contain an address/RLE or data. The 82091AA asserts this signal to identify an address/RLE transfer and negates it to identify a data transfer.
PD[7:0]	I/O	**DATA:** PD[7:0] is a bi-directional data bus that transfers data, addresses, or RLE data.
SELECTIN#	O	**ECP MODE (ECPmode):** The host (via the PCON Register) negates this signal during ECP mode operation.

2.7 Hard Reset Signal Conditions

Table 1 shows the state of all 82091AA output and bi-directional signals during hard reset (RSTDRV asserted). The strapping options described in Section 4.0, AIP Configuration are sampled when the 82091AA is hard reset.

Table 2. Output and I/O Signal States During a Hard Reset

Signal Name	State
ACK#	—
AEN	—
AUTOFD#	Tri-state
BUSY	—
CTS[A,B]#	—
DCD[A,B]#	High
DEN#	High(1)
DIR#	High
DRVDEN[1:0]0	Low
DSKCHG#	—
DTR[A,B]#	High(1)
FAULT#	—
FDDACK#	—
FDDREQ	Tri-state
FDME0#/MEEN#	High
FDME1#/DSEN#	High
FDS0#/MDS0	High
FDS1#/MDS1	High
HDSEL	High
HEN#	High(1)
IDECS[1,0]#	High(1)
INDX#	—
INIT#	Low
IO16#	—
IOCHRDY	Tri-state(2)
IORC#	—
IOWC#	—
IRQ[7:3]	Tri-state
NOWS#	Tri-state
PD[7:0]	Low
PERROR	—
PPDACK#	—
PPDREQ	Tri-state
PPDIR/GCS#	High(1)
RDDATA	—
RI[A,B]#	—
RSTDRV	—
RTS[A,B]#	High(1)
SA[10,0]	—
SD[7:0]	Tri-state
SELECT	—
SELECTIN#	Tri-state
SIN[A,B]	—
SOUT[A,B]	High(1)
STEP#	High
STROBE#	Tri-state
TC	—
TRK0#	—
WE#	High
WP#	—
WRDATA#	High
X1/OSC	—
X2	—

NOTES:

1. During and immediately after a hard reset, this signal is an input for hardware configuration. After the hardware configuration time, these signals go to the state specified in the table.
2. If IORC# or IOWC# is asserted, IOCHRDY will be asserted by the IOCHRDY.
3. Dashes represent input signals.

2.8 Power And Ground

Signal Name	Type	Description
V_{SS}	I	**GROUND:** The ground reference for the 82091AA.
V_{CC}	I	**POWER:** The 5V/3.3V(1) modes are selected via strapping options at power-up (see Section 4.2, hardware Configuration). When strapping options (V_{SEL}) are set to 5V, the V_{CC} pins must be connected to 5V. When strapping options are set to 3.3V, the V_{CC} pins must be connected to 3.3V.
V_{CCF}	I	**POWER:** The 5V/3.3V(1) power supply for the 82091AA. In 5V or 3.3V power supply modes (non-mixed mode), the voltage applied to V_{CCF} is the same voltage as applied to V_{CC}. For mixed mode operations, 5V is applied to V_{CCF}. This voltage provides 5V reference for the parallel port and floppy disk controller interfaces. Note that in mixed mode, 3.3V is applied to V_{CC}.

NOTE:
1. 3.3V operation is available only in the 82091AA.

3.0 I/O ADDRESS ASSIGNMENTS

The 82091AA assigns CPU I/O address locations to its game port chip select, IDE interface, serial ports, parallel port, floppy disk controller, and the 82091AA configuration registers as indicated in Table 3. Except for the game port chip select (address 201h), address assignments are configurable. For example, the serial port can be assigned to one of eight address blocks. The parallel port can be assigned to one of three address blocks, and the IDE interface and floppy disk controller can be assigned to one of two address blocks. These address assignments are made during 82091AA configuration (either hardware configuration at powerup or a hard reset, or software configuration by programming the 82091AA configuration registers). In addition, the 82091AA configuration registers can be located at one of two address blocks during hardware configuration.

All of the 82091AA address locations are located in the host I/O address space. The address block assignments are shown in Table 3. The first hex address in the Address Block column represents the base address for that particular block.

Table 3. AIP Address Assignments

Address Block (ISA Bus)	Assignment
170–177h	IDE Interface—Secondary Address Block
1F0–1F7h	IDE Interface—Primary Address Block
201h	Game Port Chip Select
220–227h	Serial Port
228–22Fh	Serial Port
238–23Fh	Serial Port
26E–26Fh	82091AA Configuration Registers—Primary Address Block (022–023h on X-Bus)
278–27Fh	Parallel Port
2E8–2EFh	Serial Port
2F8–2FFh	Serial Port
338–33Fh	Serial Port
370–377h	Floppy Disk Controller—Secondary Address Block (376h and 377h are Shared with the IDE Drive Interface Secondary Address)
378–37Fh	Parallel Port
398–399h	82091AA Configuration Registers—Secondary Address Block (024–025h on X-Bus)
3BC–3BFh	Parallel Port (All Mopes Except EPP)
3E8–3EFh	Serial Port
3F0–3F7h	Floppy Disk Controller—Primary Address (3F6h and 3F7h are Shared with the IDE Drive Interface Primary Address)
3F8–3FFh	Serial Port
678–67Ah	Parallel Port (ECP Mode Peripheral Interface Protocol)
778–77Ah	Parallel Port (ECP Mode Peripheral Interface Protocol)
7BC–7BEh	Parallel Port (ECP Mode Peripheral Interface Protocol)

NOTES:

1. The 82091AA does not contain IDE registers. However, the 82091AA provides the address block assignments for accessing the IDE registers that are located in the IDE device.
2. The standard PC/AT* compatible logical I/O address assignments are supported. For example, COM1 (3F8–3FFh) and COM2 (2F8–2FFh) are part of the serial port assignments and LPT1 (3BC–3BFh), LPT2 (378–37Fh), and LPT3 (278–27Fh) are part of the parallel port assignments.

*Other brands and names are the property of their respective owners.

4.0 AIP CONFIGURATION

82091AA configuration consists of setting up overall device operations along with certain functions pertaining to the individual 82091AA modules (parallel port, serial ports, floppy disk controller, and IDE interface). Overall device operations include selecting the clock frequency, power supply voltage, and address assignment for the configuration registers. Overall device operations also enable/disable access to the configuration registers and provide interrupt signal level control. For the individual modules, 82091AA configuration includes module address assignment, interrupt control, module enable/disable, powerdown control, test mode control, module reset, and certain functions specific to each module. The remainder of the functions unique to each module are handled via the individual module registers.

Two methods are provided for configuring the 82091AA—hardware configuration via strapping options at powerup (or whenever RSTDRV is asserted) and software configuration by programming the configuration registers. (For information on hardware configuration, see Section 4.2, Hardware Configuration. For information on software configuration, see Section 4.1, Configuration Registers.)

NOTE:

1. There are four hardware configuration modes—SWMB (Software Motherboard), SWAI (Software Add-In), HWB (Hardware Basic), and HWE (Hardware Extended). Some of these modes can be used without the need for programming the 82091AA configuration registers. Other modes use both hardware configuration strapping options and programming the configuration registers to set up the 82091AA.
2. The 82091AA's operating power supply voltage level, 82091AA clock frequency, and address assignment for the 82091AA configuration registers can only be configured by hardware configuration.

4.1 Configuration Registers

82091AA Configuration Space contains 13 configuration registers. Four of the registers (Product and Revision Identification Registers and the 82091AA Configuration 1 and 2 Registers) provide control and status information for the entire chip. In addition, two registers each for the floppy disk controller, parallel port, serial port A, and serial port B and one register for the IDE interface provide certain module status and control information. The 82091AA configuration registers are indirectly addressed by first writing to the 82091AA Configuration Index Register as described in Section 4.1.1. Thus, the 13 configuration registers occupy two address locations in the host's I/O address space—one for indirectly selecting the specific configuration register and the other for transfering register data. All 82091AA configuration registers are 8-bits wide and are accessed as byte quantities.

Some of the 82091AA Configuration registers described in this section contain reserved bits. These bits are labeled "R". Software must deal correctly with fields that are reserved. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved bit positions are preserved. That is, the value of reserved bit positions must first be read, merged with the new values for other bit positions, and then written back.

In addition to reserved bits within a register, the 82091AA configuration space contains address locations that are labeled "Reserved" (Table 5). While the 82091AA responds to accesses to these I/O addresses by completing the host cycle, writing to a reserved I/O address can result in unintended device operations. Values read from a reserved I/O address should not be used to permit future expansion and upgrades.

During a hard reset (RSTDRV asserted), the 82091AA sets its configuration registers to pre-determined **default** states. The default values are indicated in the individual register descriptions. The following nomenclature is used for register access attributes:

RO **Read Only.** If a register is read only, writes have no effect.

R/W **Read/Write.** A register with this attribute can be read and written. Note that individual bits in some read/write registers may be read only.

4.1.1 CFGINDX, CFGTRGT—CONFIGURATION INDEX REGISTER AND TARGET PORT

I/O Address: Hardware Configurable (see Table 4)
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

CFGINDX and CFGTRGT are used to access 82091AA configuration space where all of the 82091AA configuration registers are located. CFGINDX and CFGTRGT are located in the host I/O address space and the address locations are hardware configurable as shown in Table 4. CFGINDX is an 8-bit register that contains the address index of the 82091AA configuration register to be accessed. CFGTRGT is a port for reading data from or writing data to the configuration register whose index address matches the address stored in the CFGINDX Register. Thus, to access a configuration register, CFGINDX must first be programmed with the index address. A software example is provided in this section demonstrating how to access the configuration registers.

Table 4. Configuration Register Access Addresses

Address Selection	X-Bus Implementation		ISA Bus Implementation	
	Index	Target	Index	Target
Primary Address	22h	23h	26Eh	26Fh
Secondary Address	24h	25h	398h	399h

Table 5 summarizes the 82091AA configuration space. Following the table, is a detailed description of each register. The register descriptions are arranged in the order that they appear in Table 5.

Bit	Description
7:0	**82091AA Configuration Register Address Index:** Bits[7:0] correspond to SD[7:0].

Software Configuration

Access Addresses for the two Software Configuration Modes:

	Index	Target
For SWMB Mode Primary Address:	22h	23h
For SWMB Mode Secondary Address:	24h	25h
For SWAI, HWE, and HWB Modes Primary Address:	26Eh	26Fh
For SWAI, HWE, and HWB Modes Secondary Address:	398h	399h

The following pseudo code sequence could be used to access the configuration registers under SWMB primary address:

Configuration register write:

```
OUT 22h, ConfigRegAddr
OUT 23h, ConfigRegData
```

Configuration register read:

```
OUT 22h, ConfigRegAddr
IN 23h
```

Table 5. AIP Configuration Registers

82091AA Configuration Address Index	Abbreviation	Register Name	Access
00h	AIPID	Product Identification	RO
01h	AIPREV	Revision Identification	RO
02h	AIPCFG1	82091AA Configuration 1	R/W
03h	AIPCFG2	82091AA Configuration 2	R/W
04–0Fh	—	Reserved	—
10h	FCFG1	FDC Configuration	R/W
11h	FCFG2	FDC Power Management and Status	R/W
12–1Fh	—	Reserved	—
20h	PCFG	Parallel Port Configuration	R/W
21h	PCFG2	Parallel Port Power Management and Status	R/W
22–2Fh	—	Reserved	—
30h	SACFG1	Serial Port A Configuration	R/W
31h	SACFG2	Serial Port A Power Management and Status	R/W
32–3Fh	—	Reserved	—
40h	SBCFG1	Serial Port B Configuration	R/W
41h	SBCFG2	Serial Port B Power Management and Status	R/W
42–4Fh	—	Reserved	—
50h	ICFG	IDE Configuration	R/W
51–FFh	—	Reserved	—

NOTE:
Writing to a reserved I/O address should not be attempted and can result in unintended device operations.

4.1.2 AIPID—AIP IDENTIFICATION REGISTER

Index Address:	00h
Default Value:	A0h
Attribute:	Read Only
Size:	8 bits

Bit	Description
7:0	**AIP IDENTIFICATION (AIPID):** A value of A0h is assigned to the 82091AA. This 8-bit register combined with the 82091AA Revision Identification Register uniquely identifies the device.

4.1.3 AIPREV—AIP REVISION IDENTIFICATION

Index Address:	01h
Default Value:	00h
Attribute:	Read Only
Size:	8 bits

This register contains two fields that identify the revision of the 82091AA device. The revision number will be incremented for every stepping, even if change is invisible to software.

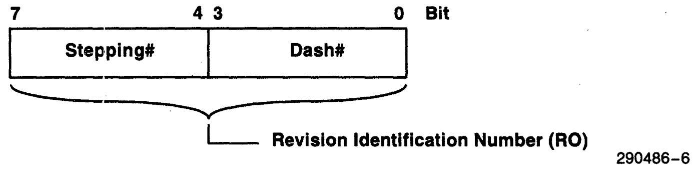


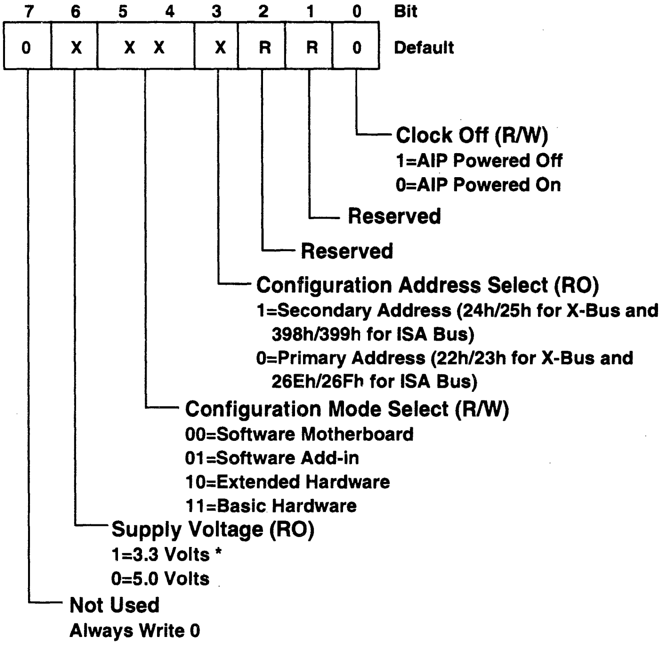
Figure 6. AIP Revision Identification Register

Bit	Description
7:4	**STEP NUMBER:** Contains the hexadecimal representation of the device stepping.
3:0	**DASH NUMBER:** Contains the hexadecimal representation of the dash number of the device stepping.

4.1.4 AIPCFG1—AIP CONFIGURATION 1 REGISTER

Index Address:	02h
Default Value:	Depends upon hardware strap
Attribute:	Read/Write
Size:	8 bits

The AIPCFG1 Register enables/disables master clock circuitry for power management, enables/disables access to the configuration registers, and selects the 82091AA configuration mode. This register provides status for certain hardware configuration selections—the 82091AA clock frequency, power supply voltage, and address assignment for the configuration registers (address locations of the INDEX and TARGET Registers).



NOTES:
*3.3V operation is available only in the 82091AA.
X = Value is determined by hardware strapping options as described in Section 4.2, Hardware Configuration.

Figure 7. AIP Configuration 1 Register

Bit	Description
7	**NOT USED:** Always write to 0.
6	**VOLTAGE SELECT (VSEL):** This bit indicates whether 3.3V or 5V has been selected for the operating power supply voltage during hardware configuration. A 1 indicates that 3.3V is selected and a 0 indicates that 5V is selected. This bit is read only and writes have no effect. **NOTE:** 3.3V operation is available only in the 82091AA.
5:4	**CONFIGURATION MODE SELECT (CFGMOD):** These bits indicate the configuration mode for the 82091AA. After a hard reset, these bits reflect the mode selected by hardware configuration. If configuration register access is not locked out during hardware configuration, software can change the configuration mode by writing to this field. For configuration mode details, (see Section 4.2, Hardware Configuration). **Bits[5:4]** **Configuration Mode** 0 0 Software Motherboard (SWMB) 0 1 Software Add-in (SWAI) 1 0 Extended Hardware (HWE) 1 1 Basic Hardware (HWB)
3	**CONFIGURATION ADDRESS SELECT (CFGADS):** This read only bit indicates the address assignment for the 82091AA configuration registers as selected by hardware configuration. Hardware configuration selects between primary addresses (22h/23h and 26Eh/26Fh) and secondary addresses (24h/25h and 398h/399h) for accessing the 82091AA configuration registers. When CFGADS = 0, the primary addresses are selected and when CFGADS = 1, the secondary addresses are selected.
2	**RESERVED**
1	**RESERVED**
0	**CLOCK OFF (CLKOFF):** The CLKOFF bit is used to implement clock circuitry power management. When CLKOFF = 0, the main clock circuitry is powered on. When CLKOFF = 1, the main clock circuitry is powered off. This capability is independent of the 82091AA's powerdown state. Note that auto powerdown mode and powerdown have no effect over the power state of the clock circuitry.

4.1.5 AIPCFG2—AIP CONFIGURATION 2 REGISTER

Index Address:	03h
Default Value:	0000 0RRR
Attribute:	Read/Write
Size:	8 bits

This register selects the active signal level for IRQ[7:3]. The interrupt signals can be individually programmed for either active high or active low drive characteristics. The active high mode is ISA (non-share) compatible and has tri-state drive characteristic. The active low mode is EISA (sharable) compatible and has an open collector drive characteristic.

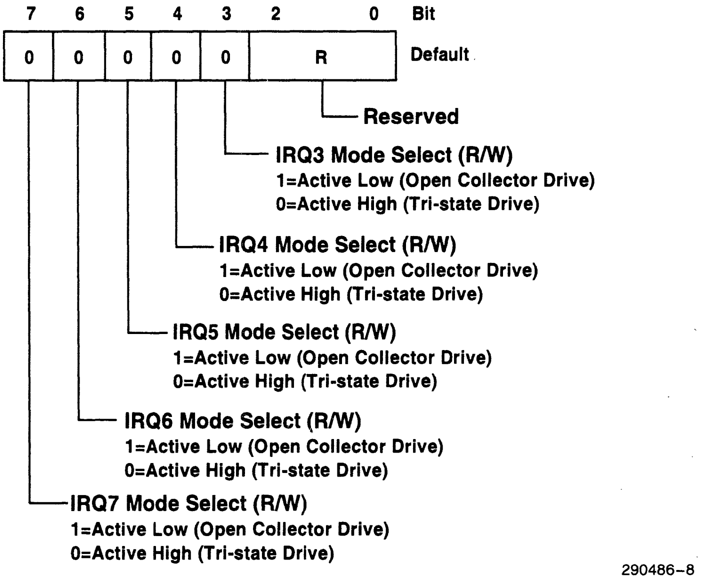

7 6 5 4 3 2 0 Bit
0 0 0 0 0 R Default
Reserved
IRQ3 Mode Select (R/W)
1=Active Low (Open Collector Drive)
0=Active High (Tri-state Drive)
IRQ4 Mode Select (R/W)
1=Active Low (Open Collector Drive)
0=Active High (Tri-state Drive)
IRQ5 Mode Select (R/W)
1=Active Low (Open Collector Drive)
0=Active High (Tri-state Drive)
IRQ6 Mode Select (R/W)
1=Active Low (Open Collector Drive)
0=Active High (Tri-state Drive)
IRQ7 Mode Select (R/W)
1=Active Low (Open Collector Drive)
0=Active High (Tri-state Drive)
290486-8

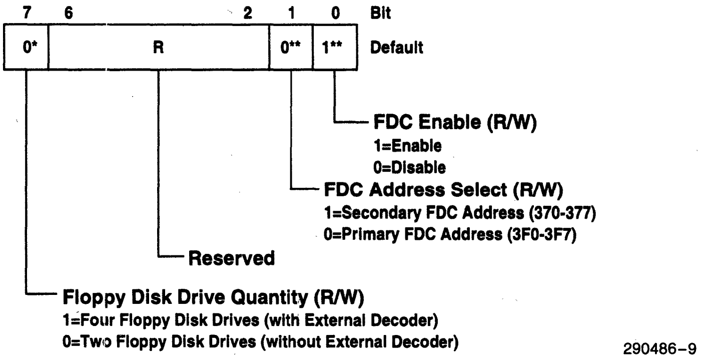
Figure 8. AIP Configuration 2 Register

Bit	Description
7	**IRQ7 MODE SELECT (IRQ7MOD):** When IRQ7MOD = 0, IRQ7 is an active high tri-state drive signal. When IRQ7MOD = 1, IRQ7 is an active low open collector drive signal.
6	**IRQ6 MODE SELECT (IRQ6MOD):** When IRQ6MOD = 0, IRQ6 is an active high tri-state drive signal. When IRQ6MOD = 1, IRQ6 is an active low open collector drive signal.
5	**IRQ5 MODE SELECT (IRQ5MOD):** When IRQ5MOD = 0, IRQ5 is an active high tri-state drive signal. When IRQ5MOD = 1, IRQ5 is an active low open collector drive signal.
4	**IRQ4 MODE SELECT (IRQ4MOD):** When IRQ4MOD = 0, IRQ4 is an active high tri-state drive signal. When IRQ4MOD = 1, IRQ4 is an active low open collector drive signal.
3	**IRQ3 MODE SELECT (IRQ3MOD):** When IRQ3MOD = 0, IRQ3 is an active high tri-state drive signal. When IRQ3MOD = 1, IRQ3 is an active low open collector drive signal.
2:0	**RESERVED**

4.1.6 FCFG1—FDC CONFIGURATION REGISTER

Index Address:	10h
Default Value:	0RRR RR01
Attribute:	Read/Write
Size:	8 bits

This register selects between a 2 and 4 floppy drive system, selects primary/secondary ISA address range for the FDC, and enables/disables the FDC. All bits in this register are read/write.


7
6
2
1
0
Bit
0*
R
0**
1**
Default
FDC Enable (R/W)
1=Enable
0=Disable
FDC Address Select (R/W)
1=Secondary FDC Address (370-377)
0=Primary FDC Address (3F0-3F7)
Reserved
Floppy Disk Drive Quantity (R/W)
1=Four Floppy Disk Drives (with External Decoder)
0=Two Floppy Disk Drives (without External Decoder)
290486-9

NOTES:
*Default shown is for SWMB, SWAI, and HWB hardware configuration modes. For HWE, the default is determined by hardware strapping options as described in Section 4.2, Hardware Configuration.
**Default shown is for SWMB and SWAI configuration modes. For HWB and HWE configuration modes, the default is determined by hardware strapping options as described in Section 4.2, Hardware Configuration.

Figure 9. FDC Configuration Register

Bit	Description
7	**FLOPPY DISK DRIVE QUANTITY (FDDQTY):** This bit selects between two and four floppy disk drive capability. When FDDQTY = 0, the 82091AA can control two floppy disk drives directly without an external decoder. When FDDQTY = 1, the 82091AA can control four floppy disk drives with an external decoder. When FDDQTY = 1, the PDEN feature in the powerdown command is disabled. For further details, see Appendix A, FDC Four Drive Support. This bit can be configured by hardware extended configuration (HWE) at powerup. For all other hardware configuration modes (SWMB, SWAI, and HWB), the floppy disk drive quantity is not configurable by hardware strapping options and defaults to 2 drives.
6:2	**RESERVED**
1	**FLOPPY DISK CONTROLLER ADDRESS SELECT (FADS):** When FADS = 0, the primary FDC address (3F0–3F7) is selected. When FADS = 1, the secondary FDC address (370–377) is selected. For SWMB and SWAI configuration modes, the default is 0 (primary address). For HWB and HWE hardware configuration modes, the default is determined by signal pin strapping options.
0	**FLOPPY DISK CONTROLLER ENABLE (FEN):** This bit enables/disables the FDC. When FEN = 1, the FDC is enabled. When FEN = 0, the FDC module is disabled. For SWMB and SWAI configuration modes, the default is 1 (enabled). For HWB and HWE hardware configuration modes, the default is determined by signal pin strapping options. Note that, when the FDC is disabled, IRQ6 and FDDREQ are tri-stated.

4.1.7 FCFG2—FDC POWER MANAGEMENT AND STATUS REGISTER

Index Address:	11h
Default Value:	RRRR 0000
Attribute:	Read/Write
Size:	8 bits

This register enables/disables FDC auto powerdown and can place the FDC into direct powerdown. The register also provides FDC idle status and FDC reset control.

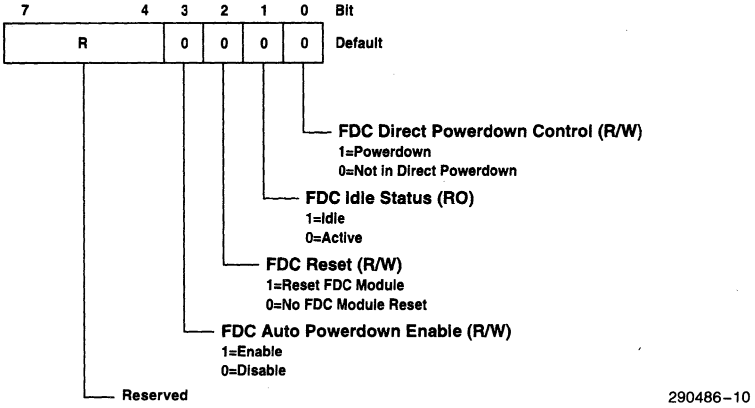


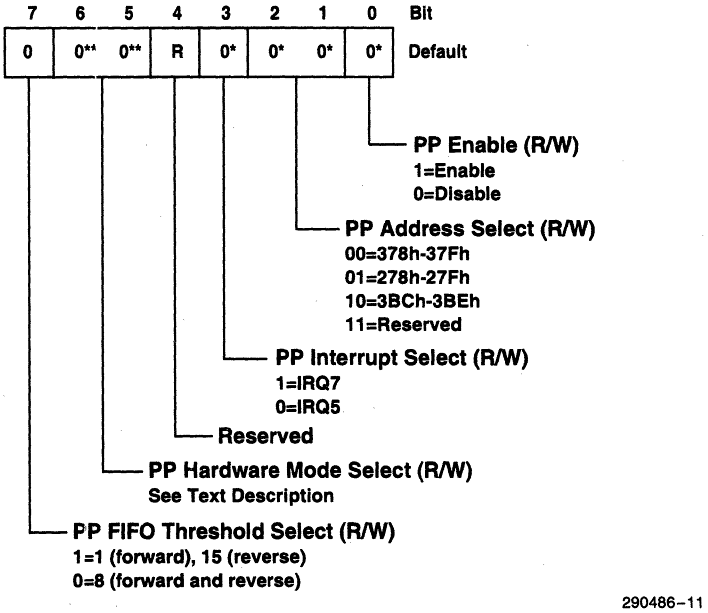
Figure 10. FDC Power Management and Status Register

Bit	Description
7:4	**RESERVED**
3	**FLOPPY DISK AUTO POWERDOWN ENABLE (FAPDN):** This bit is used to enable/disable auto powerdown for the FDC. When FAPDN = 1, the FDC will enter auto powerdown when the required conditions are met. When FAPDN = 0, FDC auto powerdown is disabled.
2	**FLOPPY DISK CONTROLLER RESET (FRESET):** FRESET is a reset for the FDC. When FRESET = 1, the FDC is reset (i.e., all programming and current state information is lost). FRESET = 1 has the same affect on the FDC as a hard reset (asserting the RSTDRV signal). When resetting the FDC via this configuration bit, the software must toggle this bit and ensure the reset active time (FRESET = 1) of 1.13 μs minimum is met.
1	**FLOPPY DISK CONTROLLER IDLE STATUS (FIDLE):** When the FDC is in the idle state, this bit is set to 1 by the 82091AA hardware. In the idle state the FDC's Main Status Register (MSR) = 80h, IRQ6 = inactive, and the head unload timer has expired. When the FDC exits its idle state, this bit is set to 0. This bit is read only.
0	**FLOPPY DISK CONTROLLER POWERDOWN (FDPDN):** When FDPDN is set to 1, the FDC is placed in direct powerdown. Once in powerdown the following procedure should be used to bring the FDC out of powerdown: • Write this bit low • Apply a hardware reset (via bit 2 of this register) or a software reset (via either bit 2 of the FDC's DOR or bit 7 of the FDC's DSR). **NOTE:** A hard reset via the RSTDRV pin also removes the FDC powerdown.

4.1.8 PCFG1—PARALLEL PORT CONFIGURATION REGISTER

Index Address:	20h
Default Value:	000R 0000
Attribute:	Read/Write
Size:	8 bits

The PCFG1 Register enables/disables the parallel port, selects the parallel port address, and selects the parallel port interrupt. This register also selects the hardware operation mode for the parallel port.



NOTES:

*Default shown is for SWMB and SWAI configuration modes. For HWB and HWE modes, the default is determined by hardware configuration options as described in Section 4.2, Hardware Configuration.

**Default shown is for SWMB, SWAI, and HWB configuration modes. For HWE mode, the default is determined by hardware configuration options as described in Section 4.2, Hardware Configuration.

Figure 11. Parallel Port Configuration Register

Bit	Description
7	**PARALLEL PORT FIFO THRESHOLD SELECT (PTHRSEL):** This bit controls the FIFO threshold and only affects parallel port operations when the parallel port is in ECP mode or ISA-Compatible FIFO mode. When PTHRSEL = 1, the FIFO threshhold is 1 in the forward direction and 15 in the reverse direction. When PTHRSEL = 0, the FIFO threshold is 8 in both directions. This bit can only be programmed when the parallel port is in ISA-Compatible or PS/2-Compatible mode. These modes can be selected via bits[6:5] of this register or the ECP Extended Control Register (ECR). **NOTE:** In the reverse direction, a threshold of 15/8 means that a request (DMA or Interrupt is enabled) is generated when 15/8 bytes are in the FIFO. In the forward direction, a threshold of 1/8 means that a request is generated when 1/8 byte locations are available.
6:5	**PARALLEL PORT HARDWARE MODE SELECT (PPHMOD):** This field selects the parallel port hardware mode. The ISA-Compatible mode is for compatibility and nibble mode peripheral interface protocols. The PS/2-Compatible mode is for the byte mode peripheral interface protocol. The EPP and ECP modes are for the EPP and ECP mode peripheral interface protocols, respectively. This field can be configured by strapping options at powerup for hardware extended configuration (HWE) mode only. For all other hardware configuration modes (SWMB, SWAI, and HWB), the default is 00 (ISA-Compatible). **Bits [6:5]** — **Read** — **Write** 0 0 — ISA-Compatible — ISA-Compatible(1) 0 1 — PS/2-Compatible — PS/2-Compatible(1) 1 0 — EPP — EPP(1, 3) 1 1 — ECP(2) — Reserved; do not write(2) **NOTES:** 1. ISA-Compatible, PS/2-Compatible, and EPP modes are selected via this field or hardware configuration. In addition, ISA-Compatible and PS/2-Compatible modes can be selected via the ECP Extended Control Register (ECR). When the ECR is programmed for one of these two modes (ECR[7:5] = 000, 001), this field is updated to match the selected mode. 2. ECP Mode can not be entered by programming this field. ECP Mode can only be selected through the ECR. When the ECR is programmed for ECP mode, the 82091AA sets this field to 11. 3. Parallel port interface signals controlled by the PCON Register (SELECTIN#, INIT#, AUTOFD#, and STROBE#) should be negated before entering EPP mode.
4	**RESERVED**
3	**PARALLEL PORT IRQ SELECT (PIRQSEL):** When PIRQSEL = 1, IRQ7 is selected as the parallel port interrupt. When PIRQSEL = 0, IRQ5 is selected as the parallel port interrupt. This field can be configured by strapping options at powerup for HWB and HWE modes only. For all other hardware configuration modes (SWMB and SWAI), the default is 0 (IRQ5).

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>2:1</td><td><b>PARALLEL PORT ADDRESS SELECT (PADS):</b> This field selects the address for the parallel port as follows:
<table>
<tr><th>Bits[2:1]</th><th>Address</th><th>Parallel Port Hardware Mode</th></tr>
<tr><td>0 0</td><td>378–37F</td><td>All</td></tr>
<tr><td>0 1</td><td>278–27F</td><td>All</td></tr>
<tr><td>1 0</td><td>3BC–3BE</td><td>All except EPP</td></tr>
<tr><td>1 1</td><td>Reserved</td><td>None, do not write</td></tr>
</table>
This field can be configured by strapping options at powerup for HWB and HWE modes only. For all other hardware configuration modes (SWMB and SWAI), the default is 00 (378h–37Fh). Note that the SWMB and SWAI default settings for PIRQSEL (bit 3) and PADS (bits[2,1]) do not match a standard PC/AT* combination for address assignment and interrupt setting. However, for SWMB and SWAI, the parallel port defaults to a disabled condition and this register must be programmed to enable the parallel port (i.e., bit 0 set to 1). At this time, the selections for interrupt and address assignments should be made.</td></tr>
<tr><td>0</td><td><b>PARALLEL PORT ENABLE (PEN):</b> When PEN = 0, the parallel port is disabled. When PEN = 1, the parallel port is enabled. This bit can be configured by hardware strapping options at powerup for HWB and HWE modes only. For all other hardware configuration modes (SWMB and SWAI), the default is 0 (disabled). Note that when the parallel port is disabled, IRQ[7,5] and PPDREQ are tri-stated.</td></tr>
</table>

4.1.9 PCFG2—PARALLEL PORT POWER MANAGEMENT AND STATUS REGISTER

Index Address:	21h
Default Value:	RR0R 0000
Attribute:	Read/Write
Size:	8 bits

This register enables/disables parallel port auto powerdown and can place the parallel port into a powerdown mode directly. The register also provides parallel port idle status, resets the parallel port, and reports FIFO underrun or overrun errors.

*Other brands and names are the property of their respective owners.

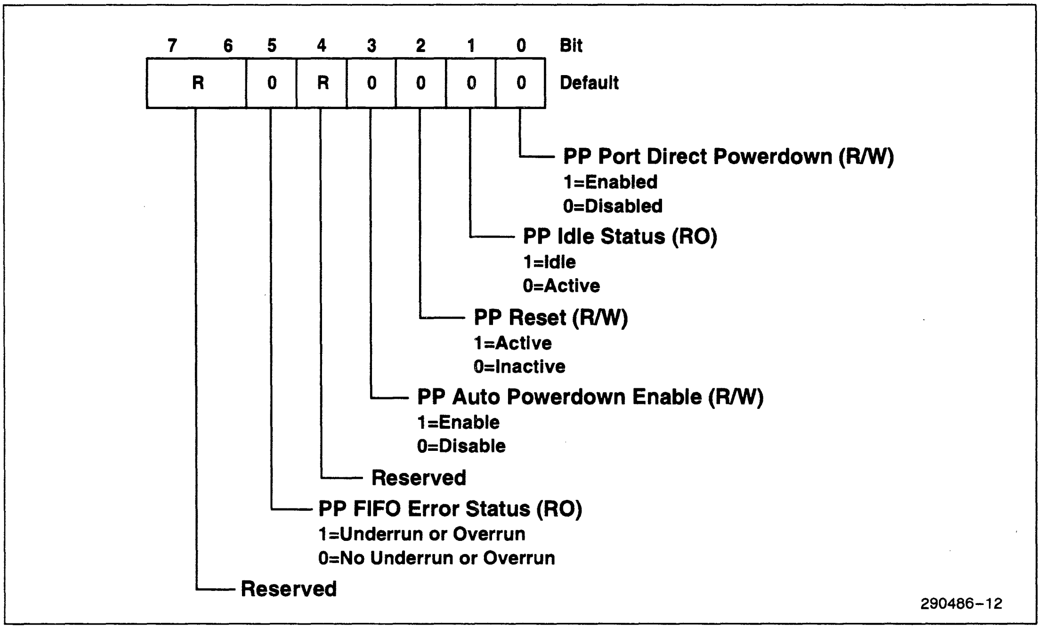


Figure 12. Parallel Port Power Management and Status Register

Bit	Description
7:6	**RESERVED**
5	**PARALLEL PORT FIFO ERROR STATUS (PFERR):** When PFERR = 1, a FIFO underrun or overrun condition has occurred. This bit is read only. Setting PRESET to 1 clears this bit to 0.
4	**RESERVED**
3	**PARALLEL PORT AUTO POWERDOWN ENABLE (PAPDN):** When PAPDN = 1, the parallel port can enter auto powerdown if the required auto powerdown conditions are met. When PAPDN = 0, auto powerdown is disabled.
2	**PARALLEL PORT RESET (PRESET):** When PRESET is set to 1, the parallel port is reset (i.e., all programming and current state information is lost). This is the same state the module would be in after a hard reset (RSTDRV asserted) to the 82091AA. When resetting the parallel port via this configuration bit, the software must toggle this bit and ensure the reset active time (PRESET = 1) of 1.13 μs minimum is met.
1	**PARALLEL PORT IDLE STATUS (PIDLE):** This bit reflects the idle state of the parallel port. When the parallel port is in an idle state (i.e., when the same conditions are met that apply to entering auto powerdown) the 82091AA sets this bit to 1. The parallel port idle state is defined as the FIFO empty and no activity on the parallel port interface. This bit is read only.
0	**PARALLEL PORT DIRECT POWERDOWN (PDPDN):** When PDPDN is set to 1, the parallel port enters direct powerdown. When PDPDN is set to 0, the parallel port is not in direct powerdown. Note that a parallel port module reset (PRESET bit in this register) also brings the parallel port out of the direct powerdown state.

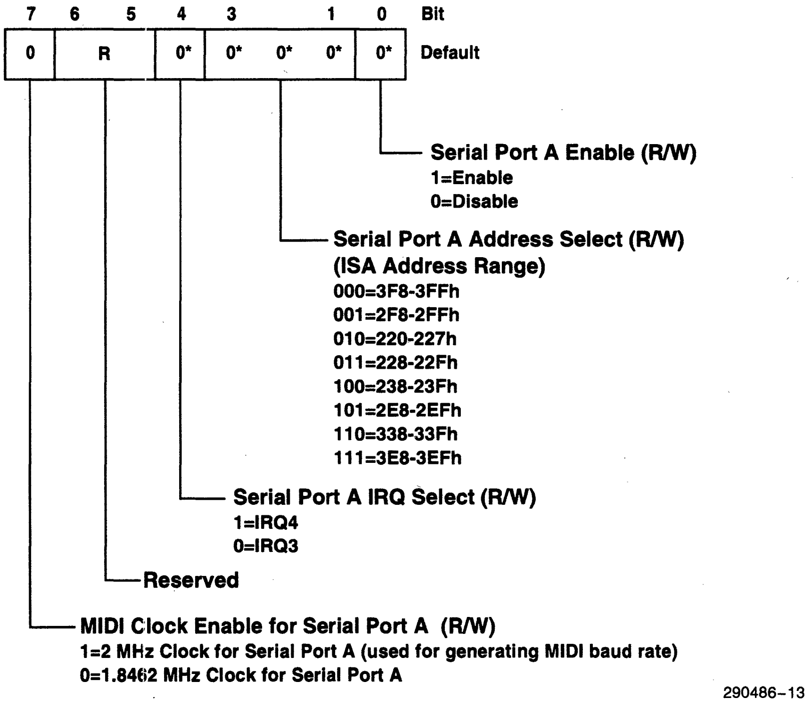
4.1.10 SACFG1—SERIAL PORT A CONFIGURATION REGISTER

Index Address:	30h
Default Value:	0RR0 0000
Attribute:	Read/Write
Size:	8 bits

The SACFG1 register enables/disables Serial Port A, selects the Serial Port A address range, and selects between IRQ3 and IRQ4 as the Serial Port A interrupt. This register also selects the appropriate clock frequency for use with MIDI.

NOTES:

1. Through programming of this register and the SBCFG1 Register, the 82091AA permits serial ports A and B to be configured for the same interrupt assignment. However, software must take care in responding to interrupts correctly.
2. It is possible to enable and assign both serial ports to the same address through software. In this configuration, the 82091AA disables serial port B, but does not set serial port B into it's powerdown condition. Although this is a safe configuration for the 82091AA, it is not power conservative and is not recommended.



7 6 5 4 3 1 0 Bit
0 R 0* 0* 0* 0* 0* Default
Serial Port A Enable (R/W)
1=Enable
0=Disable
Serial Port A Address Select (R/W)
(ISA Address Range)
000=3F8-3FFh
001=2F8-2FFh
010=220-227h
011=228-22Fh
100=238-23Fh
101=2E8-2EFh
110=338-33Fh
111=3E8-3EFh
Serial Port A IRQ Select (R/W)
1=IRQ4
0=IRQ3
Reserved
MIDI Clock Enable for Serial Port A (R/W)
1=2 MHz Clock for Serial Port A (used for generating MIDI baud rate)
0=1.8462 MHz Clock for Serial Port A
290486-13

NOTE:
*Default shown is for SWMB and SWAI hardware configuration modes. For HWB and HWE modes, the default is determined by hardware strapping options as described in Section 4.2, Hardware Configuration.

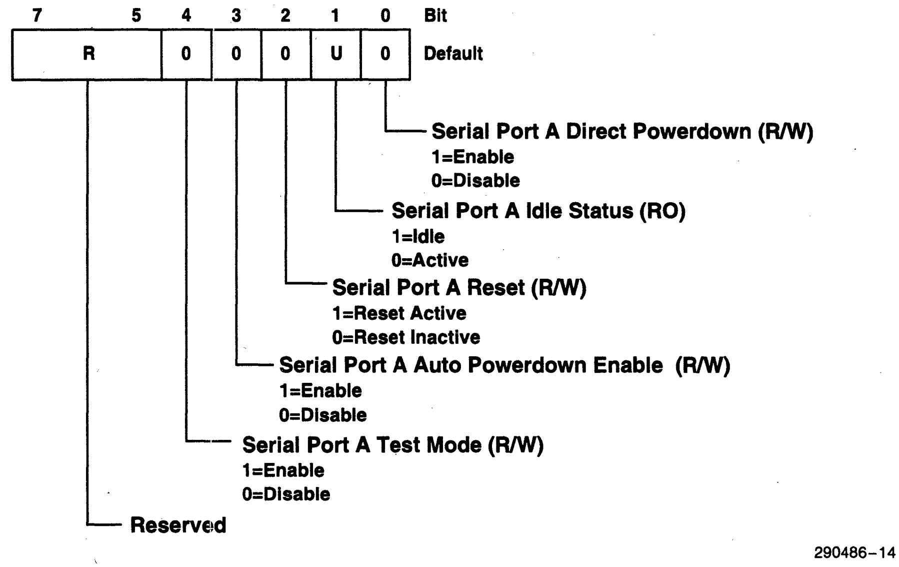
Figure 13. Serial Port A Configuration Register

Bit	Description
7	**MIDI CLOCK FOR SERIAL PORT A ENABLE (SAMIDI):** When SAMIDI = 1, the clock into Serial Port A is changed from 1.8462 MHz–2 MHz. The 2 MHz clock is needed to generate the MIDI baud rate. When SAMIDI = 0, the clock frequency is 1.8462 MHz.
6:5	**RESERVED**
4	**SERIAL PORT A IRQ SELECT (SAIRQSEL):** When SAIRQSEL = 0, IRQ3 is selected for the Serial Port A interrupt. When SAIRQSEL = 1, IRQ4 is selected for the Serial Port A interrupt. This bit can be configured by strapping options at powerup for HWB and HWE modes only. For SWMB and SWAI hardware configuration modes, the default is 0 (IRQ3). Note that, while the default address and IRQ assignments for SWMB and SWAI modes are the same for both serial ports, the serial ports are disabled and programming of this register is required for operation.
3:1	**SERIAL PORT A ADDRESS SELECT (SAADS):** This field selects the ISA address range for Serial Port A as follows: **Bits[3:1] ISA Address Range** 0 0 0 3F8–3FFh 0 0 1 2F8–2FFh 0 1 0 220–227h 0 1 1 228–22Fh 1 0 0 238–23Fh 1 0 1 2E8–2EFh 1 1 0 338–33Fh 1 1 1 3E8–3EFh This field can be configured by strapping options at powerup for HWB and HWE modes only. For SWMB and SWAI hardware configuration modes, the default is 000 (3F8–3FFh). Note that, while the default address and IRQ assignments for SWMB and SWAI modes are the same for both serial ports, the serial ports are disabled and programming of this register is required for operation.
0	**SERIAL PORT A ENABLE (SAEN):** When SAEN = 1, Serial Port A is enabled. When SAEN = 0, Serial Port A is disabled. This bit can be configured by strapping options at powerup for HWB and HWE modes only. For SWMB and SWAI hardware configuration modes, the default is 0 (disabled).

4.1.11 SACFG2—SERIAL PORT A POWER MANAGEMENT AND STATUS REGISTER

Index Address:	31h
Default Value:	RRR0 00U0
Attribute:	Read/Write
Size:	8 bits

This register enables/disables the Serial Port A module auto powerdown and can place the module into a direct powerdown mode. The register also provides Serial Port A idle status, resets the Serial Port A module, and places Serial Port A into test mode.


7
5
4
3
2
1
0
Bit
R
0
0
0
U
0
Default
Serial Port A Direct Powerdown (R/W)
1=Enable
0=Disable
Serial Port A Idle Status (RO)
1=Idle
0=Active
Serial Port A Reset (R/W)
1=Reset Active
0=Reset Inactive
Serial Port A Auto Powerdown Enable (R/W)
1=Enable
0=Disable
Serial Port A Test Mode (R/W)
1=Enable
0=Disable
Reserved
290486-14

NOTE:
U = Undefined

Figure 14. Serial Port A Power Management and Status Register

Bit	Description
7:5	**RESERVED**
4	**SERIAL PORT A TEST MODE (SATEST):** The serial port test mode provides user access to the output of the baud out generator. When SATEST = 1 (and the DLAB bit is 1 in the LCR), the Serial Port A test mode is enabled and the baud rate clock is output on the SOUTA pin (Figure 15). When SATEST = 0, the Serial Port A test mode is disabled.

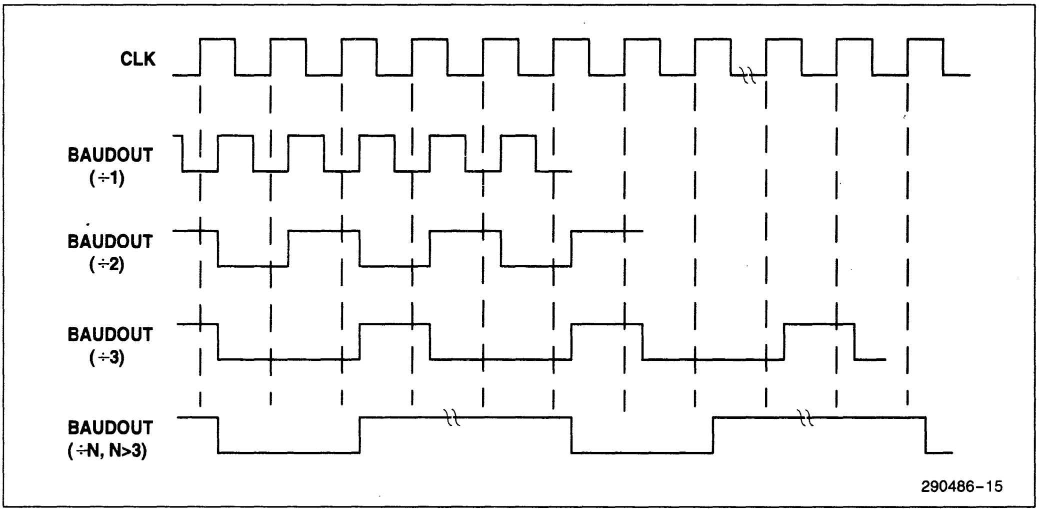


Figure 15. Test Mode Output (SOUTA and SOUTB)

Bit	Description
3	**SERIAL PORT A AUTO POWERDOWN ENABLE (SAAPDN):** This bit enables/disables auto powerdown. When SAAPDN = 1, Serial Port A can enter auto powerdown if the required conditions are met. The required conditions are that the transmit and receive FIFOs are empty and the timeout counter has expired. When SAAPDN = 0, auto powerdown is disabled.
2	**SERIAL PORT A RESET (SARESET):** When SARESET = 1, the Serial Port A module is reset (i.e. all programming and current state information is lost). This is the same state the module would be in after a hard reset (RSTDRV asserted). When resetting the serial port via this configuration bit, the software must toggle this bit and ensure the reset active time (SARESET = 1) of 1.13 μs minimum is met.
1	**SERIAL PORT A IDLE STATUS (SAIDLE):** When Serial Port A is in an idle state the 82091AA sets this bit to 1. Serial Port A is in the idle state when the transmit and receive FIFOs are empty and the timeout counter has expired. Note that these are the same conditions that apply to entering auto powerdown. When serial port A is not in an idle state, the 82091AA sets this bit to 0. Direct powerdown does not affect this bit and in auto powerdown SAIDLE is only set to a 1 if the receive and transmit FIFOs are empty. This bit is read only. During a hard reset (RSTDRV asserted), The 82091AA sets SAIDLE to 0. However, because the serial port is typically initialized by software before the idle conditions are met, the default state is shown as undefined.
0	**SERIAL PORT A DIRECT POWERDOWN (SADPDN):** When SADPDN = 1, Serial Port A is placed in direct powerdown mode. Setting this bit to 0 brings Serial Port A out of direct powerdown mode. Setting bit 2 (SARESET) of this register to 1 will also bring Serial Port A out of the direct powerdown mode. **NOTE:** Direct powerdown resets the receiver and transmitter portions of the serial port including the receive and transmit FIFOs. To ensure that the resetting of the FIFOs does not cause data loss, the SAIDLE bit should be 1 before placing the serial port into direct powerdown.

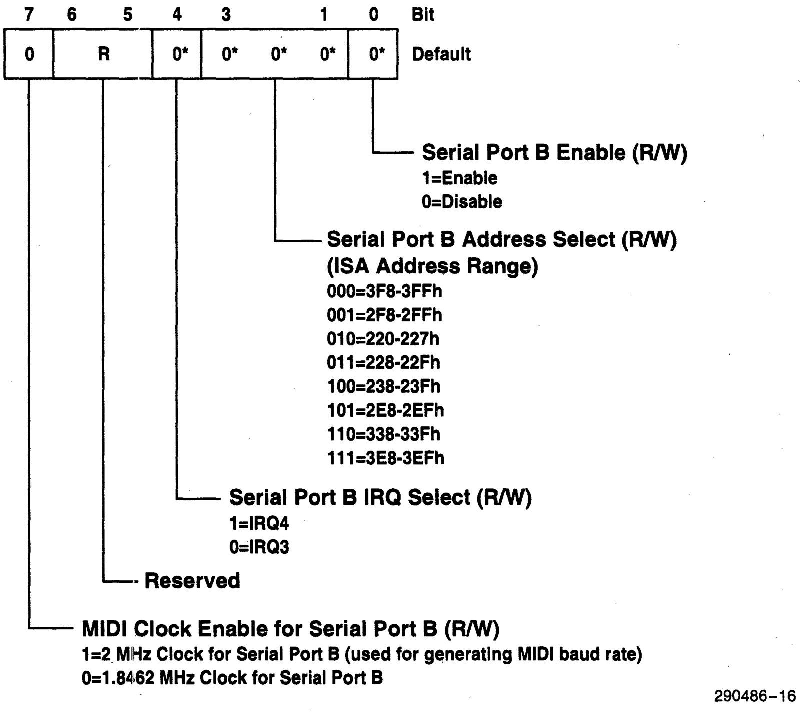
4.1.12 SBCFG1—SERIAL PORT B CONFIGURATION REGISTER

Index Address:	40h
Default Value:	0RR0 0000
Attribute:	Read/Write
Size:	8 bits

The SBCFG1 register enables/disables Serial Port B, selects the Serial Port B address range, and selects between IRQ3 and IRQ4 as the Serial Port B interrupt. This register also selects the appropriate clock frequency for use with MIDI.

NOTES:

1. Through programming of this register and the SBCFG1 Register, the 82091AA permits serial ports A and B to be configured for the same interrupt assignment. However, software must take care in responding to interrupts correctly.
2. It is possible to enable and assign both serial ports to the same address through software. In this configuration, the 82091AA disables serial port B, but does not set serial port B into it's powerdown condition. Although this is a safe configuration for the 82091AA, it is not power conservative and is not recommended.


7
6
5
4
3
1
0
Bit
0
R
0*
0*
0*
0*
0*
Default
Serial Port B Enable (R/W)
1=Enable
0=Disable
Serial Port B Address Select (R/W)
(ISA Address Range)
000=3F8-3FFh
001=2F8-2FFh
010=220-227h
011=228-22Fh
100=238-23Fh
101=2E8-2EFh
110=338-33Fh
111=3E8-3EFh
Serial Port B IRQ Select (R/W)
1=IRQ4
0=IRQ3
Reserved
MIDI Clock Enable for Serial Port B (R/W)
1=2 MHz Clock for Serial Port B (used for generating MIDI baud rate)
0=1.8462 MHz Clock for Serial Port B
290486-16

NOTE:
*Default shown is for SWMB and SWAI hardware configuration modes. For HWB and HWE modes, the default is determined by hardware strapping options as described in Section 4.2, Hardware Configuration.

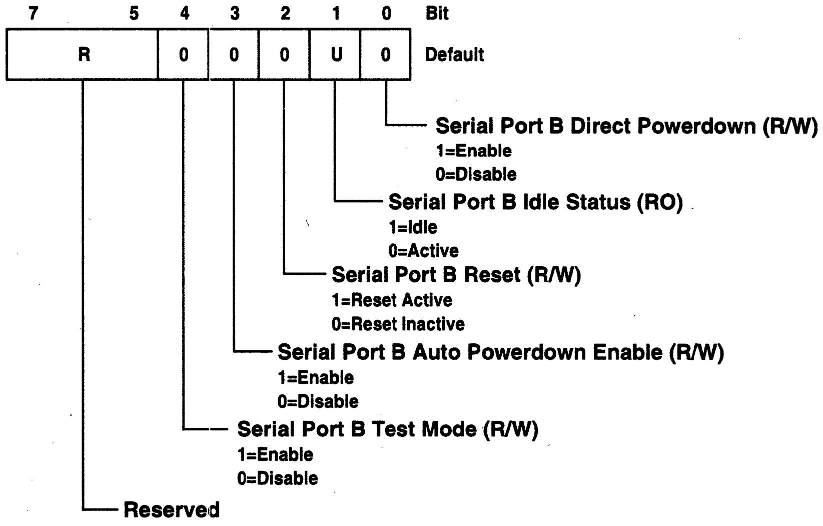
Figure 16. Serial Port B Configuration Register

Bit	Description
7	**MIDI CLOCK FOR SERIAL PORT B ENABLE (SBMIDI):** When SBMIDI = 1, the clock into Serial Port B is changed from 1.8462 MHz to 2 MHz. The 2 MHz clock is needed to generate the MIDI baud rate. When SBMIDI = 0, the clock frequency is 1.8462 MHz. The default value is 0.
6:4	**RESERVED**
4	**SERIAL PORT B IRQ SELECT (SBIRQSEL):** When SBIRQSEL = 0, IRQ3 is selected for the Serial Port B interrupt. When SBIRQSEL = 1, IRQ4 is selected for the Serial Port B interrupt. The default value is 0. This bit can be configured by strapping options at powerup for HWB and HWE modes only. For SWMB and SWAI configuration modes, the default is 0 (IRQ3). Note that, while the default address and IRQ assignments for SWMB and SWAI modes are the same for both serial ports, the serial ports are disabled and programming of this register is required for operation.
3:1	**SERIAL PORT B ADDRESS SELECT (SBADS):** This field selects the ISA address range for Serial Port B as follows: **Bits[3:1]** **ISA Address Range** 0 0 0 3F8-3FFh 0 0 1 2F8-2FFh 0 1 0 220-227h 0 1 1 228-22Fh 1 0 0 238-23Fh 1 0 1 2E8-2EFh 1 1 0 338-33Fh 1 1 1 3E8-3EFh This field can be configured by strapping options at powerup for HWB and HWE modes only. For SWMB and SWAI configuration modes, the default is 000 (3F8-3FFh). Note that, while the default address and IRQ assignments for SWMB and SWAI modes are the same for both serial ports, the serial ports are disabled and programming of this register is required for operation.
0	**SERIAL PORT B ENABLE (SBEN):** When SBEN = 1, Serial Port B is enabled. When SAEN = 0, Serial Port B is disabled. This bit can be configured by strapping options at powerup for HWB and HWE modes only. For SWMB and SWAI configuration modes, the default is 0 (disabled).

4.1.13 SBCFG2—SERIAL PORT B POWER MANAGEMENT AND STATUS REGISTER

Index Address:	41h
Default Value:	RRR0 00U0
Attribute:	Read/Write
Size:	8 bits

This register enables/disables the Serial Port B module auto powerdown and can place the module into a powerdown mode directly. The register also provides Serial Port B idle status, resets the Serial Port B module, and enables/disables Serial Port B test mode.


7
5
4
3
2
1
0
Bit
R
0
0
0
U
0
Default
Serial Port B Direct Powerdown (R/W)
1=Enable
0=Disable
Serial Port B Idle Status (RO)
1=Idle
0=Active
Serial Port B Reset (R/W)
1=Reset Active
0=Reset Inactive
Serial Port B Auto Powerdown Enable (R/W)
1=Enable
0=Disable
Serial Port B Test Mode (R/W)
1=Enable
0=Disable
Reserved

290486–17

NOTE:
U = Undefined

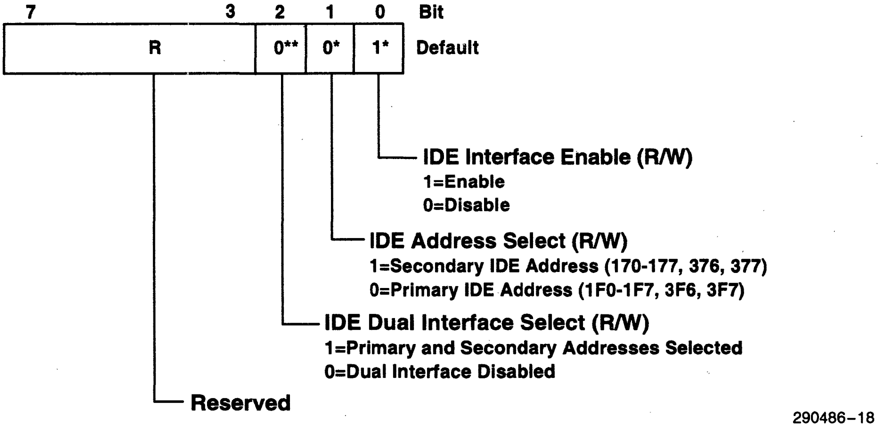
Figure 17. Serial Port B Power Management and Status Register

Bit	Description
7:5	**RESERVED**
4	**SERIAL PORT B TEST MODE (SBTEST):** The serial port test mode provides user access to the output of the baud out generator. When SBTEST = 1 (and the DLAB bit is 1 in the LCR), the Serial Port B test mode is enabled and the baud rate clock is output on the SOUTB pin (Figure 15). When SBTEST = 0, the Serial Port B test mode is disabled.
3	**SERIAL PORT B AUTO POWERDOWN ENABLE (SBAPDN):** This bit enables/disables auto powerdown. When SBAPDN = 1, Serial Port B can enter auto powerdown if the required conditions are met. The required conditions are that the transmit and receive FIFOs are empty and the timeout counter has expired. When SBAPDN = 0, auto powerdown is disabled.
2	**SERIAL PORT B RESET (SBRESET):** When SBRESET = 1, Serial Port B is reset (i.e., all programming and current state information is lost). This is the same state the module would be in after a hard reset (RSTDRV asserted). When resetting the serial port via this configuration bit, the software must toggle this bit and ensure the reset active time (SBRESET = 1) of 1.13 μs minimum is met.
1	**SERIAL PORT B IDLE STATUS (SBIDLE):** When Serial Port B is in an idle state the 82091AA sets this bit to 1. Serial Port B is in the idle state when the transmit and receive FIFOs are empty and the timeout counter has expired. Note that these are the same conditions that apply to entering auto powerdown. When serial port B is not in an idle state, the 82091AA sets this bit to 0. Direct powerdown does not affect this bit and in auto powerdown, this bit is only set to a 1 if the receive and transmit FIFOs are empty. This bit is read only. During a hard reset (RSTDRV asserted), the 82091AA sets this bit to 0. However, because the serial port is typically initialized by software before the idle conditions are met, the defaullt state is shown as undefined.
0	**SERIAL PORT B DIRECT POWERDOWN (SBDPDN):** When SBDPDN = 1, Serial Port B is placed in powerdown mode. Setting this bit to 0 brings the module out of direct powerdown mode. Setting bit 2 (SBRESET) of this register to 1 will also bring Serial Port B out of the direct powerdown mode. **NOTE:** Direct powerdown resets the receiver and transmitter portions of the serial port including the receive and transmit FIFOs. To ensure that the resetting of the FIFOs does not cause data loss, the SBIDLE bit should be 1 before placing the serial port into direct powerdown.

4.1.14 IDECFG—IDE CONFIGURATION REGISTER

Index Address:	50h
Default Value:	RRRR R001
Attribute:	Read/Write
Size:	8 bits

The IDECFG Register sets up the 82091AA IDE interface. This register enables the IDE interface and selects the address for accessing the IDE.


7
3
2
1
0
Bit
R
0**
0*
1*
Default
IDE Interface Enable (R/W)
1=Enable
0=Disable
IDE Address Select (R/W)
1=Secondary IDE Address (170-177, 376, 377)
0=Primary IDE Address (1F0-1F7, 3F6, 3F7)
IDE Dual Interface Select (R/W)
1=Primary and Secondary Addresses Selected
0=Dual Interface Disabled
Reserved
290486-18

NOTES:

* Default shown is for SWMB and SWAI configuration modes. For HWB and HWE hardware configuration modes, the default is determined by hardware strapping options as described in Section 4.2, Hardware Configuration.

** Not hardware configurable.

Figure 18. IDE Configuration Register

Bit	Description
7:3	**RESERVED**
2	**IDE DUAL SELECT (IDUAL):** When IDUAL = 0, the IDE address selection is determined by the IADS bit. When IDUAL = 1, both the primary and secondary IDE addresses are selected and the setting of the IADS bit does not affect IDE address selection.
1	**IDE ADDRESS SELECT (IADS):** When IADS = 0, the primary IDE address is selected (1F0h–1F7h, 3F6h, 3F7h). When IADS = 1, the secondary IDE address is selected (1F0h–1F7h, 376h, 377h). For all hardware configuration modes (SWMB, SWAI, HWB, and HWE), the default is determined by signal pin strapping options.
0	**IDE INTERFACE ENABLE (IEN):** When IEN = 0, the IDE interface is disabled (i.e., the IDE chip selects (IDECS[1:0]), DEN#, and HEN# are negated (remain inactive) for accesses to the IDE primary and secondary addresses). When IEN = 1, the IDE interface is enabled. For all hardware configuration modes (SWMB, SWAI, HWB, and HWE), the default is determined by signal pin strapping options.

4.2 Hardware Configuration

Hardware configuration provides a mechanism for configuring certain 82091AA operations at powerup. Four hardware configuration modes provide different levels of configuration depending on the type of application and the degree of hardware/software configuration desired. The hardware configuration modes are:

- Software Motherboard (SWMB)
- Software Add-In (SWAI)
- Hardware Extended (HWE)
- Hardware Basic (HWB)

These modes support a variety of system implementations. For example, with Hardware Basic (HWB) and Hardware Extended (HWE) modes, an extensive set of 82091AA configuration options are available for setting up the 82091AA at powerup. This permits the 82091AA to be used in systems without 82091AA software drivers. For many of these systems, access to the 82091AA configuration registers may not be necessary. As such, access to these registers can be disabled via hardware configuration. This option could be used to prevent software from inadvertently re-configuring the 82091AA.

NOTE:

If the 82091AA is configured in HWB or HWE configuration mode at powerup, and reconfiguration with software is desired, the 82091AA configuration mode must first be changed to SWAI configuration mode by writing the AIPCFG1 register. The 82091AA can then remain in SWAI configuration mode to accomodate software programmable configuration changes as desired.

Software Motherboard (SWMB) and Software Add-In (SWAI) modes provide a minimum hardware configuration in systems where software/firmware drivers are used for configuration. Because access to the 82091AA configuration registers after powerup/hardware configuration is needed, the SWMB and SWAI modes do not provide disabling access to these registers (i.e., the strapping of the HEN# signal has no effect).

The desired hardware configuration mode and options within the mode are selected by strapping certain 82091AA signal pins at powerup. These signal pins are sampled when the 82091AA receives a hard reset (via RSTDRV). This section describes how to select the configuration mode and options within the mode. The section also provides example hardware connection diagrams for the different modes.

4.2.1 SELECTING THE HARDWARE CONFIGURATION MODE

During powerup or a hard reset, four signal pins (DEN#, PPDIR/GCS#, DTRA, and HEN#) select the hardware configuration mode, I/O address assignment for the 82091AA configuration registers, and whether software access to these configuration registers is permitted. The following mnemonics and signal pins are assigned for these functions:

CFGMOD[1,0] **Hardware Configuration Mode.** The 82091AA samples the CFGMOD0 (DEN#) and CFGMOD1 (PPDIR/GCS#) signal pins to select one of the four hardware configuration modes as shown in Table 6.

CFGADS **82091AA Configuration Register Address Assignment.** The 82091AA samples the DTRA# signal (CFGADS function) to determine the address assignment of the 82091AA configuration registers as shown in Table 6. CFGADS works in conjunction with CFGDIS. Note that the 82091AA configuration register address assignment for Hardware Basic mode is not selectable.

CFGDIS **82091AA Configuration Register Disable.** The 82091AA samples CFGDIS (HEN# signal) to enable/disable access to the 82091AA configuration registers as shown in Table 6. Note that CFGDIS only affects the HWE and HWB modes.

NOTE:

For Extended Hardware Configuration, the time immediately following the RSTDRV pulse is required to complete the configuration time. If IORC#/IOWC# are asserted during this time, IOCHRDY will be negated (wait-states inserted) until the 82091AA configuration time expires.

Table 6. AIP Configuration Mode Register Address Assignment

CFGDIS (HEN#)	CFGMOD1 (PPDIR)	CFGMOD0 (DEN#)	CFGADS (DTRA#)	Configuration Mode	Configuration Register ISA Address (INDEX/TARGET)
X	0	0	0	SWMB	22h/23h
X	0	0	1	SWMB	24h/25h
X	0	1	0	SWAI	26Eh/26Fh
X	0	1	1	SWAI	398h/399h
0	1	0	0	HWE	26Eh/26Fh
0	1	0	1	HWE	398h/399h
1	1	0	X	HWE	Access Disabled
0	1	1	n/a	HWB	398h/399h
1	1	1	n/a	HWB	Access Disabled

4.2.2 SELECTING HARDWARE CONFIGURATION MODE OPTIONS

Within each hardware configuration mode, a number of options are available. For the HWB and HWE hardware configuration modes, the user can enable/disable the floppy disk controller and the IDE interface via the IDE chip select pins (see Table 7). If enabled, these signal pins also select the address assignment. For SWMB and SWAI configuration modes, these signal pins have no effect.

Table 7. FDC and IDE Enable/Disable

DDCFG1 (IDECS1#)	DDCFG0 (IDECS0#)	Floppy Disk Controller	IDE
0	0	Disable	Disable
0	1	Enabled (3F6–3F7h; Primary)	Disable
1	0	Enabled (370–377h; Secondary)	Enabled (170–177h; Secondary)
1	1	Enabled (3F6–3F7h; Primary)	Enabled (1F0–1F7h; Primary)

The 82091AA provides additional hardware configuration options through the SOUTA, SOUTB, RTSA#, RTSB#, DTRA#, and DTRB# signal pins as shown in Table 8. In the case of the Hardware Extended Mode, the 82091AA samples the signal pins at two different times (once for HWEa options and again for HWEb options). The timing for signal sampling is discussed in Section 4.2.3, Hardware Configuration Timing Relationships. The options provide configuration of the serial ports, floppy disk controller, parallel port, IDE interface, 82091AA operating power supply voltage, 82091AA clock frequency, and address assignment for the 82091AA configuration registers. Table 8 provides a matrix of the options available for each hardware configuration mode. The configuration options are selected as shown in Table 8 through Table 14.

Note that for the SWAI and SWMB modes, the selection of the operating frequency (CLKSEL), power supply voltage level (VSEL), and 82091AA configuration register address assignment (CFGADS) are the only hardware configuration options (Table 8). In these modes, software/firmware provides the remainder of the 82091AA configuration by programming the 82091AA configuration registers (see Section 4.1, Configuration Registers). For the SWAI and SWMB modes, the 82091AA modules are placed in the following states after powerup or a hard reset:

- Serial ports disabled
- Parallel port disabled
- FDC enabled for two drives (primary address)
- IDE enabled (primary address)

Table 8. Hardware Configuration Mode Option Matrix

Signal Name	Basic Hardware Configuration	Extended Hardware Configuration		Software Add-In Configuration	Software MotherBoard Configuration
	HWB	HWEa	HWEb	SWAI	SWMB
SOUTA	SPCFG0	CLKSEL[3]	SPCFG0	CLKSEL[3]	CLKSEL[3]
SOUTB	SPCFG1	PPMOD0	SPCFG1	—	—
RTSA#	SPCFG2	PPMOD1	SPCFG2	—	—
RTSB#	SPCFG3	FDDQTY	SPCFG3	—	—
DTRA#	PPCFG0	CFGADS	PPCFG0	CFGADS	CDGADS
DTRB#	PPCFG1	VSEL	PPCFG1	VSEL	VSEL

NOTES:

1. HWEa and HWEb reference the switching banks shown in Figure 22.
2. The following mnemonics are used in the table: SPCFGx = serial port configuration, PPCFGx = parallel port configuration, CLKSEL = clock select, PPMODx = parallel port hardware mode, FDDQTY = floppy disk drive quantity, VSEL = power supply voltage select, CFGADS = 82091AA configuration register address assignment select.
3. Always tie this signal low with a 10K resistor.

Table 9. Serial Port Address and Interrupt Assignments

SPCFG3 (RTSB#)	SPCFG2 (RTSA#)	SPCFG1 (SOUTB)	SPCFG0 (SOUTA)	Serial Port B		Serial Port A	
				Address Assignment	Interrupt Assignment	Address Assignment	Interrupt Assignment
0	0	0	0	Disable	—	Disable	—
0	0	0	1	Disable	—	3F8–3FFh	IRQ4
0	0	1	0	Disable	—	2F8–2FFh	IRQ3
0	0	1	1	Disable	—	3E8–3EFh	IRQ4
0	1	0	0	3F8–3FFh	IRQ4	Disable	—
0	1	0	1	3E8–3EFh	IRQ4	Disable	—
0	1	1	0	3F8–3FFh	IRQ4	2F8–2FFh	IRQ3
0	1	1	1	3F8–3FFh	IRQ4(1)	3E8–3EFh	IRQ4(1)
1	0	0	0	2F8–2FFh	IRQ3	Disable	—
1	0	0	1	2F8–2FFh	IRQ3	3F8–3FFh	IRQ4
1	0	1	0	Disable	—	2E8–2EFh	IRQ3
1	0	1	1	2F8–2FFh	IRQ3	3E8–3EFh	IRQ4
1	1	0	0	2E8–2EFh	IRQ3	Disable	—
1	1	0	1	2E8–2EFh	IRQ3	3F8–3FFh	IRQ4
1	1	1	0	2E8–2EFh	IRQ3(1)	2F8–2FFh	IRQ3(1)
1	1	1	1	2E8–2EFh	IRQ3	3E8–3EFh	IRQ4

NOTE:

1. In this configuration, the two serial ports share the same interrupt line. Responding correctly to interrupts generated in this configuration is the exclusive responsibility of software.

Table 10. Parallel Port Address and Interrupt Assignments

PPCFG1 (DTRB#)	PPCFG0 (DTRA#)	Parallel Port Address Assignment	Parallel Port Interrupt Assignment
0	0	Disable	—
0	1	378–37Fh	IRQ7
1	0	278–27Fh	IRQ5
1	1	3BC–3BFh	IRQ7

Table 11. Parallel Port Hardware Mode Select

PPMOD1 (RTSA#)	PPMOD0 (SOUTB)	Mode
0	0	ISA-Compatible
0	1	PS/2-Compatible
1	0	EPP
1	1	Reserved

NOTES:
1. PPMODx hardware configuration is effective in HWE mode only.
2. ECP mode is not selectable via hardware configuration.
3. For EPP mode, address assignment must be either 278h or 378h.

Table 12. AIP Clock Select

CLKSEL (SOUTA)	
0	24 MHz

NOTE:
Always tie this low.

Table 13. AIP Power Supply Voltage

VSEL (DTRB#)	Power Supply Voltage
0	5.0V Operation
1	3.3V Operation

NOTES:
1. VSEL hardware configuration is not available in HWB mode only.
2. To operate the 82091AA and all of the interfaces at 5V or 3.3V, both V_{CC} and V_{CCF} are connected to 5V or 3.3V power supplies, respectively. However, in the mixed mode, hardware configuration (V_{SEL}) is set to 3.3V, V_{CC} is connected to 3.3V, and V_{CCF} connected to 5V.
3. 3.3V operation is available only in the 82091AA.

Table 14. Floppy Drive Quantity Select

FDDQTY (RTSB#)	Number of Supported Floppy Drives
0	2 Floppy Drives
1	4 Floppy Drives

NOTES:
1. FDDQTY hardware configuration is effective in HWE mode only.
2. Four floppy drive support requires external logic to decode.

4.2.3 HARDWARE CONFIGURATION TIMING RELATIONSHIPS

The 82091AA samples all of the hardware configuration signals on the high-to-low transition of RSTDRV. For the HWB, SWMB, and SWAI modes, the 82091AA completes hardware configuration on this sampling (Figure 19). For HWE mode, the 82091AA samples some of the signals twice (Figure 20). The first sampling occurs on the high-to-low transition of RSTDRV. As Figure 22 shows (see Section 4.2.5, Extended Hardware Configuration Mode), the HC367 tri-states its outputs when RSTDRV is negated. This permits the strapping options from the HWEb block to be sampled. A short time after RSTDRV is negated (the time is specified in Section 11.0, Electrical Characteristics), the 82091AA samples the SOUTA, RTSA#, DTRA#, SOUTB, RTSB#, and DTRB# signals.

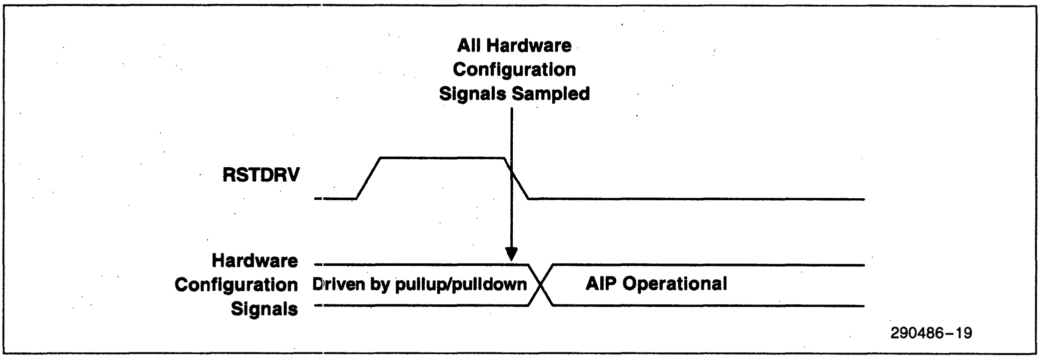


Figure 19. HWB, SWMB, and SWAI Hardware Configuration Mode Timing

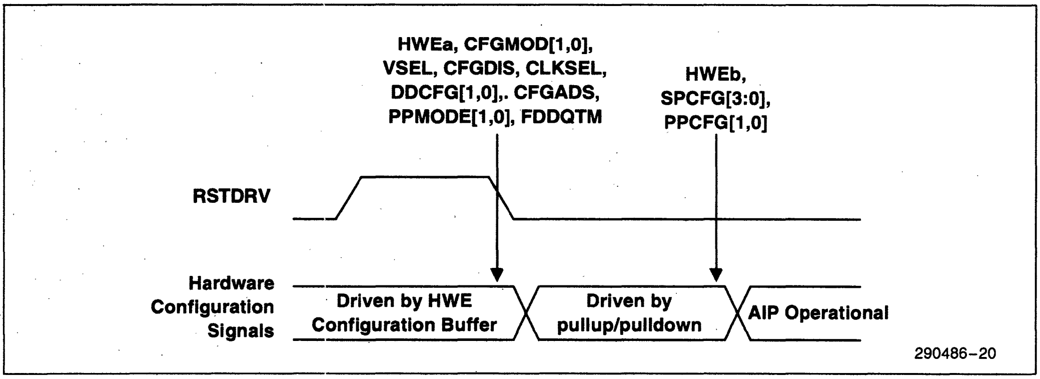


Figure 20. HWE Hardware Configuration Mode Timing

4.2.4 HARDWARE BASIC CONFIGURATION

The Hardware Basic configuration mode permits the user to assign addresses to the serial ports and parallel ports. This is achieved by sampling several of the serial port connections at the end of a hardware reset. The PPDIR/GCS# signal defaults to game port chip select output (GCS#). The 82091AA power supply voltage is not selectable in this mode and is fixed at 5V. The parallel port mode is set to ISA-Compatible. In addition, the FDC floppy drive support is set at two floppy drives. If configuration register access is enabled, the access address is fixed at 398h/399h. To reconfigure the 82091AA using software, the 82091AA configuration mode must be changed to SWAI mode (refer to AIPCFG1 register). Figure 21 shows the implementation of a basic hardware configuration.

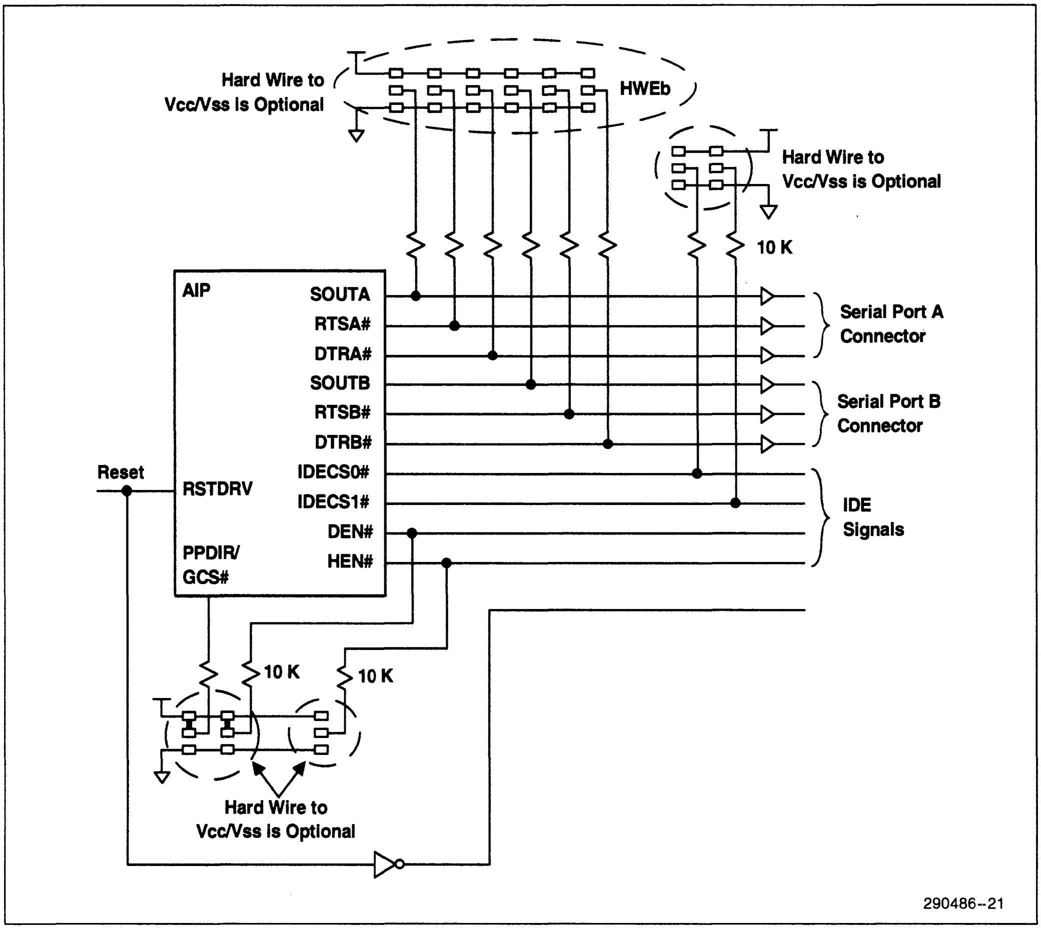


Figure 21. Hardware Basic Configuration

4.2.5 HARDWARE EXTENDED CONFIGURATION MODE

The Hardware Extended configuration mode provides all of the features of the Hardware Basic configuration mode. Additional features in Hardware Extended configuration permit the user to select quantity of floppy drives can be selected for either 2 or 4 floppy drive support. The 82091AA operating voltage is selectable between 3.3V* and 5V. In addition, the parallel port can be configured to operate in ISA-Compatible, PS/2-Compatible, or EPP modes. Hardware extended configuration provides these additional hardware configuration options by sampling the pins on the serial ports at two different times. When RSTDRV is asserted, the HC367 drives the values on SOUTA, RTSA#, DTRA#, SOUTB, RTSB#, and DTRB# (Figure 22). When RTSDRV is negated, the HC367 is disabled and these serial port signals are driven by HWEb pullup/down resistors. The PPDIR/GCS# signal defaults to a game port chip select (GCS#). To reconfigure the 82091AA using software, the 82091AA configuration mode must be changed to SWAI mode (refer to AIPCFG1 register).

NOTE:
*3.3V operation is only available in the 82091AA.

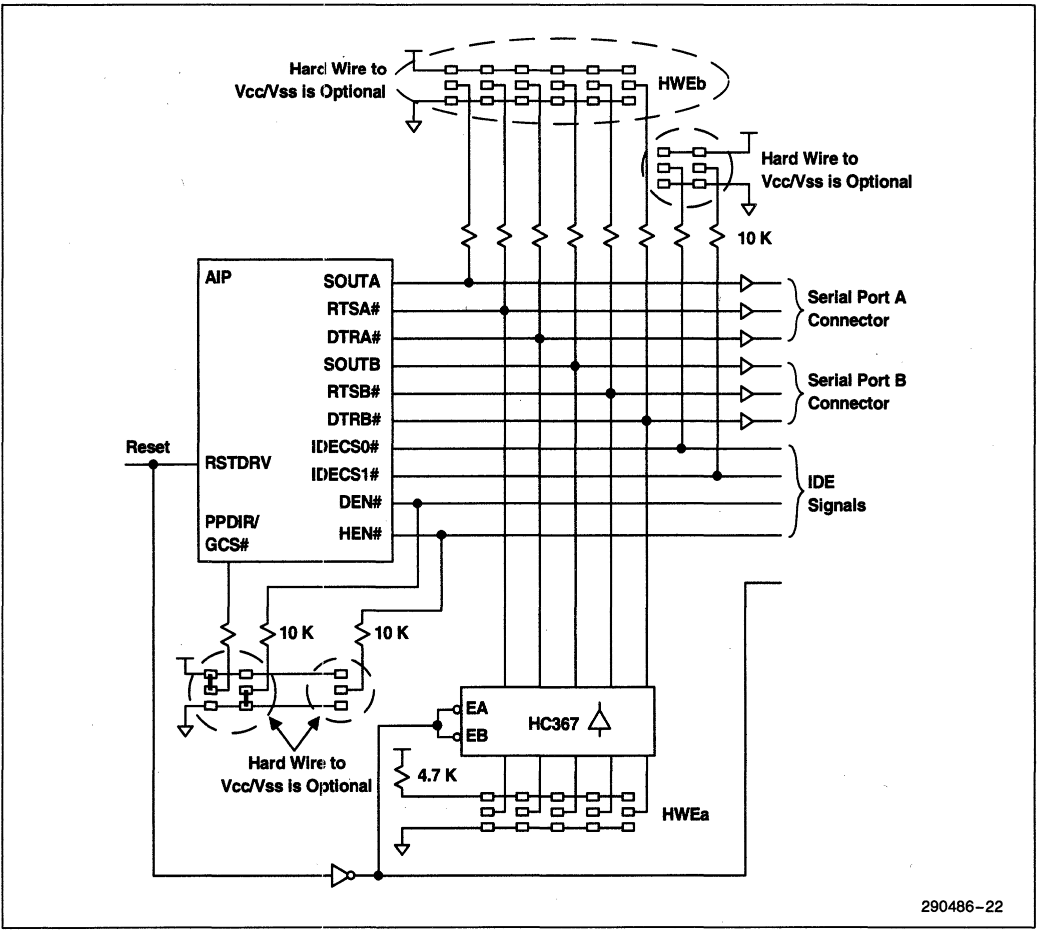


Figure 22. Hardware Extended Configuration

4.2.6 SOFTWARE ADD-IN CONFIGURATION

The Software Add-in configuration mode permits the user to assign the address for the 82091AA configuration registers, and select the power supply voltage for the 82091AA. The 82091AA configuration registers are accessible. The registers are located in the ISA Bus I/O address space and can be selected to be at either 398h/399h or 26Eh/26Fh. The PPDIR/GCS# signal defaults to a game port chip select (GCS#).

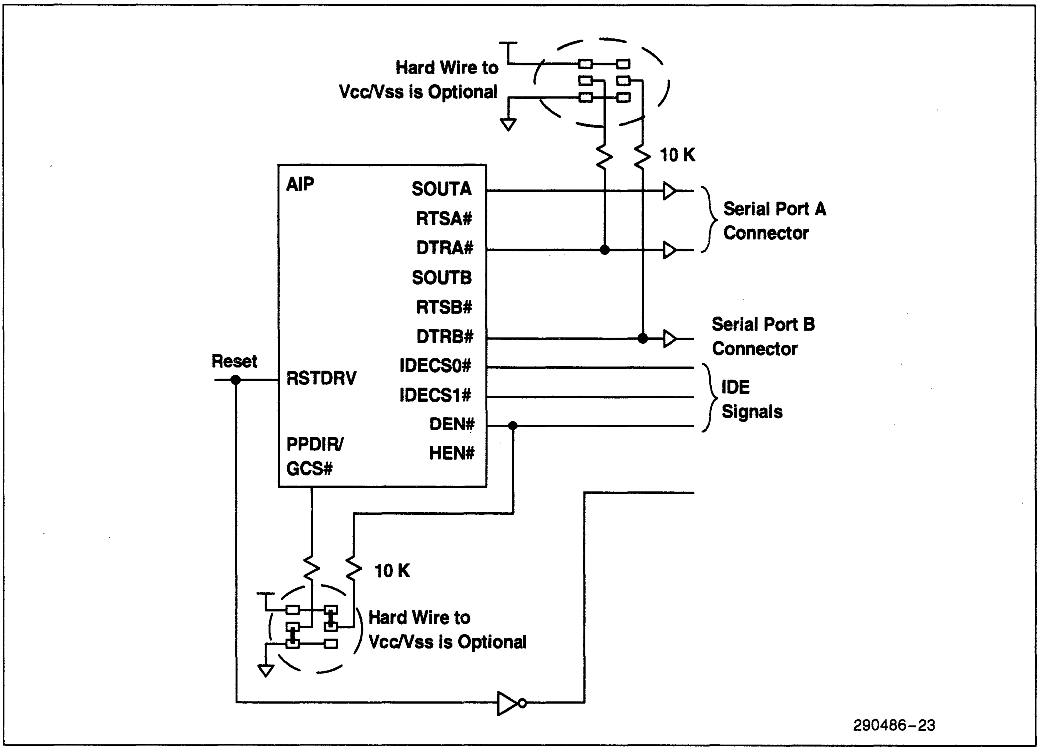


Figure 23. Software Add-In Configuration

4.2.7 SOFTWARE MOTHERBOARD CONFIGURATION

The Software Motherboard configuration mode permits the 82091AA to be located on the motherboard. In this mode, the 82091AA configuration registers are accessible via the X-Bus I/O address space and can be selected to be at either 22h/23h or 24h/25h. In addition, the user selects the power supply voltage for the 82091AA. The PPDIR/GCS# signal defaults to a Parallel Port Direction Control Output (PPDIR).

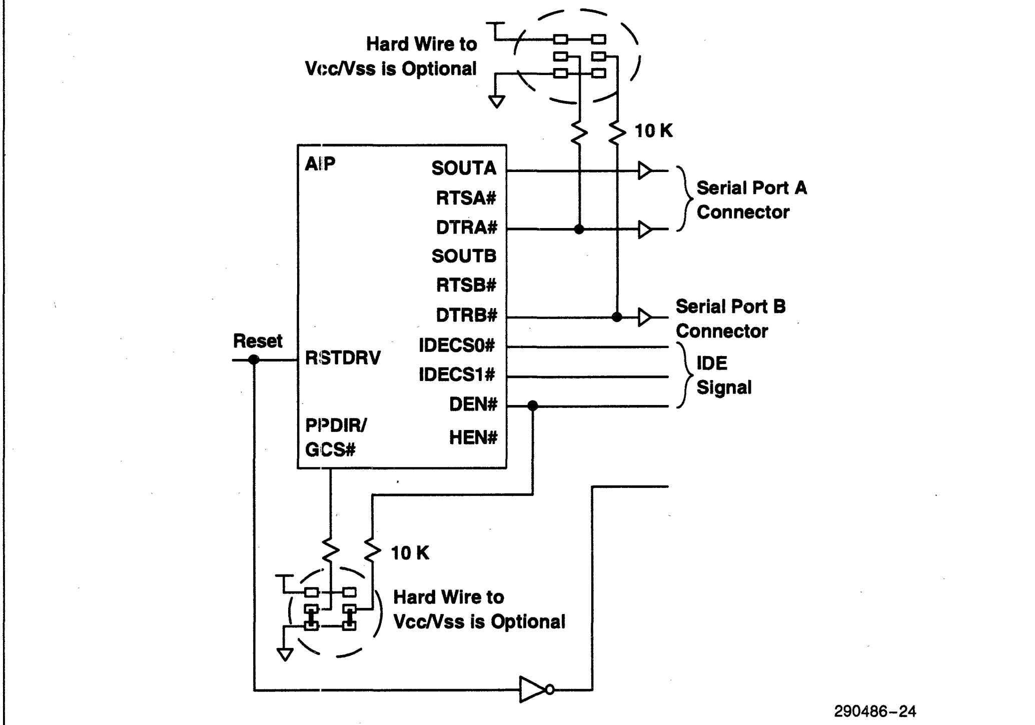


Figure 24. Software Motherboard Hardware Configuration

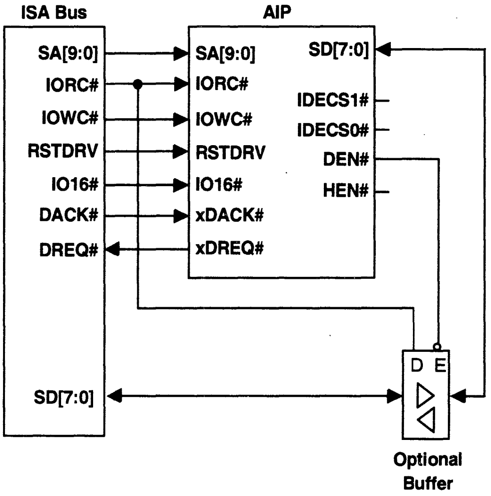
5.0 HOST INTERFACE

The 82091AA host interface is an 8-bit direct-drive (24 mA) ISA Bus/X-Bus interface that permits the CPU to access its registers through read/write operations in I/O space. These registers may be accessed by programmed I/O and/or DMA bus cycles. With the exception of the IDE Interface, all functions on the 82091AA require only 8-bit data accesses. The 16-bit access required for the IDE Interface is supported through the appropriate chip selects and data buffer enables from the 82091AA. The 82091AA does not participate in 16-bit IDE DMA transfers.

Although the 82091AA has an ISA/X-Bus host interface, there are a few features that differentiate it from conventional ISA/X-Bus peripherals. These features are as follows:

- **Internal, Configurable Chip Select Decode Logic.** SA[9:0] allow full decoding of the ISA I/O address space such that the functional modules contained in the 82091AA can be relocated to the desired I/O address. This feature can be used to resolve potential system configuration conflicts.
- **IOCHRDY for ISA Cycle Extension.** During certain I/O cycles to the parallel port controller in the 82091AA, it is necessary to extend the current bus cycle to match the access time of the device connected to the Parallel Port. The IOCHRDY signal is used by the 82091AA to extend ISA Bus cycles, as needed, according to the ISA protocol. IOCHRDY overrides all other strobes that attempt to shorten the bus cycle.
- **NOWS# for 3 BCLK I/O Cycles.** All programmed I/O accesses to 82091AA registers can be completed in a total of 3 BCLK cycles. This is possible because the 82091AA register access times have been minimized to allow data transfers to occur with shortened read/write control strobes. As a result, the 82091AA is well suited for use in embedded control designs that use an asynchronous microprocessor interface without any particular reference to ISA cycle timings.
- **DMA Transfers:** The 82091AA supports DMA compatible, type A, type B and type F DMA cycles. Some newer system DMA controllers are capable of generating fast DMA cycles (type F) on all DMA channels. If such a controller is used in conjunction with the 82091AA, it will be possible to accomplish a DMA transfer in 2 BCLKs.

The 82091AA ISA data lines (SD[7:0]) can be connected directly to the ISA Bus. If external buffers are used to isolate the SD[7:0] signals from the 240 pF loading of the ISA Bus, the DEN# signal can be used to control the external buffers as shown in Figure 25.



290486-25

Figure 25. ISA Interface (with Optional Data Buffer)

6.0 PARALLEL PORT

The 82091AA parallel port can be configured for four parallel port modes. These parallel port modes and the associated parallel interface protocols are:

Parallel Port Mode	Parallel Interface Protocol
ISA-Compatible Mode	Compatibility, Nibble
PS/2-Compatible Mode	Byte
EPP Mode	EPP
ECP	ECP

ISA-Compatible, PS/2-Compatible, and EPP modes are selected through 82091AA configuration (see Section 4.0, AIP Configuration). ECP is selected by programming the ECP Extended Control Register (ECR).

In ISA-Compatible mode, the parallel port exactly emulates a standard ISA-style parallel port. The parallel port data bus (PD[7:0]) is uni-directional. The compatibility protocol transfers data to the peripheral device via PD[7:0] (forward direction). Note that the Nibble protocol permits data transfers from the peripheral device (reverse direction) by using four peripheral status signal lines to transfer 4 bits of data at a time.

PS/2-Compatible mode differs from ISA-Compatible mode by providing bi-directional transfers on PD[7:0]. A bit is added to the PCON Register to allow software control of the data transfer direction.

For both the ISA-Compatible and PS/2-Compatible modes, the actual data transfer over the parallel port interface is accomplished by software handshake (i.e., automatic hardware handshake is not used). Software controls data transfer by monitoring handshake signal status from the peripheral device via the PSTAT Register and controlling handshake signals to the peripheral device via the PCON Register.

EPP mode provides bi-directional transfers on PD[7:0]. The 82091AA automatically generates the address and data strobes in hardware.

ECP is a high performance peripheral interface mode. This mode uses an asynchronous automatic handshake to transfer data over the parallel port interface. In addition, the parallel port contains a FIFO for transferring data in ECP mode. The ECP register set contains an Extended Control Register (ECR) that provides a wide range of functions including the ability to operate the parallel port in either ECP, ISA-Compatible, or PS/2-Compatible modes.

NOTE:

In general, this document describes parallel port operations and functions in terms of how the 82091AA parallel port hardware operates. Detailed descriptions of the parallel interface protocols are beyond the scope of this document. Readers should refer to the proposed IEEE Standard 1284 for detailed descriptions of the Compatibility, Nibble, Byte, EPP, and ECP protocols.

Special circuitry on the 82091 prevents it from being powered up or being damaged while a parallel port peripheral is powered on and the 82091 is powered off.

6.1 Parallel Port Registers

This section is organized into three sub-sections—ISA-Compatible and PS/2-Compatible Modes, EPP Mode, and ECP Mode. Since the register sets are similar for ISA-Compatible and PS/2-Compatible modes (differing by a direction control bit in the PCON Register) the register set descriptions are combined. The EPP mode and ECP mode register sets are described separately. Each register set description contains the I/O address assignment and a complete description of the registers and register bits. Note that the PSTAT and PCON Registers are common to all modes and for completeness are repeated in each sub-section. Any difference in bit operations for a particular mode is noted in that particular register description.

The registers provide parallel port control/status information and data paths for transferring data between the parallel port interface and the 8-bit host interface. All registers are accessed as byte quantities. The base address is determined by hardware configuration at powerup (or a hard reset) or via software configuration by programming the 82091AA configuration registers as described in Section 4.0, AIP Configuration. The parallel port can be disabled or configured for a base address of 378h (all modes), 278h (all modes), or 3BCh (all modes except EPP and ECP). This provides the system designer with the option of using additional parallel ports on add-in cards that have fixed address decoding.

Some of the parallel port registers described in this section contain reserved bits. These bits are labeled "R". Software must deal correctly with fields that are reserved. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved bit positions are preserved. That is, the value of reserved bit positions must first be read, merged with the new values for other bit positions, and then written back.

During a hard reset (RSTDRV asserted), the 82091AA registers are set to pre-determined **default** states. The default values are indicated in the individual register descriptions.

The following nomenclature is used for register access attributes:

RO **Read Only**. Note that for registers with read only attributes, writes to the I/O address have no affect on parallel port operations.

R/W **Read/Write**. A register with this attribute can be read and written. Note that individual bits in some read/write registers may be read only.

6.1.1 ISA-COMPATIBLE AND PS/2-COMPATIBLE MODES

This section contains the registers used in ISA-Compatible and PS/2-Compatible modes. The I/O address assignment for this register set is shown in Table 15 and the register descriptions are presented in the order that they appear in the table.

Table 15. Parallel Port Register (ISA-Compatible and PS/2-Compatible)

Parallel Port Register Address Access (AEN = 0) Base +	Abbreviation	Register Name	Access
0h	PDATA	Data Register	R/W
1h	PSTAT	Status Register	RO
2h	PCON	Control Register	R/W

NOTE:
Parallel port base addresses are 278h, 378h and 3BCh.

6.1.1.1 PDATA—Parallel Port Data Register (ISA-Compatible and PS/2-Compatible Modes)

I/O Address:	Base +00h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

ISA-Compatible Mode

The PDATA Register is a uni-directional data port that transfers 8-bit data from the host to the peripheral device (forward transfer). A write to this register drives the written data onto PD[7:0]. Reads of this register should not be performed in ISA-Compatible mode. For a host read of this address location, the 82091AA completes the handshake on the ISA Bus and the value is the last value stored in the PDATA Register.

PS/2-Compatible Mode

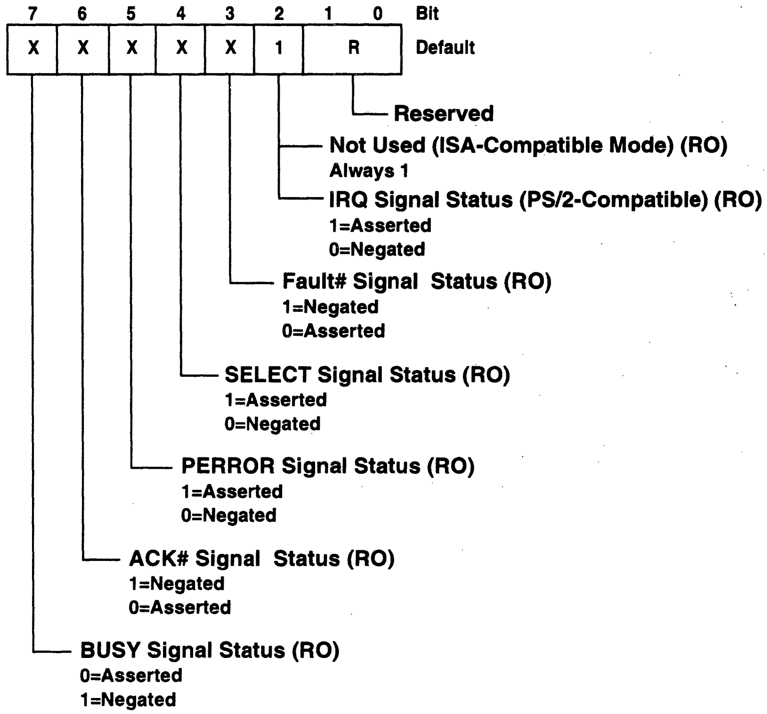
The PDATA Register is a bi-directional data port that transfers 8-bit data between the peripheral device and host. The direction of transfer is determined by the DIR# bit in the PCON Register. If DIR# = 0 (forward direction), and the host writes to this register, the data is stored in the PDATA Register and driven onto PD[7:0]. If DIR# = 1 (reverse direction), a host read of this register returns the data on PD[7:0]. Note that read data is not stored in the PDATA Register.

Bit	Description
7:0	**PARALLEL PORT DATA:** Bits[7:0] correspond to parallel port data lines PD[7:0] and ISA Bus data lines SD[7:0].

6.1.1.2 PSTAT—Status Register (ISA-Compatible and PS/2-Compatible Modes)

I/O Address:	Base +01h
Default Value:	XXXX X1RR
Attribute:	Read Only
Size:	8 bits

The PSTAT Register provides the status of certain parallel port signals and whether a CPU interrupt has been generated by the parallel port. This register indicates the current state of the BUSY, ACK#, PERROR, SELECT, and FAULT# signals.



7	6	5	4	3	2	1	0	Bit
X	X	X	X	X	1	R		Default

Reserved

Not Used (ISA-Compatible Mode) (RO)
Always 1

IRQ Signal Status (PS/2-Compatible) (RO)
1=Asserted
0=Negated

Fault# Signal Status (RO)
1=Negated
0=Asserted

SELECT Signal Status (RO)
1=Asserted
0=Negated

PERROR Signal Status (RO)
1=Asserted
0=Negated

ACK# Signal Status (RO)
1=Negated
0=Asserted

BUSY Signal Status (RO)
0=Asserted
1=Negated

290486-26

NOTE:
X = Default value is determined by signal state at reset.

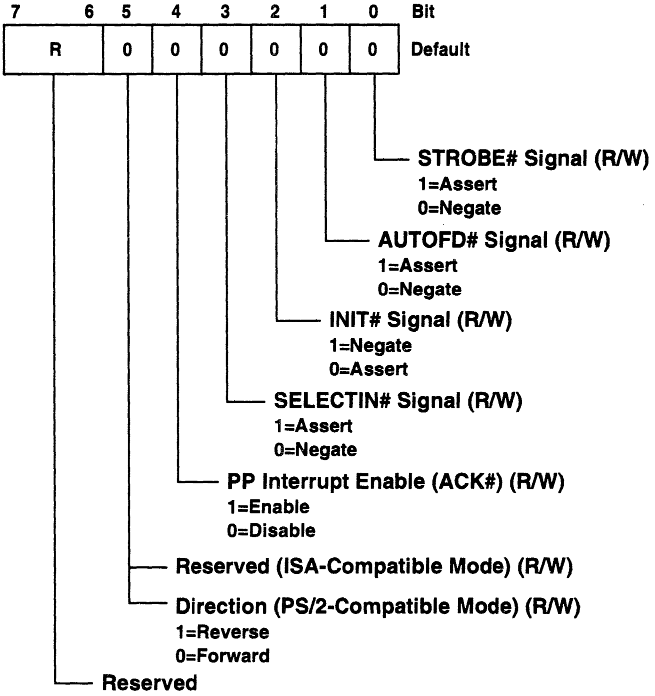
Figure 26. Status Register (ISA-Compatible and PS/2-Compatible Modes)

Bit	Description
7	**BUSY STATUS (BUSYS):** This bit indicates the state of the parallel port interface BUSY signal. When BUSY is asserted, BUSYS = 0. When BUSY is negated, BUSYS = 1.This bit is an inverted version of the parallel port BUSY signal.
6	**ACK# STATUS (ACKS):** This bit indicates the state of the parallel port interface ACK# signal. This bit indicates when the peripheral has received a data byte and is ready for another. When ACK# is asserted, ACKS = 0. When ACK# is negated, ACKS = 1. Note that if interrupts are enabled (via bit 4 of the PCON Register), the assertion of the ACK# signal generates an interrupt to the CPU.
5	**PERROR STATUS (PERRS):** This bit indicates the state of the parallel port interface PERROR signal. This bit indicates when an error has occurred in the peripheral paper path (e.g., out of paper). When PERROR is asserted, PERRS = 1, When PERROR is negated, PERRS = 0.
4	**SELECT STATUS (SELS):** This bit indicates the state of the parallel port interface SELECT signal. When the SELECT signal is asserted, SELS = 1, When the SELECT signal is negated, SELS = 0.
3	**FAULT# STATUS (FAULTS):** This bit indicates the state of the parallel port interface FAULT# signal being driven by the peripheral device. When the FAULT# signal is asserted, FAULTS = 0. When the FAULT# signal is negated, FAULTS = 1.
2	**PARALLEL PORT INTERRUPT STATUS (PIRQ):** This bit indicates a CPU interrupt by the parallel port. PIRQ indicates that the printer has accepted the previous character and is ready for another. In ISA-Compatible mode, interrupt status is not reported in this register and this bit is always 1. In PS/2-Compatibile mode, if interrupts are enabled via the PCON Register and the ACK# signal is asserted (low-to-high transition), PIRQ is set to a 0 (and an IRQ generated to the CPU). The 82091AA sets PIRQ to 1 when this register is read or by a hard reset. If interrupts are disabled via the PCON Register, this bit is never set to 0.
1:0	**RESERVED**

6.1.1.3 PCON—Control Register (ISA-Compatible And PS/2-Compatible Mode)

I/O Address:	Base + 02h
Default Value:	RR00 0000
Attribute:	Read/Write
Size:	8 bits

The PCON Register controls certain parallel port interface signals and enables/disables parallel port interrupts. This register permits software to control the STROBE#, AUTOFD#, INIT#, and SELECTIN# signals. For PS/2-Compatible mode, this register also controls the direction of transfer on PD[7:0].



290486-27

Figure 27. Control Register (ISA-Compatible and PS/2-Compatible Modes)

Bit	Description
7:6	**RESERVED**
5	**RESERVED (ISA-COMPATIBLE MODE):** Not used and undefined when read. Writes have no affect on parallel port operations. **DIRECTION (DIR#) (PS/2-COMPATIBLE MODE):** This bit is used to control the direction of data transfer on the parallel port data bus (PD[7:0]). When DIR# = 0, PD[7:0] are outputs. When DIR# = 1, PD[7:0] are inputs.
4	**ACK# INTERRUPT ENABLE (ACKINTEN):** ACKINTEN enables CPU interrupts (via either IRQ5 or IRQ7) to be generated when the ACK# signal on the parallel port interface is asserted. When ACKINTEN = 1, a CPU interrupt is generated when ACK# is asserted. When ACKINTEN = 0, the ACK# interrupt is disabled.
3	**SELECTIN# CONTROL (SELINC):** This bit controls the SELECTIN# signal. SELINC is set to 1 to select the printer. When SELINC = 1, the SELECTIN# signal is asserted, When SELINC = 0, the SELECTIN# signal is negated.
2	**INIT# CONTROL (INITC):** This bit controls the INIT# signal. When INITC = 1, the INIT# signal is negated. When INITC = 0, the INIT# signal is asserted.
1	**AUTOFD# CONTROL (AUTOFDC):** This bit controls the AUTOFD# signal. AUTOFDC is set to 1 to instruct the printer to advance the paper one line each time a carriage return is received. When AUTOFDC = 1, the AUTOFD# signal is asserted. When AUTOFDC = 0, the AUTOFD# signal is negated.
0	**STROBE# CONTROL (STROBEC):** This bit controls the STROBE# signal. The STROBE# signal is set active to instruct the printer to accept the character being presented on the data lines. When STROBEC = 1, the STROBE# signal is asserted. When STROBEC = 0, the STROBE# signal is negated.

6.1.2 EPP MODE

This section contains the registers used in EPP mode. The I/O address assigment for this register set is shown in Table 16 and the register descriptions are presented in the order that they appear in the table.

Table 16. Parallel Port Registers (EPP Mode)

Parallel Port Register Address Access (AEN = 0) Base +	Abbreviation	Register Name	Access
0h	PDATA	Data Register	R/W
1h	PSTAT	Status Register	RO
2h	PCON	Control Register	R/W
3h	ADDSTR	Address Strobe Register	R/W
4h–7h	DATASTR	Data Strobe Registers	R/W

NOTE:
Parallel port base addresses are 278h (LPT2) and 378h (LPT1). Base address 3BCh is not available in EPP or ECP modes.

6.1.2.1 PDATA—Parallel Port Data Register (EPP Mode)

I/O Address:	Base + 00h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

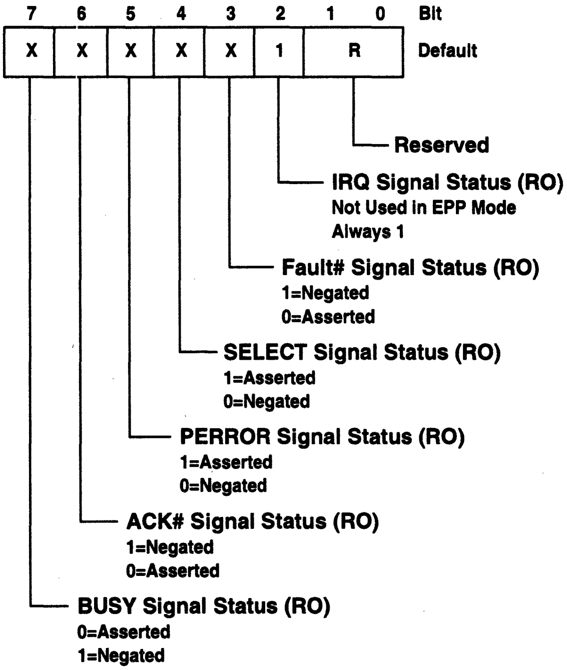
The PDATA Register is a bi-directional data port that transfers 8-bit data between the peripheral device and host. The direction of transfer is determined by the DIR# bit in the PCON Register. If DIR# = 0 (forward direction) and the host writes to this register, the data is stored in the PDATA Register and driven onto PD[7:0]. If DIR# = 1 (reverse direction), a host read of this register returns the data on PD[7:0]. However, read data is not stored in the PDATA Register.

Bit	Description
7:0	**PARALLEL PORT DATA:** Bits[7:0] correspond to parallel port data lines PD[7:0] and ISA Bus data lines.

6.1.2.2 PSTAT—Status Register (EPP Mode)

I/O Address:	Base +01h
Default Value:	XXXX X1RR
Attribute:	Read Only
Size:	8 bits

The PSTAT Register provides the status of certain parallel port signals. It also indicates whether a CPU interrupt has been generated by the parallel port. This register indicates the current state of the BUSY, ACK#, PERROR, SELECT, and FAULT# signals.



290486–28

NOTE:
X = Default value is determined by signal state at reset.

Figure 28. Status Register (EPP Mode)

Bit	Description
7	**BUSY STATUS (BUSYS):** This bit indicates the state of the parallel port interface BUSY signal. When BUSY is asserted, BUSYS = 0. When BUSY is negated, BUSYS = 1. This bit is an inverted version of the parallel port BUSY signal.
6	**ACK# STATUS (ACKS):** This bit indicates the state of the parallel port interface ACK# signal. This bit indicates when the peripheral has received a data byte and is ready for another. When ACK# is asserted, ACKS = 0. When ACK# is negated, ACKS = 1. Note that if interrupts are enabled (via bit 4 of the PCON Register), the assertion of the ACK# signal generates an interrupt to the CPU.
5	**PERROR STATUS (PERRS):** This bit indicates the state of the parallel port interface PERROR signal. This bit indicates when an error has occurred in the peripheral paper path (e.g., out of paper). When PERROR is asserted, PERRS = 1. When PERROR is negated, PERRS = 0.
4	**SELECT STATUS (SELS):** This bit indicates the state of the parallel port interface SELECT signal. When the SELECT signal is asserted, SELS = 1. When the SELECT signal is negated, SELS = 0.
3	**FAULT# STATUS (FAULTS):** This bit indicates the state of the parallel port interface FAULT# signal being driven by the peripheral device. When the FAULT# signal is asserted, FAULTS = 0. When the FAULT# signal is negated, FAULTS = 1.
2	**PARALLEL PORT INTERRUPT (PIRQ):** In EPP mode interrupt status is not reported in this register and this bit is always 1.
1:0	**RESERVED**

6.1.2.3 PCON—Control Register (EPP Mode)

I/O Address:	Base + 02h
Default Value:	RR00 0000
Attribute:	Read/Write
Size:	8 bits

The PCON Register controls certain parallel port interface signals, enables/disables parallel port interrupts, and selects the direction of data transfer on PD[7:0]. This register permits software to control the INIT# signal. Note that in the EPP parallel interface protocol, the STROBE#, AUTOFD#, and SELECTIN# signals are automatically generated by the parallel port and are not controlled by software.

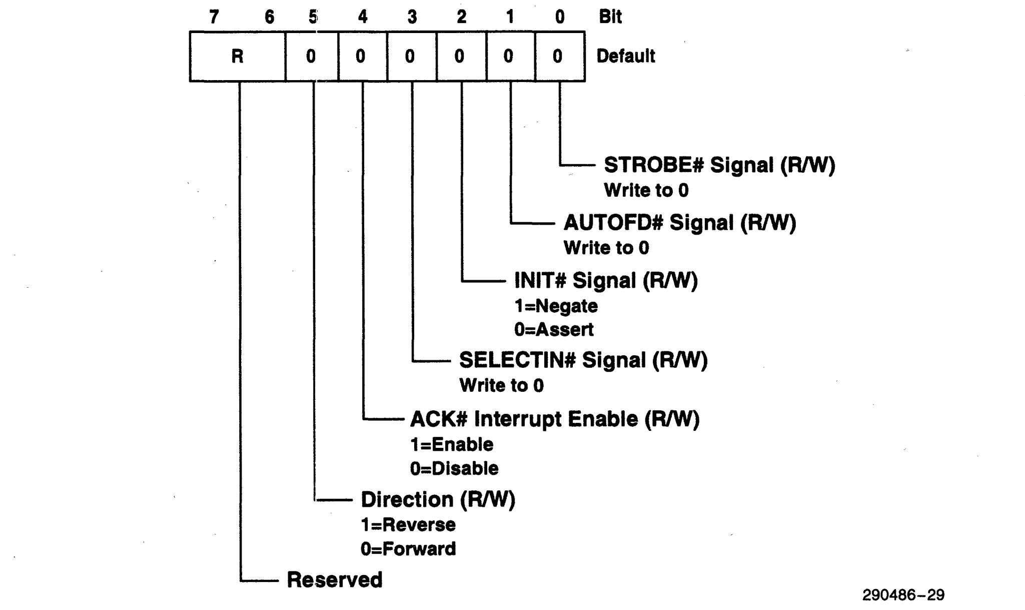


Figure 29. Control Register (EPP Mode)

Bit	Description
7:6	**RESERVED**
5	**DIRECTION (DIR#):** This bit is used to control the direction of data transfer on the parallel port data bus (PD[7:0]). When DIR# = 0 (forward direction), PD[7:0] are outputs. When DIR# = 1 (reverse direction), PD[7:0] are inputs.
4	**ACK# INTERRUPT ENABLE (ACKINTEN):** ACKINTEN enables CPU interrupts (via IRQ5 or IRQ7) to be generated when the ACK# signal on the parallel port interface is asserted. When ACKINTEN = 1, a CPU interrupt is generated when ACK# is asserted. When ACKINTEN = 0, the ACK# interrupt is disabled.
3	**SELECTIN# CONTROL (SELINC):** Write to 0 when programming this register. This bit must be 0 for the parallel port handshake to operate properly.
2	**INIT# CONTROL (INITC):** This bit controls the INIT# signal. When INITC = 1, the INIT# signal is negated. When INITC = 0, the INIT# signal is asserted.
1	**AUTOFD# CONTROL (AUTOFDC):** Write to 0 when programming this register.
0	**STROBE# CONTROL (STROBEC):** Write to 0 when programming this register. This bit must be 0 for the parallel port handshake to operate properly.

6.1.2.4 ADDSTR—EPP Auto Address Strobe Register (EPP Mode)

I/O Address:	Base + 03h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

The ADDSTR Register provides a peripheral address to the peripheral (via PD[7:0]) during a host address write operation and to the host (via PD[7:0]) during a host address read operation. An automatic address strobe is generated on the parallel port interface when data is read from or written to this register.

Bit	Description
7:0	**EPP ADDRESS:** Bits[7:0] correspond to SD[7:0] and PD[7:0].

6.1.2.5 DATASTR—Auto Data Strobe Register (EPP Mode)

I/O Address:	Base +04h, 05h, 06h, 07h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

The DATASTR Register provides data from the host to the peripheral device (via PD[7:0]) during host write operations and from the peripheral device to the host (via PD[7:0]) during a host read operation. An automatic data strobe is generated on the parallel port interface when data is read from or written to this register. To maintain compatibility with Intel's 82360SL I/O device that has a 32-bit Host Bus interface, four consecutive byte address locations are provided for transferring data.

Bit	Description
7:0	**EPP DATA:** Bits[7:0] correspond to SD[7:0] and PD[7:0].

6.1.3 ECP MODE

This section contains the registers used in ECP mode. The I/O address assignment for this register set is shown in Table 17 and the register descriptions are presented in the order that they appear in the table. The Extended Control Register (ECR) permits various modes of operation. Note that ECR[7:5] = 000 selects ISA-Compatible mode and ECR[7:5] = 001 selects PS/2-Compatibile mode. These modes are discussed in Section 6.1.1, ISA-Compatible and PS/2 Compatible modes. The other modes selected by ECR[7:5] are discussed in this section.

Table 17. Parallel Port Registers (ECP Mode)

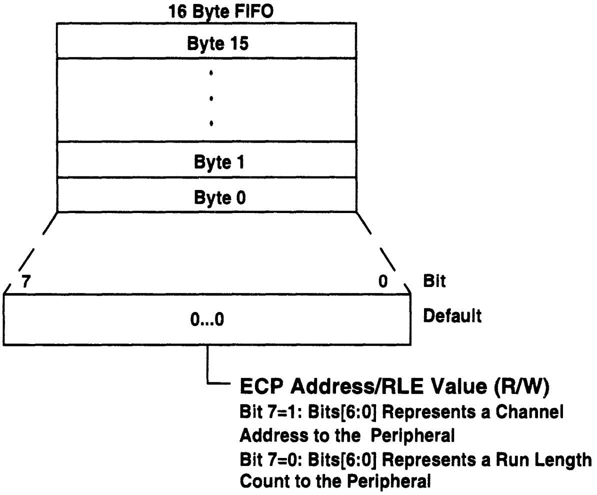
Parallel Port Register Address Access (AEN = 0) Base +	Abbreviation	Register Name	Access	
			ECR[7:5]	Read/Write Attribute
0h	ECPAFIFO	ECP Address/RLE FIFO	011	R/W
1h	PSTAT	Status Register	All	RO
2h	PCON	Control Register	All	R/W
400h	SDFIFO	Standard Parallel Port Data FIFO	010	R/W
400h	ECPDFIFO	ECP Data FIFO	011	R/W
400h	TFIFO	Test FIFO	110	R/W
400h	ECPCFGA	ECP Configuration A	111	R/W
401h	ECPCFGB	ECP Configuration B	111	R/W
402h	ECR	Extended Control Register	All	R/W

NOTES:
1. Parallel port base addresses are 278h, 378h, and 3BCh.
2. A register is accessible when the ECR[7:5] field contains the value specified in the ECR[7:5] column. The register is not accessible if the ECR[7:5] field does not match the value specified in this column. The term "All" means that the register is accessible in all modes selected by ECR[7:5].

6.1.3.1 ECPAFIFO—ECP Address/RLE FIFO Register (ECP Mode)

I/O Address:	Base +00h
Default Value:	UUUU UUUU (Undefined)
Attribute:	Read/Write
Size:	8 bits

The ECPAFIFO Register provides a channel address or a Run Length Count (RLE) to the peripheral, depending on the state of bit 7. This I/O address location is only used in ECP mode (ECR bits[7:5]=011). In this mode, bytes written to this register are placed in the parallel port FIFO and transmitted over PD[7:0] using ECP protocol.


16 Byte FIFO
Byte 15
Byte 1
Byte 0
7
0
Bit
0...0
Default
ECP Address/RLE Value (R/W)
Bit 7=1: Bits[6:0] Represents a Channel Address to the Peripheral
Bit 7=0: Bits[6:0] Represents a Run Length Count to the Peripheral

290486-30

NOTE:
U = Undefined

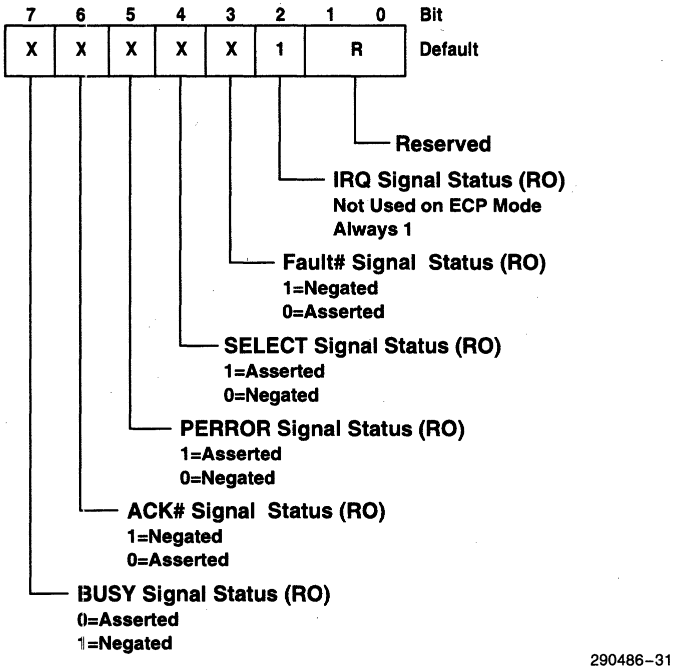
Figure 30. ECP Address/RLE FIFO Register (ECP Mode)

Bit	Description
7:0	**ECP ADDRESS/RLE VALUE:** Bits[7:0] correspond to parallel port data lines PD[7:0] and ISA Bus data lines SD[7:0]. The peripheral device should interpret bits[6:0] as a channel address when bit 7 = 1 and as a run length count when bit 7 = 0. Note that this interpretation is performed by the peripheral device and the value of bit 7 has no affect on 82091AA operations. Note that the 82091AA asserts AUTOFD# to indicate that the information on PD[7:0] represents an ECP address/RLE count. The 82091AA negates AUTOFD# (drives high) when PD[7:0] is transferring data.

6.1.3.2 PSTAT—Status Register (ECP Mode)

I/O Address:	Base +01h
Default Value:	XXXX X1RR
Attribute:	Read Only
Size:	8 bits

The PSTAT Register provides the status of certain parallel port signals and whether a CPU interrupt has been generated by the parallel port. This register indicates the current state of the BUSY, ACK#, PERROR, SELECT, and FAULT# signals.



NOTE:
X = Default value is determined by the state of the corresponding signal pin at reset.

Figure 31. Status Register (ECP Mode)

Bit	Description
7	**BUSY STATUS (BUSYS):** This bit indicates the state of the parallel port interface BUSY signal. When BUSY is asserted, BUSYS = 0. When BUSY is negated, BUSYS = 1. This is an inverted version of the parallel port BUSY signal. Refer to Section 6.2.3 ECP Mode for more detail.
6	**ACK# STATUS (ACKS):** This bit indicates the state of the parallel port interface ACK# signal. This bit indicates when the peripheral has received a data byte and is ready for another. When ACK# is asserted, ACKS = 0. When ACK# is negated, ACKS = 1. Note that if interrupts are enabled (via bit 4 of the PCON Register), the assertion of the ACK# signal generates an interrupt to the CPU. Refer to Section 6.2.3 ECP Mode for more detail.
5	**PERROR STATUS (PERRS):** This bit indicates the state of the parallel port interface PERROR signal. This bit indicates when an error has occurred in the peripheral paper path (e.g., out of paper). When PERROR is asserted, PERRS = 1, When PERROR is negated, PERRS = 0.
4	**SELECT STATUS (SELS):** This bit is used in all parallel port modes and indicates the state of the parallel port interface SELECT signal. When the SELECT signal is asserted, SELS = 1. When the SELECT signal is negated, SELS = 0.
3	**FAULT# STATUS (FAULTS):** This bit is used in all parallel port modes and indicates the state of the parallel port interface FAULT# signal being driven by the peripheral device. When the FAULT# signal is asserted, FAULTS = 0. When the FAULT# signal is negated, FAULTS = 1.
2	**PARALLEL PORT INTERRUPT (PIRQ):** In ECP mode, interrupt status is not reported in this register and this bit is always 1.
1:0	**RESERVED**

6.1.3.3 PCON—Control Register (ECP Mode)

I/O Address:	Base + 02h
Default Value:	RR00 0000
Attribute:	Read/Write
Size:	8 bits

The PCON Register controls certain parallel port interface signals, enables/disables parallel port interrupts, and selects the direction of data transfer on PD[7:0]. Note that the function of some bits depends on the programming of the ECR.

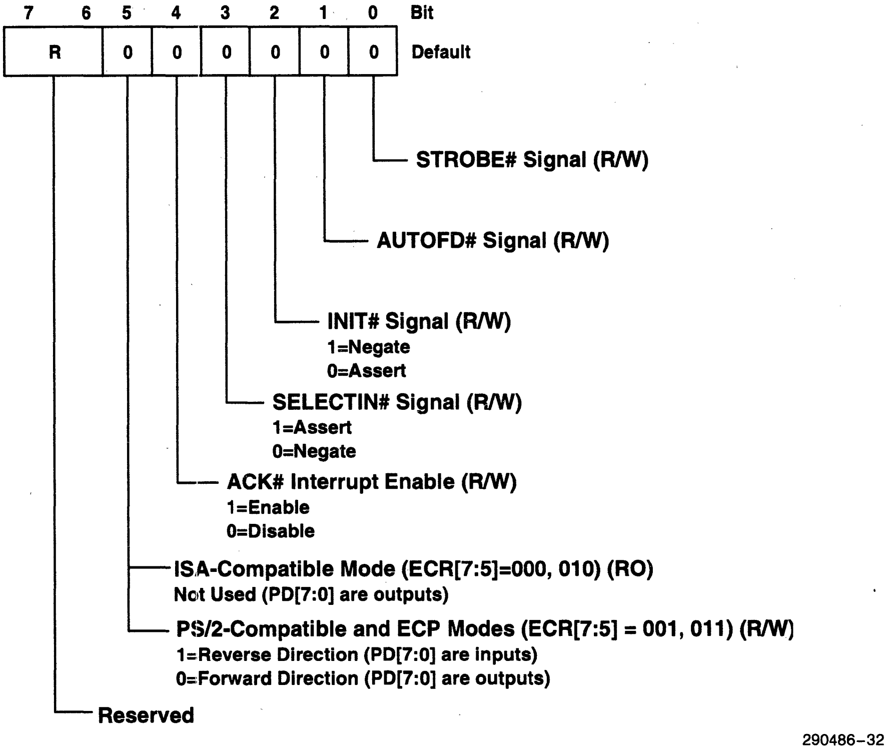

7 6 5 4 3 2 1 0 Bit
R 0 0 0 0 0 0 Default
STROBE# Signal (R/W)
AUTOFD# Signal (R/W)
INIT# Signal (R/W)
1=Negate
0=Assert
SELECTIN# Signal (R/W)
1=Assert
0=Negate
ACK# Interrupt Enable (R/W)
1=Enable
0=Disable
ISA-Compatible Mode (ECR[7:5]=000, 010) (RO)
Not Used (PD[7:0] are outputs)
PS/2-Compatible and ECP Modes (ECR[7:5] = 001, 011) (R/W)
1=Reverse Direction (PD[7:0] are inputs)
0=Forward Direction (PD[7:0] are outputs)
Reserved
290486–32

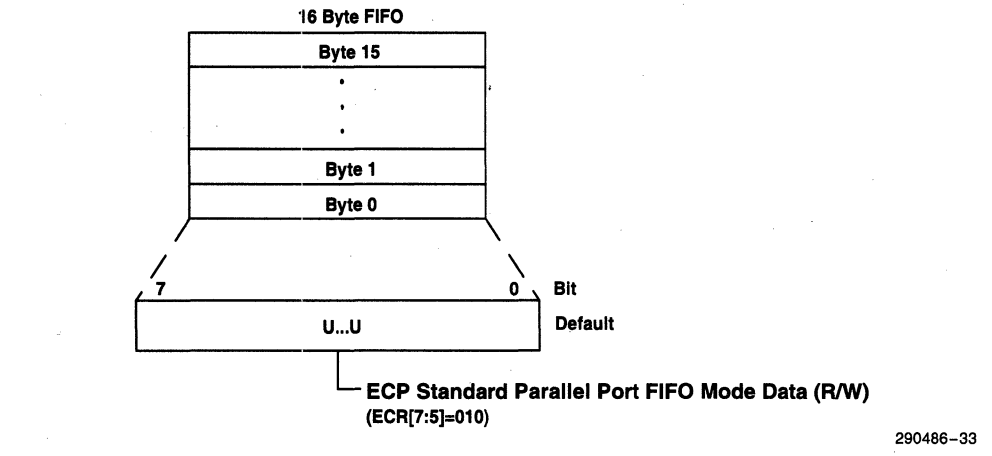
Figure 32. Control Register (ECP Mode)

Bit	Description
7:6	**RESERVED**
5	**DIRECTION (DIR#):** This bit is used to control the direction of data transfer on the parallel port data bus (PD[7:0]). When DIR# = 0 (forward direction), PD[7:0] are outputs. When DIR# = 1 (reverse direction), PD[7:0] are inputs.
4	**INTERRUPT ENABLE (ACK#) (IRQEN):** IRQEN enables interrupts to the CPU to be generated when the ACK# signal on the parallel port interface is asserted and is used in all parallel port interface modes. When IRQEN = 1, a CPU interrupt is generated when ACK# is asserted. When IRQEN = 0, parallel port interrupts are disabled.
3	**SELECTIN# CONTROL (SELINC):** This bit controls the SELECTIN# signal. SELINC is set to 1 to select the printer. When SELINC = 1, the SELECTIN# signal is asserted, When SELINC = 0, the SELECTIN# signal is negated.
2	**INIT# CONTROL (INITC):** This bit controls the INIT# signal When INITC = 1, the INIT# signal is negated. When INITC = 0, the INIT# signal is asserted.
1	**AUTOFD# CONTROL (AUTOFDC):** In ECP mode or ISA-Compatible FIFO mode (ECR[7:5] = 011, 010), this bit has no effect. Refer to Section 6.2.3 ECP Mode for more details.
0	**STROBE# CONTROL (STROBEC):** In ECP mode or ISA-Compatible FIFO mode (ECR[7:5] = 011, 010), this bit has no effect. Refer to Section 6.2.3 ECP Mode for more details.

6.1.3.4 SDFIFO—Standard Parallel Port Data FIFO

I/O Address:	Base +400h and (ECR[7:5]=010)
Default Value:	UUUU UUUU (undefined)
Attribute:	Read/Write
Size:	8 bits

SDFIFO is used to transfer data from the host to the peripheral when the ECR Register is set for ISA-Compatible FIFO mode (bits[7:5]=010). Data bytes written or DMAed from the system to this FIFO are transmitted by a hardware handshake to the peripheral using the standard ISA-Compatible protocol. Note that bit 5 in the PCON Register must be set to 0 for a forward transfer direction.



NOTE:
U = Undefined

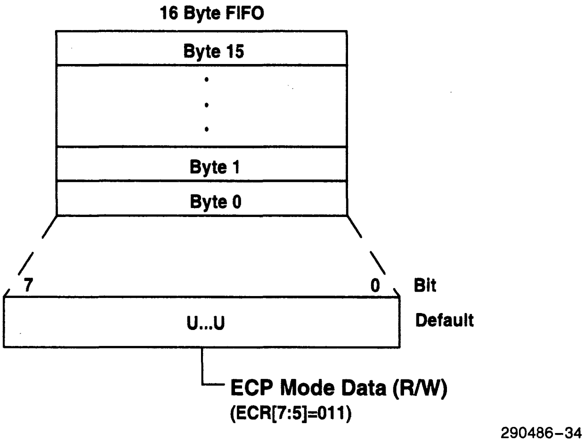
Figure 33. ECP ISA-Compatible Data FIFO

Bit	Description
7:0	**ECP STANDARD PARALLEL PORT DATA:** Bits[7:0] correspond to SD[7:0] and PD[7:0].

6.1.3.5 DFIFO—Data FIFO (ECP Mode)

I/O Address: Base +400h and (ECR[7:5]=011)
Default Value: UUUU UUUU (undefined)
Attribute: Read/Write
Size: 8 bits

This I/O address location transfers data between the host and peripheral device when the parallel port is in ECP mode (ECR Bits[7:5]=011). Transfers use the parallel port FIFO. Data is transferred on PD[7:0] via hardware handshakes on the parallel port interface using ECP parallel port interface handshake protocol.



NOTE:
U=Undefined

Figure 34. ECP Data FIFO (ECP Mode)

Bit	Description
7:0	**ECP MODE DATA:** Data bytes written or DMAed from the system to this FIFO in the forward direction (PCON bit 5=0) are transmitted to the peripheral by an ECP mode protocol hardware handshake. In the reverse direction (PCON bit 5=1) data bytes from the peripheral are transferred to the FIFO using the ECP mode protocol hardware handshake. Reads or DMAs from the FIFO return bytes of ECP data to the system. Bits[7:0] correspond to SD[7:0] and PD[7:0].

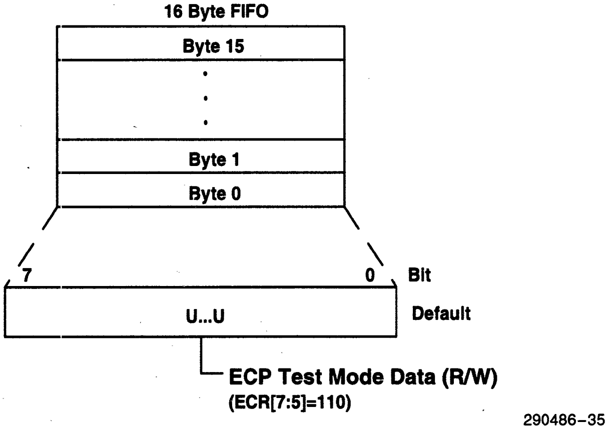
6.1.3.6 TFIFO—ECP Test FIFO Register (ECP Mode)

I/O Address:	Base +400 and (ECR[7:5]=110)
Default Value:	UUUU UUUU (undefined)
Attribute:	Read/Write
Size:	8 bits

The TFIFO Register provides a test mechanism for the ECP mode FIFO. Test mode is enabled via the ECR Register. In test mode (ECR[7:5]=110), data can be read, written or DMAed to/from the FIFO by accessing this register I/O address.

Data bytes may be read, written or DMAed to or from the system to this FIFO in any direction. The parallel port interface signals are not affected by TFIFO accesses and TFIFO data is not transmitted to PD[7:0]. The test FIFO does not stall when overwritten or underrun. Data is simply re-written or over-run. The full and the empty bits in the ECR always keep track of the correct FIFO state.

The test FIFO transfers data at the maximum ISA rate so that software can generate performance metrics. The FIFO write threshold can be determined by starting with a full TFIFO and emptying it a byte at a time until a service interrupt is set to 1 in the ECR. The FIFO read threshold can be determined by setting the direction bit in the PCON Register to 1, and filling the FIFO a byte at a time until the service interrupt is set to 1 in the ECR.



NOTE:
U=Undefined

Figure 35. ECP Test FIFO Register (ECP Mode)

Bit	Description
7:0	**ECP TEST FIFO** Data: Bits[7:0] correspond to SD[7:0].

6.1.3.7 ECPCFGA—ECP Configuration A Register (ECP Mode)

I/O Address:	Base +400h and (ECR[7:5] = 111)
Default Value:	1001 RRRR
Attribute:	Read/Write
Size:	8 bits

The ECPCFGA Register provides information about the ECP mode implementation. Access to this register is enabled by programming the ECR Register (ECR[7:5] = 111).

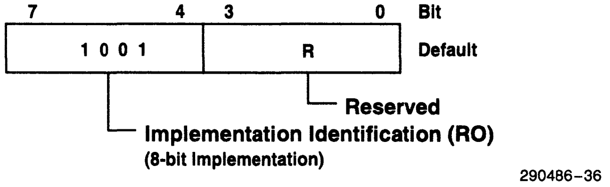


Figure 36. ECP Configuration A Register (ECP Mode)

Bit	Description
7:4	**IMPLEMENTATION IDENTIFICATION (IMPID):** This field is hardwired to 1001 to indicate an 8-bit implementation (bit 4) and an ISA-style interrupt (bit 7). This field is read only and writes have no affect.
3:0	**RESERVED**

6.1.3.8 ECPCFGB—ECP Configuration B Register (ECP Mode)

I/O Address:	Base +401h and (ECR[7:5] = 111)
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

The ECPCFGB Register is part of the ECP specification and is implemented in the 82091AA as a scratchpad register. Software can use the fields in this register to maintain system information. Programming these bits does not affect parallel port operations. Access to the ECPCFGB Register is enabled by programming the ECR Register (ECR[7:5] = 111).

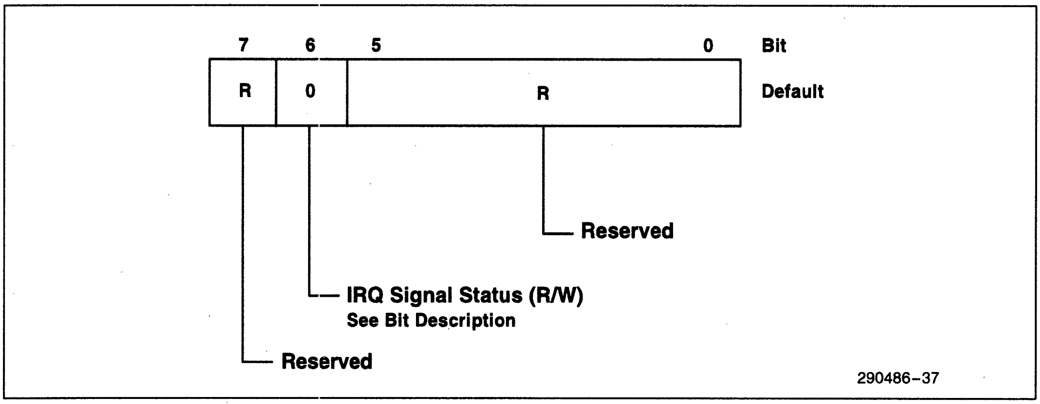


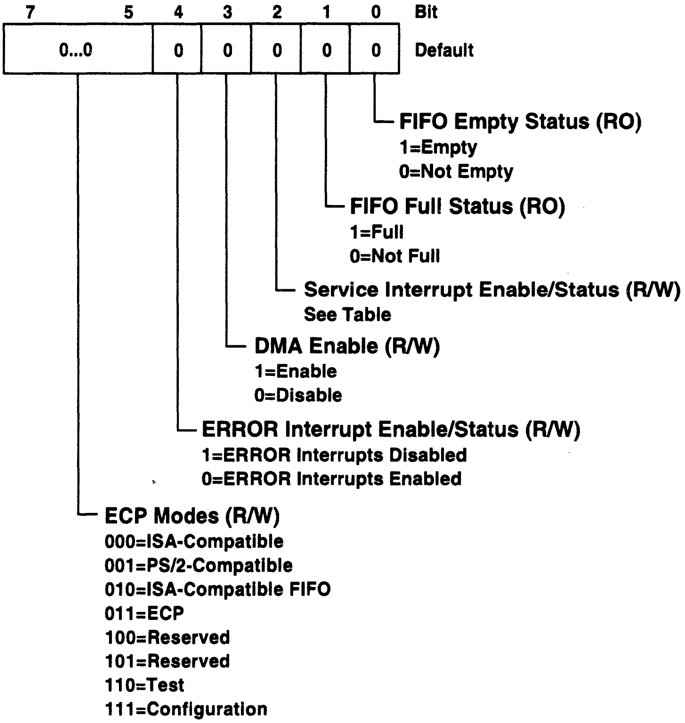
Figure 37. ECP Extended Control Register (ECP Mode)

Bit	Description
7	**RESERVED:** This bit always reads back as 0.
6	**INTRVALUE (INTRV):** This bit returns the value on the ISA IRQ line (IRQ5/IRQ7) to determine possible conflicts. The value of either IRQ5 or IRQ7 is read back based on the parallel port interrupt selection in the 82091AA configuration registers. IRQ5/IRQ7 are tri-stated in ECP configuration mode (ECR[7:5] = 111] to allow the state of the selected parallel port interrupt line to be read back. Note that the ACKINTEN bit in the PCON register must be written to 0 before the interrupt status can be read on this bit.
5:0	**RESERVED:** These bits always read back as 0.

6.1.3.9 ECR ECP—Extended Control Register (ECP Mode)

I/O Address:	Base +402h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

This register selects the ECP mode, enables service and error interrupts and provides interrupt status. The ECR also enables/disables DMA operations and provides FIFO empty and FIFO full status. The FIFO empty and FIFO full status bits are also used to report FIFO overrun and underrun conditions.



290486–38

Figure 38. ECP Extended Control Register (ECP Mode)

Bit	Description
7:5	**ECP MODE SELECT:** This field selects one of the following ECP Modes: **Mode Operation** **0 0 0** **ISA-Compatible Mode.** In this mode the parallel port operates in ISA-Compatible mode. The FIFO is reset and common collector drivers are used on the control lines (STROBE#, AUTOFD#, INIT# and SELECTIN#). Setting the direction bit to 1 in the PCON Register does not affect the parallel port interface. For register descriptions in this mode, See Section 6.1.1, ISA-Compatible and PS/2-Compatible Modes. **0 0 1** **PS/2-Compatible Mode.** In this mode the parallel port operates in PS/2-Compatible mode. The FIFO is reset and common collector drivers are used on the control lines (STROBE#, AUTOFD#, INIT# and SELECTIN#). Unlike mode 000 above, setting the direction bit to 1 in the PCON Register tri-states the data lines and reading the data register returns the value on the PD[7:0]. For register descriptions in this mode, see Section 6.1.1, ISA-Compatible and PS/2-Compatible Modes. **0 1 0** **ISA-Compatible FIFO Mode.** This mode is the same as mode 000 above, except that data is written or DMAed to the FIFO. FIFO data is automatically transmitted using the ISA-style protocol. For this mode, the direction control bit in the PCON register must be 0. **0 1 1** **ECP Mode.** In the forward direction, bytes written to the ECP DFIFO location and bytes written to the ECP AFIFO location are placed in the ECP FIFO and transmitted automatically to the peripheral using ECP protocol. In reverse direction bytes are transferred from PD[7:0] to the ECP FIFO. **1 0 0** **Reserved** **1 0 1** **Reserved** **1 1 0** **Test Mode.** In this mode, the FIFO may be written and read, but the data will not be transmitted on PD[7:0]. **1 1 1** **Configuration Mode.** In this mode, the ECP Configuration A and B Registers are accessible. ECP Mode Switching Guidelines Software will execute P1284 negotiations and all operation prior to a data transfer phase under programmed I/O (using mode 000 or 001). Hardware provides an automatic control line handshake, moving data between the FIFO and the ECP port only in the data transfer phase (using modes 011 or 010). Setting the mode to 011 or 010 causes the hardware to initiate the data transfer. If the parallel port is in mode 000 or 001, the port can be switched to any other mode. If the parallel port is not in mode 000 or 001, the port can only be switched into mode 000 or 001. The direction and the FIFO threshold can only be changed in modes 000 or 001. Note that the FIFO, FIFO Error, and TC conditions are also reset when the mode is switched to 000 or 001. Once in an extended forward mode, the software should wait for the FIFO to be empty before switching back to mode 000 or 001. In this case, all control signals are negated before the mode switch. In an ECP reverse mode the software waits for all the data to be read from the FIFO before changing to mode 000 or 001.

Bit	Description
4	**ERROR INTERRUPT DISABLE (ERRINTREN):** This bit enables error interrupts to the host. In ECP Mode, When ERRINTREN = 1, interrupts are disabled. When ERRINTREN = 0, interrupts are enabled. When enabled and a high-to-low transition occurs on the FAULT# signal (FAULT# asserted), an interrupt is generated to the host. Note that if this bit is written from a 1 to a 0 while FAULT# is asserted, an interrupt is generated to the host.
3	**DMA ENABLE (DMAEN):** This bit enables/disables DMA. When DMAEN = 1, DMA is enabled and the host uses PPDREQ, PPDACK, and TC to transfer data. When DMAEN = 0, DMA is disabled and the PPDREQ output is tri-stated. In this case, programmed I/O is used to transfer data between the host and the 82091AA FIFO. The Service Interrupt (bit 2) needs to be set to 0 to allow generation of a TC interrupt. This bit must be written to 0 to reset the TC interrupt.
2	**SERVICE INTERRUPT (SERVICEINTR):** This bit enables FIFO and TC service interrupts. When the CPU writes SERVICEINTR = 1, FIFO request interrupts, FIFO error interrupts, and TC interrupts are disabled. Setting this bit to a 0 enables interrupts for one of the four cases listed below. When enabled (set to 0) and one of the four conditions below occurs, the 82091AA sets SERVICEINTR to a 1 and generates an interrupt to the host. 1. During DMA operations (DMAEN = 1), when terminal count is reached (TC asserted). To clear the TC interrupt, switch to ISA-Compatible or PS/2-Compatible mode (write ECR[7:5] to 000, 001) or set DMAEN to 0. 2. In the forward direction and DMAEN = 0, when there is a threshold number of bytes in the FIFO to be written. 3. In the reverse direction and DMAEN = 0, when there is a threshold number of bytes in the FIFO to be read. 4. In either DMA or programmed I/O mode when there is a FIFO overrun or underrun. Reading the SERVICEINTR bit indicates the presence of an active interrupt when this bit has been written to a 0 prior to reading it back. To disable interrupts, the SERVICEINTR bit must be explicitly written to a 1. **NOTE:** The ACK# and FAULT# interrupts can be generated independent of the value of the SERVICEINTR bit. ACK# and FAULT# interrupts are enabled via the ACKINTREN bit in the PCON Register and the ERRINTREN bit in the ECR Registers, respectively. The parallel port IRQ output (IRQ5/IRQ7) is enabled when ACKINTREN = 1 in the PCON Register or when ECR[7:5] = 010, 011, or 110. Otherwise, the IRQ output is tri-stated.
1	**FIFO FULL STATUS (FIFOFS):** This bit indicates when the FIFO is full. When FIFOFS = 1 (and FIFOES = 0), the FIFO is full and cannot accept another byte of data. When FIFOFS = 0, at least one byte location is free in the FIFO. This bit is read only and writes have no affect. When a FIFO overrun or underrun occurs, the 82091AA sets both FIFOES and FIFOFS to 1. To clear the FIFO error condition interrupt, swiitch the parallel port mode from ECP (011) to either ISA-Compatible or PS/2-Compatible modes (000 or 001).
0	**FIFO EMPTY STATUS (FIFOES):** This bit indicates when the FIFO is empty. When FIFOES = 1 (and FIFOFS = 0), the FIFO is empty. When FIFOES = 0, the FIFO contains at least one byte. This bit is read only and writes have no affect. When a FIFO overrun or underrun occurs, the 82091AA sets both FIFOES and FIFOFS to 1. To clear the FIFO error condition interrupt, swiitch the parallel port mode from ECP (011) to either ISA-Compatible or PS/2-Compatible modes (000 or 001).

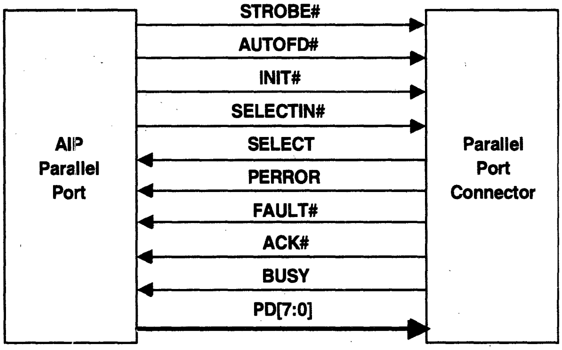
6.2 Parallel Port Operations

The parallel port can be placed in ISA-Compatible, PS/2-Compatible, or EPP mode by hardware configuration or by writing to the 82091AA's configuration registers (PCFG1 Register). If access to the configuration registers is not disabled by hardware configuration, a hardware configured parallel port mode can be changed by programming the PCFG1 Register.

ECP mode is entered by programming the ECP Extended Control Register (ECR). Writing to this register changes any previously selected parallel port mode (via hardware configuration or writing the PCFG1 Register) to one of the ECP ECR Register modes selected via ECR[7:5]. Note that ECP mode cannot be entered by hardware configuration or programming the 82091AA configuration registers.

6.2.1 ISA-COMPATIBLE AND PS/2-COMPATIBLE MODES

The ISA-Compatible mode is used for standard ISA-type parallel port interfaces. Figure 39 shows the parallel port interface for ISA-Compatible mode. STROBE#, AUTOFD#, INIT#, and SELECTIN# are controlled by software via the PCON Register and the status of SELECT, PERROR, FAULT, ACK#, and BUSY are reported in the PSTAT Register. PD[7:0] are outputs only. Note that for a reverse data transfer using the Nibble protocol, the peripheral device supplies data, 4 bits at a time, using the BUSY, SELECT, PERROR, and FAULT# signals.



290486-39

Figure 39. ISA-Compatible Mode

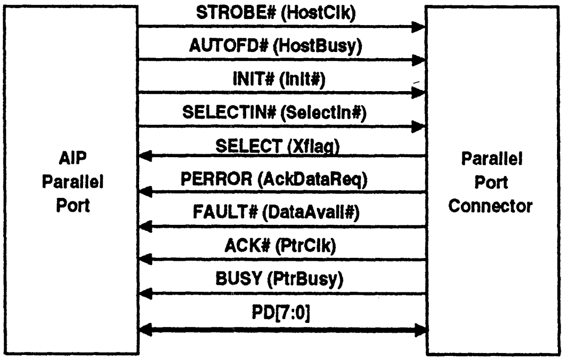
The following general protocol is used when communicating with a printer or other parallel port device.

Software selects the peripheral device by asserting the SELECTIN# signal. The peripheral device, in turn, acknowledges that it is selected by asserting the SELECT signal. The INIT# signal is used to initialize the peripheral device. If an error is encountered during initialization or normal operations, the peripheral device asserts FAULT#. If a printer (or plotter) encounters problems in the paper path, the device asserts PERROR. Other peripheral devices may not use the PERROR signal.

During normal operation, the peripheral device asserts BUSY when it is not ready to receive data. When it has finished processing the previous data, the peripheral device asserts ACK# and negates BUSY. If interrupts are enabled, a low-to-high transition on ACK# when the signal is negated generates an interrupt. If interrupts are not enabled, software must poll the PSTAT Register to determine when ACK# is pulsed.

The parallel port driver software sends data to the peripheral device by writing to the PDATA Register and asserting the STROBE# signal after an appropriate data stabilization interval. After a sufficient setup time has elapsed, software then negates STROBE#. Valid data is read by the peripheral device.

In the PS/2-Compatible mode, data can be written to or read from the parallel port. Figure 40 shows the parallel port interface for PS/2-Compatible mode. The Byte protocol signal names are shown in parenthesis. Before reading or writing the PDATA Register, the direction control bit in the PCON Register must be set to the proper transfer direction on PD[7:0]. During a write to the PDATA Register (with DIR# = 0), data is latched by the PDATA Register and driven onto PD[7:0]. During a parallel port read of the PDATA Register (with DIR# = 1), the data on PD[7:0] is driven onto SD[7:0]. The data is not latched by the PDATA Register during the read cycle.

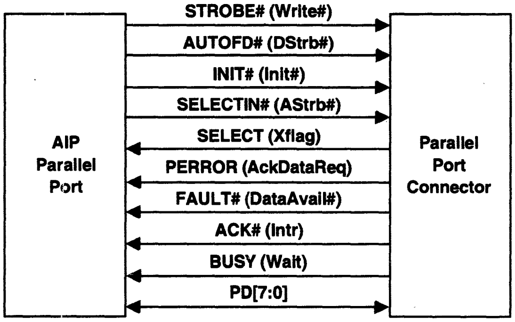


290486-40

Figure 40. PS/2-Compatible Mode

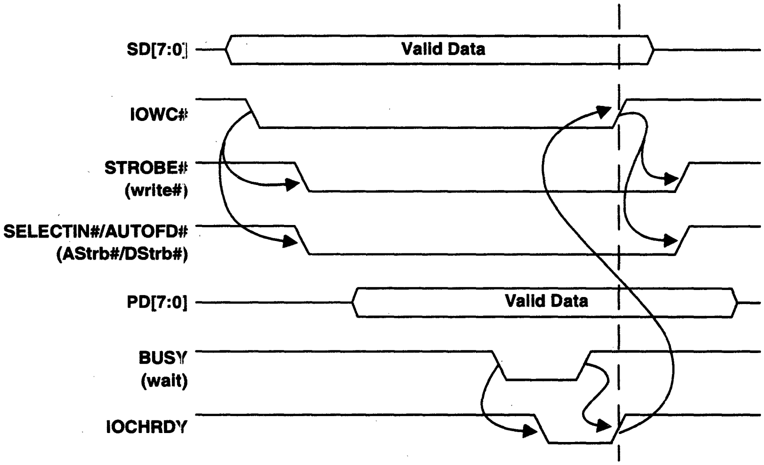
6.2.2 EPP MODE

The 82091AA is EPP 1.7 compliant. This means EPP cycles will begin with WAIT (Busy) inactive, however, WAIT will still prolong the cycle when active. Figure 41 shows the parallel port interface for EPP mode. The EPP parallel port interface protocol signal names are shown in parenthesis. In EPP mode, the initialization, printer selection, and error signals (PERROR and FAULT#) work the same way as in the ISA-Compatible mode. However, in EPP mode, SELECTIN# and AUTOFD# are automatically generated and become Address Strobe (AStrb#) and Data Strobe (DStrb#), respectively, enabling direct access to parallel port devices. STROBE (Write#) is used to indicate a read or write cycle. Note that BUSY (Wait) is an active low signal in EPP mode rather than an active high signal as in ISA-Compatible mode. In addition, BUSY, in combination with IOCHRDY on the ISA Bus extends clock cycles to enable the completion of a read or write without additional wait states. EPP write and read cycles are shown in Figure 42 and Figure 43.



290486-41

Figure 41. EPP Mode



290486-42

Figure 42. EPP Mode Write Cycle

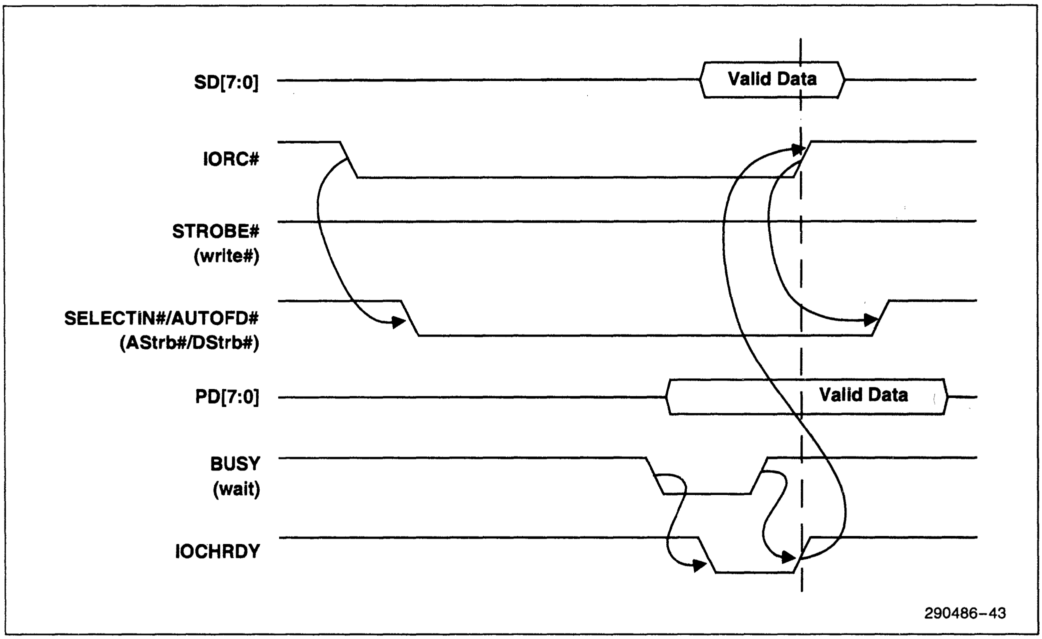


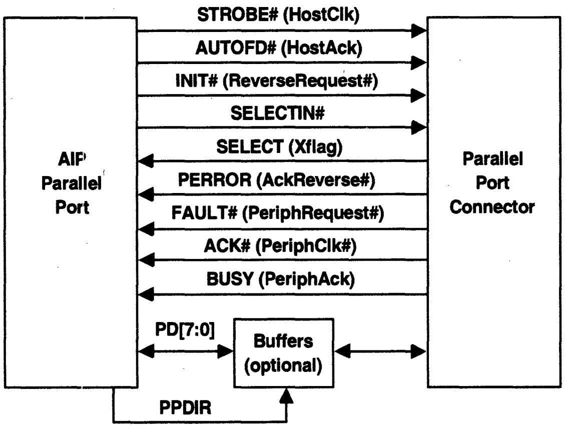
Figure 43. EPP Mode Read Cycle

6.2.3 ECP MODE

Figure 44 shows the parallel port interface for ECP mode with the ECP protocol signal names in parenthesis. The ECP modes are selected by programming the Extended Control Register (ECR bits[7:5]). Two of the modes (Test and Configuration) provide information about the 82091AA's parallel port and are not used for interfacing with a peripheral device. Four peripheral interface modes are selectable via the ECR—ISA-Compatible mode, ISA-Compatible FIFO mode, PS/2-Compatible mode, and ECP mode.

ISA-Compatible and PS/2-Compatible Modes (ECR[7:5] = 000,001)

The ISA-Compatible and PS/2-Compatible mode selections in the ECR are equivalent to selecting these modes via hardware configuration or programming the PCFG1 Register. For these modes the operation is the same as described in Section 6.2.1, ISA-Compatible and PS/2-Compatible Modes.



290486–44

Figure 44. ECP Mode

ISA-Compatible FIFO Mode (ECR[7:5] = 010)

The ISA-Compatible FIFO mode uses the same signaling protocol on the parallel port interface as the ISA-Compatible mode. However, there are two major operational differences. First, data is written to a 16-byte FIFO (via the SDFIFO address location). The FIFO empty and FIFO full bits in the ECR provide FIFO status. In addition, DMA can be used to transfer data to the FIFO by enabling this feature in the ECR.

Second, the data is transferred to the peripheral using an automatic hardware handshake. This handshake emulates the standard ISA-Compatible style software generated handshake (Figure 45). For ISA-Compatible FIFO mode, the 82091AA does not monitor the ACK# signal. Service interrupts are enabled and reported via the ECR. The generation of service interrupts is based on the state of the FIFO and not individual transfers (using ACK#) as is the case in standard ISA-Compatible mode.

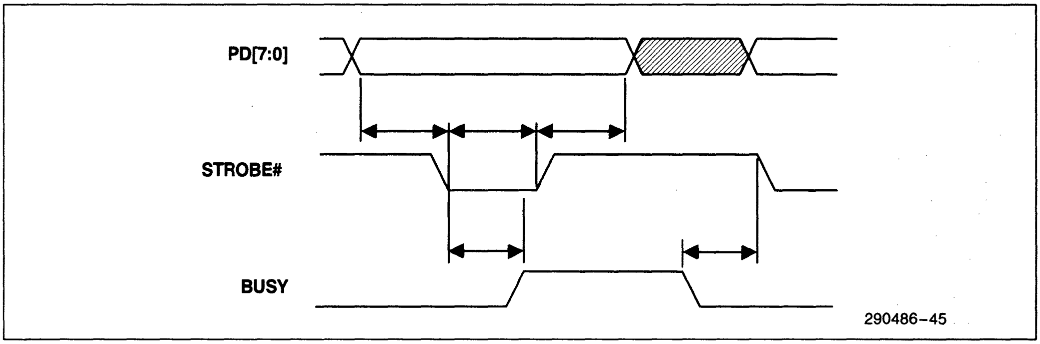


Figure 45. ISA-Compatible Timing

ECP Mode (ECR[7:5] = 011)

When ECR[7:5] = 011, the parallel port operates in ECP mode. In ECP mode, both data and commands (addresses and RLE) are transferred using the parallel port 16-byte FIFO. This information can be either written to or read from the FIFO using DMA or non-DMA ISA Bus transfers. The parallel port interface transfers use an automatic handshake generated by the 82091AA. The host controls the transfer direction by programming the DIR# bit in the PCON Register.

When the host is writing to the peripheral device (forward direction), STROBE#, and BUSY provide the automatic handshake for transfer on the parallel port interface (Figure 46). The peripheral device negates BUSY when it is ready to receive data or commands. AUTOFD# indicates whether PD[7:0] contain data (AUTOFD# is high) or a command (AUTOFD# is low). For commands (address or RLE), the host writes to the ECPAFIFO Register I/O address and for data, the host writes to the DFIFO Register I/O address. The addresses and data are placed in the same 16-byte FIFO. When the FIFO is full and cannot accept more data/addresses, the FIFO Full status bit is set in the ECR.

Data/addresses written to the FIFO are transferred to the peripheral device via PD[7:0]. To begin a transfer on the peripheral interface, the 82091AA checks BUSY to make sure the peripheral is in the ready state. If BUSY is negated, the 82091AA drives PD[7:0] and AUTOFD#, and asserts STROBE# to indicate that the data/command is on PD[7:0]. The peripheral device asserts BUSY to indicate that it is receiving the data/command. BUSY asserted causes the 82091AA to negate STROBE#.

When the host is reading from the peripheral device (reverse direction), AUTOFD# and ACK# provide the automatic handshake for transfer on the parallel port interface (Figure 47). Data/commands from the peripheral device are placed in the parallel port FIFO using this handshake. In this case, BUSY indicates whether PD[7:0] contain data (BUSY is high) or a command (BUSY is low).

The peripheral device asserts ACK# to indicate that a data/command is on PD[7:0]. The 82091AA negates AUTOFD# when it is ready for a peripheral transfer and asserts AUTOFD# to indicate that it is receiving the data/command. AUTOFD# asserted causes the peripheral device to negate ACK#. The peripheral transfers are to the parallel port 16-byte FIFO.

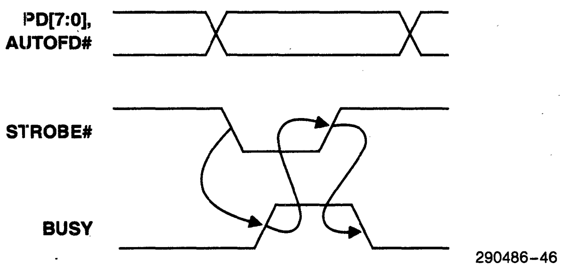


Figure 46. ECP Mode Handshake (Forward Direction)

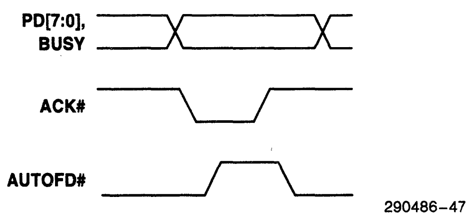


Figure 47. ECP Mode Handshake (Reverse Direction)

Test Mode (ECR[7:5] = 110) and Configuration Mode (ECR7:5] = 111)

The test mode can be used to check the FIFO read and write interrupt thresholds as described in Section 6.1.3.7, TFIFO—ECP Test FIFO Register. Note that for the 82091AA parallel port, the read and write FIFO interrupt thresholds are the same. The FIFO threshold is set by programming the PCFG1 Register in the 82091AA configuration space. The configuration mode is used to access the ECPCFGA and ECPCFGB Registers. This mode must first be set before the ECPCFGA and ECPCFGB Registers can be accessed.

6.2.3.1 FIFO Operations

The parallel port FIFO is used for ECP transfers (ECR[7:5] = 011), ISA-Compatible FIFO transfers (ECR[7:5] = 010), and Test mode (ECR[7:5] = 110). Either DMA or programmed I/O can be used for transfers between the host and the parallel port.

The FIFO threshold value is selected via the 82091AA configuration registers (PCFG1 Register). The threshold is set to either 1 (forward)/15 (reverse) or 8 in both directions. A threshold setting of 1 (forward)/15 (reverse) results in longer periods of time between service request, but requires faster servicing of both read and write requests. A threshold setting of 8 results in more service requests, but tolerates slower servicing of the requests.

In modes 010 and 011, an internal temporary holding register is used in conjunction with the 16-byte FIFO to provide 17 bytes of storage for both forward and reverse transfers. Thus, in the forward direction if the peripheral asserts the BUSY signal during the filling of the FIFO, the host needs to write 17 bytes before the FIFO full flag in the ECR is set to 1. In Test mode (110) only the 16-byte FIFO is used and the temporary holding register is not used.

The FIFO is reset by a hard reset (RSTDRV asserted) or whenever the parallel port is placed in ISA-Compatible or PS/2-Compatible modes. Note that the FIFO threshold can only be changed when the parallel port is in ISA-Compatible or PS/2-Compatible mode.

6.2.3.2 DMA Transfers

The 82091AA contains parallel port DMA request (PPDREQ) and acknowledge (PPDACK#) signals to communicate with a standard PC DMA controller. Before initiating a DMA transfer the direction bit in the PCON Register must be set to the proper direction. To initiate DMA transfers, software sets the DMAEN bit to 1 and the SERVICEINTR bit to 0 in the ECR. The PPDREQ and PPDACK# signals will then be used to fill (forward direction) or empty (reverse direction) the FIFO. When the DMA controller reaches terminal count and asserts the TC signal, an interrupt is generated and the SERVICEINTR bit is set to 1. To reset the TC interrupt, software can either switch the mode to 000 or 001 or write the DMAEN bit to 0.

In DMA mode, if 32 consecutive reads or writes are performed to the FIFO and PPDREQ is still asserted, the 82091AA negates PPDREQ for the length of the last PPDACK#/command pulse to force an arbitration cycle on the ISA Bus.

6.2.3.3 Reset FIFO and DMA Terminal Count Interrupt

The following operations are used to reset the parallel port FIFO and TC interrupt

Function	Reset Operations
FIFO	-Changing to modes 000 or 001 -Hard reset
FIFO Error	-Changing to modes 000 or 001 -Hard reset
TC Interrupt	-Changing to modes 000 or 001 -Setting DMAEN to 0 in ECR -Hard reset

6.2.3.4 Programmed I/O Transfers

Programmed I/O (non-DMA) can also be used for transfers between the host and the parallel port FIFO. Software can determine the read/write FIFO thresholds and the FIFO depth by accessing the FIFO in Test mode. To use programmed I/O transfers software sets the direction bit in the PCON Register to the desired direction and sets the DMAEN bit to 0 and the SERVICEINTR bit to 0 in the ECR. The parallel port requests programmed I/O transfers from the host by asserting IRQ5/IRQ7.

In the reverse direction an interrupt occurs when SERVICEINTR = 0 either 8 or 15 bytes (depending on threshold setting) are in the FIFO. IRQ5/IRQ7 can be used in an interrupt-driven system. The host must respond to the interrupt request by reading data from the FIFO.

In the forward direction an interrupt occurs when SERVICEINTR = 0 and there are either 8 or 1 byte locations available in the FIFO (depending on threshold setting). IRQ5/IRQ7 can be used in an interrupt-driven system. The host must respond to the interrupt request by writing data to the FIFO.

6.2.3.5 Data Compression

The 82091AA supports Run Length Encoded (RLE) decompression in hardware and can transfer compressed data to the peripheral. To transfer compressed data to the peripheral (forward direction), the compression count is written to the ECPAFIFO location and the data is written to the ECPDFIFO location. The most significant bit (bit 7) in the byte written to the ECPAFIFO Register informs the peripheral whether the value is a channel address (bit 7 = 1) or a run length count (bit 7 = 0). The RLE count in the ECPAFIFO (bits[6:0]) informs the peripheral of how many times the data in the ECPDFIFO is to be repeated. An RLE count of 0 indicates that only one byte of the data is present and a count of 127 indicates to the peripheral that the next byte should be expanded 128 times. An RLE count of 1 should be avoided as it will cause unnecessary expansions. Note that the 82091AA asserts AUTOFD# to indicate that PD[7:0] contains address/RLE instead of data.

In the reverse direction, the peripheral negates the BUSY signal to indicate that PD[7:0] contains address/RLE. During an address/RLE cycle, the 82091AA checks bit 7 to see if the next byte received needs to be decompressed. If bit 7 is 0, the 82091AA decompresses (replicates) the next data received by the RLE count received on bits[6:0].

6.2.4 PARALLEL PORT EXTERNAL BUFFER CONTROL

A multi-function signal (GCS#/PPDIR) is provided for controlling optional external parallel port data buffers. The PPDIR function is only available when the 82091AA configuration is in software motherboard (SWMB) mode. In this mode, this signal operates as a parallel port direction control signal (PPDIR). Note that, if any other configuration is used (SWAI, HWB, or HWE configuration modes), this multi-function signal operates as a game port chip select (GCS#). In SWMB, PPDIR is low when PD[7:0] are outputs and high when PD[7:0] are inputs. Figure 44 shows an example of external buffers being used when the parallel port is in ECP mode.

External buffering affects the ability of the port to read software security devices. Typically these software secutiry devices are designed to hold the data pins of the parallel port connector at either high or low logic levels when the pins are not being driven by the parallel port. The bit pattern read from the parallel port by the security software may not be correctly transferred through the external buffer.

6.2.5 PARALLEL PORT SUMMARY

Table 18 summarizes the parallel port interrupt, DMA, and parallel port signal drive type for the various modes of operation.

Table 18. Parallel Port Summary

Parallel Port Mode	ECR[7:5]	PD[7:0] Direction	Parallel Port Control Signals Controlled By PCON	IRQ Enable	DMA Enable
ISA-Compatible	000	Output	Open Drain	ACKINTEN	
PS/2-Compatible	001	Bi-directional	Open Drain	ACKINTEN	
EPP	N/A	Bi-directional	Push Pull	ACKINTEN	
ISA-Compatible FIFO	010	Output	Push Pull	Always Enabled	DMAEN
ECP	011	Bi-directional	Push Pull	Always Enabled	DMAEN
ECP Test	110	Output	Push Pull	Always Enabled	DMAEN
ECP Configuration	111	Bi-directional	Push Pull	ACKINTEN	DMAEN

NOTES:

1. The selected IRQ pin (IRQ5/IRQ7) is enabled if ACKINTEN is enabled in the PCON Register. Otherwise, the IRQ pin is tri-stated.
2. PPDREQ is enabled whenever the DMAEN bit is enabled in the ECR, independent of the parallel port mode.

7.0 SERIAL PORT

The two 82091AA serial ports are identical. This section describes the serial port registers and FIFO operations.

7.1 Register Description

The register descriptions in this section apply to both serial port A and serial port B and provide a complete operational description of the serial ports. Table 19 shows the I/O address assignments for the serial port registers. The individual register descriptions follow in the order that they appear in the table. Note that serial port interrupt assignments (IRQ3 or IRQ4) and the base address assignments are made by 82091AA configuration as described in Section 4.0, AIP Configuration.

All registers are accessed as byte quantities. The base address is determined by hardware configuration at powerup (or a hard reset) or via software configuration by programming the 82091AA configuration registers as described in Section 4.0, AIP Configuration. Note that access to certain serial port registers requires prior programming of the DLAB bit in the Line Control Register (LCR).

During a hard reset (RSTDRV asserted), the 82091AA registers are set to pre-determined **default** states. The default values are indicated in the individual register descriptions. Reserved bits in the 82091AA's serial port registers must be programmed to 0 when writing the register and these bits are 0 when read. The following bit notation is used for default settings:

X Default bit position value is determined by conditions on an 82091AA signal pin.

The following nomenclature is used for serial port register access attributes:

RO **Read Only**. Note that for all registers with read only attributes, writes to the I/O address access a different register.

WO **Write Only**. Note that for all registers with write only attributes, reads to the I/O address access a different register.

R/W **Read/Write**. A register with this attribute can be read and written. Note that some read/write registers contain bits that are read only.

Table 19. Serial Port Registers

Register Address Access (AEN = 0)		Abbreviation	Register Name	Access
Base +	DLAB			
0h	0	THR	Transmit Holding Register	WO
0h	0	RBR	Receiver Buffer Register	RO
0h	1	DLL	Divisor Latch LSB	R/W
1h	1	DLM	Divisor Latch MSB	R/W
1h	0	IER	Interrupt Enable Register	R/W
2h	—	IIR	Interrupt Identification Register	RO
2h	—	FCR	FIFO Control Register	WO
3h	—	LCR	Line Control Register	R/W
4h	—	MCR	Modem Control Register	R/W
5h	—	LSR	Line Status Register	R/W
6h	—	MSR	Modem Status Register	R/W
7h		SCR	Scratch Pad Register	R/W

Table 20. Serial Port Register Summary

Bit #	Receiver Buffer Register	Transmitter Holding Register	Interrupt Enable Register	Interrupt Identification Register	FIFO Control Register	Line Control Register
0	Data Bit 0	Data Bit 0	Enable Received Data Available Interrupt	0 if Interrupt Pending	FIFO Enable	Word Length Select Bit 0
1	Data Bit 1	Data Bit 1	Enable XMTR Holding Register Empty Interrupt	Interrupt ID Bit	RCVR FIFO Reset	Word Length Select Bit 1
2	Data Bit 2	Data Bit 2	Enable RCVR Line Status Interrupt	Interrupt ID Bit	XMIT FIFO Reset	Number of Stop Bits
3	Data Bit 3	Data Bit 3	Enable Modem Status Interrupt	Interrupt ID Bit (Non-FIFO = 0)	DMA Mode Select	Parity Enable
4	Data Bit 4	Data Bit 4	0	0	Reserved	Event Parity Select
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Non-FIFO = 0)	RCVR Trigger (LSB)	Set Break
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Non-FIFO = 0)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)

Table 20. Serial Port Register Summary (Continued)

Bit #	Modem Control Register	Line Status Register	Modem Status Register	ScratchPad Register	Divisor Latch - MSB	Divisor Latch - LSB
0	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send	Bit 0	Bit 0	Bit 8
1	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready	Bit 1	Bit 1	Bit 9
2	Out 1 Bit	Parity Error (PE)	Trailing Edge Ring Indicator	Bit 2	Bit 2	Bit 10
3	IRQ Enable	Framing Error (FE)	Delta Data Carrier Detect	Bit 3	Bit 3	Bit 11
4	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	0	Error in RCVR FIFO	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

7.1.1 THR(A,B)—TRANSMITTER HOLDING REGISTER

I/O Address:	Base +0h (DLAB=0)
Default Value:	00h
Attribute:	Write Only
Size:	8 bits

The THR contains data to be transmitted out on the SOUT[A,B] signal line. Bit 0 is the least significant bit and is the first bit serially transmitted. If the serial word length is less than 8 bits (as selected in the LCR), the data word must be written to this register right-justified. Bit positions above the number of bits selected for the word size are discarded (not transmitted).

Bit	Description
7:0	**Transmit Data:** Bits[7:0] correspond to SD[7:0].

7.1.2 RBR(A,B)—RECEIVER BUFFER REGISTER

I/O Address:	Base +0h (DLAB=0)
Default Value:	00h
Attribute:	Read Only
Size:	8 bits

The RRB contains data received from the SIN[A,B] signal line. Bit 0 is the least significant bit and is the first bit serially received. If the serial word length is less than 8 bits (as selected in the LCR), the data word in this register is right-justified. Bit positions above the number of bits selected for the word size are 0.

Bit	Description
7:0	**Receiver Data:** Bits[7:0] correspond to SD[7:0].

7.1.3 DLL(A,B), DLM(A,B)—DIVISOR LATCHES (LSB AND MSB) REGISTERS

I/O Address:	Base +0h,1h (DLAB=1)
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

The 82091AA contains two independently programmable baud rate generators. The 24 MHz crystal oscillator frequency input is divided by 13, resulting in a frequency of 1.8462 MHz. This frequency is the input to each baud rate generator and is divided by the divisor of the associated serial port. The output frequency of the baud rate generator (BOUT[A,B]) is 16 x the baud rate.

divisor # = (frequency input)/(baud rate x 16)

The output of each baud rate generator drives the transmitter and receiver sections of the associated serial port. Two 8-bit latches per serial port store the divisor in a 16-bit binary format. These divisor latches must be loaded during initialization to ensure proper operation of the baud rate generator. Upon loading either of the divisor latches, a 16-bit baud counter is loaded. Table 21 provides decimal divisors to use with crystal frequencies of 24 MHz. Using a divisor of zero is not recommended.

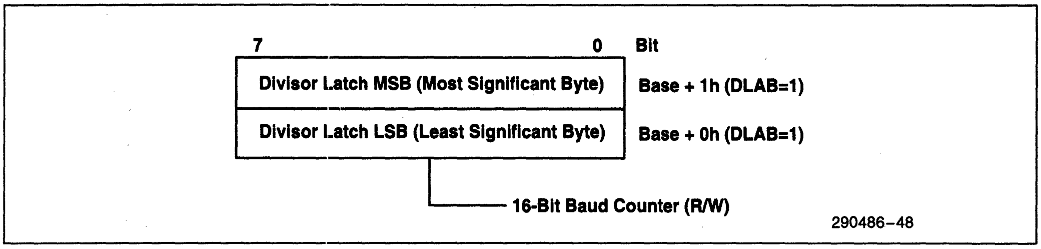

7
0
Bit
Divisor Latch MSB (Most Significant Byte)
Base + 1h (DLAB=1)
Divisor Latch LSB (Least Significant Byte)
Base + 0h (DLAB=1)
16-Bit Baud Counter (R/W)
290486-48

Figure 48. Divisor Latches (LSB and MSB) Registers

Bit	Description
7:0	**Divisor Latch Data:** Bits[7:0] correspond to SD[7:0].

Table 21. AIP Serial Port A and B Divisors, Baud Rates, and Clock Frequencies

	24 MHz Input Divided to 1.8461 MHz	
Baud Rate	**Decimal Divisor for 16x Clock**	**Percent Error**
50	2304	0.1
75	1536	
110	1047	
134.5	857	0.4
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.5
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	3.0
115200	1	—

7.1.4 IER(A,B)—INTERRUPT ENABLE REGISTER

I/O Address:	Base + 1h (DLAB = 0)
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

This register enables/disables interrupts for five types of serial port conditions. If a particular condition occurs whose interrupt is disabled in this register, the corresponding interrupt status bit in the IIR will not be set and an interrupt request (IRQ3 or IRQ4) will not be generated.

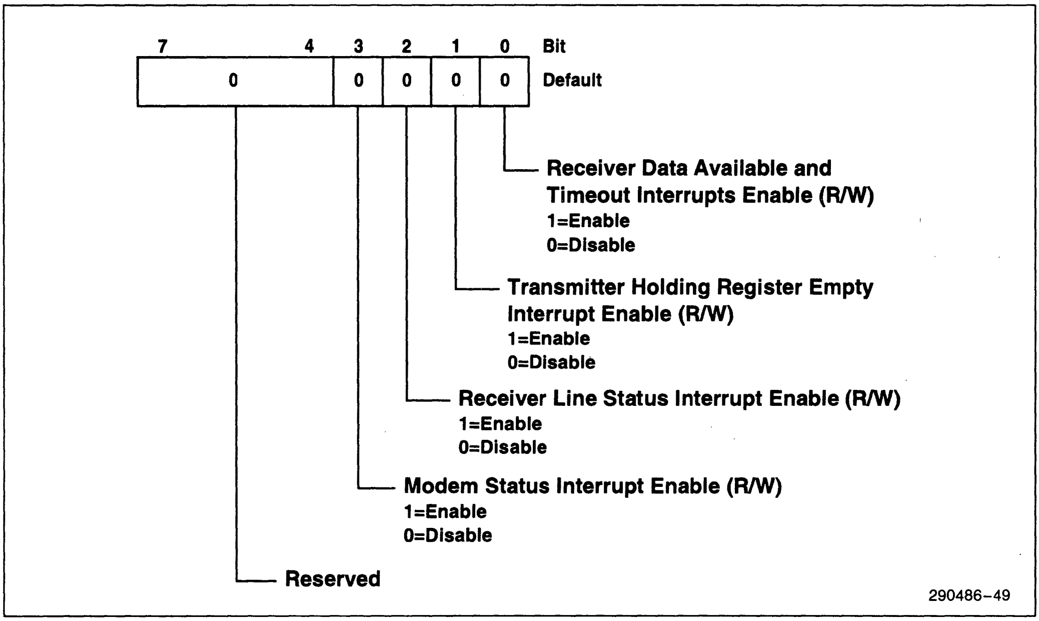

7
4
3
2
1
0
Bit
0
0
0
0
0
Default
Receiver Data Available and Timeout Interrupts Enable (R/W)
1=Enable
0=Disable
Transmitter Holding Register Empty Interrupt Enable (R/W)
1=Enable
0=Disable
Receiver Line Status Interrupt Enable (R/W)
1=Enable
0=Disable
Modem Status Interrupt Enable (R/W)
1=Enable
0=Disable
Reserved
290486-49

Figure 49. Interrupt Enable Register

Bit	Description
7:4	**RESERVED**
3	**MODEM INTERRUPT ENABLE (MIE):** When MIE = 1, the Modem Status Interrupt is enabled. When MIE = 0, the Modem Status Interrupt is disabled.
2	**RECEIVER INTERRUPT ENABLE (RIE):** When RIE = 1, the Receiver Line Status interrupt is enabled. When RIE = 0, the receiver line status interrupt is disabled.
1	**TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT ENABLE (THEIE):** When THREIE = 1, the Transmitter Holding Register Empty Interrupt is enabled. When THREIE = 0, the Transmitter Holding Register Empty Interrupt is disabled.
0	**RECEIVER DATA AVAILABLE INTERRUPT ENABLE AND TIMEOUT INTERRUPT ENABLE IN FIFO MODE (RAVIE):** When RAVIE = 1, the Received Data Available Interrupt is Enabled. When RAVIE = 0, the Received Data Available Interrupt is disabled. In addition, in the FIFO Mode, this bit enables the Timeout Interrupt when set to 1 and disables the Timeout Interrupt when set to 0.

7.1.5 IIR(A,B)—INTERRUPT IDENTIFICATION REGISTER

I/O Address:	Base +2h
Default Value:	01h
Attribute:	Read Only
Size:	8 bits

This register provides interrupt status and indicates whether the serial port receive/transmit FIFOs are enabled (FIFO mode) or disabled (non-FIFO mode). In order to provide minimum software overhead during data character transfers, the serial port prioritizes interrupts into four levels and records these in the Interrupt Identification Register. The four levels of interrupt conditions in order of priority are Receiver Line Status; Received Data Ready; Transmitter Holding Register Empty; and Modem Status. When the CPU accesses the IIR, the serial port freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the serial port records new interrupts, but does not change its current indication until the current access is complete.

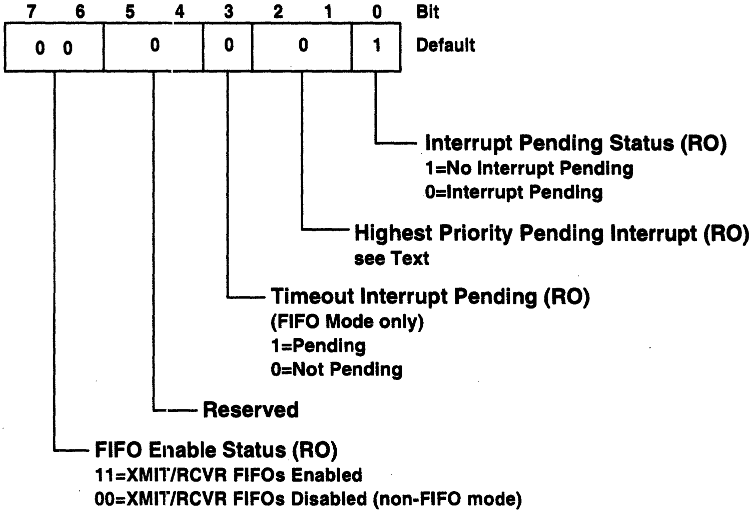

7 6 5 4 3 2 1 0 Bit
0 0 | 0 | 0 | 0 | 1 Default
Interrupt Pending Status (RO)
1=No Interrupt Pending
0=Interrupt Pending
Highest Priority Pending Interrupt (RO)
see Text
Timeout Interrupt Pending (RO)
(FIFO Mode only)
1=Pending
0=Not Pending
Reserved
FIFO Enable Status (RO)
11=XMIT/RCVR FIFOs Enabled
00=XMIT/RCVR FIFOs Disabled (non-FIFO mode)
290486-50

Figure 50. Interrupt Identification Register

Bit	Description
7:6	**FIFO MODE ENABLE STATUS (FIFOES):** This status field indicates whether the serial port is in FIFO mode or non-FIFO mode (FIFO/non-FIFO mode is selected via the FCR). When FIFOES = 11, the serial port is in FIFO mode (FIFOs enabled). When FIFOES = 00, the serial port is in non-FIFO mode (FIFOs disabled). The 82091AA never sets this field to either = 01 or 10.
5:4	**RESERVED**
3	**TIMEOUT INTERRUPT PENDING (TOUTIP)—FIFO MODE ONLY:** In the non-FIFO mode, this bit is 0. In FIFO mode TOUTIP is set to 1 when no characters have been removed from or input to the receive FIFO during the last 4 character times and there is at least 1 character in the FIFO during this time. When a timeout interrupt is pending, the 82091AA sets this bit along with bit 2 of this register.
2:1	**HIGHEST PRIORITY INTERRUPT INDICATOR:** This field identifies the highest priority interrupt pending as indicated in Table 22.
0	**INTERRUPT PENDING STATUS (IPS):** This bit can be used in an interrupt environment to indicate whether an interrupt condition is pending. When IPS = 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When IPS = 1, no interrupt is pending.

Table 22. Interrupt Priority

FIFO Mode Only	Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	—	None	None	—
0	1	1	0	Highest	Receiver Line Status	Overrun Error, Parity Error, Framing Error, or Break Interrupt	Reading the Line Status Register
0	1	0	0	Second	Received Data Available	Receiver Data Available	Read Receiver Buffer
1	1	0	0	Second	Character Timeout Indication	No Characters Have Been Removed from or Input to the RCVR FIFO during the Last 4 Char. Times and there is at least 1 Char. in it during this time	Reading the Receiver Buffer Register
0	0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if Source or Interrupt) or Writing the Transmitter Holding Register
0	0	0	0	Fourth	Modem Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect.	Reading the Modem Status Register

7.1.6 FCR(A,B)—FIFO CONTROL REGISTER

I/O Address:	Base +2h
Default Value:	00h
Attribute:	Write Only
Size:	8 bits

FCR is a write only register that is located at the same address as the IIR (the IIR is a read only register). FCR enables/disables the transmit/receive FIFOs, clears the transmit/receive FIFOs, and sets the receive FIFO trigger level.

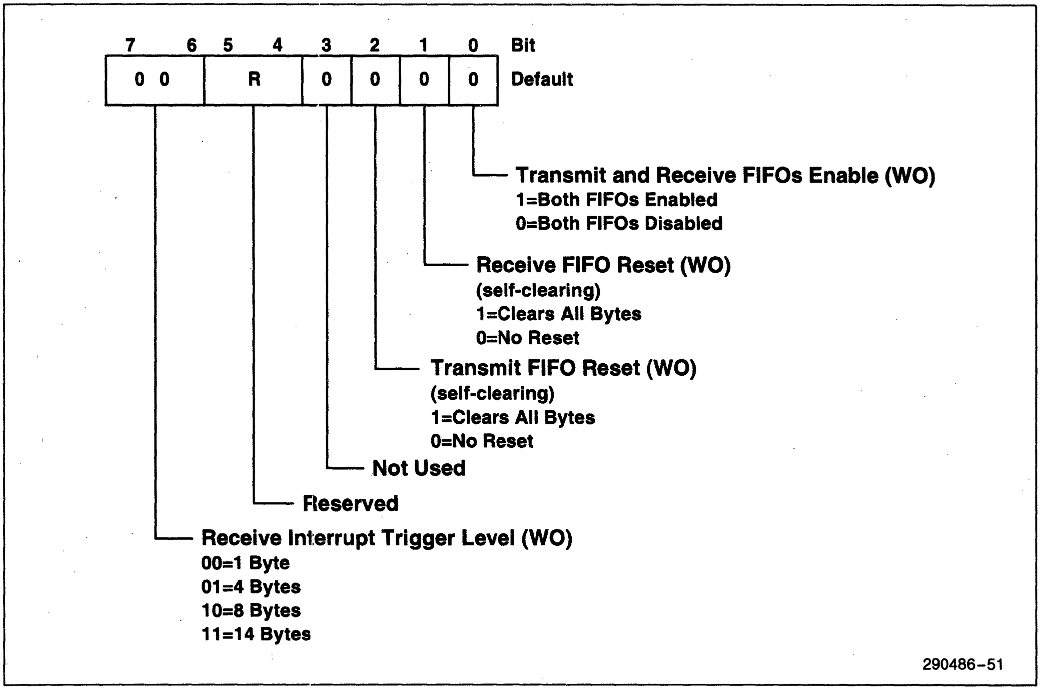


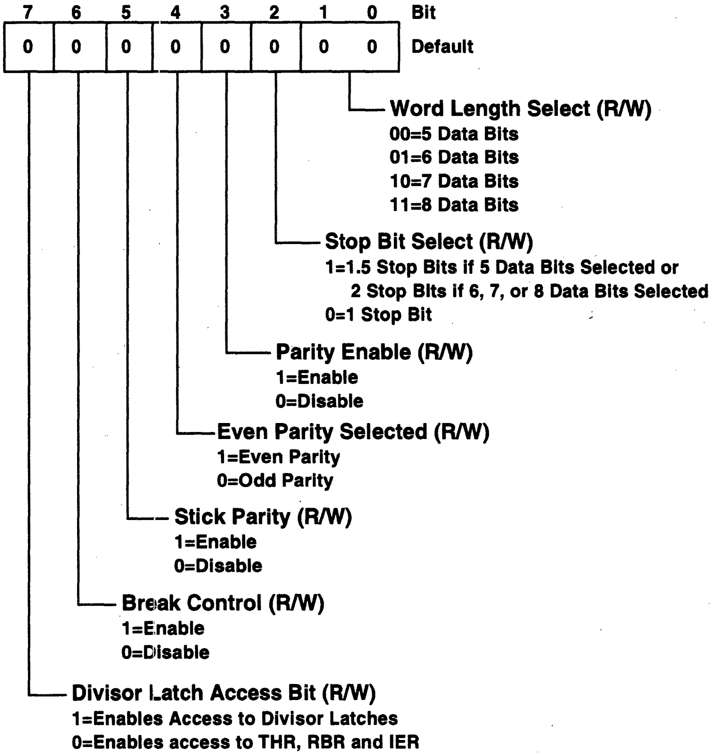
Figure 51. FIFO Control Register

Bit	Description
7:6	**INTERRUPT TRIGGER LEVEL (ITL):** The ITL field indicates the interrupt trigger level. When the number of bytes in the receive FIFO equals the interrupt trigger level programmed into this field and the Received Data Available Interrupt enabled (via the IER), an interrupt will be generated and the appropriate bits set in the IIR. **Bits [7:6] Trigger Level (Bytes)** 0 0 01 (default) 0 1 04 1 0 08 1 1 14
5:4	**RESERVED**
3	**NOT USED:** Writing to this bit causes no change in serial port operations. The serial port does not support DMA operations. Note that the TXRDY# and RXRDY# pins are not available in the 82091AA.
2	**RESET TRANSMITTER FIFO (RESETTF):** When RESETTF is set to a 1, the FIFO counter is set to 0. The shift register is not cleared. When the FIFO is cleared, the 82091AA sets this bit to 0.
1	**RESET RECEIVER FIFO (RESETRF):** When RESETRF is set to a 1, the FIFO counter is set to 0. The shift register is not cleared. When the FIFO is cleared, the 82091AA sets this bit to 0.
0	**TRANSMIT AND RECEIVE FIFO ENABLE (TRFIFOE):** TRFIFOE enables/disables the transmit and receive FIFOs. When TRFIFOE = 1, both FIFOs are enabled (FIFO Mode). When TRFIFOE = 0, the FIFOs are both disabled (non-FIFO MODE). Writing a 0 to this bit clears all bytes in both FIFOs. When changing from FIFO mode to non-FIFO mode and vice versa, data is automatically cleared from the FIFOs. This bit must be written with a 1 when other bits in this register are written or the other bits will not be programmed.

7.1.7 LCR(A,B)—LINE CONTROL REGISTER

I/O Address:	Base +3h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

This register specifies the format of the asynchronous data communications exchange. LCR also enables/disables access to the Baud Rate Generator Divisor latches or the Transmitter Data Holding Register, Receiver Buffer Register, and Interrupt Enable Register.



7	6	5	4	3	2	1	0	Bit
0	0	0	0	0	0	0	0	Default

Word Length Select (R/W)
00=5 Data Bits
01=6 Data Bits
10=7 Data Bits
11=8 Data Bits

Stop Bit Select (R/W)
1=1.5 Stop Bits if 5 Data Bits Selected or
2 Stop Bits if 6, 7, or 8 Data Bits Selected
0=1 Stop Bit

Parity Enable (R/W)
1=Enable
0=Disable

Even Parity Selected (R/W)
1=Even Parity
0=Odd Parity

Stick Parity (R/W)
1=Enable
0=Disable

Break Control (R/W)
1=Enable
0=Disable

Divisor Latch Access Bit (R/W)
1=Enables Access to Divisor Latches
0=Enables access to THR, RBR and IER

290486–52

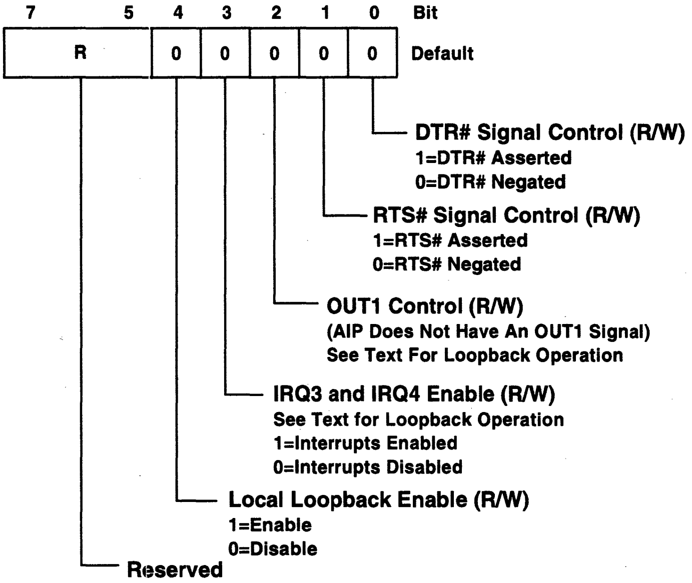
Figure 52. Line Control Register

Bit	Description
7	**DIVISOR LATCH ACCESS BIT (DLAB):** DLAB controls access to the Baud Rate Generator Divisor Latches (and to the Transmit Holding Register, Receiver Buffer Register and Interrupt Enable Register which are located at the same I/O addresses). When DLAB = 1, access to the two Divisor Latches is selected and access to the THR, RBR, and IER is disabled. When DLAB = 0, access to the two Divisor Latches is disabled and access to the THR, RBR, and IER is selected. During test mode operations, DLAB must be set to 1 for the BOUT signal to appear on the SOUT pin.
6	**BREAK CONTROL (BRCON):** When BRCON = 1, a break condition is transmitted from the 82091AA serial port to the receiving device. When BRCON = 1, the serial output (SOUT) is forced to the 'spacing' state (logical 0). BRCON only affects the SOUT signal and has no effect on the transmitter logic. Note that this feature permits the CPU to alert a terminal. If the following sequence is used, no erroneous characters will be transmitted because of the break. 1. Wait for the transmitter to be idle (TEMT = 1). 2. Set break (BRCON = 1) for the appropriate amount of time. If the transmitter will be used to time the break duration, then check that TEMT = 1 before clearing the BRCON. 3. Clear break (BRCON = 0) when normal transmission has to be restored. During the break, the transmitter can be used as a character timer to accurately establish the break duration by sending characters and monitoring THRE and TEMT.
5	**STICKY PARITY (STICPAR):** STICPAR is the Stick Parity bit. When parity is enabled (PAREN = 1) this bit is used in conjunction with EVENPAR to select "Mark" or "Space" Parity. When bits PAREN, EVENPAR and STICPAR are 1, the parity bit is transmitted and checked as a 0 (Space Parity). If bits PAREN and STICPAR are 1 and EVENPAR is 0, the parity bit is transmitted and checked as a 1 (Mark Parity). When STICPAR = 0, stick parity is disabled.
4	**EVEN PARITY SELECT (EVENPAR):** EVENPAR selects between even and odd parity. When parity is enabled (PAREN = 1) and EVENPAR = 0, an odd number of 1s is transmitted or checked in the data word bits and parity bit. When parity is enabled and EVENPAR = 1, an even number of 1s is transmitted or checked.
3	**PARITY ENABLE (PAREN):** This bit enables/disables parity generation and checking. When PAREN = 1, a parity bit is generated (transmit data) or checked (receive data) between the last data bit and stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data bits and the Parity bit are summed.) When PAREN = 0, parity generation and checking is disabled.
2	**STOP BITS (STOPB):** This bit specifies the number of stop bits transmitted with each serial character. When STOPB = 0, one stop bit is generated in the transmitted data. When STOPB = 1 and a 5-bit data length is selected, one and a half stop bits are generated. When STOPB = 1 and either a 6-, 7-, or 8-bit data length is selected, two stop bits are generated. The receiver checks the first Stop bit only, regardless of the number of Stop bits selected.
1:0	**SERIAL DATA BITS (SERIALDB):** This field specifies the number of data bits in each transmitted or received serial character as follows: **Bits[1:0]** **Data Length** 0 0 5 Bits - Default 0 1 6 Bits 1 0 7 Bits 1 1 8 Bits

7.1.8 MCR(A,B)—MODEM CONTROL REGISTER

I/O Address:	Base +4h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

This register controls the interface with the modem or data set (or a peripheral device emulating a modem).


7
5
4
3
2
1
0
Bit
R
0
0
0
0
0
Default
DTR# Signal Control (R/W)
1=DTR# Asserted
0=DTR# Negated
RTS# Signal Control (R/W)
1=RTS# Asserted
0=RTS# Negated
OUT1 Control (R/W)
(AIP Does Not Have An OUT1 Signal)
See Text For Loopback Operation
IRQ3 and IRQ4 Enable (R/W)
See Text for Loopback Operation
1=Interrupts Enabled
0=Interrupts Disabled
Local Loopback Enable (R/W)
1=Enable
0=Disable
Reserved

290486–53

Figure 53. Modem Control Register

Bit	Description
7:5	**RESERVED**
4	**LOOPBACK MODE ENABLE (LME):** LME provides a local loopback feature for diagnostic testing of the serial port module. When LME = 1, the following occurs: 1. The transmitter Serial Output (SOUT) is set to the Marking (logic 1) state. 2. The receiver Serial Input (SIN) is disconnected. 3. The output of the Transmitter Shift Register is "looped back"(connected) to the Receiver Shift Register. 4. The four modem control inputs (DSR#, CTS#, RI and DCD#) are disconnected. 5. The DTRC, RTSC, OUT1C, IE bits in the MCR are internally connected to DSRS, CTSS, RIS, and DCDS in MSR, respectively. 6. The modem control output pins are forced to their high (inactive) state. 7. Data that is transmitted is immediately received. This feature allows the CPU to verify the transmit and received data paths of the serial port. In the loopback mode, the receiver and transmitter interrupts are fully operational. The modem status interrupts are fully operational. The modem status interrupts are also operational, but the interrupt sources are the lower four bits of MCR instead of the four modem control inputs. Writing a 1 to any of these 4 MCR bits (bits[3:0]) causes an interrupt. In Loopback Mode the interrupts are still controlled by the Interrupt Enable Register. The IRQ3 and IRQ4 signal pins are tri-stated in the loopback mode.
3	**INTERRUPT ENABLE (IE):** When IE = 1, the associated interrupt is enabled (either IRQ3 or IRQ4 as selected via the associated serial port configuration register - A or B). In Local Loopback Mode, this bit controls bit 7 of the Modem Status Register.
2	**OUT1 BIT CONTROL (OUT1C):** This bit is the OUT1 bit. It does not have an output pin associated with it. It can be written to and read by the CPU. In Local Loopback Mode, this bit controls bit 6 of the Modem Status Register.
1	**REQUEST TO SEND CONTROL (RTS):** This bit controls the Request to Send (RTS#) output. When RTSC = 1, the RTS# output is asserted. When RTSC = 0, the RTS# output is negated. In Local Loopback Mode, this bit controls bit 4 of the Modem Status Register.
0	**DATA TERMINAL READY CONTROL (DTRC):** This bit controls the Data Terminal Ready (DTR#) output. When DTRC = 1, the DTR# output is asserted. When DTRC = 0, the DTR# output is negated. In Local Loopback Mode, this bit controls bit 5 of the Modem Status Register. **NOTE:** The DTR# and RTS# outputs of the serial port may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the modem or data set.

7.1.9 LSR(A,B)—LINE STATUS REGISTER

I/O Address:	Base + 5h
Default Value:	60h
Attribute:	Read/Write
Size:	8 bits

This 8-bit register provides data transfer status information to the CPU. Note that the Line Status Register is intended for read operations only. Writing to this register is not recommended and could result in unintended operations. For this reason, the figure shows these bits as RO (read only).

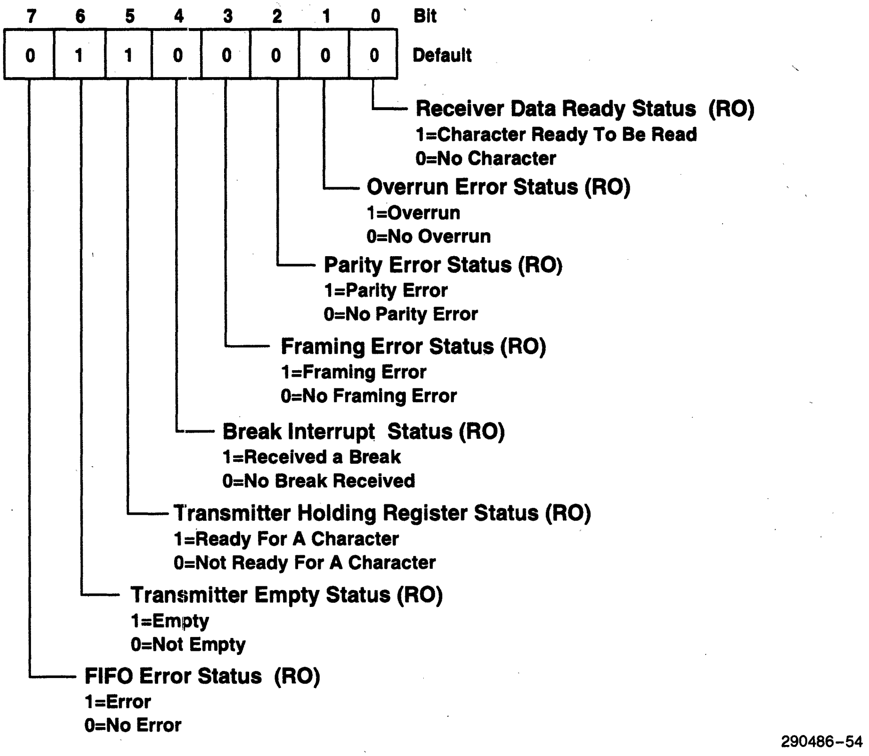


Figure 54. Line Status Register

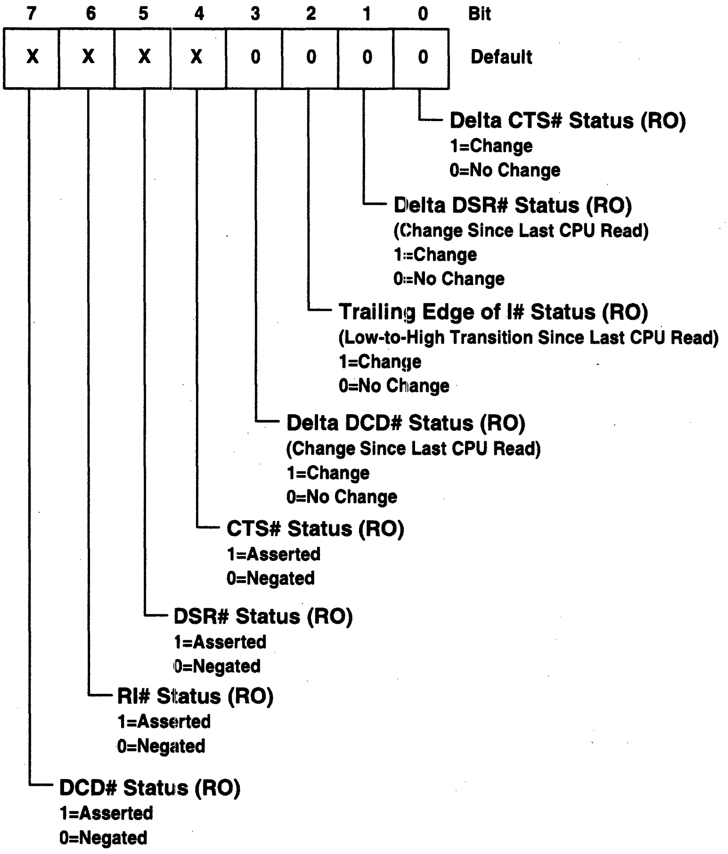
Bit	Description
7	**FIFO ERROR STATUS (FIFOE):** In the non-FIFO Mode this is a 0. In the FIFO Mode, FIFOE is set to 1 when there is at least one parity error, framing error, or break indication in the FIFO. FIFOE is set to 0 when the CPU reads the LSR, if there are no subsequent errors in the FIFO.
6	**TRANSMITTER EMPTY STATUS (TEMT):** This bit is the Transmitter Empty (TEMT) indicator. When the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty, the 82091AA sets TEMT to a 1. When either the THR or TSR contains a data character, TEMT is set to a 0. The default is 0. In FIFO mode, this bit is set to 1 when the transmitter FIFO and the shift register are both empty.

Bit	Description
5	**TRANSMITTER HOLDING REGISTER STATUS (THRE):** This bit is the Transmitter Holding Register Empty (THRE) indicator. THRE indicates that the serial port module is ready to accept a new character for transmission. In addition, this bit causes the serial port module to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set to a 1. THRE is set to 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. THRE is set to 0 when the CPU loads the Transmitter Holding Register. In the FIFO mode, this bit is set to a 1 when the transmit FIFO is empty, and is set to 0 when at least 1 byte is written to the transmit FIFO.
4	**BREAK INTERRUPT STATUS (BI):** This bit is the Break Interrupt (BI) indicator. BI is set to a 1 when the received data input is held in the Spacing state (logic 0) for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). When the CPU reads the contents of the Line Status Register, BI is set to 0. In FIFO mode, this error is associated with the particular character in the FIFO associated with the Break. BI is indicated to the CPU when its associated character is at the top of the FIFO. When break occurs only one character is loaded into the FIFO. Restarting after a break is received requires the SIN pin to be a logical 1 for at least ½ bit times. **NOTE:** Bits[3:0] are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and that interrupt is enabled.
3	**FRAMING ERROR STATUS (FE):** This bit is the Framing Error (FE) indicator. FE indicates that the received character did not have a valid stop bit. FE is set to a 1 when the stop bit following the last data bit or parity bit is 0 (spacing level). FE is set to 0 when the CPU reads the contents of the Line Status Register. In FIFO mode, this error is associated with the particular character in the FIFO that it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. When a framing error is due to the next start bit, the serial port attempts to resynchronize. In this case, the serial port module samples this start bit twice and, if no FE exists, then the module takes in the rest of the bits.
2	**PARITY ERROR STATUS (PE):** This bit is the Parity Error (PE) indicator. PE indicates that the received data character does not have the correct even or odd parity, as selected by the EVENPAR bit in the Line Status Register. When a parity error is detected, PE is set to 1. PE is set to 0 when the CPU reads the contents of the Line Status Register. In the FIFO mode, this error is associated with the particular character in the FIFO that it applies to. This error is indicated to the CPU when its associated character is at the top of the FIFO.
1	**OVERRUN ERROR STATUS (OE):** OE indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register. In this case, the previous character is overwritten. When an overrun is detected, OE is set to 1. when the CPU reads the Line Status Register, OE is set to 0. This bit is read only. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is completely full and the next character has been received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register is overwritten, but it is not transferred to the FIFO.
0	**RECEIVER DATA READY STATUS (DR):** DR is set to 1 when a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. When the data in the Receiver Buffer Register or FIFO is read, DR is set to 0. This bit is read only.

7.1.10 MSR(A,B)—MODEM STATUS REGISTER

I/O Address:	Base + 6h
Default Value:	XXXX 0000
Attribute:	Read/Write
Size:	8 bits

The MSR provides the current state of the control lines from the Modem (or peripheral device) to the CPU. Bits[7:4] provide the status of the DCD#, RI, DSR#, and CTS# Modem signals. In addition to the current-state information of the Modem signals, bits[3:0] provide change information for these signals. Bits[3:0] are set to a 1 when the corresponding input signal changes state. Bits[3:0] are set to a 0 when the CPU reads the Modem Status Register.



290486-55

NOTE:
X = Value determined by state of the corresponding modem control signal.

Figure 55. Modem Status Register

Bit	Description
7	**DATA CARRIER DETECT STATUS:** This bit is the compliment of the Data Carrier Detect (DCD#) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to IRQ ENABLE in the MCR.
6	**RING INDICATOR STATUS:** This bit is the compliment of the Ring Indicator (RI) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT1 in the MCR.
5	**DATA SET READY STATUS:** This bit is the compliment of the Data Set Ready (DSR#) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to DTR in the MCR.
4	**CLEAR TO SEND STATUS:** This bit is the compliment of the Clear to Send (CTS#) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.
3	**DELTA DATA CARRIER DETECT STATUS:** This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the DCD# input to the chip has changed state. **NOTE:** Whenever bit 0, 1, 2, or 3 is set to logic 1, a Modem Status Interrupt is generated.
2	**TRAILING EDGE OF RING INDICATOR STATUS:** This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the RI# input to the chip has changed from a low to a high state.
1	**DELTA DATA SET READY STATUS:** This bit is the Delta Data Set Ready (DDSR) indictor. Bit 1 indicates that the DSR# input to the chip has changed state since the last time it was read by the CPU.
0	**DELTA CLEAR TO SEND STATUS:** This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the CTS# input to the chip has changed state since the last time it was read by the CPU.

7.1.11 SCR(A,B)—SCRATCHPAD REGISTER

I/O Address:	Base +7h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

This 8-bit read/write register does not control the serial port module in any way. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

Bit	Description
7:0	**SCRATCHPAD DATA:** Bits[7:0] of this register correspond to SD[7:0].

7.2 FIFO Operations

This section describes the FIFO operations for interrupt and polled modes.

7.2.1 FIFO INTERRUPT MODE OPERATION

When the Receive FIFO and receiver interrupts are enabled (FCR0 = 1 and IER0 = 1), receiver interrupts occur as follows:

1. The receive data available interrupt is invoked when the FIFO has reached its programmed trigger level. The interrupt is cleared when the FIFO drops below the programmed trigger level.
2. The IIR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt, the bits are cleared when the FIFO drops below the trigger level.
3. The receiver line status interrupt (IIR-06h), as before, has higher priority than the received data available (IIR = 04h) interrupt.
4. The data ready bit (LSR0) is set as soon as a character is transferred from the shift register to the receive FIFO. This bit is set to 0 when the FIFO is empty.

When receiver FIFO and receiver interrupts are enabled, receiver FIFO timeout interrupts occur as follows:

1. A FIFO timeout interrupt occurs, if the following conditions exist:
   a. At least one character is in the FIFO.
   b. The most recent serial character received was longer than 4 continous character times ago (if 2 stop bits are programmed, the second one is included in this time delay).
   c. The most recent CPU read of the FIFO was longer than 4 continous character times ago.

   The maximum time between a received character and a timeout interrupt is 160 ms at 300 baud with a 12-bit receive character (i.e., 1 start, 8 data, 1 parity, and 2 stop bits).
2. Character times are calculated by using the RCLK input for a clock signal (this makes the delay proportional to the baud rate).
3. When a timeout interrupt occurs, it is cleared and the timer reset when the CPU reads one character from the receiver FIFO.
4. When a timeout interrupt does not occur, the timeout timer is reset after a new character is received or after the CPU reads the receiver FIFO.

When the transmit FIFO and transmitter interrupts are enabled (FCR0 = 1, IER1 = 1), transmit interrupts occur as follows:

1. The transmitter holding register interrupt occurs when the transmit FIFO is empty. The interrupt is cleared as soon as the transmitter holding register is written (1 to 16 characters may be written to the transmit FIFO while servicing the interrupt) or the IIR is read.

Character timeout and receiver FIFO trigger level interrupts have the same priority as the current received data available interrupt. Transmit FIFO empty has the same priority as the current transmitter holding register empty interrupt.

7.2.2 FIFO POLLED MODE OPERATION

With FIFO = 1, setting IER[3:0] to all 0s puts the serial port in the FIFO polled mode of operation. Since the receiver and transmitter are controlled separately, either one or both can be in the polled mode of operation.

In this mode, software checks receiver and transmitter status via the LSR. As stated in the register description:

- LSR0 is set as long as there is one byte in the receiver FIFO.
- LSR1 and LSR4 specify which error(s) has occurred. Character error status is handled the same way as interrupt mode. The IIR is not affected since IER2 = 0.
- LSR5 indicates when the transmitter FIFO is empty.
- LSR6 indicates that both the transmitter FIFO and shift register are empty.
- LSR7 indicates whether there are any errors in the receiver FIFO.

8.0 FLOPPY DISK CONTROLLER

The 82091AA's Floppy Disk Controller (FDC) is functionally compatible with 82078/82077SL/82077AA/8272A floppy disk controllers. During 82091AA configuration, the FDC can be configured for either two drive support or four drive support via the FCFG1 Register. This section provides a complete description of the FDC when it is configured for two drive support. Additional information on four drive support is provided in Appendix A, FDC Four Drive Support.

NOTE:

For FDC compatibility and programming guidelines, refer to the 82078 Floppy Disk Controller Data sheet.

8.1 Floppy Disk Controller Registers

The FDC contains seven status, control, and data registers. Table 23 shows the I/O address assignments for the FDC registers and the individual register descriptions follow in the order that they appear in the table. The registers provide control/status information and data paths for transfering data between the floppy disk controller interface and the 8-bit host interface. In some cases, two different registers occupy the same I/O address. In these cases, one register is read only and the other is write only (i.e., a read to the I/O address accesses one register and a write accesses the other register).

All registers are accessed as byte quantities. The base address is determined by hardware configuration at powerup (or a hard reset) or via software configuration by programming the 82091AA configuration registers as described in Section 4.0, AIP Configuration.

During a hard reset (RSTDRV asserted), the 82091AA registers are set to pre-determined **default** states. The default values are indicated in the individual register descriptions. Reserved bits in the FDC registers must be programmed to 0 when writing the register and these bits are 0 when read. The following bit notation is used for default settings:

X Default bit position value is determined by conditions on an 82091AA signal pin.

The following nomenclature is used for register access attributes:

RO **Read Only.** Note that for registers with read only attributes, writes to the I/O address have no affect on floppy disk operations.

WO **Write Only.** Note that for all FDC registers with write only attributes, reads of the I/O address access a different register.

R/W **Read/Write.** A register with this attribute can be read and written. Note that individual bits in some read/write registers may be read only.

Table 23 lists the register accesses that bring the FDC out of a powerdown state. All other registers accesses are possible without waking the part from a powerdown state and reads from these registers reflects the true status as shown in the register description. For writes that do not affect the powerdown state, the FDC retains the data and will subsequently reflect it when the FDC awakens. Note that for accesses that do not affect powerdown, the access may cause a temporary increase in FDC power consumption. The FDC reverts back to low power mode when the access has been completed. None of the extended registers effect the behavior of the powerdown mode.

Table 23. Floppy Disk Controller Registers(1)

FDC Register Address Access Base +	Abbreviation	Register Name	Access Wakes Up FDC	Access
0h	—	Reserved	—	—
1h	SRB	Status Register B	No	RO
2h	DOR	Digital Output Register	No(2)	R/W
3h	TDR	Tape Drive Register	No	R/W
4h	MSR	Main Status Register	Yes	RO
4h	DSR	Datarate Select Register	No(2)	WO
5h	FIFO	Data FIFO	Yes	R/W
6h	—	Reserved	—	—
7h	DIR#	Digital Input Register	No	RO
7h	CCR	Configuration Control Register		WO

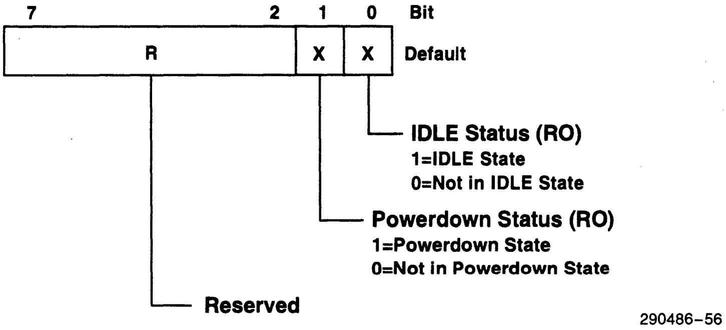
NOTES:

1. The base address is 3F0h (primary address) or 370 (secondary address).
2. While writing to the DOR or DSR does not wake up the FDC, writing any of the motor enable bits in the DOR or invoking a software reset (either via DOR or DSR reset bits) will wake up the FDC.

8.1.1 SRB—STATUS REGISTER B (EREG EN = 1)

I/O Address:	Base + 1h
Default Value:	RRRR RRXX
Attribute:	Read/Write
Size:	8 bits

SRB provides status and control information when auto powerdown is enabled. In the AT/EISA mode the SRB is made available whenever the EREG EN bit in the POWERDOWN MODE Command is set to 1. When EREG EN bit is set to 0, this register is not accessible. In this case, writes have no affect and reads return indeterminate values.



NOTE:
X = Value is determined by the state of the corresponding signal pin.

Figure 56. Status Register B

Bit	Description
7:2	**RESERVED**
1	**POWERDOWN STATUS (PD):** This bit reflects the powerdown state of the FDC module. The 82091AA sets PD to 1 when the FDC is in the powerdown state. When PD = 0, the FDC is not in the powerdown state.
0	**IDLE STATUS (IDLE):** This bit reflects the idle state of the FDC module. The 82091AA sets IDLE to 1 when the FDC is in the idle state. When IDLE = 0, the FDC is not in the idle state.

8.1.2 DOR—DIGITAL OUTPUT REGISTER

I/O Address:	Base + 2h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

The Digital Output Register enables/disables the floppy disk drive motors, selects the disk drives, enables/disables DMA, and provides a FDC module reset. The DOR reset bit and the motor enable bits have to be inactive when the FDC is in powerdown. The DMAGATE# and drive select bits are unchanged. During powerdown, writing to the DOR does not wake up the FDC, except for activating any of the motor enable bits. Setting the motor enable bits to 1 wakes up the FDC.

NOTES:

1. The descriptions in this section for DOR only apply when two-drive support is selected in the FCFG1 Register (FDDQTY = 0). For four-drive support (FDDQTY = 1), refer to Appendix A, FDC Four Drive Support.
2. The drive motor can be enabled separately without selecting the drive. This permits the motor to come up to speed before selecting the drive. Note also that only one drive can be selected at a time. However, the drive should not be selected without enabling the appropriate drive motor via bits[5:4] of this register.

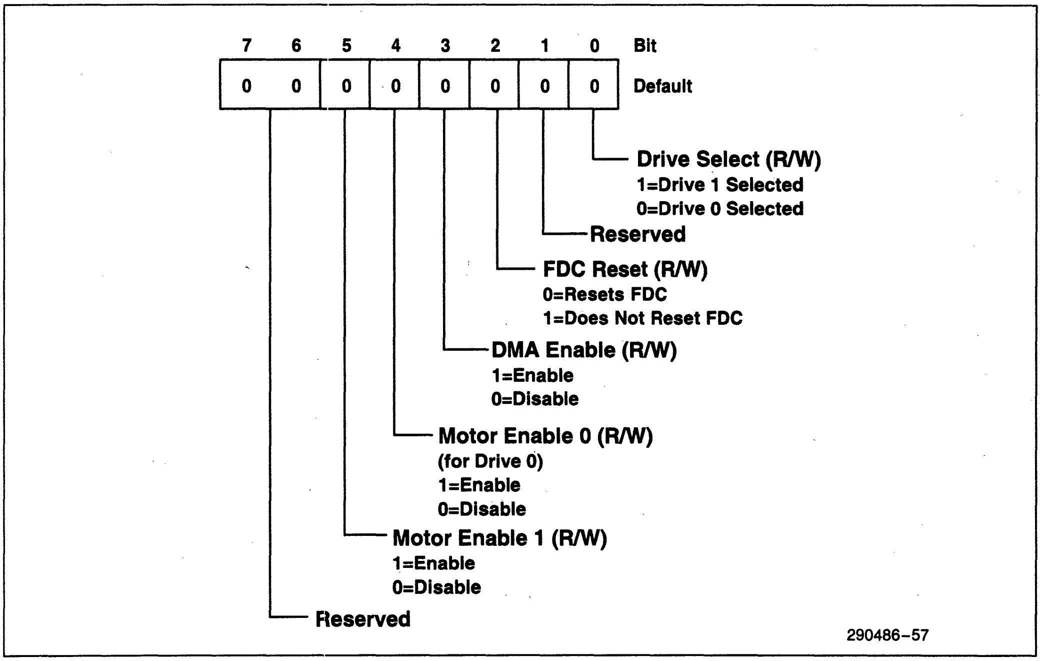


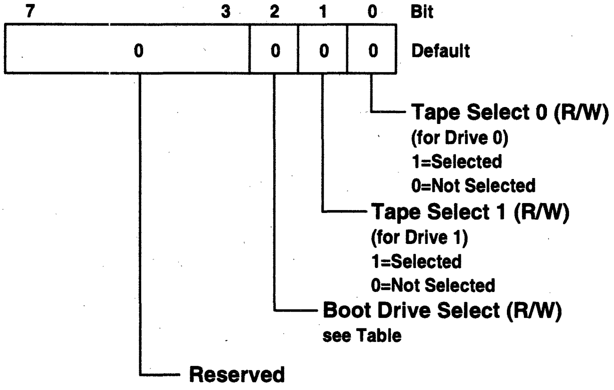
Figure 57. Digital Output Register

Bit	Description
7:6	**RESERVED:** For a two-drive system, these bits are not used and have no affect on FDC operation. For a four drive system, see Appendix A, FDC Four Drive Support.
5	**MOTOR ENABLE 1 (ME1):** This bit controls a motor drive enable signal. ME1 directly controls either the FDME1# signal or FDME0# signal, depending on the state of the BOOTSEL bit in the TDR. When ME1 = 1, the selected motor enable signal (FDME1# or FDME0#) is asserted and when ME1 = 0, the selected motor enable signal is negated.
4	**MOTOR ENABLE 0 (ME0):** This bit controls a motor drive enable signal. ME1 directly controls either the FDME0# signal or FDME1# signal, depending on the state of the BOOTSEL bit in the TDR. When ME0 = 1, the selected motor enable signal (FDME0# or FDME1#) is asserted and when ME0 = 0, the selected motor enable signal is negated.
3	**DMA GATE (DMAGATE):** This bit enables/disables DMA for the FDC. When DMAGATE = 1, DMA for the FDC is enabled. In this mode, FDDREQ, TC, IRQ6, and FDDACK# are enabled. When DMAGATE = 0, DMA for the FDC is disabled. In this mode the IRQ6, and DRQ outputs are tri-stated and the DACK# and TC inputs are disabled to the FDC. Note that the TC input is only disabled to the FDC module. Other functional units in the 82091AA (e.g., parallel port or IDE interface) can still use the TC input signal for DMA activities.
2	**FDC RESET (DORRST):** DORRST is a software reset for the FDC module. When DORRST is set to 0, the basic core of the FDC and the FIFO circuits are cleared conditioned by the LOCK bit in the CONFIGURE Command. This bit is set to 0 by software or a hard reset (RSTDRV asserted). The FDC remains in a reset state until software sets this bit to 1. This bit does not affect the DSR, CCR and other bits of the DOR. DORRST must be held active for at least 0.5 μs at 250 Kbps. This is less than a typical ISA I/O cycle time. Thus, in most systems consecutive writes to this register to toggle this bit allows sufficient time to reset the FDC.
1	**RESERVED:** For a two-drive system, this bit is not used and must be programmed to 0. For a four drive system, see Appendix A, FDC Four Drive Support.
0	**DRIVE SELECT (DS):** This selects the floppy drive by controlling the FDS0# and FDS1# output signals. DS directly controls FDS1 and FDS0 as follows: **Bit 0** **Output Pin Status** 0 FDS0# asserted (FDS1 asserted if BOOTSEL = 1) 1 FDS1# asserted (FDS1 asserted if BOOTSEL = 1)

8.1.3 TDR—ENHANCED TAPE DRIVE REGISTER

I/O Address:	Base +3h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

This register allows the user to assign tape support to a particular drive during initialization. Any future references to that drive number automatically invokes tape support. A hardware reset sets all bits in this register to 0 making drive 0 not available for tape support. A software reset via bit 2 of the DOR does not affect this register. Drive 0 is reserved for the floppy boot drive. Bits[7:2] are only available when EREG EN = 1; otherwise the bits are tri-stated. EREG EN is a bit in the POWERDOWN Command.



290486-58

Figure 58. Enhanced Tape Drive Register

Bit	Description
7:3	**RESERVED**
2	**BOOT DRIVE SELECT (BOOTSEL):** The BOOTSEL bit is used to remap the drive selects and motor enables. The functionality is as described below: **BOOTSEL** **Mapping** 0 DS0 → FDS0, ME0 → FDME0 (default) DS1 → DS1, ME1 → FDME1 1 DS0 → DS1, ME0 → FDME1 DS1 → FDS0, ME1 → FDME0 Note that this mapping also applies to a four drive system (FDDQTY = 1 in the FCFG1 Register). In a four drive system, only drive 0 or drive 1 can be selected as the boot drive.
1	**RESERVED:** For a two-drive system, this bit is not used and must be programmed to 0. For a four drive system, see Appendix A, FDC Four Drive Support.
0	**TAPE SELECT (TAPESEL):** This bit is used by software to assign logical drive number 1 to be a tape drive. Other than adjusting precompensation delays for tape support, this bit does not affect the FDC hardware. The bit can be written and read by software as an indication of the tape drive assignment. Drive 0 is not available as a tape drive and is reserved as the floppy disk boot drive. The tape drive assignment is as follows: **Bit 0** **Drive Selected** 0 None (all are floppy disk drives) 1 Drive 1 is a tape drive.

8.1.4 MSR—MAIN STATUS REGISTER

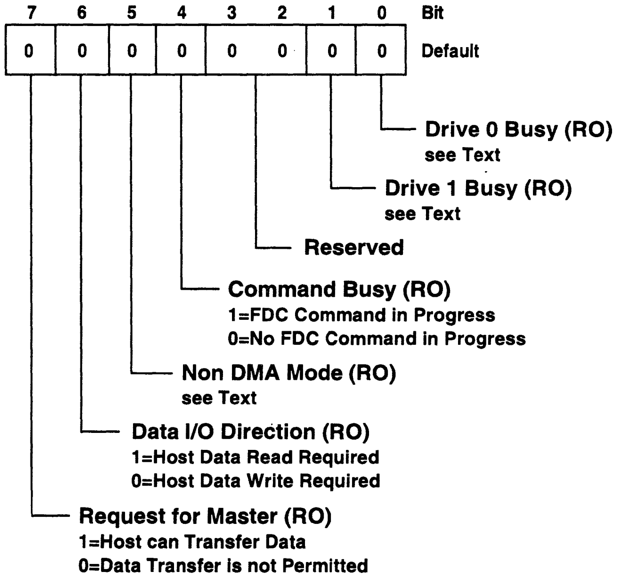
I/O Address:	Base +4h
Default Value:	00h
Attribute:	Read Only
Size:	8 bits

This read only register provides FDC status information. This information is used by software to control the flow of data to and from the FIFO (accessed via the FDCFIFO Register). The MSR indicates when the FDC is ready to send or receive data through the FIFO. During non-DMA transfers, this register should be read before each byte is transferred to or from the FIFO.

After a hard or soft reset or recovery from a powerdown state, the MSR is available to be read by the host. The register value is 00h until the oscillator circuit has stabilized and the internal registers have been initialized. When the FDC is ready to receive a new command, MSR[7:0] = 80h. The worst case time allowed for the MSR to report 80h (i.e., RQM is set to 1) is 2.5 μs after a hard or soft reset.

Main Status Register is used for controlling command input and result output for all commands. Some example values of the MSR are:

- MSR = 80H; The controller is ready to receive a command.
- MSR = 90H; executing a command or waiting for the host to read status bytes (assume DMA mode).
- MSR = D0H; waiting for the host to write status bytes.



290486-59

Figure 59. Main Status Register

Bit	Description
7	**REQUEST FOR MASTER (RQM):** When RQM = 1, the FDC is ready to send/receive data through the FIFO (FDCFIFO Register). The FDC sets this bit to 0 after a byte transfer and then sets the bit to 1 when it is ready for the next byte. During non-DMA execution phase, RQM indicates the status of IRQ6.
6	**DIRECTION I/O (DIO):** When RQM = 1, DIO indicates the direction of a data transfer. When DIO = 1, the FDC is requesting a read of the FDCFIFO. When DIO = 0, the FDC is requesting a write to the FDCFIFO.
5	**NON-DMA (NONDMA):** Non-DMA mode is selected via the SPECIFY Command. In this mode, the FDC sets this bit to a 1 during the execution phase of a command. This bit is for polled data transfers and helps differentiate between the data transfer phase and the reading of result bytes.
4	**COMMAND BUSY (CMDBUSY):** CMDBUSY indicates when a command is in progress. When the first byte of the command phase is written, the FDC sets this bit to 1. CMDBUSY is set to 0 after the last byte of the result phase is read. If there is no result phase (e.g., SEEK or RECALIBRATE Commands), CMDBUSY is set to 0 after the last command byte is written.
3:2	**RESERVED:** For a two-drive system, these bits are not used and must be programmed to 0. For a four drive system, see Appendix A, FDC Four Drive Support.
1	**DRIVE 1 BUSY (DRV1BUSY):** The FDC module sets this bit to 1 after the last byte of the command phase of a SEEK or RECALIBRATE Command is issued for drive 1. This bit is set to 0 after the host reads the first byte in the result phase of the SENSE INTERRUPT Command for this drive.
0	**DRIVE 0 BUSY (DRV0BUSY):** The FDC module sets this bit to 1 after the last byte of the command phase of a SEEK or RECALIBRATE Command is issued for drive 0. This bit is set to 0 after the host reads the first byte in the result phase of the SENSE INTERRUPT Command for this drive.

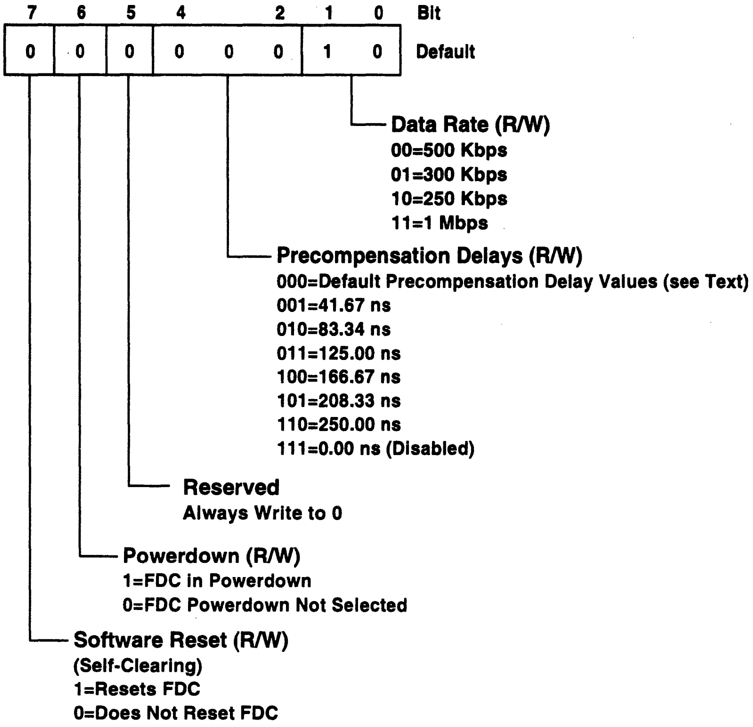
8.1.5 DSR—DATA RATE SELECT REGISTER

I/O Address:	Base + 4h
Default Value:	02h
Attribute:	Write Only
Size:	8 bits

The DSR selects the data rate, amount of write precompenstion, invokes direct powerdown, and invokes a FDC software reset. This write only register ensures backward compatibility with the Intel series of floppy disk controllers. Changing the data rate changes the timings of the drive control signals. To ensure that drive timings are not violated when changing data rates, choose a drive timing such that the fastest data rate will not violate the timing.

In the default state, the PDOSC bit is low and the oscillator is powered up. When this bit is programmed to a 1, the oscillator is shut off. Hardware reset sets this bit to a 0. Neither of the software resets (via DOR or DSR) have any effect on this bit. Note that PDOSC should only be set to a 1 when the FDC module is in the powerdown state. Otherwise, the FDC will not function correctly and must be hardware reset once the oscillator has turned back on and stabilized. Setting the PDOSC bit has no effect on the clock input to the FDC (the X1 pin). The clock input is separately disabled when the part is powered down. The Save Command checks the status of PDOSC. However the Restore Command will not restore this bit to a 1.

Software resets do not affect the DRATE or PRECOMP bits.



290486-60

Figure 60. Data Rate Select Register

Bit	Description
7	**SOFTWARE RESET (DSRRST):** DSRRST operates the same as the DORRST bit in the DOR, except that this bit is self clearing.
6	**POWERDOWN (FPD):** FPD provides direct powerdown for the FDC module. When FPD = 1, the FDC module enters the powerdown state, regardless of the state of the module. The FDC module is internally reset and then put into powerdown. No status is saved and any operation in progress is aborted. A hardware or software reset causes the 82091AA to exit the FDC module powerdown state.
5	**RESERVED**
4:2	**PRECOMPENSATION (PRECOMP):** Bits[4:2] adjusts the WRDATA output to the disk to compensate for magnetic media phenomena known as bit shifting. The data patterns that are susceptible to bit shifting are well understood and the FDC compensates the data pattern as it is written to the disk. The amount of precompensation depends on the drive and media but in most cases the default value is acceptable. The FDC module starts pre-compensating the data pattern starting on Track 0. The CONFIGURE Command can change the track where pre-compensating originates. **Bits[4:2]** **Precompensation Delays (ns)** 0 0 0 Default mode 0 0 1 41.67 0 1 0 83.34 0 1 1 125.00 1 0 0 166.67 1 0 1 208.33 1 1 0 250 1 1 1 0.00 (disabled) The default precompensation delay mode provides the following delays: **Data Rate** **Default Precompensation Delays (ns)** 1 Mbps 41.67 0.5 Mbps 125.00 0.3 Mbps 125.00 0.25 Mbps 125.00
1:0	**DATA RATE SELECT (DRATESEL):** DRATESEL[1:0] select one of the four data rates as listed below. The default value is 250 Kbps. **Bits[1:0]** **Date Rate** 1 1 1 Mbps 0 0 500 Kbps 0 1 300 Kbps 1 0 250 Kbps - default

8.1.6 FDCFIFO—FDC FIFO (DATA)

I/O Address:	Base +5h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

All command parameter information and disk data transfers go through the 16-byte FIFO. The FIFO has programmable threshold values. Data transfers are governed by the RQM and DIO bits in the MSR. At the start of a command, the FIFO action is always disabled and command parameters must be sent based upon the RQM and DIO bit settings. At the start of the command execution phase, the FDC clears the FIFO of any data to ensure that invalid data is not transferred. An overrun or underrun will terminate the current command and the transfer of data. Disk writes complete the current sector by generating a 00 pattern and valid CRC.

The FIFO defaults to an 8272A compatible mode after a hardware reset (via RSTDRV pin). Software resets (via DOR or DSR) can also place the FDC into 8272A compatible mode, if the LOCK bit is set to 0 (see the definition of the LOCK bit) maintaining PC-AT hardware compatibility. The default values can be changed through the CONFIGURE Command (enable full FIFO operation with threshold control). The FIFO provides the system a larger DMA latency without causing a disk error. The following table gives several examples of the delays with a FIFO. The data is based upon the formula: Threshold# × 1/DATA RATE × 8 − 1.5 µs = DELAY.

FIFO Threshold	Maximum Service Delay (1 Mbps Data Rate)	Maximum Delay to Servicing at 500 Kbps Data Rate
1 byte	1 × 8 µs − 1.5 µs = 6.5 µs	1 × 16 µs − 1.5 µs = 14.5 µs
2 bytes	2 × 8 µs − 1.5 µs = 14.5 µs	2 × 16 µs − 1.5 µs = 30.5 µs
8 bytes	8 × 8 µs − 1.5 µs = 62.5 µs	8 × 16 µs − 1.5 µs = 126.5 µs
15 bytes	15 × 8 µs − 1.5 µs = 118.5 µs	15 × 16 µs − 1.5 µs = 238.5 µs

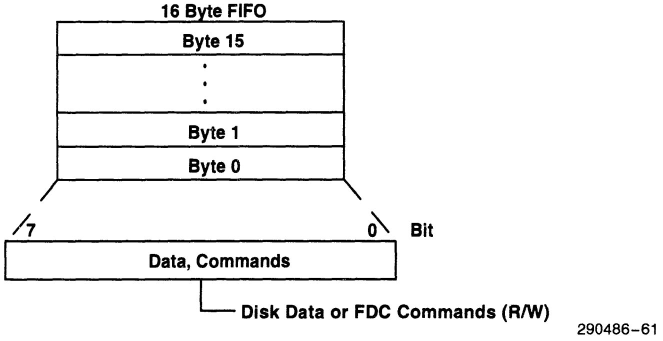


Figure 61. FDC FIFO

Bit	Description
7:0	**FIFO DATA:** Bits[7:0] correspond to SD[7:0].

8.1.7 DIR—DIGITAL INPUT REGISTER

I/O Address: Base +7h
Default Value: 00h
Attribute: Read Only
Size: 8 bits

This register is read only in all modes. In PC-AT mode only bit 7 is driven and all other bits remain tri-stated.

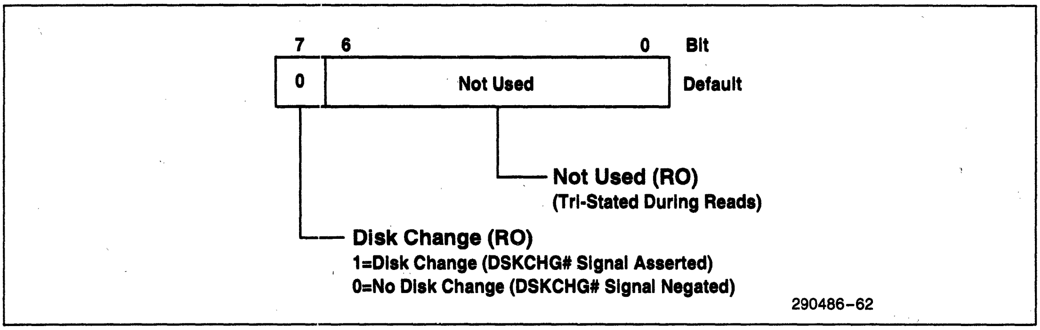

7
6
0
Bit
0
Not Used
Default
Not Used (RO)
(Tri-Stated During Reads)
Disk Change (RO)
1=Disk Change (DSKCHG# Signal Asserted)
0=No Disk Change (DSKCHG# Signal Negated)
290486-62

Figure 62. Digital Input Register

Bit	Description
7	**DISK CHANGE (DSKCHG):** This bit monitors a disk change in the floppy disk drive. DSKCHG is set to a 1 when the DSKCHG# signal on the floppy interface is asserted. DSKCHG is set to a 0 when the DSKCHG# signal on the floppy interface is negated. During powerdown, this bit is invalid.
6:0	**NOT USED:** These bits are tri-stated during a read.

8.1.8 CCR—CONFIGURATION CONTROL REGISTER

I/O Address:	Base +7h
Default Value:	02h
Attribute:	Write Only
Size:	8 bits

This register sets the data rate.

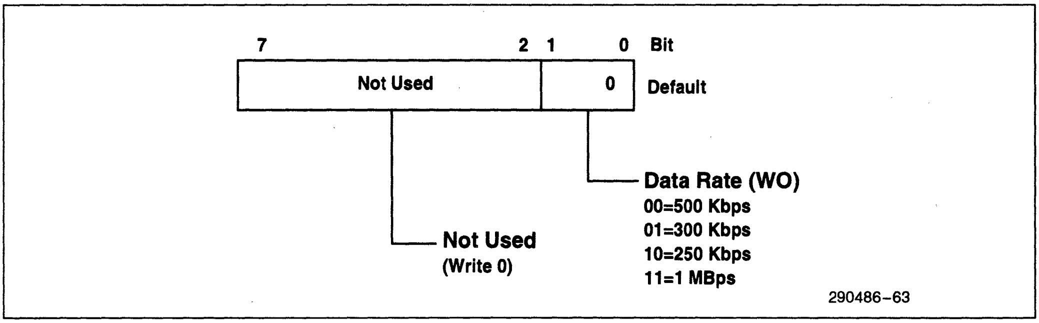


Figure 63. Configuration Control Register

8.2 Reset

There are four sources of FDC reset—a hard reset via the RSTDRV signal and three software resets (via the FCFG2, DOR, and DSR Registers). At the end of the reset, the FDC comes out of the powerdown state. Note that the DOR reset condition remains in effect until software programs the DORRST bit to 1 in the DOR. All operations are terminated and the FDC enters an idle state. Invoking a reset while a disk write activity is in progress will corrupt the data and CRC. On exiting the reset state, various internal registers are cleared, and the FDC waits for a new command. Drive polling will start unless disabled by a new CONFIGURE Command.

8.2.1 HARD RESET AND CONFIGURATION REGISTER RESET

A hard reset (asserting RSTDRV) and a software reset through the FCFG2 Registers have the same affect on the FDC. These resets clear all FDC registers, except those programmed by the SPECIFY command. The DOR reset bit is enabled and must be set to 0 by the host to exit the reset state.

8.2.2 DOR RESET vs DSR RESET

The DOR and DSR resets are functionally the same. The DSR reset is included to maintain 82072 compatibility. Both reset the 8272 core, which affects drive status information. The FIFO circuits are also reset if the LOCK bit is a 0 (see definition of the LOCK bit). The DSR reset is self-clearing (exits the reset state automatically) while the DOR reset remains in the reset state until software writes the DOR reset bit to 0. DOR reset has precedence over the DSR reset. The DOR reset is set automatically when a hard reset or configuration reset occurs. Software must set the DOR reset bit to 0 to exit the reset state.

The AC Specifications gives the minimum amount of time that the DOR reset must be held active. This amount of time that the DOR reset must be held active is dependent upon the data rate. FDC requires that the DOR reset bit must be held active for at least 0.5 µs at 250 Kbps. This is less than a typical ISA I/O cycle time.

8.3 DMA Transfers

DMA transfers are enabled with the SPECIFY Command. When enabled, The FDC initiates DMA transfers by asserting the FDDREQ signal during a data transfer command. The FIFO is enabled directly by asserting FDDACK# and addresses need not be valid.

8.4 Controller Phases

The FDC handles commands in three phases—*command*, *execution* and *result*. Each phase is described in the following sections. When not processing a command, the FDC can be in the *idle*, *drive polling* or *powerdown state*. This section describes the command, execute and result phases.

8.4.1 COMMAND PHASE

After a reset, the FDC enters the command phase and is ready to accept a command from the host. For each of the commands, a defined set of command code bytes and parameter bytes must be written to the FDC (as described in Section 8.8, Command Set Description) before the command phase is complete. These bytes of data must be transferred in the order described.

Before writing to the FDC, the host must examine the RQM and DIO bits of the Main Status Register. RQM must be 1 and DIO must be 0, before command bytes may be written. The FDC sets RQM to 0 after each write cycle and keeps the bit at 0 until the received byte is processed. After processing the byte, the FDC sets RQM to 1 again to request the next parameter byte of the command, unless an illegal command condition is detected. After the last parameter byte is received, RQM remains 0, and the FDC automatically enters the next phase (execution or result phase) as defined by the command definition.

The FIFO is disabled during the command phase to retain compatibility with the 8272A, and to provide for the proper handling of the Invalid Command condition.

8.4.2 EXECUTION PHASE

The following paragraphs detail the operation of the FIFO flow control. In these descriptions, threshold is defined as the number of bytes available to the FDC when service is requested from the host, and ranges from 1 to 16. The FIFOTHR parameter, which the user programs, is one less and ranges from 0 to 15.

A low threshold value (e.g., 2) results in longer periods of time between service requests but requires faster servicing of the request for both read and write cases. The host reads (writes) from (to) the FIFO until empty (full), then the transfer request goes inactive. The host must be very responsive to the service request. This is the desired case for use with a "fast" system.

A high value of threshold (e.g., 12) is used with a "sluggish" system by affording a long latency period after a service request, but results in more frequent service requests.

8.4.2.1 Non-DMA Mode Transfers from the FIFO to the Host

The IRQ6 pin and RQM bits in the Main Status Register are activated when the FIFO contains 16 (or set threshold) bytes, or the last bytes of a full sector transfer have been placed in the FIFO. The IRQ6 pin can be used for interrupt driven systems and RQM can be used for polled sytems. The host must respond to the request by reading data from the FIFO. This process is repeated until the last byte is transferred out of the FIFO, then FDC negates the IRQ6 pin and RQM bit.

8.4.2.2 Non-DMA Mode Transfers from the Host to the FIFO

The IRQ6 pin and RQM bit in the Main Status Register are activated upon entering the execution phase of data transfer commands. The host must respond to the request by writing data into the FIFO. The IRQ6 pin and RQM bit remain true until the FIFO becomes full. They are set true again when the FIFO has (threshold) bytes remaining in the FIFO. The IRQ6 pin is also negated if TC and DACK# both go inactive. The FDC enters the result phase after the last byte is taken by the FDC from the FIFO (i.e. FIFO empty condition).

8.4.2.3 DMA Mode Transfers from the FIFO to the Host

The FDC asserts the FDDREQ signal when the FIFO contains 16 (or set threshold) bytes or the last byte of a full sector transfer has been placed in the FIFO. The DMA controller must respond to the request by reading data from the FIFO. The FDC negates FDDREQ when the FIFO is empty. FDDREQ is negated after FDDACK# is asserted for the last byte of a data transfer (or on the active edge of RD#, on the last byte, if no edge is present on FDDACK#).

NOTE:

FDDACK# and TC must overlap for at least 50 ns for proper functionality. A data underrun may occur if FDDREQ is not removed in time to prevent an unwanted cycle.

8.4.2.4 DMA Mode Transfers from the Host to the FIFO

The FDC asserts FDDREQ when entering the execution phase of data transfer commands. The DMA controller must respond by asserting FDDACK# and WR# signals and placing data in the FIFO. FDDREQ remains asserted until the FIFO becomes full. FDDREQ is again asserted when the FIFO has (threshold) bytes remaining in the FIFO. The FDC also negates the FDDREQ when the FIFO becomes empty (qualified by DACK# and TC overlapping by 50 ns) indicating that no more data is required. FDDREQ is negated after FDDACK# is asserted for the last byte of a data transfer (or on the active edge of WR# of the last byte, if no edge is present on DACK#). A data overrun may occur if FDDREQ is not removed in time to prevent an unwanted cycle.

8.4.3 DATA TRANSFER TERMINATION

The FDC supports terminal count explicitly through the TC signal and implicitly through the underrun/overrun and end-of-track (EOT) functions. For full sector transfers, the EOT parameter can define the last sector to be transferred in a single or multi-sector transfer. If the last sector to be transferred is a partial sector, the host can stop transferring the data in mid-sector and the FDC will continue to complete the sector as if a hardware TC was received. The only difference between these implicit functions and TC is that they return "abnormal termination" result status. Such status indications can be ignored if they were expected.

> **NOTE:**
> When the host is sending data to the FIFO, the internal sector count will be complete when the FDC reads the last byte from its side of the FIFO. There may be a delay in the removal of the transfer request signal of up to the time taken for the FDC to read the last 16 bytes from the FIFO. The host must be able to tolerate this. In a DMA system, FDDREQ is removed (negated) as soon as TC is received indicating the termination of the transfer. The reception of TC also generates an interrupt on IRQ6. However, in a non-DMA system the interrupt will not be generated until the FIFO is empty.

The generation of IRQ6 determines the beginning of the result phase. For each of the commands, a defined set of result bytes has to be read from the FDC before the result phase is complete (refer to Section 8.5, Command Set/Descriptions). These bytes of data must be read out for another command to start.

RQM and DIO must both be 1 before the result bytes may be read from the FIFO. After all the result bytes have been read, RQM = 1, DIO = 0, and CMDBUSY = 0 in the MSR. This indicates that the FDC is ready to accept the next command.

8.5 Command Set/Descriptions

Commands can be written whenever the FDC is in the command phase. Each command has a unique set of needed parameters and status results. The FDC checks to see that the first byte is a valid command and, if valid, proceeds with the command. If it was invalid, the next time the RQM bit in the MSR register is 1 the DIO and CB bits will also be 1, indicating the FIFO must be read. A result byte of 80h will be read out of the FIFO, indicating an invalid command was issued. After reading the result byte from the FIFO, the FDC returns to the command phase. Table 23 shows the FDC Command set.

Table 24. FDC Command Set

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
Read Data										
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	......	......	......	C		......	......	......	Sector ID
	W	......	......	......	H		......	......	......	Information Prior to
	W	......	......	......	R		......	......	......	Command
										Execution
	W	......	......	......	N		......	......	......	
	W	......	......	......	EOT		......	......	......	
	W	......	......	......	GPL		......	......	......	
	W	......	......	......	DTL		......	......	......	
Execution										Data Transfer
										Between the FDD
										and System
Result	R	......	......	......	ST 0		......	......	......	Status Information
	R	......	......	......	ST 1		......	......	......	After Command
	R	......	......	......	ST 2		......	......	......	Execution
	R	......	......	......	C		......	......	......	
	R	......	......	......	H		......	......	......	Sector ID
	R	......	......	......	R		......	......	......	Information After
	R	......	......	......	N		......	......	......	Command
										Execution
Read Deleted Data										
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	......	......	......	C		......	......	......	Sector ID
	W	......	......	......	H		......	......	......	Information Prior to
	W	......	......	......	R		......	......	......	Command
										Execution
	W	......	......	......	N		......	......	......	
	W	......	......	......	EOT		......	......	......	
	W	......	......	......	GPL		......	......	......	
	W	......	......	......	DTL		......	......	......	
Execution										Data Transfer
										Between the FDD
										and System
Result	R	......	......	......	ST 0		......	......	......	Status Information
	R	......	......	......	ST 1		......	......	......	After Command
	R	......	......	......	ST 2		......	......	......	Execution
	R	......	......	......	C		......	......	......	
	R	......	......	......	H		......	......	......	Sector ID
	R	......	......	......	R		......	......	......	Information After
	R	......	......	......	N		......	......	......	Command
										Execution

Table 24. FDC Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
Write Data										
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID Information Prior to Command Execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data Transfer Between the FDD and System
Result	R	ST 0								Status Information After Command Execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Sector ID Information After Command Execution
	R	R								
	R	N								
Write Deleted Data										
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	C								Sector ID Information Prior to Command Execution
	W	H								
	W	R								
	W	N								
	W	EOT								
	W	GPL								
	W	DTL								
Execution										Data Transfer Between the FDD and System
Result	R	ST 0								Status Information After Command Execution
	R	ST 1								
	R	ST 2								
	R	C								
	R	H								Sector ID Information After Command Execution
	R	R								
	R	N								

Table 24. FDC Command Set (Continued)

<table>
<tr><th rowspan="2">Phase</th><th rowspan="2">R/W</th><th colspan="8">Data Bus</th><th rowspan="2">Remarks</th></tr>
<tr><th>D7</th><th>D6</th><th>D5</th><th>D4</th><th>D3</th><th>D2</th><th>D1</th><th>D0</th></tr>
<tr><th colspan="11">Read Track</th></tr>
<tr><td>Command</td><td>W</td><td>0</td><td>MFM</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>Command Codes</td></tr>
<tr><td></td><td>W</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>HDS</td><td>DS1</td><td>DS0</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. C ....</td><td rowspan="3">Sector ID Information Prior to Command Execution</td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. H ....</td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. R ....</td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. N ....</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. EOT</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. GPL ....</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. DTL ....</td><td></td></tr>
<tr><td>Execution</td><td></td><td colspan="8"></td><td>Data Transfer Between the FDD and System. FDC Reads All Sectors From Index Hole to EOT</td></tr>
<tr><td>Result</td><td>R</td><td colspan="8">.................. ST 0 ....</td><td rowspan="3">Status Information After Command Execution</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. ST 1 ....</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. ST 2 ....</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. C ....</td><td></td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. H ....</td><td rowspan="3">Sector ID Information After Command Execution</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. R ....</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. N ....</td></tr>
<tr><th colspan="11">Verify</th></tr>
<tr><td>Command</td><td>W</td><td>MT</td><td>MFM</td><td>SK</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>Command Codes</td></tr>
<tr><td></td><td>W</td><td>EC</td><td>0</td><td>0</td><td>0</td><td>0</td><td>HDS</td><td>DS1</td><td>DS0</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. C ....</td><td rowspan="3">Sector ID Information Prior to Command Execution</td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. H ....</td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. R ....</td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. N ....</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. EOT ....</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. GPL ....</td><td></td></tr>
<tr><td></td><td>W</td><td colspan="8">.................. DTL/SC ....</td><td></td></tr>
<tr><td>Execution</td><td></td><td colspan="8"></td><td>Data Transfer Between the FDD and System</td></tr>
<tr><td>Result</td><td>R</td><td colspan="8">.................. ST 0 ....</td><td rowspan="3">Status Information After Command Execution</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. ST 1 ....</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. ST 2 ....</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. C ....</td><td></td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. H ....</td><td rowspan="3">Sector ID Information After Command Execution</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. R ....</td></tr>
<tr><td></td><td>R</td><td colspan="8">.................. N ....</td></tr>
</table>

Table 24. FDC Command Set (Continued)

Phase	R/W	Data Bus									Remarks
		D7	D6	D5	D4		D3	D2	D1	D0	
						Version					
Command	W	0	0	0	1		0	0	0	0	Command Codes
Result	W	1	0	0	1		0	0	0	0	Enhanced Controller
						Format Track					
Command	W	0	MFM	0	0		1	1	0	1	Command Codes
	W	0	0	0	0		0	HDS	DS1	DS0	
	W					N					Bytes/Sector
	W					SC					Sector/Cylinder
	W					GPL					Gap 3
	W					D					Filler Byte
Execution											
For Each	W					C					
Sector	W					H					Input Sector
Repeat:	W					R					Parameters
	W					N					
											FDC Formats an
											Entire Cylinder
Result	R					ST 0					Status Information
	R					ST 1					after Command
	R					ST 2					Execution
	R					Undefined					
	R					Undefined					
	R					Undefined					
	R					Undefined					
						Scan Equal					
Command	W	MT	MFM	SK	1		0	0	0	0	Command Codes
	W	0	0	0	0		0	HDS	DS1	DS0	
	W					C					Sector ID
	W					H					Information Prior
	W					R					to Command
	W					N					Execution
	W					EOT					
	W					GPL					
	W					STP					
Execution											
											Data Compared
											Between the FDD
											and Main-System
Result	R					ST 0					Status Information
	R					ST 1					After Command
	R					ST 2					Execution
	R					C					
	R					H					Sector ID
	R					R					Information After
	R					N					Command
											Execution

Table 24. FDC Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
Scan Low or Equal										
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W				C					Sector ID Information Prior to Command Execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				STP					
Execution										Data Compared Between the FDD and Main-System
Result	R				ST 0					Status Information After Command Execution
	R				ST 1					
	R				ST 2					
	R				C					
	R				H					Sector ID Information After Command Execution
	R				R					
	R				N					
Scan High or Equal										
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W				C					Sector ID Information Prior to Command Execution
	W				H					
	W				R					
	W				N					
	W				EOT					
	W				GPL					
	W				STP					
Execution										Data Compared Between the FDD and Main-System
Result	R				ST 0					Status Information After Command Execution
	R				ST 1					
	R				ST 2					
	R				C					
	R				H					Sector ID Information After Command Execution
	R				R					
	R				N					
Recalibrate										
Command	W	0	0	0	0	0	1	1	1	Command Codes Enhanced Controller
	W	0	0	0	0	0	0	DS0	DS1	
Execution										Head Retracted to Track 0 Interrupt

Table 24. FDC Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
Sense Interrupt Status										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R	....			ST 0				....	Status Information at the End of Each Seek Operation
	R	....			PCN				....	
Specify										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	....	SRT		....	.....	HUT		.....	
	W	....		HLT	....				ND	
Sense Drive Status										
Command	W	0	0	0	0	0	1	0	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Result	R	....			ST 3				....	Status Information About FDD
Drive Specification Command										
Command	W	1	0	0	0	1	1	1	0	Command Code
	W	0	FD1	FD0	PTS	DRT1	DRT0	DT1	DT0	
	:	:	:	:	:	:	:	:	:	0–4 bytes issued
	W	DN	NRP	0	0	0	0	0	0	
Result	R	0	0	0	PTS	DRT1	DRT0	DT1	DT0	Drive 0
	R	0	0	0	PTS	DRT1	DRT0	DT1	DT0	Drive 1
	R	0	0	0	0	0	0	0	0	RSVD
	R	0	0	0	0	0	0	0	0	RSVD
Seek										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
Execution	W	....			NCN				....	Head is Positioned Over Proper Cylinder on Diskette
Configure										
Command	W	0	0	0	1	0	0	1	1	Command Code
	W	0	0	0	0	0	0	0	0	
	W	0	EIA	EFIFO	POLL	.....	FIFOTHR		....	
	W	....			PRETRK				....	
Relative Seek										
Command	W	1	DIR#	0	0	1	1	1	1	Command Code
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	....			RCN				....	

Table 24. FDC Command Set (Continued)

Phase	R/W	Data Bus D7	D6	D5	D4	D3	D2	D1	D0	Remarks
DUMPREG										
Command Execution	W	0	0	0	0	1	1	1	0	Note: Registers placed in FIFO
Result	R	....	....	....		PCN-Drive 0	....	....	....	
	R	....	....	....		PCN-Drive 1	....	....	....	
	R	....	....	....		PCN-Drive 2	....	....	....	
	R	....	....	....		PCN-Drive 3	....	....	....	
	R	.....	SRT	....	....	....	HUT	....	....	
	R	....	....	HLT	....	....	....	....	ND	
		....	....	....		SC/EOT	....	....	....	
	R	LOCK	0	0	0	D1	D0	GAP	WGATE	
	R	0	EIS	EFIFO		POLL	FIFOTHR	....	....	
	R	....	....	....		PRETRK	....	....	....	
Read ID										
Command	W	0	MFM	0	0	1	0	1	0	Commands
	W	0	0	0	0	0	HDS	DS1	DS0	
										The First Correct ID Information on the Cylinder is Stored in Data Register
Result	R	....	....	....		ST 0	....	....	....	Status Information After Command Execution
	R	....	....	....		ST 1	....	....	....	
	R	....	....	....		ST 2	....	....	....	
	R	....	....	....		C	....	....	....	
	R	....	....	....		H	....	....	....	Disk Status After the Command has Completed
	R	....	....	....		R	....	....	....	
	R	....	....	....		N	....	....	....	
Perpendicular Mode										
Command	W	0	0	0	1	0	0	1	0	Command Codes
	W	OW	0	0	0	D1	D0	GAP	WGATE	
Lock										
Command	W	LOCK	0	0	1	0	1	0	0	Command Codes
Result	R	0	0	0	LOCK	0	0	0	0	
Part ID										
Command	W	0	0	0	1	1	0	0	0	Command Code
Result	R	0	0	0	.....	Stepping	....	....	1	Part ID Number
Powerdown Mode										
Command	W	0	0	0	1	0	1	1	1	Command Code
	W	0	0	EREG EN	0	0	FDI TRI	MIN DLY	AUTO PD	
Result	R	0	0	EREG EN	0	0	FDI TRI	MIN DLY	AUTO PD	

Table 24. FDC Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
					Option					
Command	W	0	0	1	1	0	0	1	1	Command Code
	W	....		RSVD		....	....	....	ISO	
					Save					
Command	W	0	0	1	0	1	1	1	0	Command Code
Result	R	RSVD	RSVD	PD OSC	PC2	PC1	PC0	DRATE1	DRATE0	Save Information to Reprogram the FDC
		0	0	0	0	0	0	0	ISO	
	R	....	....	....		PCN-Drive 0	....	....	....	
	R	....	....	....		PCN-Drive 1	....	....	....	
	R	....	....	....		PCN-Drive 2	....	....	....	
	R	....	....	....		PCN-Drive 3	....	....	....	
	R	.....	SRT	.....		....	HUT	....	....	
	R	....	....	....		HLT	....	....	ND	
	R	....	....	....		SC/EOT	....	....	....	
	R	LOCK	0	0	0	D1	D0	GAP	WGATE	
	R	0	EIS	EFIFO	POLL	—	FIFOTHR		....	
	R	....	....	....		PRETRK	....	....	....	
	R	0	0	EREG EN	0	RSVD	FDI TRI	MIN DLY	AUTO PD	
	R	....	....	....		DISK/STATUS	....	....	....	
	R	....	....	....		RSVD	....	....	....	
	R	....	....	....		RSVD	....	....	....	
					Restore					
Command	W	0	1	0	0	1	1	1	0	Command Code
	W	0	0	0	PC2	PC1	PC0	DRATE1	DRATE0	Restore Original Register Status
	W	0	0	0	0	0	0	0	ISO	
	W	....	....	....		PCN-Drive 0	....	....	....	
	W	....	....	....		PCN-Drive 1	....	....	....	
	W	....	....	....		PCN-Drive 2	....	....	....	
	W	....	....	....		PCN-Drive 3	....	....	....	
	W	.....	SRT	.....		....	HUT	....	....	
	W	....	....	....	HLT	....	....	....	ND	
	W	....	....	....		SC/EOT	....	....	....	
	W	LOCK	0	0	0	D1	D0	GAP	WGATE	
	W	0	EIS	EFIFO	POLL	....	FIFOTHR	....	....	
	W	....	....	....		PRETRK	....	....	....	
	W	0	0	EREG EN	0	RSVD	FDI TRI	MIN DLY	AUTO PD	
	W	....	....	....		DISK/STATUS	....	....	....	
	W	....	....	....		RSVD	....	....	....	
	W	....	....	....		RSVD	....	....	....	

Table 24. FDC Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
Format and Write										
Command	W	1	MFM	1	0	1	1	0	1	Command Code
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	N								
	W	SC								
	W	GPL								
	W	D								
Execution repeated for each sector	W	C								Input Sector Parameters
	W	H								
	W	R								
	W	N								
	W	Data Transfer Of N Bytes								
										FDC Formats and Writes Entire Track
Result	R	ST 0								
	R	ST 1								
	R	ST 2								
	R	Undefined								
	R	Undefined								
	R	Undefined								
	R	Undefined								
Invalid										
Command	W	Invalid Codes								Invalid Command Codes (Noop—FDC goes into Standby State)
Result	R	ST 0								ST 0 = 80

Parameter Abbreviations

Symbol **Description**

AUTO PD **AUTO POWERDOWN CONTROL:** When AUTO PD=0, automatic powerdown is disabled. When AUTO PD=1, automatic powerdown is enabled.

C **CYLINDER ADDRESS:** The currently selected cylinder address, 0 to 255.

D0, D1 **DRIVE SELECT 0-1:** Designates which drives are Perpendicular drives. A 1 indicates Perpendicular drive.

D **DATA PATTERN:** The pattern to be written in each sector data field during formatting.

DN **DONE:** This bit indicates that this is the last byte of the drive specification command. The FDC checks to see if this bit is 1 or 0. When DN=0, the FDC expects more bytes.

DN=0 FDC expects more subsequent bytes.

DN=1 Terminates the command phase and enters the results phase. An additional benefit is that by setting this bit to 1, a direct check of the current drive specifications can be done.

DIR# **DIRECTION CONTROL:** When DIR# =0, the head steps out from the spindle during a relative seek. When DIR# =1, the head steps in toward the spindle.

DS0, DS1 **DISK DRIVE SELECT:**

DS1	DS0	Drive Slot
0	0	drive 0
0	1	drive 1
1	0	drive 2*
1	1	drive 3*

*Available when FDDQTY=1 in the FCFG1 Register (see Appendix A, FDC Four Drive Support)

DTL **SPECIAL SECTOR SIZE:** By setting N to zero (00), DTL may be used to control the number of bytes transferred in disk read/write commands. The sector size (N=0) is set to 128. If the actual sector (on the diskette) is larger than DTL, the remainder of the actual sector is read but is not passed to the host during read commands; during write commands, the remainder of the actual sector is written with all zero bytes. The CRC check code is calculated with the actual sector. When N is not zero, DTL has no meaning and should be set to FFh.

DRATE[0:1] **DATA RATE:** Data rate values from the DSR register.

Symbol	Description
DRT0, DRT1	**DATA RATE TABLE SELECT:** These two bits select between the different data rate tables. The default is the conventional table. These also provide mapping of the data rates selected in the DSR and CCR. The table below shows this.

Bits in DSR					
DRT1	**DRT0**	**DRATE1**	**DRATE0**	**Data Rate**	**Operation**
0	0	1	1	1 Mbps	Default
		0	0	500 Kbps	
		0	1	300 Kbps	
		1	0	250 Kbps	
0	1	RSVD	RSVD	RSVD	RSVD
1	0	RSVD	RSVD	RSVD	RSVD
1	1	1	1	1 Mbps	Perpendicular mode FDDs
		0	0	500 Kbps	
		0	1	Illegal	
		1	0	250 Kbps	

DT0,DT1	**DRIVE DENSITY SELECT TYPE:** These bits select the outputs on DRVDEN0 and DRVDEN1 (see DRIVE SPECIFICATION Command).
EC	**ENABLE COUNT:** When EC = 1, the DTL parameter of the Verify Command becomes SC (Number of sectors per track).
EFIFO	**Enable FIFO:** When EFIFO = 0, the FIFO is enabled. EFIFO = 1 puts the FDC in the 8272A compatible mode where the FIFO is disabled.
EIS	**ENABLE IMPLIED SEEK:** When EIS = 1, a seek operation is performed before executing any read or write command that requires the C parameter in the command phase. EIS = 0 disables the implied seek.
EOT	**END OF TRACK:** The final sector number of the current track.
EREG EN	**ENHANCED REGISTER ENABLE:** When EREG EN = 1, the TDR register is extended and SRB is made visible to the user. When EREG EN = 0, the standard registers are used.
FDI TRI	**FLOPPY DRIVE INTERFACE TRI-STATE:** When FDI TRI = 0, the output pins of the floppy disk drive interface are tri-stated. This is also the default state. When FDI TRI = 1, the floppy disk drive interface remains unchanged.

Symbol **Description**

FD0, FD1 **FLOPPY DRIVE SELECT:** These two bits select which physical drive is being specified. The FDn corresponds to FDSn and FDMEn on the floppy drive interface. The drive is selected independent of the BOOTSEL bit in the TDR. Refer to Section 8.1.3, TDR—Enhanced Tape Drive Register, which explains the distinction between physical drives and their virtual mapping as defined by the BOOTSEL bit.

FD1	FD0	Drive slot
0	0	drive 0
1	0	drive 1
0	1	drive 2*
1	1	drive 3*

*Available if the four floppy drive option is selected in the FCFG1 Register.

GAP **GAP:** Alters Gap 2 length when using Perpendicular Mode.

GPL **GAP LENGTH:** The gap 3 size. (Gap 3 is the space between sectors excluding the VCO synchronization field).

H/HDS **HEAD ADDRESS:** Selected head: 0 or 1 (disk side 0 or 1) as encoded in the sector ID field.

HLT **HEAD LOAD TIME:** The time interval that FDC waits after loading the head and before initiating a read or write operation. Refer to the SPECIFY Command for actual delays.

HUT **HEAD UNLOAD TIME:** The time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Refer to the SPECIFY Command for actual delays.

ISO **ISO FORMAT:** When ISO = 1, the ISO format is used for all data transfer commands. When ISO = 0, the normal IBM system 34 and perpendicular is used. The default is ISO = 0.

LOCK **LOCK:** Lock defines whether EFIFO, FIFOTHR, and PRETRK parameters of the CONFIGURE Command can be reset to their default values by a software reset (Reset made by setting the proper bit in the DSR or DOR registers).

MFM **MFM MODE:** A one selects the double density (MFM) mode. A zero is reserved.

Symbol **Description**

MIN DLY **MINIMUM POWERUP TIME CONTROL:** This bit is active only if AUTO PD bit is enabled. When MIN DLY = 0, a 10 ms minimum powerup time is assigned and when MIN DLY = 1, a 0.5 sec. minimum powerup time is assigned.

MT **MULTI-TRACK SELECTOR:** When MT = 1, the multi-track operating mode is selected. In this mode, the FDC treats a complete cylinder, under head 0 and 1, as a single track. The FDC operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set, a multitrack read or write operation will automatically continue to the first sector under head 1 when the FDC finishes operating on the last sector under head 0.

N **SECTOR SIZE CODE:** This specifies the number of bytes in a sector. When N = 00h, the sector size is 128 bytes. The number of bytes transferred is determined by the DTL parameter. Otherwise the sector size is (2 raised to the "N'th" power) times 128. All values up to 07h are allowable. A value of 07h equals a sector size of 16 Kbytes. It is the users responsibility to not select combinations that are not possible with the drive.

N	Sector Size
00	128 bytes
01	256 bytes
02	512 bytes
03	1024
..	...
07	16 Kbytes

NCN **NEW CYLINDER NUMBER:** The desired cylinder number.

ND **NON-DMA MODE FLAG:** When ND = 1, the FDC operates in the non-DMA mode. In this mode, the host is interrupted for each data transfer. When ND = 0, the FDC operates in DMA mode and interfaces to a DMA controller by means of the DRQ and DACK# signals.

NRP **NO RESULTS PHASE:** When NRP = 1, the result phase is skipped. When NRP = 0, the result phase is generated.

OW **OVERWRITTEN:** The bits denoted D0 and D1 of the PERPENDICULAR MODE Command can only be overwritten when OW = 1.

Symbol	Description
PCN	**PRESENT CYLINDER NUMBER:** The current position of the head at the completion of SENSE INTERRUPT STATUS Command.
PC2,PC1,PC0	**PRECOMPENSATION VALUES:** Precompensation values from the DSR register.
PDOSC	**POWERDOWN OSCILLATOR:** When this bit is set, the internal oscillator is turned off.
PTS	**PRECOMPENSATION TABLE SELECT:** This bit selects whether to enable the precompensation value programmed in the DSR or not. In the default state, the value programmed in DSR will be used. More information regarding the precompensation is available in Section 8.1.5. PTS = 0 DSR programmed precompensation delays PTS = 1 No precompensation delay is selected for the corresponding drive.
POLL	**POLLING DISABLE:** When POLL = 1, the internal polling routine is disabled. When POLL = 0, polling is enabled.
PRETRK	**PRECOMPENSATION START TRACK NUMBER:** Programmable from track 00 to FFh.
R	**SECTOR ADDRESS:** The sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of the first sector to be read or written.
RCN	**RELATIVE CYLINDER NUMBER:** Relative cylinder offset from present cylinder as used by the RELATIVE SEEK Command.
SC	**NUMBER OF SECTORS:** The number of sectors to be initialized by the FORMAT Command. The number of sectors to be verified during a Verify Command, when EC = 1.
SK	**SKIP FLAG:** When SK = 1, sectors containing a deleted data address mark will automatically be skipped during the execution of a READ DATA Command. If a READ DELETED DATA Command is executed, only sectors with a deleted address mark will be accessed. When SK = 0, the sector is read or written the same as the read and write commands.
SRT	**STEP RATE INTERVAL:** The time interval between step pulses issued by the FDC. Programmable from 0.5 ms to 8 ms, in increments of 0.5 ms at the 1 Mbit data rate. Refer to the SPECIFY Command for actual delays.
ST0-3	**STATUS REGISTERS 0-3.**Registers within the FDC that store status information after a command has been executed. This status information is available to the host during the result phase after command execution.
WGATE	**WRITE GATE:** Write gate alters timing of WE, to allow for pre-erase loads in perpendicular drives.

8.5.1 STATUS REGISTER ENCODING

The contents of these registers are available only through a command sequence.

8.5.1.1 Status Register 0

Bit #	Symbol	Name	Description
7,6	IC	Interrupt Code	00 Normal termination of command. The specified command was properly executed and completed without error. 01 Abnormal termination of command. Command execution was started, but was not successful completed. 10 Invalid command. The requested command could not be executed. 11 Abnormal termination caused by Polling.
5	SE	Seek End	The 82091AA completed a SEEK or RECALIBRATE command, or a READ or WRITE with implied seek command.
4	EC	Equipment Check	The TRK pin failed to become a "1" after: 1. 80 step pulses in the RECALIBRATE COMMAND. 2. The RELATIVE SEEK command causes the 82078 to step outward beyond Track 0.
3	—	—	Unused. This bit is always "0".
2	H	Head Address	The current head address.
1,0	DS1,0	Drive Select	The current selected drive.

8.5.1.2 Status Register 1

Bit #	Symbol	Name	Description
7	EN	End of Cylinder	The 82078 tried to access a section beyond the final sector of the track (255D). Will be set if TC is not issued after Read or Write Data Command.
6	—	—	Unused. This bit is always "0".
5	DE	Data Error	The 82078 detected a CRC error in either the ID field or the data field of a sector.
4	OR	Overrun/Underrun	Becomes set if the 82078 does not receive CPU or DMA service within the required time interval, resulting in data overrun or underrun.
3	—	—	Unused. Ths bit is always "0".
2	ND	No Data	Any one of the following: 1. READ DATA, READ DELETED DATA command, the 82091AA did not find the specified sector. 2. READ ID command, the 82091AA cannot read the ID field without an error. 3. READ TRACK command, the 82091AA cannot find the proper sector sequence.
1	NW	Not Writable	WP pin became a "1" while the 82091AA is executing a WRITE DATA, WRITE DELETED DATA, or FORMAT TRACK command.
0	MA	Missing Address Mark	Any one of the the following: 1. The 82091AA did not detect an ID address mark at the specified track after encountering the index pulse from the INDX# pin twice. 2. The 82091AA cannot detect a data address mark or a deleted data address mark on the specified track.

8.5.1.3 Status Register 2

Bit #	Symbol	Name	Description
7	—	—	Unused. This bit is always "0".
6	CM	Control Mark	Any one of the following: 1. READ DATA command, the 82078 encounters a deleted data address mark. 2. READ DELETED DATA command, the 82078 encountered a data address mark.
5	DD	Data Error in Data Field	The 82091AA detected a CRC error in the data field.
4	WC	Wrong Cylinder	The track address from the sector ID field is different from the track address maintained inside the 82091AA.
3	—	—	Unused. This bit is always "0".
2	—	—	Unused. This bit is always "0".
1	BC	Bad Cylinder	The track address from the sector ID field is different from the track address maintained inside the 82091AA and is equal to FF hex which indicates a bad track with a hard error according to the IBM soft-sectored format.
0	MD	Missing Data Address Mark	The 82091AA cannot detect a data address mark or a deleted data address mark.

8.5.1.4 Status Register 3

Bit #	Symbol	Name	Description
7	—	—	Unused. This bit is always "0".
6	WP	Write Protected	Indicates the status of the WP pin.
5	—	—	Unused. This bit is always "0".
4	T0	Track 0	Indicates the status of the TRK0 pin.
3	—	—	Unused. This bit is always "0".
2	HD	Head Address	Indicates the status of the HDSEL pin.
1,0	DS1,0	Drive Select	Indicates the status of the DS1, DS0 pins.

8.5.2 DATA TRANSFER COMMANDS

All of the READ DATA, WRITE DATA and VERIFY type commands use the same parameter bytes and return the same results information. The only difference being the coding of bits[4:0] in the first byte.

An implied seek will be executed if the feature was enabled by the CONFIGURE Command. This seek is completely transparent to the user. The Drive Busy bit for the drive will go active in the Main Status Register during the seek portion of the command. A seek portion failure is reflected in the results status normally returned for a READ/WRITE DATA Command. Status Register 0 (ST0) contains the error code and C contains the cylinder that the seek failed.

8.5.2.1 Read Data

A set of nine bytes is required to place the FDC into the Read Data Mode. After the READ DATA Command has been issued, the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the SPECIFY Command), and begins reading ID address marks and ID fields. When the sector address read from the diskette matches with the sector address specified in the command, the FDC reads the sector's data field and transfers the data to the FIFO.

After completion of the read operation from the current sector, the sector address is incremented by one, and the data from the next logical sector is read and output via the FIFO. This continuous read function is called "Multi-Sector Read Operation". Upon receipt of TC or an implied TC (FIFO overrun/underrun), the FDC stops sending data. However, the FDC will continue to read data from the current sector, check the CRC bytes, and, at the end of the sector, terminate the READ DATA Command.

N determines the number of bytes per sector (Table 25). If N is set to zero, the sector size is set to 128. The DTL value determines the number of bytes to be transferred. If DTL is less than 128, the FDC transfers the specified number of bytes to the host. For reads, it continues to read the entire 128 byte sector and checks for CRC errors. For writes it completes the 128 byte sector by filling in zeroes. If N is not set to 00h, DTL should be set to FFh, and has no impact on the number of bytes transferred.

Table 25. Sector Sizes

N	Sector Size
00	128 Bytes
01	256 Bytes
02	512 Bytes
03	1024 Bytes
...	...
07	16 KBytes

The amount of data that can be handled with a single command to the FDC depends on MT (multi-track) and N (Number of bytes/sector).

Table 26. Effects of MT and N Bits

MT	N	Max. Transfer Capacity	Final Sector Read from Disk
0	1	256 × 26 = 656	26 at side 0 or 1
1	1	256 × 52 = 13312	26 at side 1
0	2	512 × 15 = 7680	15 at side 0 or 1
1	2	512 × 30 = 15360	15 at side 1
0	3	1024 × 8 = 8192	8 at side 0 or 1
1	3	1024 × 16 = 16384	16 at side 1

The Multi-Track function (MT) allows the FDC to read data from both sides of the diskette. For a particular cylinder, data will be transferred starting at sector 1, side 0 and completing at the last sector of the same track at side 1.

If the host terminates a read or write operation in the FDC, the ID information in the result phase is dependent on the state of the MT bit and EOT byte. Refer to Table 29. The termination must be normal.

At the completion of the READ DATA Command, the head is not unloaded until after the Head Unload Time Interval (specified in the SPECIFY Command) has elapsed. If the host issues another command before the head unloads, the head settling time may be saved between subsequent reads.

If the FDC detects a pulse on the INDEX# pin twice without finding the specified sector (meaning that the diskette's index hole passes through index detect logic in the drive twice), the FDC sets the IC code in Status Register 0 to 01 (Abnormal termination), sets the ND bit in Status Register 1 to 1 indicating a sector not found and terminates the READ DATA Command.

After reading the ID and data fields in each sector, the FDC checks the CRC bytes. If a CRC error occurs in the ID or data field, the FDC sets the IC code in Status Register 0 to 01 (Abnormal termination), sets the DE bit flag in Status Register 1 to 1, sets the DD bit in Status Register 2 to 1 if CRC is incorrect in the ID field, and terminates the READ DATA Command.

Table 27 describes the affect of the SK bit on the READ DATA command execution and results.

8.5.2.2 Read Deleted Data

This command is the same as the READ DATA Command, except that it operates on sectors that contain a deleted data address mark at the beginning of a data field. Table 28 describes the affect of the SK bit on the READ DELETED DATA Command execution and results.

Table 27. Skip Bit vs READ DATA Command

SK Bit Value	Data Address Mark Type Encountered	Sector Read	Results CM Bit of ST2 Set?	Description of Results
0	Normal Data	Yes	No	Normal Termination
0	Deleted Data	Yes	Yes	Address Not Incremented. Next Sector Not Searched For.
1	Normal Data	Yes	No	Normal Termination
1	Deleted Data	No	Yes	Normal Termination Sector Not Read ("Skipped")

Except where noted in Table 27, the C or R value of the sector address is automatically incremented (see Table 29).

Table 28. Skip Bit vs READ DELETED DATA Command

SK Bit Value	Data Address Mark Type Encountered	Sector Read	Results CM Bit of ST2 Set?	Description of Results
0	Normal Data	Yes	Yes	Normal Termination
0	Deleted Data	Yes	No	Address Not Incremented. Next Sector Not Searched For.
1	Normal Data	No	Yes	Normal Termination Sector Not Read ("Skipped")
1	Deleted Data	Yes	No	Normal Termination

Except where noted in Table 28, the C or R value of the sector address is automatically incremented (see Table 29).

Table 29. Result Phase

MT	Head	Final Sector Transferred to Host	ID Information at Result Phase			
			C	H	R	N
	0	Less than EOT	NC	NC	R+1	NC
0		Equal to EOT	C+1	NC	01	NC
	1	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	NC	01	NC
	0	Less than EOT	NC	NC	R+1	NC
1		Equal to EOT	NC	LSB	01	NC
	1	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	LSB	01	NC

NOTE:
1. NC=no change; the same value as the one at the beginning of command execution.
2. LSB=least significant bit; the LSB of H is complemented.

8.5.2.3 Read Track

This command is similar to the READ DATA Command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering a pulse on the INDEX# pin, the FDC starts to read all data fields on the track as continuous blocks of data without regard to logical sector numbers. If the FDC finds an error in the ID or DATA CRC check bytes, it continues to read data from the track and sets the appropriate error bits at the end of the command. The FDC compares the ID information read from each sector with the specified value in the command and sets the ND flag to 1 in Status Register 1 if there is no comparison. Multi-track or skip operations are not allowed with this command. The MT and SK bits (Bits D7 and D5 of the first command byte respectively) should always be set to 0.

This command terminates when the EOT specified number of sectors have been read. If the FDC does not find an ID address mark on the diskette after the second occurrence of a pulse on the INDEX# pin, then it sets the IC code in Status Register 0 to 01 (Abnormal termination), sets the MA bit in Status Register 1 to 1, and terminates the command.

8.5.2.4 Write Data

After the WRITE DATA Command has been issued, the FDC loads the head (if it is in the unloaded state), waits the specified head load time if unloaded (defined in the SPECIFY Command), and begins reading ID fields. When the sector address read from the diskette matches the sector address specified in the command, the FDC reads the data from the host via the FIFO, and writes it to the sector's data field.

After writing data into the current sector, the FDC computes the CRC value and writes it into the CRC field at the end of the sector transfer. The sector number stored in R is incremented by one, and the FDC continues writing to the next data field. The FDC continues this multi-sector write operation. If a terminal count signal is received or a FIFO over/under run occurs while a data field is being written, the remainder of the data field is filled with zeros.

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a CRC error in one of the ID fields, it sets the IC code in Status Register 0 to 01 (Abnormal termination), sets the DE bit of Status Register 1 to 1, and terminates the WRITE DATA Command.

The WRITE DATA Command operates in much the same manner as the READ DATA Command. The following items are the same. Please refer to the READ DATA Command for details:

- Transfer Capacity
- EN (End of Cylinder) bit
- ND (No Data) bit
- Head Load, Unload Time Interval
- ID information when the host terminates the command
- Definition of DTL when N=0 and when N does not=0

8.5.2.5 Verify

The VERIFY Command is used to verify the data stored on a disk. This command acts exactly like a READ DATA Command except that no data is transferred to the host. Data is read from the disk, and CRC is computed and checked against the previously stored value.

Because no data is transferred to the host, the TC signal cannot be used to terminate this command. By setting the EC bit to 1, an implicit TC will be issued to the FDC. This implicit TC occurs when the SC value has decrement to 0 (a SC value of 0 verifies 256 sectors). This command can also be terminated by setting the EC bit to 0 and the EOT value equal to the final sector to be checked. When EC=0, DTL/SC should be programmed to 0FFh. Refer to Table 29 and Table 30 for information concerning the values of MT and EC versus SC and EOT value.

Definitions:

# Sectors Per Side = Number of formatted sectors per each side of the disk.

# Sectors Remaining = Number of formatted sectors left that can be read, including side 1 of the disk when MT=1.

Table 30. Verify Command Result Phase

MT	EC	SC/EOT Value	Termination Result
0	0	SC=DTL EOT ≤ # Sectors Per Side	Successful Termination Result Phase Valid
0	0	SC=DTL EOT > # Sectors Per Side	Unsuccessful Termination Result Phase Invalid
0	1	SC ≤ # Sectors Remaining AND EOT ≤ # Sectors Per Side	Successful Termination Result Phase Valid
0	1	SC > # Sectors Remaining OR EOT > # Sectors Per Side	Unsuccessful Termination Result Phase Invalid
1	0	SC=DTL EOT ≤ # Sectors Per Side	Successful Termination Result Phase Valid
1	0	SC=DTL EOT > # Sectors Per Side	Unsuccessful Termination Result Phase Invalid
1	1	SC ≤ # Sectors Remaining AND EOT ≤ # Sectors Per Side	Successful Termination Result Phase Valid
1	1	SC > # Sectors Remaining OR EOT > # Sectors Per Side	Unsuccessful Termination Result Phase Invalid

NOTE:
When MT=1 and the SC value is greater than the number of remaining formatted sectors on Side 0, verification continues on Side 1 of the disk.

8.5.2.6 Format Track

The FORMAT TRACK Command allows an entire track to be formatted. After a pulse from the INDEX# pin is detected, the FDC starts writing data on the disk including gaps, address marks, ID fields and data fields, per the IBM* System 34 (MFM). The particular values written to the gap and data field are controlled by the values programmed into N, SC, GPL, and D which are specified by the host during the command phase. The data field of the sector is filled with the data byte specified by D. The ID field for each sector is supplied by the host. That is, four data bytes per sector are needed by the FDC for C, H, R, and N (cylinder, head, sector number, and sector size, respectively).

After formatting each sector, the host must send new values for C, H, R, and N to the FDC for the next sector on the track. The R value (sector number) is the only value that must be changed by the host after each sector is formatted. This allows the disk to be formatted with nonsequential sector addresses (inter-leaving). This incrementing and formatting continues for the whole track until the FDC encounters a pulse on the INDEX# pin again and it terminates the command.

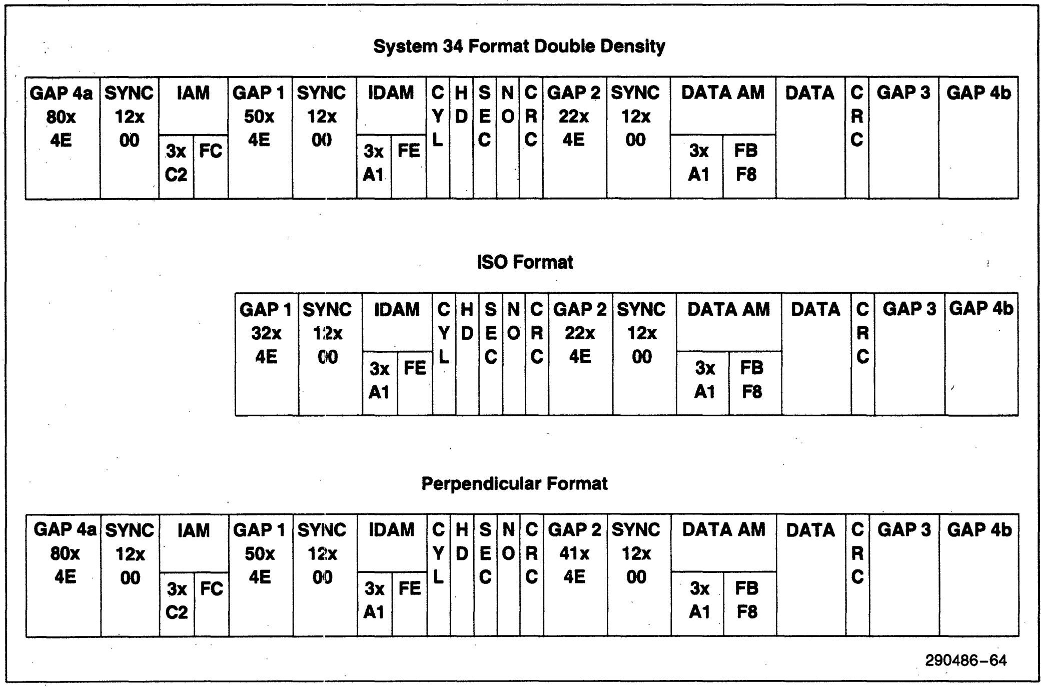
Table 31 contains typical values for gap fields that are dependent on the size of the sector and the number of sectors on each track. Actual values can vary due to drive electronics.

Table 31. Typical PC/AT Values for Formatting

Drive Form	MEDIA	Sector Size	N	SC	GPL1	GPL2
5.25″	1.2 MB	512	02	0F	2A	50
	360 KB	512	02	09	2A	50
3.5″	2.88 MB	512	02	24	38	53
	1.44 MB	512	02	18	1B	54
	720 KB	512	02	09	1B	54

NOTES:
1. All values are in hex, except sector size.
2. Gap3 is programmable during reads, writes, and formats.
3. GPL1 = suggested Gap3 values in read and write commands to avoid splice point between data field and ID field of contiguous sections.
4. GPL2 = suggested Gap3 value in FORMAT TRACK Command.

8.5.2.7 Format Field



System 34 Format Double Density

GAP 4a	SYNC	IAM		GAP 1	SYNC	IDAM		CYL	HD	SEC	NO	CRC	GAP 2	SYNC	DATA AM		DATA	CRC	GAP 3	GAP 4b
80x 4E	12x 00	3x C2	FC	50x 4E	12x 00	3x A1	FE						22x 4E	12x 00	3x A1	FB F8				

ISO Format

GAP 1	SYNC	IDAM		CYL	HD	SEC	NO	CRC	GAP 2	SYNC	DATA AM		DATA	CRC	GAP 3	GAP 4b
32x 4E	12x 00	3x A1	FE						22x 4E	12x 00	3x A1	FB F8				

Perpendicular Format

GAP 4a	SYNC	IAM		GAP 1	SYNC	IDAM		CYL	HD	SEC	NO	CRC	GAP 2	SYNC	DATA AM		DATA	CRC	GAP 3	GAP 4b
80x 4E	12x 00	3x C2	FC	50x 4E	12x 00	3x A1	FE						41x 4E	12x 00	3x A1	FB F8				

290486–64

Figure 64. System 34, ISO and Perpendicular Formats

8.5.3 CONTROL COMMANDS

Control commands differ from the other commands in that no data transfer takes place. Three commands generate an interrupt when complete; READ ID, RECALIBRATE and SEEK. The other control commands do not generate an interrupt.

8.5.3.1 READ ID Command

The READ ID Command is used to find the present position of the recording heads. The FDC stores the values from the first ID field it is able to read into its registers. If the FDC does not find an ID address mark on the diskette after the second occurrence of a pulse on the INDEX# pin, it then sets the IC code in Status Register 0 to 01 (Abnormal termination), sets the MA bit in Status Register 1 to 1, and terminates the command.

The following commands will generate an interrupt upon completion. They do not return any result bytes. It is recommended that control commands be followed by the SENSE INTERRUPT STATUS Command. Otherwise, valuable interrupt status information will be lost.

8.5.3.2 RECALIBRATE Command

This command causes the read/write head within the FDC to retract to the track 0 position. The FDC clears the contents of the PCN counter, and checks the status of the TRK0 pin from the FDD. As long as the TRK0 pin is low, the DIR# pin remains 0 and step pulses are issued. When the TRK0 pin goes high, the SE bit in Status Register 0 is set to 1, and the command is terminated. If the TRK0 pin is still low after 79 step pulses have been issued, the FDC sets the SE and the EC bits of Status Register 0 to 1 and terminates the command. Disks capable of handling more than 80 tracks per side may require more than one RECALIBRATE Command to return the head back to physical Track 0.

The RECALIBRATE Command does not have a result phase. The SENSE INTERRUPT STATUS Command must be issued after the RECALIBRATE Command to effectively terminate it and to provide verification of the head position (PCN). During the command phase of the recalibrate operation, the FDC is in the busy state, but during the execution phase it is in a non-busy state. At this time another RECALIBRATE Command may be issued, and in this manner, parallel RECALIBRATE operations may be done on up to 2 drives simultaneously.

After powerup, software must issue a RECALIBRATE Command to properly initialize all drives and the controller.

8.5.3.3 DRIVE SPECIFICATION Command

The FDC uses two pins, DRVDEN0 and DRVDEN1 to select the density for modern drives. These signals inform the drive of the type of diskette in the drive. The DRIVE SPECIFICATION Command specifies the polarity of the DRVDEN0 and DRVDEN1 pins. It also enables/disables DSR programmed precompensation.

This command removes the need for a hardware work-around to accommodate differing specifications among drives. By programming this command during BIOS's POST routine, the floppy disk controller internally configures the correct values for DRVDEN0 and DRVDEN1 with corresponding precompensation value and data rate table enabled for the particular type of drive.

This command is protected from software resets. After executing the DRIVE SPECIFICATION Command, subsequent software resets will not clear the programmed parameters. Only another DRIVE SPECIFICATION Command or hard reset can reset it to default values. The 6 LSBs of the last byte of this command are reserved for future use.

The DRATE0 and DRATE1 are values as programmed in the DSR register. See Table 32 for pin decoding at different data rates.

Table 32 describes the drives that are supported with the DT0, DT1 bits of the DRIVE SPECIFICATION Command:

Table 32. DRVDENn Polarities

DT1	DT0	Data Rate	DRVDEN1	DRVDEN0
0*	0*	1 Mbps	1	1
		500 Kbps	0	1
		300 Kbps	1	0
		250 Kbps	0	0
0	1	1 Mbps	1	0
		500 Kbps	0	0
		300 Kbps	1	1
		250 Kbps	0	1
1	0	1 Mbps	1	1
		500 Kbps	0	0
		300 Kbps	1	0
		250 Kbps	0	1
1	1	1 Mbps	1	1
		500 Kbps	0	0
		300 Kbps	0	1
		250 Kbps	1	0

NOTE:
(*) Denotes the default setting

8.5.3.4 SEEK Command

The read/write head within the drive is moved from track to track under the control of the SEEK Command. The FDC compares the PCN which is the current head position with the NCN and performs the following operation if there is a difference:

PCN < NCN: Direction signal to drive set to 1 (step in), and issues step pulses.

PCN > NCN: Direction signal to drive set to 0 (step out), and issues step pulses.

The rate at which step pulses are issued is controlled by SRT (Stepping Rate Time) in the SPECIFY Command. After each step pulse is issued, NCN is compared against PCN, and when NCN = PCN, then the SE bit in Status Register 0 is set to 1, and the command is terminated.

During the command phase of the seek or recalibrate operation, the FDC is in the busy state, but during the execution phase it is in the non-busy state.

Note that if implied seek is not enabled, the read and write commands should be preceded by:

1. SEEK Command;
   Step to the proper track
2. SENSE INTERRUPT STATUS Command;
   Terminate the SEEK Command
3. READ ID.
   Verify head is on proper track
4. Issue READ/WRITE Command.

The SEEK Command does not have a result phase. Therefore, it is highly recommended that the SENSE INTERRUPT STATUS Command be issued after the SEEK Command to terminate it and to provide verification of the head position (PCN). The H bit (Head Address) in ST0 will always return a 0. When exiting DSR Powerdown mode, the FDC clears the PCN value and the status information to zero. Prior to issuing the DSR POWERDOWN Command, it is highly recommended that the user service all pending interrupts through the SENSE INTERRUPT STATUS Command.

8.5.3.5 SENSE INTERRUPT STATUS Command

An interrupt signal on the INT pin is generated by the FDC for one of the following reasons:

1. Upon entering the Result Phase of:
   a. READ DATA Command
   b. READ TRACK Command
   c. READ ID Command
   d. READ DELETED DATA Command
   e. WRITE DATA Command
   f. FORMAT TRACK Command
   g. WRITE DELETED DATA Command
   h. VERIFY Command
2. End of SEEK, RELATIVE SEEK or RECALIBRATE Command
3. FDC requires a data transfer during the execution phase in the non-DMA Mode

The SENSE INTERRUPT STATUS Command resets the interrupt signal and via the IC code and SE bit of Status Register 0, identifies the cause of the interrupt. If a SENSE INTERRUPT STATUS Command is issued when no active interrupt condition is present, the status register ST0 will return a value of 80h (invalid command).

The SEEK, RELATIVE SEEK and the RECALIBRATE Commands have no result phase. The SENSE INTERRUPT STATUS Command must be issued immediately after these commands to terminate them and to provide verification of the head position (PCN). The H (Head Address) bit in ST0 will always return a 0. If a SENSE INTERRUPT STATUS is not issued, the drive, will continue to be busy and may effect the operation of the next command.

8.5.3.6 SENSE DRIVE STATUS Command

The SENSE DRIVE STATUS Command obtains drive status information. It has no execution phase and goes directly to the result phase from the command phase. STATUS REGISTER 3 contains the drive status information.

8.5.3.7 SPECIFY Command

The SPECIFY Command sets the initial values for each of the three internal timers. The HUT (Head Unload Time) defines the time from the end of the execution phase of one of the read/write commands to the head unload state. The SRT (Step Rate Time) defines the time interval between adjacent step pulses. Note that the spacing between the first and second step pulses may be shorter than the remaining step pulses. The HLT (Head Load Time) defines the time between the command phase to the execution phase of a READ DATA or Write Data Command. The Head Unload Time (HUT) timer goes from the end of the execution phase to the begining of the result phase of a READ Data or Write Data Command. The values change with the data rate speed selection and are documented in Table 34.

Table 33. Interrupt Identification

SE	IC	Interrupt Due To
0	11	Polling
1	00	Normal Termination of SEEK or RECALIBRATE Command
1	01	Abnormal Termination of SEEK or RECALIBRATE Command

Table 34. Drive Control Delays (ms)

	HUT				SRT			
	1 M	500K	300K	250K	1 M	500K	300K	250K
0	128	256	426	512	8.0	16	26.7	32
1	8	16	26.7	32	7.5	15	25	30
..	..	..	..	..	..	..	..	..
A	80	160	267	320	3.0	6.0	10.2	12
B	88	176	294	352	2.5	5.0	8.3	10
C	96	192	320	384	2.0	4.0	6.68	8
D	104	208	346	416	1.5	3.0	5.01	6
E	112	224	373	448	1.0	2.0	3.33	4
F	120	240	400	480	0.5	1.0	1.67	2

Table 35. Head Load Time (ms)

	HLT			
	1M	500K	300K	250K
00	128	256	426	512
01	1	2	3.3	4
02	2	4	6.7	8
..	..	..	..	..
7E	126	252	420	504
7F	127	254	423	508

The choice of DMA or non-DMA operations is made by the ND bit. When ND = 1, the non-DMA mode is selected, and when ND = 0, the DMA mode is selected. In DMA mode, data transfers are signalled by the DRQ pin. Non-DMA mode uses the RQM bit and the IRQ6 pin to signal data transfers.

8.5.3.8 CONFIGURE Command

Issue the configure command to enable features like the programmable FIFO and set the begining track for precompensation. A CONFIGURE Command need not be issued if the default values of the FDC meets the system requirements.

CONFIGURE DEFAULT VALUES:

EIS	**No Implied Seeks**
EFIFO	FIFO Disabled
POLL	Polling Enabled
FIFOTHR	FIFO Threshold Set to 1 Byte
PRETRK	Pre-Compensation Set to Track 0

EIS—Enable Implied Seek. When EIS = 1, the FDC will perform a SEEK operation before executing a read/write command. The default value is 0 (no implied seek).

EFIFO—Enable FIFO. When EFIFO = 1, the FIFO is disabled (8272A compatible mode). This means data transfers are asked for on a byte by byte basis. The default value is 1 (FIFO disabled). The threshold defaults to one.

POLL—Disable Polling. When POLL = 1, polling of the drives is disabled. POLL Defaults to 0 (polling enabled). When enabled, a single interrupt is generated after a reset. No polling is performed while the drive head is loaded and the head unload delay has not expired.

FIFOTHR—The FIFO threshold in the execution phase of a read/write command. This is programmable from 1 to 16 bytes. FIFOTHR defaults to one byte. A 00 selects one byte and a 0F selects 16 bytes.

PRETRK—Precompensation start track number. Programmable from track 0 to 255. PRETRK defaults to track 0. A 00h selects track 0 and a FFh selects 255.

8.5.3.9 VERSION Command

The VERSION Command checks to see if the controller is an enhanced type (82077, 82077AA, 82077SL) or the older type (8272A/765A). A value of 90h is returned as the result byte, defining an enhanced FDD controller is in use. No interrupts are generated.

8.5.3.10 RELATIVE SEEK Command

The RELATIVE SEEK Command is coded the same as for the SEEK Command, except for the MSB of the first byte and the DIR# bit.

DIR# Head Step Direction Control

DIR#	ACTION
0	Step Head Out
1	Step Head In

RCN Relative Cylinder Number that determines how many tracks to step the head in or out from the current track number.

The RELATIVE SEEK Command differs from the SEEK Command in that it steps the head the absolute number of tracks specified in the command instead of making a comparison against an internal register. The SEEK Command is good for drives that support a maximum of 256 tracks. RELATIVE SEEKs cannot be overlapped with other RELATIVE SEEKs. Only one RELATIVE SEEK can be active at a time. Bit 4 of Status Register 0 (EC) will be set to 1 if RELATIVE SEEK attempts to step outward beyond Track 0.

As an example, assume that a floppy drive has 300 useable tracks and that the host needs to read track 300 and the head is on any track (0–255). If a SEEK Command is issued, the head stops at track 255. If a RELATIVE SEEK Command is issued, the FDC moves the head the specified number of tracks, regardless of the internal cylinder position register (but increments the register). If the head had been on track 40 (D), the maximum track that the FDC could position the head on using RELATIVE SEEK, is 296 (D), the initial track, +256 (D). The maximum count that the head can be moved with a single RELATIVE SEEK Command is 256 (D).

The internal register, PCN, would overflow as the cylinder number crossed track 255 and would contain 40 (D). The resulting PCN value is thus (NCN + PCN) mod 256. Functionally, the FDC starts counting from 0 again as the track number goes above 255(D). It is the users responsibility to compensate FDC functions (precompensation track number) when accessing tracks greater than 255. The FDC does not keep track that it is working in an "extended track area" (greater than 255). Any command issued uses the current PCN value, except for the RECALIBRATE Command that only looks for the TRACK0 signal. RECALIBRATE returns an error if the head is farther than 79 due to its limitation of issuing a maximum 80 step pulses. The user simply needs to issue a second RECALIBRATE Command. The SEEK Command and implied seeks function correctly within the 44 (D) track (299–255) area of the extended track area. It is the users responsibility not to issue a new track position that exceeds the maximum track that is present in the extended area.

To return to the standard floppy range (0–255) of tracks, a RELATIVE SEEK is issued to cross the track 255 boundary.

A RELATIVE SEEK Command can be used instead of the normal SEEK Command but the host is required to calculate the difference between the current head location and the new (target) head location. This may require the host to issue a READ ID Command to ensure that the head is physically on the track that software assumes it to be. Different FDC commands return different cylinder results which may be difficult to keep track of with software without the READ ID Command.

8.5.3.11 DUMPREG Command

The DUMPREG Command is designed to support system run-time diagnostics and application software development and debug. The command returns pertinent information regarding the status of many of the programmed fields in the FDC. This can be used to verify the values initialized in the FDC.

8.5.3.12 PERPENDICULAR MODE Command

An added capability of the FDC is the ability to interface directly to perpendicular recording floppy drives. Perpendicular recording differs from the traditional longitudinal method by orienting the magnetic bits vertically. This scheme packs in more data bits for the same area.

The PERPENDICULAR MODE Command allows the system designers to designate specific drives as Perpendicular recording drives. Data transfers between Conventional and Perpendicular drives are allowed without having to issue PERPENDICULAR MODE Commands between the accesses of the two different drives, nor having to change write precompensation values.

With this command, the length of the Gap2 field and VCO enable timing can be altered to accommodate the unique requirements of these drives. Table 36 describes the effects of the WGATE and GAP bits for the PERPENDICULAR MODE Command.

When both GAP and WGATE equal 0 the PERPENDICULAR MODE Command will have the following effect on the FDC:

1. If any of the new bits D0 and D1 are programmed to 1, the corresponding drive is automatically programmed for Perpendicular mode (ie: GAP2 being written during a write operation, the programmed Data Rate will determine the length of GAP2), and data will be written with 0 ns write precompensation.
2. Any of the new bits (DO/D1) are programmed for 0, the designated drive is programmed for Conventional Mode and data will be written with the currently programmed write precompensation value.
3. Bits D0 and D1 can only be over-written when the OW bit is 1. The status of these bits can be determined by interpreting the eighth result byte of the DUMPREG Command. (Note: if either the GAP or WGATE bit is 1, bits D0 and D1 are ignored.)

Software and Hardware reset have the following effects on the enhanced PERPENDICULAR MODE Command:

1. A software reset (Reset via DOR or DSR registers) only sets GAP and WGATE bits to 0; D0 and D1 retain their previously programmed values.
2. A hardware reset (Reset via pin 32) sets all bits (GAP, Wgate, D0, and D1) to 0 (All Drives Conventional Mode).

8.5.3.13 POWERDOWN MODE Command

The POWERDOWN MODE Command allows the automatic power management and enables the enhanced registers (EREG EN) of the FDC. The use of the command can extend the battery life in portable PC applications. To enable auto powerdown the command may be issued during the BIOS power on self test (POST).

This command includes the ability to configure the FDC into the enhanced mode extending the SRB and TDR registers. These extended registers accommodate bits that give more information about floppy drive interface, allow for boot drive selection, and identify the values of the PD and IDLE status.

As soon as the command is enabled, a 10 ms or a 0.5 sec minimum powerup timer is initiated, depending on whether the MIN DLY bit is set to 0 or 1. This timer is one of the required conditions that has to be satisfied before the FDC will enter auto powerdown.

Table 36. Effects of WGATE and GAP Bits

GAP	WGATE	MODE	VCO Low Time after Index Pulse	Length of Gap2 Format Field	Portion of Gap2 Written by Write Data Operation	Gap2 VCO Low Time for Read Operations
0	0	Conventional Mode	33 Bytes	22 Bytes	0 Bytes	24 Bytes
0	1	Perpendicular Mode (500 Kbps and Lower Data Rates)	33 Bytes	22 Bytes	19 Bytes	24 Bytes
1	0	Reserved (Conventional)	33 Bytes	22 Bytes	0 Bytes	24 Bytes
1	1	Perpendicular Mode (1 Mbps Data Rate)	18 Bytes	41 Bytes	38 Bytes	43 Bytes

NOTE:
When either GAP or WGATE bit is set, the current value of precompensation in the DSR is used.

Any software reset will re-initialize the timer. The timer countdown is also extended by up to 10 ms if the data rate is changed during the timer's countdown. Without this timer, the FDC would have been put to sleep immediately after FDC is idle. The minimum delay gives software a chance to interact with the FDC without incurring an additional overhead due to recovery time.

The command also allows the output pins of the floppy disk drive interface to be tri-stated or left unaltered during auto powerdown. This is done by the FDI TRI bit. In the default condition (FDI TRI = 0) the output pins of the floppy disk drive are tri-stated. Setting this bit leaves the interface unchanged from the normal state.

The results phase returns the values programmed for MIN DLY, FDI TRI and AUTO PD. The auto powerdown mode is disabled by a hardware reset. Software results have no effect on the POWERDOWN MODE Command parameters.

8.5.3.14 PART ID Command

This command can be used to identify the floppy disk controller as an enhanced controller. The first stepping of the FDC (all versions) will yield 0x02 in the result phase of this command. Any future enhancements on these parts will be denoted by the 5 LSBs (0x01 to 0x1F).

8.5.3.15 OPTION Command

The standard IBM format includes an index address field consisting of 80 bytes of GAP 4a, 12 bytes of the sync field, four bytes identifying the IAM and 50 bytes of GAP 1. Under the ISO format most of this preamble is not used. The ISO format allows only 32 bytes of GAP 1 after the index mark. The ISO bit in this command allows the FDC to configure the data transfer commands to recognize this format. The MSBs in this command are reserved for any other enhancements made available to the user in the future.

8.5.3.16 SAVE Command

The first byte corresponds to the values programmed in the DSR with the exception of CLKSEL. The DRATE1, DRATE0 used here are unmapped. The second byte is used for configuring the bits from the OPTION Command. All future enhancements to the OPTION Command will be reflected in this byte as well. The next nine result bytes are explained in the Parameter Abbreviations section after the command summary. The 13th byte is the value associated with the POWERDOWN MODE Command. The disk status is used internally by the FDC. There are two reserved bytes at the end of this command for future use.

This command is similar to the DUMPREG Command but it additionally allows the user to read back the precompensation values as well as the programmed data rate. It also allows the user to read the values programmed in the POWERDOWN MODE Command. The precompensation values will be returned as programmed in the DSR register. This command, used in conjunction with the RESTORE Command, should prove very useful for SMM power management. This command reserves the last two bytes for future enhancements.

8.5.3.17 RESTORE Command

Using the RESTORE Command with the SAVE Command, allows the SMM power management to restore the FDC to its original state after a system powerdown. It also serves as a succinct way to provide most of the initialization requirements normally handled by the system. The sequence of initializing the FDC after a reset occurred and assuming a SAVE Command was issued follows:

- Issue the DRIVE SPECIFICATION Command (if the design utilizes this command)
- Issue the RESTORE Command (pass the 16 bytes retrieved previously during SAVE)

The RESTORE Command programs the data rate and precompensation value via the DSR. It then restores the values normally programmed through the CONFIGURE, SPECIFY, and PERPENDICULAR Commands. It also enables the previously selected values for the POWERDOWN Mode Command. The PCN values are set restored to their previous values and the user is responsible for issuing the SEEK and RECALIBRATE Commands to restore the head to the proper location. There are some drives that do not recalibrate in which case the RESTORE Command restores the previous state completely. The PDOSC bit is retrievable using the SAVE Command, however, the system designer must set it correctly. The software must allow at least 20 μs to execute the RESTORE Command. When using the BOOTSEL bits in the TDR, the user must restore or reinitialize these bits to their proper values.

8.5.3.18 FORMAT AND WRITE Command

The FORMAT AND WRITE Command is capable of simultaneously formatting and writing data to the diskette. It is essentially the same as the normal FORMAT Command. With the exception that included in the execution for each sector is not only the C, H, R, and N but also the data transfer of N bytes. The D value is ignored. This command formats the entire track. High speed floppy diskette duplication can be done fast and efficiently with this command. The user can format the diskette and put data on it in a single pass. This is very useful for software duplication applications by reducing the time required to format and copy diskettes.

9.0 IDE INTERFACE

The 82091AA supports the IDE (Integrated Drive Electronics) interface by providing two chip selects, and lower and upper data byte controls. DMA and 16-bit data transfers are supported. Minimal external logic is required to complete the optional 16-bit IDE I/O and DMA interfaces. With external logic, a fully buffered interface is also supported.

9.1 IDE Registers

The 82091AA does not contain IDE registers. All of the IDE device registers are located in the IDE device, except bit 7 of the Drive Address Register which is the Floppy Controller Disk Change status bit and is driven by the 82091AA.

The IDE interface contains two chip selects (IDECS0# and IDECS1#). These signals are asserted for accesses to the Command and Control Block registers located at 01Fxh and 03Fxh, respectively (Table 37).

Table 37. IDE Register Set (Located in IDE Device)

Primary Address	Secondary Address	Chip Select	Registers	Access
1F0h	170h	IDECS0#	Data Register	R/W
1F1h	171h	IDECS0#	Error Register	RO
1F1h	171h	IDECS0#	Write Precomp/Features Register	WO
1F2h	172h	IDECS0#	Sector Count Register	R/W
1F3h	173h	IDECS0#	Sector Number Register	R/W
1F4h	174h	IDECS0#	Cylinder Low Register	R/W
1F5h	175h	IDECS0#	Cylinder High Register	R/W
1F6h	176h	IDECS0#	Drive/Head Register	R/W
1F7h	177h	IDECS0#	Status Register	RO
1F7h	177h	IDECS0#	Command Register	WO
3F6h	376h	IDECS1#	Alternate Status Register	RO
3F6h	376h	IDECS1#	Digital Output Register	WO
3F7h	377h	IDECS1#	Drive Address Register	RO
3F7h	377h	IDECS1#	Not Used	

9.2 IDE Interface Operation

The 82091AA implements the chip select signals for the IDE interface and decodes the standard PC/AT primary and secondary I/O locations.

The 82091AA provides a data buffer enable signal (DEN#) to control the lower data byte path for buffered designs. Buffering the lower data byte path is an application option that requires an external transceiver/buffer. For buffered applications, DEN# controls an external transceiver and enables data bits IDED[7:0] onto the system data bus SD[7:0]. For non-buffered applications (typically the X-Bus configuration), IDED[7:0] are connected directly to the bus and DEN# is not used and becomes a no-connect. For 16-bit applications the upper data byte path (IDED[15:8]) is controlled by the HEN# signal.

Figure 65 shows an example IDE interface without DMA capability. In this case all IDE accesses for setting up the IDE registers and transferring data is programmed via I/O. The 82091AA generates the chip selects (IDECS0# and IDECS1#). The 82091AA also generates the DEN# and HEN# signals to enable the data buffers.

Figure 66 shows an example DMA IDE interface for type "F" DMA cycles. To set up the IDE interface, the host accesses the IDE registers on the IDE device. For programmed I/O accesses, the 82091AA generates the chip selects (IDECS0# and IDECS1#) to access the IDE registers and the DEN# and HEN# signals to control the data buffers. During DMA transfers the DMA handshake is between the DMA controller and IDE device via the DREQ and DACK# signals. The DACK# signal is ORed with the DEN# and HEN# signals to control the upper and lower byte buffers during DMA transfers.

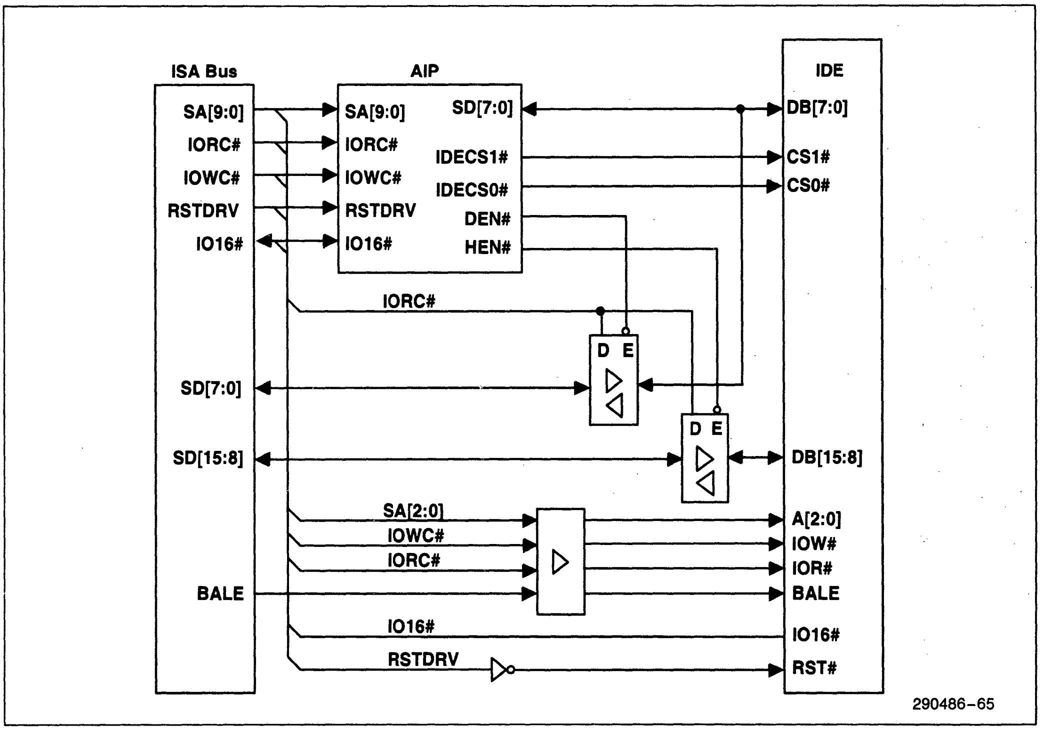


Figure 65. IDE Interface Example (without DMA)

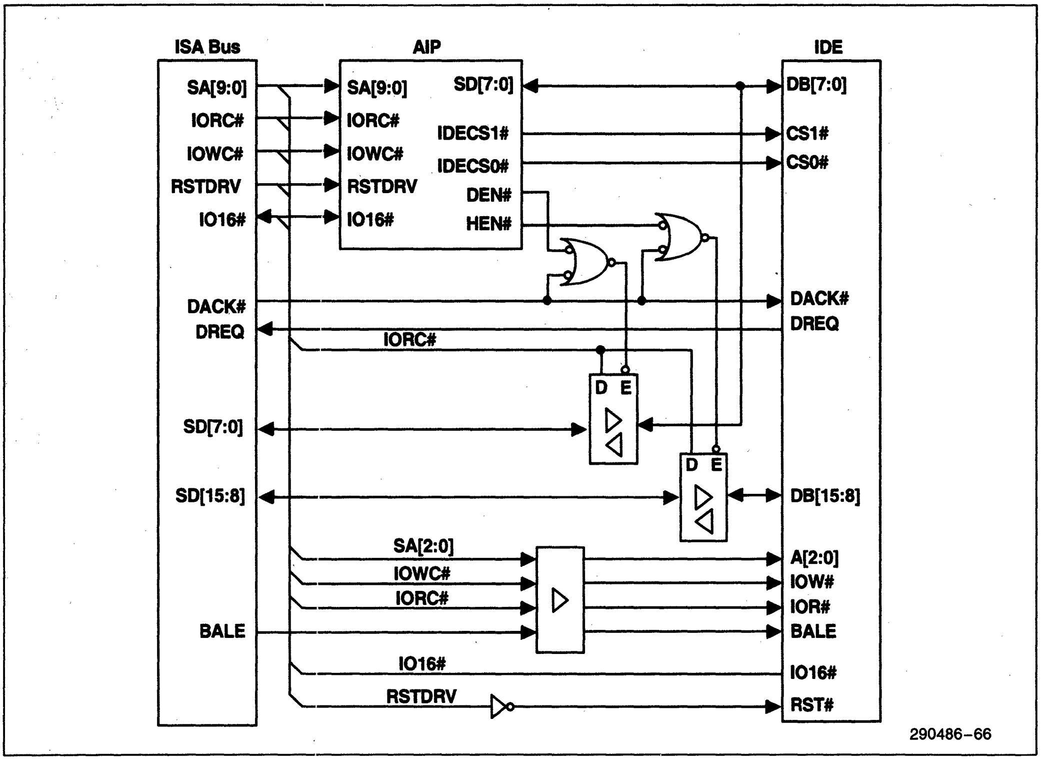


Figure 66. IDE Interface Example (with DMA)

10.0 POWER MANAGEMENT

The 82091AA provides power management capabilities for its primary functional modules (parallel port, floppy disk controller, serial port A, and serial port B). For each module, the 82091AA implements two types of power management—direct powerdown and auto powerdown. Direct powerdown, enabled via control bits in the 82091AA configuration registers, immediately places the module in a powerdown mode by turning off the clock to the associated module. Direct powerdown removes the clock regardless of the activity or status of the module. By contrast, when auto powerdown is enabled (via control bits in the 82091AA configuration registers), the associated module only enters a powerdown mode if it is in an idle state.

NOTE:
The entire 82091AA can be placed in direct powerdown by writing to the CLKOFF bit in the AIPCFG1 Register.

10.1 Power Management Registers

The floppy disk controller, parallel port, serial port A, and serial port B each have two 82091AA configuration registers. For each module, three configuration register bits control power management—xDPDN, xIDLE, and xAPDN.

- xAPDN: auto-powerdown, shuts off the oscillator to the module when the module is idle.
- xIDLE: idle status, a read only pin that indicates idle status.
- xDPDN: direct powerdown, shuts off module oscillator when active regardless of module status.

The 82091AA exits any powerdown mode after a hardware reset (RSTDRV asserted) or reset via the xRESET bit in the 82091AA configuration registers. Direct powerdown can also be exited by writing the corresponding xPDN bit in the configuration register to 0. Auto powerdown is exited by events at the module (e.g., CPU read/write or module interface activity).

NOTE:
The configuration registers also contain the xEN bit. This bit is used to completely disable an unused module. Enabling a disabled module takes much longer than restoring a module from powerdown. Therefore, this bit is not recommend for temporarily disabling a module as a powerdown scheme.

10.2 Clock Power Management

The internal clock circuitry of the 82091AA can be turned on or off as part of a power management scheme. The clock circuitry is controlled via the CLKOFF bit in the AIPCFG1 Register. If an external clock source exists, the user may want to turn off the internal oscillator to save power and provide minimum recovery time.

Auto powerdown and direct powerdown (in each module) have no effect on the state of internal oscillator.

10.3 FDC Power Management

This section describes the FDC direct and auto powerdown modes and recovery from the powerdown modes.

Auto Powerdown

Automatic powerdown (APDN) has an advantage over direct powerdown (PDN) since the register contents are not lost under APDN. Automatic powerdown is invoked by either the Auto Powerdown command, or by enabling the FAPDN bit in the FDC configuration register. There are four conditions required before the FDC will enter powerdown:

1. The motor enable pins ME[3:0] must be inactive.
2. The FDC must be in an idle state. FDC idle is indicated by MSR = 80h and the IRQ6 signal is negated (IRQ6 may be asserted even if MSR = 80h due to polling interrupt).
3. The head unload timer (HUT, explained in the SPECIFY Command) must have expired.
4. The auto powerdown timer must have timed out.

An internal timer is initiated when the POWERDOWN MODE Command is executed. The amount of time can be set by the user via the MIN DLY bits in the POWERDOWN MODE Command. The module is then powered down, provided all the remaining conditions are met. A software reset reinitializes the timer. When using the FDC FAPDN bit to enable the automatic powerdown feature, the MIN DLY bit is set to the default condition.

Recovery from Auto Powerdown

When the FDC is in auto powerdown, the module is awakened by a reset or access to the DOR, MSR or FIFO registers. The module remains in auto powerdown mode after a software reset (i.e., it will powerdown again after being idle for the time specified by MIN DLY). However, the FDC does not remain in auto powerdown mode after a hardware reset or DSR reset.

Direct Powerdown

Direct powerdown is invoked via the Powerdown bit in the Data Rate Select Register (bit 6), or the FDPDN bit in the FCFG2 Register. Setting FDPDN to 1 will powerdown the FDC. All status is lost when this type of powerdown mode is used. The FDC exits powerdown mode after any hardware or software reset. Direct powerdown overrides automatic powerdown.

Recovery from Direct Powerdown

The FDC exits the direct powerdown state by setting the FDPDN bit to 0 followed by a software or hardware reset.

After reset, the FDC goes through a normal sequence. The drive status is initialized. The FIFO mode is set to default mode on a hardware or software reset if the LOCK Command has not blocked it. Finally, after a delay, the polling interrupt is issued.

10.4 Serial Port Power Management

This section describes the serial port direct and auto powerdown modes and recovery from the powerdown modes.

Auto Powerdown

When auto powerdown is enabled in the SxCFG2 Register (SxAPDN bit is 1), the serial port enters auto powerdown based on monitoring line interface activity. During auto powerdown, the status of the serial port is maintained (the FIFO and registers are not reset). Access to any serial port register is allowed during auto powerdown. The transmitter and the receiver enter powerdown individually, depending on certain conditions. When there are no characters to transmit (TEMPTY = 1 in the LSR), the transmitter clock is shut off placing the transmitter in auto powerdown. In the case of the receiver, when serial input signal is inactive for approximately 5 character times, indicating that no character is being received, the receiver goes into auto powerdown.

Recovery from Auto Powerdown

The serial port recovers from auto powerdown when either the transmitter or receiver are active. If data is written to the transmitter or data is present at the receiver, the serial port exits from auto powerdown.

Direct Powerdown

Direct Powerdown is invoked via the SxCFG2 Register (setting the SxDPDN bit to 1). When in direct powerdown, the clock to the module is shut off. All registers are accessible while in direct powerdown. A host read of the Receiver Buffer Register or a write to the Transmitter Holding Register should not be performed during powerdown. The SINx input should remain static.

When direct powerdown is invoked, the transmit and receive sections of the serial port are reset, including the transmit and receive FIFOs. Thus, to prevent possible data loss when the FIFOs are reset, software should not invoke direct powerdown until the serial port is in the idle state as indicated by the SxIDLE bit in the SxCFG2 Register.

Recovery from Direct Powerdown

Recovery from direct powerdown is accomplished by writing the SxDPDN bit in the configuration register to 0 or by a module reset.

10.5 Parallel Port Power Management

Auto Powerdown

Auto powerdown is enabled via the PAPDN bit in the PCFG2 Register. When enabled, the parallel port enters auto powerdown when the module is in an idle state. If the parallel port FIFO is being used to transfer data, the parallel port is in an idle state when the FIFO is empty.

Recovery from Auto Powerdown

Recovery from auto powerdown occurs when the FIFO is written or as a result of parallel port interface activity.

Direct Powerdown

Direct powerdown is invoked via the PCFG2 Register (setting the PDPDN bit to 1). When PDPDN = 1, the clock to the printer state machine is disabled and the state machine goes into an idle state.

Recovery from Direct Powerdown

Recovery from direct powerdown is accomplished by setting the PDPDN bit to 0 or the PRESET bit to a 1 in the PCFG2 Register. An 8209†AA hard reset (RSTDRV asserted) also brings the part out of direct powerdown.

11.0 ELECTRICAL CHARACTERISTICS

11.1 Absolute Maximum Ratings

Storage Temperature .... −65°C to +150°C
Supply Voltage .... −0.5V to +8.0V
Voltage on Any Input............ GND−2V to 6.5V
Voltage on Any Output ... GND−0.5V to V_{CC} +0.5V
Power Dissipation 1W

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.

******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.*

11.2 DC Characteristics

Table 38. DC Specifications (V_{CC} = 5V ±10%, T_{amb} = 0°C to 70°C)

Symbol	Parameter	V_{CC} = +5V ±10			V_{CC} = 3.3V ±0.3V		
		Min(V)	Max(V)	Notes	Min(V)	Max(V)	Notes
V_{ILC}	Input Low Voltage, X1	−0.5	0.8		−0.3	0.8	
V_{IHC}	Input High Voltage, X1	3.9	V_{CC} + 0.5		2.4	V_{CC} + 0.3	
V_{IL}	Input Low Voltage (all pins except X1)	−0.5	0.8		−0.3	0.8	
V_{IH}	Input High Voltage (all pins except X1)	2.0	V_{CC} + 0.5		2.0	V_{CC} + 0.3	
I_{CC}	V_{CC} Supply Current −1 Mbps FDC Data Rate V_{IL} = 0.45V, V_{IH} = 2.4V		50 mA	1, 2		40 mA	1, 2
I_{CCSB}	I_{CC} in Powerdown		100 μA	3, 4, 5		100 μA	3, 4, 5
I_{IL}	Input Load Current (all input pins)		+10μA −10 μA	6		+10 μA −10 μA	6
I_{OFL}	Data Bus Output Float Leakage		+10 μA −10 μA	7		+10 μA −10 μA	8
I_{BPL}	Parallel Port Back-Power Leakage (All Parallel Port Signals)		+10 μA	9		+10 μA	9

NOTES:
1. Test Conditions: Only the data bus inputs may float. All outputs are open.
2. Test Conditions: Tested while reading a sync field of "00". Outputs not connected to DC loads. This specification reflects the supply current when all modules within the 82091AA are active.
3. Test Conditions: V_{IL} = V_{SS}, V_{IH} = V_{CC}; Outputs not connected to DC loads.
4. Test Conditions: Typical value with the oscillator off.
5. Test Conditions: All 82091AA modules are in their powerdown state.
6. Test Conditions: 10 μA (V_{IN} = V_{CC}), −10μA (V_{IN} = 0V)
7. Test Conditions: 0V < V_{OH} < V_{CC}
8. Test Conditions: 0.45V < V_{OH} < V_{CC}
9. Test Conditions: Device in Circuit V_{CC} = 0V, V_{IN} = 5.5V max.

Table 39. Capacitance Specifications (V_{CC} = 5V ±10%, T_{amb} = 0°C to 70°C)

C_{IN}	Input Capacitance	10	pF	F = 1 MHz, T_A = 25°C
C_{IN1}	Clock Input Capacitance	20	pF	Sampled, not 100% Tested
$C_{I/O}$	Input/Output Capacitance	20	pF	

NOTE:
All pins except pins under test are tied to AC ground.

The following pin groupings are used in Table 40 and Table 41.

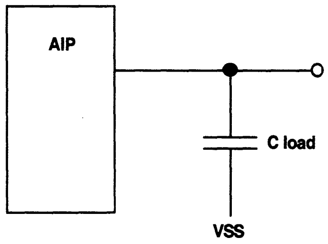
DMA	FDDREQ, PPDREQ
IRQx	IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
Serial Port	SOUTA, SOUTB, DTRA#, DTRB#, RTSA#, RTSB#
Parallel Port	PD[7:0], STROBE#, AUTOFD#, INIT#, SELECTIN#
FDC Interface	WRDATA, HDSEL#, STEP#, DIR#, WE#, FDME0#, FDME1#, FDS0#, FDS1#, DRVDEN[1:0]

Table 40. V_{OL} Specifications (V_{CC} = 5V ±10%, T_{amb} = 0°C to 70°C)

Symbol	Signal	V_{CC} = 5V ± 10%			V_{CC} 3.3V ± 0.3V		
		Min	Max	I_{OL}	Min	Max	I_{OL}
V_{OL}	SD[7:0]		0.45V	24 mA		0.45V	12 mA
V_{OL}	NOWS#, IOCHRDY		0.45V	24 mA		0.45V	12 mA
V_{OL}	DMA,IRQx		0.45V	12 mA		0.45V	6 mA
V_{OL}	Serial Port		0.45V	4 mA		0.45V	2 mA
V_{OL}	Parallel Port		0.45V	16 mA		0.45V	8 mA
V_{OL}	PPDIR,GCS#		0.45V	4 mA		0.45V	2 mA
V_{OL}	FDC Interface		0.45V	12 mA		0.45V	6 mA
V_{OL}	DEN#,HEN#		0.45V	4 mA		0.45V	2 mA
V_{OL}	IDECS[1:0]#		0.45V	12 mA		0.45V	6 mA

Table 41. V_{OH} Specifications (V_{CC} = 5V ±10%, T_{amb} = 0°C to 70°C)

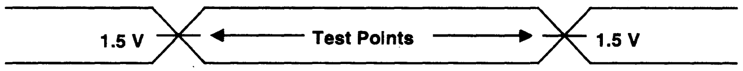
Symbol	Signal	V_{CC} = 5V ± 10%			V_{CC} 3.3V ± 0.3V		
		Min	Max	I_{OH}	Min	Max	I_{OH}
V_{OH}	SD[7:0]	2.4V		4 mA	2.4V		2 mA
V_{OH}	DMA,IRQx	2.4V		4 mA	2.4V		2 mA
V_{OH}	Serial Port	2.4V		1 mA	2.4V		1 mA
V_{OH}	Parallel Port	2.4V		4 mA	2.4V		50 μA
V_{OH}	PPDIR,GCS#	2.4V		1 mA	2.4V		1 mA
V_{OH}	FDC Interface	2.4V		4 mA	2.4V		2 mA
V_{OH}	DEN#, HEN#	2.4V		1 mA	2.4V		1 mA
V_{OH}	IDECS[1:0]#	2.4V		4 mA	2.4V		2 mA



C load = 50 pF for all logic outputs and 100 pF for the data bus.

290486-67

Figure 67. Load Circuit



290486-68

Figure 68. AC Testing Input, Output

11.3 Oscillator

The 24 MHz clock can be supplied either by a crystal (Figure 69) or a MOS level square wave. All internal timings are referenced to this clock or a scaled count that is data rate dependent. The crystal oscillator must be allowed to run for 10 ms after V_{CC} has reached 4.5V or exiting the POWERDOWN mode to guarantee that it is stable.

Crystal Specifications:

Freq:	24 MHz ± 0.1%
Mode:	Parallel Resonant Fundamental Mode
Series Resistance:	< 40Ω
Shunt Capacitance:	< 5 pF
C1, C2:	20 pF–25 pF

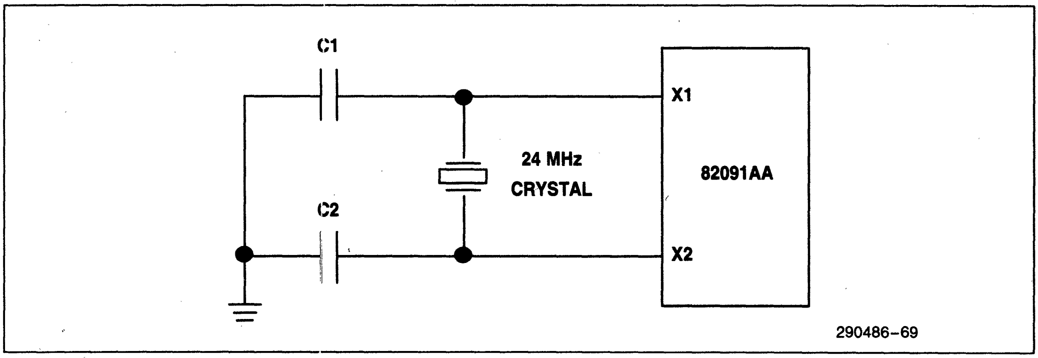


Figure 69. Crystal Connections

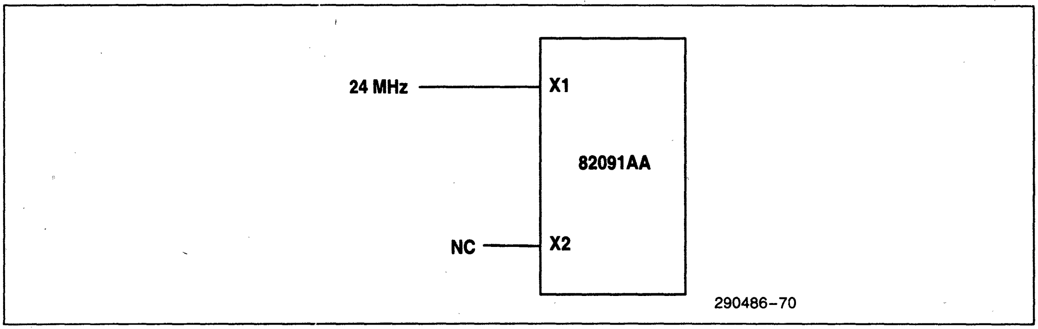


Figure 70. Oscillator Connections

11.4 AC Characteristics

Table 42. AC Specifications (V_{CC} = 5V ± 10%, T_{amb} = 0°C to 70°C)

Symbol	Parameter	24 MHz		Units	Notes	Figure
		Min	Max			
t1a	Clock Rise and Fall Time		10	ns	1	71
t1b	Clock High Time	16		ns	1	71
t1c	Clock Low Time	16		ns	1	71
t1d	Clock Period	41.66	41.66	ns	2	71
t1e	Internal Clock Period				3	

NOTES:

1. Clock input high level test points for clock high time and clock rise/fall times are 3.5V with V_{CC} at 5V ±10% and 2.0V with V_{CC} at 3.3V V_{CC} ±10%. Clock input low level test point for clock low time and clock rise/fall time is 0.8V.
2. Clock input test point for clock period is 0.8V.
3. Certain Floppy Disk Controller module timings are a function of the selected data rate. The nominal values for the internal clock period (t1e) for the various data rates are:

Disk Drive Disk Rate	Internal Clock Period (*nominal values)
	24 MHz
1 Mbps	125 ns
500 Kbps	250 ns
300 Kbps	420 ns
250 Kbps	500 ns

All information contained in ( ) in the following tables represents 3.3V specifications.

Table 43. AC Specifications (V_{CC} = 5V ± 10%, or [3.3V ± 0.3V] T_{amb} = 0°C to 70°C)

Symbol	Parameter	Min	Max	Units	Notes	Figure
Host						
SA[10:0]						
t2a	SA[10:0] Setup to IORC#/IOWC# Active	18 (25)		ns		72, 73
t2b	SA[10:0] Hold from IORC#/IOWC# Inactive	0		ns		72, 73
SD[7:0]						
t3a	SD[7:0] Valid Delay from IORC# Active		70 (100)	ns	1	72
t3b	SD[7:0] Float Delay from IORC# Inactive	5	35 (40)	ns		72
t3c	SD[7:0] Setup to IOWC# Inactive	35		ns		73
t3d	SD[7:0] Hold from IOWC# Inactive	0		ns		73

Table 43. AC Specifications (V_{CC}= 5V ± 10%, or (3.3V ± 0.3V) T_{amb}=0°C to 70°C) (Continued)

Symbol	Parameter	Min	Max	Units	Notes	Figure
IOCHRDY						
t4a	IOCHRDY Propagation Delay from IORC#/ IOWC#		55 (75)	ns	EPP	82, 83
t4b	IOCHRDY Propagation Delay from BUSY		34 (65)	ns	EPP	82, 83
IORC#						
t5a	IORC# Active Pulse Width	90		ns		72
t5b	IORC# Recovery Time	60		ns		72
IOWC#						
t6a	IOWC# Active Pulse Width	90		ns		73
t6b	IOWC# Recovery Time	60		ns		73
AEN						
t7a	AEN Setup to IORC#/IOWC# Active	18		ns		72, 73
t7b	AEN Hold from IORC#/IOWC# Inactive	0		ns		72, 73
NOWS#						
t8a	NOWS# Delay from IORC#/IOWC#		35 (50)	ns		72, 73
TC						
t9a	TC Active Pulse Width	50		ns	6	74
RESET						
RSTDRV						
t10a	RSTDRV Active Pulse Width	0.5		μs		75
t10b	Hardware Configuration Input Setup to RSTDRV Inactive	100		ns	All Configuration Modes	76
t10c	Hardware Configuration Input Hold from RSTDRV Inactive	0			All Configuration Modes	76
INTERRUPTS						
RQ[4,3] (Serial Ports)						
t11b	IRQ[4,3] Inactive Delay from IORC#/ IOWC# Active		100	ns	THR wr, RBR rd, MSR rd	90, 91
t11c	IRQ[4,3] Inactive Delay from IORC# Inactive		100	ns	IIR rd, LSR rd	90
t11d	IRQ[4,3] Active Delay from DCD#/DSR#/ CTS#/RI#		80	ns		91

Table 43. AC Specifications (V_{CC} = 5V ± 10%, or (3.3V ± 0.3V) T_{amb} = 0°C to 70°C) (Continued)

Symbol	Parameter	Min	Max	Units	Notes	Figure
INTERRUPTS						
IRQ[7,5] (Parallel Port)						
t12b	IRQ[7,5] Inactive Delay from IORC#/ IOWC# Active		70 (90)	ns	ECP rev, fwd to FIFO	81
t12c	IRQ[7,5] Inactive Delay from IOWC# Inactive		70 (95)	ns	ECP fwd to ECR	81
t12d	IRQ[7,5] Delay from ACK#		70 (90)	ns	All Modes	81
t12e	IRQ[7,5] Delay from FAULT#		70 (90)	ns	ECP	81
IRQ6 (FDC)						
t13b	IRQ6 Inactive Delay from IORC#/ IOWC# Active		t1e+125	ns	2	80
DMA						
FDDREQ, PPDREQ						
t14a	xDREQ Inactive Delay from xDACK# Active		75 (100)	ns	4	74
t14b	FDREQ Cycle Time (Non-Burst DMA)	6.25		µs	3	74
t14c	xDREQ Active from IORC#/IOWC# Inactive	100		ns		74
t14d	xDREQ Setup IORC#/IOWC#	0		ns	3	74
t14e	xDREQ Delay from IORC#/IOWC# Active		75 (100)	ns	5	74
t14f	FDREQ Inactive Delay from TC Active PPDREQ Inactive Delay from TC Active		110 80 (90)	ns		74
t14g	xDREQ to xDACK# Inactive	⅔ t1e				74
FDDACK#, PPDACK#						
t15a	xDACK# Active Delay from xDREQ Active	0		ns		74
t15b	xDACK# Setup to IORC#/IOWC# Active	18		ns		74
t15c	xDACK# Hold from IORC#/IOWC# Inactive	0		ns		74

Table 43. AC Specifications (V_{CC} = 5V ± 10%, or (3.3V ± 0.3V) T_{amb} = 0°C to 70°C) (Continued)

Symbol	Parameter	Min	Max	Units	Notes	Figure
PARALLEL PORT						
PD[7:0]						
t16a	PD[7:0] Delay from IOWC# Inactive		60 (90)	ns	ISA,PS/2 wr	87
t16b	PD[7:0] Delay from IOWC# Active		70 (100)	ns	EPP wr	82
t16c	PD[7:0] Float Delay from IOWC# Inactive	50		ns	EPP wr	82
t16f	PD[7:0] Setup to STROBE# Active	450			ISA FIFO	84
t16g	PD[7:0] Hold from STROBE# Inactive	450			ISA FIFO	84
t16h	PD[7:0] Hold from BUSY Inactive	0			ECP fwd	85
t16i	PD[7:0] Setup to ACK# High	0			ECP rev	86
t16j	PD[7:0] Hold from AUTOFD# Low	0			ECP rev	86
STROBE#						
t17a	STROBE# Delay from IOWC# Inactive		60/ 90		ISA, PS/2	87
t17b	STROBE# Delay from IORC#/IOWC#		60/ 90		EPP	82, 83
t17c	STROBE# Active from BUSY Inactive	500			ISA FIFO	84
t17d	STROBE Active Pulse Width	450			ISA FIFO	84
t17e	STROBE# Active from BUSY Inactive	0			ECP fwd	85
t17f	STROBE# Inactive Delay from BUSY Active	0			ECP fwd	85
AUTOFD#						
t18a	AUTOFD# Delay from IOWC# Inactive		60 (90)	ns	ISA,PS/2	87
t18b	AUTOFD# Delay from IORC#/IOWC#		60 (90)	ns	EPP	82, 83
t18c	AUTOFD# Hold from BUSY Inactive	80		ns	ECP fwd	85
t18d	AUTOFD# Low Delay from ACK# Inactive	0		ns	ECP rev	86
t18e	AUTOFD# High Delay from ACK# Active	0		ns	ECP rev	86
INIT#						
t19a	INIT# Delay from IOWC# Inactive		60 (90)	ns	All Modes	87

Table 43. AC Specifications (V_{CC} = 5V ± 10%, or (3.3V ± 0.3V) T_{amb} = 0°C to 70°C) (Continued)

Symbol	Parameter	Min	Max	Units	Notes	Figure
SELECTIN#						
t20a	SELECTIN# Delay from IOWC# Inactive#		60 (90)	ns	ISA, PS/2	87
t20b	SELECTIN# Delay from IOWC#/IORC# Active		60 (90)		EPP	82
BUSY						
t21a	BUSY Active Delay from STROBE# Active		500		ISA, PS/2	84
t21b	BUSY Active Delay from STROBE# Active	0				85
t21c	BUSY Inactive Delay from STROBE# Inactive	0			ECP fwd	85
t21d	BUSY Setup to ACK# Active	0			ECP rev	86
t21f	BUSY Hold from AUTOFD# Inactive	0			ECP rev	86
ACK#						
t22a	ACK# Active Hold from AUTOFD# High	0			ECP rev	86
t22b	ACK# Inactive Hold from AUTOFD# Low	0			ECP rev	86
PPDIR/GCS#						
t23a	GCS# Delay from SA[10:0]		60 (90)			89
t23b	PPDIR Delay from IOWC# Inactive		60 (90)		ISA, PS/2, ECP	87
t23c	PPDIR Delay from IOWC# Active		60 (90)		EPP	82
IDE Interface						
IDECS[1:0]#						
t24a	IDECSx# Delay from SA[10:0]		40 (70)			88
DEN#						
t25a	DEN# Delay from SA[10:0]		40 (70)			72, 73, 88
t25b	DEN# Delay from xDACK#		40 (70)			74
HEN#						
t26a	HEN# Delay from IO16#		35 (65)			88

Table 43. AC Specifications (V_{CC} = 5V ± 10%, or (3.3V ± 0.3V) T_{amb} = 0°C to 70°C) (Continued)

Symbol	Parameter	Min	Max	Units	Notes	Figure
SERIAL PORTS						
DTRx#, RTSx#, DCDx#						
t27a	DTRx#/RTSx#/DCDx# Active Delay from IOWC#		55 (70)	ns	MCR wr	91
FLOPPY DISK CONTROLLER						
RDDATA#						
t28a	Read Data Pulse Width	50		ns		95
t28c	PLL Data Rate		1M	bits/sec		na
t28d	Lockup Time		64	t28c		na
WRDATA#						
t29a	Data Width	see note	see note		7	77
HDSEL#						
t30a	WE# to HDSEL# Change	see note	see note		10	78
STEP#						
t31a	STEP# Active Time	2.5		μs		78
t31b	STEP# Cycle Time	see note	see note	μs	9	78
DIR#						
t32a	DIR# Setup to STEP# Active	1		μs	8	78
t32b	DIR# Hold from STEP# Inactive	10		μs		78
WE#						
t33a	WE# Inactive Delay from RSTDRV Inactive Edge		2	μs		75
INDEX#						
t34a	INDEX# Pulse Width	5	t1e			78

NOTES:

1. The FDC Status Register's status bits which are not latched may be updated during a host read operation.
2. The timing t13b is specified for the FDC interrupt signal in the polling mode only. These timings in case of the result phase of the read and write commands are microcode dependent.
3. This timing is for FDC FIFO threshold = 1. When FIFO threshold is N bytes, the value should be multiplied by N and subtract 1.5 μs. The value shown is for 1 Mbps, scales linearly with data rate.
4. This timing is a function of the internal clock period (t1e) and is given as (2/3) t1e. The values of t1e are shown in Note 3.
5. If DACK# transitions before RD#, then this specification is ignored. If there is no transition on DACK#, then this becomes the DRQ inactive delay.
6. TC width is defined as the time that both TC and DACK# are active. Note that TC and DACK# must overlap at least 50 ns.

NOTES: (Continued)

7. Based on the internal clock period (t1e). For various data rates, the read and write data width minimum values are:

Disk Drive Data Rate	24 MHz
1 Mbps	150 ns
500 Kbps	360 ns
300 Kbps	615 ns
250 Kbps	740 ns

8.This timing is a function of the selected data rate as follows:

Disk Drive Data Rate	Timing
1 Mbps	1.0 μs Min
500 Kbps	2.0 μs Min
300 Kbps	3.3 μs Min
250 Kbps	4.0 μs Min

9. This value can range from 0.5 ms to 8.0 ms and is dependent upon data rate and the Specify Command value.

10. The minimum MFM values for WE# to HDSEL# change for the various data rates are:

Disk Drive Data Rate	Min MFM Value
1 Mbps	0.5 ms + [8 × GPL]
500 Kbps	1.0 ms + [16 × GPL]
300 Kbps	1.6 ms + [26.66 × GPL]
250 Kbps	2.0 ms + [32 × GPL]

GPL is the size of gap 3 defined in the sixth byte of a Write Command.

11. Based on internal clock period.

12. Jitter tolerance is defined as:
(Maximum bit shift from nominal position ÷ ¼ period of nominal data rate) × 100 percent is a measure of the allowable bit jitter that may be present and still be correctly detected. The data separator jitter tolerance is measured under dynamic conditions that jitters the bit stream according to a reverse precompensation algorithm.

13. The minimum reset active period for a software reset is dependent on the data rate, after the FDC module has been properly reset using the t10a spec. The minimum software reset period then becomes:

Disk Drive Data Rate	Minimum Software Reset Active Period
	24 MHz
1 Mbps	125 ns
500 Kbps	250 ns
300 Kbps	420 ns
250 Kbps	500 ns

11.4.1 CLOCK TIMINGS

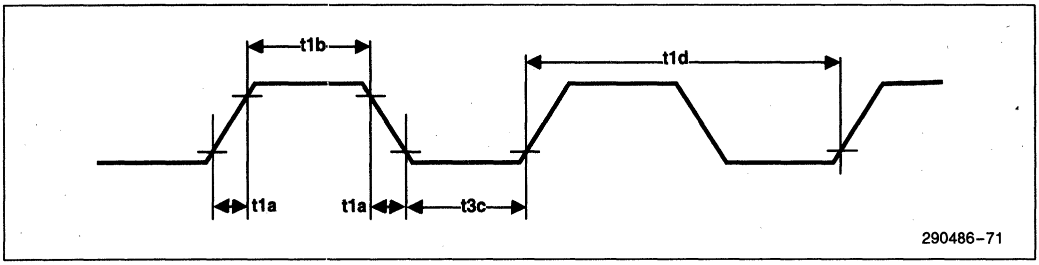


Figure 71. Clock Timing

11.4.2 HOST TIMINGS

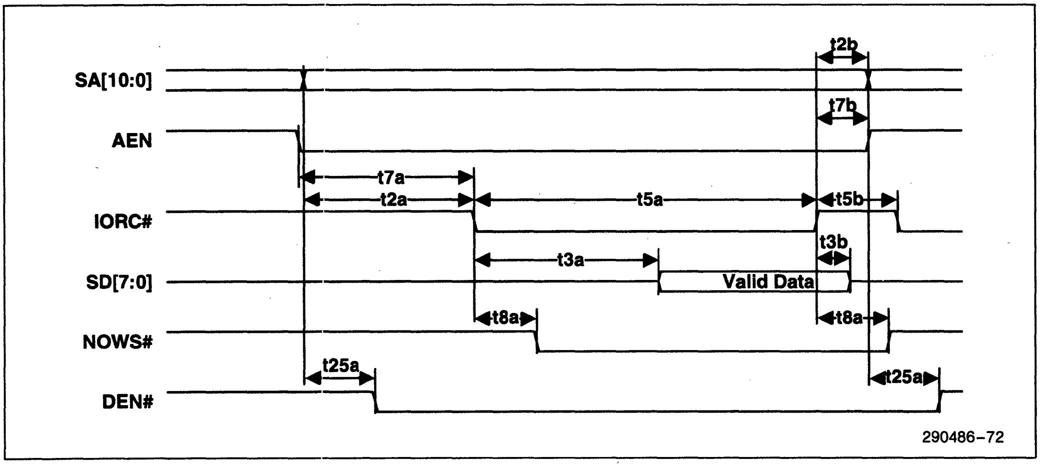


Figure 72. Host Read

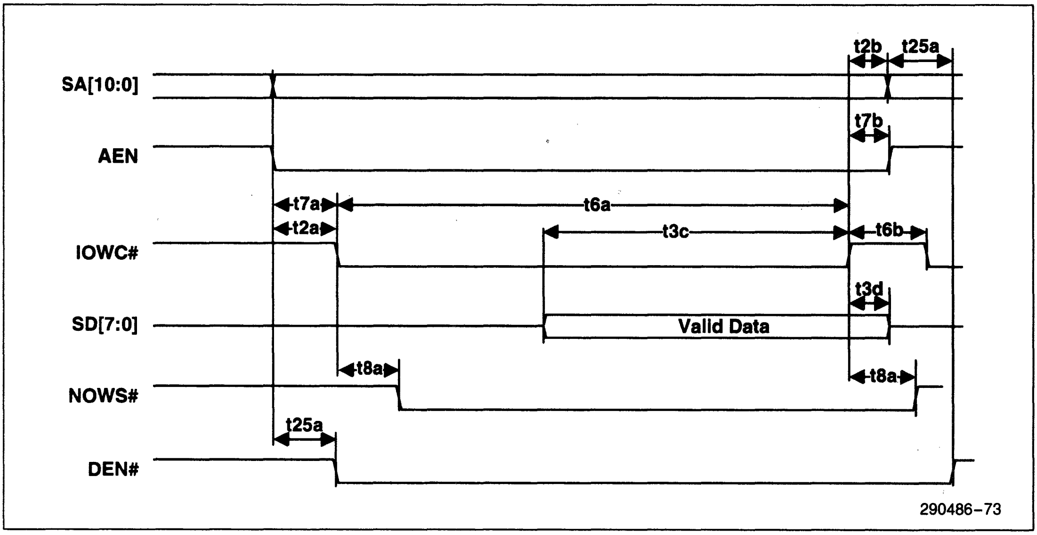


Figure 73. Host Write

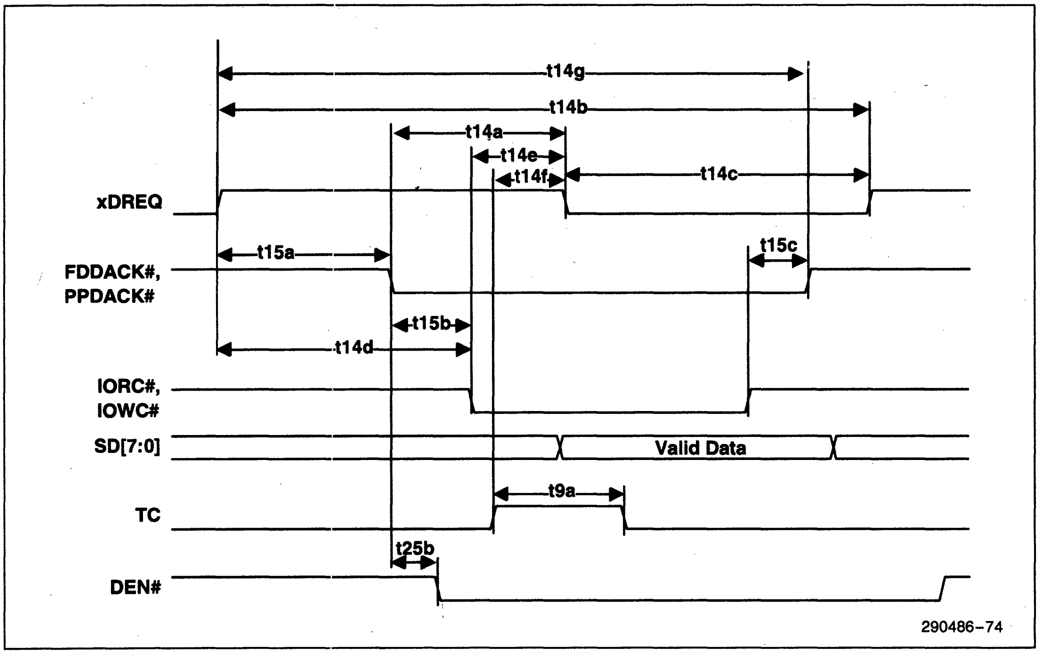
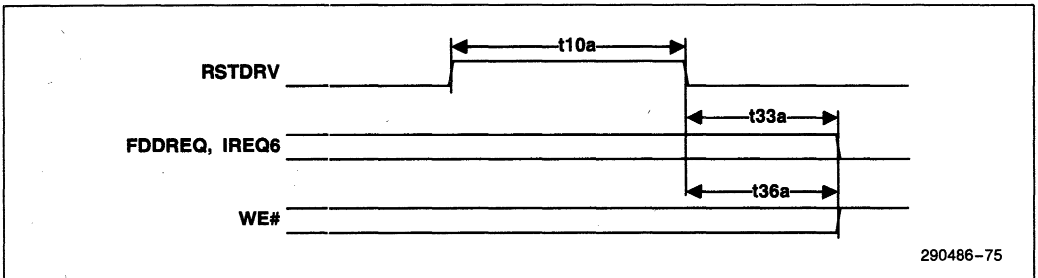


Figure 74. DMA Timing



NOTE:
FDDREQ, IRQ6 depicts the FDC enabled condition under hardware configuration. Otherwise, these signals tri-state with the same timing as IRQ[7,5,4,3].

Figure 75. Reset Timing

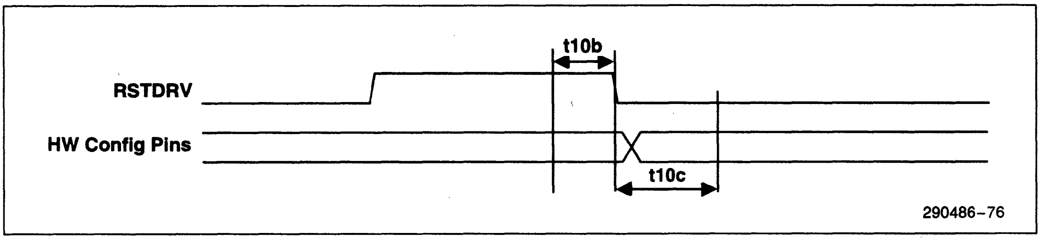


Figure 76. Reset Timing (Hardware Extended Configuration Mode)

11.4.3 FDC TIMINGS

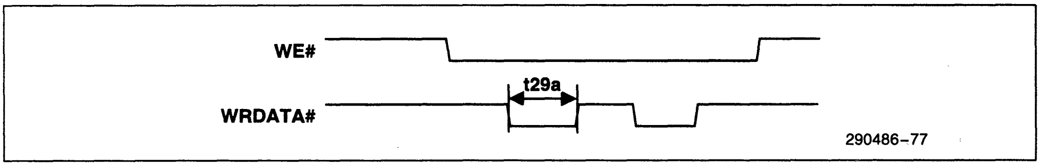
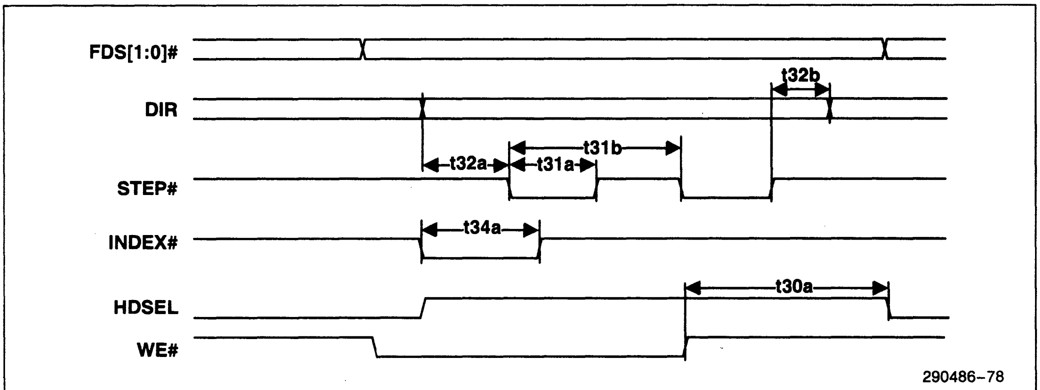


Figure 77. Write Data Timing



NOTE:
For overlapped seeks, only one step pulse per drive selection is issued. Non-overlapped seeks will issue all programmed step pulses.

Figure 78. FDC Drive Control/Timing

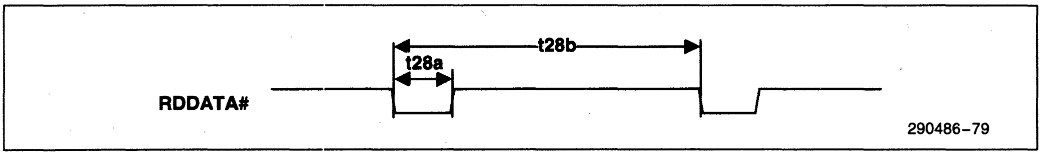


Figure 79. FDC Internal PLL Timing

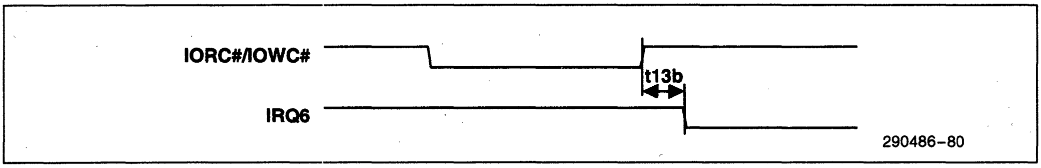


Figure 80. Floppy Disk Controller Interrupts

11.4.4 PARALLEL PORT TIMINGS

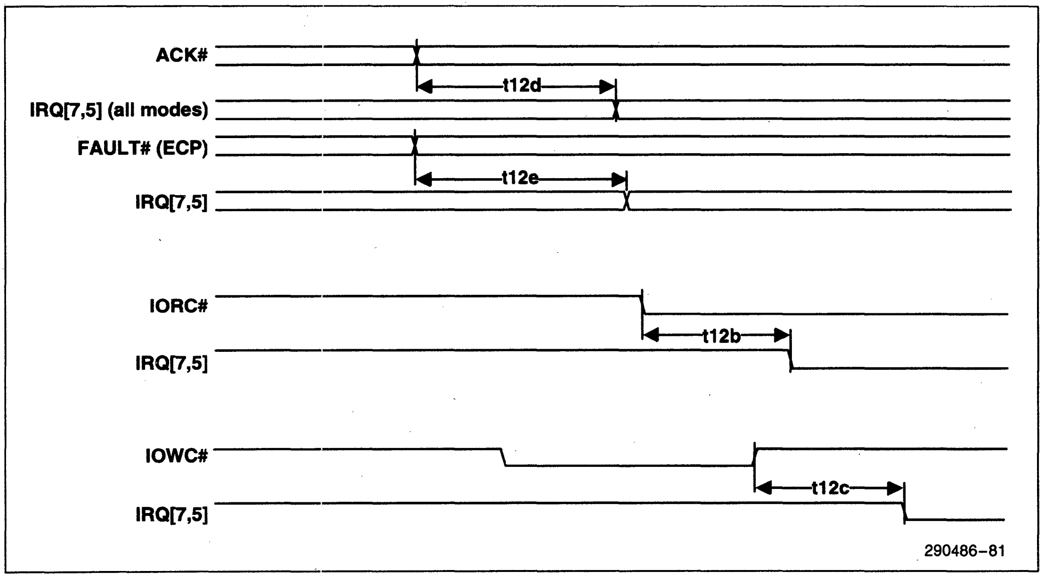


Figure 81. Parallel Port Interrupt Timing

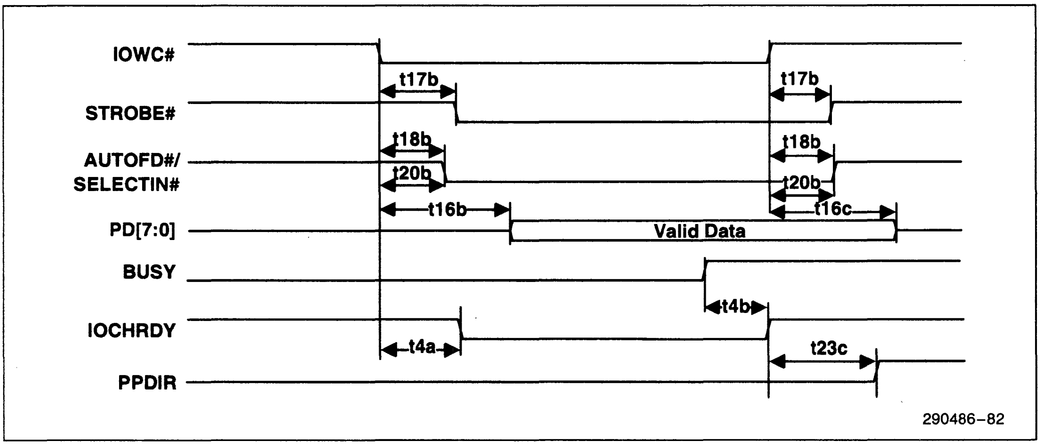


Figure 82. EPP Write Timing

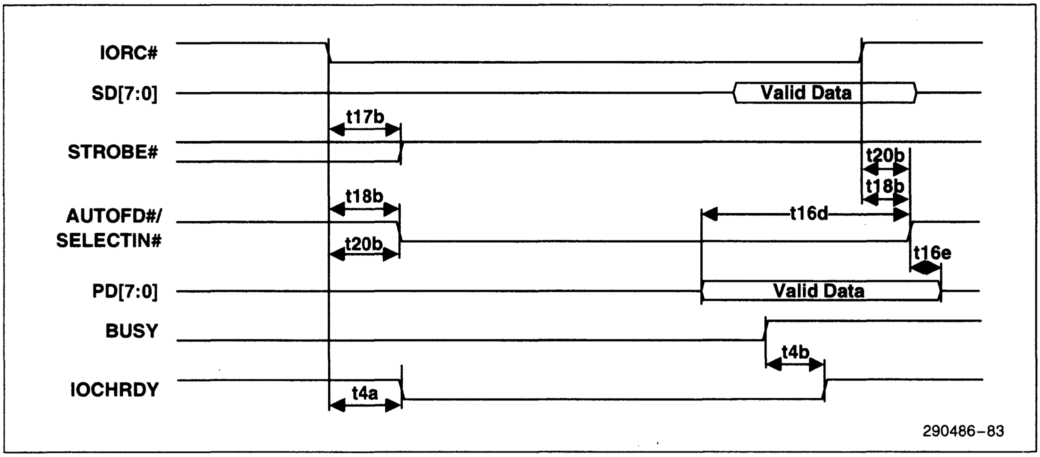


Figure 83. EPP Read Timing

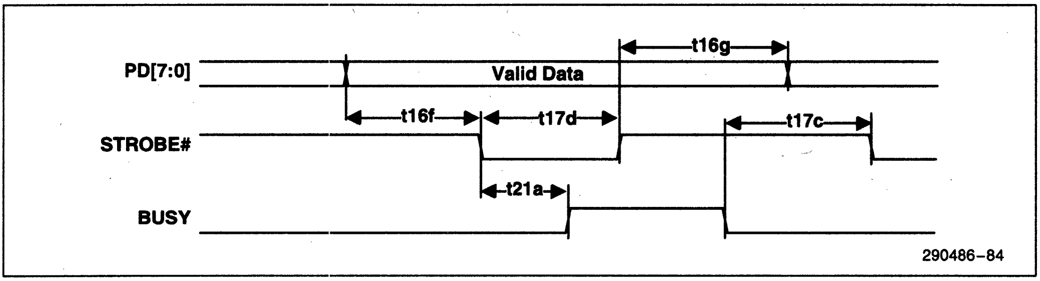


Figure 84. ISA-Compatible FIFO Timing

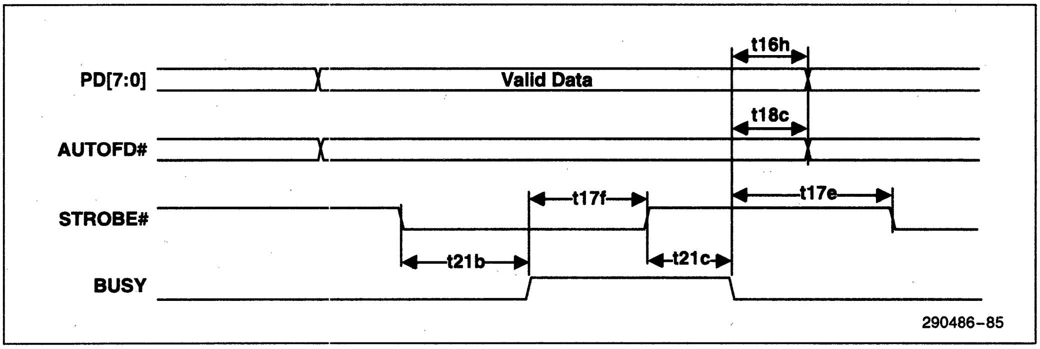


Figure 85. ECP Write Timing (Forward Direction)

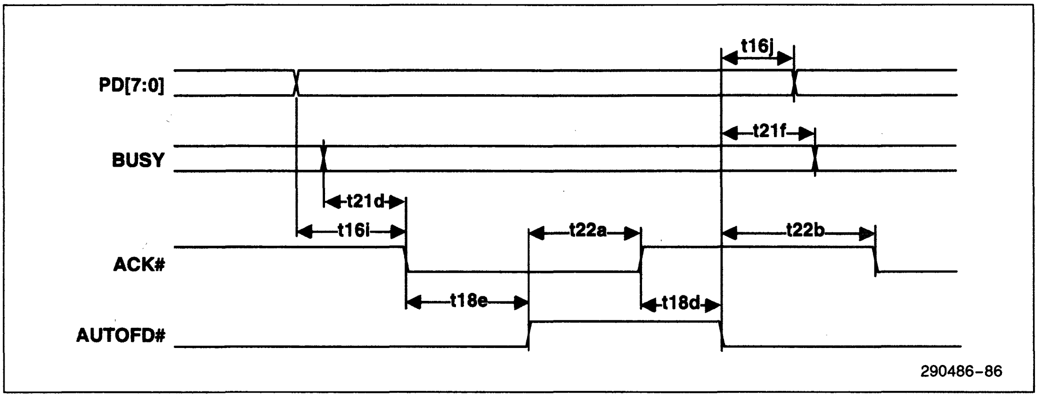


Figure 86. ECP Read Timing (Reverse Direction)

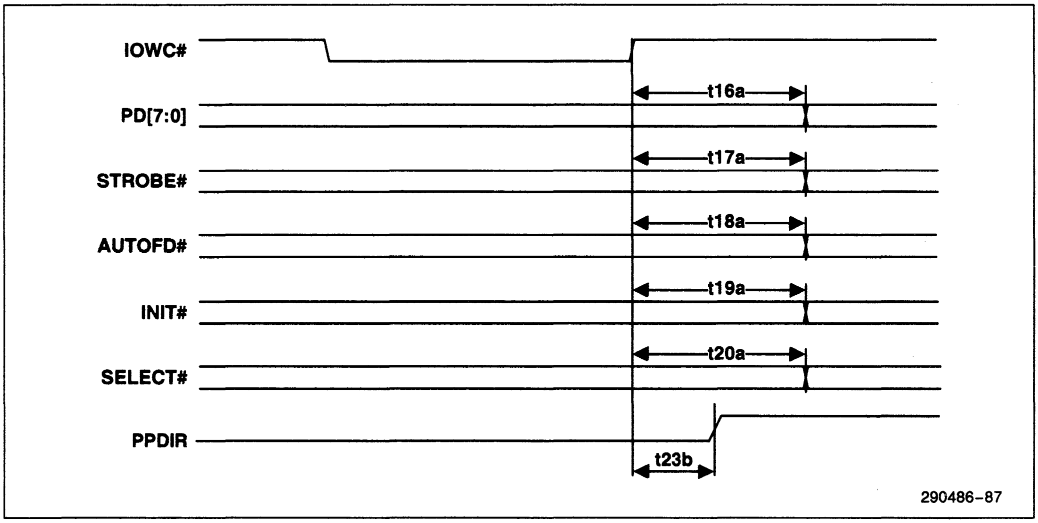


Figure 87. ISA-Compatible Write Timing

11.4.5 IDE TIMINGS

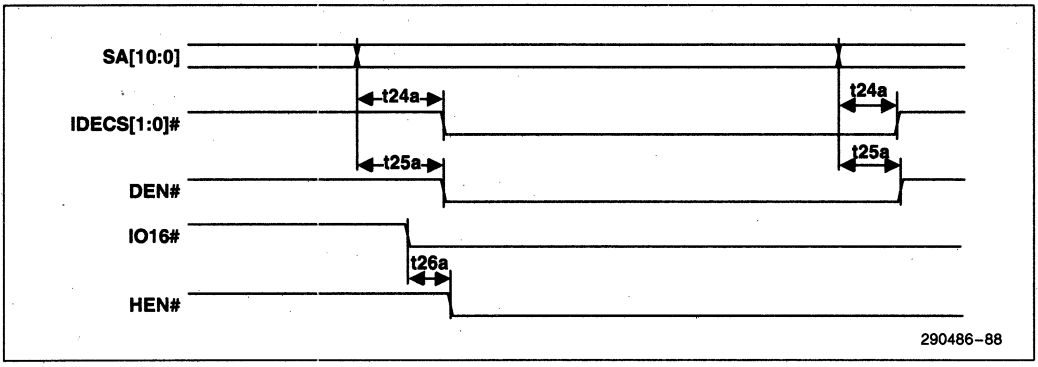


Figure 88. IDE Timing

11.4.6 GAME PORT TIMINGS

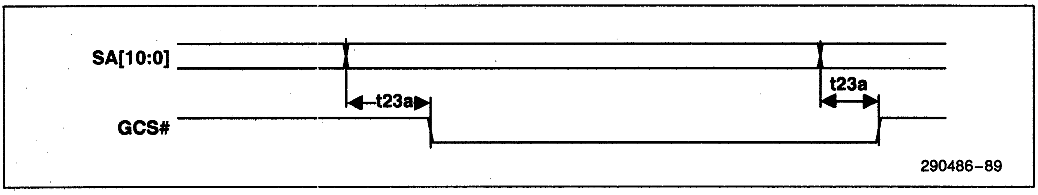


Figure 89. Game Port Timing

11.4.7 SERIAL PORT TIMINGS

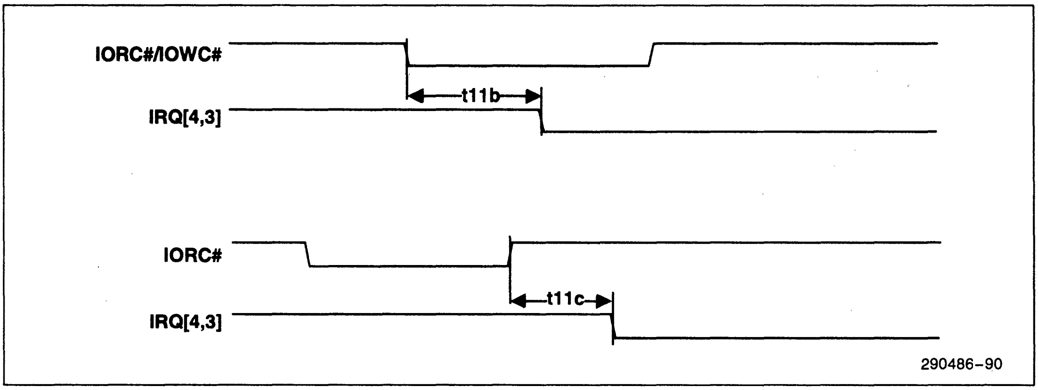


Figure 90. Serial Port Interrupt Timing

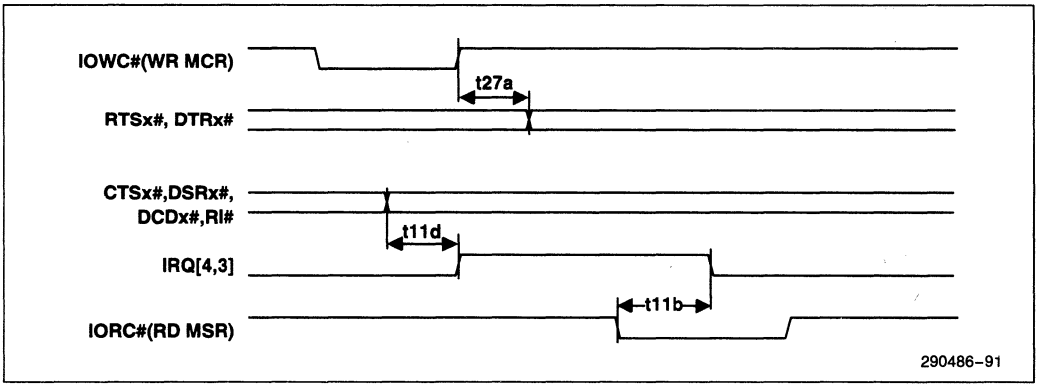


Figure 91. Modem Control Timing

12.0 PINOUT AND PACKAGE INFORMATION

12.1 Pin Assignment

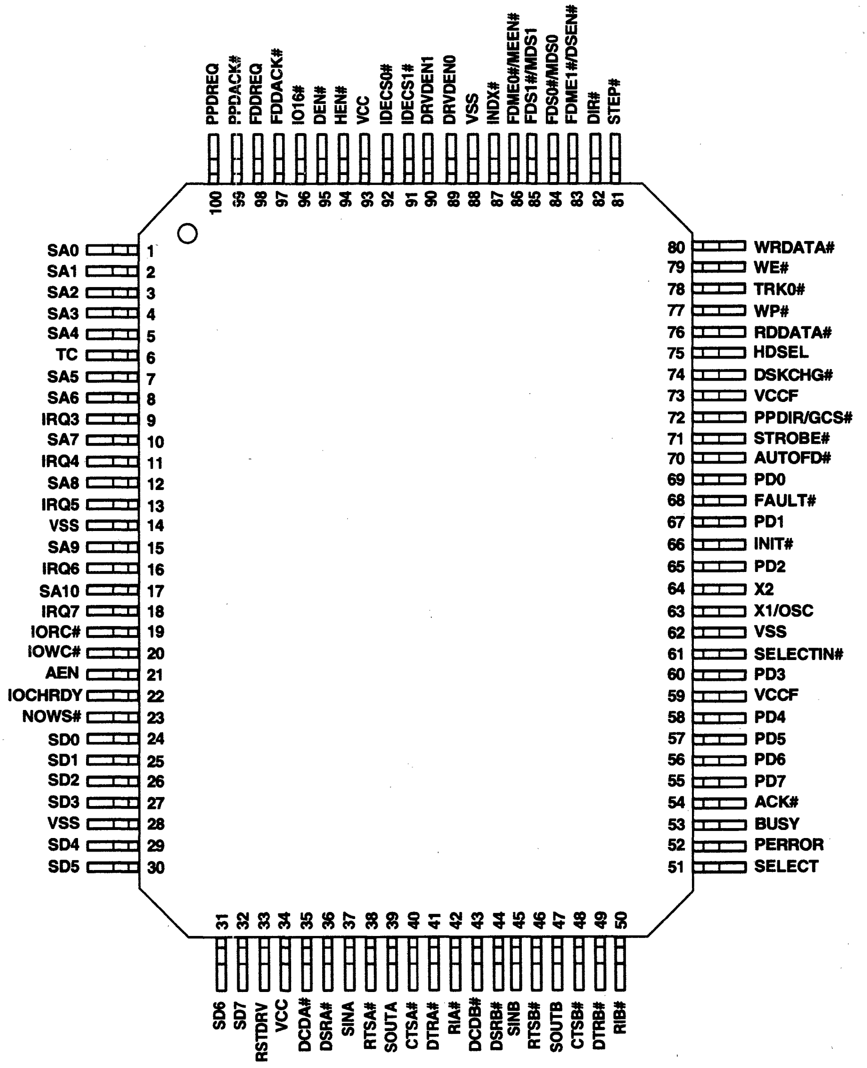


Figure 92. 82091AA Pin Diagram

Table 44. Alphabetical 82091AA Pin Assignment

Signal Name	Pin #	Type
ACK#	54	I
AEN	21	I
AUTOFD#	70	O
BUSY	53	I
CTSA#	40	I
CTSB#	48	I
DCDA#	35	O
DCDB#	43	O
DEN#	95	I/O
DIR#	82	O
DRVDEN0	89	O
DRVDEN1	90	O
DSKCHG#	74	I
DSRA#	36	I
DSRB#	44	I
DTRA#	41	I/O
DTRB#	49	I/O
FAULT#	68	I
FDDACK#	97	I
FDDREQ	98	O
FDME0#/MEEN#	86	O
FDME1#/DSEN#	83	O
FDS0#/MDS0	84	O
FDS1#/MDS1	85	O
HDSEL	75	O
HEN#	94	I/O
IDECS0#	92	I/O
IDECS1#	91	I/O
INDX#	87	I
INIT#	66	O
IO16#	96	I
IOCHRDY	22	O
IORC#	19	I
IOWC#	20	I
IRQ3	9	O
IRQ4	11	O
IRQ5	13	O
IRQ6	16	O
IRQ7	18	O
NOWS#	23	O
PD0	69	I/O
PD1	67	I/O
PD2	65	I/O
PD3	60	I/O
PD4	58	I/O
PD5	57	I/O
PD6	56	I/O
PD7	55	I/O
PERROR	52	I
PPDACK#	99	I
PPDREQ	100	O
PPDIR/GCS#	72	I/O
RDDATA#	76	I
RIA#	42	I
RIB#	50	I
RSTDRV	33	I
RTSA#	38	I/O
RTSB#	46	I/O
SA0	1	I
SA1	2	I
SA2	3	I
SA3	4	I
SA4	5	I
SA5	7	I
SA6	8	I
SA7	10	I
SA8	12	I
SA9	15	I

Table 44. Alphabetical 82091AA Pin Assignment (Continued)

Signal Name	Pin #	Type	Signal Name	Pin #	Type
SA10	17	I	STROBE#	71	O
SD0	24	I/O	TC	6	I
SD1	25	I/O	TRK0#	78	I
SD2	26	I/O	V_{CC}	34	V
SD3	27	I/O	V_{CC}	93	V
SD4	29	I/O	V_{CCF}	59	V
SD5	30	I/O	V_{CCF}	73	V
SD6	31	I/O	V_{SS}	14	V
SD7	32	I/O	V_{SS}	28	V
SINA	37	I	V_{SS}	62	V
SINB	45	I	V_{SS}	88	V
SELECT	51	I	WE#	79	O
SELECTIN#	61	O	WP#	77	I
SOUTA	39	I/O	WRDATA#	80	O
SOUTB	47	I/O	X1/OSC	63	I
STEP#	81	O	X2	64	I

Table 45. Numerical 82091AA Pin Assignment

Pin#	Signal Name	Type	Pin#	Signal Name	Type
1	SA0	I	16	IRQ6	O
2	SA1	I	17	SA10	I
3	SA2	I	18	IRQ7	O
4	SA3	I	19	IORC#	I
5	SA4	I	20	IOWC#	I
6	TC	I	21	AEN	I
7	SA5	I	22	IOCHRDY	O
8	SA6	I	23	NOWS#	O
9	IRQ3	O	24	SD0	I/O
10	SA7	I	25	SD1	I/O
11	IRQ4	O	26	SD2	I/O
12	SA8	I	27	SD3	I/O
13	IRQ5	O	28	V_{SS}	V
14	V_{SS}	V	29	SD4	I/O
15	SA9	I	30	SD5	I/O

Table 45. Numerical 82091AA Pin Assignment (Continued)

Pin #	Signal Name	Type
31	SD6	I/O
32	SD7	I/O
33	RSTDRV	I
34	V_{CC}	V
35	DCDA#	O
36	DSRA#	I
37	SINA	I
38	RTSA#	I/O
39	SOUTA	I/O
40	CTSA#	I
41	DTRA#	I/O
42	RIA#	I
43	DCDB#	O
44	DSRB#	I
45	SINB	I
46	RTSB#	I/O
47	SOUTB	I/O
48	CTSB#	I
49	DTRB#	I/O
50	RIB#	I
51	SELECT	I
52	PERROR	I
53	BUSY	I
54	ACK#	I
55	PD7	I/O
56	PD6	I/O
57	PD5	I/O
58	PD4	I/O
59	V_{CCF}	V
60	PD3	I/O
61	SELECTIN#	O
62	V_{SS}	V
63	X1/OSC	I
64	X2	I
65	PD2	I/O
66	INIT#	O
67	PD1	I/O
68	FAULT#	I
69	PD0	I/O
70	AUTOFD#	O
71	STROBE#	O
72	PPDIR/GCS#	I/O
73	V_{CCF}	V
74	DSKCHG#	I
75	HDSEL	O
76	RDDATA#	I
77	WP#	I
78	TRK0#	I
79	WE#	O
80	WRDATA#	O
81	STEP#	O
82	DIR#	O
83	FDME1#/DSEN#	O
84	FDS0#/MDS0	O
85	FDS1#/MDS1	O
86	FDME0#/MEEN#	O
87	INDX#	I
88	V_{SS}	V
89	DRVDEN0	O
90	DRVDEN1	O
91	IDECS1#	I/O
92	IDECS0#	I/O
93	V_{CC}	I/O
94	HEN#	V
95	DEN#	I/O
96	IO16#	I
97	FDDACK#	I
98	FDDREQ	O
99	PPDACK#	I
100	PPDREQ	O

12.2 Package Characteristics

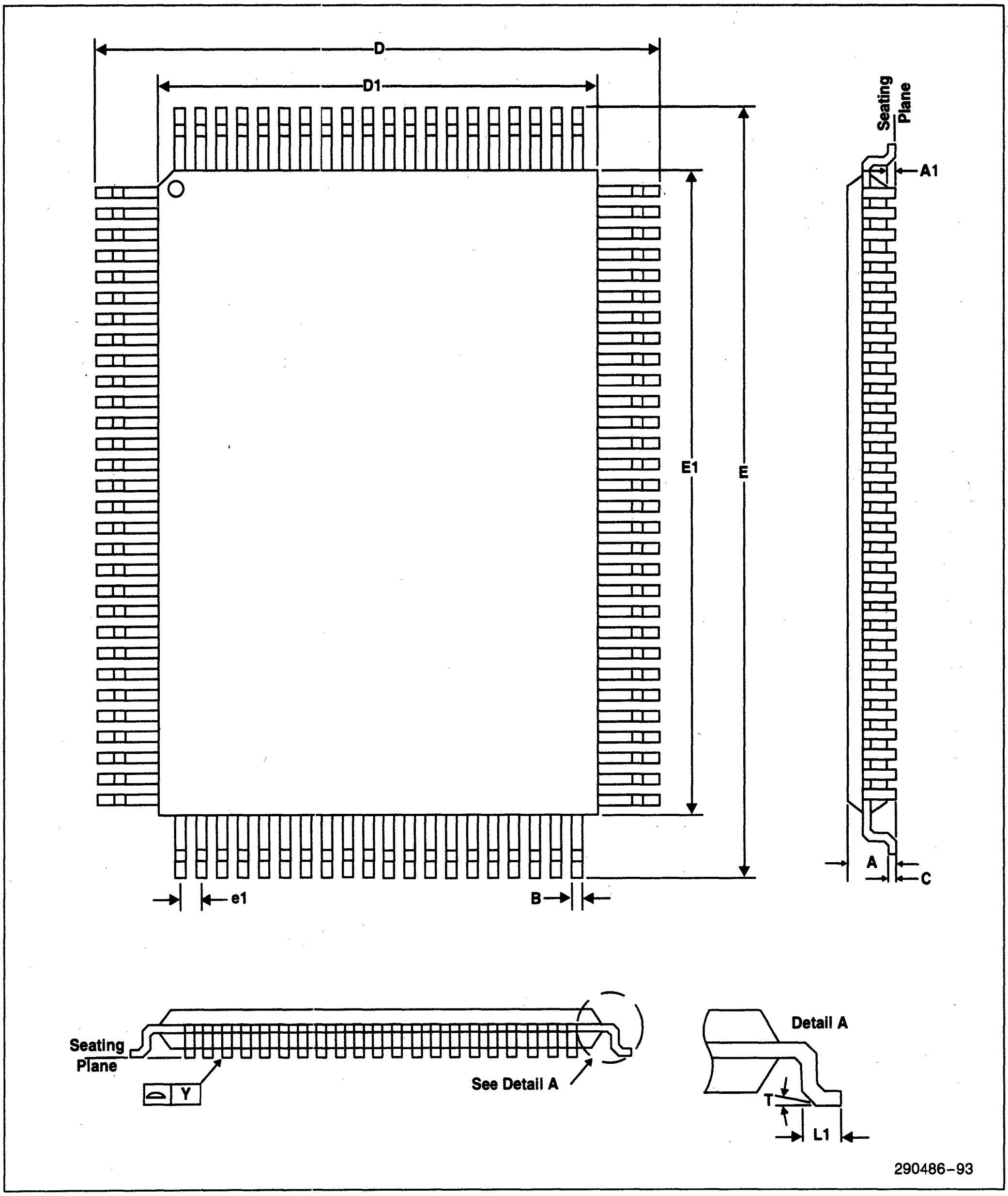
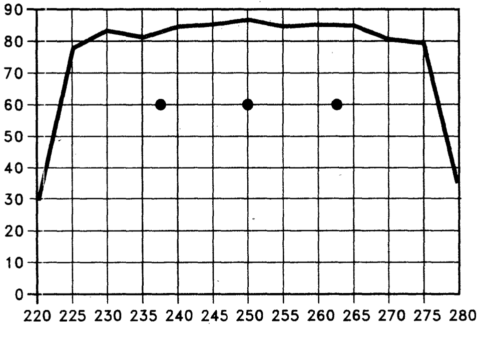


Figure 93. 100-Pin Quad Flat Pack (QFP) Dimensions

Quad Flat Pack Package

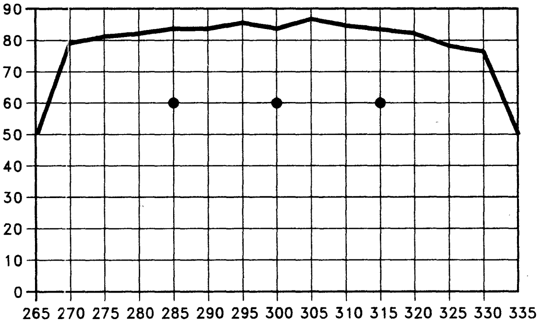
Symbol	Millimeters			
	Minimum	**Nominal**	**Maximum**	**Notes**
A			3.15	
A1	0.0			
B	0.20	0.30	0.40	
C	0.10	0.15	0.20	
D	17.5	17.9	18.3	
D1		14.0		
E	23.5	23.9	24.3	
E1		20.0		
e1	0.53	0.65	0.77	
L1	0.60	0.80	1.00	
N	100			Rectangle
T	0.00		10.0	
Y			0.10	
ISSUE	JEDEC			

13.0 DATA SEPARATOR CHARACTERISTICS FOR FLOPPY DISK MODE



290486–94

Figure 94. Typical Jitter Tolerance vs Data Rate (Capture Range 250 Kbps)



290486–95

Figure 95. Typical Jitter Tolerance vs Data Rate (Capture Range 300 Kbps)

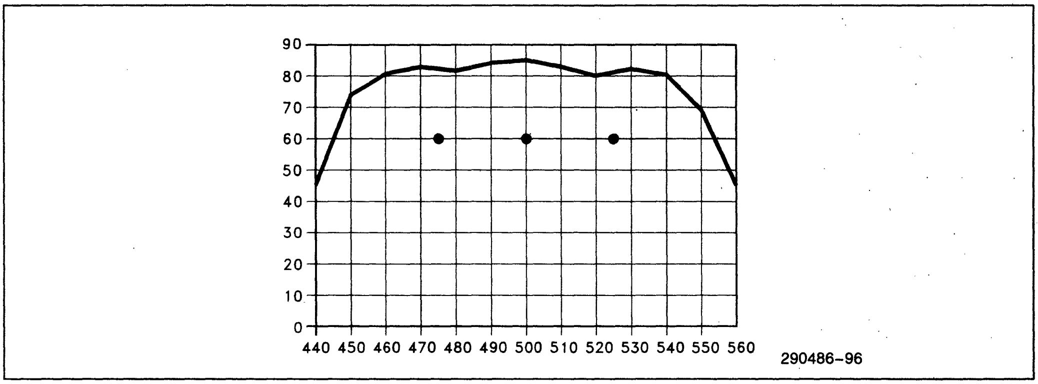


Figure 96. Typical Jitter Tolerance vs Data Rate (Capture Range 500 Kbps)

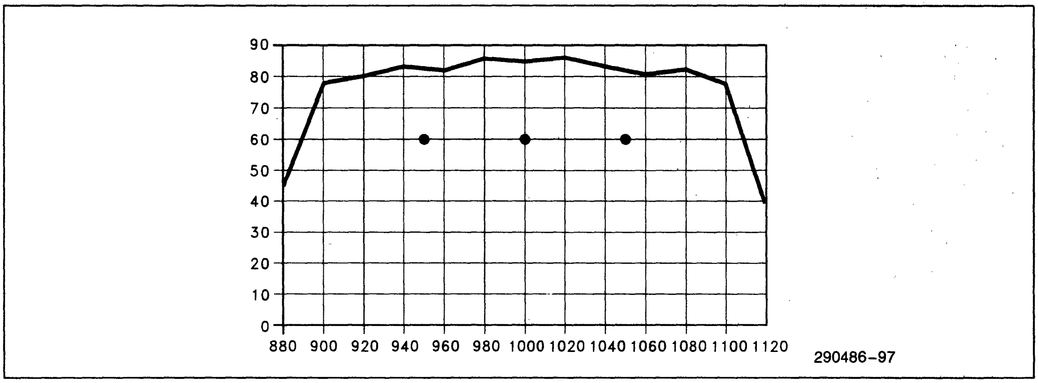


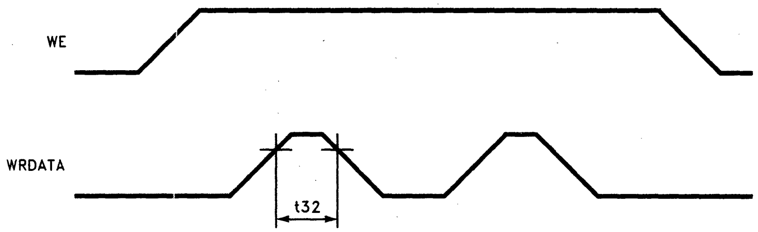
Figure 97. Typical Jitter Tolerance vs Data Range (Capture Range 1 Mbps)

Jitter Tolerance measured in percent. Capture range expressed as a percent of data rate, i.e., ±3% percent.

● = Test Points: 250 Kbps, 300 Kbps, 500 Kbps and 1 Mbps are center, ±5 percent @ 60 percent jitter.

Test points are tested at temparture and V_{CC} limits. Refer to the datasheet. Typical conditions are: room temperature, nominal V_{CC}.

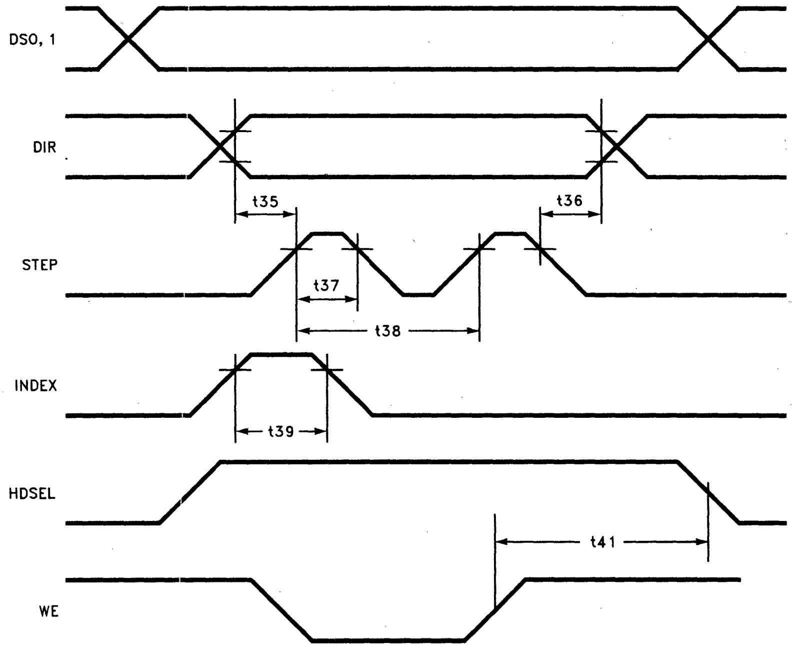
13.1 Write Data Timing



290486–98

NOTE:
Invert high.

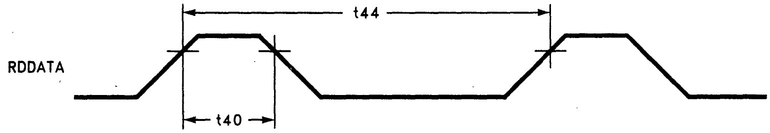
13.2 Drive Control



290486–99

NOTE:
For overlapped seeks, only one step pulse per drive selection is issued. Non-overlapped seeks will issue all programmed step pulses. Invert high.

13.3 Internal PLL



290486-A0

NOTE:
Invert high.

APPENDIX A
FDC FOUR DRIVE SUPPORT

Section 8.0 of this document completely describes the FDC when the module is configured for two drive support. In addition, the FDC commands in Section 8.0 provide four drive support information. This appendix provides additional information concerning four drive support. The signal pins that are affected by four drive support are described in Section A.1. Note that the FDC signals not discussed in this appendix operate the same for both two and four drive systems. The following registers are described in this appendix; Digital Output Register (DOR), Enhanced Tape Drive Register (TDR), and the Main Status Register (MSR). Some bits in these registers operate differently in a four drive configuration than a two drive configuration.

NOTES:

- The descriptions in this appendix assume that four floppy drive support has been selected by setting FDDQTY to 1 in the AIPCFG1 Register.
- Only drive 0 or drive 1 can be selected as the boot drive.

A.1 Floppy Disk Controller Interface Signals

These signal descriptions are for a four drive system (FDDQTY=1 in the AIPCFG1 Register). See Section 2.0 for two drive system signal descriptions.

Signal Name	Type	Description
FDME1#/DSEN#(1)	O	**FLOPPY DRIVE MOTOR ENABLE 1, or DRIVE SELECT ENABLE:** In a four drive system, this signal functions as a drive select enable (DSEN#). When DSEN# is asserted, MDS1 and MDS0 reflect the selection of the drive.
FDS1#/MDS1(1)	O	**FLOPPY DRIVE SELECT1, or MOTOR DRIVE SELECT 1:** In a four drive system, this signal functions as a motor drive select (MDS1). MDS1, together with MDS0, indicate which of the four drives is selected, as shown in note 1.
FDME0#/MEEN#(1)	O	**FLOPPY DRIVE MOTOR ENABLE 0 or MOTOR ENABLE ENABLE:** In a four drive system, this signal functions as a motor enable enable (MEEN#). MEEN# is asserted to enable the external decoding of MDS1 and MDS0 for the appropriate motor enable (see note 1).
FDS0#/MDS0(1)	O	**FLOPPY DRIVE SELECT 0 or MOTOR DRIVE SELECT 0:** In a four drive system, this signal functions as motor drive select (MDS0). MDS0, together with MDS1, indicate which of the four drives is selected as shown in note 1.

NOTE:

1. These signal pins are used to control an external decoder for four floppy disk drives as shown below. Refer to the DOR Register Description in Section A.2 for details.

MDS1	MDS0	DSEN# = 0	MEEN# = 0
0	0	Drive 0	ME0
0	1	Drive 1	ME1
1	0	Drive 2	ME2
1	1	Drive 3	ME3

A.2 DOR—Digital Output Register

I/O Address:	Base +2h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

The Digital Output Register enables/disables the floppy disk drive motors, selects the disk drives, enables/disables DMA, and provides a FDC module reset. The DOR reset bit and the Motor Enable bits have to be inactive when the 82091AA's FDC is in powerdown. The DMAGATE# and Drive Select bits are unchanged. During powerdown, writing to the DOR does not wake up the 82091AA's FDC, except for activating any of the motor enable bits. Setting the motor enable bits to 1 will wake up the module. The four internal drive select and four internal motor enable signals are encoded to a total of four output pins as described in Table 47. Figure 99 shows an example of how these four output pins can be decoded to provide four drive select and four motor enable signals. Note that only drive 0 or drive 1 can be used as the boot drive when four disk drives are enabled.

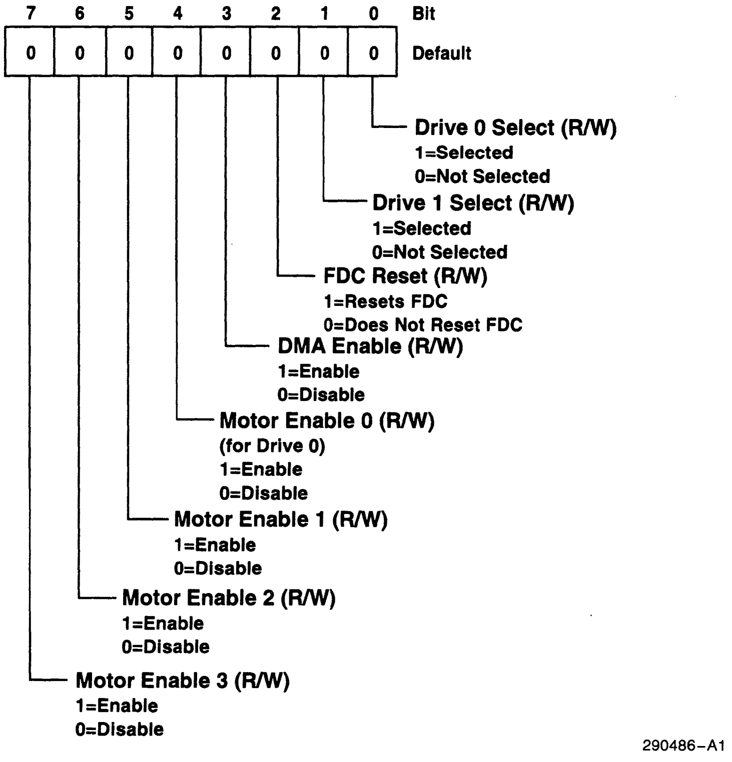


Figure 98. Digital Output Register

Bit	Description
7	**Motor Enable 3 (ME3):** This bit controls a motor drive enable output signal and provides the signal output for the floppy drive 3 motor (via external decoding) as shown in Table 46.
6	**Motor Enable 2 (ME2):** This bit controls a motor drive enable output signal and provides the signal output for the floppy drive 2 motor (via external decoding) as shown in Table 46.
5	**Motor Enable 1 (ME1):** This bit controls a motor drive enable signal and provides the signal output for the floppy drive 1 motor (via external decoding) as shown in Table 46.
4	**Motor Enable 0 (ME0):** This bit controls a motor drive enable signal and provides the signal output for the floppy drive 0 motor (via external decoding) as shown in Table 46.
3	**DMA Gate (DMAGATE):** This bit enables/disables DMA for the FDC. When DMAGATE = 1, DMA for the FDC is enabled. In this mode FDDREQ, TC, IRQ6, and FDDACK# are enabled. When DMAGATE = 0, DMA for the FDC is disabled. In this mode, the IRQ6 and DRQ outputs are tri-stated and the DACK# and TC inputs are disabled to the FDC. Note that the TC input is only disabled to the FDC module. Other functional units in the 82091AA (e.g., parallel port or IDE interface) can still use the TC input signal for DMA activities.
2	**FDC Reset (DORRST):** DORRST is a software reset for the FDC module. When DORRST is set to 0, the basic core of the 82091AA's FDC and the FIFO circuits are cleared conditioned by the LOCK bit in the Configure Command. This bit is set to 0 by software or a hard reset (RSTDRV asserted). The FDC remains in a reset state until software sets this bit to 1. This bit does not affect the DSR, CCR and other bits of the DOR. DORRST must be held active for at least 0.5 μs at 250 Kbps. This is less than a typical ISA I/O cycle time. Thus, in most systems consecutive writes to this register to toggle this bit allows sufficient time to reset the FDC.
1:0	**Drive Select (DS[1:0]):** This field provides the output signals to select a particular floppy drive (via external decoding) as shown in Table 47. Note that the drive motor can be enabled separately without selecting the drive. This permits the motor to come up to speed before selecting the drive. Note also that only one drive can be selected at a time. However, the drive should not be selected without enabling the appropriate drive motor via bits[7:4] of this register.

Table 46. Output Pin Status for Four Disk Drives

	FDC DOR Register Bits						Signal Pins			
Description	**ME3**	**ME2**	**ME1**	**ME0**	**DS1**	**DS0**	**MDS1#**	**MDS0#**	**DSEN#**	**MEEN#**
ME0 and DS0 enable	X	X	X	1	0	0	0	0	0	0
ME1 and DS1 enable	X	X	1	X	0	1	0	1	0	0
ME2 and DS2 enable	X	1	X	X	1	0	1	0	0	0
ME3 and DS3 enable	1	X	X	X	1	1	1	1	0	0
ME0 enable only	X	X	X	1	DS[1:0]≠00		0	0	1	0
ME1 enable only	X	X	1	0	DS[1:0]≠01		0	1	1	0
ME2 enable only	X	1	0	0	DS[1:0]≠10		1	0	1	0
ME3 enable only	1	0	0	0	DS[1:0]≠11		1	1	1	0
No ME or DS enable	0	0	0	0	X	X	1	1	1	1

NOTE:
To enable a particular drive motor and select the drive, the value for DS[1:0] must match the appropriate motor enable bit selected as indicated in the first four rows of the table. For example, to enable the drive 0 motor and select the drive, ME0 is set to 1 and DS[1:0] must be set to 00. To enable the drive motor and keep the drive de-selected the value for DS[1:0] must not match the particular motor enable as shown in the first four rows. For example, to enable the motor for drive 0 while the drive remains de-selected, ME0 is set to 1 and DS[1:0] is set to 01, 10, or 11.

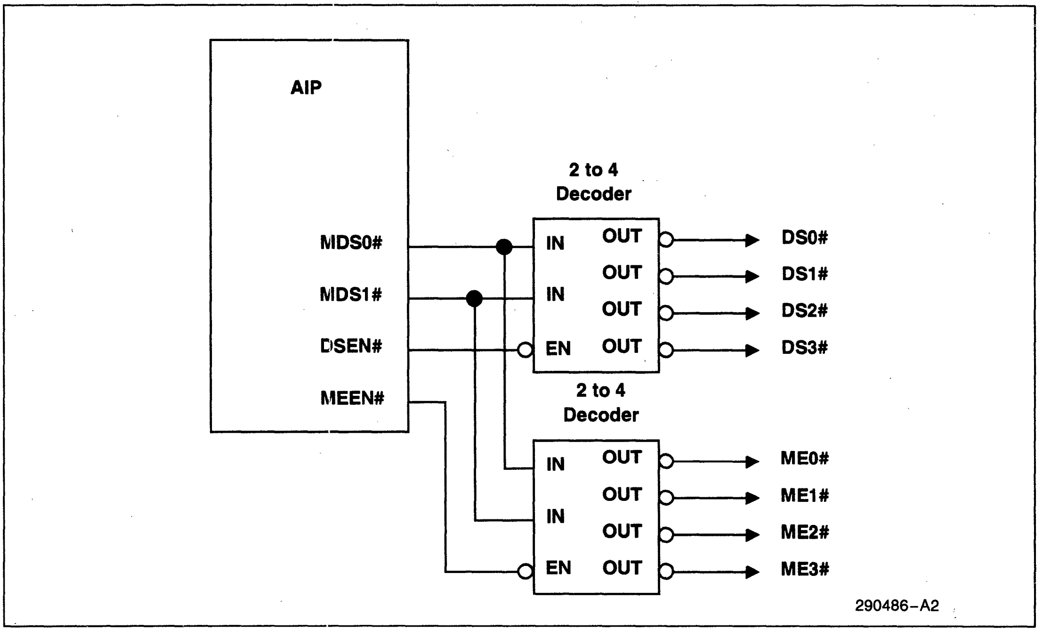


Figure 99. Example External Decoder (Four Drive System)

A.3 TDR—Enhanced Tape Drive Register

I/O Address:	Base +3h
Default Value:	00h
Attribute:	Read/Write
Size:	8 bits

This register allows the user to assign tape support to a particular drive during initialization. Any future references to that drive number automatically invokes tape support. A hardware reset sets all bits in this register to 0 making drive 0 not available for tape support. A software reset via bit 2 of the DOR does not affect this register. Drive 0 is reserved for the floppy boot drive. Bits[7:2] are only available when EREG EN = 1; otherwise the bits are tri-stated.

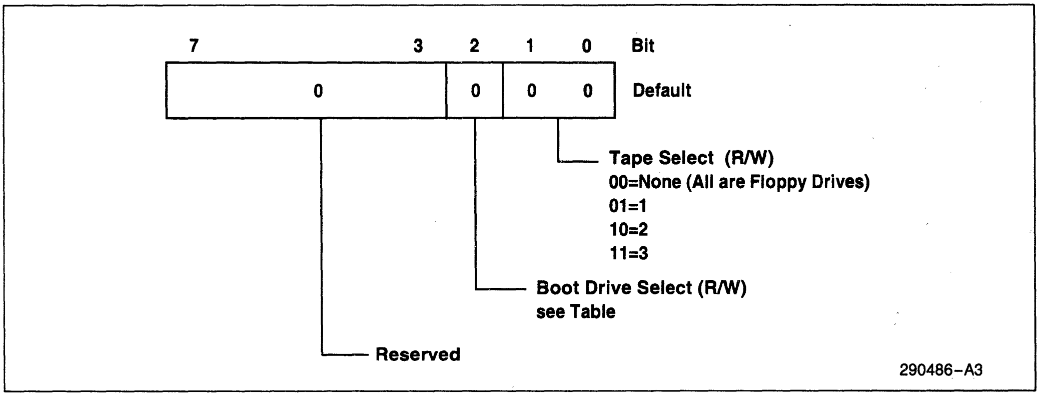


Figure 100. Enhanced Tape Drive Register

Bit	Description
7:3	**Reserved:**
2	**Boot Drive Select (BOOTSEL):** The BOOTSEL bit is used to remap the drive selects and motor enables. The functionality is shown below: **BOOTSEL** **Mapping** 0 DS0 → FDS0, ME0 → FDME0 (default) DS1 → DS1, ME1 → FDME1 1 DS0 → DS1, ME0 → FDME1 DS1 → FDS0, ME1 → FDME0 Only drive 0 or drive 1 can be selected as the boot drive.
1:0	**Tape Select (TAPESEL[1:0]):** These two bits are used by software to assign a logical drive number to be a tape drive. Other than adjusting precompensation delays for tape support, these two bits do not affect the FDC hardware. They can be written and read by software as an indication of the tape drive assignment. Drive 0 is not available as a tape drive and is reserved as the floppy disk boot drive. The tape drive assignments are as follows: **Bits[1:0]** **Drive Selected** 0 0 None (all are floppy disk drives) 0 1 1 1 0 2 1 1 3

A.4 MSR—Main Status Register

I/O Address:	Base +4h
Default Value:	00h
Attribute:	Read Only
Size:	8 bits

This read only register provides FDC status information. This information is used by software to control the flow of data to and from the FIFO (accessed via the FDCFIFO Register). The MSR indicates when the FDC is ready to send or receive data through the FIFO. During non-DMA transfers, this register should be read before each byte is transferred to or from the FIFO.

After a hard or soft reset or recovery from a powerdown state, the MSR is available to be read by the host. The register value is 00h until the oscillator circuit has stabilized and the internal registers have been initialized. When the FDC is ready to receive a new command, MSR[7:0] = 80h. The worst case time allowed for the MSR to report 80h (i.e., RQM is set to 1) is 2.5 μs after a hard or soft reset.

Main Status Register is used for controlling command input and result output for all commands. Some example values of the MSR are:

- MSR = 80H; The controller is ready to receive a command.
- MSR = 90H; Executing a command or waiting for the host to read status bytes (assume DMA mode).
- MSR = D0H; Waiting for the host to write status bytes.

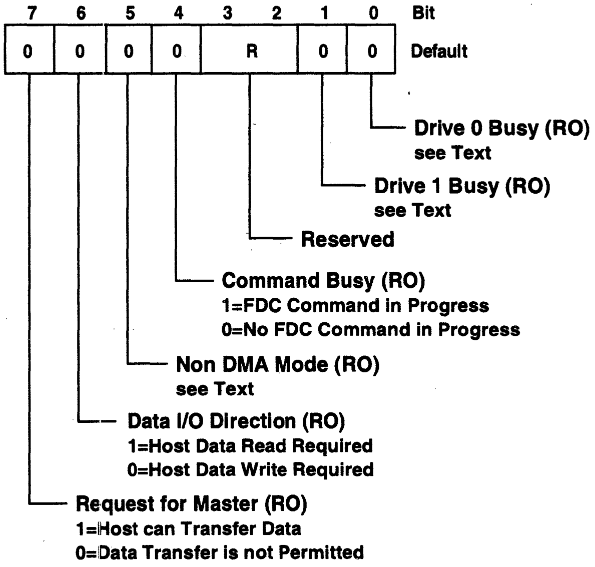


Figure 101. Main Status Register

Bit	Description
7	**Request For Master (RQM):** When RQM = 1, the FDC is ready to send/receive data through the FIFO (FDCFIFO Register). The FDC sets this bit to 0 after a byte transfer and then sets the bit to 1 when it is ready for the next byte. During non-DMA execution phase, RQM indicates the status of IRQ6.
6	**Direction I/O (DIO):** When RQM = 1, DIO indicates the direction of a data transfer. When DIO = 1, the FDC is requesting a read of the FDCFIFO. When DIO = 0, the FDC is requesting a write to the FDCFIFO.
5	**NON-DMA (NONDMA):** Non-DMA mode is selected via the SPECIFY Command. In this mode, the FDC sets this bit to a 1 during the execution phase of a command. This bit is for polled data transfers and helps differentiate between the data transfer phase and the reading of result bytes.
4	**Command Busy (CMDBUSY):** CMDBUSY indicates when a command is in progress. When the first byte of the command phase is written, the FDC sets this bit to 1. CMDBUSY is set to 0 after the last byte of the result phase is read. If there is no result phase (e.g., SEEK or RECALIBRATE Commands), CMDBUSY is set to 0 after the last command byte is written.
3	**Drive 3 Busy (DRV1BUSY):** The FDC module sets this bit to 1 after the last byte of the command phase of a SEEK or RECALIBRATE Command is issued for drive 3. This bit is set to 0 after the host reads the first byte in the result phase of the SENSE INTERRUPT Command for this drive.
2	**Drive 2 Busy (DRV1BUSY):** The FDC module sets this bit to 1 after the last byte of the command phase of a SEEK or RECALIBRATE Command is issued for drive 2. This bit is set to 0 after the host reads the first byte in the result phase of the SENSE INTERRUPT Command for this drive.
1	**Drive 1 Busy (DRV1BUSY):** The FDC module sets this bit to 1 after the last byte of the command phase of a SEEK or RECALIBRATE Command is issued for drive 1. This bit is set to 0 after the host reads the first byte in the result phase of the SENSE INTERRUPT Command for this drive.
0	**Drive 0 Busy (DRV0BUSY):** The FDC module sets this bit to 1 after the last byte of the command phase of a SEEK or RECALIBRATE Command is issued for drive 0. This bit is set to 0 after the host reads the first byte in the result phase of the SENSE INTERRUPT Command for this drive.

82078 CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER

- Small Footprint and Low Height Packages
- Supports Standard 5.0V as Well as Low Voltage 3.3V Platforms
  - Selectable 3.3V and 5.0V Configuration
  - 5.0V Tolerant Drive Interface
- Enhanced Power Management
  - Application Software Transparency
  - Programmable Powerdown Command
  - Save and Restore Commands for 0V Powerdown
  - Auto Powerdown and Wakeup Modes
  - Two External Power Management Pins
  - Consumes No Power While in Powerdown
- Programmable Internal Oscillator
- Floppy Drive Support Features
  - Drive Specification Command
  - Media ID Capability Provides Media Recognition
  - Drive ID Capability Allows the User to Recognize the Type of Drive
  - Selectable Boot Drive
  - Standard IBM and ISO Format Features
  - Format with Write Command for High Performance in Mass Floppy Duplication
- Integrated Host/Disk Interface Drivers
- Integrated Analog Data Separator
  - 250 Kbits/sec
  - 300 Kbits/ sec
  - 500 Kbits/ sec
  - 1 Mbits/sec
  - 2 Mbits/sec
- Integrated Tape Drive Support
  - Standard 1 Mbps/500 Kbps/ 250 Kbps Tape Drives
  - New 2 Mbps Tape Drive Mode
- Perpendicular Recording Support for 4 MB Drives
- Fully Decoded Drive Select and Motor Signals
- Programmable Write Precompensation Delays
- Addresses 256 Tracks Directly, Supports Unlimited Tracks
- 16 Byte FIFO
- Single-Chip Floppy Disk Controller Solution for Portables and Desktops
  - 100% PC-AT* Compatible
  - 100% PS/2* Compatible
  - 100% PS/2 Model 30 Compatible
  - Fully Compatible with Intel's 386SL Microprocessor SuperSet
  - Integrated Drive and Data Bus Buffers
- Available in 64 Pin QFP and 44 Pin QFP Package
  (See Package Specification Order Number 240800, Package Type S)

The 82078 Product Family brings a set of enhanced floppy disk controllers. These include several features that allow for easy implementation in both the portable and desktop market. The current family includes a 64 pin and a 44 pin part in the smaller form factor QFP package. The 3.3V version of the 64 pin part provides an ideal solution for the rapidly emerging 3.3V platforms. It also allows for a 5.0V tolerant floppy drive interface that lets the users retain their normal 5.0V drives. Another version of the 64 pin part provides support for 2 Mbps data rate tape drives.

*Other brands and names are the property of their respective owners.

February 1994
Order Number: 290468-003

Table 1-0. 64 Pin Part Versions

	3.3V	5.0V	2 Mbps Data Rate
82078SL	X	X	
82078-1		X	X

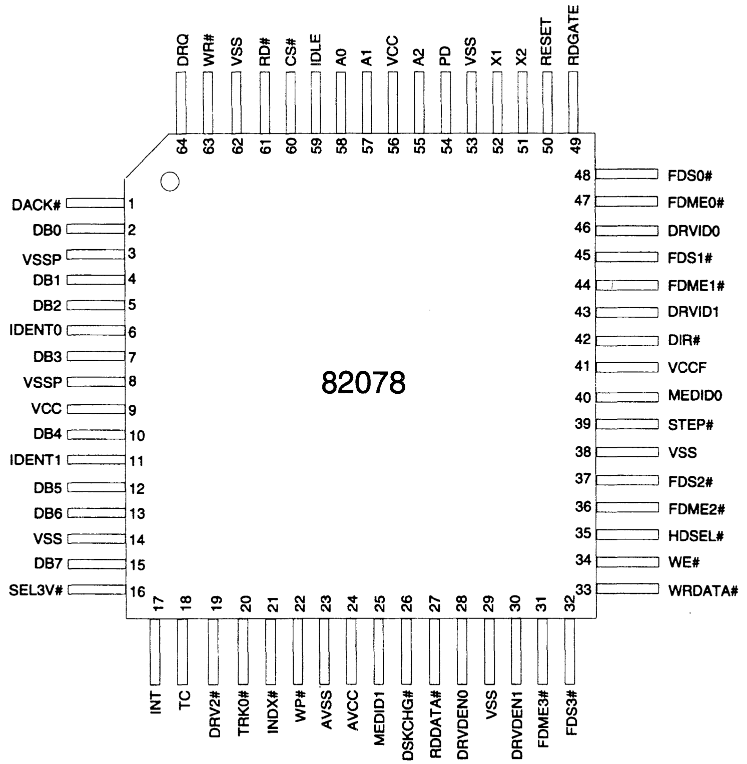
The 44 pin is targeted for platforms that are operated at 3.3V or 5.0V and do not require more than two drive support. The 82078-5 is designed for price sensitive 5.0V designs which do not include 4 MB drive support.

Table 2-0. 44 Pin Part Versions

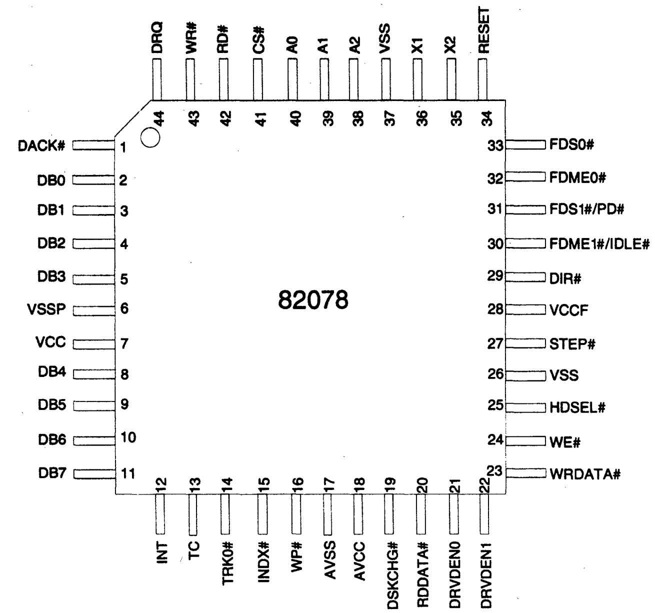
	3.3V	5.0V	1 Mbps Data Rate
82078		X	X
82078-5		X	
82078-3	X		X

Both parts can be operated at 1 Mbps/500 Kbps/300 Kbps/250 Kbps. Additionally, one version of the 64 pin part provides 2 Mbps data rate operation specific for the new tape drives.

The 82078 is fabricated with Intel's advanced CHMOS III technology.



290468-1


DRQ
WR#
RD#
CS#
A0
A1
A2
VSS
X1
X2
RESET
44
43
42
41
40
39
38
37
36
35
34
DACK# 1
DB0 2
DB1 3
DB2 4
DB3 5
VSSP 6
VCC 7
DB4 8
DB5 9
DB6 10
DB7 11
82078
33 FDS0#
32 FDME0#
31 FDS1#/PD#
30 FDME1#/IDLE#
29 DIR#
28 VCCF
27 STEP#
26 VSS
25 HDSEL#
24 WE#
23 WRDATA#
12
13
14
15
16
17
18
19
20
21
22
INT
TC
TRK0#
INDX#
WP#
AVSS
AVCC
DSKCHG#
RDDATA#
DRVDEN0
DRVDEN1

290468-2

82078 44 PIN
CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER

- **Small Footprint and Low Height Package**
- **Enhanced Power Management**
  - **—Application Software Transparency**
  - **—Programmable Powerdown Command**
  - **—Save and Restore Commands for Zero-Volt Powerdown**
  - **—Auto Powerdown and Wakeup Modes**
  - **—Two External Power Management Pins**
  - **—Consumes No Power While in Powerdown**
- **Integrated Analog Data Separator**
  - **—250 Kbps**
  - **—300 Kbps**
  - **—500 Kbps**
  - **—1 Mbps**
- **Programmable Internal Oscillator**
- **Floppy Drive Support Features**
  - **—Drive Specification Command**
  - **—Selectable Boot Drive**
  - **—Standard IBM and ISO Format Features**
  - **—Format with Write Command for High Performance in Mass Floppy Duplication**
- **Integrated Tape Drive Support**
  - **—Standard 1 Mbps/500 Kbps/ 250 Kbps Tape Drives**
- **Perpendicular Recording Support for 4 MB Drives**
- **Integrated Host/Disk Interface Drivers**
- **Fully Decoded Drive Select and Motor Signals**
- **Programmable Write Precompensation Delays**
- **Addresses 256 Tracks Directly, Supports Unlimited Tracks**
- **16 Byte FIFO**
- **Single-Chip Floppy Disk Controller Solution for Portables and Desktops**
  - **—100% PC/AT* Compatible**
  - **—Fully Compatible with Intel386™ SL**
  - **—Integrated Drive and Data Bus Buffers**
- **Separate 5.0V and 3.3V Versions of the 44 Pin part are Available**
- **Available in a 44 Pin QFP Package**

The 82078, a 24 MHz crystal, a resistor package, and a device chip select implements a complete solution. All programmable options default to 82078 compatible values. The dual PLL data separator has better performance than most board level/discrete PLL implementations. The FIFO allows better system performance in multi-master (e.g., Microchannel, EISA).

The 82078 maintains complete software compatibility with the 82077SL/82077AA/8272A floppy disk controllers. It contains programmable power management features while integrating all of the logic required for floppy disk control. The power management features are transparent to any application software.

The 82078 is fabricated with Intel's advanced CHMOS III technology and is also available in a 64-lead QFP package.

*Other brands and names are the property of their respective owners.

Refer to the 1995 Peripheral Components Handbook for the complete data sheet on this device.
The complete document for this product is available on Intel's "Data-on-Demand" CD-ROM product. Contact your local Intel field sales office, Intel technical distributor, or call 1-800-548-4725.

December 1994
Order Number: 290474-003

82078 64 PIN
CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER

- Small Footprint and Low Height Packages
- Supports Standard 5.0V as well as Low Voltage 3.3V Platforms
  - Selectable 3.3V and 5.0V Configuration
  - 5.0V Tolerant Drive Interface
- Enhanced Power Management
  - Application Software Transparency
  - Programmable Powerdown Command
  - Save and Restore Commands for Zero-Volt Powerdown
  - Auto Powerdown and Wakeup Modes
  - Two External Power Management Pins
  - Consumes no Power when in Powerdown
- Integrated Analog Data Separator
  - 250 Kbps
  - 300 Kbps
  - 500 Kbps
  - 1 Mbps
  - 2 Mbps
- Programmable Internal Oscillator
- Floppy Drive Support Features
  - Drive Specification Command
  - Media ID Capability Provides Media Recognition
  - Drive ID Capability Allows the User to Recognize the Type of Drive
  - Selectable Boot Drive
  - Standard IBM and ISO Format Features
  - Format with Write Command for High Performance in Mass Floppy Duplication
- Integrated Tape Drive Support
  - Standard 1 Mbps/500 Kbps/ 250 Kbps Tape Drives
  - New 2 Mbps Tape Drive Mode
- Perpendicular Recording Support for 4 MB Drives
- Integrated Host/Disk Interface Drivers
- Fully Decoded Drive Select and Motor Signals
- Programmable Write Precompensation Delays
- Addresses 256 Tracks Directly, Supports Unlimited Tracks
- 16 Byte FIFO
- Single-Chip Floppy Disk Controller Solution for Portables and Desktops
  - 100% PC AT* Compatible
  - 100% PS/2* Compatible
  - 100% PS/2 Model 30 Compatible
  - Fully Compatible with Intel386™ SL Microprocessor SuperSet
- Integrated Drive and Data Bus Buffers
- Available in 64 Pin QFP Package

The 82078, a 24 MHz crystal, a resistor package, and a device chip select implements a complete solution. All programmable options default to 82078 compatible values. The dual PLL data separator has better performance than most board level/discrete PLL implementations. The FIFO allows better system performance in multi-master (e.g., Microchannel, EISA).

The 82078 maintains complete software compatibility with the 82077SL/82077AA/8272A floppy disk controllers. It contains programmable power management features while integrating all of the logic required for floppy disk control. The power management features are transparent to any application software. There are two versions of 82078 floppy disk controllers, the 82078SL and 82078-1.

The 82078 is fabricated with Intel's advanced CHMOS III technology and is also available in a 44-lead QFP package.

*Other brands and names are the property of their respective owner.

Refer to the 1995 Peripheral Components Handbook for the complete data sheet on this device.
The complete document for this product is available on Intel's "Data-on-Demand" CD-ROM product. Contact your local Intel field sales office, Intel technical distributor, or call 1-800-548-4725.

November 1994
Order Number: 290475-004

82077SL
CHMOS SINGLE-CHIP FLOPPY DISK CONTROLLER

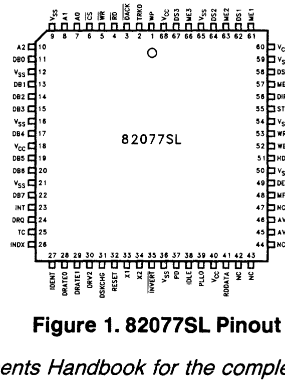
- **Completely Compatible with Industry Standard 82077AA**
- **Single-Chip Laptop Desktop Floppy Disk Controller Solution**
  - **—100% PC AT* Compatible**
  - **—100% PS/2* Compatible**
  - **—100% PS/2 Model 30 Compatible**
  - **—Fully Compatible with Intel's 386SL Microprocessor SuperSet**
  - **—Integrated Drive and Data Bus Buffers**
- **Power Management Features**
  - **—Application Software Transparency**
  - **—Programmable Powerdown Command**
  - **—Auto Powerdown and Wakeup Modes**
  - **—Two External Power Management Pins**
  - **—Typical Power Consumption in Power Down: 10 μA**
- **High Speed Processor Interface**
- **Integrated Analog Data Separator**
  - **—250 Kbits/sec**
  - **—300 Kbits/sec**
  - **—500 Kbits/sec**
  - **—1 Mbits/sec**
- **Programmable Crystal Oscillator for On or Off**
- **Integrated Tape Drive Support**
- **Perpendicular Recording Support**
- **12 mA Host Interface Drivers, 40 mA Disk Drivers**
- **Four Fully Decoded Drive Select and Motor Signals**
- **Programmable Write Precompensation Delays**
- **Addresses 256 Tracks Directly, Supports Unlimited Tracks**
- **16 Byte FIFO**
- **68-Pin PLCC**

  (See Packaging Handbook Order Number #240800, Package Type N)

The 82077SL, a 24 MHz crystal, a resistor package, and a device chip select implements a complete laptop solution. All programmable options default to 82077AA compatible values. The dual PLL data separator has better performance than most board level/discrete PLL implementations. The FIFO allows better system performance in multi-master systems (e.g., Microchannel, EISA).

The 82077SL is a superset of 82077AA. The 82077SL incorporates power management features while maintaining complete compatibility with the 82077AA/8272A floppy disk controllers. It contains programmable power management features while integrating all of the logic required for floppy disk control. The power management features are transparent to any application software. The 82077SL is available in three versions—82077SL-5, 82077SL and 82077SL-1. 82077SL-1 has all features listed in this data sheet. It supports both tape drives and 4 MB floppy drives. The 82077SL supports 4 MB floppy drives and is capable of operation at all data rates through 1 Mbps. The 82077SL-5 supports 500/300/250 Kbps data rates for high and low density floppy drives.

The 82077SL is fabricated with Intel's advanced CHMOS III technology and is available in a 68-lead PLCC (plastic) package.



290410–1

Figure 1. 82077SL Pinout

*PS/2 and PC AT are trademarks of IBM.

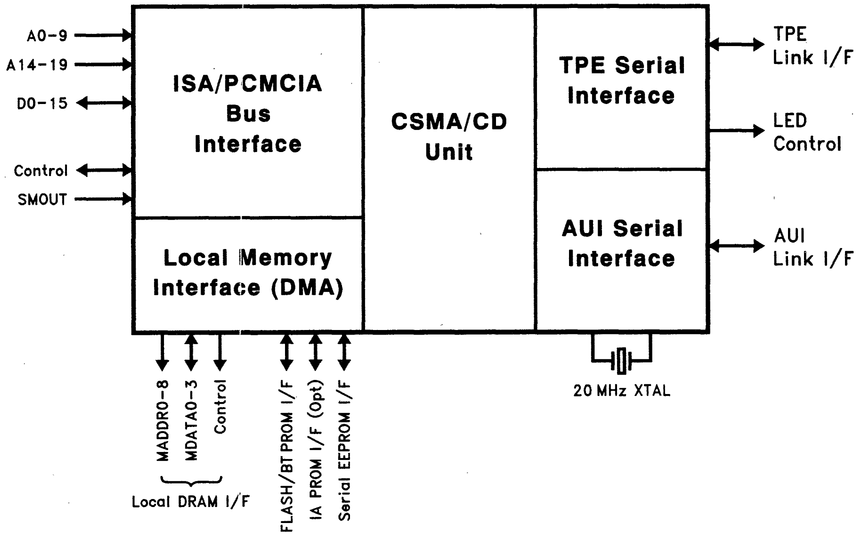
Refer to the 1995 Peripheral Components Handbook for the complete data sheet on this device.
The complete document for this product is available on Intel's "Data-on-Demand" CD-ROM product. Contact your local Intel field sales office, Intel technical distributor, or call 1-800-548-4725.

December 1994
Order Number: 290410-004

PRELIMINARY

82595 ISA/PCMCIA HIGH INTEGRATION ETHERNET CONTROLLER

- Optimal Integration for Lowest Cost Solution
  - — Glueless 8-Bit/16-Bit ISA/PCMCIA 2.0 Bus Interface
  - — Provides Fully 802.3 Compliant AUI and TPE Serial Interface
  - — Local DRAM Support up to 64 Kbytes
  - — FLASH/EPROM Boot Support
  - — Hardware and Software Portable between Motherboard, Adapter, and PCMCIA IO Card Solution
- High Performance Networking Functions
  - — 16-Bit IO Accesses to Local DRAM with Zero Added Wait-States
  - — Ring Buffer Structure for Continuous Frame Reception and Transmit Chaining
  - — Automatic Retransmission on Collision
  - — Automatically Corrects TPE Polarity Switching Problems
- Low Power CHMOS IV Technology
- Ease of Use
  - — Design Time Reduced by High Integration
  - — EEPROM Interface to Support Jumperless Design
  - — Software Structures Optimized to Reduce Processing Steps
  - — Automatically Maps into Unused PC IO Location to Help Eliminate LAN Setup Problems
  - — All Software Structures Contained in One 16-Byte IO Space
  - — Automatic or Manual Switching between TPE and AUI Ports
  - — JTAG Port for Reduced Board Testing Times
- Power Management
  - — SL Compatible $\overline{\text{SMOUT}}$ Power Down Input
  - — Software Power Down Command for non-SL Systems
- 144-Lead tQFP Package Provides Smallest Available Form Factor
  (See Packaging Spec., Order No. 240800)



290458-1

Figure 1. 82595 Block Diagram

For the complete data sheet on this product, refer to the 1995 Networking handbook.

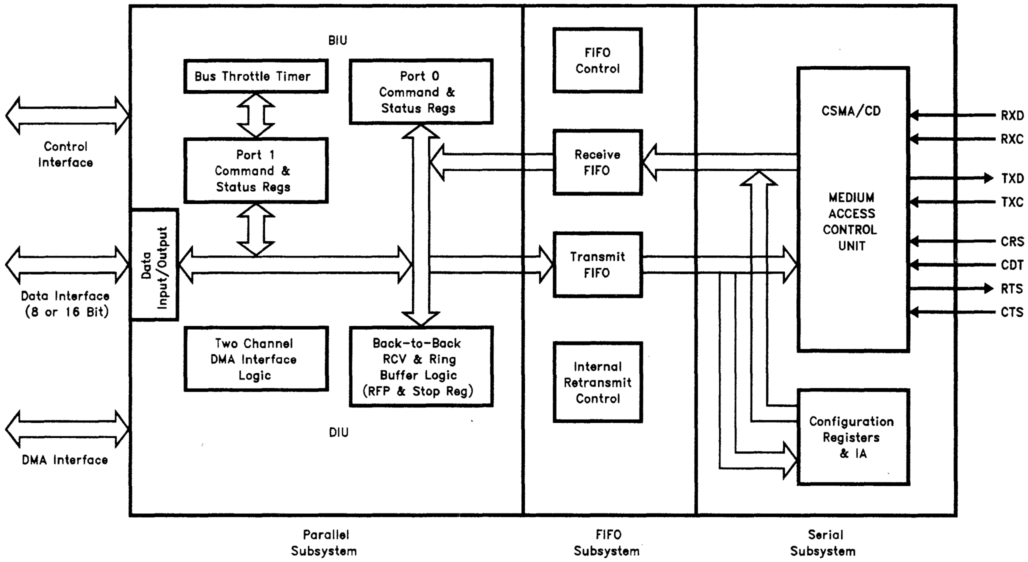
February 1994
Order Number: 290458-003

intel®

82593
CSMA/CD CORE LAN CONTROLLER

- Supports Industry Standard LANs
  - —IEEE 10BASE5 (Ethernet*)
  - —IEEE 10BASE2 (Cheapernet)
  - —IEEE 10BASE-T (TPE)
- Simple, High-Performance Control and Data Interface
  - —Control and Status via $\overline{\text{RD}}$, $\overline{\text{WR}}$, and $\overline{\text{CS}}$ Lines
  - —Data Transfers via DMA Interface
  - —Two Clocks per DMA Transfer
  - —Programmable Bus Throttle Timer
- High-Performance Networking Functions
  - —Automatic Retransmission from Internal FIFO
  - —Back-to-Back Frame Reception with No CPU Intervention
  - —Receive Ring Buffer Memory Structure
  - —Transmit Frame Chaining
  - —96-Byte Transmit FIFO and 96-Byte Receive FIFO
- High Speed, 5V CHMOS IV (P648.8) Technology
- Serial Bit Rates up to 20 Mb/s (82593SX)
  - —Direct Interface to Intel 82C501AD ESI or AMD 7992 SIA
  - —Conforms to 802.3 CSMA/CD Standard
- On-Board Diagnostics
  - —Internal and External Loopback Operation
  - —Internal Register Dump
  - —TDR Functionality
- 44-Lead PLCC Package Type N (82593SX), 44-Lead QFP Package Type S (82593SX) or 28-Pin PDIP
  - —82593SX (8/16-Bit) System Clock up to 20 MHz
  - —82593SX Package Pin Compatible with Intel 82592 PLCC

(See Packaging Spec., Order No. 240800-001 Package Type N, S and P)


BIU
Bus Throttle Timer
Port 0 Command & Status Regs
Port 1 Command & Status Regs
Control Interface
Data Input/Output
Data Interface (8 or 16 Bit)
Two Channel DMA Interface Logic
Back-to-Back RCV & Ring Buffer Logic (RFP & Stop Reg)
DIU
DMA Interface
FIFO Control
Receive FIFO
Transmit FIFO
Internal Retransmit Control
CSMA/CD
MEDIUM ACCESS CONTROL UNIT
Configuration Registers & IA
RXD
RXC
TXD
TXC
CRS
CDT
RTS
CTS
Parallel Subsystem
FIFO Subsystem
Serial Subsystem

290411-1

Figure 1. 82593 Block Diagram

*Ethernet is a registered trademark of Xerox Corporation.

For the complete data sheet on this product, refer to the 1995 Networking handbook.

82503
DUAL SERIAL TRANSCEIVER (DST)

82503 PRODUCT FEATURE SET OVERVIEW

- **Single Component Ethernet* Interface to Both 802.3 10BASE-T and AUI**
- **Automatic or Manual Port Selection**
- **Manchester Encoder/Decoder and Clock Recovery**
- **No Glue Interface to Industry-Standard LAN Controllers**
  - **— Intel 82586, 82590, 82593 and 82596**
  - **— AMD 7990 (LANCE*)**
  - **— National Semiconductor 8390 and 83932 (SONIC*)**
  - **— Western Digital 83C690**
  - **— Fujitsu 86950 (Etherstar*)**
- **Diagnostic Loopback**
- **Reset, Low Power Modes**
- **Network Status Indicators**
- **Defeatable Jabber Timer**
- **User Test Modes**
- **10 MHz Transmit Clock Generator**
- **One Micron CHMOS** IV (Px48) Technology**
- **Single 5-V Supply**

INTERFACE FEATURES

TPE

- **Complies with 10BASE-T, IEEE Std. 802.3i-1990 for Twisted Pair Ethernet**
- **Selectable Polarity Switching**
- **Direct Interface to TPE Analog Filters**
- **On-Chip TPE Squelch**
- **Defeatable Link Integrity (LI)**
- **Support of Cable Lengths > 100m**

AUI

- **Complies with IEEE 802.3 AUI Standard**
- **Direct Interface to AUI Transformers**
- **On-Chip AUI Squelch**

A block diagram of a typical application is shown in Figure 1. The 82503 Dual Serial Transceiver is a high-integration CMOS device designed to simplify interfacing industry standard Ethernet LAN Controllers to IEEE 802.3 local area network applications (10BASE5, 10BASE2, and 10BASE-T). The component supports both an attachment unit interface (AUI) and a Twisted Pair Ethernet interface (TPE). It allows OEMs to design a state-of-the-art media interface that is jumperless and fully automatic. The 82503 includes on-chip AUI and TPE drivers and receivers; it offers designers a cost-effective, integrated solution for interfacing LAN controllers to the wire medium.

**CHMOS is a patented process of Intel Corporation.
*Ethernet is a registered trademark of Xerox Corporation.
LANCE is a registered trademark of Advanced Micro Devices.
Etherstar is a registered trademark of Fujitsu Electronics.
Sonic is a registered trademark of National Semiconductor Corporation.

For the complete data sheet on this product, refer to the 1995 Networking handbook.

November 1992
Order Number: 290421-003

Ethernet* LAN Card Product Brief

Product Highlights

- Complete plug and play PCMCIA LAN solution. Comes with all drivers, installation, card and socket services and card management software necessary for reliable operation in a PCMCIA slot.
- Drivers for all major network operating systems.
- Industry-recognized Intel SoftSet installation software. Easy to operate and manage.
- Based on highly integrated Intel 82595 Ethernet Controller
- Complies with PCMCIA 2.0/ JEIDA 4.1 68-pin standard.
- 5 mm-thick PCMCIA Type II card.
- Detachable line adapter module (LAM) for multiple media attachment.
- Ethernet IEEE 802.3 compatibility (10BASE-T/TPE, 10BASE- 2/BNC).
- Activity and link integrity LEDs.

The Intel PCMCIA Ethernet LAN Card brings high performance and ease of use to PCMCIA networking. It lets you put networking capabilities into laptop computers without a lot of hassle, headache or expense.

It's a plug and play solution, providing PCMCIA network- readiness right out of the box. All the software you need comes with the card: drivers for the most popular network operating systems (Novell NetWare 2.2, 3.11, 4.0 and Lite 1.0; Microsoft* LAN Manager* 2.X; IBM* LAN Server* 2.X; Banyan Vines* 4.x; and Microsoft Windows for Workgroup* 3.1), PCMCIA- compliant card and socket services from SystemSoft*, and Intel's own card manager and Softset auto-configuration, auto-installation software that gets users on the network fast.*

The Intel PCMCIA Ethernet LAN Card is based on the highly integrated Intel 82595 ISA/PCMCIA Ethernet Controller, giving you 16-bit desktop performance in a PCMCIA form factor. As we develop additional software functionality for our 82595 line, you'll be able to leverage it across your entire 82595-based product line, from chips to NICs to PCMCIA form factor products. You decrease your time to market by taking advantage of Intel's product development efforts, while increasing the value of your products.

The card is standard PCMCIA 2.0 68-pin form factor and only 5 mm thick. It's passed Intel's extensive quality and reliability testing to ensure that it stands up to the rigors of mobile users on the go. These include PCMCIA mechanical qualification testing such as torque, bend, shock, vibration and environmental testing across extreme temperatures and voltages. Our CMOS technology and highly integrated 82595 are very power efficient, letting mobile users stay connected to the network for a long time. Maximum power draw is 85 mA; in idle mode, power consumption drops to 20 mA.

Finally, we've made it easy to customize our card and its accessories to your own OEM marketing needs. Manuals, software diskettes and the cards themselves can all be tailored to reflect your company's look.

Product Description

The Intel PCMCIA Ethernet LAN Card is the fastest, easiest way to deliver network-ready laptops to your customers. It snaps in and installs in minutes, not hours, thanks to a disk-full of software to make your job easier. Our industry-standard SoftSet installation utility automatically configures the card and sets up the software with a single command. Built-in card and socket services software provides card recognition and compatibility. Built-in card management software performs IRQ management and allows the card to be installed even when the system is running, so you don't have to reboot. There's even built-in driver support for popular network operating systems from Novell, Microsoft, IBM and Banyan. There are no jumpers or switches to set, no IRQ addresses to labor over. The card's installed and working in five minutes. Once installed, it's easy to operate too, improving customer satisfaction and decreasing the number of support calls you receive.

The card is fully PCMCIA 2.0/JEIDA 4.1 compliant with a standard 68-pin form factor. It's also fully compliant with Ethernet IEEE 802.3 standard for 10BASE-T and 10BASE-2 wiring.

There's a detachable line adapter module (LAM) for attachment to multiple media and LEDs to indicate active and link integrity.

Features	Benefits
— Installation, card manager, and card and socket services software	— A complete solution; no need for any other pieces Easy to install, easy to configure, easy to use
— Broad client driver support	— Meets broad target market. Usable on industry-standard networks like Novell, LAN Manager and Banyan
— Complies with PCMCIA 2.0/JEIDA 4.1 standards	— No jumpers Easy to transport Small form factor
— Intel-based 82595	— Great performance; 8 & 16-bit data path Glueless interface to PCMCIA Bus
— Activity and link integrity LEDs	— Indicates card status, improves diagnostics
— Supports both twisted pair Ethernet (10BASE-T) and ThinCoax (BNC 10BASE-2)	— Flexibility to connect to multiple network media

Product Codes

MBLA8110	TPE all geographies
MBLA8120US	BNC US
MBLA8120EU	BNC Europe

Additional Literature

Local Area Networking Family Product Brief	297085-002
EtherExpress Family Brochure	D413.01
TokenExpress Family Brochure	D414.02
Intel 82595 Data Sheet	290458-003
Intel 82595 User's Manual	TBD

* Other brands and names are the property of their respective owners.

Order Number: 297120-003

DataFax 14.4 Card Product Brief

Product Highlights

- LAM-less design (Integrated DAA)
- Complies with PCMCIA 2.0 and JEIDA 4.1 standards
- PCMCIA Type II card — 5.0 mm thick
- Automatic power down mode
- DTMF AND PULSE dialing
- Fully compliant with CCITT T.30 and T.4 (Group 3 fax)
- Compliant with CCITT V.17 (14.4 Kbps fax)
- Provides both Send and Receive fax capability
- 8-bit I/O bus interface
- Includes Ring Detect notification to host computer in the power down mode
- FCC Class B, UL and DOC/UL (Canada)
- Compatible with EIA/TIA 602 (AT command set)
- Data modem complies with CCITT V.32bis, V.32, V.22bis, V.22, V.21, Bell* 212a, 103
- Supports CCITT V.42 error detection and V.42bis data compression
- Provides MNP/5* data compression for backward compatibility
- Requires no external power
- MNP/10
- Hayes AutoSync

With the design of the DataFax 14.4, Intel introduces a high speed fax card with no external line adapter module. Lightweight. Easy to carry. And less chance of damage or loss. Not only does the DataFax 14.4 feature integrated DAA, but its high speed transmission helps your customers reduce the telephone expenses associated with faxes and modems.

The DataFax 14.4 fax card is also Group 3 compliant, assuring worldwide compatibility with most fax machines operating today. It sends and receives faxes and transfers files to or from notebook or hand-held computers over the public telephone network.

The Intel DataFax 14.4 also conforms to PCMCIA 2.0 AND JEIDA 4.1 physical and electrical standards for portable computers. The size of a credit card, the DataFax 14.4 slides into an external slot in the notebook computer, and connects easily into the public switched telephone network. The card features four power management modes — on-line, active, power save, and power down — to ensure the lowest possible power consumption. No external power is required.

Product Description

The DataFax 14.4 combines high speed with an integrated DAA design to provide optimal connectivity for notebooks and sub-notebooks. PCMCIA cards enhance the multiple functionality of the notebook computer. Intel's integrated DAA design incorporates the telephone interface circuitry directly onto the card. This includes a ring detector and telephone line coupling transformer. A six-foot cable connects the DataFax 14.4 card to a standard RJ-11 modular telephone jack for connectivity to the telephone network.

The DataFax 14.4 supports V.42 error correction which ensures that errors caused by the phone system are automatically corrected. The DataFax 14.4 increases data throughput with V.42bis data compression. This detects redundant characters, character sequences, and uses fewer bits to send more frequently occurring sequences. DataFax 14.4 users enjoy an effective throughput of up to 57,600 bits per second.

Unlike other card manufacturers, Intel provides a full-service program for card labeling, custom kitting with third party applications, and fulfillment. The Intel DataFax 14.4 is the only fax card designed with recessed covers in order to accept adhesive labels, front and back, insuring quick turnaround for custom orders.

Features	Benefits
— Integrated DAA	— No external circuitry — Compact and lightweight — Easy to carry — Fits in briefcase with notebook computer
— Exchangeable with other cards	— Single slot serves multiple functions
— CCITT V.32bis (14.4 Kbps)	— Ensures connectivity worldwide — Faster transmission means less costly phone bills
— Supports V.42/V.42bis	— Provides high throughput (57.6 Kbps)
— MNP/10	— Enhanced data throughput with cellular connections (required phone adapter)
— Hayes AutoSync	— Provides synchronous communications capability (software required)

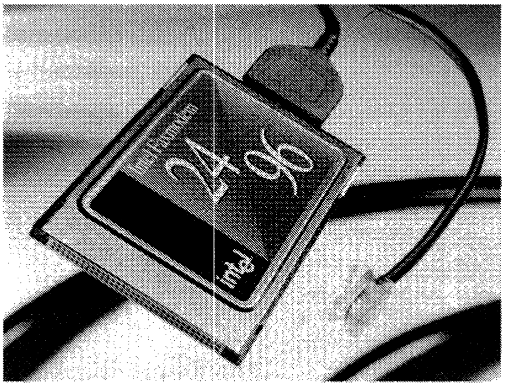
*Other brands and names are the property of their respective owners.

Order Number: 297311-002

Faxmodem 24/96 Card Product Brief

Product Highlights

- Integrated DAA on the card for U.S./Canada
- Complies with PCMCIA 2.0 and JEIDA 4.1 standards
- PCMCIA Type II card
  - 5mm thick
- Fully compliant with CCITT T.30 and T.4 (group 3 fax)
- Provides both Send and Receive fax capability
- Fully compliant with EIA/TIA 578 (fax class 1 command set)
- Data modem complies with CCITT V.22bis, V.22, V.21, Bell* 212a and 103
- Supports CCITT V.42 error detection and V.42bis data compression
- Provides MNP5 data compression for backward compatibility
- Includes Ring Detect notification to host computer in the power down mode
- Multiple power conservation modes
- Requires no external power



The Faxmodem 24/96 enables you to send or receive faxes and transfer files to or from notebook computers over the public telephone network. It is the latest member of Intel's I/O card "plug and play" family.

The Intel Faxmodem 24/96 also conforms to PCMCIA 2.0 AND JEIDA 4.1 physical and electrical standards for portable computers. The card's light weight and low power consumption combine to provide high-performance, low-cost connectivity for notebooks and sub-notebooks.

Approximately the size of a credit card, the Faxmodem 24/96 slides into an external slot in the notebook computer, connecting easily into the public switched telephone network. The card features four power management modes — on-line, active, power save, and power down — to ensure the lowest possible power consumption. No external power is required.

Product Description

The Faxmodem 24/96 card provides convenient, high-performance mobile fax and data communications capability for the notebook computer user. Adherence to accepted national and international standards provides the user with worldwide connectivity for both fax (approximately 30 million machines in use worldwide) and data transfer (V.22bis is the most widely used communications standard in the world). Its exchangeability with other PCMCIA cards enhances the multiple functionality of the notebook computer.

The Faxmodem 24/96 card contains the Intel 89C124FX integrated data-fax modem chipset, a UART, a microcontroller, an analog front-end and other supporting devices. The 16450-type UART provides an 8-bit data bus interface. This, together with the Class 1 AT command set support, provides compatibility with all the major fax application software.

For U.S. and Canada, the DAA is integrated on the card. For rest of world, the detachable line adapter module incorporates country-specific telephone interface circuitry. This consists of the ring detector and telephone line coupling transformer. A standard RJ-11 modular telephone jack provides connectivity to the telephone network. A short cable connects the line adapter module to the Faxmodem 24/96 card.

The Faxmodem 24/96 supports V.42 error correction which ensures that errors caused by the phone system are automatically corrected. The Faxmodem 24/96 also supports V.42bis data compression. This increases data throughput by detecting redundant characters, character sequences, and using fewer bits to send more frequently occurring sequences. Throughput for file transfer operations is increased by as much as 400 percent, providing the user with an effective 9600 bits per second.

Features	Benefits
— Integrated DAA	— No external circuitry — Compact and lightweight — Easy to carry — Fits in briefcase with notebook computer
— Exchangeable with other cards	— Single slot serves multiple functions
— Group 3 fax	— Compatible with worldwide installed base of fax machines
— Class 1 fax command set	— Compatible with standard communications packages
— CCITT V.22bis, V.22, V.21, Bell 212 and 103	— Ensures connectivity worldwide
— Supports V.42/V.42bis	— Faster; up to 4 to 1 data compression providing an effective throughput of 9600 bits per second for file transfers
— Modem supports AT command set	— Compatible with standard communications packages
— Multiple power conservation modes	— Prolong system battery life
— Factory Configuration Option for Cellular Network	— Ease of use

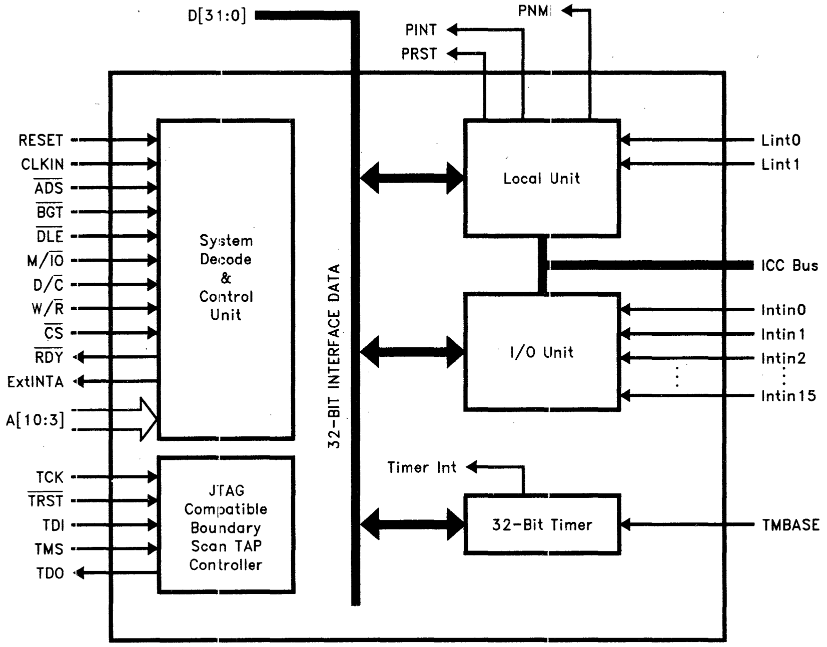
Order Number: 297390-001

82489DX
ADVANCED PROGRAMMABLE INTERRUPT CONTROLLER

82489DX FEATURES OVERVIEW

- **Advanced Interrupt Controller for 32-Bit Operating Systems**
- **Solution for Multiprocessor Interrupt Management**
- **Dynamic Interrupt Distribution for Load Balancing in MP Systems**
- **Separate Nibble Bus (Interrupt Controller Communications (ICC) Bus) for Interrupt Messages**
- **Inter-Processor Interrupts**
- **Various Addressing Schemes— Broadcast, Fixed, Lowest Priority, etc.**
- **Compatibility Mode with 8259A**
- **32-Bit Internal Registers**
- **Integrated Timer Support**
- **33 MHz Operation**
- **132-Lead PQFP Package, Package Type KU**
(See Packaging Specification. Order Number: 240800)

82489DX Block Diagram



290446–1

Refer to Application Note AP-388: 82489DX User's Manual (Order Number 292116) when evaluating your design needs.

82489DX
Advanced Programmable Interrupt Controller

CONTENTS

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

1.0 INTRODUCTION

The 82489DX Advanced Programmable Interrupt Controller provides multiprocessor interrupt management, providing both static and dynamic symmetrical Interrupt distribution across all processors.

The main function of the 82489DX is to provide interrupt management across all processors. This dynamic interrupt distribution includes routing of the interrupt to the lowest-priority processor. The 82489DX works in systems with multiple I/O subsystems, where each subsystem can have its own set of interrupts. This chip also provides inter-processor interrupts, allowing any processor to interrupt any processor or set of processors. Each 82489DX I/O unit Interrupt Input pin is individually programmable by software as either edge or level triggered. The interrupt vector and interrupt steering information can be specified per pin. A 32-bit wide timer is provided that can be programmed to interrupt the local processor. The timer can be used as a counter to provide a time base to software running on the processor, or to generate time slice interrupts locally to that processor. The 82489DX provides 32-bit software access to its internal registers. Since no 82489DX register reads have any side effects, the 82489DX registers can be aliased to a user read-only page for fast user access (e.g., performance monitoring timers).

The 82489DX supports a generalized naming/addressing scheme that can be tailored by software to fit a variety of system architectures and usage models. It also supports 8259A compatibility by becoming virtually transparent with regard to an externally connected 8259A style controller, making the 8259A visible to software.

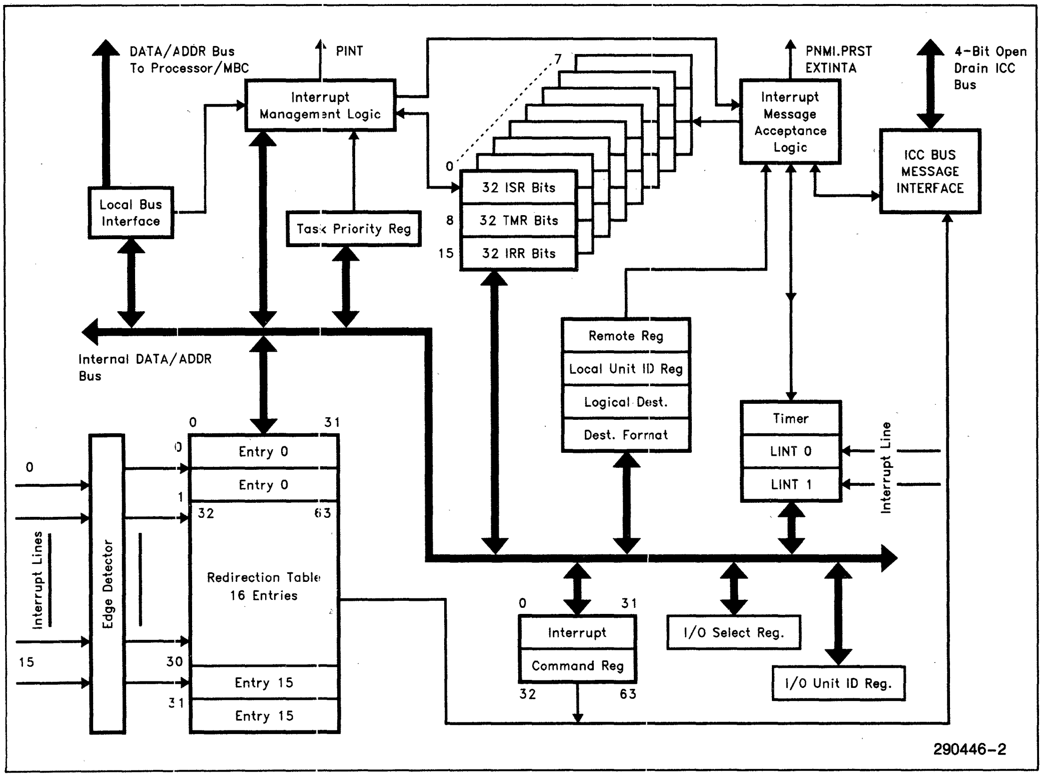


Figure 1. 82489DX Architecture

2.0 FUNCTIONAL OVERVIEW

82489DX Functional Blocks

82489DX contains one Local Unit, one I/O unit and a timer. The ICC bus is used to pass interrupt messages.

ICC BUS

The ICC bus is a 5-wire synchronous bus connecting all 82489DXs (all I/O Untis and all Local Units). The Local Units and I/O Units communicate over this ICC bus. Four of these five wires are used for data transmissions and arbitration, and one wire is a clock.

LOCAL UNIT

The Local Unit contains the necessary intelligence to determine whether or not its processor should accept interrupt messages sent on the ICC bus by other Local Units and I/O Units. The Local Unit also provides local pending of interrupts, nesting and masking of interrupts, and handles all interactions with its local processor such as the INT/INTA/EOI protocol. The Local Unit further provides inter-processor interrupt functionality and a timer to its local processor. The interface of a processor to its 82489DX Local Unit is identical for every processor.

I/O UNIT

The I/O Unit provides the interrupt input pins on which I/O devices inject interrupts into the system in the form of an edge or a level. The I/O unit also contains a Redirection Table for the interrupt input pins. Each entry in the Redirection Table can be individually programmed to indicate whether an interrupt on the pin is recognized as either an edge or a level; what vector and also what priority the interrupt has; and which of all possible processors should service the interrupt and how to select that processor (statically or dynamically). The information in the table is used to send interrupt messages to all 82489DX Units via the ICC bus.

TIMER

The 82489DX provides a 32-bit wide timer that can be programmed to interrupt the local processor. The timer can be used as a counter to provide a timebase to software running on the processor, or to generate time-slice interrupts local to that processor.

3.0 PIN DESCRIPTION

The 82489DX pin description is organized in a small number of functional groups. Pin definitions and protocols have been designed to minimize interface issues. In particular, they support the notion of independently controlled address and data phases. The primary host interface is synchronous in nature.

In the following pin definition table if the signal name has (__) over it, the signal is in its active state when it has a low level. The signal direction column identifies output only signals as a continuous drive (O), tristate (T/S), or open drain (O/D). All bi-directional (BI-D) signals have tri-stating outputs.

Pin Definition Table

Symbol	Pin No.	Type	Function
SYSTEM PINS			
RESET	65	I	The **RESET INPUT** forces 82489DX to enter its initial state. The 82489DX Local Unit in turn asserts it PRST (Processor Reset) output. All tri-state outputs remain in high impedance until explicitly enabled.
ExtINTA	41	O	The **EXTERNAL INTERRUPT ACKNOWLEDGE** output is asserted (high) when an external interrupt controller (e.g., 8259) is expected to respond to the current INTA cycle. If deasserted (low), 82489DX will respond, and the INTA cycle must not be delivered to the external controller.
CLKIN	57	I	**CLOCK INPUT** provides reference timing for most of the bus signals.
$\overline{\text{TRST}}$	56	I	**TEST RESET** is the JTAG compatible boundary scan TAP controller reset pin. A weak pull-up keeps the pin high if not driven.
TCK	55	I	**TEST CLOCK** is the clock input for the JTAG compatible boundary scan controller and latches.
TDI	53	I	**TEST DATA INPUT** is the test data input pin for the JTAG compatible boundary scan chain and TAP controller. A weak pull-up keeps this pin high if not driven.
TDO	52	O	**TEST DATA OUTPUT** is the test data output for the JTAG compatible boundary scan chain.
TMS	54	I	**TEST MODE SELECT** is the test mode select pin for the JTAG boundary scan TAP controller. A weak pull-up keeps this pin high if not driven.
TIMER PIN			
TMBASE	59	I	The **TIME BASE** input provides a standard frequency that is only used by the 82489DX timer and that is independent of the system clock.
INTERRUPT PINS			
INTIN[15:0]	82–97	I	These 16 **INTERRUPT INPUT** pins accept edge or level sensitive interrupt requests from I/O or other devices. The pin numbers are specified respectively. INTIN15 corresponds to pin number 82, INTIN14 corresponds to pin number 83 etc., and INTIN0 corresponds to pin number 97. These pins are active high.
LINTIN[1] LINTIN[0]	80 81	I I	Two **LOCAL INTERRUPT INPUT** pins accept edge or level sensitive interrupt requests that can only be delivered to the connected processor. These pins are active high.
REGISTER ACCESS PINS			
$\overline{\text{ADS}}$	64	I	**ADDRESS STROBE** signal indicating the start of a bus cycle. 82489DX does not commit to start the cycle internally until BUS GRANT is detected active.

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
REGISTER ACCESS PINS (Continued)			
$M/\overline{IO}$, $D/\overline{C}$, $W/\overline{R}$	63 61 62	I I I	Bus cycle definition signals. Note that since the 82489DX registers can be mapped in either memory or I/O space, the $M/\overline{IO}$ pin is not used for register access cycles; it is only used to decode interrupt acknowledge cycles. 82489DX does not respond to code read cycles.
$\overline{BGT}$	66	I	The **BUS GRANT** input is optional and is used to indicate the address phase of a bus cycle in configurations where address timing cannot be inferred from $\overline{ADS}$. This signal is really used as an address latch enable, but is named as it is to indicate that it can normally be connected to the Intel Cache Controller generated signal of the same name. Must be tied low if not used.
$\overline{CS}$	74	I	The **CHIP SELECT** input indicates that the 82489DX registers are being addressed.
A3 A4 A5 A6 A7 A8 A9 A10	31 29 28 27 26 24 22 21	BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D	The address pins are used as inputs in addressing internal register space. Output function is reserved. They are also used to latch local unit ID on reset.
$\overline{DLE}$	73	I	**DATA LATCH/ENABLE** is optional and is used to indicate committing the data phase of a bus cycle in configurations where data timing cannot be inferred from other cycle timings. Must be tied low if not used.
D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12 D11	105 107 109 110 111 112 114 115 116 118 119 121 122 123 124 125 128 129 130 131 2	BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D	The DATA BUS is for all register accesses and interrupt vectoring.

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
REGISTER ACCESS PINS (Continued)			
D10	3	BI-D	
D9	4	BI-D	
D8	7	BI-D	
D7	8	BI-D	
D6	9	BI-D	
D5	11	BI-D	
D4	12	BI-D	
D3	13	BI-D	
D2	14	BI-D	
D1	16	BI-D	
D0	18	BI-D	
DP3	101	BI-D	One Data Parity pin for each byte on the data bus. EVEN parity is generated any time the data bus is driven by the 82489DX.
DP2	102	BI-D	
DP1	103	BI-D	
DP0	104	BI-D	
$\overline{RDY}$	43	O	**READY** output indicates that the current bus cycle is complete. In the case of a read cycle, valid data and the return to inactive state after going active low may be delayed till $\overline{DLE}$ goes active.
PROCESSOR PINS			
PINT	35	T/S	The **PROCESSOR INTERRUPT OUTPUT** indicates to the processor that one or more maskable interrupts are pending. This pin is tri-stated at reset, and has an internal pull-down resistor to prevent false signaling to the processor until the 82489DX Local Unit is enabled and this pin is actively driven.
PRST	38	O	The **PROCESSOR RESET OUTPUT** is asserted/de-asserted upon 82489DX reset, and also in response to ICC bus messages with "RESET" delivery mode. This pin should be used with care.
PNMI	37	T/S	The **NON-MASKABLE INTERRUPT** output is signaled in respone to ICC bus messages with "NMI" delivery mode. This pin is tri-stated at reset, and has an internal pull-down resistor to prevent false signaling to the processor until the Local Unit is enabled and this pin is actively driven.
ICC BUS PINS			
ICLK	60	I	The **ICC BUS CLOCK** input provides synchronous operation of the ICC bus.
MBI[3:0]	76–79	I	The four **ICC BUS IN** inputs are used for incoming ICC bus messages. In smaller configurations the ICC bus input and outputs may be tied directly together at the pins. Pin number for MBI3 is 76, MBI2 is 77, MBI1 is 78 and MBI0 is 79.
MBO3	45	O/D	The four **ICC BUS OUT** outputs are used for outgoing ICC bus messages. The current capacity is only 4 mA. So external bufferes will be needed.
MBO2	48		
MBO1	49		
MBO0	51		

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
RESERVED PINS			
Reserved	34, 42	NC	These pins **MUST BE LEFT OPEN.**
Reserved	70, 72, 75		**Reserved** by Intel. **These pins should be strapped to V_{CC}.**
Reserved	71, 19, 20		**Reserved** by Intel. **These pins should be strapped to GND.**
POWER AND GROUND PINS			
V_{CC}	1, 32, 69, 98	POWER	Nominally +5V. These pins along with V_{SS} and V_{SSI} should be separately bypassed.
V_{CCP}	6, 15, 25,100, 108, 117, 126	POWER	Nominally +5V. These pins along with V_{SSP} should be separately bypassed.
V_{CCPO}	39, 46	POWER	Nominally +5V. These pins along with V_{SSPO} should be separately bypassed.
V_{SS}	5, 33, 67, 68, 99	GND	Nominally 0V. These pins along with V_{CC} should be separately bypassed.
V_{SSP}	10, 17, 23, 30, 106, 113, 120, 127, 132,	GND	Nominally 0V. These pins along with V_{CCP} should be separately bypassed.
V_{SSPO}	36, 40, 44, 47, 50	GND	Nominally 0V. These pins along with V_{CCPO} should be separately bypassed.
V_{SSI}	58	GND	Nominally 0V. These pins along with V_{CC} should be separately bypassed.

NOTE:
V_{CC}, V_{CCP} and V_{CCPO} should be of same voltage. V_{SS}, V_{SSP}, V_{SSPO} and V_{SSI} should be 0V.

4.0 FUNCTIONAL DESCRIPTION

As far as interrupt management is concerned, the 82489DX's interrupt control function spans over two functional units, the I/O Unit of which there is one per I/O subsystem, and the Local Unit of which there is one per processor. 82489DX has one I/O unit and one Local Unit in a single package. This section takes a detailed look at both local and I/O Units.

I/O Unit

The I/O Unit consists of a set of Interrupt Input pins, an Interrupt Redirection Table, and a message unit for sending and receiving messages from the ICC bus. The I/O Unit is where I/O devices inject their interrupts, the I/O Unit selects the corresponding entry in the Redirection Table and uses the information in that entry to format an interrupt request message. The message unit then broadcasts this message over the ICC bus. The content of the Redirection Table is under software control and is assigned benign defaults upon reset. The masks in the Redirectional Table entries are set to 1 at *hardware reset* to disable the interrupts.

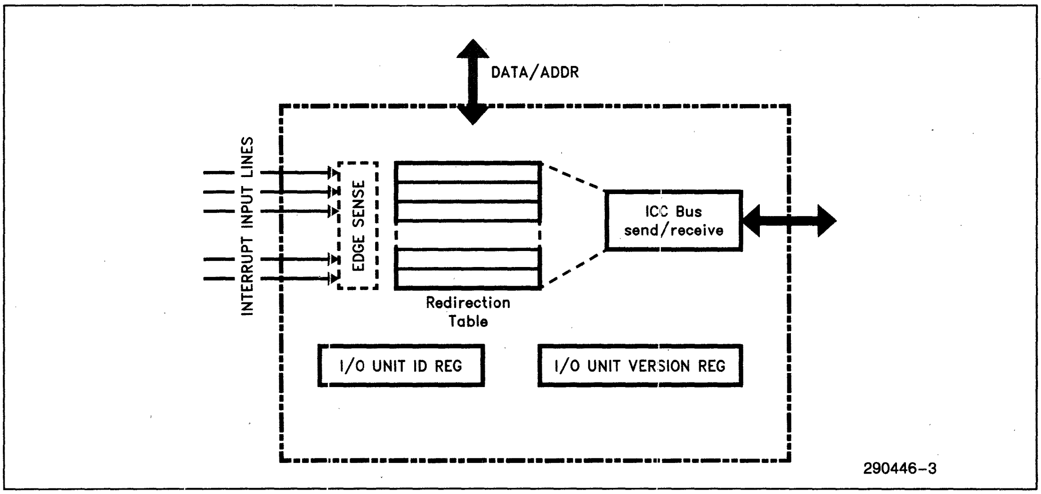


Figure 2. 82489DX I/O Unit Block Diagram

Local Unit

Interrupt Management of the Local Unit is responsible for local interrupt sources, interrupt acceptance, dispensing interrupts to the processor, and sending inter-processor interrupts. Depending on the delivery mode of the interrupt, zero, one or more units can accept an interrupt. A Local Unit accepts an interrupt only if it will deliver the interrupt to its processor. Accepting an interrupt is purely an inter-82489DX matter; dispensing an interrupt to the local processor only involves a 82489DX and its local processor.

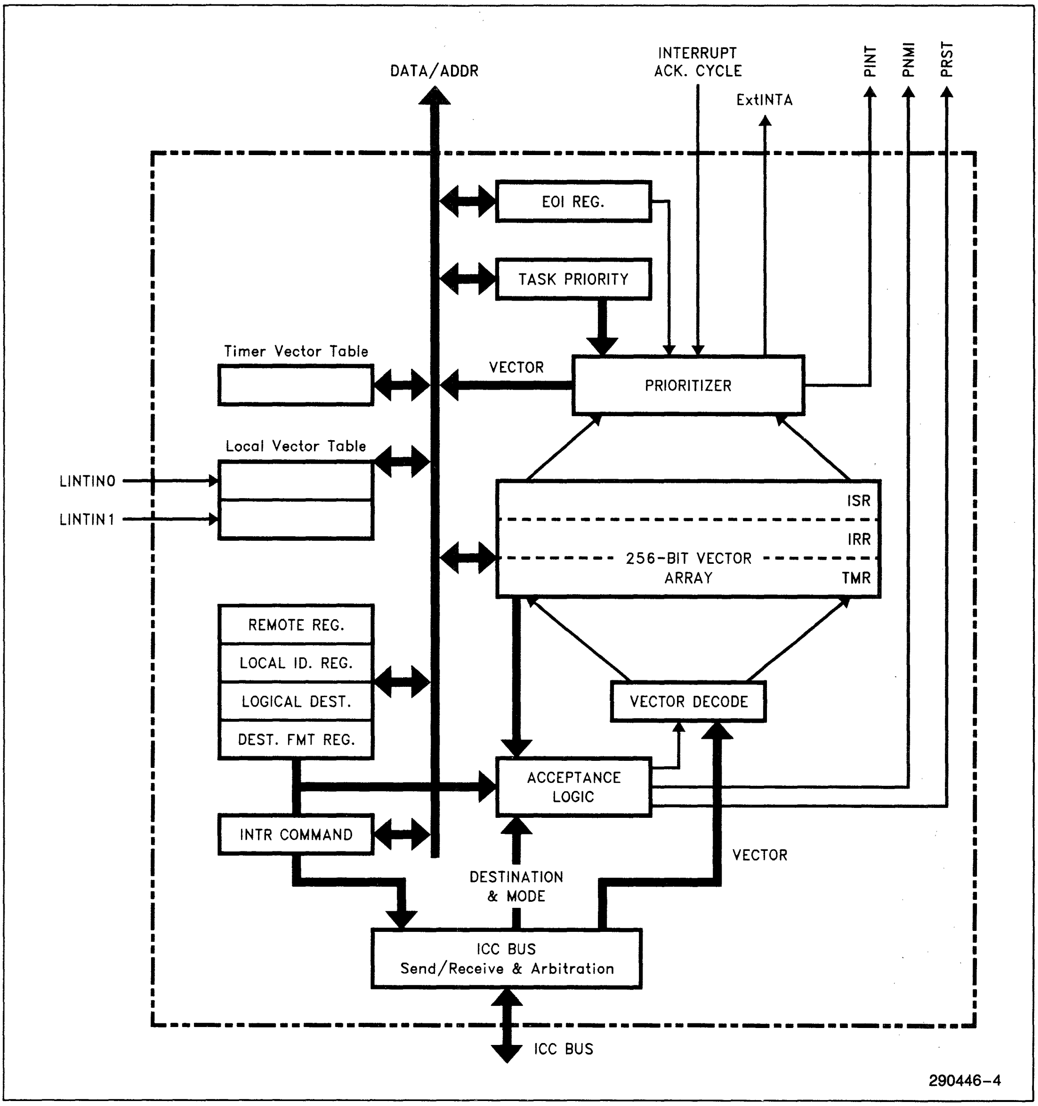


Figure 3. 82489DX Local Unit Block Diagram

5.0 INTERRUPT CONTROL MECHANISM

This section describes briefly the interrupt control mechanism in the 82489DX.

5.1 Interrupts

The interrupt control function of all 82489DXs are collectively responsible for delivering interrupts from interrupt sources to interrupt destinations in the multiprocossor system. When a processor accepts an interrupt, it uses the vector to locate the entry point of the handler in its interrupt table. The 82489DX architecture allows for 16 possible interrupt priorities; zero being the lowest priority and 15 being the highest. Priority of interrupt A "is higher than" the priority of interrupt B if servicing A is more urgent than servicing B. An interrupt's priority is implied by its vector; namely priority = vector/16.

With 256 vectors and 16 different priorities, this implies that 16 different interrupt vectors can share a single interrupt priority.

TOTAL ALLOWED INTERRUPT VECTORS

Out of 256 vectors, interrupt vectors 0 to 15 should not be used in the 82489DX. Only 240 interrupt vectors (vectors from 16 to 255) are supported in the 82489DX.

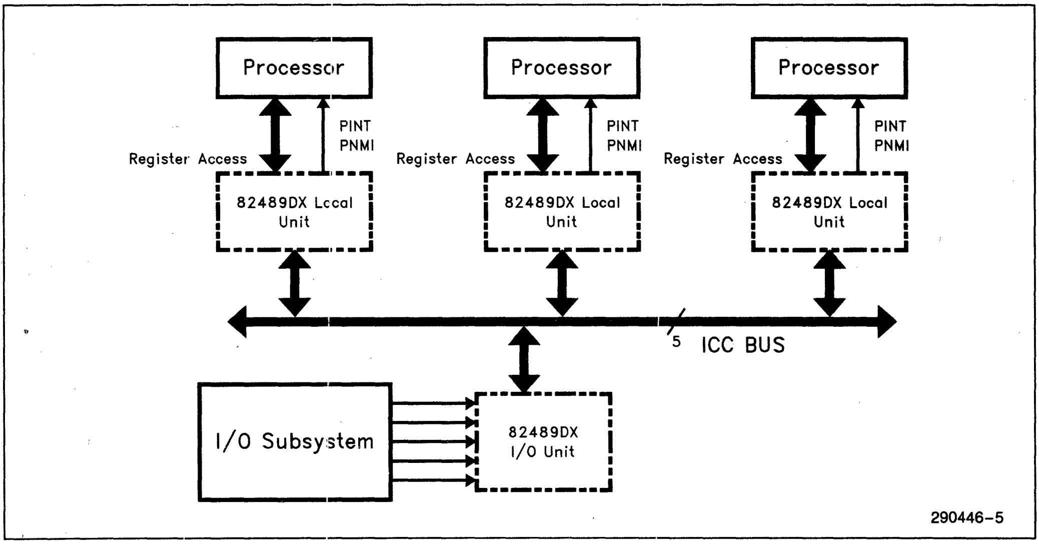


Figure 4. I/O Units and Local Units

INTERRUPT SOURCES

Interrupts are generated by a number of different interrupt sources in the system.

Possible interrupt sources are:

- Externally connected (I/O) devices. Interrupts from these external sources manifest themselves as edges or levels on interrupt input pins and can be redirected to any processor.
- Locally connected devices. These originate as edges or levels on interrupt pins, but they are al ways directed to the local processor only.
- 82489DX timer generated interrupts. Like locally connected devices, 82489DX timer can only interrupt its local processor.
- Processors. A processor can interrupt any individual processor or sets of processors. This supports software self-interrupts, preemptive scheduling, TLB flushing, and interrupt forwarding. A processor generates interrupts by writing to the interrupt command register in its Local Unit.

INTERRUPT DESTINATIONS

I/O Units can only source interrupts whereas Local Units can both source and accept interrupts, so whenever "interrupt destination" is discussed, it is implied that the Local Unit is the destination of the interrupt. In physical mode the destination processor is specified by a unique 8-bit 82489DX local ID. Only a single destination or a broadcast to all (LOCAL ID of all ones) can be specified in physical destination mode.

In logical mode destinations are specified using a 32-bit destination field. All Local Units contain a 32-bit Logical Destination register against which the destination field of the interrupt is matched to determine if the receiver is being targeted by the interrupt. An additional 32-bit Destination Format register in each Local Unit enables the logical mode addressing.

INTERRUPT DELIVERY

The description of interrupt delivery makes frequent use of the following terms:

- Each processor has a processor priority that reflects the relative importance of the code the processor is currently executing. This code can be part of a process or thread, or can be an interrupt handler. A processor's priority fluctuates as a processor switches threads, a thread or handler raises and lowers its priority level to mask out interrupt, and the processor enters an interrupt handler and returns from an interrupt handler to previously interrupted activity.
- A processor is lowest priority within a given group of processors if its processor priority is the lowest of all processors in the group. Note that more than one processor can be the lowest priority in a given group.
- A processor is the focus of an interrupt if it is currently servicing that interrupt, or if it currently has a request pending for the interrupt.

Interrupt delivery begins with an interrupt source injecting its interrupt into the interrupt system at one of the 82489DX. Delivery is complete only when the servicing processor tells its 82489DX Local Unit it is complete by issuing an end-of-interrupt (EOI) command to its 82489DX Local Unit. Only then has all (relevant) internal state regarding that occurrence of the interrupt been erased. The interrupt system guarantees exactly-once delivery semantics of interrupts to the specified destinations. Exactly-once guaranteed delivery implies a number of things:

- The interrupt system never rejects interrupts; it never NAKs interrupt injection, interrupts are never lost, and the same interrupt (occurrence) is never delivered more than once.

Clearly a single edge interrupt or level interrupt counts as a single occurrence of an interrupt. In uniprocessor systems, an occurrence of an interrupt that is already pending (IRR) cannot be distinguished from the previous occurrence. All occurrences are recorded in the same IRR bit. They are therefore treated as "the same" interrupt occurrence.

For lowest-priority delivery mode, by delivering an interrupt first to its focus processor (if it currently has one), the identical behavior can be achieved in a MP (Multiprocessor) system. If an interrupt has a focus processor then the interrupt will be delivered to the interrupt's focus processor independent of priority information. This means that even if there is a lower priority processor compared to the focus processor, the interrupt still gets delivered to the focus processor.

Each edge occurring on an edge triggered interrupt input pin is clearly a one-shot event; each occurrence of an edge is delivered. An active level on a level triggered interrupt input pin represents more of a "continuous event". Repeatedly broadcasting an interrupt message while the level is active would cause flooding of the ICC bus, and in effect transmits very little useful information since the same processor (the focus) would have to be the target.

Instead, for level triggered interrupts the 82489DX merely recreate the state of the interrupt input pin at the destination . The source 82489DX accomplishes this by tracking the state of the appropriate destination 82489DX's Interrupt Request Register (or pending bit) and only sending inter-82489DX messages when the state of the interrupt input pin and the destination's interrupt request enter a disagreement. Unlike edge triggered interrupts, when a level interrupt goes into service, the interrupt request at the servicing 82489DX is not automatically removed. If the handler of a level sensitive interrupt executes an EOI then that interrupt will immediately be raised to the processor again, unless the processor has explicitly raised its task priority, or the source of that interrupt has been removed.

5.2 Interrupt Redirection

This section specifically talks about how a processor is picked during interrupt delivery. The 82489DX supports two modes for selecting the destination processor: Fixed and Lowest Priority.

- *Fixed Dellvery Mode*
  In fixed delivery mode, the interrupt is unconditionally delivered to all local 82489DXs that match in the destination information supplied with the interrupt. Note that for I/O device interrupts typically only a single 82489DX would be listed in the destination. Priority and focus information are ignored. If the priority of a destination processor equal to or higher than the priority of the interrupt, then the interrupt is held pending locally in the destination processor's Local Unit, until the processor priority becomes low enough at which time the interrupt is dispensed to the processor. More than one processor can be the destination in fixed-delivery mode.
- *Lowest Priority Dellvery Mode*
  Under the lowest priority delivery mode, the processor to handle the interrupt is the one in the specified destination with the lowest processor priority value. If more than one processor is at the lowest priority, then a unique arbitration ID is used to break ties. For lowest priority dynamic delivery, the interrupt will always be taken by its focus processor if it has one. The lowest priority delivery method assures minimum interruption of high priority tasks. Since each Local Unit only knows its own processor priority, determining the lowest priority processor is done by arbitration on the ICC bus. Only one processor can be the destination in lowest-priority delivery mode.

INTER-82489DX COMMUNICATION

All I/O and Local Units communicate during interrupt delivery. Interrupt information is exchanged between different units on a dedicated five wire ICC bus in the form of broadcast messages. A 82489DX Unit's 8-bit ID is used as its name for the purpose of using the ICC bus, and all 82489DX units using one ICC bus should be assigned a different ID. The Arbitration ID of the Local Units used to resolve tles during lowest priority arbitration is also derived from the Local Unit's ID.

6.0 82489DX LOCAL UNIT REGISTERS DESCRIPTION

6.1 Local Unit ID Register

Each 82489DX Local Unit has a register that contains the Local Unit's 8-bit ID. The Local Unit ID serves as a physical name of the 82489DX Local Unit. It can be used in specifying destination information and is also used for accessing the ICC bus. Eight address lines A[10]-A[3] are sampled on every clock edge while RESET is asserted. The last sample remains in the Local Unit ID register after reset. Alternatively, the ID can be loaded with a register write as part of software initialization. The Local Unit ID is read-write by software.

Bits [31..24]	Bits [23..0]

82489DX Local Unit ID Register

Bits [31..24] Local Unit ID: The Local Unit ID serves as the physical "name" of the Local Unit used for addressing the 82489DX in physical destination mode and for the ICC bus usage. In a system with say four 82489DX, there are 4 Local Units and 4 I/O Units. All the 8 units should be assigned different IDs. For future compatibility use only IDs from 0 to 14.

Bits [23..0] Bits [23..0] are Reserved. They should be written with 0.

6.2 Destination Format Register

Interrupt Destination can be either addressed physically or logically. When the interrupt message addresses the destination physically, each 82489DX in the ICC bus compares the address with its own unit ID. If the message is a broadcast type then every Local Unit accepts the interrupt.

When the message addresses the destination using logical addressing scheme each Local Unit in the ICC bus compares the logical address in the interrupt message with its own Logical Destination Register. If there is a bit match (i.e., if at least one of the corresponding pair of bits match) this local unit is selected for delivery.

All the 32 bits of Destination Format Register of all 82489DX connected in the ICC bus should be written with "1" to enable the addressing scheme.

Destination Format Register

Bits [31:0]

Logical Destination Register

Bits [31:0]

For future compatibility, use only bits 31–24 of Logical Destination Register. For binary compatibility, it is strongly recommended that 82489DX software use only 8 MSB of Logical Destination Register.

6.3 Local Interrupt Vector Table Registers

The Redirection Table serves to steer interrupts that originate in the I/O subsystems to the processors. The Local Vector Table is its equivalent for interrupts that are restricted to only the local processor. The Local Vector Table contains three 32-bit registers. Register 0 corresponds to the timer, registers 1 and 2 correspond to local interrupt input pins, LINTINO and LINTIN1.

The format of both the Local 0 and Local 1 interrupt vector tables are identical. The following register description talks about both Local Interrupts 0,1 vectors.

Local Interrupts 0, 1 Interrupt Vectors

Vector: [Bits 7–0]

This is the vector to use when generating an interrupt for this entry.

Delivery Mode: [Bits 10–8]

000: Fixed

001: <reserved>

010: <reserved>

011: <reserved>

100: NMI

101: <reserved>

110: <reserved>

111: ExtINT

000: (fixed) means deliver the signal on the INT pin of the local processor. Trigger mode for "fixed" Delivery Mode can be edge or level.

100: (NMI) means deliver the signal on the NMI pin of the local processor. Vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" triggered mode.

111: (ExtINTA) means deliver the signal to the INT pin of the local processor as an interrupt that originated in an externally connected (8259A compatible) interrupt controller. ExtINTA pin is as- serted also. The INTA cycle that corresponds to this ExtINTA delivery, should be routed to the external controller that is expected to supply the vector. A delivery mode of ExtINT requires an edge trigger mode. (See the section on compatiblity for more details.)

Bit 11: Bit 11 is Reserved. It should be written 0.

Delivery Status: [Bit 12]

This field is software-read only. Software writes to this field (as part of a 32-bit word) have no effect on this bit. Delivery status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined.

0: (idle) means that there is currently no activity for this interrupt.

1: (send pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by the recently injected interrupts that are in the process of being delivered.

Local INT0 Vector Table

Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

Local INT1 Vector Table

Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

Figure 5. Local Vector Table

Bit 13: Bit 13 is Reserved. It should be written 0.

Remote IRR: [Bit 14]

This bit is used for level triggered local interrupts; its meaning is undefined for edge triggered interrupts. Remote IRR mirrors the interrupt's IRR bit of this local unit. Remote IRR is software read-only; software writes to this bit do not affect it.

Trigger Mode: [Bit 15]

The Trigger Mode field indicates the type of signal on the local interrupt pin that triggers an interrupt.

0: indicates edge sensitive,

1: indicates level sensitive.

Only the local interrupt pins can be programmed as edge or level triggered. Timer interrupts are always treated as edges.

MASK: [Bit 16]

0: enables injection of the interrupt,

1: masks injection of the interrupt.

Bits [31:17] Bits [31:17] are Reserved. Should be written 0.

6.4 Inter-Processor Interrupt Registers

A processor generates inter-processor interrupts by writing to the Interrupt Command Register in its 82489DX Local Unit. Conceptually, this can be thought of as the processor providing the interrupt's Redirection Table Entry on the fly. Not surprisingly, the layout of the Interrupt Command Register resembles that of an entry in the Redirection Table. Note that the format of this register allows a processor to generate any interrupt. A processor may use this to forward device interrupts originally accepted by it to other processors.

All fields of the Interrupt Command Register are read-write by software with the exception of the Delivery Status field which is read-only. Writing to the 32-bit word that contains the interrupt vector causes the interrupt message to be sent.

Vector: [Bits 7–0]

The vector identifies the interrupt being sent. If the Delivery Mode is "Remote Read", then the Vector field contains the address of the register to be read in the remote 82489DX's Local Unit. The addresses are listed in the section discussing 82489DX Local Unit register summary. For example, for ID register, remote read address of 02 should be specified in vector field.

Delivery Mode: [Bits 10–8]

The Delivery Mode is a 3-bit field that specifies how the 82489DX listed in the destination field (bits 63:32) should act upon reception of this signal. Note that certain Delivery Modes will only operate as intended when used in conjunction with a specific Trigger Mode. These restrictions are indicated for each Delivery Mode.

000: (Fixed) means deliver the signal on the INT pin of all processors listed in the destination. Trigger Mode for "fixed" Delivery Mode can be edge or level.

001: (Lowest Priority) means deliver the signal on the INT pin of the processor that is executing at the lower priority among all the processors listed in the specified destination; Trigger Mode for "lowest priority" Delivery Mode can be edge or level.

010: Intel Reserved. Should not be used.

011: (Remote Read) is a request to a remote 82489DX Local Unit to send the value of one of its registers over the ICC bus. The register is selected by providing its address in the Vector field. The register value is latched by the requesting 82489DX and stored in the Remote Register where it can be read by the local processor. A Delivery Mode of "Remote Read" requires an "edge" Trigger Mode.

Interrupt Command Register [31:0]

Bits [31:20]	Bits [19:18]	Bits [17:16]	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

100: (NMI) means deliver the signal on the NMI pin of all processors listed in the destination, vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" Trigger Mode.

101: (Reset) means deliver the signal to all local units listed in the destination. The destination local unit will assert/deassert its PRST output pin. All addressed 82489DX Local Units will assume their reset state but preserve their ID. One side effect of an ICC bus message with Delivery Mode equal to "Reset" that results in a deassert of reset is that all Local Units (whether listed in the destination or not) will reset their lowest-priority tie breaker arbitration ID to their Local Unit ID (see the section on the ICC bus for details). A Delivery Mode of "Reset" requires a "level" Trigger Mode. "RESET" should not be used with "self" or "all incl self" Shorthand mode since it will leave the system in non-recoverable reset state. If "RESET" is used with "all excl self" mode software should make sure that only one CPU executes this instruction in a MP system.

110: Intel Reserved. Should not be used.

111: Intel Reserved. Should not be used

Destination Mode: [Bit 11]

This field determines the interpretation of the Destination field.

0: (Physical Mode): in Physical Mode, a destination 82489DX is identified by its Local Unit ID. Bits 56 through 63 (8 MSB of the destination field) specify the 8-bit 82489DX Local Unit ID.

1: (Logical Mode): in Logical Mode, destinations are identified by matching on Logical Destination under the control of the Destination Format Register in each Local 82489DX. The 32-bit Destination field is the logical destination.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered;

Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit. Software can read to find out if the current interrupt has been sent, and the Interrupt Command Register is available to send the next interrupt. If the Interrupt Command Register is overwritten before the Delivery Status is "Idle", then the destiny of that interrupt is undefined; i.e., the interrupt may have been lost.

Bit 13: Bit 13 is Reserved . Should be written 0.

Level: [Bit 14]

Software can use this bit in conjunction with the Trigger Mode bit when issuing an inter-processor interrupt to simulate assertion/deassertion of level sensitive interrupts.

To assert: Trigger mode = 1 and Level = 1.

To deassert: Trigger mode = 1 and Level = 0.

For example, a message with Delivery Mode of "Reset", a Trigger Mode of "Level", and Level bit of 0 deasserts Reset to the processor of the addressed 82489 DX Local Unit(s). As a side effect, this will also cause all 82489DX to reset their Arbitration ID to their unit ID. (The Arb ID is used for tie breaking in lowest priority arbitration.)

Trigger Mode: [Bit 15]

Software can use this in conjunction with Level Assert/Deassert to generate interrupts that behave as edges or levels.

0: Edge

1: Level

Remote Read Status: [Bits 17,16]

This field indicates the status of the data contained in the Remote Read register. This field is read-only to software. Whenever software writes to the Interrupt Command Register using Delivery Mode "Remote Read" the Remote Read status becomes "in-progress" (waiting for the remote data to arrive). The remote 82489DX Local Unit is expected to respond in a fixed amount of ICC bus cycles. If the remote 82489DX Local Unit is unable to do so, then the Remote Read status becomes "Invalid". If successful, the Remote Read status resolves to "Valid". Software should poll this field to determine completion and success of the Remote Read command.

00: (invalid): the content of the Remote Read Register is invalid. This is the case after a Remote Read command issued and the remote 82489DX Local Unit was unable to deliver the Register content in time.

01: (in progress): a Remote Read command has been issued and this 82489DX is waiting for the data to arrive from the remote 82489DX Local Unit.

10: (valid): the most recent Remote Read command has completed and the remote register content in the Remote Read Register is valid.

11: reserved.

Destination Shorthand: [Bits 19,18]

This field indicates whether a shorthand notation is used to specify the destination of the interrupt and if so, which shorthand is used. Destination Shorthands do no use the 32-bit Destination field, and can be sent by software with a single 32-bit write to the 82489DX's Interrupt Command Register. Shorthands are defined for the following common cases: software self interrupt, interrupt to all processors in the system including the sender, interrupts to all processors in the system excluding the sender.

00: (dest field): means that no shorthand is used. The destination is specified in the 32-bit Destination field in the second word (bits 32 to 63) of the Interrupt Command Register.

01: (self): means that the current 82489DX Local Unit is the single destination of the interrupt. This is useful for software interrupts. The Destination field in the Interrupt Command Register is ignored. RESET Delivery mode should not be used with self destination. Only FIXED delivery mode should be used with SELF.

10: (all incl self): means that the interrupt is to be sent to "all" processors in the system including the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones. RESET assert Delivery mode should not be used with "all incl self" destination.

11: (all excl self): means that the interrupt is to be sent to "all" processors in the system with the exception of the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones. All-excl-self is useful during selection of a boot processor (init) and also for TLB flush where "self" is flushed using the processor flush instruction. Only one CPU in a MP system should execute "all excl self" destination if used with RESET Delivery mode.

Bits [31:20] Bits [31:20] are Reserved. They should be written 0.

Interrupt Command Register [63:32]

Bits [63:32]

Destination: [Bits 63–32]

This field is only used when the Destination Shorthand is set to "Dest Field". If Destination Mode is Physical Mode, **then the 8 MSB contain a Destination unit ID.** If Logical Mode, the full 32-bit Destination field contains the logical address. The enabling is done by Destination Format Register.

6.5 IRR, ISR, TMR Registers

INTERRUPT ACCEPTANCE

All 82489DX Local Units listen to all messages sent over the ICC bus. For each message, the local unit first checks if it belongs to the destination in the message. It does this by matching the 32-bit Destination field in the message against its logical Destination Register, if the message addresses in logical mode, and against its physical ID, if the message addresses in physical mode. All 82489DX Local Units that match are said to "belong to the group".

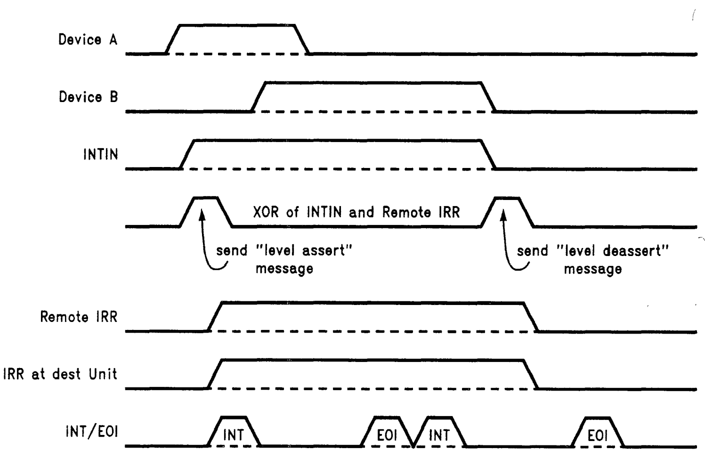
Each 82489DX Local Unit contains three 256-bit registers that play a role in the acceptance of interrupts and in dispensing accepted interrupts to the local processor. Each of these registers is a bit array where bit position i tracks information about the interrupt with vector i. These bits track information about the (PINT) maskable interrupts only. They are not relevant for NMI, RESET or ExtINT type of interrupts. The Interrupt Request Register (IRR) contains the interrupts accepted by this 82489DX Local Unit but not yet dispensed to the processor. The In Service Register (ISR) contains the interrupts that are currently in service by the processor, i.e., the interrupts that have been dispensed to the processor but for which the processor has not yet signaled the End-Of-Interrupt.

Note that the 82489DX's IRR and ISR registers have the same meaning and operation as in the 8259A in fully nested/non-specific EOI mode. Note also that these registers play no role in providing 8259A compatibility. Compatibility is handled by making an external 8259A-style controller directly visible to the processor and having the 82489DX become transparent.

Each interrupt has a vector associated with it, which determines the bit position, and hence the priority for the interrupt. When an interrupt is being serviced, all equal or lower priority interrupts are automatically masked by the 82489DX Local Unit.

The Trigger Mode Register (TMR) indicates for each interrupt whether the interrupt is edge or level. This information is transmitted with each 82489DX interrupt request message and reflects the Trigger Mode bit in the interrupt's Redirection Table entry. If an interrupt goes in service and the TMR bit is 0 (edge), then the interrupt's IRR bit is cleared at the same time the ISR bit is set. If the TMR bit is 1 (level), then the IRR bit is not cleared when the interrupt goes in service. In the latter case, the IRR bit mirrors the state of the interrupt's input pin.

The following diagram shows 82489DX operation with devices A and B sharing a level triggered interrupt input. The diagram illustrates how Remote IRR, and the IRR bit at the destination 82489DX track the state of INTIN. It also illustrates how an EOI is followed immediately be re-raising the interrupt as long as the INTIN is still asserted by some device.


Device A
Device B
INTIN
XOR of INTIN and Remote IRR
send "level assert" message
send "level deassert" message
Remote IRR
IRR at dest Unit
INT/EOI
INT
EOI
INT
EOI

290446-6

Figure 6. Interrupt Sharing

ISR, IRR, and TMR are read-only by software. Each of these 256-bit registers is accessed as four separate 32-bit registers. Note that there is no general Interrupt Mask Register (IMR) as in the 8259A. The processor masks interrupts temporarily by writing to the Icoal unit's Task Priority Register (described shortly).

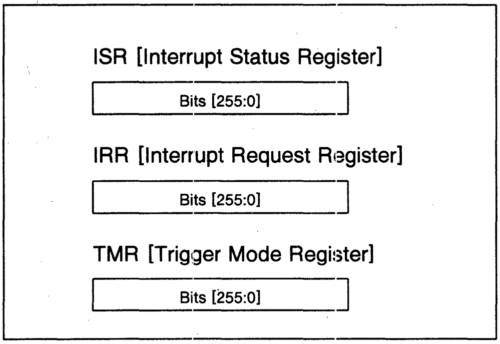

ISR [Interrupt Status Register]
Bits [255:0]
IRR [Interrupt Request Register]
Bits [255:0]
TMR [Trigger Mode Register]
Bits [255:0]

Figure 7. ISR, IRR, and TMR

TMR (Trigger Mode Register):

If 0 [edge triggered] the corresponding IRR bit is automatically cleared when interrupt service starts. If 1[level triggered] this is not the case; instead, the source 82489DX must explicitly request the IRR bit be cleared (upon deassert of the interrupt input pin or upon sending an appropriate interprocessor interrupt). Upon acceptance of an interrupt, the TMR bit is cleared for edge triggered interrupts and set for level triggered interrupts. This information is carried in the accepted interrupt message. The source 82489DX I/O unit also tracks the state of the destination unit's IRR bit (Remote IRR bit in the Redirection Table). When a level triggered interrupt input is deasserted, the source 82489DX I/O unit detects the discrepancy between the input pin state and the Remote IRR, and automatically sends a message telling the destination 82489DX to clear IRR for the interrupt.

IRR (Interrupt Request Register):

It contains the active interrupt requests that have been accepted, but not yet dispensed by this 82489DX Local Unit. A bit in IRR is set when the 82489DX Local Unit accepts the interrupt. When TMR is 0, it is cleared when the interrupt is serviced; when TMR is 1, it is cleared when the 82489DX Local Unit receives a message to clear it.

ISR (In Service Register):

It marks the interrupts that have been delivered to the processor, but that have not been fully serviced in that an End-Of-Interrupt has not yet been received. The ISR register reflects the current state of the processor's interrupt stack.

ACCEPTANCE MECHANISM

Interrupt acceptance proceeds as follows. If the delivery mode is Fixed, then each unit in the destination group unconditionally accepts the interrupt message and sets the interrupt's IRR bit. If the delivery mode is Lowest Priority, then each processor in the group first checks if it is currently the focus of the interrupt by checking its ISR and IRR. If an 82489DX finds one of these bits set for the incoming interrupt, then that 82489DX Local Unit accepts the interrupt independent of priority, and "signals" the other 82489DX Local Units to abort the priority arbitration. This avoids multiple delivery of a same interrupt occurrence to different processors, consistent with interrupt delivery semantics in uniprocessor systems as described above. If a message is to be delivered for NMI or Reset, then all 82489DX Local Units listed in the destination unconditionally assert/deassert the corresponding output pin. ISR, IRR, etc. are bypassed for NMI or reset and vector information is undefined.

The acceptance decision process is illustrated in the flow chart below.

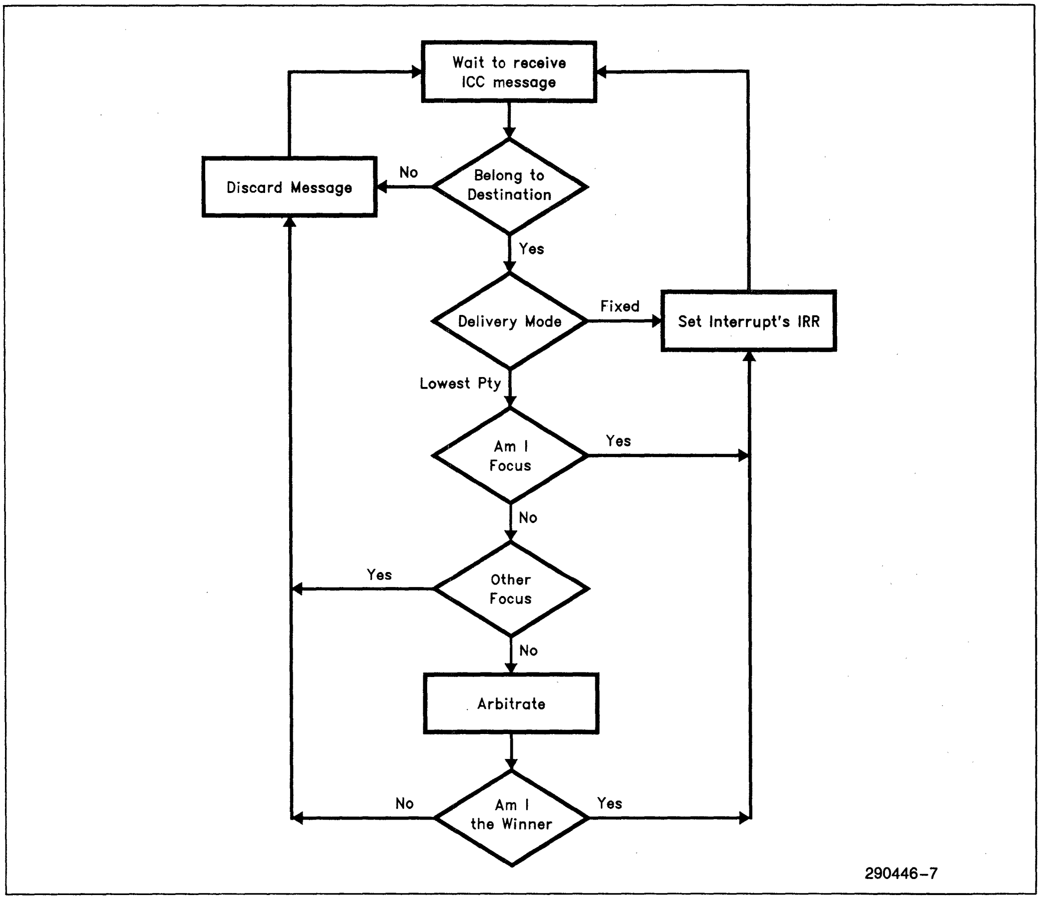


Figure 8. Interrupt Acceptance Flow Chart

6.6 Tracking Processor Priority

Each 82489DX Local Unit should be programmed with task priority so that it can mask interrupts that are less priority than that of the processor temporarily.

Task switching and task priority changes are the result of explicit software action. The operating system may define a number of task scheduling classes. Examples are an idle class, a background class, a foreground class, and a time critical class. Alternatively, different classes can be assigned to user code versus system code. If tasks in different classes are executing when an interrupt comes in, then it may be advantageous to interrupt the processor currently running the task in the least important class. Clearly, if one processor is idle while others are doing work, the idle processor would be the obvious target for servicing the interrupt. This implies that there is use in defining priority levels below all interrupt levels that can participate in lowest priority delivery selection.

At times, the operating system may need to block out interrupts from being serviced. For example, to synchronize access to a shared data structure between a device driver and its interrupt handler the driver raises it priority to equal or higher than the interrupt's priority.

The local 82489DX supports this via its Task Priority Register (the 8259A supports this via the interrupt mask register (IMR).) Software that wants to make use of this is required to inform its 82489DX Local Unit of the prioity change by updating the Task Priority Register. The Task Priority field is 8 bits providing up to 256 distinct priorities. The 4 MSB of this register correspond to the 16 interrupt priorities while the 4 LSB provide more precision. Priorities are best noted as x:y, where x is the value of the 4 MSB and y is the value of the 4 LSB. For example, Task Priority Register values 0:y with $0 < y < 15$ (and 0 in the 4 MSB) can be used to represent the priorities of the task scheduling classes described above ($y = 0$ for idle; $y = 1$ for background; etc.). Except for interrupts with vectors 0 through 15 (which are often predefined by the processor) which all have priority 0:0, the priorities of all other interrupts and their handlers is x:0 with $1 < x < 15$ and is above the base task priorities 0:y.

For example, interrupt vector 123 has priority 7:0 ($123/16 = 7$) and can be masked by any task that raises its priority to a value equal or higher than 7:0.

82489DX uses Task priority register for the purpose of masking the interrupts. The task priority register should be programmed with a priority value to specifiy the priority of task the processor is executing. 82489DX masks any interrupts of lower or equal priority when compared with task priority.

When task priority register is programmed with the priority 15, all the interrupts are masked. When task priority register is programmed with priority level X, by definition, all the interrupts of priority X and below X will be masked. When task priority register is programmed with the priority 0 then all the interrupts above priority 0 are allowed to interrupt the processor. This means that when task priority register is programmed even with the lowest value, i.e., 0, interrupts of priority 0 will be masked. So only 240 interrupt vectors should be used in 82489DX. Interrupt vecotrs from 0 to 15 should not be used.

The first priority value computed is the maximum of:

- Task Priority (4msb : 4lsb) and
- the priority of the highest order ISR bit set ((vector/16) :0).

The value is used to determine whether or not a pending interrupt can be dispensed to the processor.

The second priority value computed is the maximum of:

- Task Priority (4msb : 4lsb), and
- the priority of the highest order ISR bit set ((vector/16) :0), and
- the priority of the highest order IRR bit set ((vector/16) :0).

This value is used during arbitration as part of lowest-priority delivery.

Task Priority Register

Bits [31:8]	Bits [7:0]

From the information in the Task Priority Register and the priority information derived from the ISR and IRR register, the 82489DX Local Unit computes two additional priority values:

Bits [31:8] Bits [31:8] are Reserved. They should be written 0.

Bits [7:0] Task Priority

Bits [7:0] are used to specify the task priority.

6.7 Dispensing Interrupts

DISPENSING INTERRUPTS TO THE LOCAL PROCESSOR

Once a 82489DX Local Unit accepts an interrupt, it guarantees delivery of the interrupt to its local processor. (This part of the 82489DX functions similarly to an 8259A.) Dispensing a maskable interrupt to the local processor begins when the Local Unit asserts the INT pin of its processor. If the processor has interrupts enabled, it will respond by issuing an INTA cycle. This causes the Local Unit to freeze its internal priority state and release the 8-bit vector of the highest priority interrupt on the data bus where it is read by the processor and used to find the handler's entry point. The INT/INTA protocol also causes the interrupt's ISR bit to be set. The corresponding bit in the IRR register is only cleared if the TMR register indicates it should do so (edge triggered interrupts), otherwise (level triggered interrupts), IRR is only cleared when the Interrupt Input Pin is deasserted.

6.8 Spurious Interrupt Vector Register

SPURIOUS INTERRUPT

Note that it can happen that a level-triggered interrupt is deasserted right before its INTA cycle. In that case, all IRR bits may be clear and the prioritizer may not find a vector to give to the processor. To satisfy the processor's demand for a vector, instead, the 82489DX will return a spurious interrupt vector instead.

A similar situation may occur when the processor raises its Task Priority at or above the level of the interrupt for which the Processor INT pin is currently being asserted. When the INTA cycle is issued, the interrupt that was to be dispensed has become masked (masked but remembered).

Dispensing the spurious interrupt vector does not affect the ISR register, so the handler for this vector should just return without EOI. If the vector is shared with a valid interrupt, then the handler can read the vector's bit in the ISR register to check if it is invoked for the valid interrupt (ISR bit set) or not (ISR bit clear). Given the range of 240 vectors, overloading the spurious interrupt with a valid interrupt is not expected to be common practice. The spurious interrupt vector to be used by a Local Unit is programmable via the Spurious Interrupt Vector Register.

UNIT ENABLE

It is possible that Local Units exist in the system that do not have a processor to which to dispense interrupts. The only danger this represents in the system is that if any interrupt is broadcast to all processors using lowest priority delivery mode when all processors are at the lowest priority, there is a chance that a Local Unit without the processor may accept the interrupt if this Local Unit happens to have the lowest Arb ID at the time. To prevent this from happening, all Local Units initialize in the disabled state and must be explicitly enabled before they can either start accepting or transmitting messages from the ICC bus. A disabled 82489DX Local Unit only responds to messages with Delivery Modes set to "Reset". Reset deassert messages should be sent in Physical Destination mode using the target's Local Unit ID since the logical destination information in the lcoal units is undefined (all zeroes) when the 82489DX comes out of Reset.

Bits [31:9]	Bit 8	Bits [7:0]

Figure 9. Spurious Interrupt Vector Register

Bits [31 .. 9] Reserved Bits. Should be written 0.

Unit Enable: [Bit 8]

0: When a 0 is written to this bit, this Local Unit gets disabled with regard to responding to messages sent as well as transmitting on the ICC bus. It only responds to messages with Delivery Mode set to "Reset". Reading a 0 at this bit indicates that the unit is disabled.

1: When a 1 is written to this bit, the current Local Unit is enabled for both transmitting and receiving unit messages. Reading a 1 at this bit indicates that the unit is enabled.

Spurious Vector: [Bits 7–0]

For future compatibility, bits [3–0] should be 1111.

6.9 End-of-Interrupt (EOI) Register

Before returning from the interrupt handler, software must issue an End-Of-Interrupt (EOI) command to the 82489DX Local Unit. The data written to EOI register is don't care. This tells the 82489DX to clear the highest priority bit in the ISR register since the interrupt is no longer in service. Upon EOI, 82489DX goes through prioritization returning to the next highest priority activity. This can be a previously interrupted handler (from ISR), a pending interrupt request (from IRR), or an interrupted task (from Task Priority).

Bits [31:0]

Figure 10. EOI Register

Bits [31:0]: are don't care.

6.10 Remote Read Register

Since all 82489DX Local Units would typically occupy the same address range, an 82489DX local unit's registers can only be accessed by the local processor. From a system debugging point of view, this would mean that a large amount of state would become inaccessible if its corresponding processor hangs for whatever reason. To assist in the debugging of MP systems, the 82489DX support a mechanism that provides read-only access to any register in any other 82489DX lcoal unit in the system.

To read any register in a "remote" 82489DX Local Unit, the processor writes to the Interrupt Command Register specifying a Delivery Mode equal to "Remote Read". The remote 82489DX is specified in the Destination field of the Interrupt Command Register in the usual fashion. Debug software would make sure that this selects a single 82489DX only—for example by using the target's 82489DX Local Unit ID in physical destination mode. Since no vector is associated with remote register access, the Vector field in the Interrupt Command Register is used to select the individual remote 32-bit register to be read. The selector value corresponds to the address (offset) of the register in the local 82489DX's address space. Sending a "Remote Read" command results in sending a message on the ICC bus. The destination 82489DX responds by placing the 32-bit content of the selected register on the ICC bus. This value is read by sending the 82489DX and place it in the Remote Register where software can get at it using regualr register access to its 82489DX Local Unit. The Remote Register is software read-only. The contents of the Remote Register is valid when the Delivery Status in the Interrupt Command Register has become "Idle" again.

Remote Read Register

Bits [31:0]

Figure 11. Remote Register

Bits [31:0] Bits [31:0] contain the contents of Remote Read Register.

6.11 82489DX Local Configuration

LOCAL VERSION REGISTER

Each 82489DX Local Unit contains a hardware Version Register that identifies this 82489DX Local Unit version. This register is read only.

Local Version Register

Bits [31:8]	Bits [7:0]

Figure 12. Local Version Register

Version: [Bits 7–0]

This is a version number that identifies this version. This field is hardwired and is read-only. Will be read as "1" for 82489DX.

Bits [31:8] Bits 31:8 are reserved.

6.12 82489DX Timer Registers

Overview

82489DX Local Unit contains one 32-bit wide programmable binary timer for use by the local processor. The timer can select its clock base from one of three possible clock inputs. A timer mode can be programmed to operate in either one-shot mode or periodic mode. The timer can be configured to interrupt the local processor with a vector.

Time Base

The 82489DX has two independent clock input pins:

1. The CLK pin provides the clock signal that drives the 82489DX's internal operation.
2. The TMBASE pin allows an independent clock signal to be connected to the 82489DX for use by the timer functions.

Signals from both CLK and TMBASE can be used as clock inputs that feed the timer. In addition, the 82489DX contains a divider that can be configured to divide either input clock signal. The divider can be programmed to divide the selected input clock by 2, 4, 8, or 16. CLK, TMBASE, and the output of the divider together provide three time bases: Base 0, Base 1, and Base 2. Base 0 is always equal to CLK: Base 1 is always equal to TMBASE; and base 2 is one of; CLK/2, CLK/4, CLK/8, CLK/16, TMBASE/2, TMBASE/4, TMBASE/8, or TMBASE/16. The timer can independently select one of these three time bases as its clock input as depicted in the following diagram.

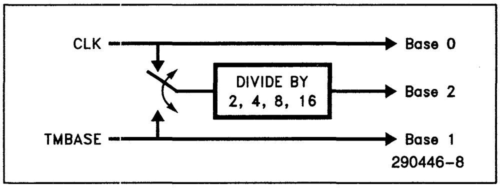


Figure 13. Time Bases

Bits [31:3]	Bit 2	Bits [1:0]

Figure 14. Divider Configuration Register

Bits [31:3] Bits 31 to 3 are reserved. They should be written 0.

Divider Input:

[Bit 2] Selects whether divider's input connects to the 82489DX Local Unit's CLK pin or TMBASE pin.

0: means the divider takes its input signal from CLK,

1: means use TMBASE.

Divide By: [Bits 1,0]

This field selects by how much the divider divides.

00: divide by 2

01: divide by 4

10: divide by 8

11: divide by 16

Timer

Software starts a timer going by programming its Initial Count Register. The timer copies this value into the Current Count Register and starts counting down at the rate of one count for each time base pulse. The time is one of Base 0, Base 1, or Base 2.

The timer has a programmable mode which can be One-Shot or Periodic. After the timer reaches zero in One-Shot mode, the timer simply stays at zero until it is reprogrammed. In Periodic mode, the timer automatically reloads its Current Count from the Initial Count and starts counting down again.

For the timer, interrupt generation can be disabled or enabled, and an arbitrary interrupt vector can be specified. When enabled and the timer reaches zero, an interrupt is generated at the 82489DX Local Unit. Timer generated interrupts are always treated as edges. They can only generate maskable interrupts to the local processor.

A timer set up with its interrupt masked is useful as a time base that can be sampled by the local processor by reading the Current Count Register, for the purpose of measuring the intervals. By mapping the 82489DX's register space into a read-only user page, safe and efficient performance monitoring of user programs can be supported.

If necessary, software may want to ensure that periodic timer interrupts on the different 82489DX Local Units are staggered such that the 82489DXs don't all deliver their interrupt (e.g., a timer slice interrupt) to their local processor at the same time. This staggering avoids bursts of contention for shared resources (bus, cache lines, dispatch queue, locks). Randomness occurring "naturally" may be sufficient to ensure staggering.

Initial Count Register

Bits [31:0] Initial Count

Current Count Register

Bits [31:0] Current Count

Figure 15. Initial Count and Current Count Registers

Initial Count: Software writes to this register to set the initial count for timer. This register can be written at any time. When written, its value is copied to the Current Count Register and countdown starts or continues from there. The Initial Count Register is read-write by software.

Current Count: This is the current count of timer. It is read-only by software and can be read at any time.

The timer is configured via its Local Vector Table entry shown below (see also Interrupt Control in this section).

Vector: [Bits 7–0]

This is the 8-bit interrupt vector to be used when timer generates an interrupt.

TIMER VECTOR TABLE

Bits [31:20]	Bits [19:18]	Bit 17	Bit 16	Bits [15:13]	Bit 12	Bits [11:8]	Bits [7:0]

Figure 16. Local Vector Table: Timer Entry

Bits 11–8 Reserved. Should be written 0.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered; Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit.

Bits 15–13: Reserved. Should be written 0.

MASK: [Bit 16]

This bit serves to mask timer interrupt generation.

0: means not masked, when timer reaches 0, it generates an interrupt with vector at the 82489DX Local Unit

1: means masked, and no interrupt is generated.

Timer Mode: [Bits 17]

This field indicates the operation mode of timer.

0: (One-Shot): the Current Count Register remains at zero after the timer reaches zero, and software needs to reassign the timer's Initial Count Register to rearm the timer.

1: (Periodic): when the timer reaches zero, the Current Count Register is automatically reloaded with the value in the Initial Count Register, and the timer counts down again.

Timer Base: [Bits 19,18]

This field selects the time base input to be used by timer.

00: (Base 0) uses "CLKIN" as input;

01: (Base 1) uses "TMBASE";

10: (Base 2) uses the output of the divider (Base 2).

Bits [31:20] Bits [31:20] are Reserved. Should be written 0.

7.0 82489DX I/O UNIT REGISTERS

REGISTERS ADDRESSING SCHEME

The I/O Unit indirect addressing scheme uses two registers directly mapped into the processor's address space: the I/O Register Select register and the I/O Window register. The I/O register select register selects which I/O unit Register appears in the I/O Window register where it can be manipulated by software.

I/O Register Select Register

Bits [31:8]	Bits [7:0]

Figure 17. I/O Register Select Register

Bits [31:8]: Reserved. Should be written 0.

Bits [7:0]: I/O REGISTER SELECT: This register selects an 82489DX I/O unit register. The contents of the selected 32-bit register can be manipulated via the I/O Window Register. The I/O Register Select register is read-write by software.

I/O Window Register

Bits [31:0]

Figure 18. I/O Window Register

Bits [31:0] I/O WINDOW REGISTER: This register is mapped onto the I/O Unit's register selected by the I/O Register Select register. Readability/writability by software is determined by the I/O unit register that is currently selected.

The addresses (offsets to a platform-defined base address) of all registers are listed in the register summary section. Note that register offsets are aligned on 128-bit boundaries; in other words, registers are located only at every fourth 32-bit address. This eliminates the need for lane-steering glue logic when connecting the 82489DX's 32-bit data bus to a wider (64-bit and 128-bit) bus.

82489DX I/O UNIT CONFIGURATION

I/O Unit ID Register

Each 82489DX I/O Unit has a register that contains the I/O Unit's 8-bit ID. The I/O unit ID serves as a physical name of the 82489DX I/O Unit. It is used in arbitrating for ICC bus ownership when the I/O unit wants to access the ICC bus for sending any interrupt message. Unlike the local unit ID, the I/O unit ID is not latched-in from the address bus during hardware reset. The I/O unit ID is set to 0 during reset. The software has to write different ID into the I/O Units before starting interrupt messages on the ICC bus.

I/O Unit ID

Bits [31:24]	Bits [23:0]

Bits [31:24] I/O Unit ID:

The I/O unit ID serves as the physical "name" of the 82489DX unit used for arbitration purposes for the ICC bus usage. In a system with, say, four 82489DX, there are 4 Local Units and 4 I/O Units. All the 8 units should be assigned different ID. The IDs should start with 0 and each unit should have different ID.

Bits [23:0] Bits 23..0 are reserved. Should be written 0.

I/O Unit Version Register

Each 82489DX I/O Unit contains a hardware Version Register that identifies this 82489DX I/O unit version. This register is read only.

I/O Unit Version Register

Bits [31:24]	Bits [23:16]	Bits [15:8]	Bits [7:0]

Version: [Bits 7–0]

This is a version number that identifies this version. This field is hardwired and is read-only. Will be read as "1" for 82489DX.

Bits [15:8] Bits [15:8] are reserved.

Max Redir Entry: [Bits 23–16]

This is the entry number (0 being the lowest entry) of the highest entry in the Redirection Table. It is equal to the number of Interrupt Input Pins minus one of this I/O Unit. This field is hardwired and is read-only.

In the 82489DX I/O unit this is read as 15.

Bits [31:24] Bits [31:24] are reserved.

I/O UNIT INTERRUPT SOURCE REGISTERS

Redirection Tables

The Redirection Table has a dedicated entry for each interrupt input pin. Unlike IRQ pins of the 8259A, the notion of interrupt priority is completely unrelated to the position of the physical interrupt input pin on the 82489DX. Instead, software can decide for each pin individually what it wants the vector (and therefore the priority) of the corresponding interrupt to be. For each individual pin, the operating system can also specify whether the interrupt is signaled as edges or levels, as well as the destination and delivery mode of the interrupt. The information in the Redirection Table is used to translate the interrupt manifestation on the corresponding interrupt pin into an inter-82489DX message.

In order for a signal on an edge-sensitive Interrupt Input pin to be recognized as a valid edge (and not a glitch) the input level on the pin must remain asserted until the time 82489DX I/O Unit sends the corresponding message over the ICC bus. Only then will the source 82489DX be able to recognize a new edge on that Interrupt Input pin. That new edge will only result in a new invocation of the handler if its acceptance by the destination 82489DX causes the Interrupt Request Register bit to go from 0 to 1. (In other words, if the interrupt wasn't already pending at the destination.)

82489DX I/O unit has 16 Redirection Table entries. The layout of an entry in the Redirection Table is as follows:

Redirection Table Entry

Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

Vector (Bits [7:0]

Interrupt vector for this interrupt

Delivery Mode (Bits [10:8])

000: Fixed

001: Lowest Priority

010: <reserved>

011: <reserved>

100: NMI

101: Reset

110: <reserved>

111: ExtINT

Destination Mode (Bit 11)

0: Physical

1: Logical

Delivery Status (Bit 12)

0: Idle

1: Send Pending

Bit 13 Bit 13: Reserved. Should be written 0.

Remote IRR (Bit 14)

Reflects the Remote IRR bit

0: Remote IRR bit is clear.

1: Remote IRR bit is set.

Trigger Mode (Bit 15)

0: Edge

1: Level

Mask (Bit 16)

0: Not Masked

1: Masked

Bits [31:17] Reserved. Should be written 0.

DESCRIPTIONS

Vector: [Bits 7–0]

The vector field is an 8-bit field containing the interrupt for this interrupt.

Delivery Mode: [Bits 10–8]

The Delivery Mode is a 3-bit field that specifies how the 82489DXs listed in the destination field should act upon reception of this signal. Note that remote read is not supported for I/O device interrupts. Note that certain Delivery Modes will only operate as intended when used in conjuction with a specific Trigger Mode. These restrictions are indicated for each Delivery Mode.

000: (Fixed) means deliver the signal on the INT pin of all processors listed in the destination. Trigger Mode for "fixed" Delivery Mode can be edge or level.

001: (Lowest Priority) means deliver the signal on the INT pin of the processor that is executing at the lower priority among all the processors listed in the specified destination; Trigger Mode for "lowest priority" Delivery Mode can be edge or level.

100: (NMI) means deliver the signal on the NMI pin of all processors listed in the destination, vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" Trigger Mode.

101: (Reset) means deliver the signal to all processors listed in the destination by asserting/deasserting the 82489DX's Reset output pin. All addressed 82489DXs' Local Units will assume their reset state but preserve their unit ID. One side effect of a unit message with Delivery Mode equal to "Reset" that results in a deassert of reset is that all 82489DXs' Local Units (whether listed in the destination or not) will reset their lowest-priority tie breaker arbitration ID to their unit ID (see the section on the ICC bus for details). A Delivery Mode of "Reset" requires a "level" Trigger Mode.

111: (ExtINT) means deliver the signal to the INT pin of all processors listed in the destination as an interrupt that originated in an externally connected (8259A-compatible) interrupt controller. The Local Unit receiving this interrupt will activate ExtINTA in response to this interrupt message. A Delivery Mode of "ExtINT" requires an "edge" Trigger Mode. (See the section on Compatibility for details.)

Destination Mode [Bit 11]

This field determines the interpretation of the Destination field.

0: (Physical Mode): in Physical Mode, a destination 82489DX Local Unit is identified by its unit ID. Bits 56 through 63 (8 MSB of the destination field) specify the 8-bit unit ID.

1: (Logical Mode): in Logical Mode, destinations are identified by matching on Logical Destination under the control of the Destination Format Register in each 82489DX Local Unit. The 32-bit Destination field is the logical destination.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered; Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit.

Bit 13: Bit 13 is Reserved. Should be written 0.

Remote IRR: [Bit 14]

This bit is used for level triggered interrupts; its meaning is undefined for edge-triggered interrupts. Remote IRR mirrors the interrupt's IRR bit of the destination 82489DX Local Unit. When the value of the bit disagrees with the state of the Interrupt Input line, a unit message is automatically sent to make the destination's IRR both reflect the new state of the Interrupt Input line, and then the Remote IRR bit is updated to track its associated IRR bit. Remote IRR is software read-only; software writes to this bit do not affect it.

Trigger Mode: [Bit 15]

The Trigger Mode field indicates the type of signal on the interrupt pin that triggers an interrupt.

0: indicates edge sensitive,

1: indicates level sensitive.

Mask: [Bit 16]

Use this bit to mask injection of this interrupt.

0: indicates that injection of this interrupt is not masked. An edge or level on an interrupt pin that is not masked results in the delivery of the interrupt to the destination.

1: indicates that injection of this interrupt is masked. Edge-sensitive interrupts signaled on a masked interrupt Input pin are simply ignored (i.e., it is not delivered and is not held pending). Level-asserts or deasserts occurring on a masked level-sensitive pin are also ignored and have no side effects. As expected, changing the mask bit from unmasked to masked while the level remains asserted has the side effect of deasserting the level. It is software's responsibility to deal with the case where the Mask bit is set after the interrupt message has been sent but before the interrupt is dispensed to the processor.

Bits [31:17] Bits [31:17] are reserved. Should be written 0.

Destination

Bits [63:32]

Destination: If the Destination Mode of this entry is "Physical Mode", then the 8 MSB [bits 56 through 63] contain an 82489DX Local Unit ID. If Logical Mode, then the Destination field potentially defines a set of processors. The interpretation of the 32-bit destination field is further enabled by the Destination Format Register in the 82489DX Local Units.

8.0 ICC BUS DEFINITION

Physical Characteristics

The ICC bus is a 5-wire synchronous bus connecting all 82489DXs (all I/O Units and all Local Units). Four of these five wires are used for data transmissions and arbitration, and one wire is a clock. The description refers to the logical state of the ICC bus. Electrical levels are just inverse of the logical state described. For example, the section describes that the ICC bus is 0000 when not transmitting any message. This refers to logical state. Electrically, the ICC bus is 1111 when not transmitting any message.

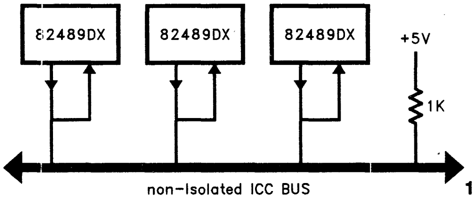
The bus is electrically an open-drain connection providing for both bus use arbitration and arbitration for lowest priority. Being open-drain, the bus is run at a "comfortable" speed such that design-specific termination tuning is not required. Furthermore, each 82489DX receiving a message or participating in an arbitration must be given enough time in a single bus cycle to latch the bus and perform some simple logic operations on the latched information in order to determine whether the next drive cycle must be inhibited.

Note that it is likely in MP systems that additional processors be located on plug-in boards. Since the ICC bus would be part of the connector, the 82489DX to ICC bus connection is defined so that it can be electrically isolated using external drivers. The 82489DX has separate ICC bus input and output pins that can be connected externally to the 82489DX to either provide or not provide isolation.

The isolation can also be used to provide a hierarchical connection of ICC buses electrically supporting large numbers of processors. The number of 82489DXs supported using the hierarchical connection is limited only by ICC bus bandwidth. It should be noted that ICC bus output low current is just 4 mA.

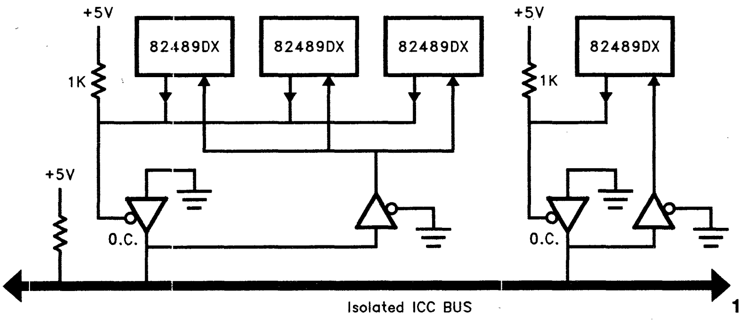
Bus Arbitration

Arbitration (both for use of the bus and for determining the lowest priority 82489DX) depends on all 82489DX message units operating synchronously. To deal with the event where multiple agents start transmitting simultaneously, a distributed arbitration approach is used. Bus arbitration uses a small number of arbitration cycles in the ICC bus. During



290446-10

Figure 20. ICC Bus: Simple Direct Connection



290446-11

Figure 21. ICC Bits: Hierachical Connection

these cycles, arbitration losers progressively drop off the bus until only the winner remains transmitting. The winner then transmits its actual inter-unit message. Once the sending of a message (including bus arbitration) has started, any possible contender must suppress transmission until enough cycles have elapsed for the message to be fully sent. The number of message cycles depends on the type of message being sent.

A bus arbitration cycle starts by the agent driving its unit ID on the ICC bus. High-order ID bits are driven first, successive cycles proceeding to the low bits of the ID. All losers in a given cycle drop off the bus, using every subsequent cycle as a tie breaker for the previous cycle. By the time all arbitration cycles are completed, there will be only a single agent left driving the bus.

The 8-bit unit ID (I7 I6 I5 I4 I3 I2 I1 I0) is chopped up in successive groups of 2 bits (I7 I6)(I5 I4)(I3 I2) (I1 I0). Each of these tuples is first decoded before driving them on the bus. The 0s and 1 indicate logical levels and not signal levels. The ICC bus is 0000 when not transmitting any message. The decoding used is:

ID Tuple		→	ICC Bus			
(I[i + 1]	I[i])		B3	B2	B1	B0
0	0	→	0	0	0	1
0	1	→	0	0	1	0
1	0	→	0	1	0	0
1	1	→	1	0	0	0

Note that the pattern generated on the ICC bus by tuple (I3 I2) will be represented as i32 i32 i32 i32. The lower case signifies this encoding.

Each tuple of the ID only contributes to a single wire, making it possible for an agent to determine with certainty whether to "drop off" or to continue arbitrating in the next cycle for the following two bits of the unit ID simply by checking whether the bus line the agent is driving is also the highest order 1 on the bus. Each ICC bus cycle therefore arbitrates 2 bits.

1:	i76	i76	i76	i76	ICC bus arbitration
2:	i54	i54	i54	i54	
3:	i32	i32	i32	i32	
4:	i10	i10	i10	i10	
	<message body>				

Lowest-Priority Arbitration

Arbitration is also used to find the 82489DX Local Unit with the lowest processor priority. Lowest-priority arbitration uses the value of the 82489DX's Processor Priority value appended with an 8-bit Arbitration ID (Arb ID) to break ties in case there are multiple units executing at the lowest priority.

Using the constant 8-bit unit ID as the Arb ID has a tendency to skew symmetry since it would favor 82489DXs with low ID values. An 82489DX Local Unit's Arb ID is therefore not the unit ID itself but is derived from it. At reset, an 82489DX Local Unit's Arb ID is equal to its unit ID. Each time a message is broadcast over the ICC bus in lowest priority mode, all 82489DX Local Units increment their Arb ID by one, which gives them a different Arb ID value for the next arbitration. The Arb ID is then endian-reversed (LSB becomes MSB, etc.) to ensure better rotation of which 82489DX gets to have the lowest Arb ID next time around. The reversed Arb ID is then decoded to generate arbitration signals on the ICC bus as described above.

To support hot insertion of processor boards in a running MP system, a mechanism is provided to allow the 82489DX of the added processor to synchronize its Arb ID with the existing 82489DXs. This is accomplished by broadcasting a message with Delivery Mode equal to "Reset", Trigger Mode equal to "Level", and Level equal to 0. This message must be broadcast before the newly added 82489DX is allowed to participate in a lowest-priority arbitration. Depending on the exact sequence under which the newly inserted board is powered-up and initialized, this Arb ID synchronization may occur naturally if a Reset-deassert to the new 82489DX is part of that sequence. If not, the local processor can always send this as an inter processor interrupt (with a null destination), causing only lhe side effect of resetting all 82489DX Arb IDs.

ICC BUS MESSAGE FORMATS

The short message format is described first. Note that the first 19 cycles of both short and long message formats have the same interpretation.

1:	i76	i76	i76	i76	ICC bus arbitration
2:	i54	i54	i54	i54	
3:	i32	i32	i32	i32	
4:	i10	i10	i10	i10	
5:	DM	M2	M1	M0	destination mode and delivery mode
6:	"0"	"0"	L	TM	control bits
7:	V7	V6	V5	V4	vector
8:	V3	V2	V1	V0	

9:	D31	D30	D29	D28	destination
10:	D27	D26	D25	D24	
11:	D23	D22	D21	D20	
12:	D19	D18	D17	D16	
13:	D15	D14	D13	D12	
14:	D11	D10	D09	D08	
15:	D07	D06	D05	D04	
16:	D03	D02	D01	D00	
17:	C	C	C	C	checksum for cycle 5 through 16
18:	"1"	"1"	"1"	"1"	post amble
19:	A	A	A	A	accept (1000 if OK, 1110 if preempt, else error)
20:	"0"	"0"	"0"	"0"	idle 1
21:	"0"	"0"	"0"	"0"	idle 2

Cycles 1 through 4 are bus arbitration as described earlier. Cycle 5 (DM M2 M1 M0) is the Destination Mode which is 0 for Physical mode and 1 for Logical Mode, and the Delivery Mode of the message. The encoding used for the Delivery Mode in the message is identical to the encoding used for the Delivery Mode in the Redirection Table, Local Vector Table, and Interrupt Command Register.

M2	M1	M0	Delivery Mode
0	0	0	Fixed
0	0	1	Lowest Priority
0	1	0	<reserved>
0	1	1	Remote Read
1	0	0	NMI
1	0	1	Reset
1	1	1	ExtINT

Cycle 6 contains the Control Bits of the message. The control bits are:

- TM (Trigger Mode): indicates whether this message corresponds to an edge or level;
- L (Level): indicates whether this is an Assert or a Deassert of a "level" signal. L is undefined when TM is edge.

6: "0" "0" L TM Control Bits
TM = Trigger Mode (0 = edge, 1 = level)
L = Level (0 = deassert, 1 = assert)

The length of the message is derived from the Delivery Mode, the Control Bits, and the Accept cycle of the message.

TM/L (AAAA)

	Edge	Level = Assert	Level = Deassert
Fixed	Short	Short	Short
Lowest Priority	Short (1110)	Short (1110)	Short (1110)
	Long (1000)	Long (1000)	Short
Remote Read	Long	Long	Short
NMI	Short	Short	Short
Reset	Short	Short	Short
ExtINTA	Short	Short	Short

Cycles 7 and 8 are the 8-bit interrupt vector. The vector is only defined for Delivery Modes Fixed, and Lowest-priority. For Delivery Mode of "Remote Read", the vector field contains the address of the register to be read remotely.

If DM is 0 (physical mode), then cycles 9 and 10 are the unit ID and cycles 11 through 16 are zero. If DM is 1 (logical mode), then cycles 9 through 16 are the 32-bit Destination field. The interpretation of the logical mode 32-bit Destination field is performed by the Local Units using the Destination Format Register. The sending 82489DX knows whether it should (incl) or should not (excl) respond to its own message.

Cycle 17 is a checksum over the data in cycles 5 through 16. The checksum is computed by adding all 4-bit quantities of cycles 5 through 16, feeding carry out of the MSB back into the LSB. This protects the data in these cycles against transmission errors. The (single) 82489DX driving the message provides this checksum in cycle 17.

Cycle 18 is a post amble cycle driven as 1111 by the sending 82489DX allowing all 82489DXs to perform various internal computations based on the information contained in the received message. One of the computations takes the computed checksum of the data received in cycles 5 through 16 and compares it against the value in cycle 17. If any 82489DX computes different checksum than the one passed in cycle 17, then that 82489DX will signal an error on the ICC bus in cycle 19 by driving it as 1111. If this happens, all 82489DXs will assume the message was never sent and the sender must try sending the message again, which includes re-arbitrating for the ICC bus. In lowest priority delivery when the interrupt has a focus processor, the focus 82489DX will signal this by driving 1110 during cycle 19. This tells all the other 82489DXs that the interrupt has been accepted, the 82489DXs is preempted, and short message format is used. All (non-focus) 82489DXs will drive 1000 in cycle 19. Under lowest priority mode, 1000 implies that the interrupt currently has no focus processor and that priority arbitration is required to complete the delivery. In that case, long message format is used. If cycle 19 is 1000 for non Lowest Priority mode, then the message has been accepted and is considered sent.

19:EEEE	
1000	OK
1110	preempt
<others>	error (drive error as 1111)

When an 82489DX detects and reports an error during the error cycle, that 82489DX will simply listen to the bus until it encounters two consecutive idle (0000) cycles. These two idle cycles indicate that the message has passed and a new message may be started by anyone. This allows an 82489DX that got itself out of cycle on the ICC bus to get back in sync with the other 82489DXs.

Long Message Format

Cycles 1 through 19 of the long message format are identical to cycles 1 through 19 of the short message format. As mentioned, long message format is used in two cases:

(1) Lowest Priority delivery when the interrupt does not have a focus. Cycles 20 through 27 are eight arbitration cycles where the destination 82489DXs determine the one 82489DX with lowest processor priority/ARB ID value.

(2) Remote Read messages. Cycles 20 through 27 are the 32-bit content of the remotely read register. This information is driven on the bus by the remote 82489DX.

Cycle 28 is an Accept cycle. In lowest priority delivery, all 82489DXs that did not win the arbitration (including those that did not participate in the arbitration) drive cycle 28 with 1000 (co accept), while the winner 82489DX drives 1111. If cycle 28 reads 1111, then all 82489DXs know that the interrupt has been accepted and the message is considered delivered. If cycle 28 reads 1100 (or anything but 1111 for that matter), then all 82489DXs assume the message was unaccepted or an error occurred during arbitration. The message is considered undelivered, and the sending 82489DX will try delivering the message again.

For Remote Read messages, cycle 28 is driven as 1100 by all 8 2489DXs except the responding remote 82489DX, who drives the bus as 1111 in case it was able to successfully supply the requested data in cycles 20 through 27. If cycle 28 reads 1111 the data in cycles 20 through 27 is considered valid; otherwise, the data is considered invalid. The source 82489DX that issued the Remote Read uses cycle 28 to determine the state of the Remote Read Status field in the Interrupt Command Register (valid or invalid). In any case, a Remote Read request is always successful (although the data may be valid or invalid) in that a Remote Read is never retried. The reason for this is that Remote Read is a debug feature, and a "hung" remote 82489DX that is unable to respond should not cause the debugger to hang.

Cycles 29 and 30 are two idle cycles. The ICC bus is available for sending the next message at cycle 31. The two idle cycles at the end of both short and long messages, together with non zero (i.e., non idle) encoding for certain other bus cycles allow an ICC bus agent that happens to be out of phase by one cycle to sync back up in one message simply by waiting for two consecutive idle cycles after reporting its checksum error. This makes use of the fact that valid arbitration cycles are never 0000.

1:	i76	i76	i76	i76	ICC bus arbitration
2:	i54	i54	i54	i54	
3:	i32	i32	i32	i32	
4:	i10	i10	i10	i10	
5:	DM	M2	M1	M0	delivery mode
6:	"0"	"0"	L	TM	control bits
7:	V7	V6	V5	V4	vector
8:	V3	V2	V1	V0	
9:	D31	D30	D29	D28	destination
10:	D27	D26	D25	D24	
11:	D23	D22	D21	D20	
12:	D19	D18	D17	D16	
13:	D15	D14	D13	D12	
14:	D11	D10	D09	D08	
15:	D07	D06	D05	D04	
16:	D03	D02	D01	D00	
17:	C	C	C	C	checksum for cycles 5 through 16
18:	"1"	"1"	"1"	"1"	postamble
19:	A	A	A	A	accept (1000 if OK, 1110 if preempt, else error)
20:	p76	p76	p76	p76	lowest priority arbitration or 32 bits of remote register
21:	p54	p54	p54	p54	processor priority
22:	p32	p32	p32	p32	
23:	p10	p10	p10	p10	
24:	a76	a76	a76	a76	
25:	a54	a54	a54	a54	arbitration ID
26:	a32	a32	a32	a32	
27:	a10	a10	a10	a10	
28:	A	A	A	A	accept
29:	"00"	"0"	"0"	"0"	idle1
30:	"0"	"0"	"0"	"0"	idle2

9.0 HARDWARE TIMINGS

This section covers the following:

— Timing Diagram Notation

— 82489DX Register Access Timing Diagrams with Descriptions

A block diagram of the configuration of the CPU module of a MP system is shown. This in no way is intended to be a complete representation of 486/Intel Cache/Intel Cache Controller connections. It is intended to show all the 82489DX connections, and how they connect to other components on and off the module. This module has arbitrarily been drawn with a 64-bit data bus to show how the expanded address space architecture fits. The unit can be similarly attached to either a 32-bit or 128-bit data bus, with total transparency to shrink-wrap software.

In this configuration, the 82489DX uses the same clock source as the processor and cache. However, it is quite possible to consider 82489DX as a memory bus device and hence supply 82489DX with the memory bus clock, which can be slower than the CPU module clock frequency.

In the configuration shown, the processor's INT and NMI pins could be supplied by other source to allow for the possibility that 82489DX can be totally bypassed if desired, by allowing those signals to be driven from off the module while the 82489DX is disabled. The reset signal generated by the 82489DX goes to the MBC (memory bus controller) which is required to drive configuration lines at reset time. This would probably be configured as a "warm" reset by the MBC.

A future version of cache controller may generate the chip select for 82489DX at a fixed memory location of hexFEE00000. By having the cache controller to provide the chip select signal, it would encourage a standard mapping for 82489DX address space. In some MBC designs, this signal should be connected to the MBC since 82489DX cycles limit bus pipelining by constraining how soon the next bus cycle can come. The 82489DX chip select can be generated by the MBC completely.

The address, data and most of the bus control signals share the respective bus with cache and cache controller. The block diagram shows attachment for only 6 address lines: A4–A9. A10 should be 0. This is all the 82489DX needs for operation, however, if the address lines are used to initialize 82489DX local ID at reset time, 8 address lines are required, A3–A10.

INTERFACING TO THE ICC BUS

The 82489DX has separate ICC bus input and output pins to facilitate using external drivers. The ICC bus input pins (MBI0-3) are TTL-level compatible CMOS inputs. The output pins (MBO0–3) are open-drain pins which required external pull-ups. The open-drain output buffers are small buffers with:

> Sink current of < 4 mA. Special consideration must be exercised when driving large capacitive loads or long transmission lines. The pull-up resistor and the capacitive load constitute RC time constant that will affect the output transition times. This in turn will limit the operating frequency of the ICC bus.
>
> When designing in the ICC bus, one needs to consider the loads that each 82489DX will be driving and whether external drivers should be used. In most situations, the ICC bus driven high (MBO pins pulled high by the external pull-up resistors) poses the most challenge. Simulating the target design on an electrical simulator (such as SPICE) will help greatly. as shown in the following examples.

First Order Buffer Models

Figure 21a and 21b are first order input buffer and output buffer models of the MBI and MBO pins. The open-drain of the MBO is modeled as a switch as the primary interest here is the MBO pins going high. These models can be used on SPICE simulations to obtain first order behaviors. The parameters for these models are as follows:

Cp (package capacitance) = 3 pF
Lp (package inductance) = 15 nH
Rb (bond wire resistance) = 0.08Ω
Ci (input buffer capacitance) = 3 pF
Co (output buffer capacitance) = 6 pF
Ro (output buffer impedance) = 30Ω–80Ω

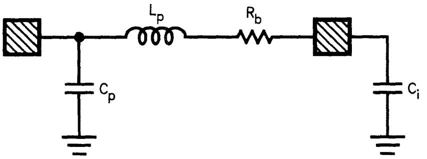
MBO Pull-up Resistor

To minimize the RC time constant, one would like to use the smallest pull-up resistor value possible. The MBO pins has a worst case Iol-spec of 4 mA and V_{CC} = 4.75V. This translates to a minimum pull-up of about 1 KΩ. Where stronger drive is needed (smaller pull-up resistors), external drivers must be used.

Driving Lumped Capacitance

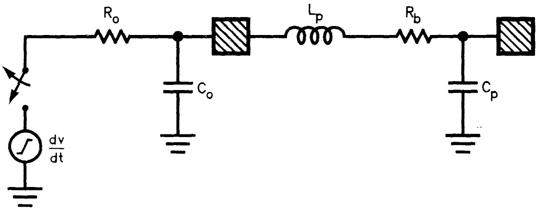
In systems where external drivers are not used, the MBI pins will be tied to the MBO pins. Figure 21d is a SPICE simulation of the MBO output with a 1 KΩ pull-up driving lumped capacitive loads from 10 pF to 150 pF.

At a load of 50 pF, it takes about 30 ns to charge up to 2V. At 100 pF, it takes an additional 25 ns. Figure 21d can be used to estimate the loading delay at different lumped capacitive loads.



290446-29

Figure 21a. First Order Input Buffer for MBI Pins



290446-30

Figure 21b. First Order Open-Drain Output Buffer for MBO Pins

In real systems, the loads are made up of lumped capacitance and transmission lines. More accurate results can be obtained using transmission line models.

Driving Transmission Lines

Two device model

In this example the ICC bus is a signal line on an FR-4 printed circuit board. The line width is 6 mils. Line length of 12 inches and 18 inches are modeled. The FR-4 PC board has the following characteristics:

resistivity =	0.6 mΩ/sq. (0.1Ω/inch for 6 mil width)
inductance =	60 pH/sq. (10 nH/inch for 6 mil width)
capacitance =	0.55 nF/sq. in. (3.3 pF/inch for 6 mil width)

The ICC bus is shared by two 82489DXs, one at each end. The ICC bus is modeled as a transmission line. For the simulation, only one of the 82489DX is driving. A pull-up resistor of 2 KΩ is used at each end (1 KΩ equivalent value) as shown in Figure 21e. Figure 21f shows the signals at each end of the 12 inch transmission line. Trace 1 is the wave form at the driven end and trace 2 is the signal at the receiving end of the line. The 2 ns delay between the two signals is the propagation delay (or flight time) through the 12 inch transmission line. It takes about 35 ns for the voltage to charge up to 2V.

Figure 21g shows the received signal with different line length and with additional lumped capacitance. Trace 1 is for 12 inch only. Trace 2 is for 12 inch with additional 20 pF lumped capacitance to represent interconnect socket capacitance. Trace 3 is for 18 inch plus 20 pF. The presence of the 20 pF at each end of the 12-inch transmission line increases the delay time by 20 ns at 2V.

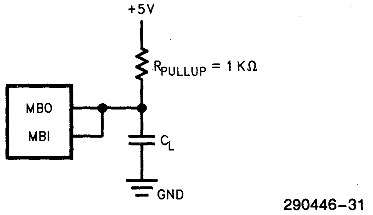


Figure 21c. ICC Bus Driving Lumped Capacitance

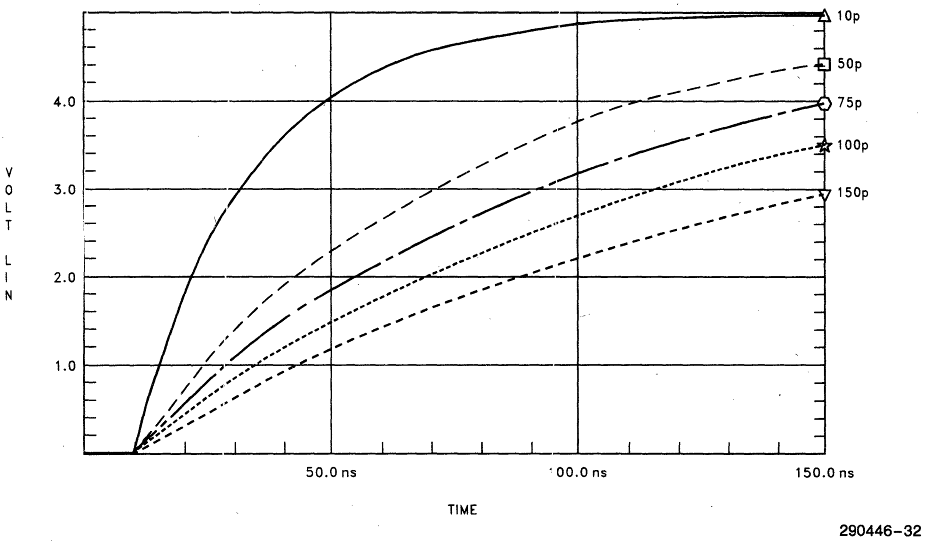


Figure 21d. 1 KΩ Pull-Up Driving Lumped Capacitance

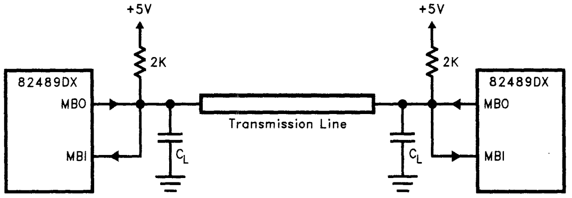
Four Device Model

In this example (Figure 21h), the ICC bus is a 12-inch transmission line with four 82489DXs connected at 4 inch intervals. The loading at each junction consisted of the MBI and MBO buffers and a 20 pF lumped capacitance. 2 KΩ pull-ups are at each end of the transmission line.

As shown in Figure 21i, it takes more than 90 ns for the signal level at both ends to reach 2V.

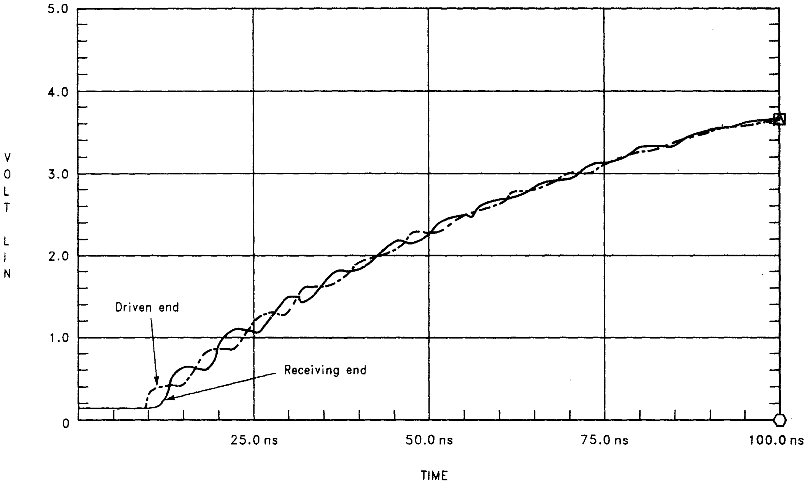
One way to improve the low to high transition time is to use a stronger pull-up (smaller resistor value) which is possible using external line drivers with their larger current drive capabilities.

Figure 21j shows the difference in output when the model is used with 300Ω pull-ups at each end of the transmission line.



290446-33

Figure 21e. Unbuffered ICC Bus with Two 82489DX



290446-34

Figure 21f. Waveform at Both Ends of 12″ T-line

External Drivers/Buffered ICC bus

The 82489DX has separate ICC Bus input (MBI) and output (MBO) pins that can be connected to external line drivers in systems that has appreciable loading on the ICC Bus or where modularity of the bus is needed.

Figure 21k is a typical implementation using external drivers with tri-state outputs. Drivers such as 74F125 or its equivalent can be used. The drivers should be placed as close to the MBO pins as possible. The input buffer on MBI is optional depending on the users ICC Bus scheme. The total delay through the drivers, buffers, transmission line, clock skews etc. must be calculated to ensure that all the ICC bus timing requirements are met.

A hierarchical bus connection can also be used in applications that cannot afford driver/buffer per unit and where bus loading are localized in cluster groups. Figure 21l shows such a connection where each cluster group is connected directly and drivers are used to connect to other clusters. Each cluster group is assumed to be close together physically with small loading on the local ICC bus.

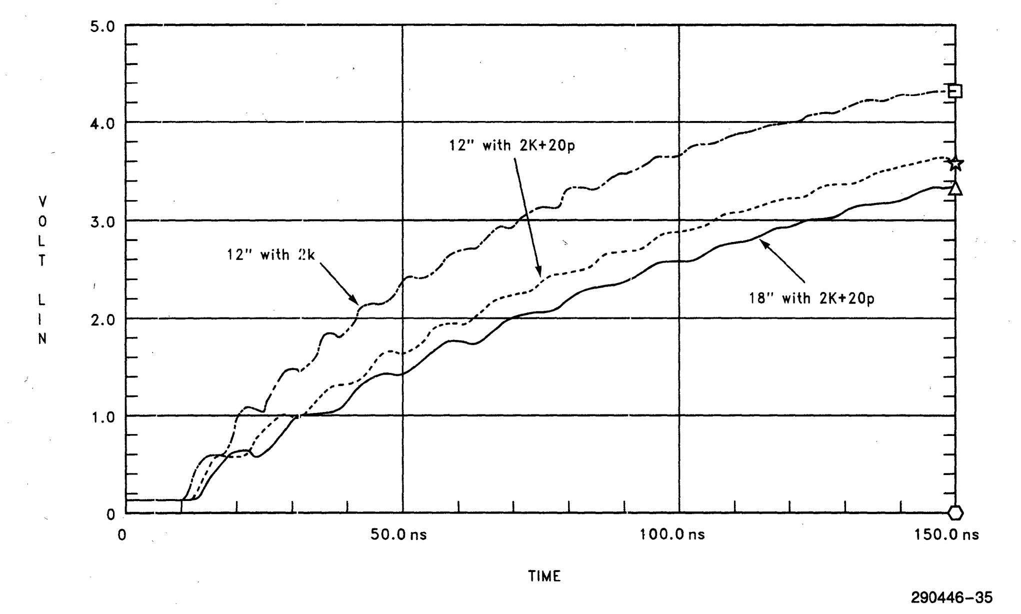


Figure 21g. Waveform at End of T-line with Load

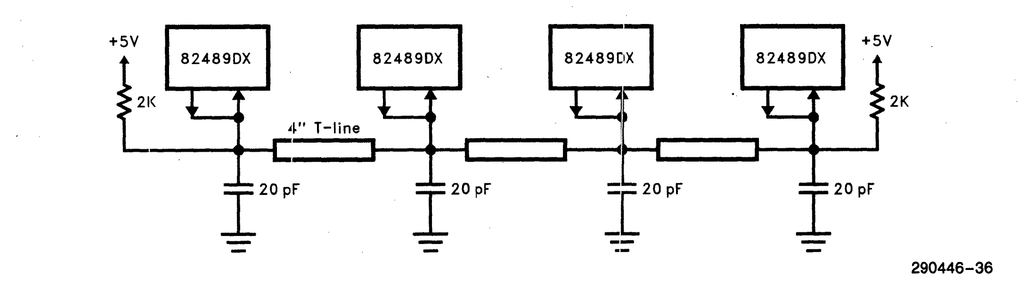


Figure 21h. Four 82489DX Configuration

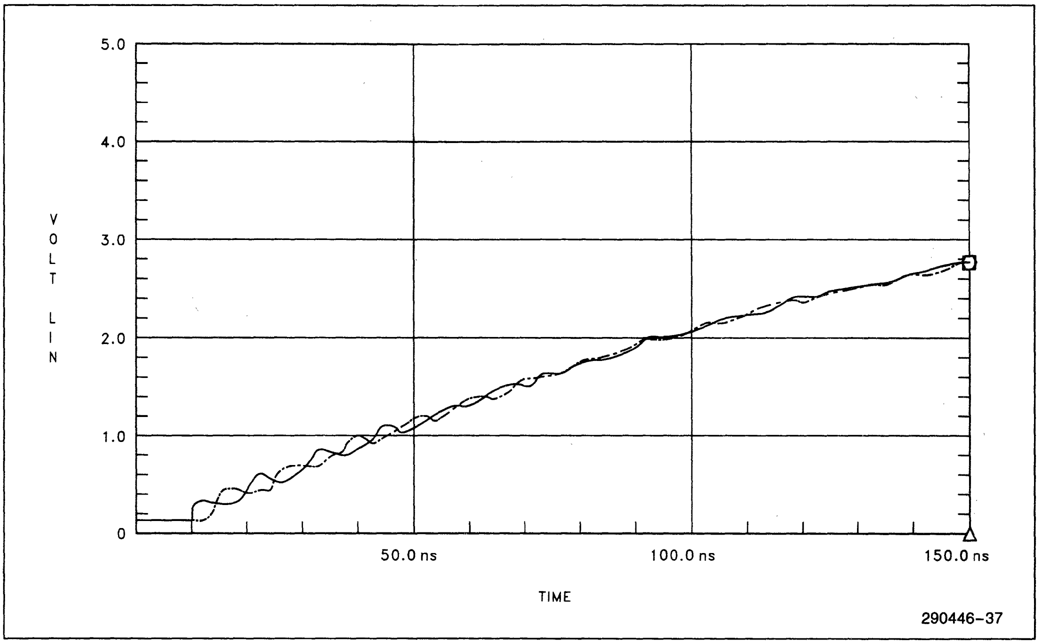


Figure 21i. Waveform for Four Devices on 12″ T-line

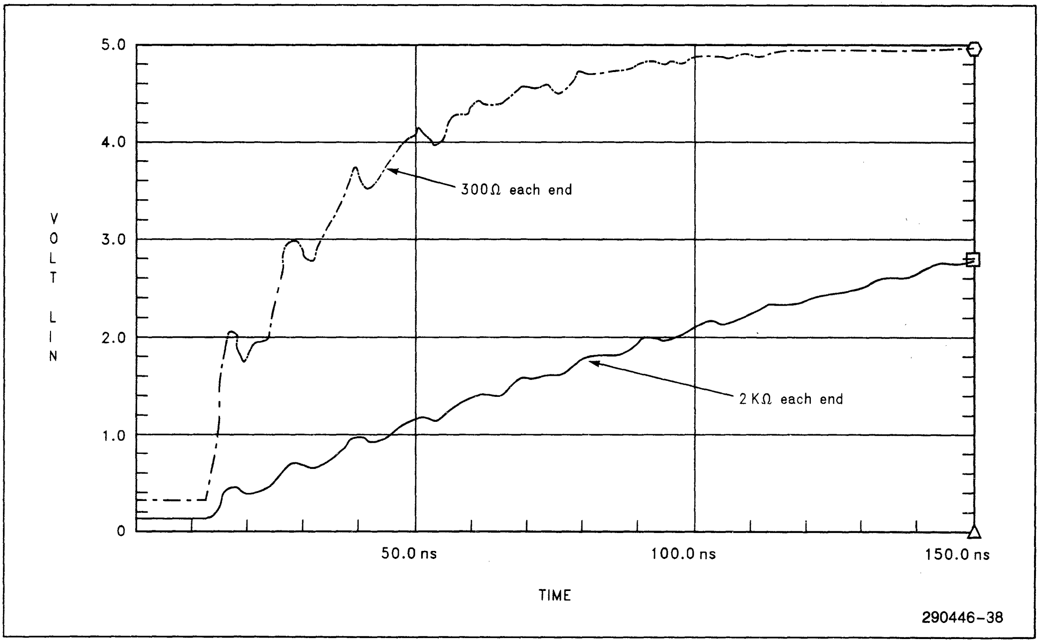


Figure 21j. Waveform with Different Pull-Ups

Transmission Line Termination

As with all high speed designs, one has to consider transmission line effects on signals, especially clock signals. Even though the ICC bus clock, ICLK is usually operated in the 10 MHz range, one has to consider proper transmission line termination also for short rise times. Figure 21m shows the ICLK wave form at the end of a 12 inch T-line when driven by a clock generator with and without series matched termination.

Series termination should not be used for the ICC bus data lines (MBO). The combination of the pull-up resistor and series resistor would degrade the output low voltage, Vol. For example, with a pull-up of 300Ω and a series termination of 50Ω at each end, the Vol voltage at the receiving end would be at 1.55V if the driving end is at 0.4V (see Figure 21n).

ICC BUS Operating Frequency

The 82489DX ICC BUS has a design target of operating up to 16 MHz (62 ns period). As shown in the examples above, the MBO low-to-high transition times are strongly dictated by the loads and the pull-ups used. This will in turn affect the maximum operating frequency of ICLK.

In general, the minimum period is the larger of 62 ns or MBO-to-MBI low data time or MBO-to-MBI high data time.

MBO-to-MBI low time =
(ICLK skew + MBO valid low delay + T-line prop.delay + ext. buffer delay + MBI setup time)

MBO-to-MBI high time =
(ICLK skew + MBO Hi-Z delay + pull-high time + T-line prop.delay + ext. buffer delay + MBI setup time)

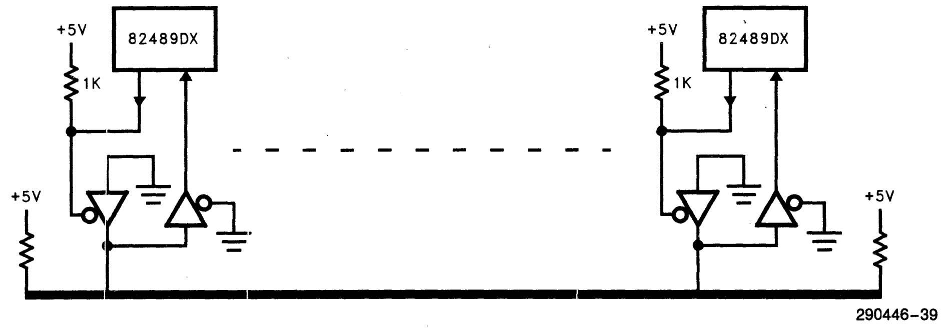
Maximum MBO valid low delay = 50 ns
Maximum MBO H-Z delay = 15 ns
MBI minimum setup time = 8 ns

In the example shown earlier where two 82489DXs are at each end of a 12-inch T-line with no other loads, the pull-high time to 2V is 35 ns (trace 1 in Figure 21g). If the ICLK skew is 2 ns, then this configuration can operate to 62 ns period or 16 MHz.

If the same configuration has additional 20 pF loads at each end, then the pull-high time is 55 ns (trace 2 in Figure 21g). The maximum frequency decreases to 12 MHz (82 ns period).

In the four device model discussed earlier, where the ICC Bus is unbuffered, the pull-high time is 90 ns (Figure 21j). The operating frequency will be less than 8 MHz (117 ns period). If external buffers are used (whereby allowing use of 300Ω pull-up) and assuming the external buffers have delays of 10 ns, the operating frequency is limited by the MBO-to-MBI low time of 72 ns or 14 MHz.

NOTE:
Each application is unique in its configuration and loading on the ICC Bus. The above examples highlighted some of the factors that need to be considered. It is important to do electrical simulation to ascertain if the propose implementation is viable before committing to the printed circuit board.



Note:
Input buffer is optional

Figure 21k. External Driver/Buffer Implementation

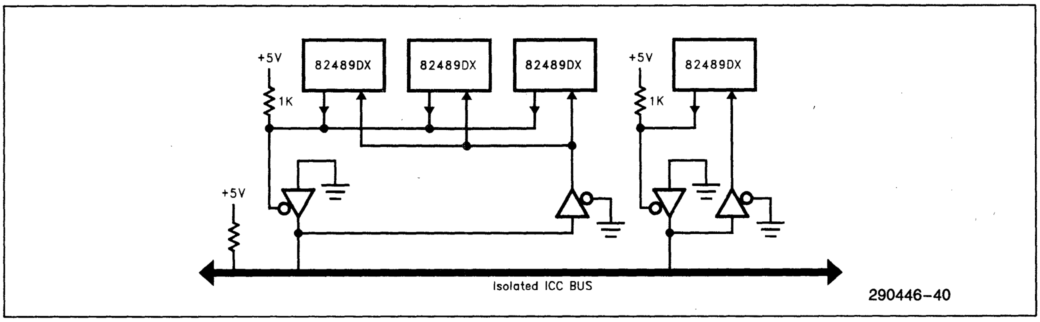


Figure 21l. ICC Bus: Hierarchical Implementation

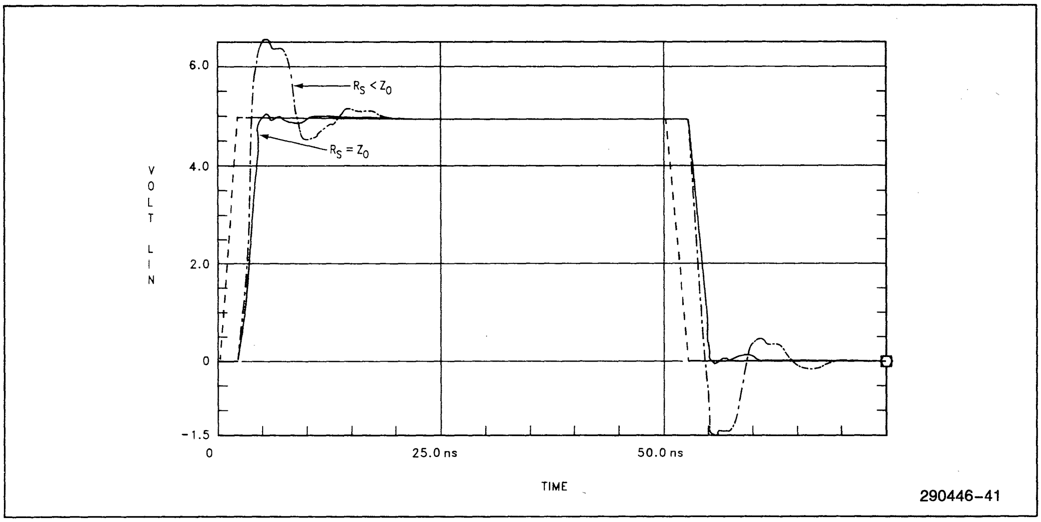


Figure 21m. ICLK Waveform on 12″ T-line

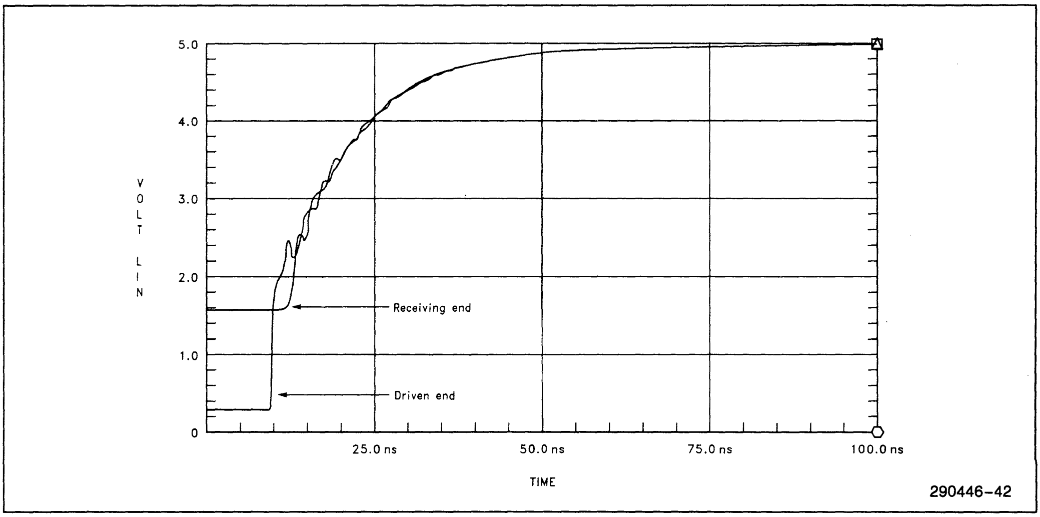


Figure 21n. Effect of Series Termination on MBO VOL

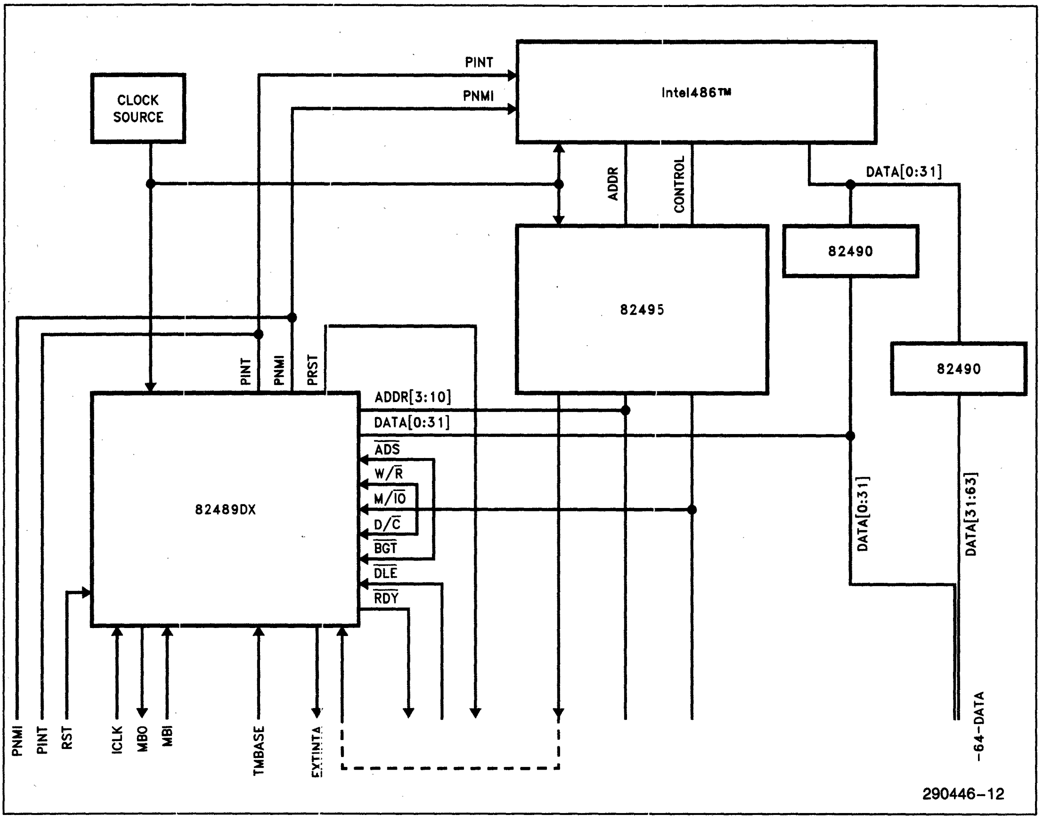


Figure 22. Possible Configuration of the CPU Module

9.1 82489DX Register Access Timing

This section provides descriptions of the four basic cycle timings for the 82489DX, which are:

- Register Write
- Register Read
- Interrupt Acknowledge
- Reset

In addition, Inter-Unit (nibble) bus timings are presented.

Register access occurs in three distinct phases:

1. Control Phase,
2. Address Phase and
3. Data Phase.

They always occur in this order, although in some cases address and data phases can occur in the same clock cycle, as will be seen in the diagrams.

NOTE:
As mentioned previously, the clock signal in all these timing diagrams is assumed to be the processor clock.

TIMING DIAGRAM NOTATION

The 82489DX bus (register access) interface is synchronous. Unless otherwise noted, setup and hold times for inputs, and delay times for outputs are measured with respect to a rising clock edge. Therefore the timing diagrams contain very few construction lines to identify timing parameters and ones that exist are explicitly discussed. High and low level for signals are obvious; dashed lines at the middle of the signal range indicate a tristate output buffer in high impedance mode; hashed or cross-hatched lines indicate a "don't care" state for inputs.

Cycle expansion marks—short curved lines breaking the waveform—appear throughout. These appear in sets or groups which are identified by construction arrows and/or vertical column alignment. Conditions resulting in cycle expansion are listed at the bottom of each diagram, and the associated set of expansion marks indicates which signals must be stretched for that condition. Signals not so indicated are not affected by that condition, and continue without any cycle stretching. For example, the data phase of a transaction may be delayed, stretching all data related signals, while address related signals can continue to the next cycle.

Sample points for input signals are marked with bold, downward pointing arrows. Since sample timing for address, data and cycle definition signals is dependent on the timing of related control signals, a bar is used on top of the arrow to indicate the signal with the independent timing. Each signal group has exactly one independent signal. In general, independent signals are sampled on every clock, and therefore must meet setup and hold times on every clock edge. Signals having dependent timing, (indicated by the arrow with no bar), are only sampled when the associated independent signal is active, and therefore setup and hold times for dependent signals need only be met at the indicated sample points.

REGISTER WRITE TIMING

For discussion of this bus cycle, refer to Figure 23, 82489DX Timing Diagram 1. This shows the relationship between the three phases of the bus cycle.

The control phase is independently timed by the ADS signal. The cycle definition signals [M/IO, D/C, are dependently sampled with ADS as indicated by the bold sample point arrows labeled "C". The cycle definition signals will be sampled in the first clock when ADS is active (low). The control signal should remain stable until ADS goes inactive. For any valid 82489DX cycle, the memory bus controller should ensure that the ADS pulse for a subsequent bus cycle is NOT presented until after the 82489DX asserts its RDY pin (low) as shown.

The address phase is independently timed by the (BGT) signal, as indicated by the bold sample point arrows labeled "A". This signal is actually used an address latch enable, however, its name is intended to imply that in most cases it can be directly driven by the Intel cache controller signal of the same name. The BGT pulse may be delayed until the address bus is available, in which case all address and data phase signals will be stretched. Note that DLE must not occur before BGT. 82489DX does not start the internal cycle until BGT is recognized with the appropriate chip select signal, If multiple ADS has been issued without BGT and a valid chip select, the internal cycle starts with the most recent ADS cycle definition preceding the BGT with valid chip select.

NOTE C:
Address information, including chip select (CS), is sampled in the first clock when BGT is active (low), and they must remain stable until BGT goes inactive, OR until RDY is asserted (low), whichever occurs first. CS should be stable when ADS is active.

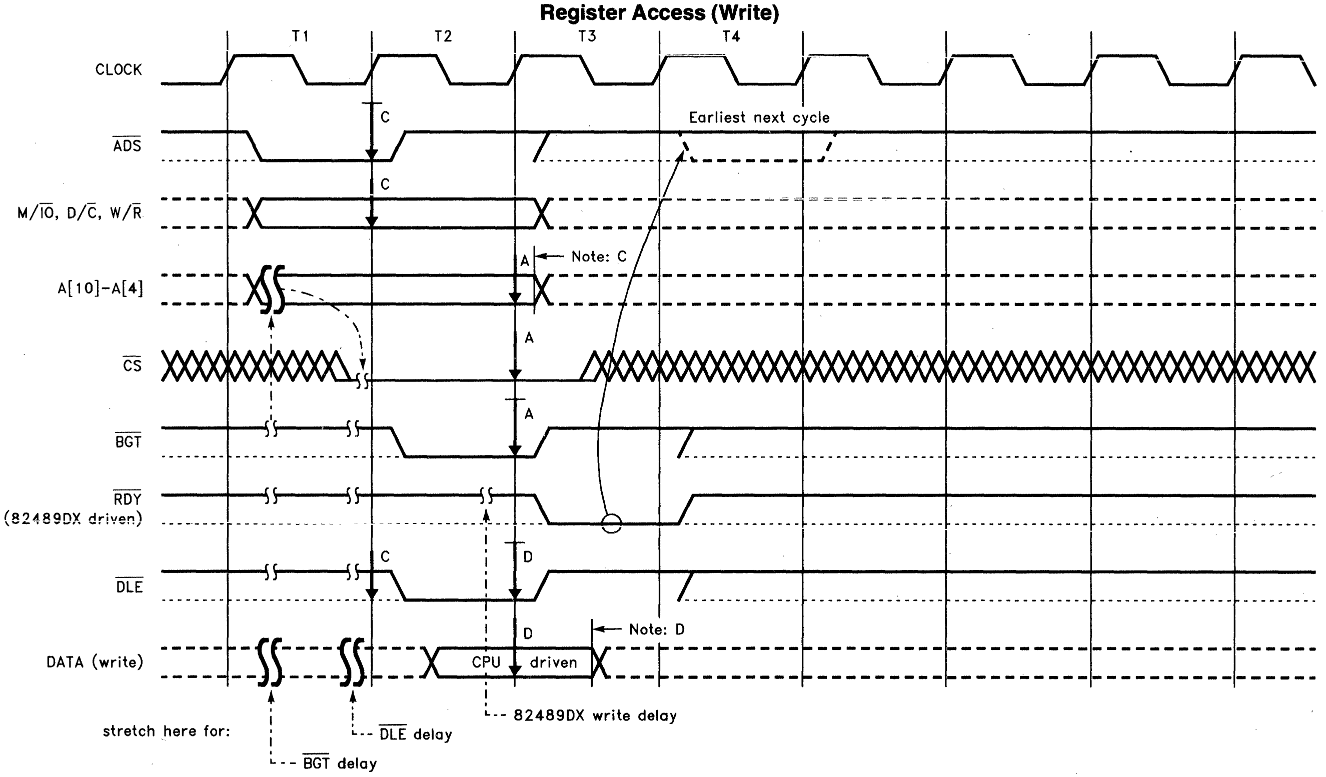
In some çonfigurations, BGT may not be provided, and can be permanently tied low. In this case, the independent address timing will occur exactly one clock after the ADS signal is first sampled low, and the dependent address information (address and chip select) will be assumed stable at this time. 82489DX recognize that independent BGT timing is not provided by sampling a low state of BGT at the time ADS is first sampled low.

The data phase is independently timed by the DLE signal, as indicated by the bold sample point arrows labeled "D". In the case of register writes, this signal work logically like a synchronous data latch enable. The DLE pulse may be delayed until the data bus is available, in which case data and RDY will be stretched.

NOTE D:
Write data are sampled the first clock when DLE is active (low), and should remain stable until DLE goes inactive, OR until RDY is asserted (low), whichever comes first.

In some configurations, DLE may not be provided, and can be permanently tied low. In this case, the independent data timing will occur exactly one clock after the ADS signal is first sampled low, OR on the same clock as BGT is first sampled low, whichever occurs later. The data bus will be assumed stable at this time. 82489DX recognize that independent DLE timing is not provided by sampling a low state of DLE at the time ADS is first sampled low.

Cycle completion is signaled by the RDY signal. Its relative positioning on any of the timing diagrams does NOT imply the number of clock cycles required for an access. RDY is delayed as needed in order for the 82489DX to complete the cycle. It is then asserted (low) for one clock cycle and then deasserted. Again, the next ADS cannot start until after RDY has been driven low. ADS must return to an inactive high state before the next cycle can be issued. It is highly recommended not to have, ADS more than one clock wide.



290446-13

Figure 23. Timing Diagram 1.

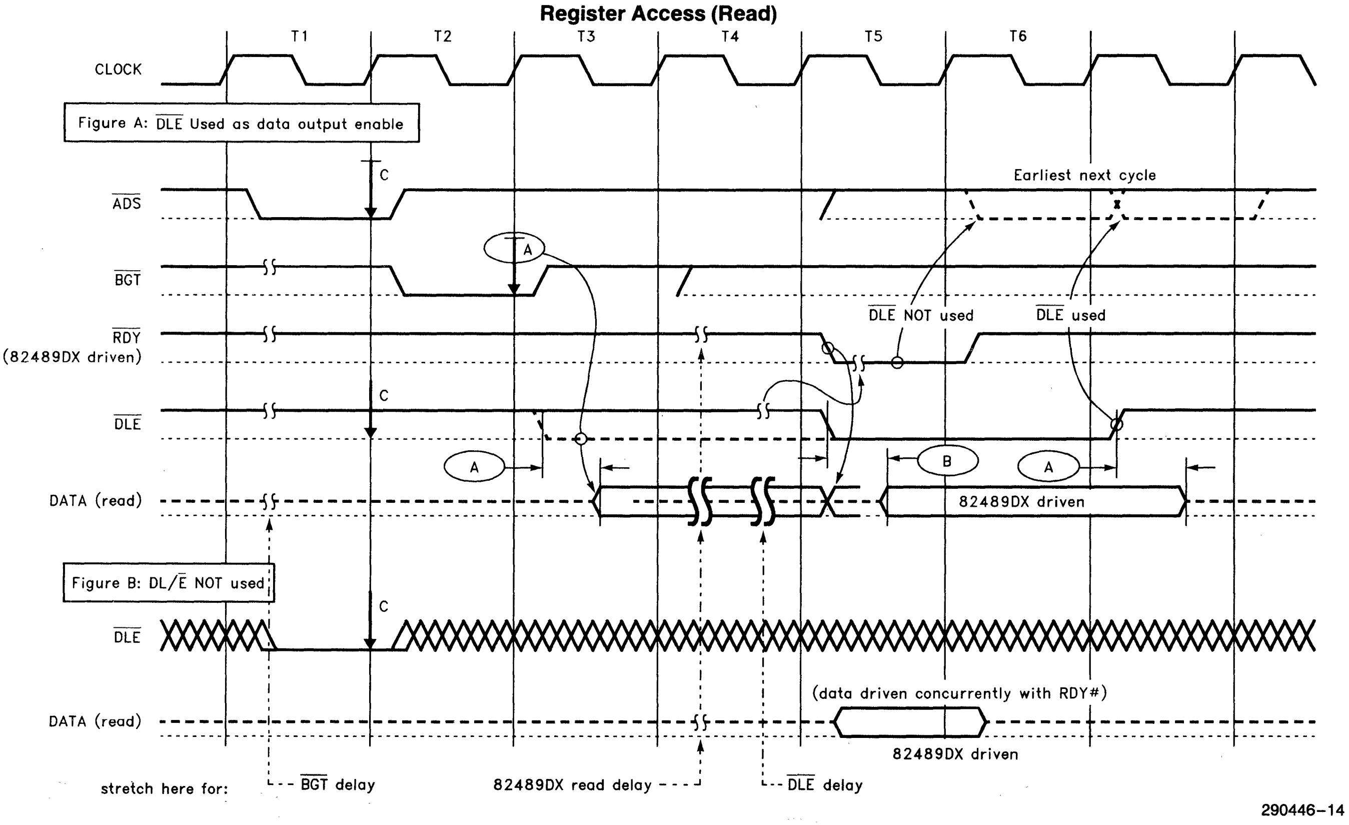


Figure 24. Timing Diagram 2

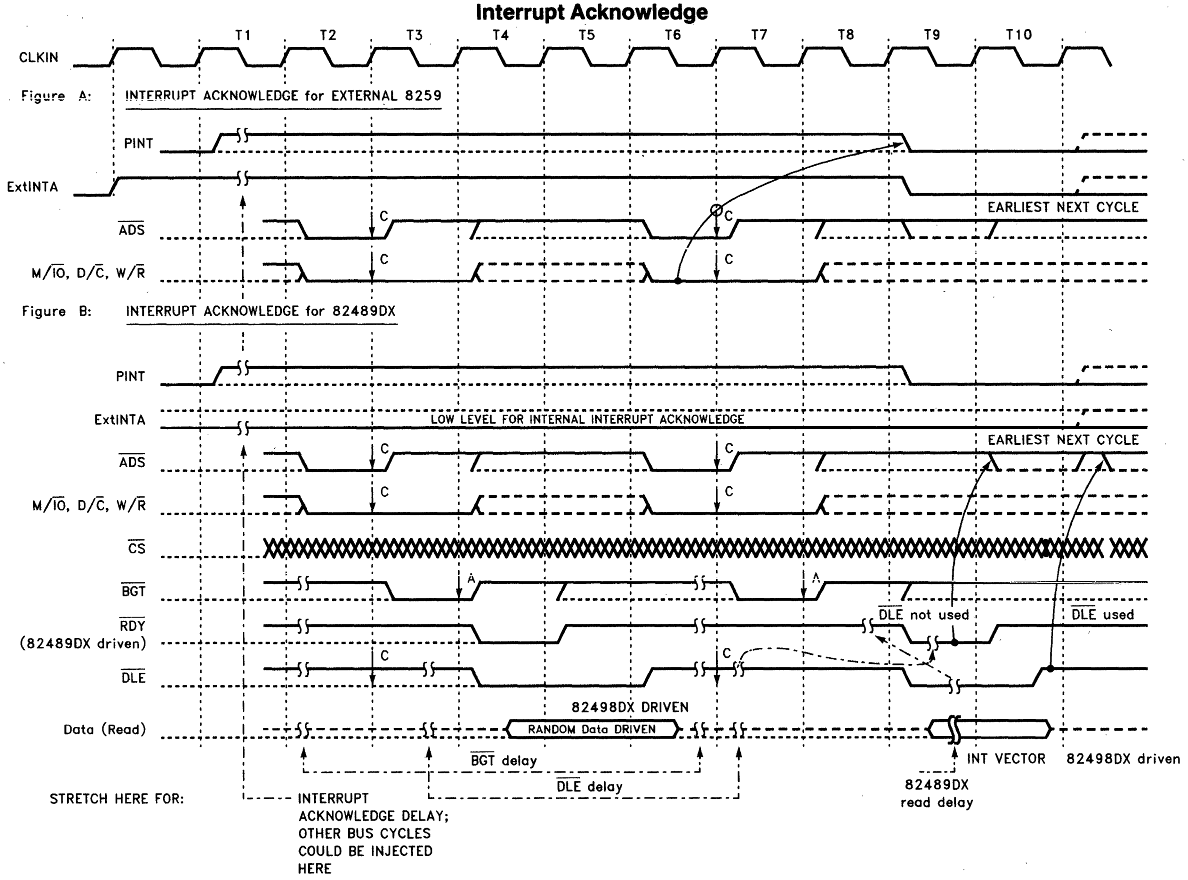
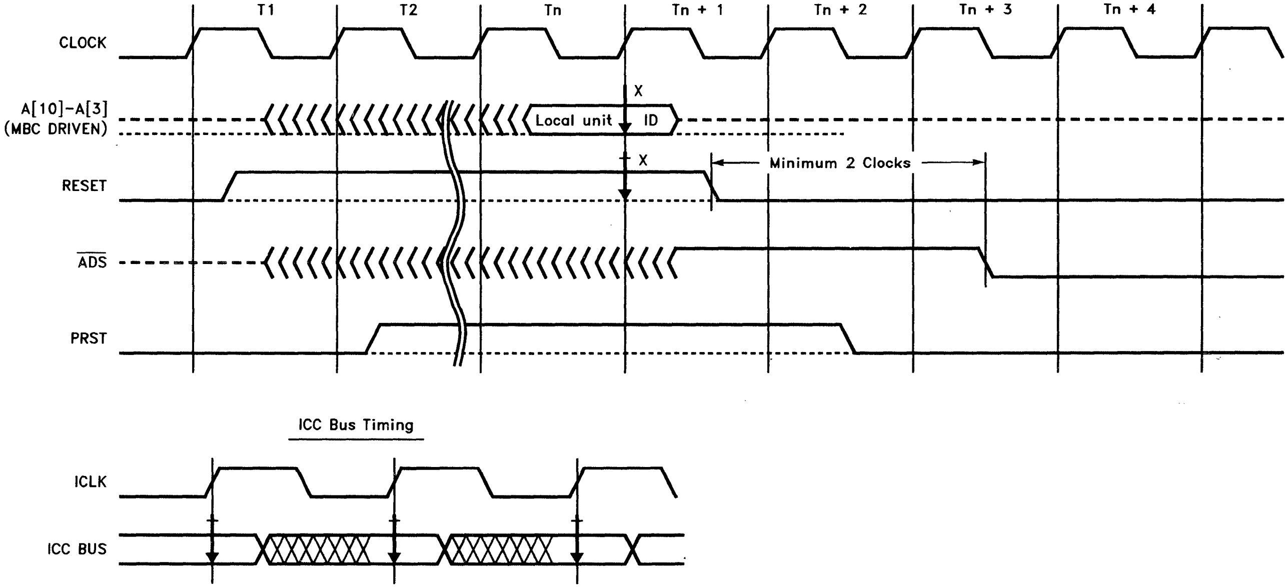


Figure 25. Timing Diagram 3

ADVANCE INFORMATION



290446-16

Figure 26. Timing Diagram 4

REGISTER READ TIMING

For discussion of this bus cycle, refer to Figure 24, 82489DX timing Diagram 2. It shows the relationship of the three phases of the bus cycle, however, the dependent control and address signals are not shown here, since they behave exactly as in the case of a register write. See the previous section for the description of control and address phases of the bus cycle.

In the case of a read when DLE is used (Figure 24A), it works logically like an asynchronous, output data enable. The 82489DX drives the data bus within time delay "A" after DLE is asserted, which must not occur before BGT. Note that even though the bus is being driven, the data only becomes valid during the clock cycle in which RDY is asserted, after that point, valid data is maintained on the bus as long as DLE remains asserted, after which the data bus returns to high impedance state within time delay "B" of DLE deassertion. If DLE is asserted late, 82489DX could complete its internal read cycle and return RDY early. In that case, RDY active low state will be maintained until DLE is asserted.

In the case of a read when DLE is NOT used (Figure 24B) the data is driven for exactly one clock cycle, coincident with RDY being asserted.

DLE is sampled with the control signals to determine whether it is being used. If sampled in the asserted state when ADS is active, the DLE will be considered not used, and its state during the remainder of the cycle doesn't matter. This is consistent with the notion of permanently tying this signal low when not used, as described in the previous section.

Indication of the end of the bus cycle is dependent on the use of DLE. When it is not used, (i.e., permanently tied to ground) RDY indicates the end of the bus cycle, as it does in the case of write access. When it is used, DLE deassertion indicates the end of the cycle, since the 82489DX could be driving the bus well after RDY is deasserted. In either case, the 82489DX can accept the next ADS pulse anytime after RDY has been asserted. However, note that if DLE is being used, the next ADS should be delayed until DLE can be safely sampled inactive. Note these two options for earliest next cycle in the timing diagram.

INTERRUPT ACKNOWLEDGE TIMING

For discussion of this bus cycle, refer to Figure 25 82489DX timing diagram 3, This cycle is the result of the 82489DX posting an interrupt to the processor by asserting PINT. After PINT is asserted, other bus cycles may occur before the interrupt acknowledge cycle.

PINT can be asserted for any external (8259) interrupt as shown in Figure 25A, or for an 82489DX generated interrupt as shown in Figure 25B. ExtINTA indicates whether PINT was asserted in response to an 8259 request or an 82489DX request. This signal is used by external control logic to either allow or preclude the 8259 from responding to the subsequent interrupt acknowledge cycle. It should be noted that ExtINTA pin gets deactivated at third clock after the ADS of the second INTA cycle.

When ExtINTA is high, the 82489DX will not respond to the acknowledge cycle other than to deassert PINT signal, and clear the pending external interrupt two full clock cycles after the ADS of the second INTA cycle, as shown in Figure 25A. Once PINT is asserted in response to an external interrupt, it can only be deasserted by an INTA cycle. An INTA cycle is recognized by 82489DX as soon as the bus cycle definition is sampled with ADS in a low state. BGT is not needed in this case.

When ExtINTA is low, the 82489DX will respond to the acknowledge cycle, as shown in Figure 25B. In this case, external logic (e.g., the memory bus controller) is expected to prevent any attached 8259 from seeing the acknowledge cycle. When PINT is asserted in response to an 82489DX internal interrupt, it can be deasserted by an INTA cycle. The PINT signal will be deasserted 5 full clocks after BGT of the second INTA cycle.

Note that ExtINTA is stable at all times while PINT is asserted. That means that even if new interrupts arrive between the time an interrupt is posted to the processor, and the acknowledge occurs, the 82489DX will not change its commitment for an external (8259) or internal (82489DX) acknowledge cycle, regardless of priority. This also means that PINT may be raised for a high priority internal interrupt right after responding to the external Interrupt. In any event, PINT will be kept low for a minimum of two clocks before reasserting itself.

The interrupt acknowledge cycle is indicated by the bus cycle definition signals all being low, and looks like two consecutive read cycles, except that there is no explicit address information. The actual content of the address pins during this cycle is processor dependent, and therefore there is no chip select either. Chip select is implied by a combination of the bus cycle definition signals (all low) and BGT.

Note that there is a "dummy" data phase in the first interrupt acknowledge cycle. This allows parity to be generated on the bus for processors like i860XP. During this cycle, 82489DX drives random data on the bus with the appropriate parity. The interrupt Request Register is "frozen" and the highest priority pending interrupt vector is returned to the processor in the second acknowledge cycle.

The second acknowledge cycle has a complete data phase with timings identical to those of an ordinary register read. The data returned is the vector of the highest priority internally pending 82489DX interrupt, or the spurious interrupt vector, if there is no interrupt pending higher than the current processor priority.

Note that the timing diagram shows $\overline{\text{DLE}}$ being used (sampled high during $\overline{\text{ADS}}$). However, just as in a normal read cycle, the option exists not to use $\overline{\text{DLE}}$ (i.e., permanently tied to ground).

RESET AND MISCELLANEOUS TIMING

For discussion of this bus cycle, refer to Figure 26 82489DX Timing Diagram 4. It shows the 82489DX reset cycle, the timing of some related signals, and the ICC bus. The RESET input has a setup and hold time to the system clock edge, CLKIN, as do other independently timed signals, The RESET signal will reset the two asynchronous system on the chip, namely the ICC bus unit running synchronously to the ICLK and all the other unit running synchronous to the system clock, CLKIN. RESET must meet the minimum reset time with respect to both clocks and there should be at least one ICLK rising edge during reset. The TAP controller should also be initialized.

During reset, an eight-bit 82489DX Local Unit ID can be optionally initialized. Eight address lines, A10–A3 are sampled on every clock edge while RESET is asserted. The last sample remains in the 82489DX Local Unit ID register after reset. Alternatively, the 82489DX Local Unit ID can be loaded with a register write as part of software initialization, before 82489DX operation is started. In any event, the register must be initialized before the 82489DX can communicate on the ICC bus, including sending/receiving RESET messages. All valid signal to 82489DX should wait at least two full clocks after RESET is deasserted.

The PRST signal (reset output) is asserted both with RESET input, or under software control. Its on-off delay times are relative to the rising clock edge. The duration of PRST under software control is defined by the software itself. Also note that the PNMI pin has the same timing as does PRST when the latter is software controlled.

The ICC bus signals are both input and output on each cycle. Setup, hold and delay times are all measured with respect to the ICC bus Clock ICLK which has no relationship to the Processor clock on which the remainder of the 82489DX runs. This means that the ICC bus is independently sampled on each ICLK edge, as shown. It also implies that largest possible hold time will not exceed the minimum delay time.

After reset, all 82489DX registers are reset to "0" state. The mask bits in the local vector table and the redirection table are reset to "1" state to mask out all interrupts. All reserved bits are all wired to "0" state permanently on chip.

10.0 BOUNDARY SCAN DESCRIPTION

The 82489DX is equipped with the JTAG boundary scan standard. This feature allows the user to test the interconnections between 82489DX and the external hardware once they have been assembled onto a printed circuit board or other substrate. In addition to the JTAG mandatory instruction set, 82489DX also provides the INTEST instruction which allows static testing of the on-chip logic.

The detailed information related to the IEEE Std 1149.1-1990 (the JTAG standard) can be obtained from the reference document *IEEE Standard Test Access Port and Boundary Scan Architecture (IEEE Std 1149.1-1990).*

10.1 Boundary Scan Architecture

The boundary scan logic contains the following elements:

- **Five Test Access Ports (TAP):** They are labeled as $\overline{\text{trst}}$, tck, tdi, tdo and tms. All ports are input pins except tdo, which is a tri-state output pin.
- **A TAP Controller:** The logic is used to control the boundary scan activity.
- **82489DX Device ID Register:** This is a 32-bit read-only register. The DID can be shifted out in ascending order to the tdo pin.
- **JTAG Instruction Register (IR):** This is a 4-bit register which accepts instruction code shifted in from the tdi pin. The opcode stored in the IR register is used to control operation.
- **Boundary Scan Register:** This is a 137 stages scan path which connects almost all 82489DX signal pins for boundary scan purposes.
- **Bypass Register:** This register simply allows the data which goes into tdi pin to be shifted out directly from tdo.

The following block diagram illustrates the implementation of the JTAG architecture in the 82489DX design.

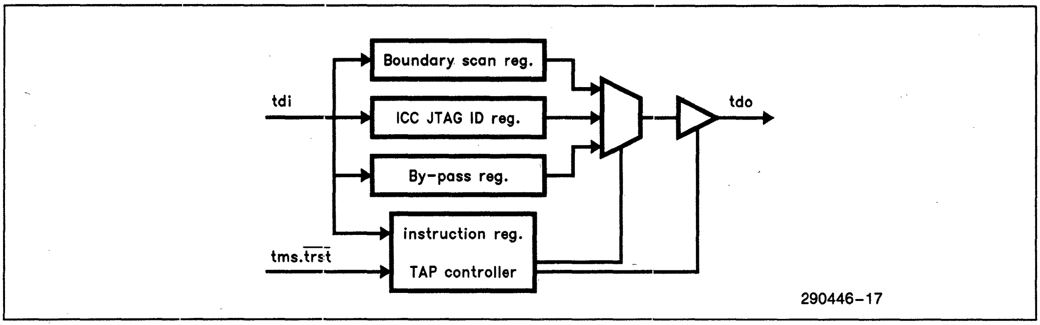


Figure 27. Block Diagram of the JTAG Architecture

Test Access Ports

$\overline{\text{trst}}$ TAP controller master reset pin. When $\overline{\text{trst}}$ is low, the TAP controller's state machine will be reset to "test-logic-reset" state asynchronously. This pin is tied to a weak internal pull-up for keeping to be a logical 1 when not driven.

tck This is the test logic clock. The test logic will change state on the rising edge of the tck.

tdi Test data input. Data is shifted into the tdi pin on the rising edge of tck. This pin is tied to a weak internal pull-up for keeping it to be a logical 1 when not driven.

tms Test mode select. This pin is used to select the state of the TAP controller. This pin is synchronous to the rising edge of the tck. This pin is tied to a weak internal pull-up for keeping it to be a logical 1 when not driven.

tdo Test data output. This is a tri-state pin which allows the data to be shifted out.

TAP CONTROLLER

The TAP controller in 82489DX is implemented to conform the IEEE1149.1 standard. The TAP controller is a single phase clock, synchronous finite state machine. It controls the sequence of the operation of the test logic.

The value of the test mode state (tms) pin at a rising edge of tck controls the sequence of the state changes. The state diagram for the TAP controller is shown in Figure 28. Test designers must consider the operation of the state machine in order to have the correct sequence of value to drive on tms.

The behavior of the TAP controller and other test logic in each of the controller states is briefly described as follows.

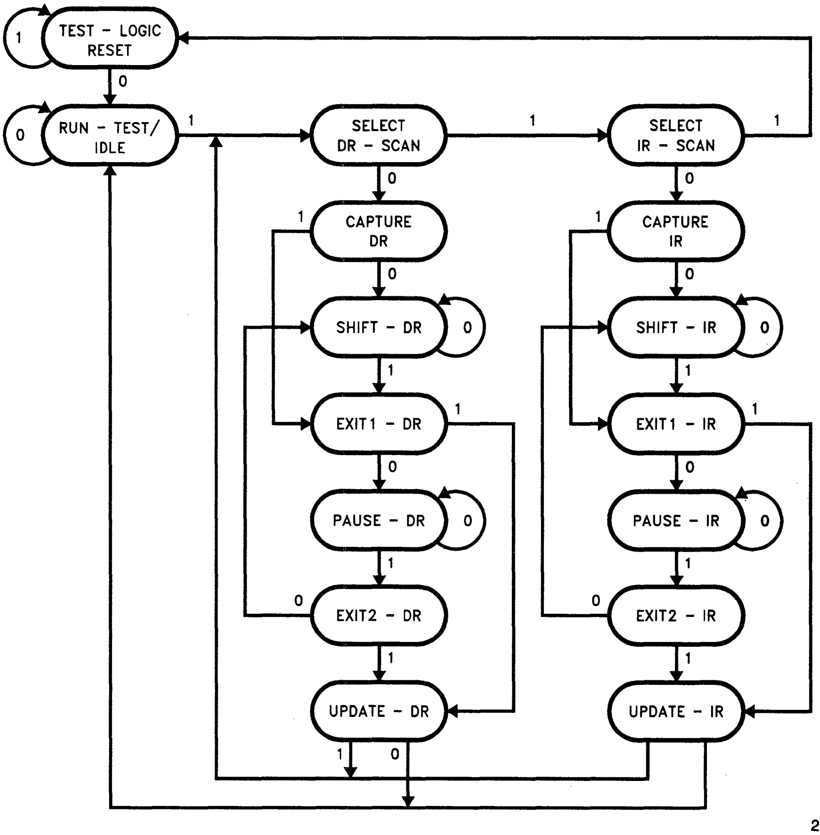


Figure 28. TAP Controller State Diagram

TEST-LOGIC-RESET

The test logic is disabled so that normal operation of the on-chip system logic (i.e., in response to stimuli received through the system pins only) can continue unhindered. This is achieved by initializing the instruction register to contain the IDCODE instruction. No matter what the original state of the controller, it will enter Test-Logic-Reset when tms is held high for at least five rising edges of tck. The controller remains in this state while tms is high.

If the controller should leave the Test-Logic-Reset controller state as a result of an erroneous low signal on the tms line at the time of the rising edge on tck (for example, a glitch due to external interference), it will return to the Test-Logic-Reset state following three rising edges of tck with the tms line at the intended high logic level. The operation of the test logic is such that no disturbance is caused to on-chip system logic operation as the results of such an error. On leaving the Test-Logic-Reset controller state, the controller moves into the Run-Test/Idle controller state where no action will occur because the current instruction has been set to select operation of the device identification register. The test logic is also inactive in the Select-DR-Scan and Select-IR-Scan controller states.

Note that the TAP controller will also be forced to the Test-Logic-Reset controller state asynchronously by applying a low logic level at $\overline{\text{trst}}$.

RUN-TEST/IDLE

A controller state between scan operations. Once entered, the controller will remain in the Run-Test/Idle state as long as tms is held low. When tms is high and a rising edge is applied at tck, the controller moves to the Select-DR-Scan state.

In the Run-Test/Idle controller state, activity in selected test logic occurs only when certain instructions are present such as RUNBIST. Since 82489DX does not have RUNBIST instruction, this state is acting like an idle state.

The instruction does not change while the TAP controller is in this state.

SELECT-DR-SCAN

This is a temporary controller state in which all test data register (82489DX has one test data register which is the boundary scan shift registers path) selected by the current instruction retain their previous state.

If tms is held low and a rising edge is applied to tck when the controller is in this state, then 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 on to the Select-IR-Scan state.

The instruction does not change while the TAP controller is in this state.

SELECT-IR-SCAN

This is a temporary controller state in which all test data registers selected by the current instructing retain their previous state.

If tms is held low and a rising edge is applied to tck when the controller is in this state, then 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 returns to the Test-Logic-Reset state. **The instruction does not change while the TAP controller Is In this state**

CAPTURE-DR

In this controller state data may be parallel-loaded into test data registers selected by the current instruction on the rising edge of tck. If a test data register selected by the current instruction does not have a parallel input, or if capturing is not required for the selected test, then the register retains its previous state unchanged. ***The instruction does not change while the TAP controller is in this state.***

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit 1-DR state if tms is held at 1 or the Shift-DR state if tms is held at 0.

SHIFT-DR

In this controller state, the test data register connected between tdi and tdo as a result of the current instruction shifts data one stage towards its serial output on each rising edge of tck. Test data registers that are selected by the current instruction, but are not placed in the serial path, retain their previous state unchanged. ***The instruction does not change while the TAP controller is In this state.***

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit 1-DR state if tms is held at 1 or remains in the Shift-DR state if tms is held at 0.

EXIT 1-DR

This is a temporary controller state, if tms is held high, a rising edge applied to tck while in this state 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. **All test date registers selected by the current instructions retain their previous state unchanged.**

The instruction does not change while the TAP controller is in this state.

PAUSE-DR

This controller state allows shifting of the test data register in the serial path between tdi and tdo to be temporarily halted. All test data registers selected by the current instruction retain their previous state unchanged.

The controller remains in this state while tms is low. When tms goes high and a rising edge is applied to tck, the controller moves on to the Exit 2-DR state. The instruction does not change while the TAP controller is in this state.

EXIT 2-DR

This is a temporary controller state. If tms is held high and a rising edge is applied to tck while in this state, the scanning process terminates and the TAP controller enters the Update-DR controller state. If tms is held low and a rising edge is applied to tck, the controller enters the Shift-DR state.

All test data registers selected by the current instruction retain their previous state unchanged. The instruction does not change while the TAP controller is in this state.

UPDATE-DR

Some test date registers may be provided with a latched parallel output to prevent changes at the parallel output while data is shifted in the associated shift-register path in response to certain instructions (e.g., EXTENT, INTEST, and RUNBIST). Data is latched onto the parallel output of these test data registers from the shift-register path on the falling edge of tck in the Update-DR controller state. The data held at the latched parallel output should not change other than in this controller state unless operation during the execution of a self test is required (e.g., during the Run-Test/Idle controller state in response to a design-specific public instruction).

All shift-register stages in test data registers selected by the current instruction retain their previous state unchanged. **The instruction does not change while the TAP controller is in this state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Select-DR-Scan state if tms is held at 1 or the Run-Test/Idle state if tms is held at 0.

CAPTURE-IR

In this controller state the shift-register contained in the instruction register loads a pattern of fixed logic values on the rising edge of tck. In addition, design-specific data may be loaded into shift-register stages that are not required to be set to fixed values.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and rising edge is applied to tck, the controller enters either the Exit 1-IR state tms is held at 1 or the Shift-IR state if tms is held at 0.

SHIFT-IR

In this controller 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.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state. When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit1-IR state if tms is held at 1 or remains in Shift-IR state if tms is held at 0.

EXIT 1-IR

This is a temporary controller state. If tms is held high, a rising edge applied to tck while in this state 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.

Test data registers selected by the current instruction retain their previous state. The instruction does not changes while the TAP controller is in this state and the instruction register retains its state.

PAUSE-IR

This controller state allows shifting of the instruction register to be halted temporarily.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

The controller remains in this state while tms is low. When tms goes high and a rising edge is applied to tck, the controller moves on to the Exit 2-IR state.

EXIT 2-IR

This is a temporary controller state. If tms is held high and a rising edge is applied to tck while in this state, termination of the scanning process results, and the TAP controller enters the Update-IR controller state. If tms is held low and a rising edge is applied to tck, the controller enters the Shift-IR state.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

UPDATE-IR

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of tck in this controller state. Once the new instruction has been latched, it becomes the current instruction. **Test data registers selected by the current instruction retain their previous state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters the Select-DR-Scan states if tms is held at 1 or the Run-Test/Idle state if tms is held at 0.

INSTRUCTION REGISTER

The function of the instruction register is to select the operating mode of the test logic. For instance, read the ID register, or capture the 82489DX output signals. 82489DX has implemented 4 instructions.

Instruction	Mandatory/Optional	Opcode
bypass	m	1 1 1 1
extest	m	0 0 0 0
sample/preload	m	0 0 0 1
idcode	m	0 0 1 0
reserved	o	1 0 0 1

Bypass Instruction

The bypass instruction selects the bypass register to be connected to tdi and tdo, effectively bypassing the test logic on the 82489DX boundary scan path and reducing the shift length to be on 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 internal pull-up on the tdi pin. This has been done to prevent any unwanted interference with the proper operation of the system logic.

Extest Instruction

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 82489DX's boundary scan register out on the output pins corresponding to each boundary scan cell and capturing the values on 82489DX's 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 located into the output control cell. Values shifted into input latch in the boundary scan register are never used by the internal logic of the 82489DX.

NOTE:
82489DX must be reset after extest instruction has been executed.

Sample/Preload Instruction

The sample/preload instruction has two functions that it can perform. When the TAP controller is in the CAPTURE-DR state, the sample/preload instruction allows a snap-shot of the normal operation of the 82489DX without interfering with that normal operation. The instruction causes boundary scan register cells associated with outputs to sample the value being driven into the 82489DX. On both outputs and inputs the sampling occurs on the rising edge of tck. When preloads data into the 82489DX 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.

dcode Instruction

The idcode instruction selects the device identification register to be connected to tdi and tdo, allowing the device ID code to be shifted out of the device on tdo. Note that the bit stream shifted into tdi will appear on tdo after all 32 bits of the DID has been shifted out.

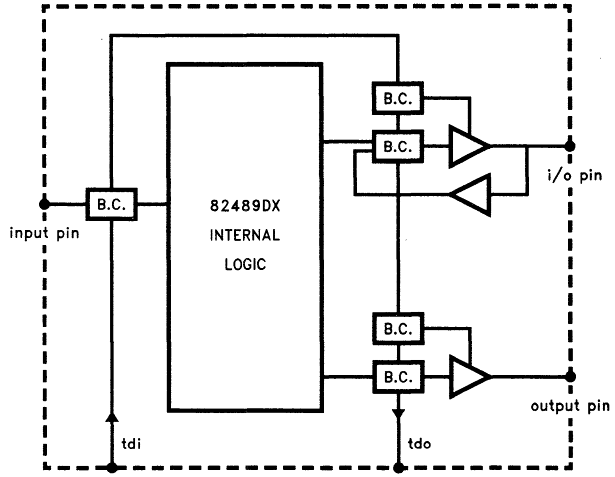
DEVICE IDENTIFICATION REGISTER (DID)

The device identification is a 32 bits number which can be read by the external hardware by using the idcode instruction. The 82489DX device ID is assigned to 1489A013 (hex). This is subject to change. The upper 4 bits of DID may be changed for different version. The 16-bit number (bit 27–bit 12) 489A (hex) is the part ID. The lower 12 bits are the manufacturer ID for Intel which must be 013 (hex).

BOUNDARY SCAN REGISTER

82489DX has only one test data register, i.e., the boundary scan register. The boundary scan register is a single shift register path containing the boundary scan cells that are connected to all signal input and output pins of the 82489DX. There are three generic type of boundary scan cells—input, output, and bi-directional. For each input only cell, one stage of shift register is added to the boundary scan path.

All output pins will become tri-stateable when boundary scan is activated, regardless whether they are tri-stateable or not in the normal operation. To explain further, the user will enable/disable an output driver with a specific tri-state control cell in the scan path. The user must shift in a proper control signal for these tri-state control cells in the scan path.



290446–19

Figure 29. Logical Structure of Boundary Scan Register

BOUNDARY SCAN CELL NAMES IN ORDER FROM tdi TO tdo

The following table is a list of the boundary scan cell names in the order from tdi to tdo. The type information indicates the purpose of the cells.

I = input only cell

B = bi-directional cell

T = tri-state output cell

C = tri-state control. Note that the signal name enclosed within the parenthesis is controlled by this cell.

Cell Number	Type	Name	Pin #
1	I	CLKIN	57
2	I	TMBASE	59
3	I	ICLK	60
4	I	$\overline{DC}$	61
5	I	$\overline{WR}$	62
6	I	$M/\overline{IO}$	63
7	I	$\overline{ADS}$	64
8	I	RESET	65
9	I	$\overline{BGT}$	66
10	I	reserved	70
11	I	reserved	71
12	I	reserved	72
13	I	$\overline{DLE}$	73
14	I	$\overline{CS}$	74
15	I	reserved	75
16	I	MBI3	76
17	I	MBI2	77
18	I	MBI1	78
19	I	MBI0	79
20	I	LINTIN1	80
21	I	LINTIN0	81
22	T	reserved	
23	C	(reserved)	
24	I	INTIN15	82
25	I	INTIN14	83
26	T	reserved	
27	C	(reserved)	
28	I	INTIN13	84
29	I	INTIN12	85
30	T	reserved	
31	C	(reserved)	
32	I	INTIN11	86
33	I	INTIN10	87
34	T	reserved	
35	C	(reserved)	
36	I	INTIN9	88
37	I	INTIN8	89
38	T	reserved	
39	C	(reserved)	
40	I	INTIN7	90
41	I	INTIN6	91
42	T	reserved	
43	C	(reserved)	
44	I	INTIN5	92
45	I	INTIN4	93
46	T	reserved	
47	C	(reserved)	
48	I	INTIN3	94
49	I	INTIN2	95
50	T	reserved	
51	C	(reserved)	
52	I	INTIN1	96
53	I	INTIN0	97
54	I	reserved	
55	B	DP3	101
56	C	(DP[3:0])	
57	B	DP2	102
58	B	DP1	103
59	B	DP0	104
60	T	reserved	
61	C	(reserved)	
62	B	D31	105
63	C	(D[31:0])	
64	B	D30	107

Cell Number	Type	Name	Pin #
65	B	D29	109
66	B	D28	110
67	T	reserved	
66	C	(reserved)	
69	B	D27	111
70	B	D26	112
71	T	reserved	
72	C	(reserved)	
73	B	D25	114
74	B	D24	115
75	B	D23	116
76	T	reserved	
77	C	(reserved)	
78	B	D22	118
79	B	D21	119
80	B	D20	121
81	B	D19	122
82	B	D18	123
83	B	D17	124
84	B	D16	125
85	B	D15	128
86	B	D14	129
87	B	D13	130
88	B	D12	131
89	B	D11	2
90	T	reserved	
91	C	(reserved)	
92	B	D10	3
93	B	D9	4
94	T	reserved	
95	C	(reserved)	
96	B	D8	7
97	B	D7	8
98	B	D6	9
99	B	D5	11
100	B	D4	12
101	B	D3	13

Cell Number	Type	Name	Pin #
102	B	D2	14
103	B	D1	16
104	B	D0	18
105	I	reserved	19
106	B	reserved	20
107	C	(cell106, A[10:3])	
108	B	A10	21
109	B	A9	22
110	B	A8	24
111	B	A7	26
112	B	A6	27
113	B	A5	28
114	B	A4	29
115	B	A3	31
116	T	reserved	34
117	C	reserved	
118	T	PINT	35
119	C	(PINT)	
120	T	PNMI	37
121	C	(PNMI)	
122	T	PRST	38
123	C	(PRST)	
124	T	ExtINTA	41
125	C	(reserved)	
126	T	reserved	42
127	C	(reserved)	
128	T	$\overline{\text{RDY}}$	43
129	C	($\overline{\text{RDY}}$)	
130	T	MBO3	45
131	C	(MBO3)	
132	T	MBO2	48
133	C	(MBO2)	
134	T	MBO1	49
135	C	(MBO1)	
136	T	MBO0	51
137	C	(MBO0)	

BYPASS REGISTER

The bypass register is simply a 1-bit shift register which connects between the tdi and tdo. When selected by using the bypass instruction, the data shifted into tdi will be shifted out from tdo one tck clock later.

JTAG TAP Controller Initialization

The TAP controller must be reset to test-logic-reset state when 82489DX is first powered up. There are two ways to reset the TAP controller:

1. Assert $\overline{\text{trst}}$ to be 0, it will reset the TAP controller asynchronously.
2. Assert tms to be 1, and clock the TAP controller at least five times, the TAP controller will be reset after the fifth rising edge of the tck.

After reset, the idcode instruction is loaded into the IR automatically.

Note that the tms and $\overline{\text{trst}}$ pins both have an internal weak pull-up device to keep them to be logic 1 level. Therefore the user can simply apply 5 clocks at the tck input to reset the TAP controller. If the TAP controller is not reset properly, 82489DX may not function because the boundary scan logic might be active which will impact the signals flow in and out to the chip.

11.0 ELECTRICAL CHARACTERISTICS

11.1 D.C. Specifications

ABSOLUTE MAXIMUM RATINGS

Case Temperature 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} + 0.5

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.

**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.*

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Min (ns)	Max	Units	Notes
V_{IL}	Input LOW Voltage (TTL)	−0.3	+0.8	V	
V_{IH}	Input HIGH Voltage (TTL)	2.0	V_{CC} + 0.3	V	
V_{OL}	Output LOW Voltage (TTL)		+0.45	V	(Note 1)
V_{QH}	Output HIGH Voltage (TTL)	2.4		V	(Note 2)
I_{CC}	33 MHz Power Supply Current		200	mA	
I_{LI}	Input Leakage Current		15	μA	
I_{LL}	Input Leakage Current		−600	μA	(Note 5)
I_{LH}	Output Leakage Current		600	μA	(Note 4)
I_{LO}	Output Leakage Current		15	μA	(Note 3)
C_{IN}	Input Capacitance		3	pF	
C_O	I/O or Output Capacitance		6	pF	
C_{CLKIN}	Clock Capacitance		3	pF	
I_{MLO}	ICC Bus Output Low Current		4	mA	(Note 6)
C_{MC}	ICC Bus Total Capacitance		100	pF	
V_{MH}	ICC Bus Input High (TTL)	2.0	V_{CC} + 0.3	V	
V_{ML}	ICC Bus Input Low (TTL)	−0.3	+0.8	V	

NOTES:
1. This parameter is measured with current load of 4 mA.
2. This parameter is measured with current load of 1.0 mA.
3. This parameter is for output without pulldown.
4. This parameter is for tri-state output with pulldown and V_{OH} = 3.0V.
5. This parameter is for input with pullup at V_{IL} = 0V.
6. ICC bus output low current is measured at 0.6V.

11.2 A.C. Specifications

A.C. Parameters Referencing 33 MHz System Clock

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Load (pF)	Min (ns)	Max (ns)	Notes
tc	CLKIN Period	30		30	100	(Note 1)
t1	CLKIN High Time	30		5		
t2	CLKIN Low Time	30		5		
t3	CLKIN Rise Time	30			3	(Note 2)
t4	CLKIN Fall Time	30			3	(Note 2)
t5	$\overline{ADS}$, $\overline{BGT}$, $\overline{DLE}$, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$, $\overline{CS}$ Setup Time	31		8		
t6	D31–D0, DP3–DP0, A9–A3 Setup Time	31		8		
t8	$\overline{ADS}$, $\overline{BGT}$, $\overline{DLE}$, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$, $\overline{CS}$ Hold Time	31		5		
t10	D31–D0, DP3–DP0, A9–A3 Hold Time	31		5		
t11	D31–D0, DP3–DP0, Valid Delay	30	50		18	
t12	D31–D0, DP3–DP0, Low-Z Delay When $\overline{DLE}$ is Not Used	32	50	3		(Note 7)
t13	D31–D0, DP3–DP0, High-Z Delay When $\overline{DLE}$ is Not Used	32	50		14	(Note 7)
t14	D31–D0, DP3–DP0 Enable Delay When $\overline{DLE}$ is Used	33	50	3	42	
t15	D31–D0, DP3–DP0 Disable Delay When $\overline{DLE}$ is Used	33	50	3	14	
t20	$\overline{RDY}$ Valid Delay	30	50	3	18	
t21	PRST, PNMI, PINT Valid Delay	30	50	3	34	
t22	RESET Setup Time	31		8		(Note 5)
t23	RESET Hold Time	31		5		(Note 5)
	RESET Cycle Time			5 tc		(Note 3)
				1 tic		(Note 3)
t24	INTIN[15:0], LINTIN[1:0] Low Time			10		(Note 6)

All parameters are given in nanoseconds.
TTL Level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:
1. ICC bus clock ICLK period must be at least 5 ns longer than system clock CLKIN for proper synchronization of the internal asynchronous signals.
2. System clock CLKIN measured from 0.8V–2.0V.
3. Minimum Reset cycle is the greater of the two cycle times.
4. Minimum pulse width must be met for valid level to be attained on the DATA or ADDRESS output.
5. Set up and hold time is required for RESET to start at the next rising edge of the clock.
6. INTIN and LINTIN low time is measured from 1.5V of the falling edge to 1.5V of rising edge.
7. Not 100% tested. Guaranteed by design characterization.

Time Base A.C. Parameters

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Note
tmc	TMBASE Period	35	40	10000	
t30	TMBASE High Time	35	10		
t31	TMBASE Low Time	35	10		
t32	TMBASE Rise Time	35		8	
t33	TMBASE Fall Time	35		8	

TAP Controller A.C. Parameters V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Note
ttc	TCK Period	35	40	1000	
t50	TCK High Time	35	10		
t51	TCK Low Time	35	10		
t52	TCK Rise Time	35		8	
t53	TCK Fall Time	35		8	
t54	TDI, TMS, $\overline{\text{TRST}}$ Setup Time	34	10		
t55	TDI, TMS, $\overline{\text{TRST}}$ Hold Time	34	5		
t56	TDO VALID Delay	34	5	24	(Note 1)
t57	Output Delay in EXTest in EXTEST Mode	34	5	27	(Note 1)
t58	$\overline{\text{TRST}}$ Minimum Low Time		10		(Note 2)

All parameters are given in nanoseconds.
TTL level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:
1. These parameters are specified for 50 pF load.
2. This parameter is measured at 1.5V between the rising and falling edges.

A.C. Parameters for ICC Bus

V_{CC} = 5V ±5%; T_C = 0°C to +85°C

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Notes
tic	ICLK Period	35	60		(Note 1)
t40	ICLK High Time	35	20		
t41	ICLK Low Time	35	20		
t42	ICLK Rise Time	35		10	
t43	ICLK Fall TIme	35		10	
t44	MBI3-MB10 Setup Time	36	8		
t45	MBI3-MB10 Hold Time	36	5		
t46	MB03-MBO0 VALID Low Delay	36		50	(Note 2)
t47	MB03-MBO0 VALID High-Z Delay	36	5	15	(Note 3)
t48	MB03-MBO0 VALID Low-Z Delay	36	12	25	(Note 3)

All parameters are given in nanoseconds.
TTL level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:
1. MBI3-0 and MBO3-0 timing is tested at 150 ns cycle time.
2. This parameter is specified for 50 pF load.
3. Not 100% tested. Guaranteed by design characterization.

12.0 REGISTER SUMMARY

82489DX registers can be located at any 1 Kbyte boundary in either memory or I/O space for as far as the 82489DX architecture itself is concerned. From a platform standard point of view, it is recommended to locate all 82489DX Local Units in memory space at address 0xFEE0—0000. It is further recommended that all 82489DX I/O Units also be located in memory space; I/O Unit 1 at address 0xFEC0—0000, I/O Unit 2 (if present) at address 0xFEC0—1000, and so on. Chip select for the 82489DX should be based on a full decode of address pins A31-A10.

All directly accessible 82489DX registers are 32 bits wide and are aligned at 128-bit boundaries. The register being accessed is determined by bits 4 through 9 of the address. This is listed in the tables below. Addresses not listed are reserved by the architecture. The tables also show whether the register is readable and/or writable by software, and what the side effects are of software accessing the register.

After reset, all registers are initialized to all zeroes with the following exceptions, The Local Unit ID field is initialized with data present on the 8 LSB address pins. The Mask bit is initialized to 1 ("masked" state) in all entries in both the local vector table and the redirection table.

For the I/O Unit, only the I/O register select and I/O window registers are directly accessible in the address space. The other I/O unit registers are accessed indirectly through the select and window register.

I/O Unit Registers

Register	Address (9:4)	SW	Side Effects
I/O Register Select	00 0000	W	
I/O Window Register	00 0001		

Register	I/O Reg Select (7:0)	SW	Side Effects
I/O Unit ID Register	0000 0000	rw	
Version Register	0000 0001	r	
Redirection Table [0] (31:0)	0001 0000	rw	
Redirection Table [0] (63:32)	0001 0001	rw	
Redirection Table [1] [31:0]	0001 0010	rw	
Redirection Table [1] [63:32]	0001 0011	rw	
Redirection Table [2] (31:0)	0001 0100	rw	
Redirection Table [2] (63:32)	0001 0101	rw	
Redirection Table [3] [31:0]	0001 0110	rw	
Redirection Table [3] [63:32]	0001 0111	rw	
Redirection Table [4] (31:0)	0001 1000	rw	
Redirection Table [4] (63:32)	0001 1001	rw	
Redirection Table [5] [31:0]	0001 1010	rw	
Redirection Table [5] [63:32]	0001 1011	rw	
Redirection Table [6] (31:0)	0001 1100	rw	
Redirection Table [6] (63:32)	0001 1101	rw	
Redirection Table [7] [31:0]	0001 1110	rw	
Redirection Table [7] [63:32]	0001 1111	rw	
Redirection Table [8] (31:0)	0010 0000	rw	
Redirection Table [8] (63:32)	0010 0001	rw	
Redirection Table [9] [31:00]	0010 0010	rw	
Redirection Table [9] [63:32]	0010 0011	rw	
Redirection Table [10] (31:00)	0010 0100	rw	
Redirection Table [10] (63:32)	0010 0101	rw	
Redirection Table [11] [31:0]	0010 0110	rw	
Redirection Table [11] [63:32]	0010 0111	rw	
Redirection Table [12] (31:0)	0010 1000	rw	
Redirection Table [12] (63:32)	0010 1001	rw	
Redirection Table [13] [31:0]	0010 1010	rw	
Redirection Table [13] [63:32]	0010 1011	rw	
Redirection Table [14] (31:0)	0010 1100	rw	
Redirection Table [14] (63:32)	0010 1101	rw	
Redirection Table [15] [31:0]	0010 1110	rw	
Redirection Table [15] [63:32]	0010 1111	rw	

LOCAL UNIT REGISTERS

Registers	Address (9:4)	SW	Side Effects
Local Unit ID Register	00 0010	rw	
Version Register	00 0011	r	
Reserved	00 0100		
Reserved	00 0101		
Reserved	00 0110		
Reserved	00 0111		
Task Priority Register	00 1000	rw	mask intr dispense
Reserved	00 1001		
Reserved	00 1010		
EOI Register	00 1011	rw	prioritization cycle
Remote Register	00 1100	r	
Logical Destination Reg.	00 1101	rw	
Destination Format Reg.	00 1110	rw	
Spurious Vector Register	00 1111	rw	
ISR (31:0)	01 0000	r	
ISR (63:32)	01 0001	r	
ISR (95:64)	01 0010	r	
ISR (127:96)	01 0011	r	
ISR (159:128)	01 0100	r	
ISR (191:160)	01 0101	r	
ISR (223:192)	01 0110	r	
ISR (255:224)	01 0111	r	
TMR (31:0)	01 1000	r	
TMR (63:32)	01 1001	r	
TMR (95:64)	01 1010	r	
TMR (127:96)	01 1011	r	
TMR (159:128)	01 1100	r	
TMR (191:160)	01 1101	r	
TMR (223:192)	01 1110	r	
TMR (255:224)	01 1111	r	
IRR (31:0)	10 0000	r	
IRR (63:32)	10 0001	r	
IRR (95:64)	10 0010	r	
IRR (127:96)	10 0011	r	
IRR (159:128)	10 0100	r	
IRR (191:160)	10 0101	r	

LOCAL UNIT REGISTERS (Continued)

Registers	Address (9:4)	SW	Side Effects
IRR (223:192)	10 0110	r	
IRR (255:224)	10 0111	r	
Intrpt Comnd Reg. (31:0)	11 0000	rw	send interrupt
Intrpt Comnd Reg. (63:32)	11 0001	rw	
Local Vector Table [timer]	11 0010	rw	
Reserved	11 0011		
Reserved	11 0100		
Local Vector Table [local int 0]	11 0101	rw	
Local Vector Table [local int 1]	11 0110	rw	
Reserved	11 0111		
Initial Count Register	11 1000	rw	
Current Count Register	11 1001	r	
Reserved	11 1010		
Reserved	11 1011		
Reserved	11 1100		
Reserved	11 1101		
Divider Configuration Reg.	11 1110	rw	
Reserved	11 1111		

NOTE:
Address space 101000 to 101111 and 111111 are reserved

13.0 TIMING DIAGRAMS

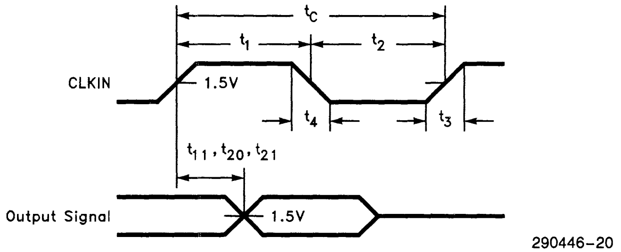


Figure 30. Output Waveform

13.0 TIMING DIAGRAMS (Continued)

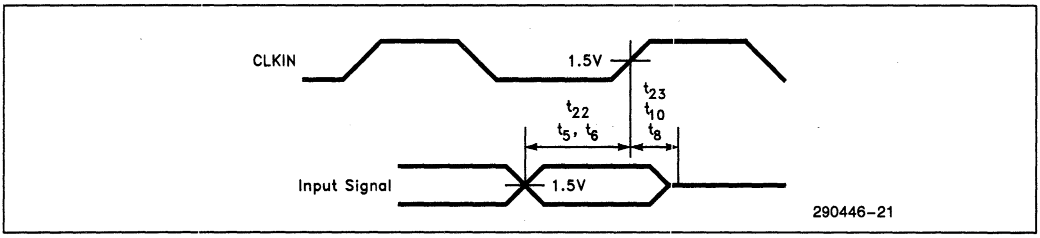


Figure 31. Input Waveforms

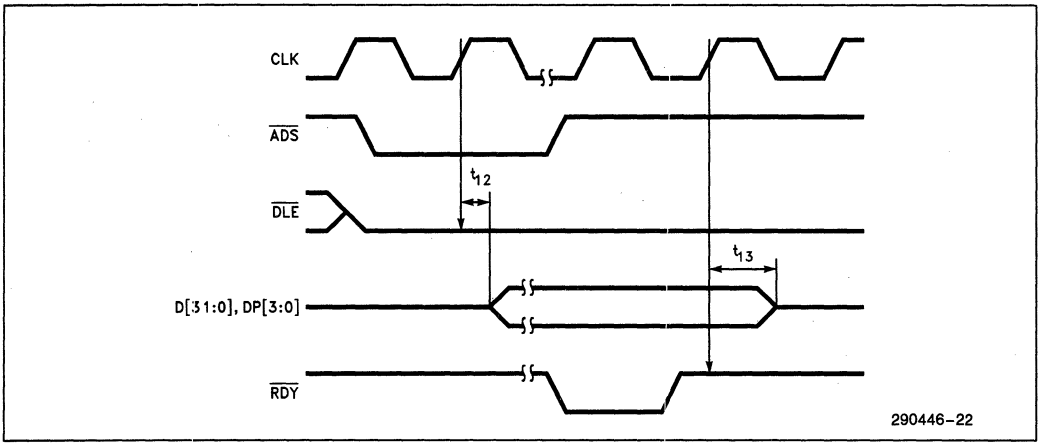


Figure 32. Data Bus Tri-State Delays when $\overline{\text{DLE}}$ Sampled Low with $\overline{\text{ADS}}$

13.0 TIMING DIAGRAMS (Continued)

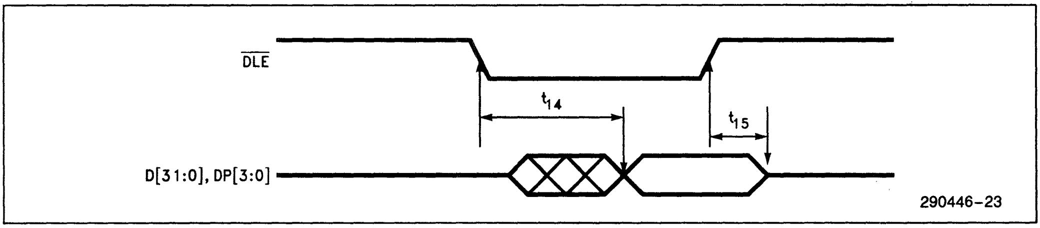


Figure 33. Data Enable/Disable Delay when $\overline{\text{DLE}}$ is Sampled High with $\overline{\text{ADS}}$

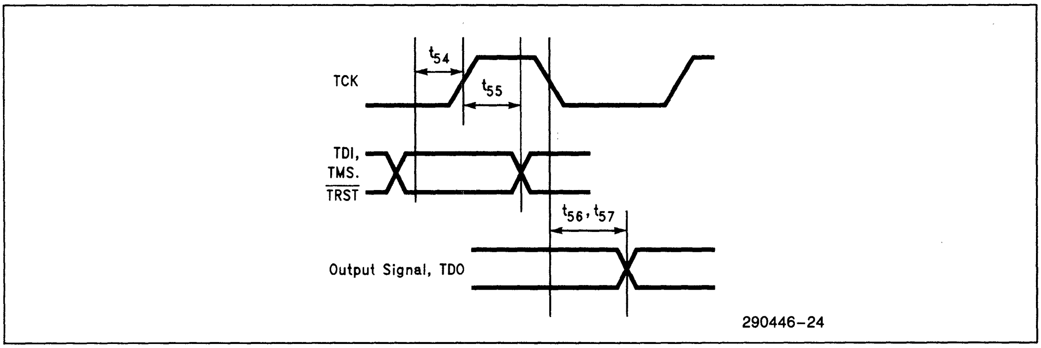


Figure 34. TAP Signal Timings

13.0 TIMING DIAGRAMS (Continued)

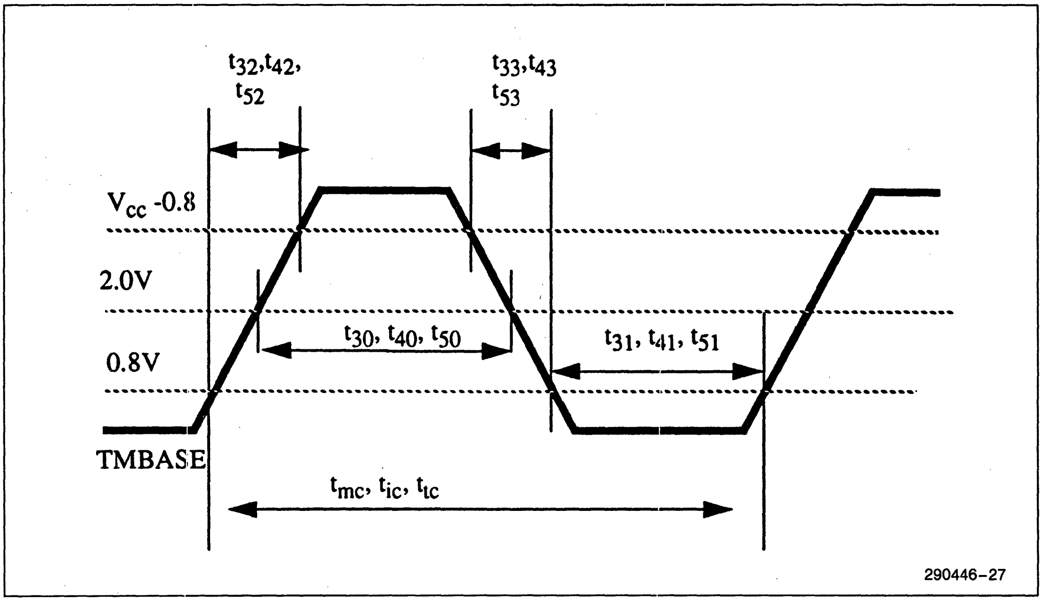


Figure 35. TMBASE, ICLK, TCK Timing

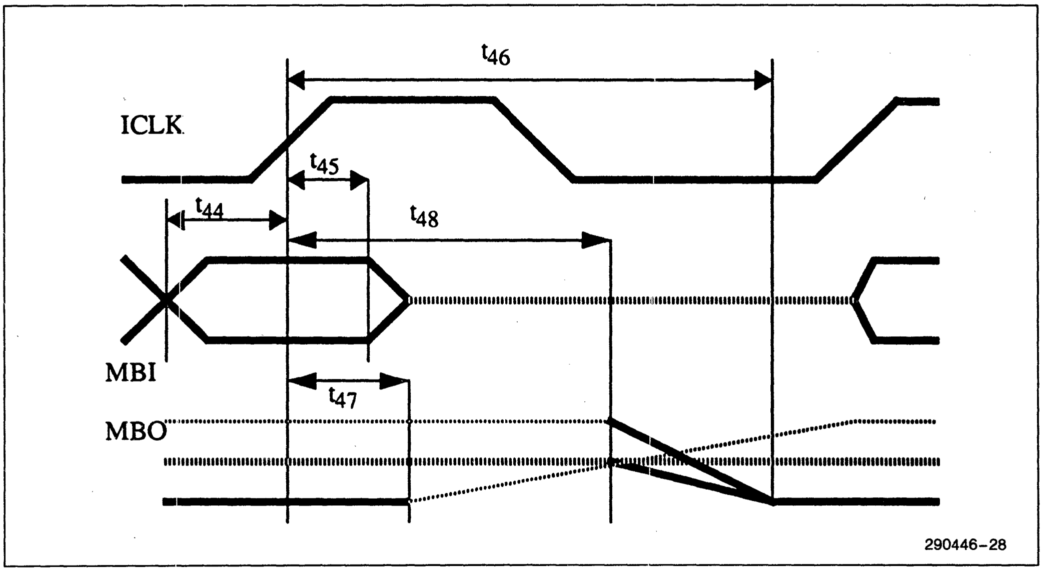
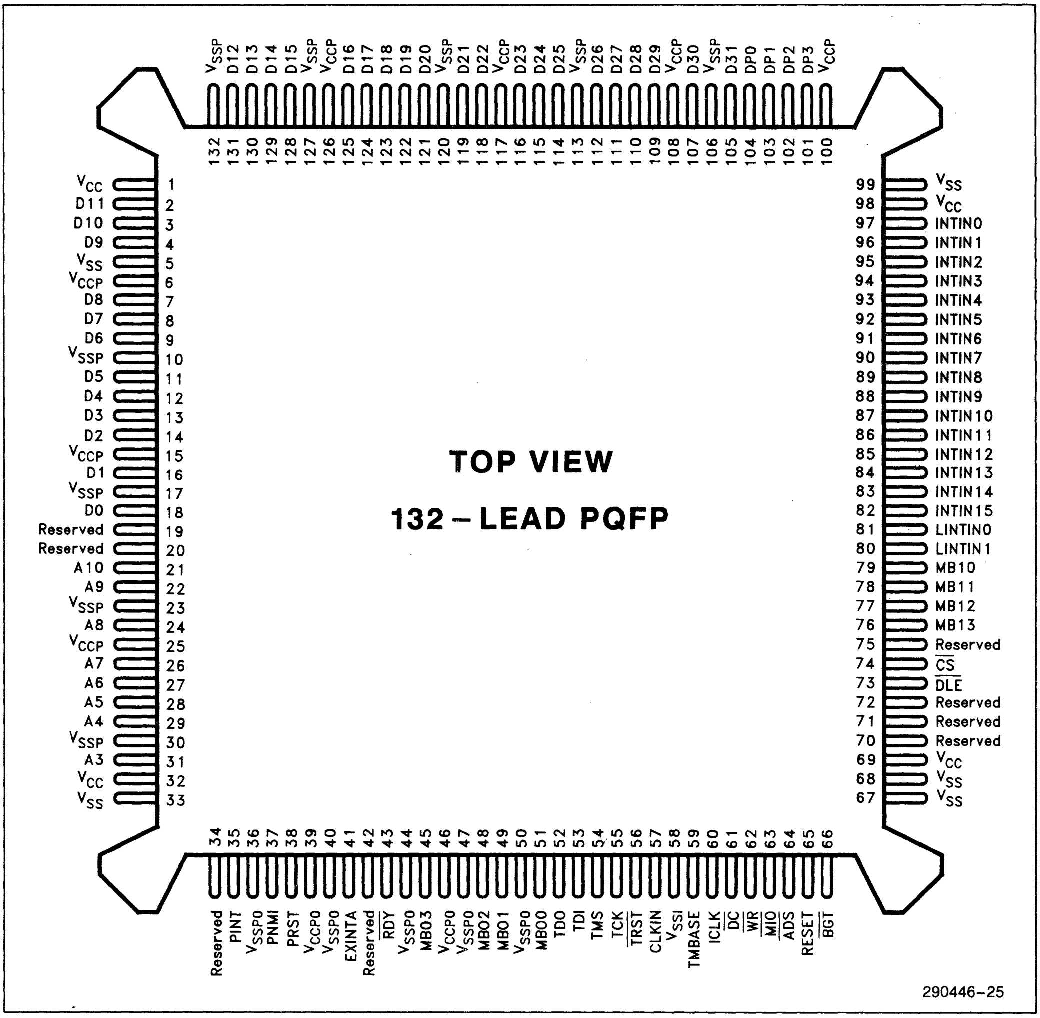


Figure 36. ICC BUS Open-Drain Output Delay

14.0 PACKAGE PIN-OUT

132-Lead PQFP
Package Type KU (See Packaging Specification. Order Number 240800)

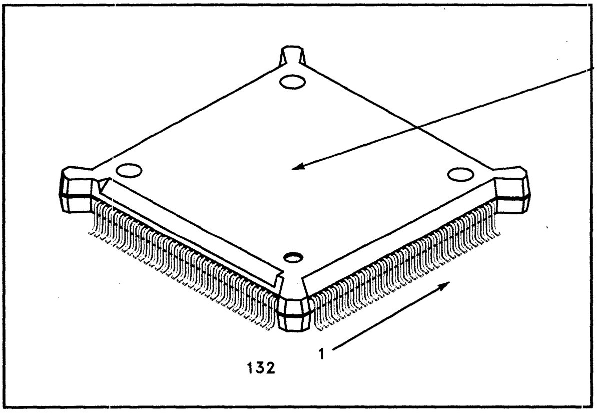


NOTE:
See pin description section for appropriate pin-strapping of the reserved pins.

15.0 PACKAGE THERMAL SPECIFICATION

The 82489DX is specified for operation when the case temperature is within the range of 0°C to +85°C. The case temperature may be measured in any environment, to determine whether the device is within the specified operating range.

The PQFP case temperature should be measured at the center of the top surface opposite the pins, as shown in Figure below.



MEASURE PQFP CASE TEMPERATURE AT CENTER OF TOP SURFACE

290446–26

Plastic Quad Flat Pack (PQFP)

PQFP Package Thermal Characteristics

Thermal Resistance—°C/W						
Parameter	**Air Flow Rate (Ft./Min)**					
	0	**200**	**400**	**600**	**800**	**1000**
θ Junction to Case	8	8	8	8	8	8
θ Junction to Ambient	32.5	25.5	20	18.5	16	15

NOTES:
1. Table above applies to 82489DX PQFP plugged into a socket or soldered directly into the board.
2. $\theta_{JA} = \theta_{JC} + \theta_{CA}$.

16.0 GUIDELINES FOR 82489DX USERS

16.1 Initialization

This section outlines one possible initialization scenario. Other scenarios are certainly possible, and one would be selected as part of a platform standard initialization scheme. The intent of this section is to illustrate that the initialization support provided by the 82489DX is adequate to support MP (Multiprocessor) system initialization.

Each 82489DX has a RESET input pin connected to a common Reset line. Upon system reset, this common reset line is activated, causing all the 82489DXs to go through reset. All 82489DX local units (note: only local units and not I/O units) latch their ID from their address bus on reset. The ID can be provided by the bus control agent based on slot number.

The local units next assert their processor's Reset pin, holding the processor in reset, and next perform their internal reset, setting all registers to their initial state. The initial state of all 82489DX Units (both local and I/O units) is "all masks set" and all Local Units disabled; registers are otherwise initialized to zero. Note that the PINT and PNMI output pins are in tri-state mode when the local unit is disabled. After this, each 82489DX local unit will deassert its processor's Reset pin, allowing the processors to come out of reset and perform self test and start executing initialization code.

Note that while connecting PRST pin it should be noted that whenever PRST pin is activated by 82489DX either because of software reset message or hardware reset, the 82489DX itself is reset. It should be taken care in the cases of Warm reset where only processors need to be reset and not the interrupt controller. In brief, the usage of PRST depends upon the system requirement on various reset.

Somewhere in this code sequence, the processors that are "alive" will enable their 82489DX local units, and attempt to force all the other processors back into Reset. Forcing the other processors into reset is performed by sending them the inter-processor interrupt with Destination Mode = "Physical", Delivery Mode = "Reset", Trigger Mode = "Level", Level = "1", and Destination Shorthand = "All Excl Self". Only the first processor to get the ICC bus will succeed in sending this signal and reset all other 82489DXs and their processors. The other processors are kept in reset until such time that an MP operating system decides they can become active again. The only running processor next performs the rest of system initialization.

Eventually, an MP operating system will be booted at which time the operating system would send "deassert reset" interprocessor signals to activate the other processors in the system. A mechanism must be provided by the platform that allows the added processors to differentiate the very first reset from a subsequent one.

16.2 Compatibility

COMPATIBILITY LEVELS

The 82489DX can be used in conjunction with standard 8259A-style interrupt controllers to provide a range of compatibility levels.

At the lowest level we have "PC shrink-wrap" compatibility. This level effectively creates a uniprocessor hardware environment within the MP platform capable of booting/running DOS shrinkwrap software. In this mode, only the 8259A generates interrupts and the 82489DX becomes a virtual wire. The interrupt latency can be minimized by connecting the 8259A interrupt to local unit directly.

The next level preserves the software compatible view of an 8259A but it allows more than one processor to be active in the system. This results in an asymmetrical arrangement, with one processor fielding all 8259A interrupts but with added inter-processor interrupt capability. In this mode, 82489DX "merges" 8259A interrupts with inter-processor interrupts. Existing I/O drivers would be bound to the compatible CPU and interface directly with the 8259A.

At the next compatibility level, 8259A compatible drivers can be mixed with native 82489DX drivers. Devices can generate interrupts at either 8259A or an 82489DX. This provides for partial symmetry as individual drivers migrate from the 8259A to native 82489DXs.

Another 8259A compatible point can be defined for MP systems. Each processor could have its own compatible 8259A controllers, allowing multiple processors to run compatible I/O drivers, but statically spreading the load across the available processors.

82489DX/8259A INTERACTION

The principle of compatible operation is very straightforward; the 82489DX(s) become a virtual wire connecting the 8259A's INT output through to the processor, while at the same time making 8259A visible to the processor.

The two connection schemes described only differ in the number of 82489DX(s) (one or two) that are located in the path from the 8259A to the processor. In the one 82489DX example illustrated in Figure 37, the INT output of the 8259A connects to one of the Interrupt Input pins of the 82489DX through an edge generation logic. This could be an interrupt pin on the 82489DX's I/O unit or local unit; assume a local interrupt input is used. The Local Vector Table entry for the interrupt pin that connects to the 8259A is set up with a Delivery Mode of "ExtINT" and edge trigger mode. This indicates that the interrupt is generated by an external controller. The processor's INT pin connects to the 82489DX PINT pin.

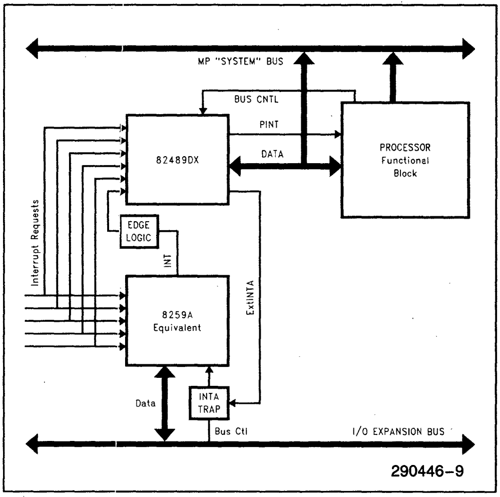
This setup enables the 82489DX local unit to detect assertions (up-edges) of the 8259A's INT output pin and pass this on to the processor's INT input. 82489DX asserts ExtINTA pin along with (one clock prior to) PINT pin to indicate "8259" interrupt. When the processor performs its INTA cycle the 82489DX itself does not respond other than deasserting PINT to the processor. At the third clock after $\overline{\text{ADS}}$ in the second bus cycle of INTA cycle ExtINTA is deasserted. External logic should make use of the ExtINTA signal to make the INTA cycle visible to the 8259A and the 8259A should provide the vector. At the same time, the local unit considers the external request as delivered, and need not wait for the external 8259A's INT to be deasserted. *A new up-edge must be generated on the 8259A INT pin before the local unit will assert the processor's INT pin on behalf of the 8259A. External edge generation logic should be used for this.* Compatible software interacts directly with the 8259A.

The mechanism is essentially the same in the two-82489DX scheme. The difference is that the 8259A connects to an interrupt input pin of the 82489DX I/O unit in the I/O system. The Redirection Table entry for this pin is again programmed with an "ExtINT" Delivery Mode, and the (single) 82489DX destination local ID corresponding to the compatible DOS processor. Capturing the up-edges of the 8259A's INT pin by the 82489DX local unit now involves sending messages from the 82489DX I/O unit to.the 82489DX local unit via the ICC bus. The "virtual wire" now includes messages over the ICC bus.

Adding inter-processor ICC interrupts (or any other 82489DX generated interrupts) to the compatible operation is accomplished by having the 82489DX internally OR the 8259A's INT request with any 82489DX interrupt request.

Before the 82489DX actually sends the interrupt signal to the processor, the 82489DX decides whether it does this for an 82489DX interrupt or whether it does this on behalf of the external controller. When the processor performs the corresponding INTA cycle, only the 82489DX knows whether it should respond with a vector, or whether the external 8259A should.

If the 82489DX needs to respond, then it will enable an externally implemented trap that prevents the 8259A from seeing the INTA cycle. If the 8259A needs to respond, then the 82489DX will not enable the INTA trap, and the INTA will be allowed to reach the 8259A. 82489DX implements this by asserting its EXTINTA pin to indicate external 8259A should respond with the vector. The 82489DX local unit controls the INTA trap via its "ExtINTA" output pin; the 82489DX does not actually provide the trap itself.



290446–9

Figure 37. Edge Logic

82489DX/8259A DUAL MODE CONNECTION

In systems that can be booted either as a configuration with compatible 8259A or without, device interrupt lines are connected to both the Interrupt Request pins of the 8259A and Interrupt Input pins of the 82489DX with all interrupts either masked at the 82489DX or at the 8259A. Some EISA and Micro-Channel chip sets that include on-chip 8259As also have internally connected interrupt requests. For example, the 82357 (the ISP of the EISA chipset) generates timer and DMA chaining interrupts internally. These are not available as separate interrupts outside the ISP. In non compatible mode the ISP timers are not used, since each local 82489DX unit provides its own timer. Therefore, the ISP's 8259A is configured to mask out all interrupts except the DMA chaining interrupt which is configured in level-sensitive, auto EOI mode. This causes the 8259A's INT output to track the state of the internal DMA interrupt request. The 8259A's INT output is then connected to one of the 82489DX interrupt input pins programmed to generate a regular (i.e., not

"ExtINT") level-sensitive interrupt. The ISP 8259A then no longer functions as an external interrupt controller; it has been logically disabled, and it needs no interrupt acknowledge or EOI. The INTA and EOI cycles occur only at the 82489DX. It should be noted that 82489DX accepts only active high level/edge interrupt inputs. External programmable logic should take care of polarity reversal that may be needed in EISA system for sharing of interrupts.

16.3 Hardware Guidelines

82489DX HARDWARE STATE ON RESET

The 82489DX goes to reset state either by Hardware Reset state or Software Reset message received on the ICC bus. On reset, 82489DX is disabled. The following is the hardware state of 82489DX after reset.

PRST	Active (HIGH)
PNMI	TRI-STATED (Internal Pull-Down Provided)
PINT	TRI-STATED (Internal Pull-Down Provided)

82489DX is disabled on Reset and unless specifically enabled, it does not start its interrupt mechanism.

The difference between hardware reset and software reset message is that during hardware reset 82489DX samples the address bus and stores the last sample in Local Unit ID whereas for software reset it does not sample and store the unit ID. In addition, during the hardware reset pulse should be wide enough to accommodate for at least one rising and falling edge of ICLK. On hardware reset ExtINTA is held high.

PULL UP AND PULL DOWN RESISTORS

PNMI, PINT are tri-stated at power on and they are maintained in tri-state condition till the unit is enabled. Eventhough internal pull down resistor is provided on PNMI and PINT external additional pull down resistor may be needed depending upon the loading on these pins by external logic. The DC characteristics gives the control specification from which the value of resistor, if needed, can be calculated.

It should be kept in mind that the ICC bus being electrically open drain bus requires pull up resistors at the MBO pins. ICC bus output low current is just 4 mA.

PINT and ExtINTA Timings

It should be noted that for ExtINTA type of interrupts **PINT gets activated one clock after ExtINTA gets activated.** When getting deactivated, both PINT and ExtINTA gets deactivated at the same clock edge.

ExtINTA Timings

In the interrupt acknowledge cycle for External Interrupt control, 82489DX asserts ExtINTA. It decodes the type of cycle from CPU control signals like M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$ and W/$\overline{\text{R}}$. CPU does two bus cycles back to back for interrupt acknowledge cycle. 82489DX maintains ExtINTA active throughout the first cycle. For next cycle (when the vector will be given by external 8259) after 82489DX senses the start of the cycle (by $\overline{\text{ADS}}$) 82489DX deactivates ExtINTA. External control logic may be inserting wait states to match the 8259 timings. Since 82489DX has no way of finding out the cycle completion, 82489DX deasserts ExtINTA before the second bus cycle gets completed. This should be kept in mind while using ExtINTA for external interrupt control logic.

82489DX AND MEMORY MAPPING

The 82489DX is a 32-bit high performance interrupt controller. It allows the CPU to do 32-bit read and write to it. By memory mapping 82489DX the system performance can be enhanced. It should be noted that 82489DX does not support pipelining. Eventhough 82489DX can be memory mapped, its functionality as an interrupt controller should be kept in mind while programming the virtual memory management control data structure. The caching policy for the page where 82489DX is mapped should also be done with the functionality of 82489DX in mind. For example, the reads to 82489DX should not be cached and writes should be write-through. Since 82489DX registers are aligned at 128-bit boundaries, memory mapping 82489DX with interleaved memory system should not be a problem.

JTAG CIRCUIT CONSIDERATIONS

The JTAG circuit is used for boundary scan test. The JTAG pins has a TCK, (JTAG clock), $\overline{\text{TRST}}$, (JTAG Reset), TDI, (Test Data Input), TMS, (Test Mode Select) and TDO, (Test Data Output).

The JTAG circuitry, if not used, should be properly deactivated so that it will not interfere in the normal functional operations. The JTAG can be inactivated in any one of the following ways:

1. JTAG inactivation through $\overline{\text{TRST}}$: The TMS, Test mode Select should be either left open (internal pull up is provided) or tied to V_{CC}. The $\overline{\text{TRST}}$ can be pulsed low (bring it low and after meeting the pulse width requirement bring it back high again) to keep the JTAG circuitry to idle state. The $\overline{\text{TRST}}$ pulse brings the JTAG circuitry to idle state and TMS being kept high maintains the JTAG circuitry in idle state.

2. JTAG inactivation through TCK: The TMS, Test mode select should be either left open (internal pull up is provided) or tied to V_{CC}. The TCK within 5 clocks brings the JTAG circuitry to idle state. The TMS, being held at logic high level, maintains the JTAG circuitry in idle state.

16.4 Programming Guidelines

The 82489DX register data structure contains different fields to specify the mode of operations and the options available within each mode. Since certain options are applicable to specific modes only (for example "Remote Read" mode applies only to interrupt command register, it does not have relevance to I/O unit's redirection tables) the following programming guidelines are provided.

UNIQUE ID REQUIREMENT

All the local units and I/O units hooked in a ICC bus should have unique ID before they can use the bus. This should be ensured by the programmer since for ICC bus arbitration the units (whether it is local unit or I/O unit) arbitrate with their unit ID.

For future compatibility, the Units should be assigned IDs starting with 0, 1, 2 etc. with the highest ID in the system being number of units minus 1. So in a four 82489DX system there are four local units and four I/O units. The ID starts with 0 and the highest ID in the system will be 7. Note that each unit should have different ID in the system.

ATOMIC WRITE READ TO TASK PRIORITY REGISTER

Normally, the task priority register is written with highest priority to mask certain low level interrupts before entering into critical section code. In a system where 82489DX is memory mapped the CPU may buffer this task priority register write to its on chip write buffer. The following scenerio can happen in such situation: CPU posts task priority register write to its on chip write buffer and enters into the critical code. A lower priority interrupt (which should not enter the critical code) interrupts the CPU before the write buffer gets flushed into task priority register). The CPU accepts the lower priority interrupt. To avoid the situation atomic write read to task priority register should be done. The read following write ensures that the write buffer is flushed to task priority register and the atomicity ensures that no interrupt will be accepted by the CPU during its write to task priority.

It should be noted that if the CPU does interrupt acknowledge cycle only after flushing the write buffers then the above situation may not arise.

CRITICAL REGIONS AND MUTUAL EXCLUSION

Each 82489DX has a single Interrupt Command Register that it uses to send interrupts to other processors. The programmer should make sure to synchronize access to this register. Specifically, 1). writing all fields of the register, 2). Sending the interrupt message (by writing the LSB register), and 3). waiting for Delivery State to become Idle again, should occur as a single atomic operation. For example, if interrupt handlers are allowed to send inter-processor interrupts, then interrupt dispensing to the processor must be disabled for the duration of these activities.

INTERRUPT COMMAND REGISTER PROGRAMMING SEQUENCE

The interrupt command register (31:0) has a side effect of sending interrupt once it is written. The destination is provided in the interrupt command register (63:32). So always interrupt command register (63:32) should be programmed before programming interrupt command register(31:0).

Program Interrupt Command Register (63:32)

↓

Program Interrupt Command Register (31:0)

INTERRUPT VECTOR

Two different interrupts should not be programmed with the same interrupt vector.

LOCAL AND I/O UNIT

Only Interrupt command register supports "Remote Read" Delivery mode. Local and I/O unit interrupts do not support "Remote Read".

ICR (INTERRUPT COMMAND REGISTER)

1. ExtINTA delivery mode is not supported for all destination shorthands.
2. "Remote Read" should always be programmed as "Edge" triggered interrupt.
3. "Remote Read" should always be programmed with physical destination mode (and not with Logical Destination mode). Broadcast addressing should not be used for Remote Read.
4. For "all incl Self" and "All exc. self" destination shorthands, "remote read" delivery mode should not be used.
5. For "all incl self" and "self" destination shorthands "Reset" delivery mode should not be used.

ICR (INTERRUPT COMMAND REGISTER)

(Continued)

6. For "all exc self" destination shorthand if "Reset" delivery mode is used, it should be ensured at system level that only one processor executes this instruction at any time.
7. Messages could be sent out in "Logical" or "Physical" mode with destination ID of all 1's depending on the way Destination mode entry is programmed. In brief, "All incl self" and "All exc. Self" support both "Logical" and "Physical" addressing mode.
8. When destination shorthand (i.e., broadcast) is used with lowest priority destination mode, then even though all participates in arbitrating for destination, only the lowest priority gets the message. So even though the addressing is broadcast since the destination mode is lowest priority only one gets the message.
9. When destination shorthand (i.e., broadcast) is used with "Fixed" destination mode, then all the units get the message.

ISR/IRR/TMR

Bits 0–15 of IRR/ISR/TMR do not track interrupt. No interrupt of vector numbers from 0–15 can be posted. The total interrupt supported are 240. When reading the lowest 32 bits of these registers, 0 will be returned for the lower 16 bits.

FOCUS PROCESSOR

Focus processor is applicable only within the addressed units.

ExtINT Interrupt Posting

The external interrupt has no priority relationship with the 82489DX priority. But when posting an interrupt to the processor, if both an external interrupt and a 82489DX interrupt are pending, 82489DX could post either one to the processor. In 82489DX implementation, it would post external interrupt whenever there is no other 82489DX interrupt that can be posted to the processor. It should be also noted that External interrupts can not be masked by raising task priority. However, they can be masked by the mask bit in the table entry for the ("ExtINTA") external interrupt.

The extINT interrupts are specific in their characteristics in that they do not have any priority relationship with the rest of the interrupt structure. ISR and IRR bits in 82489DX are used to do the housekeeping functions for interrupt priority. Since extINT interrupts do not have any priority relationship, ISR and IRR bits are not maintained for external interrupts. As far as interrupt acceptance is concerned, if more than one extINT interrupts are directed towards a local unit, that local unit treats all the extINT interrupts directed to it as only one extINT interrupt. This leads to an important point that in a system not more than one interrupt should be programmed as extINT interrupt type with the same destination. It should be noted that there can be more than one extINT type of interrupt in a system with each having different local unit as destination.

Synchronlzing Arb IDs

Initialization of an 82489DX's local unit ID is implementation dependent. In some platforms, power-on reset will latch the right values into the 82489DXs; in other platforms, unique IDs may be assigned by initialization firmware. In both cases the 82489DX I/O unit should be assigned unique ID by initialization firmware. The important point is that the 82489DXs are required to have unique IDs before they can use the bus, and in addition, all their Arb IDs must be "in sync". Synchronizing Arb IDs is accomplished as a side effect of a "deassert reset" interrupt command. This resets the (rotating) Arb ID to the (constant) unit ID; it assumes that all 82489DXs have their unique ID.

LOWEST PRIORITY

"Only once delivery" semantics for a group destination is guaranteed only if multiple fixed delivery of the same interrupt vector are not mixed. For lowest priority arbitration to work, all the arbitration ID of local 82489DXs in the system should be in sync. This means after local unit IDs are written in all local units (each ID should be different from other IDs) a RESET DEASSERT message should be sent in ALL INCLUSIVE mode. The RESET DEASSERT message should be sent before system is used for lowest priority arbitration. This ensures that all ARB IDs are also different. (Arb IDs are copied from local unit IDs during RESET DEASSERT message.)

The RESET DEASSERT message, if not sent, only one delivery semantics may not be guaranteed in the cases where lowest arbitration is used in the system.

DISABLING LOCAL UNIT

Once the 82489DX is enabled by setting bit 8 of spurious vector register to 1, the user should not disable the local unit by resetting the bit to 0. The result will put the local unit in an inconsistant state. The local unit can be disabled by sending "reset" interrupt message to the local unit.

ISSUING EOI

EOI, End of Interrupt issuing indicates end of service routine to 82489DX. The ISR bit which is set during INTA cycle gets cleared by EOI. This section discusses the relevence of EOI to the specific types of interrupts and its timing related to interrupt deassertion.

EXTERNAL INTERRUPTS AND EOI

External Interrupts should be programmed as edge type. INTA cycles to external interrupts are taken automatically as EOI by 82489DX. This is similar to AEOI, Automatic End of Interrupt of 8259A. So there is no need to issue EOI to 82489DX for external interrupt servicing. This is done to achieve software transparency in the compatible mode.

SPURIOUS INTERRUPTS AND EOI

Spurious interrupts do not have any priority relationship to other interrupts in the system. So IRR is not set for spurious interrupts. EOI should not be issued for spurious interrupts. It is advisable not to share the spurious interrupt with any vector. If spurious interrupt vector is shared with some other interrupt then while servicing issuing EOI depends on the source of interrupt. If the source is spurious interrupt (for which the corresponding IRR is not set) then EOI should not be used. If the source is a valid interrupt sharing the spurious interrupt vector (for which the IRR is set) then EOI should be issued.

NM AND EOI

For NM type of interrupt no IRR bit is set. So EOI should not be issued while servicing NMI type of interrupts.

TASK PRIORITY REGISTER

Task priority register is used to specify the priority of the task the processor is executing. In 8259 the priority is defined only among the interrupts that it handles. 82489DX goes farther ahead in handling priority. In multitasking system, in addition to device interrupts various tasks have different priority and 82489DX allows consideration of the priority at system level. The processor specifies the priority of the task it executes by writing to task priority register. Now any interrupts at and below the task priority will be masked temporarily till the task priority gets lowered. The masking granularity is at priority level. Out of 256 interrupt vectors 16 priority levels are specified and 16 vectors share one priority level. Since the masking granularity by the task priority register is at priority level, group of 16 vectors get masked when task priority register is increased by one level.

When task priority is at its minimum level of 0, interrupt vectors having level 1 to 16 are passed to CPU. Stated in other words, even when the task priority register is at its minimum (of level 0), interrupt vectors at level 0 will be masked. This means that the interrupt should not be programmed with vectors 0 to 15. So out of 256 interrupt vectors, only 240 interrupt vectors (vector 16 to 255) can be used in 82489DX.

ExtINT INTERRUPT AND TASK PRIORITY

ExtINT interrupt does not have any priority relationship with other interrupts or task priority register. So ExtINT interrupt can not be masked by raising task priority. They can be masked by writing to the vector table entry which corresponds to ExtINT interrupt.

REMOVING MASKS

When enabling units and removing Mask bits in situations where a device may already be injecting interrupts into the 82489DX system, the Mask in the Redirection Table should be removed last to ensure proper initial state (e.g., Remote IRR bit matching IRR in local unit).

DELIVERY MODE AND TRIGGER MODE

It is software's responsibility to make sure that Delivery Mode and Trigger Mode are set to meaningful combinations as listed below.

Delivery Mode	Trigger Mode
Fixed	edge/level
Lowest Priority	edge/level
Remote Read	edge
NMI	level
Reset	level
ExtINT	edge

Software is also responsible for not using meaningless Delivery Modes in Redirection Table entries and local Vector Table entries (e.g., use of Remote Read delivery mode).

ASSIGNING INTERRUPT VECTORS

Software has total control over the assignment of interrupt vectors to interrupt sources. The operating system writer should be aware of a number of things when doing this assignment.

Some processor architectures assign a predefined meaning to some of the vectors (i.e., entries in the interrupt table) as entry points to certain trap and exception handlers (e.g., divide error, invalid opcode, page fault, etc.). The programmer is strongly advised not to reuse these vectors.

The programmer must also be careful when using the same vector number to represent different interrupt sources (sharing vectors). This is especially true for level triggered interrupts. When multiple sources with different Redirection table entries share an interrupt vector, any of the sources deactivating its level signal will remove the interrupt request for all sources. Giving each interrupt source its interrupt vector in any case is the preferred approach.

SENDING INTER-PROCESSOR INTERRUPTS

Each 82489DX has a single Interrupt Command Register that it uses to send interrupts to other processors. It is software's responsibility to synchronize access to this register. Specifically, 1) writing all fields of the register, 2) sending the interrupt message (by writing the low register), and 3) waiting for Delivery State to become Idle again, should occur as a single atomic operation. For example, if interrupt handlers are allowed to send inter-processor interrupts, then interrupt dispensing to the processor must be disabled for the duration of these activities.

DELAY WITH LEVEL TRIGGERED INTERRUPTS

When a level triggered interrupt source deasserts its interrupt input, the destination will clear the interrupt's IRR bit only after receiving the message from the ICC bus. This introduces a small delay between the removal of the interrupt at the source and the removal of the interrupt at the processor. To avoid generating unnecessary interrupts, the interrupt handler should remove the interrupt at the source (at the device) as early as possible in the handler.

In any case, handlers should be able to deal with unnecessary interrupts.

RESET DEASSERT

A side effect of a reset deassertion message broadcast in the ICC bus is that all 82489DX local units reset their Arb ID to their unit ID. Interrupt commands that use the "self" Destination Shorthand do not generate a message on the ICC bus. If software only wants to generate the side effect of resetting Arb IDs, it should use a command with Logical Destination Mode and a Destination field containing all zeroes.

INTERRUPT MASKING

There are a number of levels at which interrupts can be masked, each resulting in a different behavior on interrupt delivery.

- First, interrupt injection (or deliver) can be masked by setting the Mask bit in the interrupt's Redirection Table or local Vector Table entry. These interrupts are ignored, no message is sent for them. Granularity is an individual interrupt.
- Second, each 82489DX can individually mask interrupt dispensing by raising its Task Priority to some level. This 82489DX will not dispense interrupts to its processor of this and lower priority unless it is currently the focus of the interrupt. Note again that the 82489DX is designed to operate as fully nested with non-specific EOI (to use existing 8259A terminology). There is no explicit interrupt mask (such as MR) and there is no notion of specific EOI.
- Third, each processor may provide a mechanism that masks all interrupt dispensing to it using the processor supplied instructions or status bits to do so. This does not interfere with lowest priority arbitration of the processor's 82489DX local unit.

CHANGING REDIRECTION TABLES

Redirection Tables are typically set up at initialization time. When modifying a Redirection Table entry "on the fly" the programmer must be aware of state kept at other 82489DXs relative to the interrupt being modified.

DEVICE DRIVERS WITH 82489DX

It is strongly recommended to read the device status registers before servicing the device. This is because if an edge triggered device deasserts its interrupt before interrupt acknowledge cycle (it should NOT) 82489DX will NOT give spurious vector. It will give genuine interrupt vector corresponding to the device. So, interrupt service routine should validate the interrupt request before servicing the device.

SYSTEM HARDWARE AND SOFTWARE DESIGN CONSIDERATIONS

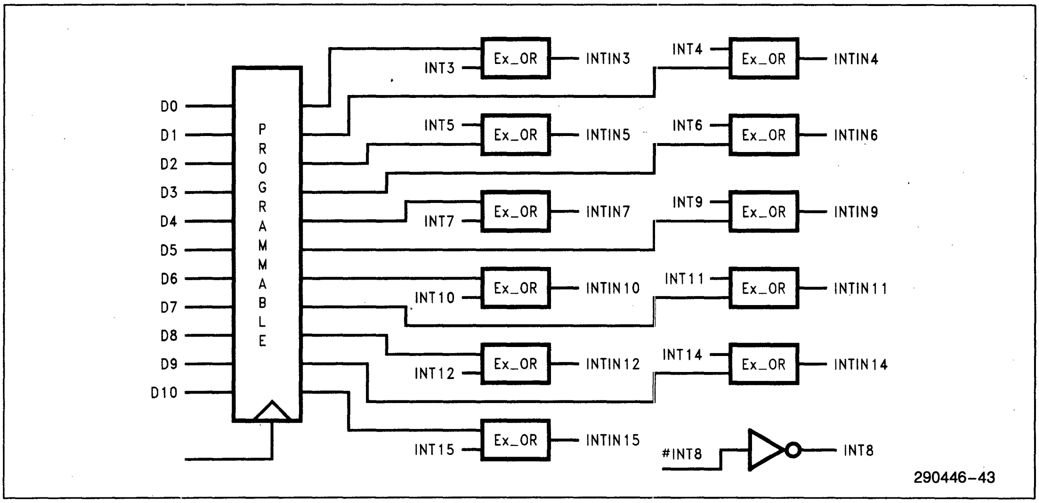
Design Consideration 1

Description: The following design consideration has to be taken care of when using ISP (82357) as external interrupt controller. 82489DX allows connecting external 8259 type interrupt controller at one of its inputs. The mode associated with the interrupt input which has 8259 connected to it is called ExtINTA mode. 82489DX allows only EDGE TRIGGERED programming option for ExtINTA mode. But in the case of 82357, the INT output from ISP stays high in case more than one interrupt is pending at its inputs. It does not always inactivate its INT output after INTA cycle. This will lead to a situation where ISP keeps the interrupt at high level continously and waits for INTA cycle. But since 82489DX expects an edge for interrupt sensing (for ExtINTA interrupts) it does not pass the interrupt to CPU and further interrupts are lost. So External circuitry should monitor the end of SECOND CYCLE of INTA cycle and force an inactive state at 82489DX's input. To avoid glitches at 82489DX input, this external logic should clear its output only at the end of second INTA cycle. It should be set by high going 82357 output. It should never be cleared by low going 82357 output. That is it should not follow 82357 output.

Design Consideration 2

Description: The following design consideration has to be taken care of when using 82489DX in EISA systems. EISA ISP(82357) chip integrates 8259A. It additionally allows sharing of interrupts. To facilitate this sharing it has a programmable register, ELCR (Edge / Level trigger control register) by which certain interrupt inputs can be programmed as edge (low to high except for RTC) or level (the level is active low). The determination of edge or level is done during initial configuration of EISA system by reading EISA add in boards from the interrupt description data structures. The solution is to have programmable logic at the interrupt inputs so that 82489DX is compatible with EISA ISP. This will introduce one more register and logic to support this. This should be an 11-bit programmable register and an array of ExOR logic (12 ExOR gates or equivalent PLD). The ISP allows programmability of the following interrupts.

INT3 INT4 INT5 INT6 INT7 INT9 INT10 INT11 INT12 INT14 INT15. In addition to the above 11 interrupts, it fixes **INT8** to be active low edge triggered interrupt. INT8 is the only case where it is active low edge triggered type. So the following logic can be used to add programmability in 82489DX based EISA system. Before connecting these 11 interrupt lines directly (#INT8 which is from Real Time Clock is always active low edge triggered. #INT8 can be passed through an inverter since there is no need for programmability) to the 82489DX they should pass through an array of 11 Ex_OR gates. One input of Ex_OR gate connects to the corresponding INT pin and other input connects to a bit of programmable register. The output of Ex_OR gate is connected to 82489DX. The idea of Ex_OR is to use as a controlled inverter.



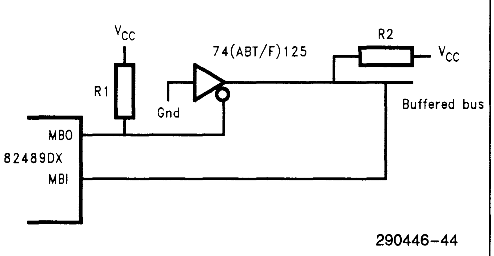
290446-43

INTIN are the interrupt inputs to the 82489DX and INT are the system interrupt. The Ex_OR gating register is programmed after EISA configuration is found from add in boards as how these interrupt lines are going to be used in that particular configuration. If a particular input is edge triggered, then the corresponding bit in the register is written with 0. If a particular input is level triggered, then the corresponding bit in the register is written with 1.

8259 by itself does not have polarity control whereas 8259 when implemented in EISA chipsets have the polarity control. Similarly APIC does not have by itself polarity register. So polarity register should be programmed as a part of system BIOS and not APIC BIOS.

Design Consideration 3

I_{CC} bus drive is an open drain bus with drive capacity of 4 mA only. Since data is transmitted at each I_{CC} clock, the "charging" of I_{CC} bus should be fast enough to ensure proper logic level at each clock edge. The I_{CC} bus needs pull up resistors since it is open drain bus. Since the drive is only 4 mA, the pull up resistor value can not be less than 5V/4 mA. This being the limit of the resistor value, the length and the characteristics of the I_{CC} trace forces a capacitance value. Both the resistor and capacitance brings a RC time constant to the I_{CC} bus waveform. So, Electrical consideration has to be given to and practice of controlled impedence should be exercised for layout of the I_{CC} bus. The length of the trace should be kept as minimum as possible. If the length of the I_{CC} bus can't be kept less, than say 6 inch, because of mechanical design of the system, the external line drivers should be added to I_{CC} bus and I_{CC} bus should be simulated with the added driver characteristics.



NOTE:
R1 can be typically 1K.
R2 is designed from the simulation results.

Design Consideration 4

This is related to ADS#, BGT# and CS# timings. For bus cycles not intended for 82489DX, (CS# = 1 where 82489DX is supposed to sample it), any change in CS# line while the ADS# is still active, may erroneously cause a RDY# returned from 82489DX. Anomolous behavior may result if for BGT# ties low cases

a) BGT# goes away just one clock after ADS# or

b) ADS# is still active, and CS# changes during this period.

For other cases anomolous behavior results if CS# changes when ADS# is still active. The following considerations are important from timing point of view. Always limit the pulse width of 82489DX ADS# to one CLKIN. Also avoid changing levels on BGT#/CS# line, when ADS# is active for cases being identified as BGT# tied low (BGT# sampled low when ADS# goes active). Also avoid changing levels on CS# line when BGT# is active.

Design Consideration 5

82489DX does not recognize the interrupt when an edge occurs at the interrupt input pin while interrupt is masked. When later it is unmasked there is no further edge and so 82489DX never passes that forgotten edge and that interrupt channel becomes unusable after that.

The recommendation is that first 82489DX should be unmasked and then the device interrupt should be enabled in the device register. By this, software can ensure that always an edge will occur after an interrupt is unmasked.

Design Consideration 6

Description: Edge triggered interrupts should not deassert their output till they are acknowledged by INTA cycle from CPU.

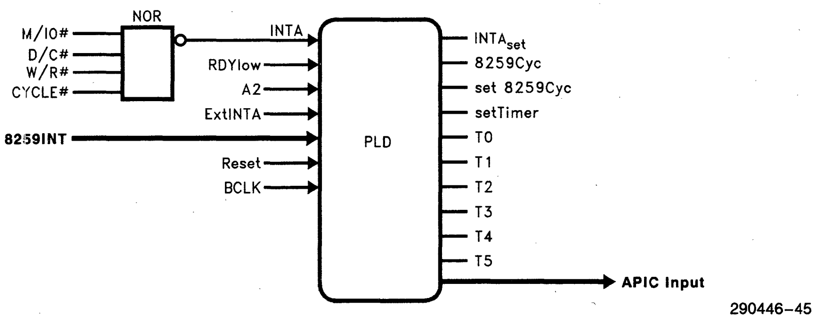
Issue:

82489DX employs glitch detection logic for edge triggered logic. To make sure the detected edge interrupt is not a glitch, 82489DX samples the input again before sending the interrupt message. The time difference between the first sampling of interrupt to be active and second sampling (just before sending the interrupt) is not a constant number. This is because the ICC bus might have been occupied by other messages. So, for example if during first sampling it was detected that INTIN0 and INTIN15 are both active and after sending INTIN0 it samples INTIN15 again before sending message for

INTIN15. But between this time ICC bus might have been occupied by other messages. So even if an edge triggered interrupt is held active high for a really long time and then brought low before INTA cycle, it is considered as a glitch. Because it may happen that the second sampling occurred just when the interrupt line got low.

Once the glitch detection circuitry found this "glitch", it goes back to the state where it will start sampling and waiting for an active edge to occur. This takes more than one clock cycle (CLK) and if the "glitch interrupt" generates an edge before that time after the second sampling of low level is done, then the edge is lost forever.

Since the time when the second sampling is done is unknown, the best way is to make sure the edge triggered interrupts do not deassert their outputs till they are acknowledged by INTA cycle from CPU. It is found that in some cases 8259 can generate brief active low pulses on its output. So the glue logic between 8259 and 82489DX input pin should make sure that 82489DX input pin is clear only after getting second interrupt acknowledge cycle. The glue logic should not just follow the 8259 output. Put in other words, after interrupt acknowledge cycle to 8259, if the 8259 input is seen active high, it should generate an edge at 82489DX input. Moreover, even if 8259 output goes low the glue logic should not lower its output since the only time when the glue logic can deassert its output is when it finds an interrupt acknowledge cycle for 8259. The following PLD equations and schematics serves as an example for the glue logic between 8259 and 82489DX.



BCLK is the standard EISA BCLK.
CYCLE# is an external signal to indicate ADS# is asserted earlier and RDY low is not yet returned.

APIC input = /T0 * /Reset * /APIC input * 8259INT * INTA_{set} ; Sample 8259 interrupt

+ /T0 * /Reset * APICinput * INTA_{set} ; Hold till it is cleared by delayed interrupt acknowledge

+ / INTA_{set} * 8259INT

Set 8259Cyc = /Reset * INTA * ExtINTA ; This INTA cycle is for 8259

8259Cyc = set8259Cyc * /T0 * /Reset ; Set 8259cyc will set 8259 cycle and T0 will clear it

+ /T0 * /Set8259Cyc * 8259Cyc * /Reset ; Hold 8259 cycle till T0 clears it

INTA_{set} = /Reset * /INTA_{set} * INTA ; wait for very first INTA cycle after reset

+ /Reset * INTA_{set} ; once first INTA cycle after Reset is found, set the INTA_{set}

Set timer = 8259cyc * /A2 * /RDYlow ; Start the timer at end of second INTA cycle

T0 = Set timer * /T5 * /Reset ; Set timer will set T0 and T5 will clear

+ /Set timer * T0 * /T5 * /Reset ; Till T5 clears it hold T0

T1 := T0 * /Reset * /T5 ; Follow T0 after one clock for setting but clear along with T0

T2 := T1 * /Reset * /T5 ; Follow T0 after two clock for setting but clear along with T0

T3 := T2 * /Reset * /T5 ; Follow T0 after three clock for setting but clear along with T0

T4 := T3 * /Reset * /T5 ; Follow T0 after four clock for setting but clear along with T0

T5 := T4 * /Reset ; Follow T0 after 5 clock for setting

290446-46

NOTES:
T1, T2, T3, T4 and T5 are clocked signals and others are combinatorials.
This circuit and PLD equations are given for concept clarification purpose. They are not tested.
INTA_{set} is needed so that some 8259 logic at power on activates its INT output to 1 and it deactivates its output after only 8259 initialization (which should happen after APIC initialization) and since APIC needs to detect rising edge at 8259, it is essential to follow the 8259 until first interrupt. This is the only occasion 8259 output will be just followed.

DIRECTIONS FOR EASY MIGRATION TO FUTURE INTEGRATED APIC

The following are the software programming directions Intel strongly recommends for easy migration from 82489DX to integrated APIC. The audience to this portion of the document are both hardware designers and firmware developers for APIC based systems. In the following discussions, the APIC BIOS is viewed functionally as two subsections 1) APIC BIOS which are all interrupt vector, priority, interrupt distribution related functions and the remaining portion of BIOS which is referred to part of system BIOS which is responsible for interrupt polarity programming, starting next processor, etc.

Note that the names APIC BIOS and APIC DRIVER are interchangeably used in the following discussion. Different Operating systems refer such functional module differently.

Consideration 1

Question: The logical destination register in future implemented APIC may have only 8 MSBs defined and 82489DX has 32 bits specified. Will this hinder binary level compatibility?

Response: In logical destination (flat addressing mode) 82489DX can go up to 32 CPUs whereas future APIC can go up to 8 CPUs with flat logical addressing mode. ***For binary compatibility, it is strongly recommended that 82489DX software use ONLY 8 MSB of logical destination register.***

Consideration 2

Question: The present day MP systems with external control ports for starting next processor may program those external control ports for starting next processor. APIC DRIVER may use external control ports for starting next processor. In future implementations of APIC, the starting of next processors may use more refined mechanisms which may not use external control ports. Will this introduce compatibility problem?

Response: Again, the starting of next processor is really part of MP system DRIVER and depending of the mechanism used to start next processor it will vary. In future implementation if starting next processor is done using new mechanisms, the starting next processor portion of MP DRIVER will be changed accordingly. Even though this will not result in any change in the APIC DRIVER which deals with interrupt priority, distribution, etc., the corresponding change will be needed in the starting application processors portion of DRIVER.

One possible method of implementing software is using version register. Version register is different in 489DX and future implementations of APIC. Taking care of these differences, such as mechanism for starting next processor, should be possible using version register.

Consideration 3

Question: APIC architecture, by its nature, seems to misinterpret spurious interrupts as genuine interrupts. That is, if an edge triggered interrupt goes inactive before interrupt acknowledge cycle, APIC, instead of giving spurious interrupt vector, gives genuine interrupt vector. Is it true that this is not the case with 8259? If that is the case, drivers which do not check device status registers for servicing the device may work with 8259 but may not work with APIC. Is this a compatibility problem?

Response: No, this is not true. Even with 8259 there is a time window in which a similar thing can happen. For example if interrupt goes inactive just after first INTA cycle but before second INTA cycle 8259 will also signal this spurious interrupt as genuine interrupt. So drivers which do not check device status registers may also fail with 8259.

Our strong recommendation to device drivers is to read device status register before servicing the device. If the device status register indicates that there is no valid source of the interrupt, the service routine should just issue EOI and return. It should not service the device. This should take care of the new drivers that will be written for APIC. To coexist with 8259, the APIC interrupt input connected to 8259 will be programmed for virtual wire mode. In virtual wire mode, the time window of 8259 will apply. So the driver will behave same way as it was behaving with 8259.

Consideration 4

Question: EISA system has active low level polarity. 82489DX itself does not have polarity control register to support this EISA feature. Implementations using external polarity register may implement the polarity register at different address. Will this introduce a problem for achieving the goal of single binary?

Response: 8259 by itself does not have polarity control whereas 8259 when implemented in EISA chipset has the polarity control. Similarly APIC does not have by itself polarity register. When implemented in ESC chipset, it will have polarity control register. So polarity register should be programmed as a part of EISA BIOS and not APIC BIOS. Since system BIOS or EISA BIOS should be able to take charge of changes, if any, to polarity control register. APIC BIOS should not be affected by differences in the address for polarity register.

Consideration 5

Question: 8259 recognizes the interrupt when an edge occurs at the interrupt input pin even if the interrupt was masked. So when the interrupt input is later unmasked, the interrupt is posted to the CPU. 82489DX does not register this edge and if interrupt happens when the interrupt is masked 82489DX just ignores the interrupt. When later it is unmasked there is no further edge and so 82489DX never passes that forgotten edge and that interrupt channel becomes unusable after that.

Response: When the interrupt is masked, logically interrupt controller should ignore whatever happens there. It is strongly recommended that first 82489DX should be unmasked and then the device interrupt should be enabled. By this sequence, software can ensure that always an edge will occur at the APIC input only after the interrupt is unmasked.

Please contact Intel for platform level specification in Multiprocessor system design using APIC.

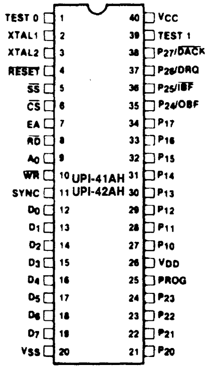
UPI-41AH/42AH UNIVERSAL PERIPHERAL INTERFACE 8-BIT SLAVE MICROCONTROLLER

- **UPI-41: 6 MHz; UPI-42: 12.5 MHz**
- **Pin, Software and Architecturally Compatible with all UPI-41 and UPI-42 Products**
- **8-Bit CPU plus ROM/OTP EPROM, RAM, I/O, Timer/Counter and Clock in a Single Package**
- **2048 x 8 ROM/OTP, 256 x 8 RAM on UPI-42, 1024 x 8 ROM/OTP, 128 x 8 RAM on UPI-41, 8-Bit Timer/Counter, 18 Programmable I/O Pins**
- **One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface**
- **DMA, Interrupt, or Polled Operation Supported**
- **Fully Compatible with all Intel and Most Other Microprocessor Families**
- **Interchangeable ROM and OTP EPROM Versions**
- **Expandable I/O**
- **Sync Mode Available**
- **Over 90 Instructions: 70% Single Byte**
- **Available in EXPRESS**
  — Standard Temperature Range
- **int$_e$ligent Programming Algorithm**
  — Fast OTP Programming
- **Available in 40-Lead Plastic and 44-Lead Plastic Leaded Chip Carrier Packages**
  (See Packaging Spec., Order #240800-001)
  Package Type P and N

The Intel UPI-41AH and UPI-42AH are general-purpose Universal Peripheral Interfaces that allow the designer to develop customized solutions for peripheral device control.

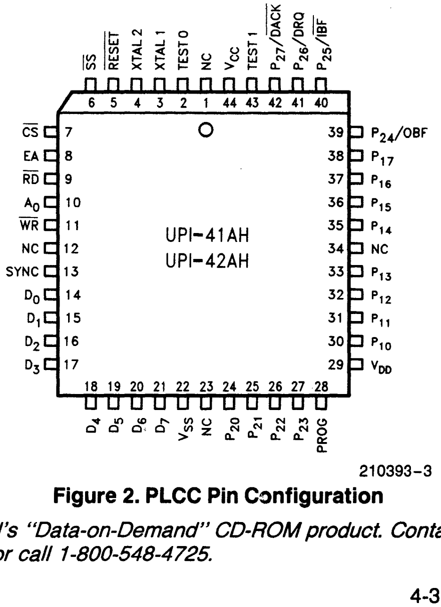
They are essentially "slave" microcontrollers, or microcontrollers with a slave interface included on the chip. Interface registers are included to enable the UPI device to function as a slave peripheral controller in the MCS Modules and iAPX family, as well as other 8-, 16-, and 32-bit systems.

To allow full user flexibility, the program memory is available in ROM and One-Time Programmable EPROM (OTP). All UPI-41AH and UPI-42AH devices are fully pin compatible for easy transition from prototype to production level designs.



210393-2

Figure 1. DIP Pin Configuration



210393-3

Figure 2. PLCC Pin Configuration

The complete document for this product is available on Intel's "Data-on-Demand" CD-ROM product. Contact your local Intel field sales office, Intel technical distributor, or call 1-800-548-4725.

November 1994
Order Number: 210393-008

UPI-C42/UPI-L42
UNIVERSAL PERIPHERAL INTERFACE
CHMOS 8-BIT SLAVE MICROCONTROLLER

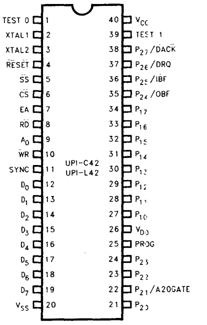
- **Pin, Software and Architecturally Compatible with all UPI-41 and UPI-42 Products**
- **Low Voltage Operation with the UPI-L42**
  — Full 3.3V Support
- **Integrated Auto A20 Gate Support**
- **Suspend Power Down Mode**
- **Security Bit Code Protection Support**
- **8-Bit CPU plus ROM/OTP EPROM, RAM, I/O, Timer/Counter and Clock in a Single Package**
- **4096 x 8 ROM/OTP, 256 x 8 RAM 8-Bit Timer/Counter, 18 Programmable I/O Pins**
- **DMA, Interrupt, or Polled Operation Supported**
- **One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface**
- **Fully Compatible with all Intel and Most Other Microprocessor Families**
- **Interchangeable ROM and OTP EPROM Versions**
- **Expandable I/O**
- **Sync Mode Available**
- **Over 90 Instructions: 70% Single Byte**
- **Quick Pulse Programming Algorithm**
  — Fast OTP Programming
- **Available in 40-Lead Plastic, 44-Lead Plastic Leaded Chip Carrier, and 44-Lead Quad Flat Pack Packages**
  (See Packaging Spec., Order #240800, Package Type P, N, and S)

The UPI-C42 is an enhanced CHMOS version of the industry standard Intel UPI-42 family. It is fabricated on Intel's CHMOS III-E process. The UPI-C42 is pin, software, and architecturally compatible with the NMOS UPI family. The UPI-C42 has all of the same features of the NMOS family plus a larger user programmable memory array (4K), integrated auto A20 gate support, and lower power consumption inherent to a CHMOS product.

The UPI-L42 offers the same functionality and socket compatibility as the UPI-C42 as well as providing low voltage 3.3V operation.

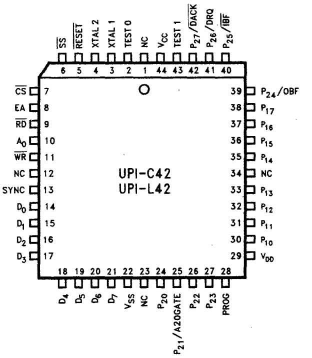
The UPI-C42 is essentially a "slave" microcontroller, or a microcontroller with a slave interface included on the chip. Interface registers are included to enable the UPI device to function as a slave peripheral controller in the MCS Modules and iAPX family, as well as other 8-, 16-, and 32-bit systems.

To allow full user flexibility, the program memory is available in ROM and One-Time Programmable EPROM (OTP).



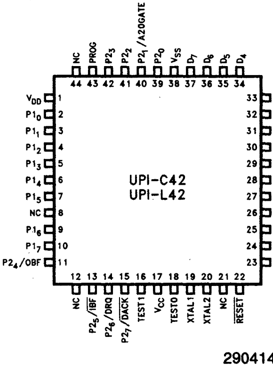
290414–1

Figure 1. DIP Pin Configuration



290414–2

Figure 2. PLCC Pin Configuration



290414–3

Figure 3. QFP Pin Configuration

The complete document for this product is available on Intel's "Data-on-Demand" CD-ROM product. Contact your local Intel field sales office, Intel technical distributor, or call 1-800-548-4725.

October 1994
Order Number: 290414-003

8XC51SL/LOW VOLTAGE 8XC51SL KEYBOARD CONTROLLER

80C51SL—CPU with RAM and I/O; V_{CC} = 5V ±10%

81C51SL— 16K ROM Preprogrammed with SystemSoft Keyboard Controller and Scanner Firmware. V_{CC} = 5V ±10%.

83C51SL— 16K Factory Programmed ROM. V_{CC} = 5V ±10%.

87C51SL— 16K OTP ROM. V_{CC} = 5V ±10%.

Low Voltage 80C51SL—CPU with RAM and I/O; V_{CC} = 3.3V ±0.3V

Low Voltage 81C51SL— 16K ROM Preprogrammed with SystemSoft Keyboard Controller and Scanner Firmware. V_{CC} = 3.3V ±0.3V.

Low Voltage 83C51SL— 16K Factory Programmed ROM. V_{CC} = 3.3V ±0.3V.

Low Voltage 87C51SL— 16K OTP ROM. V_{CC} = 3.3V ±0.3V.

- **Proliferation of 8051 Architecture**
- **Complete 8042 Keyboard Control Functionality**
- **8042 Style Host Interface**
- **Optional Hardware Speedup of GATEA20 and RCL**
- **Local 16 x 8 Keyboard Switch Matrix Support**
- **Two Industry Standard Serial Keyboard Interfaces; Supported via Four High Drive Outputs**
- **5 LED Drivers**
- **Low Power CHMOS Technology**
- **4-Channel, 8-Bit A/D**
- **Interface for up to 32 Kbytes of External Memory**
- **Slew Rate Controlled I/O Buffers Used to Minimize Noise**
- **256 Bytes Data RAM**
- **Three Multifunction I/O Ports**
- **10 Interrupt Sources with 6 User-Definable External Interrupts**
- **2 MHz-16 MHz Clock Frequency**
- **100-Pin PQFP (8XC51SL)**
  100-Pin SQFP (Low Voltage 8XC51SL)

The 8XC51SL, based on Intel's industry-standard MCS® 51 microcontroller family, is designed for keyboard control in laptop and notebook PCs. The highly integrated keyboard controller incorporates an 8042-style UPI host interface with expanded memory, keyboard scan, and power management. The 8XC51SL supports both serial and scanned keyboard interfaces and is available in pre-programmed versions to reduce time to market. The Low Voltage 8XC51SL is the 3.3V version optimized for even further power savings. Throughout the remainder of this document, both devices will generally be referred to as 51SL.

The 8XC51SL is a pin-for-pin compatible replacement for the 8XC51SL-BG. It does, however have some additional functionality. Those additional functions are as follows:

1. 16K OTP ROM: The 8XC51SL-BG had only 8K of ROM.
2. New Register Set: The 8XC51SL adds a second set of host interface registers available for use in supporting power management. This required an additional address line (A1) for decoding. To accommodate this, one V_{CC} pin was removed. However, in order to maintain compatibility with the -BG version, an enable bit for this new register set was added in configuration register 1. This allows the 8XC51SL to be drop in compatible to existing 8XC51SL-BG designs; no software modifications required.

NOTE:
The changes made to the V_{CC} pins require that all three V_{CC} pins be properly connected. Failing to do so could result in high leakage current and possible damage to the device.

The complete document for this product is available on Intel's "Data-on-Demand" CD-ROM product. Contact your local Intel field sales office, Intel technical distributor, or call 1-800-548-4725.

November 1994
Order Number: 272271-002

intel®

AP-366

APPLICATION NOTE

89C124FX Data/FAX Modem Chip Set

Reduction of Power Consumption

JIN LIEN LIN
TECHNICAL MARKETING ENGINEER

July 1992

Order Number: 292101-001

89C124FX Data/FAX Modem Chip Set Reduction of Power Consumption

INTRODUCTION

The 89C124FX Data/Fax Modem Chip Set Application Note provides the end user with applications and layout guidelines to reduce power consumption to a minimum when in the Power Down Mode.

GENERAL DESCRIPTION

When the 89C124FX is in the power down mode, the microcontroller (89C126FX) consumes very little power (less than 0.5 mW). However, the external memory, voltage regulators and peripherals draw excess current that makes the overall system power consumption more than 80 mW. This application note describes a method to reduce overall system power consumption to less than 1 mW when the 89C124FX is in the power down mode. Adding a power down feature in the microcontroller and reducing power sink to a minimum accomplishes this goal.

Three steps are required to reduce the overall system power consumption in the power down mode:

1. Detect power down in the microcontroller and isolate the power source from potential current sink from other components.
2. Inhibit the DC path from the power supply, through other components, to ground when the microcontroller is powered down.
3. Solve current drift problems due to floating inputs when power is removed from peripherals.

POWER DOWN DETECTION

Monitoring the clock output (CLKOUT, pin 65) from the microcontroller detects the power down condition. The CLKOUT pin is held high when the controller is in the power down mode. When power down is detected, the detector shuts off the +5V supply to all components except the microcontroller, RAM, and logic gates. The detector also shuts off power to the voltage regulators.

BLOCKING DC PATH

When the power down detector shuts off the bipolar switches, some of the device input pins become current sinks and drain current from the controller output pins when the output pins are high. These controller output pins are the CLKOUT and SCLK outputs. To eliminate these DC paths in the power down mode, add an inverter from the controller clock output (CLKOUT) to the AFE clock input (CLKIN) and use an AND gate to change the SCLK output to the AFE to a low.

CURRENT DRIFT

When the controller is in the power down mode, the SDATA pin becomes a floating input that can draw current in excess of 300 μA. Placing a 510 KΩ resistor between SDATA pin and ground solves this problem. Using a 100 KΩ resistor or lower may impede circuit functionality.

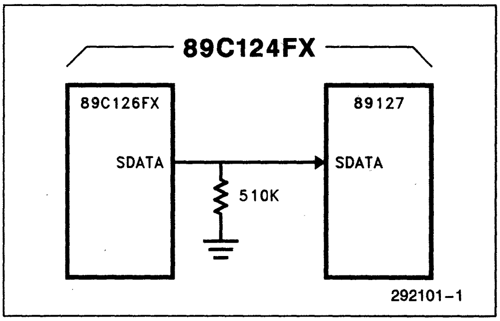


Figure 1. Placement of Current Drain Blocking Resistor

DESIGN TRADE OFF

The reduced power drain feature design trade-off, besides adding circuit complexity, is an additional 20 mW power consumption when the 89C124FX is active.

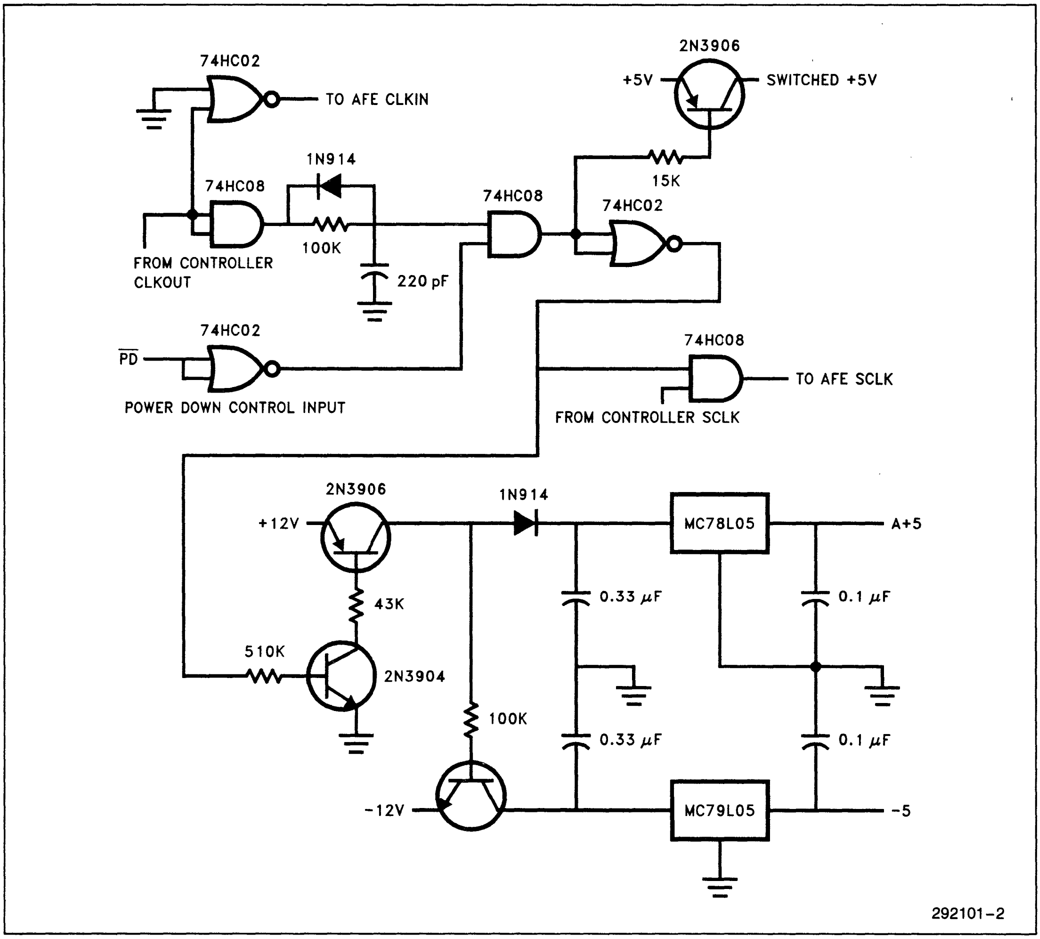


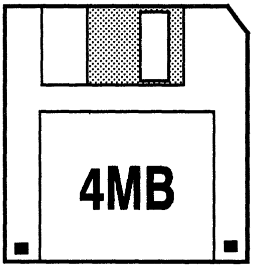
Figure 2. 89C124FX Power Consumption Circuit Modifications

intel®

AP-358

APPLICATION NOTE

Intel 82077SL for Super Dense Floppies



292093-1

KATEN A. SHAH
APPLICATION ENGINEER

September 1992

Order Number: 292093-002

Intel 82077SL for Super Dense Floppies

CONTENTS PAGE

INTRODUCTION

The evolution of the floppy has been marked in little over a decade by a significant increase in capacity accompanied by a noticeable decrease in the form factor from the early 8 inch floppy disks to the present day 3.5 inch floppy disks. This decade will also be remarkable as OEMs adopt "Super" dense floppies.

The most commonly seen floppies today are invariably one of the form factors – the 5.25″ or the 3.5″. Each form factor has several associated capacity ranges. The 5.25″ floppies available are: 180 KB (single density), 360 KB (double density) and 1.2 MB (high density). The 3.5″ floppies available are: 720 KB (double density) and 1.44 MB (high density). The emerging super dense floppies will evolve on the installed base of 3.5″ floppies. The latest member of this set is the 2.88 MB (extra density) floppy, pioneered by Toshiba. The cornerstone of market acceptance of newer drives is compatibility to the older family. The 2.88 MB (formatted) floppy drive allows the user to format, read from and write to the lower density diskettes.

As programs and data files get bigger, the demand for higher capacity floppies becomes obvious. There are several 3.5″ higher density drives available from various vendors with capacities well into the 20 MB range. NEC has introduced a 13 MB drive and companies such as Insite have introduced 20 MB drives. Both drives require servo-mechanisms to accurately position the head over the right track. NEC's drive has the standard floppy drive interface whereas Insite's interface is SCSI based. The market for these floppy drives will remain a niche unless they receive more OEM support.

Initiated by Toshiba's research and innovation of the higher density 4 MB floppy disk media, the market is headed towards the super dense floppy drive. After IBM's endorsement of the 4 MB (unformatted) floppy disk drives on their PS/2 model 57 and PS/2 model 90, several OEMs have shown a growing interest in "super" dense floppy disk drives. The latest DOS 5.0 supports the new 4 MB floppy media and BIOS vendors like Pheonix, AMI, Award, Quadtel, System Soft, and Microid all support the newer 4 MB floppy media.

PURPOSE

An important consideration to implement the 4 MB floppy drive is the floppy disk controller. Intel's highly integrated floppy disk controller, 82077AA/SL, has led the market in supporting the 4 MB floppy drive. Two ingredients are necessary to fully support these drives: 1 Mbps transfer rate and the perpendicular recording mode. This paper deals with a discussion of what the perpendicular mode is and how can a 4 MB floppy disk drive be implemented in a system using the 82077AA/SL.

PERPENDICULAR RECORDING MODE

Toshiba has taken the 2 MB floppy and doubled the storage capacity by doubling the number of bits per track. Toshiba achieved this by an innovative magnetic recording mode, called the vertical or the perpendicular recording mode. This mode utilizes magnetization perpendicular to the recording medium plane. This is in contrast to the current mode of longitudinal recording which uses the magnetization parallel to the recording plane. By making the bits stand vertical as opposed to on their side, recording density is effectively doubled, Figure 1. The new perpendicular mode of recording not only produces sharp magnetization transitions necessary at higher recording densities, but is also more stable.

The 4 MB disks utilize barium ferrite coated substrates to achieve perpendicular mode of magnetization. Current disks use cobalt iron oxide (Co-g-Fe_2O_3) coating for longitudinal recording. The barium ferrite ensures good head to medium contact, stable output and durability in terms of long use. High coercivity is required to attain high recording density for a longitudinal recording medium (coercivity specification of a disk refers to the magnetic field strength required to make an accurate record on the disk). A conventional head could not be used in this case; however, the barium ferrite disk has low coercivity and the conventional ferrite head can be used. The new combination heads include a pre-erase mechanism, i.e., the ferrite ring heads containing erase elements followed by the read/write head. These erase elements have deep overwrite penetration and ensure complete erasure for writing new data. The distance between the erase elements and the read/write head is about 200mm. This distance is important from the floppy disk controller point of view and will be discussed in later sections.

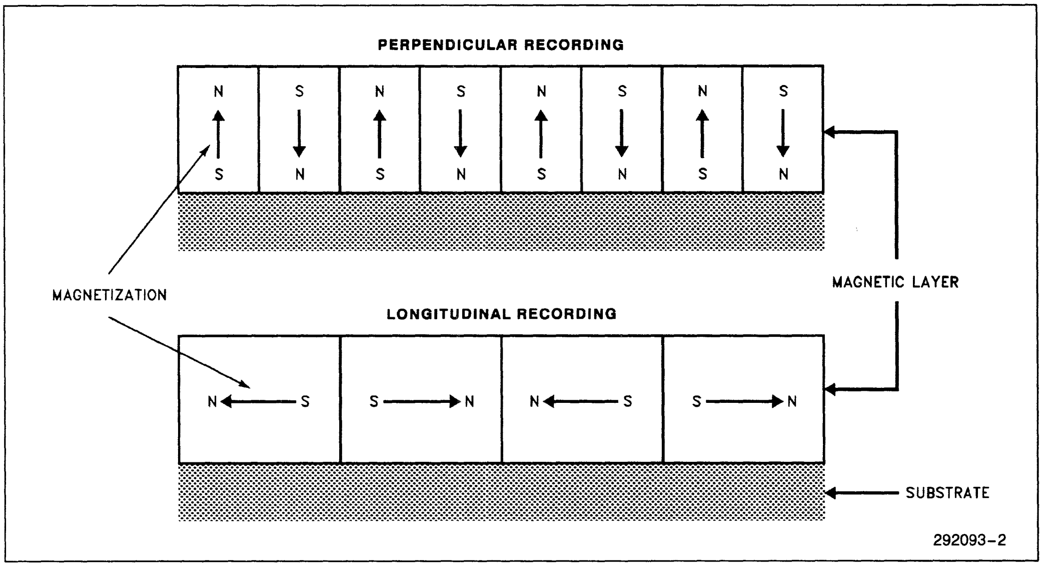


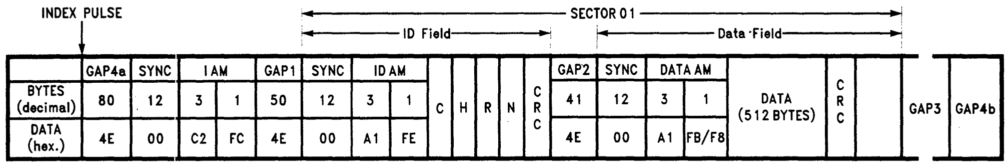
Figure 1. Perpendicular vs Longitudinal Recording

PERPENDICULAR DRIVE FORMAT AND SPECIFICATION

Figures 2a and 2b show the IBM drive format for both double density and perpendicular modes of recording. The main difference in recording format is the length of Gap2 between the ID field and the Data field. The main reason for the increased Gap2 length is the pre-erase head preceding the read/write head on the newer 4 MB floppy drives. The size of the data field is maintained at 512 KBytes standard. The increase in the capacity is implemented by increasing the number of sectors from 18 to 36. Table 1 shows the specifications of the various capacity 3.5″ drives.

PERPENDICULAR MODE COMMAND

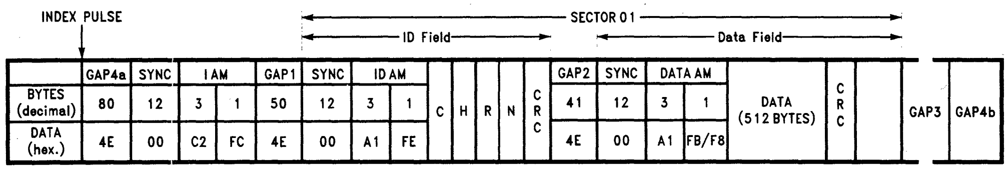
The current 82077AA/SL parts contain the "enhanced" perpendicular mode command as shown in Figure 3. This is a two byte command with the first byte being the command code (0x12H). The 2nd byte contains the parameters required to enable perpendicular mode recording. The former command (in the older 82077 parts) included only the WGATE and GAP bits. This command is compatible to the older mode where only the two LSBs are written. The enhanced mode allows system designers to designate specific drives as perpendicular recording drives. The second byte will be referenced as the PR[0:7] byte for ease of discussion. The following discusses the use of the enhanced perpendicular recording mode.



	GAP4a	SYNC	IAM		GAP1	SYNC	ID AM		C	H	R	N	CRC	GAP2	SYNC	DATA AM		DATA (512 BYTES)	CRC		GAP3	GAP4b
BYTES (decimal)	80	12	3	1	50	12	3	1						41	12	3	1					
DATA (hex.)	4E	00	C2	FC	4E	00	A1	FE						4E	00	A1	FB/F8					

292093-3

Figure 2a. Conventional IBM 1 MB and 2 MB Format (MFM)



	GAP4a	SYNC	IAM		GAP1	SYNC	ID AM		C	H	R	N	CRC	GAP2	SYNC	DATA AM		DATA (512 BYTES)	CRC		GAP3	GAP4b
BYTES (decimal)	80	12	3	1	50	12	3	1						41	12	3	1					
DATA (hex.)	4E	00	C2	FC	4E	00	A1	FE						4E	00	A1	FB/F8					

292093-4

Figure 2b. Perpendicular 4 MB Format (MFM)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
PERPENDICULAR MODE COMMAND										
Command	W	0	0	0	1	0	0	1	0	Command Code
	W	OW	0	D3	D2	D1	D0	GAP	WGATE	PR

Figure 3. Perpendicular Mode Command

Table 1. Specifications of FDDs

Various Parameters Used in the Different Kinds of FDDs.	5.25″ 360 KB	5.25″ 1.2 MB	3.5″ 720 KB	3.5″ 1.44 MB	3.5″ 2.88 MB
Number of Cylinders	40	80	80	80	80
Sectors/Track	9	15	9	18	36
Formatted Capacity	354 KB	1.2 MB	720 KB	1.44 MB	2.88MB
Unformatted Capacity	360 KB	1.6 MB	1 MB	2 MB	4 MB
Rotation Speed (rpm) XT AT	300 360	360	300	300	300
Track Density (tpi)	48	96	135	135	135
Recording Density (bpi)	5876	9870	8717	17432	34868
Data Transfer Rate (Mbps) XT AT	0.25 0.30	0.5	0.25	0.5	1
Gap Length for Read/Write	42	42	27	27	56
Gap Length for Format	80	80	84	84	83
Sector Size	512 KB	512 KB	512 KB	512 KB	512 KB
Density Notation	DD/DS	HD/DS	DD/DS	HD/DS	ED/DS

The following describes the various functions of the programmed bits in the PR:

OW If this bit is not set high, all PR[2:5] are ignored. In other words, if OW = 0, only GAP and WGATE are considered. In order to select a drive as perpendicular, it is necessary to set OW = 1 and select the Dn bit.

Dn This refers to the drive specification bits and corresponds to PR[2:5]. These bits are considered only if OW = 1. During the READ/WRITE/FORMAT command, the drive selected in these commands is compared to Dn. If the bits match then perpendicular mode will be enabled for that drive. For example, if D0 is set then drive 0 will be configured for perpendicular mode.

GAP This alters the Gap2 length as required by the perpendicular mode format.

WGATE Write gate alters timing of WE to allow for pre-erase loads in perpendicular drives.

The VCOEN timing and the length of the Gap2 field (explained above) can be altered to accommodate the unique requirements of the 4 MB floppy drives by GAP and WGATE bits of the PR. Table 2 describes the effects of the GAP and WGATE bits for the perpendicular command.

82077AA/SL's PERPENDICULAR MODE SUPPORT

The 82077AA and 82077SL both support 4 MB recording mode. The 82077SL has power management features included as well. Both AA and SL product lines have three versions each out of which two of the versions support the 4 MB floppy drives. The 82077AA-1, 82077AA, 82077SL, and 82077SL-1 all support the 4 MB floppy drives. A single command puts the 82077AA/SL into the perpendicular mode. This mode also requires the data rate to be set at 1 Mbps. The FIFO that is unique to Intel's 82077AA/SL parts may become necessary to remove the host interface bottleneck due to the higher data rate. The 4 MB floppy disk drives are downward compatible to 1 MB and 2 MB floppy diskettes. The following discussion explains the implications of the new 4 MB combination head and the functionality of the perpendicular mode command.

Table 2. Effects of GAP and WGATE Bits

GAP	WGATE	Mode	VCO Low Time after Index Pulse	Length of Gap2 Format Field	Portion of Gap2 Written by Write Data Operation	Gap2 VCO Low Time for Read Operations
0	0	Conventional	33 Bytes	22 Bytes	0 Bytes	24 Bytes
0	1	Perpendicular (Data Rate = 500 kbps)	33 Bytes	22 Bytes	19 Bytes	24 Bytes
1	0	Conventional	33 Bytes	22 Bytes	0 Bytes	24 Bytes
1	1	Perpendicular	33 Bytes	41 Bytes	38 Bytes	43 Bytes

The implementation of 4 MB drives requires understanding the Gap2 (see Figures 2a and 2b) and VCO timing requirements unique to these drives. These new requirements are dictated by the design of the "combination head" in these drives. Rewriting of disks in the 4 MB drives requires a pre-erase gap to erase the magnetic flux on the disk preceding the writing by the read/write gap. The read/write gap in the 4 MB drive does not have sufficient penetration (as shown in Figure 4a) to overwrite the existing data. In the conventional drives, the read/write gap had sufficient depth and could effectively overwrite the older data as depicted in Figure 4b. It must be noted that it is necessary to write the conventional 2 MB media in the 4 MB drive at 500 Kbps perpendicular mode. This ensures proper erasure of existing data and reliable write of the new data. The pre-erase gap in the 4 MB floppy drives is activated only during format and write commands. Both the pre-erase gap and read/write gap are activated at the same time.

As shown in Figure 4a, the pre-erase gap precedes the read/write gap by 200mm. This distance translated to bytes is about 38 bytes at a data rate of 1 Mbps and 19 bytes at 500 Kbps. Whenever the read/write gap is enabled by the Write Gate signal the pre-erase gap is activated at the same time.

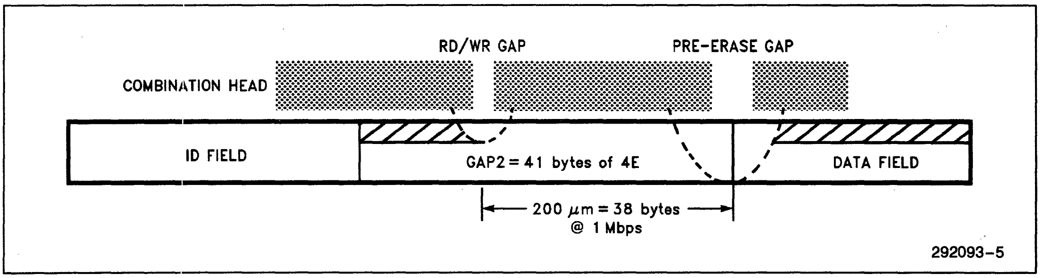


Figure 4a. Head Design for the 4 MB Perpendicular Mode

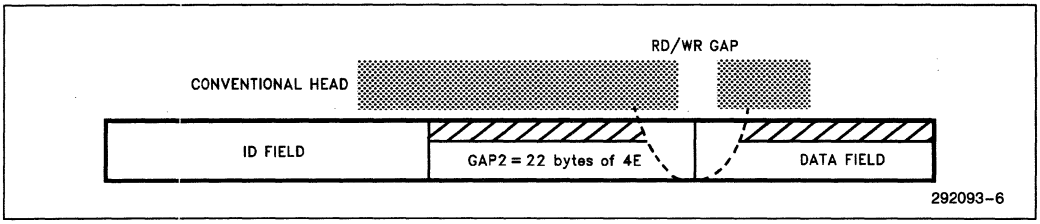


Figure 4b. Head Design for the Conventional 2 MB Mode

In conventional drives, the Write Gate is asserted at the beginning of the sync field, i.e., when the read/write is at the beginning of the data field. The controller then writes the new sync field, data address mark, data field and CRC (see Figure 2a). With the combination head, the read/write gap must be activated in the Gap2 field to ensure proper write of the new sync field. To accommodate both the distance between the pre-erase gap and read/write gap and the head activation and deactivation time, the Gap2 field is expanded to a length of 41 bytes at 1 Mbps (see Figure 2b). Since the bit density is proportional to the data rate, 19 bytes will be written in the Gap2 field at 500 Kbps data rate in the perpendicular mode.

On the read back by the 82077AA/SL, the controller must begin the synchronization at the beginning of the sync field. For conventional mode, the internal PLL VCO is enabled (VCOEN) approximately 24 bytes from the start of the Gap2 field. However, at 1 Mbps perpendicular mode the VCOEN goes active after 43 bytes to accomodate the increased Gap2 field size. For each case, a 2 byte cushion is maintained from the beginning of the sync field to avoid write splices caused by motor speed variation.

It should be noted that none of the alterations in Gap2 size, VCO timing or Write Gate timing affect the normal program flow. Once the perpendicular command is invoked, 82077AA/SL behaviour from the user standpoint is unchanged.

PROGRAMMING PERPENDICULAR MODE

Figures 5a and 5b show a flowchart on how the perpendicular recording mode is implemented on the 82077AA/SL. The perpendicular mode command can be issued during initialization. As shown in Figure 5a the perpendicular command stores the PR value internally. This value is used during the data transfer commands for configuration in order to deal with the perpendicular drives. Table 2 shows how the Gap2 length, VCOEN timing or Write Gate timing is affected. The OW bit is also tested for in this part of the loop. The enhanced perpendicular mode is enabled by setting the OW = 1, setting the Dn bits corresponding to the installed perpendicular drive high and leaving PR[0:1] = '00'.

As shown in Figure 5b, the Gap2 length is initially set to the conventional length of 22 bytes. Next the PR[0:1] bits (GAP, WGATE) are checked if they are set to '00'. If the PR[0:1] bits are set to '10' then, perpendicular mode is disabled and conventional mode is retained. If the PR[0:1] = '01' or '11' the VCOEN is

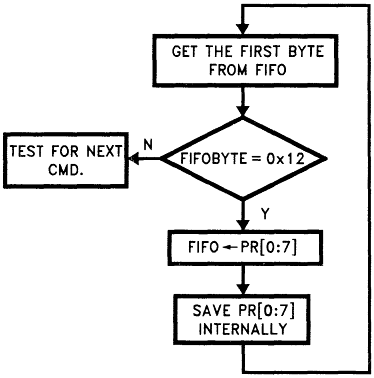
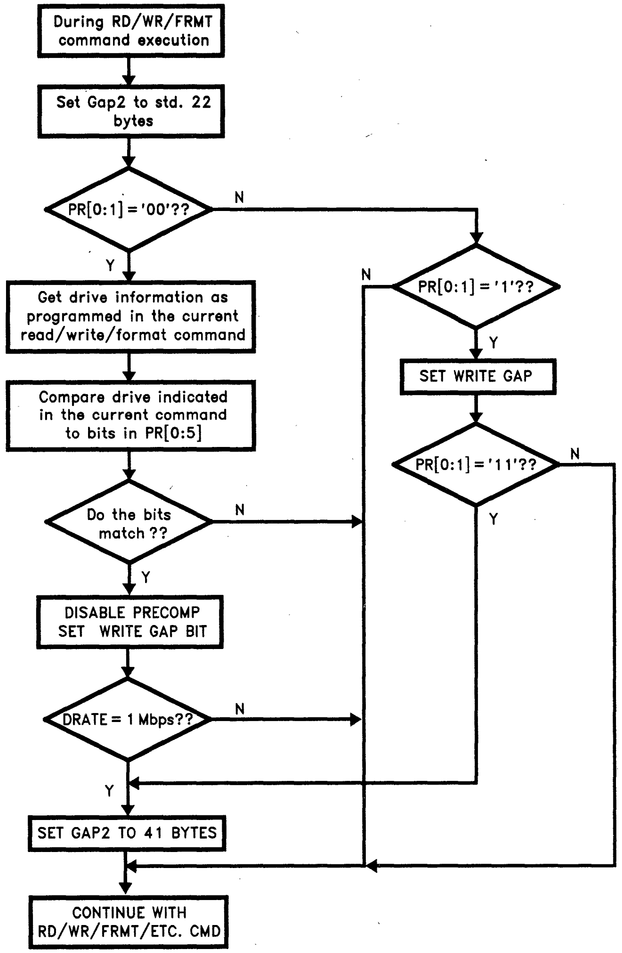


Figure 5a. Perpendicular Command Handling


During RD/WR/FRMT command execution
Set Gap2 to std. 22 bytes
PR[0:1] = '00'??
N
Y
PR[0:1] = '1'??
N
Y
Get drive information as programmed in the current read/write/format command
SET WRITE GAP
Compare drive indicated in the current command to bits in PR[0:5]
PR[0:1] = '11'??
N
Y
Do the bits match ??
N
Y
DISABLE PRECOMP SET WRITE GAP BIT
DRATE = 1 Mbps??
N
Y
SET GAP2 TO 41 BYTES
CONTINUE WITH RD/WR/FRMT/ETC. CMD

292093–8

Figure 5b. During Data Transfer Commands

set to activate 43 bytes or 24 bytes from the start of the Gap2 field, depending on the value as shown in Table 2. After this, PR[0:1] = '11' is checked; if not true (programmed '01') the program is exited with only the VCOEN timing being set for perpendicular mode. If true, however, the Gap2 length is set up for perpendicular mode (note: this is done independent of the data rate). It must be noted that if the PR[0:1] bits are set to '11' then it is up to the user to disable precompensation before accessing perpendicular drives. The other branch of the flowchart refers to setting of PR[0:1] to '00'. In this case, the perpendicular command will have the following effect:

1. If any of the Dn bits in PR[2:5] programmed high, then precompensation is automatically disabled (0 ns is selected for the specified drive regardless of the data rate) and VCOEN is set to activate appropriately. All the bits that are set low will enable the 82077 to be configured for conventional mode, i.e., exit the program without modifications (shown Figure 5b).
2. Next the data rate is checked for 1 Mbps. If the data rate is at 1 Mbps, then Gap2 length is set to 41 bytes, otherwise, the program is exited without setting up the Gap2 to 41 bytes.

It must be noted that if PR[2:5] are to be recognized in the command the OW bit must be set high. If this bit is low, setting of Dn bits will have no effect. Setting the OW bit will enable the storage of the Dn bit. Also setting PR[0:1] to any other value than '00' will override anything written in the Dn bits. In other words, setting PR[0:1] to a value other than '00' enables the effect of that for all drives. It must be noted that if PR[0:1] bits are set to a value other than '00' then it is recommended not to use the enhanced command mode, i.e., all other bits should be zero. Consider the following examples:

a. PR[0:7] = 0x84; This is the way to use the command in the enhanced mode. In this case, the OW = 1 and D0 is set high. During the data transfer command, if D0 is selected it will be automatically configured for perpendicular mode. If D1 is accessed, however, it will be configured for conventional mode. Similarly, if PR[0:7] = 0x88 then D1 is configured for perpendicular mode and D0 is configured for conventional mode. Software resets do not clear this mode.

b. PR[0:7] = 0x03; This is the way to use the command in the old mode. If the user decides to use this mode, then it must be noted that the command has to be issued before every data transfer command. Also when used this way, all the drives are configured for perpendicular mode. The user must also remember to disable precompensation and set the data rate to 1 Mbps while accessing the perpendicular drive in the system. Any software reset clears the command.

c. PR[0:7] = 0x87; In this case, the OW = 1, D0 = 1 and PR[0:1] = 11. This may be called a mixed mode and should be refrained from usage. This is similar to setting PR[0:7] = 0x03, because setting PR[0:1] high overrides automatic configuration. In this case the user has to be aware that precompensation must be disabled and the data rate must be set to 1 Mbps while accessing drive 0. After software reset, bits GAP and WGATE will be cleared, but OW and D0 will retain their previously set values. In other words, after software reset, the part will see PR[0:7] = 0x84. Evidently, this would cause problems and, therefore, it is recommended this mode *not* be used.

d. PR[0:7] = 0x80; In this case, the OW = 1, Dn = 0 and PR[0:1] = 00. This has the effect of clearing the perpendicular mode command without doing a hardware reset. Another way to do this would be to set PR[0:7] = 0x02; this can then be used to temporarily disable perpendicular mode configuration without affecting the previously programmed Dn values. Software reset following this will reenable the previously programmed enhanced mode command.

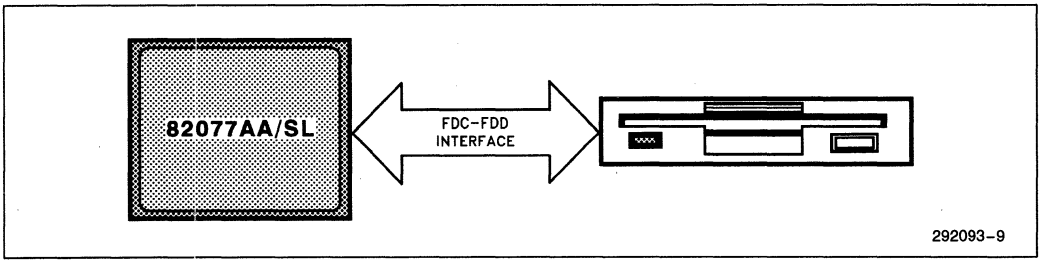
Using the enhanced perpendicular command removes the requirement of issuing the perpendicular command for each data transfer command and manually setting the perpendicular configuration.

"Software" RESETs (via DOR or DSR registers) will only clear the PR[0:1] values to '0'. Dn bits will retain their previously programmed values. "Hardware" RESETs will clear all the programmed bits including OW and Dn bits to '0'. The status of these bits can be determined by issuing the dumpreg command and checking the 8th result byte. This byte will contain the programmed values of the Dn and PR[0:1] bits as shown in Figure 6. The OW bit is *not* returned in this result byte.

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
DUMPREG COMMAND										
Command	R	Eighth Result Byte LOCK	0	D3	D2	D1	D0	GAP	WGATE	

Figure 6. Dumpreg Command

INTERFACE BETWEEN 82077AA/SL AND THE DRIVE



There is currently no industry-wide standard for the FDC to FDD interface. There are numerous floppy drive vendors, each with their own modes and interface pins to enable 4 MB perpendicular mode. The drive interface not only varies from manufacturer to manufacturer but also within a manufacturer's product line. The differences on the interface mainly originate from configuring the floppy drive into the 4 MB mode. Depending on the drive, the differences can create problems of daisy-chaining a 4 MB drive with the standard 1 MB and 2 MB drives. Of course, for laptops this is not a problem since most of them use a single floppy drive. Lack of an industry standard makes it necessary to look at each drive and build a interface for that particular drive.

The following is a brief discussion about some of the floppy drives available in the market and how these can be interfaced with the 82077AA/SL. It is important to note that although a manufacturer's name may be given in connection with the interface described, Intel does not guarantee that the interface discussed will apply to all the drives from that manufacturer. The main goal is introduce to the reader how to interface the 82077AA/SL with a 4 MB floppy drive.

Previously, for the conventional 1 MB and 2 MB AT mode drives, a single Density Select input was used by floppy drives to select between high density and low density drives. A high on this input enabled high density operation (500 Kbps) whereas a low enabled low density operation (300 Kbps/250 Kbps). This signal was asserted high or low by the floppy disk controller depending on the data rate programmed. For the 4 MB operation, there are two inputs defined by the floppy drive manufacturers. The polarity of these inputs enables the selected density operation. Implementing this requires at least 1 new pin to be defined on the FDC-FDD interface. Most floppy vendors have elected to take pin 2 (originally density select) and redefine the polarity to conform to one of these new density select inputs and another pin to be the other density select input. However, the new density select on pin 2 is not compatible to the old density select input in many of the floppy drives. This precludes the user from daisy chaining 4 MB drives with conventional drives. Another problem is that the second density select pin varies on its location on the FDC-FDD interface from drive to drive.

The way that the BIOS determines what type of diskette is in what type of drive is by trial and error. The system tries to read the diskette at 250 Kbps; if it fails then it will set the data rate to higher value and retry. The BIOS does this until the right data rate is selected. This method will still be implemented for the 4 MB drives by some BIOS vendors. However, the 4 MB drives available today also have two media sense ID pins that relate to the user what type of media is present in the floppy drive. This information will also require two pins on the FDC-FDD interface. The location of these pins is once again variable from drive to drive.

Some manufacturers have circumvented the entire standardization problem by including an auto configuration in the drive. In these cases, the type of floppy put into the drive is sensed by the hole (each 4/2/1 MB diskette has a hole in different locations identifying it) on the diskette. Then the drive automatically sets itself up for this mode. The BIOS must obviously set up the floppy disk controller for the correct data rate which could be done if the media sense ID was read and decoded as to the data rate. Due to lack of extra pins on the even side of the floppy connector the newer locations of some of the functions are migrating to the odd pins (previously all grounded). Some drive manufacturers have even made this configurable via jumpers. For instance, the new TEAC drives have a huge potpourri of configurations that would satisfy the appetite of some of the most finicky system interfaces.

The 82077AA/SL currently has two output pins DRATE0 and DRATE1 (pins 28 and 29 respectively) which directly reflect the data rate programmed in the DSR and CCR registers. These two pins can be used to select the correct density on the drive. These two can also be used with the combination of DENSEL to select the correct data rate. At the present time the 82077AA/SL does not support media sense ID. However, the user could easily make it readable directly by BIOS. The following is a discussion on what combination of DRATE0, DRATE1, and DENSEL could be used to interface to some of the currently available floppy drives.

1. TEAC 235J-600/Toshiba PD-211/Sony (Old Version)

These were among the first 4 MB drives available in the market. Each of them has a mode select input on pins 2 and 6. The polarity required for each different data rate is as shown below:

Data Rate	Capacity	DRATE1	DRATE0	MODSEL0 pin 2	MODSEL1 pin 6
1 Mbps	4 MB	1	1	1	0
500 Kbps	2 MB	0	0	0	1
300 Kbps/ 1 Mbps	4 MB	0	1	1	1
250 Kbps	1 MB	1	0	0	0

It is clear from the above that DRATE0 = MODSEL0 and MODSEL1 = DRATE1#. This would mean taking the drate signals onto pins 2 and 6 of the FDC-FDD interface. Unfortunately this solution requires an inverting gate. TEAC has recently, however, come out with a new version called TEAC 235J-3653. On this drive there are a number of possible configurations into which the drive can be put into, however, only the best way to interface to the 82077AA/SL will be discussed. The requirements are as shown below. This shows that HDIN = DENSEL (original signal for conventional drives) and EDIN = DRATE0. As suggested in the TEAC spec for method 1, the straps connected are MSC, HI2 (sets HDIN on pin 2), DC34 and EI6 (sets EDIN on pin 6). Pins 4, 29, and 33 are left open. Since pin 2 has the same polarity as the conventional drive requirement and the secondary input is connected via pin 6 (no connect on the conventional drives) daisy chaining this TEAC drive with a conventional drive does not cause any incompatibility. Figure 7 shows how the TEAC can be connected to the 82077AA/SL. It also shows daisy chaining of the TEAC drive with a conventional drive.

Data Rate	Capacity	DENSEL	DRATE1	DRATE0	HDIN pin 2	EDIN pin 6
1 Mbps	4 MB	1	1	1	X	1
500 Kbps	2 MB	1	0	0	1	0
300 Kbps/ 1 Mbps	4 MB	0	0	1	X	1
250 Kbps	1 MB	0	1	0	0	0

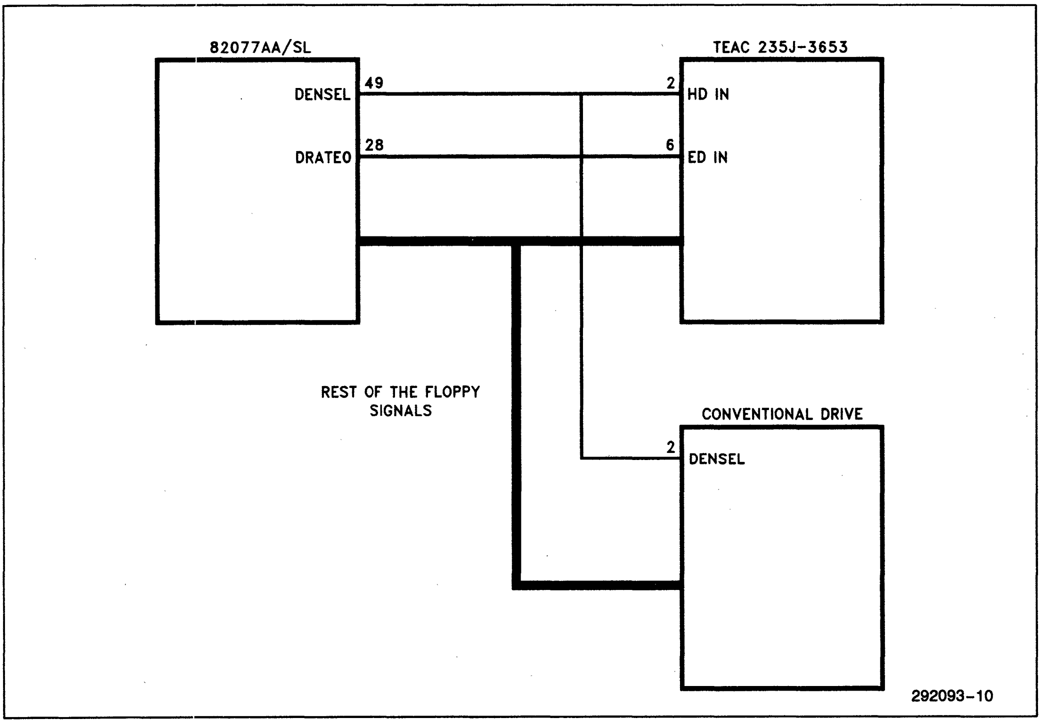


Figure 7. Interfacing 82077AA/SL to TEAC 235J-3653

2. Panasonic JU-259A (New Version)

This is Panasonic's new drive and has the HDIN signal on pin 2 and EDIN signal on pin 6. The requirements are shown below. This type of interface allows for daisy chaining the Panasonic drive with a conventional drive. The DENSEL signal can be connected to pin 2 and the DRATE0 should be connected to pin 6.

Data Rate	Capacity	DENSEL	DRATE1	DRATE0	HDIN pin 2	EDIN pin 6
1 Mbps	4 MB	1	1	1	1	1
500 Kbps	2 MB	1	0	0	1	0
300 Kbps/ 1 Mbps	4 MB	0	0	1	0	1
250 Kbps	1 MB	0	1	0	0	0

3. Mitsubishi MF356C (Model 252UG/788UG)

There are two models of this drive. The 252UG has DENSEL1 on pin 2 and DENSEL0 on pin 33, whereas the 788UG has DENSEL0 located on pin 2 and DENSEL1 located on pin 6. Via jumpers, it is possible to configure the drives to different polarity for the density select line. The following table shows the configuration for the 252UG in which jumper setting is 2MS = I/F and 4 MS = I/F.

Data Rate	Capacity	DENSEL	DRATE1	DRATE0	DENSEL1 pin 2	DENSEL0 pin 33
1 Mbps	4 MB	1	1	1	1	1
500 Kbps	2 MB	1	0	0	1	0
300 Kbps/ 1 Mbps	4 MB	0	0	1	0	1
250 Kbps	1 MB	0	1	0	0	0

The correct connection requirement is: DENSEL (from 82077AA/SL) = DENSEL1 and DRATE0 = DENSEL0. Although there are other configurations, this provides the best one, since daisy chaining is possible without any problem.

4. Epson SMD-1060

This drive has 3 different modes of operation. Mode B is the best and is similar to Mitsubishi's drives as described above. In this mode, HDI signal is connected to pin 2 and EDI is connected to pin 33. Mode B is enabled by inserting jumpers across 3-4 and 7-8 (SS01 B block) and 1-2 and 3-4 (SS03 block) for the drive with the power separated type (i.e., a connector for the floppy signals and another one for power supply) of 34-pin connector.

Data Rate	Capacity	DENSEL	DRATE1	DRATE0	HDI pin 2	EDI pin 33
1 Mbps	4 MB	1	1	1	1	1
500 Kbps	2 MB	1	0	0	1	0
300 Kbps/ 1 Mbps	4 MB	0	0	1	0	1
250 Kbps	1 MB	0	1	0	0	0

As demonstrated by the table, HDI = DENSEL and EDI = DRATE0. These connections would ensure daisy chaining capability without any problems.

5. Sony MP-F40W-14/15

The dash 14 and 15 are two drives from Sony that handle 4 MB requirements. The MP-F40W-14 has the DENSITY SELECT 1, DENSITY SELECT 0 on pins 2 and 33 respectively, whereas the MP-F40W-15 has the DENSITY SELECT 1, DENSITY SELECT 0 on pins 2 and 6 respectively. As it is obvious from the table below, daisy chaining is easily done if the 82077AA/SL is connected in the PS/2 mode (by tying IDENT low) with either type of drive, the only difference being the location of DENSITY SELECT 0.

Data Rate	Capacity	DENSEL PS/2 mode (IDENT = 0)	DRATE1	DRATE0	DENSITY SELECT1 pin 2	DENSITY SELECT0 pin 6/33
1 Mbps	4 MB	0	1	1	0	1
500 Kbps	2 MB	0	0	0	0	0
300 Kbps/ 1 Mbps	4 MB	1	0	1	1	1
250 Kbps	1 MB	1	1	0	1	0

If the drive is used in the PS/2 mode, then DENSITY SELECT1 = DENSEL and DENSITY SELECT0 = DRATE0. To use the drive in AT mode, DENSITY SELECT1 = DRATE1 and DENSITY SELECT0 = DRATE0, as shown below. However, daisy chaining is not possible.

Data Rate	Capacity	DENSEL PS/2 mode (IDENT = 0)	DRATE1	DRATE0	DENSITY SELECT1 pin 2	DENSITY SELECT0 pin 6/33
1 Mbps	4 MB	0	1	1	1	1
500 Kbps	2 MB	0	0	0	0	0
300 Kbps/ 1 Mbps	4 MB	1	0	1	0	1
250 Kbps	1 MB	1	1	0	1	0

6. Toshiba ND3571

Toshiba MB drive has the HD mode selection on pin 6 and ED mode selection on pin 2. This causes daisy chaining problems with conventional drives as shown in the figure below:

Data Rate	Capacity	DENSEL	DRATE1	DRATE0	ED Mode pin 2	HD Mode pin 6
1 Mbps	4 MB	1	1	1	1	1
500 Kbps	2 MB	1	0	0	0	1
300 Kbps/ 1 Mbps	4 MB	0	0	1	1	0
250 Kbps	1 MB	0	1	0	0	0

The DENSEL from the 82077 is connected to pin 6 and DRATE0 is connected to pin 2.

82077SL 4 MB DESIGN

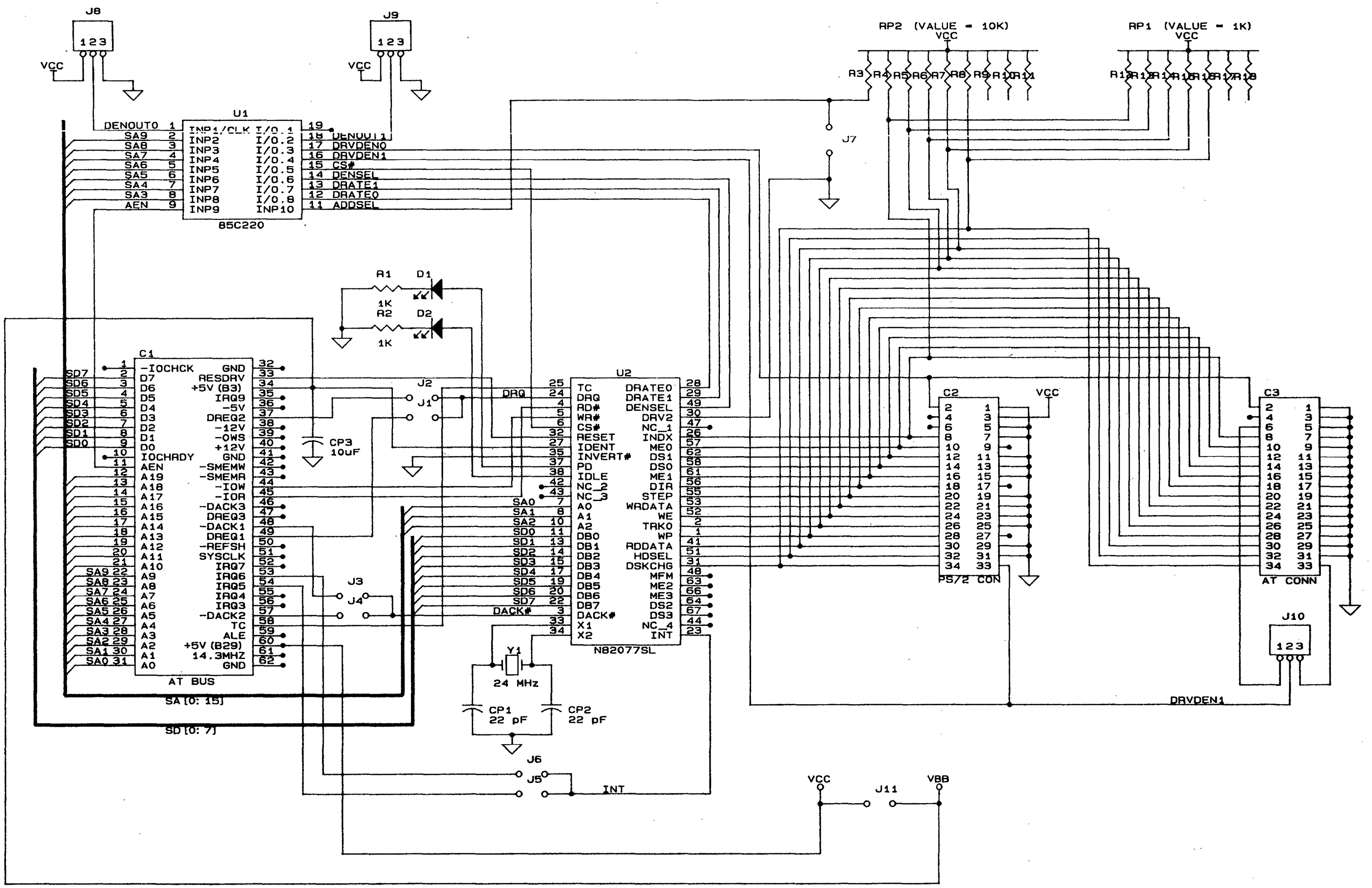
This section presents a design application of a PC/AT compatible floppy disk controller. The 82077SL integrates the entire PC/AT controller design with the exception of the address decode on a single chip. The schematic for this solution is shown in Figure 8. The chip select for the 82077SL is generated by a 85C220 μPLD that is programmed to decode addresses 03F0H through 03F7H when AEN is low. The programming equations for the μPLD is in the Intel's .ADF format and can be processed using the IPLSII compiler (available from Intel).

A floppy disk interface is provided by on-chip output buffers with a 40 mA sink capability. The outputs from the disk drive are terminated at the floppy disk controller with a 1 KΩ resistor pack. The 82077SL disk interface inputs contain a Schmitt trigger input structure for higher noise immunity. The host interface is a similar direct connection with on-chip 12 mA sink capable buffers on DB0–7, INT and DRQ.

The schematic shows eleven jumpers numbered J1 through J11. The table below describes the functions of these jumpers as well as their normal connections. The normal connections allow the BIOS to work without modification. In the normal mode, the 82077SL responds to DRQ2 and DACK2# as well as IRQ6. Depending on the type of drive interfaced to this board, the DENOUT0 and DENOUT1 signals can be tied. With the setting to 2–3 on J8 and J9, the default setting is DENSEL on DRVDEN0 and DRATE0 on DRVDEN1. PIN6/33 SELECT is used to set for pin 6 as the EDIN input. The J11 should always be closed. It can be used to measure the current consumption of 82077SL. J7 selects between the primary and secondary address spaces. There are two resistor packs used for pullups on input signals from the floppy drive interface. These resistors are rated at 1K. Please note that if using older 5.25″ drives, the pullup on some of them is 150Ω. Most modem 5.25″ drives use a 1K value. In order to ensure the correct value please refer to the floppy drive specification manual.

For further information, please contact your local Intel sales office.

Jumper	Description	Normal Connection
J1	DRQ1: DMA request 1 used with DACK1# to allow for DMA transfers	Open
J2	DRQ2: DMA request 2 used with DACK2# to allow for DMA transfers	Closed
J3	DACK1: DMA acknowledge 1 used with DRQ1 to allow for DMA transfers	Open
J4	DACK2: DMA acknowledge 2 used with DRQ2 to allow for DMA transfers	Closed
J5	IRQ5: Interrupt line 5 used to generate floppy interrupts	Open
J6	IRQ6: Interrupt line 6 used to generate floppy interrupts	Closed
J7	DRV2: Address selection (between 3FX and 37X address ranges)	Open
J8	DENOUT0: Used with DENOUT1 to select the values of DRVDEN1,0	2–3
J9	DENOUT1: Used with DENOUT0 to select the values of DRVDEN1,0	2–3
J10	PIN6/33 SELECT: Used to select between pin 6 and pin 33 for EDIN input	1–2 or 2–3
J11	V_{BB}/V_{CC}: Connection between two power layers	Closed



292093-11

Figure 8. 82077SL Evaluation Board

```
Designer: K. Shah
Company: Intel Corp.
Dept:    IMD Marketing
Date:    April '92
Rev.#:
% The µPLD used in the 82077SL Evaluation board design, Rev.#1.0. %
85C220 dip package

OPTIONS: TURBO = ON

PART:    85C220

INPUTS:
        SA9@2, % System Address Inputs %
        SA8@3,
        SA7@4,
        SA6@5,
        SA5@6,
        SA4@7,
        SA3@8,
        AEN@9,

        DENOUT0@1, % Maps the DRVDEN0 and DRVDEN1 to appropriate polarity table %
        DENOUT1@18, % Maps the DRVDEN0 and DRVDEN1 to appropriate polarity table %

        ADDSEL@11, % Selects between primary and secondary address spaces %

        DRATE0@12, % DRATE0 signal from the 82077SL %
        DRATE1@13, % DRATE1 signal from the 82077SL %
        DENSEL@14 % DENSEL signal from the 82077SL %

OUTPUTS:
        CS_@15, % 82077SL chip select signal %

        DRVDEN1@16, % Drive density signal connected to EDIN of the drive %
        DRVDEN0@17 % Drive density signal connected to HDIN of the drive %

NETWORK:
        % Inputs %

        SA9 = INP(SA9)
        SA8 = INP(SA8)
        SA7 = INP(SA7)
        SA6 = INP(SA6)
        SA5 = INP(SA5)
        SA4 = INP(SA4)
        SA3 = INP(SA3)
        AEN = INP(AEN)
        ADDSEL = INP(ADDSEL)
        DRATE0 = INP(DRATE0)
        DRATE1 = INP(DRATE1)
        DENSEL = INP(DENSEL)
        DENOUT0 = INP(DENOUT0)
        DENOUT1 = INP(DENOUT1)

        % Outputs %

        CS_ = CONF(CSeq, VCC)

        DRVDEN0 = CONF(DEN0eq, VCC)
        DRVDEN1 = CONF(DEN1eq, VCC)
```

```
EQUATIONS:

       % CS_is activated for 3F0-3F7 and 370-377 address spaces %
       CSeq = (AEN' * SA9 * SA8 * SA7' * SA6 * SA5 * SA4 * SA3' * ADDSEL'
              + AEN' * SA9 * SA8 * SA7 * SA6 * SA5 * SA4 * SA3' * ADDSEL)';

       % These are the signals generated on DRVDEN0 and DRVDEN1 for the FDC-FDD
         interface
       DENOUT1  DENOUT0  DRVDEN0  DRVDEN1
            0    0        DENSEL   DRATE0
            0    1        DENSEL'  DRATE0
            1    0        DRATE1   DRATE0
            1    1        DRATE0   DRATE1
       %

       DEN0eq = DENSEL * (DENOUT0' * DENOUT1') + DENSEL' * (DENOUT0 * DENOUT1')
               + DRATE1 * (DENOUT0' * DENOUT1) + DRATE0 * (DENOUT0 * DENOUT1);
       DEN1eq = DRATE1 * (DENOUT0 * DENOUT1) + DRATE0 * (DENOUT0' + DENOUT1');

END$
```

82077SL Application Note Revision Summary

The following changes have been made since revision 001:

Table 2 kBps was corrected to kbps.

Page 4-323 3. Mitsubishi MF356C description modified to read: "There are two models of this drive. The 252UG has DENSEL1 on pin 2 and DENSEL0 on pin 33, whereas the 788UG has DENSEL0 located on pin 2 and DENSEL1 located on pin 6. Via jumpers, it is possible to configure the drives to different polarity for the density select lines. The following table shows the configuration for the 252UG in which jumper setting is 2 MS = I/F and 4 MS = I/F."

Figure 8 Arrow added to diagram.

Page 4-328 Columns corrected to line up properly.

5

Flash Memory Components

28F001BX-T/28F001BX-B
1M (128K x 8) CMOS FLASH MEMORY

- **High Integration Blocked Architecture**
  - **One 8 KB Boot Block w/Lock Out**
  - **Two 4 KB Parameter Blocks**
  - **One 112 KB Main Block**
- **100,000 Erase/Program Cycles Per Block**
- **Simplified Program and Erase**
  - **Automated Algorithms via On-Chip Write State Machine (WSM)**
- **SRAM-Compatible Write Interface**
- **Deep-Powerdown Mode**
  - **0.05 μA I_{CC} Typical**
  - **0.8 μA I_{PP} Typical**
- **12.0V ±5% V_{PP}**
- **High-Performance Read**
  - **70/75 ns, 90 ns, 120 ns, 150 ns Maximum Access Time**
  - **5.0V ±10% V_{CC}**
- **Hardware Data Protection Feature**
  - **Erase/Write Lockout during Power Transitions**
- **Advanced Packaging, JEDEC Pinouts**
  - **32-Pin PDIP**
  - **32-Lead PLCC, TSOP**
- **ETOX II Nonvolatile Flash Technology**
  - **EPROM-Compatible Process Base**
  - **High-Volume Manufacturing Experience**
- **Extended Temperature Options**

Intel's 28F001BX-B and 28F001BX-T combine the cost-effectiveness of Intel standard flash memory with features that simplify write and allow block erase. These devices aid the system designer by combining the functions of several components into one, making boot block flash an innovative alternative to EPROM and EEPROM or battery-backed static RAM. Many new and existing designs can take advantage of the 28F001BX's integration of blocked architecture, automated electrical reprogramming, and standard processor interface.

The 28F001BX-B and 28F001BX-T are 1,048,576 bit nonvolatile memories organized as 131,072 bytes of 8 bits. They are offered in 32-pin plastic DIP, 32-lead PLCC and 32-lead TSOP packages. Pin assignment conform to JEDEC standards for byte-wide EPROMs. These devices use an integrated command port and state machine for simplified block erasure and byte reprogramming. The 28F001BX-T's block locations provide compatibility with microprocessors and microcontrollers that boot from high memory, such as Intel's MCS-186 family, 80286, i386™, i486™, i860™ and 80960CA. With exactly the same memory segmentation, the 28F001BX-B memory map is tailored for microprocessors and microcontrollers that boot from low memory, such as Intel's MCS-51, MCS-196, 80960KX and 80960SX families. All other features are identical, and unless otherwise noted, the term 28F001BX can refer to either device throughout the remainder of this document.

The boot block section includes a reprogramming write lock out feature to guarantee data integrity. It is designed to contain secure code which will bring up the system minimally and download code to the other locations of the 28F001BX. Intel's 28F001BX employs advanced CMOS circuitry for systems requiring high-performance access speeds, low power consumption, and immunity to noise. Its access time provides no-WAIT-state performance for a wide range of microprocessors and microcontrollers. A deep-powerdown mode lowers power consumption to 0.25 μW typical through V_{CC}, crucial in laptop computer, handheld instrumentation and other low-power applications. The RP# power control input also provides absolute data protection during system powerup or power loss.

Manufactured on Intel's ETOX process base, the 28F001BX builds on years of EPROM experience to yield the highest levels of quality, reliability, and cost-effectiveness.

The complete document for this product can be ordered by calling 1-800-548-4725. It is also available on Intel's "Data-on-Demand" CD-ROM product; contact your local Intel field sales office or Intel technical distributor.

November 1994
Order Number: 290406-006

28F200BX-T/B, 28F002BX-T/B 2-MBIT (128K x 16, 256K x 8) BOOT BLOCK FLASH MEMORY FAMILY

- x8/x16 Input/Output Architecture
  - — 28F200BX-T, 28F200BX-B
  - — For High Performance and High Integration 16-bit and 32-bit CPUs
- x8-only Input/Output Architecture
  - — 28F002BX-T 28F002BX-B
  - — For Space Constrained 8-bit Applications
- Upgradable to Intel's SmartVoltage Products
- Optimized High Density Blocked Architecture
  - — One 16-KB Protected Boot Block
  - — Two 8-KB Parameter Blocks
  - — One 96-KB Main Block
  - — One 128 KB Main Block
  - — Top or Bottom Boot Locations
- Extended Cycling Capability
  - — 100,000 Block Erase Cycles
- Automated Word/Byte Write and Block Erase
  - — Command User Interface
  - — Status Registers
  - — Erase Suspend Capability
- SRAM-Compatible Write Interface
- Automatic Power Savings Feature
  - — 1 mA Typical I_{CC} Active Current in Static Operation
- Hardware Data Protection Feature
  - — Erase/Write Lockout during Power Transitions
- Very High-Performance Read
  - — 60/80/120 ns Maximum Access Time
  - — 30/40/40 ns Maximum Output Enable Time
- Low Power Consumption
  - — 20 mA Typical Active Read Current
- Reset/Deep Power-Down Input
  - — 0.2 μA I_{CC} Typical
  - — Acts as Reset for Boot Operations
- Extended Temperature Operation
  - — −40°C to +85°C
- Write Protection for Boot Block
- Industry Standard Surface Mount Packaging
  - — 28F200BX: JEDEC ROM Compatible
    44-Lead PSOP
    56-Lead TSOP
  - — 28F002BX: 40-Lead TSOP
- 12V Word/Byte Write and Block Erase
  - — V_{PP} = 12V ±5% Standard
  - — V_{PP} = 12V ±10% Option
- ETOX III Flash Technology
  - — 5V Read
- Independent Software Vendor Support

The complete document for this product can be ordered by calling 1-800-548-4725. It is also available on Intel's "Data-on-Demand" CD-ROM product; contact your local Intel field sales office or Intel technical distributor.

November 1994
Order Number: 290448-004

28F200BL-T/B, 28F002BL-T/B 2-MBIT (128K x 16, 256K x 8) LOW POWER BOOT BLOCK FLASH MEMORY FAMILY

- Low Voltage Operation for Very Low Power Portable Applications
  - V_{CC} = 3.0V–3.6V
- Expanded Temperature Range
  - −20°C to +70°C
- x8/x16 Input/Output Architecture
  - 28F200BL-T, 28F200BL-B
  - For High Performance and High Integration 16-bit and 32-bit CPUs
- x8-only Input/Output Architecture
  - 28F002BL-T, 28F002BL-B
  - For Space Constrained 8-bit Applications
- Upgradeable to Intel's SmartVoltage Products
- Optimized High Density Blocked Architecture
  - One 16-KB Protected Boot Block
  - Two 8-KB Parameter Blocks
  - One 96-KB Main Block
  - One 128-KB Main Block
  - Top or Bottom Boot Locations
- Extended Cycling Capability
  - 10,000 Block Erase Cycles
- Automated Word/Byte Write and Block Erase
  - Command User Interface
  - Status Registers
  - Erase Suspend Capability
- SRAM-Compatible Write Interface
- Automatic Power Savings Feature
  - 0.8 mA Typical I_{CC} Active Current in Static Operation
- Very High-Performance Read
  - 150 ns Maximum Access Time
  - 65 ns Maximum Output Enable Time
- Low Power Consumption
  - 15 mA Typical Active Read Current
- Reset/Deep Power-Down Input
  - 0.2 μA I_{CC} Typical
  - Acts as Reset for Boot Operations
- Write Protection for Boot Block
- Hardware Data Protection Feature
  - Erase/Write Lockout during Power Transitions
- Industry Standard Surface Mount Packaging
  - 28F200BL: JEDEC ROM Compatible
    44-Lead PSOP
    56-Lead TSOP
  - 28F002BL: 40-Lead TSOP
- 12V Word/Byte Write and Block Erase
  - V_{PP} = 12V ±5% Standard
- ETOX III Flash Technology
  - 3.3V Read
- Independent Software Vendor Support

The complete document for this product can be ordered by calling 1-800-548-4725. It is also available on Intel's "Data-on-Demand" CD-ROM product; contact your local Intel field sales office or Intel technical distributor.

November 1994
Order Number: 290449-004

28F400BX-T/B, 28F004BX-T/B
4 MBIT (256K x16, 512K x8) BOOT BLOCK FLASH MEMORY FAMILY

- x8/x16 Input/Output Architecture
  - 28F400BX-T, 28F400BX-B
  - For High Performance and High Integration 16-bit and 32-bit CPUs
- x8-only Input/Output Architecture
  - 28F004BX-T, 28F004BX-B
  - For Space Constrained 8-bit Applications
- Upgradeable to Intel's Smart Voltage Products
- Optimized High Density Blocked Architecture
  - One 16-KB Protected Boot Block
  - Two 8-KB Parameter Blocks
  - One 96-KB Main Block
  - Three 128-KB Main Blocks
  - Top or Bottom Boot Locations
- Extended Cycling Capability
  - 100,000 Block Erase Cycles
- Automated Word/Byte Write and Block Erase
  - Command User Interface
  - Status Registers
  - Erase Suspend Capability
- SRAM-Compatible Write Interface
- Automatic Power Savings Feature
  - 1 mA Typical I_{CC} Active Current in Static Operation
- Very High-Performance Read
  - 60/80/120 ns Maximum Access Time
  - 30/40/40 ns Maximum Output Enable Time
- Low Power Consumption
  - 20 mA Typical Active Read Current
- Reset/Deep Power-Down Input
  - 0.2 μA I_{CC} Typical
  - Acts as Reset for Boot Operations
- Extended Temperature Operation
  - −40°C to +85°C
- Write Protection for Boot Block
- Hardware Data Protection Feature
  - Erase/Write Lockout During Power Transitions
- Industry Standard Surface Mount Packaging
  - 28F400BX: JEDEC ROM Compatible
    44-Lead PSOP
    56-Lead TSOP
  - 28F004BX: 40-Lead TSOP
- 12V Word/Byte Write and Block Erase
  - V_{PP} = 12V ±5% Standard
  - V_{PP} = 12V ±10% Option
- ETOX III Flash Technology
  - 5V Read

The complete document for this product can be ordered by calling 1-800-548-4725. It is also available on Intel's "Data-on-Demand" CD-ROM product; contact your local Intel field sales office or Intel technical distributor.

November 1994
Order Number: 290451-004

PRELIMINARY

28F008SA
8-MBIT (1-MBIT x 8) FlashFile™ MEMORY

Extended Temperature Specifications Included

- **High-Density Symmetrically Blocked Architecture**
  - **Sixteen 64-Kbyte Blocks**
- **Extended Cycling Capability**
  - **100,000 Block Erase Cycles**
  - **1.6 Million Block Erase Cycles per Chip**
- **Automated Byte Write and Block Erase**
  - **Command User Interface**
  - **Status Register**
- **System Performance Enhancements**
  - **RY/BY# Status Output**
  - **Erase Suspend Capability**
- **Deep-Powerdown Mode**
  - **0.20 μA I_{CC} Typical**
- **Very High-Performance Read**
  - **85 ns Maximum Access Time**
- **SRAM-Compatible Write Interface**
- **Hardware Data Protection Feature**
  - **Erase/Write Lockout during Power Transitions**
- **Industry Standard Packaging**
  - **40-Lead TSOP, 44-Lead PSOP**
- **ETOX III Nonvolatile Flash Technology**
  - **12V Byte Write/Block Erase**
- **Independent Software Vendor Support**
  - **Microsoft* Flash File System (FFS)**

Intel's 28F008SA 8-Mbit FlashFile™ Memory is the highest density nonvolatile read/write solution for solid state storage. The 28F008SA's extended cycling, symmetrically blocked architecture, fast access time, write automation and low power consumption provide a more reliable, lower power, lighter weight and higher performance alternative to traditional rotating disk technology. The 28F008SA brings new capabilities to portable computing. Application and operating system software stored in resident flash memory arrays provide instant-on, rapid execute-in-place and protection from obsolescence through in-system software updates. Resident software also extends system battery life and increases reliability by reducing disk drive accesses.

For high density data acquisition applications, the 28F008SA offers a more cost-effective and reliable alternative to SRAM and battery. Traditional high density embedded applications, such as telecommunications, can take advantage of the 28F008SA's nonvolatility, blocking and minimal system code requirements for flexible firmware and modular software designs.

The 28F008SA is offered in 40-lead TSOP (standard and reverse) and 44-lead PSOP packages. Pin assignments simplify board layout when integrating multiple devices in a flash memory array or subsystem. This device uses an integrated Command User Interface and state machine for simplified block erasure and byte write. The 28F008SA memory map consists of 16 separately erasable 64-Kbyte blocks.

Intel's 28F008SA employs advanced CMOS circuitry for systems requiring low power consumption and noise immunity. Its 85 ns access time provides superior performance when compared with magnetic storage media. A deep powerdown mode lowers power consumption to 1 μW typical thru V_{CC}, crucial in portable computing, handheld instrumentation and other low-power applications. The RP# power control input also provides absolute data protection during system powerup/down.

Manufactured on Intel's 0.8 micron ETOX process, the 28F008SA provides the highest levels of quality, reliability and cost-effectiveness.

*Microsoft is a trademark of Microsoft Corporation.

The complete document for this product can be ordered by calling 1-800-548-4725. It is also available on Intel's "Data-on-Demand" CD-ROM product; contact your local Intel field sales office or Intel technical distributor.

November 1994
Order Number: 290429-005

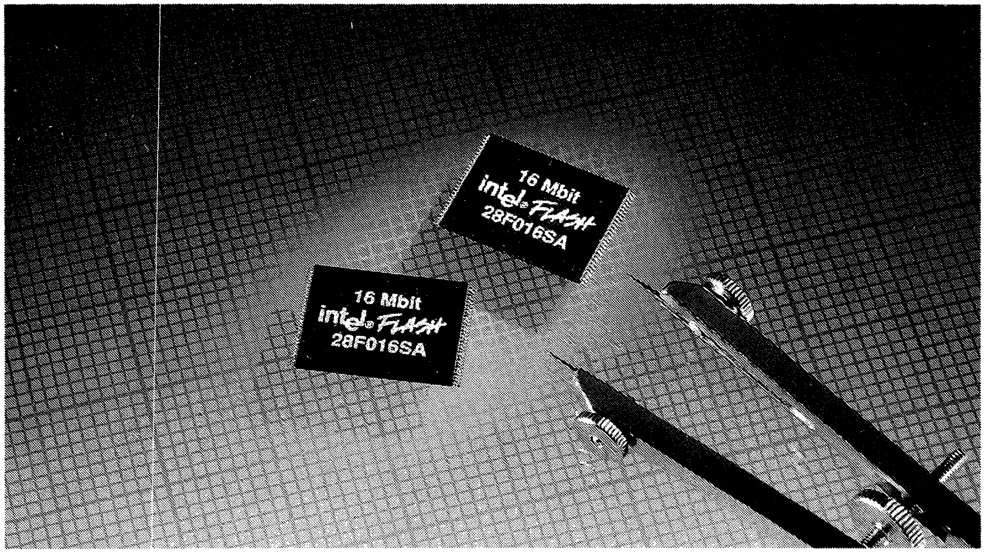
PRELIMINARY

28F016SA
16 MBIT (1 MBIT x 16, 2 MBIT x 8) FlashFile™ MEMORY

- **User-Selectable 3.3V or 5V V_{CC}**
- **User-Configurable x8 or x16 Operation**
- **70 ns Maximum Access Time**
- **28.6 MB/sec Burst Write Transfer Rate**
- **1 Million Typical Erase Cycles per Block**
- **56-Lead, 1.2mm x 14mm x 20mm TSOP Package**
- **56-Lead, 1.8mm x 16mm x 23.7mm SSOP Package**
- **Revolutionary Architecture**
  - **—Pipelined Command Execution**
  - **—Write During Erase**
  - **—Command Superset of Intel 28F008SA**
- **1 mA Typical I_{CC} in Static Mode**
- **1 μA Typical Deep Power-Down**
- **32 Independently Lockable Blocks**
- **State-of-the-Art 0.6 μm ETOX™ IV Flash Technology**

Intel's 28F016SA 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, extended temperature operation and high read/write performance, the 28F016SA enables the design of truly mobile, high-performance communications and computing products.

The 28F016SA is the highest density, highest performance non-volatile read/write solution for solid-state storage applications. Its symmetrically blocked architecture (100% compatible with the 28F008SA 8-Mbit FlashFile memory), extended cycling, extended temperature operation, flexible V_{CC}, fast write and read performance and selective block locking provide highly flexible memory components suitable for resident flash arrays, high-density memory cards and PCMCIA-ATA flash drives. The 28F016SA 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. Its high read performance and flexible block locking enable both storage and execution of operating systems and application software. Manufactured on Intel's 0.6 μm ETOX™ IV process technology, the 28F016SA is the most cost effective, highest density monolithic 3.3V FlashFile memory.



290489–1

ETOX™ and FlashFile™ are trademarks of Intel Corporation.

The complete document for this product can be ordered by calling 1-800-548-4725. It is also available on Intel's "Data-on-Demand" CD-ROM product; contact your local Intel field sales office or Intel technical distributor.

November 1994
Order Number: 290489-002

Intel486™ Microprocessor SmartDie™ Products

Intel486™ DX2 MICROPROCESSOR

SmartDie™ Product Specification

- **SL-Technology for Energy Efficiency**
  - **—Intel's System Management Mode**
  - **—Stop Clock, Auto HALT and Auto Idle Power Down**
- **Binary-Compatible with Large Software Base**
  - **—MS-DOS*, OS/2*, Windows***
  - **—UNIX* System V/Intel386™**
  - **—iRMX® Software, iRMK Kernels**
- **High Integration Enables On-Chip**
  - **—8 Kbyte Code and Data Cache**
  - **—Floating Point Unit**
  - **—Paged, Virtual Memory Management**
- **Easy to Use**
  - **—Built-In Self Test**
  - **—Hardware Debugging Support**
- **IEEE 1149.1 Boundary Scan Compatibility**
- **High-Performance Design**
  - **—40/50 MHz Core Speed Using 20/25 MHz Bus Clock at 3.3V**
  - **—RISC Integer Core with Frequent Instructions Executing in One Core Clock**
  - **—64/80 Mbyte/sec Burst Bus @40/50 MHz**
  - **—Dynamic Bus Sizing for 8-, 16- and 32-Bit Buses**
  - **—Complete 32-Bit Architecture**
- **Multiprocessor Support**
  - **—Cache Consistency Protocols**
  - **—Support for Second-Level Cache**
- **Intel SmartDie Product**
  - **—Full AC/DC Testing at Die Level**
  - **—0°C–80°C (Junction) Temperature Range**
  - **—40 MHz and 50 MHz Core Speeds @3.3V**

NOTICE: This document contains preliminary information on new products in production. It is valid for the devices indicated in the revision history. This specification is 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. 242202) for additional product information and specifications not found in this document.

*Other brands and names are the property of their respective owners.

PRELIMINARY

Intel486™ SX MICROPROCESSOR

SmartDie™ Product Specification

- **SL Technology for Energy Efficiency**
  - **— Intel's System Management Mode**
  - **— Stop Clock, Auto HALT and Auto Idle Power Down**
- **Binary-Compatible with Large Software Base**
  - **— MS-DOS*, OS/2*, Windows***
  - **— UNIX* System V/386**
  - **— iRMX Software, iRMK Kernels**
- **High Integration Enables On-Chip**
  - **— 8 Kbyte Code and Data Cache**
  - **— Paged, Virtual Memory Management**
- **Easy to Use**
  - **— Built-In Self Test**
  - **— Hardware Debugging Support**
  - **— Intel Software Support**
  - **— Extensive Third Party Software Support**
- **High-Performance Design**
  - **— Intel486 One Clock Instruction Core**
  - **— 80/100 Mbyte/sec Burst Bus at 25/33 MHz**
  - **— CHMOS V Process Technology**
  - **— Dynamic Bus Sizing for 8-Bit, 16-Bit and 32-Bit Buses**
- **Complete 32-Bit Architecture**
  - **— Address and Data Buses**
  - **— Registers**
  - **— 8-Bit, 16-Bit and 32-Bit Data Types**
- **Multiprocessor Support**
  - **— Multiprocessor Instructions**
  - **— Cache Consistency Protocols**
  - **— Support for Second-Level Cache**
- **IEEE 1149.1 Boundary Scan Compatibility**
- **Intel SmartDie Product**
  - **— Full AC/DC Testing at Die Level**
  - **— 0°C to +80°C (Junction) Temperature Range**
  - **— 25 MHz and 33 MHz Speeds @ 3.3V**

NOTICE: This document contains preliminary information on new products in production. It is valid for the devices indicated in the revision history. This specification is 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/Book (Order No. 242202) for additional product information and specifications not found in this document.

December 1994
Order Number: 271292-002

Intel486™ Microprocessors and Related Products

In 1994, Intel enhanced the Intel486™ microprocessor family with the IntelDX4™ Processor, the world's fastest 486, and the Write-Back Enhanced IntelDX2™ Processor. These new processors enable development of higher performance, lower-cost, entry-level desktop and mobile system products. Supporting PCIsets are also available, allowing the system designer to support the Peripheral Component Interconnect (PCI) bus. PCI allows a glueless interface for high performance peripherals to be placed onto a fast local bus. By providing multiple performance options, Intel makes desktop and mobile computing power affordable for more and more users.

This databook contains extensive information on Intel486 microprocessor families, OverDrive™ processors, supporting PCIsets, floppy and hard disk controllers, mobile peripheral products and flash memory components for the desktop and mobile family of Intel486 microprocessors. The datasheets and application notes contained in this databook include comprehensive charts, diagrams and instruction and hardware information for leading-edge 32-bit system development.

intel.

Order Number: 241731-002
Printed in USA/0195/14K/RRD/MY
Microprocessors

Printed on
Recycled Material

ISBN 1-55512-235-3
90000>
9 781555 122355