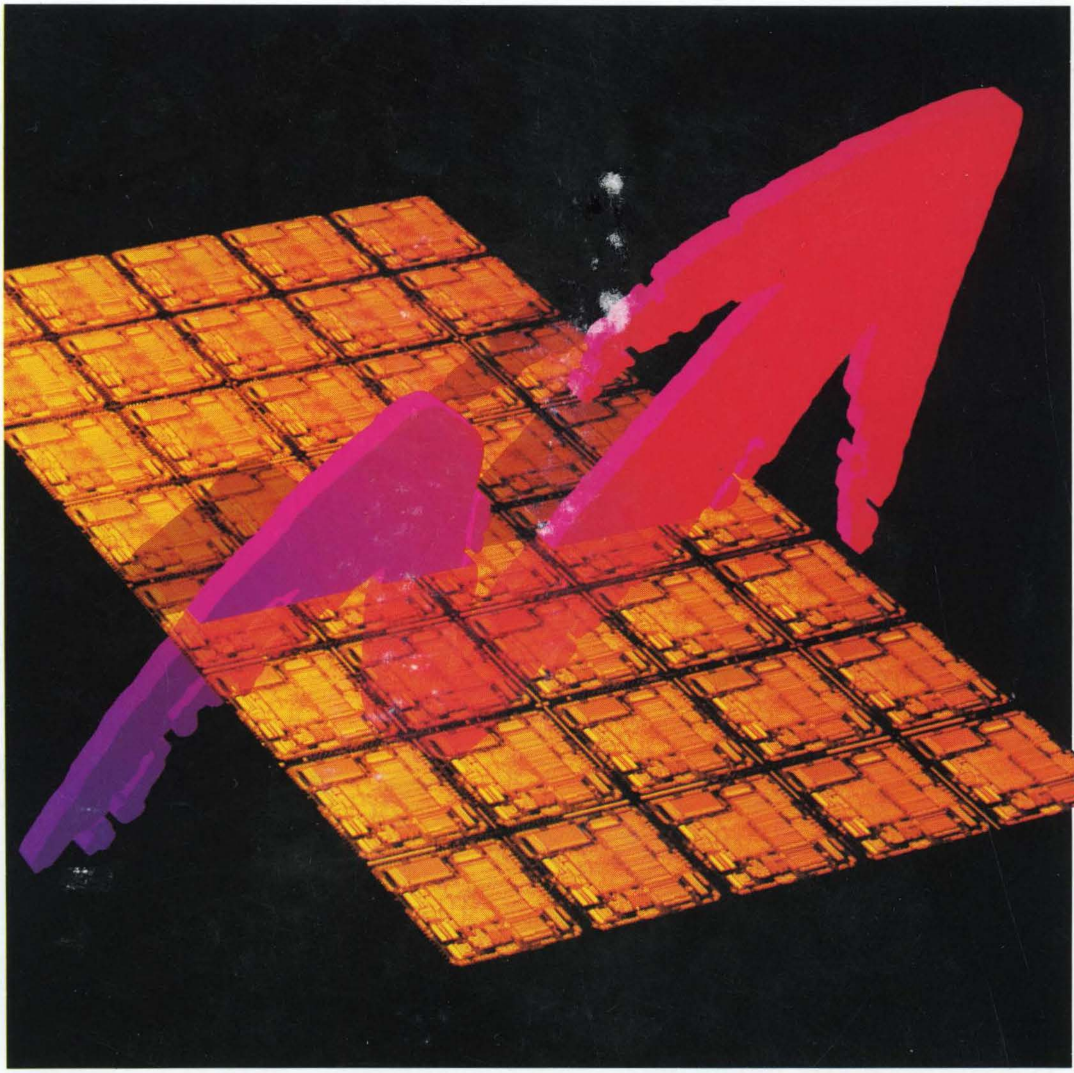
Personal Computer Microprocessors Data Book

Advanced
Micro
Devices

PC WEEK
LABS
TOP PRODUCTS OF 1991
Am386™ Microprocessor



AMD

PC WEEK
LABS
TOP PRODUCTS OF 1991
Am386™ Microprocessor

Personal Computer Microprocessors

1992

Personal Computer Microprocessors

Data Book

ADVANCED MICRO DEVICES

Advanced Micro Devices reserves the right to make changes in its products without notice in order to improve design or performance characteristics.

This publication neither states nor implies any warranty of any kind, including but not limited to implied warrants of merchantability or fitness for a particular application. AMD® assumes no responsibility for the use of any circuitry other than the circuitry in an AMD product.

The information in this publication is believed to be accurate in all respects at the time of publication, but is subject to change without notice. AMD assumes no responsibility for any errors or omissions, and disclaims responsibility for any consequences resulting from the use of the information included herein. Additionally, AMD assumes no responsibility for the functioning of undescribed features or parameters.

Trademarks

AMD, PAL, and PALASM are registered trademarks of Advanced Micro Devices, Inc.

AMD 80C287, AMD 80EC287, Am286, Am386, Am29000, Am29005, Am29050, 29K, Digital Subscriber Controller, ExpressROM, and MACH are trademarks of Advanced Micro Devices, Inc.

FusionPLD is a servicemark of Advanced Micro Devices, Inc.

Postscript is a registered trademark of Adobe Systems

TapePak is a registered trademark of National Semiconductor

Product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Advanced Micro Devices provides high-performance microprocessors for personal computer manufacturers.

This data book details all 80X86 processors and coprocessors produced by AMD, including the 32-bit Am386™ Microprocessor Family, NMOS and CMOS 16-bit 80286 Family, Am286™ ZX/LX integrated microprocessors, 8-bit 8086 and 8088, and CMOS 80287 math coprocessors. Also included in this book is a system integration guide highlighting other AMD PC-related products.

Am386 microprocessors are the fastest available, with 40 MHz on the Am386DXL microprocessor and 25 MHz on the Am386SXL microprocessor. We also offer microprocessors optimized for notebook applications.

Remember, our partnership helps you gain and keep the competitive edge. We are not your competition.

Bob McConnell
Vice President
Personal Computer Products Division

PERSONAL COMPUTER MICROPROCESSORS DATA BOOK TABLE OF CONTENTS

Chapter 1 Microprocessors

Chapter 2 General Information

SYSTEM INTEGRATION

The Am386DX and Am386SX microprocessors are 32-bit microprocessors that form the basis for high-performance 32-bit systems. Both processors incorporate multitasking support, memory management, pipelined architecture, address translation caches, and a high-speed bus interface on a single chip.

The low-power versions of these microprocessors, the Am386DXL CPU and Am386SXL CPU, offer innovations in performance and battery life. These devices have higher clock speeds than the Intel i386DX and Intel i386SX—up to 40 MHz for the Am386DXL CPU and up to 25 MHz for the Am386SXL CPU. Their true static design offers near-zero power consumption in standby mode as well as low-operating power consumption at maximum clock speeds. These power-saving features allow designers to improve the battery life in portable systems as well as facilitate the reduction or elimination of the system cooling fan for cost-savings and size/noise reduction in desktop PCs.

While the Am386 microprocessor represents a significant improvement over previous generations of microprocessors, compatibility with the earlier processors is preserved. Software compatibility at the object-code level is provided, so that an existing investment in 8086 and 80286 software can be maintained. New software can be built upon existing routines, reducing the time-to-market for new products. Hardware compatibility is preserved through the dynamic bus-sizing feature.

The Am386 microprocessor is fully supported by peripherals and performance enhancement components from AMD as well as third party manufacturers. The major components of an Am386 microprocessor system and their functions are described throughout this section.

Am386DX Microprocessor

The Am386 microprocessor is a compatible implementation of the Intel i386DX. It is an advanced 32-bit microprocessor designed for applications needing high performance. The 32-bit registers and data paths support 32-bit addresses and data types. The processor addresses up to 4 Gb of physical memory and 64 tb of virtual memory. The integrated memory management and protection architecture allow high-performance execution of operating systems including DOS, Windows, OS/2, and UNIX.

A pipelined architecture enables the Am386DX microprocessor to perform instruction fetching, decoding, execution, and memory management functions in parallel. Because the Am386DX microprocessor prefetches instructions and queues them internally, instruction fetch and decode times are absorbed in the pipeline; the processor rarely has to wait for an instruction to execute.

Am386DXL Microprocessor

The Am386DXL microprocessor is a high-speed, true static implementation of the Intel i386DX microprocessor. It is ideal for both desktop and battery-powered portable computers. For desktop PCs, the Am386DXL microprocessor offers a 21% performance increase in maximum operating speed from 33 MHz to 40 MHz. Also, this device offers lower heat dissipation allowing system designers to remove or reduce the size and cost of the system cooling fan.

For portable systems, the Am386DXL microprocessor's true static design allows longer battery life by offering low-operating power consumption and a standby mode. At 33 MHz, this device has 40% lower operating Icc than the Intel i386DX. Standby mode allows the Am386 microprocessor to be clocked down to 0 MHz (DC) and retain full register contents. In standby mode, typical current draw is less than 0.02 mA; nearly a 1000× reduction in power consumption over the Intel i386DX or Intel i386SX at minimum frequency.

Additionally, the Am386DXL microprocessor offers a cost- and space-saving 132-pin plastic quad flat pack (PQFP) package. This surface mount package is 40% smaller than a PGA package, allowing smaller board designs without the need for a socket.

Am386DXLV Microprocessor

The Am386DXLV microprocessor is a low-voltage, true static implementation of the Intel i386DX microprocessor. The operating voltage range is 3.0 V to 5.5 V. The low-voltage operation makes it ideal for both desktop and battery-powered portable personal computers. For desktop PCs, low heat dissipation allows the system designers to remove or reduce the size and cost of the system cooling fan. The Am386DXLV microprocessor operates at a maximum speed of 25 MHz from 3.0 to 5.5 V and at a maximum speed of 33 MHz from 4.5 to 5.5 V.

The Am386DXLV microprocessor's lower operating voltage and true static design enable longer battery life and/or lower weight for portable applications. At 25 MHz, this device has 80% lower operating Icc than the Intel i386DX. Lowering typical operating voltage from 5.0 V to 3.3 V doubles the battery life. Standby Mode allows the Am386DXLV microprocessor to be clocked down to 0 MHz (DC) and retain full register contents. In Standby Mode, typical current draw is 0.01 mA, a greater than 1000X reduction in power consumption versus the Intel i386DX or Intel i386SX.

Additionally, the Am386DXLV processor comes with SMM for system and power management. SMI is a non-maskable, higher priority interrupt than NMI and has its own code space (1 Mb). SMI can be coupled with the I/O instruction break feature to implement transparent power management of peripherals. SMM can be used by system designers to implement system and power management code independent of the operating system or the Processor Mode.

Am386SX Microprocessor

The Am386SX microprocessor is a compatible implementation of the Intel i386SX. It is a 32-bit CPU with a 16-bit external data bus, and a 24-bit external address bus. It provides the performance and compatibility benefits of the 386 internal 32-bit architecture with the cost savings associated with 16-bit external hardware.

Am386SXL Microprocessor

The Am386SXL microprocessor is a high-speed, true static implementation of the Intel i386SX. It is ideal for both desktop and battery-powered portable personal computers. For notebooks, the Am386SXL microprocessor's true static design allows longer battery life by offering low operating power consumption and a standby mode. At 20 MHz, this device offers 22% lower current than the Intel i386SX. Standby mode allows the Am386SXL microprocessor to be clocked down to 0 MHz (DC) and retain internal register contents. Typical current in standby mode is reduced to less than 0.02 mA; nearly a 1000x reduction in power consumption over the Intel i386SX at 25 MHz.

For desktop PCs, the Am386SXL microprocessor offers a 25% increase in the maximum operating speed, from 20 to 25 MHz. This device also offers lower heat dissipation, allowing system designers to remove or reduce the size and cost of the system cooling fan.

Am386SXLV Microprocessor

The Am386SXLV microprocessor is a low-voltage, true static implementation of the Intel i386SX microprocessor. With the operating range of 3.0 V to 5.5 V, it is ideal for both desktop and battery-powered notebook personal computers. For desktop PCs, this device offers lower heat dissipation, allowing system designers to remove or reduce the size and cost of the cooling fan.

At 20 MHz, this device has 60% lower operating Icc than the Intel i386SX. Lowering typical operating voltage from 5.0 V to 3.3 V enables battery life to increase by a factor of two. In Standby Mode, typical current draw is less than 0.01 mA, a greater than 1000X reduction in power consumption versus the Intel i386SX, while retaining full register contents.

Additionally, the Am386SXLV microprocessor comes with System Management Mode (SMM) for system and power management. SMI (System Management Interrupt) is a non-maskable, higher priority interrupt than NMI and has its own code space (1 Mb). SMI can be coupled with the I/O instruction break feature to implement transparent power managment of peripherals. SMM can be used by system designers to implement system and power management code independent of the operating system or the Processor Mode.

Math Coprocessors

The performance of many numeric-intensive applications can be enhanced by the use of math coprocessors. A math coprocessor provides the hardware to perform floating point functions that would otherwise be performed by software. Coprocessors extend the instruction set of the microprocessor to include floating point operations.

The Am386DX and Am386SX microprocessors allow an interface to a 387-compatible math coprocessor. For applications that benefit from high-precision integer and floating point calculations (such as CAD, spreadsheet, graphics, data base, font generation, and statistics), the math coprocessor provides full support of the IEEE standard for floating point operations.

Cache Controllers

Low-speed memory systems (<= 20 MHz) have traditionally been implemented with dynamic RAMs (DRAMs) using a direct or page-interleaved approach. Page-interleaving is a memory design technique for achieving near-zero wait state system performance with inexpensive DRAMs. However, as the speed of processors increases these methods may become inadequate. With increased processor speed, the speed of the DRAM memory increases as well as the cost. In order to reduce cost and improve performance, cache controllers are used. A cache-based memory system contains a small amount of fast SRAM memory and a large amount of slower DRAM memory. This provides the performance of SRAMs at a cost approaching that of DRAMs.

Many cache controllers are available from third party manufactures for both the Am386DX and Am386SX microprocessors.

Integrated Peripheral Chip Sets

Many discrete features such as memory, I/O, and bus control for Am386 processors have become integrated into chip sets. These chip sets have become the standard for cost-effective designs for desktop and portable personal computers. A wide variety of chip sets are available from third party manufacturers to support both the Am386DX and Am386SX microprocessors in high-speed desktop and long battery life portable designs.

Programmable Logic

With over a decade of PLD leadership, AMD manufactures a broad line of programmable logic to improve time-to-market when designing systems around Am386 microprocessors. The MACH™ (Macro Array CMOS High-Density) Family fills the system design gap between high-density field programmable gate arrays and high-speed PAL® device ICs.

The MACH Family offers a breakthrough combination of high density (900 to 3600 gates) and high performance (15 ns and 50 MHz). AMD offers proven production methods and the right technology for attractive pricing on high-density programmable logic. MACH Family devices are manufactured with EE CMOS technology offering the following benefits.

–Reprogrammable in low-cost plastic packages

–Quick design revisions

–Insurance on production programmed parts

–Quick market entry

The MACH Family is supported by major third party PLD design software packages through AMD's FusionPLD℠ program. It is also supported by AMD's own PALASM® software. Several of these software packages run on 286- and 386-based systems and provide low-cost computer aided design (CAD) capability while easing the important phases of the designer's task: design entry, implementation, verification, programming, and testing. The MACH Family devices can be programmed on conventional PAL device programmers with appropriate personality and socket adapter modules.

AMD also offers several zero-power PAL devices for power-conscious designs. These devices' ultra-low standby power is ideal for laptop, notebook, and hand-held PC designs where faster time-to-market and longer battery life are required.

EPROMs and Flash

AMD is a leading supplier of EPROMs (Erasable Programmable Read Only Memory) to the personal computer market. EPROMs are traditionally used to store BIOS in one or two chips on the motherboard. AMD offers higher density, lower power, and faster EPROMs than any other manufacturer in the world. AMD's 1 Mb EPROM, for example, runs as fast as 45 ns.

This 1 Mb high-speed EPROM could be used with the 40-MHz Am386DXL microprocessor to execute the BIOS and keyboard code directly from EPROM with no wait states and no shadowing of the EPROM. EPROM shadowing traditionally allowed designers to run slow EPROM code from faster DRAM. However, with AMD's high-performance EPROMs, the DRAM is freed up for use by operating systems such as DOS, OS/2, and UNIX.

AMD's EPROMs are offered in UV-erasable windowed ceramic versions as well as in lower cost plastic "one-time programmable" DIP and PLCC packages. For medium- to high-volume stable code, AMD offers the ExpressROM™ memory, a factory-programmed plastic packaged EPROM.

AMD also offers the highest densities and speeds in Flash memories. Flash memories may be reprogrammed in systems and allow the user to reconfigure the BIOS without system disassembly. Hand-held and application specific personal computers can also use Flash memories as a read-write solid state hard disk. A silicon disk is ideal for battery operated portable PCs since they are fast, rugged, economical, and drain little current from the system battery.

Networking Solutions

AMD provides a wide range of networking solutions to be designed into PC add-in cards or integrated onto the motherboard. The AMD Am7900 Ethernet/Cheapernet chip set provides a low component count, minimum board space, low-cost Ethernet interface. AMD also offers solutions for 10BASE-T and FDDI standards.

In addition, AMD manufactures the Am79C30 Digital Subscriber Controller™ circuit. This device may be used for voice annotation through electronic mail systems across networks. This feature allows users to electronically attach voice dialog to Email messages.

Serial Controllers/SCSI

Some of the newest portable personal computers offer pen-based operating systems such as GO Computer's PenPoint and Microsoft's PenWindows. These systems are designed to recognize a light pen or other stylus as the input device. One of the features offered with these new notebook systems is handwriting recognition—the user prints the characters with the pen and the system converts it to ASCII characters. The AMD Am85C80 is being used to perform handwriting character recognition as well as provide an integrated SCSI interface. The Am85C80 integrates an 85C30 dual serial communication controller as well as a 5380 SCSI controller. The sleep-mode feature of the Am85C80 makes it particularly well suited for notebook applications.

SCSI adapters offer tremendous flexibility as a standard for personal computer peripherals such as hard disks, CD ROMs, optical drives, laser printers, and scanners, many of which are very useful for the emerging multimedia market.

Keyboard Controllers

AMD's 80C51-based 8-bit microcontrollers offer a cost-effective solution for both desktop and portable personal computers. These devices may be used for keyboard scanning as well as power management. The 87C51 EPROM-based microcontroller can replace separate 8042/80C42 and 8048/8049 controllers and is supported by the major BIOS manufacturers.

Bus Interface Products

The Am29C800A High-Performance CMOS Bus Interface Family is ideal for Am386 CPU based bus interface applications. Typically, these devices are used with integrated peripheral chip sets to drive extra I/O bus slots for peripheral add-in cards and large DRAM arrays. Their low ground-bounce outputs are particularly well suited for systems with clock speeds greater than 25 MHz.

The Am29C800A family provides bipolar-comparable performance while consuming much less power than bipolar devices. Pin-for-pin compatible with the Am29800A/Am29C800 families, these devices offer lower propagation delays and consume less power than their predecessors. They offer the necessary output drive for driving heavily loaded buses while achieving the fast switching speeds required for high-performance systems.

Graphics

AMD's 29K™ Family of general-purpose 32-bit RISC microprocessors is meeting the needs of today's distributed graphics display systems. The Am29000™ microprocessor can be found in graphics applications ranging from Apple Macintosh QuickDraw accelerators to X Window System terminals (X terminals). In addition, the 29K Family is the heart of a wide range of PostScript-based laser printer products.

The 29K Family represents the broadest range of pin- and software-compatible 32-bit RISC microprocessors on the market today. The low-end Am29005 processor provides 6–9 MIPS performance for cost-sensitive applications. The Am29000 processor provides 12–23 MIPS performance for mainstream 2D graphics. The Am29050 microprocessor, with its tightly integrated on-chip floating point unit, provides up to 32 MIPS and 34 MFLOPS sustained performance (80 MFLOPS peak) for demanding 3D graphics and imaging applications. AMD's Embedded Processor Division regularly introduces new members of the 29K Family to support the embedded graphics, laser printer, and network controller markets.

The IBM PC-compatible market has entered the age of Graphical User Interfaces (GUIs) with the introduction of Microsoft Windows 3.0, joining Apple and the UNIX world. The 29K Family provides a flexible architecture for providing high-speed acceleration for the Application Programming Interfaces (APIs) associated with each GUI.

The power of the 29K Family of microprocessors ensures that the entire graphics library of well-distributed GUIs can be offloaded to a graphics accelerator. This not only provides graphics horsepower, but also frees the central processor to devote more time to executing the PC user's applications—increasing the speed of the entire system. The Am386 Family and the 29K Family can work together to provide high-performance, cost-effective solutions for the latest GUIs, including Windows and the emerging Display PostScript standard.

For more information on any of AMD's products, contact your AMD sales representative.

Chapter 1

Microprocessors

CHAPTER 1
Microprocessors

8086

16-Bit Microprocessor
iAPX86 Family
FINAL

DISTINCTIVE CHARACTERISTICS

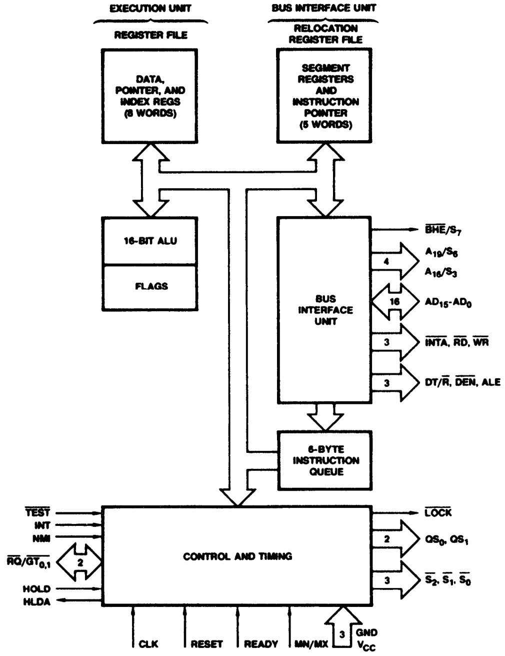
- Directly addresses up to 1 Mbyte of memory
- 24 operand addressing modes
- Efficient implementation of high level languages
- Instruction set compatible with 8080 software
- Bit, byte, word, and block operations
- 8 and 16-bit signed and unsigned arithmetic in binary or decimal
- MULTIBUS® system interface
- Three speed options
  - 5MHz for 8086
  - 8MHz for 8086-2
  - 10MHz for 8086-1

GENERAL DESCRIPTION

The 8086 is a general purpose 16-bit microprocessor CPU. Its architecture is built around thirteen 16-bit registers and nine 1-bit flags. The CPU operates on 16-bit address spaces and can directly address up to 1 megabyte using offset addresses within four distinct memory segments, designated as code, data, stack and extra code. The 8086 implements a powerful instruction set with 24 operand addressing modes. This instruction set is compatible with that of the 8080 and 8085. In addition, the 8086 is particularly effective in executing high level languages.

The 8086 can operate in minimum and maximum modes. Maximum mode offloads certain bus control functions to a peripheral device and allows the CPU to operate efficiently in a multi-processor system. The CPU and its high performance peripherals are MULTIBUS compatible. The 8086 is implemented in N-channel, depletion load, silicon gate technology and is contained in a 40-pin CERDIP package, Molded DIP package, or Plastic Leaded Chip Carrier.

BLOCK DIAGRAM

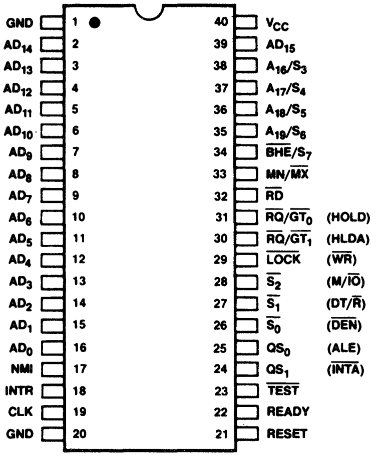


BD003740

Publication # Rev. Amendment
01966 D /0
Issue Date: August 1989

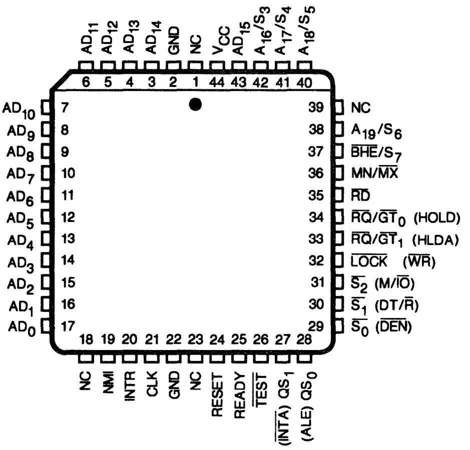
CONNECTION DIAGRAMS
Top View

DIP



CD005511

PLCC



CD010701

Note: Pin 1 is marked for orientation.

ORDERING INFORMATION

Commercial Products

AMD commercial products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of: **a. Temperature Range**
b. Package Type
c. Device Number
d. Speed Option
e. Optional Processing

I D 8086 -2 B

e. OPTIONAL PROCESSING
Blank = Standard Processing
B = Burn-in

d. SPEED OPTION
Blank = 5 MHz
-2 = 8 MHz
-1 = 10 MHz

c. DEVICE NUMBER/DESCRIPTION
8086
16-Bit Microprocessor

b. PACKAGE TYPE
P = 40-Pin Plastic DIP (PD 040)
D = 40-Pin Ceramic DIP (CD 040)
N = 44-Pin Plastic Leaded Chip Carrier (PL 044)

a. TEMPERATURE RANGE
Blank = Commercial (0 to +70°C)
I = Industrial (-40 to +85°C)

Valid Combinations	
P, D, N	8086
	8086-2
	8086-1
D, ID	8086-2B
D	8086-1
ID	8086B

Valid Combinations

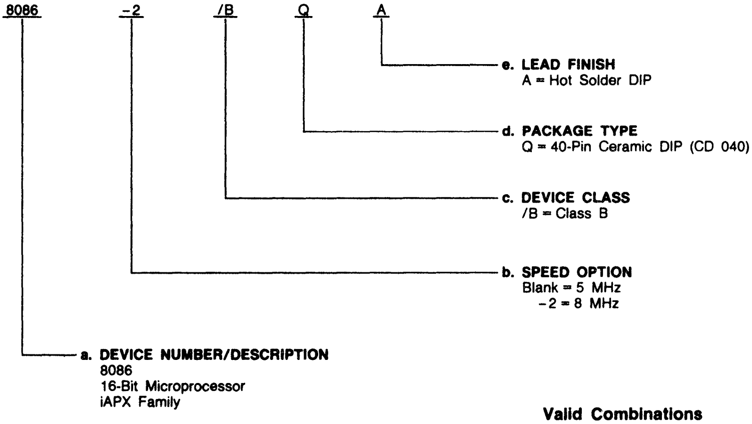
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

ORDERING INFORMATION

Military Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) for APL products is formed by a combination of: **a. Device Number**

b. Speed Option (if applicable)
c. Device Class
d. Package Type
e. Lead Finish



e. LEAD FINISH
A = Hot Solder DIP

d. PACKAGE TYPE
Q = 40-Pin Ceramic DIP (CD 040)

c. DEVICE CLASS
/B = Class B

b. SPEED OPTION
Blank = 5 MHz
-2 = 8 MHz

a. DEVICE NUMBER/DESCRIPTION
8086
16-Bit Microprocessor
iAPX Family

Valid Combinations	
8086	/BQA
8086-2	

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check for newly released valid combinations.

Group A Tests

Group A tests consist of Subgroups 1, 2, 3, 7, 8, 9, 10, 11.

PIN DESCRIPTION

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

<table>
<tr><th>Pin No.*</th><th>Name</th><th>I/O</th><th>Description</th></tr>
<tr><td>39, 2–16</td><td>AD_{15}–AD_0</td><td>I/O</td><td>Address Data Bus. These lines constitute the time multiplexed memory/IO address (T_1) and data (T_2, T_3, T_W, T_4) bus. A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7–D_0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A_0 to condition chip select functions. (See $\overline{BHE}$.) These lines are active HIGH and float to three-state OFF during interrupt acknowledge and local bus "hold acknowledge."</td></tr>
<tr><td>35–38</td><td>A_{19}/S_6,
A_{18}/S_5,
A_{17}/S_4,
A_{16}/S_3</td><td>O</td><td>Address/Status. During T_1 these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W, and T_4. The status of the interrupt enable FLAG bit (S_5) is updated at the beginning of each CLK cycle. A_{17}/S_4 and A_{16}/S_3 are encoded as shown.
This information indicates which relocation register is presently being used for data accessing. These lines float to three-state OFF during local bus "hold acknowledge."
<table>
<tr><th>A_{17}/S_4</th><th>A_{16}-S_3</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>Alternate Data</td></tr>
<tr><td>0</td><td>1</td><td>Stack</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>Code or None</td></tr>
<tr><td>1</td><td>1</td><td>Data</td></tr>
<tr><td>S_6 is 0 (LOW)</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>34</td><td>$\overline{BHE}/S_7$</td><td>O</td><td>Bus High Enable/Status. During T_1 the bus high enable signal ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D_{15}–D_8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is LOW during T_1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S_7 status information is available during T_2, T_3, and T_4. The signal is active LOW and floats to three-state OFF in "hold." It is LOW during T_1 for the first interrupt acknowledge cycle.
<table>
<tr><th>$\overline{BHE}$</th><th>A_0</th><th>Characteristics</th></tr>
<tr><td>0</td><td>0</td><td>Whole word</td></tr>
<tr><td>0</td><td>1</td><td>Upper byte from/ to odd address</td></tr>
<tr><td>1</td><td>0</td><td>Lower byte from/ to even address</td></tr>
<tr><td>1</td><td>1</td><td>None</td></tr>
</table></td></tr>
<tr><td>32</td><td>$\overline{RD}$</td><td>O</td><td>Read. Read strobe indicates that the processor is performing a memory of I/O read cycle, depending on the state of the S_2 pin. This signal is used to read devices which reside on the 8086 local bus. $\overline{RD}$ is active LOW during T_2, T_3, and T_W of any read cycle and is guaranteed to remain HIGH in T_2 until the 8086 local bus has floated.
This signal floats to three-state OFF in "hold acknowledge."</td></tr>
<tr><td>22</td><td>READY</td><td>I</td><td>READY. Is the acknowledgment from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory/IO is synchronized by the 8284A Clock Generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the set-up and hold times are not met.</td></tr>
<tr><td>18</td><td>INTR</td><td>I</td><td>Interrupt Request. Is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.</td></tr>
<tr><td>23</td><td>$\overline{TEST}$</td><td>I</td><td>$\overline{TEST}$. Input is examined by the "Wait" instruction. If the $\overline{TEST}$ input is LOW, execution continues; otherwise, the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.</td></tr>
<tr><td>17</td><td>NMI</td><td>I</td><td>Non-Maskable Interrupt. An edge-triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.</td></tr>
<tr><td>21</td><td>RESET</td><td>I</td><td>Reset. Causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the Instruction Set description, when RESET returns LOW. RESET is internally synchronized.</td></tr>
<tr><td>19</td><td>CLK</td><td>I</td><td>Clock. Provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.</td></tr>
<tr><td>40</td><td>V_{CC}</td><td></td><td>V_{CC}. The + 5 V power supply pin.</td></tr>
<tr><td>1, 20</td><td>GND</td><td></td><td>Ground. The ground pin.</td></tr>
<tr><td>33</td><td>MN/$\overline{MX}$</td><td>I</td><td>Minimum/Maximum. Indicates what mode the processor is to operate in. The two modes are discussed in the following sections.</td></tr>
</table>

*Pin numbers correspond to DIPs only.

PIN DESCRIPTION (continued)

<table>
<tr><th>Pin No.*</th><th>Name</th><th>I/O</th><th>Description</th></tr>
<tr><td>28–26</td><td>$\overline{S}_2$, $\overline{S}_1$, $\overline{S}_0$</td><td>O</td><td>Status. Active during T_4, T_1, and T_2 and is returned to the passive state (1, 1, 1) during T_3 or during T_W when READY is HIGH. This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals. Any change by S_2, S_1, or S_0 during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle. These signals float to three-state OFF in "hold acknowledge." These status lines are encoded as shown.
<table>
<tr><th>$\overline{S}_2$</th><th>$\overline{S}_1$</th><th>$\overline{S}_0$</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table>
</td></tr>
<tr><td>31, 30</td><td>$\overline{RQ}/\overline{GT}_0$,
$\overline{RQ}/\overline{GT}_1$</td><td>I/O</td><td>Request/Grant. Pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT}_0$ having higher priority than $\overline{RQ}/\overline{GT}_1$. $\overline{RQ}/\overline{GT}$ has an internal pull-up resistor so it may be left unconnected. The request/grant sequence is as follows:
1. A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 8086 (pulse 1).
2. During a T_4 or T_1 clock cycle, a pulse 1 CLK wide from the 8086 to the requesting master (pulse 2), indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge."
3. A pulse 1 CLK wide from the requesting master indicates to the 8086 (pulse 3) that the "hold" request is about to end and that the 8086 can reclaim the local bus at the next CLK.
Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.
If the request is made while the CPU is performing a memory cycle, it will release the local bus during T_4 of the cycle when all the following conditions are met:
1. Request occurs on or before T_2.
2. Current cycle is not the low byte of a word (on an odd address).
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.
If the local bus is idle when the request is made, two possible events will follow:
1. Local bus will be released during the next clock.
2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.</td></tr>
<tr><td>29</td><td>$\overline{LOCK}$</td><td>O</td><td>$\overline{LOCK}$. Output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to three-state OFF in "hold acknowledge."</td></tr>
<tr><td>24, 25</td><td>QS_1, QS_0</td><td>O</td><td>Queue Status. The queue status is valid during the CLK cycle after which the queue operation is performed.
QS_1 and QS_0 provide status to allow external tracking of the internal 8086 instruction queue.</td></tr>
<tr><td>28</td><td>M/$\overline{IO}$</td><td>O</td><td>Status line. Logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/IO becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle (M = HIGH, IO = LOW). M/$\overline{IO}$ floats to three-state OFF in local bus "hold acknowledge."</td></tr>
<tr><td>29</td><td>$\overline{WR}$</td><td>O</td><td>Write. Indicates that the processor is performing a write memory or write I/O cycle, depending on the state of M/IO signal. WR is active for T_2, T_3, and T_W of any write cycle. It is active LOW and floats to three-state OFF in local bus "hold acknowledge."</td></tr>
<tr><td>24</td><td>$\overline{INTA}$</td><td>O</td><td>$\overline{INTA}$. Is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3, and T_W of each interrupt acknowlegde cycle.</td></tr>
<tr><td>25</td><td>ALE</td><td>O</td><td>Address Latch Enable. Provided by the processor to latch the address into 8282/8283 address latch. It is a HIGH pulse active during T_1 of any bus cycle. Note that ALE is never floated.</td></tr>
<tr><td>27</td><td>DT/$\overline{R}$</td><td>O</td><td>Data Transmit/Receive. Needed in minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{R}$ is equivalent to $\overline{S}_1$ in the maximum mode, and its timing is the same as for M/$\overline{IO}$. (T = HIGH, R = LOW.) This signal floats to three-state OFF in local bus "hold acknowledge."</td></tr>
<tr><td>26</td><td>$\overline{DEN}$</td><td>O</td><td>Data Enable. Provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access and for INTA cycles. For a read or $\overline{INTA}$ cycle, it is active from the middle of T_2 until the middle of T_4, while for a write cycle, it is active from the beginning of T_2 until the middle of T_4. $\overline{DEN}$ floats to three-state OFF in local bus "hold acknowledg</td></tr>
</table>

*Pin numbers correspond to DIPs only.

PIN DESCRIPTION (continued)

Pin No.*	Name	I/O	Description
31, 30	HOLD, HLDA	I/O	HOLD. Indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T_4 or T_1 clock cycle. Simultaneous with the issuance of HLDA, the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will LOWer HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. The same rules as for $\overline{RQ}/\overline{GT}$ apply, regarding when the local bus will be released. HOLD is not asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the set-up time.

*Pin numbers correspond to DIPs only.

DETAILED DESCRIPTION

The 8086 CPU is internally organized into two processing units. These two units are the Bus Interface Unit (BIU) and the Execution Unit (EU). A block diagram of this organization is shown on page 1.

The BIU performs instruction fetch and queuing, operand fetch and store, address relocation, and basic bus control. The EU receives operands and instructions from the BIU and processes them on a 16-bit ALU. The EU accesses memory and peripheral devices through requests to the BIU. The BIU generates physical addresses in memory using the 4 segment registers and offset values.

The BIU and EU usually operate asynchronously. This permits the 8086 to overlap execution fetch and execution. Up to 6 instruction bytes can be queued. The instruction queue acts as a FIFO buffer for instructions, from which the EU extracts instruction bytes as required.

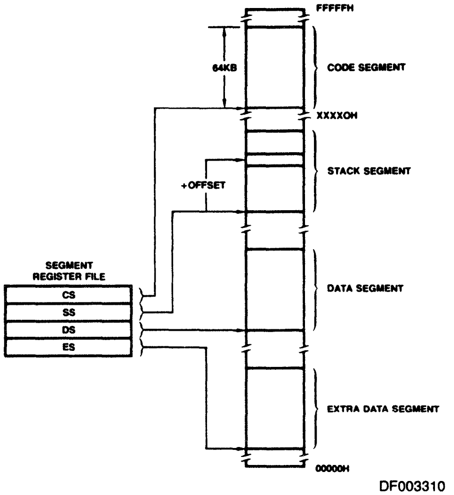
Memory Organization

The 8086 addresses up to 1 megabyte of memory. The address space is organized as a linear array, from 00000 to FFFFF in hexadecimal. Memory is subdivided into segments of 64K bytes each. There are 4 segments: code, stack, data, and extra (usually employed as an extra data segment). Each segment thus contains information of a similar type. Selection of a destination segment is automatically performed using the rules in the table below. This segmentation makes memory more easily relocatable and supports a more structured programming style.

Physical addresses in memory are generated by selecting the appropriate segment, obtaining the segment "base" address from the segment register, shifting the base address 4 digits to the left, and then adding this base to the "offset" address. For programming code, the offset address is obtained from the instruction pointer. For operands, the offset address is calculated in several ways, depending upon information contained in the addressing mode. Memory organization and address generation are shown in Figure 1a.

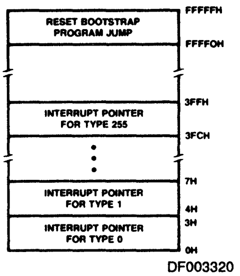
Certain memory locations are reserved for specific CPU operations. These are shown in Figure 1b. Addresses FFFF0H through FFFFFH are reserved for operations which include a jump to the initial program loading routine. After RESET, the CPU will always begin execution at location FFFF0H, where the jump must be located.

Addresses 00000H through 003FFH are reserved for interrupt operations. The service routine of each of the 256 possible interrupt types is signaled by a 4-byte pointer. The pointer elements must be stored in reserved memory addresses before the interrupts are invoked.



DF003310

Figure 1a. Memory Organization


RESET BOOTSTRAP PROGRAM JUMP
FFFFFH
FFFF0H
3FFH
INTERRUPT POINTER FOR TYPE 255
3FCH
7H
INTERRUPT POINTER FOR TYPE 1
4H
3H
INTERRUPT POINTER FOR TYPE 0
0H

DF003320

Figure 1b. Reserved Memory Locations

Memory Reference Need	Segment Register Used	Segment Selection Rule
Instructions	CODE (CS)	Automatic for all prefetching of instructions.
Stack	STACK (SS)	All stack pushes and pops, and all memory references relative to BP base register except data references.
Local Data	DATA (DS)	Data references which are relative to the stack, the destination of a string operation, or explicitly overriden.
External (Global) Data	EXTRA (ES)	Destination of string operations, when they are explicitly selected using a segment override.

Minimum and Maximum Modes

The 8086 has two system configurations, minimum and maximum mode. The CPU has a strap pin, MN/$\overline{MX}$, which defines the system configuration. The status of this strap pin defines the function of pin numbers 24 through 31.

When MN/$\overline{MX}$ is strapped to GND, the 8086 operates in maximum mode. The operations of pins 24 through 31 are redefined. In maximum mode, several bus timing and control functions are "off-loaded" to the 8288 bus controller, thus freeing up the CPU. The CPU communicates status information to the 8288 through pins S_0, S_1, and S_2. In maximum mode, the 8086 can operate in a multiprocessor system, using the LOCK signal within a Multibus format.

When MN/$\overline{MX}$ is strapped to V_{CC}, the 8086 operates in minimum mode. The CPU sends bus control signals itself through pins 24 through 31. This is shown in the Connection Diagrams (in parentheses). Examples of minimum and maximum mode systems are shown in Figure 2.

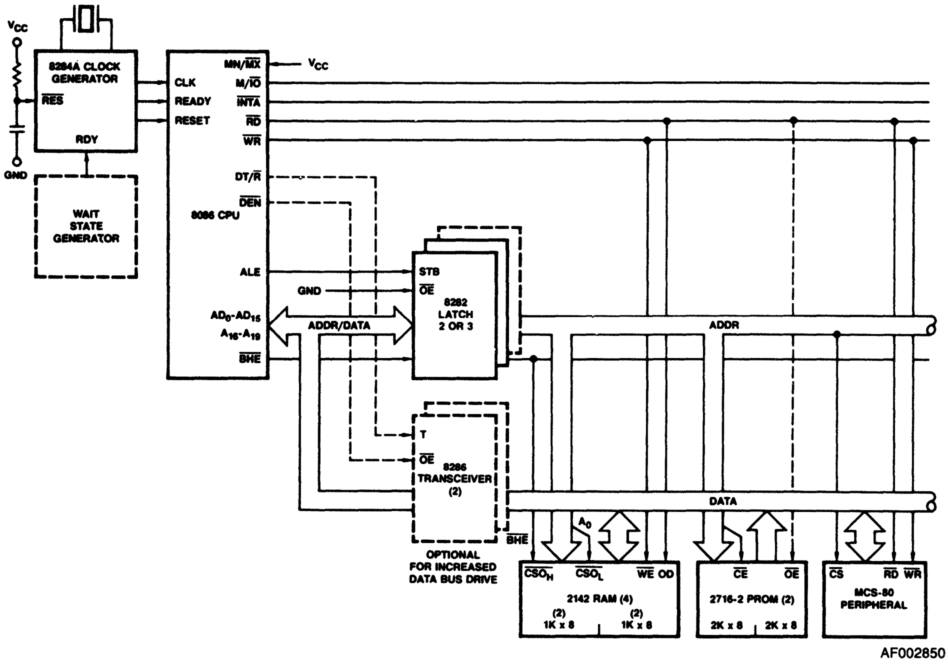


Figure 2a. Minimum Mode 8086 Typical Configuration

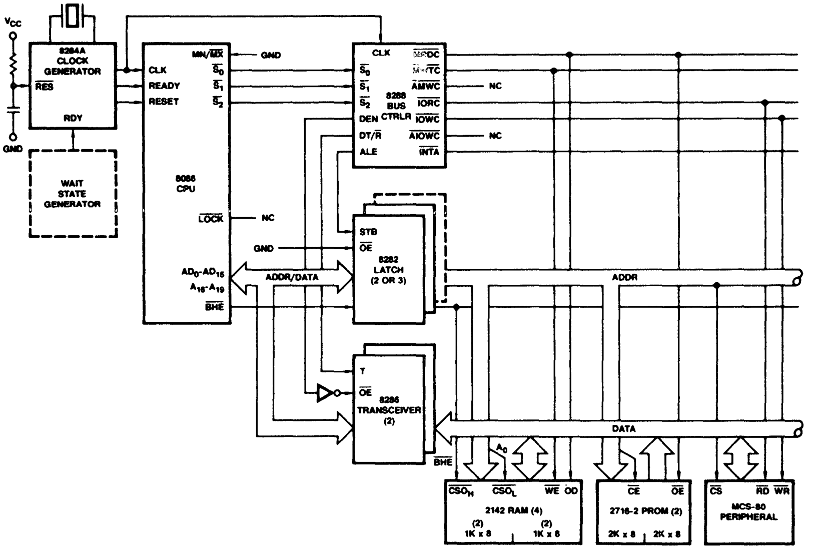


Figure 2b. Maximum Mode 8086 Typical Configuration

Bus Operation

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

Each bus cycle consists of at least four CLK cycles. These are referred to as T_1, T_2, T_3 and T_4 (see Figure 5). The address is sent from the processor during T_1. Data transfer occurs on the bus during T_3 and T_4. T_2 is used for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "Wait" states (T_W) are inserted between T_3 and T_4. Each inserted "Wait" state is of the same duration as a CLK cycle. "Idle" states (T_1) or inactive CLK cycles can occur between 8086 bus cycles. The processor uses these cycles for internal housekeeping.

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

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

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

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

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

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

I/O Addressing

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

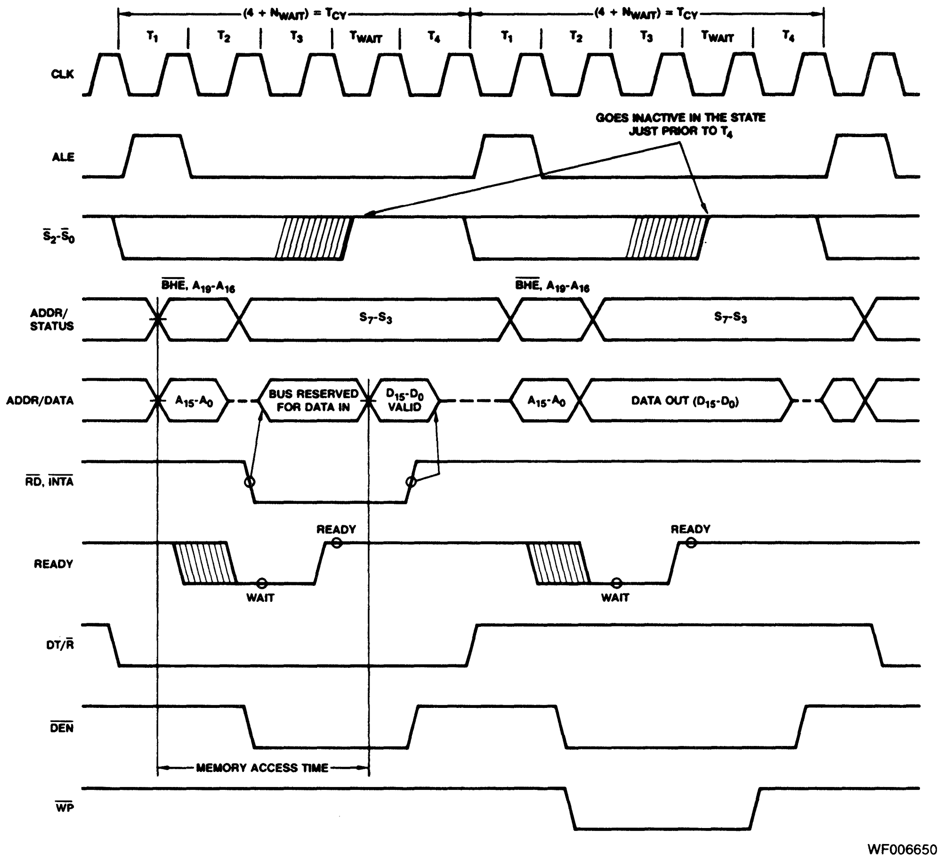


Figure 3. Basic System Timing

EXTERNAL INTERFACE

Processor Reset and Initialization

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8086 RESET is required to be HIGH for greater than 4 CLK cycles. The 8086 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 10 CLK cycles. After this interval the 8086 operates normally beginning with the instruction in absolute location FFFF0H (see Figure 1b). The details of this operation are explained in the Instruction Set description of the MCS-86 Family User's Manual. The RESET input is internally synchronized to the processor clock. At initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50μs after power-up, to allow complete initialization of the 8086.

NMI may not be asserted prior to the 2nd CLK cycle following the end of RESET.

Interrupt Operations

Interrupt operations fall into two classes: software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are described in the Instruction Set description. Hardware interrupts are either non-maskable or maskable.

Interrupts transfer control to a new program location. A 256-element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 1b), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type." An interrupting device supplies an 8-bit type number during the interrupt acknowledge sequence, which is used to "vector" through the appropriate element to the new interrupt service program location.

Non-Maskable Interrupt (NMI)

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

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

Maskable Interrupt (INTR)

The 86/10 provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable FLAG status bit. The interrupt request signal is level-triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block-type instruction. During the interrupt response sequence, further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt, or single-step), although the FLAGS register, which is automatically pushed onto the stack, reflects the state of the processor prior to the Interrupt. Until the old FLAGS register is restored, the enable bit will be zero unless specifically set by an instruction.

During the response sequence (Figure 4), the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 8086 emits the LOCK signal from T_2 of the first bus cycle until T_2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle, a byte is fetched from the external interupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop, which returns the status of the original interrupt enable bit when it restores the FLAGS.

HALT

When a software "HALT" instruction is executed, the processor indicates that it is entering the "HALT" state in one of two ways depending upon which mode is strapped. In minimum mode, the processor issues one ALE with no qualfying bus control signals. In Maximum Mode, the processor issues appropiate HALT status on $\overline{S_2}\overline{S_1}\overline{S_0}$, and the 8288 bus controller issues one ALE. The 8086 will not leave the "HALT" state when a local bus "hold" is entered while in "HALT." In this case, the processor reissues the HALT indicator. An interrupt request or RESET will force the 8086 out of the "HALT" state.

Read/Modify/Write (Semaphore) Operation Via Lock

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

External Synchronization Via Test

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

Basic System Timing

Typical system configurations for the processor operating in minimum mode and in maximum mode are shown in Figures 2a and 2b, respectively. In minimum mode, the processor emits bus control signals in a manner similar to the 8085. In maximum mode, the processor emits coded status information which the 8288 bus controller uses to generate MULTIBUS compatible bus control signals. Figure 3 illustrates the signal timing relationships.

System Timing – Minimum System

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (low-going) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into the 8282/8283 latch. The $\overline{\text{BHE}}$ and A_0 signals address the low, high, or both bytes. From T_1 to T_4, the M/$\overline{\text{IO}}$ signal indicates a memory or I/O operation. At T_2 the address is removed from the local bus and the bus goes to a high impedance state. The read control signal is also asserted at T_2. The read ($\overline{\text{RD}}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver (8286/8287) is required to buffer the 8086 local bus, signals DT/$\overline{\text{R}}$ and $\overline{\text{DEN}}$ are provided by the 8086.

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

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

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

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

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal ($\overline{INTA}$) is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 6.) In the second of two successive INTA cycles, a byte of information is read from bus lines D_7–D_0 as supplied by the interrupt system logic (i.e., 8259A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into a interrupt vector lookup table, as described earlier.

Bus Timing — Medium Size Systems

For medium size systems, the MN/$\overline{MX}$ pin is connected to V_{SS}, and the 8288 Bus Controller is added to the system as well as an 8282/8283 latch for latching the system address and a 8286/8287 transceiver to allow for bus loading greater than the 8086 is capable of handling. Signals ALE, DEN, and DT/$\overline{R}$ are generated by the 8288 instead of the processor in this configuration, although their timing remains relatively the same. The 8086 status ($\overline{S}_2$, $\overline{S}_1$, and $\overline{S}_0$) provide type-of-cycle information and become 8288 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence, data isn't valid at the leading edge of write. The 8286/8287 transceiver receives the usual T and OE inputs from the 8288's DT/$\overline{R}$ and DEN.

The pointer into the interrupt vector table, which is passed during the second INTA cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8259A Priority Interrupt Controller is positioned on the local bus, a TTL gate is required to disable the 8286/8287 transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll."

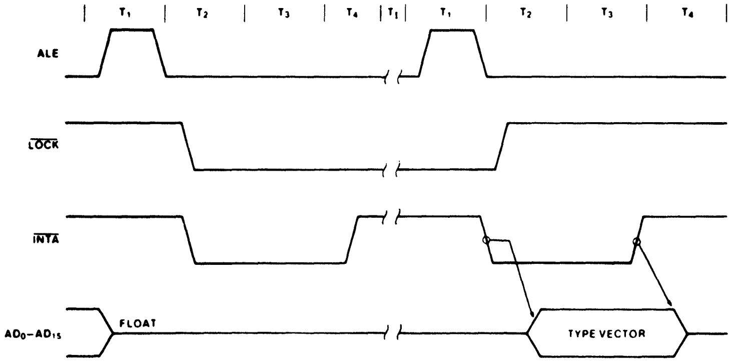


Figure 4. Interrupt Acknowledge Sequence

WF009371

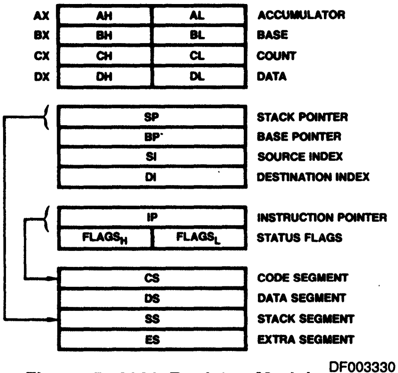


Figure 5. 8086 Register Model

DF003330

ABSOLUTE MAXIMUM RATINGS

Storage Temperature .... −65 to +150°C
Ambient Temperature Under Bias .... 0 to 70°C
Voltage on any Pin
  with Respect to Ground .... −1 to +7.0 V
Power Dissipation .... 2.5 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices
  Temperature (T_A) .... 0 to +70°C
  Supply Voltage (V_{CC})
    8086 .... 5 V ± 10%
    8086-1, 8086-2 .... 5 V ± 5%

Industrial (I) Devices
  Temperature (T_A) .... −40 to +85°C
  Supply Voltage (V_{CC})
    8086 .... 5 V ± 10%
    8086-1, 8086-2 .... 5 V ± 5%

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating range

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

SWITCHING CHARACTERISTICS over COMMERCIAL operating range
MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

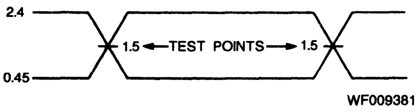
Parameters	Description	Test Conditions	8086		8086-2		8086-1		Units
			Min	Max	Min	Max	Min	Max	
TCLCL	CLK Cycle Period		200	500	125	500	100	500	ns
TCLCH	CLK Low Time		118		68		53		ns
TCHCL	CLK High Time		69		44		39		ns
TCH1CH2	CLK Rise Time	From 1.0 to 3.5V		10		10		10	ns
TCL2CL1	CLK Fall Time	From 3.5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

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

SWITCHING CHARACTERISTICS over COMMERCIAL operating range (continued)
TIMING RESPONSES

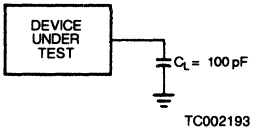
Parameters	Description	Test Conditions	8086		8086-2		8086-1		Units
			Min	Max	Min	Max	Min	Max	
TCLAV	Address Valid Delay	$^*C_L$ = 20-100 pF for all 8086 Outputs (in addition to 8086 self-load). Typical C_L = 100 pF.	10	110	10	60	10	50	ns
TCLAX	Address Hold Time		10		10		10		ns
TCLAZ	Address Float Delay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

SWITCHING TEST INPUT/OUTPUT WAVEFORM



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

SWITCHING TEST LOAD CIRCUIT



C_L includes jig capacitance

SWITCHING CHARACTERISTICS over **COMMERCIAL** operating range (continued)
MAX MODE SYSTEM (USING 8288 BUS CONTROLLER)
TIMING REQUIREMENTS

Parameters	Description	Test Conditions	8086		8086-2		8086-1		Units
			Min	Max	Min	Max	Min	Max	
TCLCL	CLK Cycle Period		200	500	125	500	100	500	ns
TCLCH	CLK Low Time		118		68		53		ns
TCHCL	CLK High Time		69		44		39		ns
TCH1CH2	CLK Rise Time	From 1.0 to 3.5 V		10		10		10	ns
TCL2CL1	CLK Fall Time	From 3.5 to 1.0 V		10		10		10	ns
TDVCL	Data in Set-up Time		30		20		5		ns
TCLDX	Data in Hold Time		10		10		10		ns
TR1VCL	RDY Set-up Time into 8284A (See Notes 1, 2)		35		35		35		ns
TCLR1X	RDY Hold Time into 8284A (See Notes 1, 2)		0		0		0		ns
TRYHCH	READY Set-up Time into 8086		118		68		53		ns
TCHRYX	READY Hold Time into 8086		30		20		20		ns
TRYLCL	READY Inactive to CLK (See Note 4)		−8		−8		−10		ns
TINVCH	Set-up Time for Recognition (INTR, NMI, $\overline{\text{TEST}}$ (See Note 2)		30		15		15		ns
TGVCH	$\overline{\text{RQ}}/\overline{\text{GT}}$ Set-up Time		30		15		12		ns
TCHGX	$\overline{\text{RQ}}$ Hold Time into 8066		40		30		20		ns
TILIH	Input Rise Time (Except CLK)	From 0.8 to 2.0 V		20		20		20	ns
TIHIL	Input Fall Time (Except CLK)	From 2.0 to 0.8 V		12		12		12	ns

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

SWITCHING CHARACTERISTICS over **COMMERCIAL** and **INDUSTRIAL** ranges (continued)
TIMING RESPONSES

Parameters	Description	Test Conditions	8086		8086-2		8086-1		Units
			Min	Max	Min	Max	Min	Max	
TCLML	Command Active Delay (See Note 1)	C_L = 20-100 pF for all 8086 Outputs (In addition to 8086 self-load)	10	35	10	35	10	35	ns
TCLMH	Command Inactive Delay (See Note 1)		10	35	10	35	10	35	ns
TRYHSH	READY Active to Status Passive (See Note 3)			110		65		45	ns
TCHSV	Status Active Delay		10	110	10	60	10	45	ns
TCLSH	Status Inactive Delay		10	130	10	70	10	55	ns
TCLAV	Address Valid Delay		10	110	10	60	10	50	ns
TCLAX	Address Hold Time		10		10		10		ns
TCLAZ	Address Float Delay		TCLAX	80	TCLAX	50	10	40	ns
TSVLH	Status Valid to ALE High (See Note 1)			15		15		15	ns
TSVMCH	Status Valid to MCE High (See Note 1)			15		15		15	ns
TCLLH	CLK Low to ALE Valid (See Note 1)			15		15		15	ns
TCLMCH	CLK Low to MCE High (See Note 1)			15		15		15	ns
TCHLL	ALE Inactive Delay (See Note 1)			15		15		15	ns
TCLMCL	MCE Inactive Delay (See Note 1)			15		15		15	ns
TCLDV	Data Valid Delay		10	110	10	60	10	50	ns
TCHDX	Data Hold Time		10		10		10		ns
TCVNV	Control Active Delay (See Note 1)		5	45	5	45	5	45	ns
TCVNX	Control Inactive Delay (See Note 1)		10	45	10	45	10	45	ns
TAZRL	Address Float to Read Active		0		0		0		ns
TCLRL	$\overline{RD}$ Active Delay		10	165	10	100	10	70	ns
TCLRH	$\overline{RD}$ Inactive Delay		10	150	10	80	10	60	ns
TRHAV	$\overline{RD}$ Inactive to Next Address Active		TCLCL-45		TCLCL-40		TCLCL-35		ns
TCHDTL	Direction Control Active Delay (See Note 1)			50		50		50	ns
TCHDTH	Direction Control Inactive Delay (See Note 1)			30		30		30	ns
TCLGL	GT Active Delay		0	85	0	50	0	38	ns
TCLGH	GT Inactive Delay		0	85	0	50	0	45	ns
TRLRH	$\overline{RD}$ Width		2TCLCL-75		2TCLCL-50		2TCLCL-40		ns
TOLOH	Output Rise Time	From 0.8 to 2.0 V		20		20		20	ns
TOHOL	Output Fall Time	From 2.0 to 0.8 V		12		12		12	ns

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

ABSOLUTE MAXIMUM RATINGS

Storage Temperature .... −65 to +150°C
Ambient Temperature Under Bias .... 0 to 70°C
Voltage on any Pin
 with Respect to Ground .... −1 to +7.0 V
Power Dissipation .... 2.5 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Military (M) Devices
 Temperature (T_C) .... −55 to +125°C
 Supply Voltage (V_{CC}) .... 5 V ±10%

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over **MILITARY** operating range (for APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted)

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
V_{IL} †	Input LOW Voltage	V_{CC} = Min. & Max.	−0.5*	+0.8	V
V_{IH} †	Input HIGH Voltage	V_{CC} = Min. & Max.	2.0	V_{CC} + 0.5*	V
V_{OL}	Output LOW Voltage	I_{OL} = 2.0 mA, V_{CC} = Min.		0.45	V
V_{OH}	Output HIGH Voltage	I_{OH} = −400 μA, V_{CC} = Min.	2.4		V
I_{CC}	Power Supply Current (Note 1)	T_C = 25°C, V_{CC} = Max.		340	mA
I_{LI}	Input Leakage Current	V_{CC} = Max., V_{IN} = 5.5 V & 0 V	−10	10	μA
I_{LO} ††	Output Leakage Current	V_{CC} = Max., V_{OUT} = 5.5 V & 0.45 V	−10	10	μA
V_{CL} †	Clock Input LOW Voltage	V_{CC} = Min. & Max.	−0.5*	+0.6	V
V_{CH} †	Clock Input HIGH Voltage	V_{CC} = Min. & Max.	3.9	V_{CC} + 1.0*	V
C_{IN} †††	Capacitance of Input Buffer (All Input Except AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)	fc = 1 MHz		20*	pF
C_{IO} †††	Capacitance of I/O Buffer (AD_0–AD_{15}, $\overline{RQ}/\overline{GT}$)	fc = 1 MHz		20*	pF

* Guaranteed by design; not tested.
† Group A, Subgroups 7 and 8 only are tested.
†† Group A, Subgroups 1 and 2 only are tested.
††† Not included in Group A tests.

Notes: 1. I_{CC} is measured while running a functional pattern with spec value I_{OL}/I_{OH} loads applied.

SWITCHING CHARACTERISTICS over MILITARY operating range (for APL Products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8086		8086-2		Unit
			Min.	Max.	Min.	Max.	
TCLCL	CLK Cycle Period (Note 11)		200	500	125	500	ns
TCLCH	CLK LOW Time		118		68		ns
TCHCL	CLK HIGH Time		69		44		ns
TCH1CH2	CLK Rise Time (Note 5)	From 1.0 to 3.5 V		10		10	ns
TCL2CL1	CLK Fall Time (Note 5)	From 3.5 to 1.0 V		10		10	ns
TDVCL	Data in Setup Time		30		20		ns
TCLDX	Data in Hold Time		10		10		ns
TR1VCL	RDY Setup Time into 8284A (Notes 1 & 2)		35		35		ns
TCLR1X	RDY Hold Time into 8284A (Notes 1 & 2)		0		0		ns
TRYHCH	READY Setup Time into 8086		118		68		ns
TCHRYX	READY Hold Time into 8086		30		20		ns
TRYLCL	READY Inactive to CLK (Note 4)		−8		−8		ns
THVCH	HOLD Setup Time		35		20		ns
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (Note 2)		30		15		ns
TILIH	Input Rise Time (Except CLK) (Note 5)	From 0.8 to 2.0 V		2 0		20	ns
TIHIL	Input Fall Time (Except CLK) (Note 5)	From 2.0 to 0.8 V		12		12	ns

Notes:
1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
   V_{IL} = .45 V V_{IHC} = 4.3 V
   V_{ILC} = .25 V V_{OH} = 1.6 V
   V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
    V_{CC} = 4.5 V V_{OL} = 1 V
    V_{IL} = 0 V V_{IH} = 4 V
    V_{ILC} = 0 V V_{IHC} = 5 V

SWITCHING CHARACTERISTICS over MILITARY operating range (continued)

TIMING RESPONSES

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8086		8086-2		Unit
			Min.	Max.	Min.	Max.	
TCLAV	Address Valid Delay	C_L = 100 pF for all 8086 Outputs (in addition to 8086 internal loads)	10	110	10	60	ns
TCLAX	Address Hold Time (Notes 7 & 8)		10		10		ns
TCLAZ	Address Float Delay (Note 8)		10	80	10	50	ns
TLHLL	ALE Width (Note 10)		98		58		ns
TCLLH	ALE Active Delay (Note 8)			80		50	ns
TCHLL	ALE Inactive Delay (Note 8)			85		55	ns
TLLAX	Address Hold Time to ALE Inactive (Note 7)		59		34		ns
TCLDV	Data Valid Delay (Note 8)		10	110	10	60	ns
TCHDX	Data Hold Time (Note 10)		10		10		ns
TWHDX	Data Hold Time After WR (Note 9)		88		38		ns
TCVCTV	Control Active Delay 1 (Note 8)		10	110	10	70	ns
TCHCTV	Control Active Delay 2 (Note 8)		10	110	10	60	ns
TCVCTX	Control Inactive Delay (Note 8)		10	110	10	70	ns
TAZRL	Address Float to READ Active (Note 9)		0		0		ns
TCLRL	$\overline{RD}$ Active Delay (Note 8)		10	165	10	100	ns
TCLRH	$\overline{RD}$ Inactive Delay (Note 8)		10	150	10	80	ns
TRHAV	$\overline{RD}$ Inactive to Next Address Active (Note 10)		155		85		ns
TCLHAV	HLDA Valid Delay (Note 8)		10	160	10	100	ns
TRLRH	$\overline{RD}$ Width (Note 10)		325		200		ns
TWLWH	$\overline{WR}$ Width (Note 10)		340		210		ns
TAVAL	Address Valid to ALE LOW (Note 9)		58		28		ns
TOLOH	Output Rise Time (Note 9)	From 0.8 to 2.0 V		20		20	ns
TOHOL	Output Fall Time (Note 9)	From 2.0 to 0.8 V		12		12	ns

Notes:
1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
   V_{IL} = .45 V V_{IHC} = 4.3 V
   V_{ILC} = .25 V V_{OH} = 1.6 V
   V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
    V_{CC} = 4.5 V V_{OL} = 1 V
    V_{IL} = 0 V V_{IH} = 4 V
    V_{ILC} = 0 V V_{IHC} = 5 V

SWITCHING CHARACTERISTICS over MILITARY operating range (continued)

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER) TIMING REQUIREMENTS

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8086		8086-2		Unit
			Min.	Max.	Min.	Max.	
TCLCL	CLK Cycle Period (Note 11)		200	500	125	500	ns
TCLCH	CLK LOW Time		118		68		ns
TCHCL	CLK HIGH Time		69		44		ns
TCH1CH2	CLK Rise Time (Note 5)	From 1.0 to 3.5 V		10		10	ns
TCL2CL1	CLK Fall Time (Note 5)	From 3.5 to 1.0 V		10		10	ns
TDVCL	Data in Setup Time		30		20		ns
TCLDX	Data in Hold Time		10		10		ns
TR1VCL	RDY Setup Time into 8284A (Notes 1 & 2)		35		35		ns
TCLR1X	RDY Hold Time into 8284A (Notes 1 & 2)		0		0		ns
TRYHCH	READY Setup Time into 8086		118		68		ns
TCHRYX	READY Hold Time into 8086		30		20		ns
TRYLCL	READY Inactive to CLK (Note 4)		−8		−8		ns
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{\text{TEST}}$) (Note 2)		30		15		ns
TGVCH	$\overline{\text{RQ}}/\overline{\text{GT}}$ Setup Time		30		15		ns
TCHGX	$\overline{\text{RQ}}$ Hold Time into 8066		40		30		ns
TILIH	Input Rise Time (Except CLK) (Note 5)	From 0.8 to 2.0 V		20		20	ns
TIHIL	Input Fall Time (Except CLK) (Note 5)	From 2.0 to 0.8 V		12		12	ns

Notes:
1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
   V_{IL} = .45 V V_{IHC} = 4.3 V
   V_{ILC} = .25 V V_{OH} = 1.6 V
   V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
    V_{CC} = 4.5 V V_{OL} = 1 V
    V_{IL} = 0 V V_{IH} = 4 V
    V_{ILC} = 0 V V_{IHC} = 5 V

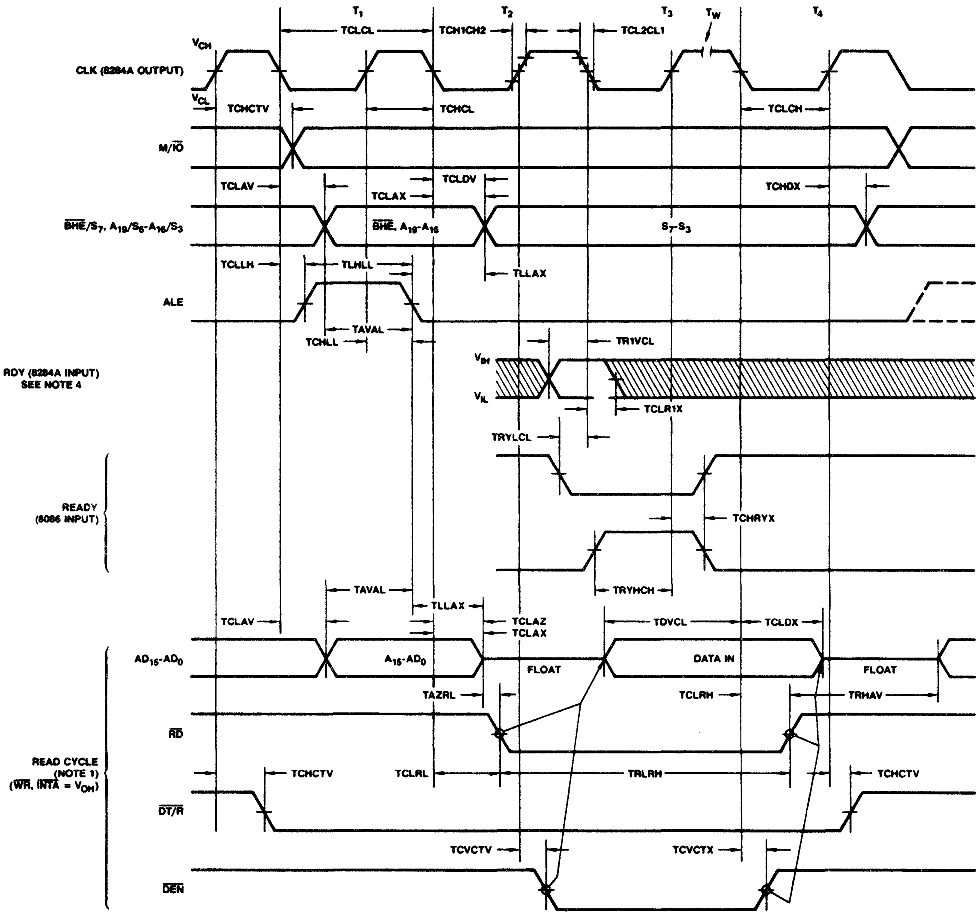
SWITCHING CHARACTERISTICS over **MILITARY** operating range (continued)
TIMING RESPONSES

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8086		8086-2		Unit
			Min.	Max.	Min.	Max.	
TCLML	Command Active Delay (Note 1)	C_L = 100 pF for all 8086 Outputs (In addition to 8086 internal loads)	10	35	10	35	ns
TCLMH	Command Inactive Delay (Note 1)		10	35	10	35	ns
TRYHSH	READY Active to Status Passive (Note 3)			110		65	ns
TCHSV	Status Active Delay (Notes 7 & 8)		10	110	10	60	ns
TCLSH	Status Inactive Delay		10	130	10	70	ns
TCLAV	Address Valid Delay		10	110	10	60	ns
TCLAX	Address Hold Time		10		10		ns
TCLAZ	Address Float Delay		10	80	10	50	ns
TSVLH	Status Valid to ALE HIGH (Note 1)			15		15	ns
TSVMCH	Status Valid to MCE HIGH (Note 1)			15		15	ns
TCLLH	CLK LOW to ALE Valid (Note 1)			15		15	ns
TCLMCH	CLK LOW to MCE HIGH (Note 1)			15		15	ns
TCHLL	ALE Inactive Delay (Note 1)			15		15	ns
TCLMCL	MCE Inactive Delay (Note 1)			15		15	ns
TCLDV	Data Valid Delay		10	110	10	60	ns
TCHDX	Data Hold Time		10		10		ns
TCVNV	Control Active Delay (Note 1)		5	45	5	45	ns
TCVNX	Control Inactive Delay (Note 1)		10	45	10	45	ns
TAZRL	Address Float to Read Active		0		0		ns
TCLRL	$\overline{RD}$ Active Delay		10	165	10	100	ns
TCLRH	$\overline{RD}$ Inactive Delay		10	150	10	80	ns
TRHAV	$\overline{RD}$ Inactive to Next Address Active		155		85		ns
TCHDTL	Direction Control Active Delay (Note 1)			50		50	ns
TCHDTH	Direction Control Inactive Delay (Note 1)			30		30	ns
TCLGL	GT Active Delay (Note 8)		0	85	0	50	ns
TCLGH	GT Inactive Delay (Note 8)		0	85	0	50	ns
TRLRH	$\overline{RD}$ Width		325		200		ns
TOLOH	Output Rise Time	From 0.8 to 2.0 V		20		20	ns
TOHOL	Output Fall Time	From 2.0 to 0.8 V		12		12	ns

Notes:
1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
   V_{IL} = .45 V V_{IHC} = 4.3 V
   V_{ILC} = .25 V V_{OH} = 1.6 V
   V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
    V_{CC} = 4.5 V V_{OL} = 1 V
    V_{IL} = 0 V V_{IH} = 4 V
    V_{ILC} = 0 V V_{IHC} = 5 V

SWITCHING WAVEFORMS

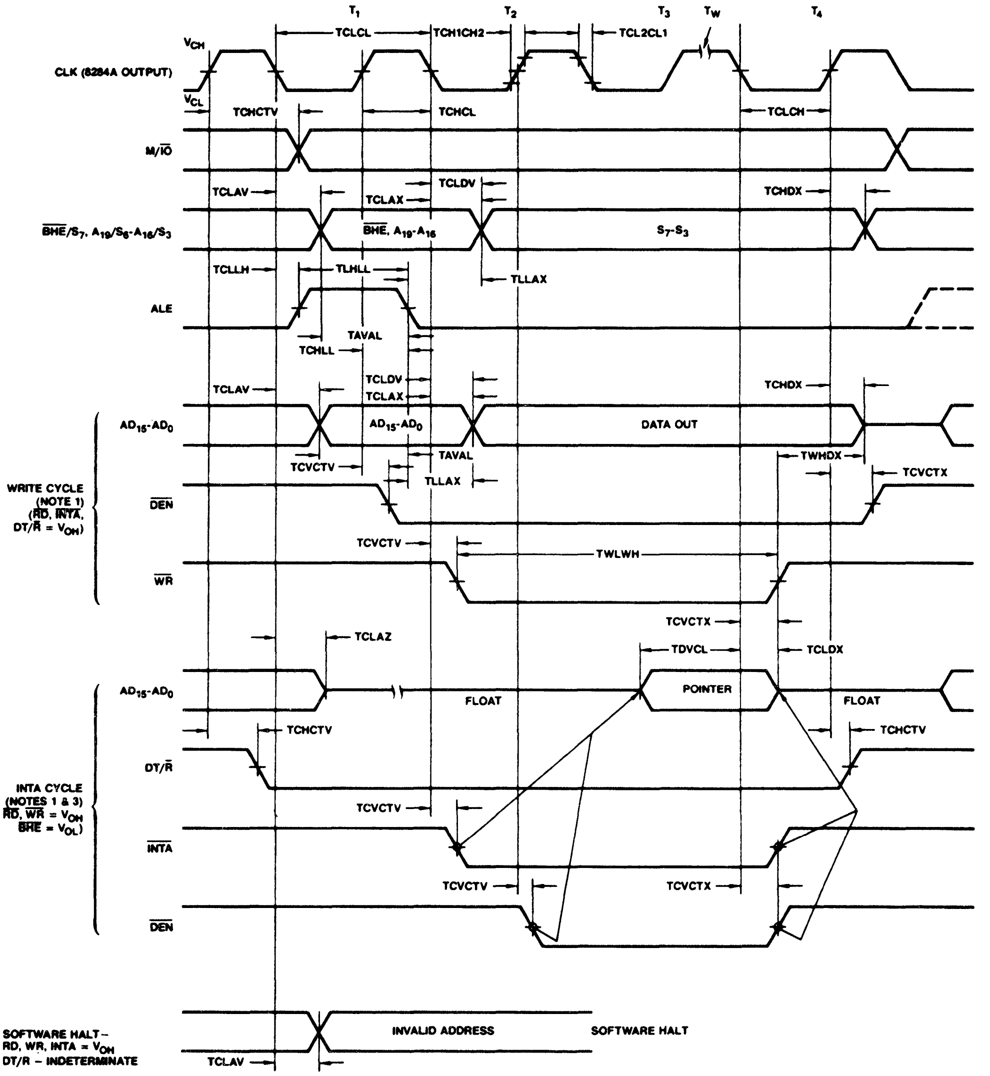
MINIMUM MODE



WF006660

SWITCHING WAVEFORMS (continued)

MINIMUM MODE

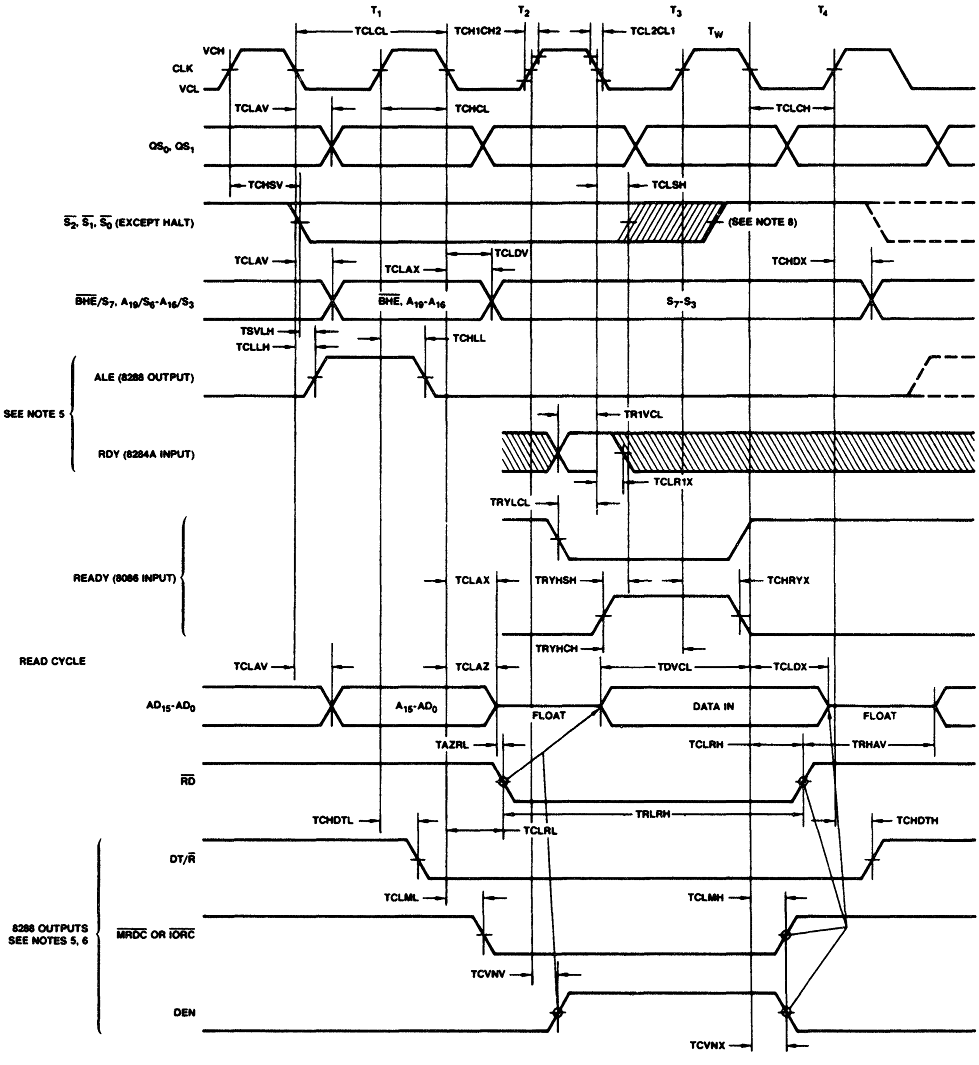


WF006670

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

SWITCHING WAVEFORMS (continued)

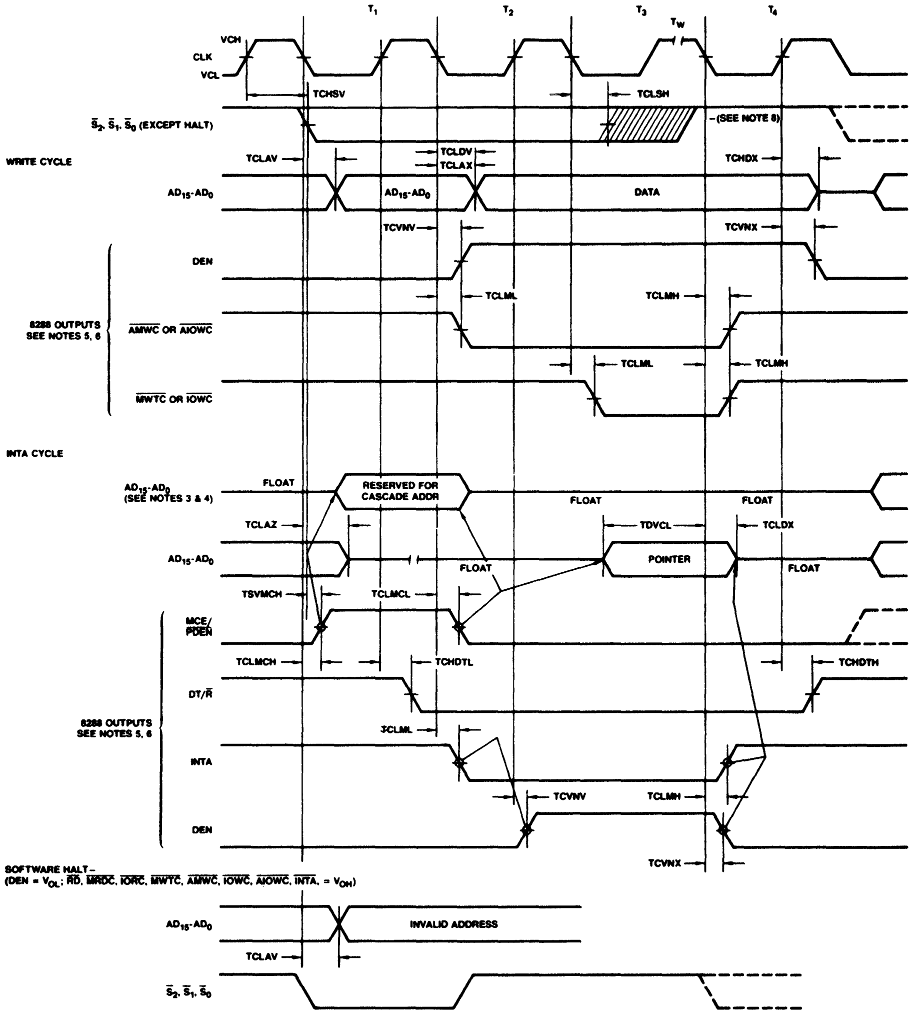
MAXIMUM MODE



WF006680

SWITCHING WAVEFORMS (continued)

MAXIMUM MODE (continued)



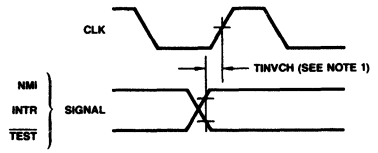
WF006730

Notes: 1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.

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

SWITCHING WAVEFORMS (continued)

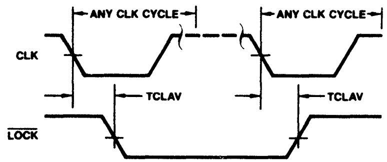
ASYNCHRONOUS SIGNAL RECOGNITION



WF006690

Note: Set-up Requirements for Asynchronous signals only to guarantee recognition at next CLK.

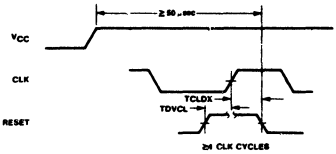
BUS LOCK SIGNAL TIMING



WF006700

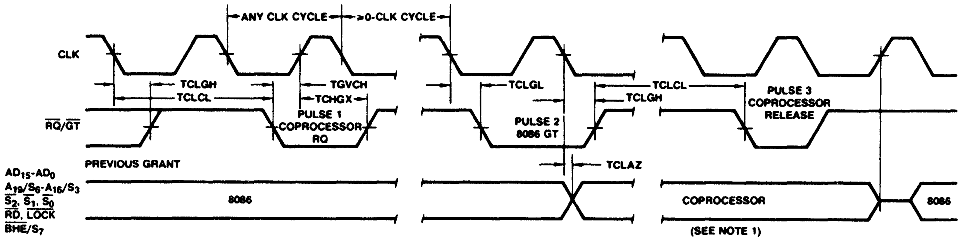
(MAXIMUM MODE ONLY)

RESET TIMING



WF009530

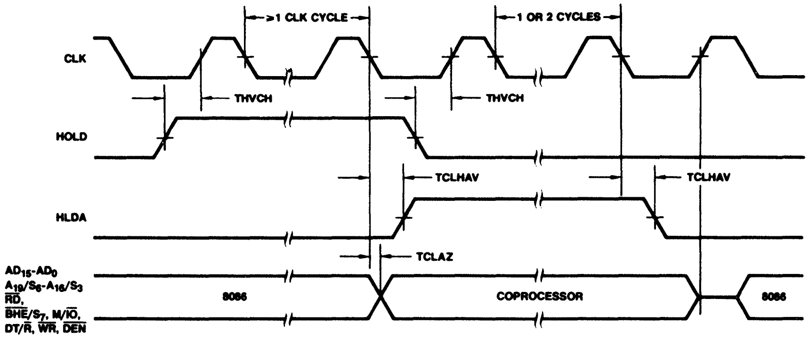
REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



WF006710

Note: The Coprocessor may not drive the buses outside the region shown without risking contention.

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



WF006720

8086/8088
INSTRUCTION SET SUMMARY

DATA TRANSFER

MOV = Move	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to /from register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
Immediate to register	1 0 1 1 w reg	data	data if w = 1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		
PUSH = Push:				
Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m		
Register	0 1 0 1 0 reg			
Segment register	0 0 0 reg 1 1 0			
POP = Pop:				
Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m		
Register	0 1 0 1 1 reg			
Segment register	0 0 0 reg 1 1 1			
XCHG = Exchange:				
Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m		
Register with accumulator	1 0 0 1 0 reg			
IN = Input from:				
Fixed port	1 1 1 0 0 1 0 w	port		
Variable port	1 1 1 0 1 1 0 w			
OUT = Ouput to:				
Fixed port	1 1 1 0 0 1 1 w	port		
Variable port	1 1 1 0 1 1 1 w			
XLAT = Transtate byte to AL	1 1 0 1 0 1 1 1			
LEA = Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m		
LDS = Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		
LES = Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		
LANF = Load AH with flags	1 0 0 1 1 1 1 1			
SANF = Store AH into flags	1 0 0 1 1 1 1 0			
PUSHF = Push flags	1 0 0 1 1 1 0 0			
POPF = Pop flags	1 0 0 1 1 1 0 1			

INSTRUCTION SET SUMMARY (continued)

ARITHMETIC

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
ADD = Add				
Reg/memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register / memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w = 01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w = 1	
ADC = Add with carry:				
Reg/memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w = 01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w = 1	
INC = Increment:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		
Register	0 1 0 0 0 reg			
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1			
SUB = Subtract:				
Reg/memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w = 01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w = 1	
SBB = Subtract with borrow:				
Reg/memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w = 01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w = 1	
DEC = Decrement:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		
CMP = Compare:				
Register/memory with register	0 0 1 1 1 0 1 w	mod reg r/m		
Register with register/memory	0 0 1 1 1 0 0 w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w = 01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w = 1	
AAS ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL Mulitiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL Integer multiply (signed):	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV Divide (unsigned):	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD ASCH adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW Convert byte to word	1 0 0 1 1 0 0 0			
CWD Convert word to double word	1 0 0 1 1 0 0 1			

INSTRUCTION SET SUMMARY (continued)

LOGIC

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
NOT Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m		
SHL/SAL Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m		
SHR Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m		
SAR Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m		
ROL Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m		
ROR Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m		
RCL Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m		
RCR Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m		
AND = And:				
Reg/memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w = 1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w = 1	
TEST = And function to flags, no result:				
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w = 1	
OR = Or:				
Reg/memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w = 1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w = 1	
XOR = Exclusive or:				
Reg/memory and registor to either	0 0 1 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w = 1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w = 1	

STRING MANIPULATION:

REP = Repeat	1 1 1 1 0 0 1 z
MOVS = Move byte/word	1 0 1 0 0 1 0 w
CMPS = Compare byte/word	1 0 1 0 0 1 1 w
SCAS = Scan byte/word	1 0 1 0 1 1 1 w
LODS = Load byte/wd to AL/AX	1 0 1 0 1 1 0 w
STOS = Stor byte/wd from AL/A	1 0 1 0 1 0 1 w

INSTRUCTION SET SUMMARY (continued)

CONTROL TRANSFER

CALL = Call	76543210	76543210	76543210	76543210
Direct within segment	11101000	disp-low	disp-high	
indirect within segment	11111111	mod 010 r/m		
Direct intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect intersegment	11111111	mod 011 r/m		
JMP = Unconditional jump:				
Direct within segment	11101001	disp-low	disp-high	
Direct within segment-short	11101011	disp		
Indirect within segment	11111111	mod 100 r/m		
Direct intersegment	11101010	offset-low	offset-high	
		seg-low	seg-high	
Indirect intersegment	11111111	mod 101 r/m		
RET = Return from CALL:				
Within segment	11000011			
Within seg adding immed to SP	11000010	data-low	data-high	
Intersegment	11001011			
Intersegment adding immediate to SP	11001010	data-low	data-high	
JE/JZ = Jump on equal/zero	01110100	disp		
JL/JNGE = Jump on less/not greater or equal	01111100	disp		
JLE/JNG = Jump on less or equal/not greater	01111110	disp		
JB/JNAE = Jump on below/not above or equal	01110010	disp		
JBE/JNA = Jump on below or equal/not above	01110110	disp		
JP/JPE = Jump on parity/parity even	01111010	disp		
JO = Jump on overflow	01110000	disp		
JS = Jump on sign	01111000	disp		
JNE/JNZ = Jump on not equal/not zero	01110101	disp		
JNL/JGE = Jump on not less/greater or equal	01111101	disp		
JNLE/JG = Jump on not less or equal/greater	01111111	disp		
JNB/JAE = Jump on not below/above or equal	01110011	disp		
JNBE/JA = Jump on not below or equal/above	01110111	disp		
JNP/JPO = Jump on not par/par odd	01111011	disp		
JNO = Jump on not overflow	01110001	disp		
JNS = Jump on not sign	01111001	disp		
LOOP = Loop CX times	11100010	disp		
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp		
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp		
JCXZ = Jump on CX zero	11100011	disp		

INSTRUCTION SET SUMMARY (continued)

CONTROL TRANSFER (Cont'd.)

INT = Interrupt	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Type specified	1 1 0 0 1 1 0 1	type		
Type 3	1 1 0 0 1 1 0 0			
INTO = Interrupt on overflow	1 1 0 0 1 1 1 0			
IRET = Interrupt return	1 1 0 0 1 1 1 1			
PROCESSOR CONTROL				
CLC = Clear carry	1 1 1 1 1 0 0 0			
CMC = Complement carry	1 1 1 1 0 1 0 1			
STC = Set carry	1 1 1 1 1 0 0 1			
CLD = Clear direction	1 1 1 1 1 1 0 0			
STD = Set direction	1 1 1 1 1 1 0 1			
CLI = Clear interrupt	1 1 1 1 1 0 1 0			
STI = Set interrupt	1 1 1 1 1 0 1 1			
HLT = Halt	1 1 1 1 0 1 0 0			
WAIT = Wait	1 0 0 1 1 0 1 1			
ESC = Processor Extension Escape	1 1 0 1 1 x x x	mod x x x r/m		
LOCK = Bus lock prefix	1 1 1 1 0 0 0 0			

Footnotes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive.
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
w = 1 then word instruction; if w = 0 then byte instruction

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

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

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

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

SEGMENT OVERRIDE PREFIX

0	0	1	reg	1	1	0

REG is assigned according to the following table:

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

Instructions which reference the flag register files as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

8088

8-Bit Microprocessor CPU
iAPX86 Family
FINAL

DISTINCTIVE CHARACTERISTICS

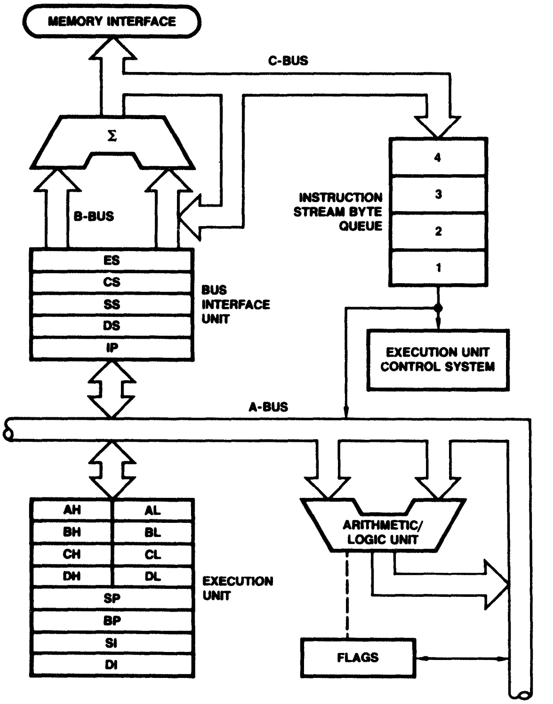
- 8-bit data bus, 16-bit internal architecture
- Directly addresses 1 Mbyte of memory
- Software compatible with 8086 CPU
- Byte, word, and block operations
- 24 operand addressing modes
- Powerful instruction set
- Efficient high level language implementation
- Three speed options: 5MHz 8088
  8MHz 8088-2
  10MHz 8088-1

GENERAL DESCRIPTION

The 8088 CPU is an 8-bit processor designed around the 8086 internal structure. Most functions of the 8088 are identical to the equivalent 8086 functions. The pinout is slightly different. The 8088 handles the external bus the same way the 8086 does, but it handles only 8 bits at a time. Sixteen-bit words are fetched or written in two consecutive bus cycles. Both processors will appear identical to the software engineer, with the exception of execution time.

The 8088 is made with N-channel silicon gate technology and is packaged in a 40-pin Plastic dip, CERDIP or Plastic Leaded Chip Carrier.

BLOCK DIAGRAM



BD003750

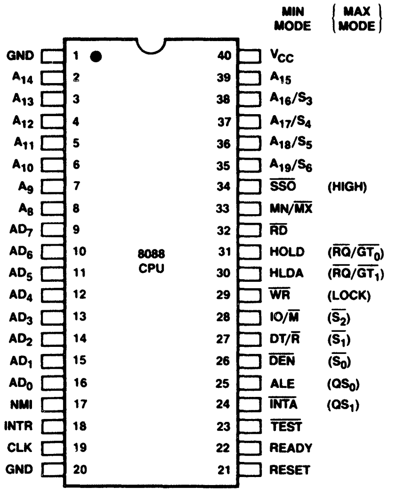
Publication #	Rev.	Amendment
02338	D	/0

Issue Date: August 1989

CONNECTION DIAGRAMS

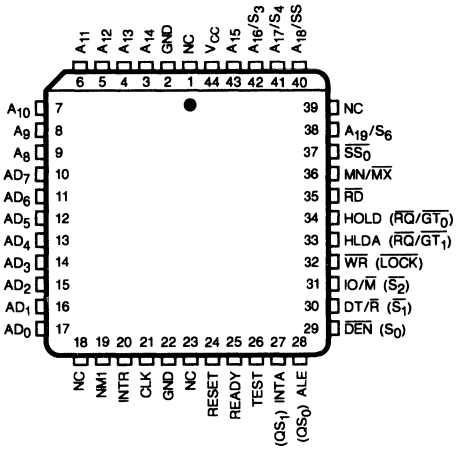
Top View

DIPs



CD005520

PLCC



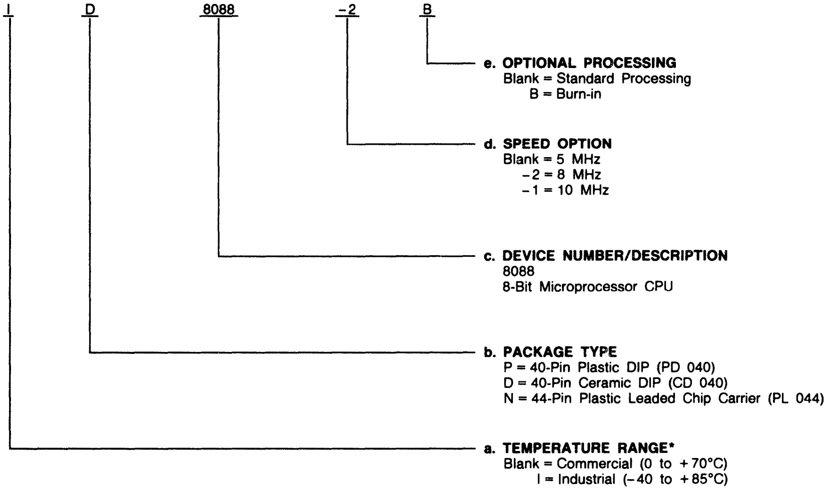
CD010680

Note: Pin 1 is marked for orientation.

ORDERING INFORMATION

Commodity Products

AMD commodity products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of: **a. Temperature Range**
b. Package Type
c. Device Number
d. Speed Option
e. Optional Processing



I D 8088 -2 B

e. OPTIONAL PROCESSING
Blank = Standard Processing
B = Burn-in

d. SPEED OPTION
Blank = 5 MHz
-2 = 8 MHz
-1 = 10 MHz

c. DEVICE NUMBER/DESCRIPTION
8088
8-Bit Microprocessor CPU

b. PACKAGE TYPE
P = 40-Pin Plastic DIP (PD 040)
D = 40-Pin Ceramic DIP (CD 040)
N = 44-Pin Plastic Leaded Chip Carrier (PL 044)

a. TEMPERATURE RANGE*
Blank = Commercial (0 to +70°C)
I = Industrial (-40 to +85°C)

Valid Combinations	
P, N	8088
	8088-2
	8088-1
D	8088B, 8088
	8088-2B, 8088-2
	8088-1B
ID	8088B
	8088-2B

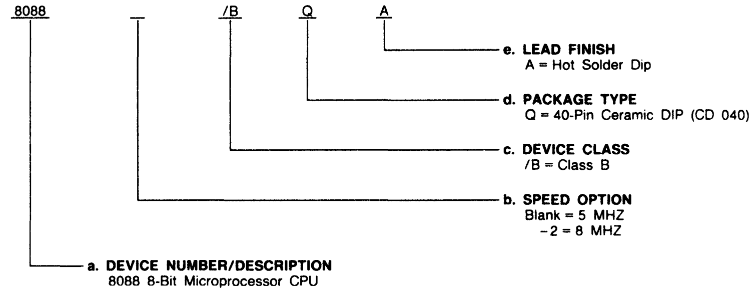
Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

MILITARY ORDERING INFORMATION
APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) for APL products is formed by a combination of:

a. Device Number
b. Speed Option (if applicable)
c. Device Class
d. Package Type
e. Lead Finish



8088 _ /B Q A

e. LEAD FINISH
A = Hot Solder Dip

d. PACKAGE TYPE
Q = 40-Pin Ceramic DIP (CD 040)

c. DEVICE CLASS
/B = Class B

b. SPEED OPTION
Blank = 5 MHZ
-2 = 8 MHZ

a. DEVICE NUMBER/DESCRIPTION
8088 8-Bit Microprocessor CPU

Valid Combinations	
8088	/BQA
8088-2	

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check for newly released valid combinations.

Group A Tests

Group A tests consist of Subgroups 1, 2, 3, 7, 8, 9, 10, 11.

PIN DESCRIPTION

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

<table>
<tr><th>Pin No.*</th><th>Name</th><th>I/O</th><th>Description</th></tr>
<tr><td>9-16</td><td>AD_7–AD_0</td><td>I/O</td><td>Address Data Bus. These lines constitute the time multiplexed memory/IO address (T_1) and data (T_2, T_3, T_W and T_4) bus. These lines are active HIGH and float to three-state OFF during interrupt acknowledge and local bus "hold acknowledge."</td></tr>
<tr><td>39, 2-8</td><td>A_{15}–A_8</td><td>O</td><td>Address Bus. These lines provide address bits 8 through 15 for the entire bus cycle (T_1–T_4). These lines do not have to be latched by ALE to remain valid. A_{15}–A_8 are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge."</td></tr>
<tr><td>35-38</td><td>A_{19}/S_6,
A_{18}/S_5,
A_{17}/S_4,
A_{16}/S_3</td><td>O</td><td>Address/Status. During T_1, these are the four most significant address lines for memory operations. During I/O operations, these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W and T_4. S_6 is always LOW. The status of the interrupt enable flat bit (S_5) is updated at the beginning of each clock cycle. S_4 and S_3 are encoded as shown.
This information indicates which segment register is presently being used for data accessing.
These lines float to three-state OFF during local bus "hold acknowledge."
<table>
<tr><th>S_4</th><th>S_3</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>Alternate Data</td></tr>
<tr><td>0</td><td>1</td><td>Stack</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>Code or None</td></tr>
<tr><td>1</td><td>1</td><td>Data</td></tr>
<tr><td>S_6 is 0 (LOW)</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>32</td><td>$\overline{RD}$</td><td>O</td><td>Read. Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the IO/$\overline{M}$ pin or S_2. This signal is used to read devices which reside on the 8088 local bus. $\overline{RD}$ is active LOW during T_2, T_3 and T_W of any read cycle and is guaranteed to remain HIGH in T_2 until the 8088 local bus has floated.
This signal floats to 3-state OFF in "hold acknowledge."</td></tr>
<tr><td>22</td><td>READY</td><td>I</td><td>READY. The acknowledgment from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory or I/O is synchronized by the 8284 clock generator to form READY. This signal is active HIGH. The 8088 READY input is not synchronized. Correct operation is not guaranteed if the set-up and hold times are not met.</td></tr>
<tr><td>18</td><td>INTR</td><td>I</td><td>Interrupt Request. A level-triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.</td></tr>
<tr><td>23</td><td>$\overline{TEST}$</td><td>I</td><td>$\overline{TEST}$. Input is examined by the "wait for test" instruction. If the $\overline{TEST}$ input is LOW, execution continues; otherwise, the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.</td></tr>
<tr><td>17</td><td>NMI</td><td>I</td><td>Non-Maskable Interrupt. An edge-triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This Input is internally synchronized.</td></tr>
<tr><td>21</td><td>RESET</td><td>I</td><td>RESET. Causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the instruction set description, when RESET returns LOW. RESET is internally synchronized.</td></tr>
<tr><td>19</td><td>CLK</td><td>I</td><td>Clock. Provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.</td></tr>
<tr><td>40</td><td>V_{CC}</td><td></td><td>V_{CC}. The +5 V ±10% power supply pin.</td></tr>
<tr><td>1, 20</td><td>GND</td><td></td><td>GND. The ground pins.</td></tr>
<tr><td>33</td><td>MIN/$\overline{MX}$</td><td>I</td><td>Minimum/Maximum. Indicates what mode the processor is to operate in. The two modes are discussed in the following sections.</td></tr>
<tr><td>28</td><td>IO/$\overline{M}$</td><td>O</td><td>Status Line. An inverted maximum mode $\overline{S}_2$. It is used to distinguish a memory access from an I/O access. IO/$\overline{M}$ becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle (I/O = HIGH, M = LOW). IO/$\overline{M}$ floats to three-state OFF in local bus "hold acknowledge."</td></tr>
<tr><td>29</td><td>$\overline{WR}$</td><td>O</td><td>Write. Strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the IO/$\overline{M}$ signal. WR is active for T_2, T_3 and T_W of any write cycle. It is active LOW and floats to 3-state OFF in local bus "hold acknowledge."</td></tr>
<tr><td>24</td><td>$\overline{INTA}$</td><td>O</td><td>INTA. Used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3 and T_W of each interrupt acknowledge cycle.</td></tr>
<tr><td>25</td><td>ALE</td><td>O</td><td>Address Latch Enable. Provided by the processor to latch the address into 8282/8283 address latch. It is a HIGH pulse active during clock low of T_1 of any bus cycle. Note that ALE is never floated.</td></tr>
<tr><td>27</td><td>DT/$\overline{R}$</td><td>O</td><td>Data Transmit/Receive. Needed in a minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{R}$ is equivalent to $\overline{S}_1$ in the maximum mode, and its timing is the same as for IO/$\overline{M}$ (T = HIGH, R = LOW.) This signal floats to three-state OFF in local bus "hold acknowledge."</td></tr>
<tr><td>26</td><td>$\overline{DEN}$</td><td>O</td><td>Data Enable. Provided as an output enable for the 8286/8287 in a minimum system that uses the transceiver. $\overline{DEN}$ is active LOW during each memory and I/O access and for $\overline{INTA}$ cycles. For a read or $\overline{INTA}$ cycle, it is active from the middle of T_2 until the middle of T_4; while for a write cycle, it is active from the beginning of T_2 until the middle of T_4. $\overline{DEN}$ floats to 3-state OFF during local bus "hold acknowledge."</td></tr>
</table>

*Pin numbers correspond to DIPs only.

PIN DESCRIPTION (continued)

<table>
<tr><th>Pin No.*</th><th>Name</th><th>I/O</th><th>Description</th></tr>
<tr><td>31, 30</td><td>HOLD,
HLDA</td><td>I/O</td><td>HOLD. Indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgment in the middle of a T_4 or T_1 clock cycle. Simultaneous with the issuance of HLDA, the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor lowers HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines.
HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the set-up time.</td></tr>
<tr><td>34</td><td>$\overline{SSO}$</td><td>O</td><td>Status Line. Logically equivalent to $\overline{SO}$ in the maximum mode. The combination of $\overline{SSO}$, IO/$\overline{M}$ and DT/$\overline{R}$ allows the system to completely decode the current bus cycle status.
<table>
<tr><th>IO/$\overline{M}$</th><th>DT/$\overline{R}$</th><th>$\overline{SSO}$</th><th>Characteristics</th></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read I/O port</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write I/O port</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>0 (LOW)</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read memory</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write memory</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Passive</td></tr>
</table></td></tr>
<tr><td>28-26</td><td>$\overline{S}_2$, $\overline{S}_1$, $\overline{S}_0$</td><td>O</td><td>Status. Active during clock high of T_4, T_1 and T_2 and is returned to the passive state (1, 1, 1) during T_3 or during T_W when READY is HIGH. This status is used by the 8288 bus controller to generate all memory and I/O access control signals. Any change by $\overline{S}_2$, $\overline{S}_1$ or $\overline{S}_0$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle.
These signals float to three-state OFF during "hold acknowledge." During the first clock cycle after RESET becomes active, these signals are active HIGH. After this first clock, they float to three-state OFF.
<table>
<tr><th>$\overline{S}_2$</th><th>$\overline{S}_1$</th><th>$\overline{S}_0$</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Read I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Write I/O Port</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Halt</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>Code Access</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Read Memory</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Write Memory</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Passive</td></tr>
</table></td></tr>
<tr><td>31, 30</td><td>$\overline{RQ}/\overline{GT}_0$,
$\overline{RQ}/\overline{GT}_1$</td><td>I/O</td><td>Request/Grant. Pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT}_0$ having higher priority than $\overline{RQ}/\overline{GT}_1$. $\overline{RQ}/\overline{GT}$ has an internal pull-up resistor, so may be left unconnected. The request/grant sequence is as follows:
1. A pulse of one CLK wide from another local bus master indicates a local bus request ("hold") to the 8088 (pulse 1).
2. During a T_4 or T_1 clock cycle, a pulse one clock wide from the 8088 to the requesting master (pulse 2), indicates that the 8088 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge." The same rules as for HOLD/HLDA apply as for when the bus is released.
3. A pulse one CLK wide from the requesting master indicates to the 8088 (pulse 3) that the "hold" request is about to end and that the 8088 can reclaim the local bus at the next CLK. The CPU then enters T_4.
Each master-master exchange of the local bus is a sequence of three pulses. There must be one idle CLK cycle after each bus exchange. Pulses are active LOW.
If the request is made while the CPU is performing a memory cycle, it will release the local bus during T_4 of the cycle when all the following conditions are met:
1. Request occurs on or before T_2.
2. Current cycle is not the low bit of a word.
3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
4. A locked instruction is not currently executing.
If the local bus is idle when the request is made, two possible events will follow:
1. Local bus will be released during the next clock.
2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.</td></tr>
<tr><td>29</td><td>$\overline{LOCK}$</td><td>O</td><td>$\overline{LOCK}$. Indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active (LOW). The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW and floats to 3-state off in "hold acknowledge."</td></tr>
</table>

*Pin numbers correspond to DIPs only.

PIN DESCRIPTION (continued)

<table>
<tr><th>Pin No.*</th><th>Name</th><th>I/O</th><th>Description</th></tr>
<tr><td>24, 25</td><td>QS_1, QS_0</td><td>O</td><td>Queue Status. Provides status to allow external tracking of the internal 8088 instruction queue. The queue status is valid during the CLK cycle after which the queue operation is performed.
<table>
<tr><th>QS_1</th><th>QS_0</th><th>Characteristics</th></tr>
<tr><td>0 (LOW)</td><td>0</td><td>No Operation</td></tr>
<tr><td>0</td><td>1</td><td>First Byte of Opcode from Queue</td></tr>
<tr><td>1 (HIGH)</td><td>0</td><td>Empty the Queue</td></tr>
<tr><td>1</td><td>1</td><td>Subsequent Byte from Queue</td></tr>
</table>
</td></tr>
<tr><td>34</td><td>–</td><td>O</td><td>Pin 34 is always HIGH in the maximum mode.</td></tr>
</table>

*Pin numbers correspond to DIPs only.

DETAILED DESCRIPTION

The 8088 Compared to the 8086

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

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

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

- $A_8 - A_{15}$ — These pins are only address outputs on the 8088. These address lines are latched internally and remain valid throughout a bus cycle in a manner similar to the 8085 upper address lines.
- $\overline{BHE}$ has no meaning on the 8088 and has been eliminated.
- $\overline{SSO}$ provides the $\overline{SO}$ status information in the minimum mode. This output occurs on pin 34 in minimum mode only. DT/$\overline{R}$, IO/$\overline{M}$, and $\overline{SSO}$ provide the complete bus status in minimum mode.
- IO/$\overline{M}$ has been inverted to be compatible with the MCS-85 bus structure.
- ALE is delayed by one clock cycle in the minimum mode when entering HALT, to allow the status to be latched with ALE.

I/O Addressing

In the 8088, I/O operations can address up to a maximum of 64K I/O registers. The I/O address appears in the same format as the memory address on bus lines $A_{15} - A_0$. The address lines $A_{19} - A_{16}$ are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer, have full address capability, while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

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

Bus Operation

The 8088 address/data bus is broken into three parts — the lower eight address/data bits ($AD_0 - AD_7$), the middle eight address bits ($A_8 - A_{15}$) and the upper four address bits ($A_{16} - A_{19}$). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor, permitting the use of a standard 40 lead package. The middle eight address bits are not multiplexed; i.e., they remain valid throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T1, T2, T3 and T4. The address is emitted from the processor during T1 and data transfer occurs on the bus during T3 and T4. T2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "wait" states (Tw) are inserted between T3 and T4. Each inserted "wait" state is of the same duration as a CLK cycle. Periods can occur between 8088 driven bus cycles. These are referred to as "idle" states (Ti) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

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

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

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

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

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

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

External Interface

Processor Reset and Initialization

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8088 RESET is required to be HIGH for greater than four clock cycles. The 8088 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 clock cycles. After this interval the 8088 operates normally, beginning with the instruction in absolute location FFFF0H (see Figure 3). The RESET input is internally synchronized to the processor clock. At initialization, the HIGH to LOW transition of RESET must occur no sooner than 50 μs after power up, to allow complete initialization of the 8088.

If INTR is asserted sooner than nine clock cycles after the end of RESET, the processor may execute one instruction before responding to the interrupt.

All three-state outputs float to three-state OFF during RESET. Status is active in the idle state for the first clock after RESET becomes active and then floats to three-state OFF.

Interrupt Operations

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

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

Non-Maskable Interrupt (NMI)

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

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

Maskable Interrupt (INTR)

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

During the response sequence (see Figure 1), the processor executes two successive (back to back) interrupt acknowledge cycles. The 8088 emits the LOCK signal (maximum mode only) from T2 of the first bus cycle until T2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle, a byte is fetched from the external interrupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The interrupt return instruction includes a flags pop which returns the status of the original interrupt enable bit when it restores the flags.

HALT

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

Read/Modify/Write (Semaphore) Operations via LOCK

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

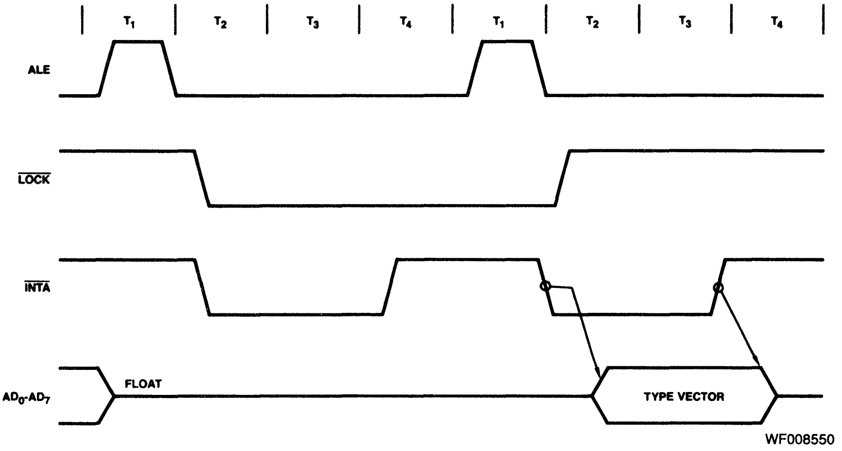


Figure 1. Interrupt Acknowledge Sequence

External Synchronization via $\overline{\text{TEST}}$

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

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

Basic System Timing

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

System Timing — Minimum System

The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal. The trailing (low going) edge of this signal is used to latch the address information, which is valid on the address/data bus (AD0 - AD7) at this time, into the 8282/8283 latch. Address lines A8 through A15 do not need to be latched because they remain valid throughout the bus cycle. From T1 to T4 the IO/$\overline{\text{M}}$ signal indicates a memory or I/O operation. At T2 the address is removed from the address/data bus, and the bus goes to a high impedance state. The read control signal is also asserted at T2. The read ($\overline{\text{RD}}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later, valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again three-state its bus drivers. If a transceiver (8286/8287) is required to buffer the 8088 local bus, signals DT/$\overline{\text{R}}$ and $\overline{\text{DEN}}$ are provided by the 8088.

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

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge ($\overline{\text{INTA}}$) signal is asserted in place of the read ($\overline{\text{RD}}$) signal and the address bus is floated (see Figure 1). In the second of two successive $\overline{\text{INTA}}$ cycles, a byte of information is read from the data bus, as supplied by the interrupt system logic (i.e., 8259A priority interrupt controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into the interrupt vector lookup table, as described earlier.

Bus Timing — Medium Complexity Systems

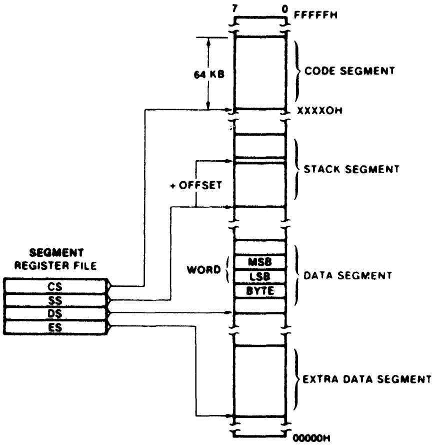
For medium complexity systems, the MN/$\overline{\text{MX}}$ pin is connected to GND and the 8288 bus controller is added to the system, as well as an 8282/8283 latch for latching the system address, and an 8286/8287 transceiver to allow for bus loading greater than the 8088 is capable of handling. Signals ALE, $\overline{\text{DEN}}$ and DT/$\overline{\text{R}}$ are generated by the 8288 instead of the processor in this configuration, although their timing remains relatively the same. The 8088 status outputs ($\overline{\text{S2}}$, $\overline{\text{S1}}$ and $\overline{\text{S0}}$) provide type of cycle information and become 8288 inputs. This bus cycle information specifies read (code, data or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence, data is not valid at the leading edge of write. The 8286/8287 transceiver receives the usual T and $\overline{OE}$ inputs from the 8288's DT/$\overline{R}$ and $\overline{DEN}$ outputs.

The pointer into the interrupt vector table, which is passed during the second $\overline{INTA}$ cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8289A priority interrupt controller is positioned on the local bus, a TTL gate is required to disable the 8286/8287 transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll."

Memory Organization

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries (see Figure 2).

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



DF004530

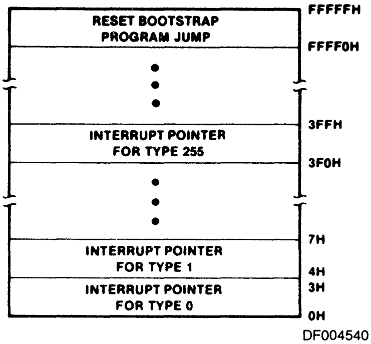
Figure 2. Memory Organization

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

Certain locations in memory are reserved for specific CPU operations (see Figure 3). Locations from addresses FFFF0H through FFFFFH are reserved for operations including a jump to the initial system initialization routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be located. Locations 00000H through 003FFH are reserved for interrupt operations. Four-byte pointers consisting of a 16-bit segment address and a 16-bit offset address direct program flow to one of the 256 possible interrupt service routines. The pointer elements are assumed to have been stored at their respective places in reserved memory prior to the occurrence of interrupts.

Minimum and Maximum Modes

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



DF004540

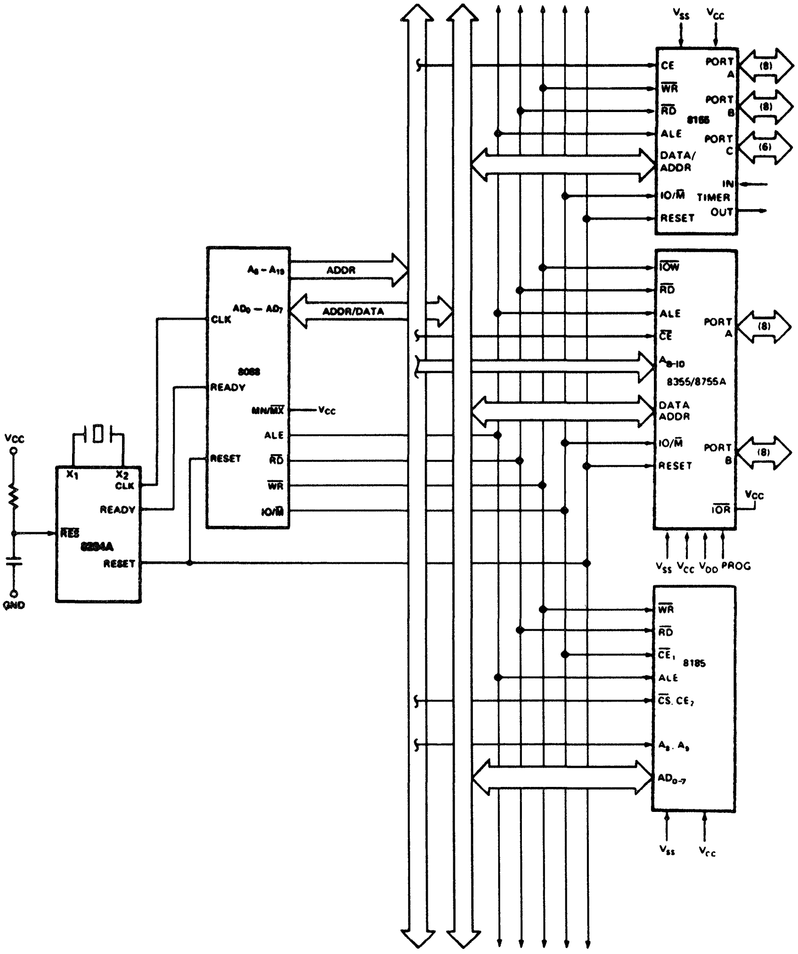
Figure 3. Reserved Memory Locations

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

The minimum mode 8088 can be used with either a multiplexed or demultiplexed bus. The multiplexed bus configuration is compatible with the MCS-85^{TM} multiplexed bus peripherals (8155, 8156, 8355, 8755A, and 8185). This configuration (see Figure 4) provides the user with a minimum chip count system. This architecture provides the 8088 processing power in a highly integrated form.

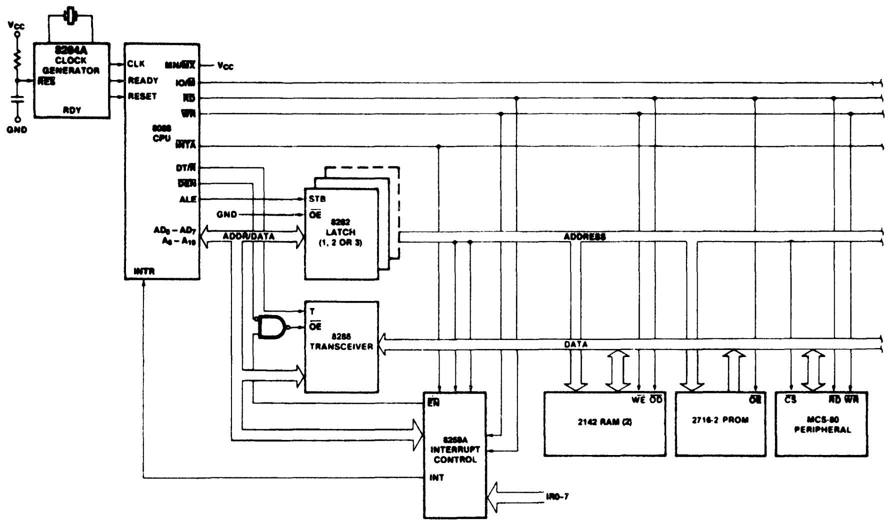
The demultiplexed mode requires one latch (for 64K addressability) or two latches (for a full megabyte of addressing). A third latch can be used for buffering if the address bus loading requires it. An 8286 or 8287 transceiver can also be used if data bus buffering is required (see Figure 5). The 8088 provides $\overline{\text{DEN}}$ and DT/$\overline{\text{R}}$ to control the transceiver, and ALE to latch the addresses. This configuration of the minimum mode provides the standard demultiplexed bus structure with heavy bus buffering and relaxed bus timing requirements.

The maximum mode employs the 8288 bus controller (see Figure 6). The 8288 decodes status lines $\overline{\text{S0}}$, $\overline{\text{S1}}$ and $\overline{\text{S2}}$ and provides the system with all bus control signals. Moving the bus control to the 8288 provides better source and sink current capability to the control lines and frees the 8088 pins for extended large system features. Hardware lock, queue status and two request/grant interfaces are provided by the 8088 in maximum mode. These features allow co-processors in local bus and remote bus configurations.



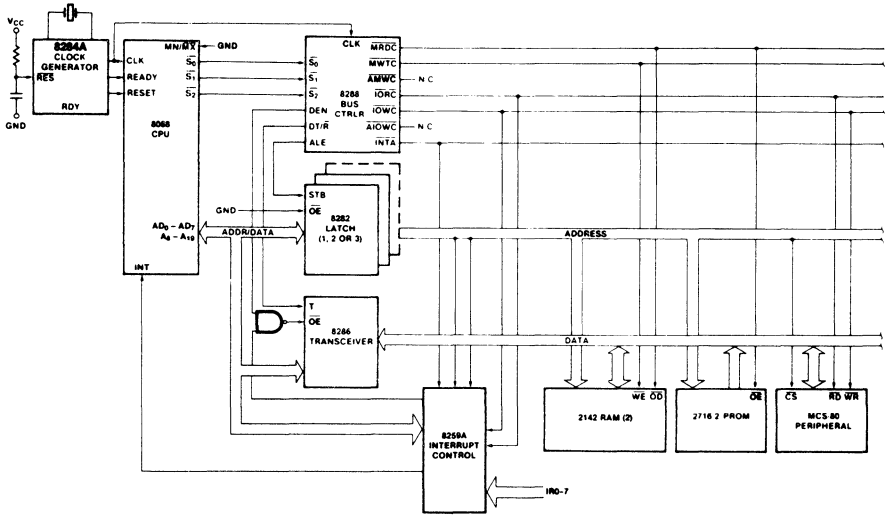
AF003451

Figure 4. Multiplexed Bus Configuration



AF003461

Figure 5. Demultiplexed Bus Configuration



AF003470

Figure 6. Fully Buffered System Using Bus Controller

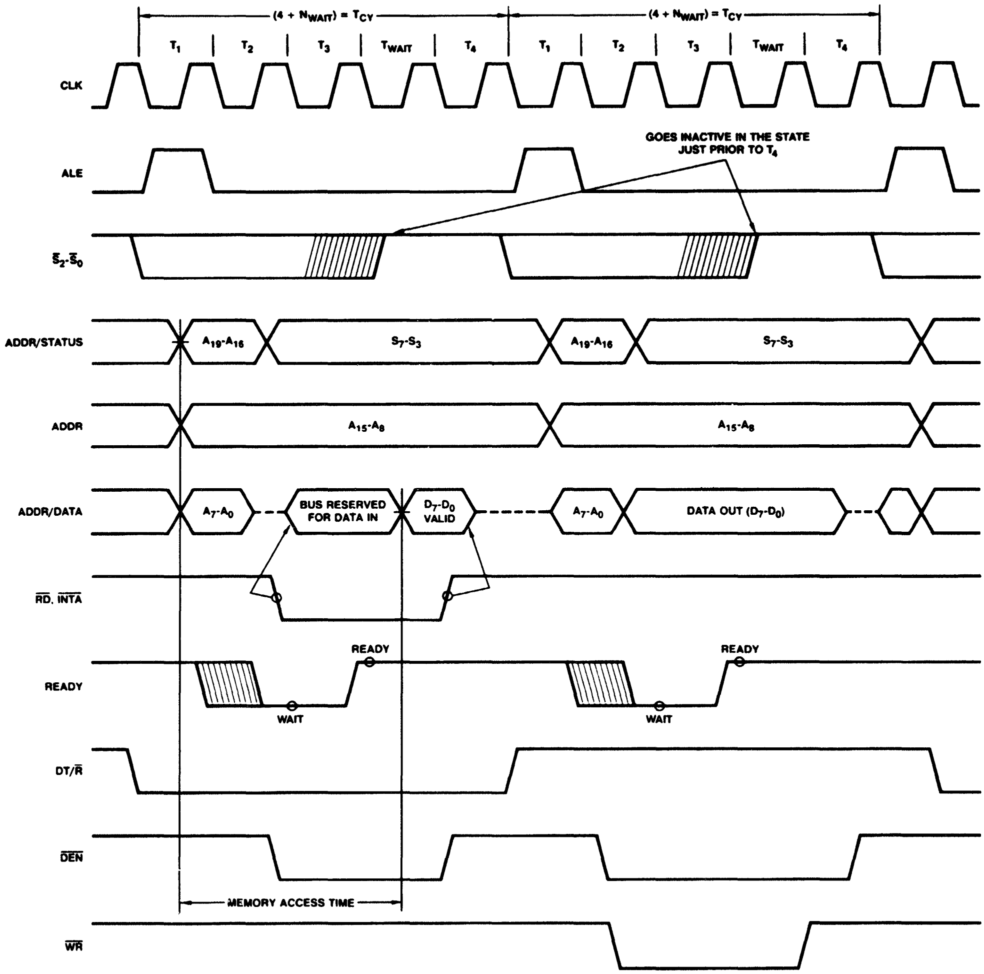
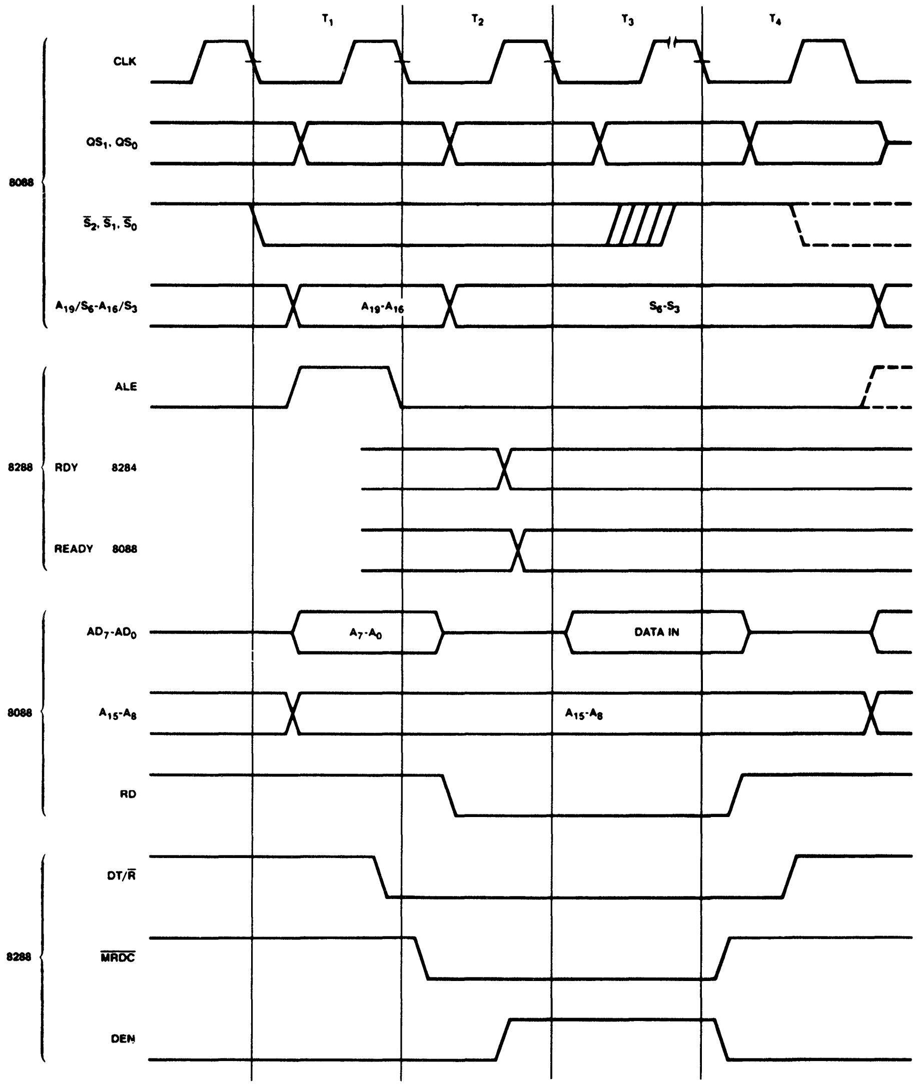


Figure 7. Basic System Timing



WF006751

Figure 8. Medium Complexity System Timing

ABSOLUTE MAXIMUM RATINGS

Storage Temperature ....−65 to +150°C
Voltage on any Pin
with Respect to Ground ....−1.0 to +7.0 V
Power Dissipation ....2.5 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices
Temperature (T_A)................................0 to +70°C
Supply Voltage (V_{CC})
8088 ....5 V ± 10%
8088-1, 8088-2..................................5 V ± 5%

Industrial (I) Devices
Temperature (T_A)..............................−40 to +85°C
Supply Voltage (V_{CC})
8088 ....5 V ± 10%
8088-1, 8088-2..................................5 V ± 5%

Military (M) Devices
Temperature (T_C).............................−55 to +125°C
Supply Voltage (V_{CC}) ....5 V ± 10%

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating range (for APL, Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted)

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Units
V_{IL}†	Input Low Voltage	COML: see Note 1	−0.5*	+0.8	V
		MIL: V_{CC} = Min. & Max.			
V_{IH}†	Input High Voltage	COML: see Notes 1 & 2	2.0	V_{CC} +0.5*	V
		MIL: V_{CC} = Min. & Max.			
V_{OL}	Output Low Voltage	COML: I_{OL} = 2.0 mA		0.45	V
		MIL: I_{OL} = 2.0 mA V_{CC} = Min.			
V_{OH}	Output High Voltage	COML: I_{OH} = −400 µA	2.4		V
		MIL: I_{OH} = −400 µA V_{CC} = Min.			
I_{CC}	Power Supply Current (Note 6)	MIL: T_C = 25°C, V_{CC} = Max.		340	mA
I_{LI}	Input Leakage Current	COML: 0 V ≤ V_{IN} ≤ V_{CC}		±10	µA
		MIL: V_{CC} = Max. V_{IN} = 5.5 V & 0 V	−10	10	
I_{LO}††	Output Leakage Current	COML: 0.45 V ≤ V_{OUT} ≤ V_{CC}		COML ±10	µA
		MIL: V_{CC} = Max. V_{OUT} = 5.5 V & 0.45 V	MIL −10	MIL 10	
V_{CL}	Clock Input Low Voltage		−0.5	+0.6	V
V_{CH}	Clock Input High Voltage		3.9	V_{CC} +1.0	V
C_{IN}	Capacitance of Input Buffer (All input except AD_0–AD_7, RQ/GT)	fc = 1 MHz		15	pF
C_{IO}	Capacitance of I/O Buffer (AD_0–AD_7, RQ/GT)	fc = 1 MHz		15	pF
I_{CC}	Power Supply Current	T_A = 25°C 8088		340	mA
		8088-1, -2		350	
		P8088		250	

Notes: 1. V_{IL} tested with MN/$\overline{MX}$ pin = 0 V; V_{IH} tested with MN/$\overline{MX}$ pin = 5 V; MN/$\overline{MX}$ is a strap pin.
2. Not applicable to $\overline{RQ}/\overline{GT0}$ and $\overline{RQ}/\overline{GT1}$ pins (pins 30 and 31).
3. Signal at 8284 or 8288 shown for reference only.
4. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
5. Applies only to T_3 and Wait states.
6. I_{CC} is measured while running a functional pattern with spec value I_{OL}/I_{OH} loads applied.
* Guaranteed by design; not tested.
† Group A, Subgroups 7 and 8 only are tested.
†† Group A, Subgroups 1 and 2 only are tested.

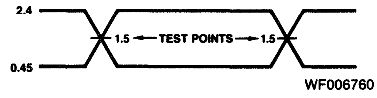
SWITCHING CHARACTERISTICS over COMMERCIAL operating range
MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Parameter Symbol	Parameter Description	Test Conditions	8088		8088-2		8088-1		Units
			Min	Max	Min	Max	Min	Max	
TCLCL	CLK Cycle Period		200	500	125	500	100	500	ns
TCLCH	CLK Low Time		118		68		53		ns
TCHCL	CLK High Time		69		44		39		ns
TCH1CH2	CLK Rise Time	From 1.0 to 3.5 V		10		10		10	ns
TCL2CL1	CLK Fall Time	From 3.5 to 1.0 V		10		10		10	ns
TDVCL	Data in Set-up Time		30		20		5		ns
TCLDX	Data in Hold Time		10		10		10		ns
TR1VCL	RDY Set-up Time into 8284 (See Notes 3, 4)		35		35		35		ns
TCLR1X	RDY Hold Time into 8284 (See Notes 3, 4)		0		0		0		ns
TRYHCH	READY Set-up Time into 8088		118		68		53		ns
TCHRYX	READY Hold Time into 8088		30		20		20		ns
TRYLCL	READY Inactive to CLK (See Note 5)		-8		-8		-10		ns
THVCH	HOLD Set-up Time		35		20		20		ns
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Set-up Time (See Note 4)		30		15		15		ns
TILIH	Input Rise Time (Except CLK)	From 0.8 to 2.0 V		20		20		20	ns
TIHIL	Input Fall Time (Except CLK)	From 2.0 to 0.8 V		12		12		12	ns

SWITCHING CHARACTERISTICS over COMMERCIAL operating range (continued)
TIMING RESPONSES

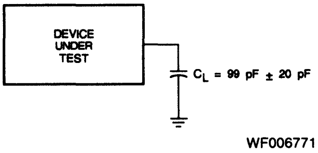
Parameter Symbol	Parameter Description	Test Conditions	8088		8088-2		8088-1		Units
			Min	Max	Min	Max	Min	Max	
TCLAV	Address Valid Delay	C_L = 20-100 pF for all 8088 Outputs (in addition to internal loads)	10	110	10	60	10	50	ns
TCLAX	Address Hold Time		10		10		10		ns
TCLAZ	Address Float Delay		TCLAX	80	TCLAX	50	10	40	ns
TLHLL	ALE Width		TCLCH -20		TCLCH -10		TCLCH -10		ns
TCLLH	ALE Active Delay			80		50		40	ns
TCHLL	ALE Inactive Delay			85		55		45	ns
TLLAX	Address Hold Time to ALE Inactive		TCHCL -10		TCHCL -10		TCHCL -10		ns
TCLDV	Data Valid Delay		10	110	10	60	10	50	ns
TCHDX	Data Hold Time		10		10		10		ns
TWHDX	Data Hold Time After $\overline{\text{WR}}$		TCLCH -30		TCLCH -30		TCLCH -25		ns
TCVCTV	Control Active Delay 1		10	110	10	70	10	50	ns
TCHCTV	Control Active Delay 2		10	110	10	60	10	45	ns
TCVCTX	Control Inactive Delay		10	110	10	70	10	50	ns
TAZRL	Address Float to READ Active		0		0		0		ns
TCLRL	$\overline{\text{RD}}$ Active Delay		10	165	10	100	10	70	ns
TCLRH	$\overline{\text{RD}}$ Inactive Delay		10	150	10	80	10	60	ns
TRHAV	$\overline{\text{RD}}$ Inactive to Next Address Active		TCLCL -45		TCLCL -40		TCLCL -35		ns
TCLHAV	HLDA Valid Delay		10	160	10	100	10	60	ns
TRLRH	$\overline{\text{RD}}$ Width		2TCLCL -75		2TCLCL -50		2TCLCL -40		ns
TWLWH	$\overline{\text{WR}}$ Width		2TCLCL -60		2TCLCL -40		2TCLCL -35		ns
TAVAL	Address Valid to ALE Low		TCLCH -60		TCLCH -40		TCLCH -35		ns
TOLOH	Output Rise Time	From 0.8 to 2.0 V		20		20		20	ns
TOHOL	Output Fall Time	From 2.0 to 0.8 V		12		12		12	ns

SWITCHING TEST INPUT/OUTPUT WAVEFORM



AC testing inputs are driven at 2.4 V for a logic "1" and 0.45 V for a logic "0." The clock is driven at 4.3 V and 0.25 V. Timing measurements are made at 1.5 V for both a logic "1" and "0."

SWITCHING TEST LOAD CIRCUIT



C_L Includes JIG Capacitance.

SWITCHING CHARACTERISTICS over COMMERCIAL operating range (continued)

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER) TIMING REQUIREMENTS

Parameter Symbol	Parameter Description	Test Conditions	8088		8088-2		8088-1		Units
			Min	Max	Min	Max	Min	Max	
TCLCL	CLK Cycle Period		200	500	125	500	100	500	ns
TCLCH	CLK Low Time		118		68		53		ns
TCHCL	CLK High Time		69		44		39		ns
TCH1CH2	CLK Rise Time	From 1.0 to 3.5 V		10		10		10	ns
TCL2CL1	CLK Fall Time	From 3.5 to 1.0 V		10		10		10	ns
TDVCL	Data in Set-up Time		30		20		5		ns
TCLDX	Data in Hold Time		10		10		10		ns
TR1VCL	RDY Set-up Time into 8284 (See Notes 1, 2)		35		35		35		ns
TCLR1X	RDY Hold Time into 8284 (See Notes 1,2)		0		0		0		ns
TRYHCH	READY Set-up Time into 8088		118		68		53		ns
TCHRYX	READY Hold Time into 8088		30		20		20		ns
TRYLCL	READY Inactive to CLK (See Note 3)		-8		-8		-10		ns
TINVCH	Set-up Time for Recognition (INTR, NMI, TEST) (See Note 2)		30		15		15		ns
TGVCH	RQ/GT Set-up Time		30		15		12		ns
TCHGX	RQ Hold Time into 8086		40		30		20		ns
TILIH	Input Rise Time (Except CLK)	From 0.8 to 2.0 V		20		20		20	ns
TIHIL	Input Fall Time (Except CLK)	From 2.0 to 0.8 V		12		12		12	ns

Notes: 1. Signal at 8284 or 8288 shown for reference only.
2. Set-up requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T_3 and Wait states.

SWITCHING CHARACTERISTICS over COMMERCIAL operating range (continued)
TIMING RESPONSES

Parameter Symbol	Parameter Description	Test Conditions	8088		8088-2		8088-1		Units
			Min	Max	Min	Max	Min	Max	
TCLML	Command Active Delay (See Note 1)	C_L = 20-100 pF for all 8088 outputs (in addition to internal loads)	10	35	10	35	10	35	ns
TCLMH	Command Inactive Delay (See Note 1)		10	35	10	35	10	35	ns
TRYHSH	READY Active to Status Passive (See Note 3)			110		65		45	ns
TCHSV	Status Active Delay		10	110	10	60	10	45	ns
TCLSH	Status Inactive Delay		10	130	10	70	10	55	ns
TCLAV	Address Valid Delay		10	110	10	60	10	50	ns
TCLAX	Address Hold Time		10		10		10		ns
TCLAZ	Address Float Delay		TCLAX	80	TCLAX	50	10	40	ns
TSVLH	Status Valid to ALE High (See Note 1)			15		15		15	ns
TSVMCH	Status Valid to MCE High (See Note 1)			15		15		15	ns
TCLLH	CLK Low to ALE Valid (See Note 1)			15		15		15	ns
TCLMCH	CLK Low to MCE High (See Note 1)			15		15		15	ns
TCHLL	ALE Inactive Delay (See Note 1)			15		15		15	ns
TCLMCL	MCE Inactive Delay (See Note 1)			15		15		15	ns
TCLDV	Data Valid Delay		10	110	10	60	10	50	ns
TCHDX	Data Hold Time		10		10		10		ns
TCVNV	Control Active Delay (See Note 1)		5	45	5	45	5	45	ns
TCVNX	Control Inactive Delay (See Note 1)		10	45	10	45	10	45	ns
TAZRL	Address Float to Read Active		0		0		0		ns
TCLRL	RD Active Delay		10	165	10	100	10	70	ns
TCLRH	RD Inactive Delay		10	150	10	80	10	60	ns
TRHAV	RD Inactive to Next Address Active		TCLCL -45		TCLCL -40		TCLCL -35		ns
TCHDTL	Direction Control Active Delay (See Note 1)			50		50		50	ns
TCHDTH	Direction Control Inactive Delay (See Note 1)			30		30		30	ns
TCLGL	GT Active Delay			85		50	0	45	ns
TCLGH	GT Inactive Delay			85		50	0	45	ns
TRLRH	RD Width		2TCLCL -75		2TCLCL -50		2TCLCL -40		ns
TOLOH	Output Rise Time	From 0.8 to 2.0 V		20		20		20	ns
TOHOL	Output Fall Time	From 2.0 to 0.8 V		12		12		12	ns

Notes: 1. Signal at 8284 or 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T_2 state (8 ns into T_3 state).

SWITCHING CHARACTERISTICS over **MILITARY** operating range (for APL Products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8088		8088-2		Unit
			Min.	Max.	Min.	Max.	
TCLCL	CLK Cycle Period (Note 11)		200	500	125	500	ns
TCLCH	CLK LOW Time		118		68		ns
TCHCL	CLK HIGH Time		69		44		ns
TCH1CH2	CLK Rise Time (Note 5)	From 1.0 to 3.5 V		10		10	ns
TCL2CL1	CLK Fall Time (Note 5)	From 3.5 to 1.0 V		10		10	ns
TDVCL	Data in Setup Time		30		20		ns
TCLDX	Data in Hold Time		10		10		ns
TR1VCL	RDY Setup Time into 8284A (Notes 1 & 2)		35		35		ns
TCLR1X	RDY Hold Time into 8284A (Notes 1 & 2)		0		0		ns
TRYHCH	READY Setup Time into 8088		118		68		ns
TCHRYX	READY Hold Time into 8088		30		20		ns
TRYLCL	READY Inactive to CLK (Note 3)		-8		-8		ns
THVCH	HOLD Setup Time		35		20		ns
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (Note 2)		30		15		ns
TILIH	Input Rise Time (Except CLK) (Note 5)	From 0.8 to 2.0 V		20		20	ns
TIHIL	Input Fall Time (Except CLK) (Note 5)	From 2.0 to 0.8 V		12		12	ns

Notes: 1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
V_{IL} = .45 V V_{IHC} = 4.3 V
V_{ILC} = .25 V V_{OH} = 1.6 V
V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
V_{CC} = 4.5 V V_{OL} = 1 V
V_{IL} = 0 V V_{IH} = 4 V
V_{ILC} = 0 V V_{IHC} = 5 V

SWITCHING CHARACTERISTICS over MILITARY operating range (continued)
TIMING RESPONSES

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8088		8088-2		Unit
			Min.	Max.	Min.	Max.	
TCLAV	Address Valid Delay	C_L = 100 pF for all 8088 Outputs (in addition to internal loads).	10	110	10	60	ns
TCLAX	Address Hold Time (Notes 7 & 8)		10		10		ns
TCLAZ	Address Float Delay (Note 8)		10	80	10	50	ns
TLHLL	ALE Width (Note 10)		98		58		ns
TCLLH	ALE Active Delay (Note 8)			80		50	ns
TCHLL	ALE Inactive Delay (Note 8)			85		55	ns
TLLAX	Address Hold Time to ALE Inactive (Note 7)		59		34		ns
TCLDV	Data Valid Delay (Note 8)		10	110	10	60	ns
TCHDX	Data Hold Time (Note 10)		10		10		ns
TWHDX	Data Hold Time After WR (Note 9)		88		38		ns
TCVCTV	Control Active Delay 1 (Note 8)		10	110	10	70	ns
TCHCTV	Control Active Delay 2 (Note 8)		10	110	10	60	ns
TCVCTX	Control Inactive Delay (Note 8)		10	110	10	70	ns
TAZRL	Address Float to READ Active (Note 9)		0		0		ns
TCLRL	$\overline{RD}$ Active Delay (Note 8)		10	165	10	100	ns
TCLRH	$\overline{RD}$ Inactive Delay (Note 8)		10	150	10	80	ns
TRHAV	$\overline{RD}$ Inactive to Next Address Active (Note 10)		155		85		ns
TCLHAV	HLDA Valid Delay (Note 8)		10	160	10	100	ns
TRLRH	$\overline{RD}$ Width (Note 10)		325		200		ns
TWLWH	$\overline{WR}$ Width (Note 10)		340		210		ns
TAVAL	Address Valid to ALE Low (Note 9)		58		28		ns
TOLOH	Output Rise Time (Note 9)	From 0.8 to 2.0 V		20		20	ns
TOHOL	Output Fall Time (Note 9)	From 2.0 to 0.8 V		12		12	ns

Notes:
1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
   V_{IL} = .45 V V_{IHC} = 4.3 V
   V_{ILC} = .25 V V_{OH} = 1.6 V
   V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
    V_{CC} = 4.5 V V_{OL} = 1 V
    V_{IL} = 0 V V_{IH} = 4 V
    V_{ILC} = 0 V V_{IHC} = 5 V

SWITCHING CHARACTERISTICS over MILITARY operating range (continued)
MAX MODE SYSTEM (USING 8288 BUS CONTROLLER) TIMING REQUIREMENTS

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8088		8088-2		Unit
			Min.	Max.	Min.	Max.	
TCLCL	CLK Cycle Period (Note 11)		200	500	125	500	ns
TCLCH	CLK LOW Time		118		68		ns
TCHCL	CLK HIGH Time		69		44		ns
TCH1CH2	CLK Rise Time (Note 5)	From 1.0 to 3.5 V		10		10	ns
TCL2CL1	CLK Fall Time (Note 5)	From 3.5 to 1.0 V		10		10	ns
TDVCL	Data in Setup Time		30		20		ns
TCLDX	Data in Hold Time		10		10		ns
TR1VCL	RDY Setup Time into 8284A (Notes 1 & 2)		35		35		ns
TCLR1X	RDY Hold Time into 8284A (Notes 1 & 2)		0		0		ns
TRYHCH	READY Setup Time into 8088		118		68		ns
TCHRYX	READY Hold Time into 8088		30		20		ns
TRYLCL	READY Inactive to CLK (Note 3)		-8		-8		ns
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{\text{TEST}}$ (Note 2)		30		15		ns
TGVCH	$\overline{\text{RQ}}/\overline{\text{GT}}$ Setup Time		30		15		ns
TCHGX	$\overline{\text{RQ}}$ Hold Time into 8086		40		30		ns
TILIH	Input Rise Time (Except CLK) (Note 5)	From 0.8 to 2.0 V		20		20	ns
TIHIL	Input Fall Time (Except CLK) (Note 5)	From 2.0 to 0.8 V		12		12	ns

Notes: 1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
V_{IL} = .45 V V_{IHC} = 4.3 V
V_{ILC} = .25 V V_{OH} = 1.6 V
V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
V_{CC} = 4.5 V V_{OL} = 1 V
V_{IL} = 0 V V_{IH} = 4 V
V_{ILC} = 0 V V_{IHC} = 5 V

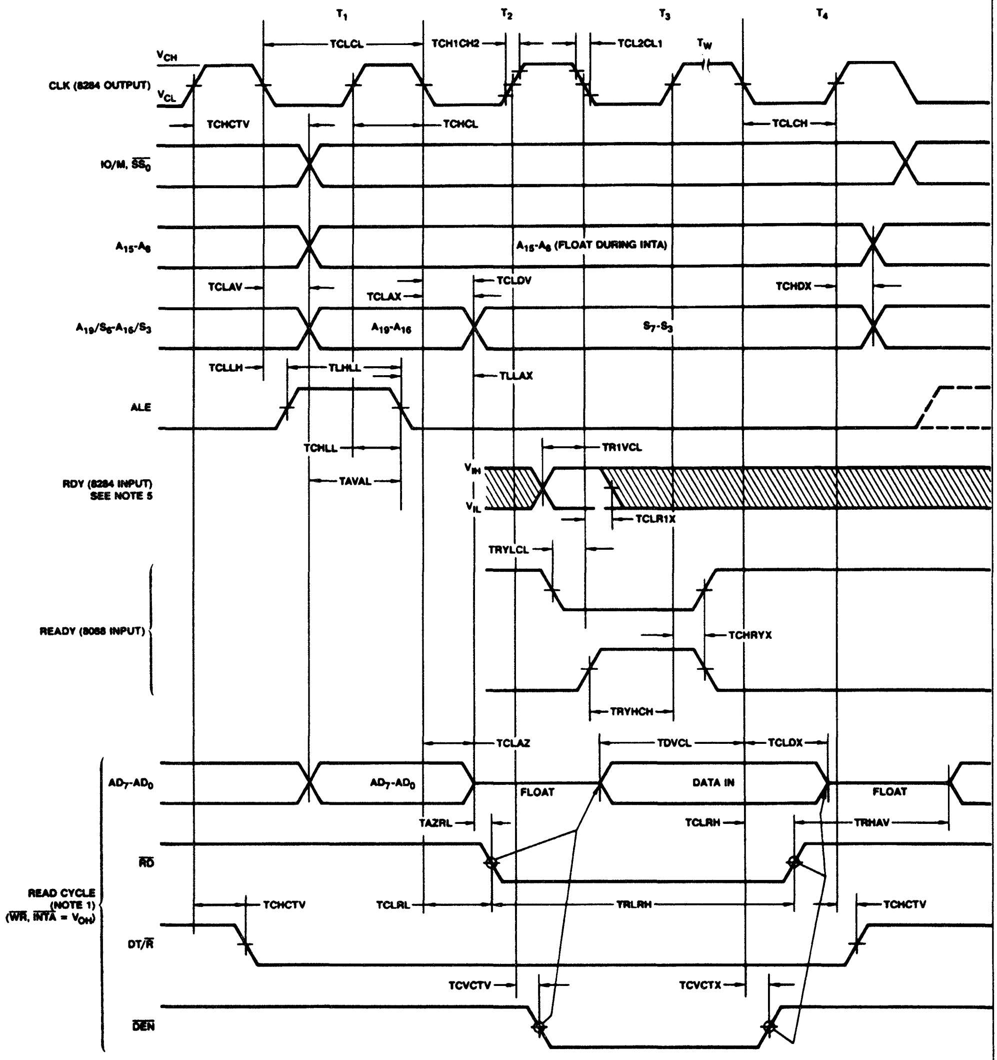
SWITCHING CHARACTERISTICS over MILITARY operating range (continued)
TIMING RESPONSES

Parameter Symbol	Parameter Description	Test Conditions (Note 6)	8088		8088-2		Unit
			Min.	Max.	Min.	Max.	
TCLML	Command Active Delay (Note 1)	C_L = 100 pF for all 8088 Outputs (In addition to internal loads)	10	35	10	35	ns
TCLMH	Command Inactive Delay (Note 1)		10	35	10	35	ns
TRYHSH	READY Active to Status Passive (Note 4)			110		65	ns
TCHSV	Status Active Delay (Notes 7 & 8)		10	110	10	60	ns
TCLSH	Status Inactive Delay		10	130	10	70	ns
TCLAV	Address Valid Delay		10	110	10	60	ns
TCLAX	Address Hold Time		10		10		ns
TCLAZ	Address Float Delay		10	80	10	50	ns
TSVLH	Status Valid to ALE HIGH (Note 1)			15		15	ns
TSVMCH	Status Valid to MCE HIGH (Note 1)			15		15	ns
TCLLH	CLK LOW to ALE Valid (Note 1)			15		15	ns
TCLMCH	CLK LOW to MCE HIGH (Note 1)			15		15	ns
TCHLL	ALE Inactive Delay (Note 1)			15		15	ns
TCLMCL	MCE Inactive Delay (Note 1)			15		15	ns
TCLDV	Data Valid Delay		10	110	10	60	ns
TCHDX	Data Hold Time		10		10		ns
TCVNV	Control Active Delay (Note 1)		5	45	5	45	ns
TCVNX	Control Inactive Delay (Note 1)		10	45	10	45	ns
TAZRL	Address Float to Read Active		0		0		ns
TCLRL	$\overline{RD}$ Active Delay		10	165	10	100	ns
TCLRH	$\overline{RD}$ Inactive Delay		10	150	10	80	ns
TRHAV	$\overline{RD}$ Inactive to Next Address Active		155		85		ns
TCHDTL	Direction Control Active Delay (Note 1)			50		50	ns
TCHDTH	Direction Control Inactive Delay (Note 1)			30		30	ns
TCLGL	GT Active Delay (Note 8)			110		50	ns
TCLGH	GT Inactive Delay (Note 8)			85		50	ns
TRLRH	$\overline{RD}$ Width		325		200		ns
TOLOH	Output Rise Time	From 0.8 to 2.0 V		20		20	ns
TOHOL	Output Fall Time	From 2.0 to 0.8 V		12		12	ns

Notes:
1. Signal at 8284A and 8288 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).
5. Not tested; these specs are controlled by the Teradyne J941 tester.
6. V_{CC} = 4.5 V, 5.5 V V_{IH} = 2.4 V
   V_{IL} = .45 V V_{IHC} = 4.3 V
   V_{ILC} = .25 V V_{OH} = 1.6 V
   V_{OL} = 1.4 V
7. Minimum spec tested at V_{CC} Max. (5.5 V) only.
8. Maximum spec tested at V_{CC} Min. (4.5 V) only.
9. Tested at V_{CC} Max. (5.5 V) only.
10. Tested at V_{CC} Min. (4.5 V) only.
11. Test conditions for TCLCL Max. are:
    V_{CC} = 4.5 V V_{OL} = 1 V
    V_{IL} = 0 V V_{IH} = 4 V
    V_{ILC} = 0 V V_{IHC} = 5 V

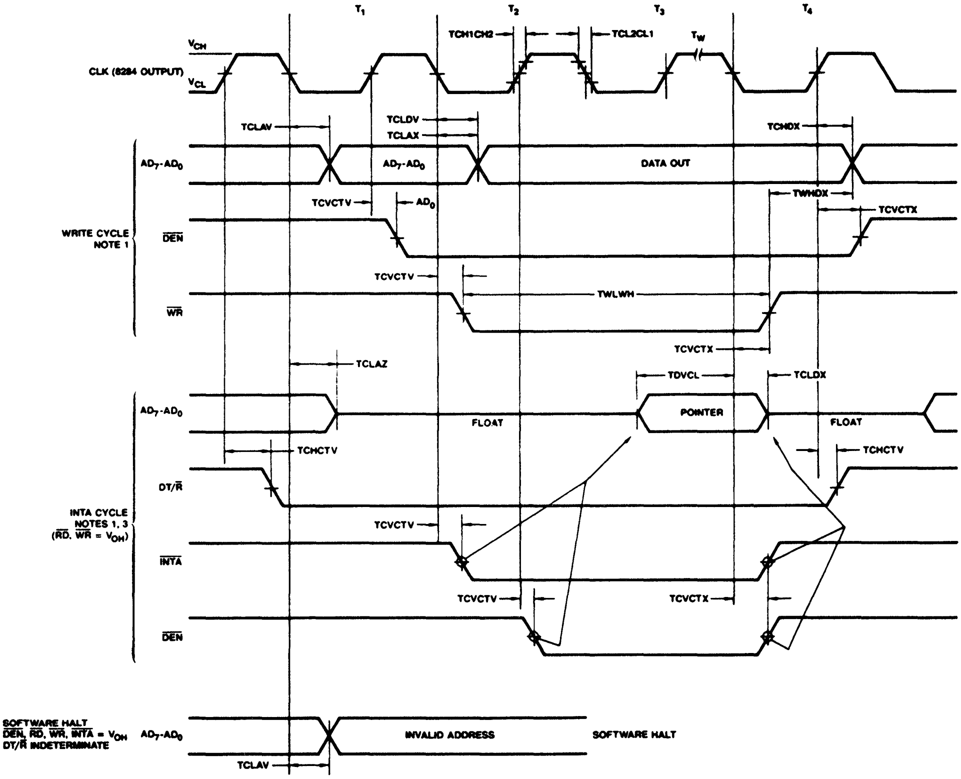
SWITCHING WAVEFORMS

BUS TIMING - MINIMUM MODE SYSTEM



SWITCHING WAVEFORMS

BUS TIMING – MINIMUM MODE SYSTEM (continued)



WF006780

Notes: 1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.

2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.

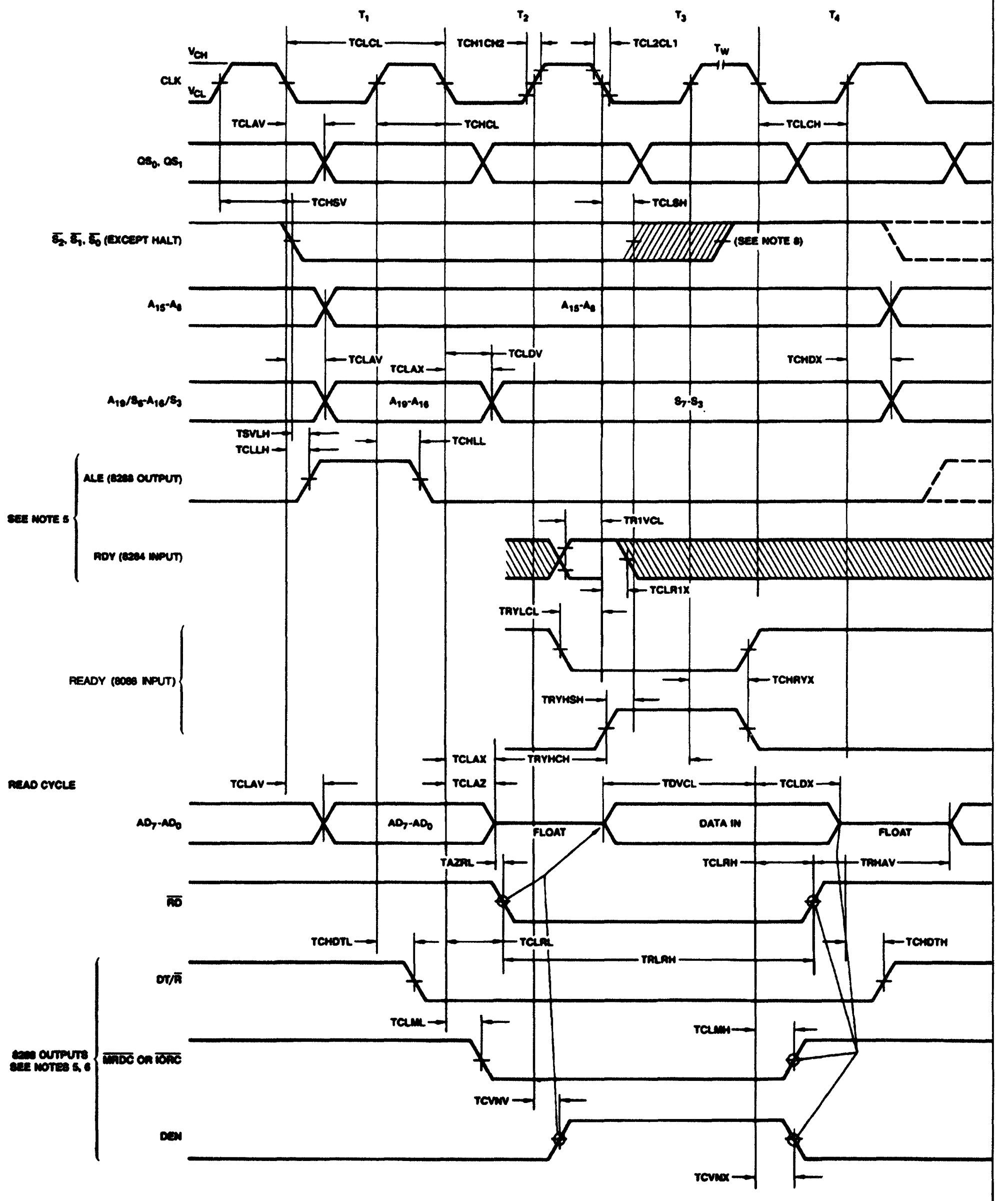
3. Two INTA cycles run back-to-back. The 8088 local ADDR/DATA bus is floating during both INTA cycles. Control signals are shown for the second INTA cycle.

4. Signals at 8284 are shown for reference only.

5. All timing measurements are made at 1.5 V unless otherwise noted.

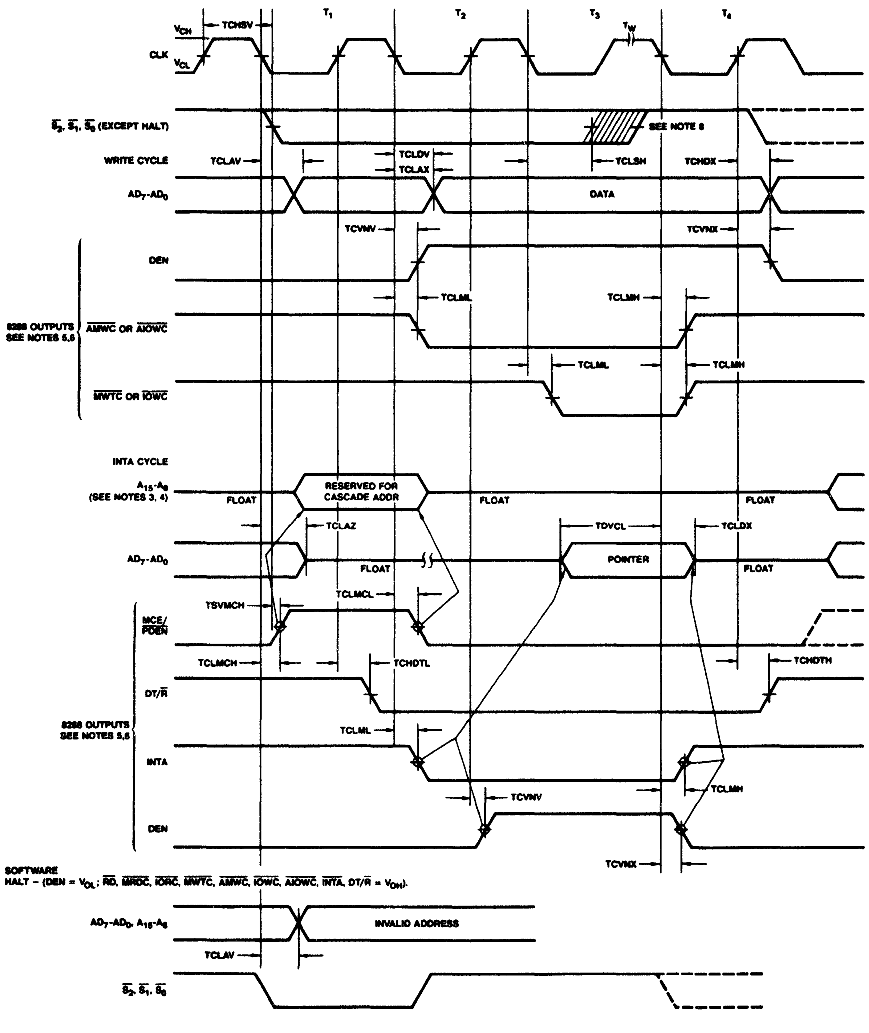
SWITCHING WAVEFORMS (continued)

BUS TIMING - MAXIMUM MODE



SWITCHING WAVEFORMS (continued)

BUS TIMING – MAXIMUM MODE SYSTEM (USING 8288)

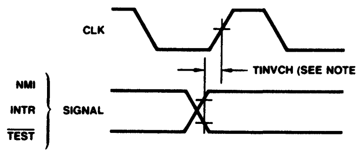


WF006801

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

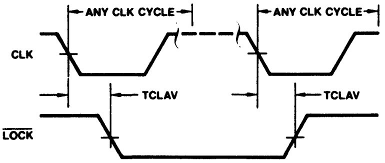
SWITCHING WAVEFORMS (continued)

ASYNCHRONOUS SIGNAL RECOGNITION



WF006820

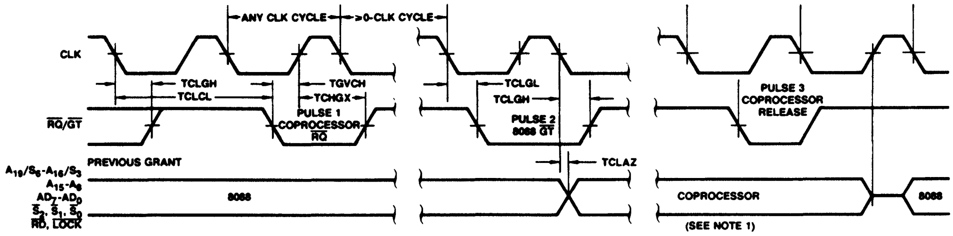
BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



WF006830

Note: Set-up requirements for asynchronous signals only to guarantee recognition at next CLK.

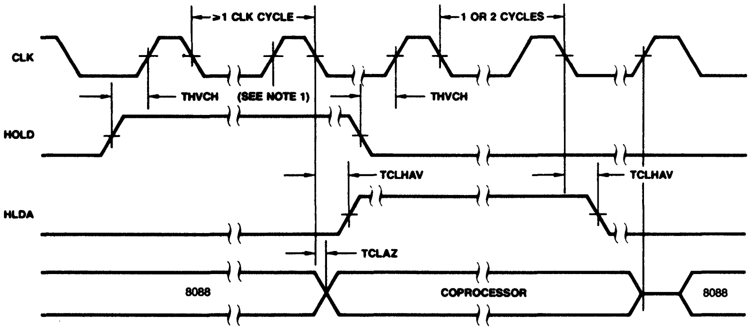
REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



WF006840

Note: The coprocessor may not drive the buses outside the region shown without rising contention.

HOLD/HOLD ACKNOWLEDGE TIMING (MIMIMUM MODE ONLY)



WF006851

Note: All signals switch between V_{OH} and V_{OL} unless otherwise specified.

8086/8088
INSTRUCTION SET SUMMARY

DATA TRANSFER

MOV = Move	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
Immediate to register	1 0 1 1 w reg	data	data if w = 1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		
PUSH = Push:				
Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m		
Register	0 1 0 1 0 reg			
Segment register	0 0 0 reg 1 1 0			
POP = Pop:				
Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m		
Register	0 1 0 1 1 reg			
Segment register	0 0 0 reg 1 1 1			
XCHG = Exchange:				
Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m		
Register with accumulator	1 0 0 1 0 reg			
IN = Input from:				
Fixed port	1 1 1 0 0 1 0 w	port		
Variable port	1 1 1 0 1 1 0 w			
OUT = Output to:				
Fixed port	1 1 1 0 0 1 1 w	port		
Variable port	1 1 1 0 1 1 1 w			
XLAT = Transtate byte to AL	1 1 0 1 0 1 1 1			
LEA = Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m		
LDS = Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		
LES = Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		
LANF = Load AH with flags	1 0 0 1 1 1 1 1			
SANF = Store AH into flags	1 0 0 1 1 1 1 0			
PUSHF = Push flags	1 0 0 1 1 1 0 0			
POPF = Pop flags	1 0 0 1 1 1 0 1			

INSTRUCTION SET SUMMARY (continued)

ARITHMETIC

ADD = Add	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Reg/memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register / memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w = 01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w = 1	
ADC = Add with carry:				
Reg/memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w = 01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w = 1	
INC = Increment:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		
Register	0 1 0 0 0 reg			
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1			
SUB = Subtract:				
Reg/memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w = 01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w = 1	
SBB = Subtract with borrow:				
Reg/memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w = 01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w = 1	
DEC = Decrement:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		
CMP = Compare:				
Register/memory with register	0 0 1 1 1 0 1 w	mod reg r/m		
Register with register/memory	0 0 1 1 1 0 0 w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w = 01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w = 1	
AAS ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL Mulitiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL Integer multiply (signed):	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV Divide (unsigned):	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD ASCH adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW Convert byte to word	1 0 0 1 1 0 0 0			
CWD Convert word to double word	1 0 0 1 1 0 0 1			

INSTRUCTION SET SUMMARY (continued)

LOGIC

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
NOT Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m		
SHL/SAL Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m		
SHR Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m		
SAR Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m		
ROL Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m		
ROR Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m		
RCL Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m		
RCR Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m		
AND = And:				
Reg/memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w = 1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w = 1	
TEST = And function to flags, no result:				
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w = 1	
OR = Or:				
Reg/memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w = 1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w = 1	
XOR = Exclusive or:				
Reg/memory and register to either	0 0 1 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w = 1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w = 1	

STRING MANIPULATION:

REP = Repeat	1 1 1 1 0 0 1 z			
MOVS = Move byte/word	1 0 1 0 0 1 0 w			
CMPS = Compare byte/word	1 0 1 0 0 1 1 w			
SCAS = Scan byte/word	1 0 1 0 1 1 1 w			
LODS = Load byte/wd to AL/AX	1 0 1 0 1 1 0 w			
STOS = Store byte/wd from AL/A	1 0 1 0 1 0 1 w			

INSTRUCTION SET SUMMARY (continued)

CONTROL TRANSFER

CALL = Call	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high	
indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m		
Direct intersegment	1 0 0 1 1 0 1 0	offset-low	offset-high	
		seg-low	seg-high	
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m		
JMP = Unconditional jump:				
Direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high	
Direct within segment-short	1 1 1 0 1 0 1 1	disp		
Indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m		
Direct intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high	
		seg-low	seg-high	
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m		
RET = Return from CALL:				
Within segment	1 1 0 0 0 0 1 1			
Within segment adding immediate to SP	1 1 0 0 0 0 1 0	data-low	data-high	
Intersegment	1 1 0 0 1 0 1 1			
Intersegment adding immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high	
JE/JZ = Jump on equal/zero	0 1 1 1 0 1 0 0	disp		
JL/JNGE = Jump on less/not greater or equal	0 1 1 1 1 1 0 0	disp		
JLE/JNG = Jump on less or equal/not greater	0 1 1 1 1 1 1 0	disp		
JB/JNAE = Jump on below/not above or equal	0 1 1 1 0 0 1 0	disp		
JBE/JNA = Jump on below or equal/not above	0 1 1 1 0 1 1 0	disp		
JP/JPE = Jump on parity/parity even	0 1 1 1 1 0 1 0	disp		
JO = Jump on overflow	0 1 1 1 0 0 0 0	disp		
JS = Jump on sign	0 1 1 1 1 0 0 0	disp		
JNE/JNZ = Jump on not equal/not zero	0 1 1 1 0 1 0 1	disp		
JNL/JGE = Jump on not less/greater or equal	0 1 1 1 1 1 0 1	disp		
JNLE/JG = Jump on not less or equal/greater	0 1 1 1 1 1 1 1	disp		
JNB/JAE = Jump on not below/above or equal	0 1 1 1 0 0 1 1	disp		
JNBE/JA = Jump on not below or equal/above	0 1 1 1 0 1 1 1	disp		
JNP/JPO = Jump on not par/par odd	0 1 1 1 1 0 1 1	d isp		
JNO = Jump on not overflow	0 1 1 1 0 0 0 1	disp		
JNS = Jump on not sign	0 1 1 1 1 0 0 1	disp		
LOOP = Loop CX times	1 1 1 0 0 0 1 0	disp		
LOOPZ/LOOPE = Loop while zero/equal	1 1 1 0 0 0 0 1	disp		
LOOPNZ/LOOPNE = Loop while not zero/equal	1 1 1 0 0 0 0 0	disp		
JCXZ = Jump on CX zero	1 1 1 0 0 0 1 1	disp		

INSTRUCTION SET SUMMARY (continued)

CONTROL TRANSFER (continued)

INT = Interrupt	76543210	76543210	76543210	76543210
Type specified	1 1 0 0 1 1 0 1	type		
Type 3	1 1 0 0 1 1 0 0			
INTO = Interrupt on overflow	1 1 0 0 1 1 1 0			
IRET = Interrupt return	1 1 0 0 1 1 1 1			

PROCESSOR CONTROL

CLC = Clear carry	1 1 1 1 1 0 0 0	
CMC = Complement carry	1 1 1 1 0 1 0 1	
STC = Set carry	1 1 1 1 1 0 0 1	
CLD = Clear direction	1 1 1 1 1 1 0 0	
STD = Set direction	1 1 1 1 1 1 0 1	
CLI = Clear interrupt	1 1 1 1 1 0 1 0	
STI = Set interrupt	1 1 1 1 1 0 1 1	
HLT = Halt	1 1 1 1 0 1 0 0	
WAIT = Wait	1 0 0 1 1 0 1 1	
ESC = Processor Extension Escape	1 1 0 1 1 x x x	mod x x x r/m
LOCK = Bus lock prefix	1 1 1 1 0 0 0 0	

Footnotes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive.
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
w = 1 then word instruction; if w = 0 then byte instruction

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

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

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

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

SEGMENT OVERRIDE PREFIX

0	0	1	reg	1	1	0

REG is assigned according to the following table:

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

Instructions which reference the flag register files as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

80286

High-Performance Microprocessor with Memory Management and Protection

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

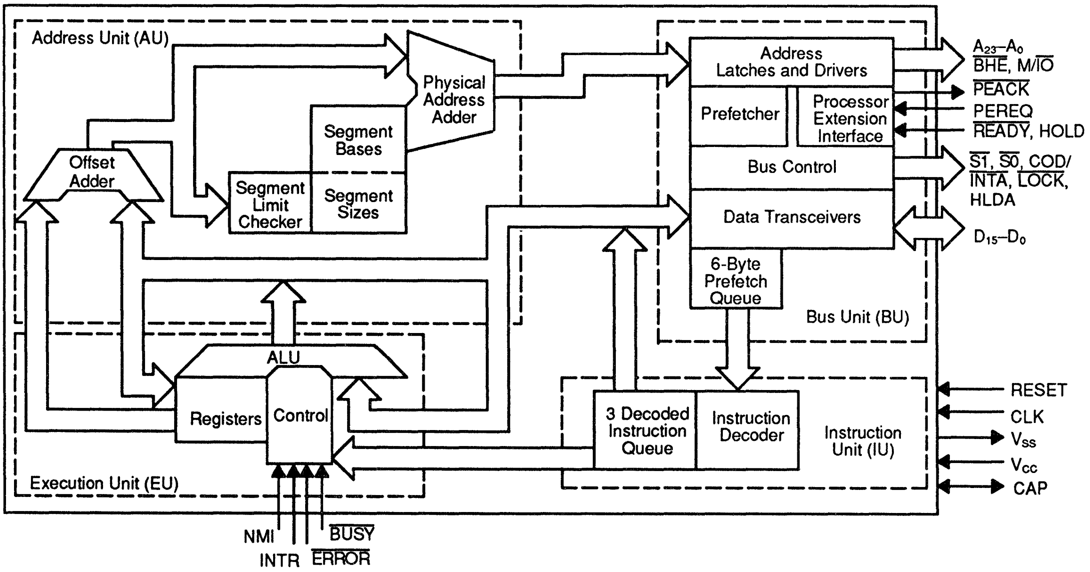
- **High-performance processor (up to 13.3 times iAPX 86 when using the 16-MHz 80286)**
- **Large address space**
  - –16 Mb physical
  - –1 Gb virtual memory per task
- **Integrated memory management, four-level memory protection and support for virtual memory and operating systems**
- **Two iAPX 86 upward-compatible operating modes**
  - –iAPX 86 real address mode
  - –Protected virtual address mode
- **High bandwidth bus interface (16 Mb/s)**
- **Range of clock rates**
  - – 8 MHz 80286–8
  - –10 MHz 80286–10
  - –12 MHz 80286–12
  - –16 MHz 80286–16

GENERAL DESCRIPTION

The 80286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multitasking systems. The 80286 has built-in memory protection that supports operating system and task isolation as well as program and data privacy within tasks. A 16-MHz 80286 provides up to 13.3 times greater throughput than the standard 5-MHz 8086. The 80286 includes memory management capabilities that map up to 2^{30} bytes (one gigabyte) of virtual address space per task into 2^{24} bytes (16 Mb) of physical memory.

The 80286 is upward-compatible with iAPX 86 and 88 software. Using iAPX 86 real address mode, the 80286 is object-code compatible with existing iAPX 86, 88 software.

BLOCK DIAGRAM



03552–1

Publication # 03552 Rev. G Amendment /0
Issue Date: December 1991

GENERAL DESCRIPTION (continued)

In protected virtual address mode, the 80286 is source-code compatible with iAPX 86, 88 software and may require upgrading to use virtual addresses supported by the 80286's integrated memory management and protection mechanism. Both modes operate at full 80286 performance and execute a superset of the iAPX 86 and 88 instructions.

The 80286 provides special operations to support the efficient implementation and execution of operating systems. For example, one instruction can end execution of one task, save its state, switch to a new task, load its state, and start execution of the new task. The 80286 also supports virtual memory systems by providing a segment-not-present exception and restartable instructions.

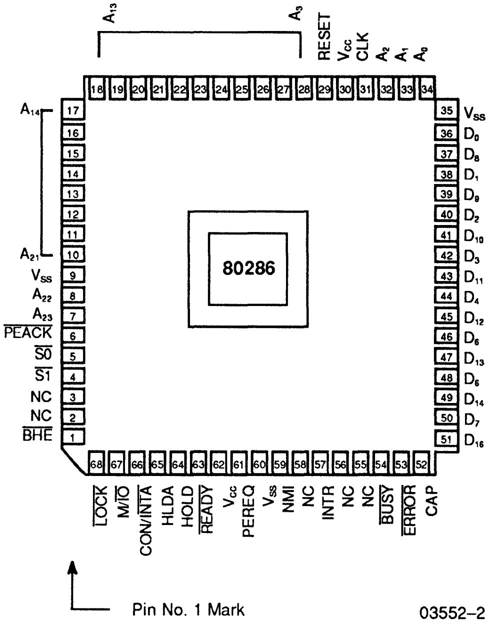
Related AMD Products

Part No.	Description
82284	Clock Driver
82C54	Programmable Interval Timer
Am9517A	DMA Controller

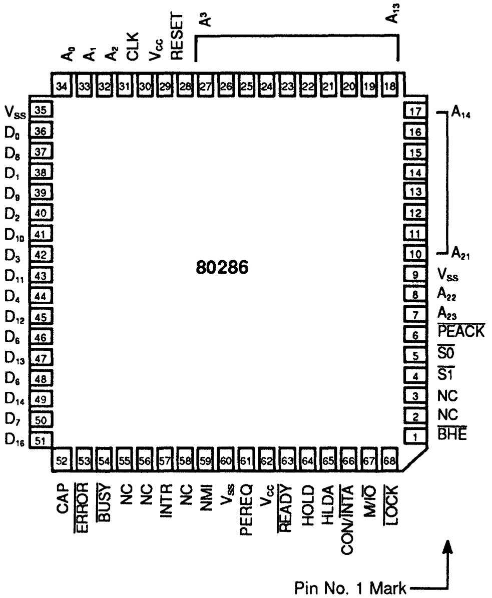
CONNECTION DIAGRAMS

LCC

Component Pad Views—As viewed from underside of component on the PC Board



PC Board Views—As viewed from the component side of the PC Board



There are no electrical connections on the bottom of this package

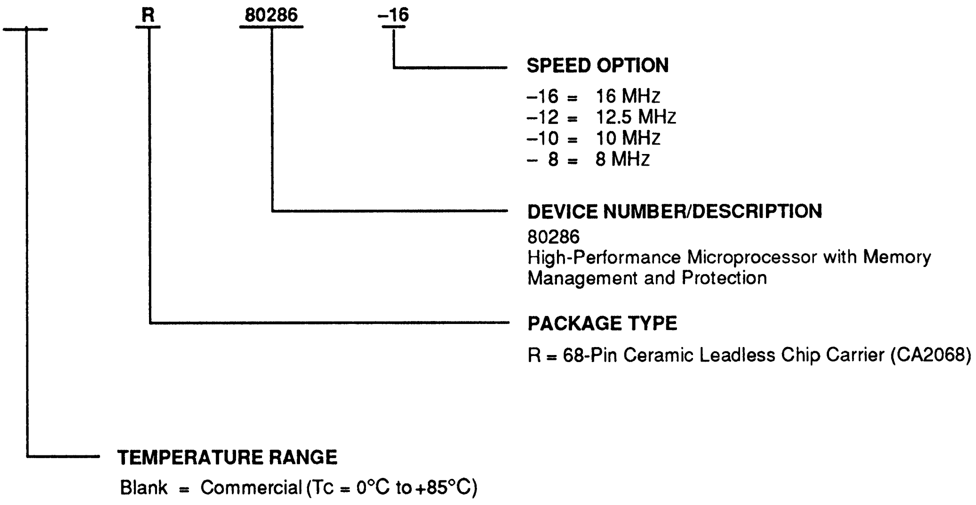
PIN DESIGNATIONS
(Sorted by pin number)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	$\overline{BHE}$	24	A_7	47	D_{13}
2	NC	25	A_6	48	D_6
3	NC	26	A_5	49	D_{14}
4	$\overline{S1}$	27	A_4	50	D_7
5	$\overline{S0}$	28	A_3	51	D_{15}
6	$\overline{PEACK}$	29	RESET	52	CAP
7	A_{23}	30	V_{CC}	53	$\overline{ERROR}$
8	A_{22}	31	CLK	54	$\overline{BUSY}$
9	V_{SS}	32	A_2	55	NC
10	A_{21}	33	A_1	56	NC
11	A_{20}	34	A_0	57	INTR
12	A_{19}	35	V_{SS}	58	NC
13	A_{18}	36	D_0	59	NMI
14	A_{17}	37	D_8	60	V_{SS}
15	A_{16}	38	D_1	61	PEREQ
16	A_{15}	39	D_9	62	V_{CC}
17	A_{14}	40	D_2	63	$\overline{READY}$
18	A_{13}	41	D_{10}	64	HOLD
19	A_{12}	42	D_3	65	HLDA
20	A_{11}	43	D_{11}	66	COD/$\overline{INTA}$
21	A_{10}	44	D_4	67	M/$\overline{IO}$
22	A_9	45	D_{12}	68	$\overline{LOCK}$
23	A_8	46	D_5		

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



SPEED OPTION

−16 = 16 MHz
−12 = 12.5 MHz
−10 = 10 MHz
− 8 = 8 MHz

DEVICE NUMBER/DESCRIPTION

80286
High-Performance Microprocessor with Memory Management and Protection

PACKAGE TYPE

R = 68-Pin Ceramic Leadless Chip Carrier (CA2068)

TEMPERATURE RANGE

Blank = Commercial (Tc = 0°C to +85°C)

Valid Combinations	
R	80286–8
	80286–10
	80286–12
	80286–16

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

PIN DESCRIPTION

CLK

System Clock (Input; Active High)

System Clock provides the fundamental timing for 80286 systems. It is divided by two inside the 80286 to generate the processor clock. The internal divide-by-two circuitry can be synchronized to an external clock generator by Low-to-High transition on the RESET input.

D_0–D_{15}

Data Bus (Input/Output; Active High)

Data Bus inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles. The data bus is active High and floats to three-state OFF during bus hold acknowledge.

A_{23}–A_0

Address Bus (Output; Active High)

Address Bus outputs physical memory and I/O port addresses. A_0 is Low when data is to be transferred on pins D_7–D_0. A_{23}–A_{16} are Low during I/O transfers. The address bus is active High and floats to three-state OFF during bus hold acknowledge.

$\overline{\text{BHE}}$

Bus High Enable (Output; Active Low)

Bus High Enable indicates transfer of data on the upper byte of the data bus D_{15}–D_8. Eight-bit oriented devices assigned to the upper byte of the data bus would normally use $\overline{\text{BHE}}$ to condition chip select functions. $\overline{\text{BHE}}$ is active Low and floats to three-state OFF during bus hold acknowledge.

$\overline{\text{BHE}}$ and A_0 Encodings

$\overline{\text{BHE}}$ Value	A_0 Value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus (D_{15-8})
1	0	Byte transfer on lower half of data bus (D_{7-0})
1	1	Reserved

$\overline{\text{S1}}$, $\overline{\text{S0}}$

Bus Cycle Status (Output; Active Low)

Bus Cycle Status indicates initiation of a bus cycle and, along with M/$\overline{\text{IO}}$ and COD/$\overline{\text{INTA}}$, defines the type of bus cycle. The bus is in a T_S state whenever one or both are Low. $\overline{\text{S1}}$ and $\overline{\text{S0}}$ are active Low and float to three-state OFF during bus hold acknowledge.

80286 Bus Cycle Status Definition

COD/ $\overline{\text{INTA}}$	M/$\overline{\text{IO}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$	Bus Cycle Status Definition
0 (Low)	0	0	0	Interrupt acknowledge
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	None; not a status cycle
0	1	0	0	If $A_1 = 1$ then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (High)	0	0	0	Reserved
1	0	0	1	I/O Read
1	0	1	0	I/O Write
1	0	1	1	None; not a status cycle
1	1	0	0	Reserved
1	1	0	1	Memory instruction read
1	1	1	0	Reserved
1	1	1	1	None; not a status cycle

M/$\overline{\text{IO}}$

Memory/$\overline{\text{IO}}$ Select (Output)

Memory/$\overline{\text{IO}}$ Select distinguishes memory access from I/O access. If High during T_S, a memory cycle or a halt/shutdown cycle is in progress. If Low, an I/O cycle or an interrupt acknowledge cycle is in progress. M/$\overline{\text{IO}}$ floats to three-state OFF during bus hold acknowledge.

COD/$\overline{\text{INTA}}$

Code/Interrupt Acknowledge (Output)

Code/Interrupt Acknowledge distinguishes instruction fetch cycles from memory data read cycles. Also distinguishes interrupt acknowledge cycles from I/O cycles. COD/$\overline{\text{INTA}}$ floats to three-state OFF during bus hold acknowledge.

$\overline{\text{LOCK}}$

Bus Lock (Output; Active Low)

Bus Lock indicates that other system bus masters are not to gain control of the system bus following the current bus cycle. The $\overline{\text{LOCK}}$ signal may be activated explicitly by the LOCK instruction prefix or automatically by 80286 hardware during memory XCHG instructions, interrupt acknowledge, or descriptor table access. $\overline{\text{LOCK}}$ is active Low and floats to three-state OFF during hold acknowledge.

PIN DESCRIPTION (continued)

$\overline{\text{READY}}$

Bus Ready (Input; Active Low)

Bus Ready terminates a bus cycle. Bus cycles are extended without limit until terminated by $\overline{\text{READY}}$ Low. $\overline{\text{READY}}$ is an active Low synchronous input requiring set-up and hold times relative to the system clock be met for correct operation. $\overline{\text{READY}}$ is ignored during bus hold acknowledge.

HOLD, HLDA

Bus Hold Request and Hold Acknowledge (Input/Output; Active High)

Bus Hold Request and Hold Acknowledge control ownership of the 80286 local bus. The HOLD input allows another local bus master to request control of the local bus. When control is granted, the 80286 will float its bus drivers to three-state OFF and then active HLDA, thus entering the bus hold acknowledge condition. The local bus will remain granted to the requesting master until HOLD becomes inactive which results in the 80286 deactivating HLDA and regaining control of the local bus. This terminates the bus hold acknowledge condition. HOLD may be asynchronous to the system clock. These signals are active High.

INTR

Interrupt Request (Input; Active High)

Interrupt Request requests the 80286 to suspend its current program execution and service a pending external request. Interrupt requests are masked whenever the interrupt enable bit in the flag word is cleared. When the 80286 responds to an interrupt request, it performs two interrupt acknowledge bus cycles to read an 8-bit interrupt vector that identifies the source of the interrupt. To assure program interruption, INTR must remain active until the first interrupt acknowledge cycle is completed. INTR is sampled at the beginning of each processor cycle and must be active High at least two processor cycles before the current instruction ends in order to interrupt before the next instruction. INTR is level sensitive, active High, and may be asynchronous to the system clock.

NMI

Non-Maskable Interrupt Request (Input; Active High)

Non-maskable Interrupt Request interrupts the 80286 with an internally supplied vector value of 2. No interrupt acknowledge cycles are performed. The interrupt enable bit in the 80286 flag word does not affect this input. The NMI input is active High, may be asynchronous to the system clock, and is edge triggered after internal synchronization. For proper recognition, the input must have been previously Low for at least four system clock cycles and remain High for at least four system clock cycles.

PEREQ, $\overline{\text{PEACK}}$

Processor Extension Operand Request and Acknowledge (Input/Output)

Processor Extension Operand Request and Acknowledge extended the memory management and protection capabilities of the 80286 to processor extensions. The PEREQ input requests the 80286 to perform a data operand transfer for a processor extension. The $\overline{\text{PEACK}}$ output signals the processor extension when the requested operand is being transferred. PEREQ is active High and may be asynchronous to the system clock. $\overline{\text{PEACK}}$ is active Low.

$\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$

Processor Extension Busy and Error (Input/Input, Active Low)

Processor Extension Busy and Error indicate the operating condition of a processor extension to the 80286. An active $\overline{\text{BUSY}}$ input stops 80286 program execution on WAIT and some ESC instructions until $\overline{\text{BUSY}}$ becomes inactive (High). The 80286 may be interrupted while waiting for $\overline{\text{BUSY}}$ to become inactive. An active $\overline{\text{ERROR}}$ input causes the 80286 to perform a processor extension interrupt when executing WAIT or some ESC instructions. These inputs are active Low and may be asynchronous to the system clock.

RESET

System Reset (Input; Active High)

System Reset clears the internal logic of the 80286 and is active High. The 80286 may be reinitialized at any time with a Low-to-High transition on RESET which remains active for more than 16 system clock cycles. During RESET active, the output pins of the 80286 enter the state shown below.

80286 Pin State During Reset

Pin Value	Pin Names
1 (High)	$\overline{\text{S0}}$, $\overline{\text{S1}}$, $\overline{\text{PEACK}}$, A_{23}–A_0, $\overline{\text{BHE}}$, $\overline{\text{LOCK}}$
0 (Low)	M/$\overline{\text{IO}}$, COD/$\overline{\text{INTA}}$, HLDA
Three-state OFF	D_{15}–D_0

Operation of the 80286 begins after a High-to-Low transition on RESET. The High-to-Low transition of RESET must be synchronous to the system clock. Approximately 50 system clock cycles are required by the 80286 for internal initializations before the first bus cycle to fetch code from the power-on execution address is performed.

PIN DESCRIPTION (continued)

A Low-to-High transition of RESET synchronous to the system clock will begin a new processor cycle at the next High-to-Low transition of the system clock. The Low-to-High transition of RESET may be asynchronous to the system clock; however, in this case it cannot be predetermined which phase of the processor clock will occur during the next system period. Synchronous Low-to-High transitions of RESET are only required for systems where the processor clock must be phase synchronous to another clock.

V_{SS}
System Ground (Input)

System Ground: 0 V.

V_{CC}
System Power (Input)

System Power: +5 V power supply.

CAP
Substrate Filter Capacitor (Input; Active High)

A 0.047 µF ±20% 12 V capacitor must be connected between this pin and ground. This capacitor filters the output of the internal substrate bias generator. A maximum DC leakage current of 1 µA is allowed through the capacitor.

For correct operation of the 80286, the substrate bias generator must charge this capacitor to its operating voltage. The capacitor charge-up time is 5 ms (Max) after V_{CC} and CLK reach their specified AC and DC parameters. RESET may be applied to prevent spurious activity by the CPU during this time. After this time, the 80286 processor clock can be phase synchronized to another clock by pulsing RESET Low synchronous to the system clock.

FUNCTIONAL DESCRIPTION

Introduction

The 80286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multi-tasking systems. Depending on the application, the 80286's performance is up to 13.3 times faster than the standard 5-MHz 8086's, while providing complete upward software compatibility with AMD's iAPX 86, 88, and 186 family of CPUs.

The 80286 operates in two modes: iAPX 86 real address mode and protected virtual address mode. Both modes execute a superset of the iAPX 86 and 88 instruction set.

In iAPX 86 real address mode programs use real addresses with up to one megabyte of address space. Programs use virtual addresses in protected virtual address mode, also called protected mode. In protected mode, the 80286 CPU automatically maps 1 gigabyte of virtual addresses per task into a 16-megabyte real address space. This mode also provides memory protection to isolate the operating system and ensure privacy of each task's programs and data. Both modes provide the same base instruction set, registers, and addressing modes.

The following pages describe first, the base 80286 architecture common to both modes; second, iAPX 86 real address mode; and third, protected mode.

80286 Base Architecture

The iAPX 86, 88, 186, and 286 CPU family all contain the same basic set of registers, instructions, and addressing modes. The 80286 processor is upward-compatible with the 8086, 8088, and 80186 CPUs.

Register Set

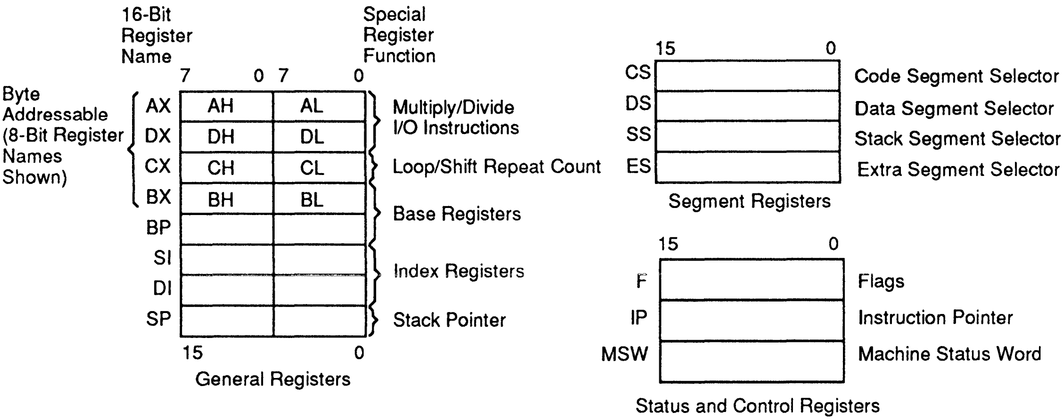
The 80286 base architecture has fifteen registers as shown in Figure 1. These registers are grouped into the following four categories:

General Registers: Eight 16-bit general purpose registers used to contain arithmetic and logical operands. Four of these (AX, BX, CX, and DX) can be used either in their entirety as 16-bit words or split into pairs of separate 8-bit registers.

Segment Registers: Four 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data. (For usage, refer to Memory Organization.)

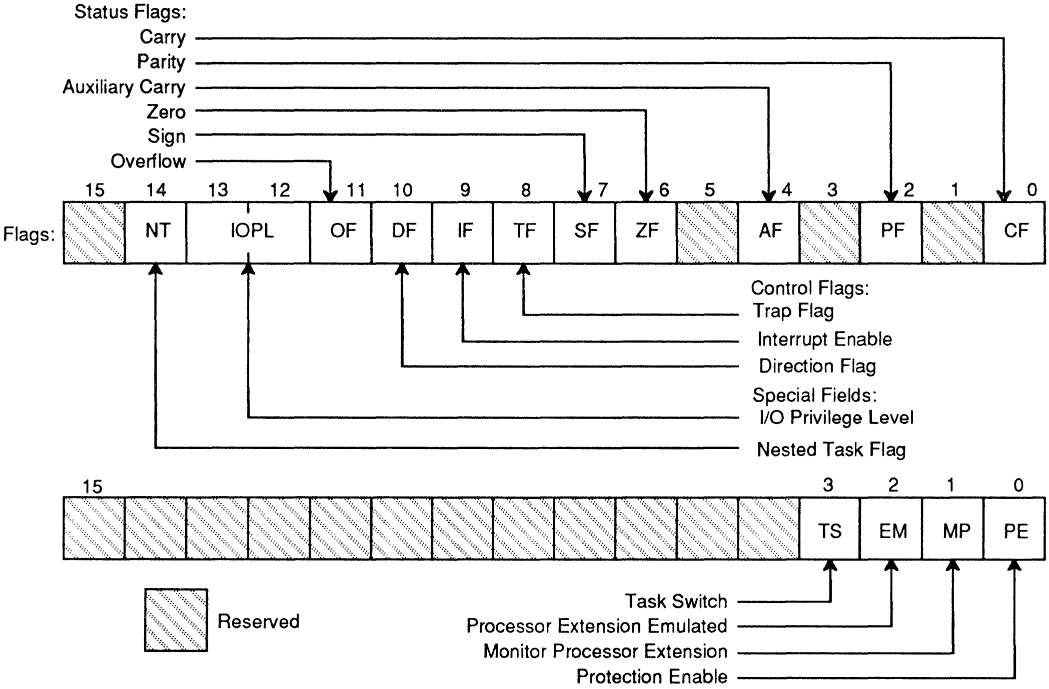
Base and Index Registers: Four of the general purpose registers may also be used to determine offset addresses of operands in memory. These registers may contain base addresses or indexes to particular locations within a segment. The addressing mode determines the specific registers used for operand address calculations.

Status and Control Registers: Three 16-bit special purpose registers record or control certain aspects of the 80286 processor state. These include the Instruction Pointer, which contains the offset address of the next sequential instruction to be executed.



03552–7

Figure 1. Register Set



03552–8

Figure 2. Status and Control Register Bit Functions

Flags Word Description

The Flags Word (Flags) records specific characteristics of the result of logical and arithmetic instructions (bits 0, 2, 4, 6, 7, and 11) and controls the operation of the 80286 within a given operating mode (bits 8 and 9). Flags is a 16-bit register. The function of the flag bits is given in Table 1.

Table 1. Flags Word Bit Functions

Bit Position	Name	Function
0	CF	Carry Flag—Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity Flag—Set if low-order 8 bits of result contain an even number of 1 bits; cleared otherwise
4	AF	Set on carry-from or borrow-to the low-order four bits of AL; cleared otherwise
6	ZF	Zero Flag—Set if result is zero; cleared otherwise
7	SF	Sign Flag—Set equal to high-order bit of result (0 if positive, 1 if negative)
11	OF	Overflow Flag—Set if result is a too-large positive number or a too-small negative number (excluding sign-bit) to fit in destination operand; cleared otherwise
8	TF	Single Step Flag—Once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt
9	IF	Interrupt-Enable Flag—When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location
10	DF	Direction Flag—Causes string instructions to auto-decrement the appropriate index registers when set. Clearing DF causes auto increment.

Instruction Set

The instruction set is divided into seven categories: data transfer, arithmetic, shift/rotate/logical, string manipulation, program transfer, high-level instructions, and processor control. These categories are summarized in Figures 3–9.

An 80286 instruction can reference zero, one, or two operands where an operand resides in a register, in the instruction itself, or in memory. Zero-operand instructions (e.g., NOP and HLT) are usually one byte long. One-operand instructions (e.g., INC and DEC) are usually two bytes long, but some are encoded in only one byte. One-operand instructions may reference a register or memory location. Two-operand instructions permit the following six types of instruction operations.

Register to Register
Memory to Register
Immediate to Register
Memory to Memory
Register to Memory
Immediate to Memory

Two-operand instructions (e.g., MOV and ADD) are usually three to six bytes long. Memory to memory operations are provided by a special class of string instructions requiring one to three bytes. For detailed instruction formats and encodings, refer to the instruction set summary at the end of this document.

	General Purpose
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push all registers on stack
POPA	Pop all registers from stack
XCHG	Exchange byte or word
XLAT	Translate byte
	Input/Output
IN	Input byte or word
OUT	Output byte or word
	Address Object
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
	Flag Transfer
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack

Figure 3. Data Transfer Instructions

Addition	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition

Subtraction	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction

Multiplication	
MUL	Multiply byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply

Division	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to double word

Figure 4. Arithmetic Instructions

MOVS	Move byte or word string
INS	Input bytes or word string
OUTS	Output bytes or word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string
LODS	Load byte or word string
STOS	Store byte or word string
REP	Repeat
REPE/REPZ	Repeat while equal/zero
REPNE/REPNZ	Repeat while not equal/not zero

Figure 5. String Instructions

Logicals	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word

Shifts	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word

Rotates	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

Figure 6. Shift/Rotate/Logical

Conditional Transfers	
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if sign

Unconditional Transfers	
CALL	Call procedure
RET	Return from procedure
JMP	Jump

Iteration Controls	
LOOP	Loop
LOOPE/LOOPZ	Loop if equal/zero
LOOPNE/LOOPNZ	Loop if not equal/not zero
JCXZ	Jump if register CX = 0

Interrupts	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Interrupt return

Figure 7. Program Transfer Instructions

Flag Operations

STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag

External Synchronization

HLT	Halt until interrupt or reset
WAIT	Wait for $\overline{\text{BUSY}}$ not active
ESC	Escape to extension processor
LOCK	Lock bus during next instruction

No Operation

NOP	No operation

Execution Environment Control

LMSW	Load machine status word
SMSW	Store machine status word

Figure 8. Processor Control Instructions

ENTER	Format stack for procedure entry
LEAVE	Restore stack for procedure exit
BOUND	Detects values outside prescribed range

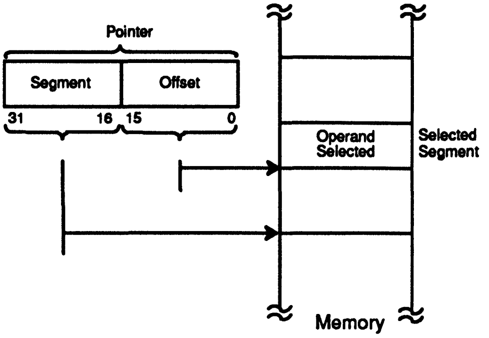
Figure 9. High-Level Instructions

Memory Organization

Memory is organized as sets of variable length segments. Each segment is a linear contiguous sequence of up to 64K(2^{16}) 8-bit bytes. Memory is addressed using a two-component address (a pointer) that consists of a 16-bit segment selector and a 16-bit offset. The segment selector indicates the desired segment in memory. The offset component indicates the desired byte address within the segment.

Most instructions need not explicitly specify which segment register is used. The correct segment register is automatically chosen according to the rules of Table 2. These rules follow the way programs are written (see Figure 11) as independent modules that require areas for code and data, a stack, and access to external data areas.

All instructions that address operands in memory must specify the segment and the offset. For speed and compact instruction encoding, segment selectors are usually stored in the high speed segment registers. An instruction need specify only the desired segment register and an offset to address a memory operand.



03552–9

Figure 10. Two-Component Address

Table 2. Segment Register Selection Rules

Memory Reference Needed	Segment Register Used	Implicit Segment Selection Rule
Instructions	Code (CS)	Automatic with instruction prefetch
Stack	Stack (SS)	All stack pushes and pops. Any memory reference which uses BP as a base register.
Local Data	Data (DS)	All data references except when relative to stack or string destination.
External (Global) Data	Extra (ES)	Alternate data segment and destination of string operation.

Special segment override instruction prefixes allow the implicit segment register selection rules to be overridden for special cases. The stack, data, and extra segments may coincide for simple programs. To access operands that do not reside in one of the four immediately available segments, either a full 32-bit pointer can be used or a new segment selector must be loaded.

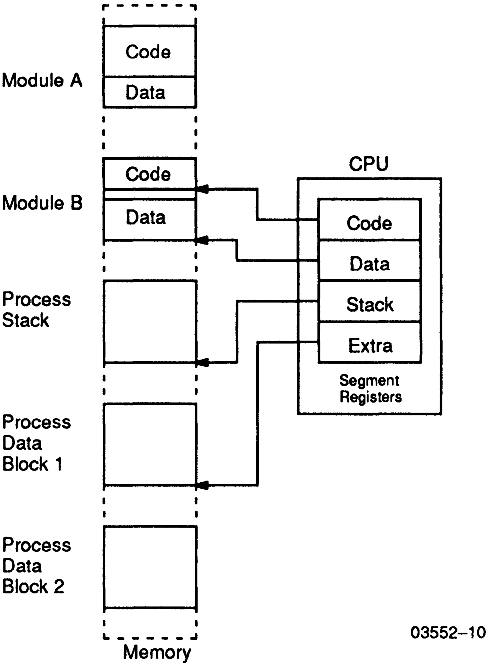


Figure 11. Segmented Memory Helps Structure Software

Addressing Modes

The 80286 provides a total of eight addressing modes for instructions to specify operands. Two addressing modes are provided for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8- or 16-bit general registers.

Immediate Operand Mode: The operand is included in the instruction.

Six modes are provided to specify the location of an operand in a memory segment. A memory operand address consists of two 16-bit components: segment selector and offset. The segment selector is supplied by a segment register either implicitly chosen by the addressing mode or explicitly chosen by a segment override prefix. The offset is calculated by summing any combination of the following three address elements:

the displacement (an 8- or 16-bit immediate value contained in the instruction)

the base (contents of either the BX or BP base registers)

the index (contents of either the SI or DI index registers)

Any carry out from the 16-bit addition is ignored. Eight-bit displacements are sign extended to 16-bit values.

Combinations of these three address elements define the six memory addressing modes here described:

Direct Mode: The operand's offset is contained in the instruction as an 8- or 16-bit displacement element.

Register Indirect Mode: The operand's offset is in one of the registers SI, DI, BX, or BP.

Based Mode: The operand's offset is the sum of an 8- or 16-bit displacement and the contents of a base register (BX or BP).

Indexed Mode: The operand's offset is the sum of an 8- or 16-bit displacement and the contents of an index register (SI or DI).

Based Indexed Mode: The operand's offset is the sum of the contents of a base register and an index register.

Based Indexed Mode with Displacement: The operand's offset is the sum of a base register's contents, an index register's contents, and an 8- or 16-bit displacement.

Data Types

The 80286 directly supports the following data types:

Integer:	A signed binary numeric value contained in an 8-bit byte or a 16-bit word. All operations assume a two's complement representation. Signed 32- and 64-bit integers are supported using the 80287 Numeric Data Processor.
Ordinal:	An unsigned binary numeric value contained in an 8-bit byte or 16-bit word.
Pointer:	A 32-bit quantity, composed of a segment selector component and an offset component. Each component is a 16-bit word.
String:	A contiguous sequence of bytes or words. A string may contain from 1 byte to 64K bytes.
ASCII:	A byte representation of alphanumeric and control characters using the ASCII standard of character representation.
BCD:	A byte (unpacked) representation of the decimal digits 0–9.
Packed BCD:	A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble of the byte.
Floating Point:	A signed 32-, 64-, or 80-bit real number representation. (Floating point operands are supported using the iAPX 287 Numeric Processor configuration.)

Figure 12 graphically represents the data types supported by the 80286.

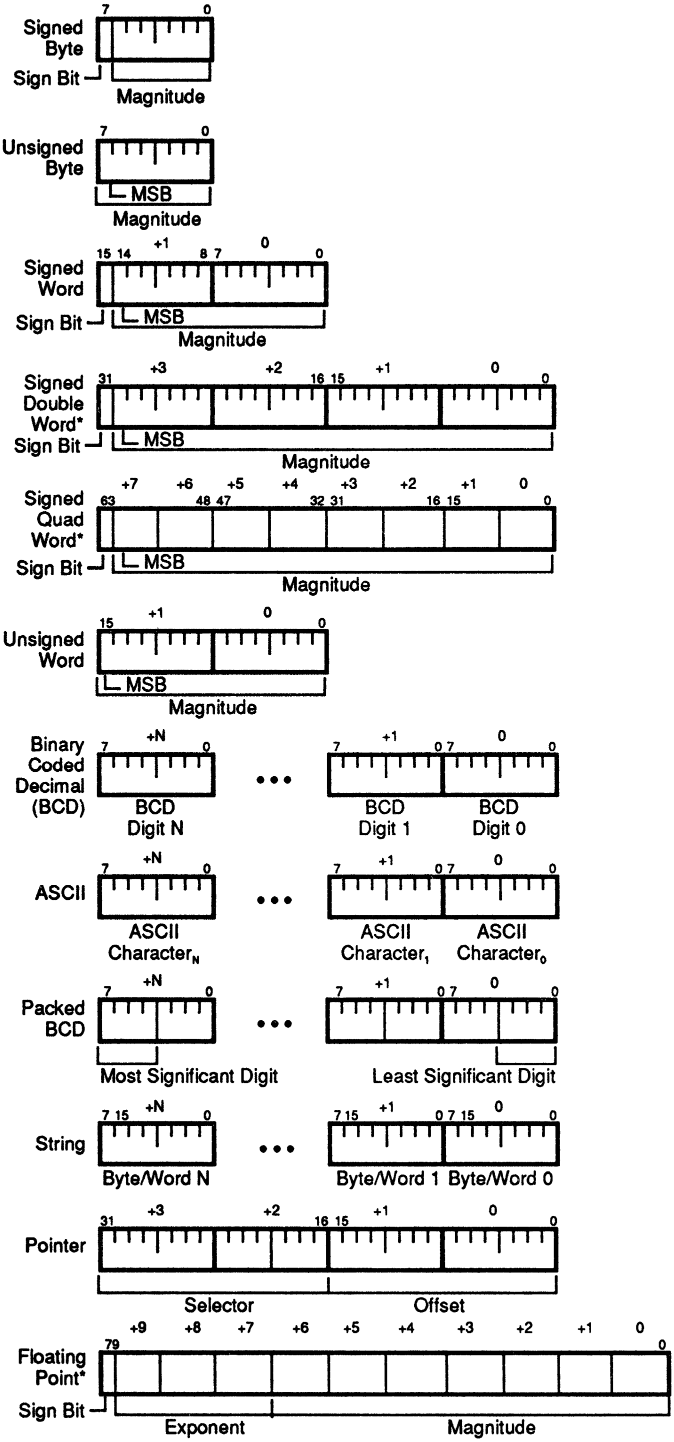


Figure 12. 80286 Supported Data Types

*Supported by iAPX 286/287 Numeric Data Processor Configuration

03552–11

I/O Space

The I/O space consists of 64K 8-bit or 32K 16-bit ports. I/O instructions address the I/O space with either an 8-bit port address, specified in the instruction, or a 16-bit port address in the DX register. Eight-bit port addresses are zero extended such that A_{15}–A_8 are Low. I/O port addresses 00F8(H) through 00FF(H) are reserved.

Interrupts

An interrupt transfers execution to a new program location. The old program address (CS:IP) and machine state (Flags) are saved on the stack to allow resumption of the interrupted program. Interrupts fall into three classes: hardware initiated, INT instructions, and instruction exceptions. Hardware initiated interrupts occur in response to an external input and are classified as non-maskable or maskable. Programs may cause an interrupt with an INT instruction. Instruction exceptions occur when an unusual condition, which prevents further instruction processing, is detected while attempting to execute an instruction. The return address from an exception will always point at the instruction causing the exception and include any leading instruction prefixes.

A table containing up to 256 pointers defines the proper interrupt service routine for each interrupt. Interrupts 0–31, some of which are used for instruction exceptions, are reserved. For each interrupt, an 8-bit vector must be supplied to the 80286 which identifies the appropriate table entry. Exceptions supply the interrupt vector internally. INT instructions contain or imply the vector and allow access to all 256 interrupts. Maskable hardware-initiated interrupts supply the 8-bit vector to the CPU during an interrupt acknowledge bus sequence. Non-maskable hardware interrupts use a predefined internally supplied vector.

Maskable Interrupt (INTR)

The 80286 provides a maskable hardware interrupt request pin, INTR. Software enables this input by setting the interrupt flag bit (IF) in the flag word. All 224 user-defined interrupt sources can share this input, yet they can retain separate interrupt handlers. An 8-bit vector read by the CPU during the interrupt acknowledge sequence (discussed in the System Interface section) identifies the source of the interrupt.

Further maskable interrupts are disabled while servicing an interrupt by resetting the IF but as part of the response to an interrupt or exception. The saved flag word will reflect the enable status of the processor prior to the interrupt. Until the flag word is restored to the flag register, the interrupt flag will be zero unless specifically set. The interrupt return instruction includes restoring the flag word, thereby restoring the original status of IF.

Non-Maskable Interrupt Request (NMI)

A non-maskable interrupt input (NMI) is also provided. NMI has higher priority than INTR. A typical use of NMI would be to activate a power failure routine. The activation of this input causes an interrupt with an internally supplied vector value of 2. No external interrupt acknowledge sequence is performed.

While executing the NMI servicing procedure, the 80286 will not service further NMI requests, INTR requests, or the processor extension segment overrun interrupt until an interrupt return (IRET) instruction is executed or the CPU is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. IF is cleared at the beginning of an NMI interrupt to inhibit INTR interrupts.

Single Step Interrupt

The 80286 has an internal interrupt that allows programs to execute one instruction at a time. It is called the single step interrupt and is controlled by the single step flag bit (TF) in the flag word. Once this bit is set, an internal single step interrupt will occur after the next instruction has been executed. The interrupt clears the TF bit and uses an internally supplied vector of 1. The IRET instruction is used to set the TF bit and transfer control to the next instruction to be single stepped.

Table 3. Interrupt Vector Assignments

Function	Interrupt Number	Related Instructions	Return Address Before Instruction Causing Exception?
Divide error exception	0	DIV, IDIV	Yes
Single step interrupt	1	All	
NMI interrupt	2	All	
Breakpoint interrupt	3	INT	
INTO detected overflow exception	4	INTO	No
BOUND range exceeded exception	5	BOUND	Yes
Invalid opcode exception	6	Any undefined op-code	Yes
Processor extension not available exception	7	ESC or WAIT	Yes
Reserved	8–15		
Processor extension error input	16	ESC or WAIT	
Reserved	17–31		
User-defined	32–255		

Interrupt Priorities

When simultaneous interrupt requests occur, they are processed in a fixed order as shown in Table 4. Interrupt processing involves saving the flags, return address, and setting CS:IP to point at the first instruction of the interrupt handler. If other interrupts remain enabled, they are processed before the first instruction of the current interrupt handler is executed. The last interrupt processed is therefore the first one serviced.

Table 4. Interrupt Processing Order

Order	Interrupt
1	INT instruction or exception
2	Single step
3	NMI
4	Processor extension segment overrun
5	INTR

Initialization and Processor Reset

Processor initialization or start up is accomplished by driving the RESET input pin High. RESET forces the 80286 to terminate all execution and local bus activity. No instruction or bus activity will occur as long as RESET is active. After RESET becomes inactive and an internal processing interval elapses, the 80286 begins execution in real address mode with the instruction at physical location FFFFF0(H). RESET also sets some registers to predefined values as shown in Table 5.

Table 5. 80286 Initial Register State after RESET

Flag word	0002(H)
Machine Status Word	FFF0(H)
Instruction pointer	FFF0(H)
Code segment	F000(H)
Data segment	0000(H)
Extra segment	0000(H)
Stack segment	0000(H)

Machine Status Word Description

The machine status word (MSW) records when a task switch takes place and controls the operating mode of the 80286. It is a 16-bit register of which the lower four bits are used. One bit places the CPU into protected mode, while the other three bits, as shown in Table 6, control the processor extension interface. After RESET, this register contains FFF0(H) which places the 80286 in iAPX 86 real address mode.

The LMSW and SMSW instructions can load and store the MSW in real address mode. The recommended use of TS, EM, and MP is shown in Table 7.

Table 6. MSW Bit Functions

Bit Position	Name	Function
0	PE	Protected mode Enable places the 80286 into protected mode and cannot be cleared except by RESET.
1	MP	Monitor Processor extension allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	Emulate processor extension causes a processor extension not present exception (number 7) on ESC instructions to allow emulating a processor extension.
3	TS	Task Switched indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task.

Table 7. Recommended MSW Encodings For Processor Extension Control

TS	MP	EM	Recommended Use	Instructions Causing Exception
0	0	0	iAPX 86 real address mode only. Initial encoding after RESET. 80286 operation is identical to iAPX 86, 88.	None
0	0	1	No processor extension is available. Software will emulate its function.	ESC
1	0	1	No processor extension is available. Software will emulate its function. The current processor extension context may belong to another task.	ESC
0	1	0	A processor extension exists.	None
1	1	0	A processor extension exists. The current processor extension context may belong to another task. The exception on WAIT allows software to test for an error pending from a previous processor extension operation.	ESC or WAIT

Halt

The HLT instruction stops program execution and prevents the CPU from using the local bus until restarted. Either NMI, INTR with IF = 1, or RESET will force the 80286 out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

iAPX 286 Real Address Mode

The 80286 executes a fully upward-compatible superset of the 8086 instruction set in real address mode. In real address mode the 80286 is object code compatible with 8086 and 8088 software. The real address mode architecture (registers and addressing modes) is exactly as described in the 80286 Base Architecture section.

Memory Size

Physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A_0 through A_{19} and $\overline{BHE}$. A_{20} through A_{23} are ignored.

Memory Addressing

In real address mode the processor generates 20-bit physical addresses directly from a 20-bit segment base address and a 16-bit offset.

The selector portion of a pointer is interpreted as the upper 16 bits of a 20-bit segment address. The lower four bits of the 20-bit segment address are always zero. Segment addresses, therefore, begin on multiples of 16 bytes. See Figure 13 for a graphic representation of address formation.

All segments in real address mode are 64 kbytes in size and may be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment (e.g., a word with its low order byte at offset FFFF(H) and its high order byte at offset 0000(H)). If, in real address mode, the information contained in a segment does not use the full 64 kbytes, the unused end of the segment may be overlaid by another segment to reduce physical memory requirements.

Reserved Memory Locations

The 80286 reserves two fixed areas of memory in real address mode (see Figure 1): system initialization area and interrupt table area. Locations from addresses FFFF0(H) through FFFFF(H) are reserved for system initialization. Initial execution begins at location FFFF0 (H). Locations 00000(H) through 003FF(H) are reserved for interrupt vectors.

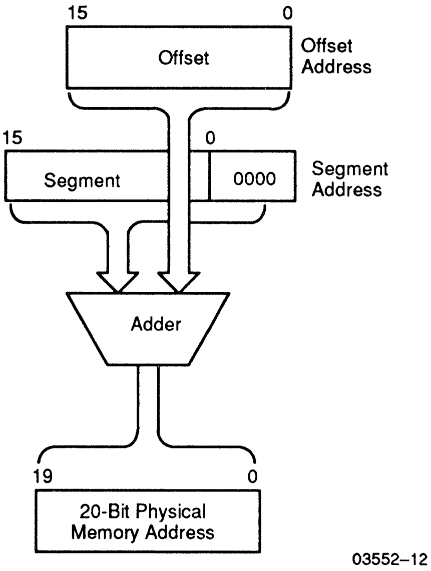
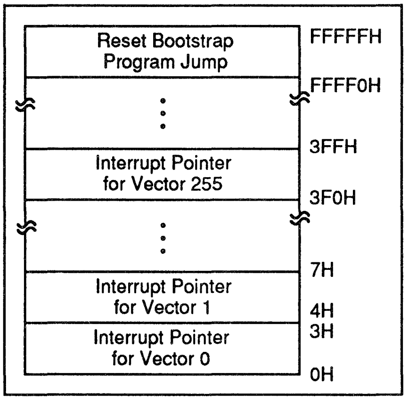


Figure 13. iAPX 86 Real Address Mode Address Calculation


Reset Bootstrap Program Jump
FFFFFH
FFFF0H
3FFH
Interrupt Pointer for Vector 255
3F0H
7H
Interrupt Pointer for Vector 1
4H
3H
Interrupt Pointer for Vector 0
0H

03552–13

Figure 14. iAPX 86 Real Address Mode Initially Reserved Memory Locations

Table 8. Real Address Mode Addressing Interrupts

Function	Interrupt Number	Related Instructions	Return Address Before Instruction?
Interrupt table limit too small exception	8	INT vector is not within table limit	Yes
Processor extension segment overrun interrupt	9	ESC with memory operand extending beyond offset FFFF(H)	No
Segment overrun exception	13	Word memory reference with offset = FFFF(H) or an attempt to execute past the end of a segment	Yes

Interrupts

Table 8 shows the interrupt vectors reserved for exceptions and interrupts which indicate an addressing error. The exceptions leave the CPU in the state existing before attempting to execute the failing instruction (except for PUSH, POP, PUSHA, or POPA). Refer to the next section on protected mode initialization for a discussion on exception 8.

Protected Mode Initialization

To prepare the 80286 for protected mode, the LIDT instruction is used to load the 24-bit interrupt table base and 16-bit limit for the protected mode interrupt table. This instruction can also set a base and limit for the interrupt vector table in real address mode. After reset, the interrupt table base is initialized to 000000(H) and its size set to 03FF(H). These values are compatible with iAPX 86, 88 software. LIDT should only be executed in preparation for the protected mode.

Shutdown

Shutdown occurs when a severe error is detected that prevents further instruction processing by the CPU. Shutdown and halt are externally signaled via a halt bus operation. They can be distinguished by A_1 High for halt and A_1 Low for shutdown. In real address mode, shutdown can occur under two conditions:

- Exceptions 8 or 13 happen and the IDT limit does not include the interrupt vector.
- A CALL, INT, or POP instruction attempts to wrap around the stack segment when SP is not even.

An NMI input can bring the CPU out of shutdown if the IDT limit is at least 000F(H) and SP is greater than 0005(H); otherwise, shutdown can only be exited via the RESET input.

Protected Virtual Address Mode

The 80286 executes a fully upward-compatible superset of the 8086 instruction set in protected virtual address mode (protected mode). Protected mode also provides memory management and protection mechanisms and associated instructions.

The 80286 enters protected virtual address mode from real address mode by setting the PE (Protection Enable) bit of the machine status word with the Load Machine Status Word (LMSW) instruction. Protected mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

All registers, instructions, and addressing modes described in the 80286 Base Architecture section remain the same. Programs for the iAPX 86, 88, 186, and real address mode 80286 can be run in protected mode; however, embedded constants for segment selectors are different.

Memory Size

The protected mode 80286 provides a 1 gigabyte virtual address space per task mapped into a 16-megabyte physical address space defined by the address pin A_{23}–A_0 and $\overline{BHE}$. The virtual address space may be larger than the physical address space since any use of an address that does not map to a physical memory location will cause a restartable exception.

Memory Addressing

As in real address mode, protected mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector, however, specifies an index into a memory resident table rather than the upper 16 bits of a real memory address.

The 24-bit base address of the desired segment is obtained from the tables in memory. The 16-bit offset is added to the segment base address to form the physical address as shown in Figure 15. The tables are automatically referenced by the CPU whenever a segment register is loaded with a selector. All 80286 instructions which load a segment register will reference the memory-based tables without additional software. The memory-based tables contain 8-byte values called descriptors.

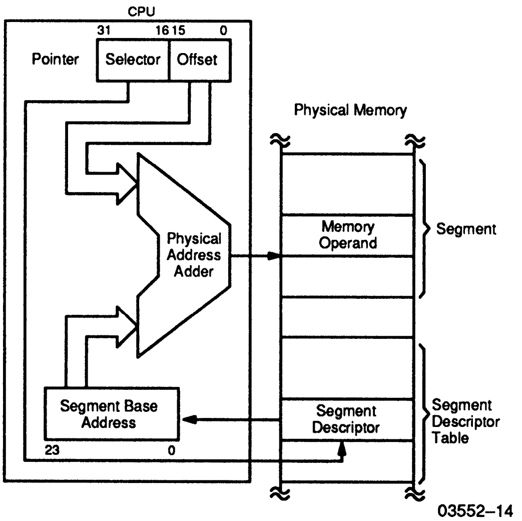


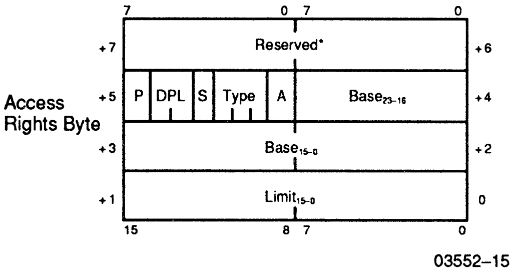
Figure 15. Protected Mode Memory Addressing

Descriptors

Descriptors define the use of memory. Special types of descriptors also define new functions for transfer of control and task switching. The 80286 has segment descriptors for code, stack and data segments, and system control descriptors for special system data segments and control transfer operations. Descriptor accesses are performed as locked bus operations to assure descriptor integrity in multi-processor systems.

Code and Data Segment Descriptors (S = 1)

Besides segment base addresses, code and data descriptors contain other segment attributes, including segment size (1 to 64 kbytes), access rights (read-only, read/write, execute-only, and execute/read), and presence in memory (for virtual memory systems) (see Figure 16). Any segment usage violating a segment attribute indicate by the segment descriptor will prevent the memory cycle and cause an exception or interrupt.



*Must be set to 0 for compatibility with iAPX 386.

Code and data are stored in two types of segments: code segments and data segments. Both types are identified and defined by segment descriptors. Code segments are identified by the executable (E) bit set to 1 in the descriptor access rights byte. The access rights byte of both code and data segment descriptor types have three fields in common: present (P) bit, Descriptor Privilege Level (DPL), and accessed (A) bit. If P = 0, any attempted use of this segment will cause a not-present exception. DPL specifies the privilege level of the segment descriptor. DPL controls when the descriptor may be used by a task (refer to privilege discussion). The A bit shows whether the segment has been previously accessed for usage profiling, a necessity for virtual memory systems. The CPU will always set this bit when accessing the descriptor.

Data segments (S = 1, E = 0) may be either read-only or read-write as controlled by the W bit of the access rights byte. Read-only (W = 0) data segments may not be written into. Data segments may grow in two directions, as determined by the Expansion Direction (ED) bit: upwards (ED = 0) for data segments, and downwards (ED = 1) for a segment containing a stack. The limit field for a data segment descriptor is interpreted differently depending on the ED bit (see Figure 16).

A code segment (S = 1, E = 1) may be execute-only or execute/read as determined by the Readable (R) bit. Code segments may never be written into and execute-only code segments (R = 0) may not be read. A code segment may also have an attribute called Conforming (C). A conforming code segment may be shared by programs that execute at different privilege levels. The DPL of a conforming code segment defines the range of privilege levels at which the segment may be executed (refer to privilege discussion).

System Segment Descriptors (S = 0, Type 1–3)

In addition to code and data segment descriptors, the protected mode 80286 defines system segment descriptors. These descriptors define special system data segments which contain a table of descriptors (Local Descriptor Table Descriptor) or segments which contain the execution state of a task (Task State Segment Descriptor).

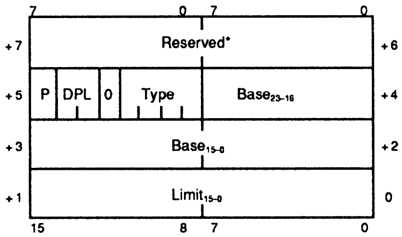
Figure 17 gives the formats for the special system data segment descriptors. The descriptors contain a 24-bit base address of the segment and a 16-bit limit. The access byte defines the type of descriptor, its state and privilege level. The descriptor contents are valid and the segment is in physical memory if P = 1. If P = 0, the segment is not valid. The DPL field is only used in Task State Segment descriptors and indicates the privilege level at which the descriptor may be used (see Privilege). Since the Local Descriptor Table descriptor may only be used by a special privileged instruction, the DPL field is not used. Bit 4 of the access byte is 0 to indicate that it is a system control descriptor. The Type field specifies the descriptor type as indicated in Figure 17.

Access Rights Byte Definition

	Bit Position	Name	Function		
	7	Present (P)	P = 1	Segment is mapped into physical memory.	
			P = 0	No mapping to physical memory exists; base and limit are not used. Segment privilege attribute used in privilege tests.	
	6–5	Descriptor Privilege Level (DPL)			
	4	Segment Descriptor (S)	S = 1	Code or Data segment descriptor	
			S = 0	Non-segment descriptor	
Type Field Definition	3	Executable (E)	E = 0	Data segment descriptor type is:	Data Segment
	2	Expansion Direction (ED)	ED = 0	Grow up segment, offsets must be ≤ limit.	
			ED = 1	Grow down segment, offsets must be > limit.	
	1	Writable (W)	W = 0	Data segment may not be written into.	
			W = 1	Data segment may be written into.	
	3	Executable (E)	E = 1	Code Segment Descriptor type is:	Code Segment
	2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL.	
	1	Readable (R)	R = 0	Code segment may not be read.	
			R = 1	Code segment may be read.	
	0	Accessed (A)	A = 0	Segment has not been accessed.	
			A = 1	Segment selector has been loaded into segment register or used by selector test instructions.	

Figure 16. Code and Data Segment Descriptors

System Segment Descriptor



*Must be set to 0 for compatibility with iAPX 386.

03552–16

System Segment Descriptor Fields

Name	Value	Description
Type	0	Invalid Descriptor
	1	Available Task State Segment
	2	Local Descriptor Table Descriptor
	3	Busy Task State Segment
	4–7	Control Descriptor
	8	Invalid Descriptor
	9–F	Reserved
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid
DPL	0–3	Descriptor Privilege Level
Base	24-bit number	Base Address of special system data segment in real memory
Limit	16-bit number	Offset of last byte in segment

Figure 17. System Segment Format

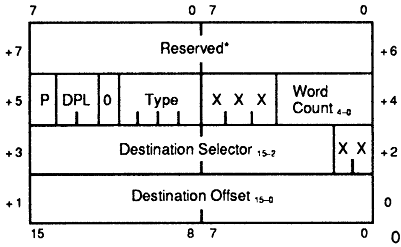
Gate Descriptors (S = 0, Type = 4–7)

Gates are used to control access to entry points within the target code segment. The gate descriptors are call gates, task gates, interrupt gates and trap gates. Gates provide a level of indirection between the source and destination of the control transfer. This indirection allows the CPU to automatically perform protection checks and control the entry point of the destination. Call gates are used to change privilege levels (see Privilege); task gates are used to perform a task switch; and interrupt and trap gates are used to specify interrupt service routines. The interrupt gate disables interrupts (resets IF) while the trap gates does not.

Figure 18 shows the format of the gate descriptors. The descriptor contains a destination pointer that points to the descriptor of the target segment and the entry point offset. The destination selector in an interrupt gate, trap gate, and call gate must refer to a code segment descriptor. These gate descriptors contain the entry point to prevent a program from constructing and using an illegal entry point. Task gates may only refer to a task state segment. Since task gates invoke a task switch, the destination offset is not used in the task gate.

Exception 13 is generated when the gate is used if a destination selector does not refer to the correct descriptor type. The Word Count field is used in the call gate descriptor to indicate the number of parameters (0–31 words) to be automatically copied from the caller's stack to the stack of the called routine when a control transfer changes privilege levels. The Word Count field is not used by any other gate descriptor.

Gate Descriptor



03552–17

*Must be set to 0 for compatibility with iAPX 386.

Gate Descriptor Fields

Name	Value	Description
Type	4	Call Gate
	5	Task Gate
	6	Interrupt Gate
	7	Trap Gate
P	0	Descriptor Contents are not valid
	1	Descriptor Contents are valid
DPL	0–3	Descriptor Privilege Level
Word Count	0–31	Number of words to copy from callers stack to called procedures stack. Only used with call gate.
Destination segment Selector	16-bit selector	Selector to the target code (Call, Interrupt or Trap Gate) Selector to the target task state segment (Task Gate)
Destination Offset	16-bit offset	Entry point within the target code segment

Figure 18. Gate Descriptor Format

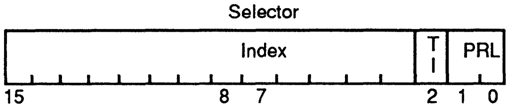
The access byte format is the same for all gate descriptors. P = 1 indicates that the gate contents are valid. P = 0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the Descriptor Privilege Level and specifies when this descriptor may be used by a task (refer to privilege discussion). Bit 4 must equal 0 to indicate a system control descriptor. The Type field specifies the descriptor type as indicated in Figure 18.

Segment Descriptor Cache Registers

A segment descriptor cache register is assigned to each of the four segment registers (CS, SS, DS, ES). Segment descriptors are automatically loaded (cached) into a segment descriptor cache register (Figure 20) whenever the associated segment register is loaded with a selector. Only segment descriptors may be loaded into segment descriptor cache registers. Once loaded, all references to that segment of memory use the cached descriptor information instead of reaccessing memory. The descriptor cache registers are not visible to programs. No instructions exist to store their contents. They only change when a segment register is loaded.

Selector Fields

A protected mode selector has three fields: descriptor entry index, local or global descriptor table indicator (TI), and selector privilege (RPL), as shown in Figure 19. These fields select one of two memory-based tables of descriptors, select the appropriate table entry, and allow high-speed testing of the selector's privilege attribute (refer to privilege discussion).



Bits	Name	Function
1–0	Requested Privilege Level (RPL)	Indicates Selector Privilege Level Desired

2	Table Indicator (TI)	TI = 0 Use Global Descriptor Table (GDT)
		TI = 1 Use Local Descriptor Table (LDT)

15–3	Index	Select Descriptor Entry in Table

Figure 19. Selector Fields

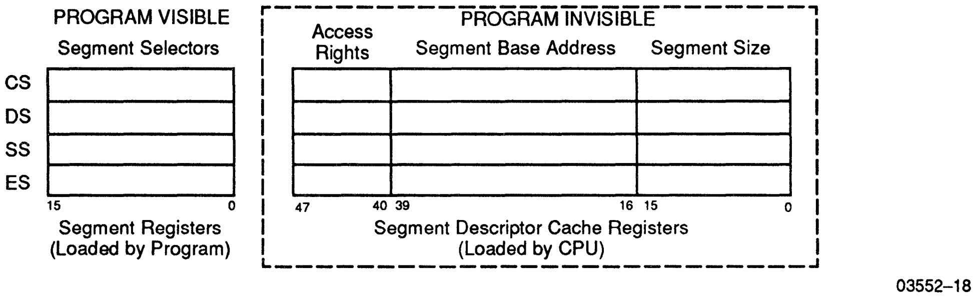


Figure 20. Descriptor Cache Registers

Local and Global Descriptor Tables

Two tables of descriptors, called descriptor tables, contain all descriptors accessible by a task at any given time. A descriptor table is a linear array of up to 8192 descriptors. The upper 13 bits of the selector value are an index into a descriptor table. Each table has a 24-bit base register to locate the descriptor table in physical memory and a 16-bit limit register that confines descriptor access to the defined limits of the table as shown in Figure 21. A restartable exception (13) will occur if an attempt is made to reference a descriptor outside the table limits.

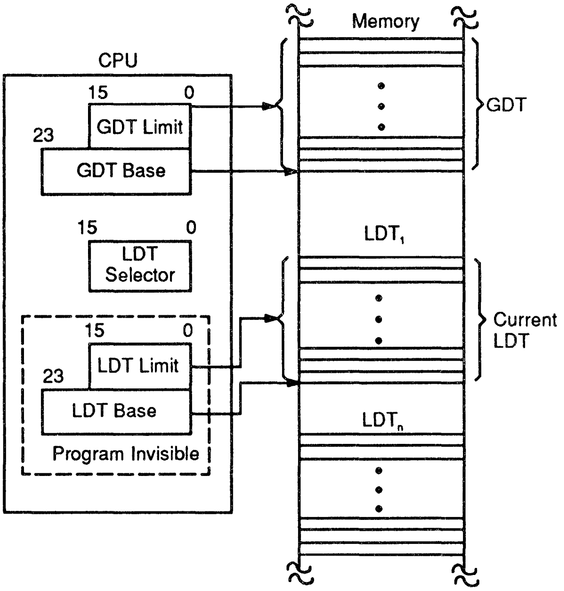
One table, called the Global Descriptor Table (GDT), contains descriptors available to all tasks. The other table, called the Local Descriptor Table (LDT), contains descriptors that can be private to a task. Each task may have its own private LDT. The GDT may contain all descriptor types except interrupt and trap descriptors. The LDT may contain only segment, task gate, and call gate descriptors. A segment cannot be accessed by a task if its segment descriptor does not exist in either descriptor table at the time of access.

The LGDT and LLDT instructions load the base and limit of the global and local descriptor tables. LGDT and LLDT are protected. They may only be executed by trusted programs operating at level 0. The LGDT instruction loads a six-byte field containing the 16-bit table limit and 24-bit base address of the Global Descriptor Table as shown in Figure 22. The LLDT instruction loads a selector which refers to a descriptor in the Local Descriptor Table. This descriptor contains the base address and limit for an LDT, as shown in Figure 21.

Interrupt Descriptor Table

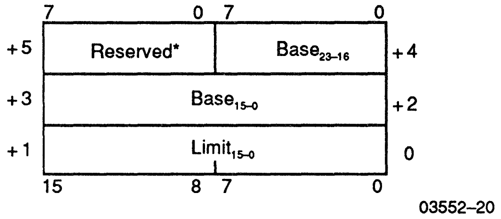
The protected mode 80286 has a third descriptor table, called the Interrupt Descriptor Table (IDT) (see Figure 23), used to define up to 256 interrupts. It may contain only task gates, interrupt gates and trap gates. The IDT (Interrupt Descriptor Table) has a 24-bit base and 16-bit limit register in the CPU. The protected LIDT instruction loads these registers with a six-byte value of identical form to that of the LGDT instruction (see Figure 22 and Protected Mode Initialization).

References to IDT entries are made via INT instructions, external interrupt vectors, or exceptions. The IDT must be at least 256 bytes in size to allocate space for all reserved interrupts.



03552–19

Figure 21. Local and Global Descriptor Table Definitions



*Must be set to 0 for compatibility with iAPX 386.

Figure 22. Global Descriptor Table and Interrupt Descriptor Data Types

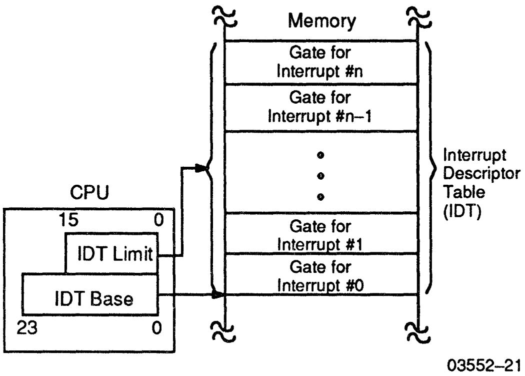


Figure 23. Interrupt Descriptor Table Definition

Privilege

The 80286 has a four-level hierarchical privilege system which controls the use of privileged instructions and access to descriptors (and their associated segments) within a task. Four-level privilege, as shown in Figure 24, is an extension of the user/supervisor mode commonly found in minicomputers. The privilege levels are numbered 0 through 3. Level 0 is the most privileged level. Privilege levels provide protection *within* a task. (Tasks are isolated by providing private LDT's for each task.) Operating system routines, interrupt handlers, and other system software can be included and protected within the virtual address space of each task using the four levels of privilege. Tasks may also have a separate stack for each privilege level.

Tasks, descriptors, and selectors have a privilege level attribute that determines whether the descriptor may be used. Task privilege effects the use of instructions and descriptors. Descriptor and selector privilege only effect access to the descriptor.

Task Privilege

The task always executes at one of the four privilege levels. A task privilege level at any specific instant is called the Current Privilege Level (CPL) and is defined by the lower two bits of the CS register. CPL cannot change during execution in a single code segment. A task's CPL may only be changed by control transfers through gate descriptors to a new code segment (See Control Transfer). Tasks begin executing at the CPL value specified by the code segment when the task is initiated via a task switch operation. A task executing at Level 0 can access all data segments defined in the GDT and the task's LDT and is considered the most trusted level. A task executed at Level 3 has the most restricted access to data and is considered the least trusted level.

Descriptor Privilege

Descriptor privilege is specified by the Descriptor Privilege Level (DPL) field of the descriptor access byte. DPL specifies the least trusted privilege level (CPL) at which a task may access the descriptor. Descriptors with DPL = 0 are the most protected. Only tasks executing at privilege level 0 (CPL = 0) may access them. Descriptors with DPL = 3 are the least protected (i.e., have the least restricted access) since tasks can access them when CPL = 0, 1, 2, or 3. This rule applies to all descriptors, except LDT descriptors.

Selector Privilege

Selector privilege is specified by the Requested Privilege Level (RPL) field in the least significant two bits of a selector. Selector RPL may establish a less trusted privilege level than the current privilege level for the use of a selector. This level is called the task's effective privilege level (EPL). RPL can only reduce the scope of a task's access to data with this selector. A task's effective privilege is the numeric maximum of RPL and CPL. A selector with RPL = 0 imposes no additional restriction on its use while a selector with RPL = 3 can only refer to segments at privilege Level 3 regardless of the task's

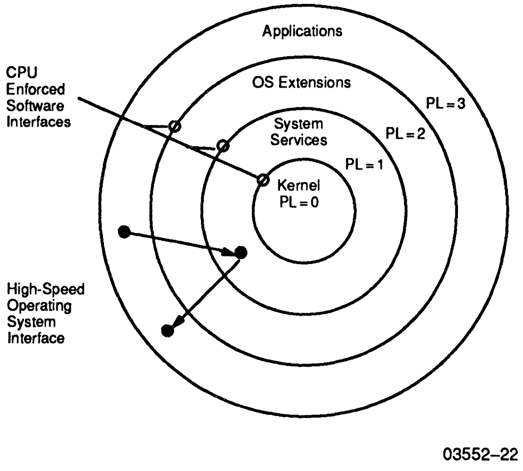


Figure 24. Hierarchical Privilege Levels

CPL. RPL is generally used to verify that pointer parameters passed to a more trusted procedure are not allowed to use data at a more privileged level than the caller (refer to pointer testing instructions).

Descriptor Access and Privilege Validation

Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used and CPL, RPL, and DPL. The two basic types of segment accesses are control transfer (selectors loaded into CS) and data (selectors loaded into DS, ES, or SS).

Data Segment Access

Instructions that load selectors into DS and ES must refer to a data segment descriptor or readable code segment descriptor. The CPL of the task and the RPL of the selector must be the same as or more privileged (numerically equal to or lower than) than the descriptor DPL. In general, a task can only access data segments at the same or less privileged levels than the CPL or RPL (whichever is numerically higher) to prevent a program from accessing data it cannot be trusted to use.

An exception to the rule is a readable conforming code segment. This type of code segment can be read from any privilege level.

If the privilege checks fail (e.g., DPL is numerically less than the maximum of CPL and RPL) or an incorrect type of descriptor is referenced (e.g., gate descriptor or execute only code segment), exception 13 occurs. If the segment is not present, exception 11 is generated.

Instructions that load selectors into SS must refer to data segment descriptors for writable data segments. The descriptor privilege (DPL) and RPL must equal CPL.

All other descriptor types or privilege level violation will cause exception 13. A not-present fault causes exception 12.

Control Transfer

Four types of control transfer can occur when a selector is loaded into CS by a control transfer operation (see Table 9). Each transfer type can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules (e.g., JMP through a call gate or RET to a Task State Segment) will cause exception 13.

The ability to reference a descriptor for control transfer is also subject to rules of privilege. A CALL or JUMP instruction may only reference a code segment descriptor with DPL equal to the task CPL or a conforming segment with DPL of equal or greater privilege than CPL. The RPL of the selector used to reference the code descriptor must have as much privilege as CPL.

RET and IRET instructions may only reference code segment descriptors with descriptor privilege equal to or less privileged than the task CPL. The selector loaded into CS is the return address from the stack. After the return, the selector RPL is the task's new CPL. If CPL changes, the old stack pointer is popped after the return address.

When a JMP or CALL references a Task State Segment descriptor, the descriptor DPL must be the same or less privileged than the task's CPL. Reference to a valid Task State Segment descriptor causes a task switch (see Task Switch Operation). Reference to a Task State Segment descriptor at a more privileged level than the task's CPL generates exception 13.

When an instruction or interrupt references a gate descriptor, the gate DPL must have the same or less privilege than the task CPL. If DPL is at a more privileged level than CPL, exception 13 occurs. If the destination

Table 9. Descriptor Types Used for Control Transfer

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL	CALL	Call Gate	GDT/LDT
	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
Task Switch	CALL, JMP	Task State Segment	GDT
	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

* NT (Nested Task bit of flag word) = 0
** NT (Nested Task bit of flag word) = 1

selector contained in the gate references a code segment descriptor, the code segment descriptor DPL must be the same or more privileged than the task CPL. If not, Exception 13 is issued. After the control transfer, the code segment descriptor DPL is the task's new CPL. If the destination selector in the gate references a task state segment, a task switch is automatically performed (see Task Switch Operation).

The privilege rules on control transfer require:

— JMP or CALL direct to a code segment (code segment descriptor) can only be to a conforming segment with DPL of equal or greater privilege than CPL or a non-conforming segment at the same privilege level.

— interrupts within the task or calls that may change privilege levels can only transfer control through a gate at the same or a less privileged level than CPL to a code segment at the same or more privileged level than CPL.

— return instructions that don't switch tasks can only return control to a code segment at the same or less privileged level.

— task switch can be performed by a call, a jump or an interrupt which references either a task gate or task state segment at the same or less privileged level.

Privilege Level Changes

Any control transfer that changes CPL within the task causes a change of stacks as part of the operation. Initial values of SS:SP for privilege levels 0, 1, and 2 are kept in the task state segment (refer to Task Switch Operation). During a JMP or CALL control transfer, the new stack pointer is loaded into the SS and SP registers and the previous stack pointer is pushed onto the new stack.

When returning to the original privilege level, its stack is restored as part of the RET or IRET instruction operation. For subroutine calls that pass parameters on the stack and cross privilege levels, a fixed number of words, as specified in the gate, are copied from the previous stack to the current stack. The intersegment RET instruction with a stack adjustment value will correctly restore the previous stack pointer upon return.

Protection

The 80286 includes mechanisms to protect critical instructions that affect the CPU execution state (e.g., HLT) and code or data segments from improper usage. These mechanisms are grouped under the term "protection" and have three forms:

- Restricted usage of segments (e.g., no write allowed to read-only data segments). The only segments available for use are defined by descriptors in the Local Descriptor Table (LDT) and Global Descriptor Table (GDT).
- Restricted access to segments via the rules of privilege and descriptor usage.
- Privileged instructions or operations that may only be executed at certain privilege levels as determined by the CPL and I/O Privilege Level (IOPL). The IOPL is defined by bits 14 and 13 of the flag word.

These checks are performed for all instructions and can be split into three categories: segment load checks (Table 10), operand reference checks (Table 11), and privileged instruction checks (Table 12). Any violation of the rules shown will result in an exception. A not-present exception related to the stack segment causes exception 12.

The IRET and POPF instructions do not perform some of their defined functions if CPL is not of sufficient privilege (numerically small enough). Precisely, these are:

- The IF bit is not changed if CPL > IOPL.
- The IOPL field of the flag word is not changed if CPL > 0.

No exceptions or other indication are given when these conditions occur.

Table 10. Segment Register Load Checks

Error Description	Exception Number
Descriptor table limit exceeded	13
Segment descriptor not present	11 or 12
Privilege rules violated	13
Invalid descriptor/segment type segment register load: — Read only data segment load to SS — Special control descriptor load to DS, ES, SS — Execute only segment load to DS, ES, SS — Data segment load to CS — Read/Execute code segment load to SS	13

Table 11. Operand Reference Checks

Error Description	Exception Number
Write into code segment	13
Read from execute-only code segment	13
Write to read-only data segment	13
Segment limit exceeded[1]	12 or 13

Note: Carry out in offset calculations is ignored.

Table 12. Privileged Instruction Checks

Error Description	Exception Number
CPL≠0 when executing the following instructions: LIDT, LLDT, LGDT, LTR, LMSW, CTS, HLT	13
CPL > IOPL when executing the following instructions: INS, IN, OUTS, OUT, STI, CLI, LOCK	13

Exceptions

The 80286 detects several types of exceptions and interrupts in protected mode (see Table 13). Most are restartable after the exceptional condition is removed. Interrupt handlers for most exceptions receive an error code, pushed on the stack after the return address, that identifies the selector involved (0 if none). The return address normally points to the failing instruction, including all leading prefixes. For a processor extension segment overrun exception, the return address will not point at the ESC instruction that caused the exception; however, the processor extension registers may contain the address of the failing instruction.

Special Operations

Task Switch Operation

The 80286 provides a built-in task switch operation which saves the entire 80286 execution state (registers, address space, and a link to the previous task), loads a new execution state, and commences execution in the new task. Like gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS) or task gate descriptor in the GDT or LDT. An INT n instruction, exception, or external interrupt may also invoke the task switch operation by selecting a task gate descriptor in the associated IDT descriptor entry.

The TSS descriptor points at a segment (see Figure 25) containing the entire 80286 execution state while a task gate descriptor contains a TSS selector. The limit field must be > 002B(H).

Each task must have a TSS associated with it. The current TSS is identified by a special register in the 80286 called the Task Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TR are loaded whenever TR is loaded with a new selector.

The IRET instruction is used to return control to the task that called the current task or was interrupted. Bit 14 in the flag register is called the Nested Task(NT) bit. It controls the function of the IRET instruction. If NT = 0, the IRET instruction performs the regular current task return; when NT = 1, IRET performs a task switch operation back to the previous task.

When a CALL or INT instruction initiates a task switch, the old and new TSS will be marked busy and the back link field of the new TSS set to the old TSS selector. The NT bit of the new task is set by CALL or INT initiated task switches. An interrupt that does not cause a task switch will clear NT. NT may also be set or cleared by POPF or IRET instructions.

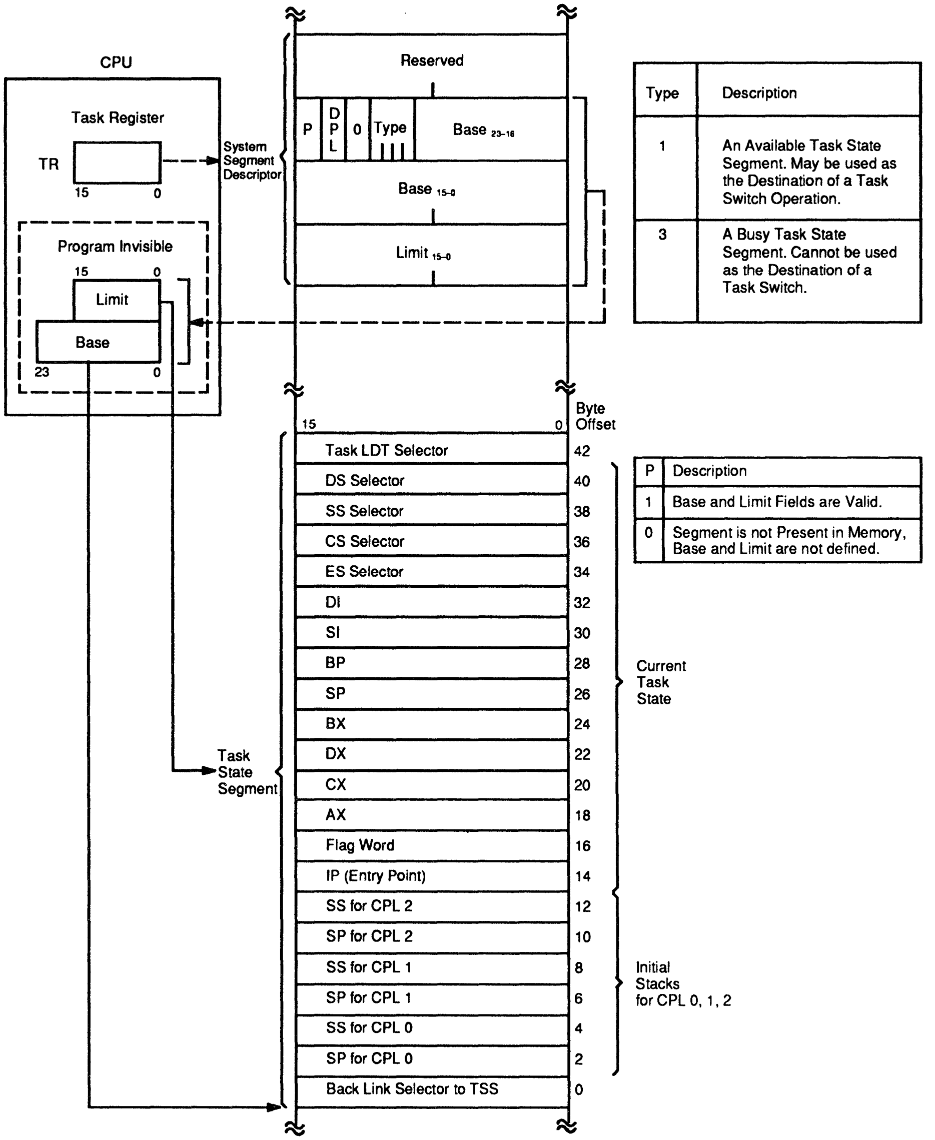
The task state segment is marked busy by changing the descriptor type field from Type 1 to Type 3. Use of a selector that references a busy task state segment causes Exception 13.

Table 13. Protected Mode Exceptions

Interrupt Vector	Function	Return Address At Failing Instruction?	Always Restartable?	Error Code on Stack?
8	Double exception detected	Yes	No[2]	Yes
9	Processor extension segment overrun	No	No[2]	No
10	Invalid task state segment	Yes	Yes	Yes
11	Segment not present	Yes	Yes	Yes
12	Stack segment overrun or segment not present	Yes	Yes[1]	Yes
13	General protection	Yes	No[2]	Yes

Notes:

1. When a PUSHA or POPA instruction attempts to wrap around the stack segment, the machine state after the exception will not be restartable because stack segment wraparound is not permitted. This condition is identified by the value of the saved SP being either 0000(H), 0001(H), FFFE(H), or FFFF(H).
2. These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.
3. All these checks are performed for all instructions and can be split into three categories: Segment Load Checks (Table 10), Operand Reference Checks (Table 11), and Privileged Instruction Checks (Table 12). Any violation of the rules shown will result in an exception. A not-present exception causes exception 11 or 12 and is restartable.



Type	Description
1	An Available Task State Segment. May be used as the Destination of a Task Switch Operation.
3	A Busy Task State Segment. Cannot be used as the Destination of a Task Switch.

P	Description
1	Base and Limit Fields are Valid.
0	Segment is not Present in Memory, Base and Limit are not defined.

15 0	Byte Offset
Task LDT Selector	42
DS Selector	40
SS Selector	38
CS Selector	36
ES Selector	34
DI	32
SI	30
BP	28
SP	26
BX	24
DX	22
CX	20
AX	18
Flag Word	16
IP (Entry Point)	14
SS for CPL 2	12
SP for CPL 2	10
SS for CPL 1	8
SP for CPL 1	6
SS for CPL 0	4
SP for CPL 0	2
Back Link Selector to TSS	0

Figure 25. Task State Segment and TSS Registers

Processor Extension Context Switching

The context of a processor extension is not changed by the task switch operation. A processor extension context need only be changed when a different task attempts to use the processor extension (which still contains the context of a previous task). The 80286 detects the first use of a processor extension after a task switch by causing the processor extension not present exception (7). The interrupt handler may then decide whether a context change is necessary.

Whenever the 80286 switches tasks, it sets the Task Switched (TS) bit of the MSW. TS indicates that a processor extension context may belong to a different task than the current one. The processor extension not present exception (7) will occur when attempting to execute an ESC or WAIT instruction if TS = 1 and a processor extension is present (MP = 1 in MSW).

Pointer Testing Instructions

The 80286 provides several instructions to speed pointer testing and consistency checks for maintaining system integrity (see Table 14). These instructions use the memory management hardware to verify that a selector value refers to an appropriate segment without risking an exception. A condition flag indicates whether use of the selector or segment will cause an exception.

Table 14. Pointer Test Instructions

Instruction	Operands	Function
ARPL	Selector, Register Selector	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 is the segment referred to by the selector can be read.
VERW	Register, Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL		Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

Double Fault and Shutdown

If two separate exceptions are detected during a single instruction execution, the 80286 performs the double fault exception (8). If an exception occurs during processing of the double fault exception, the 80286 will enter shutdown. During shutdown no further instructions or exceptions are processed. Either NMI (CPU remains in protected mode) or RESET (CPU exits protected mode) can force the 80286 out of shutdown. Shutdown is externally signaled via a HALT bus operation with A_1 High.

Protected Mode Initialization

The 80286 initially executes in real address mode after RESET. To allow initialization code to be placed at the top of physical memory, A_{23-20} will be High when the 80286 performs memory references relative to the CS register, until CS is changed. A_{23-20} will be zero for references to the DS, ES, or SS segments. Changing CS in real address mode will force A_{23-20} Low whenever using CS thereafter. The initial CS:IP value of FFOO:FFFO provides 64K bytes of code space for initialization code without changing CS.

Before placing the 80286 into protected mode, several registers must be initialized. The GDT and IDT base registers must refer to a valid GDT and IDT. After executing the LMSW instruction to set PE, the 80286 must immediately execute an intrasegment JMP instruction to clear the instruction queue of instructions decoded in real address mode.

To force the 80286 CPU registers to match the initial protected mode state assumed by software, execute a JMP instruction with a selector referring to the initial TSS used in the system. This will load the task register, local descriptor table register, segment registers and initial general register state. The TR should point at a valid TSS since a task switch operation involves saving the current

System Interface

The 80286 system interface appears in two forms: a local bus and a system bus. The local bus consists of address, data, status, and control signals at the pins of the CPU. A system bus is any buffered version of the local bus. A system bus may also differ from the local bus in terms of coding of status and control lines and/or timing and loading of signals. The 80286 family includes several devices to generate standard system buses such as the IEEE 796 Standard MULTIBUS.

Bus Interface Signals and Timing

The 80286 microsystem local bus interfaces the 80286 to local memory and I/O components. The interface has 24 address lines, 16 data lines, and 8 status and control signals.

The 80286 CPU, 82284 clock generator, 82C288 bus controller, 82289 bus arbiter, 8286/7 transceivers, and 8282/3 latches provide a buffered and decoded system bus interface. The 82284 generates the system clock and synchronizes $\overline{\text{READY}}$ and RESET. The 82C288 converts bus operation status encoded by the 80286 into command and bus control signals. These components can provide the timing and electrical power drive levels required for most system bus interfaces including the MULTIBUS.

Physical Memory and I/O Interface

A maximum of 16 megabytes of physical memory can be addressed in protected mode. One megabyte can be addressed in real address mode. Memory is accessible as bytes or words. Words consist of any two consecutive bytes addressed with the least significant byte stored in the lowest address.

Byte transfers occur on either half of the 16-bit local data bus. Even bytes are accessed over D_{7-0} while odd bytes are transferred over D_{15-8}. Even-addressed words are transferred over D_{15-0} in one bus cycle, while odd-addressed words require *two* bus operations. The first transfers data on D_{15-8}, and the second transfers data on D_{7-0}. Both byte data transfers occur automatically, transparent to software.

Two bus signals, A_0 and $\overline{BHE}$, control transfers over the lower and upper halves of the data bus. Even address byte transfers are indicated by A_0 Low and $\overline{BHE}$ High. Odd address byte transfers are indicated by A_0 High and $\overline{BHE}$ Low. Both A_0 and $\overline{BHE}$ are Low for even address word transfers.

The I/O address space contains 64K addresses in both modes. The I/O space is accessible as either bytes or words, as is memory. Byte-wide peripheral devices may be attached to either the upper or lower byte of the data bus. Byte-wide I/O devices attached to the upper data byte (D_{15-8}) are accessed with odd I/O addresses. Devices on the lower data byte are accessed with even I/O addresses. An interrupt controller such as the 8259A must be connected to the lower data byte (D_{7-0}) for proper return of the interrupt vector.

Bus Operation

The 80286 uses a double-frequency system clock (CLK input) to control bus timing. All signals on the local bus are measured relative to the system CLK input. The CPU divides the system clock by 2 to produce the internal processor clock, which determines bus state. Each processor clock is composed of two system clock cycles named phase 1 and phase 2. The 82284 clock generator output (PCLK) identifies the next phase of the processor clock. (See Figure 26.)

Six types of bus operations are supported: memory read, memory write, I/O read, I/O write, interrupt acknowledge, and halt/shutdown. Data can be transferred at a maximum rate of one word per two processor clock cycles.

The 80286 bus has three basic states: idle (T_i), send status (T_s), and perform command (T_c). The 80286 CPU also has a fourth local bus state called hold (T_h). T_h indicates that the 80286 has surrendered control of the local bus to another bus master in response to a HOLD request.

Each bus state is one processor clock long. Figure 27 shows the four 80286 local bus states and allowed transitions.

Bus States

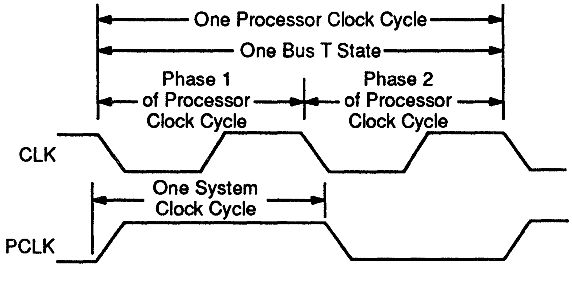
The idle (T_i) state indicates that no data transfers are in progress or requested. The first active state, T_s, is signaled by either status line $\overline{S1}$ or $\overline{S0}$ going Low also identifying phase 1 of the processor clock. During T_s, the command encoding, the address, and data (for a write operation) are available on the 80286 output pins. The 82C288 bus controller decodes the status signals and generates MULTIBUS-compatible read/write command and local transceiver control signals.

After T_s, the perform command (T_c) state is entered. Memory or I/O devices respond to the bus operation during T_c, either transferring read data to the CPU or accepting write data. T_c states may be repeated as often as necessary to assure sufficient time for the memory or I/O device to respond. The $\overline{READY}$ signal determines whether T_c is repeated. A repeated T_c state is called a wait state.

During hold (T_h), the 80286 will float all address, data, and status output pins, enabling another bus master to use the local bus. The 80286 HOLD input signal is used to place the 80286 into the T_h state. The 80286 HLDA output signal indicates that the CPU has entered T_h.

Pipelined Addressing

The 80286 uses a local bus interface with pipelined timing to allow as much time as possible for data access.


One Processor Clock Cycle
One Bus T State
Phase 1 of Processor Clock Cycle
Phase 2 of Processor Clock Cycle
CLK
One System Clock Cycle
PCLK

03552-24

Figure 26. System and Processor Clock Relationships

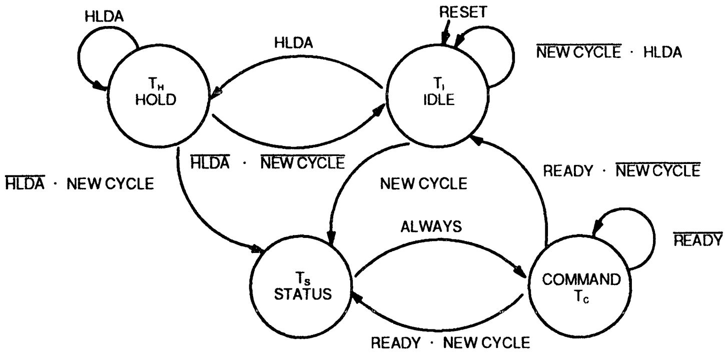
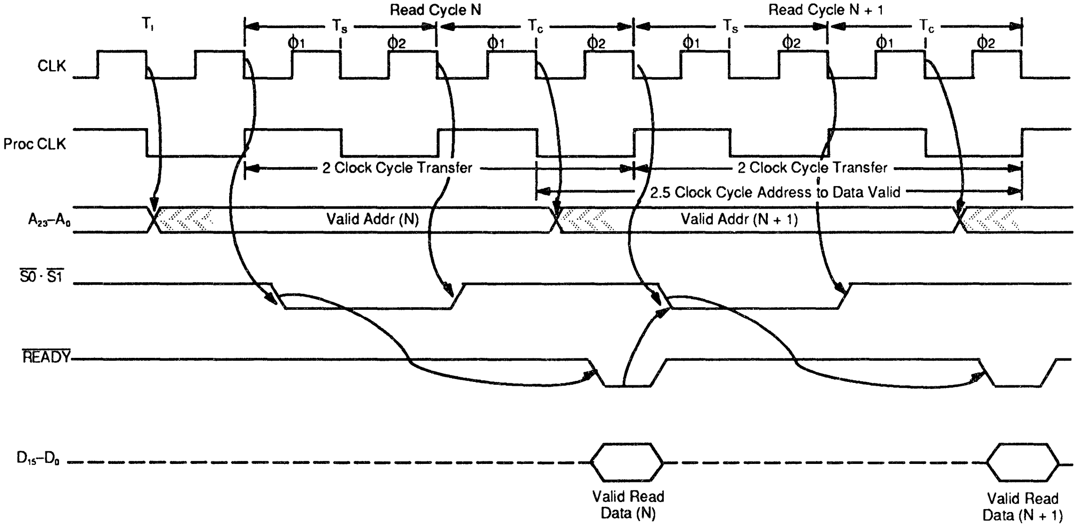


Figure 27. 80286 Bus States

Pipelined timing allows bus operations to be performed in two processor cycles, while allowing each individual bus operation to last for three processor cycles.

The timing of the address outputs is pipelined such that the address of the next bus operation becomes available during the current bus operation. Or in other words, the first clock of the next bus operation is overlapped with the last clock of the current bus operation. Therefore, address decode and routing logic can operate in advance of the next bus operation. External address latches may hold the address stable for the entire bus operation and provide additional AC and DC buffering.

The 80286 does not maintain the address of the current bus operation during all Tc states. Instead, the address for the next bus operation may be emitted during phase 2 of any Tc. The address remains valid during phase 1 of the first Tc to guarantee hold time, relative to ALE, for the address latch inputs.



Pipelining: valid address (N + 1) available in last phase of bus cycle (N).

03552-26

Figure 28. Basic Bus Cycle

Bus Control Signals

The 82C288 bus controller provides control signals: address latch enable (ALE), Read/Write commands, data transmit/receive (DT/$\overline{\text{R}}$), and data enable (DEN) that control the address latches, data transceivers, write enable, and output enable for memory and I/O systems.

The Address Latch Enable (ALE) output determines when the address may be latched. ALE provides at least one system CLK period of address hold time from the end of the previous bus operation until the address for the next bus operation appears at the latch outputs. This address hold time is required to support MULTIBUS and common memory systems.

The data bus transceivers are controlled by 82C288 outputs Data Enable (DEN) and Data Transmit/Receive (DT/$\overline{\text{R}}$). DEN enables the data transceivers while DT/$\overline{\text{R}}$ controls transceiver direction. DEN and DT/$\overline{\text{R}}$ are timed to prevent bus contention between the bus master, data bus transceivers, and system data bus transceivers.

Command Timing Controls

Two system timing customization options, command extension and command delay, are provided on the 80286 local bus.

Command extension allows additional time for external devices to respond to a command and is analogous to inserting wait states on the 80286. External logic can control the duration of any bus operation such that the operation is only as long as necessary. The $\overline{\text{READY}}$ input signal can extend any bus operation for as long as necessary.

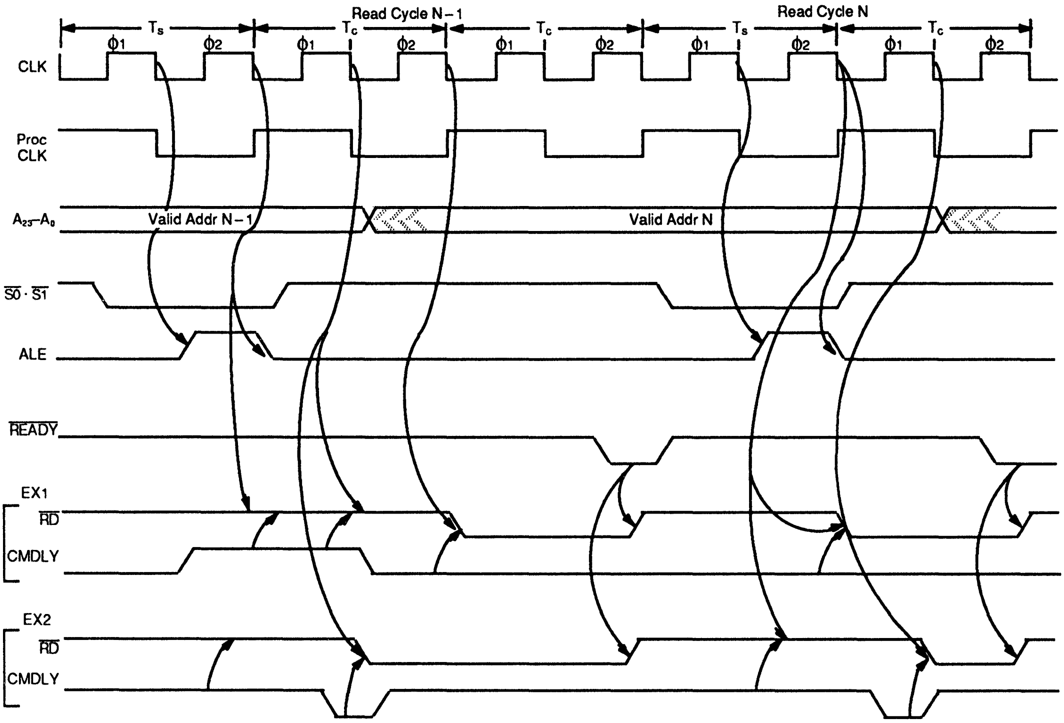
Command delay allows an increase of address or write data set-up time to system bus command active for any bus operation by delaying when the system bus command becomes active. Command delay is controlled by the 82C288 CMDLY input. After T_S, the bus controller samples CMDLY at each failing edge of CLK. If CMDLY is High, the 82C288 will not activate the command signal. When CMDLY is Low, the 82C288 will activate the command signal. After the command becomes active, the CMDLY input is not sampled.

When a command is delayed, the available response time from command active to return read data or accept write data is less. To customize system bus timing, an address decoder can determine which bus operations require delaying the command. The CMDLY input does not affect the timing of ALE, DEN, or DT/$\overline{\text{R}}$.

Figure 29 illustrates four uses of CMDLY. Example 1 shows delaying the read command two system CLKs for cycle N−1 and no delay for cycle N, and example 2 shows delaying the read command one system CLK for cycle N−1 and one system CLK delay for cycle N.

Bus Cycle Termination

At maximum transfer rates, the 80286 bus alternates between the status and command states. The bus status signals become inactive after T_S so that they may correctly signal the start of the next bus operation after the completion of the current cycle. No external indication of T_C exists on the 80286 local bus. The bus master and bus controller enter T_C directly after T_S and continue executing T_C cycles until terminated by $\overline{\text{READY}}$.



03552-27

Figure 29. CMDLY Controls and Leading Edge of the Command

READY Operation

The current bus master and 82C288 bus controller terminate each bus operation simultaneously to achieve maximum bus bandwidth. Both are informed in advance by READY active which identifies the last Tc cycle of the current bus operation. The bus master and bus controller must see the same sense of the READY signal, thereby requiring READY be synchronous to the system clock.

Synchronous Ready

The 82284 clock generator provides READY synchronization from both synchronous and asynchronous sources (see Figure 30). The synchronous ready input (SRDY) of the clock generator is sampled with the falling edge of CLK at the end of phase 1 of each Tc. The state of SRDY is then broadcast to the bus master and bus controller via the READY output line.

Asynchronous Ready

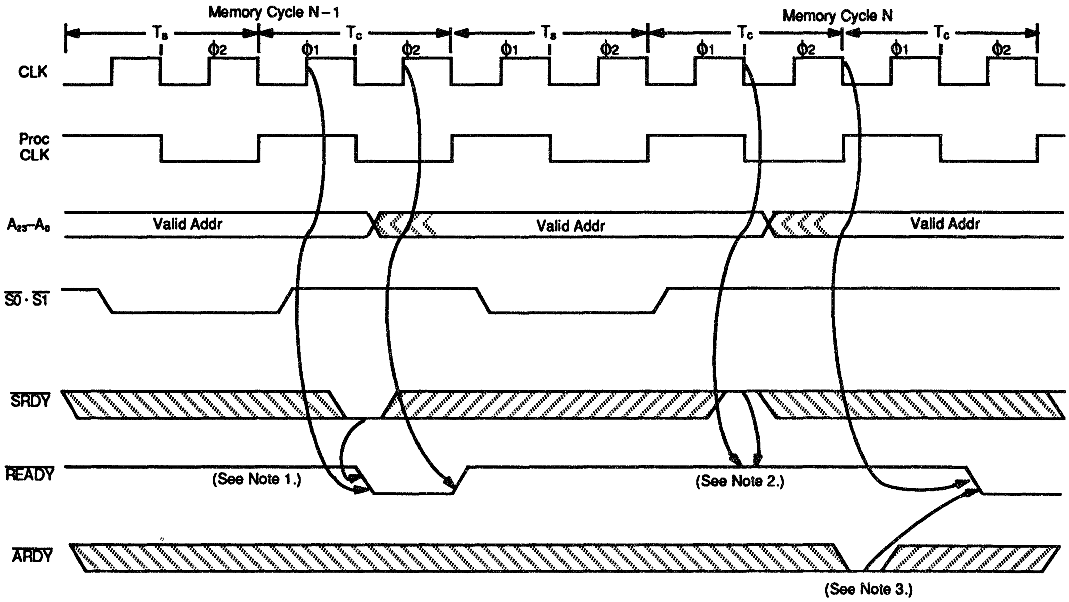
Many systems have devices or subsystems that are asynchronous to the system clock. As a result, their ready outputs cannot be guaranteed to meet the 82284 SRDY set-up and hold time requirements. The 82284 asynchronous ready input (ARDY) is designed to accept such signals. The ARDY input is sampled at the beginning of each Tc cycle by 82284 synchronization logic. This provides a system CLK cycle time to resolve its value before broadcasting it to the bus master and bus controller.

ARDY or ARDYEN must be High at the end of Ts. ARDY cannot be used to terminate bus cycle with no wait status.

Each ready input of the 82284 has an enable pin (SRDYEN and ARDYEN) to select whether the current bus operation will be terminated by the synchronous or asynchronous ready. Either of the ready inputs may terminate a bus operation. These enable inputs are active low and have the same timing as their respective ready inputs. Address decode logic usually selects whether the current bus operation should be terminated by ARDY or SRDY.

Data Bus Control

Figures 31, 32, and 33 show how the DT/R, DEN, data bus, and address signals operate for different combinations of read, write, and idle bus operations. DT/R goes



Notes:
1. $\overline{SRDYEN}$ is active Low.
2. If $\overline{SRDYEN}$ is High, the state of $\overline{SRDY}$ will not affect $\overline{READY}$.
3. $\overline{ARDYEN}$ is active Low.

Figure 30. Synchronous and Asynchronous Ready

active (Low) for a read operation. DT/$\overline{R}$ remains High before, during, and between write operations.

The data bus is driven with write data during the second phase of T_s. The delay in write data timing allows the read data drivers, from a previous read cycle, sufficient time to enter three-state OFF before the 80286 CPU begins driving the local data bus for write operations. Write data will always remain valid for one system clock past the last T_c to provide sufficient hold time for MULTIBUS or other similar memory or I/O systems. During write-read or write-idle sequences, the data bus enters three-state OFF during the second phase of the processor cycle after the last T_c. In a write-write sequence the data bus does not enter three-state OFF between T_c and T_s.

Bus Usage

The 80286 local bus may be used for several functions: instruction data transfers, data transfers by other bus masters, instruction fetching, processor extension data transfers, interrupt acknowledge, and halt/shutdown. This section describes local bus activities which have special signals or requirements.

HOLD and HLDA

HOLD and HLDA allow another bus master to gain control of the local bus by placing the 80286 bus into the T_h state. The sequence of events required to pass control between the 80286 and another local bus master are shown in Figure 34.

In this example, the 80286 is initially in the T_h, state as signaled by HLDA being active. Upon leaving T_h, as signaled by HLDA going inactive, a write operation is started. During the write operation another local bus master requests the local bus from the 80286 as shown by the HOLD signal. After completing the write operation, the 80286 performs one T_i bus cycle, to guarantee write data hold time, then enters T_h as signaled by HLDA going active.

The CMDLY signal and $\overline{ARDY}$ ready are used to start and stop the write bus command, respectively. Note that $\overline{SRDY}$ must be inactive or disabled by $\overline{SRDYEN}$ to guarantee $\overline{ARDY}$ will terminate the cycle.

HOLD must not be active during the time from the leading edge of RESET until 34 CLKs following the trailing edge of RESET unless the 80286 is in the Halt condition. To ensure that the 80286 remains in the Halt condition until the processor Reset operation is complete, no interrupts should occur after the execution of HLT until 34 CLKs after the trailing edge of the RESET pulse.

Lock

The CPU asserts an active lock signal during Interrupt-Acknowledge cycles, the XCHG instruction, and during some descriptor accesses. Lock is also asserted when

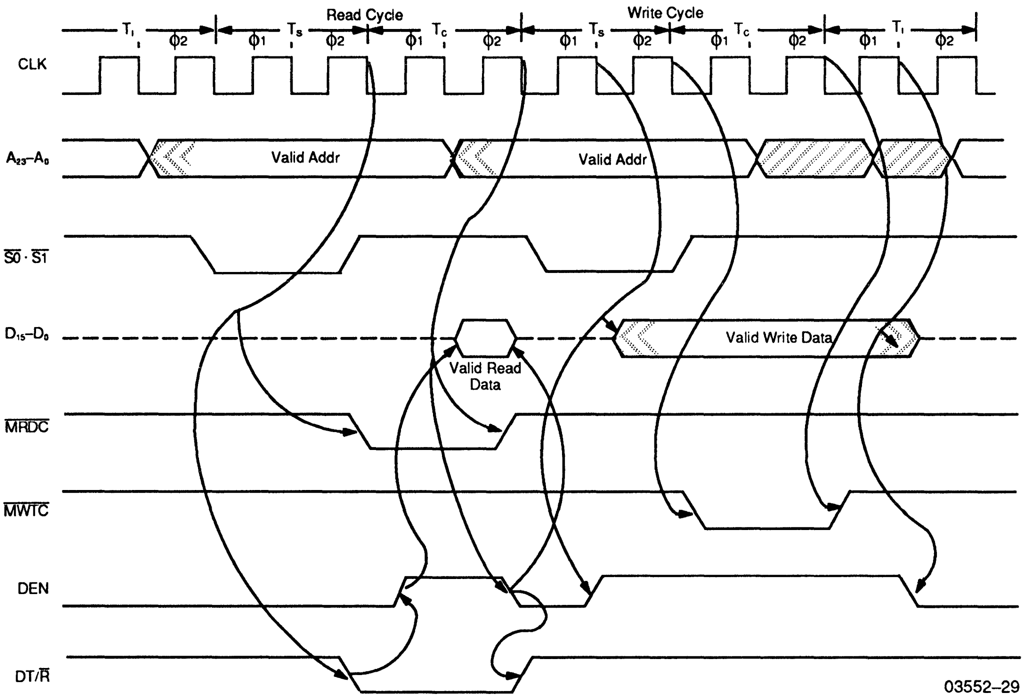


Figure 31. Back-to-Back Read-Write Cycles

the LOCK prefix is used. The LOCK prefix may be used with the following ASM-286 assembly instructions; MOVS, INS, and OUTS. For bus cycles other than Interrupt-Acknowledge cycles, Lock will be active for the first and subsequent cycles of a series of cycles to be locked. Lock will not be shown active during the last cycle to be locked. For the next-to-last cycle, Lock will become inactive at the end of the first Tc regardless of the number of wait-states inserted. For Interrupt-Acknowledge cycles, Lock will be active for each cycle, and will become inactive at the end of the first Tc for each cycle regardless of the number of wait-states inserted.

Instruction Fetching

The 80286 Bus Unit (BU) will fetch instructions ahead of the current instruction being executed. This activity is called prefetching. It occurs when the local bus would otherwise be idle and obeys the following rules:

A prefetch bus operation starts when at least two bytes of the 6-byte prefetch queue are empty.

The prefetcher normally performs word prefetches independent of the byte alignment of the code segment base in physical memory.

The prefetcher will perform only a byte code fetch operation for control transfers to an instruction beginning on a numerically odd physical address.

Prefetching stops whenever a control transfer or HLT instruction is decoded by the IU and placed into the instruction queue.

In real address mode, the prefetcher may fetch up to 5 bytes beyond the last control transfer or HLT instruction in a code segment.

In protected mode, the prefetcher will never cause a segment overrun exception. The prefetcher stops at the last physical memory word of the code segment. Exception 13 will occur if the program attempts to execute beyond the last full instruction in the code segment.

If the last byte of a code segment appears on an even physical memory address, the prefetcher will read the next physical byte of memory (perform a word code fetch). The value of this byte is ignored and any attempt to execute it causes exception 13.

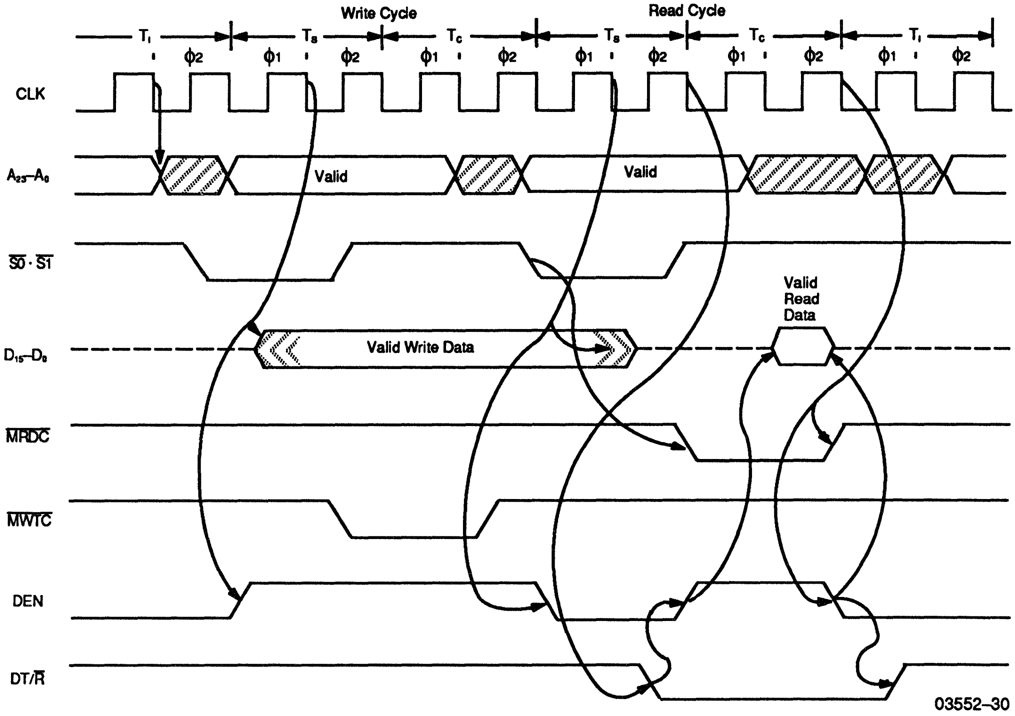


Figure 32. Back-to-Back Write-Read Cycles

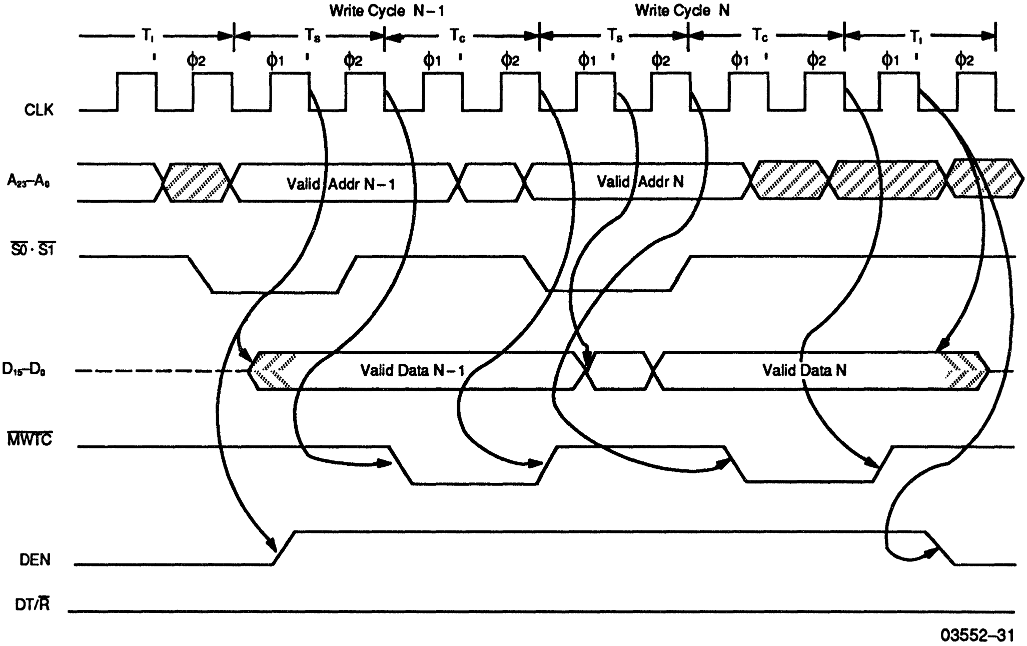
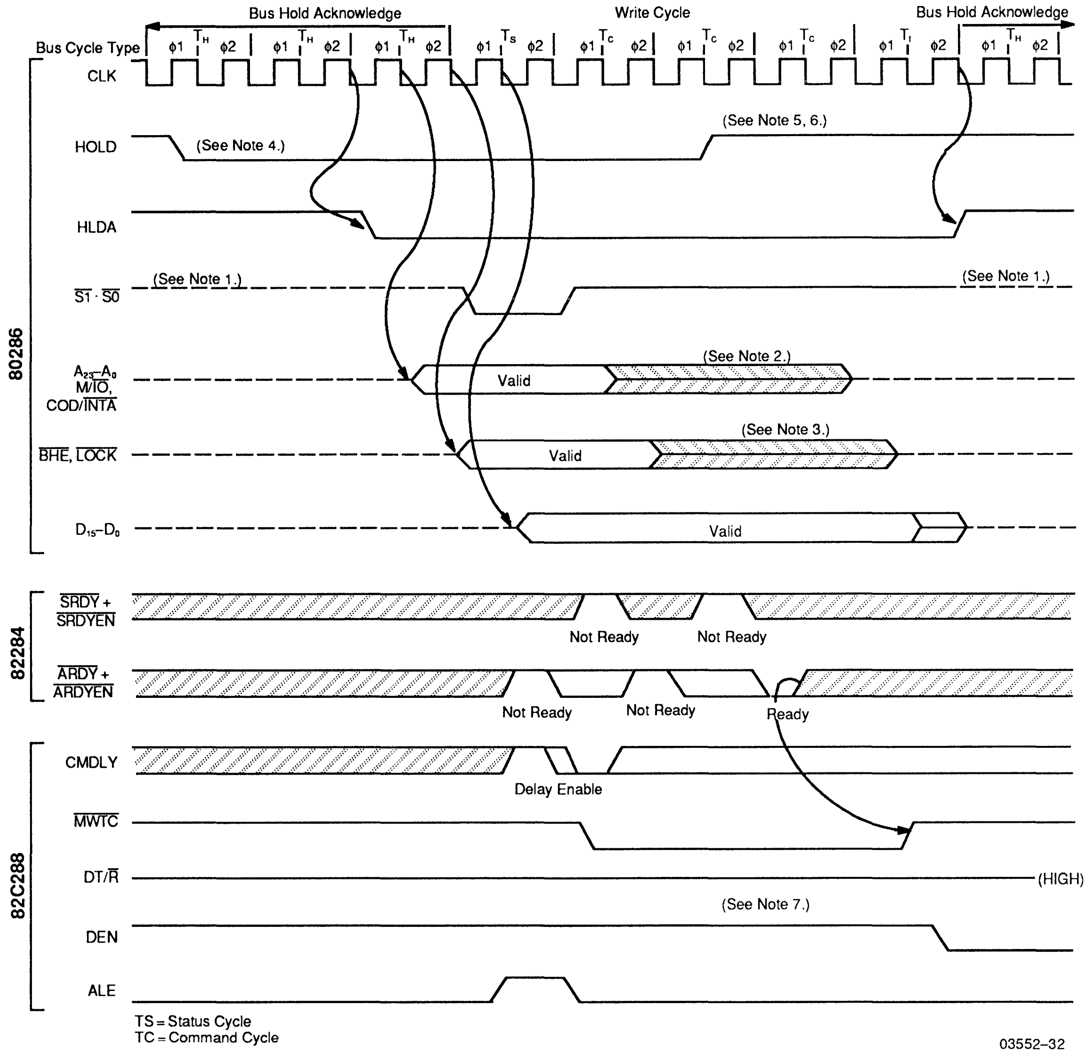


Figure 33. Back-to-Back Write-Write Cycles



Notes:
1. Status lines are not driven by 80286, yet remain high due to pull-up resistors in 82C288 and 82289 during HOLD state.
2. Address, M/IO and COD/INTA may start may start floating during any TC, depending on when internal 80286 bus arbiter decides to release bus to external HOLD. The float starts in φ2 of TC.
3. BHE and LOCK may start floating after the end of any TC, depending on when internal 80286 bus arbiter decides to release bus to external HOLD.
4. The minimum HOLD ↓ to HLDA ↓ time is shown. Maximum is one T_H longer.
5. The earliest HOLD ↑ time is shown which will always allow a subsequent memory cycle if pending.
6. The minimum HOLD ↑ to HLDA ↑ time is shown. Maximum is a function of the instruction, type of bus cycle and other machine status (i.e., Interrupts, Waits, Lock, etc.).
7. Asynchronous ready allows termination of the cycle. Synchronous ready does not signal ready in this example. Synchronous ready state is ignored after ready is signaled via the asynchronous input.

Figure 34. MULTIBUS Write Terminated by Asynchronous Ready with Bus Hold

Processor Extension Transfers

The processor extension interface uses I/O port addresses 00F8(H), and 00FA(H), and 00FC(H) which are part of the I/O port address range and is a reserved area. An ESC instruction with EM = 0 and TS = 0 will perform I/O bus operations to one or more of these I/O port addresses independent of the value of IOPL and CPL.

ESC instructions with memory references enable the CPU to accept PEREQ inputs for processor extension operand transfers. The CPU will determine the operand starting address and read/write status of the instruction. For each operand transfer, two or three bus operations, one word transfer with I/O port address 00FA(H), and one or two bus operations with memory are performed. Three bus operations are required for each word operand aligned on an odd byte address.

Interrupt Acknowledge Sequence

Figure 35 illustrates an interrupt acknowledge sequence performed by the 80286 in response to an INTR input. An interrupt acknowledge sequence consists of two INTA bus operations. The first allows a master 8259A Programmable Interrupt Controller (PIC) to determine which, if any, of its slaves should return the interrupt vector. An eight-bit vector is read by the 80286 during the second INTA bus operation to select an interrupt handler routine from the interrupt table.

The Master Cascade Enable (MCE) signal of the 82C288 is used to enable the cascade address drivers, during INTA bus operations (see Figure 35), onto the local address bus for distribution to slave interrupt controllers via the system address bus. The 80286 emits the $\overline{\text{LOCK}}$ signal (active Low) during T_S of the first INTA bus operation. A local bus "hold" request will not be honored until the end of the second INTA bus operation.

Three idle processor clocks are provided by the 80286 between INTA bus operations to allow for the minimum INTA to INTA time and CAS (cascade address) out delay of the 8259A. The second INTA bus operation must always have at least one extra T_C state added via logic controlling $\overline{\text{READY}}$. A_{23}–A_0 are in three-state OFF until after the first T_C state of the second INTA bus operation. This prevents bus contention between the cascade address drivers and CPU address drivers. The extra T_C state allows time for the 80286 to resume driving the address lines for subsequent bus operations.

Local Bus Usage Priorities

The 80286 local bus is shared among several internal units and external HOLD requests. In case of simultaneous requests, their relative priorities are:

(Highest) Any transfers which assert $\overline{\text{LOCK}}$ either explicitly (via the LOCK instruction prefix) or implicitly (i.e., segment descriptor access, interrupt acknowledge sequence, or an XCHG with memory).

The second of the two-byte bus operations required for an odd aligned word operand.

Local bus request via HOLD input.

Processor extension data operand transfer via PEREQ input.

Data transfer performed by EU as part of an instruction.

(Lowest) An instruction prefetch request from BU. The EU will inhibit prefetching two processor clocks in advance of any data transfers to minimize waiting by EU for a prefetch to finish.

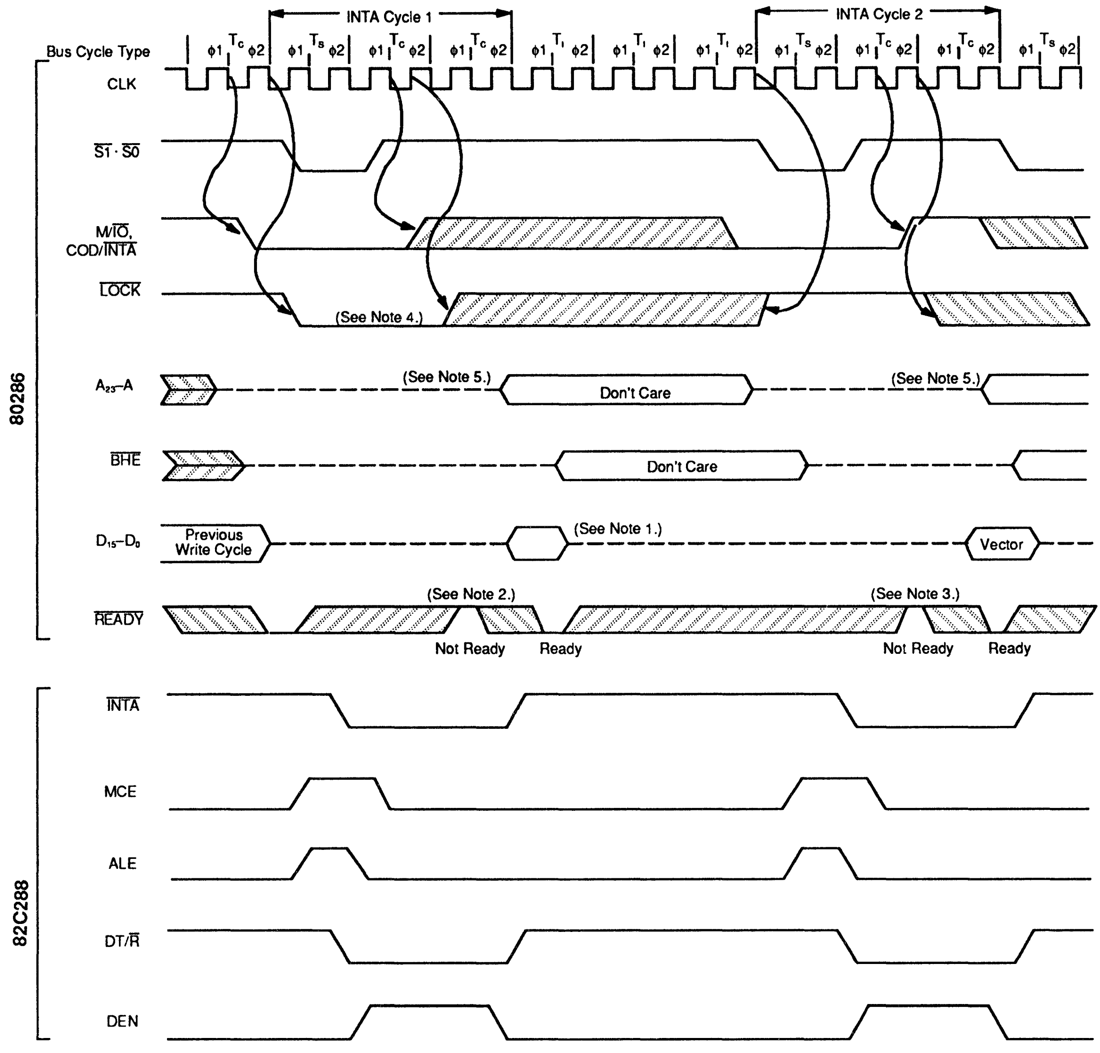
Halt or Shutdown Cycles

The 80286 externally indicates halt or shutdown conditions as a bus operation. These conditions occur due to a HLT instruction or multiple protection exceptions while attempting to execute one instruction. A halt or shutdown bus operation is signalled when $\overline{\text{S1}}$, $\overline{\text{S0}}$ and COD/$\overline{\text{INTA}}$ are Low and M/$\overline{\text{IO}}$ is High. A_1 High indicates halt, and A_1 Low indicates shutdown. The 82C288 bus controller does not issue ALE, nor is $\overline{\text{READY}}$ required to terminate a halt or shutdown bus operation.

During halt or shutdown, the 80286 may service PEREQ or HOLD requests. A processor extension segment overrun exception during shutdown will inhibit further service of PEREQ. Either NMI or RESET will force the 80286 out of either halt or shutdown. An INTR, if interrupts are enabled, or a processor extension segment overrun exception will also force the 80286 out of halt.

System Configurations

The versatile bus structure of the 80286 microsystem, with a full complement of support chips, allows flexible configuration of a wide range of systems. The basic configuration, shown in Figure 36, is similar to an iAPX 86 maximum mode system. It includes the CPU plus an 8259A interrupt controller, 82284 clock generator, and the 82C288 Bus Controller. The iAPX 86 latches (29843 and 29845) and transceivers (29833 and 29863) may be used in an 80286 microsystem.



03552-33

Notes:

1. Data is ignored.
2. First INTA cycle should have at least one wait state inserted to meet 8259A minimum INTA pulse width.
3. Second INTA cycle must have at least one wait state inserted since the CPU will not drive A_{23}–A_0, $\overline{BHE}$, and $\overline{LOCK}$ until after the first T_C state.
   The CPU imposed one/clock delay prevents bus contention between cascade address buffer being disabled by MCE ↓ and address outputs.
   Without the wait state, the 80286 address will not be valid for a memory cycle started immediately after the second INTA cycle. The 8259A also requires one wait state for minimum INTA pulse width.
4. $\overline{LOCK}$ is active for the first INTA cycle to prevent the 82289 from releasing the bus between INTA cycles in a multi-master system.
5. A_{23}–A_0 exits three-state OFF during φ2 of the second T_C in the INTA cycle.

Figure 35. Interrupt Acknowledge Sequence

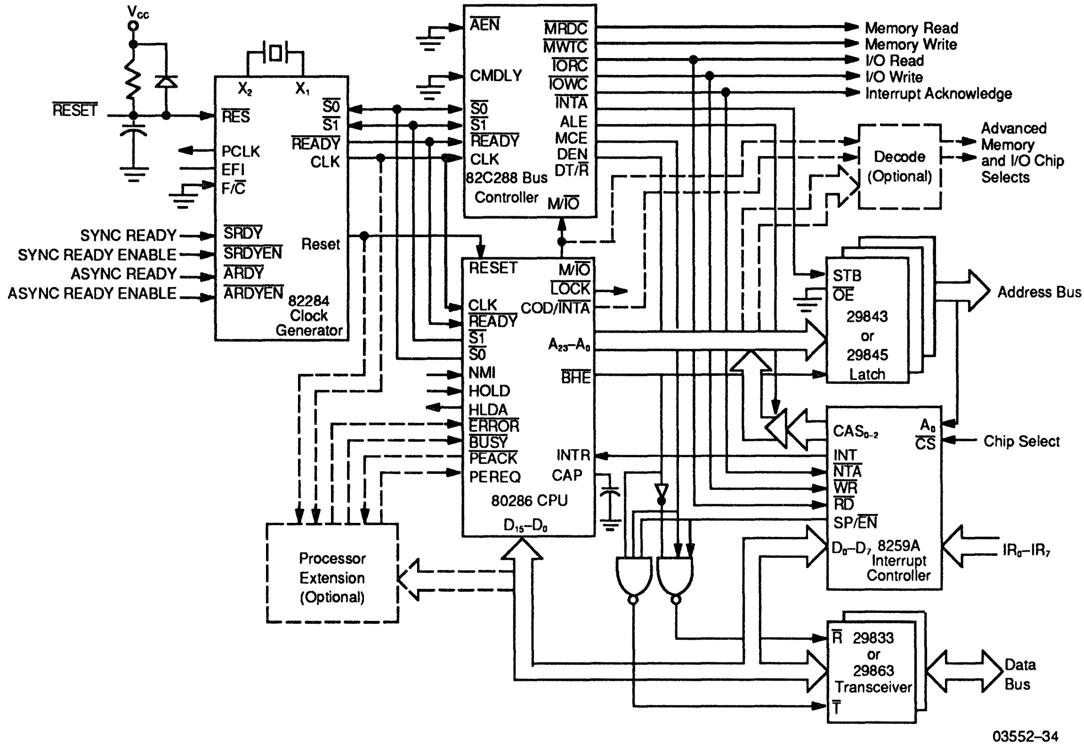


Figure 36. Basic 80286 System Configuration

As indicated by the dashed lines in Figure 36, the ability to add processor extensions is an integral feature of 80286 microsystems. The processor extension interface allows external hardware to perform special functions and transfer data concurrent with CPU execution of other instructions. Full system integrity is maintained because the 80286 supervises all data transfers and instruction execution for the processor extension.

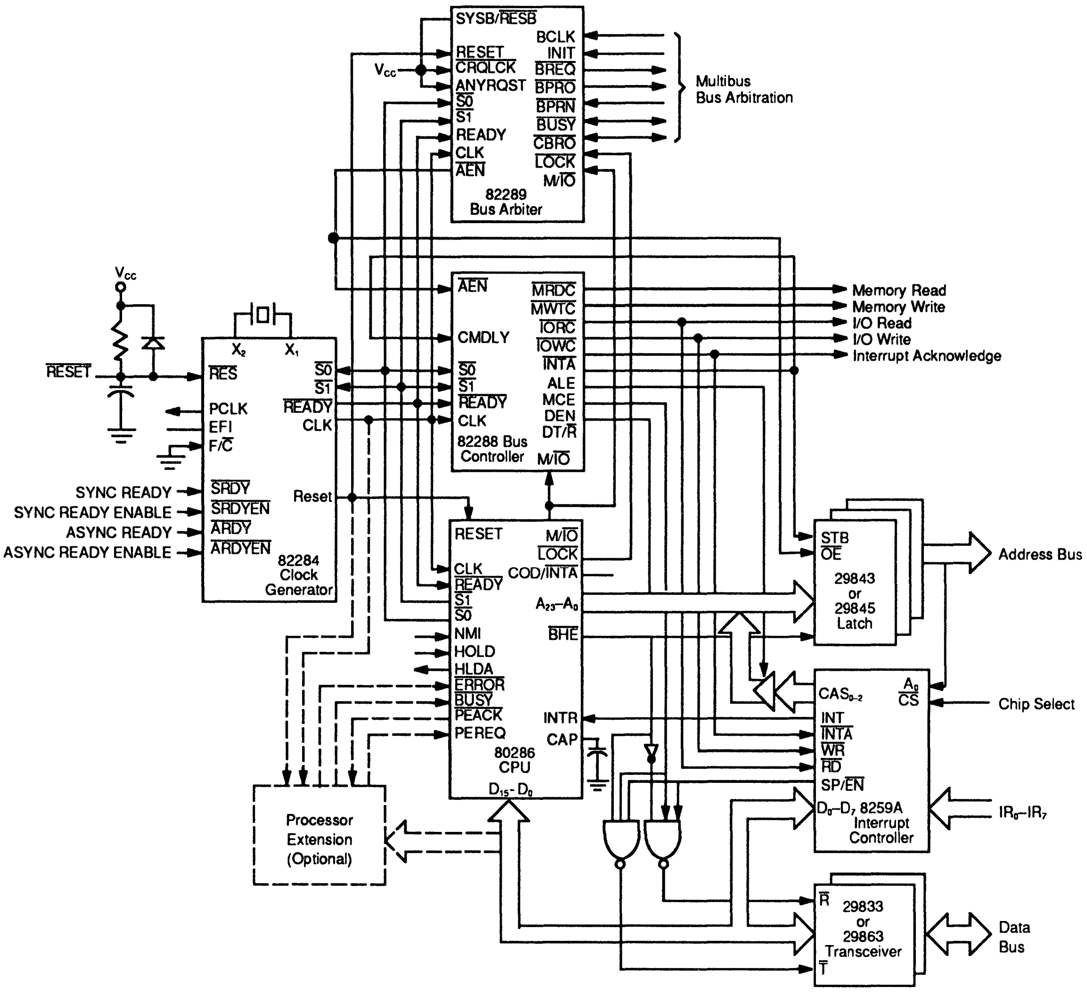
The 80286 with the 80287 numeric processor extension (NPX) uses this interface. The iAPX 286/287 has all the instructions and data types of an iAPX 86/87 or iAPX 88/87. The 80287 NPX can perform numeric calculations and data transfers concurrently with CPU program execution. Numerics code and data have the same integrity as all other information protected by the 80286 protection mechanism.

The 80286 can overlap chip select decoding and address propagation during the data transfer for the previous bus operation. This information is latched into the 29843/45's by ALE during the middle of a Ts cycle. The latched chip select and address information remains stable during the bus operation while the next cycle's address is being decoded and propagated into the system. Decode logic can be implemented with a high-speed bipolar PROM.

The optional decode logic shown in Figure 36 takes advantage of the overlap between address and data of the 80286 bus cycle to generate advanced memory and I/O-select signals. This minimizes system performance degradation caused by address propagation and decode delays. In addition to selecting memory and I/O, the advanced selects may be used with configurations supporting local and system buses to enable the appropriate bus interface for each bus cycle. The COD/INTA and M/IO signals are applied to the decode logic to distinguish between interrupt, I/O, code and data bus cycles.

By adding the 82289 bus arbiter chip, the 80286 provides a MULTIBUS system bus interface as shown in Figure 37. The ALE output of the 82C288 for the MULTIBUS bus is connected to its CMDLY input to delay the start of commands one system CLK as required to meet MULTIBUS address and write data set-up times. This arrangement will add at least one extra Tc state to each bus operation which uses the MULTIBUS.

A second 82C288 bus controller and additional latches and transceivers could be added to the local bus of Figure 37. This configuration allows the 80286 to



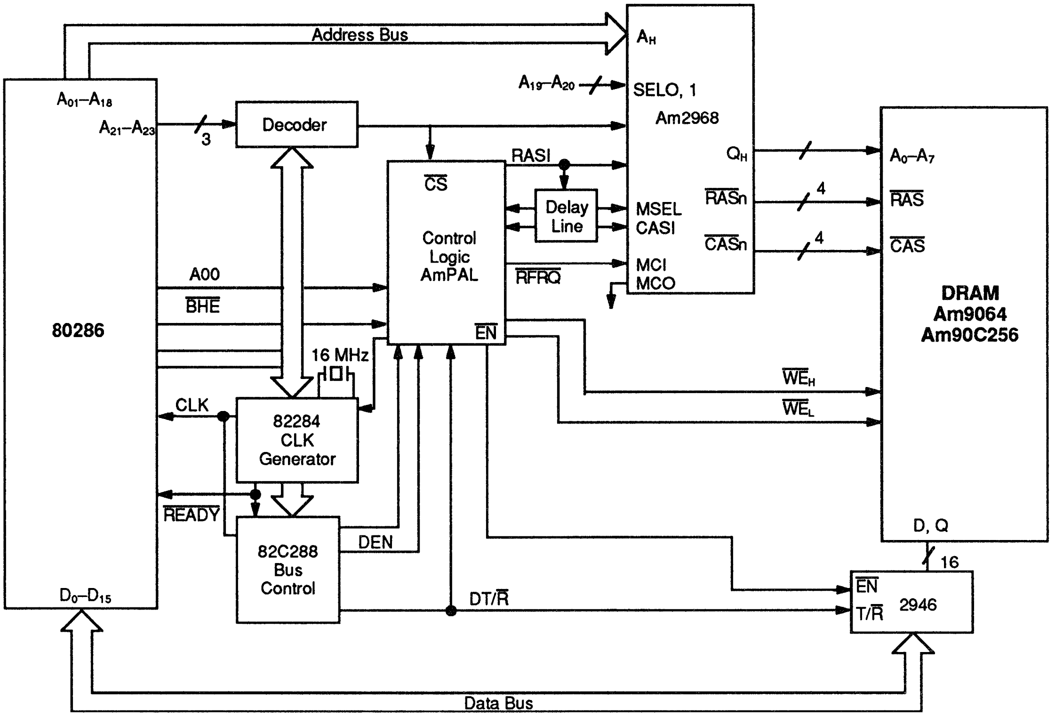
03552-35

Figure 37. MULTIBUS System Bus Interface

support an on-board bus for local memory and peripherals and the MULTIBUS for system bus interfacing.

Figure 38 shows the interface of the 80286 with the Am2968 Dynamic Memory Controller. The interface is a timing controller which consists of some control logic and a delay line. The timing controller runs asynchronously to the CPU. It arbitrates between memory requests and refresh requests by generating the proper signals to the dynamic memory controller and memory. The design described is a simple, cost-effective solution to interfacing the 80286 with the Am2968. A further description about DRAM selection based on processor speed may be found in the Am2968 Application Note.

Two-operand instructions (e.g., MOV and ADD) are usually three to six bytes long. Memory-to-memory operations are provided by a special class of string instructions requiring one to three bytes.

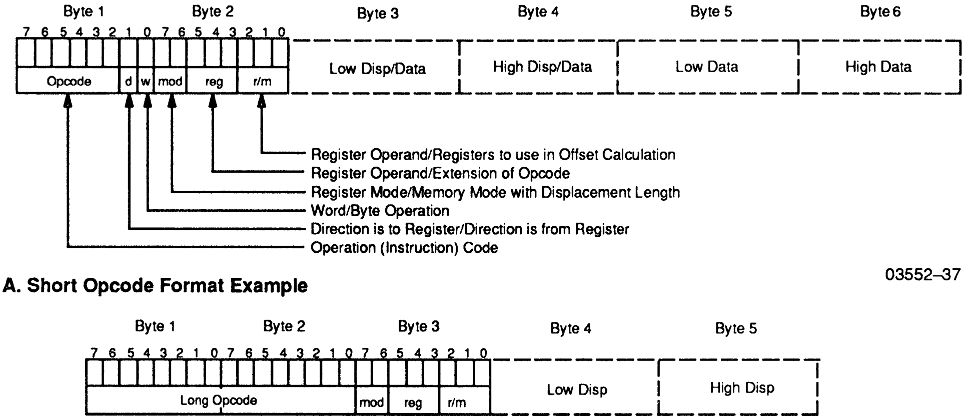


03552–36

Figure 38. 80286 Interface with the Am2968 Dynamic Memory Controller

Table 15. 80286 Systems Recommended Pull-up Resistor Values

80286 Pin and Name	Pull-up Value	Purpose
4–$\overline{\text{S1}}$	20KΩ ± 10%	Pull $\overline{\text{S0}}$, $\overline{\text{S1}}$, and $\overline{\text{PEACK}}$ inactive during 80286 hold periods.
5–$\overline{\text{S0}}$		
6–$\overline{\text{PEACK}}$		
53–$\overline{\text{ERROR}}$	20KΩ ± 10%	Pull $\overline{\text{ERROR}}$ and $\overline{\text{BUSY}}$ inactive when 80287 not present (or temporarily removed from socket).
54–$\overline{\text{BUSY}}$		
63–$\overline{\text{READY}}$	910Ω ± 5%	Pull $\overline{\text{READY}}$ inactive within required minimum time (C_L = 150 pF, I_R ≤ 7mA).



A. Short Opcode Format Example

03552–37

B. Long Opcode Format Example

03552–38

Figure 39. 80286 Instruction Format Examples

80286 INSTRUCTION SET SUMMARY

Instruction Timing Notes

The instruction clock counts listed below establish the maximum execution rate of the 80286. With no delays in bus cycles, the actual clock count of an 80286 program will average 5% more than the calculated clock count, due to instruction sequences which execute faster than they can be fetched from memory.

To calculate elapsed times for instruction sequences, multiply the sum of all instruction clock counts, as listed in the table below, by the processor clock period. An 8-MHz processor clock has a clock period of 125 nanoseconds and requires an 80286 system clock (CLK input) of 16 MHz.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution. Control transfer instruction clock counts include all time required to fetch, decode, and prepare the next instruction for execution.
2. Bus cycles do not require wait states.
3. There are no processor extension data transfer or local bus HOLD requests.
4. No exceptions occur during instruction execution.

Instruction Set Summary Notes

Addressing displacements selected by the MOD field are not shown. If necessary they appear after the instruction fields shown.

Above/below refers to unsigned value

Greater refers to positive signed value

Less refers to less positive (more negative) signed values

if d = 1 then to register; if d = 0 then from register

if w = 1 then word instruction; if w = 0 then byte instruction

if s = 0 then 16-bit immediate data to form the operand

if s = 0 then an immediate data byte is sign-extended to form the 16-bit operand

x = don't care

z = used for string primitives for comparison with ZF FLAG

If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.

* = add one clock if offset calculation requires summing 3 elements

n = number of times repeated

m = number of bytes of code in next instruction

Level (L)—Lexical nesting level of the procedure

The following comments describe possible exceptions, side effects, and allowed usage for instructions in both operating modes of the 80286.

Real Address Mode Only

1. This is a protected mode instruction. Attempted execution in real address mode will result in an undefined op-code exception (6).
2. A segment overrun exception (13) will occur if a word operand reference at offset FFFF(H) is attempted.
3. This instruction may be executed in real address mode to initialize the CPU for protected mode.
4. The IOPL and NT fields will remain 0.
5. Processor extension segment overrun interrupt (9) will occur if the operand exceeds the segment limit.

Either Mode

6. An exception may occur, depending on the value of the operand.
7. $\overline{\text{LOCK}}$ is automatically asserted regardless of the presence or absence of the LOCK instruction prefix.
8. $\overline{\text{LOCK}}$ does not remain active between all operand transfers.

Protected Virtual Address Mode Only

9. A general protection exception (13) will occur if the memory operand cannot be used due to either a segment limit or access rights violation. If a stack segment limit is violated, a stack segment overrun exception (12) occurs.
10. For segment load operations, the CPL, RPL, and DPL must agree with privilege rules to avoid an exception. The segment must be present to avoid a not-present exception (11). If the SS register is the destination, and a segment-not-present violation occurs, a stack exception (12) occurs.
11. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert $\overline{\text{LOCK}}$ to maintain descriptor integrity in multiprocessor systems.
12. JMP, CALL, INT, RET, IRET instructions referring to another code segment will cause a general protection exception (13) if any privilege rule is violated.
13. A general protection exception (13) occurs if CPL≠0.
14. A general protection exception (13) occurs if CPL > IOPL.
15. The IF field of the flag word is not updated if CPL > IOPL. The IOPL field is updated only if CPL = 0.
16. Any violation of privilege rules as applied to the selector operand do not cause a protection exception; rather, the instruction does not return a result and the zero flag is cleared.
17. If the starting address of the memory operand violates a segment limit, or an invalid access is attempted, a general protection exception (13) will occur before the ESC instruction is executed. A stack segment overrun exception (12) will occur if the stack limit is violated by the operand's starting address. If a segment limit is violated during an attempted data transfer, then a processor extension segment overrun exception (9) occurs.
18. The destination of an INT, JMP, CALL, RET, or IRET instruction must be in the defined limit of a code segment or a general protection exception (13) will occur.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature .... –65 to + 150° C
Voltage on Any Pin with
Respect to Ground .... –1.0 to + 7.0 V
Power Dissipation .... 3.15 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices
Temperature (T_C) .... 0 to +85° C
Supply Voltage (V_{CC}) .... 5 V±5%

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS

(V_{CC} = 5 V ±5%, T_{CASE} = 0 to + 85° C)

Parameter	Description	Test Conditions	Min	Max	Unit
V_{IL}	Input Low Voltage		–.5	8	V
V_{IH}	Input High Voltage		2.0	V_{CC} +0 .5	V
V_{ILC}	CLK Input Low Voltage		–.5	.6	V
V_{IHC}	CLK Input High Voltage		3.8	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	I_{OL} = 2.0 mA		0.45	V
V_{OH}	Output High Voltage	I_{OH} = –400 µA	2.4		V
I_{LI}	Input Leakage Current	0 V ≤ V_{IN} ≤ V_{CC}		±10	µA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		±10	µA
I_{CC}	Supply Current (turn on, 0°C)	Note 1		600	mA
C_{CLK}	CLK Input Capacitance	F_C = 1 MHz		20	pF
C_{IN}	Other Input Capacitance	F_C =1 MHz		10	pF
C_O	Input /Output Capacitance	F_C = 1 MHz		20	pF
I_{LO}	Output Leakage Current	0 V ≤ V_{OUT} ≤ 0.45 V		±1	mA
I_{IL}	Input Sustaining Current On $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ pins	V_{IN} = 0 V	30	500	µA
I_{LCR}	Input CLK Leakage Current	0.45 ≤ V_{IN} ≤ V_{CC}		±10	µA
I_{LCR}	Input CLK Leakage Current	0 V ≤ V_{IN} ≤ 0.45 V		+1	mA

Note: Low temperature is worst case.

SWITCHING CHARACTERISTICS

V_{CC} = +5 V ± 5%, T_{CASE} = 0° to +85° C

AC Timings are referenced to 0.8 V and 2.0 V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

			8 MHz		10 MHz		
Parameters	**Description**	**Test Conditions**	**Min.**	**Max.**	**Min.**	**Max.**	**Unit**
1	System Clock (CLK) Period		62	125	50	125	ns
2	System Clock (CLK) Low Time	at 1.0 V	15	100	12	109	ns
3	System Clock (CLK) High Time	at 3.6 V	25	110	16	113	ns
17	System Clock (CLK) Rise Time	1.0 V to 3.6 V		10		8	ns
18	System Clock (CLK) Fall Time	3.6 V to 1.0 V		10		8	ns
4	Asynchronous Inputs Setup Time	Note 1	20		20		ns
5	Asynchronous Inputs Hold Time	Note 1	20		20		ns
6	RESET Setup Time		28		23		ns
7	RESET Hold Time		5		5		ns
8	Read Data Setup Time		10		8		ns
9	Read Data Hold Time		8		8		ns
10	$\overline{\text{READY}}$ Setup Time		38		26		ns
11	$\overline{\text{READY}}$ Hold Time		25		25		ns
12	Status/$\overline{\text{PEACK}}$ Valid Delay	Note 2, Note 3	1	40	–	–	ns
12a	Status/$\overline{\text{PEACK}}$ Active Delay	Note 2, Note 3	–	–	1	22	ns
12b	Status/$\overline{\text{PEACK}}$ Inactive Delay	Note 2, Note 3	–	–	1	30	ns
13	Address Valid Delay	Note 2, Note 3	1	60	1	35	ns
14	Write Data Valid Delay	Note 2, Note 3	0	50	0	30	ns
15	Address/Status/Data Float Delay	Note 2, Note 4	0	50	0	47	ns
16	HLDA Valid Delay	Note 2, Note 3	0	50	0	47	ns
19	Address Valid to Status Valid Setup Time	Note 3, Note 5, Note 6	38		27		ns

Notes: 1. Asynchronous inputs are INTR, NMI, HOLD PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.

2. Delay from 1.0 V on the CLK to 0.8 V or 2.0 V or float on the output as appropriate for valid or floating condition.

3. Output load: C_L = 100 pF.

4. Float condition occurs when output current is less than I_{LO} in magnitude.

5. Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 2.0 V or status going inactive reaching 0.8 V.

6. For load capacitance of 10 pF on STATUS/$\overline{\text{PEACK}}$ lines, subtract typically 7 ns for 8 MHz spec, and maximum 7 ns for 10 MHz spec.

SWITCHING CHARACTERISTICS (continued)

V_{CC} = +5 V ± 5%, T_{CASE} = 0° to +85° C
AC Timings are referenced to 0.8 V and 2.0 V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Parameters	Description	Test Conditions	12.5 MHz		16 MHz		Unit
			Min.	Max.	Min.	Max.	
1	System Clock (CLK) Period		40	125	31	125	ns
2	System Clock (CLK) Low Time	at 1.0 V	11	112	10	113	ns
3	System Clock (CLK) High Time	at 3.6 V	13	114	12	115	ns
17	System Clock (CLK) Rise Time	1.0 V to 3.6 V		8	8		ns
18	System Clock (CLK) Fall Time	3.6 V to 1.0 V		8	8		ns
4	Asynchronous Inputs Setup Time	Note 1	15		11		ns
5	Asynchronous Inputs Hold Time	Note 1	15		11		ns
6	RESET Setup Time		18		14		ns
7	RESET Hold Time		5		3		ns
8	Read Data Setup Time		5		5		ns
9	Read Data Hold Time		6		5		ns
10	$\overline{\text{READY}}$ Setup Time		22		15		ns
11	$\overline{\text{READY}}$ Hold Time		20				ns
12	Status/$\overline{\text{PEACK}}$ Valid Delay	Note 2, Note 3	–	–	1	18	ns
12a	Status/$\overline{\text{PEACK}}$ Active Delay	Note 2, Note 3	3	18	1	18	ns
12b	Status/$\overline{\text{PEACK}}$ Inactive Delay	Note 2, Note 3	3	20	1	20	ns
13	Address Valid Delay	Note 2, Note 3	1	32	1	29	ns
14	Write Data Valid Delay	Note 2, Note 3	0	30	0	22	ns
15	Address/Status/Data Float Delay	Note 2, Note 4	0	32	0	29	ns
16	HLDA Valid Delay	Note 2, Note 3	0	25	0	25	ns
19	Address Valid to Status Valid Setup Time	Note 3, Note 5, Note 6	22		22		ns

Notes: 1. Asynchronous inputs are INTR, NMI, HOLD PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.

2. Delay from 1.0 V on the CLK to 0.8 V or 2.0 V or float on the output as appropriate for valid or floating condition.

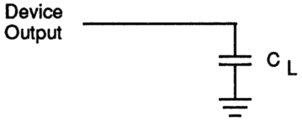
3. Output load: C_L = 100 pF.

4. Float condition occurs when output current is less than I_{LO} in magnitude.

5. Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 2.0 V or status going inactive reaching 0.8 V.

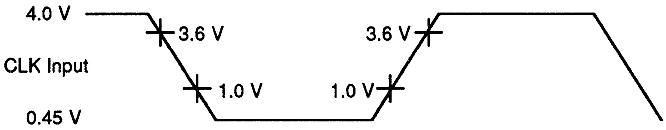
6. For load capacitance of 10 pF on STATUS/$\overline{\text{PEACK}}$ lines, subtract typically 7 ns for 8 MHz spec, and maximum 7 ns for 10 MHz spec.

AC Test Loading on Outputs



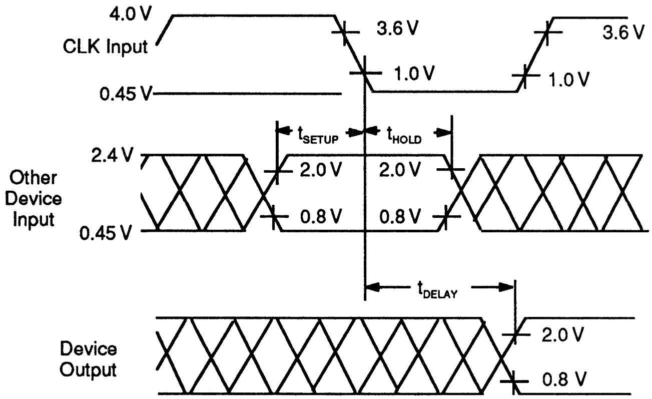
03552–39

AC Drive and Measurement Points–CLK Input



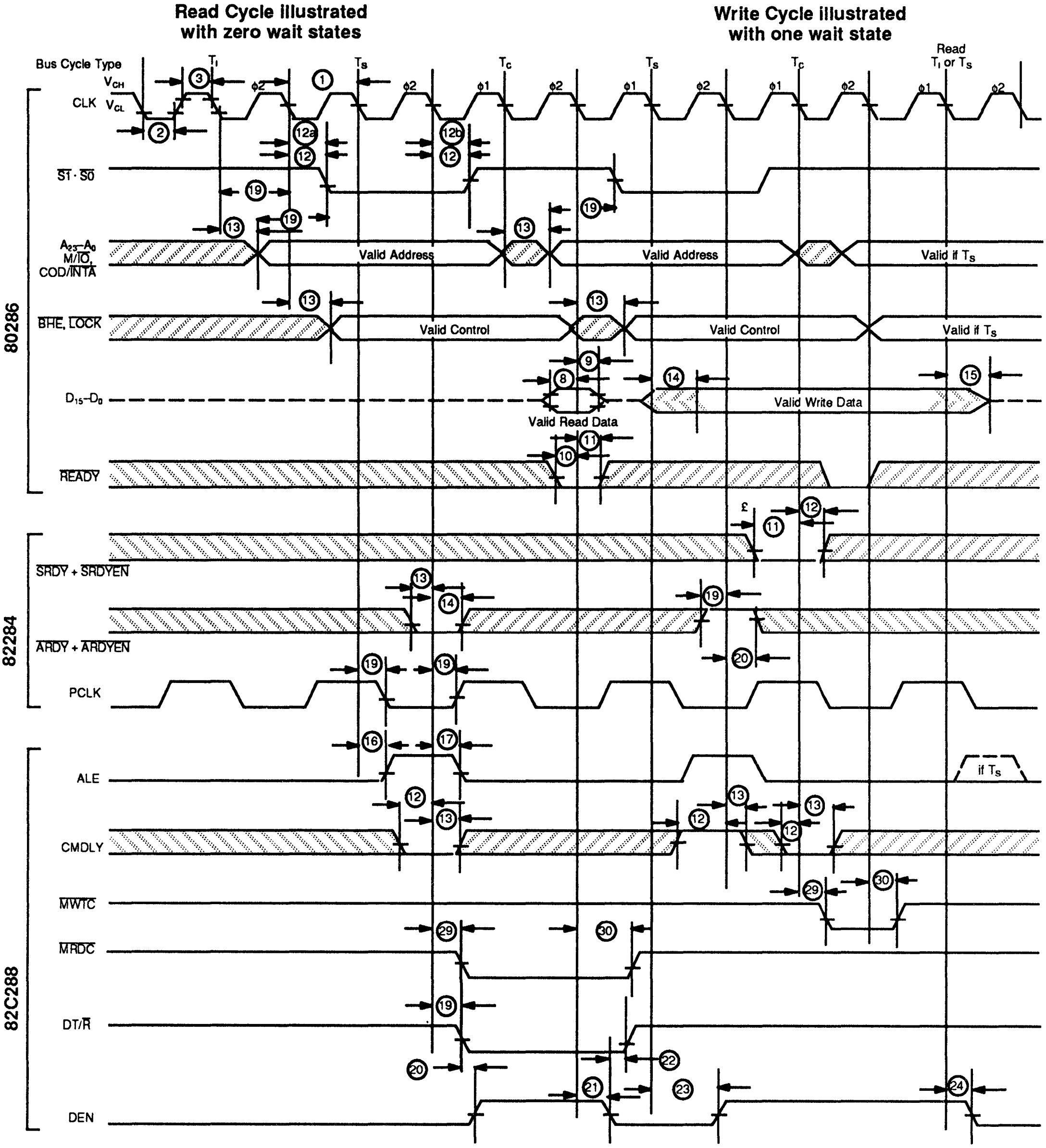
03552–40

AC Setup, Hold and Delay Time Measurement—General



03552–41

SWITCHING WAVEFORMS
Major Cycle Timing

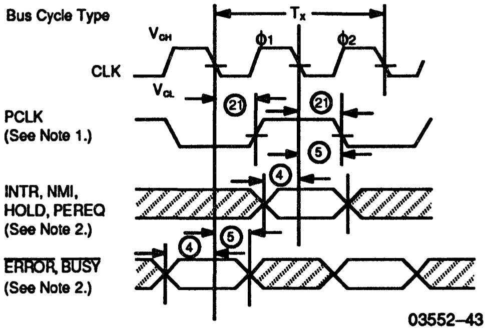


Note: The modified timing is due to the $\overline{\text{CMDLY}}$ signal being active.

03552–42

SWITCHING WAVEFORMS (continued)

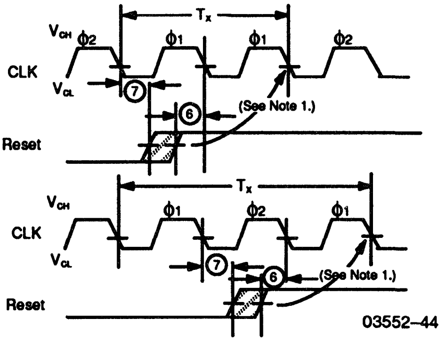
80286 Asynchronous Input Signal Timing



Notes: 1. PCLK indicates which processor cycle phase will occur on the next CLK. PCLK may not indicate the correct phase until the first bus cycle is performed.

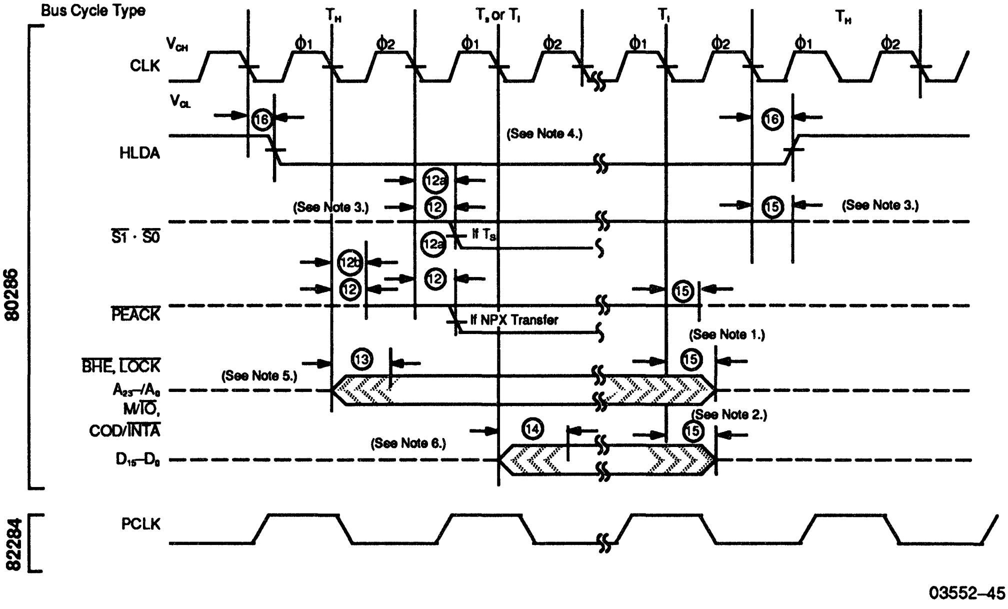
2. These inputs are asynchronous. The setup and hold times shown assure recognition for testing purposes.

80286 Reset Input Timing and Subsequent Processor Cycle Phase



Note: When RESET meets the set-up time shown, the next CLK will start or repeat φ1 of a processor cycle.

Exiting and Entering Hold



Notes: 1. These signals may not be driven by the 80286 during the time shown. The worst case in terms of latest float time is shown.

2. The data bus will be driven as shown if the last cycle before T_I in the diagram was a write T_C.

3. The 80286 floats its status pins during T_H. External 20 kΩ resistors keep these signals high (see Table 15).

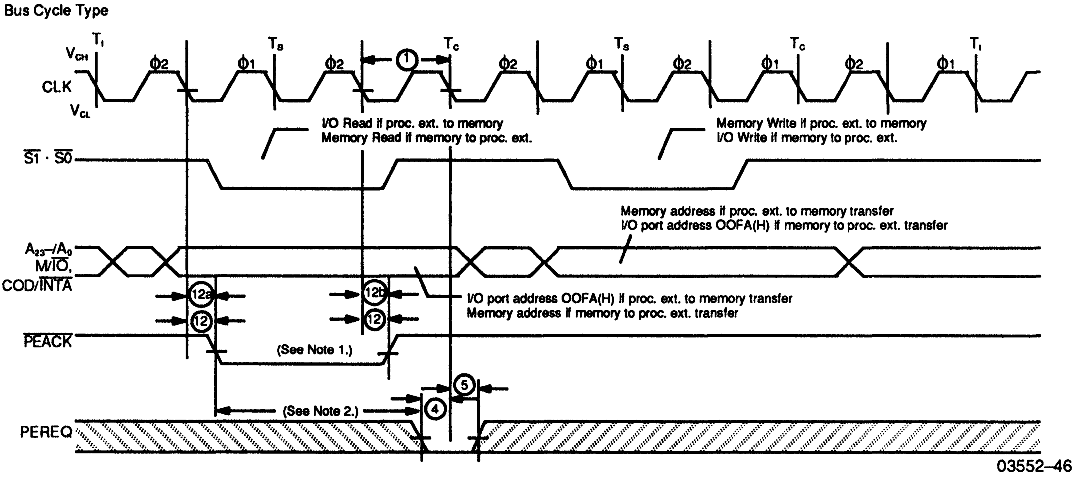
4. For HOLD request set-up to HLDA, refer to Figure 34.

5. BHE and LOCK are driven at this time but will not become valid until T_S.

6. The data bus will remain in three-state OFF if a read cycle is performed.

SWITCHING WAVEFORMS (continued)

80286 PEREQ/$\overline{\text{PEACK}}$ Timing Required PEREQ Timing for One Transfer Only


Bus Cycle Type
T_I
T_S
T_C
T_S
T_C
T_I
V_CH
CLK
V_CL
φ2
φ1
φ2
①
φ2
φ1
φ2
φ1
φ2
φ1
I/O Read if proc. ext. to memory
Memory Read if memory to proc. ext.
Memory Write if proc. ext. to memory
I/O Write if memory to proc. ext.
S̅1̅ · S̅0̅
Memory address if proc. ext. to memory transfer
I/O port address OOFA(H) if memory to proc. ext. transfer
A23–A0
M/I̅O̅,
COD/I̅N̅T̅A̅
12a
12
12b
12
I/O port address OOFA(H) if proc. ext. to memory transfer
Memory address if memory to proc. ext. transfer
P̅E̅A̅C̅K̅
(See Note 1.)
5
(See Note 2.)
4
PEREQ

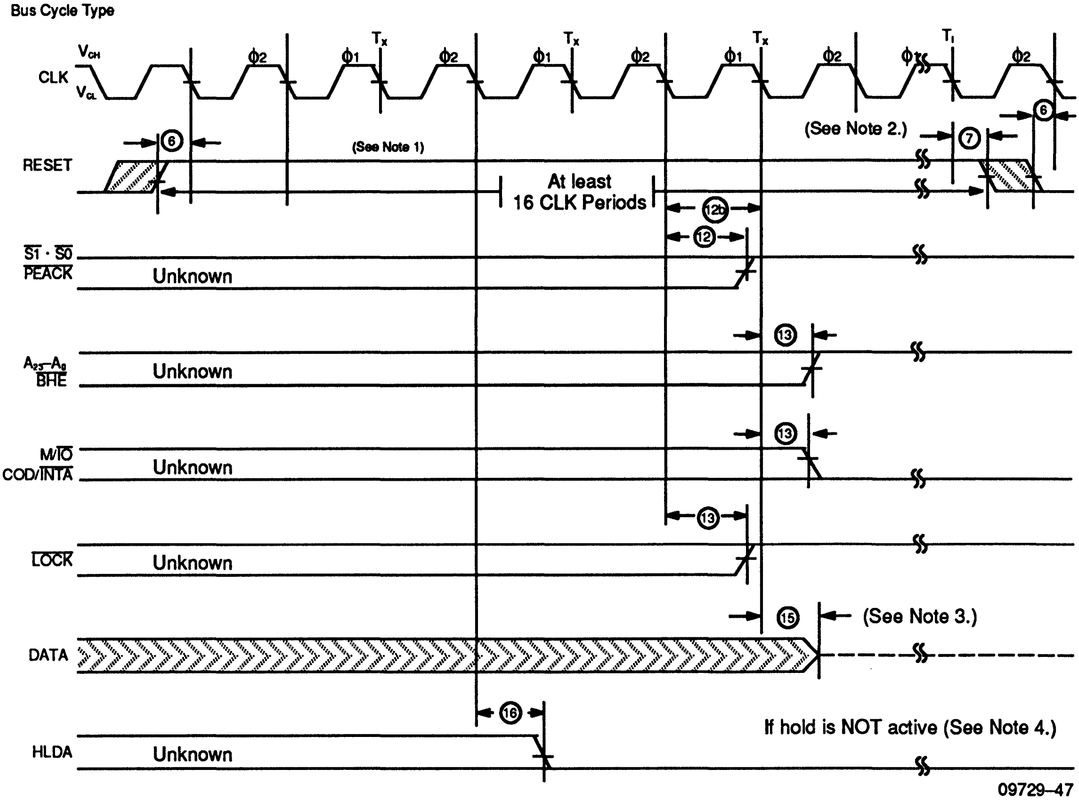
03552–46

Assuming word-aligned memory operand; if odd-aligned, 80286 transfers to/from memory byte-at-a-time with two memory cycles.

Notes: 1. $\overline{\text{PEACK}}$ always goes active during the first bus operation of a processor extension data operand transfer sequence. The first bus operation will be either a memory read at operand address or I/O read at port address 00FA(H).

2. To prevent a second processor extension data operand transfer, the worst case maximum time (shown above) is: 3 x 1 – 11 max – 4 min. The actual, configuration dependent, maximum time is: 3 x 1 – 11 max – 4 min + A x 2 x 1. A is the number of extra T_C states added to either the first or second bus operation of the processor extension data operand transfer sequence.

Initial 80286 Pin State During Reset



Notes: 1. Set-up time for RESET ↑ may be violated with the consideration that φ1 of the processor clock may begin one system CLK period later.

2. Set-up and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during φ1 or φ2.
3. The data bus is only guaranteed to be in three-state OFF at the time shown.
4. HOLD is acknowledged during RESET, causing HLDA to go active and the appropriate pins to float. If HOLD remains active while RESET goes inactive, the 80286 remains in HOLD state and will not perform any bus accesses until HOLD is deactivated.

80286 INSTRUCTION SET SUMMARY

Function	Format				Clock Count		Comments	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
DATA TRANSFER								
MOV = Move:								
Register to Register/Memory	1000100w	mod reg r/m			2,3*	2,3*	2	9
Register/Memory to Register	1000101w	mod reg r/m			2,5*	2,5*	2	9
Immediate to register/ Memory	1100011w	mod000r/m	data	data if w=1	2,3*	2,3*	2	9
Immediate to register	1011wreg	data	data if w = 1		2	2		
Memory to accumulator	1010000w	addr-low	addr-high		5	5	2	9
Accumulator to memory	1010001w	addr-low	addr-high		3	3	2	9
Register/memory to segment register	10001110	mod0reg r/m			2,5*	17,19*	2	9,10,11
Segment register to register/memory	10001100	mod0reg r/m			2,3*	2,3*	2	9
PUSH = Push:								
Memory	11111111	mod110r/m			5*	5*	2	9
Register	01010reg				3	3	2	9
Segment register	000reg110				3	3	2	9
Immediate*	0110100	data	data if s = 0		3	3	2	9
PUSHA = Push All*	01100000				17	17	2	9
POP = Pop								
Memory	10001111	mod000r/m			5*	5*	2	9
Register	01011reg				5	5	2	9
Segment register	000reg111	(reg≠01)			5	20	2	9,10,11
POPA = Pop All*	01100001				19	19	2	9
XCHG = Exchange:								
Register/memory with register	1000011w	modreg r/m			3,5*	3,5*	2,7	7,9
Register with accumulator	10010reg				3	3		
IN = Input from:								
Fixed port	1110010w	port			5	5		14
Variable port	1110110w				5	5		14
OUT = Output to:								
Fixed port	1110011w	port			3	3		14
Variable port	1110111w				3	3		14
XLAT = Translate byte to AL	11010111				5	5		9
LEA = Load EA to register	10001101	modreg r/m			3*	3*		
LDS = Load pointer to DS	11000101	mod reg r/m	(mod ≠ 11)		7*	21*	2	9,10,11
LES = Load pointer to ES	11000100	mod reg r/m	(mod ≠ 11)		7*	21*	2	9,10,11
LAHF = Load AH with flags	10011111				2	2		
SAHF = Store AH into flags	10011110				2	2		
PUSHF = Push flags	10011100				3	3	2	9
POPF = Pop flags	10011101				5	5	2,4	9,15

*Indicates instructions not available in iAPX 86, 88 microsystems.

See footnotes on page 1-124.

80286 INSTRUCTION SET SUMMARY (continued)

Function	Format				Clock Count		Comments	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC								
ADD=Add:								
Reg/memory with register to either	0 0 0 0 0 0 d w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 r/m	data	data if s:w = 01	3, 7*	3, 7*	2	9
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w = 1		3	3		
ADC = Add with carry:								
Reg/memory with register to either	0 0 0 1 0 0 d w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w = 01	3, 7*	3, 7*	2	9
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w = 1		3	3		
INC =Increment:								
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m			2, 7*	2, 7*	2	9
Register	0 1 0 0 0 reg				2	2		
SUB =Subtract:								
Reg/memory and register to either	0 0 1 0 1 0 d w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w = 1	3, 7*	3, 7*	2	9
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w = 1		3	3		
SBB = Subtract with borrow:								
Reg/memory and register to either	0 0 0 1 1 0 d w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w = 01	3, 7*	3, 7*	2	9
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w = 1		3	3		
DEC = Decrement:								
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m			2, 7*	2, 7*	2	9
Register	0 1 0 0 1 reg				2	2		
CMP = Compare:								
Register/memory with register	0 0 1 1 1 0 1 w	mod reg r/m			2, 6*	2, 6*	2	9
Register with register/memory	0 0 1 1 1 0 0 w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w = 01	3, 6*	3, 6*	2	9
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w = 1		3	3		
NEG = Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m			2	7*	2	7
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1				3	3		
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1				3	3		
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1				3	3		
DAS = Decimal adjust for subtract	0 0 1 0 1 1 1 1				3	3		
MUL = Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m						
Register-Byte					13	13		
Register-Word					21	21		
Memory-Byte					16*	16*	2	9
Memory-Word					24*	24*	2	9
IMUL = Integer multiply (signed)*	1 1 1 1 0 1 1 w	mod 1 0 1 r/m						
Register-Byte					13	13		
Register-Word					21	21		
Memory-Byte					16*	16*	2	9
Memory-Word					24*	24*	2	9

*Indicates instructions not available in iAPX 86, 88 microsystems.
See footnotes on page 1-124.

80286 INSTRUCTION SET SUMMARY (continued)

Function	Format				Clock Count		Comments	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC (Continued)								
IMUL = Integer immediate multiply: (signed)*	0 1 1 0 1 0 a 1	mod reg r/m	data	data if a = 0	21, 24*	21, 24*	2	9
DIV = Divide (unsigned):	1 1 1 1 0 1 1 w	mod 1 1 0 r/m						
Register-Byte					14	14		
Register-Word					22	22		
Memory-Byte					17*	17*	2, 6	6, 9
Memory-Word					25*	25*	2, 6	6, 9
IDIV = Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m						
Register-Byte					17	17		
Register-Word					25	25		
Memory-Byte					20*	20*	2	9
Memory-Word					28*	28*	2	9
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0			16	16		
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0				14	14	
CBW = Convert byte to word	1 0 0 1 1 0 0 0					2	2	
CWD = Convert word to double word	1 0 0 1 1 0 0 1					2	2	
LOGIC **Shift/Rotate Instructions:**								
Register/Memory by 1	1 1 0 1 0 0 0 w	mod TTT r/m			2, 7*	2, 7*	2	9
Register/Memory by CL	1 1 0 1 0 0 1 w	mod TTT r/m			5+n, 8+n*	5+n, 8+n*	2	9
Register Memory by Count*	1 1 0 0 0 0 0 w	mod TTT r/m			5+n, 8+n*	5+n, 8+n*	2	9
	TTT	**Instruction**						
	0 0 0	ROL						
	0 0 1	ROR						
	0 1 0	RCL						
	0 1 1	RCR						
	1 0 0	SHL/SAL						
	1 0 1	SHR						
	1 1 1	SAR						
AND = And:								
Reg/memory and register to either	0 0 1 0 0 0 d w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w = 1	3, 7*	3, 7*	2	9
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w = 1		3	3		
TEST = And function to flags, no result:								
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m			2, 6*	2, 6*	2	9
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1	3, 6*	3, 6*	2	9
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w = 1		3	3		
OR = Or:								
Reg/memory and register to either	0 0 0 0 1 0 d w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w = 1	3, 7*	3, 7*	2	9
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w = 1		3	3		
XOR = Exclusive or:								
Reg/memory and register to either	0 0 1 1 0 0 d w	mod reg r/m			2, 7*	2, 7*	2	9
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w = 1	3, 7*	3, 7*	2	9
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w = 1		3	3		
NOT = Invert register/memory	1 1 1 1 0 1 1 w	mod 0 1 0 r/m			2, 7*	2, 7*	2	9

*Indicates instructions not available in iAPX 86, 88 microsystems.
See footnotes on page 1-124.

80286 INSTRUCTION SET SUMMARY (continued)

Function	Format			Clock Count		Comments	
				Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
STRING MANIPULATION:							
MOVS = Move byte/word	1 0 1 0 0 1 0 w			5	5	2	9
CMPS = Compare byte/word	1 0 1 0 0 1 1 w			8	8	2	9
SCAS = Scan byte/word	1 0 1 0 1 1 1 w			7	7	2	9
LODS = Load byte/wd to AL/AX	1 0 1 0 1 1 0 w			5	5	2	9
STOS = Stor byte/wd from AL/A	1 0 1 0 1 0 1 w			3	3	2	9
INS = Input byte/wd from DX port	0 1 1 0 1 1 0 w			5	5	2	9, 14
OUTS = Output byte/wd to DX port	0 1 1 0 1 1 1 w			5	5	2	9, 14
Repeated by count in CX*							
MOVS = Move string	1 1 1 1 0 0 1 0	1 0 1 0 0 1 0 w		5+4n	5+4n	2	9
CMPS = Compare string	1 1 1 1 0 0 1 z	1 0 1 0 0 1 1 w		5+9n	5+9n	2	9
SCAS = Scan string	1 1 1 1 0 0 1 z	1 0 1 0 1 1 1 w		5+8n	5+8n	2	9
LODS = Load string	1 1 1 1 0 1 0	1 0 1 0 1 1 0 w		5+4n	5+4n	2	9
STOS = Store string	1 1 1 1 0 0 1 0	1 0 1 0 1 0 1 w		4+3n	4+3n	2	9
INS = Input string*	1 1 1 1 0 0 1 0	0 1 1 0 1 1 0 w		5+4n	5+4n	2	9, 14
OUTS = Output string*	1 1 1 1 0 0 1 0	0 1 1 0 1 1 1 w		5+4n	5+4n	2	9, 14
CONTROL TRANSFER							
CALL = Call:							
Direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high	7 + m	7 + m	2	8
Register memory indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m		7 + m,11 + m	7 + m,11 + m	2	8, 9
Direct intersegment	1 0 0 1 1 0 1 0	segment offset		13 + m	26 + m	2	8,11,12
		segment selector					
Protected Mode Only (Direct Intersegment):							
Via call gate to same privilege level				41 + m		8,11,12	
Via call gate to different privilege level, no parameters				82 + m		8,11,12	
Via call gate to different privilege level, x parameters				86 + 4x + m		8,11,12	
Via TSS				177 + m		8,11,12	
Via task gate				182 + m		8,11,12	
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	(mod ≠ 11)	16 + m	29 + m*	2	8,9,11,12
Protected Mode Only (Indirect Intersegment):							
Via call gate to same privilege level					44 + m*		8,9,11,12
Via call gate to different privilege level, no parameters					83 + m*		8,9,11,12
Via call gate to different privilege level, x parameters					90 + 4x + m*		8,9,11,12
Via TSS					180 + m*		8,9,11,12
Via task gate					185 + m*		8,9,11,12
JMP = Unconditional jump							
Short/long	1 1 1 0 1 0 1 1	disp-low		7 + m	7 + m		8
Direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high	7 + m	7 + m		8
Register/mem indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m		7 + m,11 + m*	7 + m,11 + m*	2	8, 9
Direct intersegment	1 1 1 0 1 0 1 0	segment offset		11 + m	23 + m		8,11,12
		segment selector					
Protected Mode Only (Indirect Intersegment):							
Via call gate to same privilege level					38 + m		8,11,12
Via TSS					175 + m		8,11,12
Via task gate					180 + m		8,11,12
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	(mod ≠ 11)	15 + m*	26 + m*	2	8,9,11,12

*Indicates instructions not available in iAPX 86, 88 microsystems.
See footnotes on page 1-124.

80286 INSTRUCTION SET SUMMARY (continued)

				Clock Count		Comments	
Function	**Format**			**Real Address Mode**	**Protected Virtual Address Mode**	**Real Address Mode**	**Protected Virtual Address Mode**
CONTROL TRANSFER (Continued):							
Protected Mode Only (Indirect Intersegment)							
Via call gate to same privilege level					41+m*		8,9,11,12
Via TSS					178+m*		8,9,11,12
Via task gate					183+m*		8,9,11,12
RET = Return from CALL:							
Within segment	11000011			11+m	11+m	2	8,9
Within seg adding immed to SP	11000010	data-low	data-high	11+m	11+m	2	8,9
Intersegment	11001011			15+m	25+m	2	8,9,11,12
Intersegment adding immediate to SP	11001010	data-low	data-high	15+m		2	8,9,11,12
Protected Mode Only (RET):							
To different privilege level					55+m		
JE/JZ = Jump on equal zero	01110100	disp		7+m or 3	7+m or 3		8
JL/JNGE = Jump on less not greater or equal	01111100	disp		7+m or 3	7+m or 3		8
JLE/JNG = Jump on less or equal not greater	01111110	disp		7+m or 3	7+m or 3		8
JB/JNAE = Jump on below not above or equal	01110010	disp		7+m or 3	7+m or 3		8
JBE/JNA = Jump on below or equal not above	01110110	disp		7+m or 3	7+m or 3		8
JP/JPE = Jump on parity/parity even	01111010	disp		7+m or 3	7+m or 3		8
JO = Jump on overflow	01110000	disp		7+m or 3	7+m or 3		8
JS = Jump on sign	01111000	disp		7+m or 3	7+m or 3		8
JNE/JNZ = Jump on not equal not zero	01110101	disp		7+m or 3	7+m or 3		8
JNL/JGE = Jump on not less greater or equal	01111101	disp		7+m or 3	7+m or 3		8
JNLE/JG = Jump on not less or equal greater	01111111	disp		7+m or 3	7+m or 3		8
JNB/JAE = Jump on not below above or equal	01110011	disp		7+m or 3	7+m or 3		8
JNBE/JA = Jump on not below or equal above	01110111	disp		7+m or 3	7+m or 3		8
JNP/JPO = Jump on not par/par odd	01111011	disp		7+m or 3	7+m or 3		8
JNO = Jump on not overflow	01110001	disp		7+m or 3	7+m or 3		8
JNS = Jump on not sign	01111001	disp		7+m or 3	7+m or 3		8
LOOP = Loop CX Times	11100010	disp		8+m or 4	8+m or 4		8
LOOPZ/LOOPE = Loop while zero equal	11100001	disp		8+m or 4	8+m or 4		8
LOOPNZ/LOOPNE = Loop while not zero equal	11100000	disp		8+m or 4	8+m or 4		8
JCXZ = Jump on CX zero	11100011	disp		8+m or 4	8+m or 4		8
ENTER = Enter Procedure*	11001000	data-low	data-high	L			
L=0				11	11	2	9
L=1				15	15	2	9
L>1				16–4(L–1)	16–4(L–1)	2	9
LEAVE = Leave Procedure*	11001001			5	5	2	9

*Indicates instructions not available in iAPX 86, 88 microsystems.
See footnotes on page 1-124.

80286 INSTRUCTION SET SUMMARY (continued)

Function	Format		Clock Count		Comments	
			Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
CONTROL TRANSFER (Continued):						
INT = Interrupt:						
Type specified	1 1 0 0 1 1 0 1	type	23 + m		2	
Type 3	1 1 0 0 1 1 0 0		23 + m		2	
INTO = Interrupt on overflow	1 1 0 0 1 1 1 0		24 − m or 3	24 − or 3	2	
			(3 if no)	(3 if no)		
			(Interrupt)	(Interrupt)		
Protected Mode Only:						
Via interrupt or trap gate to same privilege level				40 + m		8,11,12
Via interrupt or trap gate to fit different privilege level				78 + m		8,11,12
Via Task Gate				167 + m		8,11,12
IRET = Interrupt return	1 1 0 0 1 1 1 1		17 + m	31 + m	2,4	8,9,11, 12,15
Protected Mode Only:						
To different privilege level					55 + m	8,9,11, 12,15
To different task (NT = 1)					169 + m	8,9,11,12
BOUND = Detect value out of range*	0 1 1 0 0 0 1 0	mod reg r/m	13	13 (Use INT clock count if excep- tion 6)	2,6	
PROCESSOR CONTROL						
CLC = Clear carry	1 1 1 1 1 0 0 0		2	2		
CMC = Complement carry	1 1 1 1 0 1 0 1		2	2		
STC = Set carry	1 1 1 1 1 0 0 1		2	2		
CLD = Clear direction	1 1 1 1 1 1 0 0		2	2		
STD = Set direction	1 1 1 1 1 1 0 1		2	2		
CLI = Clear interrupt	1 1 1 1 1 0 1 0		3	3		14
STI = Set interrupt	1 1 1 1 1 0 1 1		2	2		14
HLT = Halt	1 1 1 1 0 1 0 0		2	2		13
WAIT = Wait	1 0 0 1 1 0 1 1		3	3		
LOCK = Bus lock prefix	1 1 1 1 0 0 0 0		0	0		14
CTS = Clear task switched flag	0 0 0 0 1 1 1 1	0 0 0 0 0 1 1 0	2	2	3	13
ESC = Processor Extension Escape	1 0 0 1 1 T T T	mod LLL r/m	9–20*	9–20*	5	17
	(TTT LL are opcode to processor extension)					
SEG = Segment override prefix	0 0 1 reg 1 1 0		0	0		

*Indicates instructions not available in iAPX 86, 88 microsystems.
See footnotes on page 1-124.

80286 INSTRUCTION SET SUMMARY (continued)

Function	Format			Clock Count		Comments	
				Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
PROTECTION CONTROL:							
LGDT = Load global descriptor table register*	00001111	0000001	mod 0 1 0 r/m	11*	11*	2,3	9,13
SGDT = Store global descriptor table register*	00001111	0000001	mod 0 0 0 r/m	11*	11*	2,3	9,13
LIDT = Load interrupt descriptor table register*	00001111	0000001	mod 0 1 1 r/m	12*	12*	2,3	9,13
SIDT = Store interrupt descriptor table register*	00001111	0000001	mod 0 0 1 r/m	12*	12*	2,3	9
LLDT = Load local descriptor table register from table memory*	00001111	0000000	mod 0 1 0 r/m		17,19*	1	9,11,13
SLDT = Store local descriptor table register to register/memory*	00001111	0000000	mod 0 0 0 r/m		2,3*	1	9
LTR = Load task register from register/memory*	00001111	0000000	mod 0 1 1 r/m		17,19*	1	9,11,13
STR = Store task register to register memory*	00001111	0000000	mod 0 0 1 r/m		2,3*	1	9,11,13
LMSW = Load machine status word from register/memory*	00001111	0000001	mod 1 1 0 r/m	3,6*	3,6*	2,3	9,13
SMSW = Store machine status word*	00001111	0000001	mod 1 0 0 r/m	2,3*	2,3*	2,3	9
LAR = Load access rights from register/memory*	00001111	0000010	mod reg r/m		14,16*	1	9,16
LSL = Load segment limit from register/memory*	00001111	0000011	mod reg r/m		14,16*	1	9,16
ARPL = Adjust requested privilege level from register/memory*		01100011	mod reg r/m		10,11*	2	9
VERR = Verify read access: register/memory*	00001111	0000000	mod 1 0 0 r/m		14,16*	1	9,16
VERR = Verify write access*:	00001111	0000000	mod 1 0 1 r/m		14,16*	1	9,16

*Indicates instructions not available in iAPX 86, 88 microsystems.
See footnotes on page 1-124.

Footnotes

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

if mod = 11 then r/m is treated as a REG field

if mod = 00 then DISP = 0*, disp-low and disp-high are absent

if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent

if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP

if r/m = 001 then EA = (BX) + (DI) + DISP

if r/m = 010 then EA = (BP) + (SI) + DISP

if r/m = 011 then EA = (BP) + (DI) + DISP

if r/m = 100 then EA = (SI) + DISP

if r/m = 101 then EA = (DI) + DISP

if r/m = 110 then EA = (BP) + DISP*

if r/m = 111 then EA = (BX) + DISP

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

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

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following:

REG	Segment Register
00	ES
01	CS
10	SS
11	DS

REG is assigned according to the following table:

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

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

80C286

High-Speed CMOS 80286 Microprocessor

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

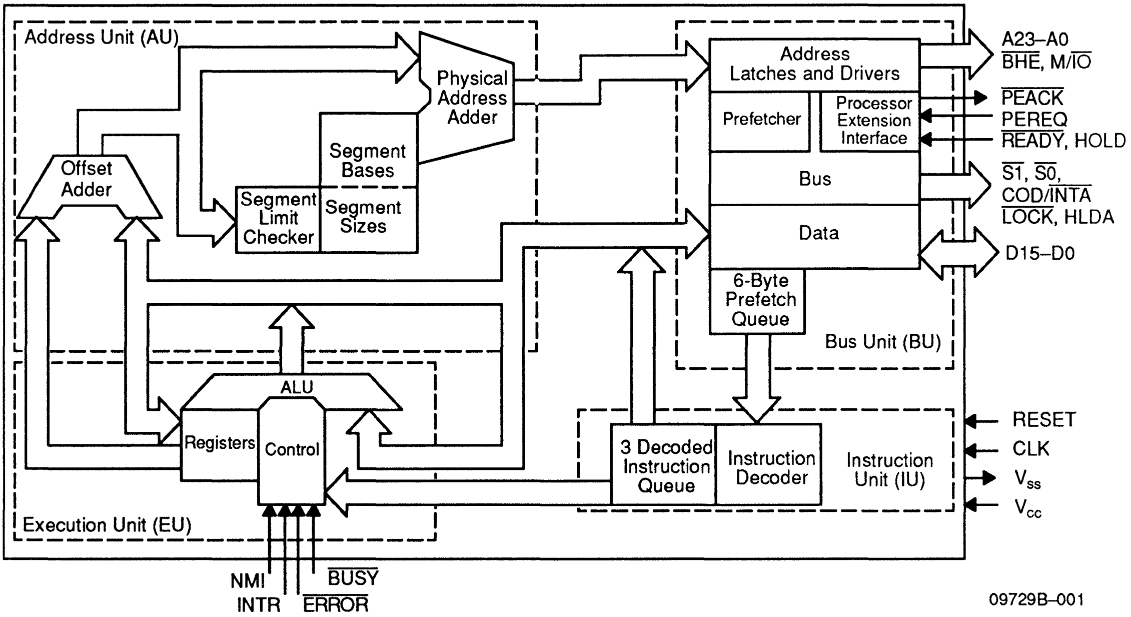
- **Ultra high-performance processor**
  —Over 20 times the performance of the 8086
- **Wide range of clock rates**
  —20 MHz (80C286-20)
  —16 MHz (80C286-16)
  —12.5 MHz (80C286-12)
- **100% functionally and pin compatible with NMOS 286**
- **Static CMOS design for low power operation**
  —Standby mode Icc = 5 mA maximum
  —Operating mode Icc
  220 mA max at 12.5 MHz
  260 mA max at 16 MHz
  310 mA max at 20 MHz
- **68-lead PLCC package**

GENERAL DESCRIPTION

The AMD 80C286 is a high-speed implementation of the industry standard 80286 microprocessor. It is 100% functionally compatible with the NMOS version and is a plug compatible replacement. AMD's high-speed CMOS process allows clock speeds much higher than those attainable with NMOS. The 80C286 operates at clock speeds up to 20 MHz.

This CMOS design is a static implementation which allows the processor to be clocked down to DC and still retain full register status. This is useful for designs where power consumption is a consideration as the 80C286 uses only 5 mA of supply current when in standby mode. The 80C286 also retains full functionality from its maximum clock frequency through very low frequencies down to DC. Since power consumption is proportional to clock speed, the 80C286 may be clocked at a slower rate to draw less current.

BLOCK DIAGRAM



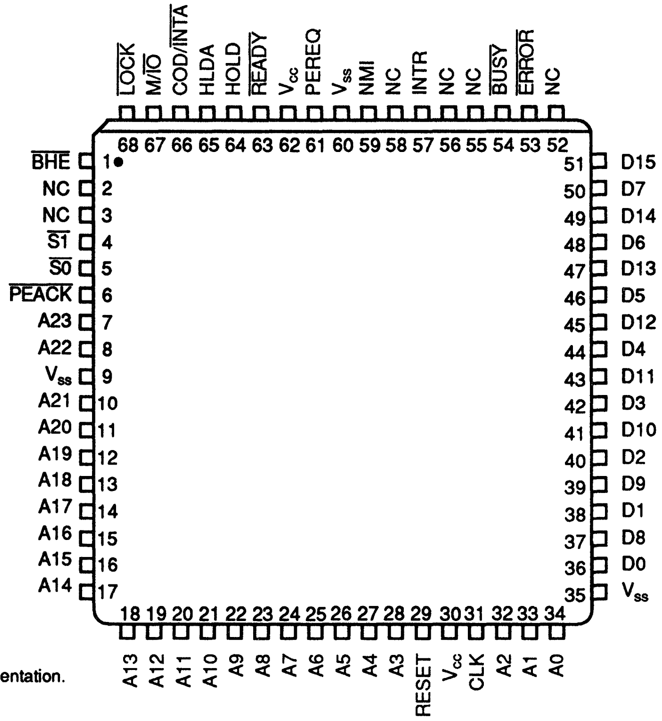
09729B–001

Publication # 11625 Rev. B Amendment /0
Issue Date: December 1991

CONNECTION DIAGRAM

68-Pin PLCC

As viewed from top of package (PC side of component board).



Note: Pin 1 is marked for orientation.

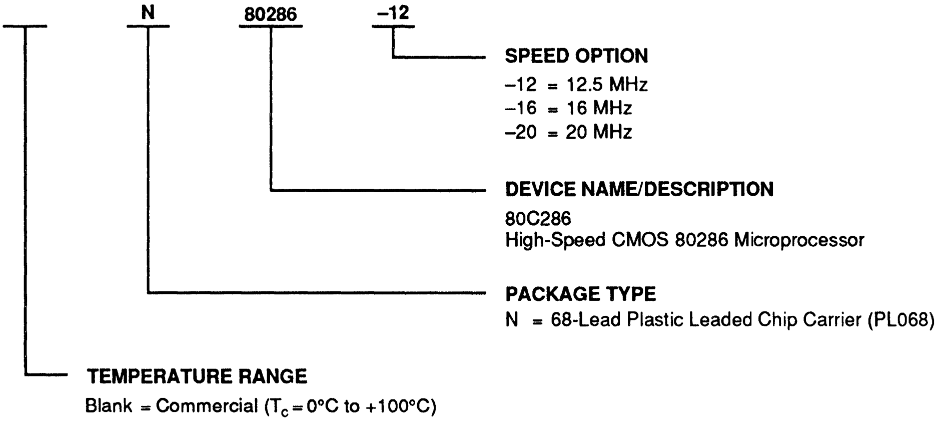
PIN DESIGNATIONS (sorted by pin number)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	$\overline{\text{BHE}}$	24	A7	47	D13
2	NC	25	A6	48	D6
3	NC	26	A5	49	D14
4	$\overline{\text{S1}}$	27	A4	50	D7
5	$\overline{\text{S0}}$	28	A3	51	D15
6	$\overline{\text{PEACK}}$	29	RESET	52	NC
7	A23	30	V_{cc}	53	$\overline{\text{ERROR}}$
8	A22	31	CLK	54	$\overline{\text{BUSY}}$
9	V_{ss}	32	A2	55	NC
10	A21	33	A1	56	NC
11	A20	34	A0	57	INTR
12	A19	35	V_{ss}	58	NC
13	A18	36	D0	59	NMI
14	A17	37	D8	60	V_{ss}
15	A16	38	D1	61	PEREQ
16	A15	39	D9	62	V_{cc}
17	A14	40	D2	63	$\overline{\text{READY}}$
18	A13	41	D10	64	HOLD
19	A12	42	D3	65	HLDA
20	A11	43	D11	66	COD/$\overline{\text{INTA}}$
21	A10	44	D4	67	M/$\overline{\text{IO}}$
22	A9	45	D12	68	$\overline{\text{LOCK}}$
23	A8	46	D5		

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



SPEED OPTION
–12 = 12.5 MHz
–16 = 16 MHz
–20 = 20 MHz

DEVICE NAME/DESCRIPTION
80C286
High-Speed CMOS 80286 Microprocessor

PACKAGE TYPE
N = 68-Lead Plastic Leaded Chip Carrier (PL068)

TEMPERATURE RANGE
Blank = Commercial (T_C = 0°C to +100°C)

Valid Combinations	
N	80C286-20 80C286-16 80C286-12

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

PIN DESCRIPTION

A23–A0

Address Bus (Outputs; Active High)

Address Bus outputs physical memory and I/O port addresses. A0 is Low when data is to be transferred on pins D7–D0. A23–A16 are Low during I/O transfers. The address bus is active High and floats to three-state Off during bus hold acknowledge.

$\overline{\text{BHE}}$

Bus High Enable (Output; Active Low)

Bus High Enable indicates transfer of data on the upper byte of the data bus D15–D8. Eight-bit oriented devices assigned to the upper byte of the data bus would normally use $\overline{\text{BHE}}$ to condition chip select functions. $\overline{\text{BHE}}$ is active Low and floats to three-state Off during bus hold acknowledge.

$\overline{\text{BHE}}$ and A_0 Encodings

$\overline{\text{BHE}}$ Value	A0 Value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus (D15–D8)
1	0	Byte transfer on lower half of data bus (D7–D0)
1	1	Reserved

$\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$

Processor Extension Busy and Error (Inputs; Active Low)

Processor Extension Busy and Error indicate the operating condition of a processor extension to the 80C286. An active $\overline{\text{BUSY}}$ input stops 80C286 program execution on WAIT and some ESC instructions until $\overline{\text{BUSY}}$ becomes inactive (High). The 80C286 may be interrupted while waiting for $\overline{\text{BUSY}}$ to become inactive. An active $\overline{\text{ERROR}}$ input causes the 80C286 to perform a processor extension interrupt when executing WAIT or some ESC instructions. These inputs are active Low and may be asynchronous to the system clock.

CLK

System Clock (Input; Active High)

System Clock provides the fundamental timing for 80C286 systems. It is divided by two inside the 80C286 to generate the processor clock. The internal divide-by-two circuitry can be synchronized to an external clock generator by Low-to-High transition on the RESET input.

COD/$\overline{\text{INTA}}$

Code/Interrupt Acknowledge (Output)

Code/Interrupt Acknowledge distinguishes instruction fetch cycles from memory data read cycles. It also distinguishes interrupt acknowledge cycles from I/O cycles. COD/$\overline{\text{INTA}}$ is pulled up internally during bus hold.

D15–D0

Data Bus (Inputs/Outputs; Active High)

Data Bus inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles. The data bus is active High and floats to three-state Off during bus hold acknowledge.

HOLD, HLDA

Bus Hold Request and Hold Acknowledge (Input/Output; Active High)

Bus Hold Request and Hold Acknowledge control ownership of the 80C286 local bus. The HOLD input allows another local bus master to request control of the local bus. When control is granted, the 80C286 will float its bus drivers to three-state Off and then active HLDA, thus entering the bus hold acknowledge condition. The local bus will remain granted to the requesting master until HOLD becomes inactive, which results in the 80C286 deactivating HLDA and regaining control of the local buys. This terminates the bus hold acknowledge condition. HOLD may be asynchronous to the system clock. These signals are active High.

INTR

Interrupt Request (Input; Active High)

Interrupt Request requests the 80C286 to suspend its current program execution and service a pending external request. Interrupt requests are masked whenever the interrupt enable bit in the flag word is cleared. When the 80C286 responds to an interrupt request, it performs two interrupt acknowledge bus cycles to read an 8-bit interrupt vector that identifies the source of the interrupt. To assure program interruption, INTR must remain active until the first interrupt acknowledge cycle is completed. INTR is sampled at the beginning of each processor cycle and must be active High at least two processor cycles before the current instruction ends in order to interrupt before the next instruction. INTR is level sensitive, active High, and may be asynchronous to the system clock.

$\overline{\text{LOCK}}$

Bus Lock (Output; Active Low)

Bus Lock indicates that other system bus masters are not to gain control of the system bus following the current bus cycle. The $\overline{\text{LOCK}}$ signal may be activated explicitly by the LOCK instruction prefix or automatically by 80C286 hardware during memory XCHG instructions, interrupt acknowledge, or descriptor table access. $\overline{\text{LOCK}}$ is active Low and floats to three-state Off during hold acknowledge.

M/$\overline{\text{IO}}$

Memory/IO Select (Output)

Memory/IO Select distinguishes memory access from I/O access. If High during T_S, a memory cycle or a halt/shutdown cycle is in progress. If Low, an I/O cycle or an interrupt acknowledge cycle is in progress. M/$\overline{\text{IO}}$ is pulled up internally during bus hold.

NC

No Connect

No Connect pins should always remain unconnected.

NMI

Non-maskable Interrupt Request (Input; Active High)

Non-maskable Interrupt Request interrupts the 80C286 with an internally supplied vector value of 2. No interrupt acknowledge cycles are performed. The interrupt enable bit in the 80C286 flag word does not affect this input. The NMI input is active High, may be asynchronous to the system clock, and is edge-triggered after internal synchronization. For proper recognition, the input must have been previously Low for at least four system clock cycles and remain High for at least four system clock cycles.

PEREQ, $\overline{\text{PEACK}}$

Processor Extension Operand Request (Input; Active High)
Processor Extension Acknowledge (Output; Active Low)

Processor Extension Operand Request and Acknowledge extends the memory management and protection capabilities of the 80C286 to processor extensions. The PEREQ input requests the 80C286 to perform a data operand transfer for a processor extension. The $\overline{\text{PEACK}}$ output signals the processor extension when the requested operand is being transferred. PEREQ is active High and may be asynchronous to the system clock. $\overline{\text{PEACK}}$ is active Low.

$\overline{\text{READY}}$

Bus Ready (Input; Active Low)

Bus Ready terminates a bus cycle. Bus cycles are extended without limit until terminated by $\overline{\text{READY}}$ Low. Bus Ready requires that set-up and hold times relative to the system clock be met for correct operation. $\overline{\text{READY}}$ is ignored during bus hold acknowledge.

RESET

System Reset (Input; Active High)

System Reset clears the internal logic of the 80C286 and is active High. The 80C286 may be reinitialized at any time with a Low-to-High transition on RESET which remains active for more than 16 system clock cycles. During RESET active, the output pins of the 80C286 enter the state shown below:

80C286 Pin State during Reset

Pin Value	Pin Names
1 (High)	$\overline{\text{S0}}$, $\overline{\text{S1}}$, $\overline{\text{PEACK}}$, A23–A0, $\overline{\text{BHE}}$, $\overline{\text{LOCK}}$
0 (Low)	M/$\overline{\text{IO}}$, COD/$\overline{\text{INTA}}$, HLDA
Three-state Off	D15–D0

Operation of the 80C286 begins after a High-to-Low transition on RESET. The High-to-Low transition of RESET must be synchronous to the system clock. Approximately 50-system clock cycles are required by the 80C286 for internal initializations before the first bus cycle to fetch code from the power-on execution address is performed.

A Low-to-High transition of RESET synchronous to the system clock will begin a new processor cycle at the next High-to-Low transition of the system clock. The Low-to-High transition of RESET may be asynchronous to the system clock; however, in this case it cannot be predetermined which phase of the processor clock will occur during the next system period. Synchronous Low-to-High transitions of RESET are only required for systems where the processor clock must be phase-synchronous to another clock.

$\overline{\text{S1}}$, $\overline{\text{S0}}$

Bus Cycle Status (Outputs; Active Low)

Bus Cycle Status indicates initiation of a bus cycle and, along with M/$\overline{\text{IO}}$ and COD/$\overline{\text{INTA}}$, defines the type of bus cycle. The bus is in a T_S state whenever one or both are Low. $\overline{\text{S1}}$ and $\overline{\text{S0}}$ are active Low and are pulled up internally during bus hold.

80C286 Bus Cycle Status Definition

COD/ $\overline{\text{INTA}}$	M/$\overline{\text{IO}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$	Bus cycle initiated
0 (Low)	0	0	0	Interrupt acknowledge
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	None; not a status cycle
0	1	0	0	If $A_1 = 1$ then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (High)	0	0	0	Reserved
1	0	0	1	I/O Read
1	0	1	0	I/O Write
1	0	1	1	None; not a status cycle
1	1	0	0	Reserved
1	1	0	1	Memory instruction read
1	1	1	0	Reserved
1	1	1	1	None; not a status cycle

V_{SS}

System Ground (Input)

System ground: 0 V.

V_{CC}

System Power (Input)

System power: +5-V power supply.

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	+8.0 V
Input, Output or I/O Voltage Applied GND	−1.0 V to V_{DD}+1.0 V
Power Dissipation/Speed	
20 MHz	1.7 W
16 MHz	1.4 W
12.5 MHz	1.2 W
Storage Temperature Range	−65°C to +150°C
Junction Temperature	+165°C
Lead Temperature (Soldering, Ten Seconds)	+275°C

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Operating Voltage Range	+4.5 V to +5.5 V
80C286-20 Only	+4.75 V to +5.25 V
Operating Temperature Range	0 to +100°C case temperature

(Meets laptop temperature requirements.)

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating range

V_{CC} = +5 V ± 10% for 80C286-12 and 80C286-16; V_{CC} = +5 V ±5% for 80C286-20, T_C = 0°C to +100°C

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
V_{IL}	Input Low Voltage		−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.5	V
V_{ILC}	CLK Input Low Voltage		−0.5	1.0	V
V_{IHC}	CLK Input High Voltage		3.6	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	I_{OL} = 2.0 mA	–	0.4	V
V_{OH}	Output High Voltage	I_{OH} = −2.0 mA I_{OH} = −100 mA	3.0 V_{CC} − 0.4	–	V V
I_I	Input Leakage Current	V_{IN} = GND or V_{CC} Pins 29, 31, 57, 59, 61, 63–64	−10	10	µA
I_{SH}	Input Sustaining Current on $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ Pins	V_{IN} = GND (see Note 5)	−30	−500	µA
I_{BHL}	Input Sustaining Current High	V_{IN} = 1.0 V (see Note 1)	38	200	µA
I_{BHH}	Input Sustaining Current High	V_{IN} = 3.0 V (see Note 2)	−50	−400	µA
I_O	Output Leakage Current	V_O = GND or V_{CC} Pins 1, 7–8, 10–28, 32–34	−10	10	µA
I_{CCOP}	Active Power Supply Current	80C286-12 (see Note 4) 80C286-16 (see Note 4) 80C286-20 (see Note 4)	– – –	220 260 310	mA mA mA
I_{CCSB}	Standby Power Supply Current	(see Note 3)	–	5	mA

Notes: 1. I_{BHL} should be measured after lowering V_{IN} to GND and then raising to 1.0 V on the following pins: 36–51, 66, 67.

2. I_{BHH} should be measured after raising V_{IN} to V_{CC} and then lowering to 3.0 V on the following pins: 4–6, 36–51, 66–68.

3. I_{CCSB} tested with the clock stopped in phase two of the processor clock cycle. V_{IN} = V_{CC} or GND, V_{CC} = 5.5 V, outputs unloaded.

4. I_{CCOP} measured at 12.5 MHz for the 80C286-12, 16 MHz for the 80C286-16, and 20 MHz for the 80C286-20. V_{IN} = 2.4 V or 0.4 V, V_{CC} = 5.5 V, outputs unloaded.

5. I_{SH} should be measured after raising V_{IN} to V_{CC} and then lowering to GND on pins 53 and 54.

CAPACITANCE (T_A = +25°C; All Measurements Referenced to Device GND)

Parameter Symbol	Parameter Description	Typ	Unit	Test Conditions
C_{CLK}	CLK Input Capacitance	10	pF	FREQ = 1 MHz
C_{IN}	Other Input Capacitance	10	pF	
$C_{I/O}$	I/O Capacitance	10	pF	

SWITCHING CHARACTERISTICS over operating range

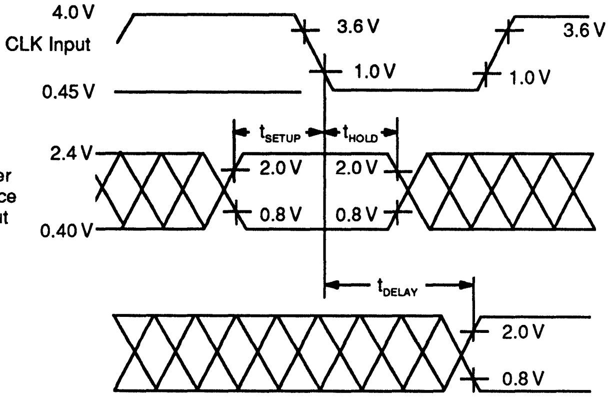
V_{CC} = +5 V ± 10% for 80C286-12 and 80C286-16; V_{CC} = +5 V ±5% for 80C286-20, T_C = 0°C to +100°C
AC Timings are referenced to 0.8 V and 2.0 V points of the signals as illustrated in datasheet waveforms, for 12.5 and 16 MHz, unless otherwise specified. For 20 MHz, AC timings are referenced to the 1.5 V point of the signals as illustrated in Data Sheet waveforms, unless otherwise specified.

Parameter		Test	12.5 MHz		16 MHz		20 MHz		
Symbol	**Description**	**Conditions**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**
Timing Requirements									
1	System Clock (CLK) Period		40	–	31	–	25	–	ns
2	System Clock (CLK) Low Time	@ 1.0 V	11	–	7	–	6	–	ns
3	System Clock (CLK) High Time	@ 3.6 V	13	–	11	–	9	–	ns
17	System Clock (CLK) RISE Time	1.0 V to 3.6 V	–	8	–	5	–	4	ns
18	System Clock (CLK) FALL Time	3.6 V to 1.0 V	–	8	–	5	–	4	ns
4	Asynchronous Inputs SETUP Time	(Note 1)	15	–	5	–	4	–	ns
5	Asynchronous Inputs HOLD Time	(Note 1)	15	–	5	–	4	–	ns
6	RESET SETUP Time		10	–	10	–	10	–	ns
7	RESET HOLD Time		0	–	0	–	0	–	ns
8	Read Data SETUP Time		5	–	5	–	3	–	ns
9	Read Data HOLD Time		4	–	3	–	2	–	ns
10	$\overline{\text{READY}}$ SETUP Time		20	–	12	–	10	–	ns
11	$\overline{\text{READY}}$ HOLD Time		20	–	5	–	3	–	ns
20	Input RISE/FALL Times	0.8 V to 2.0 V	–	8	–	6	–	6	ns
Timing Responses									
12A	Status/$\overline{\text{PEACK}}$ Active Delay	1, (Notes 3, 6, 7)	1	21	1	18	1	15	ns
12B	Status/$\overline{\text{PEACK}}$ Inactive Delay	1, (Notes 3, 6)	1	24	1	20	1	16	ns
13	Address Valid Delay	1, (Notes 2, 3)	1	32	1	27	1	23	ns
14	Write Data Valid Delay	1, (Notes 2, 3)	0	31	0	28	0	27	ns
15	Address/Status/Data Float Delay	2, (Note 5)	0	32	0	29	0	25	ns
16	HLDA Valid Delay	1, (Notes 2, 3, 8)	0	25	0	25	0	20	ns
19	Address Valid–Status SETUP Time	1, (Notes 3, 4)	22	–	16	–	9	–	ns

Notes: 1. Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. This specification is given only for testing purposes to assure recognition at a specific CLK edge.
2. Delay from 1.0 V on the CLK to 0.8 V or 2.0 V.
3. Output load: C_L = 100 pF.
4. Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 0.8 V or status going inactive reaching 2.0 V.
5. Delay from 1.0 V on the CLK to Float (no current drive) condition.
6. Delay from 1.0 V on the CLK to 0.8 V for Min (HOLD time) and to 2.0 V for Max (inactive delay).
7. Delay from 1.0 V on the CLK to 2.0 V for Min (HOLD time) and to 0.8 V for Max (active delay).
8. Delay from 1.0 V on the CLK to 2.0 V.

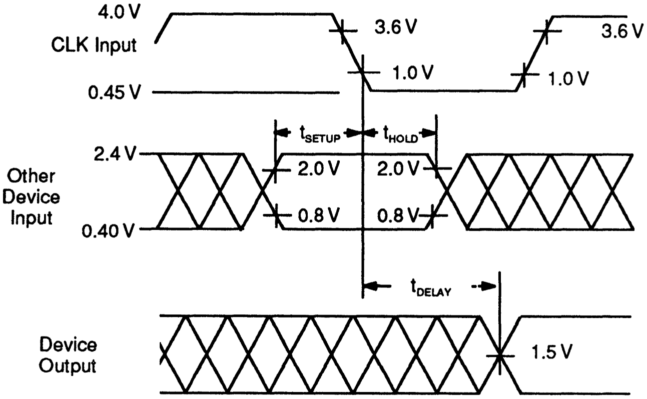
Switching Test Conditions

Test Condition	IL (Constant Current Source	CL
1	| 2.0 mA |	100 pF
2	–6 mA (V_{OH} to Float) 8 mA (V_{OL} to Float)	100 pF



09729B-035

12.5 MHz and 16 MHz AC Setup, Hold and Delay Time Measurement

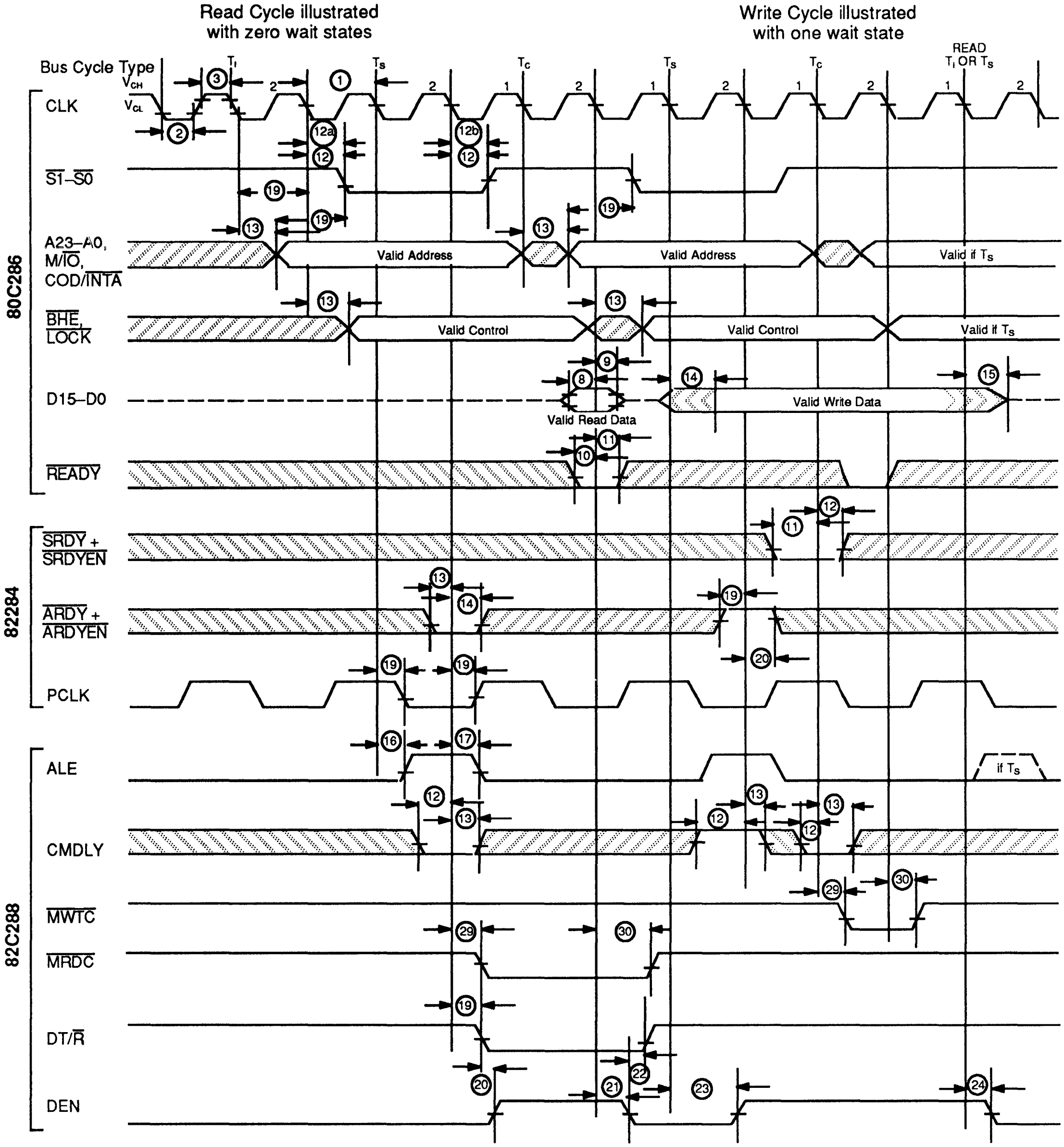


09729B-036

20 MHz AC Setup, Hold and Delay Time Measurement

SWITCHING WAVEFORMS

Major Cycle Timing

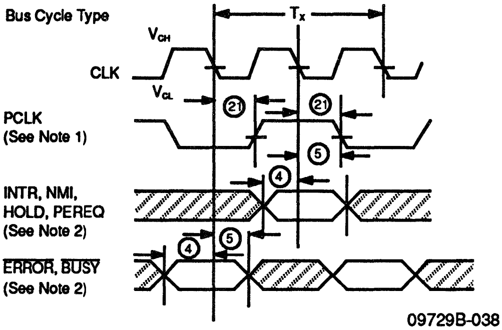

Read Cycle illustrated with zero wait states
Write Cycle illustrated with one wait state
Bus Cycle Type
T_I
T_S
T_C
T_S
T_C
READ T_I OR T_S
V_{CH}
V_{CL}
80C286
CLK
$\overline{S1}$–$\overline{S0}$
A23–A0, M/$\overline{IO}$, COD/$\overline{INTA}$
$\overline{BHE}$, $\overline{LOCK}$
D15–D0
$\overline{READY}$
Valid Address
Valid Address
Valid if T_S
Valid Control
Valid Control
Valid if T_S
Valid Read Data
Valid Write Data
82284
$\overline{SRDY}$ + $\overline{SRDYEN}$
$\overline{ARDY}$ + $\overline{ARDYEN}$
PCLK
82C288
ALE
if T_S
CMDLY
$\overline{MWTC}$
$\overline{MRDC}$
DT/$\overline{R}$
DEN
1 2 3 8 9 10 11 12 12a 12b 13 14 15 16 17 19 20 21 22 23 24 29 30

09729B-037

Note: The modified timing is due to the $\overline{CMDLY}$ signal being active.

SWITCHING WAVEFORMS (continued)

80C286 Asynchronous Input Signal Timing

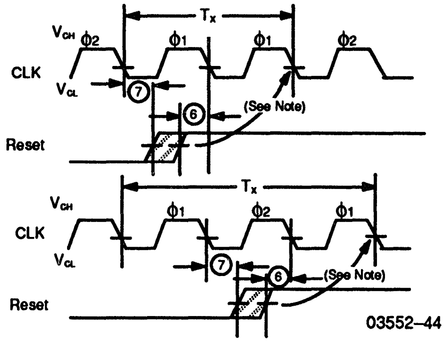


09729B-038

Notes: 1. PCLK indicates which processor cycle phase will occur on the next CLK. PCLK may not indicate the correct phase until the first bus cycle is performed.

2. These inputs are asynchronous. The setup and hold times shown assure recognition for testing purposes.

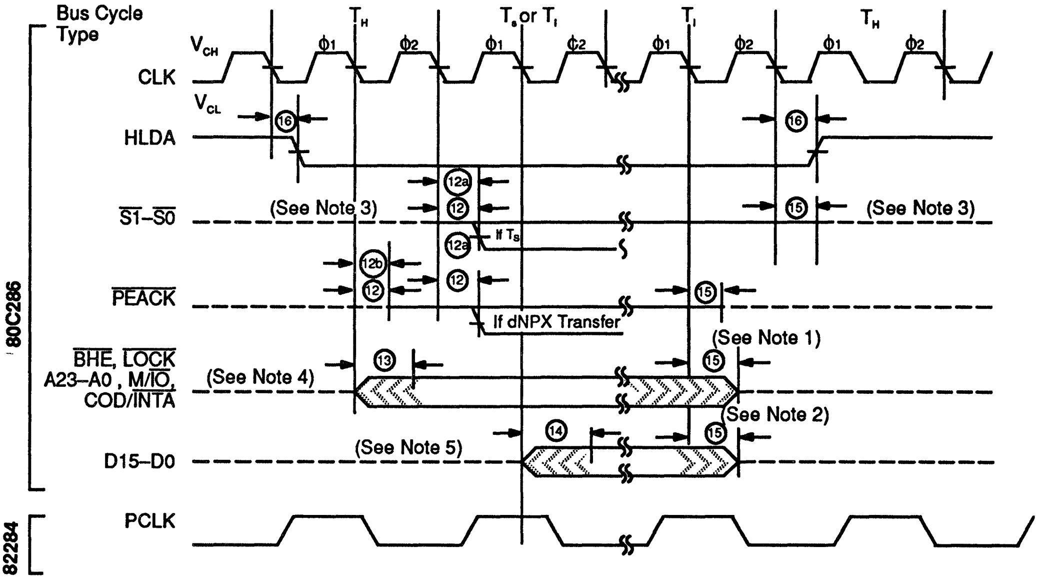
80C286 Reset Input Timing and Subsequent Processor Cycle Phase



03552-44

Note: When Reset meets the set-up time shown, the next CLK will start or repeat ϕ1 of a processor cycle.

Exiting and Entering Hold



09729B-040

Notes: 1. These signals may not be driven by the 80C286 during the time shown. The worst case in terms of latest float time is shown.

2. The data bus will be driven as shown if the last cycle before T_I in the diagram was a write T_C.

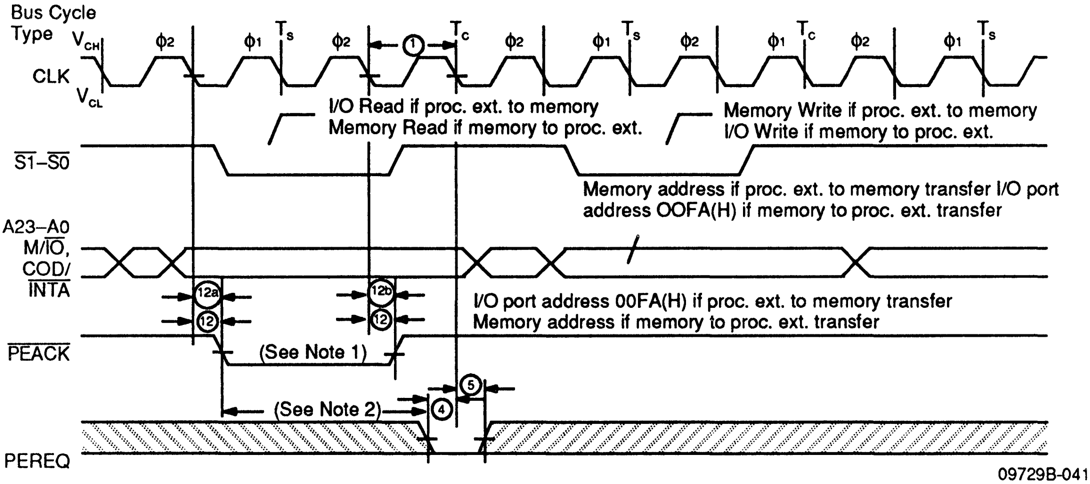
3. The 80C286 puts its status pins in a high impedance logic one state during T_H.

4. BHE and LOCK are driven at this time but will not become valid until T_S.

5. The data bus will remain in three-state Off if a read cycle is performed.

SWITCHING WAVEFORMS (continued)

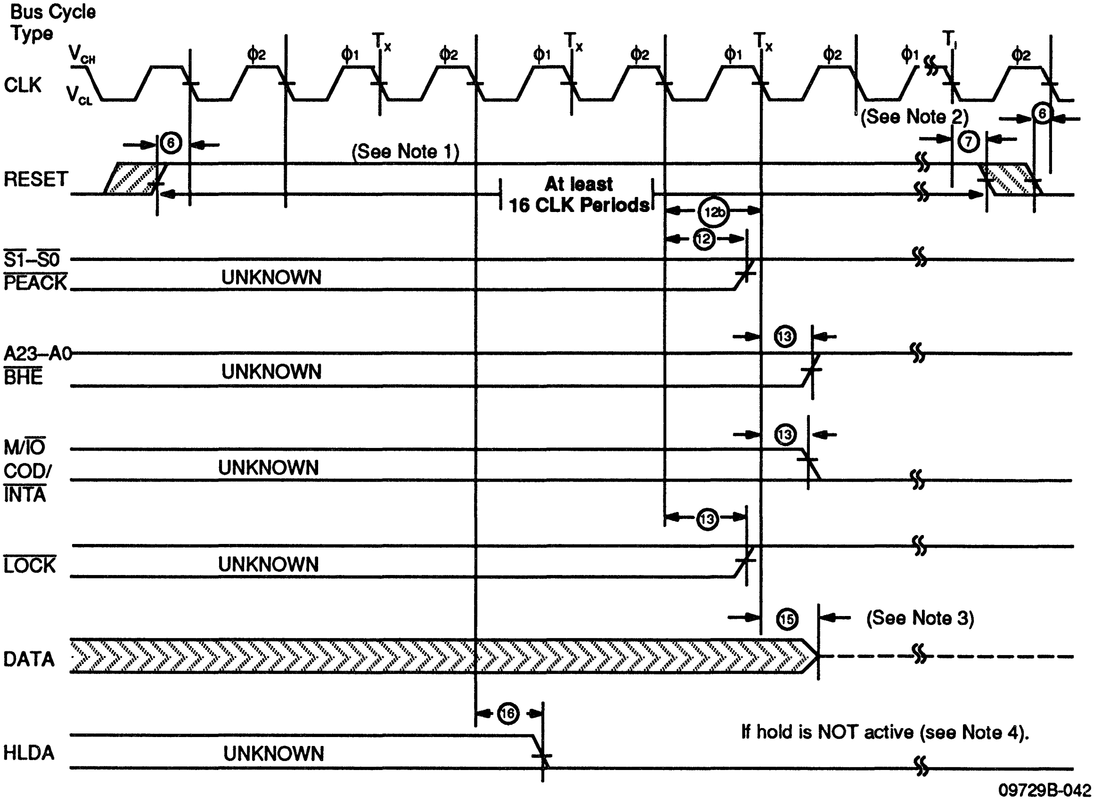
80C286 PEREQ/$\overline{\text{PEACK}}$ Timing Required PEREQ Timing for One Transfer Only


Bus Cycle Type
V_{CH}
CLK
V_{CL}
φ2
φ1
T_s
φ2
1
T_c
φ2
φ1
T_s
φ2
φ1
T_c
φ2
φ1
T_s
I/O Read if proc. ext. to memory
Memory Read if memory to proc. ext.
Memory Write if proc. ext. to memory
I/O Write if memory to proc. ext.
$\overline{S1}$–$\overline{S0}$
Memory address if proc. ext. to memory transfer I/O port address OOFA(H) if memory to proc. ext. transfer
A23–A0
M/$\overline{IO}$,
COD/
$\overline{INTA}$
12a
12
12b
12
I/O port address 00FA(H) if proc. ext. to memory transfer
Memory address if memory to proc. ext. transfer
$\overline{PEACK}$
(See Note 1)
5
(See Note 2)
4
PEREQ
09729B-041

Notes: 1. $\overline{\text{PEACK}}$ always goes active during the first bus operation of a processor extension data operand transfer sequence. The first bus operation will be either a memory read at operand address or I/O read at port address 00FA(H).

2. To prevent a second processor extension data operand transfer, the worst case maximum time (shown above) is: 3 × 1 – 12a max – 4 min. The actual, configuration dependent, maximum time is: 3 × 1–12a max – 4 min + A × 2 × 1. A is the number of extra T_C states added to either the first or second bus operation of the processor extension data operand transfer sequence.

Initial 80C286 Pin State During Reset



Notes:
1. Set-up time for RESET ↑ may be violated with the consideration that φ1 of the processor clock may begin one system CLK period later.
2. Set-up and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during φ1 or φ2.
3. The data bus is only guaranteed to be in three-state Off at the time shown.
4. HOLD is acknowledged during RESET, causing HLDA to go active and the appropriate pins to float. If HOLD remains active while RESET goes inactive, the 80C286 remains in HOLD state and will not perform any bus accesses until HOLD is deactivated.

80L286

Low-Power High-Performance Microprocessor with Memory Management and Protection

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

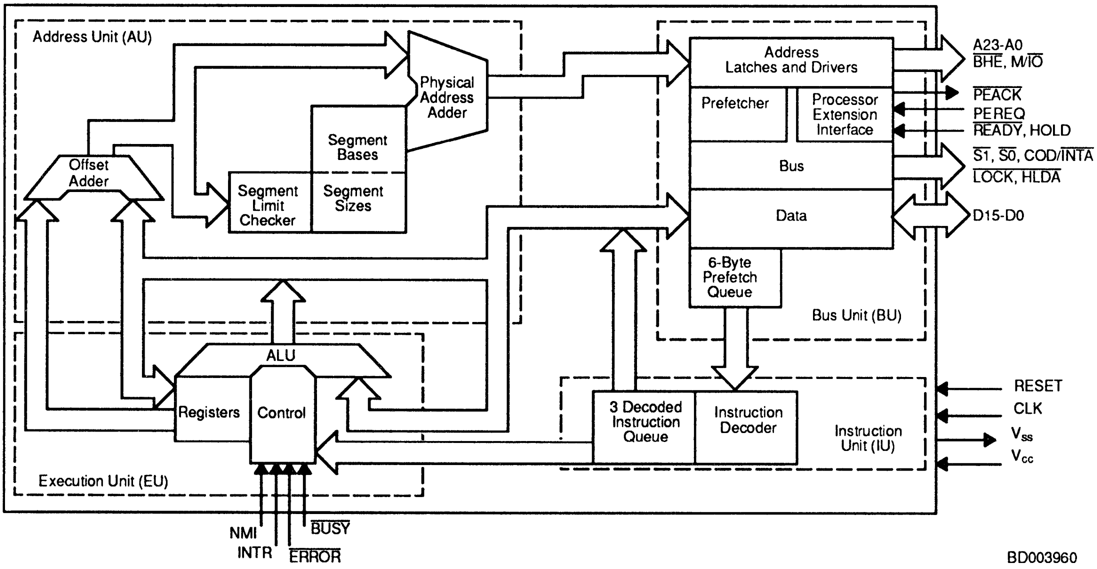
- **High-performance processor (up to 13.3 times iAPX 86 when using the 16-MHz 80L286)**
- **Identical to the 80286 except consumes less power**
- **Available in cost-effective Plastic Leaded Chip Carrier (PLCC) package**
- **Socketed PLCC footprint is compatible with socketed LCC and PGA footprints**
- **Surface-mountable PLCC for high density board utilization**
- **8-, 10-, and 12.5- and16-MHz operation**
- **Large address space**
  - 16-Mb physical
  - 1-Gb virtual memory per task
- **Integrated memory management, four-level memory protection and support for virtual memory and operating systems**

GENERAL DESCRIPTION

The 80L286 is an advanced, high performance microprocessor, identical to the 80286, except consumes less power. Its reduced power enables the 80L286 to be packaged in low-cost, Plastic Leaded Chip Carrier (PLCC) without a heat sink or heat spreader. Cooler operation also enhances reliability. The PLCC package can be surface-mounted or socketed. The footprint of the socketed PLCC package is identical to the socketed LCC or PGA packages so no board layout change is needed. The 80L286 is available in 8-, 10-, 12-, and 16-MHz speeds and is fully compatible with the 82C288 Bus Controller and the 82284 Clock Driver functions.

The 80L286 is upward compatible with iAPX 86 and 88 software. Using iAPX real address mode, the 80L286 is object code compatible with existing iAPX 86, 88 software. In protected virtual address mode, the 80L286 is source code compatible with iAPX 86, 88 software and may require upgrading to use virtual addresses supported by the 80L286's integrated memory management and protection mechanism. Both modes operate at full 80L286 performance and execute a superset of the iAPX 86 and 88 instructions.

BLOCK DIAGRAM



BD003960

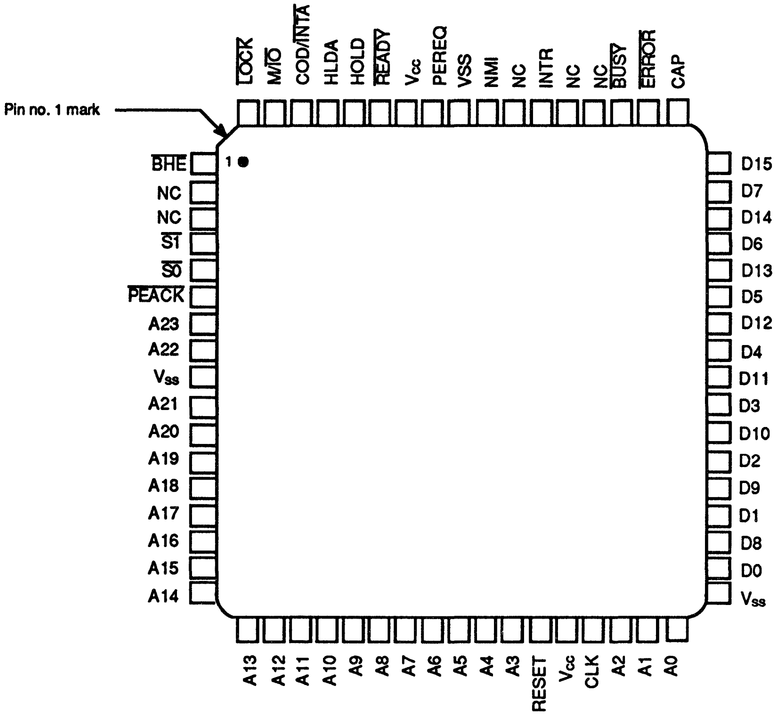
Publication # 08511 Rev. E Amendment /0
Issue Date: December 1991

GENERAL DESCRIPTION (continued)

The 80L286 provides special operations to support the efficient implementation and execution of operating systems. For example, one instruction can end execution of one task, save its state, switch to a new task, load its state, and start execution of the new task. The 80L286 also supports virtual memory systems by providing a segment-not-present exception and restartable instructions.

CONNECTION DIAGRAM
Top View

Plastic Leaded Chip Carrier
(PL 068)



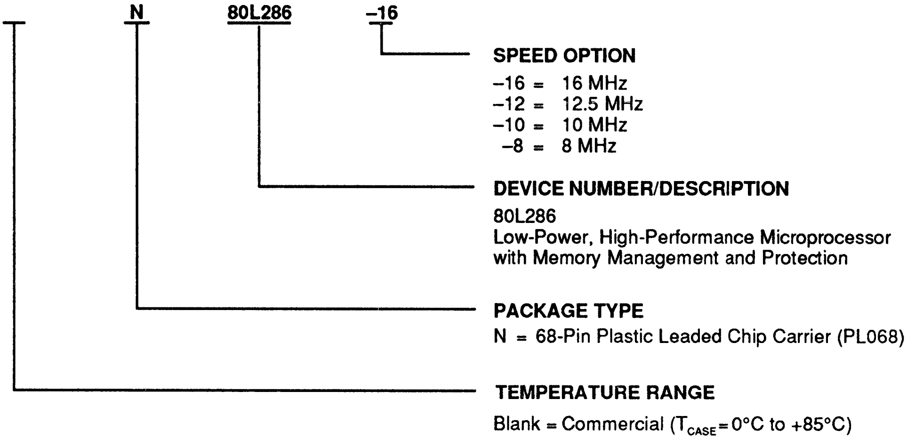
CD010641

As viewed from top of package (PC side of component board)

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



SPEED OPTION

–16 = 16 MHz
–12 = 12.5 MHz
–10 = 10 MHz
–8 = 8 MHz

DEVICE NUMBER/DESCRIPTION

80L286
Low-Power, High-Performance Microprocessor with Memory Management and Protection

PACKAGE TYPE

N = 68-Pin Plastic Leaded Chip Carrier (PL068)

TEMPERATURE RANGE

Blank = Commercial (T_{CASE} = 0°C to +85°C)

Valid Combinations	
N	80L286-16 80L286-12 80L286-10 80L286-8

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

PIN DESCRIPTION

CLK

System Clock (Input; Active High)

System Clock provides the fundamental timing for 80L286 systems. It is a 16 MHz signal divided by two inside the 80L286 to generate the 8 MHz processor clock. The internal divide-by-two circuitry can be synchronized to an external clock generator by a Low-to-High transition on the RESET input.

D0–D15

Data Bus (Input/Output; Active High)

Data Bus inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles. The data bus is active High and floats to three-state OFF during bus hold acknowledge.

A23–A0

Address Bus (Output; Active High)

Address Bus outputs physical memory and I/O port addresses. A0 is Low when data is to be transferred on pins D7–D0. A23–A16 are Low during I/O transfers. The address bus is active High and floats to three-state OFF during bus hold acknowledge.

$\overline{\text{BHE}}$

Bus High Enable (Output; Active Low)

Bus High Enable indicates transfer of data on the upper byte of the data bus D15-D8. Eight-bit oriented devices assigned to the upper byte of the data bus would normally use $\overline{\text{BHE}}$ to condition chip select functions. $\overline{\text{BHE}}$ is active Low and floats to three-state OFF during bus hold acknowledge.

$\overline{\text{BHE}}$ and A$_0$ Encodings

$\overline{\text{BHE}}$ Value	A0 Value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus (D15–D8)
1	0	Byte transfer on lower half of data bus (D7–D0)
1	1	Reserved

$\overline{\text{S1}}$, $\overline{\text{S0}}$

Bus Cycle Status (Output; Active Low)

Bus Cycle Status indicates initiation of a bus cycle and, along with M/$\overline{\text{IO}}$ and COD/$\overline{\text{INTA}}$, defines the type of bus cycle. The bus is in a T_S state whenever one or both are Low. $\overline{\text{S1}}$ and $\overline{\text{S0}}$ are active Low and float to three-state OFF during bus hold acknowledge.

80L286 Bus Cycle Status Definition

COD/$\overline{\text{INTA}}$	M/$\overline{\text{IO}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$	Bus cycle initiated
0 (Low)	0	0	0	Interrupt acknowledge
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	None; not a status cycle
0	1	0	0	IF A1 = 1 then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (High)	0	0	0	Reserved
1	0	0	1	I/O Read
1	0	1	0	I/O Write
1	0	1	1	None; not a status cycle
1	1	0	0	Reserved
1	1	0	1	Memory instruction read
1	1	1	0	Reserved
1	1	1	1	None; not a status cycle

M/$\overline{\text{IO}}$

Memory/IO Select (Output)

Memory/IO Select distinguishes memory access from I/O access. If High during T_S, a memory cycle or a halt/shutdown cycle is in progress. If Low, an I/O cycle or an interrupt acknowledge cycle is in progress M/$\overline{\text{IO}}$ floats to three-state OFF during bus hold acknowledge.

COD/$\overline{\text{INTA}}$

Code/Interrupt Acknowledge (Output)

Code/Interrupt Acknowledge distinguishes instruction fetch cycles from memory data read cycles. Also distinguishes interrupt acknowledge cycles from I/O cycles. COD/$\overline{\text{INTA}}$ floats to three-state OFF during bus hold acknowledge.

$\overline{\text{LOCK}}$

Bus Lock (Output; Active Low)

Bus Lock indicates that other system bus masters are not to gain control of the system bus following the current bus cycle. The $\overline{\text{LOCK}}$ signal may be activated explicitly by the "LOCK" instruction prefix or automatically by 80286 hardware during memory XCHG instructions, interrupt acknowledge, or descriptor table access. $\overline{\text{LOCK}}$ is active Low and floats to three-state OFF during hold acknowledge.

PIN DESCRIPTION (continued)

$\overline{\text{READY}}$

Bus Ready (Input; Active Low)

Bus Ready terminates a bus cycle. Bus cycles are extended without limit until terminated by $\overline{\text{READY}}$ Low. $\overline{\text{READY}}$ is an active Low synchronous input requiring setup and hold times relative to the system clock be met for correct operation. $\overline{\text{READY}}$ is ignored during bus hold acknowledge.

HOLD, HLDA

Bus Hold Request and Hold Acknowledge (Input/Output; Active High)

Bus Hold Request and Hold Acknowledge control ownership of the 80L286 local bus. The HOLD input allows another local bus master to request control of the local bus. When control is granted, the 80L286 will float its bus drivers to three-state OFF and then active HLDA, thus entering the bus hold acknowledge condition. The local bus will remain granted to the requesting master until HOLD becomes inactive which results in the 80L286 deactivating HLDA and regaining control of the local buys. This terminates the bus hold acknowledge condition. HOLD may be asynchronous to the system clock. These signals are active HIGH.

INTR

Interrupt Request (Input; Active High)

Interrupt Request requests the 80L286 to suspend its current program execution and service a pending external request. Interrupt requests are masked whenever the interrupt enable bit in the flag word is cleared. When the 80L286 responds to an interrupt request, it performs two interrupt acknowledge bus cycles to read an 8-bit interrupt vector that identifies the source of the interrupt. To assure program interruption, INTR must remain active until the first interrupt acknowledge cycle is completed. INTR is sampled at the beginning of each processor cycle and must be active High at least two processor cycles before the current instruction ends in order to interrupt before the next instruction. INTR is level sensitive, active High, and may be asynchronous to the system clock.

NMI

Non-maskable Interrupt Request (Input; Active High)

Non-maskable Interrupt Request interrupts the 80L286 with an internally supplied vector value of 2. No interrupt acknowledge cycles are performed. The interrupt enable bit in the 80L286 flag word does not affect this input. The NMI input is active High, may be asynchronous to the system clock, and is edge triggered after internal synchronization. For proper recognition, the input must have been previously Low for at least four system clock cycles and remain High for at least four system clock cycles.

PEREQ, $\overline{\text{PEACK}}$

Processor Extension Operand Request and Acknowledge (Input/Output)

Processor Extension Operand Request and Acknowledge extends the memory management and protection capabilities of the 80L286 to processor extensions. The PEREQ input requests the 80L286 to perform a data operand transfer for a processor extension. The $\overline{\text{PEACK}}$ output signals the processor extension when the requested operand is being transferred. PEREQ is active High and may be asynchronous to the system clock. $\overline{\text{PEACK}}$ is active Low.

$\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$

Processor Extension Busy and Error (Input; Active Low)

Processor Extension Busy and Error indicate the operating condition of a processor extension to the 80L286. An active $\overline{\text{BUSY}}$ input stops 80L286 program execution on WAIT and some ESC instructions until $\overline{\text{BUSY}}$ becomes inactive (High). The 80L286 may be interrupted while waiting for $\overline{\text{BUSY}}$ to become inactive. An active $\overline{\text{ERROR}}$ input causes the 80L286 to perform a processor extension interrupt when executing WAIT or some ESC instructions. These inputs are active Low and may be asynchronous to the system clock.

RESET

System Reset (Input; Active High)

System Reset clears the internal logic of the 80L286 and is active High. The 80L286 may be reinitialized at any time with a Low-to-High transition on RESET which remains active for more than 16 system clock cycles. During RESET active, the output pins of the 80C286 enter the state shown below:

80L286 Pin State during Reset

Pin Value	Pin Names
1 (High)	$\overline{\text{S0}}$, $\overline{\text{S1}}$, $\overline{\text{PEACK}}$, A23–A0, $\overline{\text{BHE}}$, $\overline{\text{LOCK}}$
0 (Low)	M/$\overline{\text{IO}}$, COD/$\overline{\text{INTA}}$, HLDA
Three-state OFF	D15–D0

Operation of the 80L286 begins after a High-to-Low transition on RESET. The High-to-Low transition of RESET must be synchronous to the system clock. Approximately 50 system clock cycles are required by the 80L286 for internal initializations before the first bus cycle to fetch code from the power-on execution address is performed.

PIN DESCRIPTION (continued)

A Low-to-High transition of RESET synchronous to the system clock, will begin a new processor cycle at the next High-to-Low transition of the system clock. The Low-to-High transition of RESET may be asynchronous to the system clock; however, in this case it cannot be predetermined which phase of the processor clock will occur during the next system period. Synchronous Low-to-High transitions of RESET are only required for systems where the processor clock must be phase synchronous to another clock.

V_{SS}

System Ground (Input)

System Ground: 0 V.

V_{CC}

System Power (Input)

System Power: +5-V power supply.

CAP

Substrate Filter Capacitor (Input; Active High)

Substrate Filter Capacitor: a 0.047 μF ±20% 12 V capacitor must be connected between this pin and ground. This capacitor filters the output of the internal substrate bias generator. A maximum DC leakage current of 1 μA is allowed through the capacitor.

For correct operation of the 80L286, the substrate bias generator must charge this capacitor to its operating voltage. The capacitor charge-up time is 5 ms (max.) after V_{CC} and CLK reach their specified AC and DC parameters. RESET may be applied to prevent spurious activity by the CPU during this time. After this time, the 80L286 processor clock can be phase synchronized to another clock by pulsing RESET Low synchronous to the system clock.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature . . . . –65°C to +150°C
Voltage on Any Pin with
Respect to Ground . . . . –1.0 V to 7.0 V
Power Dissipation . . . . 2.89 Watts

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Operating Voltage Range . . . . +4.75 V to +5.25 V
Case Operating Temperature Range . . 0°C to +85°C

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating range

(T_{CASE} = 0°C to 85°C, V_{CC} = 5 V ±5%)

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
V_{IL}	Input Low Voltage		–0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.5	V
V_{ILC}	CLK Input Low Voltage		–0.5	.6	V
V_{IHC}	CLK Input High Voltage		3.8	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	I_{OL} = 2.0 mA		0.45	V
V_{OH}	Output High Voltage	I_{OH} = –400 µA	2.4		V
I_{LI}	Input Leakage Current	0 V ≤ V_{IN} ≤ V_{CC}		±10	µA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		±10	µA
I_{CC}	Supply Current	T_C = 0°C T_C = 85°C		550 475	mA mA
C_{CLK}	CLK Input Capacitance	F_C = 1 MHz		20	pF
C_{IN}	Other Input Capacitance	F_C = 1 MHz		10	pF
C_O	Input/Output Capacitance	F_C = 1 MHz		20	pF
I_{LO}	Output Leakage Current	0 V ≤ V_{OUT} < 0.45 V		±1	mA
I_{IL}	Input Sustaining Current on $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ pins	V_{IN} = 0 V	30	500	µA
I_{LCR}	Input CLK Leakage Current	0.45 ≤ V_{IN} ≤ V_{CC}		±10	µA
I_{LCR}	Input CLK Leakage Current	0 V ≤ V_{IN} ≤ 0.45 V		±1	mA

Note: Low temperature is worst case.

SWITCHING CHARACTERISTICS

V_{CC} = +5 V ±5%, T_{CASE} = 0°C to +85°C

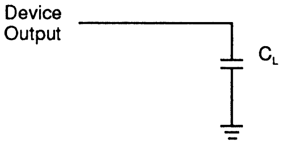
AC timings are referenced to 0.8 V and 2.0 V points of the signals as illustrated in data sheet waveforms, unless otherwise noted.

Parameter Symbol	Parameter Description	Test Conditions	8 MHz		10 MHz		Unit
			Min.	Max.	Min.	Max.	
1	System Clock (CLK) Period		62	125	50	125	ns
2	System Clock (CLK) Low Time	@ 1.0 V	15	100	12	109	ns
3	System Clock (CLK) High Time	@ 3.6 V	25	110	16	113	ns
17	System Clock (CLK) RISE Time	1.0 V to 3.6 V		10		8	ns
18	System Clock (CLK) FALL Time	3.6 V to 1.0 V		10		8	ns
4	Asynchronous Inputs SETUP Time	(Note 1)	20		20		ns
5	Asynchronous Inputs HOLD Time	(Note 1)	20		20		ns
6	RESET SETUP Time		28		23		ns
7	RESET HOLD Time		5		5		ns
8	Read Data SETUP Time		10		8		ns
9	Read Data HOLD Time		8		8		ns
10	$\overline{\text{READY}}$ SETUP Time		38		26		ns
11	$\overline{\text{READY}}$ HOLD Time		25		25		ns
12	Status/$\overline{\text{PEACK}}$ Valid Delay	(Notes 2, 3)	1	40	–	–	ns
12A	Status/$\overline{\text{PEACK}}$ Active Delay	(Notes 2, 3)	–	–	1	22	ns
12B	Status/$\overline{\text{PEACK}}$ Inactive Delay	(Notes 2, 3)	–	–	1	30	ns
13	Address Valid Delay	(Notes 2, 3)	1	60	1	35	ns
14	Write Data Valid Delay	(Notes 2, 3)	0	50	0	30	ns
15	Address/Status/Data Float Delay	(Notes 2, 4)	0	50	0	47	ns
16	HLDA Valid Delay	(Notes 2, 3)	0	50	0	47	ns
19	Address Valid to Status SETUP Time	(Notes 3, 5, 6)	38		27		ns

SWITCHING CHARACTERISTICS (continued)

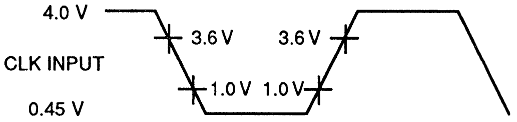
Parameter Symbol	Parameter Description	Test Conditions	12.5 MHz		16 MHz		Unit
			Min.	Max.	Min.	Max.	
1	System Clock (CLK) Period		40	125	31	125	ns
2	System Clock (CLK) Low Time	@ 1.0 V	11	112	10	113	ns
3	System Clock (CLK) High Time	@ 3.6 V	13	114	12	115	ns
17	System Clock (CLK) RISE Time	1.0 V to 3.6 V		8		5	ns
18	System Clock (CLK) FALL Time	3.6 V to 1.0 V		8		4	ns
4	Asynchronous Inputs SETUP Time	(Note 1)	15		11		ns
5	Asynchronous Inputs HOLD Time	(Note 1)	15		11		ns
6	RESET SETUP Time		18		14		ns
7	RESET HOLD Time		5		3		ns
8	Read Data SETUP Time		5		5		ns
9	Read Data HOLD Time		6		5		ns
10	$\overline{\text{READY}}$ SETUP Time		22		15		ns
11	$\overline{\text{READY}}$ HOLD Time		20		15		ns
12	Status/$\overline{\text{PEACK}}$ Valid Delay	(Notes 2, 3)	–	–	–	–	ns
12A	Status/$\overline{\text{PEACK}}$ Active Delay	(Notes 2,3)	3	18	1	18	ns
12B	Status/$\overline{\text{PEACK}}$ Inactive Delay	(Notes 2,3)	3	20	1	20	ns
13	Address Valid Delay	(Notes 2, 3)	1	32	1	29	ns
14	Write Data Valid Delay	(Notes 2, 3)	0	30	0	22	ns
15	Address/Status/Data Float Delay	(Notes 2, 4)	0	32	0	29	ns
16	HLDA Valid Delay	(Notes 2, 3)	0	25	0	25	ns
19	Address Valid to Status SETUP Time	(Notes 3, 5, 6)	22		22		ns

Notes:
1. Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.
2. Delay from 1.0 V on the CLK to 0.8 V or 2.0 V or float on the output as appropriate for valid or floating condition.
3. Output load: $C_L = 100$ pF.
4. Float condition occurs when output current is less than I_{LO} in magnitude.
5. Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 2.0 V or status going inactive reaching 0.8 V.
6. For load capacitance of 10 pF on STATUS/PEACK lines, subtract typically 7 ns for 8 MHz spec, and maximum 7 ns for 10 MHz spec.



Note 7:
AC Test Loading on Outputs

TC004190

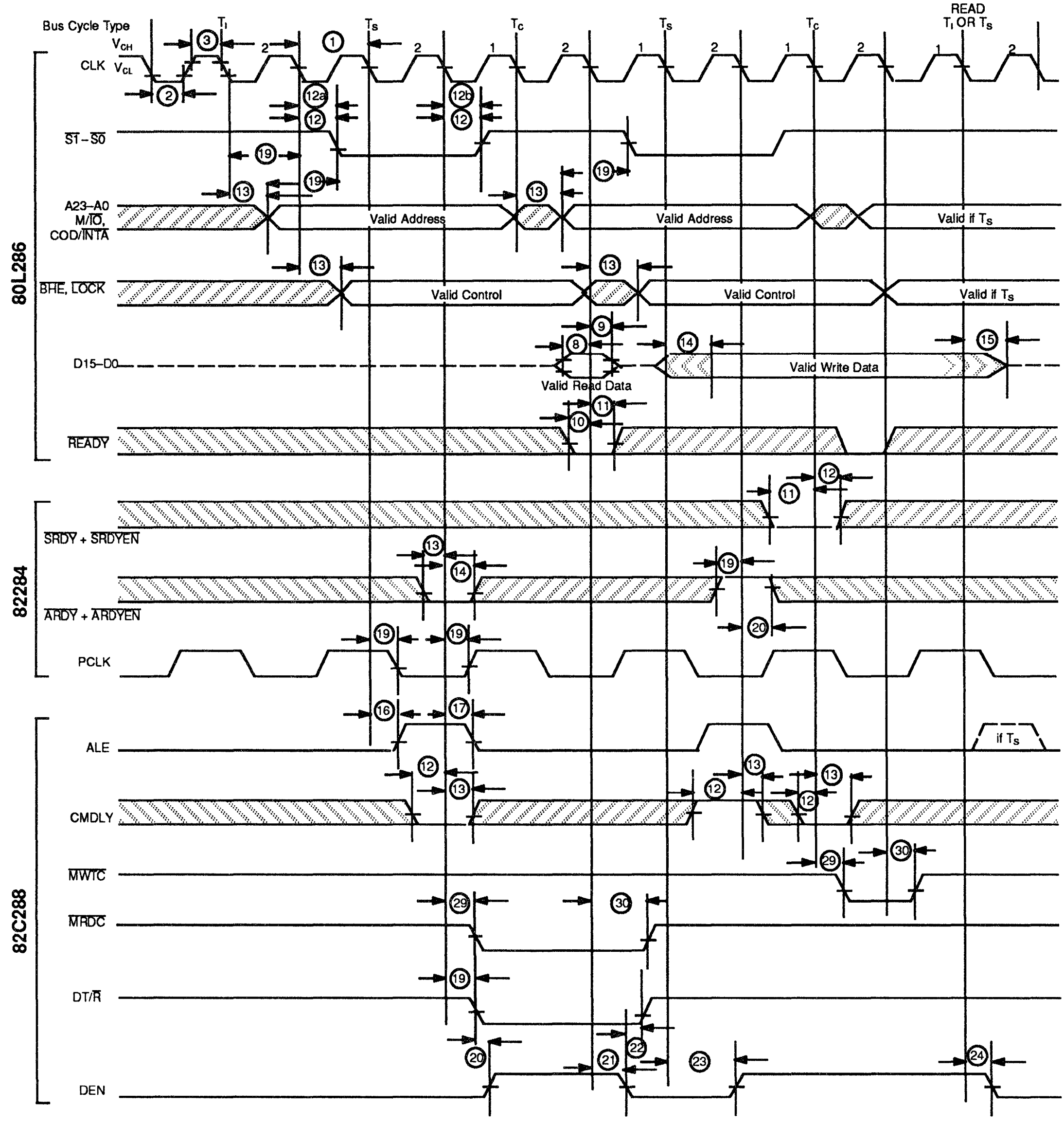


Note 8:
AC Drive and Measurement Points—CLK Input

WF024240

SWITCHING WAVEFORMS

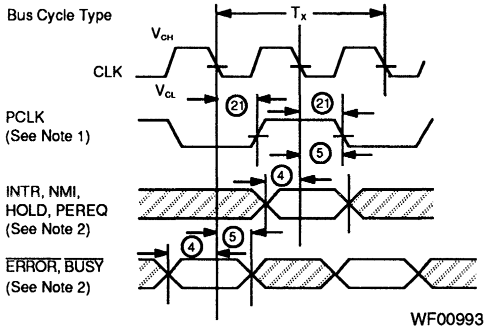
Major Cycle Timing



Note: $\overline{\text{MWTC}}$ is valid at this point only if $\overline{\text{CMDLY}}$ is Low.

SWITCHING WAVEFORMS (continued)

80L286 Asynchronous Input Signal Timing

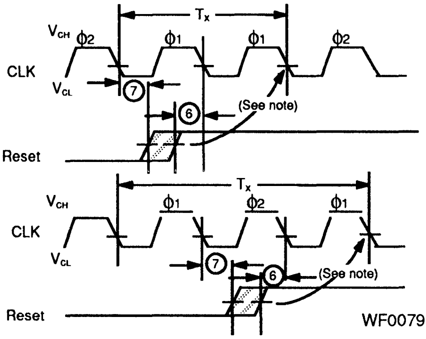


WF009930

Notes: 1. PCLK indicates which processor cycle phase will occur on the next CLK. PCLK may not indicate the correct phase until the first bus cycle is performed.

2. These inputs are asynchronous. The setup and hold times shown assure recognition for testing purposes.

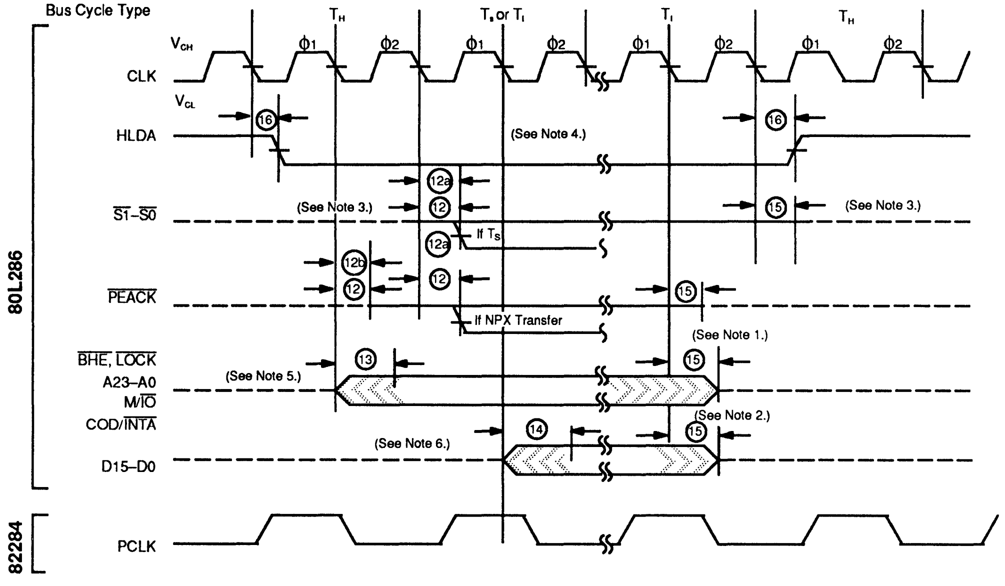
80L286 Reset Input Timing and Subsequent Processor Cycle Phase



WF007930

Note: When RESET meets the set-up time shown, the next CLK will start or repeat ϕ1 of a processor cycle.

Exiting and Entering Hold



WF009942

Notes: 1. These signals may not be driven by the 80L286 during the time shown. The worst case in terms of latest float time is shown.

2. The data bus will be driven as shown if the last cycle before T_I in the diagram was a write T_C.

3. The 80L286 floats its status pins during T_H. External 20 kΩ resistors keep these signals High.

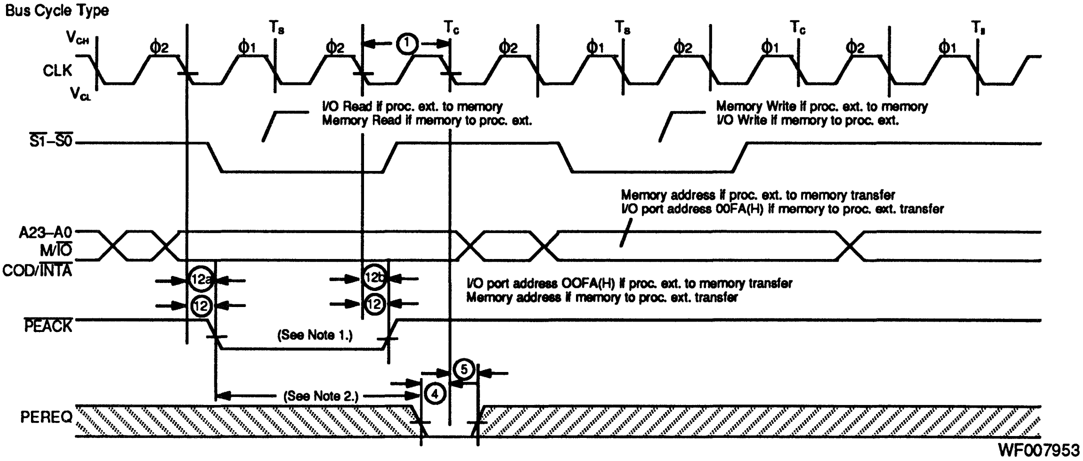
4. For HOLD request set-up to HLDA, refer to Figure 34.

5. BHE and LOCK are driven at this time but will not become valid until T_S.

6. The data bus will remain in three-state OFF if a read cycle is performed.

SWITCHING WAVEFORMS (continued)

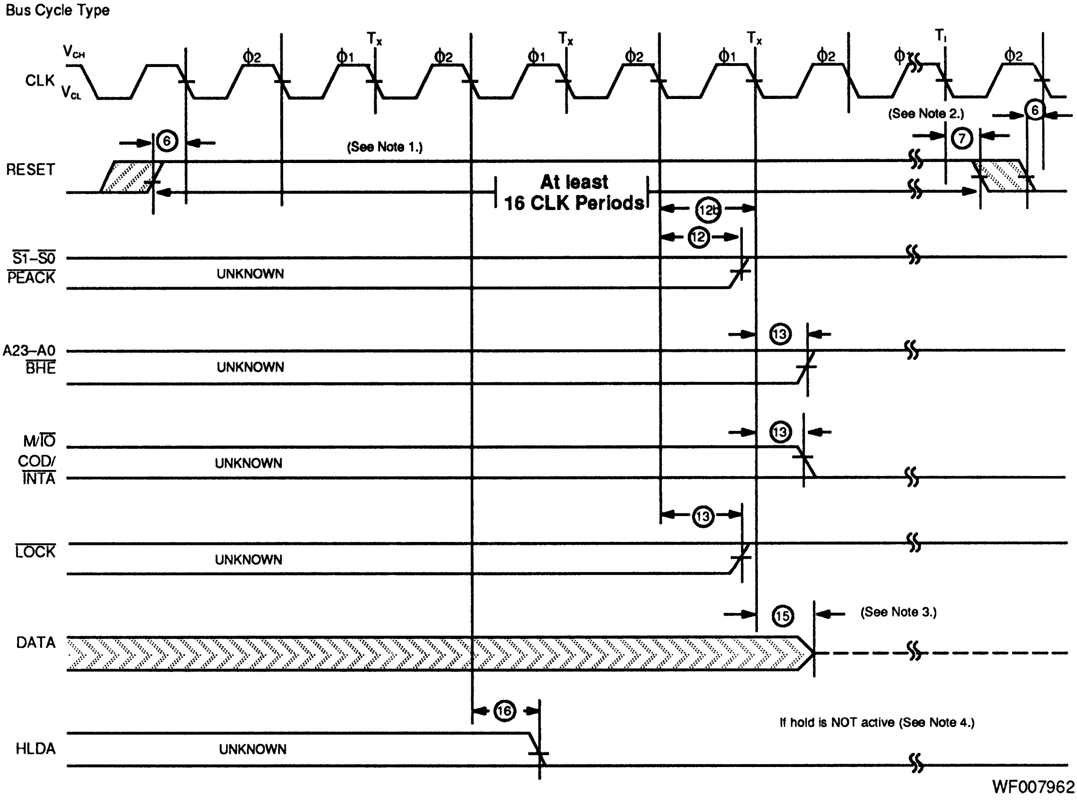
80L286 PEREQ/$\overline{\text{PEACK}}$ Timing Required PEREQ Timing for One Transfer Only


Bus Cycle Type
Ts
Tc
Ts
Tc
Ti
VCH
CLK
VCL
ϕ2
ϕ1
ϕ2
ϕ2
ϕ1
ϕ2
ϕ1
ϕ2
ϕ1
1
I/O Read if proc. ext. to memory
Memory Read if memory to proc. ext.
Memory Write if proc. ext. to memory
I/O Write if memory to proc. ext.
S1-S0
Memory address if proc. ext. to memory transfer
I/O port address 00FA(H) if memory to proc. ext. transfer
A23-A0
M/IO
COD/INTA
12a
12
12b
12
I/O port address 00FA(H) if proc. ext. to memory transfer
Memory address if memory to proc. ext. transfer
PEACK
(See Note 1.)
5
(See Note 2.)
4
PEREQ
WF007953

Notes: 1. $\overline{\text{PEACK}}$ always goes active during the first bus operation of a processor extension data operand transfer sequence. The first bus operation will be either a memory read at operand address or I/O read at port address 00FA(H).

2. To prevent a second processor extension data operand transfer, the worst case maximum time (shown above) is: 3 x 1–12a max–4 min. The actual, configuration dependent, maximum time is: 3 x 1–12a max–4 min + A x 2 x 1. A is the number of extra T_C states added to either the first or second bus operation of the processor extension data operand transfer sequence.

Initial 80L286 Pin State During Reset



Notes: 1. Set-up time for RESET ↑ may be violated with the consideration that φ1 of the processor clock may begin one system CLK period later.

2. Set-up and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during φ1 or φ2.

3. The data bus is only guaranteed to be in three-state OFF at the time shown.

4. HOLD is acknowledged during RESET, causing HLDA to go active and the appropriate pins to float. If HOLD remains active while RESET goes inactive, the 80L286 remains in HOLD state and will not perform any bus accesses until HOLD is deactivated.

1990 BYTE AWARD OF EXCELLENCE

Am286™ZX/Am286LX Integrated Processors

PC-AT Motherboard-on-a-Chip

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

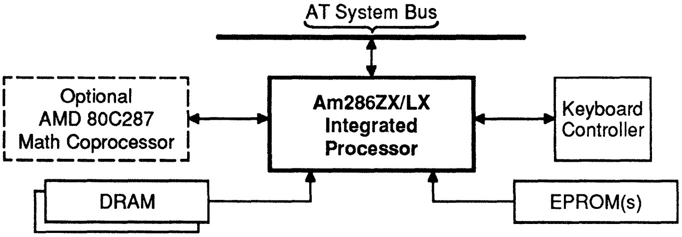
- **Integrates entire IBM PC-AT motherboard logic, plus enhancements**
  - –80C286 Microprocessor Core
  - –Enhanced Bus Controller
  - –Enhanced Clock Generator
  - –DMA Controllers
  - –Interrupt Controllers
  - –Counter/Timer
  - –Real Time Clock and CMOS Static RAM
- **Direct connection to DRAM, AMD 80C287™ math coprocessor, EPROMs, keyboard controller, and AT-bus slots eliminates the need for buffers and other glue logic**
- **Full hardware support for EMS 4.0 memory management with 128 EMS registers allows fast switching for multitasking systems**
- **100% compatible with the IBM PC-AT standard**
- **Low-power operation plus power saving features for battery powered notebook/hand-held systems (Am286LX processor only)**
- **Industry standard I/O port 92H fast reset and GATEA20 features for high-speed switching between real and protected mode**
- **Page mode/interleave DRAM Controller with direct interface to 256-kbit, 1-Mbit, and 4-Mbit DRAMs—supports up to 16 Mb of physical memory**
- **Complete line of support products available including demo boards, ICE, BIOS, and EMS drivers**
- **Flexible clock speeds up to 16 MHz**
- **28mm x 28mm, 216-Pin EIAJ Plastic Quad Flat Pack (PQFP), with socket available**

GENERAL DESCRIPTION

The Am286ZX and Am286LX integrated processors are AMD proprietary PC-AT motherboard-on-a-chip devices for personal computers. They integrate all of the logic functions from the original IBM PC-AT motherboard, plus enhancements, onto one chip. Included are the 80C286 microprocessor, all of the AT standard peripherals, and memory management to provide a high-performance, low-cost, low-power system solution for personal computers. The high level of integration provided by the Am286ZX/LX integrated processor allows designers to reduce the size, power consumption, and cost of a PC-AT compatible system, while increasing functionality and adding features.

The Am286ZX/LX integrated processor is ideal for design of desktop, notebook, handheld, embedded, and other industry standard AT personal computers, where performance, size, power consumption, and cost are critical factors. The Am286LX processor version provides additional power saving features including CPU shutdown mode, system shutdown mode, staggered DRAM refresh, and slow-refresh DRAM support.

Figure 1 shows a system block diagram for an AT motherboard, illustrating the high level of integration achieved by the Am286ZX/LX integrated processor.



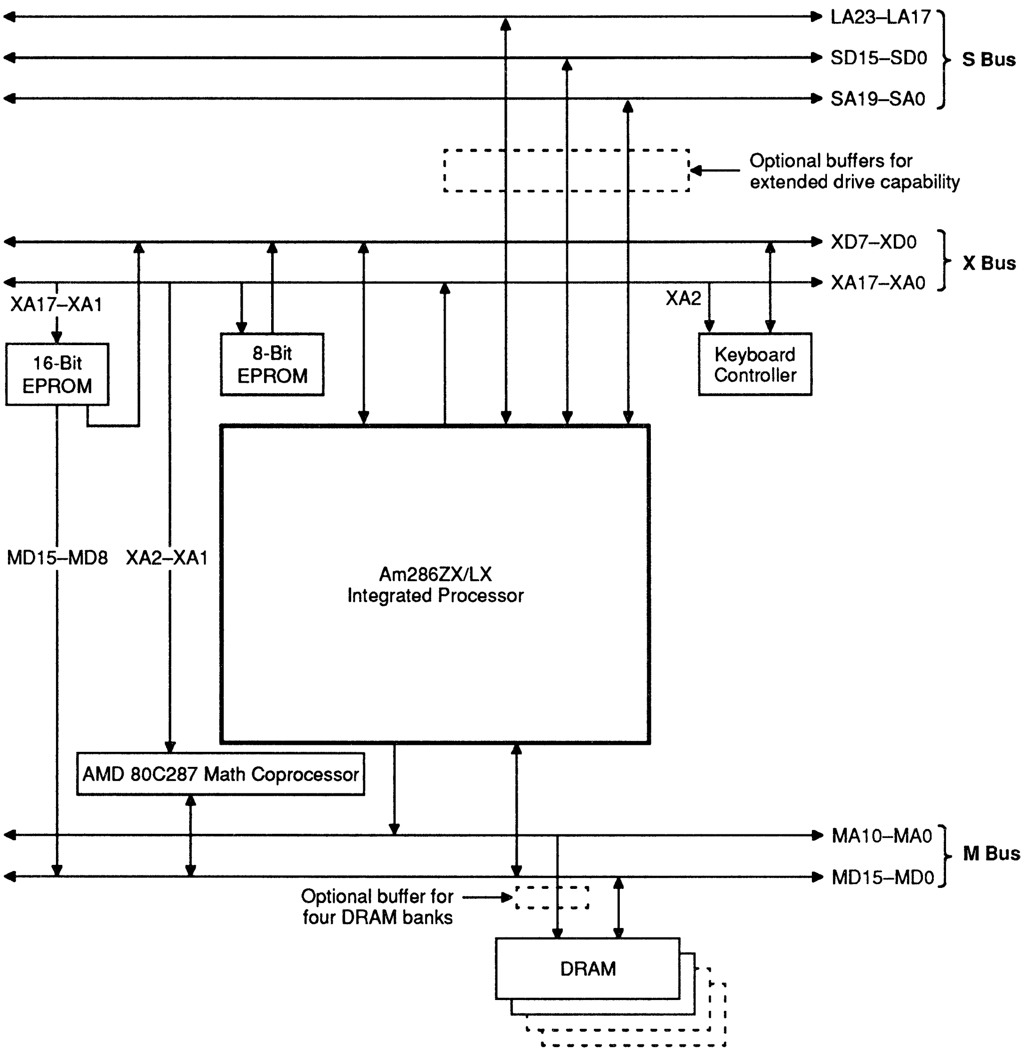
14753C–001

Figure 1. PC-AT System Block Diagram Using the Am286ZX/LX Integrated Processor

Publication #: 14753 Rev. D Amendment: /0
Issue Date: December 1991

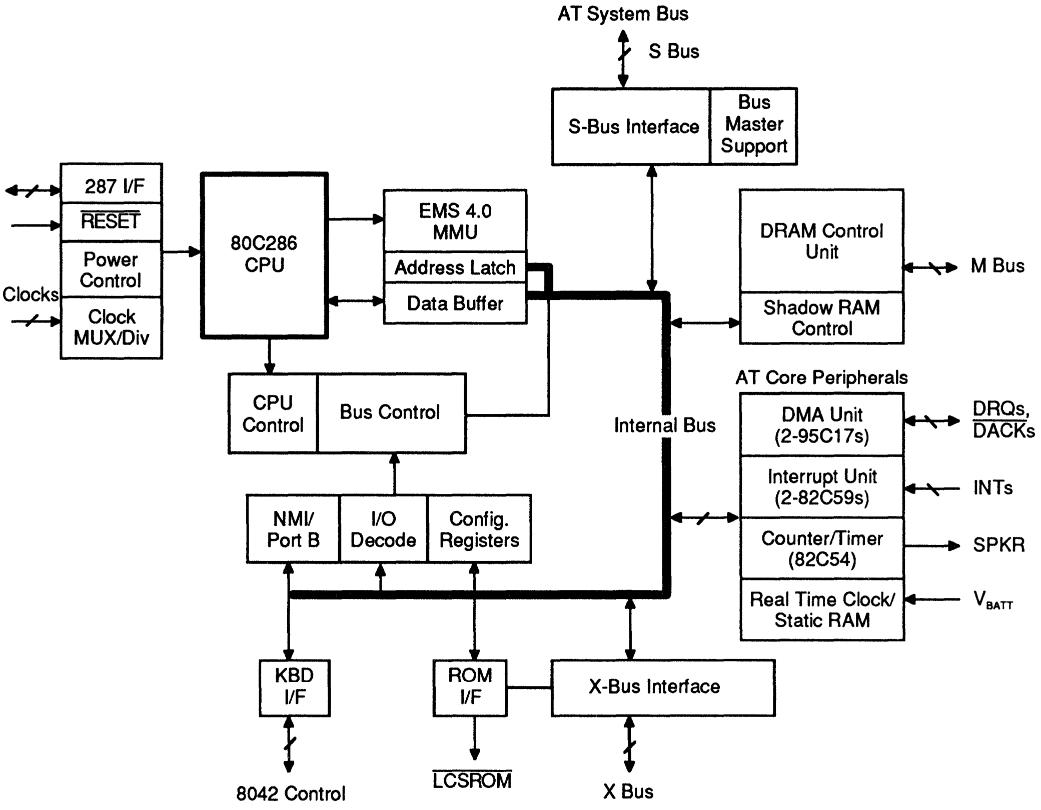
SYSTEM ARCHITECTURE

Figure 2 shows the Am286ZX/LX Integrated Processor Bus Interface. The AT System Bus (S Bus), Memory Bus (M Bus), and the I/O Bus (X Bus) directly interface to the rest of the system components. Optional buffers are shown for applications that require extended S-bus drive capability and/or four DRAM banks. The Am286ZX/LX integrated processor is designed to drive two bus slots and two DRAM banks, without requiring buffers.



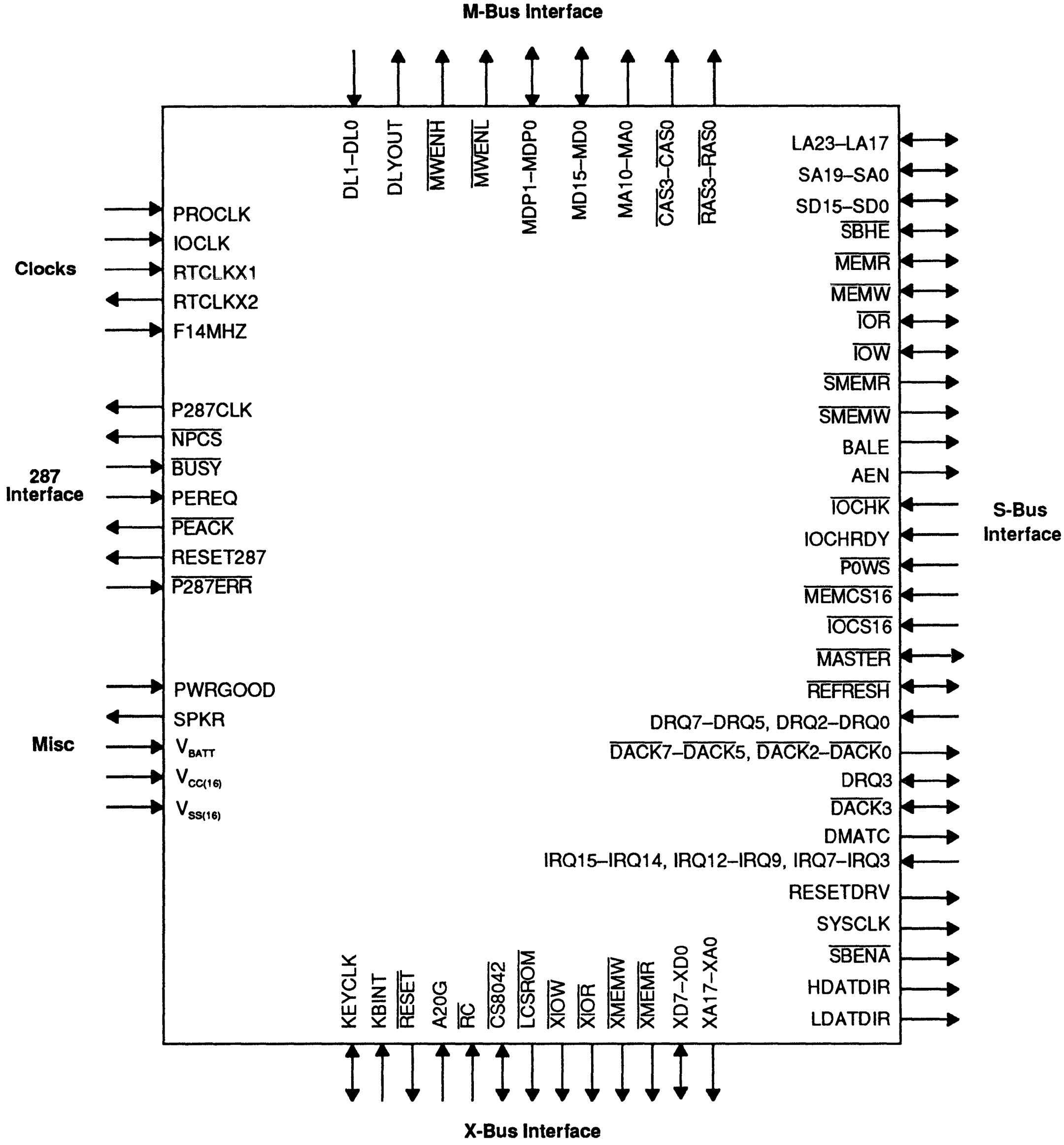
14753C–002

Figure 2. Am286ZX/LX Integrated Processor Bus Interface



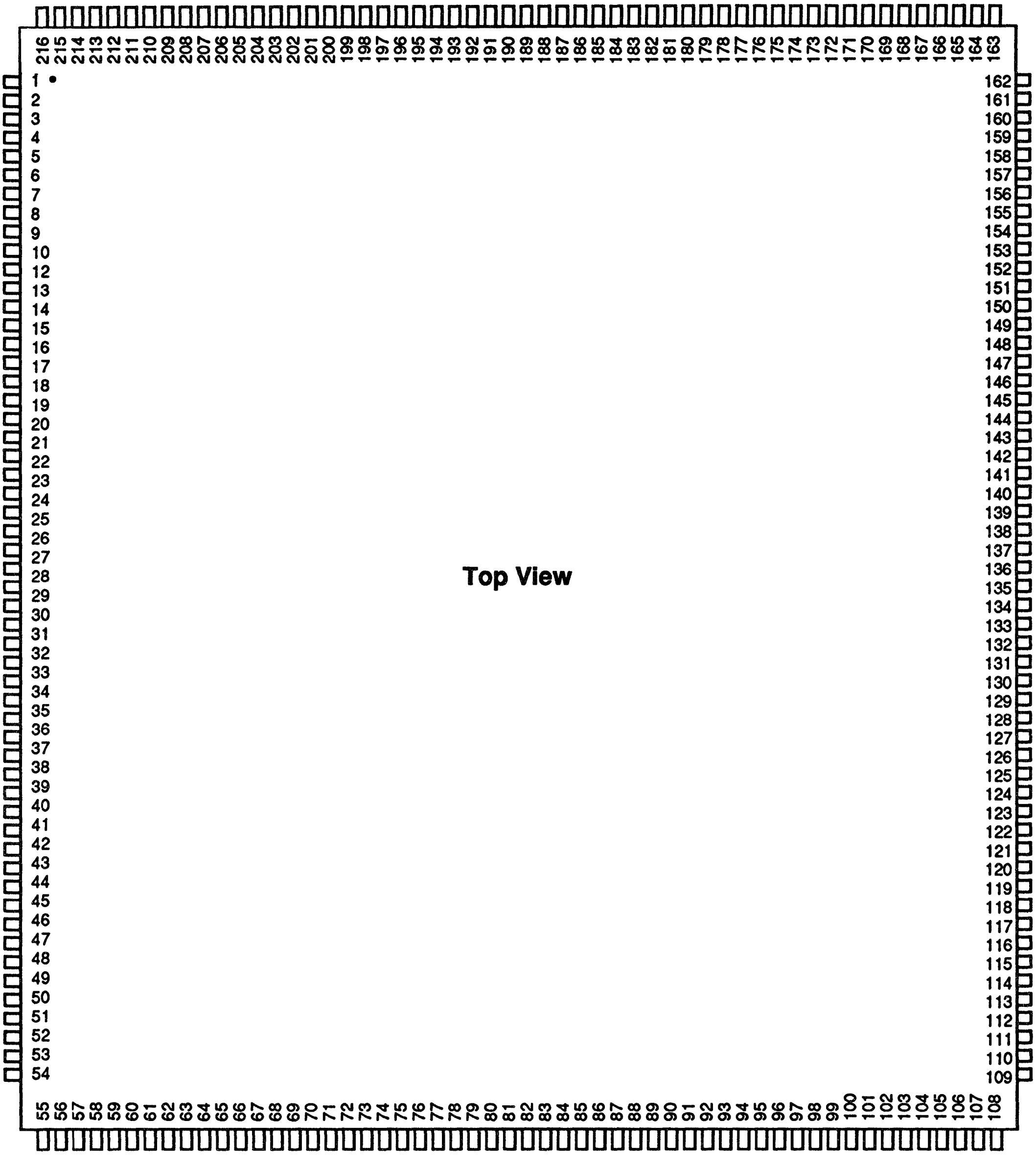
14753C–003

Figure 3. Am286ZX/LX Integrated Processor Internal Block Diagram



14753C–004

Figure 4. Am286ZX/LX Integrated Processor Logic Symbol



Note: Pin 1 is marked for orientation.

14753C–005

Figure 5. Am286ZX/LX Integrated Processor Connection Diagram

PIN DESIGNATION TABLE (Sorted by Pin Number)

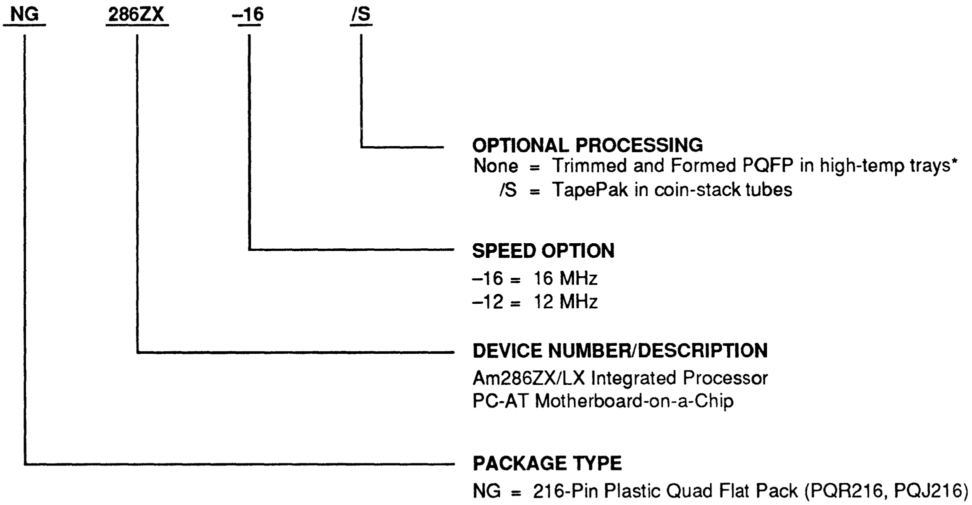
Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	LDATDIR	56	V_{CC}	111	SA18	166	MDP0
2	$\overline{\text{SBENA}}$	57	V_{SS}	112	LA18	167	MDP1
3	$\overline{\text{MASTER}}$	58	XA12	113	SA19	168	SD0
4	DRQ7	59	XA11	114	LA19	169	SD1
5	$\overline{\text{DACK7}}$	60	XA10	115	LA20	170	SD2
6	DRQ6	61	XA9	116	LA21	171	SD3
7	$\overline{\text{DACK6}}$	62	XA8	117	LA22	172	SD4
8	DRQ5	63	XA7	118	LA23	173	SD5
9	$\overline{\text{DACK5}}$	64	XA6	119	$\overline{\text{RAS0}}$	174	SD6
10	DRQ0	65	XA5	120	$\overline{\text{RAS1}}$	175	SD7
11	V_{CC}	66	XA4	121	V_{CC}	176	V_{CC}
12	V_{SS}	67	XA3	122	V_{SS}	177	V_{SS}
13	$\overline{\text{DACK0}}$	68	XA2	123	$\overline{\text{RAS2}}$	178	SD8
14	$\overline{\text{MEMR}}$	69	XA1	124	$\overline{\text{RAS3}}$	179	SD9
15	$\overline{\text{MEMW}}$	70	XA0	125	$\overline{\text{CAS0}}$	180	SD10
16	IRQ14	71	V_{CC}	126	$\overline{\text{CAS1}}$	181	SD11
17	IRQ15	72	V_{SS}	127	V_{CC}	182	SD12
18	IRQ12	73	MA0	128	V_{SS}	183	SD13
19	IRQ11	74	MA1	129	$\overline{\text{CAS2}}$	184	SD14
20	IRQ10	75	MA2	130	$\overline{\text{CAS3}}$	185	SD15
21	$\overline{\text{IOCS16}}$	76	MA3	131	$\overline{\text{MWENL}}$	186	V_{BATT}
22	$\overline{\text{MEMCS16}}$	77	MA4	132	$\overline{\text{MWENH}}$	187	RTCLKX1
23	$\overline{\text{SBHE}}$	78	V_{CC}	133	DLYOUT	188	RTCLKX2
24	BALE	79	V_{SS}	134	DL0	189	V_{CC}
25	DMATC	80	MA5	135	V_{CC}	190	V_{SS}
26	$\overline{\text{DACK2}}$	81	V_{CC}	136	V_{SS}	191	$\overline{\text{LCSROM}}$
27	V_{CC}	82	V_{SS}	137	DL1	192	$\overline{\text{XMEMW}}$
28	V_{SS}	83	MA6	138	XD0	193	$\overline{\text{XMEMR}}$
29	IRQ3	84	MA7	139	XD1	194	$\overline{\text{CS8042}}$
30	IRQ4	85	MA8	140	XD2	195	$\overline{\text{XIOR}}$
31	IRQ5	86	MA9	141	XD3	196	$\overline{\text{XIOW}}$
32	IRQ6	87	MA10	142	XD4	197	$\overline{\text{RC}}$
33	IRQ7	88	V_{CC}	143	XD5	198	A20G
34	$\overline{\text{REFRESH}}$	89	V_{SS}	144	XD6	199	$\overline{\text{RESET}}$
35	DRQ1	90	SA0	145	XD7	200	KEYCLK
36	SYSCLK	91	SA1	146	V_{CC}	201	KBINT
37	$\overline{\text{DACK1}}$	92	SA2	147	V_{SS}	202	PWRGOOD
38	DRQ3	93	SA3	148	MD0	203	PROCLK
39	$\overline{\text{DACK3}}$	94	SA4	149	MD1	204	IOCLK
40	$\overline{\text{IOW}}$	95	SA5	150	MD2	205	F14MHZ
41	$\overline{\text{IOR}}$	96	SA6	151	MD3	206	V_{CC}
42	$\overline{\text{SMEMW}}$	97	SA7	152	MD4	207	V_{SS}
43	$\overline{\text{SMEMR}}$	98	SA8	153	MD5	208	SPKR
44	IOCHRDY	99	SA9	154	MD6	209	P287CLK
45	AEN	100	SA10	155	MD7	210	$\overline{\text{NPCS}}$
46	$\overline{\text{P0WS}}$	101	SA11	156	MD8	211	$\overline{\text{BUSY}}$
47	DRQ2	102	SA12	157	MD9	212	PEREQ
48	IRQ9	103	SA13	158	MD10	213	$\overline{\text{PEACK}}$
49	RESETDRV	104	SA14	159	MD11	214	RESET287
50	$\overline{\text{IOCHK}}$	105	SA15	160	V_{CC}	215	$\overline{\text{P287ERR}}$
51	XA17	106	SA16	161	V_{SS}	216	HDATDIR
52	XA16	107	SA17	162	MD12		
53	XA15	108	V_{CC}	163	MD13		
54	XA14	109	V_{SS}	164	MD14		
55	XA13	110	LA17	165	MD15		

PIN DESIGNATION TABLE (Sorted by Pin Name)

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A20G	198	LA23	118	RESETDRV	49	V_{cc}	121
AEN	45	$\overline{\text{LCSROM}}$	191	RTCLKX1	187	V_{cc}	127
BALE	24	LDATDIR	1	RTCLKX2	188	V_{cc}	135
$\overline{\text{BUSY}}$	211	MA0	73	SA0	90	V_{cc}	146
$\overline{\text{CAS}}$0	125	MA1	74	SA1	91	V_{cc}	160
$\overline{\text{CAS}}$1	126	MA2	75	SA2	92	V_{cc}	176
$\overline{\text{CAS}}$2	129	MA3	76	SA3	93	V_{cc}	189
$\overline{\text{CAS}}$3	130	MA4	77	SA4	94	V_{cc}	206
$\overline{\text{CS8042}}$	194	MA5	80	SA5	95	V_{ss}	12
$\overline{\text{DACK}}$0	13	MA6	83	SA6	96	V_{ss}	28
$\overline{\text{DACK}}$1	37	MA7	84	SA7	97	V_{ss}	57
$\overline{\text{DACK}}$2	26	MA8	85	SA8	98	V_{ss}	72
$\overline{\text{DACK}}$3	39	MA9	86	SA9	99	V_{ss}	79
$\overline{\text{DACK}}$5	9	MA10	87	SA10	100	V_{ss}	82
$\overline{\text{DACK}}$6	7	$\overline{\text{MASTER}}$	3	SA11	101	V_{ss}	89
$\overline{\text{DACK}}$7	5	MD0	148	SA12	102	V_{ss}	109
DL0	134	MD1	149	SA13	103	V_{ss}	122
DL1	137	MD2	150	SA14	104	V_{ss}	128
DLYOUT	133	MD3	151	SA15	105	V_{ss}	136
DMATC	25	MD4	152	SA16	106	V_{ss}	147
DRQ0	10	MD5	153	SA17	107	V_{ss}	161
DRQ1	35	MD6	154	SA18	111	V_{ss}	177
DRQ2	47	MD7	155	SA19	113	V_{ss}	190
DRQ3	38	MD8	156	$\overline{\text{SBENA}}$	2	V_{ss}	207
DRQ5	8	MD9	157	$\overline{\text{SBHE}}$	23	XA0	70
DRQ6	6	MD10	158	SD0	168	XA1	69
DRQ7	4	MD11	159	SD1	169	XA2	68
F14MHZ	205	MD12	162	SD2	170	XA3	67
HDATDIR	216	MD13	163	SD3	171	XA4	66
$\overline{\text{IOCHK}}$	50	MD14	164	SD4	172	XA5	65
IOCHRDY	44	MD15	165	SD5	173	XA6	64
IOCLK	204	MDP0	166	SD6	174	XA7	63
$\overline{\text{IOCS16}}$	21	MDP1	167	SD7	175	XA8	62
$\overline{\text{IOR}}$	41	$\overline{\text{MEMCS16}}$	22	SD8	178	XA9	61
$\overline{\text{IOW}}$	40	$\overline{\text{MEMR}}$	14	SD9	179	XA10	60
IRQ3	29	$\overline{\text{MEMW}}$	15	SD10	180	XA11	59
IRQ4	30	$\overline{\text{MWENH}}$	132	SD11	181	XA12	58
IRQ5	31	$\overline{\text{MWENL}}$	131	SD12	182	XA13	55
IRQ6	32	$\overline{\text{NPCS}}$	210	SD13	183	XA14	54
IRQ7	33	$\overline{\text{P0WS}}$	46	SD14	184	XA15	53
IRQ9	48	P287CLK	209	SD15	185	XA16	52
IRQ10	20	$\overline{\text{P287ERR}}$	215	$\overline{\text{SMEMR}}$	43	XA17	51
IRQ11	19	$\overline{\text{PEACK}}$	213	$\overline{\text{SMEMW}}$	42	XD0	138
IRQ12	18	PEREQ	212	SPKR	208	XD1	139
IRQ14	16	PROCLK	203	SYSCLK	36	XD2	140
IRQ15	17	PWRGOOD	202	V_{BATT}	186	XD3	141
KBINT	201	$\overline{\text{RAS}}$0	119	V_{cc}	11	XD4	142
KEYCLK	200	$\overline{\text{RAS}}$1	120	V_{cc}	27	XD5	143
LA17	110	$\overline{\text{RAS}}$2	123	V_{cc}	56	XD6	144
LA18	112	$\overline{\text{RAS}}$3	124	V_{cc}	71	XD7	145
LA19	114	$\overline{\text{RC}}$	197	V_{cc}	78	$\overline{\text{XIOR}}$	195
LA20	115	$\overline{\text{REFRESH}}$	34	V_{cc}	81	$\overline{\text{XIOW}}$	196
LA21	116	$\overline{\text{RESET}}$	199	V_{cc}	88	$\overline{\text{XMEMR}}$	193
LA22	117	RESET287	214	V_{cc}	108	$\overline{\text{XMEMW}}$	192

ORDERING INFORMATION
Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the items below:



NG 286ZX -16 /S

OPTIONAL PROCESSING
None = Trimmed and Formed PQFP in high-temp trays*
/S = TapePak in coin-stack tubes

SPEED OPTION
–16 = 16 MHz
–12 = 12 MHz

DEVICE NUMBER/DESCRIPTION
Am286ZX/LX Integrated Processor
PC-AT Motherboard-on-a-Chip

PACKAGE TYPE
NG = 216-Pin Plastic Quad Flat Pack (PQR216, PQJ216)

Valid Combinations	
NG286LX	–12, –12/S
	–16, –16/S
NG286ZX	–12, –12/S
	–16, –16/S

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, and to check on newly released valid combinations.

*Note: Order Trimmed and Formed PQFP in multiples of 24 units/tray, 4 trays/dry pack.

PIN DESCRIPTION

Clocks

PROCLK
Processor Clock (Input)

Processor clock supplies a clock to the clock divider and multiplexer logic.

IOCLK
I/O Clock (Input)

I/O clock supplies a clock to the clock divider and multiplexer logic.

F14MHZ
14.318-MHz Frequency (Input)

14.318-MHz Frequency input to a divider that generates the clock for the internal 82C54 counter/timer.

RTCLKX1
RTC X1 Crystal Connection (Input)

Input to the crystal oscillator that supplies the internal real-time clock's 32-kHz frequency.

RTCLKX2
RTC X2 Crystal Connection (Output)

Output of the crystal oscillator that supplies the internal real-time clock's 32-kHz frequency.

AT System Bus (S Bus) Interface

SYSCLK
System Clock (Output)

System Clock is the S-bus synchronous clock. It is always half the frequency of the AT state machine clock.

LA23–LA17
Extended Address (Inputs/Outputs)

Extended Address lines are used to address up to 16 Mb of memory on the S Bus. These signals are not latched internally with BALE and are valid earlier in the cycle than the SA lines. These pins can be driven by the internal CPU or DMA controller or by an external bus master.

SA19–SA0
S-Bus Address (Inputs/Outputs)

S-Bus Address lines are used to address memory and I/O devices on the S Bus and are valid for the entire bus cycle. These pins can be driven by the internal CPU or DMA controller or by an external bus master.

$\overline{\text{SBHE}}$
S-Bus Byte High Enable (Input/Output; Active Low)

S-Bus Byte High Enable signal indicates a transfer of data on the upper byte of the data bus, SD15–SD8, to a 16-bit, S-bus peripheral. This signal can be driven by the internal CPU or DMA controller or by an external bus master.

SD15–SD0
S-Bus Data (Inputs/Outputs)

S-Bus Data lines are used to transfer data on the S Bus. All 8-bit devices use SD7–SD0. All 16-bit devices use SD15–SD0. Transfers may be initiated by the internal CPU or DMA controller or by an external bus master.

BALE
Buffered Address Latch Enable (Output; Active High)

Buffered Address Latch Enable is an active High output used to latch valid addresses and memory decodes during CPU transfer cycles. BALE is forced High during DMA and external bus master cycles.

$\overline{\text{IOW}}$
I/O Write (Input/Output; Active Low)

I/O Write is an active Low control signal for I/O write cycles on the S Bus. This signal can be driven by the internal CPU or DMA controller or by an external bus master.

$\overline{\text{IOR}}$
I/O Read (Input/Output; Active Low)

I/O Read is an active Low control signal for I/O read cycles on the S Bus. This signal can be driven by the internal CPU or DMA controller or by an external bus master.

$\overline{\text{MEMW}}$
Memory Write (Input/Output; Active Low)

Memory Write is an active Low control signal for all memory write cycles on the S Bus. This signal can be driven by the internal CPU or DMA controller or by an external bus master.

$\overline{\text{MEMR}}$
Memory Read (Input/Output; Active Low)

Memory Read is an active Low control signal for all memory read cycles on the S Bus. This signal can be driven by the internal CPU or DMA controller or by an external bus master.

$\overline{\text{SMEMW}}$
S-Memory Write (Output; Active Low)

S-Memory Write is an active Low control signal for write cycles to S-bus memory with addresses less than 1Mb.

$\overline{\text{SMEMR}}$
S-Memory Read (Output; Active Low)

S-Memory Read is an active Low control signal for read cycles to S-bus memory with addresses less than 1Mb.

$\overline{\text{MEMCS16}}$

Memory 16-Bit Chip Select (Input; Active Low)

Memory 16-Bit Chip Select is an active Low input signal indicating that the present S-bus transfer cycle is a 16-bit memory cycle. This pin should be driven by open collector or three-state buffers. There is an internal pull-up on the pin which can be disabled.

$\overline{\text{IOCS16}}$

I/O 16-Bit Chip Select (Input; Active Low)

I/O 16-Bit Chip Select is an active Low input signal indicating that the present S-bus transfer cycle is a 16-bit I/O cycle. This pin is also sampled at reset time to determine ROM size. This pin should be driven by open collector or three-state buffers. There is an internal pull-up on the pin which can be disabled.

$\overline{\text{POWS}}$

Zero Wait State (Input; Active Low)

Zero Wait State is an active Low input signal indicating to the bus controller that it can complete the present bus cycle without inserting any additional wait states. This pin should be driven by open collector or three-state buffers. There is an internal pull-up on the pin which can be disabled.

$\overline{\text{IOCHK}}$

I/O Channel Check (Input; Active Low)

I/O Channel Check is an active Low input from the S Bus which can cause a NMI to be generated to the internal CPU, indicating a I/O error condition on the S Bus. This pin should be driven by open collector or three-state buffers. There is an internal pull-up on the pin which can be disabled.

IOCHRDY

I/O Channel Ready (Input; Active High)

I/O Channel Ready is an active High signal from the S Bus. When Low it indicates a "not ready" condition and inserts wait states in AT I/O or memory cycles. When High, it allows the current S-bus cycle to complete. This pin should be driven by open collector or three-state buffers. There is an internal pull-up on the pin which can be disabled.

AEN

Address Enable (Output; Active High)

Address Enable is an active High output signal used to indicate to S-bus I/O device address decoders that a DMA cycle is in progress. When active, the internal DMA controller has control of the S-bus address, data, and control pins.

$\overline{\text{MASTER}}$

Master (Input/Output; Active Low)

$\overline{\text{MASTER}}$ is usually an active Low input signal asserted by an external device on the S Bus to allow S-bus peripherals to access system resources as a bus master. In Bus Master Mode, this pin is an output asserted when a mastering cycle is in progress. This pin should be driven by open collector or three-state buffers. There is an internal pull-up on the pin which can be disabled.

DRQ7–DRQ5, DRQ2–DRQ0

DMA Request (Inputs; Active High)

DMA Request signals 7–5 and 2–0 are asynchronous DMA channel request inputs used by peripheral devices to gain access to a DMA service. These pins can also be used by bus masters to gain S-bus control. DRQ2–DRQ0 perform 8-bit DMA transfers and DRQ7–DRQ5 perform 16-bit DMA transfers. There are weak pull-downs on these pins.

DRQ3

DMA Request (Input/Output)

DMA Request 3 is usually defined the same as the other DRQ inputs. In Bus Master Mode, this pin is an output asserted to request bus access on a host system's bus. There is a weak pull-down on this pin.

$\overline{\text{DACK7}}$–$\overline{\text{DACK5}}$, $\overline{\text{DACK2}}$–$\overline{\text{DACK0}}$

DMA Acknowledge (Outputs; Active Low)

DMA Acknowledge signals 7–5 and 2–0 are active Low output pins which acknowledge their corresponding DMA requests.

$\overline{\text{DACK3}}$

DMA Acknowledge 3 (Input/Output; Active Low)

DMA Acknowledge 3 is usually defined the same as the other $\overline{\text{DACK}}$ outputs. In Bus Master Mode, this pin is an input which senses the status of a bus request to the host system.

DMATC

DMA Terminal Count (Output; Active High)

DMA Terminal Count is an active High output signal indicating that terminal count for a DMA channel has been reached.

$\overline{\text{REFRESH}}$

Refresh (Input/Output; Active Low)

$\overline{\text{REFRESH}}$ is an active Low signal to indicate a memory refresh cycle. This signal can be driven by an external bus master. There is an internal pull-up on the pin which can be disabled.

IRQ15–IRQ14, IRQ12–IRQ9, IRQ7–IRQ3
Interrupt Request (Inputs; Active Low)

Interrupt Request input pins signal the internal 82C59 interrupt controllers that an I/O device needs attention. There are weak pull-ups on these pins.

RESETDRV
Reset Drive (Output; Active High)

Reset Drive is an active High signal used to reset or initialize system logic at power-up time. It is synchronous to SYSCLK.

$\overline{\text{SBENA}}$
S-Bus Buffer Enable (Output; Active Low)

S-Bus Buffer Enable is an active Low enable for the S-bus address and data expansion buffers.

HDATDIR
High Data Direction (Output; Active High)

High Data Direction is an active High direction control pin for the high byte of the S-bus data buffer. A logic High on this pin indicates a drive direction out from the chip.

LDATDIR
Low Data Direction (Output; Active High)

Low Data Direction is an active High direction control pin for the low byte of the S-bus data buffer. A logic High on this pin indicates a drive direction out from the chip.

DRAM Interface

$\overline{\text{RAS3}}$–$\overline{\text{RAS0}}$
Row Address Strobe (Outputs; Active Low)

Row Address Strobes are active Low outputs used by the DRAMs as RAS signals to clock in row addresses for each of the 4 possible DRAM banks.

$\overline{\text{CAS3}}$–$\overline{\text{CAS0}}$
Column Address Strobe (Outputs; Active Low)

Column Address Strobes are active Low outputs used by the DRAMs as CAS signals to clock in column addresses for each of the 4 possible DRAM banks.

MA10–MA0
Memory Address (Outputs)

Memory Address lines are multiplexed outputs and convey the following information: row addresses during RAS signals, column addresses during CAS signals, and refresh addresses during refresh cycles. 256-kbit DRAMs use MA8–MA0; 1-Mbit DRAMs use MA9–MA0; and, 4-Mbit DRAMs use MA10–MA0.

MD15–MD0
Memory Data (Inputs/Outputs)

Memory Data bus transfers data to/from the DRAMs and the AMD 80C287 math coprocessor. The high byte of the bus, MD15–MD8, is also used for transfers involving 16-bit ROM and 16-bit X-bus I/O devices.

MDP1–MDP0
Memory Data Parity (Inputs/Outputs)

Memory Data Parity bits 1 and 0 transfer parity bit information to/from the optional parity DRAM. Parity is generated and verified internally. MDP0 is the parity for the low byte and MDP1 is the parity for the high byte.

$\overline{\text{MWENH}}$
Memory Write Enable High (Output; Active Low)

Memory Write Enable High is an active Low output for the high byte DRAM write enable.

$\overline{\text{MWENL}}$
Memory Write Enable Low (Output; Active Low)

Memory Write Enable Low is an active Low output for the low byte DRAM write enable.

DLYOUT
Delay Line Out (Output; Active High)

Delay Line Out is an active High output to the delay line for generating DRAM control signals.

DL1–DL0
Delay Line (Inputs; Active High)

Delay Line inputs 1 and 0 are active High inputs from two taps of a delay line that generate DRAM control signals.

X-Bus Interface

XA17–XA0
X-Bus Address (Outputs)

X-Bus Address lines provide addressing for the BIOS ROM, keyboard controller, AMD 80C287 math coprocessor, and other X-bus peripherals.

XD7–XD0
X-Bus Data (Inputs/Outputs)

X-Bus Data lines are used to transfer the low 8 bits of data to/from X-bus devices such as the keyboard controller, BIOS ROM, and other X-bus peripherals.

$\overline{\text{XIOW}}$

X-Bus I/O Write (Output; Active Low)

X-Bus I/O Write is an active Low control signal directing an I/O port to accept data from the XD Bus.

$\overline{\text{XIOR}}$

X-Bus I/O Read (Output; Active Low)

X-Bus I/O Read is an active Low control signal directing an I/O port to place data on the XD Bus.

$\overline{\text{XMEMW}}$

X-Bus Memory (Output; Active Low)

X-Bus Memory Write is an active Low control signal for memory write cycles on the X Bus.

$\overline{\text{XMEMR}}$

X-Bus Memory Read (Output; Active Low)

X-Bus Memory Read is an active Low control signal for memory read cycles on the X Bus, such as BIOS ROM reads.

Coprocessor Interface

P287CLK

287 Clock (Output)

287 Clock provides the output clock to the math coprocessor (AMD 80C287 math coprocessor). The clock is generated by the clock divider and multiplexer logic, and is derived from PROCLK or IOCLK.

$\overline{\text{NPCS}}$

Numeric Processor Chip Select (Output; Active Low)

Numeric Processor Chip Select is an active Low output which indicates a data transfer involving the AMD 80C287 math coprocessor.

$\overline{\text{BUSY}}$

Busy (Input; Active Low)

$\overline{\text{BUSY}}$ is an active Low input from the AMD 80C287 math coprocessor indicating it is executing an instruction. There is a weak internal pull-up resistor on this pin.

PEREQ

Processor Extension Request (Input; Active High)

Processor Extension Request is an active High input pin which indicates that the AMD 80C287 math coprocessor is ready to transfer data to/from the CPU. There is a weak pull-down on this pin.

$\overline{\text{PEACK}}$

Processor Extension Acknowledge (Output; Active Low)

Processor Extension Acknowledge is an active Low output which indicates that the requested transfer due to an active PEREQ has been completed.

RESET287

287 Reset (Output; Active High)

287 Reset is an active High output signal which resets the AMD 80C287 math coprocessor.

$\overline{\text{P287ERR}}$

287 Error (Input; Active Low)

287 Error is an active Low input which indicates an AMD 80C287 math coprocessor error condition and can trigger the coprocessor interrupt to the interrupt controller. There is a weak pull-up on this pin.

Keyboard Controller Interface

$\overline{\text{CS8042}}$

8042 Chip Select (Input/Output; Active Low)

8042 Chip Select is an active Low signal used for selecting the external keyboard controller. If the XT keyboard interface is enabled, the pin definition changes to XTKBDATA, a bi-directional serial data transfer line with an internal pull-up.

$\overline{\text{RC}}$

Reset CPU (Input; Active Low)

Reset CPU is an active Low input signal which will activate the internal CPU's reset when active. This is generated from the 8042 keyboard controller.

A20G

Address 20 Gate (Input)

Address 20 Gate input for CPU address line 20. A logic High on this input will enable A20 pass-through from the CPU. A logic Low will force the internal A20 inactive. This signal is generated by the 8042 keyboard controller.

$\overline{\text{RESET}}$

Reset (Output; Active Low)

$\overline{\text{RESET}}$ is an active Low output, CPU clock synchronized reset signal derived from PWRGOOD. It is usually used to reset the keyboard controller. If the XT keyboard interface is enabled, the pin definition changes to XTKBRST, an I/O controlled reset output for an XT keyboard.

KEYCLK

Keyboard Clock (Input/Output)

Keyboard Clock provides the output clock to the keyboard controller. The clock is generated by the clock divider and multiplexer logic and can be derived from several clock sources. If the XT keyboard interface is enabled, the pin definition changes to XTKBCLK, an I/O controlled clock line with an internal pull-up.

KBINT

Keyboard Interrupt (Input)

Keyboard Interrupt is a direct connection to the IRQ1 input on the master 82C59A.

Miscellaneous Signals

$\overline{\text{LCSROM}}$
ROM Chip Select (Output; Active Low)

ROM Chip Select is an active Low output which provides the chip select for the BIOS ROM/EPROM.

PWRGOOD
Power Good (Input; Active High)

Power Good is an active High input signal which indicates a stable system power condition. This is normally driven by the power supply. A logic Low on this input resets the device. There is a schmitt trigger input on this pin.

SPKR
Speaker (Output)

Speaker output signal is the output of the internal tone generation logic which includes one channel of the internal 82C54 timer. This is a standard TTL level output.

V_{BATT}

Negative Battery Voltage input to power the internal Real Time Clock and CMOS memory.

V_{CC}

Power.

V_{SS}

Ground.

FUNCTIONAL DESCRIPTION

80C286 and Standard Peripherals

80C286 Microprocessor Core

The Am286ZX/LX integrated processor integrates AMD's CMOS 80C286 microprocessor as the core of the internal system. The 80C286 core maintains the complete functional features of a standard 80C286. This includes full 80286 instruction set, registers, status information, internal memory organization, 16 Mb of physical and 1 Gb of virtual address space, all addressing modes, address and data segmentation, pipelining schemes, real and protected mode environments with four levels of memory protection, interrupt priorities, and halt and shut down cycles. See the 80C286 data sheet (order #11625) for more information.

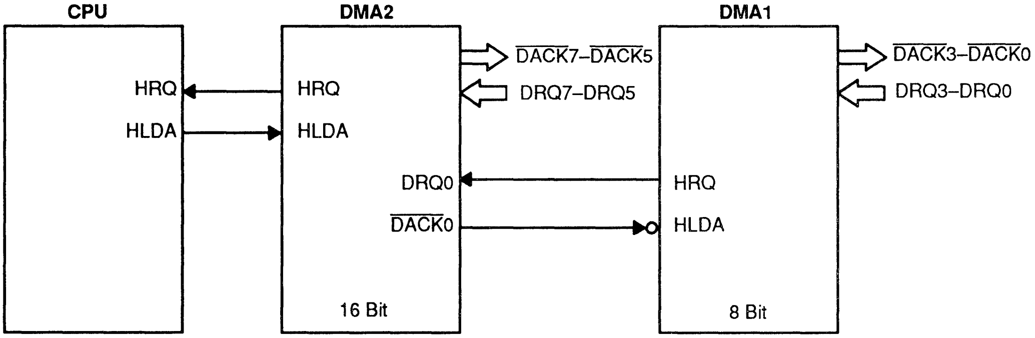
In the Am286ZX/LX integrated processor, the core is designed to be fully static. This means that the Am286ZX/LX integrated processor can operate anywhere from its maximum speed down to 0 MHz. Power consumption is significantly reduced by lowering the clock speed. For maximum power saving, the clock may be shut off completely. The Am286ZX/LX integrated processor retains its state while the processor and peripheral clocks are stopped, and then continues to operate from its original state when these clocks are resumed.

DMA Controllers

Two identical 9517/8237-compatible DMA controllers and a page register are integrated into the Am286ZX/LX integrated processor. DMA1 occupies addresses 000H–00FH and DMA2 occupies addresses 0C0H –0DFH. Each controller is a four channel DMA device which generates the memory addresses and control signals necessary to directly transfer information between a peripheral device and memory. This allows transfer of information at higher speeds with little CPU intervention. The two DMA controllers are internally cascaded to provide four DMA channels for 8-bit transfers (DMA1) and three DMA channels for 16-bit transfers (DMA2). DMA2 Channel 0 is internally cascaded.

The programmable registers in the Am286ZX/LX integrated processor allow independent control for both 8-bit and 16-bit transfers by inserting wait states. Each DMA channel has a pair of 16-bit counters and a pair of reload registers for each counter. This allows up to 64-kb block transfers with DMA1 and up to 128-kb block transfers with DMA2. Several modes of operations are possible using programmable features in the registers.

The page registers are used to generate the high order addresses during DMA cycles. Only eight of these registers are used, but all 16 are included to maintain PC-AT compatibility. Each DMA channel has a register associated with it, with the exception of Channel 0 of DMA2. The Am286ZX/LX integrated processor includes weak pull-downs on the DRQ input pins.



14753C–006

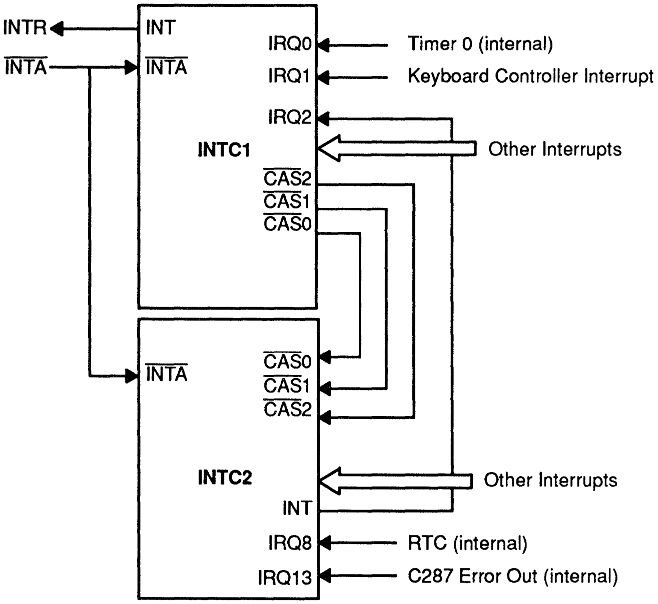
Figure 6. DMA Block Diagram

Interrupt Controllers

Two identical, 8259-compatible interrupt controllers, INTC1 and INTC2, are integrated into the Am286ZX/LX integrated processor. They accept requests from peripherals, resolve priority on pending interrupts and interrupts in service, issue an interrupt request to the processor, and provide interrupt vectors for interrupt service routines.

The two devices are internally connected and must be programmed to operate in cascade mode for operation of all 15 interrupt channels. INTC1 occupies addresses 020H-021H and is configured for master operation in cascade mode. INTC2 occupies addresses 0A0H–0A1H and is configured for slave operation. The interrupt request output signal from INTC2 (INT) is internally connected to the interrupt request input Channel 2 (IRQ2) of INTC1. This configuration is compatible with the IBM PC-AT.

The output of Timer 0 in the counter/timer section is connected to Channel 0 (IRQ0) of INTC1. The interrupt request from the Real Time Clock is connected to Channel 0 (IRQ8) of INTC2. The other interrupts are also available to external peripherals, as in the AT architecture. The Am286ZX/LX integrated processor includes weak pull-ups on the interrupt input pins.



14753C–007

Figure 7. Interrupt Controller Block Diagram

Counter/Timer

A three-channel, general purpose, 82C54-compatible, 16-bit counter/timer is integrated into the Am286ZX/LX integrated processor. It provides critical timing parameters for the PC system under software control. It can be programmed to count in binary or in BCD. Each counter operates independently of the other two and can be programmed for operation as a timer or a counter. All three are controlled from a common set of control logic, which provides controls to load, read, configure, and control each counter. This counter occupies I/O addresses 040H–043H. There are six modes of operation:

Mode 0	Interrupt on terminal count
Mode 1	Hardware re-triggerable one-shot
Mode 2	Rate generator
Mode 3	Square wave generator
Mode 4	Software triggered strobe
Mode 5	Hardware re-triggered strobe

All three counters are driven from a common clock, which is internally generated as a divide by 12 of the F14MHZ pin. The output of Counter 0 is connected to IRQ0 interrupt input of INTC1, and may be used as a software "timer tick" for time-keeping and task-switching activities. The output of Counter 1 is internally connected as a refresh request. The output of Counter 2 is internally connected to speaker logic.

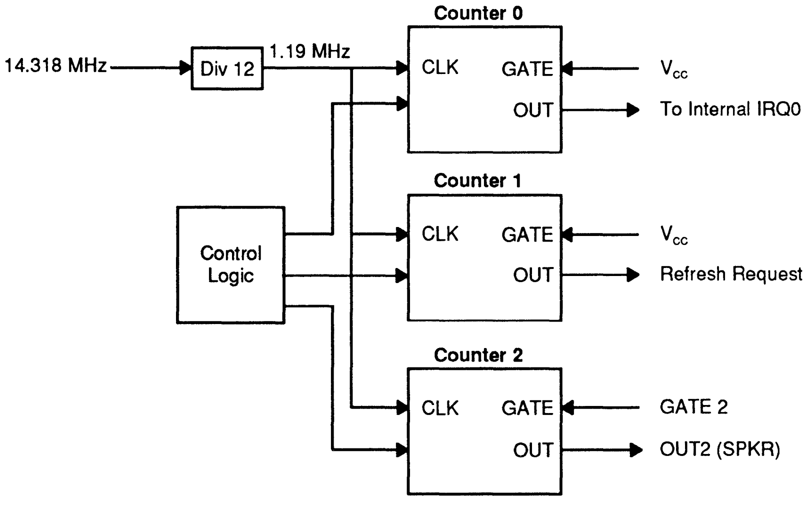
Real Time Clock and CMOS Static RAM

A Real Time Clock (RTC) function is implemented in the Am286ZX/LX integrated processor at I/O address space 070H and 071H. It combines a complete time-of-day clock with alarm, a one hundred year calendar, a programmable periodic interrupt, and 114 bytes of CMOS static RAM. The static RAM is battery backed-up to save its contents in the absence of main system power. Also, since it is battery backed-up, the clock counting continues to maintain the date and time when power is shut off .

The Am286ZX/LX integrated processor includes a low-power CMOS crystal oscillator to handle the 32-kHz clock signal to the RTC. This feature saves board space by integrating the oscillator on-chip, and it extends the life of the system's battery.

An indexed addressing scheme is implemented to write to the 128 locations of the RTC. The index register is written with the address of the memory location, which acts as an address pointer. Data is then transferred to the location. The RTC contains 128 addressable locations which include: 10 locations for time, calendar and alarm data, 4 general purpose registers, and 114 static RAM locations.

The alarm bytes are programmed to generate an interrupt at a specific time, or can be programmed to generate a periodic interrupt. The static RAM from index addresses 0EH to 7FH are not affected by the RTC. This area can be used to store configuration and calibration information. Since this section is battery powered, it will not lose data when the system is turned off.



14753C–008

Figure 8. Counter/Timer Block Diagram

System Control Logic

Enhanced Clock Generator

The Am286ZX/LX integrated processor provides very flexible clock selections for the CPU clock, AT-bus clock, keyboard clock, and AMD 80C287 math coprocessor clock. Configuration registers are provided to select these clocks to suit system applications. An 82284-compatible logic block is used to generate the READY signal for the internal CPU.

PROCLK and IOCLK are the main input clock pins for the clock generator. Depending on the system design, the Am286ZX/LX integrated processor can be configured to select internal CPU clock, AMD 80C287 math coprocessor clock, AT state machine clock, and keyboard clock from either PROCLK or IOCLK. Flexible divisors are provided to select desired operational speeds. SYSCLK is always half the frequency of the AT-state machine clock (ATSMCLK).

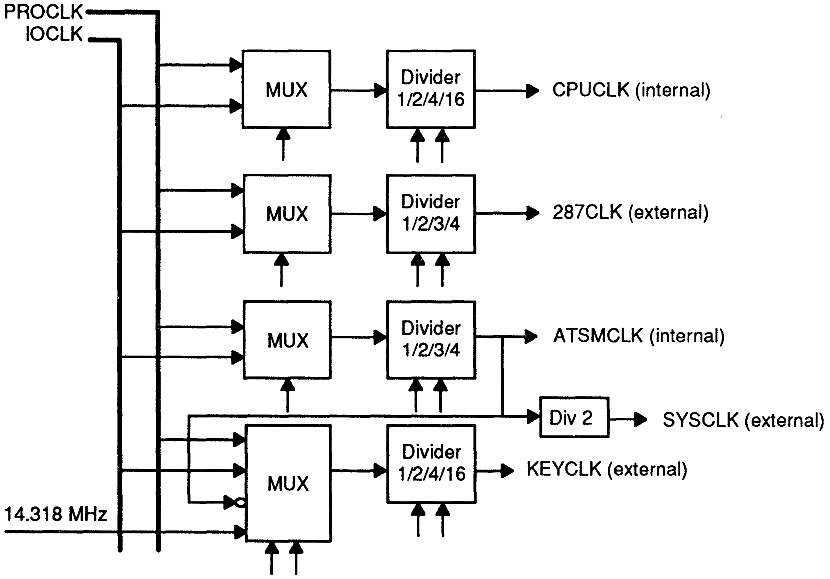
The dividers are at the divide by four setting at power up. The default selections for clock source and divider can be modified by writing to configuration registers. The Am286ZX/LX integrated processor contains logic to prevent clock pulse width violations when changing any clock.

In order to achieve maximum system performance, it is desirable to run the processor at the maximum rated speed. Internal state machines maintain asynchronous protocols for high-speed operation of the CPU while allowing slower S-bus cycles.

Enhanced Bus Controller

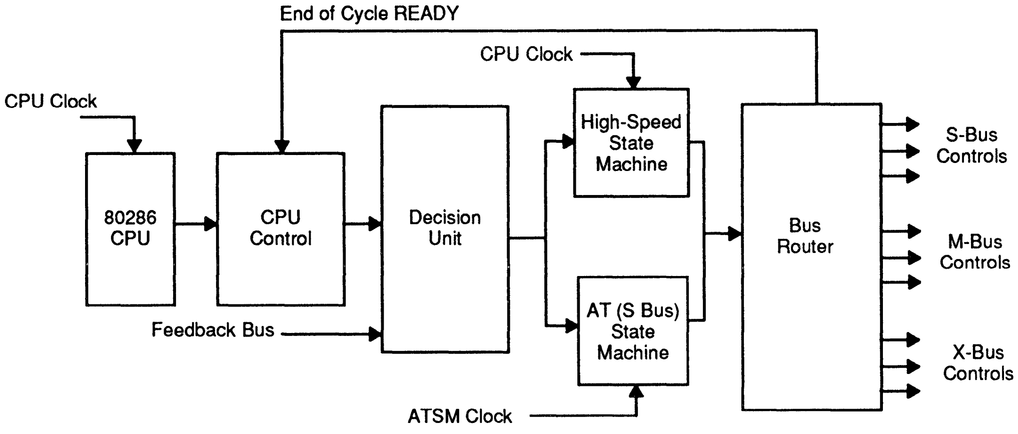
The Am286ZX/LX integrated processor's Enhanced Bus Controller provides the control logic for routing addresses and data through the Am286ZX/LX integrated processor for every type of data transaction. At its core, it provides analogous functions to the 82284 and 82288. The rest of the logic transforms the Am286ZX/LX integrated processor into advanced high-speed, AT-compatible products. These advanced functions include the following:

1. Dual state machine design—A high-speed CPU synchronous state machine for DRAM and internal register access, and an S-bus state machine which can operate asynchronously to the CPU.
2. AT-compatible, high-byte routing, and word-to-byte transaction conversion for bus transfers to 8-bit peripherals.
3. Address/data routing and buffer control for all possible data transfers, including CPU, DMA, Refresh, and bus mastering.



14753C–009

Figure 9. Clock Generator Block Diagram



14753C–010

Figure 10. Bus Controller Block Diagram

DRAM Interface

The Am286ZX/LX integrated processor supports a robust and easily implemented memory system providing zero or near zero wait state performance, while maintaining low system cost with inexpensive DRAMs. Its memory array is configured to directly support two banks of 16/18 bits (four 8/9 bit SIMMs). Two additional banks can be added by adding buffers on the memory address lines. Each bank consists of 16-bit data and two optional parity bits, one for each byte. They can be implemented with 256-kbit, 1-Mbit, or 4-Mbit DRAMs for a maximum of 8 Mb per bank, up to a total of 16 Mb in multiple banks.

The array supports page mode accesses to decrease memory speed requirements. If adjacent arrays are populated with the same DRAM size, two or four way interleaving may be selected to increase the page mode hit ratio. The Am286ZX/LX integrated processor can handle memory size mixing. If interleaving is enabled, only the same size banks are interleaved.

In page mode operation, consecutive accesses within DRAM page boundaries are reduced to zero wait states. Wait states occur only when changing pages or crossing page boundaries. In the case of identical banks, interleaving may be enabled to double (2 way) or quadruple (4 way) the page size. This increases the page mode hit ratio. The following table shows the average number of wait states, plus the recommended memory speeds.

CPU Speed	Mode	Average Wait States	DRAM Speed
12 MHz	Normal	0	80 ns
12 MHz	Page/Interleave	0.6	120 ns
12 MHz	Normal	1	100 ns
16 MHz	Normal	1	80 ns
16 MHz	Page/Interleave	0.6	100 ns

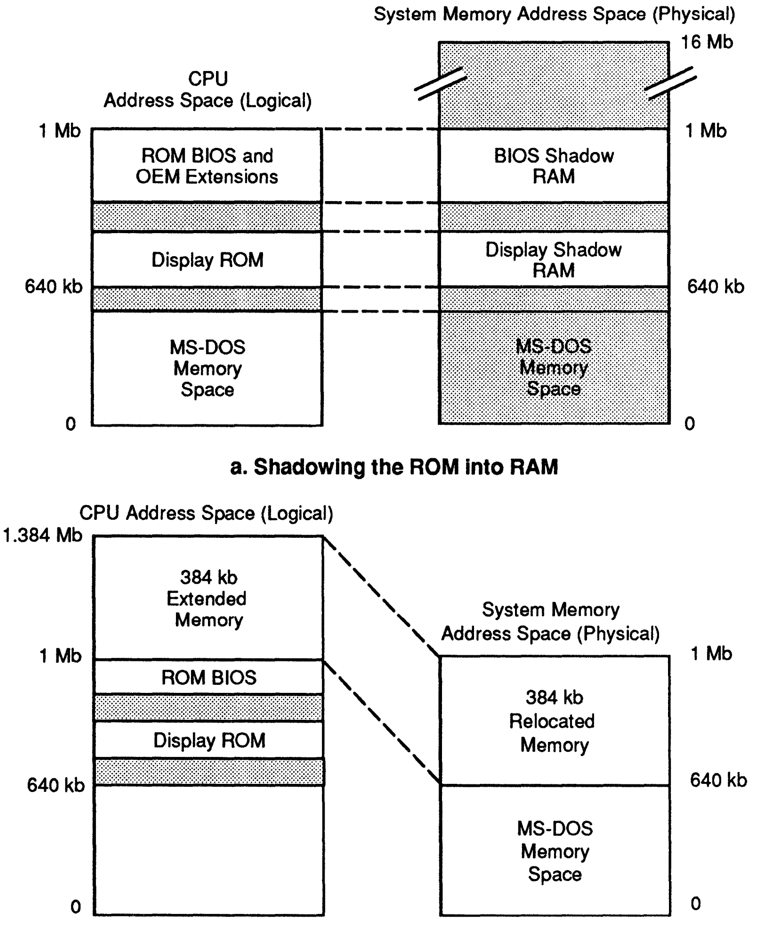
The Am286ZX/LX integrated processor can also interface to SRAM via the S Bus or the high-speed memory bus that normally connects to DRAM.

Shadow RAM/384-kb Relocation

Typically, system ROM BIOS and other ROM extensions require multiple wait states for access, presenting a significant system performance bottleneck. Shadow RAM is a method to eliminate this slow speed path. Using the ROM chip-select control facilities, slow access ROMs can be disabled and replaced with fast access write protected RAM in the 384 kb of unused RAM area. Three 8-bit registers provide Shadow RAM mapping control of 384-kb shadow memory. This area is divided into 24 blocks of 16 kb.

Upon system initialization, the BIOS copies itself and any ROM extensions into a temporary location in system memory. The ROMs are then disabled with the appropriate bits in the ROM control register, while at the same time the shadow RAM is enabled. The ROM contents are then copied into the RAM at original locations, and the appropriate 16-kb pages are write protected. Other sections of RAM can be moved and write protected for other system functions, such as consolidating ROM extensions and write protecting system scratch-pad RAM.

However, if only 1 Mb of RAM is present, the 384 kb between 640 kb and 1 Mb can be relocated to above the 1-Mb boundary, providing 384 kb of extended memory to the user. This feature is enabled by the REL bit in the DRAM Control Register (DRC). The Am286ZX/LX integrated processor provides the flexibility of offering either Shadow RAM or 384-kb relocation by setting the appropriate configuration registers. These two features are mutually exclusive.



a. Shadowing the ROM into RAM

b. 384-kb Relocation

14753C–011

Figure 11. Shadow ROM/384-kb Relocation

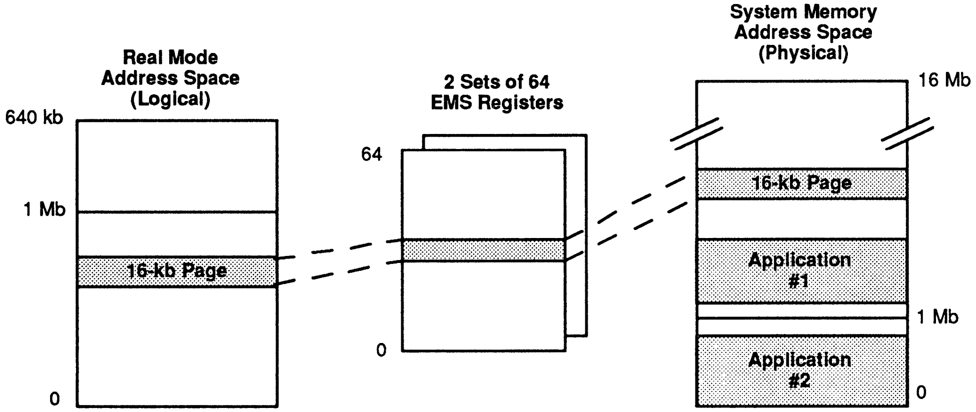
EMS 4.0 Memory Management

The principle behind EMS 4.0 is that any 16-kb page within a large protected memory area can be mapped to a location within the IBM-PC address space (0–1 Mb). This provides a method to implement extremely large data sets and multi-tasking within the memory limitations of MS-DOS. The Am286ZX/LX Address Translation Unit (ATU) provides all the basic hardware functionality to completely map the entire PC address space. The DRAM controller can be programmed to reserve up to 15.5 Mb of local memory for use with EMS. Using the ATU, any 16-kb page in the lower 1 Mb of memory can be mapped to any page in the reserved EMS address space.

The PC logical memory space of 1 Mb is considered as 64 pages, with each page 16 kb (64 x 16 kb = 1 Mb). The EMS memory mapping involves allocating one or more of these pages as a page frame. Accesses to this page frame are translated to addresses in the reserved memory area via a set of pointers in hardware. These pointers are stored in 2 sets of 64 Address Translation Registers (Main and Alternate sets), which are read and written by means of an address translation control register and four address translation register I/O address locations.

MS-DOS application software takes advantage of EMS 4.0 by writing code that conforms to the LIM EMS 4.0 specification.

The Am286ZX/LX integrated processor has two sets of 64 EMS registers. Each set of 64 registers forms a translation set which can map the entire 0–1 Mb address space. By having two sets, EMS software can perform high-speed switching between the two translation sets of EMS memory. When performing this type of multitasking, the addresses (pointers) of EMS memory for each task are loaded into both the primary and alternate EMS registers. This allows for a task switch to occur without reloading the pointers to each task. This greatly improves performance of task switching which is utilized by multitasking EMS software.



14753C–012

Figure 12. EMS Address Translation

Power Saving (Am286LX Processor Only)

Power saving modes are included in the Am286LX processor to allow the system BIOS to control power consumption. Power consumption is controlled by selectively turning off on-chip clocks. The Am286LX processor has two power saving modes—CPU stop clock and system standby. CPU stop clock mode shuts down the CPU clock until the next interrupt. By stopping CPU activity the system draws less current. In this way the system can stop the CPU when CPU interaction is not required. For example, in a keyboard polling loop, the system can stop the CPU between keyboard interrupts.

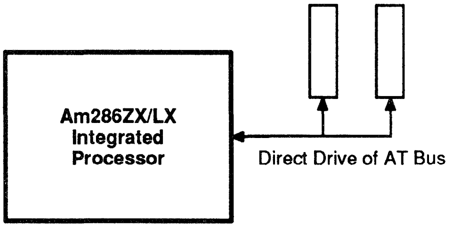
System standby mode goes even further by stopping all clocks that are not essential for DRAM refresh. This mode could be used in conjunction with slow-refresh DRAMs to basically turn the system off, except for maintaining data in the RAMs. The system can be switched into system standby mode during long periods of inactivity to substantially improve battery life. Normal implementation of the Am286LX processor's power saving modes requires no external circuitry. Instead, the modes are enabled by software.

The Am286LX processor also incorporates staggered DRAM refresh. By staggering the refresh of the DRAMs, instantaneous current demands are greatly reduced, providing a lower current surge on the system's power source. The Am286LX processor also has support for slow-refresh DRAMs. With slow-refresh memory, data does not have to be refreshed as often, which reduces power consumption.

External Interfaces

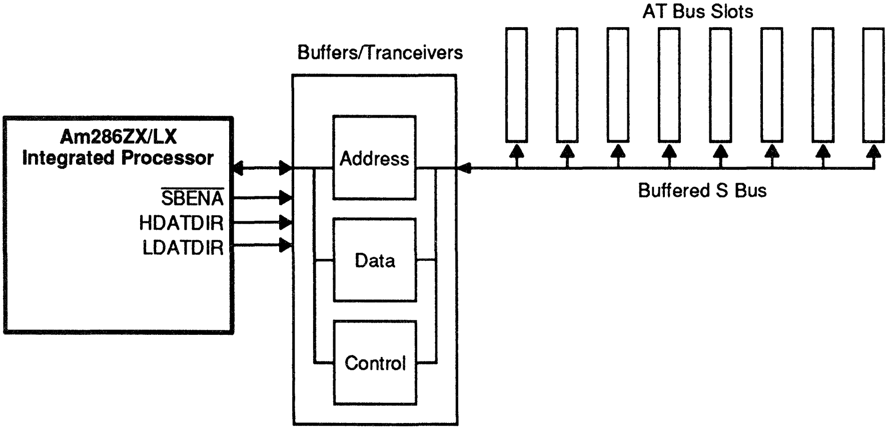
AT System Bus

The Am286ZX/LX integrated processor can directly drive the AT Bus within the limits of the DC current drive specification. For a fully buffered AT Bus, the Am286ZX/LX integrated processor provides three buffer control output pins. They are $\overline{\text{SBENA}}$, HDATDIR, and LDATDIR.



14753C–013

Figure 13. Direct S-Bus Connection



14753C–014

Figure 14. Buffered S-Bus Connection

Keyboard Controller

The Am286ZX/LX integrated processor provides a direct interface for an AT-type keyboard controller (8042), or an XT-type keyboard. The type of keyboard interface is selected with a configuration register. The KEYCLK output is driven by the clock generation logic, with four selectable clock inputs (see Enhanced Clock Generation). The keyboard controller is attached as an X-bus device.

AMD 80C287 Math Coprocessor

The Am286ZX/LX integrated processor provides a direct interface to the AMD 80C287 math coprocessor. The clock generation for the AMD 80C287 math coprocessor is driven by the clock generation logic, with two selectable clock inputs (see Enhanced Clock Generator). The AMD 80C287 math coprocessor is attached with the X-bus address lines and the M-bus data lines. The Am286ZX/LX integrated processor provides AT-standard I/O command decodes for "clear busy" (port 0F0h) and AMD 80C287 math coprocessor reset (port 0F1h). Standard coprocessor interconnect is shown in the following diagram.

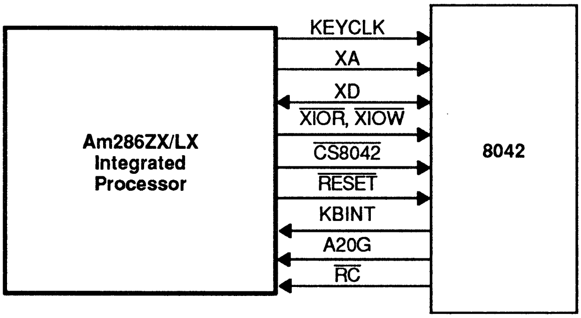
ROM/EPROM

The Am286ZX/LX integrated processor provides support for 8- or 16-bit EPROM configurations. The EPROM(s) are attached as an X-bus device, with high-byte data off the M Bus for 16-bit configurations.

Expansion Bus (X Bus)

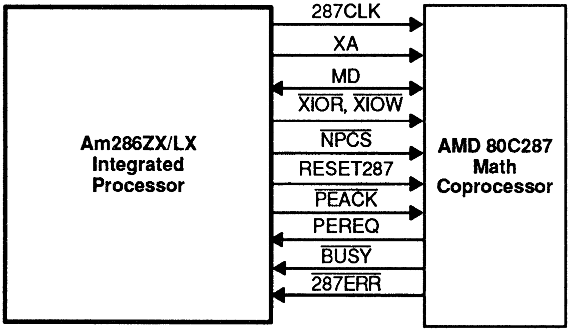
In addition to the EPROM, keyboard controller, and AMD 80C287 math coprocessor peripherals, the Am286ZX/LX integrated processor's X Bus can be used to attach other I/O devices. Eight configuration registers are available to map any 8-byte I/O address range onto the X Bus. 16-bit I/O is also supported.

Any I/O device such as serial ports, parallel ports, and floppy disk controllers may be attached to the Am286ZX/LX integrated processor's X Bus. Any of the four 8-byte DMA channels (0–3) can be mapped to the X Bus as well. For example, a floppy disk controller would use DMA Channel 2.



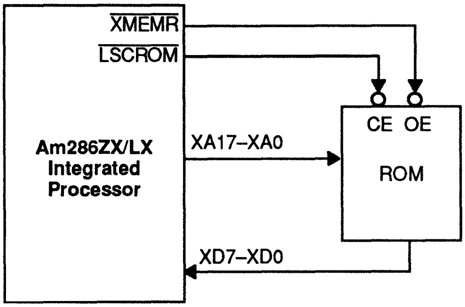
14753C–015

Figure 15. Keyboard Controller Interface



14753C–016

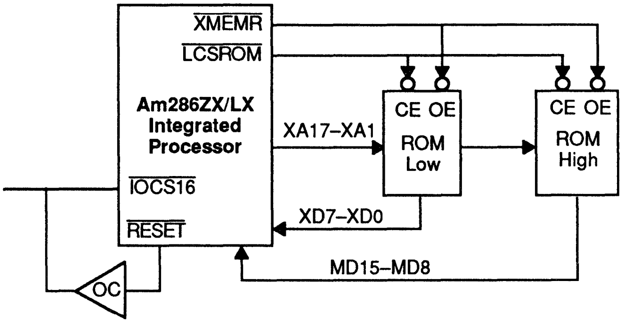
Figure 16. AMD 80C287 Math Coprocessor Interface



Note: This is the default configuration.

14753C–017

Figure 17. 8-Bit EPROM Configuration



Note: Open-collector buffer between RESET and IOCS16.

14753C–018

Figure 18. 16-Bit EPROM Configuration

Additional Features

Fast RESET and Fast GATEA20

The Am286ZX/LX integrated processor provides two features which enhance the system's performance—fast RESET and fast GATEA20. Normally, system software calls for a BIOS routine which issues two commands to switch the processor from protected mode to real mode. First, a RESET command is written to the 8042-keyboard controller; then the GATEA20 command is written to the 8042 controller. This imposes a performance limitation due to the slow 8042. The Am286ZX/LX integrated processor provides the fast RESET and fast GATEA20 features to bypass the keyboard controller and reduce the mode switch time. The Am286ZX/LX integrated processor conforms to the industry standard port 92h location to perform a reset to real mode.

These features greatly improve performance in operating environments like OS/2 and DOS extenders that frequently switch between real and protected mode during execution.

Bus Master Mode

The Am286ZX/LX integrated processor is normally the main CPU of an AT-architecture motherboard. However, the Am286ZX/LX integrated processor's Bus Master Mode allows it to act as a Bus Master peripheral connected to the expansion bus of a host system. This allows the Am286ZX/LX integrated processor to be the main CPU on a high-performance bus-master add-in card. The Am286ZX/LX integrated processor directly supports industry standard architecture (ISA) bus mastering and, with extended logic, MCA and EISA systems. Selecting bus master mode changes the operating mode of some of the S-bus pins to allow direct connection of the Am286ZX/LX integrated processor to the system expansion bus of another system.

NMI and Port B Logic

Industry standard non-maskable interrupt (NMI) logic is provided. This includes the AT-standard port B register at I/O address 061h, and the NMI enable register at I/O address 070h.

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature under Bias . . –55°C to +125°C
Storage Temperature . . . . –65°C to +150°C
Junction Temperature . . . . +175°C
Lead Temperature (10 seconds) . . . . +275°C
Supply Voltage to Ground Potential
Continuous . . . . –0.5 V to +7.0 V
DC Voltage Applied to Outputs . . . . –0.5 to Vcc Max
DC Input Voltage . . . . –0.5 to Vcc Max
DC Output Current . . . . ±30 mA
DC Input Current . . . . –10 mA to +10 mA

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Ambient Temperature (T_A) . . . . 0°C to +70°C
Supply Voltage (V_{CC}) . . . . +4.75 V to +5.25 V
Battery Voltage (V_{BATT}) . . . . –3.0 V to –4.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating ranges

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

Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
V_{IL}	Input Low Voltage		–0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.5	V
V_{ILC}	CLK Input Low Voltage (Note 11)		–0.5	0.8	V
V_{IHC}	CLK Input High Voltage (Note 11)		3.6	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	I_{OL} = See Note 10	–	0.4	V
V_{OH}	Output High Voltage	I_{OH} = See Note 10	3.0	–	V
I_I	Input Leakage Current	V_{IN} = GND or V_{CC}	–10	10	µA
I_{OZ}	Output Leakage Current	V_O = GND or V_{CC}	–10	10	µA
I_{CCSC}	Supply Current (Notes 1, 13)	16 MHz CPU 12.5 MHz CPU		30 28	mA
I_{CCSS}	Supply Current (Notes 2, 13)	16 MHz CPU 12.5 MHz CPU		24 22	mA
I_{CCFS}	Supply Current (Notes 3, 13)	16 MHz CPU 12.5 MHz CPU		5 5	mA
I_{CC}	Supply Current (Note 14)	16 MHz CPU 12.5 MHz CPU		240 190	mA
I_{CCBATT}	RTC Battery Backup Supply Current	V_{BATT} = –3.0 V (Note 12)		20	µA
IL_{LOAD1}	Pull Up Loading Current (Note 4)	V_{IN} = 0.4 V, V_{CC} = 5.25 V	–0.40	–0.10	mA
IL_{LOAD2}	Pull Up Loading Current (Note 5)	V_{IN} = 0.4 V, V_{CC} = 5.25 V	–4.50	-1.50	mA
IL_{LOAD3}	Pull Up Loading Current (Note 6)	V_{IN} = 0.4 V, V_{CC} = 5.25 V	-9.0	-3.0	mA
IL_{LOAD4}	Pull Up Loading Current (Note 7)	V_{IN} = 3.0 V, V_{CC} = 5.25 V	–0.30	–0.05	mA
IL_{LOAD5}	Pull Up Loading Current (Note 8)	V_{IN} = 0.4 V, V_{CC} = 5.25 V	–14.0	-5.0	mA
IH_{LOAD}	Pull Down Loading Current (Note 9)	V_{IN} = 3.0 V, V_{CC} = 5.25 V	0.05	0.40	µA

Notes:1. Stop Clock Mode (Am286LX processor only): CTSC bit set, input clocks running, CPU clock stopped, peripheral clocks and refresh running.
2. System Standby Mode (Am286LX processor only): SSBY bit set, input clocks running, CPU and Peripheral clocks stopped, refresh running.
3. Full Standby Mode: All input clocks stopped, including 32KHz and F14MHz. This is traditional CMOS standby current.
4. Pull up current for the SA, SD, IRQ, $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$, $\overline{SBHE}$, $\overline{BUSY}$, and $\overline{P287ERR}$ pins.
5. Pull up current for the $\overline{MASTER}$, IOCHRDY, $\overline{IOCHK}$ pins.
6. Pull up current for the $\overline{REFRESH}$, $\overline{IOCS16}$, $\overline{MEMCS16}$, and $\overline{POWS}$ pins.
7. Pull up current for the XA pin.
8. Pull up current for the KEYCLK and $\overline{CS8042}$ pins.
9. Pull down current for the DRQ and PEREQ pins.
10. The Am286ZX/LX integrated processors have several output buffer types, each with different I_{OL}/I_{OH} characteristics. They are characterized by the following table.
11. Includes pins PROCLK, IOCLK, KEYCLK (XT keyboard mode), $\overline{CS8042}$ (XT keyboard mode), and PWRGOOD, IOCHK.
12. See Battery Backup Test diagram.
13. V_{IN} = Vcc or GND, Vcc = 5.25 V, outputs loaded with 50 pF.
14. V_{IN} = 0 V or 4 V, Vcc = 5.25 V, outputs loaded with 50 pF.

Pin Group	I_{OL}, I_{OH}	C_L
SA, SD, $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$, RESETDRV	6 mA	100 pF
LA, $\overline{SMEMR}$, $\overline{SMEMW}$, SYSCLK, $\overline{MASTER}$, $\overline{REFRESH}$, $\overline{SBHE}$, DMATC, BALE, AEN, $\overline{SBENA}$, XD, XA, $\overline{XIOR}$, $\overline{XIOW}$, $\overline{RESET}$, $\overline{DACK}$	4 mA	100 pF
$\overline{XMEMR}$, $\overline{XMEMW}$, $\overline{LCSROM}$	4 mA	50 pF
SPKR	4 mA	150 pF
MA	2 mA	260 pF
MD, MDP	2 mA	100 pF
$\overline{MWENH}$, $\overline{MWENL}$	2 mA	260 pF
$\overline{RAS}$, $\overline{CAS}$	2 mA	200 pF
DLYOUT, P287CLK, $\overline{NPCS}$, RESET287, $\overline{PEACK}$, HDATDIR, LDATDIR	2 mA	50 pF

CAPACITANCE

TA = +25°C; all measurements referenced to device GND.

Parameter Symbol	Parameter Descriptions	Typ	Unit
C_{IN}	Input Pin Capacitance	10	pf
C_{IO}	Bi-directional Pin Capacitance	10	pf
C_{OUT}	Output Pin Capacitance	10	pf

Note: Pin capacitance is characterized at a frequency of 1 MHz, and is tested on a sample basis only.

SWITCHING CHARACTERISTICS over operating range

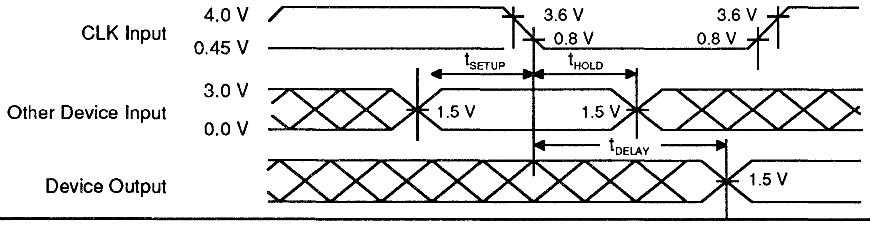
Method of Specification

The Am286ZX/LX integrated processor has several modes of operation. Given the number of options available and the internal generation of clocks, some explanation of the specification method is in order. The following points must be noted:

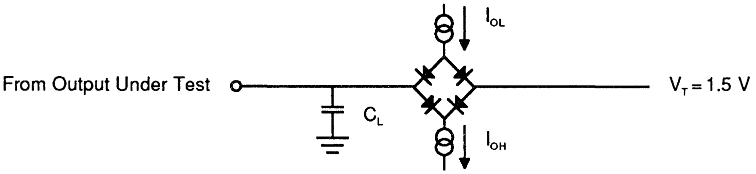
1. In order to test and verify the timing parameters, the specifications are from external input to output definitions.
2. The designator "srcCLK" refers to the input clocks PROCLK or IOCLK. Internal and external clocks are generated from these input clock signals. KEYCLK may use the 14.318 MHz input clock as its source. It is assumed for all diagrams that all PROCLK and IOCLK related dividers are set for divide by 1.
3. Output invalid delay times (Min and Max) not specified track the corresponding valid delay time (Min and Max) within ±5 ns. There are two exception on the M bus for non-paged mode accesses. The $\overline{RAS}$ invalid delay timing tracks the valid delay timing by +10 ns. The MA10–MA0 invalid timing tracks the valid delay timing by ±8 ns.
4. The AT-bus timing diagrams assume a programmed configuration of no command delays for 16-bit memory accesses, one command delay for 8-bit memory accesses, and one command delay for I/O commands. The timing parameters are referenced to clocks and are independent of command delays.

Note: Switching characteristics are targeted numbers and are subject to change without notice.

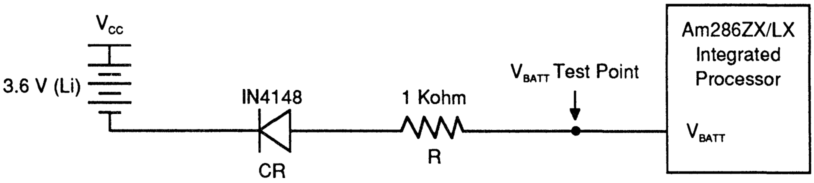
SWITCHING TEST WAVEFORM



SWITCHING TEST CIRCUIT



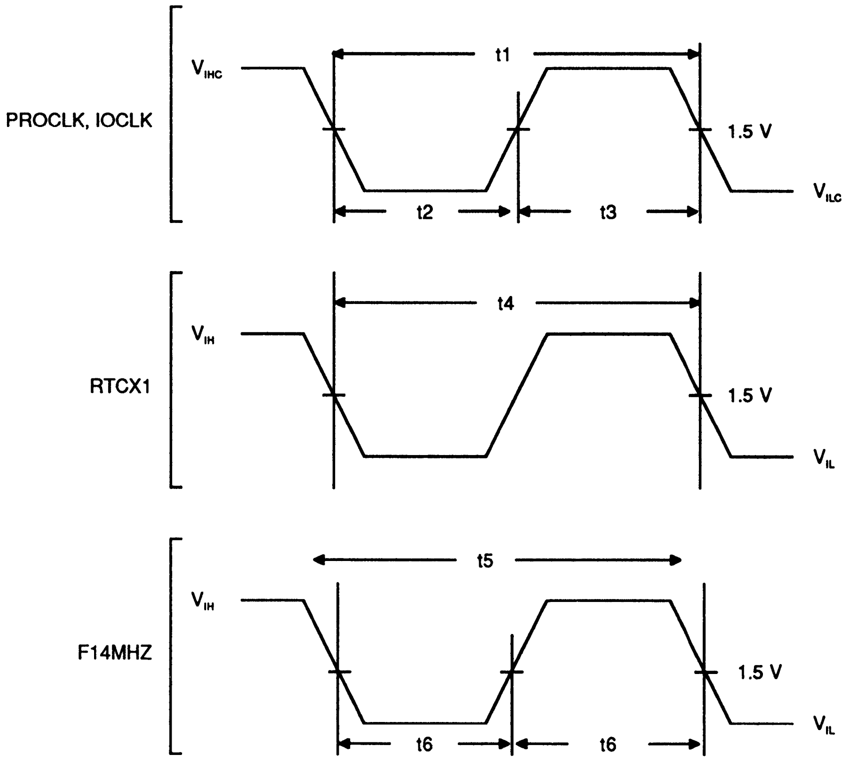
BATTERY BACKUP TEST DIAGRAM



Input Clocks

Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t1	PROCLK, IOCLK Period	16 MHz CPU 12.5 MHz CPU	31 40		ns
t2	PROCLK, IOCLK Low Time	16 MHz CPU 12.5 MHz CPU	14 16		ns
t3	PROCLK, IOCLK High Time	16 MHz CPU 12.5 MHz CPU	13 16		ns
t4	RTCCLK Period (Note 1)		30.518		µs
t5	F14 MHz Period		69.8		ns
t6	F14 MHz High, Low Time		25		ns

Note 1. 32.768 kHz ±100 ppm crystal input across RTCLKX1 and RTCLKX2 for AT compatibility.



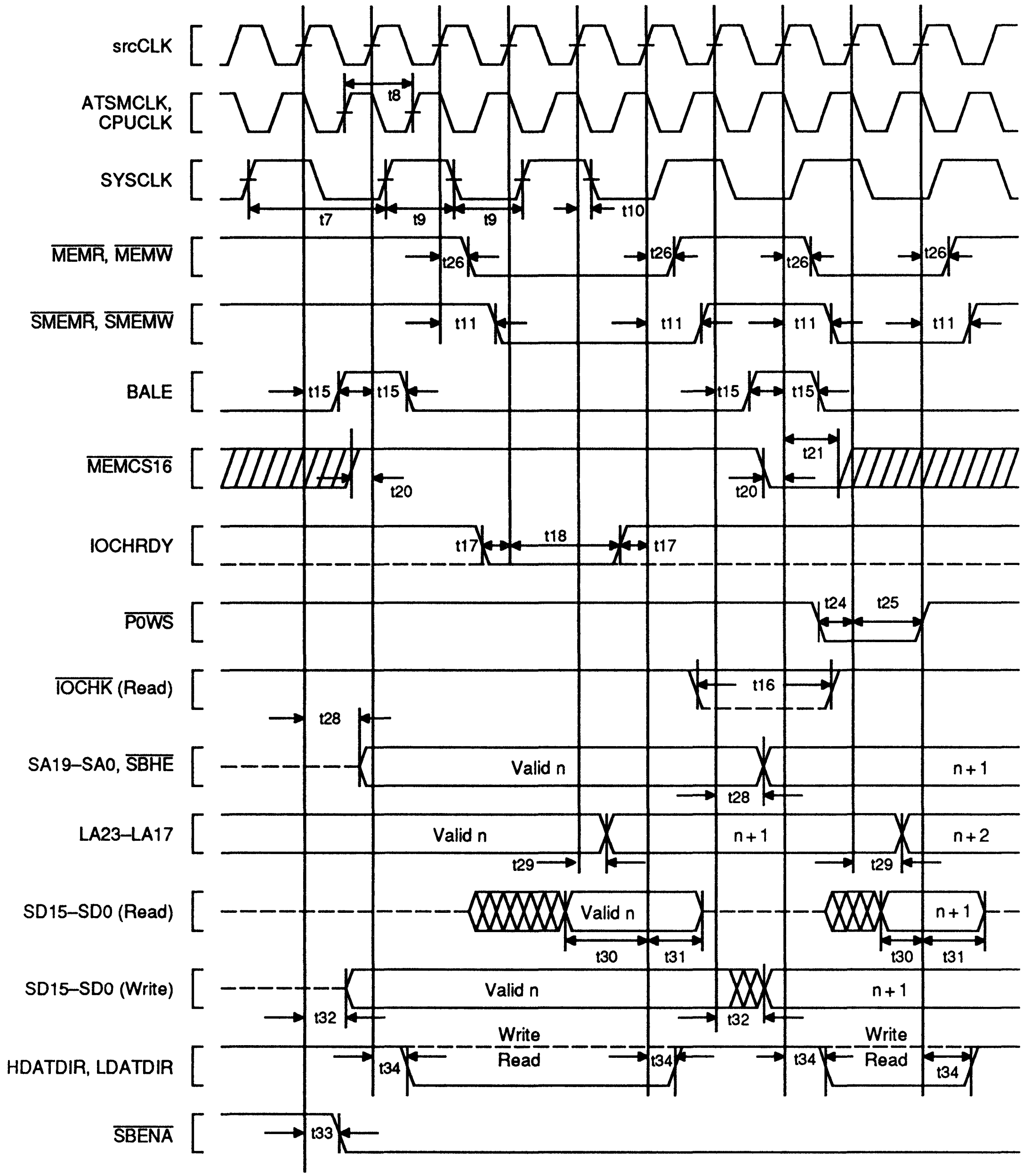
14753C–019

Figure 19. Input Clocks

AT System Bus Interface: Referenced to srcCLK

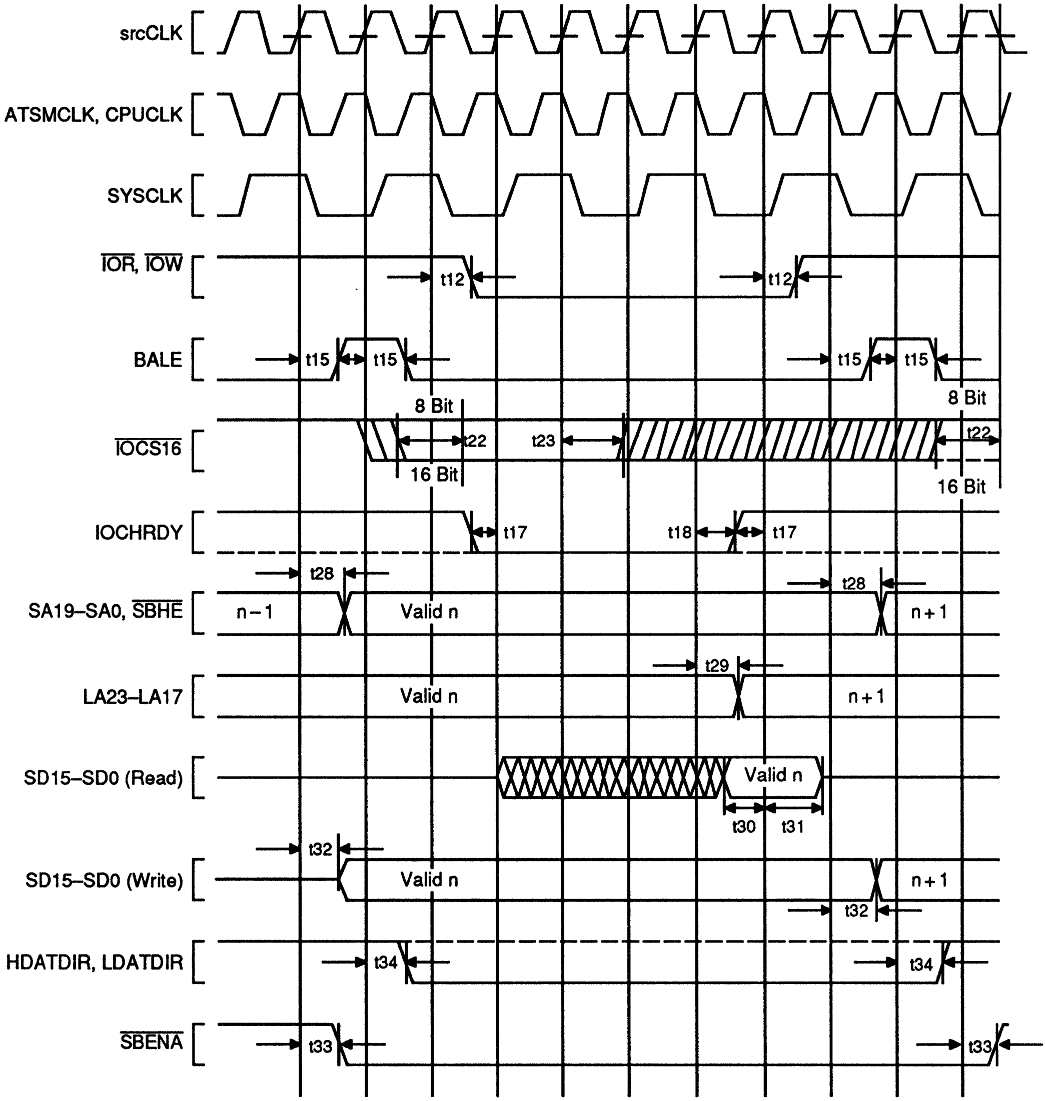
Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t7	SYSCLK Period	Note 2	125		ns
t8	ATSMCLK Period		62.5		ns
t9	SYSCLK High/Low Time		50		ns
t10	SYSCLK Delay from srcCLK	Note 1	9	31	ns
t11	$\overline{\text{SMEMR}}$/$\overline{\text{SMEMW}}$ from srcCLK	Note 1		70	ns
t12	$\overline{\text{IOR}}$/$\overline{\text{IOW}}$ from srcCLK	Note 1		50	ns
t13	$\overline{\text{REFRESH}}$ Active from srcCLK	Note 1		90	ns
t14	$\overline{\text{REFRESH}}$ Inactive from srcCLK	Note 1		90	ns
t15	BALE from srcCLK	Note 1		45	ns
t16	$\overline{\text{IOCHK}}$ Active Pulse Width	Note 3	10		ns
t17	IOCHRDY Setup to srcCLK	Notes 1, 3	5		ns
t18	IOCHRDY Hold from srcCLK	Notes 1, 3	17		ns
t19	RESETDRV from srcCLK	Note 4		65	ns
t20	$\overline{\text{MEMCS16}}$ Setup to srcCLK	Note 1	5		ns
t21	$\overline{\text{MEMCS16}}$ Hold to srcCLK	Note 1	30		ns
t22	$\overline{\text{IOCS16}}$ Setup to srcCLK	Note 1	10		ns
t23	$\overline{\text{IOCS16}}$ Hold to srcCLK	Note 1	25		ns
t24	$\overline{\text{P0WS}}$ Setup to srcCLK	Note 1	11		ns
t25	$\overline{\text{P0WS}}$ Hold to srcCLK	Note 1	15		ns
t26	$\overline{\text{MEMR}}$/$\overline{\text{MEMW}}$ from srcCLK	Note 1		47	ns
t27	IRQ Inactive Pulse Width	Notes 3, 5, 6	100		ns
t28	SA, $\overline{\text{SBHE}}$ Valid from srcCLK	Note 1		60	ns
t29	LA Valid from srcCLK	Note 1		52	ns
t30	SD Read Setup to srcCLK	Note 1	13		ns
t31	SD Read Hold to srcCLK	Note 1	35		ns
t32	SD Write Valid from srcCLK	Note 1		60	ns
t33	$\overline{\text{SBENA}}$ from srcCLK	Note 1		52	ns
t34	(H,L)DATDIR from srcCLK	Note 1		54	ns

Notes: 1. ATSMCLK reference.
2. The maximum frequency for SYSCLK is 8 MHz.
3. Asynchronous inputs not requiring setup or hold times. This specification given for testing purposes.
4. Duration of RESETDRV active is dependent on PWRGOOD and can be software controlled.
5. Low time required to clear the input latch of the interrupt controller.
6. IRQs must remain High until after the first interrupt acknowledge cycle to prevent spurious interrupts.



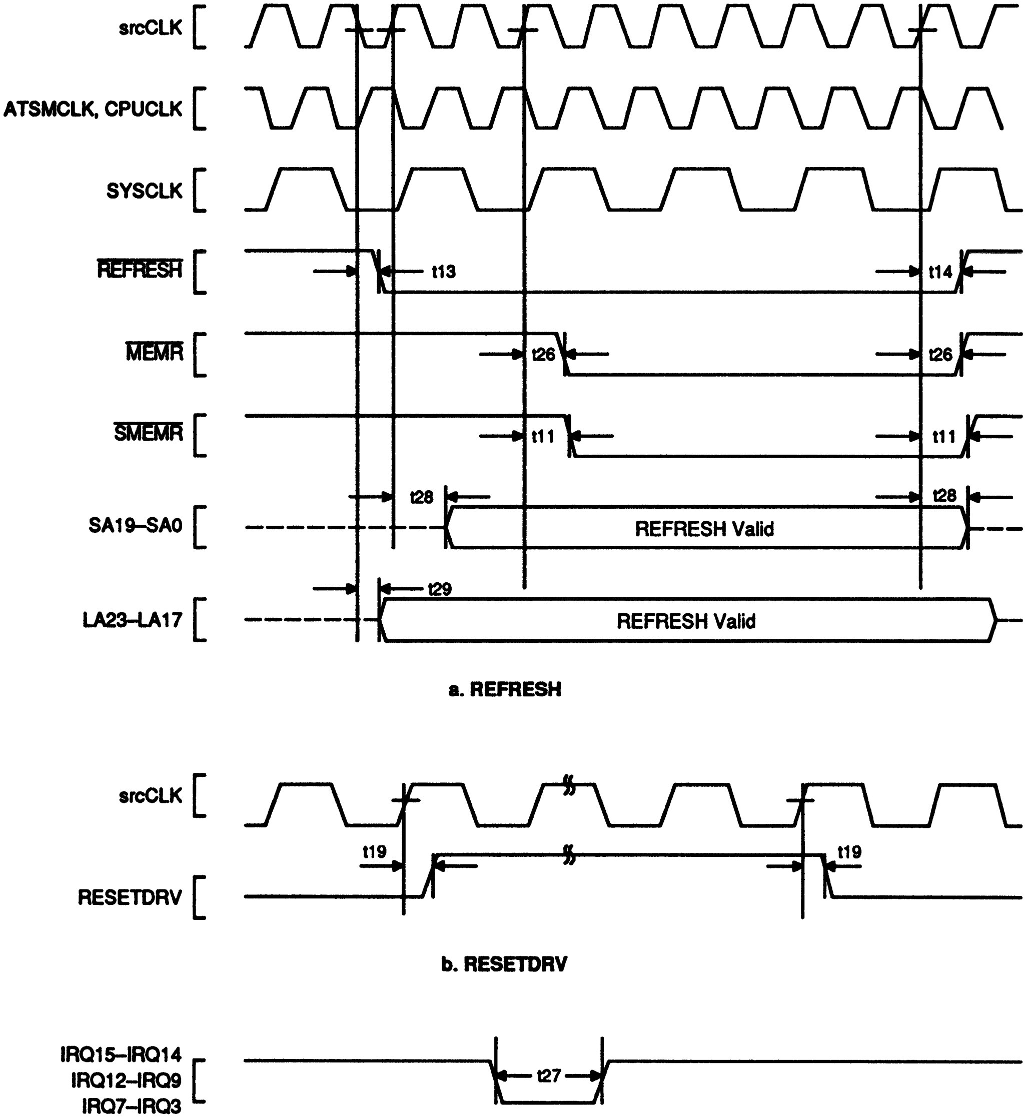
14753C–020

Figure 20. AT System Bus Interface (Processor) Memory 8-Bit-1WS, 16-Bit-0WS



14753C–021

Figure 21. AT System Bus Interface (Processor) I/O 16- and 8-Bit-2WS



a. REFRESH

b. RESETDRV

c. IRQ

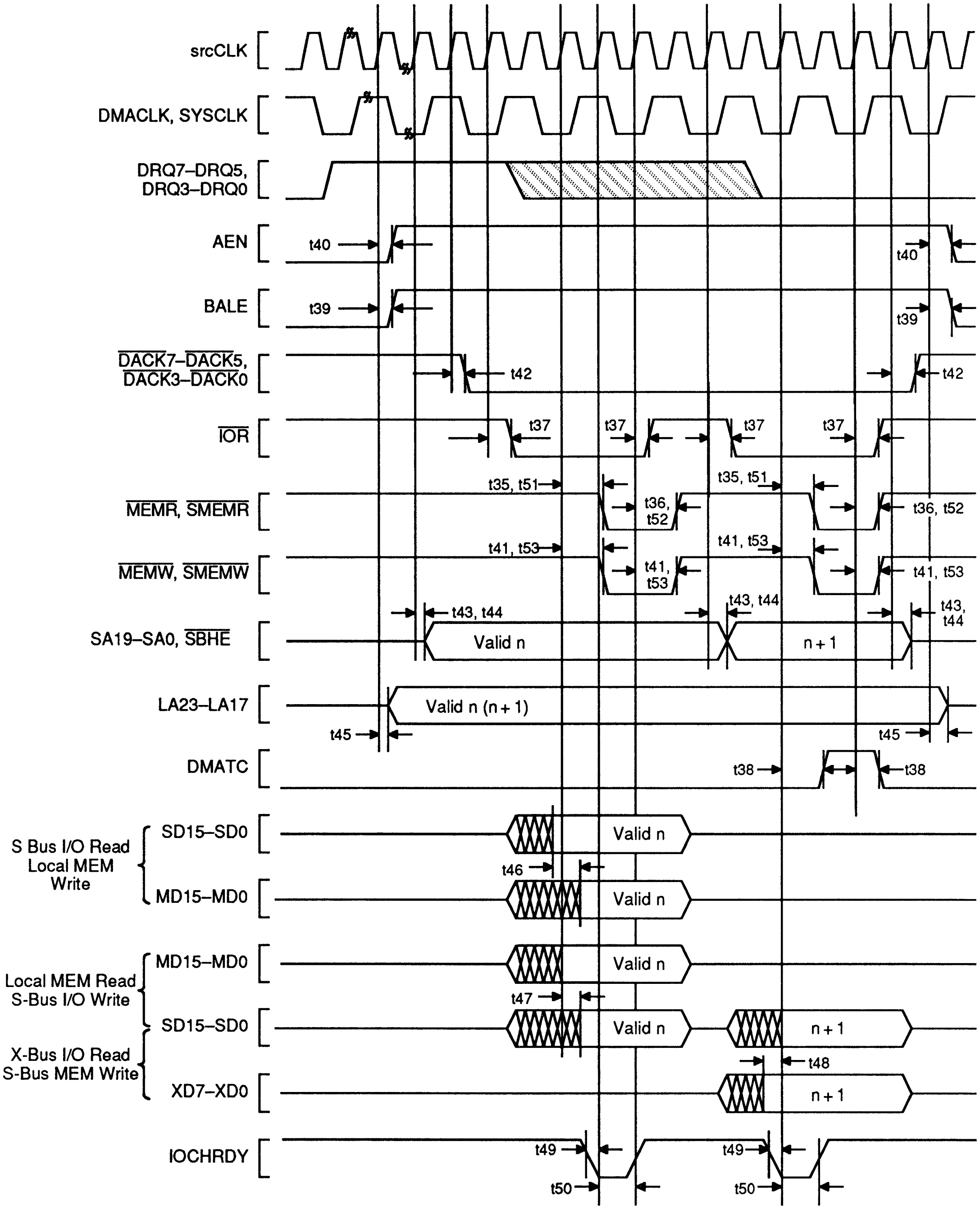
14753C–022

Figure 22. AT System Bus Interface (Processor)

AT System Bus Interface (DMA): Referenced to srcCLK

Parameter Symbol	Parameter Descriptions	Notes and Conditons	Min	Max	Unit
t35	$\overline{\text{MEMR}}$ Active from srcCLK	Note 2		50	ns
t36	$\overline{\text{MEMR}}$ Inactive from srcCLK	Note 2		75	ns
t37	$\overline{\text{IOR}}$ from srcCLK	Note 2		60	ns
t38	DMATC from srcCLK	Note 2		75	ns
t39	BALE from srcCLK	Note 1		55	ns
t40	AEN from srcCLK	Note 1		60	ns
t41	$\overline{\text{MEMW}}$, $\overline{\text{IOW}}$ from srcCLK	Note 2		60	ns
t42	$\overline{\text{DACK}}$ from srcCLK	Note 2		65	ns
t43	SA Valid from srcCLK	Note 2		120	ns
t44	$\overline{\text{SBHE}}$ from srcCLK	Note 2		80	ns
t45	LA Valid from srcCLK	Note 2		105	ns
t46	MD Valid from SD Valid ($\overline{\text{IOR}}$)			25	ns
t47	SD Valid from MD Valid ($\overline{\text{MEMR}}$)			27	ns
t48	SD Valid from XD Valid ($\overline{\text{IOR}}$)			27	ns
t49	IOCHRDY Setup to srcCLK	Notes 2, 3	5		
t50	IOCHRDY Hold from srcCLK	Notes 2, 3	25		
t51	$\overline{\text{SMEMR}}$ Active from srcCLK	Note 2		50	
t52	$\overline{\text{SMEMR}}$ Inactive from srcCLK	Note 2		80	
t53	$\overline{\text{SMEMW}}$ from srcCLK	Note 2		65	

Notes: 1. CPUCLK reference.
2. DMACLK reference.
3. Asynchronous inputs not requiring setup or hold times. This specification given for testing purposes.



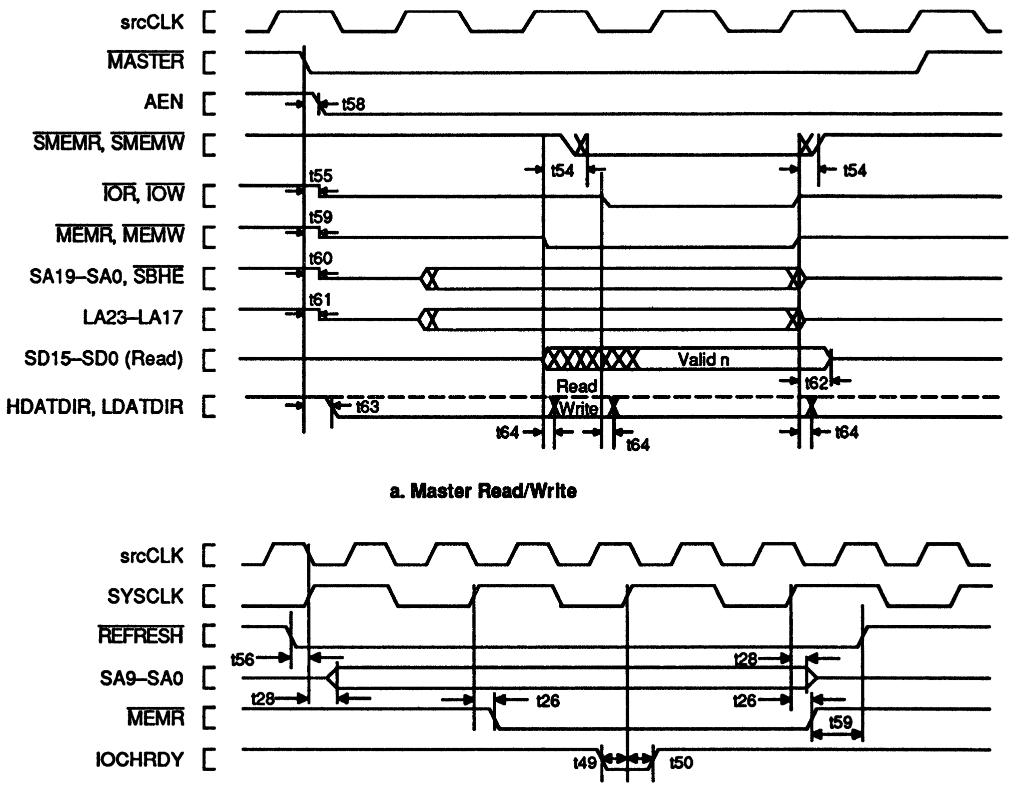
14753C–023

Figure 23. AT System Bus Interface (DMA) Local I/O Write/Memory Read, X Bus I/O Read/Memory Write

AT System Bus Interface (External Master): Referenced to srcCLK

Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t54	SMEMR/SMEMW from MEMR/MEMW			50	ns
t55	IOR/IOW Float from MASTER Active	Note 3		40	ns
t56	REFRESH Setup to srcCLK	Note 1, 2	5		ns
t57	REFRESH Hold from srcCLK	Note 1, 2	20		ns
t58	AEN from MASTER			40	ns
t59	MEMR/MEMW Float from MASTER Active	Note 3		40	ns
t60	SA, SBHE Float from MASTER Active	Note 3		55	ns
t61	LA Float from MASTER Active	Note 3		30	ns
t62	SD Hold from MEMR		12		ns
t63	(H,L)DATDIR from MASTER Active			44	ns
t64	(H,L)DATDIR from IOR/MEMR Active			44	ns
t65	SA Setup to Command		40		ns
t66	SA Hold from Command		10		ns

Notes: 1. ATSMCLK reference.
2. Asynchronous inputs not requiring setup or hold times. This specification given for testing purposes.
3. Not 100% tested.



a. Master Read/Write

b. REFRESH

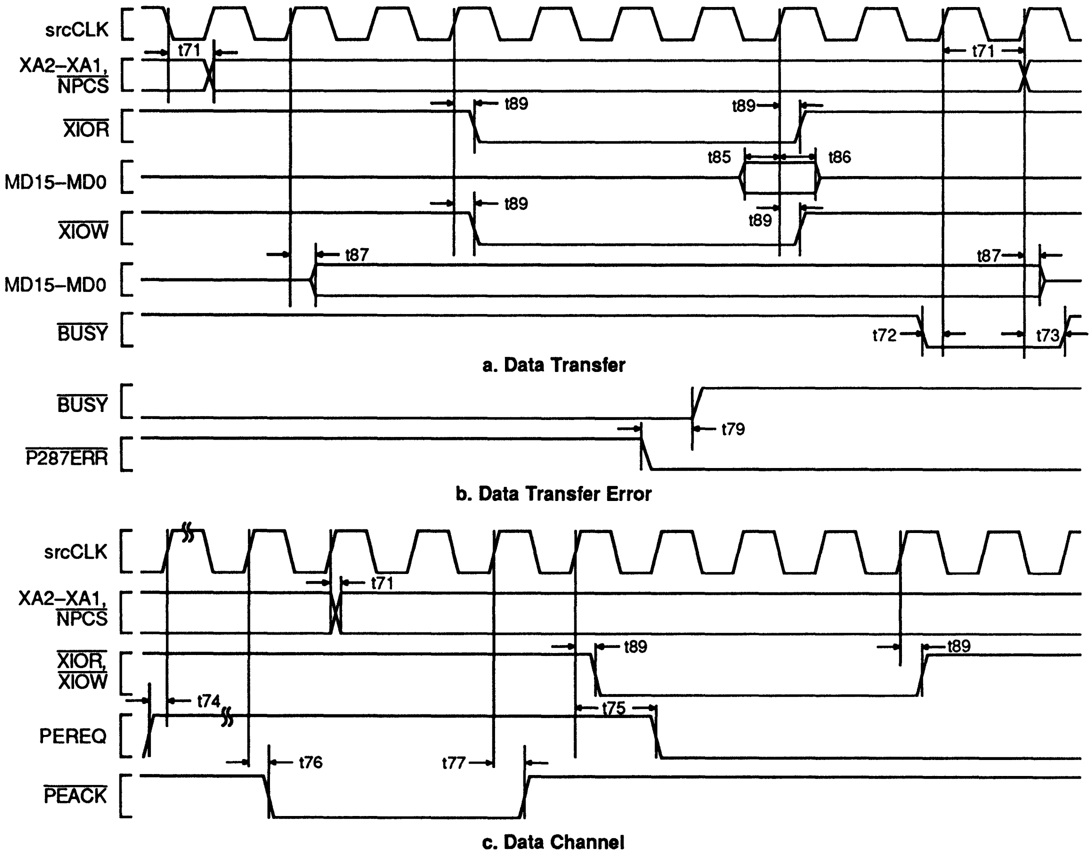
14753C–024

Figure 24. AT System Bus Interface (External Bus Master)

Coprocessor Interface

Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t68	P287CLK Period	Note 2	30		ns
t69	P287CLK Low Time		12		ns
t70	P287CLK Delay from srcCLK			20	ns
t71	$\overline{\text{NPCS}}$ from srcCLK	Notes 1, 3		47	ns
t72	$\overline{\text{BUSY}}$ Setup to srcCLK	Notes 1, 3	5		ns
t73	$\overline{\text{BUSY}}$ Hold from srcCLK	Notes 1, 3	5		ns
t74	PEREQ Setup to srcCLK	Notes 1, 3	5		ns
t75	PEREQ Hold from srcCLK	Notes 1, 3	20		ns
t76	$\overline{\text{PEACK}}$ Active from srcCLK	Note 1		35	ns
t77	$\overline{\text{PEACK}}$ Inactive from srcCLK	Note 1		247	ns
t78	RESET287 from srcCLK	Note 1		40	ns
t79	$\overline{\text{BUSY}}$ to $\overline{\text{P287ERR}}$ Active Overlap		5		ns

Notes: 1. CPUCLK reference.
2. This 50% duty cycle is internally divided by 3 within the AMD 80C287 math coprocessor.
3. Asynchronous inputs not requiring setup or hold times. This specification given for testing purposes.



a. Data Transfer

b. Data Transfer Error

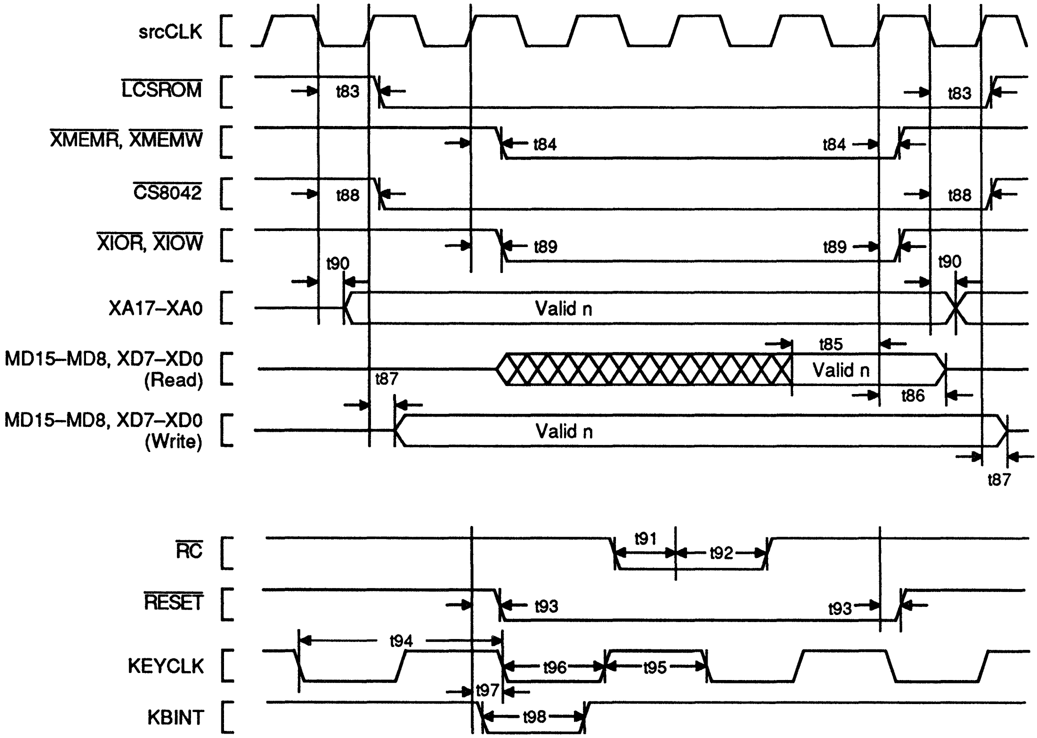
c. Data Channel

Figure 25. Coprocessor Interface

XBUS—ROM, RAM, KBD, I/O, MEM

Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t83	LCSROM from srcCLK	Note 1		60	ns
t84	XMEMR/XMEMW from srcCLK Non-DMA cycles	Note 1		45	ns
t85	MD, XD Read Setup to srcCLK	Note 1	15		ns
t86	MD, XD Read Hold from srcCLK	Note 1	30		ns
t87	MD, XD Write from srcCLK	Note 1		60	ns
t88	CS8042 from srcCLK	Note 1		47	ns
t89	XIOR/XIOW from srcCLK	Note 1		60	ns
t90	XA from srcCLK	Note 1		40	ns
t91	RC Setup to srcCLK	Notes 1, 2	5		ns
t92	RC Hold to srcCLK	Notes 1, 2	5		ns
t93	RESET from srcCLK	Note 1		25	ns
t94	KEYCLK Period		32		ns
t95	KEYCLK High Time		12		ns
t96	KEYCLK Low Time		8		ns
t97	KEYCLK Delay from srcCLK	Note 1		31	ns
t98	KBINT Inactive Pulse Width	Notes 2, 3, 4		100	ns

Notes: 1. CPUCLK reference.
2. Asynchronous inputs not requiring setup or hold times. This specification given for testing purposes.
3. This Low time is required to clear the input latch in edge triggered mode.
4. In all modes KBINT must remain High until the first Interrupt Acknowledge.



14753C–026

Figure 26. X Bus—ROM, RAM, Keyboard, I/O Device (Processor) Read, Write, 1WS

XBUS—I/O (DMA/External Master)

Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t99	XD Valid from SD Valid			30	ns
t100	XD Valid from MD Valid			30	ns
t101	MD Valid from XD Valid			30	ns
t102	$\overline{XIOR}/\overline{XIOW}$ Valid from srcCLK	Note 1		60	ns
t103	$\overline{XMEMR}/\overline{XMEMW}$ Valid from $\overline{MEMR}/\overline{MEMW}$ Valid			40	ns
t104	$\overline{XIOR}/\overline{XIOW}$ Valid from $\overline{IOR}/\overline{IOW}$ Valid			30	ns
t105	XA Valid from SA Valid			20	ns

Notes: 1. DMACLK reference.

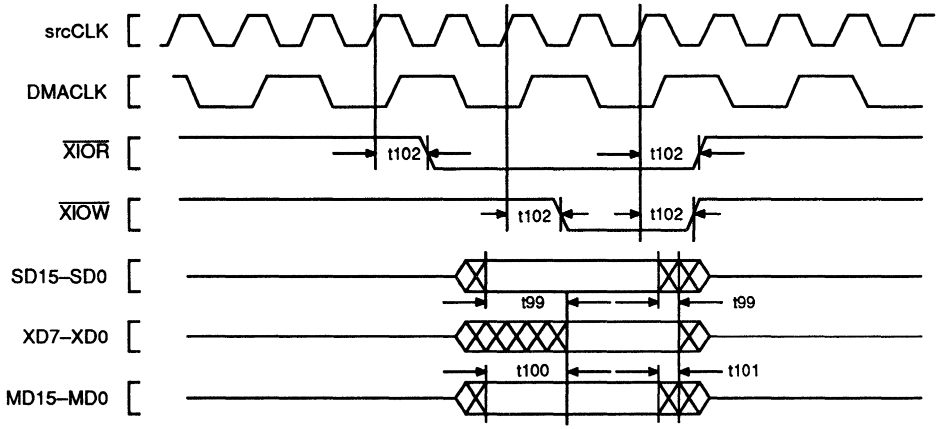


Figure 27. X Bus—I/O Device (DMA)

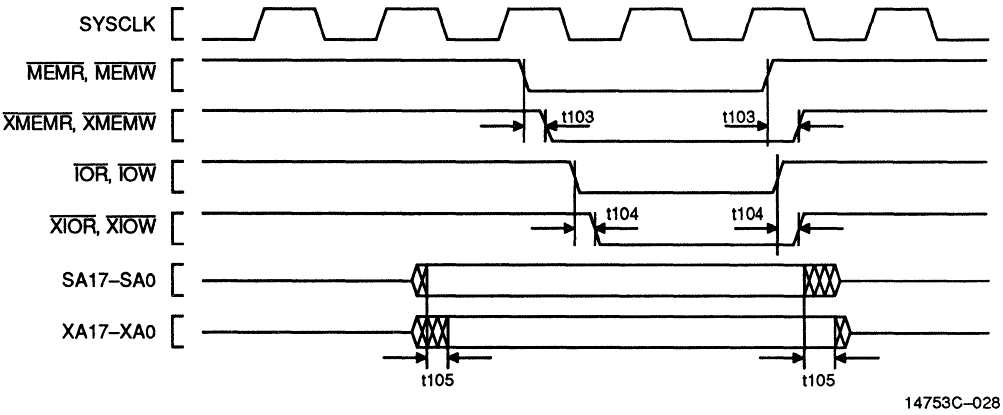

SYSCLK
$\overline{MEMR}$, $\overline{MEMW}$
$\overline{XMEMR}$, $\overline{XMEMW}$
$\overline{IOR}$, $\overline{IOW}$
$\overline{XIOR}$, $\overline{XIOW}$
SA17–SA0
XA17–XA0
t103
t103
t104
t104
t105
t105
14753C–028

Figure 28. X Bus—RAM, Keyboard, I/O Device (External Bus Master)

Miscellaneous Signals

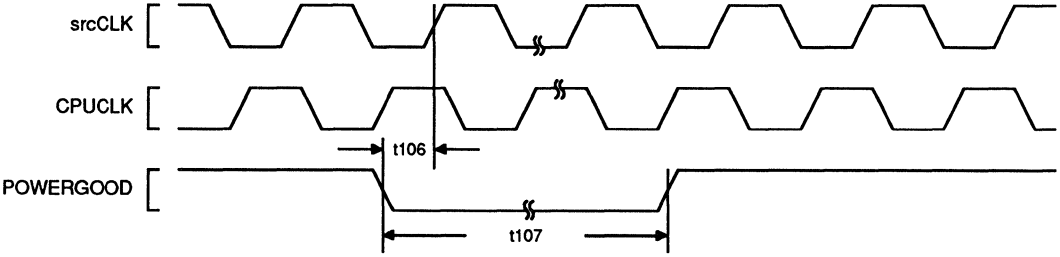
Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t106	PWRGOOD Setup to srcCLK	Notes 1, 2	15		ns
t107	PWRGOOD Pulse Width	Notes 1, 2, 3	65 · TPCLK		ns

Notes: 1. CPUCLK reference.
2. Asynchronous inputs not requiring setup or hold times. This specification given for testing purposes.
3. TPCLK is the period of PROCLK.

Am286ZX/LX Integrated Processor as a Bus Master

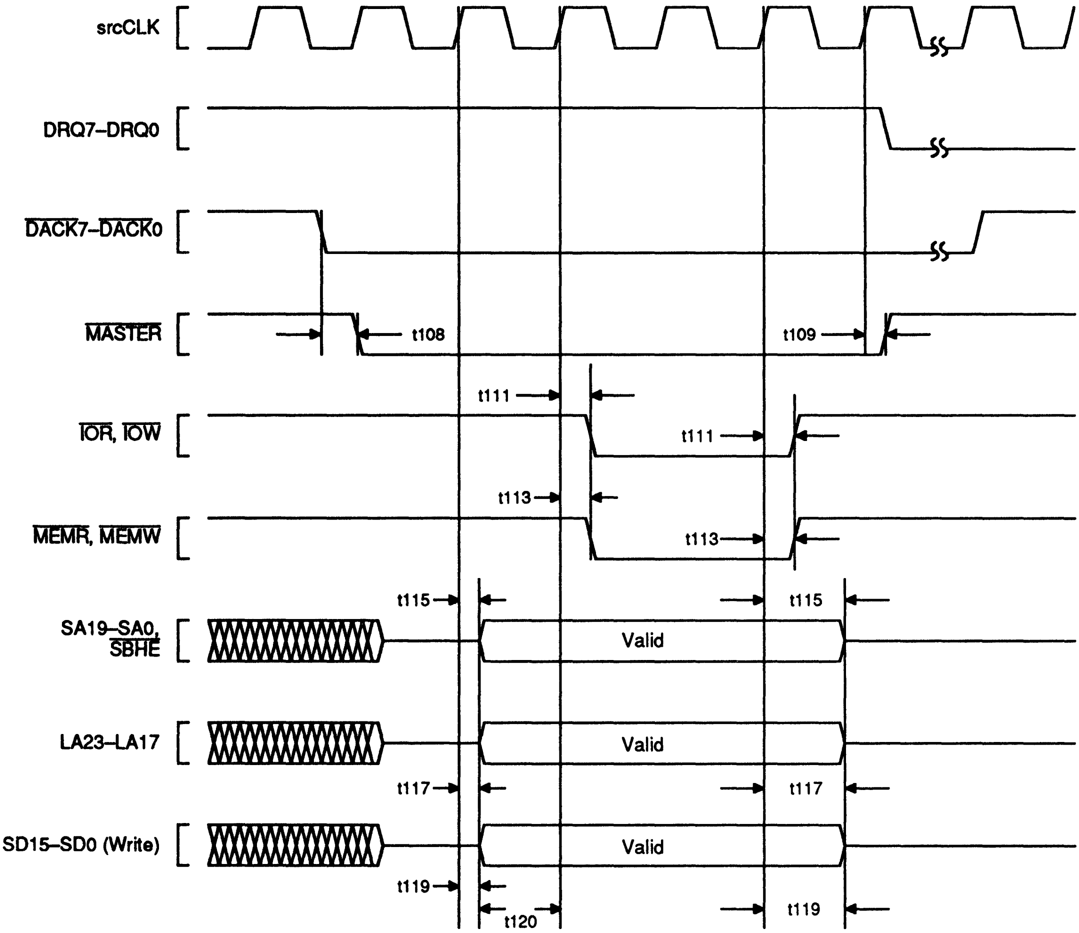
New Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t108	$\overline{\text{MASTER}}$ Active from $\overline{\text{DACK}}$ Active			25	ns
t109	$\overline{\text{MASTER}}$ Inactive from srcCLK			40	ns
t111	$\overline{\text{IOR}}$/$\overline{\text{IOW}}$ from srcCLK	Note 1		50	ns
t113	$\overline{\text{MEMR}}$/$\overline{\text{MEMW}}$ from srcCLK	Note 1		50	ns
t115	SA, $\overline{\text{SBHE}}$ from srcCLK	Note 1		60	ns
t117	LA Valid from srcCLK	Note 1		45	ns
t119	SD Write from srcCLK	Note 1		63	ns
t120	SD Write Setup to $\overline{\text{MEMW}}$ Active		12		ns

Notes: 1. ATSMCLK reference.



14753C–029

Figure 29. Miscellaneous Signals



14753C–030

Figure 30. Am286ZX/LX Integrated Processor as a Bus Master

Memory Bus

Parameter Symbol	Parameter Descriptions	Notes and Conditions	Min	Max	Unit
t121	srcCLK to RAS Active			36	ns
t122	srcCLK to RAS Inactive			43	ns
t123	srcCLK to CAS Active			35	ns
t124	srcCLK to CAS Inactive			35	ns
t125	srcCLK to MA Valid			50	ns
t126	srcCLK to MA Invalid		9		ns
t127	MWENL, MWENH Active Delay			40	ns
t128	MWENL, MWENH Inactive Delay			40	ns
t129	srcCLK to MD Valid			47	ns
t130	srcCLK to MD FLOAT	Note 1		40	ns
t131	MD Read Setup to srcCLK		6		ns
t132	MD Hold from srcCLK		20		ns
t133	srcCLK to MDP Valid			47	ns
t134	MDP Setup to srcCLK		6		ns
t135	MDP Hold from srcCLK		20		ns
t136	srcCLK to MDP FLOAT	Note 1		40	ns
t137	SA to MA Valid Delay			73	ns
t139	SBUS cmd to DLYOUT Delay		5	35	ns
t140	DL0 to MA Valid Delay			25	ns
t141	SBUS cmd to RAS Delay			50	ns
t142	SBUS cmd to MWENL, MWENH Delay			49	ns
t143	DL1 to CAS Delay		4	22	ns

Notes: 1. Not 100% tested.

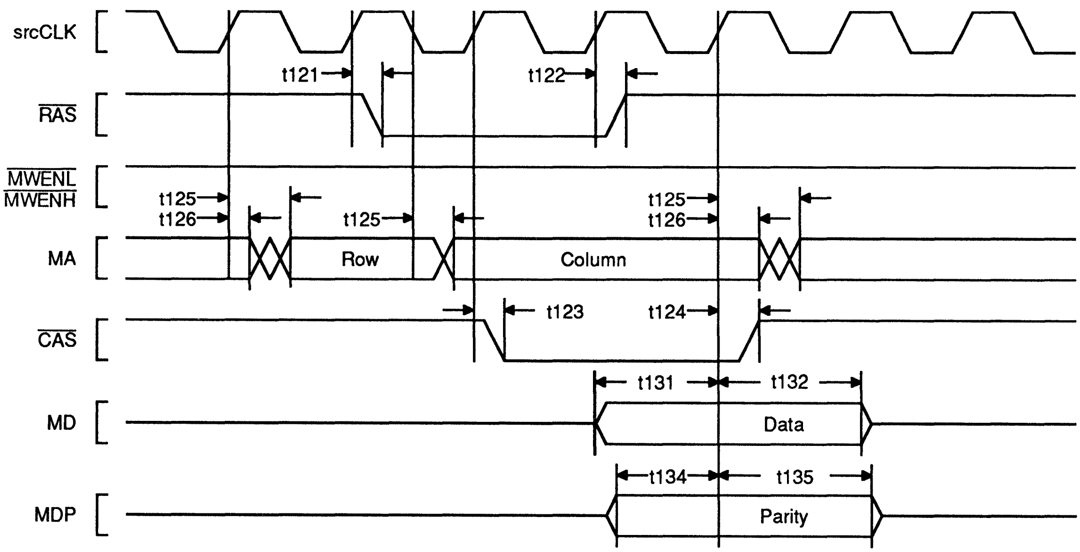


Figure 31. M Bus: Non-Paged Mode—Read/0WS

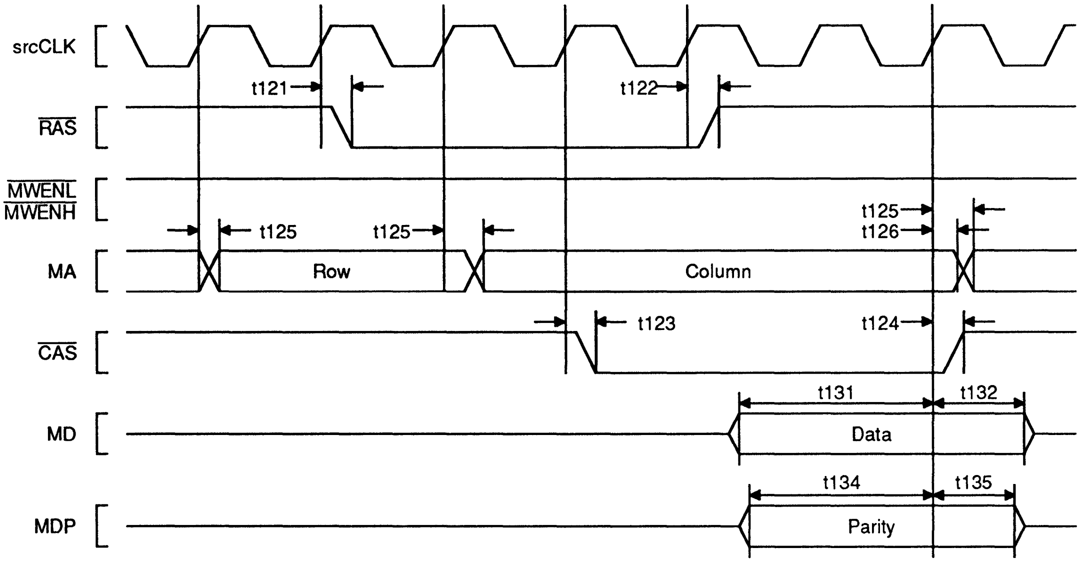


Figure 32. M Bus: Non-Paged Mode—Read/1WS

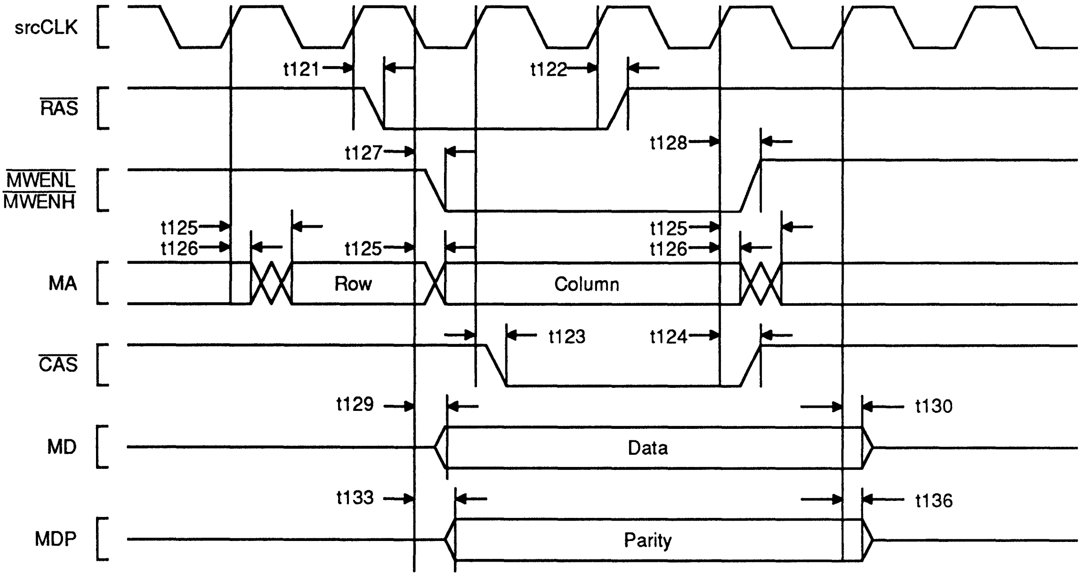


Figure 33. M Bus: Non-Paged Mode—Write/0WS

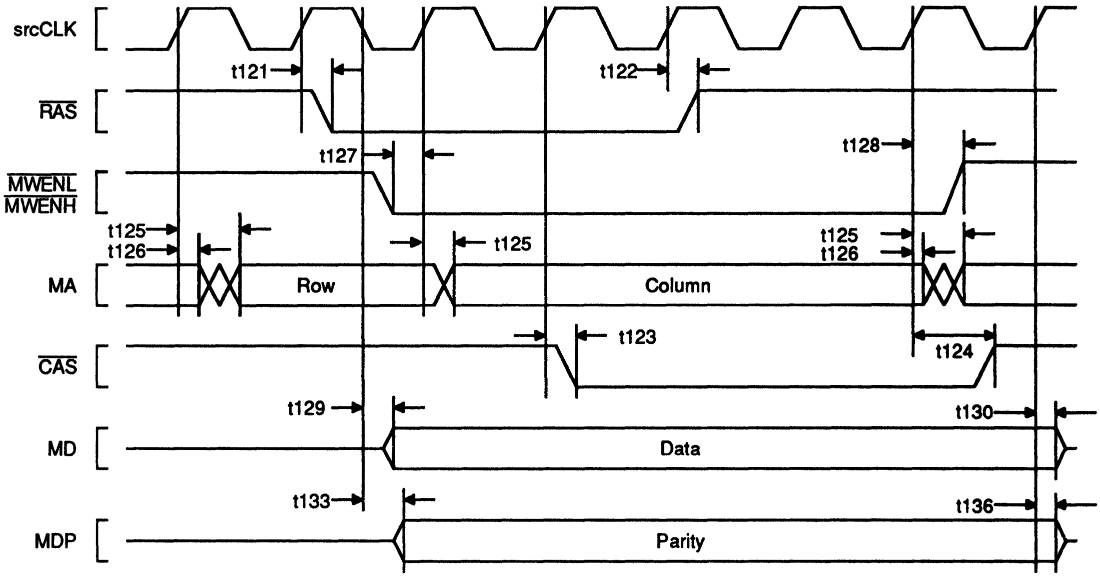
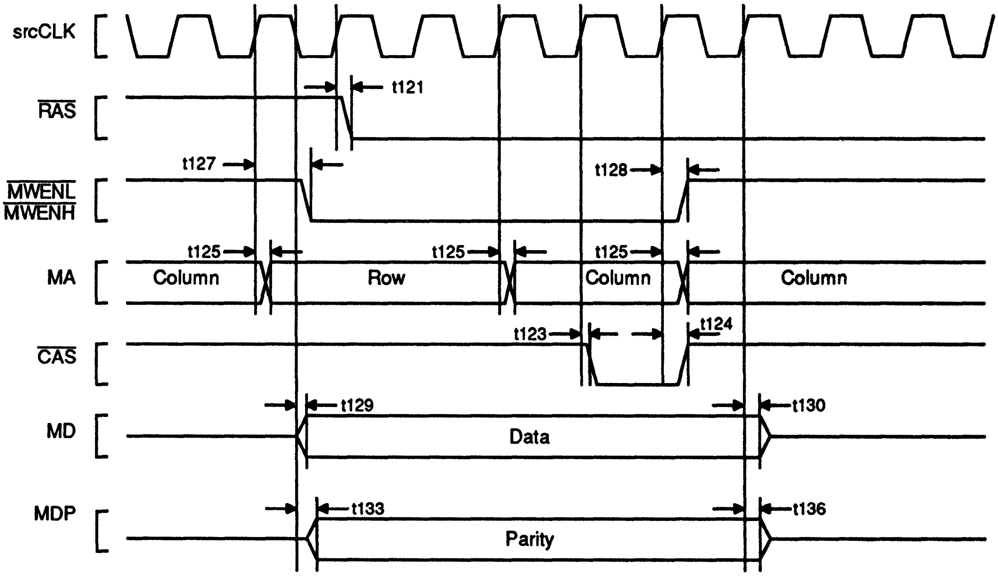


Figure 34. M Bus: Non-Paged Mode—Write/1WS



14753C–031

Figure 35. M Bus: Page Mode, Page—Hit $\overline{\text{RAS}}$ Inactive/Write/0WS

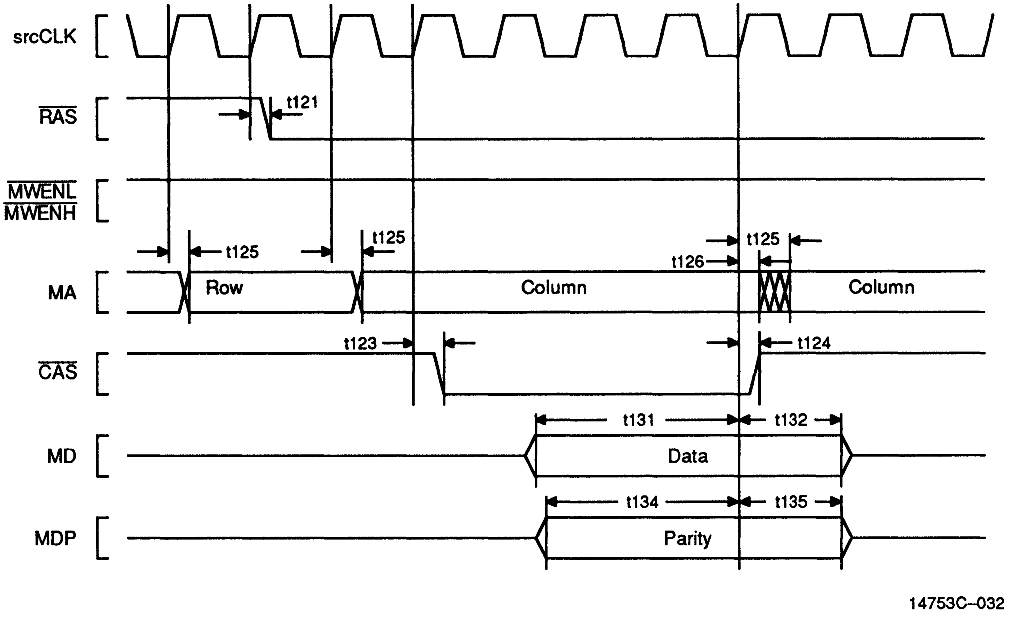


Figure 36. M Bus: Page Mode, Page—Hit RAS Inactive/Read/0WS

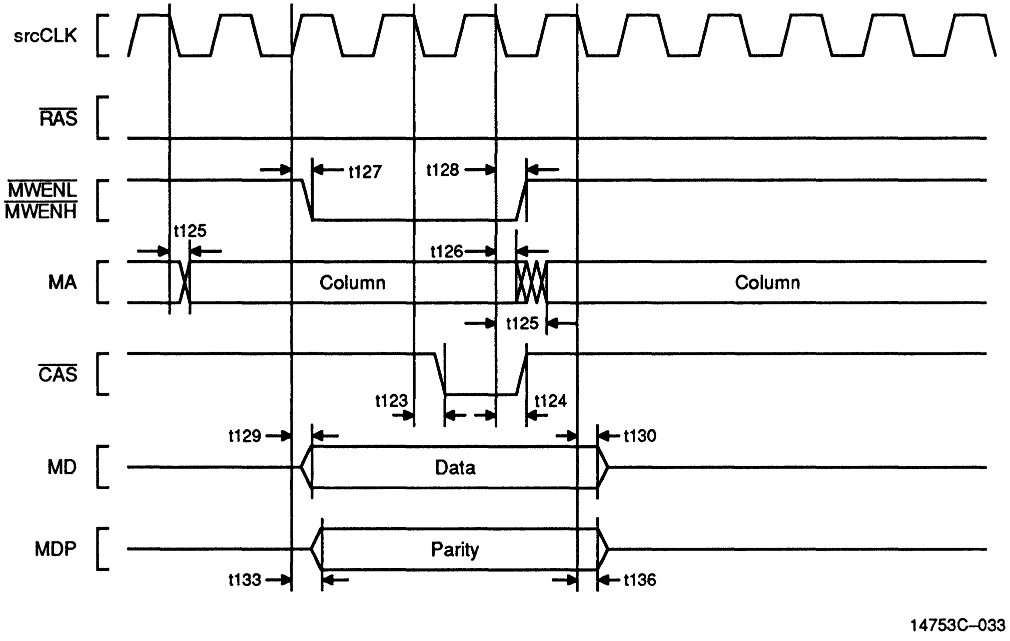


Figure 37. M Bus: Page Mode, Page—Hit/Write/0WS

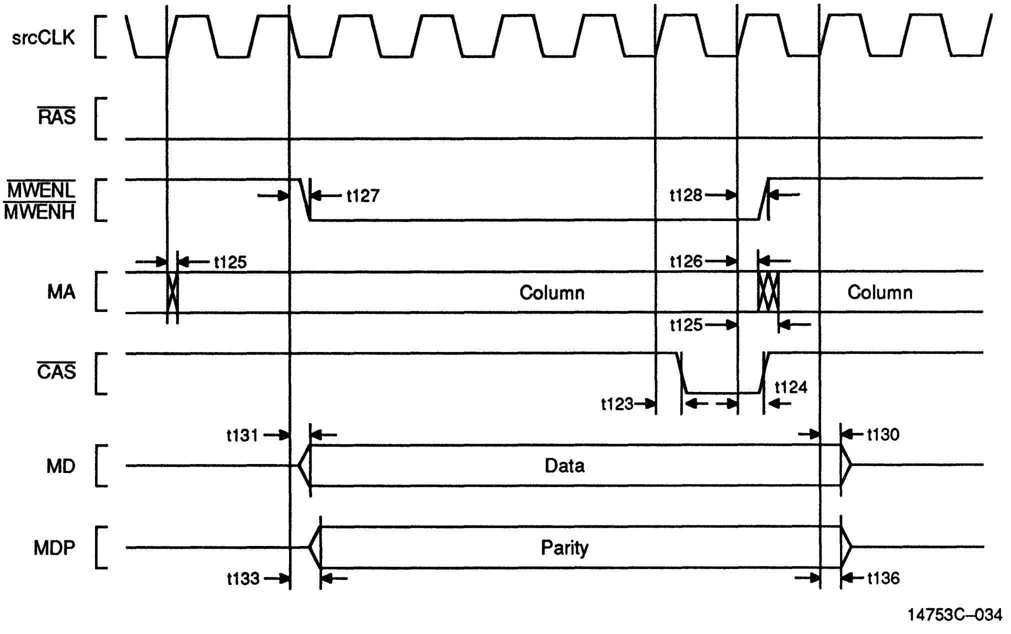


Figure 38. M Bus: Page Mode, Page—Hit $\overline{\text{RAS}}$ Active/Write/1WS

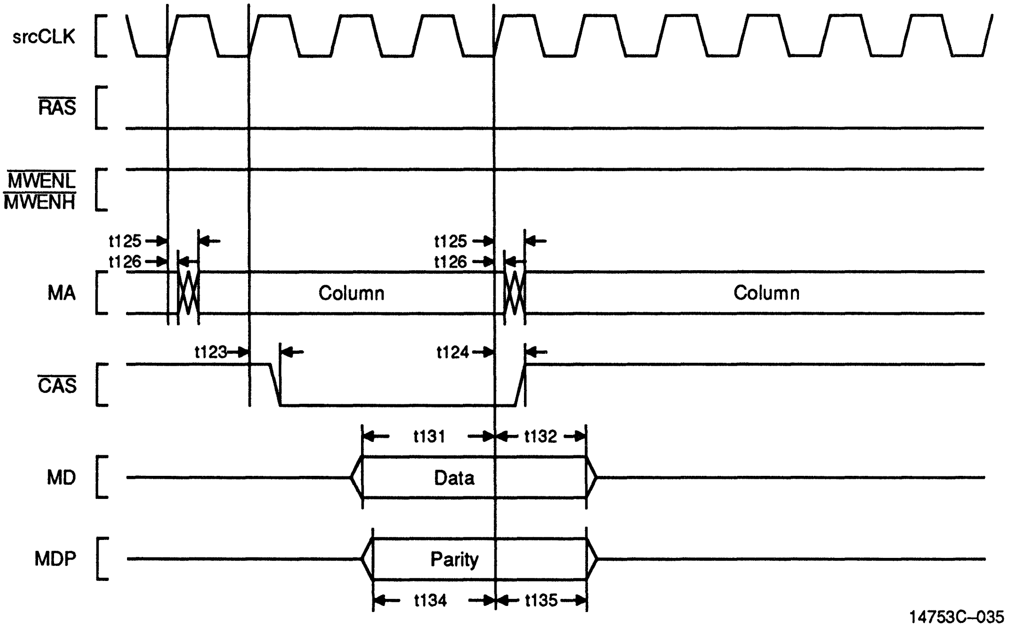
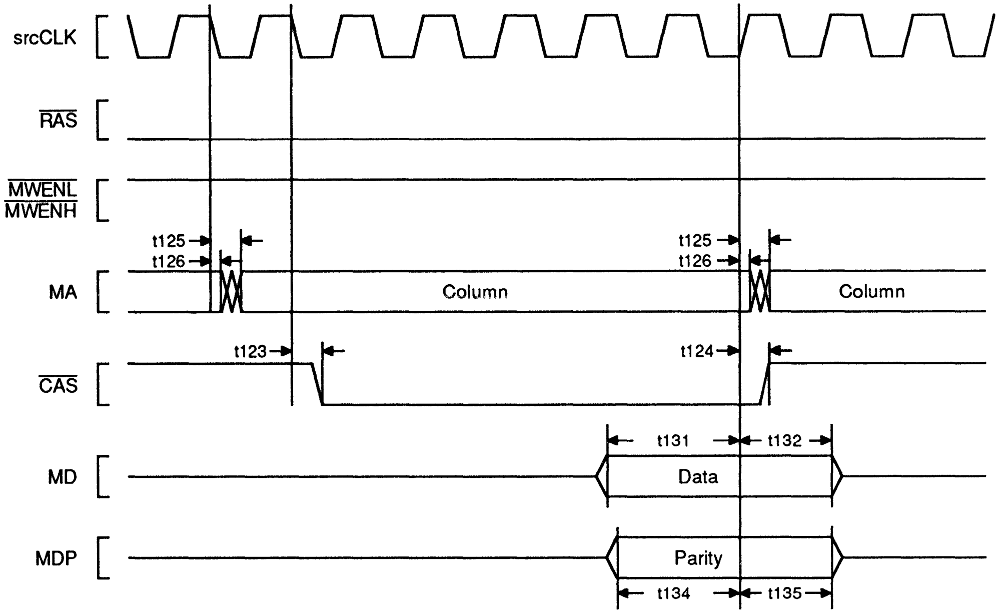
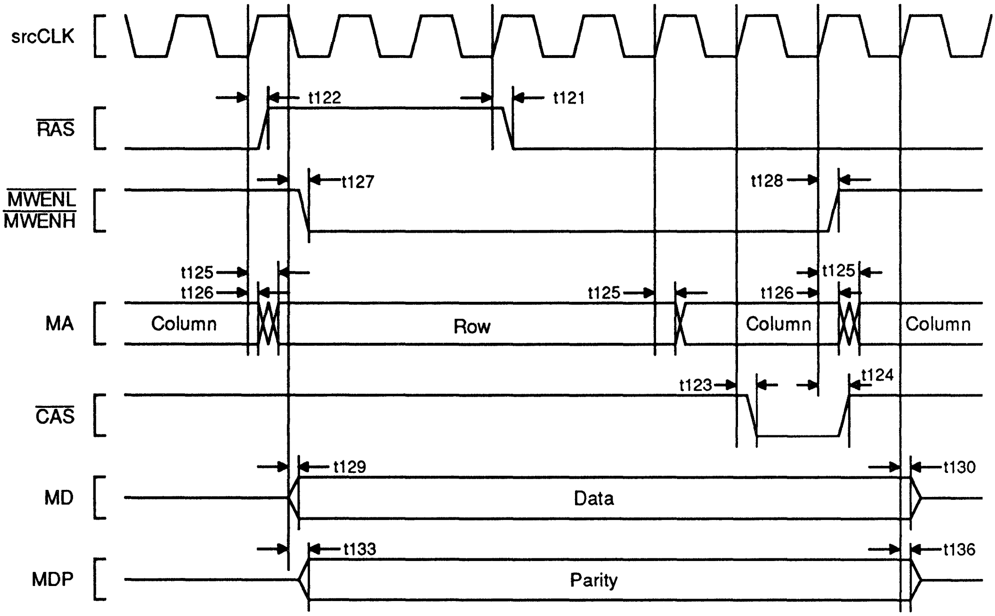


Figure 39. M Bus: Page Mode, Page—Hit $\overline{\text{RAS}}$ Active/0WS/Read



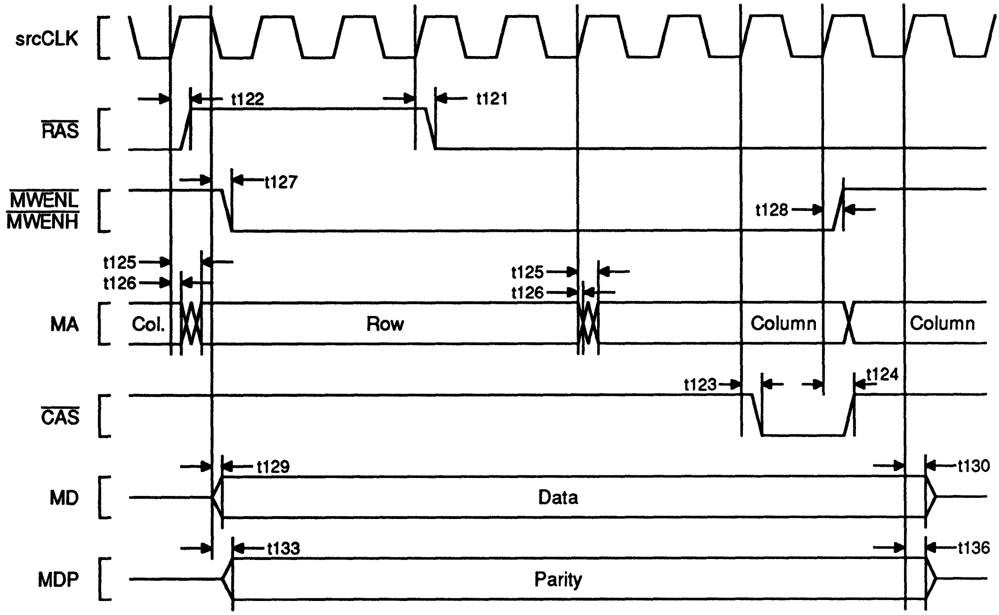
14753C–036

Figure 40. M Bus: Page Mode, Page—Hit $\overline{\text{RAS}}$ Active/1WS/Read



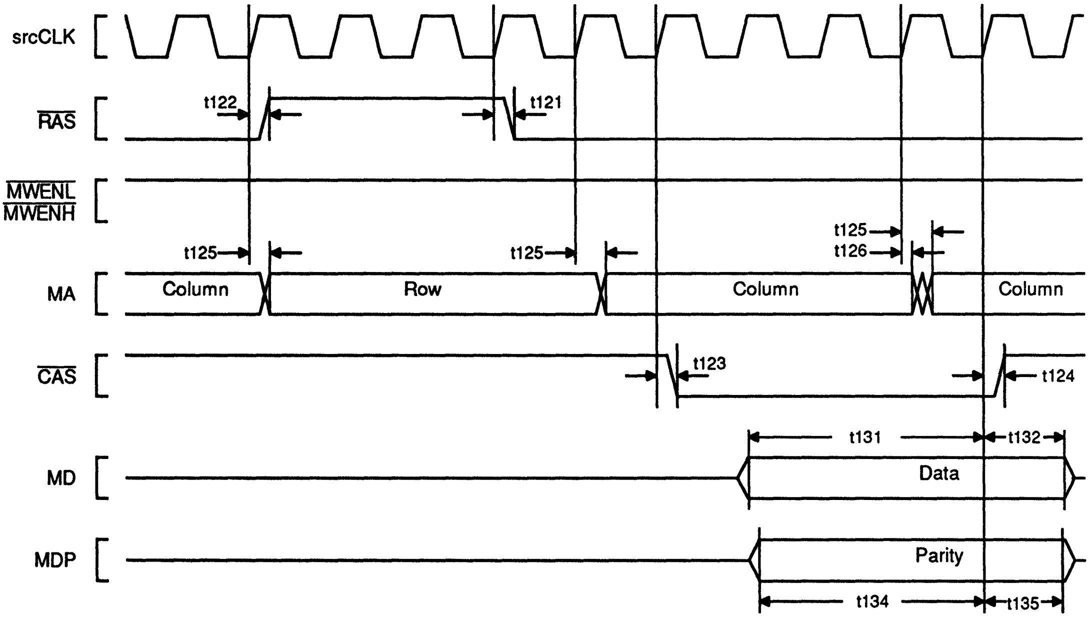
14753C–037

Figure 41. M Bus: Page Mode, Page—Miss/Write/0WS



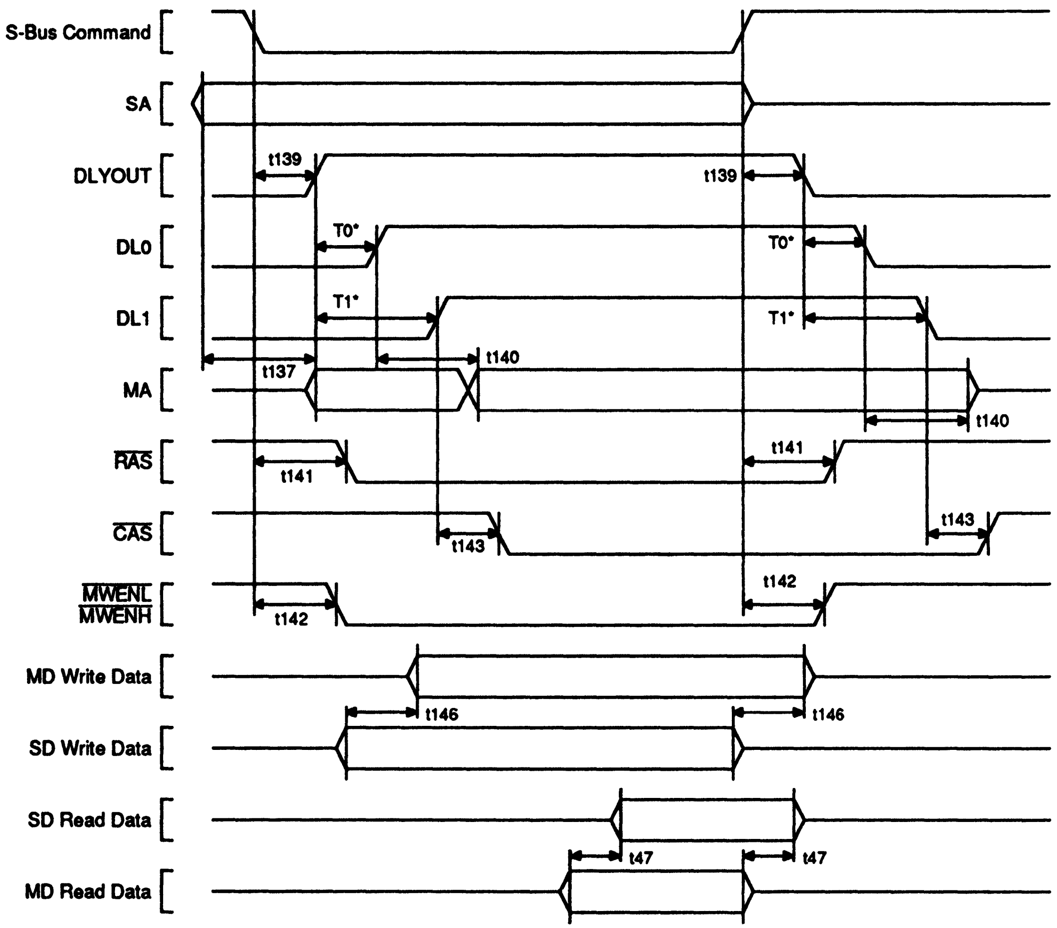
14753C–038

Figure 42. M Bus: Page Mode, Page—Miss/Write/1WS



14753C–039

Figure 43. M Bus: Page Mode, Page—Hit $\overline{\text{RAS}}$ Inactive/Read/0WS



*T0 and T1 are user selectable via an external delay circuit.

14753C–041

Figure 44. M Bus: S-Bus Master/DMA Access to Local Memory

Advanced
Micro
Devices

Am386™DXL

High-Performance, Low-Power, 32-Bit Microprocessor

DISTINCTIVE CHARACTERISTICS

- **Ideal for portable PCs**
  - –True static design for long battery life
  - –Typical standby Icc < 0.02 mA at DC (0 MHz)
  - –Typical operating Icc < 275 mA at 33 MHz
  - –Lower power consumption than Intel i386DX or Intel i386SX
  - –Small footprint 132-pin PQFP package
  - –Wide range of chip sets and BIOS available to support standby mode capabilities
  - –Performance on demand (0 to 40 MHz)
- **Ideal for desktop PCs**
  - –40-, 33-, 25-, and 20-MHz operating speeds
  - –Lower heat dissipation facilitates fan reduction or elimination for cost savings and noise reduction
  - –Pin-for-pin replacement for Intel i386DX
- **Compatible with 386DX systems and software**
- **Supports 387DX-compatible math coprocessors**
- **AMD® advanced 0.8 micron CMOS technology**

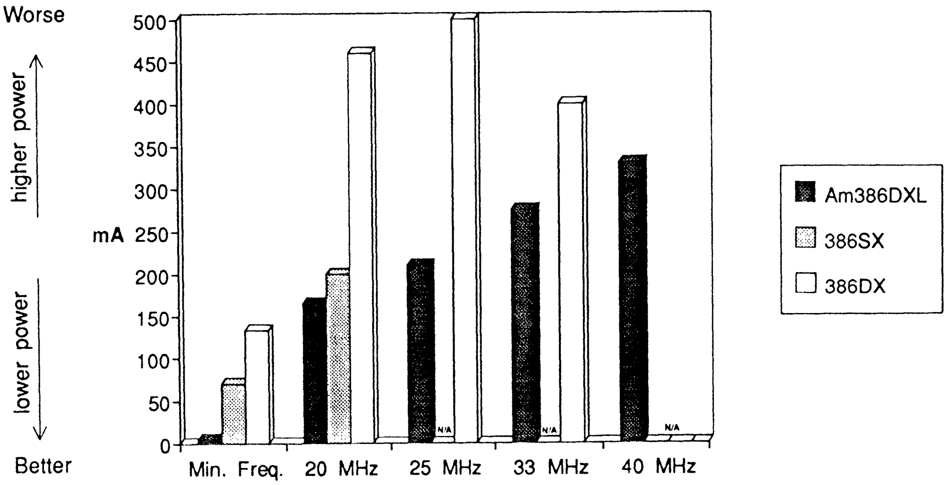
GENERAL DESCRIPTION

The Am386DXL microprocessor is a high-speed, true static implementation of the Intel i386DX microprocessor. It is ideal for both desktop and battery-powered portable personal computers. For desktop PCs, the Am386DXL microprocessor offers a 21% increase in the maximum operating speed from 33 to 40 MHz. Also, this device offers lower heat dissipation, allowing system designers to remove or reduce the size and cost of the system cooling fan.

For portables, the Am386DXL microprocessor's true static design offers longer battery life with low operating power consumption and standby mode. At 33 MHz, this device has 40% lower operating Icc than the Intel i386DX. Standby mode allows the Am386DXL microprocessor to be clocked down to 0 MHz (DC) and retain full register contents. In standby mode, typical current draw is less than 0.02 mA, a nearly 1000× reduction in power consumption versus the Intel i386DX or Intel i386SX.

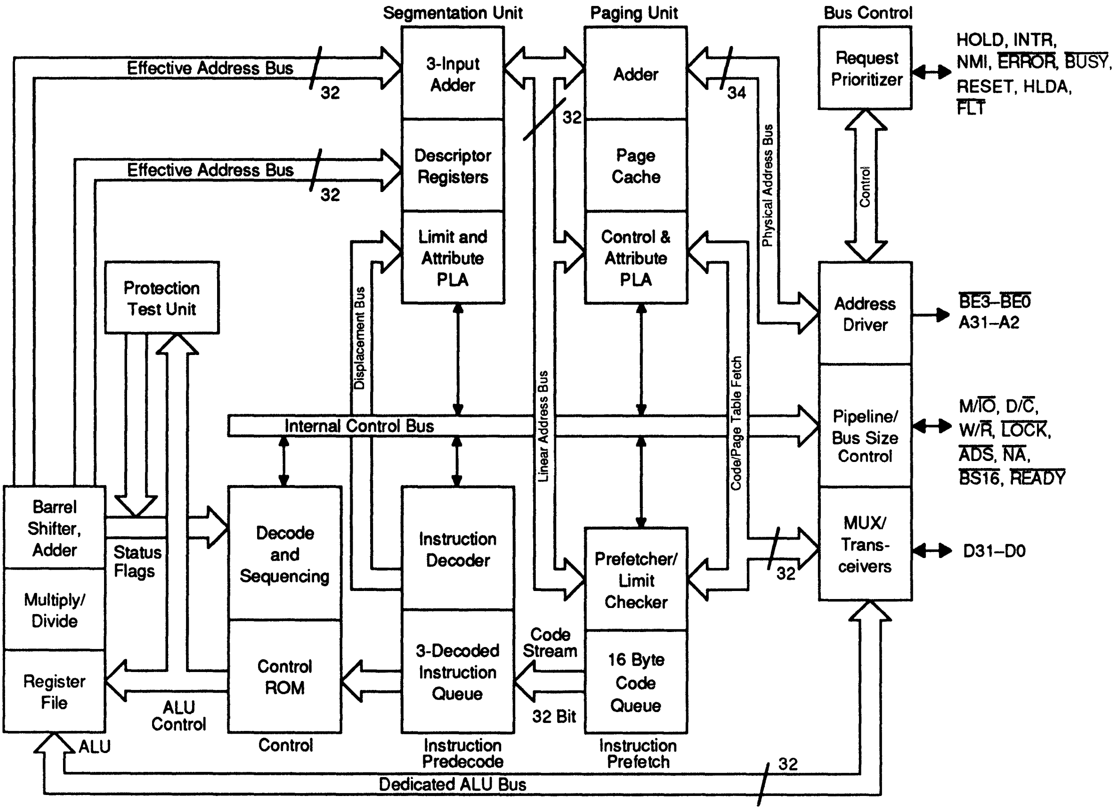
Additionally, the Am386DXL microprocessor will be available in a small footprint 132-pin plastic quad flat pack (PQFP) package. This surface-mount package is 40% smaller than PGA, allowing smaller, lower-cost board designs without the need for a socket.

Typical Icc



Publication # 15484 Rev. C Amendment /0
Issue Date: December 1991

BLOCK DIAGRAM



15021B–001

INTRODUCTION

AMD is proud to provide the Am386DXL microprocessor at a time when the personal computer market requires alternatives. Alternate source manufacturers traditionally increase availability, add features, and broaden the market. AMD has focused significant engineering resources to bring you these benefits.

AMD's track record with the 80286 shows that we were first to raise the performance of the 80286 from 8 MHz to 10, 12, and 16 MHz. We were first to offer new packaging technology with a smaller, lower cost PLCC. Today, over 90% of all 80286s sold use PLCC packaging. AMD is committed to similar advances with the Am386DXL microprocessor.

The Am386DXL microprocessor is more than just a re-creation of the Intel i386DX. Highly skilled engineers in our Austin, Texas facility added enhancements to the microprocessor, through the use of our advanced 0.8 micron CMOS technology. These enhancements include a true static design, and an increase in the maximum operating speed to 40 MHz. The true static design allows for lower operating power consumption, as well as standby mode for lower heat dissipation in desktop PCs and longer battery life in portables.

AMD engineered the Am386DXL microprocessor to insure compatibility with the installed base of hardware and software for the 386-based personal computers. The Am386DXL microprocessor is your solution to meet the demand for high-performance, 32-bit desktop and portable personal computers.

FUNCTIONAL DESCRIPTION

True Static Operation

The Am386DXL microprocessor incorporates a true static design. Unlike dynamic circuit design, the Am386DXL device eliminates the minimum operating frequency restriction. It may be clocked from its maximum speed of 40 MHz all the way down to 0 MHz (DC). System designers can use this feature to design true 32-bit battery-powered portable PCs with long battery life.

Standby Mode

This true static design allows for a standby mode. At any of its operating speeds (40 MHz to DC), the Am386DXL microprocessor will retain its state (i.e., the contents of all of its registers). By shutting off the clock completely, the device enters standby mode. Since power consumption is a function of clock frequency, operating power consumption is reduced as the frequency is lowered. In standby mode, typical current draw is reduced to less than 0.02 mA at DC.

Not only does this feature save battery life, but it also simplifies the design of power-conscious notebook computers in the following ways.

1. Eliminates the need for software in BIOS to save and restore the contents of registers.
2. Allows simpler circuitry to control stopping of the clock (since) the system does not need to know what state the processor is in.

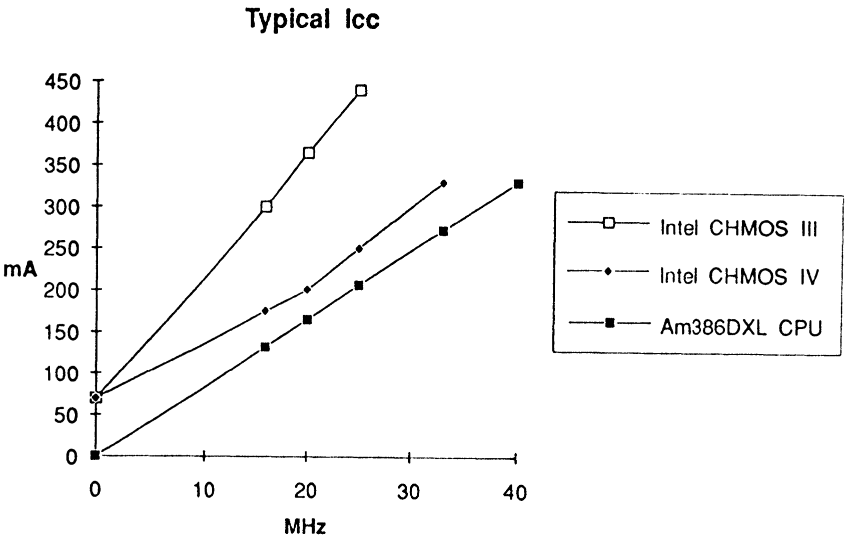
Lower Operating I_{CC}

True static design also allows lower operating I_{CC} when operating at any speed. See the following graph for typical current at operating speeds.

Performance On Demand

The Am386DXL microprocessor retains its state at any speed from 0 MHz (DC) to its maximum operating speed (20, 25, 33, or 40 MHz). With this feature, system designers may vary the operating speed of the system to extend the battery life in portable systems.

For example, the system could operate at low speeds during inactivity or polling operations. However, upon interrupt, the system clock can be increased up to its maximum speed. After a user-defined time-out period, the system can be returned to a low (or 0 MHz) operating speed without losing its state. This design maximizes life while achieving optimal performance.



CONNECTION DIAGRAMS
132-Lead Ceramic Pin Grid Array (PGA) Package

Top Side View

	P	N	M	L	K	J	H	G	F	E	D	C	B	A
1	A30	A27	A26	A23	A21	A20	A17	A16	A15	A14	A11	A8	V_{SS}	V_{CC}
2	V_{CC}	A31	A29	A24	A22	V_{SS}	A18	V_{CC}	V_{SS}	A13	A10	A7	A5	V_{SS}
3	D30	V_{SS}	V_{CC}	A28	A25	V_{SS}	A19	V_{CC}	V_{SS}	A12	A9	A6	A4	A3
4	D29	V_{CC}	V_{SS}									A2	NC	NC
5	D26	D27	D31									V_{CC}	V_{SS}	V_{CC}
6	V_{SS}	D25	D28									NC	NC	V_{SS}
7	D24	V_{CC}	V_{CC}									NC	INTR	V_{CC}
8	V_{CC}	D23	V_{SS}									PEREQ	NMI	$\overline{\text{ERROR}}$
9	D22	D21	D20									RESET	$\overline{\text{BUSY}}$	V_{SS}
10	D19	D17	V_{SS}									$\overline{\text{LOCK}}$	W/$\overline{\text{R}}$	V_{CC}
11	D18	D16	D15									V_{SS}	V_{SS}	D/$\overline{\text{C}}$
12	D14	D12	D10	V_{CC}	D7	V_{SS}	D0	V_{CC}	CLK2	$\overline{\text{BE0}}$	V_{CC}	V_{CC}	NC	M/$\overline{\text{IO}}$
13	D13	D11	V_{CC}	D8	D5	V_{SS}	D1	$\overline{\text{READY}}$	NC	NC	$\overline{\text{NA}}$	$\overline{\text{BE1}}$	$\overline{\text{BE2}}$	$\overline{\text{BE3}}$
14	V_{SS}	D9	HLDA	D6	D4	D3	D2	V_{CC}	V_{SS}	$\overline{\text{ADS}}$	HOLD	$\overline{\text{BS16}}$	V_{SS}	V_{CC}

Pin Side View

	A	B	C	D	E	F	G	H	J	K	L	M	N	P
1	V_{CC}	V_{SS}	A8	A11	A14	A15	A16	A17	A20	A21	A23	A26	A27	A30
2	V_{SS}	A5	A7	A10	A13	V_{SS}	V_{CC}	A18	V_{SS}	A22	A24	A29	A31	V_{CC}
3	A3	A4	A6	A9	A12	V_{SS}	V_{CC}	A19	V_{SS}	A25	A28	V_{CC}	V_{SS}	D30
4	NC	NC	A2									V_{SS}	V_{CC}	D29
5	V_{CC}	V_{SS}	V_{CC}									D31	D27	D26
6	V_{SS}	NC	NC									D28	D25	V_{SS}
7	V_{CC}	INTR	NC				Metal Lid					V_{CC}	V_{CC}	D24
8	$\overline{\text{ERROR}}$	NMI	PEREQ									V_{SS}	D23	V_{CC}
9	V_{SS}	$\overline{\text{BUSY}}$	RESET									D20	D21	D22
10	V_{CC}	W/$\overline{\text{R}}$	$\overline{\text{LOCK}}$									V_{SS}	D17	D19
11	D/$\overline{\text{C}}$	V_{SS}	V_{SS}									D15	D16	D18
12	M/$\overline{\text{IO}}$	NC	V_{CC}	V_{CC}	$\overline{\text{BE0}}$	CLK2	V_{CC}	D0	V_{SS}	D7	V_{CC}	D10	D12	D14
13	$\overline{\text{BE3}}$	$\overline{\text{BE2}}$	$\overline{\text{BE1}}$	$\overline{\text{NA}}$	NC	NC	$\overline{\text{READY}}$	D1	V_{SS}	D5	D8	V_{CC}	D11	D13
14	V_{CC}	V_{SS}	$\overline{\text{BS16}}$	HOLD	$\overline{\text{ADS}}$	V_{SS}	V_{CC}	D2	D3	D4	D6	HLDA	D9	V_{SS}

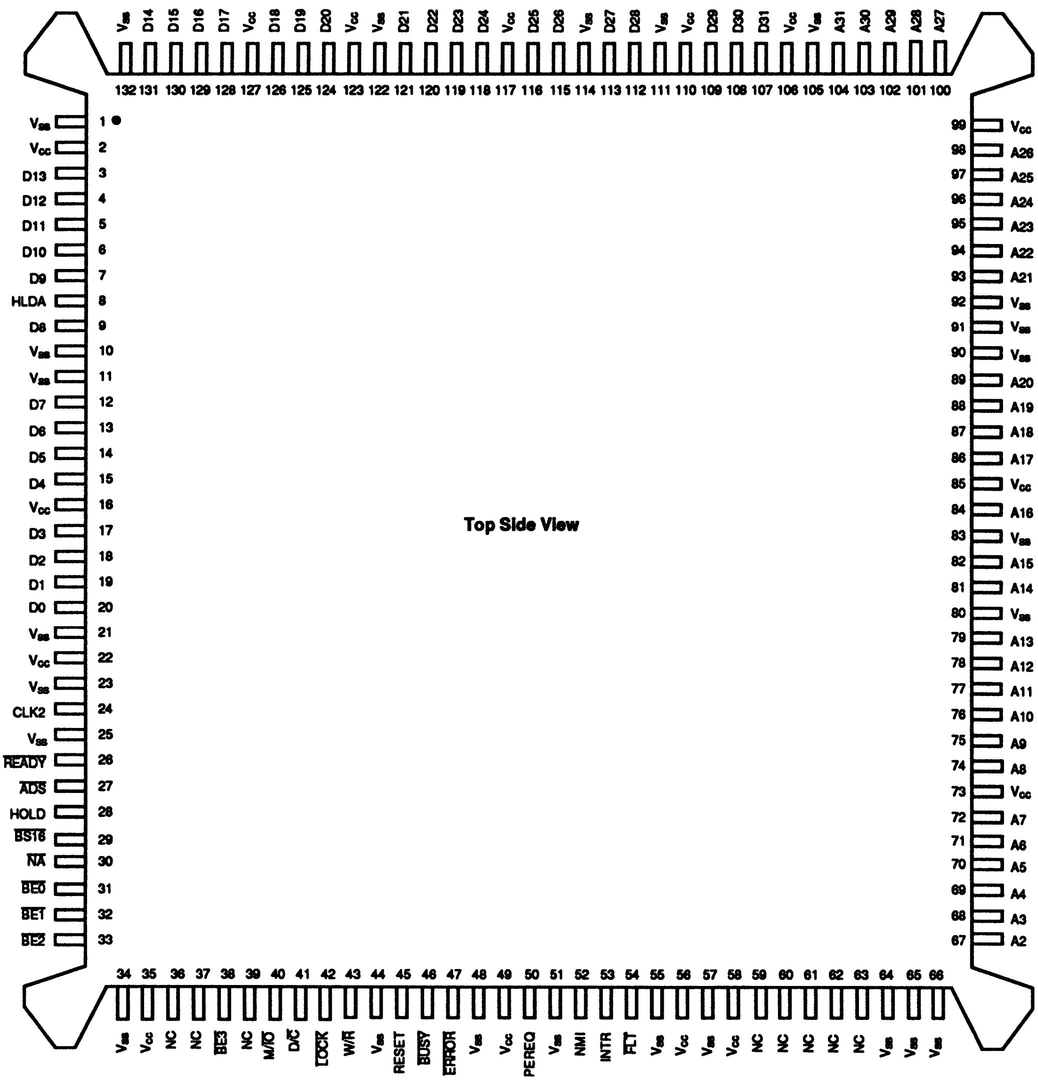
PGA Pin Designations (Functional Grouping)

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A2	C4	A24	L2	D6	L14	D28	M6	V_{cc}	C12	V_{ss}	F2
A3	A3	A25	K3	D7	K12	D29	P4		D12		F3
A4	B3	A26	M1	D8	L13	D30	P3		G2		F14
A5	B2	A27	N1	D9	N14	D31	M5		G3		J2
A6	C3	A28	L3	D10	M12	D/$\overline{\text{C}}$	A11		G12		J3
A7	C2	A29	M2	D11	N13	$\overline{\text{ERROR}}$	A8		G14		J12
A8	C1	A30	P1	D12	N12	HLDA	M14		L12		J13
A9	D3	A31	N2	D13	P13	HOLD	D14		M3		M4
A10	D2	$\overline{\text{ADS}}$	E14	D14	P12	INTR	B7		M7		M8
A11	D1	$\overline{\text{BE0}}$	E12	D15	M11	$\overline{\text{LOCK}}$	C10		M13		M10
A12	E3	$\overline{\text{BE1}}$	C13	D16	N11	M/$\overline{\text{IO}}$	A12		N4		N3
A13	E2	$\overline{\text{BE2}}$	B13	D17	N10	$\overline{\text{NA}}$	D13		N7		P6
A14	E1	$\overline{\text{BE3}}$	A13	D18	P11	NMI	B8		P2		P14
A15	F1	$\overline{\text{BS16}}$	C14	D19	P10	PEREQ	C8		P8	W/$\overline{\text{R}}$	B10
A16	G1	$\overline{\text{BUSY}}$	B9	D20	M9	$\overline{\text{READY}}$	G13	V_{ss}	A2	NC	A4
A17	H1	CLK2	F12	D21	N9	RESET	C9		A6		B4
A18	H2	D0	H12	D22	P9	V_{cc}	A1		A9		B6
A19	H3	D1	H13	D23	N8		A5		B1		B12
A20	J1	D2	H14	D24	P7		A7		B5		C6
A21	K1	D3	J14	D25	N6		A10		B11		C7
A22	K2	D4	K14	D26	P5		A14		B14		E13
A23	L1	D5	K13	D27	N5		C5		C11		F13

PGA Pin Designations (Sorted by Pin No.)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
A1	V_{cc}	B9	$\overline{\text{BUSY}}$	D3	A9	H1	A17	L13	D8	N7	V_{cc}
A2	V_{ss}	B10	W/$\overline{\text{R}}$	D12	V_{cc}	H2	A18	L14	D6	N8	D23
A3	A3	B11	V_{ss}	D13	$\overline{\text{NA}}$	H3	A19	M1	A26	N9	D21
A4	NC	B12	NC	D14	HOLD	H12	D0	M2	A29	N10	D17
A5	V_{cc}	B13	$\overline{\text{BE2}}$	E1	A14	H13	D1	M3	V_{cc}	N11	D16
A6	V_{ss}	B14	V_{ss}	E2	A13	H14	D2	M4	V_{ss}	N12	D12
A7	V_{cc}	C1	A8	E3	A12	J1	A20	M5	D31	N13	D11
A8	$\overline{\text{ERROR}}$	C2	A7	E12	$\overline{\text{BE0}}$	J2	V_{ss}	M6	D28	N14	D9
A9	V_{ss}	C3	A6	E13	NC	J3	V_{ss}	M7	V_{cc}	P1	A30
A10	V_{cc}	C4	A2	E14	$\overline{\text{ADS}}$	J12	V_{ss}	M8	V_{ss}	P2	V_{cc}
A11	D/$\overline{\text{C}}$	C5	V_{cc}	F1	A15	J13	V_{ss}	M9	D20	P3	D30
A12	M/$\overline{\text{IO}}$	C6	NC	F2	V_{ss}	J14	D3	M10	V_{ss}	P4	D29
A13	$\overline{\text{BE3}}$	C7	NC	F3	V_{ss}	K1	A21	M11	D15	P5	D26
A14	V_{cc}	C8	PEREQ	F12	CLK2	K2	A22	M12	D10	P6	V_{ss}
B1	V_{ss}	C9	RESET	F13	NC	K3	A25	M13	V_{cc}	P7	D24
B2	A5	C10	$\overline{\text{LOCK}}$	F14	V_{ss}	K12	D7	M14	HLDA	P8	V_{cc}
B3	A4	C11	V_{ss}	G1	A16	K13	D5	N1	A27	P9	D22
B4	NC	C12	V_{cc}	G2	V_{cc}	K14	D4	N2	A31	P10	D19
B5	V_{ss}	C13	$\overline{\text{BE1}}$	G3	V_{cc}	L1	A23	N3	V_{ss}	P11	D18
B6	NC	C14	$\overline{\text{BS16}}$	G12	V_{cc}	L2	A24	N4	V_{cc}	P12	D14
B7	INTR	D1	A11	G13	$\overline{\text{READY}}$	L3	A28	N5	D27	P13	D13
B8	NMI	D2	A10	G14	V_{cc}	L12	V_{cc}	N6	D25	P14	V_{ss}

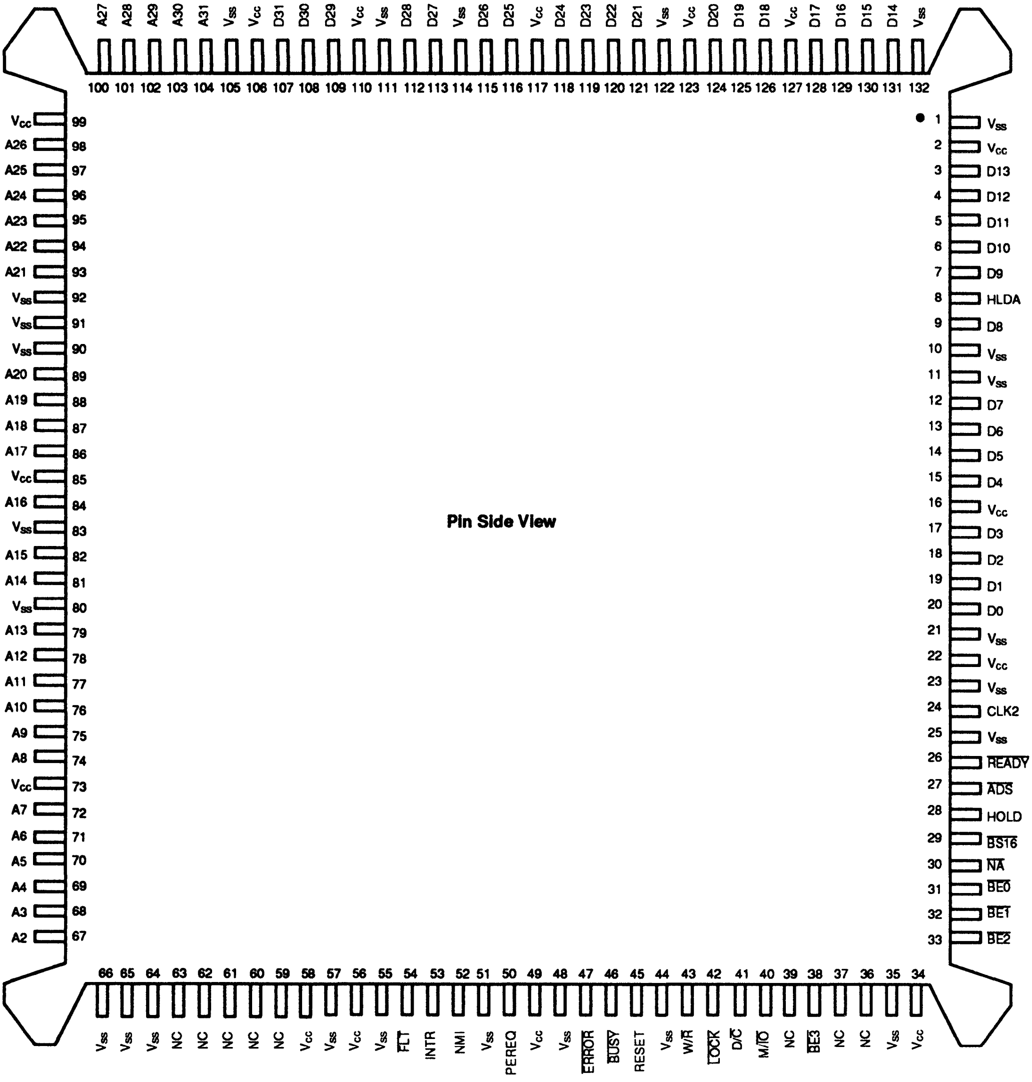
CONNECTION DIAGRAMS
132-Lead Plastic Quad Flat Pack (PQFP) Package



15022B-002

Note: Pin 1 is marked for orientation.

CONNECTION DIAGRAMS (continued)
132-Lead Plastic Quad Flat Pack (PQFP) Package



15022B-002

Notes: Pin 1 is marked for orientation.

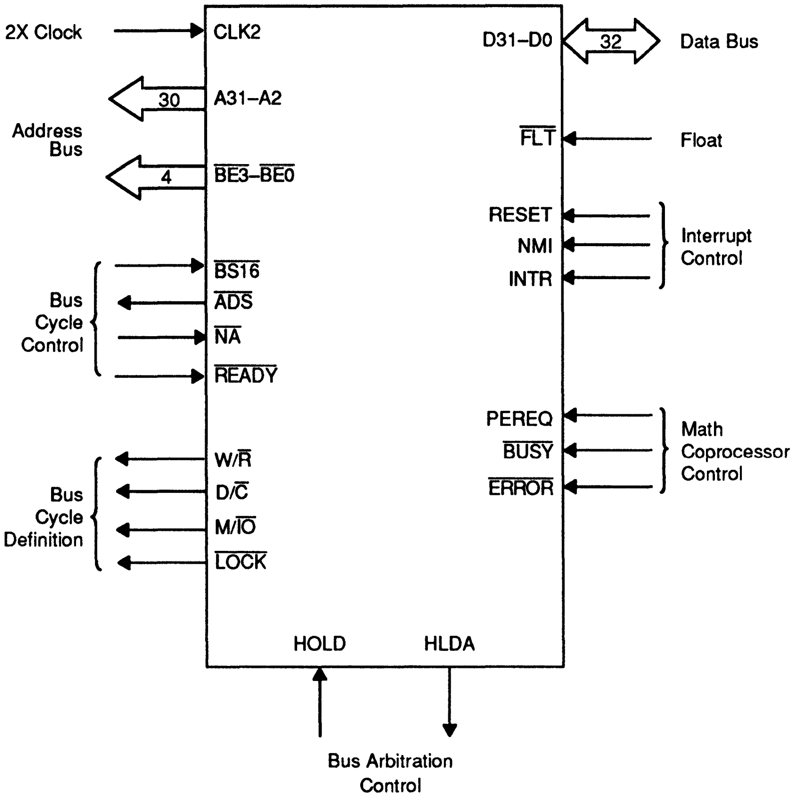
PQFP Pin Designations (Functional Grouping)

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A2	67	A24	96	D6	13	D28	112	V_{cc}	56	V_{ss}	64
A3	68	A25	97	D7	12	D29	109		58		65
A4	69	A26	98	D8	9	D30	108		73		66
A5	70	A27	100	D9	7	D31	107		85		80
A6	71	A28	101	D10	6	$D/\overline{C}$	41		99		83
A7	72	A29	102	D11	5	$\overline{ERROR}$	47		106		90
A8	74	A30	103	D12	4	HLDA	8		110		91
A9	75	A31	104	D13	3	HOLD	28		117		92
A10	76	$\overline{ADS}$	27	D14	131	INTR	53		123		105
A11	77	$\overline{BE0}$	31	D15	130	$\overline{LOCK}$	42		127		111
A12	78	$\overline{BE1}$	32	D16	129	$M/\overline{IO}$	40	V_{ss}	1		114
A13	79	$\overline{BE2}$	33	D17	128	$\overline{NA}$	30		10		122
A14	81	$\overline{BE3}$	38	D18	126	NMI	52		11		132
A15	82	$\overline{BS16}$	29	D19	125	PEREQ	50		21	$W/\overline{R}$	43
A16	84	$\overline{BUSY}$	46	D20	124	$\overline{READY}$	26		23	NC	36
A17	86	CLK2	24	D21	121	RESET	45		25		37
A18	87	D0	20	D22	120	V_{cc}	2		35		39
A19	88	D1	19	D23	119		16		44		59
A20	89	D2	18	D24	118		22		48		60
A21	93	D3	17	D25	116		34		51		61
A22	94	D4	15	D26	115		49		55		62
A23	95	D5	14	D27	113	$\overline{FLT}$	54		57		63

PQFP Pin Designations (Sorted by Pin No.)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	V_{ss}	23	V_{ss}	45	RESET	67	A2	89	A20	111	V_{ss}
2	V_{cc}	24	CLK2	46	$\overline{BUSY}$	68	A3	90	V_{ss}	112	D28
3	D13	25	V_{ss}	47	$\overline{ERROR}$	69	A4	91	V_{ss}	113	D27
4	D12	26	$\overline{READY}$	48	V_{ss}	70	A5	92	V_{ss}	114	VSS
5	D11	27	$\overline{ADS}$	49	V_{cc}	71	A6	93	A21	115	D26
6	D10	28	HOLD	50	PEREQ	72	A7	94	A22	116	D25
7	D9	29	$\overline{BS16}$	51	V_{ss}	73	V_{cc}	95	A23	117	V_{cc}
8	HLDA	30	$\overline{NA}$	52	NMI	74	A8	96	A24	118	D24
9	D8	31	$\overline{BE0}$	53	INTR	75	A9	97	A25	119	D23
10	V_{ss}	32	$\overline{BE1}$	54	$\overline{FLT}$	76	A10	98	A26	120	D22
11	V_{ss}	33	$\overline{BE2}$	55	V_{ss}	77	A11	99	V_{cc}	121	D21
12	D7	34	V_{cc}	56	V_{cc}	78	A12	100	A27	122	V_{ss}
13	D6	35	V_{ss}	57	V_{ss}	79	A13	101	A28	123	V_{cc}
14	A5	36	NC	58	V_{cc}	80	V_{ss}	102	A29	124	D20
15	D4	37	NC	59	NC	81	A14	103	A30	125	D19
16	V_{cc}	38	$\overline{BE3}$	60	NC	82	A15	104	A31	126	D18
17	D3	39	NC	61	NC	83	V_{ss}	105	V_{ss}	127	V_{cc}
18	D2	40	$M/\overline{IO}$	62	NC	84	A16	106	V_{cc}	128	D17
19	D1	41	$D/\overline{C}$	63	NC	85	V_{cc}	107	D31	129	D16
20	D0	42	$\overline{LOCK}$	64	V_{ss}	86	A17	108	D30	130	D15
21	V_{ss}	43	$W/\overline{R}$	65	V_{ss}	87	A18	109	D29	131	D14
22	V_{cc}	44	V_{ss}	66	V_{ss}	88	A19	110	V_{cc}	132	V_{ss}

LOGIC SYMBOL

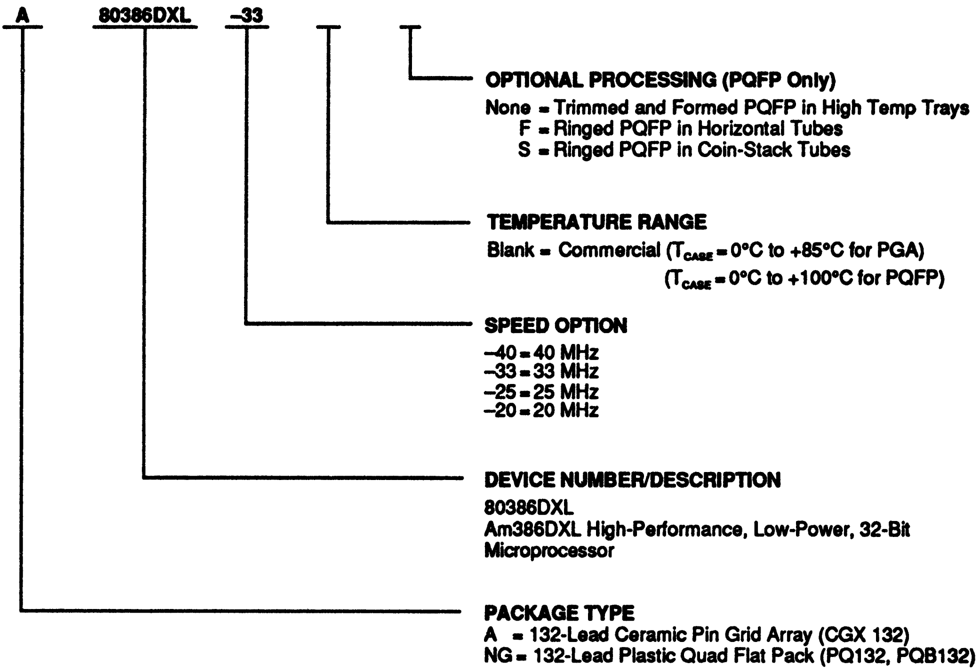


15021B–003

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



A 80386DXL -33

OPTIONAL PROCESSING (PQFP Only)
None = Trimmed and Formed PQFP in High Temp Trays
F = Ringed PQFP in Horizontal Tubes
S = Ringed PQFP in Coin-Stack Tubes

TEMPERATURE RANGE
Blank = Commercial (T_{CASE} = 0°C to +85°C for PGA)
(T_{CASE} = 0°C to +100°C for PQFP)

SPEED OPTION
–40 = 40 MHz
–33 = 33 MHz
–25 = 25 MHz
–20 = 20 MHz

DEVICE NUMBER/DESCRIPTION
80386DXL
Am386DXL High-Performance, Low-Power, 32-Bit Microprocessor

PACKAGE TYPE
A = 132-Lead Ceramic Pin Grid Array (CGX 132)
NG = 132-Lead Plastic Quad Flat Pack (PQ132, PQB132)

Valid Combinations	
A80386DXL	–40 –33 –25 –20
NG80386DXL	–33F, –33S –25F, –25S –20F, –20S

Valid Combinations

Valid Combinations lists configurations planned to be supported in volume for this device. All speeds may not be available in all package combinations. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations.

PIN DESCRIPTION

A31–A2

Address Bus (Outputs)

Outputs physical memory or port I/O addresses.

$\overline{\text{ADS}}$

Address Status (Active Low; Output)

Indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BE0}}$, $\overline{\text{BE1}}$, $\overline{\text{BE2}}$, $\overline{\text{BE3}}$, and A31–A2) are being driven at the Am386DXL microprocessor pins.

$\overline{\text{BE3}}$–$\overline{\text{BE0}}$

Byte Enables (Active Low; Outputs)

Indicate which data bytes of the data bus take part in a bus cycle.

$\overline{\text{BS16}}$

Bus Size 16 (Active Low; Input)

Allows direct connection of 32-bit and 16-bit data buses.

$\overline{\text{BUSY}}$

Busy (Active Low; Input)

Signals a busy condition from a processor extension.

CLK2

Clock (Input)

Provides the fundamental timing for the Am386DXL microprocessor.

D31–D0

Data Bus (Inputs/Outputs)

Inputs data during memory, I/O, and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.

D/$\overline{\text{C}}$

Data/Control (Output)

A bus cycle definition pin that distinguishes data cycles, either memory or I/O from control cycles which are: interrupt acknowledge, halt, and instruction fetching.

$\overline{\text{ERROR}}$

Error (Active Low; Input)

Signals an error condition from a processor extension.

$\overline{\text{FLT}}$

Float (Active Low; Input)

An input signal which forces all bi-directional and output signals, including HLDA, to the three-state condition.

$\overline{\text{FLT}}$ has an internal pull-up resistor, and if it is not used it should be unconnected.

HLDA

Bus Hold Acknowledge (Active High; Output)

Indicates that the Am386DXL microprocessor has surrendered control of its local bus to another bus master.

HOLD

Bus Hold Request (Active High; Input)

Allows another bus master to request control of the local bus.

INTR

Interrupt Request (Active High; Input)

A maskable input that signals the Am386DXL microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

$\overline{\text{LOCK}}$

Bus Lock (Active Low; Output)

A bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active.

M/$\overline{\text{IO}}$

Memory I/O (Output)

A bus cycle definition pin that distinguishes memory cycles from input/output cycles.

$\overline{\text{NA}}$

Next Address (Active Low; Input)

Used to request address pipelining.

NC

No Connect

Should always remain unconnected. Connection of a NC pin may cause the processor to malfunction or be incompatible with future steppings of the Am386DXL microprocessor.

NMI

Non-Maskable Interrupt Request (Active High; Input)

A non-maskable input that signals the Am386DXL microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

PEREQ

Processor Extension Request (Active High; Input)

Indicates that the processor extension has data to be transferred by the Am386DXL microprocessor.

$\overline{\text{READY}}$

Bus Ready (Active Low; Input)

Terminates the bus cycle.

RESET

Reset (Active High; Input)

Suspends any operation in progress and places the Am386DXL microprocessor in a known reset state.

V_{CC}

System Power (Active High; Input)

Provides the +5-V nominal DC supply input.

V_{SS}

System Ground (Input)

Provides 0 V connection from which all inputs and outputs are measured.

W/$\overline{\text{R}}$

Write/Read (Output)

A bus cycle definition pin that distinguishes write cycles from read cycles.

BASE ARCHITECTURE

Introduction

The Am386DXL microprocessor consists of a central processing unit, a memory management unit, and a bus interface.

The central processing unit consists of the execution unit and instruction unit. The execution unit contains the eight 32-bit general purpose registers that are used for both address calculation, data operations, and a 64-bit barrel shifter used to speed shift, rotate, multiply and divide operations. The multiply and divide logic uses a 1-bit per cycle algorithm. The multiply algorithm stops the iteration when the most significant bits of the multiplier are all zero. This allows typical 32-bit multiplies to be executed in under 1 ms. The instruction unit decodes the instruction op-codes and stores them in the decoded instruction queue for immediate use by the execution unit.

The Memory Management Unit (MMU) consists of a segmentation unit and a paging unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability and efficient sharing. The paging mechanism operates beneath and is transparent to the segmentation process to allow management of the physical address space. Each segment is divided into one or more 4-Kb pages. To implement a virtual memory system, the Am386DXL microprocessor supports full restartability for all page and segment faults.

Memory is organized into one or more variable length segments, each up to 4 Gb in size. A given region of the linear address space, a segment, can have attributes associated with it. These attributes include its location, size, type (i.e., stack, code or data), and protection characteristics. Each task on an Am386DXL microprocessor can have a maximum of 16,381 segments of up to 4 Gb each, thus providing 64 tb (trillion bytes) or virtual memory to each task.

The segmentation unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of system with a high degree of integrity.

The Am386DXL microprocessor has two modes of operation: Real Address Mode (Real Mode) and Protected Virtual Address Mode (Protected Mode). In Real Mode, the Am386DXL device operates as a very fast 8086 but with 32-bit extensions, if desired. Real Mode is required primarily to setup the processor for Protected Mode operation. Protected Mode provides address to the sophisticated memory management, paging, and privilege capabilities of the processor.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each such tasks behaves with 8086 semantics, thus allowing 8086 software (an application program or an entire operating system) to execute. The Virtual 8086 tasks can be isolated and protected from one another and the host Am386DXL microprocessor operating system by the use of paging and the I/O Permission Bitmap.

Finally, to facilitate high-performance system hardware designs, the Am386DXL microprocessor bus interface offers address pipelining, dynamic data bus sizing, and direct Byte Enable signals for each byte of the data bus. These hardware features are described fully beginning in the Functional Data section.

Register Overview

The Am386DXL microprocessor has 32 register resources in the following categories.

- General Purpose Registers
- Segment Registers
- Instruction Pointer and Flags
- Control Registers
- System Address Registers
- Debug Registers
- Test Registers

The registers are a superset of the 8086, 80186, and 80286 registers, so all 16-bit 80186 and 80286 registers are contained within the 32-bit Am386DXL microprocessor.

Figure 1 shows all of Am386DXL microprocessor base architecture registers that include the general address and data registers, the instruction pointer, and the flags register. The contents of these registers are task-specific, so these registers are automatically loaded with a new context upon a task switch operation.

The base architecture also includes six directly accessible segments, each up to 4 Gb in size. The segments are indicated by the selector values placed in Am386DXL CPU segment registers of Figure 1. Various selector values can be loaded as a program executes, if desired.

The selectors are also task specific, so the segment registers are automatically loaded with new context upon a task switch operation.

The other types of registers Control, System Address, Debug, and Test are primarily used by system software.

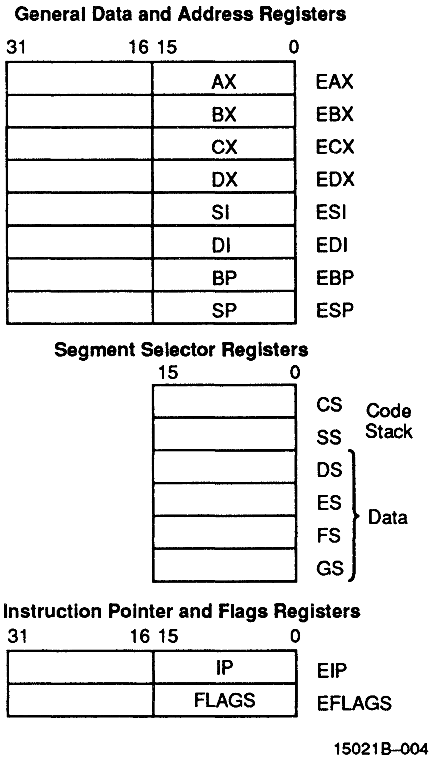


Figure 1. Base Architecture Registers

Register Descriptions

General-Purpose Registers

The eight general-purpose registers of 32 bits hold data or address quantities. The general registers, Figure 2, support data operands of 1, 8, 16, 32, and 64 bits and bit fields of 1 to 32 bits. They support address operands of 16 and 32 bits. The 32-bit registers are named EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP.

The least significant 16 bits of the registers can be accessed separately. This is done by using the 16-bit names of the registers AX, BX, CX, DX, SI, DI, BP, and SP. When accessed as a 16-bit operand, the upper 16 bits of the register are neither used nor changed.

Finally, 8-bit operations can individually access the lower byte (bits 7–0) and the higher byte (bits 15–8) of general purpose registers AX, BX, CX, and DX. The lower bytes are named AL, BL, CL, and DL, respectively. The higher bytes are named AH, BH, CH, and DH, respectively. The individual byte accessibility offers additional flexibility for data operations, but is not used for effective address calculation.

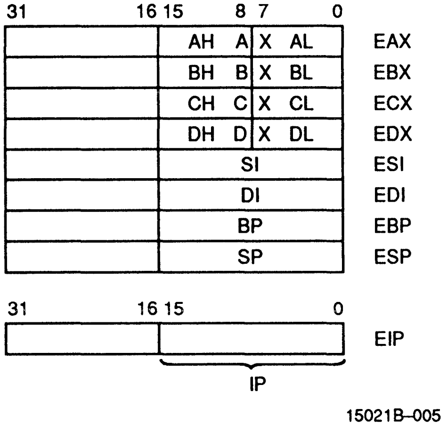


Figure 2. General Registers and Instruction Pointer

Instruction Pointer

The instruction pointer, Figure 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 15–0) of EIP contain the 16-bit instruction pointer named IP, which is used by 16-bit addressing.

Flags Register

The Flags Register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS shown in Figure 3 control certain operations and indicate status of the Am386DXL microprocessor. The lower 16 bits (bits 15–0) of EFLAGS contain the 16-bit flag register named FLAGS, which is most useful when executing 8086 and 80286 code.

VM (Virtual 8086 Mode, bit 17)

The VM bit provides Virtual 8086 Mode within Protected Mode. If set while the Am386DXL microprocessor is in Protected Mode, the Am386DXL microprocessor will switch to Virtual 8086 operation, handling segment loads as the 8086 does, but generating Exception 13 faults on privileged op-codes. 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, (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 EFLAGS image having the RF bit set and resume the program's execution at the breakpoint address without generating another breakpoint fault on the same location.

NT (Nested Task, bit 14)

This flag applies to Protected Mode. NT is set to indicate that the execution of this task is nested within another task. If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS. This bit is set or reset by control transfers to other tasks. The value of NT in EFLAGS is tested by the IRET instruction to determine whether to do an inter-task return or an intra-task return. A POPF or an IRET instruction will affect the setting of this bit according to the image popped at any privilege level.

IOPL (Input/Output Privilege Level, bits 12–13)

This two-bit field applies to Protected Mode. IOPL indicates the numerically maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an Exception 13 fault or consulting the I/O Permission Bitmap. It also indicates the maximum CPL value allowing alteration of the IF (INTR Enable Flag) bit when new values are popped into the EFLAGS 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)

OD 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 bits 7, 15, 31, respectively.

DF (Direction Flag, bit 10)

DF defines whether ESI and/or EDI registers postdecrement or postincrement during the sg instructions. Postincrement occurs if DF is reset. Postdecrement occurs if DF is set.

IF (INTR Enable Flag, bit 9)

The IF flag, when set, allows recognition of external interrupts 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 Am386DXL microprocessor generates an Exception 1 trap after the next instruction is executed. When TF is reset, Exception 1 traps occur only as a function of the breakpoint addresses loaded into debug register DR3–DR0.

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 bits 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.

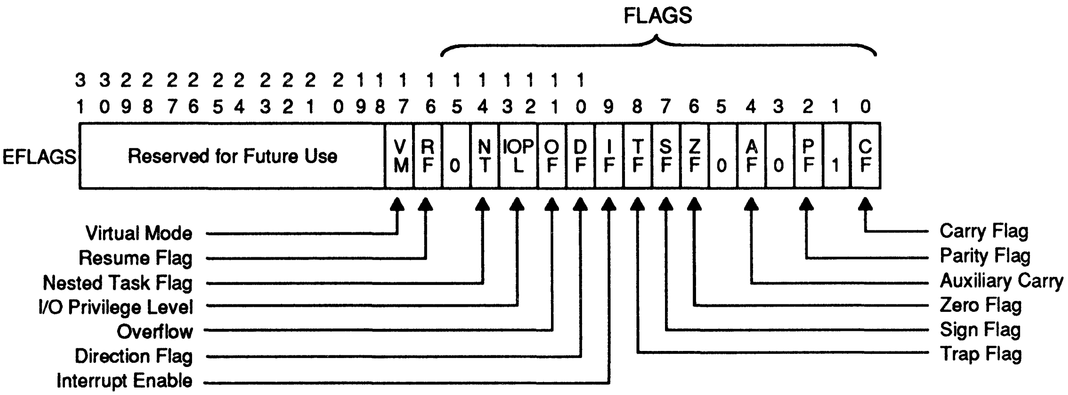
PF (Parity flags, bit 2)

PF is set if the low-order 8 bits of the operation contain an even number of 1s (even parity). PF is reset if the low-order 8 bits have odd parity. PF is a function of only the low-order 8 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 bits 7, 15, or 31, respectively.

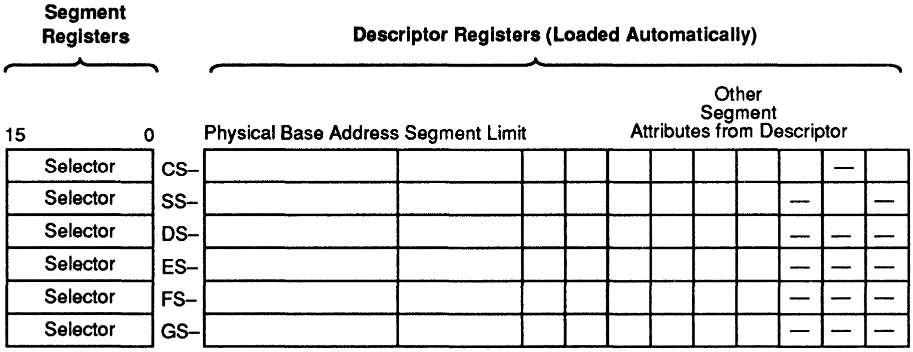
Note in these descriptions, set means set to 1 and reset means reset to 0.



Note: 0 indicates "Reserved for Future Use." Do not define; see Section Compatibility.

15021B–006

Figure 3. FLAGS Registers



15021B–007

Figure 4. Segment Registers and Associated Descriptor Registers

Segment Registers

Six 16-bit segment registers hold segment selector values identifying the currently addressable memory segments. Segment registers are shown in Figure 4. In Protected Mode, each segment may range in size from one byte up to the entire linear and physical space of the machine, 4 Gb (2^{32} bytes). If a maximum sized segment is used (limit = FFFFFFFFH) it should be Dword aligned (i.e., the least two significant bits of the segment base should be zero). This will avoid a segment limit violation (Exception 13) caused by the wrap around. In Real Address Mode, the maximum segment size is fixed at 64 Kb (2^{16} bytes).

The six segments addressable at any given moment are defined by the segment registers: CS, SS, DS, ES, FS, and GS. The selector in SS indicates the current stack segment; the selectors in DS, ES, FS, and GS indicate the current data segments.

Segment Descriptor Registers

The segment descriptor registers are not programmer visible, yet it is very useful to understand their content. Inside the Am386DXL microprocessor, a descriptor register (programmer invisible) is associated with each programmer-visible segment register, as shown by Figure 4. Each descriptor register holds a 32-bit segment base address, a 32-bit segment limit, and the other necessary segment attributes.

When a selector value is loaded into a segment register, the associated descriptor register is automatically updated with the correct information. In Real Address Mode, only the base address is updated directly (by shifting the selector value four bits to the left), since the segment maximum limit and attributes are fixed in Real Mode. In Protected Mode, the base address, the limit, and the attributes are all updated per the contents of the segment descriptor indexed by the selector.

Whenever a memory reference occurs, the segment descriptor register associated with the segment being used is automatically involved with the memory reference. The 32-bit segment base address becomes a component of the linear address calculation, the 32-bit limit is used for the limit-check operation, and the attributes are checked against the type of memory reference requested.

Control Registers

The Am386DXL microprocessor has three control registers of 32 bits: CR0, CR2, and CR3 to hold machine state of a global nature (not specific to an individual task). These registers, along with System Address Registers described in the next section, hold machine state that affects all tasks in the system. To access the Control Registers, load and store instructions are defined.

CR0: Machine Control Register (Includes 80286 Machine Status Word)

CR0, shown in Figure 5, contains 6 defined bits for control and status purposes. The low-order 16 bits of CR0 are also known as the Machine Status Word (MSW) for compatibility with 80286 Protected Mode. LMSW and SMSW instructions are taken as special aliases of the load and store CR0 operations, where only the low-order 16 bits of CR0 are involved. For compatibility with 80286 operating systems, the Am386DXL microprocessor LMSW instructions work in an identical fashion to the LMSW instruction on the 80286 (i.e., it only operates on the low-order 16 bits of CR0 and it ignores the new bits in CR0). New Am386DXL microprocessor operating systems should use the MOV CR0, Reg instruction.

The defined CR0 bits are described below.

PG (Paging Enable, bit 31)

The PG bit is set to enable the on-chip paging unit. It is reset to disable the on-chip paging unit.

R (Reserved, bit 4)

This bit is Reserved for Future Use. When loading CR0 care should be taken to not alter the value of this bit.

TS (Task Switched, bit 3)

TS is automatically set whenever a task switch operation is performed. If TS is set, a coprocessor ESCape op-code will cause a Coprocessor Not Available trap (Exception 7). The trap handler typically saves a 387DX math coprocessor context belonging to a previous task, loads a 387DX math coprocessor state belonging to the current task, and clears the TS bit before returning to the faulting coprocessor op-code.

EM (Emulate Coprocessor, bit 2)

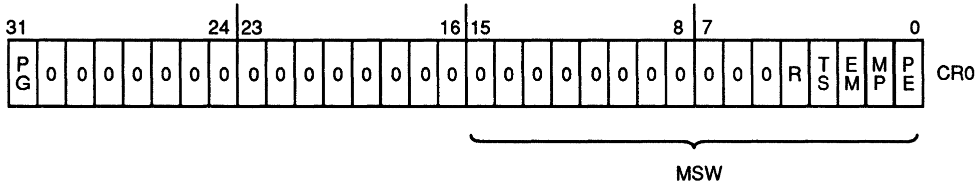
The Emulate coprocessor bit is set to cause all coprocessor op-codes to generate a Coprocessor Not Available fault (Exception7). It is reset to allow coprocessor op-codes to be executed on an actual 387DX math coprocessor (this is the default case after reset). Note that the WAIT op-code is not affected by the EM bit setting.

MP (Monitor Coprocessor, bit 1)

The MP bit is used in conjunction with the TS bit to determine if the WAIT op-code will generate a Coprocessor Not Available fault (Exception 7) when TS = 1. When both MP = 1 and TS = 1, the WAIT op-code generates a trap. Otherwise, the WAIT op-code does not generate a trap. Note that TS is automatically set whenever a task switch operation is performed.

PE (Protection Enable, bit 0)

The PE bit is set to enable the Protected Mode. If PE is reset, the processor operates again in Real Mode. PE may be set by loading MSW or CR0. PE can be reset only by a load into CR0. Resetting the PE bit is typically part of a longer instruction sequence needed for proper transition from Protected Mode to Real Mode. Note that for strict 80286 compatibility, PE cannot be reset by the LMSW instruction.



Note: 0 indicates "Reserved for Future Use." Do not define; see Section Compatibility.

Figure 5. Control Register 0

15021B–008

CR1: Reserved

CR1 is reserved for future processors.

CR2: Page Fault Linear Address

CR2, shown in Figure 6, holds the 32-bit linear address that caused the last page fault detected. The error code pushed onto the page fault handler's stack when it is invoked provides additional status information on this page fault.

CR3: Page Directory Base Address

CR3, shown in Figure 6, contains the physical base address of the page directory table. The Am386DXL microprocessor page directory table is always page-aligned (4 Kb-aligned). Therefore the lowest 12 bits of CR3 are ignored when written and they store as undefined.

A task switch through a TSS that changes the value in CR3, or an explicit load into CR3 with any value, will invalidate all cached page table entries in the paging unit cache. Note that if the value in CR3 does not change during the task switch, the cached page table entries are not flushed.

System Address Registers

Four special registers are defined to reference the tables or segments supported by the 80286 CPU and Am386DXL microprocessor protection model. These tables or segments are:

GDT (Global Descriptor Table);
IDT (Interrupt Descriptor Table);
LDT (Local Descriptor Table);
TSS (Task State Segment).

The addresses of these tables and segments are stored in special registers, the System Address and System Segment Registers illustrated in Figure 7. These registers are named GDTR, IDTR, LDTR, and TR, respectively. The Protected Mode Architecture section describes the use of these registers.

GDTR and IDTR

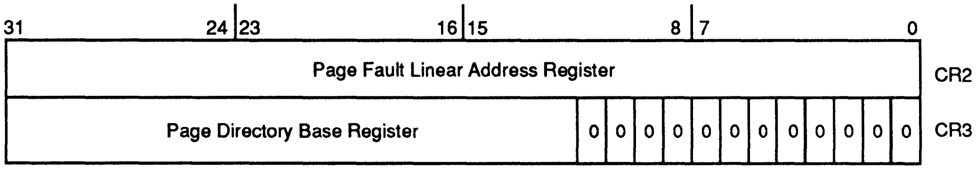
These registers hold the 32-bit linear base address and 16-bit limit of the GDT and IDT, respectively.

The GDT and IDT segments, since they are global to all tasks in the system, are defined by 32-bit linear addresses (subject to page translation if paging is enabled) and 16-bit limit values.

LDTR and TR

These registers hold the 16-bit selector for the LDT descriptor and the TSS descriptor, respectively.

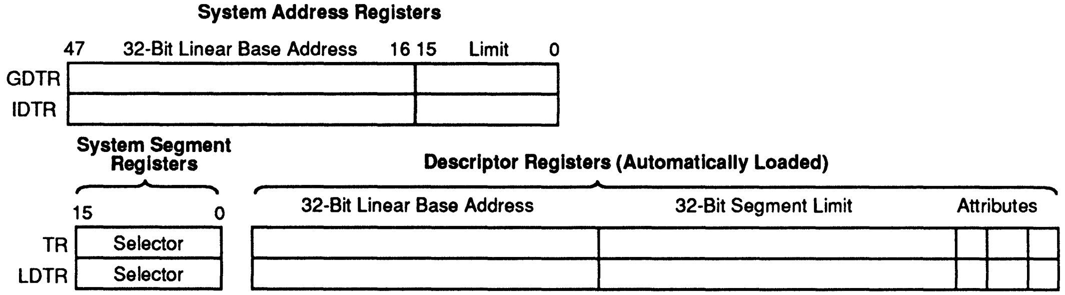
The LDT and TSS segments, since they are task-specific segments, are defined by selector values stored in the system segment registers. Note that a segment descriptor register (programmer-invisible) is associated with each system segment register.


31
24 23
16 15
8 7
0
Page Fault Linear Address Register
CR2
Page Directory Base Register
0 0 0 0 0 0 0 0 0 0 0 0
CR3

Note: 0 indicates "Reserved for Future Use." Do not define; see Section Compatibility.

15021B–009

Figure 6. Control Registers 2 and 3



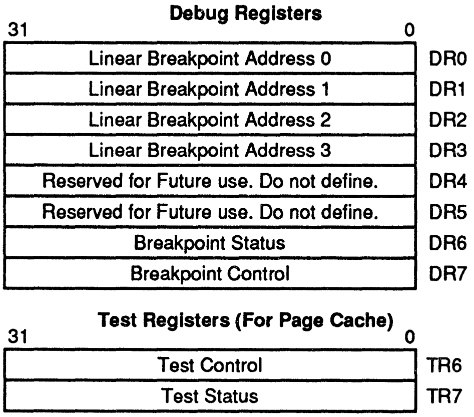
15021B–010

Figure 7. System Address and System Segment Registers

Debug and Test Registers

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. Debug Registers DR3–DR0 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 the Debugging Support section.

Test Registers: Two registers are used to control the testing of the RAM/CAM (Content Addressable Memories) in the Translation Look-Aside Buffer portion of the Am386DXL microprocessor. TR6 is the command test register, and TR7 is the data register that contains the data of the Translation Look-Aside buffer test. Their use is discussed in the Testability section. Figure 8 shows the Debug and Test registers.


Debug Registers

31 0

Linear Breakpoint Address 0 DR0

Linear Breakpoint Address 1 DR1

Linear Breakpoint Address 2 DR2

Linear Breakpoint Address 3 DR3

Reserved for Future use. Do not define. DR4

Reserved for Future use. Do not define. DR5

Breakpoint Status DR6

Breakpoint Control DR7

Test Registers (For Page Cache)

31 0

Test Control TR6

Test Status TR7

15021B–011

Figure 8. Debug and Test Registers

Register Accessibility

There are a few differences regarding the accessibility of the registers in Real and Protected Mode. Table 1 summarizes these differences. See the Protected Mode Architecture section for further details.

Compatibility

VERY IMPORTANT NOTE:
COMPATIBILITY WITH FUTURE PROCESSORS

In the preceding register descriptions, note certain Am386DXL microprocessor register bits are Reserved for Future Use. 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 state of any undefined bits when testing the values of defined register bits. Mask them out when testing.
2. Do not depend on the state 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 which have been previously stored may be reloaded without masking.

Depending upon the values of undefined register bits will make your software dependent upon the unspecified Am386DXL microprocessor handling of these bits. Depending on undefined values risks making your software incompatible with future processors that define usages for the Am386DXL CPU undefined bits. Avoid any software dependence upon the state of undefined Am386DXL CPU register bits.

Table 1. Register Usage

Register	Use In Real Mode		Use In Protected Mode		Use In Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
General Registers	Yes	Yes	Yes	Yes	Yes	Yes
Segment Registers	Yes	Yes	Yes	Yes	Yes	Yes
Flag Registers	Yes	Yes	Yes	Yes	IOPL	IOPL
Control Registers	Yes	Yes	PL = 0	PL = 0	No	Yes
GDTR	Yes	Yes	PL = 0	Yes	No	Yes
IDTR	Yes	Yes	PL = 0	Yes	No	Yes
LDTR	No	No	PL = 0	Yes	No	No
TR	No	No	PL = 0	Yes	No	No
Debug Control	Yes	Yes	PL = 0	PL = 0	No	No
Test Registers	Yes	Yes	PL = 0	PL = 0	No	No

Notes: PL = 0: The registers can be accessed only when the current privilege level is zero.
IOPL: The PUSHF and POPF instructions are made I/O Privilege Level sensitive in Virtual 8086 Mode.

Instruction Set

Instruction Set Overview

The instruction set is divided into nine categories of operations.

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High Level Language Support
Operating System Support
Processor Control

These Am386DXL microprocessor instructions are listed in Table 2.

All Am386DXL microprocessor instructions operate on either 0, 1, 2, or 3 operands where an operand resides in a register in the instruction itself or in memory. Most zero operand instructions (e.g., CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2-bytes long. Since the Am386DXL device has a 16-byte instruction queue, an average of 5 instructions will be prefetched. The use of two operands permits the following types of common instructions.

Register to Register
Memory to Register
Immediate to Register
Register to Memory
Immediate to Memory

The operands can be either 8-, 16-, or 32-bits long. As a general rule, when executing code written for the Am386DXL microprocessor (32-bit code), operands are 8 or 32 bits; when executing existing 80286 or 8086 code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to 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).

Addressing Modes

Addressing Modes Overview

The Am386DXL microprocessor provides a total of 11 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high-level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

Register and Immediate Modes

Two of the addressing modes provide for instructions that operate on register or immediate operands:

Register Operand Mode: The operand is located in one of the 8-, 16-, or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the op-code.

32-Bit Memory Addressing Modes

The remaining nine modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by using combinations of the following four address elements.

Displacement: An 8- or 32-bit immediate value following the instruction.

Base: The contents of any general-purpose register. The Base registers are generally used by compilers to point to the start of the local variable area.

Index: The contents of any general-purpose register except for ESP. The Index registers are used to access the elements of an array, or a string of characters.

Scale: The index register's value can be multiplied by a scale factor either 1, 2, 4, or 8. Scaled index mode is especially useful for accessing arrays or structures.

Combinations of these 4 components make up the 9 additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions.

The one exception is the simultaneous use of Base and Index components that requires one additional clock.

As shown in Figure 9, the effective address (EA) of an operand is calculated according to the following formula.

EA = Base Reg + (Index Reg · Scaling) + Displacement

Direct Mode: The operand's offset is contained as part of the instruction as an 8-, 16-, or 32-bit displacement.

EXAMPLE: INC Word PTR [500]

Register Indirect Mode: A Base register contains the address of the operand.

EXAMPLE: MOV [ECX], EDX

Based Mode: A Base register's contents is added to a Displacement to form the operands offset.

EXAMPLE: MOV ECX, [EAX + 24]

Table 2a. Data Transfer

General Purpose	
MOV	Move operand
PUSH	Push operand onto stack
POP	Pop operand off stack
PUSHA	Push all registers on stack
POPA	Pop all registers off stack
XCHG	Exchange operand register
XLAT	Translate
Conversion	
MOVZX	Move byte or Word, Dword with zero extension
MOVSX	Move byte or Word, Dword, sign extended
CBW	Convert byte to Word, or Word to Dword
CWD	Convert Word to Dword
CWDE	Convert Word to Dword extended
CDQ	Convert Dword to Qword
Input/Output	
IN	Input operand from I/O space
OUT	Output operand to I/O space
Address Object	
LEA	Load effective address
LDS	Load pointer into D segment register
LES	Load pointer into E segment register
LFS	Load pointer into F segment register
LGS	Load pointer into G segment register
LSS	Load pointer into S (Stack) segment register
Flag Manipulation	
LAHF	Load A register from Flags
SAHF	Store A register in Flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack
PUSHFD	Push EFLAGS onto stack
POPFD	Pop EFLAGS off stack
CLC	Clear Carry Flag
CLD	Clear Direction Flag
CMC	Complement Carry Flag
STC	Set Carry Flag
STD	Set Direction Flag

Table 2b. Arithmetic Instructions

Addition	
ADD	Add operands
ADC	Add with carry
INC	Increment operand by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
Subtraction	
SUB	Subtract operands
SBB	Subtract with borrow
DEC	Decrement operand by 1
NEG	Negate operand
CMP	Compare operands
DAS	Decimal adjust for subtraction
AAS	ASCII adjust for subtraction
Multiplication	
MUL	Multiply Double/Single Precision
IMUL	Integer multiply
AAM	ASCII adjust after multiply
Division	
DIV	Divide unsigned
IDIV	Integer divide
AAD	ASCII adjust before division

Table 2c. String Instructions

MOVS	Move byte or Word, Dword string
INS	Input string from I/O space
OUTS	Output string to I/O space
CMPS	Compare byte or Word, Dword string
SCAS	Scan Byte or Word, Dword string
LODS	Load byte or Word, Dword string
STOS	Store byte or Word, Dword string
REP	Repeat
REPE/ REPZ	Repeat while equal/zero
RENE/ REPNZ	Repeat while not equal/not zero

Table 2d. Logical Instructions

Logicals	
NOT	"NOT" operand
AND	"AND" operands
OR	"Inclusive OR" operands
XOR	"Exclusive OR" operands
TEST	"Test" operands
Shifts	
SHL/SHR	Shift logical left or right
SAL/SAR	Shift arithmetic left or right
SHLD/ SHRD	Double shift left or right
Rotates	
ROL/ROR	Rotate left/right
RCL/RCR	Rotate through carry left/right

Table 2e. Bit Manipulation Instructions

Single Bit Instructions	
BT	Bit Test
BTS	Bit Test and Set
BTR	Bit Test and Reset
BTC	Bit Test and Complement
BSF	Bit Scan Forward
BSR	Bit Scan Reverse

Table 2f. Program Control Instructions

Conditional Transfers	
SETCC	Set byte equal to condition code
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if sign

Table 2f. Program Control Instructions (continued)

Unconditional Transfers	
CALL	Call procedure/task
RET	Return from procedure
JMP	Jump
Iteration Controls	
LOOP	Loop
LOOPE/ LOOPZ	Loop if equal/zero
LOOPNE/ LOOPNZ	Loop if not equal/not zero
JCXZ	JUMP if register CX = 0
Interrupts	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Return from interrupt/task
CLI	Clear interrupt enable
STI	Set interrupt enable

Table 2g. High Level Language Instructions

BOUND	Check array bounds
ENTER	Setup parameter block for entering procedure
LEAVE	Leave procedure

Table 2h. Protection Model

SGDT	Store global descriptor table
SIDT	Store interrupt descriptor table
STR	Store task register
SLDT	Store local descriptor table
LGDT	Load global descriptor table
LIDT	Load interrupt descriptor table
LTR	Load task register
LLDT	Load local descriptor table
ARPL	Adjust requested privilege level
LAR	Load access rights
LSL	Load segment limit
VERR/ VERW	Verify segment for reading or writing
LMSW	Load machine status word (lower 16 bits of CR0)
SMSW	Store machine status word

Table 2i. Processor Control Instructions

HLT	Halt
WAIT	Wait until $\overline{\text{BUSY}}$ negated
ESC	Escape
LOCK	Lock Bus

Index Mode: An Index register's contents is added to a Displacement to form the operands offset.

EXAMPLE: ADD EAX, TABLE [ESI]

Scaled Index Mode: An Index register's contents is multiplied by a scaling factor that is added to a Displacement to form the operands offset.

EXAMPLE: IMUL EBX, TABLE [ESI • 4], 7

Based Index Mode: The contents of a Base register is added to the contents of an Index register to form the effective address of an operand.

EXAMPLE: MOV EAX, [ESI] [EBX]

Based Scaled Index Mode: The contents of an Index register is multiplied by a Scaling factor and the result is added to the contents of a Base register to obtain the operands offset.

EXAMPLE: MOV ECX, [EDX • 8] [EAX]

Based Index Mode with Displacement: The contents of an Index Register and a Base register's contents and a Displacement are all summed together to form the operand offset.

EXAMPLE: ADD EDX, [ESI] [EBP + 00FFFFF0H]

Based Scaled Index Mode with Displacement: The contents of an Index register are multiplied by a Scaling factor, the result is added to the contents of a Base register and a Displacement to form the operand's offset.

EXAMPLE: MOV EAX, LOCALTABLE[EDI • 4] [EBP + 80]

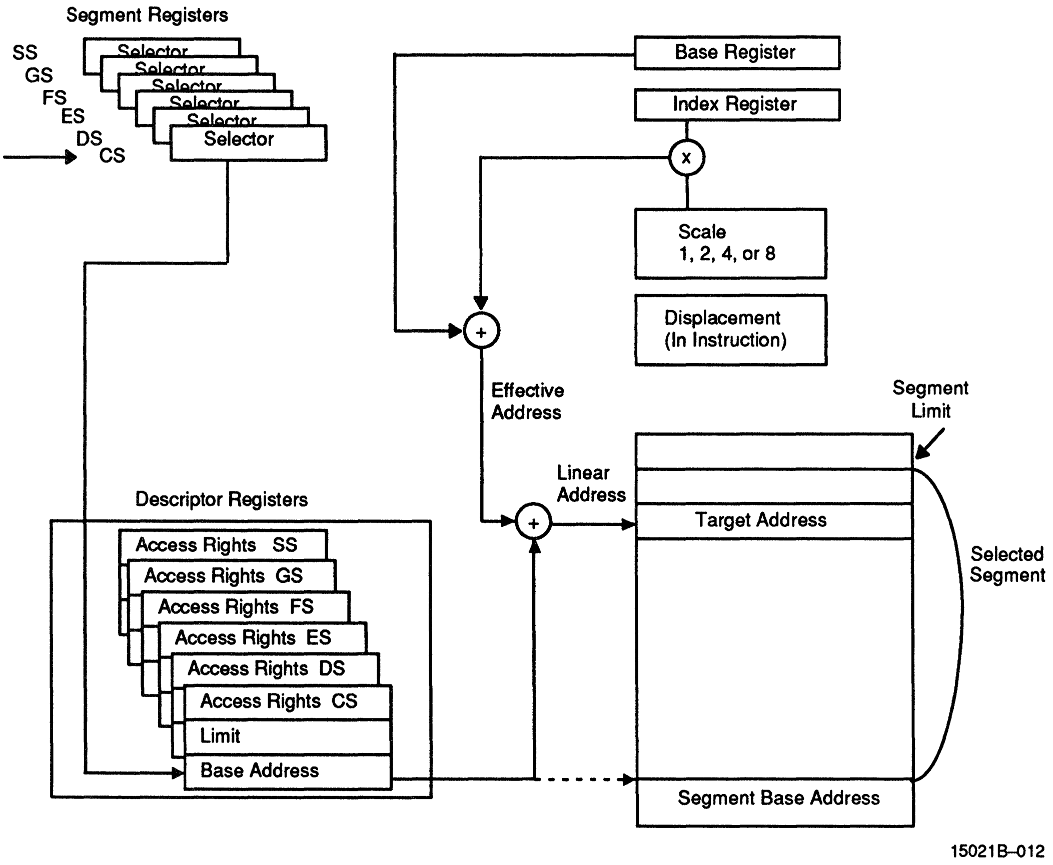


Figure 9. Addressing Mode Calculations

Differences Between 16- and 32-Bit Addresses

In order to provide software compatibility with the 80286 and the 8086, the Am386DXL microprocessor can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in the CS segment descriptor. If the D bit is 0 then all operand lengths and effective addresses are assumed to be 16 bits long. If the D bit is 1 then the default length for operands and addresses is 32 bits. In Real Mode, the default size for operands and addresses is 16 bits.

Regardless of the default precision of the operands or addresses, the Am386DXL microprocessor is able to execute either 16- or 32-bit instructions. This is specified via the use of override prefixes. Two prefixes, the Operand Size Prefix and the Address Length Prefix, override the value of the D bit on an individual instruction basis.

Example: The processor is executing in Real Mode and the programmer needs to access the EAX registers. The assembler code for this might be MOV EAX, 32-bit MEMORYOP. An assembler automatically determines that an Operand Size Prefix is needed and generates it.

Example: The D bit is 0, and the programmer wishes to use Scaled Index addressing mode to access an array. The Address Length Prefix allows the use of MOV DX, TABLE[ESI • 2]. The assembler uses an Address Length Prefix, since with D = 0, the default addressing mode is 16 bits.

Example: The D bit is 1, and the program wants to store a 16-bit quantity. The Operand Length Prefix is used to specify only a 16-bit value: MOV MEM16, DX.

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 Kb to be accessed in Real Mode. A memory address that exceeds FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional Am386DXL microprocessor addressing modes.

When executing 32-bit code, the Am386DXL microprocessor uses either 8- or 32-bit displacements, and any register can be used as base or index registers. When executing 16-bit code, the displacements are either 8 or 16 bits, and the base and index register conform to the 80286 model. Table 3 illustrates the differences.

Data Types

The Am386DXL microprocessor supports all of the data types commonly used in high-level languages.

Bit: A single bit quantity.

Bit Field: A group of up to 32 contiguous bits that spans a maximum of four bytes.

Bit String: A set of contiguous bits on the Am386DXL microprocessor bit strings can be up to 4 Gb long.

Byte: A signed 8-bit quantity.

Unsigned Byte: An unsigned 8-bit quantity.

Integer (Word): A signed 16-bit quantity.

Long Integer (Double Word): A signed 32-bit quantity. All operations assume a 2's complement representation.

Unsigned Integer (Word): An unsigned 16-bit quantity.

Unsigned Long Integer (Double Word): An unsigned 32-bit quantity.

Signed Quad Word: A signed 64-bit quantity.

Unsigned Quad Word: An unsigned 64-bit quantity.

Offset: A 16- or 32-bit offset only quantity that indirectly references another memory location.

Pointer: A full pointer which consists of a 16-bit segment selector and either a 16- or 32-bit offset.

Char: A byte representation of an ASCII alphanumeric or control character.

String: A contiguous sequence of bytes, words or Dword. A string may contain between 1 byte and 4 Gb.

BCD: A byte (unpacked) representation of decimal digits 0–9.

Packed BCD: A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

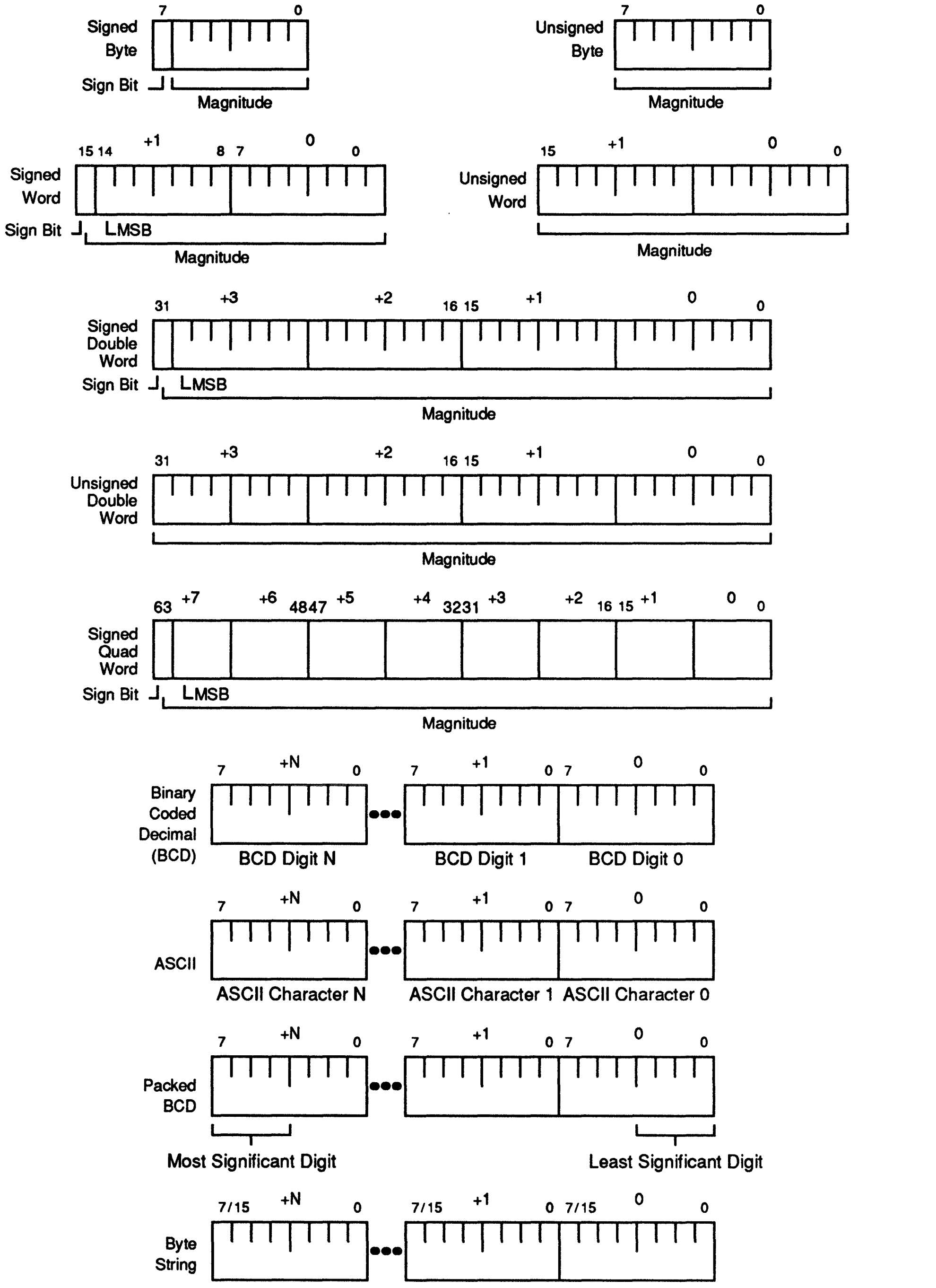
When the Am386DXL microprocessor is coupled with a 387DX math coprocessor then the following common Floating Point types are supported.

Floating Point: A signed 32-, 64-, or 80-bit real number representation. Floating point numbers are supported by a 387DX compatible math coprocessor.

Figure 10 illustrates the data types supported by the Am386DXL microprocessor and a 387DX compatible math coprocessor.

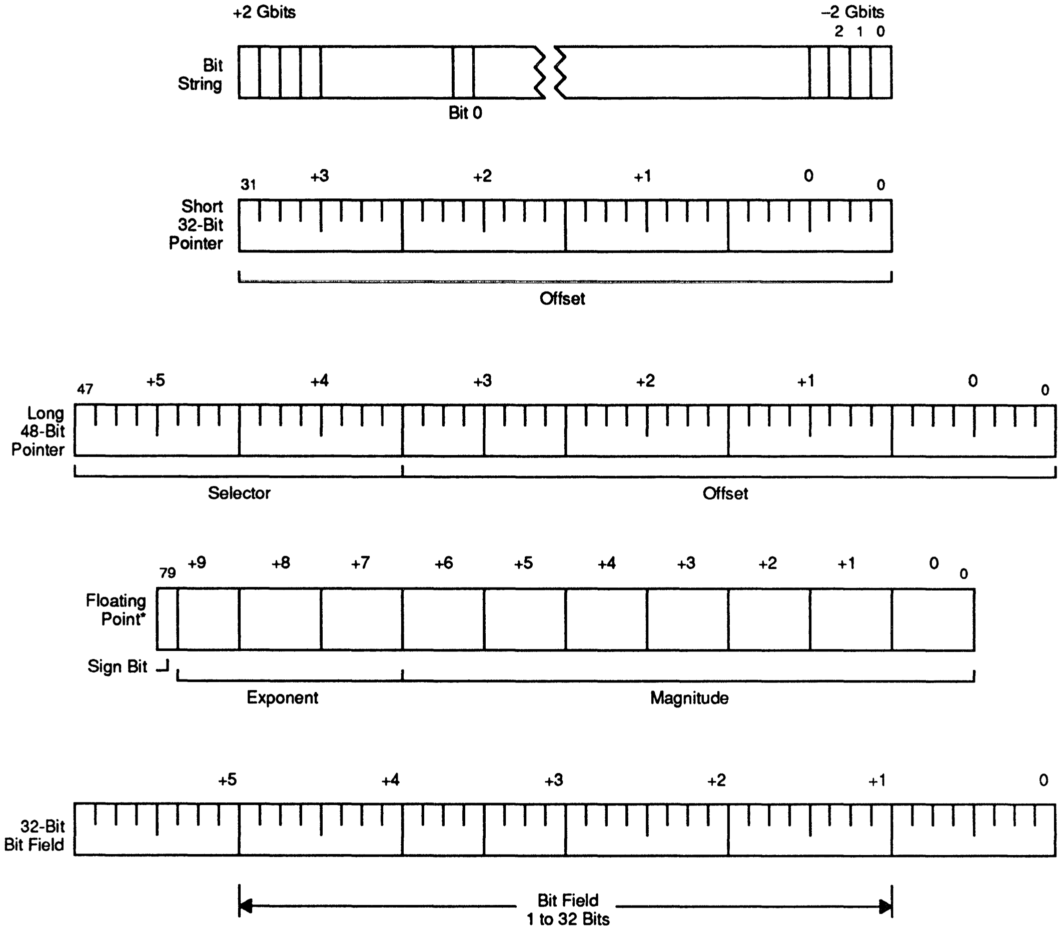
Table 3. Base and Index Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
Base Register	BX, BP	Any 32-bit GP Register
Index Register	SI, DI	Any 32-bit GP Register Except ESP
Scale Factor	None	1, 2, 4, 8
Displacement	0, 8, 16 bits	0, 8, 32 bits



15021B–013

Figure 10. Supported Data Types



*Supported by 387DX-compatible math coprocessor.

15021B–013

Figure 10. Supported Data Types (continued)

Memory Organization

Introduction

Memory on the Am386DXL microprocessor is divided up into 8-bit quantities (Bytes), 16-bit quantities (Words), and 32-bit quantities (Dword). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address, the high-order byte at the highest 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 Am386DXL microprocessor supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4-Kb pages. Finally, both segmentation and paging can be combined, gaining the advantages of both systems. The Am386DXL microprocessor supports both pages and segments in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and as such is a tool for the application programmer, while pages are useful for the system programmer for managing the physical memory of a system.

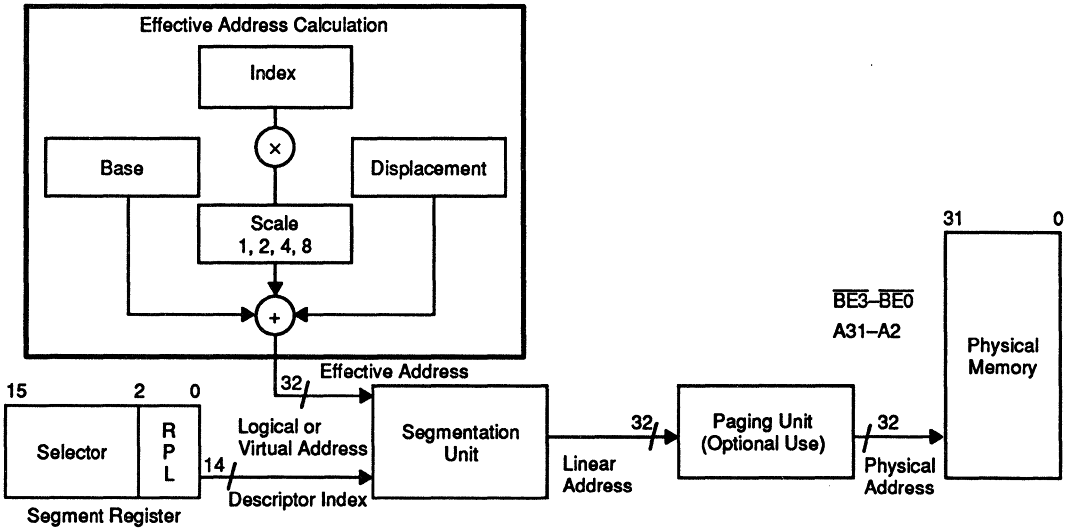
Address Spaces

The Am386DXL microprocessor has three distinct address spaces: logical, linear, and physical. A logical address (also known as a virtual address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (Base, Index, Displacement) discussed in Section Memory Address Modes into an effective address. Since each task on Am386DXL CPU has a maximum of 16K (2^{14}-1) selectors, and offsets can be 4 Gb (2^{32} bits), this gives a total of 2^{46} bits or 64 tb 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 11 shows the relationship between the various address spaces.



15021B–014

Figure 11. Address Translation

Segment Register Usage

The main data structure used to organize memory is the segment. On the Am386DXL microprocessor, segments are variable sized blocks of linear addresses that 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 Gb (2^{32} bytes).

In order to provide compact instruction encoding and increase processor performance, instructions do not need to explicitly specify which segment register is used. A default segment register is automatically chosen according to the rules of Table 4 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register; Stack references use the SS register; and Instruction fetches use the CS register. The contents of the Instruction Pointer provides the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 4. The override prefixes also allow the use of the ES, FS, and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero and create a system with a 4-Gb 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 Protected Mode Architecture.

I/O Space

The Am386DXL microprocessor has two distinct physical address spaces: Memory and I/O. Generally, peripherals are placed in I/O space although the Am386DXL CPU also supports memory-mapped peripherals. The I/O space consists of 64 Kb, it can be divided into 64K 8-bit ports, 32K 16-bit ports, or 16K 32-bit ports, or any combination of ports that add up to less than 64 Kb. The 64K I/O address space refers to physical memory rather than linear address since I/O instructions do not go through the segmentation or paging hardware. The M/$\overline{\text{IO}}$ 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/$\overline{\text{IO}}$ pin to be driven Low.

I/O port addresses 00F8H through 00FFH are reserved.

Interrupts

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

Table 4. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET Instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References with Effective Address Using Base Register of:		
[EAX]	DS	CS, SS, ES, FS, GS
[EBX]	DS	CS, SS, ES, FS, GS
[ECX]	DS	CS, SS, ES, FS, GS
[EDX]	DS	CS, SS, ES, FS, GS
[ESI]	DS	CS, SS, ES, FS, GS
[EDI]	DS	CS, SS, ES, FS, GS
[EBP]	SS	CS, SS, ES, FS, GS
[ESP]	SS	CS, SS, ES, FS, GS

handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT n instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately after the interrupted instruction. The differences between the interrupts are discussed in Sections Maskable Interrupt and Non-Maskable Interrupt.

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 Am386DXL microprocessor 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. Table 5 summarizes the possible interrupts for the Am386DXL microprocessor and shows where the return address points.

The Am386DXL microprocessor has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode (see Section Real Mode Introduction), the vectors are 4 byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8 byte quantities that are put in an Interrupt Descriptor Table (see Section Introduction). Of the 256 possible interrupts, 32 are Reserved for Future Use, the remaining 224 are free to be used by the system designer.

Table 5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	Yes	FAULT
Debug Exception	1	Any instruction	Yes	TRAP*
NMI Interrupt	2	INT 2 or NMI	No	NMI
One Byte Interrupt	3	INT	No	TRAP
Interrupt on Overflow	4	INTO	No	TRAP
Array Bounds Check	5	BOUND	Yes	FAULT
Invalid Op-Code	6	Any illegal instruction	Yes	FAULT
Device Not Available	7	ESC, WAIT	Yes	FAULT
Double Fault	8	Any instruction that can generate an Exception		ABORT
Coprocessor Segment Overrun	9	ESC	No	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	Yes	FAULT
Segment Not Present	11	Segment register instructions	Yes	FAULT
Stack Fault	12	Stack references	Yes	FAULT
General Protection Fault	13	Any memory reference	Yes	FAULT
Page Fault	14	Any memory access or code fetch	Yes	FAULT
Reserved for Future Use	15			
Coprocessor Error	16	ESC, WAIT	Yes	FAULT
Reserved for Future Use	17–31			
Two Byte Interrupt	0–255	INT n	No	TRAP

*Some debug exceptions may report both traps on the previous instruction and faults on the next instruction.

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 Am386DXL microprocessor that identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed.
- Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes a the appropriate instruction.

The 8-bit interrupt vector is supplied to the Am386DXL microprocessor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

Maskable Interrupt

Maskable interrupts are the most common way used by the Am386DXL microprocessor to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled High and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions (REPeat String instructions have an interrupt window between memory moves, which allows interrupts during long string moves). When an interrupt occurs, the processor reads an 8-bit vector supplied by the hardware that identifies the source of the interrupt (one of 224 user defined interrupts). The exact nature of the interrupt sequence is discussed in Section Functional Data.

The IF bit in the EFLAGS register is reset when an interrupt is being serviced. This effectively disables servicing additional interrupts during an interrupt service routine. However, the IF bit 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 bit is restored.

Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. A common example of the use of a non-maskable interrupt (NMI) would be to activate a power failure routine. When the NMI input is pulled High it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for NMI.

While executing the NMI servicing procedure, the Am386DXL microprocessor will not service further NMI requests until an interrupt return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

Software Interrupts

A third type of interrupt/exception for the Am386DXL microprocessor is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the nth vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3 or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt is the single step interrupt. It is discussed in Section Debugging Support.

Interrupt and Exception Priorities

Interrupts are externally-generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are both recognized at the same instruction boundary, the Am386DXL microprocessor invokes the NMI service routine first. If after the NMI service routine has been invoked, maskable interrupts are still enabled, then the Am386DXL CPU will invoke the appropriate interrupt service routine.

Table 6a. Am386DXL Microprocessor Priority for Invoking Service Routines in Case of Simultaneous External Interrupts

1.	NMI
2.	INTR

Exceptions are internally-generated events. Exceptions are detected by the Am386DXL microprocessor if in the course of executing an instruction, the Am386DXL CPU detects a problematic condition. The Am386DXL microprocessor then immediately invokes the appropriate exception service routine. The state of the Am386DXL CPU 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 Am386DXL microprocessor executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 6b. This cycle is repeated as each instruction is executed and occurs in parallel with instruction decoding and execution.

Instruction Restart

The Am386DXL microprocessor fully supports restarting all instructions after faults. If an exception is detected in the instruction to be executed (Exception Categories 4 through 10 in Table 6b), the Am386DXL device invokes the appropriate exception service routine. The Am386DXL microprocessor is in a state that permits restart of the instruction, for all cases but those in Table 6c. Note that all such cases are easily avoided by proper design of the operating system.

Table 6b. Sequence of Exception Checking

Consider the case of the Am386DXL microprocessor having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag or Data Breakpoints set in the Debug Registers).
2. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
3. Check for external NMI and INTR.
4. Check for Segmentation Faults that prevented fetching the entire next instruction (Exceptions 11 and 13).
5. Check for Paging Faults that prevented fetching the entire next instruction (Exception 14).
6. Check for Faults decoding the next instruction (Exception 6 if illegal op-code; Exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only (see Section 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 op-code, check if TS = 1 and MP = 1 (Exception 7 if both are 1).
8. If ESCAPE op-code for numeric coprocessor, check if EM = 1 or TS = 1 (Exception 7 if either are 1).
9. If WAIT op-code or ESCAPE op-code for numeric coprocessor, check $\overline{\text{ERROR}}$ input signal (Exception 16 if $\overline{\text{ERROR}}$ input is asserted).
10. Check in the following order for each memory reference required by the instruction.
    a. Check for Segmentation Faults that prevent transferring the entire memory quantity (Exceptions 11, 12, 13).
    b. Check for Page Faults that prevent transferring the entire memory quantity (Exception 14).

Note that the order stated supports the concept of the paging mechanism being underneath the segmentation mechanism. Therefore, for any given code or data reference in memory, segmentation exceptions are generated before paging exceptions are generated.

Table 6c. Conditions Preventing Instruction Restart

1. An instruction causes a task switch to a task whose Task State Segment (TSS) is partially not present. (An entire not present TSS is restartable.) Partially present TSS's can be avoided either by keeping the TSS's of such tasks present in memory or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4 Kb or less).
2. A coprocessor operand wraps around the top of a 64-Kb segment or a 4-Gb segment and spans three pages; and the page holding the middle portion of the operand is not present. This condition can be avoided by starting at a page boundary any segments containing coprocessor operands if the segments are approximately 64–200 Kb or larger (i.e., large enough for wraparound of the coprocessor operand to possibly occur).

Note that these conditions are avoided by using the operating system designs mentioned in this table.

Double Fault

A Double Fault (Exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12, or 13), but in the process of doing so, detects an exception other than a Page Fault (Exception 14).

A Double Fault (Exception 8) will also be generated when the processor attempts to invoke the Page Fault (Exception 14) service routine, and detects an exception other than a second Page Fault. In any functional system, the entire Page Fault service routine must remain present in memory.

Double Page faults however do not raise the Double Fault exception. If a second Page Fault occurs while the processor is attempting to enter the service routine for the first time, then the processor will invoke the Page Fault (Exception 14) handler a second time rather than the Double Fault (Exception 8) handler. A subsequent fault, though, will lead to shutdown.

When a Double Fault occurs, the Am386DXL microprocessor invokes the exception service routine for Exception 8.

Reset and Initialization

When the processor is initialized or Reset, the registers have the values shown in Table 7. The Am386DXL microprocessor will then start executing instructions near the top of physical memory, at location FFFFFFF0H. When the first Inter-Segment Jump or Call is executed, address lines A31–A20 will drop Low for CS-relative memory cycles, and the Am386DXL microprocessor will only execute instructions in the lower 1 Mb of physical memory. This allows the system designer to use a ROM at the top of physical memory to initialize the system and take care of Resets.

RESET forces the Am386DXL microprocessor to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350- and 450-CLK2 periods after Reset becomes inactive, the Am386DXL device will start executing instructions at the top of physical memory.

Table 7. Register Values after Reset

Flag Word	UUUU0002H	Note 1
Machine Status Word (CR0)	UUUUUUU0H	Note 2
Instruction Pointer	0000FFF0H	
Code Segment	F000H	Note 3
Data Segment	0000H	
Stack Segment	0000H	
Extra Segment (ES)	0000H	
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
DX Register	Component and Stepping ID	Note 5
All Other Registers	Undefined	Note 4

Notes:

1. EFLAGS Register. The upper 14 bits of the EFLAGS register are undefined, VM (Bit 17) and RF (Bit 16) and 0 as are all other defined flag bits.
2. CR0: (Machine Status Word). All of the defined fields in the CR0 are 0 (PG Bit 31, TS Bit 3, EM Bit 2, MP Bit 1, and PE Bit 0).
3. The code Segment Register (CS) will have its Base Address set to FFFF0000H and Limit set to 0FFFFH.
4. All undefined bits are Reserved for Future Use and should not be used.
5. DX register always holds component and stepping identifier (see Section Component and Revision Identifiers). EAX register holds self-test signature if self-test was requested (see Section Self-Test Signature).

Testability

Self-Test

The Am386DXL microprocessor has the capability to perform a self-test. The self-test checks the function of all the Control ROM and most of the non-random logic of the part. Approximately one-half of the Am386DXL microprocessor can be tested during self-test.

Self-Test is initiated on the Am386DXL microprocessor when the RESET pin transitions from High to Low, and the $\overline{\text{BUSY}}$ pin is Low. The self-test takes about 2**19 clocks or approximately 26 ms with a 20-MHz Am386DXL device. At the completion of self-test, the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX register are zero (0). If the results of EAX are not zero then the self-test has detected a flaw in the part.

TLB Testing

The Am386DXL microprocessor provides a mechanism for testing the Translation Look-Aside Buffer (TLB) if desired. This particular mechanism is unique to the Am386DXL CPU and may not be continued in the same way in future processors. When testing the TLB, paging must be turned off (PG = 0 in CR0) to enable the TLB testing hardware and avoid interference with the test data being written to the TLB.

There are two TLB testing operations:

1. Write entries into the TLB; and,
2. Perform TLB lookups. Two test registers, shown in Figure 12, are provided for the purpose of testing. TR6 is the test command register and TR7 is the test data register. The fields within these registers are defined below.

C: This is the command bit. For a write into TR6 to cause an immediate write into the TLB entry, write a 0 to this bit. For a write into TR6 to cause an immediate TLB lookup, write a 1 to this bit.

Linear Address: This is the tag field of the TLB . On a TLB write, a TLB entry is allocated to this linear address and the rest of that TLB entry is set per the value of TR7 and the value just written into TR6. On a TLB lookup, the TLB is interrogated per this value and if one and only one TLB entry matches, the rest of the fields of TR6 and TR7 are set from the matching TLB entry.

Physical Address: This is the data field of the TLB. On a write to the TLB, the TLB entry allocated to the linear address in TR6 is set to this value. On a TLB lookup, the data field (physical address) from the TLB is read out to here.

PL: On a TLB write, PL = 1 causes the REP field of TR7 to select which of four associative blocks of the TLB is to be written, but PL = 0 allows the internal pointer in the paging unit to select which TLB block is written. On a TLB lookup, the PL bit indicated whether the lookup was a hit (PL gets set to 1) or a miss (PL gets reset to 0).

V: The valid bit for this TLB entry. All valid bits can also be cleared by writing to CR3.

D, $\overline{\text{D}}$: The dirty bit for/from the TLB entry.

U, $\overline{\text{U}}$: The user bit for/from the TLB entry.

W, $\overline{\text{W}}$: The writable bit for/from the TLB entry.

For D, U and W, both the attribute and its complement are provided as tag bits to permit the option of a don't care on TLB lookups. The meaning of these pairs of bits is given in the following table.

X	X#	Effect During TLB Lookup	Value of Bit X after TLB Write
0	0	Miss All	Bit X becomes undefined
0	1	Match if X = 0	Bit X becomes 0
1	0	Match if X = 1	Bit X becomes 1
1	1	Match All	Bit X becomes undefined

For writing a TLB entry:

1. Write TR7 for the desired physical address, PL and REP values; and,
2. Write TR6 with the appropriate linear address, etc., (be sure to write C = 0 for write command).

For looking up (reading) a TLB entry:

1. Write TR6 with the appropriate linear address (be sure to write C = 1 for lookup command); and,
2. Read TR7 and TR6. If the PL bit in TR7 indicates a hit, then the other values reveal the TLB contents. If PL indicates a miss, then the other values in TR7 and TR6 are indeterminate.

Debugging Support

The Am386DXL microprocessor provides several features that simplify the debugging process. The three categories of on-chip debugging aids are:

1. The code execution breakpoint op-code (0CCH);
2. The single-step capability provided by the TF bit in the flag register; and,
3. The code and data breakpoint capability provided by the Debug Registers DR3–DR0, DR6, and DR7.

Breakpoint Instruction

A single-byte-op-code breakpoint instruction is available for use by software debuggers. The breakpoint op-code 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 op-code is an alias for the two-byte general software interrupt instruction, INT n, where n = 3. The only difference between INT 3 (0CCh) and INT n is that INT 3 is never IOPL-sensitive but INT n is IOPL-sensitive in Protected Mode and Virtual 8086 Mode.

31 12	11											0	
Linear Address	V	D	$\overline{D}$	U	$\overline{U}$	W	$\overline{W}$	0	0	0	0	C	TR6
Physical Address	0	0	0	0	0	0	0	PL	REP		0	0	TR7

Note: 0 indicates "Reserved for Future Use." Do not define; see Section Compatibility.

15021B–015

Figure 12. Test Registers

Single-Step Trap

If the single-step flag (TF, bit 8) in the EFLAGS 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 1. Precisely, Exception 1 occurs as a trap after the instruction following the instruction that set TF. In typical practice, a debugger sets the TF bit of a flag register image on the debugger's stack. It then typically transfers control to the user program and loads the flag image with a signal instruction, the IRET instruction. The single-step trap occurs after executing one instruction of the user program.

Since the Exception 1 occurs as a trap (that is, it occurs after the instruction has already executed), the CS:EIP pushed onto the debugger's stack points to the next unexecuted instruction of the program being debugged. An Exception 1 handler, merely by ending with an IRET instruction, can therefore efficiently support single-stepping through a user program.

Debug Registers

The Debug Registers are an advanced debugging feature of the Am386DXL microprocessor. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT 3 breakpoint op-code.

The Am386DXL microprocessor contains 6 Debug Registers, providing the ability to specify up to four distinct breakpoint addresses, breakpoint control options, and read breakpoint status. Initially after reset, breakpoints will occur unless the debug registers are programmed. Breakpoints set up in the Debug Registers are auto vectored to Exception 1.

Linear Address Breakpoint Registers (DR3–DR0)

Up to four breakpoint addresses can be specified by writing into Debug Registers DR3–DR0, shown in Figure 13. The breakpoint addresses specified are 32-bit linear addresses. Am386DXL microprocessor hardware continuously compares the linear breakpoint addresses in DR3–DR0 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.

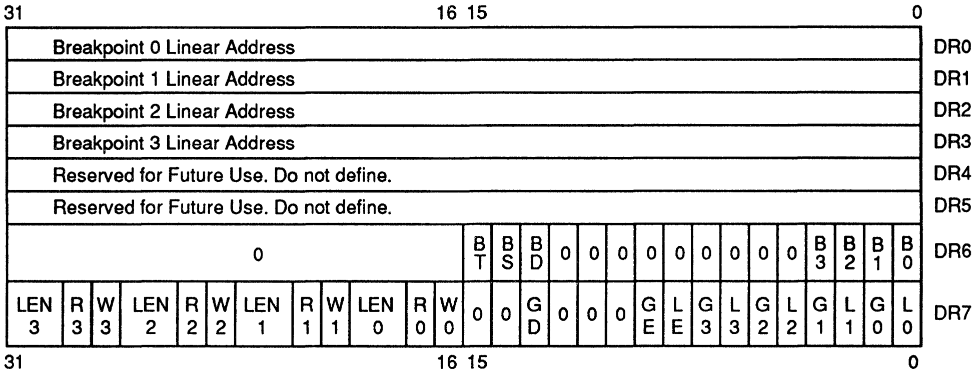
Debug Control Register (DR7)

A Debug Control Register, DR7, shown in Figure 13, allows several debug control functions, such as enabling the breakpoints and setting up other control options for the breakpoints. The fields within the Debug Control Register, DR7, are as follows.

LENi (Breakpoint Length Specification Bits)

A 2-bit LEN field exists for each of the four breakpoints. LEN specifies the length of the associated breakpoint field. The choices for data breakpoints are: 1 byte, 2 bytes, and 4 bytes. Instruction execution breakpoints must have a length of 1 (LENi = 00). Encoding of the LENi field is as follows.

LENi Encoding	Breakpoint Field Width	Usage of Least Significant Bits in Breakpoint Address Register i, (i = 0 – 3)
00	1 byte	All 32-bits used to specify a single-byte breakpoint field.
01	2 bytes	A31–A1 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	A31–A2 used to specify a four-byte, Dword-aligned breakpoint field. A0 and A1 in Breakpoint Address Register are not used.


31
16 15
0
Breakpoint 0 Linear Address
DR0
Breakpoint 1 Linear Address
DR1
Breakpoint 2 Linear Address
DR2
Breakpoint 3 Linear Address
DR3
Reserved for Future Use. Do not define.
DR4
Reserved for Future Use. Do not define.
DR5
0
BT
BS
BD
0 0 0 0 0 0 0 0 0
B3
B2
B1
B0
DR6
LEN 3
R 3
W 3
LEN 2
R 2
W 2
LEN 1
R 1
W 1
LEN 0
R 0
W 0
0 0
GD
0 0 0
GE
LE
G3
L3
G2
L2
G1
L1
G0
L0
DR7
31
16 15
0

Note: 0 indicates "Reserved for Future Use." Do not define; see Section Compatibility.

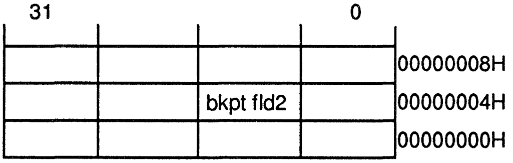
Figure 13. Debug Registers

15021B–016

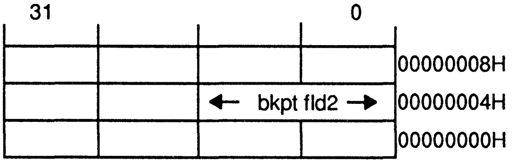
The LENi field controls the size of breakpoint field i by controlling whether all low-order linear address bits in the breakpoint address register are used to detect the breakpoint event. Therefore, all breakpoint fields are aligned; 2-byte breakpoint fields begin on Word boundaries and 4-byte breakpoint fields begin on Dword boundaries.

The following is an example of various size breakpoint fields. Assume the breakpoint linear address in DR2 is 00000005H. In that situation, the following illustration indicates the region of the breakpoint field for lengths of 1, 2, or 4 bytes.

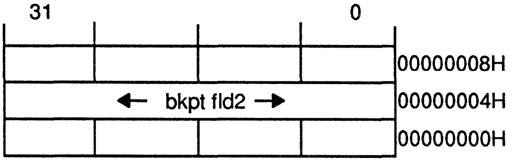
DR2 = 00000005H; LEN2 = 00B



DR2 = 00000005H; LEN2 = 01B



DR2 = 00000005H; LEN2 = 11B



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 that must occur in order to activate the associated breakpoint.

RW Encoding	Usage Causing Breakpoint
00	Instruction execution only
01	Data writes only
10	Undefined—do not use this encoding
11	Data reads and writes only

RW encoding 00 is used to set up an instruction execution breakpoint. RW encodings 01 or 11 are used to set up write-only or read/write data breakpoints.

Note that instruction execution breakpoints are taken as faults (i.e., before the instruction executes), but data breakpoints are taken as traps (i.e., after the data transfer takes place).

Using LENi and RWi to Set Data Breakpoint I

A data breakpoint can be set up by writing the linear address into DRi (i = 0–3). For data breakpoints, RWi can = 01 (write only) or 11 (write/read). LEN can = 00, 01, or 11.

If a data access entirely or partly falls within the data breakpoint field, the data breakpoint condition has occurred, and if the breakpoint is enabled, an Exception 1 trap will occur.

Using LENi and RWi to Set Instruction Execution Breakpoint I

An instruction execution breakpoint can be set up by writing address of the beginning of the instruction (including prefixes if any) into DRi (i = 0–3). RWi must = 00 and LEN must = 00 for instruction execution breakpoints.

If the instruction beginning at the breakpoint address is about to be executed, the instruction execution breakpoint condition has occurred, and if the breakpoint is enabled, an Exception 1 fault will occur before the instruction is executed.

Note that an instruction execution breakpoint address must be equal to the beginning byte address of an instruction (including prefixes) in order for the instruction execution breakpoint to occur.

GD (Global Debug Register Access Detect)

The Debug Registers can only be accessed in Real Mode or at privilege level 0 in Protected Mode. The GD bit, when set, provides extra protection against any Debug Register access even in Real Mode or at privilege level 0 in Protected Mode. This additional protection feature is provided to guarantee that a software debugger (or ICE-386) can have full control over the Debug Register resources when required. The GD bit, when set, causes an Exception 1 fault if an instruction attempts to read or write any Debug Register. The GD bit is then automatically cleared when the Exception 1 handler is invoked, allowing the Exception 1 handler free access to the debug registers.

GE and LE (Exact Data Breakpoint Match, Global and Local)

If either GE or LE is set, any data breakpoint trap will be reported exactly after completion of the instruction that caused the operand transfer. Exact reporting is provided by forcing the Am386DXL microprocessor execution unit to wait for completion of data operand transfers before beginning execution of the next instruction.

If exact data breakpoint match is not selected, data breakpoints may not be reported until several instructions later or may not be reported at all. When enabling a data breakpoint, it is therefore recommended to enable the exact data breakpoint match.

When the Am386DXL microprocessor performs a task switch, the LE bit is cleared. Thus, the LE bit supports fast task switching out of tasks that have enabled the exact data breakpoint match for their task-local breakpoints. The LE bit is cleared by the processor during a task switch to avoid having exact data breakpoint match enabled in the new task. Note that exact data breakpoint match must be re-enabled under software control.

The Am386DXL microprocessor GE bit is unaffected during a task switch. The GE bit supports exact data breakpoint match that is to remain enabled during all tasks executing in the system.

Note that instruction execution breakpoints are always reported exactly, whether or not exact data breakpoint match is selected.

Gi and Li (Breakpoint Enable, Global and Local)

If either Gi or Li is set, then the associated breakpoint (as defined by the linear address in DRi, the length in LENi and the usage criteria in RWi) is enabled. If either Gi or Li is set and the Am386DXL microprocessor detects the ith breakpoint condition, then the Exception 1 handler is invoked.

When the Am386DXL microprocessor performs a task switch to a new Task State Segment (TSS), all Li bits are cleared. Thus, the Li bits support fast task switching out of tasks that use some task-local breakpoint registers. The Li bits are cleared by the processor during a task switch to avoid spurious exceptions in the new task. Note that the breakpoints must be enabled under software control.

All Am386DXL microprocessor Gi bits are unaffected during a task switch. The Gi bits support breakpoints that are active in all tasks executing in the system.

Debug Status Register (DR6)

A Debug Status Register, DR6, shown in Figure 13, allows the Exception 1 handler to easily determine why it was invoked. Note the Exception 1 handler can be invoked as a result of one of several events.

1. DR0 Breakpoint fault/trap.
2. DR1 Breakpoint fault/trap.
3. DR2 Breakpoint fault/trap.
4. DR3 Breakpoint fault/trap.
5. Single-step (TF) trap.
6. Task switch trap.
7. Fault due to attempted debug register access when GD = 1.

The Debug Status Register contains single-bit flags for each of the possible events invoking Exception 1. Note below that some of these events are faults (exception taken before the instruction is executed), while other events are traps (exception taken after the debug events occurred).

The flags in DR6 are set by the hardware but never cleared by hardware. Exception 1 handler software should clear DR6 before returning to the user program to avoid future confusion in identifying the source of Exception 1.

The fields within the Debug Status Register, DR6 are as follows.

Bi (Debug Fault/Trap Due to Breakpoint 0–3)

Four breakpoint indicator flags, B3–B0, correspond one-to-one with the breakpoint registers in DR3–DR0. A flag Bi is set when the condition described by DRi, LENi, and RWi occurs.

If Gi or Li is set, and if the ith breakpoint is detected, the processor will invoke the Exception 1 handler. The exception is handled as a fault if an instruction execution breakpoint occurred or as a trap if a data breakpoint occurred.

Important Note: A flag, Bi, is set whenever the hardware detects a match condition on enabled breakpoint i. Whenever a match is detected on at least one enabled breakpoint i, the hardware immediately sets all Bi bits corresponding to breakpoint conditions matching at that instant, whether enabled or not. Therefore, the Exception 1 handler may see that multiple Bi bits are set, but only set Bi bits corresponding to enabled breakpoints (Li or Gi set) are true indications of why the Exception 1 handler was invoked.

BD (Debug Fault Due to Attempted Register Access When GD Bit Set)

This bit is set if the Exception 1 handler was invoked due to an instruction attempting to read or write to the debug registers when GD bit was set. If such an event occurs, then the GD bit is automatically cleared when the Exception 1 handler is invoked, allowing handler access to the debug registers.

BS (Debug Trap Due to Single-Step)

This bit is set if the Exception 1 handler was invoked due to the TF bit in the flag register being set (for single-stepping). See Section Single-Step Trap.

BT (Debug Trap Due to Task Switch)

This bit is set if the Exception 1 handler was invoked due to a task switch occurring to a task having a Am386DXL microprocessor TSS with the T-bit set. (See Figure 29.) 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.

Use of Resume Flag (RF) In Flag Register

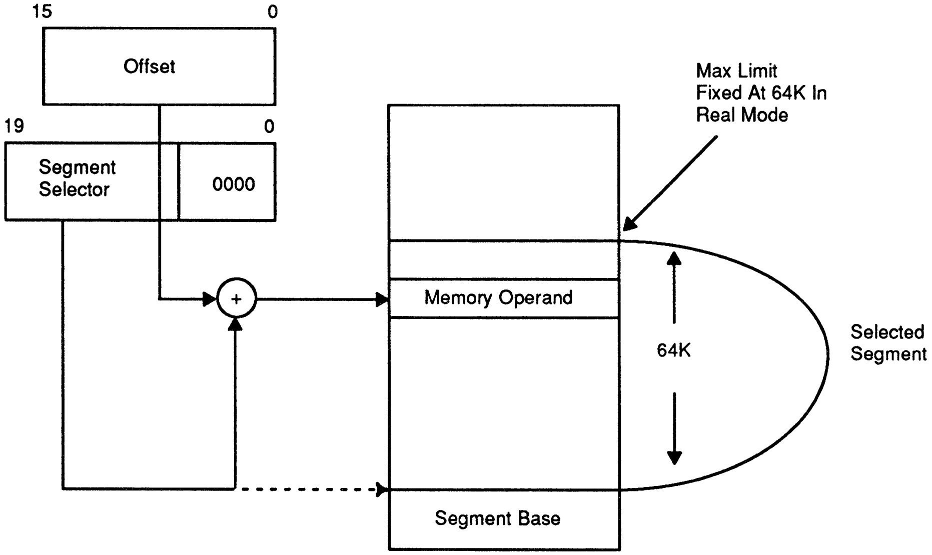
The Resume Flag (RF) in the flag word can suppress an instruction execution breakpoint when the Exception 1 handler returns to a user program at a user address that is also an instruction execution breakpoint. See Section Flags Register.

REAL MODE ARCHITECTURE

Real Mode Introduction

When the processor is reset or powered up, it is initialized in Real Mode. Real Mode has the same base architecture as the 8086, but allows access to the 32-bit register set of the Am386DXL microprocessor. The addressing mechanism, memory size, and interrupt handling are all identical to the Real Mode on the 80286.

All of the Am386DXL microprocessor instructions are available in Real Mode (except those instructions listed in Protection and I/O Permission Bitmap). The default operand size in Real Mode is 16 bits, just like the 8086. In order to use the 32-bit registers and addressing modes, override prefixes must be used. In addition, the segment size on the Am386DXL CPU in Real Mode is 64 Kb so 32-bit effective addresses must have a value less the 0000FFFFH. The primary purpose of Real Mode is to set up the processor for Protected Mode Operation.



15021B–017

Figure 14. Real Address Mode Addressing

LOCK Operation

The LOCK prefix on the Am386DXL microprocessor, even in Real Mode, is more restrictive than on the 80286. This is due to the addition of paging on the Am386DXL CPU in Protected Mode and Virtual 8086 Mode. Paging makes it impossible to guarantee that repeated string instructions can be LOCKed. The Am386DXL CPU 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.

These are the only instruction forms where the LOCK prefix is legal on the Am386DXL microprocessor.

Op-code	Operands (Dest, Source)
BIT TEST and SET/RESET/COMPLEMENT	Mem, Reg/immed
XCHG	Reg, Mem
XCHG	Mem, Reg
ADD, OR, ADC, SBB, AND, SUB, XOR	Mem, Reg/immed
NOT, NEG, INC, DEC	Mem

An Exception 6 will be generated if a LOCK prefix is placed before any instruction form or op-code not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above. For example, even the ADD Reg, Mem is not LOCKable, because the Mem operand is not the destination (and therefore no memory read/modify/operation is being performed).

Since, on the Am386DXL microprocessor, repeated string instructions are not LOCKable, it is not possible to LOCK the bus for a long period of time. Therefore, the LOCK prefix is not IOPL-sensitive on the Am386DXL device. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed above.

Memory Addressing

In Real Mode, the maximum memory size is limited to 1 Mb. Thus, only address lines A19–A2 are active. Exception, the High address lines A31–A20 are High during CS-relative memory cycles until an intersegment jump or call is executed (see Section Reset and Initialization).

Since paging is not allowed in Real Mode, the linear addresses are the same as physical addresses. Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register that is shifted left by 4 bits to an effective address. This addition results in a physical address from 00000000H to 0010FFEFH. This is compatible with 80286 Real Mode. Since segment registers are shifted left by 4 bits, this implies that Real Mode segments always start on 16-byte boundaries.

All segments in Real Mode are exactly 64-Kb long and may be read, written, or executed. The Am386DXL microprocessor will generate an Exception 13 if a data operand or instruction fetch occurs past the end of a segment (i.e., if an operand has an offset greater than FFFFH; for example, a word with a low byte at FFFFH and the high byte at 0000H).

Segments may be overlapped in Real Mode. Thus, if a particular segment does not use all 64 Kb another segment can be overlayed on top of the unused portion of the previous segment. This allows the programmer to minimize the amount of physical memory needed for a program.

Reserved Locations

There are two fixed areas in memory that 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.

Interrupts

Many of the exceptions shown in Table 5 and discussed in Section Interrupts are not applicable to Real Mode operation; in particular, Exceptions 10, 11, and 14 will not happen in Real Mode. Other exceptions have slightly different meanings in Real Mode. Table 8 identifies these exceptions.

Table 8. Other Exceptions In Real Mode

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT Vector is not within table limit.	Before Instruction
CS, DS, ES, FS, GS Segment overrun exception	13	Word memory reference 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

Shutdown and Halt

The HLT instruction stops program execution and prevents the processor from using the local bus until restarted. Either NMI, $\overline{\text{FLT}}$, INTR with interrupts enabled (IF = 1), or RESET will force the Am386DXL microprocessor out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

Shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions:

- An interrupt or an exception occur (Exception 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table (i.e., there is not an interrupt handler for the interrupt);
- A CALL, INT, or PUSH instruction attempts to wrap around the stack segment when SP is not even (e.g., pushing a value on the stack when SP = 0001 resulting in a stack segment greater than FFFFH).

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 0017H) and the stack has enough room to contain the vector and flag information (i.e., SP is greater than 0005H). Otherwise shutdown can only be exited via the RESET input.

PROTECTED MODE ARCHITECTURE

Introduction

The complete capabilities of the Am386DXL microprocessor are unlocked when the processor operates in Protected Virtual Address Mode (Protected Mode). Protected Mode vastly increases the linear address space to 4 Gb (2^{32} bytes) and allows the running of virtual memory programs of almost unlimited size (64 tb or 2^{46} bytes). In addition, Protected Mode allows the Am386DXL CPU to run all of the existing 8086 and 80286 software, while providing a sophisticated memory management and a hardware-assisted protection mechanism. Protected Mode allows the use of additional instructions especially optimized for supporting multitasking operating systems. The base architecture of the Am386DXL CPU remains the same; the registers, instructions, and addressing modes described in the previous sections are retained. The main differences between Protected Mode and Real Mode from a programmer's view is the increased address space and a different addressing mechanism.

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.

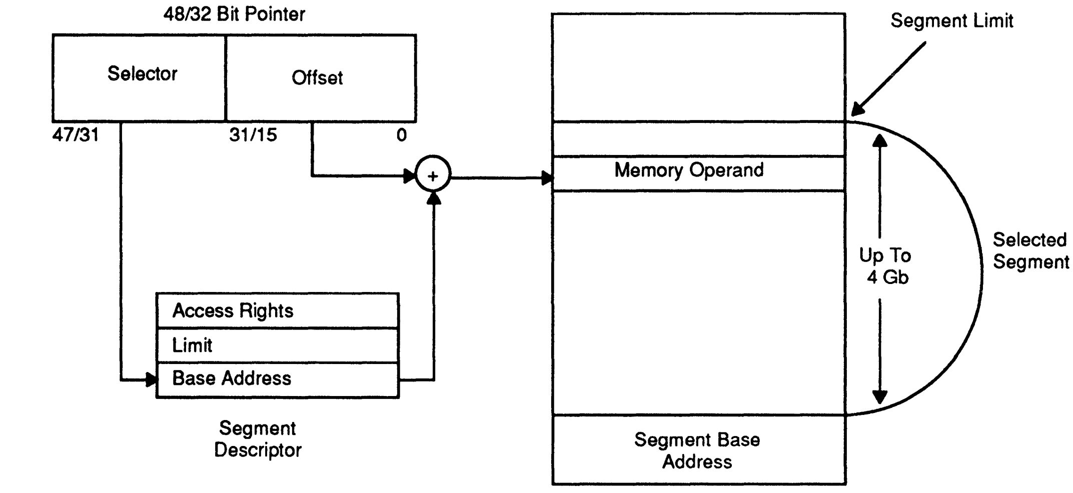
The difference between the two modes lies in calculating the base address. In Protected Mode, the selector is used to specify an index into an operating system defined table (see Figure 15). The table contains the 32-bit base address of a given segment. The physical address is formed by adding the base address obtained from the table to the offset.

Paging provides an additional memory management mechanism that operates only in Protected Mode. Paging provides a means of managing the very large segments of the Am386DXL microprocessor. As such, paging operates beneath segmentation. The paging mechanism translates the protected linear address that comes from the segmentation unit into a physical address. Figure 16 shows the complete Am386DXL device addressing mechanism with paging enabled.

Segmentation

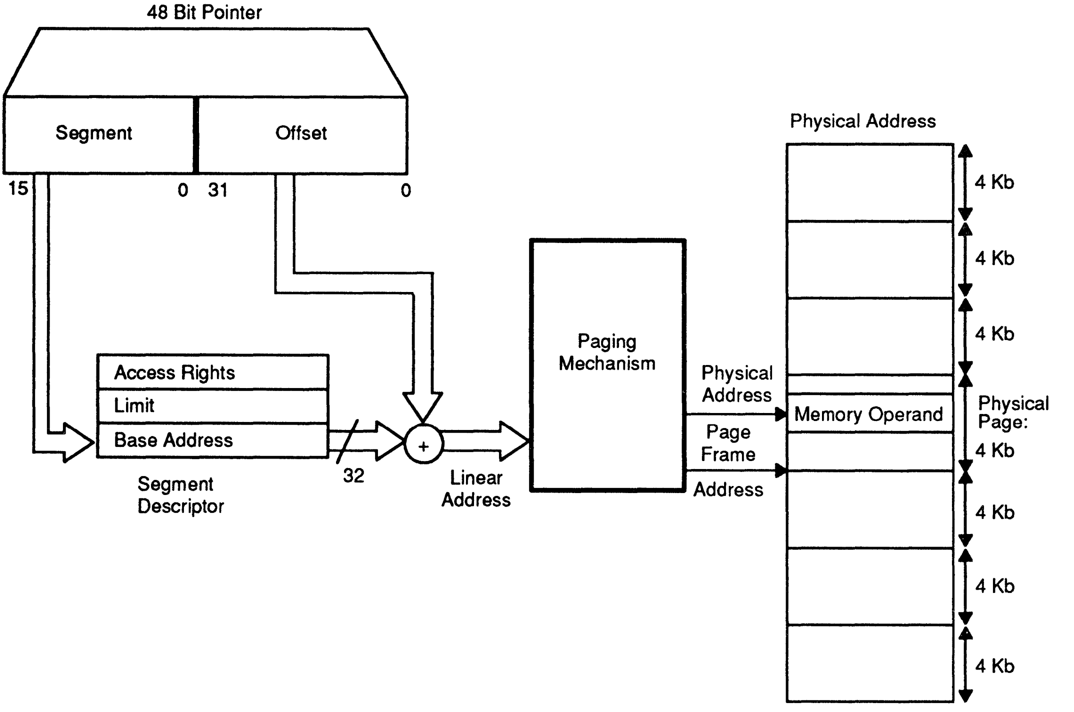
Segmentation Introduction

Segmentation is one method of memory management. Segmentation provides the basis for protection. Segments are used to encapsulate regions of memory that 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.



15021B–018

Figure 15. Protected Mode Addressing



15021B–019

Figure 16. Paging and Segmentation

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. Since small 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.

Descriptor Tables

Descriptor Tables Introduction

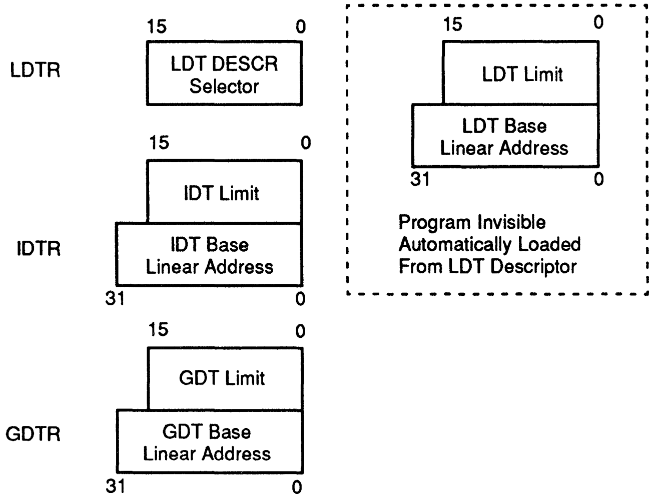
The descriptor tables define all of the segments which are used in an Am386DXL microprocessor system. There are three types of tables on the Am386DXL microprocessor that hold descriptors: the Global Descriptor Table, Local Descriptor Table, and the Interrupt Descriptor Table. All of the tables are variable length memory arrays. They can range in size between 8 bytes and 64 Kb. Each table can hold up to 8192 eight 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 that 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 17). The LGDT, LLDT, and LIDT instructions load the base and limit of the Global, Local, and Interrupt Descriptor Tables, respectively, into the appropriate register. The SGDT, SLDT, and SIDT instructions store the base and limit values. These tables are manipulated by the operating system. Therefore, the load descriptor table instructions are privileged instructions.

Global Descriptor Table

The Global Descriptor Table (GDT) contains descriptors that are possibly available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for descriptors that are used for servicing interrupts (i.e., interrupt and trap descriptors). Every Am386DXL microprocessor 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.



15021B–020

Figure 17. Descriptor Table Registers

Local Descriptor Table

LDTs contain descriptors that 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 that 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 that 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.

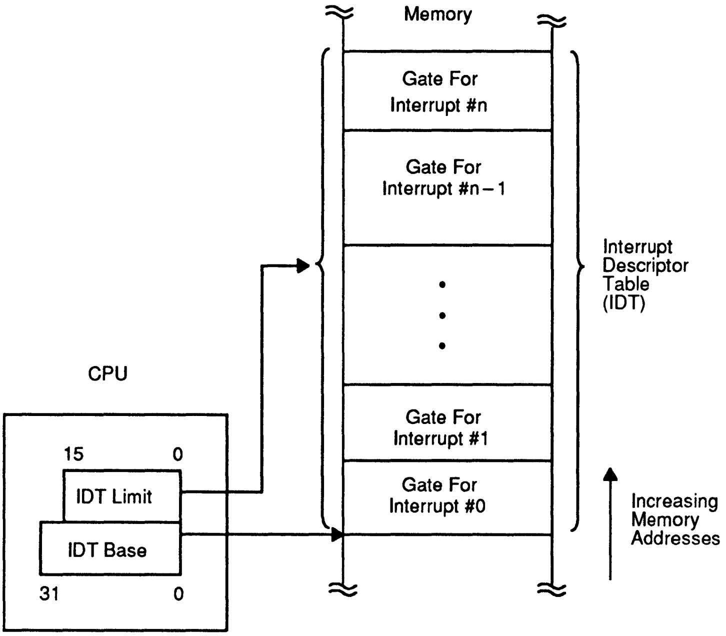
Interrupt Descriptor Table

The third table needed for Am386DXL microprocessor systems is the Interrupt Descriptor Table (see Figure 18). The IDT contains the descriptors that 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, Reserved for Future Use, 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 Interrupts.)

Descriptors

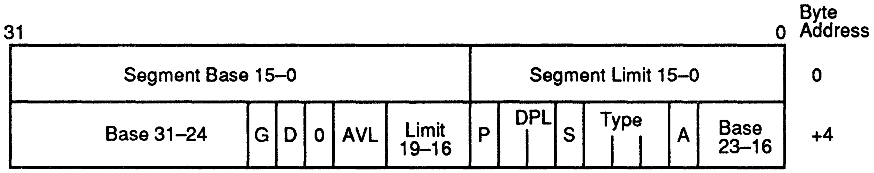
Descriptor Attribute Bits

The object to which the segment selector points is called a descriptor. Descriptors are 8-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. Figure 19 shows the general format of a descriptor. All segments on the Am386DXL microprocessor have three attribute fields in common: the P bit, the DPL bit, and the S bit. The Present P bit is 1 if the segment is loaded in physical memory; if P = 0 then any attempt to access this segment causes a not present exception (Exception 11). The Descriptor Privilege Level DPL is a 2-bit field that specifies the protection levels 0–3 associated with a segment.



15021B–021

Figure 18. Interrupt Descriptor Table Register Use



Base	Base Address of the segment
Limit	The length of the segment
P	Present Bit: 1 = Present, 0 = Not Present
DPL	Descriptor Privilege Levels 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., 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).

15021B–022

Figure 19. General Format of Segment Descriptors

The Am386DXL microprocessor has two main categories of segments: system segments and non-system segments (for code and data). The segment S bit in the segment descriptor determines if a given segment is a system segment or a code or data segment. If the S bit is 1, then the segment is either a code or data segment; if it is 0, then the segment is a system segment.

Am386DXL Microprocessor Code and Data Descriptors (S = 1)

Figure 20 shows the general format of a code and data descriptor and Table 9 illustrates how the bits in the Access Rights Byte are interpreted.

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. Am386DXL microprocessor segments can be 1 Mb long with byte granularity (G = 0) or 4 Gb with page granularity (G = 1), (i.e., 2^{20} pages—each page is 4 Kb in length). The granularity is totally unrelated to paging. An Am386DXL CPU 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.

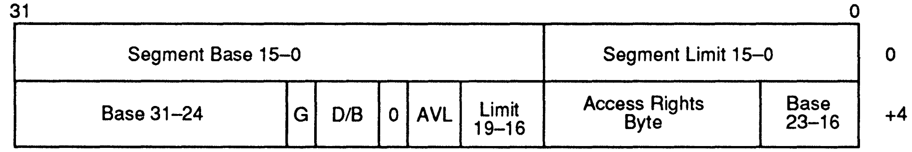
Note: Code segments may be modified via aliases. Aliases are writeable data segments that occupy the same range of linear address space as the code segment.

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 Am386DXL microprocessor assuming the D bit is set 0.

Another attribute of code segments is determined by the conforming C bit. Conforming segments, C = 1, can be executed and shared by programs at different privilege levels (see Section Protection).

Table 9. 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 exists, base and limit are not used.	
6–5	Descriptor Privilege Levels (DPL)		Segment privilege attribute used in privilege tests.	
4	Segment Descriptor (S)	S = 1	Code or Data (includes stacks) segment descriptor.	
		S = 0	System Segment Descriptor or Gate Descriptor.	
3	Executable (E)	E = 0	Descriptor type is data segment.	If Data Segment (S = 1, E = 0)
2	Expansion Direction (ED)	ED = 0	Expand up segment, offsets must be ≤ limit.	
		ED = 1	Expand down segment, offsets must be > limit.	
1	Writeable (W)	W = 0	Data segment may not be written into.	
		W = 1	Data segment may be written into.	
3	Executable (E)	E = 1	Descriptor type is code segment.	If Code Segment (S = 1, E = 1)
2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
1	Readable (R)	R = 0	Code segment may not be read.	
		R = 1	Code segment may be read.	
0	Accessed (A)	A = 0	Segment has not been accessed.	
		A = 1	Segment selector has been loaded into segment register or used by selector test instructions.	



D/B 1 = Default Instructions Attributes are 32 bits
0 = Default Instructions Attributes are 16 bits
AVL Available field for user or OS
G Granularity Bit: 1 = Segment length is page granular, 0 = Segment length is byte granular
0 Bit must be zero (0) for compatibility with future processors

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).

15021B–023

Figure 20. Code and Data Segment Descriptors

Segments identified as data segments (E = 0, S = 1) are used for two types of Am386DXL microprocessor segments: stack and data segments. The expansion direction (ED) bit specifies if a segment expands downward (stack) or upward (data). If a segment is a stack segment, all offsets must be greater than the segment limit. On a data segment all offsets must be less than or equal to the limit. In other words, stack segments start at the base linear address plus the maximum segment limit and grow down to the base linear address plus the limit. On the other hand, data segments start at the base linear address and expand to the base linear address plus limit.

The write (W) bit controls the ability to write into a segment. Data segments are read-only if W = 0. The stack segment must have W = 1.

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

System Descriptor Formats

System segments describe information about operating system tables, tasks, and gates. Figure 21 shows the general format of system segment descriptors, and the various types of system segments. The Am386DXL microprocessor system descriptors contain a 32-bit base linear address and a 20-bit segment limit. 80286 system descriptors have a 24-bit base address and a 16-bit segment limit. 80286 system descriptors are identified by the upper 16 bits being all zeros.

LDT Descriptors (S = 0, Type = 2)

LDT descriptors (S = 0, TYPE = 2) contain information about Local Descriptor Tables. LDTs contain a table of segment descriptors, unique to a particular task. Since the instruction to load the LDTR is only available at privilege level 0, the DPL field is ignored. LDT descriptors are only allowed in the Global Descriptor Table (GDT).

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 TSS. A TSS in turn is a special fixed format segment that contains all the state information for a task and a linkage field to permit nesting tasks. The Type field is used to indicate whether the task is currently BUSY (i.e., on a chain of active tasks) or the TSS is available. The Type field also indicates if the segment contains a 80286 or a Am386DXL microprocessor TSS. The Task Register (TR) contains the selector that points to the current TSS.

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 Protection), task gates are used to perform a task switch, and interrupt and trap gates are used to specify interrupt service routines.

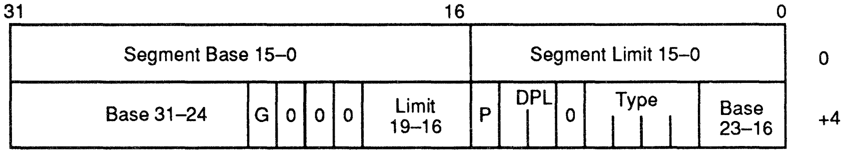
Figure 22 shows the format of the four types of gate descriptors. Call gates are primarily used to transfer program control to a more privileged level. The call gate descriptor consists of three fields: the access byte; a long pointer (selector and offset) that points to the start of a routine; and a word count that specifies how many parameters are to be copied from the caller's stack to the stack of the called routine. The word count field is only used by call gates when there is a change in the privilege level, other types of gates ignore the word count field.

Interrupt and trap gates use the destination selector and destination offset fields of the gate descriptor as a pointer to the start of the interrupt or trap handler routines. The difference between interrupt gates and trap gates is that the interrupt gate disables interrupts (resets the IF bit) while the trap gate does not.

Task gates are used to switch tasks. Task gates may only refer to a task state segment (see Section 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 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 22.

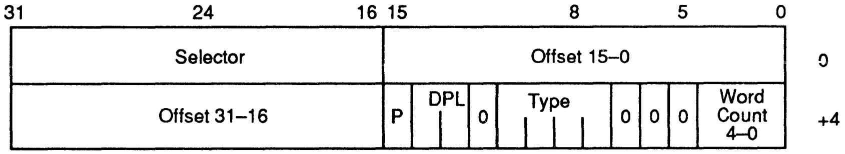


Type	Defines	Type	Defines
0	Invalid	8	Invalid
1	Available 80286 TSS	9	Available Am386DXL CPU TSS
2	LDT	A	Undefined (Reserved)
3	Busy 80286 TSS	B	Busy Am386DXL CPU TSS
4	80286 Call Gate	C	Am386DXL CPU Call Gate
5	Task Gate (for 80286 or Am386DXL CPU Task)	D	Undefined (Reserved)
6	80286 Interrupt Gate	E	Am386DXL CPU Interrupt Gate
7	80286 Trap Gate	F	Am386DXL CPU Trap Gate

Note: In a maximum-size segment (i.e., 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).

15021B–024

Figure 21. System Segments Descriptors



Gate Descriptors Fields

Name	Value	Description
Type	4	80286 Call Gate
	5	Task Gate (for 80286 or Am386DXL CPU Task)
	6	80286 Interrupt Gate
	7	80286 Trap Gate
	C	Am386DXL CPU Call Gate
	E	Am386DXL CPU Interrupt Gate
	F	Am386DXL CPU Trap Gate
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid

DPL—Least privileged level at which a task may access the gate. WORD COUNT 0–31—the number of parameters to copy from caller's stack to the called procedure's stack. The parameters are 32-bit quantities for Am386DXL CPU gates, and 16-bit quantities for 80286 gates.

DESTINATION SELECTOR	16-Bit Selector	Selector to the target code segment or Selector to the target state segment for task gate
DESTINATION OFFSET	Offset 16-bit 80286 32-bit Am386DXL CPU	Entry point within the target code segment

15021B–025

Figure 22. Gate Descriptor Formats

Difference Between Am386DXL Microprocessor and 80286 Descriptors

In order to provide operating system compatibility between the 80286 and Am386DXL microprocessor, the Am386DXL CPU supports all of the 80286 segment descriptors. Figure 23 shows the general format of an 80286 system segment descriptor. The only differences between 80286 and Am386DXL device descriptor formats are that the values of the type fields and the limit and base address fields have been expanded for the Am386DXL device. The 80286 system segment descriptors contained a 24-bit base address and 16-bit limit, while the Am386DXL microprocessor system segment descriptors have a 32-bit base address, a 20-bit limit field, and a granularity bit.

By supporting 80286 system segments, the Am386DXL microprocessor is able to execute 80286 application programs on a Am386DXL CPU operating system. This is possible because the processor automatically understands which descriptors are 80286-style descriptors and which are Am386DXL microprocessor-style descriptors. In particular, if the upper word of a descriptor is zero, then that descriptor is a 80286-style descriptor.

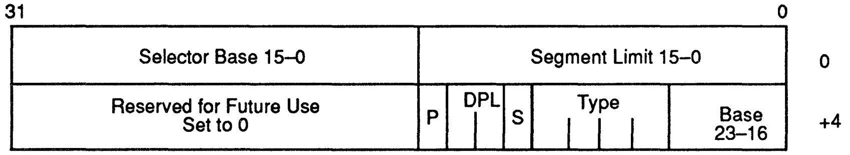
The only other differences between 80286-style descriptors and Am386DXL microprocessor descriptors is the interpretation of the word count field of call gates and the B bit. The word count field specifies the number of 16-bit quantities to copy for 80286 call gates and 32-bit quantities for Am386DXL device 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.

Selector Fields

A selector in Protected Mode has three fields: Local or Global Descriptor Table Indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL) as shown in Figure 24. 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.

Segment Descriptor Cache

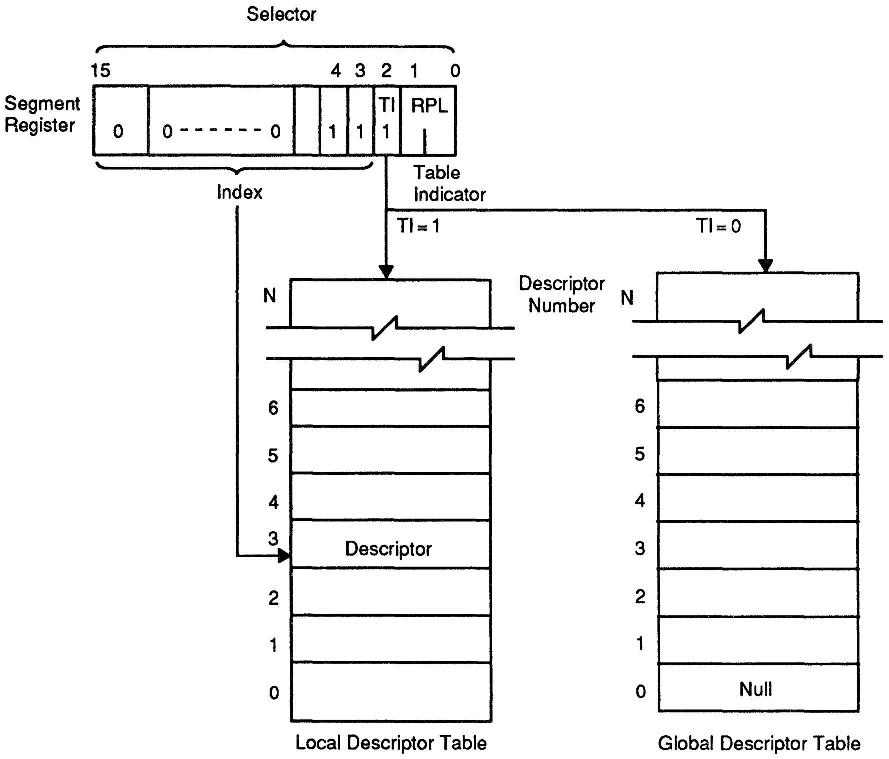
In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with that selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs that modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.



Base	Base Address of the Segment
Limit	The length of the Segment
P	Present Bit: 1 = Present, 0 = Not Present
DPL	Descriptor Privilege Levels 0–3
S	System Descriptor: 0 = System, 1 = User
Type	Type of Segment

15021B–026

Figure 23. 80286 Code and Data Segment Descriptors



15021B–027

Figure 24. Example Descriptor Selection

Segment Descriptor Register Settings

The contents of the segment descriptor cache vary depending on the mode the Am386DXL microprocessor is operating in. When operating in Real Address Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 25.

For compatibility with the 8086 architecture, the base is set to 16 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 op-codes may be executed.

When operating in Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 26. 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.

Segment Descriptor Cache Register Contents

	32-Bit Base (Updated During Selector Load into Segment Register)	32-Bit Limit (Fixed)	Other Attributes (Fixed)									
	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	–	–

Key:
Y = Yes
N = No
0 = Privilege level 0
1 = Privilege level 1
2 = Privilege level 2
3 = Privilege level 3
U = Expand up

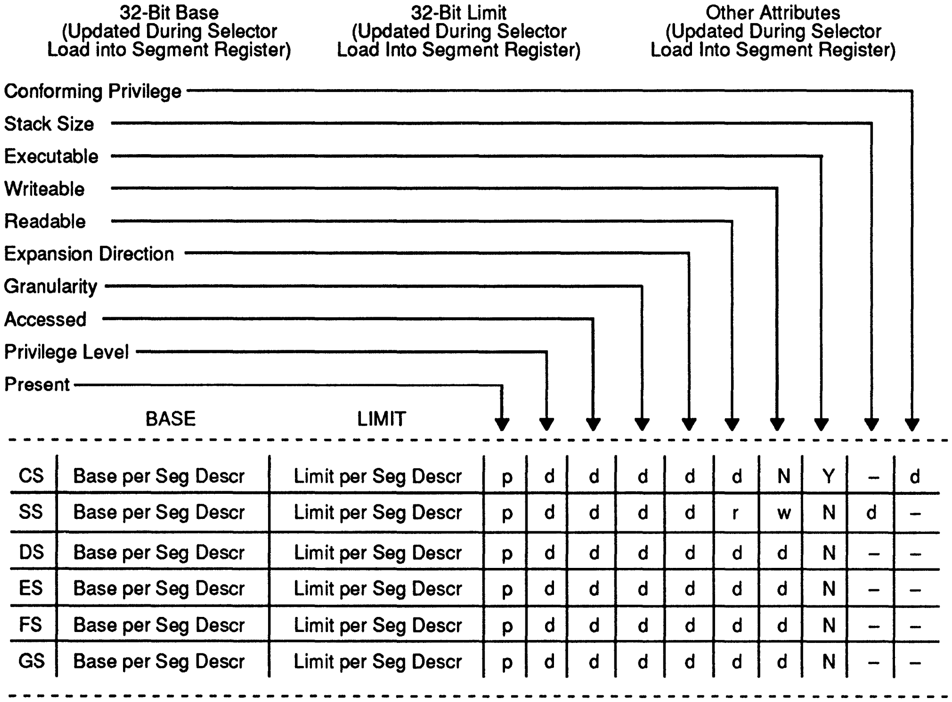
D = Expand down
B = Byte granularity
P = Page granularity
W = Push/pop 16-bit words
F = Push/pop 32-bit Dwords
– = Does not apply to that segment cache register

*Except the 32-bit CS base is initialized to FFFFF000H after reset until first intersegment control transfer (e.g., intersegment CALL, or intersegment JMP, or INT). (See Figure 27 Example.)

15021B–028

Figure 25. Segment Descriptor Caches for Real Address Mode (Segment Limit and Attributes are Fixed)

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)

	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	–	–

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

15021B–029

Figure 26. Segment Descriptor Caches for Protected Mode (Loaded per Descriptor)

When operating in a Virtual 8086 Mode within the Protected Mode, the segment base, limit, and other attributes within the segment cache registers are defined as shown in Figure 27. For compatibility with the 8086 architecture, the base is set to 16 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 (Fixed) | Other Attributes (Fixed)

	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	–	–

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		

15021B–030

Figure 27. Segment Caches for Virtual 8086 Mode within Protected Mode (Segment Limit and Attributes are Fixed)

Protection

Protection Concepts

The Am386DXL microprocessor has four levels of protection that are optimized to support the needs of a multitasking operating system to isolate and protect user programs from each other and the operating system. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. Unlike traditional microprocessor based systems where this protection is achieved only through the use of complex external hardware and software, the Am386DXL CPU provides the protection on a page basis when paging is enabled (see Section Page Level Protection).

The four-level hierarchical privilege system is illustrated in Figure 28. 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 Am386DXL microprocessor paging mechanism. The privilege levels (PL) are numbered 0 through 3. Level 0 is the most privileged or trusted level.

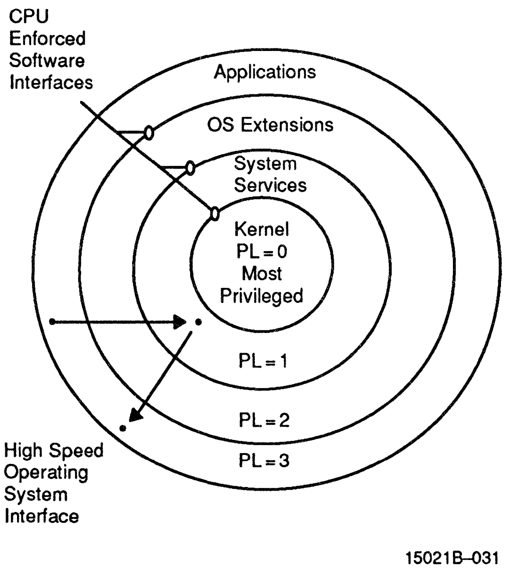

CPU Enforced Software Interfaces
Applications
OS Extensions
System Services
Kernel PL=0 Most Privileged
PL=1
PL=2
PL=3
High Speed Operating System Interface
15021B–031

Figure 28. Four-Level Hierarchical Protection

Rules of Privilege

The Am386DXL microprocessor controls access to both data and procedures between levels of a task, according to the following rules.

- Data stored in a segment with privilege level p can be accessed only by code executing at a privilege level at least as privileged as p.
- A code segment/procedure with privilege level p can only be called by a task executing at the same or a lesser privilege level than p.

Privilege Levels

Task Privilege

At any point in time, a task on the Am386DXL microprocessor always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies the task's privilege level. A task's CPL may only be changed by control transfers through gate descriptors to a code segment with a different privilege level (see Section Privilege Level Transfers). Thus, an application program running at PL=3 may call an operating system routine at PL=1 (via a gate) that would cause the task's CPL to be set to 1 until operating system routine is finished.

Selector Privilege (RPL)

The privilege level of a selector is specified by the RPL field. The RPL is the two least significant bits of the selector. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level for the use of a segment. This level is called the task's effective privilege level (EPL). The EPL is defined as being the least privileged (i.e., numerically larger) level of a task's CPL and a selector's RPL. Thus, if selector's RPL = 0, then the CPL always specifies the privilege level for making an access using the selector. On the other hand if RPL = 3, then a selector can only access segments at level 3 regardless of the task's CPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

I/O Privilege and I/O Permission Bitmap

The I/O privilege level (IOPL, a 2-bit field in the EFLAGS register) defines the least privileged level at which I/O instructions can be unconditionally performed. I/O instructions can be unconditionally performed when CPL≤ IOPL. (The I/O instructions are IN, OUT, INS, OUTS, REP INS, and REP OUTS.) When CPL > IOPL, and the current task is associated with a 286 TSS, attempted I/O instructions cause an Exception 13 fault. When CPL > IOPL, and the current task is associated with a Am386DXL CPU TSS, the I/O Permission Bitmap (part of a Am386DXL microprocessor TSS) is consulted on whether I/O to the port is allowed, or an Exception 13 fault is to be generated instead. For diagrams of the I/O Permission Bitmap, refer to Figures 29a and 29b. For further information on how the I/O Permission Bitmap is used in Protected Mode or in Virtual 8086 Mode, refer to Section 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 Am386DXL microprocessor.)

The IOPL also affects whether the IF bit (interrupts enable flag) can be changed by loading a value into the EFLAGS register. When CPL ≤ IOPL, the IF bit can be changed by loading a new value into the EFLAGS register. When CPL > IOPL, the IF bit cannot be changed by a new value POP'ed into (or otherwise loaded into) the EFLAGS register; the IF bit merely remains unchanged and no exception is generated.

Table 10. 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.

Privilege Validation

The Am386DXL CPU provides several instructions to speed pointer testing and help maintain system integrity by verifying that the selector value refers to an appropriate segment. Table 10 summarizes the selector validation procedures available for the Am386DXL microprocessor.

This pointer verification prevents the common problem of an application at PL = 3 calling an operating-systems routine at PL = 0 and passing the operating-systems routine a bad pointer that corrupts a data structure belonging to the operating system. If the operating-systems 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.

Descriptor Access

There are basically two types of segment accesses: those involving code segments, such as control transfers; and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used, and CPL, RPL, and DPL as described above.

Any time an instruction loads data segment registers (DS, ES, FS, GS) the Am386DXL microprocessor makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segments or readable code segments. The data access rules are specified in Section 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.

Privilege Level Transfers

Intersegment 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 11.

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 that loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules will cause an Exception 13 (e.g., JMP through a call gate or IRET from a normal subroutine call).

In order to provide further system security, all control transfers are also subject to the privilege rules.

The privilege rules require that:

— Privilege level transitions can only occur via gates.

— JMPs can be made to a non-conforming code segment with the same privilege or to a conforming code segment with greater or equal privilege.

— CALLs can be made to a non-conforming code segment with the same privilege or via a gate to a more privileged level.

— Interrupts handled within the task obey the same privilege rules as CALLs.

Table 11. 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 Interrup Gate	IDT
Intersegment to a lower privilege level (change task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET**, Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0 **NT (Nested Task bit of flag register) = 1

— Conforming Code segments are accessible by privilege levels that 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 that references either a task gate or task state segment whose DPL is less privileged or the same privilege as the old task's CPL.

Any control transfer that changes CPL within a task causes a change of stacks as a result of the privilege level change. The initial values of SS:ESP for privilege levels 0, 1, and 2 are retained in the task state segment (see Section Task Switching). During a JMP or CALL control transfer, the new stack pointer is loaded in 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-privilege 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 intersegment RET instruction with a stack adjustment value will correctly restore the previous stack pointer upon return.

Call Gates

Gates provide protected, indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures (such as those that allocate memory or perform I/O).

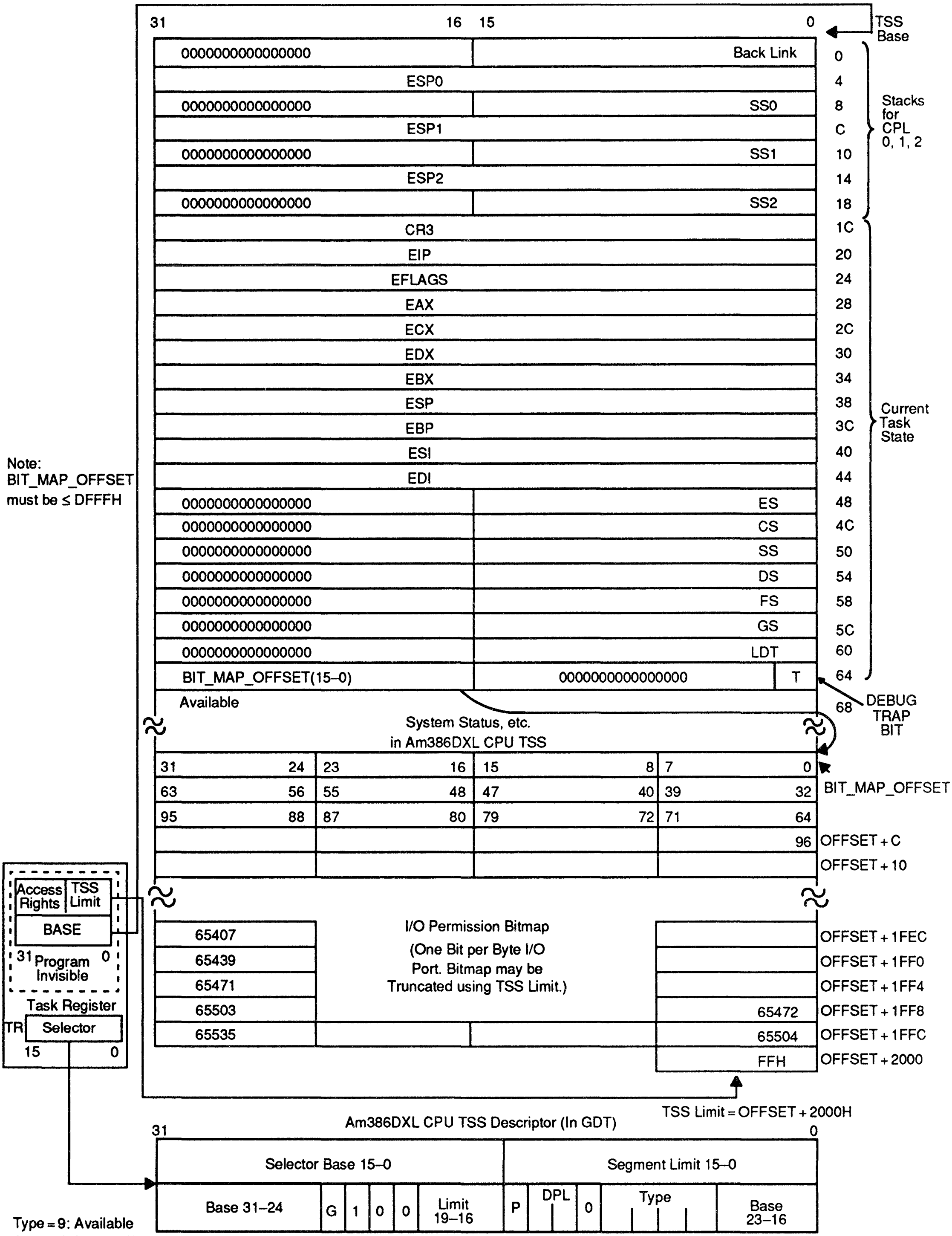
Gate descriptors follow the data access rules of privilege; that is, gates can be accessed by a task if the EPL is equal to or more privileged than the gate descriptor's DPL. Gates follow the control transfer rules of privilege and therefore may only transfer control to a more privileged level.

Call gates are accessed via a CALL instruction and are syntactically identical to calling a normal subroutine. When an interlevel Am386DXL microprocessor call gate is activated, the following actions occur:

1. Load CS:EIP from gate check for validity;
2. SS is pushed zero-extended to 32 bits;
3. ESP is pushed;
4. Copy word count 32-bit parameters from the old stack to the new stack;
5. Push return address on stack.

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

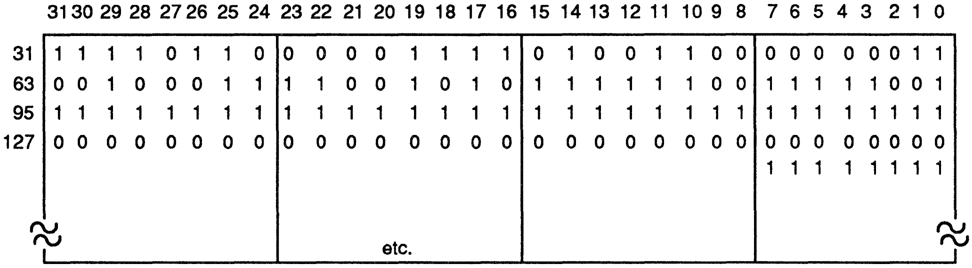
Interrupt gates and Trap gates work in a similar fashion as the call gates, except there is no copying of parameters. The only difference between Trap and Interrupt gates is that control transfers through an Interrupt gate, disable further interrupts (i.e., the IF bit is set to 0), and Trap gates leave the interrupt status unchanged.



Type = 9: Available Am386DXL CPU TSS,
Type = B: Busy Am386DXL CPU TSS

15021B–032a

Figure 29a. 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

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

15021B–032b

Figure 29b. Sample I/O Permission Bit Map

Task Switching

A very important attribute of any multitasking/multi-user operating systems is its ability to rapidly switch between tasks or processes. The Am386DXL microprocessor directly supports this operation by providing a task switch instruction in hardware. The Am386DXL CPU task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task, in about 17 ms. Like transfer of control via gates, the task switch operation is invoked by executing an intersegment JMP or CALL instruction that 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 29a) containing the entire Am386DXL microprocessor execution state while a task gate descriptor contains a TSS selector. The Am386DXL CPU supports both 80286 and Am386DXL CPU style TSSs. Figure 30 shows a 80286 TSS. The limit of a Am386DXL microprocessor TSS must be greater than 0064H (002BH for a 80286 TSS) and can be as large as 4 Gb. 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 belonging to the task.

Each task must have a TSS associated with it. The current TSS is identified by a special register in the Am386DXL microprocessor called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TR are loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The current executing task's state is saved in the TSS and the old task state is restored from its TSS.

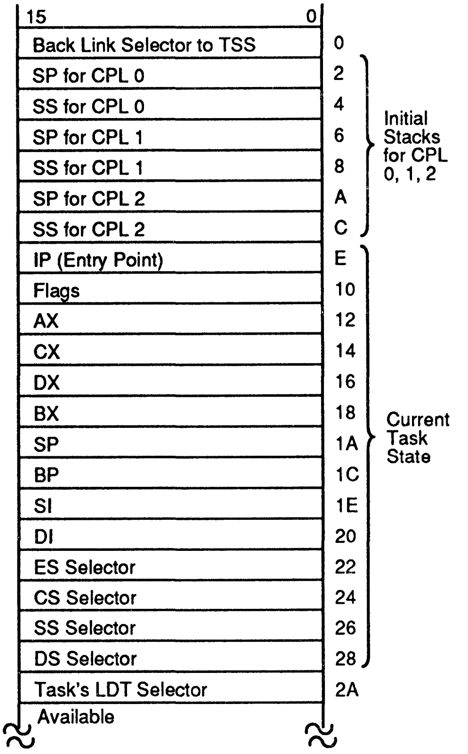
Several bits in the flag register and machine status word (CR0) give information about the state of a task that are useful to the operating system. The Nested Task (NT) (bit 14 in EFLAGS) controls the function of the IRET instruction. If NT = 0, the IRET instruction performs the regular return; when 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 Am386DXL microprocessor Task State Segment is marked busy by changing the descriptor type field from Type 9H to Type BH. An 80286 TSS is marked busy by changing the descriptor type field from Type 1 to Type 3. Use of a selector that references a busy task state segment causes an Exception 13.

The Virtual Mode (VM) bit 17 is used to indicate if a task is a virtual 8086 task. If VM = 1, then the tasks will use the Real Mode addressing mechanism. The virtual 8086 environment is only entered and exited via a task switch (see Section Virtual Mode).

The coprocessor's state is not automatically saved when a task switch occurs, because the incoming task may not use the coprocessor. The Task Switched (TS) Bit (bit 3 in the CR0) helps deal with the coprocessor's state in a multitasking environment. Whenever the Am386DXL microprocessor switches tasks, it sets the TS bit. The Am386DXL CPU detects the first use of a processor extension instruction after a task switch and causes the processor extension not available Exception 7. The exception handler for Exception 7 may then decide whether to save the state of the coprocessor. A processor extension not present Exception 7 will occur when attempting to execute an ESC or WAIT instruction if the Task Switched and Monitor coprocessor extension bits are both set (i.e., TS = 1 and MP = 1).



15 0	
Back Link Selector to TSS	0
SP for CPL 0	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
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	

15021B–033

Figure 30. 80286 TSS

The T bit in the Am386DXL microprocessor TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1, then upon entry to a new task, a debug Exception 1 will be generated.

Initialization and Transition to Protected Mode

Since the Am386DXL microprocessor begins executing in Real Mode immediately after RESET, it is necessary to initialize the system tables and registers with the appropriate values.

The GDT and IDT registers must refer to a valid GDT and IDT. The IDT should be at least 256-bytes long, and GDT must contain descriptors for the initial code and data segments. Figure 31 shows the tables and Figure 32 shows the descriptors needed for a simple Protected Mode Am386DXL microprocessor system. It has a single code and single data/stack segment each 4 Gb long and a single privilege level PL = 0.

The actual method of enabling Protected Mode is to load CR0 with PE bit set, via the MOV CR0, R/M instruction. This puts the Am386DXL microprocessor in Protected Mode.

After enabling Protected Mode, the next instruction should execute an intersegment JMP to load the CS register and flush the instruction decode queue. The final step is to load all of the data segment registers with the initial selector values.

An alternate approach to entering Protected Mode that 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. The TR should be initialized to point to a valid TSS descriptor since a task switch saves the state of the current task in a task state segment.

Paging

Paging Concepts

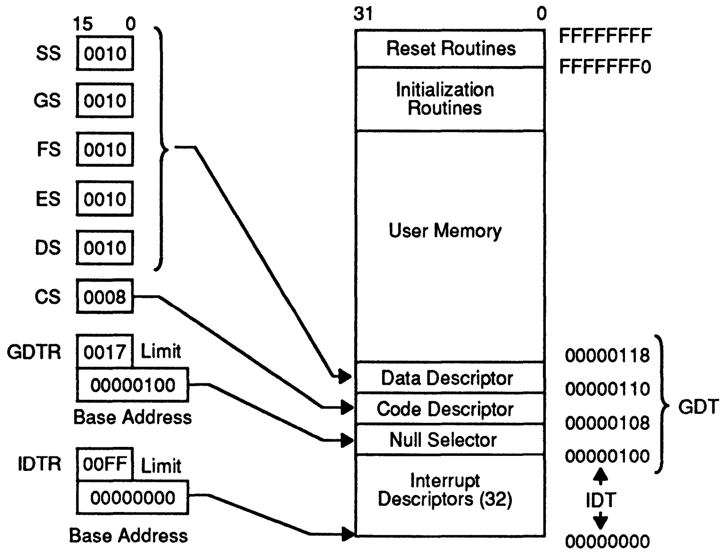
Paging is another type of memory management useful for virtual memory multitasking operating systems. Unlike segmentation that modularizes programs and data into variable length segments, paging divides programs into multiple uniform size pages. Pages bear no direct relation to the logical structure of a program. While segment selectors can be considered the logical name of a program module or data structure, a page most likely corresponds to only a portion of a module or data structure.

By taking advantage of the locality of reference displayed by most programs, only a small number of pages from each active task need be in memory at any one moment.

Paging Organization

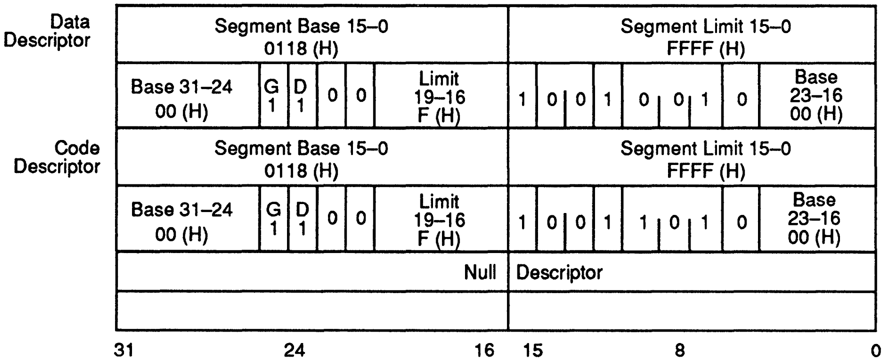
Page Mechanism

The Am386DXL microprocessor uses two levels of tables to translate the linear address (from the segmentation unit) into a physical address. There are three components to the paging mechanism of the Am386DXL CPU: the page directory, the page tables, and the page itself (page frame). All memory-resident elements of the Am386DXL CPU paging mechanism are the same size, namely, 4 Kb. A uniform size for all of the elements simplifies memory allocation and reallocation schemes, since there is no problem with memory fragmentation. Figure 33 shows how the paging mechanism works.



15021B–034

Figure 31. Simple Protected System



15021B–035

Figure 32. GDT Descriptors for Simple System

Page Descriptor Base Register

CR2 is the Page Fault Linear Address register. It holds the 32-bit linear address that 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 Translation Look-Aside Buffer.)

Page Directory

The Page Directory is 4-Kb 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 34. The upper 10 bits of the linear address (A31–A22) are used as an index to select the correct Page Directory entry.

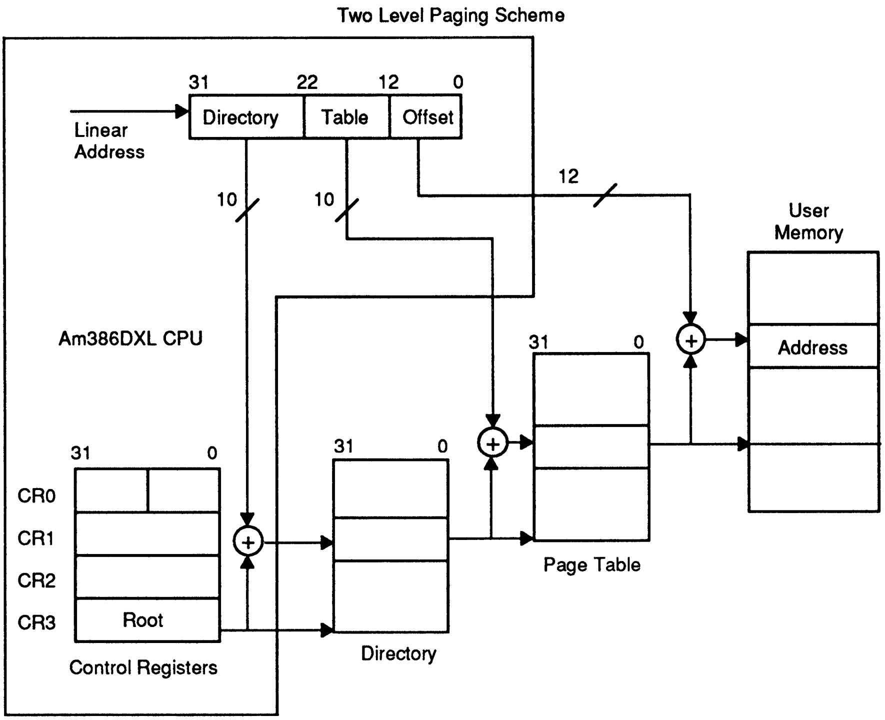


Figure 33. Paging Mechanism

31 12	11 10 9	8	7	6	5	4	3	2	1	0
Page Table Address 31–12	OS Reserved	0	0	D	A	0	0	U/S	R/W	P

Figure 34. Page Directory Entry (Points to Page Table) 15021B–037

31 12	11 10 9	8	7	6	5	4	3	2	1	0
Page Frame Address 31–12	OS Reserved	0	0	D	A	0	0	U/S	R/W	P

Figure 35. Page Table Entry (Points to Page) 15021B–038

Page Tables

Each Page Table is 4 Kb 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 35). Address bits A21–A12 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.

Page Directory/Table Entries

The lower 12 bits of the Page Table entries and Page Directory entries contain statistical information about pages and page tables respectively. The P (Present) bit 0 indicates if a Page Directory or Page Table entry can be used in address translation. If P = 1, the entry can be used for address translation; if P = 0, the entry can not be used for translation. Note that the present bit of the page table entry that points to the page where code is currently being executed should always be set. Code that marks its own page not present should not be written. All of the other bits are available for use by the software. For example the remaining 31 bits could be used to indicate where on the disk the page is stored.

The A (Accessed) bit 5 is set by the Am386DXL microprocessor for both types of entries before a read or write access occurs to an address covered by the entry. The D (Dirty) bit 6 is set to 1 before a write to an address covered by that page table entry occurs. The D bit is undefined for Page Directory entries. When the P, A, and D bits are updated by the Am386DXL CPU, the microprocessor generates a Read-Modify-Write cycle that locks the bus and prevents conflicts with other processors or peripherals. Software that modifies these bits should use the LOCK prefix to ensure the integrity of the page tables in multi-master systems.

The 3 bits marked OS Reserved in Figures 34 and 35 (bits 11–9) are software definable. OSs are free to use these bits for whatever purpose they wish. An example use of the OS Reserved bits would be to store information about page aging. By keeping track of how long a page has been in memory since being accessed, an operating system can implement a page replacement algorithm like Least Recently Used.

The (User/Supervisor) U/S bit 2 and the (Read/Write) R/W bit 1 are used to provide protection attributes for individual pages.

Page Level Protection (R/W, U/S Bits)

The Am386DXL microprocessor provides a set of protection attributes for paging systems. The paging mechanism distinguishes between two levels of protection: user, which corresponds to level 3 of the segmentation based protection, and supervisor, which encompasses all of the other protection levels (0, 1, 2). Programs executing at level 0, 1, or 2 bypass the page protection, although segmentation based protection is still enforced by the hardware.

The U/S and R/W bits are used to provide User/Supervisor and Read/Write protection for individual pages or for all pages covered by a Page Table Directory entry. The U/S and R/W bits in the first level Page Directory Table apply to all pages described by the page table pointed to by that directory entry. The U/S and R/W bits in the second level Page Table entry apply only to the page described by that entry. The U/S and R/W bits for a given page are obtained by taking the most restrictive of the U/S and R/W bits from the Page Directory Table entries and the Page Table entries and using these bits to address the page.

Example: If the U/S and R/W bits for the Page Directory entry were 10 and the U/S and R/W bits for the Page Table entry were 01, the access rights for the page would be 01, the numerically smaller of the two. Table 12 shows the effect of the U/S and R/W bits on accessing memory.

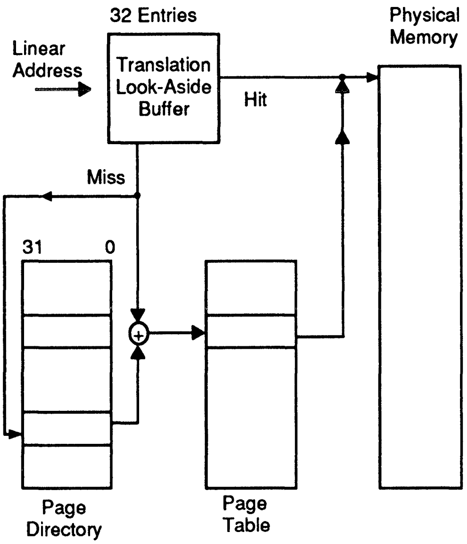
Table 12. Protection Provided by R/W and U/S

U/S	R/W	Permitted Level 3	Permitted Access Levels 0, 1, or 2
0	0	None	Read/Write
0	1	None	Read/Write
1	0	Read-Only	Read/Write
1	1	Read/Write	Read/Write

However, a given segment can be easily made read-only for level 0, 1, or 2 via the use of segmented protection mechanisms (see Section Protection).

Translation Look-Aside Buffer

The Am386DXL microprocessor paging hardware is designed to support demand paged virtual memory systems. However, performance would degrade substantially if the processor was required to access two levels of tables for every memory reference. To solve this problem, the Am386DXL device keeps a cache of the most recently accessed pages, this cache is called the Translation Look-Aside Buffer (TLB). The TLB is a four-way set associative 32-entry page table cache. It automatically keeps the most commonly used Page Table entries in the processor. The 32-entry TLB coupled with a 4K-page size results in coverage of 128 Kb of memory addresses. For many common multitasking systems, the TLB will have a hit rate of about 98%. This means that the processor will only have to access the two-level page structure on 2% of all memory references. Figure 36 illustrates how the TLB complements the Am386DXL microprocessor's paging mechanism.



•98% Hit Rate

15021B–039

Figure 36. Translation Look-Aside Buffer

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 Am386DXL microprocessor will read the appropriate Page Directory entry. If P = 1 on the Page Directory entry indicating that the page table is in memory, then the Am386DXL device 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 Am386DXL device will update the Access and Dirty bits as needed and fetch the operand. The upper 20 bits of the linear address, read from the page table, will be stored in the TLB for future accesses. However, if P = 0 for either the Page Directory entry or the Page Table Entry, then the processor will generate a Page Fault, an Exception 14.

The processor will also generate an Exception 14, Page Fault, if the memory reference violated the page protection attributes (i.e., U/S or R/W; trying to write to a read-only page). CR2 will hold the linear address that caused the page fault. If a second page fault occurs while the processor is attempting to enter the service routine for the first, then the processor will invoke the Page Fault (Exception 14) handler a second time, rather than the Double Fault (Exception 8) handler. Since Exception 14 is classified as a fault, CS:EIP will point to the instruction causing the page fault. The 16-bit error code pushed as part of the page fault handler will contain status bits which indicate the cause of the Page Fault.

The 16-bit error code is used by the operating system to determine how to handle the Page Fault. Figure 37 shows the format of the page-fault error code and the interpretation of the bits.

Note: Even though the bits in the error code (U/S, R/W, and P) have similar names as the bits in the Page Directory/Table entries, the interpretation of the error code bits is different. Figure 38 indicates what type of access caused the Page Fault.

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

15021B–040

Figure 37. Page Fault Error Code Format

U/S: The U/S bit indicates whether the access causing the fault occurred when the processor was executing the User Mode (U/S = 1) or in Supervisor mode (U/S = 0).

R/W: The R/W bit indicates whether the access causing the fault was a Read (R/W = 0) or a Write (R/W = 1).

P: The P bit indicates whether a Page Fault was caused by a not-present page (P = 0) or by a page level protection violation (P = 1).

U: Undefined.

U/S	R/W	Access Type
0	0	Supervisor* Read
0	1	Supervisor Write
1	0	User Read
1	1	User Write

*Descriptor table access will fault with U/S = 0, even if the program is executing at level 3.

15021B–041

Figure 38. Type of Access Causing Page Fault

Operating System Responsibilities

The Am386DXL microprocessor takes care of the page address translation process, relieving the burden from an operating system in a demand-paged system. The operating system is responsible for setting up the initial page tables and handling any page faults. The operating system also is required to invalidate (i.e., flush) the TLB when any changes are made to any of the Page Table entries. The operating system must reload CR3 to cause the TLB to be flushed.

Setting up the tables is simply a matter of loading CR3 with the address of the Page Directory and allocating space for the Page Directory and the Page Tables. The primary responsibility of the operating system is to implement a swapping policy and handle all of the page faults.

A final concern of the operating system is to ensure that the TLB cache matches the information in the paging tables. In particular, any time the operating system sets the P present bit of page table entry to zero, the TLB must be flushed. Operating systems may want to take advantage of the fact that CR3 is stored as part of a TSS to give every task or group of tasks its own set of page tables.

Virtual 8086 Environment

Executing 8086 Programs

The Am386DXL microprocessor allows the execution of 8086 application programs in both Real Mode and in the Virtual 8086 Mode (Virtual Mode). Of the two methods, Virtual 8086 Mode offers the system designer the most flexibility. The Virtual 8086 Mode allows the execution of 8086 applications, while still allowing the system designer to take full advantage of the Am386DXL device protection mechanism. In particular, the Am386DXL CPU allows the simultaneous execution of 8086 operating systems and its applications, and a Am386DXL CPU operating system and both 80286 and Am386DXL microprocessor applications. Thus, in a multi-user Am386DXL CPU computer, one person could be running a 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 39 illustrates this concept.

Virtual 8086 Mode Addressing Mechanism

One of the major differences between Am386DXL microprocessor Real and Protected Modes is how the segment selectors are interpreted. When the processor is executing in Virtual 8086 Mode, the segment registers are used in an identical fashion to Real Mode. The contents of the segment register is shifted left 4 bits and added to the offset to form the segment base linear address.

The Am386DXL microprocessor allows the operating system to specify which programs use the 8086 style address mechanism, and which programs use Protected Mode addressing, on a per task basis. Through the use of paging, the 1-Mb address space of the Virtual Mode task can be mapped to anywhere in the 4-Gb linear address space of the Am386DXL device. Like Real Mode, Virtual Mode effective addresses (i.e., segment offsets) that exceed 64 Kb will cause an Exception 13. However, these restrictions should not prove to be important because most tasks running in Virtual 8086 Mode will simply be existing 8086 application programs.

Paging In Virtual Mode

The paging hardware allows the concurrent running of multiple Virtual Mode tasks and provides protection and operating system isolation. Although it is not strictly necessary to have the paging hardware enabled to run Virtual Mode tasks, it is needed in order to run multiple Virtual Mode tasks or to relocate the address space of a Virtual Mode task to physical address space greater than 1 Mb.

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-Gb physical address space of the Am386DXL microprocessor. In addition, since CR3 (the Page Directory Base Register) is loaded by a task switch, each Virtual Mode task can use a different mapping scheme to map pages to different physical locations. Finally, the paging hardware allows the sharing of the 8086 operating system code between multiple 8086 applications. Figure 39 shows how the Am386DXL device paging hardware enables multiple 8086 programs to run under a virtual memory demand paged system.

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.
```

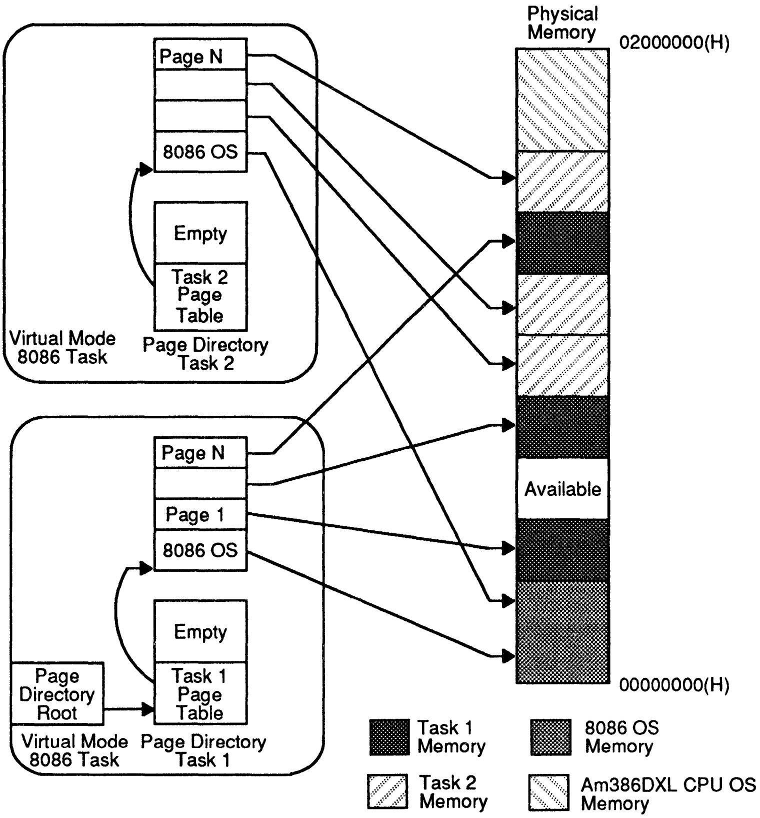


Figure 39. Virtual 8086 Environment Memory Management

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 (op-code 0CCH), INTO, and BOUND instructions are not IOPL-sensitive in Virtual 8086 Mode (they are not 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 Am386DXL CPU TSS. The I/O Permission Bitmap, automatically used by the Am386DXL microprocessor in Virtual 8086 Mode, is illustrated by Figures 29a and 29b.

The I/O Permission Bitmap can be viewed as a 0–64K bit string, that 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 that follows the bit map are all at offset ≤ 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 29a.

Each bit in the I/O Permission Bitmap corresponds to a single byte-side I/O port, as illustrated in Figure 29a. If a bit is 0, I/O to the corresponding byte-wide port can occur without generating an exception. Otherwise the I/O instruction causes an Exception 13 fault. Since every byte-wide I/O port must be protectable, all bits corresponding to a Word-wide or Dword-wide port must be 0 for the Word-wide or Dword-wide I/O to be permitted. If all the referenced bits are 0, the I/O will be allowed. If any referenced bits are 1, the attempted I/O will cause an Exception 13 fault.

Due to the use of a pointer to the base of the I/O Permission Bitmap, the bitmap may be located anywhere within the TSS or may be ignored completely by pointing the Bit_Map_Offset (15–0) beyond the limit of the TSS segment. In the same manner, only a small portion of the 64K I/O space need have an associated map bit by adjusting the TSS limit to truncate the bitmap. This eliminates the commitment of 8K of memory when a complete bitmap is not required, while allowing the fully general case if desired.

Example of Bitmap for I/O Ports 0–255: Setting the TSS limit to {Bit_Map_Offset + 31 +1**} [**see note below] will allow a 32-byte bitmap for the I/O ports 0–255, plus a terminator byte of all 1s [**see note below]. This allows the I/O bitmap to control I/O Permission to I/O ports 0–255 while causing an Exception 13 fault on attempted I/O to any I/O port 256 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 1s. The byte of all 1s must be within the limit of the Am386DXL CPU TSS segment (see Figure 29a).

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 Am386DXL CPU operating system. The Am386DXL microprocessor operating system determines if the interrupt comes from a Protected Mode application or from a Virtual Mode program by examining the VM bit in the EFLAGS image stored on the stack.

When a Virtual Mode program is interrupted and execution passes to the interrupt routine at level 0, the VM bit is cleared. However, the VM bit is still set in the EFLAGS image on the stack.

The Am386DXL microprocessor operating system in turn handles the exception or interrupt and then returns control to the 8086 program. The Am386DXL CPU 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 Am386DXL microprocessor operating system. The Am386DXL CPU operating system could emulate the 8086 operating system's call. Figure 40 shows how the Am386DXL microprocessor operating system could intercept an 8086 operating system's call to Open a File.

The Am386DXL microprocessor operating system can provide a Virtual 8086 Environment that is totally transparent to the application software via intercepting and then emulating 8086 operating system's calls, and intercepting IN and OUT instructions.

Entering and Leaving Virtual 8086 Mode

Virtual 8086 Mode is entered by executing an IRET instruction (at CPL = 0), or Task Switch (at any CPL) to a Am386DXL microprocessor task whose Am386DXL microprocessor TSS has a EFLAGS image containing a 1 in the VM bit position while the processor is executing in Protected Mode. That is, one way to enter Virtual 8086 Mode is to switch to a task with a Am386DXL device TSS that has a 1 in the VM bit in the EFLAGS image. The other way is to execute a 32-bit IRET instruction at privilege level 0, where the stack has a 1 in the VM bit in the EFLAGS image. POPF does not affect the VM bit even if the processor is in Protected Mode or level 0, and so cannot be used to enter Virtual 8086 Mode. PUSHF always pushes a 0 in the VM bit, even if the processor is in Virtual 8086 Mode, so that a program cannot tell if it is executing in Real Mode or in Virtual 8086 Mode.

The VM bit can be set by executing an IRET instruction only at privilege level 0 or by any instruction or interrupt that 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 Am386DXL microprocessor Protected Mode occurs only on receipt of an interrupt or exception (such as due to a sensitive instruction). In Virtual 8086 Mode, all interrupts and exceptions vector through the Protected Mode IDT, and enter an interrupt handler in Am386DXL CPU Protected 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 interlevel interrupt only to level 0. Interrupt or Trap gates through conforming segments or through segments with DPL > 0, will raise a GP fault with the CS selector as the error code.

Task Switches To/From Virtual 8086 Mode

Tasks which can execute in Virtual 8086 Mode must be described by a TSS with the new Am386DXL microprocessor format (Type 9 or 11 descriptor).

A task switch out of Virtual 8086 Mode will operate exactly the same as any other task switch out of a task with an Am386DXL CPU TSS. All of the programmer visible state, including the FLAGS register with the VM bit set to 1, is stored in the TSS. The segment registers in the TSS will contain 8086 segment base values rather than selectors.

A task switch into a task described by a Am386DXL microprocessor TSS will have an additional check to determine if the incoming task should be resumed in Virtual 8086 Mode. Tasks described by 80286 format TSSs cannot be resumed in Virtual 8086 Mode, so no check is required there (the FLAGS image in 80286 format TSS has only the low-order 16 FLAGS bits). Before loading the segment register images from a Am386DXL CPU TSS, the FLAGS image is loaded so that the segment registers are loaded from the TSS image as 8086 segment base values. The task is now ready to resume in Virtual 8086 Execution Mode.

Transitions Through Trap and Interrupt Gates, and IRET

A task switch is one way to enter or exit Virtual 8086 Mode. The other method is to exist through a Trap or Interrupt gate, as part of handling an interrupt, and to enter as part of executing an IRET instruction. The transition out must use a Am386DXL microprocessor Trap gate (Type 14) or Interrupt gate (Type 15) that 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. Am386DXL device gates must be used, since 80286 gates save only the lower 16 bits of the FLAGS register, so that the VM bit will not be saved on transitions through the 80286 gates. Also, the 16-bit IRET (presumably) used to terminate the 80286 interrupt handler will pop only the lower 16 bits from FLAGS, and will not affect the VM bit. The action taken for a Am386DXL microprocessor Trap or Interrupt gate if an interrupt occurs while the task is executing in Virtual 8086 Mode is given by the following sequence.

1. Save the FLAGS register in a temp to push later. Turn off the VM and TF bits, and if the interrupt is serviced by an Interrupt gate, turn off IF bit, also.
2. Interrupt and Trap gates must perform a level switch from 3 (where the VM86 program executes) to level 0 (so IRET can return). This process involves a stack switch to the stack given in the TSS for privilege level 0. Save the Virtual 8086 Mode SS and ESP registers to push in a later step. The segment register load of SS will be done as a Protected Mode segment load since the VM bit was turned off above.
3. Push the 8086 segment register values onto the new stack, in the order: GS, FS, DS, ES. These are pushed as 32-bit quantities with undefined values in the upper 16 bits. Then load these 4 registers with null selectors (0).
4. Push the old 8086 stack pointer onto the new stack by pushing the SS register (as 32-bit, 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 Am386DXL Microprocessor Mode.

The transition out of Virtual 8086 Mode performs a level change and stack switch, in addition to changing back to Protected Mode. In addition, all of the 8086 segment register images are stored on the stack (behind the SS:ESP image), and then loaded with null (0) selectors before entering the interrupt handler. This will permit the handler to safely save and restore the DS, ES, FS, and GS registers as 80286 selectors. This is needed so that interrupt handlers that "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 that 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 Am386DXL microprocessor IRET instruction (operand size = 32) can be used and must be executed at level 0 to change the VM bit to 1.

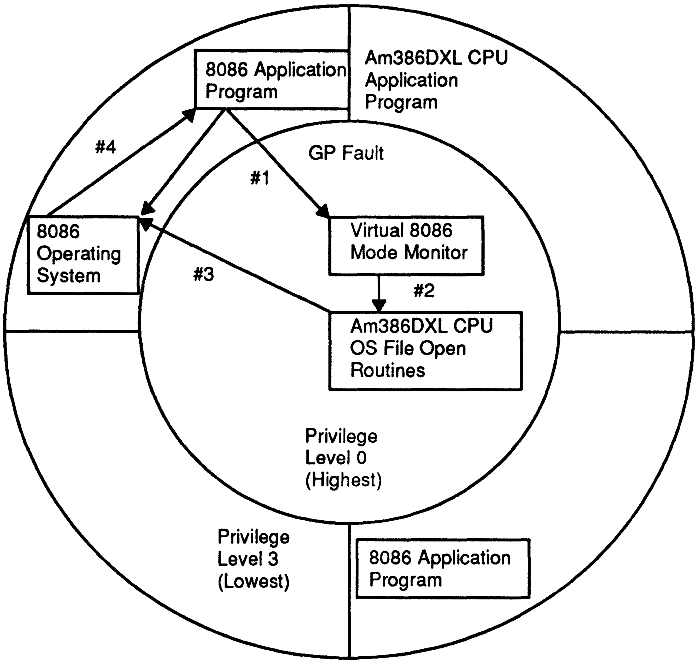
1. If the NT bit in the FLAGS register is on, an inter-task return is performed. The current state is stored in the current TSS, and the link field in the current TSS is used to locate the TSS for the interrupted task which is to be resumed.

   Otherwise, continue with the following sequence.
2. Read the FLAGS image from SS:8[ESP] into the FLAGS register. This will set VM to the value active in the interrupted routine.
3. Pop off the instruction pointer CS:EIP. EIP is popped first, then a 32-bit word is popped that contains the CS value in the lower 16 bits. If VM = 0, this CS load is done as a Protected Mode segment load. If VM = 1, this will be done as an 8086 segment load.
4. Increment the ESP register by 4 to bypass the FLAGS image which was popped in step 1.
5. If VM = 1, load segment registers ES, DS, FS, and GS from memory locations SS:[ESP+8], SS:[ESP+12], SS:[ESP+16], and SS:[ESP+20],

respectively, where the new value of ESP stored in step 4 is used. Since VM = 1, these are done as 8086 segment register loads.

Else if VM = 0, check that the selectors in ES, DS, FS, and GS are valid in the interrupted routine. Null out invalid selectors to trap if an attempt is made to access through them.

6. If (RPL(CS) > CPL), pop the stack pointer SS:ESP from the stack. The ESP register is popped first, followed by 32-bits containing SS in the lower 16 bits. If VM = 0, SS is loaded as a Protected Mode segment register load. If VM = 1, an 8086 segment register load is used.
7. Resume execution of the interrupted routine. The VM bit in the FLAGS register (restored from the interrupt routine's stack image in step 1) determines whether the processor resumes the interrupted routine in Protected Mode of Virtual 8086 Mode.


8086 Application Program
Am386DXL CPU Application Program
#4
GP Fault
#1
8086 Operating System
Virtual 8086 Mode Monitor
#3
#2
Am386DXL CPU OS File Open Routines
Privilege Level 0 (Highest)
Privilege Level 3 (Lowest)
8086 Application Program

8086 Application makes "Open File Call" → causes General Protection Fault (Arrow #1)
Virtual 8086 Monitor intercepts call. Calls Am386DXL CPU OS (Arrow #2)
Am386DXL CPU OS "Opens File" returns control to 8086 OS (Arrow #3)
8086 OS returns control to application (Arrow #4)
Transparent to Application

15021B–043

Figure 40. Virtual 8086 Environment Interrupt and Call Handling

FUNCTIONAL DATA

Introduction

The Am386DXL microprocessor features a straight forward functional interface to the external hardware. The Am386DXL CPU has separate parallel buses for data and address. The data bus is 32 bits in width and bi-directional. The address bus outputs 32-bit address values in the most directly usable form for the high-speed local bus: 4 individual Byte Enable signals and the 30 upper-order bits as a binary value. The data and address buses are interpreted and controlled with their associated control signals.

A dynamic data bus sizing feature allows the processor to handle a mix of 32- and 16-bit external buses on a cycle-by-cycle basis (see Data Bus Sizing). If 16-bit bus size is selected, the Am386DXL microprocessor automatically makes any adjustment needed even performing another 16-bit bus cycle to complete the transfer if that is necessary. Any 8-bit peripheral devices may be connected to 32- or 16-bit buses with no loss of performance. A new address pipelining option is provided and applies to 32- and 16-bit buses for substantially improved memory utilization, especially for the most heavily used memory resources.

The address pipelining option, when selected, typically allows a given memory interface to operate with one less wait state than would otherwise be required (see Address Pipelining). The pipelined bus is also well suited to interleaved memory designs. When address pipelining is requested by the external hardware, the Am386DXL microprocessor will output the address and bus cycle definition of the next bus cycle (if it is internally available) even while waiting for the current cycle to be acknowledged.

Non-pipelined address timing, however, is ideal for external cache designs, since the cache memory will typically be fast enough to allow non-pipelined cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor or from processor to system. Am386DXL microprocessor bus cycles perform data transfer in a minimum of only two clock periods. On a 32-bit data bus, the maximum Am386DXL device transfer at 20-MHz band-width is therefore 40 Mb/s, at 25-MHz bandwidth is 50 Mb/s, at 33-MHz bandwidth is 66 Mb/s, and at 40-MHz bandwidth is 80 Mb/s. Any bus cycle will be extended for more than two clock periods, however, if external hardware withholds acknowledgment of the cycle. At the appropriate time, acknowledgment is signaled by asserting the Am386DXL microprocessor $\overline{\text{READY}}$ input.

The Am386DXL CPU can relinquish control of its local buses to allow mastership by other devices, such as direct memory access channels. When relinquished, HLDA is the only output pin driven by the Am386DXL microprocessor providing near-complete isolation of the processor from its system. The near-complete isolation characteristic is ideal when driving the system from test equipment and in fault-tolerant applications.

Functional data covered in this section describes the processor's hardware interface. First, the set of signals available at the processor pins is described (see Signal Description). Following that are the signal waveforms occurring during bus cycles (see Bus Transfer Mechanism, Bus Functional Description, and Other Functional Descriptions).

Signal Description

Introduction

Ahead is a brief description of the Am386DXL CPU input and output signals arranged by functional groups.

Example signal:

M/$\overline{\text{IO}}$—High voltage indicates Memory selected
—Low voltage indicates I/O selected

The signal descriptions sometimes refer to AC timing parameters, such as t25 RESET Setup Time and t26 RESET Hold Time.

Clock (CLK2)

CLK2 provides the fundamental timing for the Am386DXL microprocessor. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two phases, phase one and phase two. Each CLK2 period is a phase of the internal clock. Figure 42 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the RESET signal falling edge meets its applicable setup and hold times, t25 and t26.

Data Bus (D31–D0)

These three-state, bi-directional signals provide the general purpose data path between the Am386DXL microprocessor and other devices. Data bus inputs and outputs indicate 1 when High. The data bus can transfer data on 32- and 16-bit buses using a data bus sizing feature controlled by the $\overline{\text{BS16}}$ input. See Section Bus Control. Data bus reads require that read data setup and hold times, t21 and t22, be met for correct operation. In addition, the Am386DXL microprocessor requires that all data bus pins be at a valid logic state (High or Low) at the end of each read cycle, when $\overline{\text{READY}}$ is asserted. During any write operation (and during halt cycles and

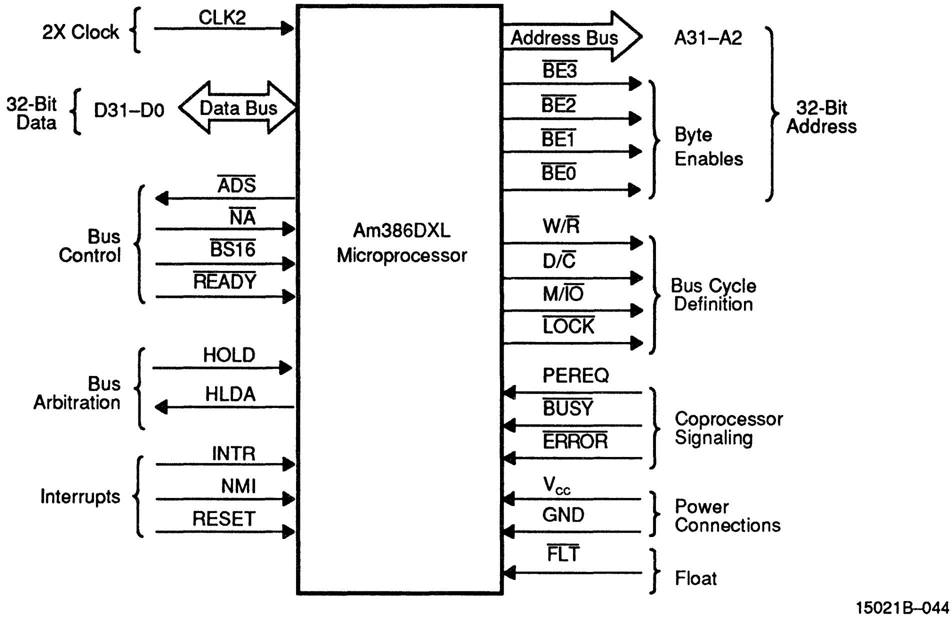


Figure 41. Functional Signal Groups

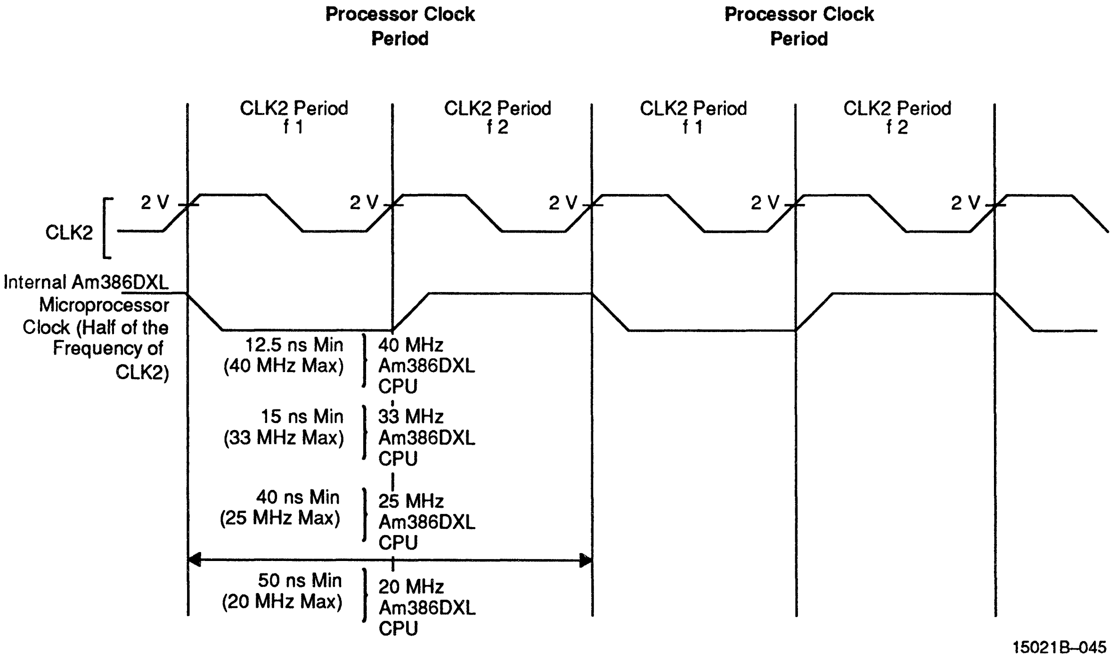


Figure 42. CLK2 Signal and Internal Processor Clock

shut down cycles), the Am386DXL microprocessor always drives all 32 signals of the data bus even if the current bus size is 16 bits.

Address Bus ($\overline{\text{BE3}}$–$\overline{\text{BE0}}$, A31–A2)

These three-state outputs provide physical memory addresses or I/O port addresses. The address bus is capable of addressing 4 Gb of physical memory space (00000000H–FFFFFFFFH), and 64 Kb of I/O address space (00000000H–0000FFFFH) for programmed I/O. I/O transfers automatically generated for Am386DXL microprocessor-to-coprocessor communication use I/O addresses 800000F8H–800000FFH, so A31 is High in conjunction with M/$\overline{\text{IO}}$ Low allows simple generation of the coprocessor select signal.

The Byte Enable outputs, $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, directly indicate which bytes of the 32-bit data bus are involved with the current transfer. This is most convenient for external hardware.

$\overline{\text{BE0}}$ applies to D7–D0

$\overline{\text{BE1}}$ applies to D15–D8

$\overline{\text{BE2}}$ applies to D23–D16

$\overline{\text{BE3}}$ applies to D31–D24

The number of Byte Enables asserted indicates the physical size of the operand being transferred (1, 2, 3, or 4 bytes). Refer to Section Operand Alignment.

When a memory write cycle or I/O write cycle is in progress and the operand being transferred occupies only the upper 16 bits of the data bus (D31–D16), duplicate data is simultaneously presented on the corresponding lower 16 bits of the data bus (D15–D0). This duplication is performed for optimum write performance on 16 bit buses. The pattern of write data duplication is a function of the Byte Enables asserted during the write cycle. Table 13 lists the write data present on D31–D0, as a function of the asserted Byte Enable outputs $\overline{\text{BE3}}$–$\overline{\text{BE0}}$.

Bus Cycle Definition Signals (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$)

These three-state outputs define the type of bus cycle being performed. W/$\overline{\text{R}}$ distinguishes between write and read cycles. D/$\overline{\text{C}}$ distinguishes between data and control cycles. M/$\overline{\text{IO}}$ distinguishes between memory and I/O cycles. $\overline{\text{LOCK}}$ distinguishes between locked and unlocked bus cycles.

The primary bus cycle definition signals are W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, and M/$\overline{\text{IO}}$, since these are the signals driven valid as the $\overline{\text{ADS}}$ (Address Status output) is driven asserted. The $\overline{\text{LOCK}}$ is driven valid at the same time as the first locked bus cycle begins, which due to address pipelining, could be later than $\overline{\text{ADS}}$ is driven asserted. See Pipelined Address. The $\overline{\text{LOCK}}$ is negated when the $\overline{\text{READY}}$ input terminates the last bus cycle that was locked.

Exact bus cycle definitions, as a function of W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, and M/$\overline{\text{IO}}$, are given in Table14. Note one combination of W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, and M/$\overline{\text{IO}}$ is never given when $\overline{\text{ADS}}$ is asserted (however, that combination, which is listed as does not occur, may occur during idle bus states when $\overline{\text{ADS}}$ is not asserted). If M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, and W/$\overline{\text{R}}$ are qualified by $\overline{\text{ADS}}$ asserted, then a decoding scheme may be simplified by using this definition of the does not occur combination.

Table 13. Write Data Duplication as a Function of $\overline{\text{BE3}}$–$\overline{\text{BE0}}$

Am386DXL CPU Byte Enables				Am386DXL CPU Write Data				Automatic Duplication?
$\overline{\text{BE3}}$	$\overline{\text{BE2}}$	$\overline{\text{BE1}}$	$\overline{\text{BE0}}$	D31–D24	D23–D16	D15–D8	D7–D0	
High	High	High	Low	Undef	Undef	Undef	A	No
High	High	Low	High	Undef	Undef	B	Undef	No
High	Low	High	High	Undef	C	Undef	C	Yes
Low	High	High	High	D	Undef	D	Undef	Yes
High	High	Low	Low	Undef	Undef	B	A	No
High	Low	Low	High	Undef	C	B	Undef	No
Low	Low	High	High	D	C	D	C	Yes
High	Low	Low	Low	Undef	C	B	A	No
Low	Low	Low	High	D	C	B	Undef	No
Low	Low	Low	Low	D	C	B	A	No

Key: D = Logical Write Data D31–D24 B = Logical Write Data D15–D8
C = Logical Write Data D23–D16 A = Logical Write Data D7–D0

Table 14. Bus Cycle Definition

M/$\overline{\text{IO}}$	D/$\overline{\text{C}}$	W/$\overline{\text{R}}$	Bus Cycle Type	Locked?
Low	Low	Low	Interrupt Acknowledge	Yes
Low	Low	High	Does Not Occur	—
Low	High	Low	I/O Data Read	No
Low	High	High	I/O Data Write	No
High	Low	Low	Memory Code Read	No
High	Low	High	Halt: Address = 2 $\overline{\text{BE0}}$ High $\overline{\text{BE1}}$ High $\overline{\text{BE2}}$ Low $\overline{\text{BE3}}$ High A31–A2 Low Shutdown: Address = 0 $\overline{\text{BE0}}$ Low $\overline{\text{BE1}}$ High $\overline{\text{BE2}}$ High $\overline{\text{BE3}}$ High A31–A2 Low	No
High	High	Low	Memory Data Read	Some Cycles
High	High	High	Memory Data Write	Some Cycles

Bus Control Signals ($\overline{\text{ADS}}$, $\overline{\text{READY}}$, $\overline{\text{NA}}$, $\overline{\text{BS16}}$)

Introduction

The following signals allow the processor to indicate when bus cycle has begun and allow other system hardware to control address pipelining, data bus width, and bus cycle termination.

Address Status ($\overline{\text{ADS}}$)

This three-state output indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, and A31–A2) is being driven at the Am386DXL microprocessor pins. It is asserted during T1 and T2P bus states (see Non-pipelined Address and Pipelined Address for additional information on bus states).

Transfer Acknowledge ($\overline{\text{READY}}$)

This input indicates the current bus cycle is complete, and the active bytes indicated by $\overline{\text{BE3}}$–$\overline{\text{BE0}}$ and $\overline{\text{BS16}}$ are accepted or provided. When $\overline{\text{READY}}$ is sampled asserted during a read cycle or interrupt acknowledge cycle, the Am386DXL microprocessor latches the input data and terminates the cycle. When $\overline{\text{READY}}$ is sampled asserted during a write cycle, the processor terminates the bus cycle.

$\overline{\text{READY}}$ is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. $\overline{\text{READY}}$ must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, $\overline{\text{READY}}$ must always meet setup and hold times, t19 and t20, for correct operation. See all sections of Bus Functional Description.

Next Address Request ($\overline{\text{NA}}$)

This is used to request address pipelining. This input indicates the system is prepared to accept new values of $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, A31–A2, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, and M/$\overline{\text{IO}}$ from the Am386DXL microprocessor even if the end of the current cycle is not being acknowledged on $\overline{\text{READY}}$. If this input is asserted when sampled, the next address is driven onto the bus provided the next bus request is already pending internally. See Address Pipelining and Read and Write Cycles. $\overline{\text{NA}}$ must always meet setup and hold times, t15 and t16, for correct operation.

Bus Size 16 ($\overline{\text{BS16}}$)

The $\overline{\text{BS16}}$ feature allows the Am386DXL microprocessor to directly connect to 32- and 16-bit data buses. Asserting this input constrains the current bus cycle to use only the lower-order half (D15–D0) of the data bus, corresponding to $\overline{\text{BE0}}$ and $\overline{\text{BE1}}$. Asserting $\overline{\text{BS16}}$ has no additional effect if only $\overline{\text{BE0}}$ and/or $\overline{\text{BE1}}$ are asserted in the current cycle. However, during bus cycles asserting $\overline{\text{BE2}}$ or $\overline{\text{BE3}}$, asserting $\overline{\text{BS16}}$ will automatically cause the Am386DXL microprocessor to make adjustments for correct transfer of the upper byte(s) using only physical data signals D15–D0.

If the operand spans both halves of the data bus and $\overline{\text{BS16}}$ is asserted, the Am386DXL microprocessor will automatically perform another 16-bit bus cycle. $\overline{\text{BS16}}$ must always meet setup and hold times, t17 and t18, for correct operation.

Am386DXL CPU I/O cycles are automatically generated for coprocessor communication. Since the Am386DXL microprocessor must transfer 32-bit quantities between itself and a 387DX math coprocessor, $\overline{\text{BS16}}$ must not be asserted during 387DX math coprocessor communication cycles.

Bus Arbitration Signals (HOLD, HLDA)

Introduction

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See

Entering and Exiting Hold Acknowledge for additional information.

Bus Hold Request (HOLD)

This input indicates some device other than the Am386DXL microprocessor requires bus mastership.

HOLD must remain asserted as long as any other device is a local bus master. HOLD is not recognized while RESET is asserted. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the hold acknowledge (high impedance) state. HOLD is level-sensitive and is a synchronous input. HOLD signals must always meet setup and hold times, t23 and t24, for correct operation.

Bus Hold Acknowledge (HLDA)

Assertion of this output indicates the Am386DXL microprocessor has relinquished control of its local bus in response to HOLD asserted, and is in the Bus Hold Acknowledge state.

The Hold Acknowledge state offers near-complete signal isolation. In the Hold Acknowledge state, HLDA is the only signal being driven by the Am386DXL microprocessor. The other output signals or bi-directional signals (D31–D0, $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, A31–A2, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, and $\overline{\text{ADS}}$) are in a high-impedance state so the requesting bus master may control them. Pullup resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See Resistor Recommendations. Also, one rising edge occurring on the NMI input during Hold Acknowledge is remembered for processing after the HOLD input is negated.

In addition to the normal usage of Hold Acknowledge with DMA controllers or master peripherals, the near-complete isolation has particular attractiveness during system test when test equipment drives the system and in hardware-fault-tolerant applications.

Coprocessor Interface Signals (PEREQ, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$)

Introduction

In the following sections are descriptions of signals dedicated to the numeric coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, these following signals control communication between the Am386DXL microprocessor and its 387DX math coprocessor extension.

Coprocessor Request (PEREQ)

When asserted, this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the Am386DXL microprocessor. In response, the Am386DXL CPU transfers information between the coprocessor and memory. Because Am386DXL microprocessor has internally stored the coprocessor op-code being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

Coprocessor Busy ($\overline{\text{BUSY}}$)

When asserted, this input indicates the coprocessor is still executing an instruction and is not yet able to accept another. When the Am386DXL microprocessor encounters any coprocessor instruction that operates on the numeric stack (e.g., load, pop, or arithmetic operation) or the WAIT instruction, this input is first automatically sampled until it is seen to be negated. This sampling of the $\overline{\text{BUSY}}$ input prevents overrunning the execution of a previous coprocessor instruction.

The FNINIT and FNCLEX coprocessor instructions are allowed to execute even if $\overline{\text{BUSY}}$ is asserted, since these instructions are used for coprocessor initialization and exception-clearing.

$\overline{\text{BUSY}}$ is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

$\overline{\text{BUSY}}$ serves an additional function. If $\overline{\text{BUSY}}$ is sampled Low at the falling edge of RESET, the Am386DXL microprocessor performs an internal self-test (see Bus Activity During and Following Reset). If $\overline{\text{BUSY}}$ is sampled High, no self-test is performed.

Coprocessor Error ($\overline{\text{ERROR}}$)

This input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the Am386DXL microprocessor when a coprocessor instruction is encountered, and if asserted, the Am386DXL device generates Exception 16 to access the error-handling software.

Several coprocessor instructions, generally those that clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the Am386DXL microprocessor generating Exception 16 even if $\overline{\text{ERROR}}$ is asserted. These instructions are FNINIT, FNCLEX, FSTSW, FSTSWAX, FSTCW, FSTENV, FSAVE, FESTENV, and FESAVE.

$\overline{\text{ERROR}}$ is level-sensitive and is allowed to be asynchronous to the CLK2 signal.

Interrupt Signals (INTR, NMI, RESET)

Introduction

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

Maskable Interrupt Request (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the Am386DXL CPU Flag Register IF bit. When the Am386DXL microprocessor responds to the INTR input, it performs two interrupt acknowledge bus cycles, and at the end of the second, latches an 8-bit interrupt vector on D17–D0 to identify the source of the interrupt.

INTR is level-sensitive and is allowed to be asynchronous to the CLK2 signal. To assure recognition of an INTR request, INTR should remain asserted until the first interrupt acknowledge bus cycle begins.

Non-Maskable Interrupt Request (NMI)

This input indicates a request for interrupt service, which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are performed when processing NMI.

NMI is rising edge-sensitive and is allowed to be asynchronous to the CLK2 signal. To assure recognition of NMI, it must be negated for at least eight CLK2 periods, and then be asserted for at least eight CLK2 periods.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

Reset (RESET)

This input signal suspends any operation in progress and places the Am386DXL microprocessor in a known reset state. The Am386DXL device is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self-test). When RESET is asserted, all other input pins, except $\overline{\mathrm{FLT}}$, are ignored, and all other bus pins are driven to an idle bus state as shown in Table 15. If RESET and HOLD are both asserted at a point in time, RESET takes priority even if the Am386DXL device was in a Hold Acknowledge state prior to RESET asserted.

RESET is level-sensitive and must be synchronous to the CLK2 signal. If desired, the phase of the internal processor clock and the entire Am386DXL microprocessor state can be completely synchronized to external circuitry by ensuring the RESET signal falling edge meets its applicable setup and hold times, t25 and t26.

Table 15. Pin State (Idle Bus) During Reset

Pin Name	Signal Level During Reset
$\overline{\mathrm{ADS}}$	High
D31–D0	High Impedance
$\overline{\mathrm{BE3}}$–$\overline{\mathrm{BE0}}$	Low
A31–A2	High
W/$\overline{\mathrm{R}}$	Low
D/$\overline{\mathrm{C}}$	High
M/$\overline{\mathrm{IO}}$	Low
$\overline{\mathrm{LOCK}}$	High
HLDA	Low

Table 16. Am386DXL Microprocessor Signal Summary

Signal Name	Function	Active State	Input/ Output	Input Synch or Asynch to CLK2	Output High Impedance During HLDA?
CLK2	Clock	—	I	—	—
D31–D0	Data Bus	High	I/O	S	Yes
$\overline{\mathrm{BE3}}$–$\overline{\mathrm{BE0}}$	Byte Enables	Low	O	—	Yes
A31–A2	Address Bus	High	O	—	Yes
W/$\overline{\mathrm{R}}$	Write-Read Indication	High	O	—	Yes
D/$\overline{\mathrm{C}}$	Data-Control Indication	High	O	—	Yes
M/$\overline{\mathrm{IO}}$	Memory-I/O Indication	High	O	—	Yes
$\overline{\mathrm{LOCK}}$	Bus Lock Indication	Low	O	—	Yes
$\overline{\mathrm{ADS}}$	Address Status	Low	O	—	Yes
$\overline{\mathrm{NA}}$	Next Address Request	Low	I	S	—
$\overline{\mathrm{BS16}}$	Bus Size 16	Low	I	S	—
$\overline{\mathrm{READY}}$	Transfer Acknowledge	Low	I	S	—
HOLD	Bus Hold Request	High	I	S	—
HLDA	Bus Hold Acknowledge	High	O	—	No
PEREQ	Coprocessor Request	High	I	A	—
$\overline{\mathrm{BUSY}}$	Coprocessor Busy	Low	I	A	—
$\overline{\mathrm{ERROR}}$	Coprocessor Error	Low	I	A	—
INTR	Maskable Interrupt Request	High	I	A	—
NMI	Non-Maskable Intrpt Request	High	I	A	—
RESET	Reset	High	I	S	—

Bus Transfer Mechanism

Introduction

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte, word, and double-word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two or even three physical bus cycles are performed as required for unaligned operand transfers. See Dynamic Data Bus Sizing and Operand Alignment.

The Am386DXL microprocessor address signals are designed to simplify external system hardware. Higher-order address bits are provided by A31–A2. Lower-order address in the form of $\overline{BE3}$–$\overline{BE0}$ directly provides linear selects for the four bytes of the 32-bit data bus. Physical operand size information is thereby implicitly provided each bus cycle in the most usable form.

Byte Enable outputs, $\overline{BE3}$–$\overline{BE0}$, are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 17. During a bus cycle, any possible pattern of contiguous asserted Byte Enable outputs can occur, but never patterns having a negated Byte Enable separating two or three asserted Enables.

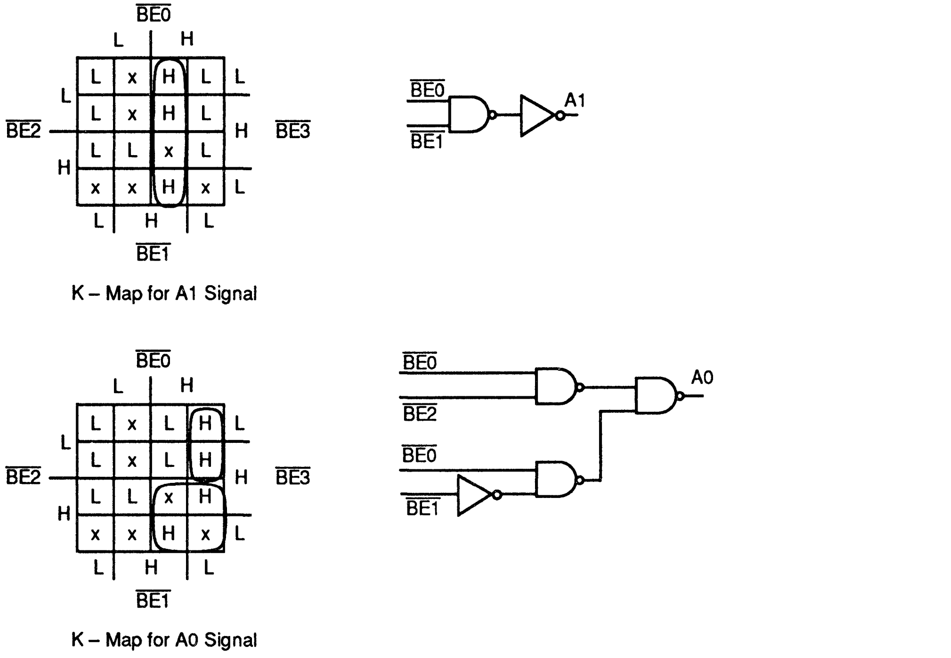
Address bits A0 and A1 of the physical operand's base address can be created when necessary (for instance, for MULTIBUS I or MULTIBUS II interface), as a function of the lowest-order asserted Byte Enable. This is shown by Table 18. Logic to generate A0 and A1 is given by Figure 43.

Table 17. Byte Enables and Associated Data and Operand Bytes

Byte Enable Signal	Associated Data Bus Signals
$\overline{BE0}$	D7–D0 (Byte 0—least significant)
$\overline{BE1}$	D15–D8 (Byte 1)
$\overline{BE2}$	D23–D16 (Byte 2)
$\overline{BE3}$	D31–D24 (Byte 3—most significant)

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See Bus Functional Description.

Since a bus cycle requires a minimum of two bus states (equal to two processor clock periods), data can be transferred between external devices and the Am386DXL CPU at a maximum rate of one 4-byte Dword every two processor clock periods, for a maximum bus bandwidth of 80 Mb/s (Am386DXL microprocessor operating at 40-MHz processor clock rate).



$\overline{BE2}$ / $\overline{BE3}$ \ $\overline{BE0}$ / $\overline{BE1}$	L / L	L / H	H / H	H / L
L / L	L	x	H	L
L / H	L	x	H	L
H / H	L	L	x	L
H / L	x	x	H	x

K – Map for A1 Signal

$\overline{BE2}$ / $\overline{BE3}$ \ $\overline{BE0}$ / $\overline{BE1}$	L / L	L / H	H / H	H / L
L / L	L	x	L	H
L / H	L	x	L	H
H / H	L	L	x	H
H / L	x	x	H	x

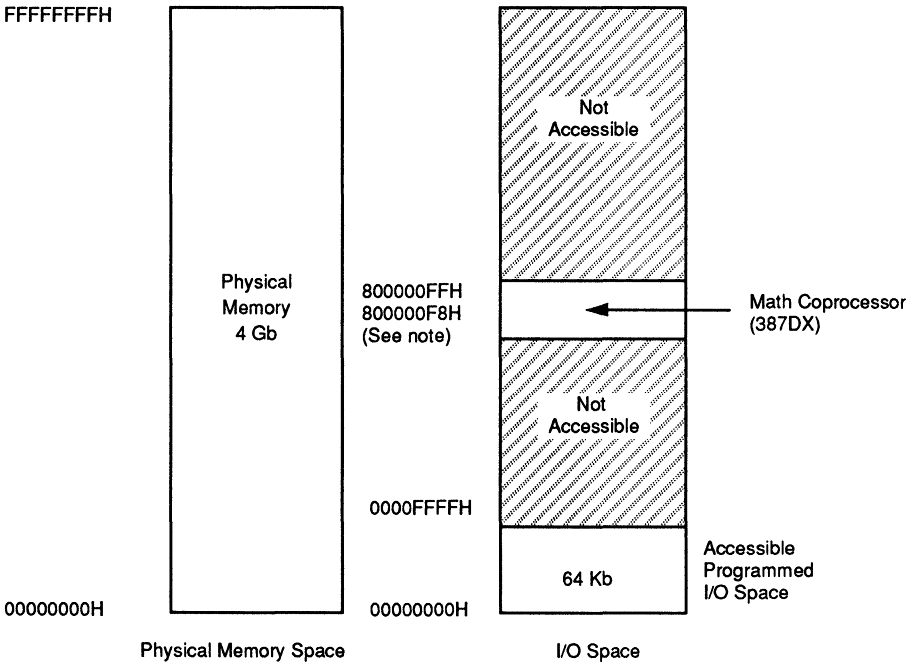
K – Map for A0 Signal

15021B–046

Figure 43. Logic to Generate A0, A1 from $\overline{BE3}$–$\overline{BE0}$

Table 18. Generating A31–A0 from $\overline{BE3}$–$\overline{BE0}$ and A31–A2

Am386DXL CPU Address Signals								
A31	**.........**	**A2**			**$\overline{BE3}$**	**$\overline{BE2}$**	**$\overline{BE1}$**	**$\overline{BE0}$**
	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



Note: Since A31 is High during automatic communication with coprocessor, A31 High and M/$\overline{IO}$ Low can be used to easily generate a coprocessor select signal.

15021B–047

Figure 44. Physical Memory and I/O Spaces

Memory and I/O Spaces

Bus cycles may access physical memory space or I/O space. Peripheral devices in the system may either be memory-mapped, or I/O-mapped, or both. As shown in Figure 44, physical memory addresses range from 00000000H to FFFFFFFFH (4 Gb) and I/O addresses from 00000000H to 0000FFFFH (64 Kb) for programmed I/O. Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 800000F8H to 800000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A31 and M/$\overline{IO}$ signals.

Memory and I/O Organization

The Am386DXL microprocessor datapath to memory and I/O spaces can be 32- or 16-bits wide. When 32-bits wide, memory and I/O spaces are organized naturally as arrays of physical 32-bit Dwords. Each memory or I/O Dword has four individually addressable bytes at consecutive byte addresses. The lowest-addressed byte is associated with data signals D17–D0; the highest-addressed byte with D31–D24.

The Am386DXL microprocessor includes a bus control input, $\overline{BS16}$, that also allows direct connection to 16-bit memory or I/O spaces organized as a sequence of 16-bit word. Cycles to 32- and 16-bit memory or I/O devices may occur in any sequence, since the $\overline{BS16}$ control is sampled during each bus cycle. (See Dynamic Data Bus Sizing.) The Byte Enable signals, $\overline{BE3}$–$\overline{BE0}$, allow byte granularity when addressing any memory or I/O structure, whether 32- or 16-bits wide.

Dynamic Data Bus Sizing

Dynamic Data Bus Sizing is a feature allowing direct processor connection to 32- or 16-bit data buses for memory or I/O. A single processor may connect to both size buses. Transfers to or from 32- or 16-bit ports are supported by dynamically determining the bus width during each bus cycle. During each bus cycle an address decoding circuit or the slave device itself may assert $\overline{BS16}$ for 16-bit ports, or negate $\overline{BS16}$ for 32-bit ports.

With $\overline{BS16}$ asserted, the processor automatically converts operand transfers larger than 16 bits, or misaligned 16-bit transfers, into two or three transfers as required. All operand transfers physically occur on D15–D0 when $\overline{BS16}$ is asserted. Therefore, 16-bit memories or I/O devices only connect on data signals D15–D0. No extra transceivers are required.

Asserting $\overline{BS16}$ only affects the processor when $\overline{BE2}$ and/or $\overline{BE3}$ are asserted during the current cycle. If only D15–D0 are involved with the transfer, asserting $\overline{BS16}$ has no affect since the transfer can proceed normally over a 16-bit bus whether $\overline{BS16}$ is asserted or not. In other words, asserting $\overline{BS16}$ has no effect when only the lower half of the bus is involved with the current cycle.

There are two types of situations where the processor is affected by asserting $\overline{BS16}$, depending on which Byte Enables are asserted during the current bus cycle.

Upper Half Only:

Only $\overline{BE2}$ and/or $\overline{BE3}$ asserted.

Upper and Lower Half:

At least $\overline{BE1}$, $\overline{BE2}$ asserted (and perhaps also $\overline{BE0}$ and/or $\overline{BE3}$).

Effect of asserting $\overline{BS16}$ during Upper Half Only read cycles:

Asserting $\overline{BS16}$ during Upper Half Only reads causes the Am386DXL microprocessor to read data on the lower 16 bits of the data bus and ignore data on the upper 16 bits of the data bus. Data that would have been read from D31–D16 (as indicated by $\overline{BE2}$ and $\overline{BE3}$) will instead be read from D15–D0, respectively.

Effect of asserting $\overline{BS16}$ during Upper Half Only write cycles:

Asserting $\overline{BS16}$ during Upper Half Only writes does not affect the Am386DXL microprocessor. When only $\overline{BE2}$ and/or $\overline{BE3}$ are asserted during a Write cycle, the Am386DXL microprocessor always duplicates data signals D31–D16 onto D15–D0 (see Table 13). Therefore, no further Am386DXL CPU action is required to perform these writes on 32- or 16-bit buses.

Effect of asserting $\overline{BS16}$ during Upper and Lower Half read cycles:

Asserting $\overline{BS16}$ during Upper and Lower Half reads causes the processor to perform two 16-bit read cycles for complete physical operand transfer. Bytes 0 and 1 (as indicated by $\overline{BE0}$ and $\overline{BE1}$) are read on the first cycle using D15–D0. Bytes 2 and 3 (as indicated by $\overline{BE2}$ and $\overline{BE3}$) are read during the second cycle, again using D15–D0. D31–D16 are ignored during both 16-bit cycles. $\overline{BE0}$ and $\overline{BE1}$ are always negated during the second 16-bit cycle. See Figure 54 Cycles 2 and 2a.

Effect of asserting $\overline{BS16}$ during Upper and Lower Half write cycles:

Asserting $\overline{BS16}$ during Upper and Lower Half writes causes the Am386DXL microprocessor to perform two 16-bit write cycles for complete physical operand transfer. All bytes are available the first write cycle allowing external hardware to receive Bytes 0 and 1 (as indicated by $\overline{BE0}$ and $\overline{BE1}$) using D15–D0. On the second cycle the Am386DXL microprocessor duplicates Bytes 2 and 3 on D15–D0 and Bytes 2 and 3 (as indicated by $\overline{BE2}$ and $\overline{BE3}$) are written using D15–D0. $\overline{BE0}$ and $\overline{BE1}$ are always negated during the second 16-bit cycle. $\overline{BS16}$ must be asserted during the second 16-bit cycle. See Figure 54 Cycles 1 and 1a.

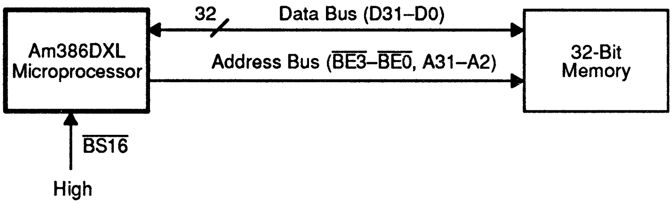
Interfacing with 32- and 16-Bit Memories

In 32-bit-wide physical memories such as Figure 45, each physical Dword begins at a byte address that is a multiple of 4. A31–A2 are directly used as a Dword selects and BE3–BE0 as byte selects. BS16 is negated for all bus cycles involving the 32-bit array.

When 16-bit-wide physical arrays are included in the system, as in Figure 46, each 16-bit physical word begins at an address that is a multiple of 2. Note the address is decoded to assert BS16 only during bus cycles involving the 16-bit array. If desiring to use pipelined address with 16-bit memories, then BE3–BE0 and W/R are also decoded to determine when BS16 should be asserted. (See Pipelined Address with Dynamic Data Bus Sizing.)

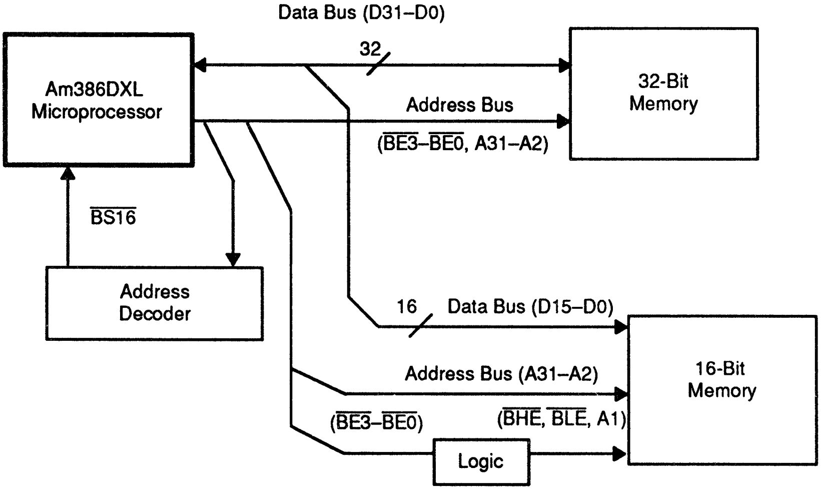
A31–A2 are directly usable for addressing 32- and 16-bit devices. To address 16-bit devices, A1 and two Byte Enable signals are also needed.

To generate an A1 signal and two Byte Enable signals for 16-bit access, BE3–BE0 should be decoded as in Table 19. Note certain combinations of BE3–BE0 are never generated by the Am386DXL microprocessor, leading to don't care conditions in the decoder. Any BE3–BE0 decoder, such as shown in Figure 47, may use the non-occurring BE3–BE0 combinations to its best advantage.



15021B–048

Figure 45. Am386DXL Microprocessor with 32-Bit Memory



15021B–049

Figure 46. Am386DXL Microprocessor with 32-Bit and 16-Bit Memory

Table 19. Generating A1, $\overline{BHE}$ and $\overline{BLE}$ for Addressing 16-Bit Devices

Am386DXL CPU Signals				16-Bit Bus Signals			Comments
$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$	A1	$\overline{BHE}$	$\overline{BLE}$ (A0)	
H*	H*	H*	H*	X	X	X	X—no active bytes
H	H	H	L	L	H	L	
H	H	L	H	L	L	H	
H	H	L	L	L	L	L	
H	L	H	H	H	H	L	
H*	L*	H*	L*	X	X	X	X—not contiguous bytes
H	L	L	H	L	L	H	
H	L	L	L	L	L	L	
L	H	H	H	H	L	H	
L*	H*	H*	L*	X	X	X	X—not contiguous bytes
L*	H*	L*	H*	X	X	X	X—not contiguous bytes
L*	H*	L*	L*	X	X	X	X—not contiguous bytes
L	L	H	H	H	L	L	
L*	L*	H*	L*	X	X	X	X—not contiguous bytes
L	L	L	H	L	L	H	
L	L	L	L	L	L	L	

$\overline{BLE}$ asserted when D7–D0 of 16-bit bus is active.
$\overline{BHE}$ asserted when D15–D8 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.

Operand Alignment

With the flexibility of memory addressing on the Am386DXL microprocessor, it is possible to transfer a logical operand that spans more than one physical Dword or Word of memory or I/O. Examples are 32-bit Dword operands beginning at addresses not evenly divisible by 4- or a 16-bit Word operand split between two physical Dwords of memory array.

Operand alignment and data bus size dictates when multiple bus cycles are required. Table 20 describes the transfer cycles generated for all combinations of logical operand lengths, alignment, and data bus sizing. When multiple bus cycles are required to transfer a multi-byte logical operand, the highest-order bytes are transferred first (but if $\overline{BS16}$ asserted requires two 16-bit cycles be performed, that part of the transfer is lowest-order first).

Bus Functional Description

Introduction

The Am386DXL microprocessor has separate, parallel buses for data and address. The data bus is 32 bits in width and bi-directional. The address bus provides a 32-bit value using 30 signals for the 30 upper-order address bits and 4 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled via several associated definition or control signals.

The definition of each bus cycle is given by three definition signals: M/$\overline{IO}$, W/$\overline{R}$, and D/$\overline{C}$. At the same time, a valid address is present on the Byte Enable signals $\overline{BE3}$–$\overline{BE0}$ and other address signals, A31–A2. A status signal, $\overline{ADS}$, indicates when the Am386DXL CPU issues a new bus cycle definition and address.

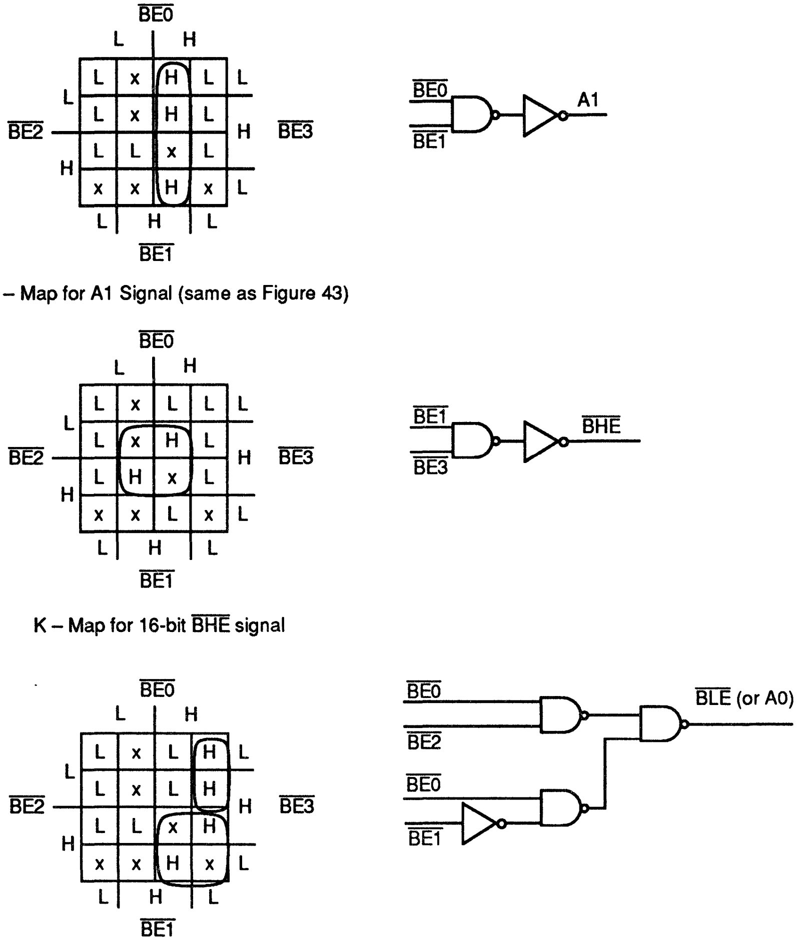
Collectively, the address bus, data bus, and all associated control signals are referred to simply as the bus.

When active, the bus performs one of the bus cycles below.

1. Read from memory space.
2. Locked read from memory space.
3. Write to memory space.
4. Locked write to memory space.
5. Read from I/O space (or coprocessor).
6. Write to I/O space (or coprocessor).
7. Interrupt acknowledge.
8. Indicate halt or indicate shutdown.

Table 14 shows the encoding of the bus cycle definition signals for each bus cycle. See Section Bus Cycle Definition.

The data bus has a dynamic sizing feature supporting 32- and 16-bit bus size. Data bus size is indicated to the Am386DXL microprocessor using its Bus Size 16 ($\overline{BS16}$) input. All bus functions can be performed with either data bus size.



K – Map for A1 Signal (same as Figure 43)

K – Map for 16-bit $\overline{BHE}$ signal

K – Map for 16-bit $\overline{BLE}$ signal (same as A0 signal in Figure 43).

15021B–050

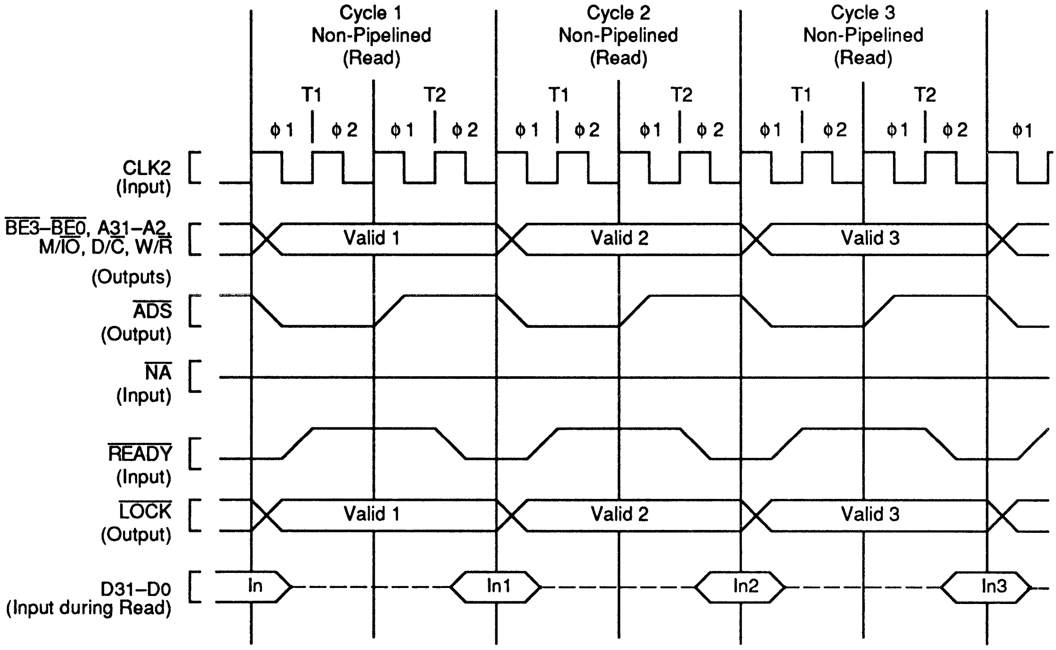
Figure 47. Logic to Generate A1, $\overline{BHE}$ and $\overline{BLE}$ for 16-Bit Buses

Table 20. Transfer Bus Cycles for Bytes, Words, and Dwords

	Byte-Length of Logical Operand								
	1	**2**				**4**			
Physical Byte Addres in Memory (low-order bits)	xx	00	01	10	11	00	01	10	11
Transfer Cycles over 32-Bit Data Bus	b	w	w	w	hb,* lb	d	hd l3	hw, lw	h3, lb
Transfer Cycles over 16-Bit Data Bus	b	w	lb, hb	w	hb, lb	lw, hw	hb, lb, mw	hw, lw	mw, hb, lb

Key: b = Byte transfer
w = Word transfer
l = low-order portion
m = mid-order portion
3 = 3-byte transfer
d = Dword transfer
h = high-order portion
x = Don't care
= $\overline{BS16}$ asserted causes second bus cycle.

*For this case, 8086, 8088, 80186, 80188, 80286 transfer lb first, then hb.



Cycle 1 Non-Pipelined (Read)
Cycle 2 Non-Pipelined (Read)
Cycle 3 Non-Pipelined (Read)
T1
T2
ϕ1
ϕ2
CLK2 (Input)
BE3–BE0, A31–A2, M/IO, D/C, W/R (Outputs)
Valid 1
Valid 2
Valid 3
ADS (Output)
NA (Input)
READY (Input)
LOCK (Output)
D31–D0 (Input during Read)
In
In1
In2
In3

Fastest non-pipelined bus cycles consist of T1 and T2

15021B–051

Figure 48. Fastest Read Cycles with Non-Pipelined Address Timing

When the Am386DXL CPU bus is not performing one of the activities listed above, it is either Idle or in the Hold Acknowledge state, which may be detected by external circuitry. The Idle state can be identified by the Am386DXL microprocessor giving no further assertions on its address strobe output (ADS) since the beginning of its most recent bus cycle, and the most recent bus cycle has been terminated. The Hold Acknowledge state is identified by the Am386DXL CPU asserting its Hold Acknowledge (HLDA) output.

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

The fastest Am386DXL microprocessor bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown by Figure 48. The bus states in each cycle are named T1 and T2. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough. The high-bandwidth, two-clock bus cycle realizes the full potential of fast main memory, or cache memory.

Every bus cycle continues until it is acknowledged by the external system hardware, using the Am386DXL microprocessor READY input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If READY is not immediately asserted, however, T2 states are repeated indefinitely until the READY input is sampled asserted.

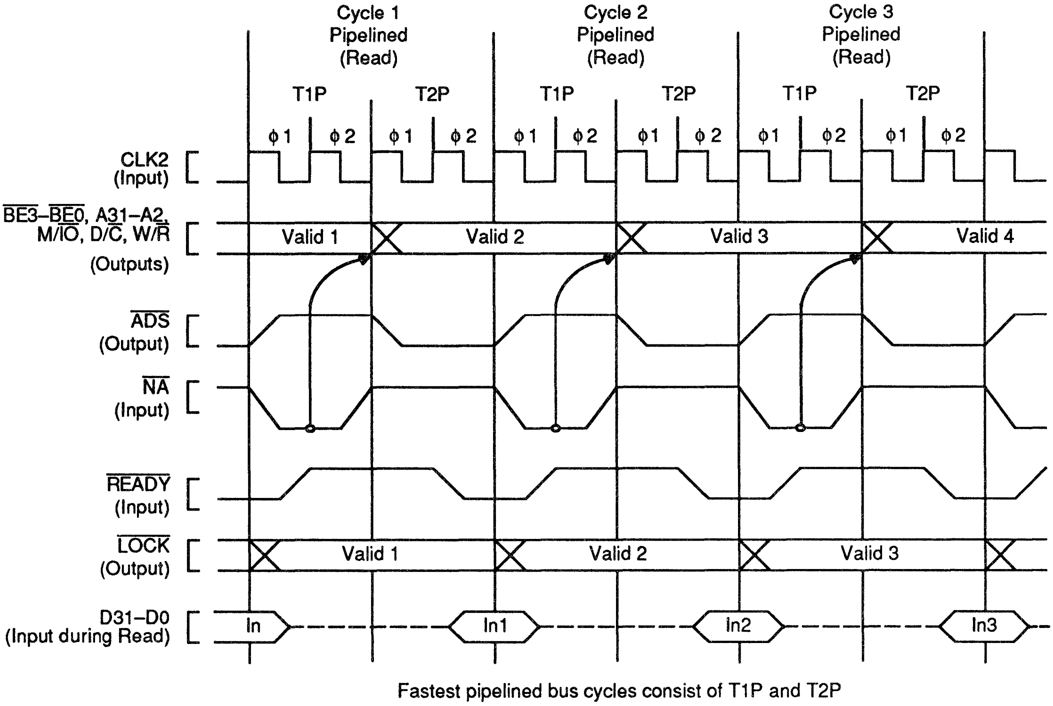
Address Pipelining

The address pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined address timing is selectable on a cycle-by-cycle basis with the Next Address (NA) input.

When address pipelining is not selected, the current address and bus cycle definition remain stable throughout the bus cycle.

When address pipelining is selected, the address (BE3–BE0, A31–A2) and definition (W/R, D/C, and M/IO) of the next cycle are available before the end of the current cycle. To signal their availability, the Am386DXL microprocessor address status output (ADS) is also asserted. Figure 49 illustrates the fastest read cycles with pipelined address timing.

Note from Figure 49, the fastest bus cycles using pipelined address require only two bus states, named T1P and T2P. Therefore, cycles with pipelined address timing allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased compared to that of a non-pipelined cycle.



Fastest pipelined bus cycles consist of T1P and T2P

15021B–052

Figure 49. Fastest Read Cycles with Pipelined Address Timing

By increasing the address-to-data access time, pipelined address timing reduces wait state requirements. For example, if one wait state is required with non-pipelined address timing, no wait states would be required with pipelined address.

Pipelined address timing is useful in typical systems having address latches. In those systems, once an address has been latched, pipelined availability of the next address allows decoding circuitry to generate chip selects (and other necessary select signals) in advance, so selected devices are accessed immediately when the next cycle begins. In other words, the decode time for the next cycle can be overlapped with the end of the current cycle.

If a system contains a memory structure of two or more interleaved memory banks, pipelined address timing potentially allows even more overlap of activity. This is true when the interleaved memory controller is designed to allow the next memory operation to begin in one memory bank while the current bus cycle is still activating another memory bank. Figure 50 shows the general structure of the Am386DXL microprocessor with two-bank and four-bank interleaved memory. Note each memory bank of the interleaved memory has full data bus width (32-bit data width typically, unless 16-bit bus size is selected).

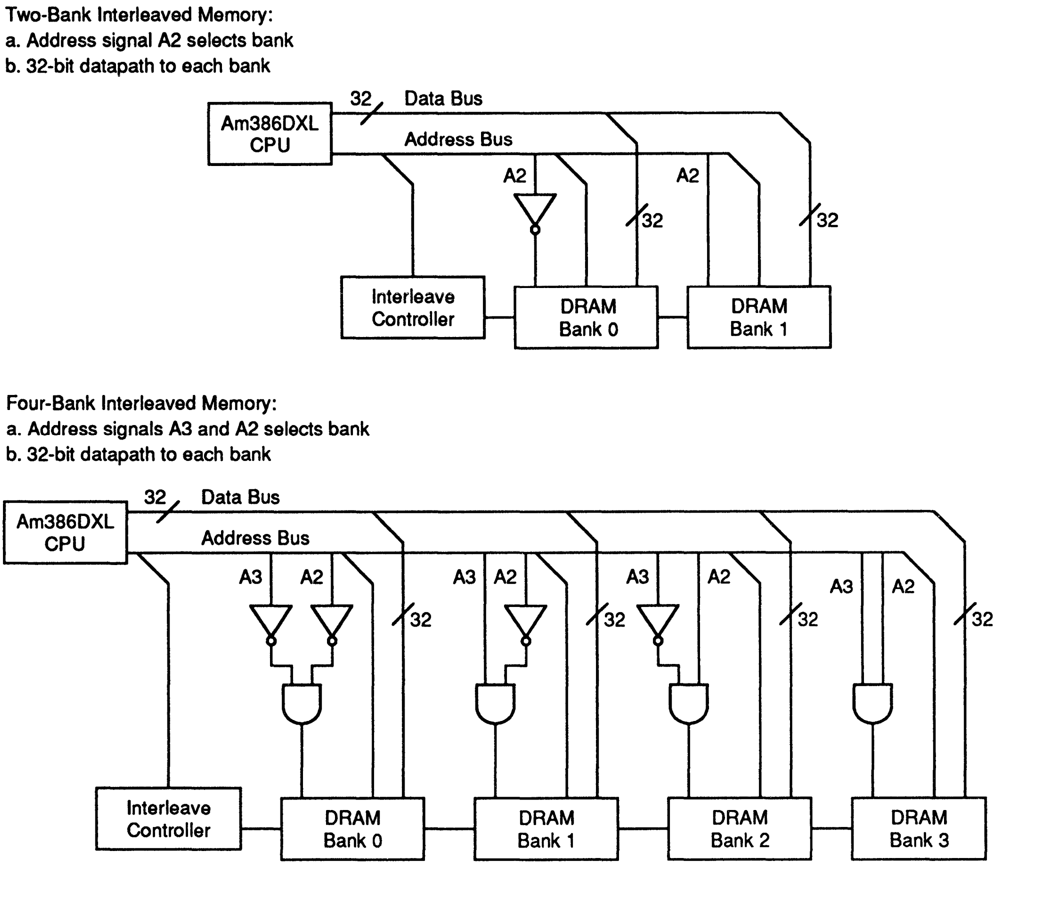
Further details of pipelined address timing are given in Pipelined Address; Initiating and Maintaining Pipelined Address; Pipelined Address with Dynamic Bus Sizing; and, Maximum Pipelined Address Usage With 16-bit Bus Size.

Read and Write Cycles

Introduction

Data transfers occur as a result of bus cycles, classified as Read or Write cycles. During Read cycles, data is transferred from an external device to the processor. During Write cycles, data is transferred in the other direction, from the processor to an external device.

Two choices of address timing are dynamically selectable: non-pipelined or pipelined. After a bus idle state, the processor always uses non-pipelined address timing. However, the NA (Next Address) input may be asserted to select pipelined address timing for the next bus cycle. When pipelining is selected and the Am386DXL microprocessor has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by READY. Generally, the NA input is sampled each bus cycle to select the desired address timing for the next bus cycle.



Two-Bank Interleaved Memory:
a. Address signal A2 selects bank
b. 32-bit datapath to each bank

Four-Bank Interleaved Memory:
a. Address signals A3 and A2 selects bank
b. 32-bit datapath to each bank

Figure 50. Two-Bank and Four-Bank Interleaved Memory Structure

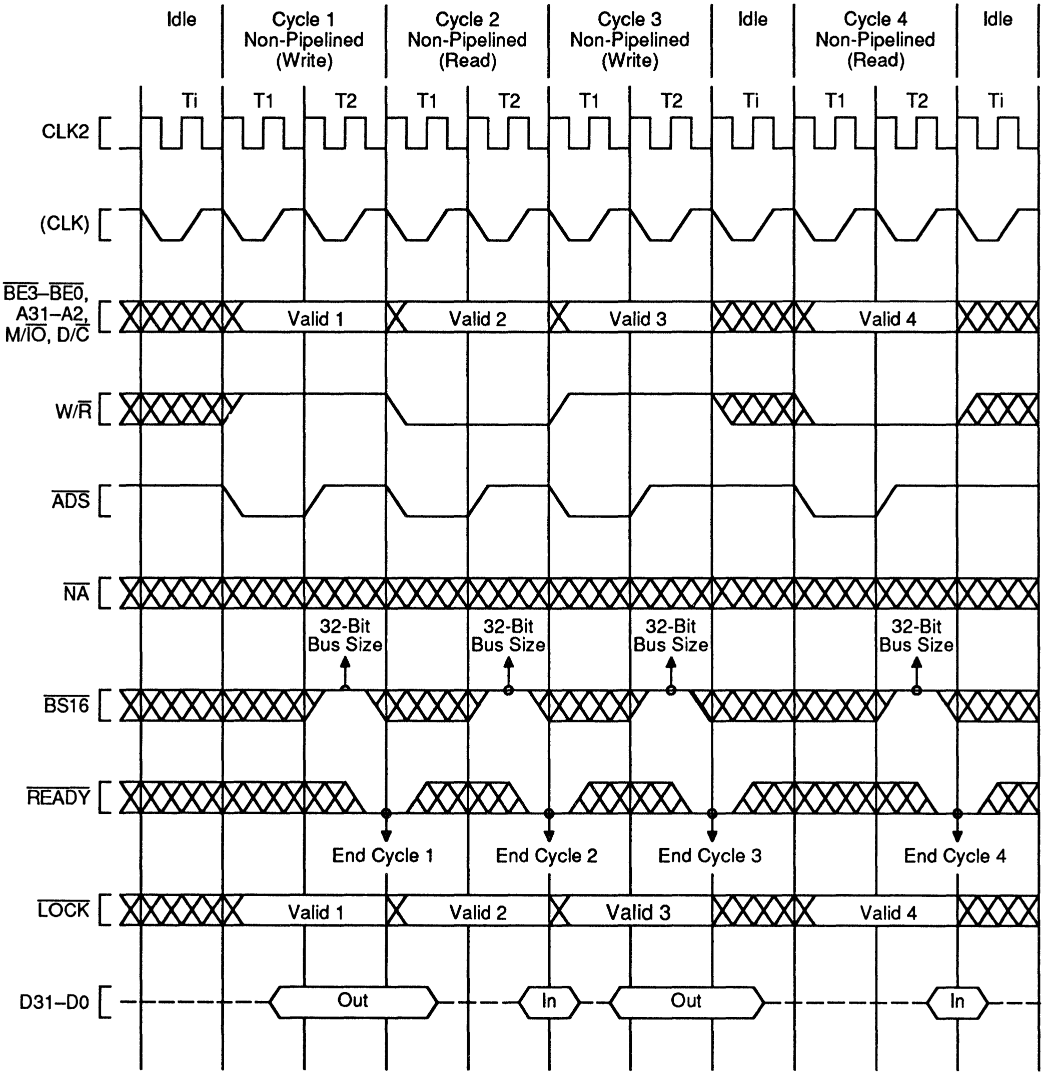
Two choices of physical data bus width are dynamically selectable: 32 bits or 16 bits. Generally, the BS16 (Bus Size 16) input is sampled near the end of the bus cycle to confirm the physical data bus size applicable to the current cycle. Negation of BS16 indicates a 32-bit size and assertion indicates a 16-bit bus size.

If 16-bit bus size is indicated, the Am386DXL CPU automatically responds as required to complete the transfer on a 16-bit data bus. Depending on the size and alignment of the operand, another 16-bit bus cycle may be required. Table 19 provides all details. When necessary, the Am386DXL microprocessor performs an additional 16-bit bus cycle, using D15–D0 in place of D31–D16.

Terminating a Read cycle or Write cycle, like any bus cycle, requires acknowledging the cycle by asserting the READY input. Until acknowledged, the processor inserts wait states into the bus cycle to allow adjustment for the speed of any external device. External hardware, that has decoded the address and bus cycle type asserts the READY input at the appropriate time.

At the end of the second bus state within the bus cycle, READY is sampled. At that time, if external hardware acknowledges the bus cycle by asserting READY, the bus cycle terminates as shown in Figure 51. If READY is negated as in Figure 52, the cycle continues another bus state (a wait state) and READY is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by READY asserted.

When the current cycle is acknowledged, the Am386DXL microprocessor terminates it. When a Read cycle is acknowledged, the Am386DXL CPU latches the information present at its data pins. When a Write cycle is acknowledged, the Am386DXL CPU write data remains valid throughout phase one of the next bus state to provide write data hold time.



Note: Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state. An active bus cycle can immediately follow the write cycle.

15021B–054

Figure 51. Various Bus Cycles and Idle States with Non-Pipelined Address (Zero Wait States)

Non-Pipelined Address

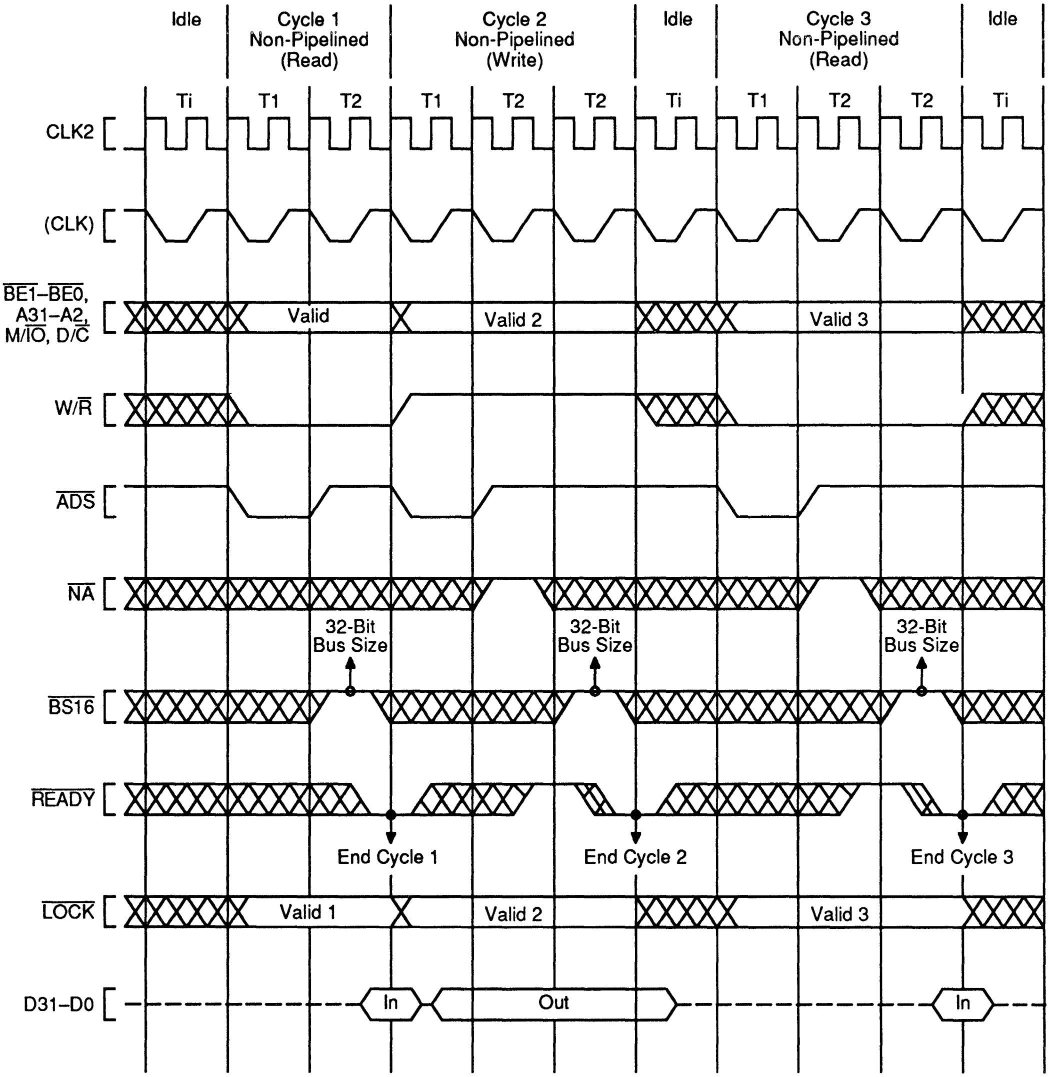
Any bus cycle may be performed with non-pipelined address timing. For example, Figure 51 shows a mixture of Read and Write cycles with non-pipelined address timing. Figure 51 shows that the fastest possible cycles with non-pipelined address have two bus states per bus cycle. The states are named T1 and T2. In phase one of the T1, the address signals and bus cycle definition signals are driven valid, and to signal their availability, address status (ADS) is simultaneously asserted.

During Read or Write cycles, the data bus behaves as follows. If the cycle is a read, the Am386DXL microprocessor floats its data signals to allow driving by the external device being addressed. The Am386DXL device requires that all data bus pins be at a valid logic state (High or Low) at the end of each read cycle, when READY is asserted, even if all byte enables are not asserted. The system **must** be designed to meet this requirement. If the cycle is a write, data signals are driven by the Am386DXL device beginning in phase two of T1 until phase one of the bus state following cycle acknowledgment.

Figure 52 illustrates non-pipelined bus cycles with one wait added to Cycles 2 and 3. READY is sampled negated at the end of the first T2 in Cycles 2 and 3. Therefore, Cycles 2 and 3 have T2 repeated. At the end of the second T2, READY is sampled asserted.

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states.

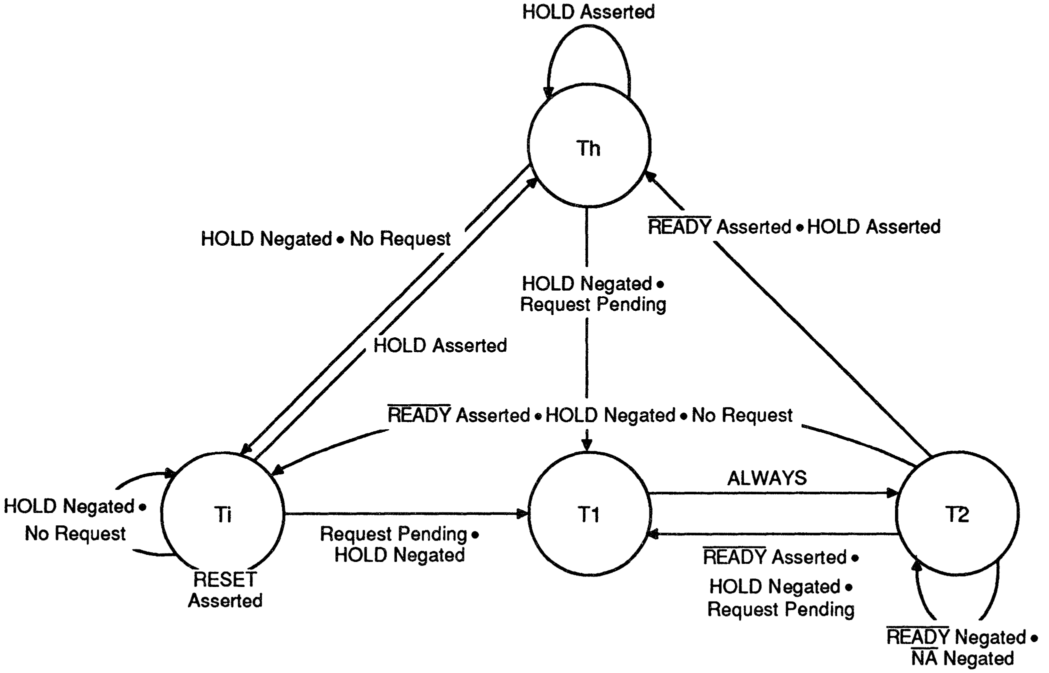
When wait states are added and you desire to maintain non-pipelined address timing, it is necessary to negate NA during each T2 state except the last one, as shown in Figure 52 Cycles 2 and 3. If NA is sampled asserted during a T2 other than the last one, the next state would be T2I (for pipelined address) or T2P (for pipelined address) instead of another T2 (for non-pipelined address).



Note: Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state. An active bus cycle can immediately follow the Write cycle.

15021B–055

Figure 52. Various Bus Cycles and Idle States with Non-Pipelined Address (Various Number of Wait States)


HOLD Asserted
Th
HOLD Negated • No Request
$\overline{\text{READY}}$ Asserted • HOLD Asserted
HOLD Negated •
Request Pending
HOLD Asserted
$\overline{\text{READY}}$ Asserted • HOLD Negated • No Request
ALWAYS
HOLD Negated •
No Request
Ti
T1
T2
Request Pending •
HOLD Negated
$\overline{\text{READY}}$ Asserted •
HOLD Negated •
Request Pending
RESET
Asserted
$\overline{\text{READY}}$ Negated •
$\overline{\text{NA}}$ Negated

Bus States:
T1— First clock of a non-pipelined bus cycle (Am386DXL microprocessor drives new address and asserts $\overline{\text{ADS}}$).
T2— Subsequent clocks of a bus cycle when $\overline{\text{NA}}$ has not been sampled asserted in the current bus cycle.
Ti — Idle state.
Th— Hold Acknowledge state (Am386DXL microprocessor asserts HLDA).

15022B–017

The fastest bus cycle consists of two states: T1 and T2.

Four basic bus states describe bus operation when not using pipelined address. These states do include $\overline{\text{BS16}}$ usage for 32-bit and 16-bit bus size. If asserting $\overline{\text{BS16}}$ requires second 16-bit bus cycle to be performed, it is performed before HOLD asserted acknowledged.

Figure 53. Bus States (Not Using Pipelined Address)

Figure 53 illustrates the bus states and transitions when address pipelining is not used. The bus transitions between four possible states: T1, T2, Ti, and Th. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise, the bus may be idle in the Ti state, or in hold acknowledge, the Th state.

When address pipelining is not used, the bus state diagram is as shown in Figure 53. When the bus is idle, it is in state Ti. Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and $\overline{\text{NA}}$ is negated, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle if a bus request is pending internally, or Ti if there is no bus request pending, or Th if the HOLD input is being asserted.

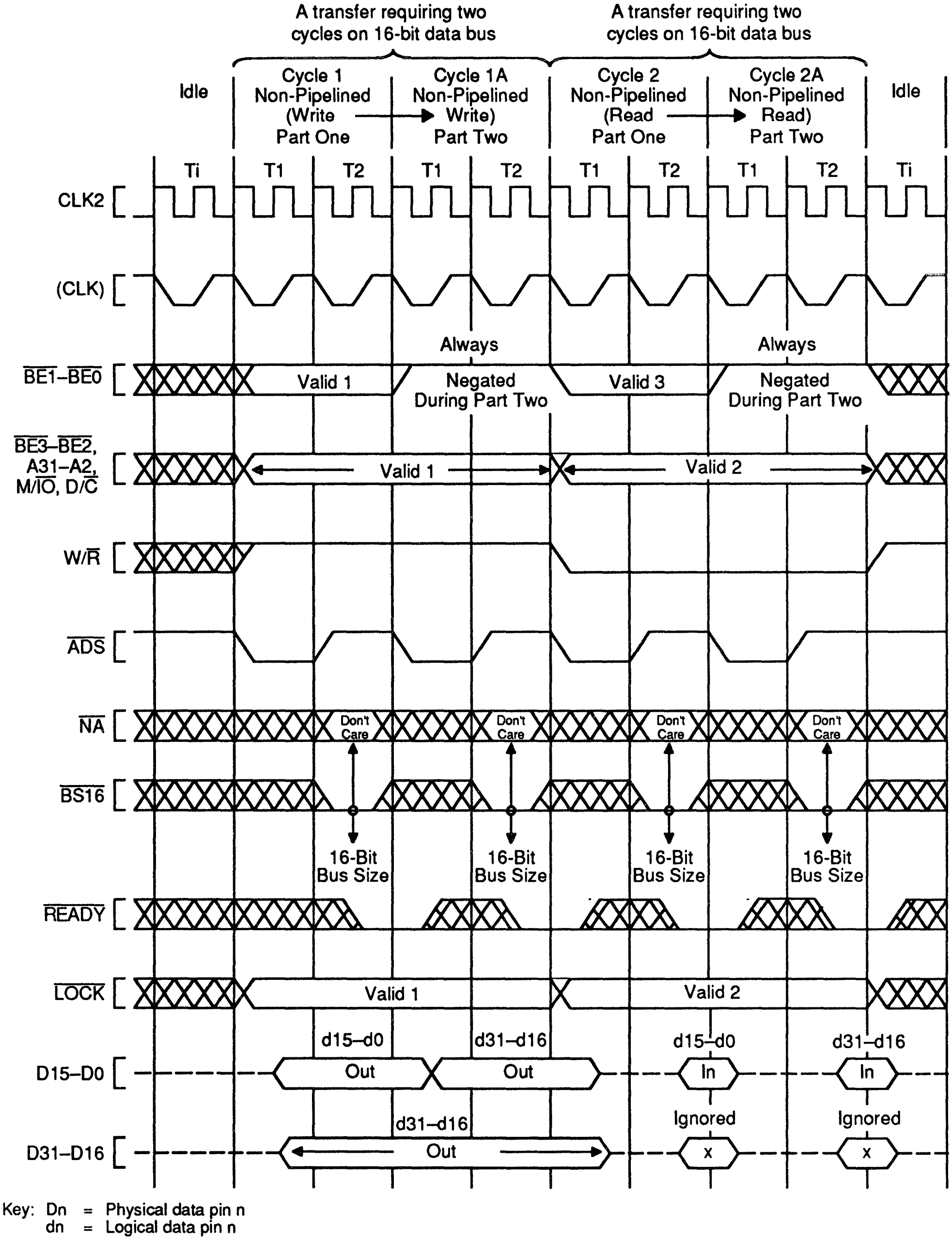
The bus state diagram in Figure 53 also applies to the use of $\overline{\text{BS16}}$. If the Am386DXL microprocessor makes internal adjustments for 16-bit bus size, the adjustments do not affect the external bus states. If an additional 16-bit bus cycle is required to complete a transfer on a 16-bit bus, it also follows the state transitions shown in Figure 53.

Use of pipelined address allows the Am386DXL CPU to enter three additional bus states not shown in Figure 53. Figure 59 in Pipelined Address is the complete bus state diagram, including pipelined address cycles.

Non-Pipelined Address With Dynamic Data Bus Sizing

The physical data bus width for any non-pipelined bus cycle can be either 32 or 16 bits. At the beginning of the bus cycle, the processor behaves as if the data bus is 32-bits wide. When the bus cycle is acknowledged by asserting $\overline{\text{READY}}$ at the end of a T2 state, the most recent sampling of $\overline{\text{BS16}}$ determines the data bus size for the cycle being acknowledged. If $\overline{\text{BS16}}$ was most recently negated, the physical data bus size is defined as 32 bits. If $\overline{\text{BS16}}$ was most recently asserted, the size is defined as 16 bits.

When $\overline{\text{BS16}}$ is asserted and two 16-bit bus cycles are required to complete the transfer, $\overline{\text{BS16}}$ must be asserted during the second cycle; 16-bit bus size is not assumed. Like any bus cycle, the second 16-bit cycle must be acknowledged by asserting $\overline{\text{READY}}$.


A transfer requiring two cycles on 16-bit data bus
A transfer requiring two cycles on 16-bit data bus
Idle
Cycle 1 Non-Pipelined (Write Part One
Cycle 1A Non-Pipelined Write) Part Two
Cycle 2 Non-Pipelined (Read Part One
Cycle 2A Non-Pipelined Read) Part Two
Idle
Ti
T1
T2
CLK2
(CLK)
Always
BE1–BE0
Valid 1
Negated During Part Two
Valid 3
BE3–BE2, A31–A2, M/IO, D/C
Valid 2
W/R
ADS
NA
Don't Care
BS16
16-Bit Bus Size
READY
LOCK
d15–d0
d31–d16
D15–D0
Out
In
Ignored
D31–D16
x
Key: Dn = Physical data pin n
dn = Logical data pin n

15021B–057

Figure 54. Asserting BS16 (Zero-Wait-States, Non-Pipelined Address)

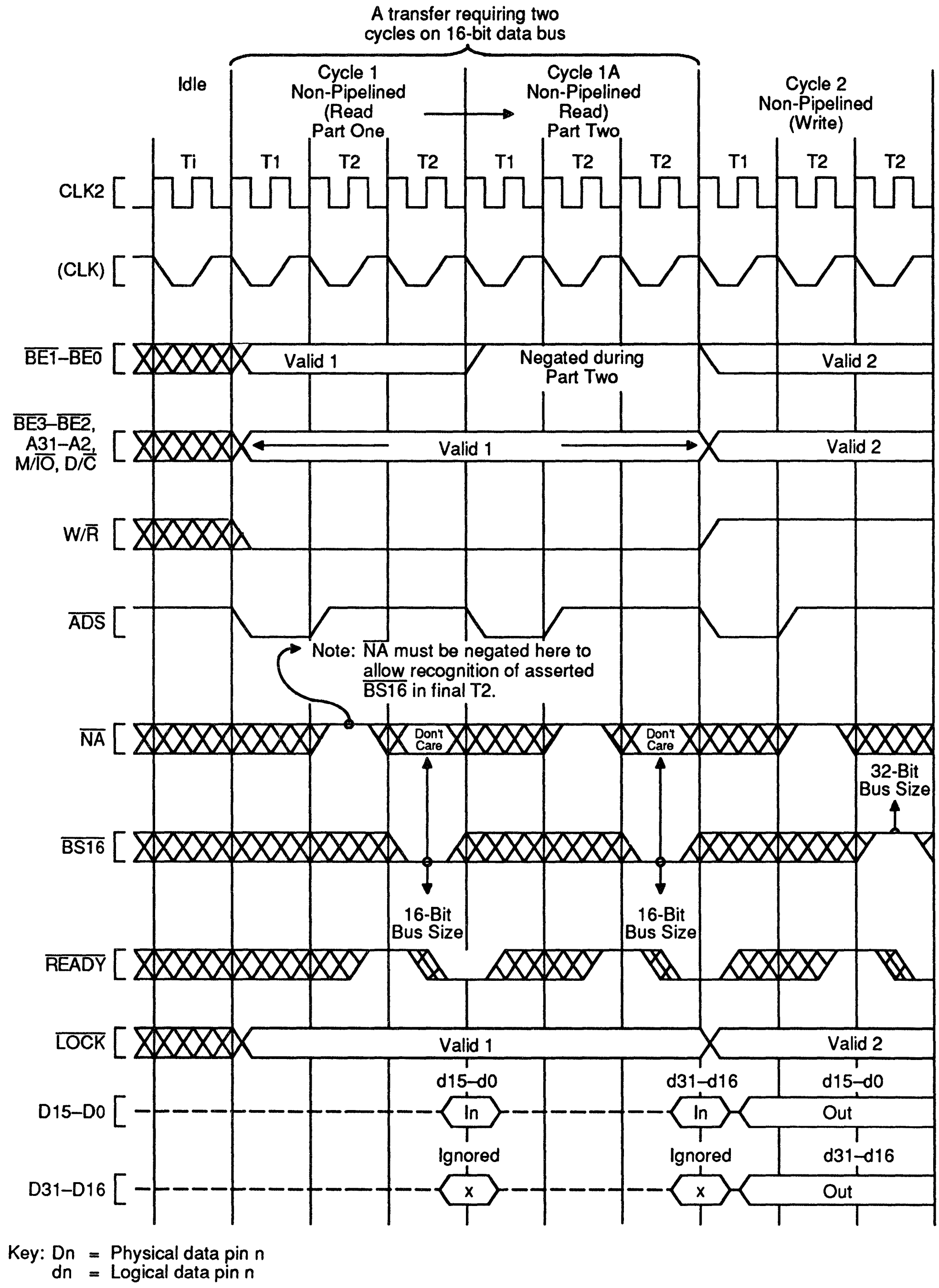


Figure 55. Asserting BS16 (One-Wait-State, Non-Pipelined Address)

When a second 16-bit bus cycle is required to complete the transfer over a 16-bit bus, the addresses generated for the two 16-bit bus cycles are closely related to each other. The addresses are the same, except BE0 and BE1 are always negated for the second cycle. This is because data on D15–D0 was already transferred during the first 16-bit cycle.

Figures 54 and 55 show cases where assertion of BS16 requires a second 16-bit cycle for complete operand transfer. Figure 54 illustrates cycles without wait states. Figure 55 illustrates cycles with one wait state. In Figure 55 Cycle 1, the bus cycle during which BS16 is asserted, note that NA must be negated in the T2 state(s) prior to the last T2 state. This is to allow the recognition of BS16 asserted in the final T2 state. The relation of NA and BS16 is given fully in Pipelined Address, but Figure 55 illustrates this only precaution you need to know when using BS16 with non-pipelined address.

Pipelined Address

Address pipelining is the option of requesting the address and the bus cycle definition of the next internally pending bus cycle before the current bus cycle is acknowledged with READY asserted. ADS is asserted by the Am386DXL microprocessor when the next address is issued. The address pipelining option is controlled on a cycle-by-cycle basis with the NA input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the NA input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles, therefore, NA is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 56, during which NA is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).

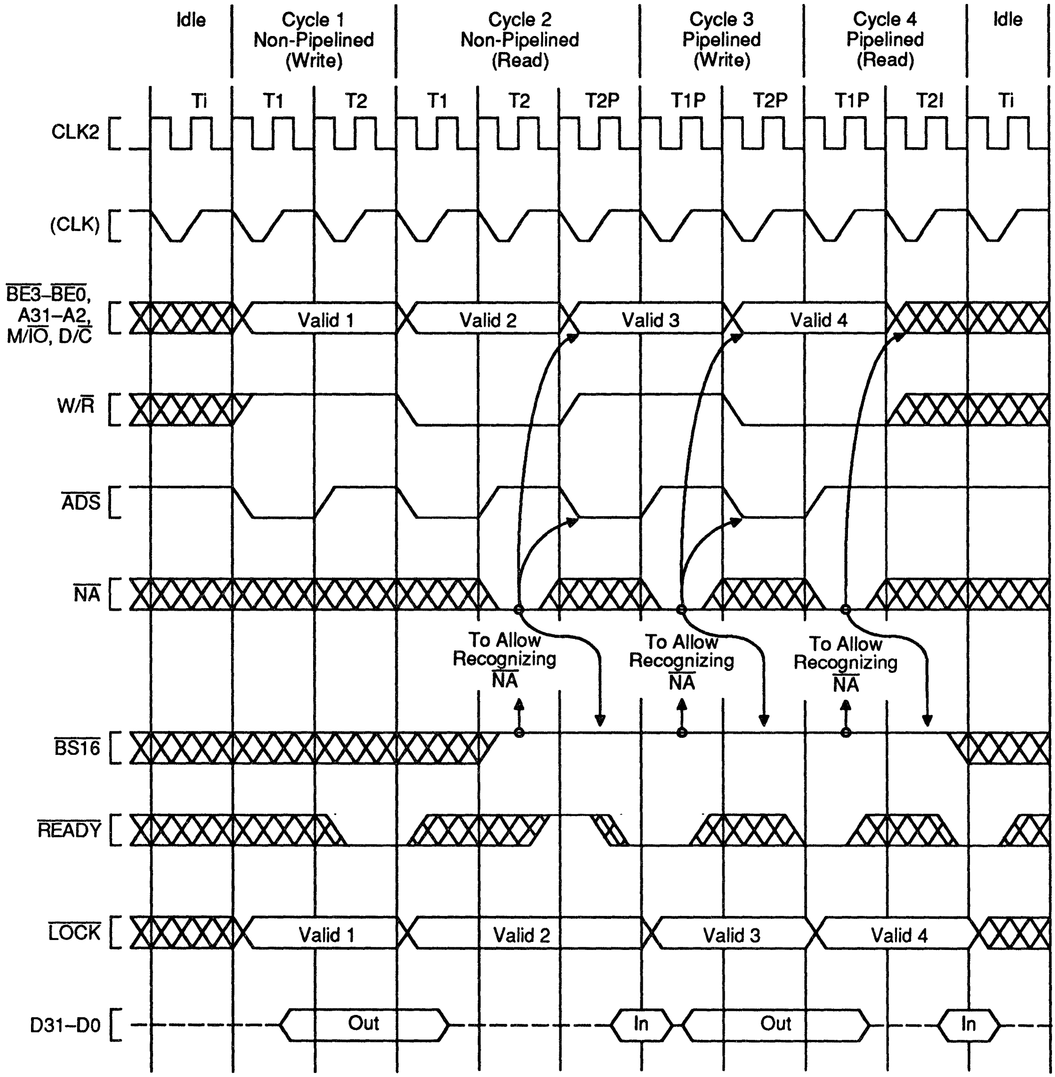
If NA is sampled asserted, the Am386DXL microprocessor is free to drive the address and bus cycle definition of the next bus cycle, and assert ADS, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of address pipelining, the Am386DXL CPU has the following characteristics.

1. For NA to be sampled asserted, BS16 must be negated at the sampling window (see Figure 56 Cycles 2 through 4, and Figure 57 Cycles 1 through 4). If NA and BS16 are both sampled asserted during the last T2 period of a bus cycle, BS16 asserted has priority. Therefore, if both are asserted, the current bus size is taken to be 16 bits and the next address is not pipelined.
2. The next address may appear as early as the bus state after NA was sampled asserted (see Figure 56 or 57). In that case, state T2P is entered immediately, However, when there is not an internal bus request already pending, the next address will not be available immediately after NA is asserted and T2I is entered instead of T2P (see Figure 58 Cycle 3). Provided the current bus cycle is not yet acknowledged by READY asserted, T2P will be entered as soon as the Am386DXL microprocessor does drive the next address. External hardware should therefore observe the ADS output as confirmation the next address is actually being driven on the bus.
3. Once NA is sampled asserted, the Am386DXL microprocessor commits itself to the highest priority bus request that is pending internally. It can no longer perform another 16-bit transfer to the same address should BS16 be asserted externally, so thereafter must assume the current bus size is 32 bits. Therefore, if NA is sampled asserted within a bus cycle, BS16 must be negated thereafter in that bus cycle (see Figures 56, 57, 58). Consequently, do not assert NA during bus cycles that must have BS16 driven asserted. See Dynamic Bus Sizing with Pipelined Address.
4. Any address which is validated by a pulse on the Am386DXL CPU ADS output will remain stable on the address pins for at least two processor clock periods. The Am386DXL microprocessor can not produce a new address more frequently than every two processor clock periods (see Figures 56, 57, 58).
5. Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 58 Cycle 1).

The complete bus state transition diagram, including operation with pipelined address is given by Figure 59. Note it is a superset of the diagram for non-pipelined address only and the three additional bus states for pipelined address are drawn in bold.

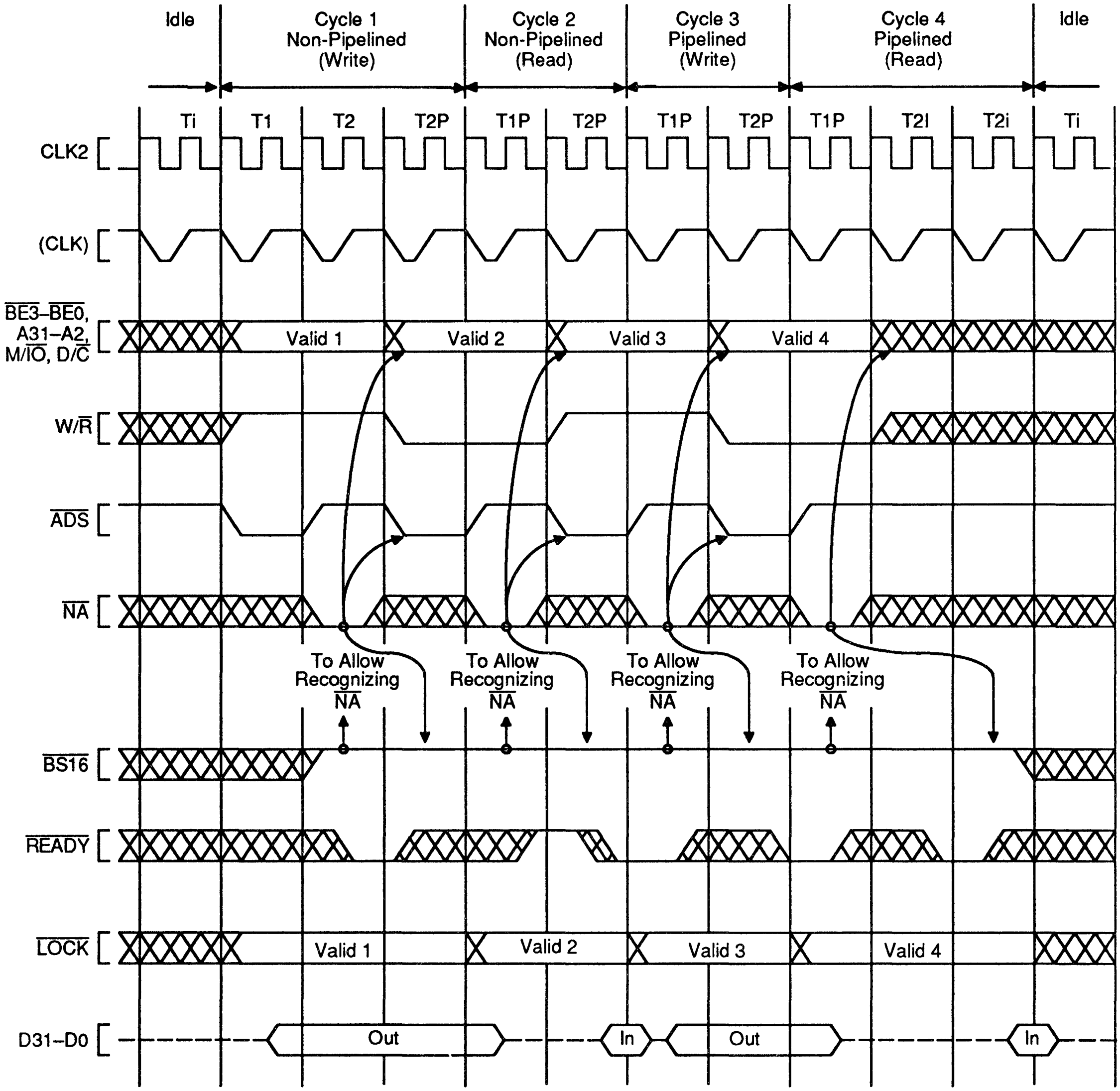
The fastest bus cycle with pipelined address consists of just two bus states, T1P and T2P (recall for non-pipelined address it is T1 and T2). T1P is the first bus state of a pipelined cycle.



Note: Following any idle bus state (Ti), addresses are non-pipelined. Within non-pipelined bus cycles, NA is only sampled during wait states. Therefore, to begin address pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cycle 2 above).

15021B–059

Figure 56. Transitioning to Pipelined Address During Burst of Bus Cycles



Note: Following any idle bus state (Ti) the address is always non-pipelined and NA is only sampled during wait states. To start address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (Cycle 1 above). The pipelined cycles (2, 3, 4 above) are shown with various numbers of wait states.

15021B–060

Figure 57. Fastest Transition to Pipelined Address Following Idle Bus State

Initiating and Maintaining Pipelined Address

Using the state diagram Figure 59, observe the transitions from an idle state, Ti, to the beginning of a pipelined bus cycle, T1P. From an idle state Ti, the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided $\overline{\text{NA}}$ is asserted and the first bus cycle ends in a T2P state (the address for the next bus cycle is driven during T2P). The fastest path from an idle state to a bus cycle with pipelined address is shown in below:

$$\underbrace{\text{Ti, Ti, Ti}}_{\substack{\text{Idle}\\\text{States}}} \qquad \underbrace{\text{T1-T2-T2P}}_{\substack{\text{Non-Pipelined}\\\text{Cycle}}} \qquad \underbrace{\text{T1P-T2P}}_{\substack{\text{Pipelined}\\\text{Cycle}}}$$

T1-T2-T2P are the states of the bus cycle that establishes address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:

$$\underbrace{\text{Th, Th, Th}}_{\substack{\text{Hold}\\\text{Acknowledge}\\\text{States}}} \qquad \underbrace{\text{T1-T2-T2P}}_{\substack{\text{Non-Pipelined}\\\text{Cycle}}} \qquad \underbrace{\text{T1P-T2P}}_{\substack{\text{Pipelined}\\\text{Cycle}}}$$

The transition to pipelined address is shown functionally by Figure 57 Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3, and 4 that are pipelined. The $\overline{\text{NA}}$ input is asserted at the appropriate time to select address pipelining for Cycles 2, 3, and 4.

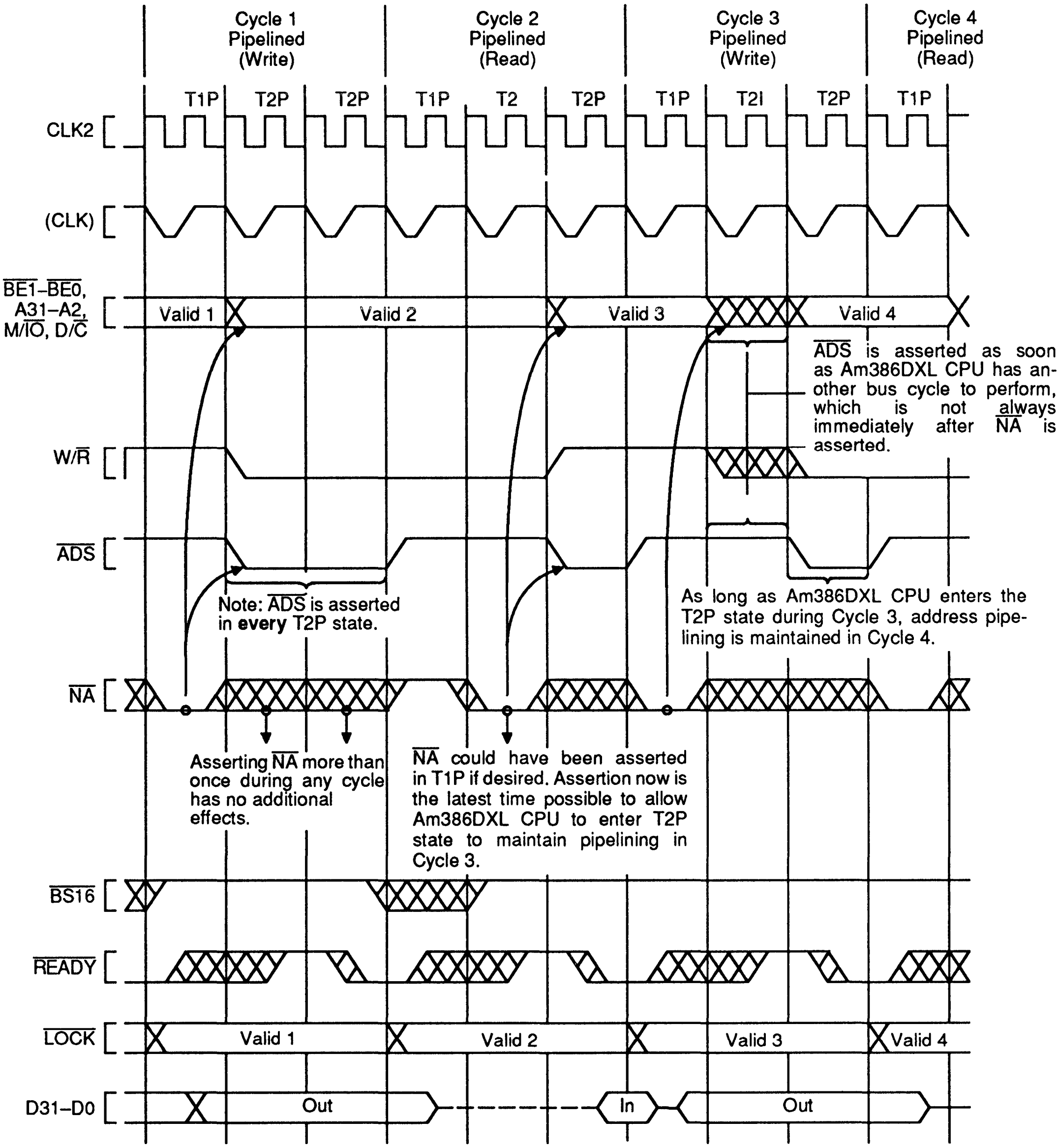
Once a bus cycle is in progress and the current address has become valid, the $\overline{\text{NA}}$ input is sampled at the end of every phase one, beginning with the next bus state, until the bus cycle is acknowledged. During Figure 57 Cycle 1 therefore, sampling begins in T2. Once $\overline{\text{NA}}$ is sampled asserted during the current cycle, the Am386DXL microprocessor is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 56 Cycle 1 for example, the next address is driven during state T2P. Thus, Cycle 1 makes the transition to pipelined address timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as $\overline{\text{READY}}$ asserted terminates Cycle 1.

Example transition bus cycles are Figure 57 Cycle 1 and Figure 56 Cycle 2. Figure 57 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 56 Cycle 2, shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 (you assert $\overline{\text{NA}}$ at that time), and T2P (provided the Am386DXL microprocessor has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note three states (T1, T2, and T2P) are only required in a bus cycle performing a transition from non-pipelined address into pipelined address timing; for example, Figure 57 Cycle 1. Figure 57 Cycles 2, 3, and 4 show that address pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting $\overline{\text{NA}}$ and detecting that the Am386DXL CPU enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of $\overline{\text{ADS}}$. Figures 56 and 57 however, show pipelining ending after Cycle 4, because Cycle 4 ends in T2P. This indicates the Am386DXL CPU did not have an internal bus request prior to the acknowledgment of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

Realistically, address pipelining is almost always maintained as long as $\overline{\text{NA}}$ is sampled asserted. This is so, because in the absence of any other request a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore, address pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD negated) and $\overline{\text{NA}}$ is sampled asserted in each of the bus cycles.



ADS is asserted as soon as Am386DXL CPU has another bus cycle to perform, which is not always immediately after NA is asserted.

Note: ADS is asserted in **every** T2P state.

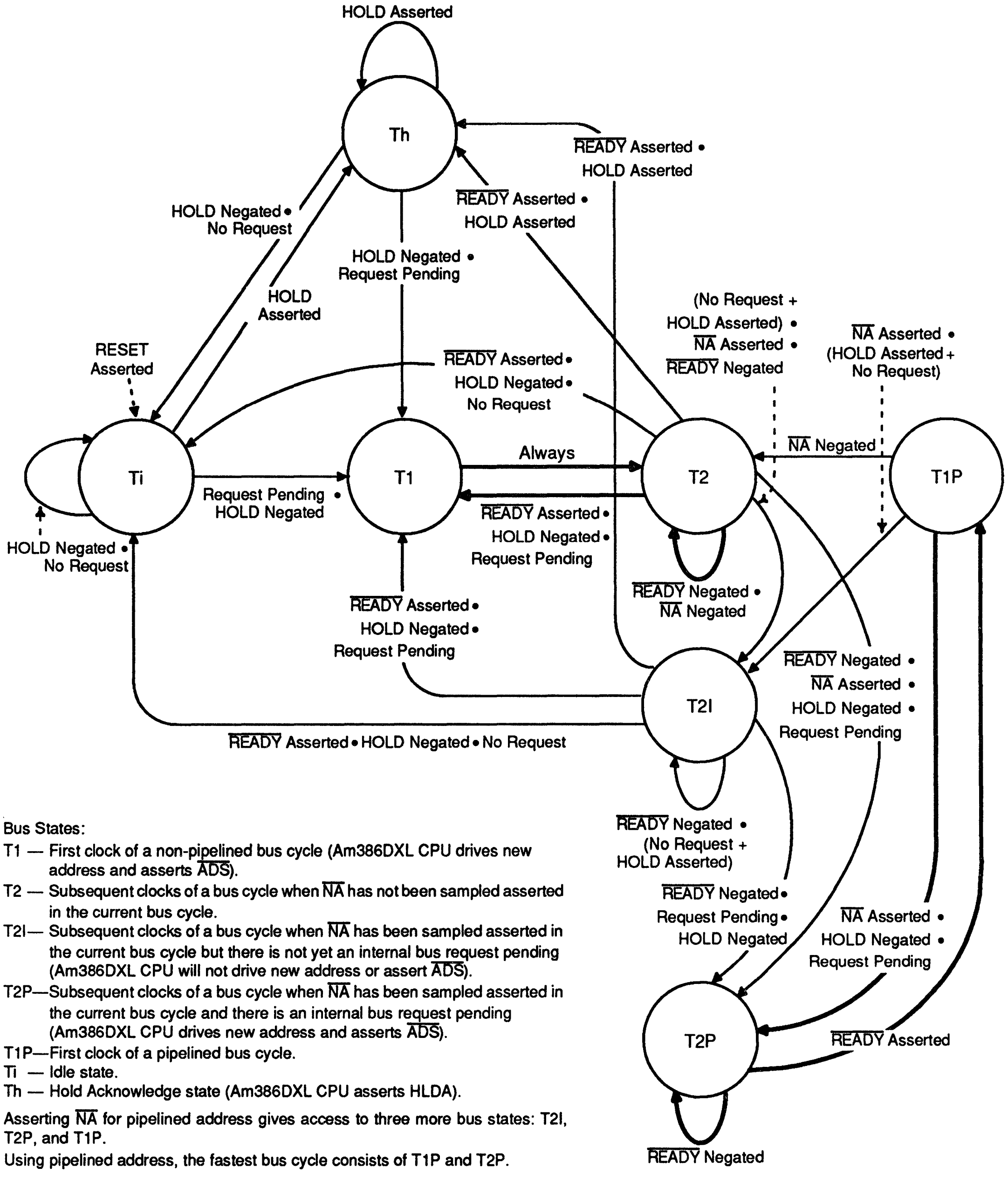
As long as Am386DXL CPU enters the T2P state during Cycle 3, address pipelining is maintained in Cycle 4.

Asserting NA more than once during any cycle has no additional effects.

NA could have been asserted in T1P if desired. Assertion now is the latest time possible to allow Am386DXL CPU to enter T2P state to maintain pipelining in Cycle 3.

15021B–061

Figure 58. Details of Address Pipelining During Cycles with Wait States


HOLD Asserted
Th
$\overline{\text{READY}}$ Asserted • HOLD Asserted
HOLD Negated • No Request
$\overline{\text{READY}}$ Asserted • HOLD Asserted
HOLD Negated • Request Pending
HOLD Asserted
(No Request + HOLD Asserted) • $\overline{\text{NA}}$ Asserted • $\overline{\text{READY}}$ Negated
$\overline{\text{NA}}$ Asserted • (HOLD Asserted + No Request)
RESET Asserted
$\overline{\text{READY}}$ Asserted • HOLD Negated • No Request
Always
$\overline{\text{NA}}$ Negated
Ti
T1
T2
T1P
Request Pending • HOLD Negated
$\overline{\text{READY}}$ Asserted • HOLD Negated • Request Pending
HOLD Negated • No Request
$\overline{\text{READY}}$ Negated • $\overline{\text{NA}}$ Negated
$\overline{\text{READY}}$ Asserted • HOLD Negated • Request Pending
$\overline{\text{READY}}$ Negated • $\overline{\text{NA}}$ Asserted • HOLD Negated • Request Pending
T2I
$\overline{\text{READY}}$ Asserted • HOLD Negated • No Request
$\overline{\text{READY}}$ Negated • (No Request + HOLD Asserted)
$\overline{\text{READY}}$ Negated • Request Pending • HOLD Negated
$\overline{\text{NA}}$ Asserted • HOLD Negated • Request Pending
T2P
$\overline{\text{READY}}$ Asserted
$\overline{\text{READY}}$ Negated

Bus States:

T1 — First clock of a non-pipelined bus cycle (Am386DXL CPU drives new address and asserts $\overline{\text{ADS}}$).

T2 — Subsequent clocks of a bus cycle when $\overline{\text{NA}}$ has not been sampled asserted in the current bus cycle.

T2I— Subsequent clocks of a bus cycle when $\overline{\text{NA}}$ has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (Am386DXL CPU will not drive new address or assert $\overline{\text{ADS}}$).

T2P—Subsequent clocks of a bus cycle when $\overline{\text{NA}}$ has been sampled asserted in the current bus cycle and there is an internal bus request pending (Am386DXL CPU drives new address and asserts $\overline{\text{ADS}}$).

T1P—First clock of a pipelined bus cycle.

Ti — Idle state.

Th — Hold Acknowledge state (Am386DXL CPU asserts HLDA).

Asserting $\overline{\text{NA}}$ for pipelined address gives access to three more bus states: T2I, T2P, and T1P.

Using pipelined address, the fastest bus cycle consists of T1P and T2P.

15021B–062

Figure 59. Am386DXL Microprocessor Complete Bus States (Including Pipelined Address)

Pipelined Address With Dynamic Data Bus Sizing

The $\overline{\text{BS16}}$ feature allows easy interface to 16-bit data buses. When asserted, the Am386DXL microprocessor bus interface hardware performs appropriate action to make the transfer using a 16-bit data bus connected on D15–D0.

There is a degree of interaction, however, between the use of Address Pipelining and the use of Bus Size 16. The interaction results from the multiple bus cycles required when transferring 32-bit operands over a 16-bit bus. If the operand requires both 16-bit halves of the 32-bit bus, the appropriate Am386DXL microprocessor action is a second bus cycle to complete the operand's transfer. It is this necessity that conflicts with $\overline{\text{NA}}$ usage.

When $\overline{\text{NA}}$ is sampled asserted, the Am386DXL microprocessor commits itself to perform the next internally pending bus request, and is allowed to drive the next internally pending address onto the bus. Asserting $\overline{\text{NA}}$ therefore makes it impossible for the next bus cycle to again access the current address on A31–A2, such as may be required when $\overline{\text{BS16}}$ is asserted by the external hardware.

To avoid conflict, the Am386DXL microprocessor is designed with following two provisions.

1. To avoid conflict, $\overline{\text{BS16}}$ must be negated in the current bus cycle if $\overline{\text{NA}}$ has already been sampled asserted in the current cycle. If $\overline{\text{NA}}$ is sampled asserted, the current data bus size is assumed to be 32 bits.
2. Also to avoid conflict, if $\overline{\text{NA}}$ and $\overline{\text{BS16}}$ are both asserted during the same sampling window, $\overline{\text{BS16}}$ asserted has priority and the Am386DXL microprocessor acts as if $\overline{\text{NA}}$ was negated at that time.

Certain types of 16- or 8-bit operands require no adjustment for correct transfer on a 16-bit bus. Those are read or write operands using only the lower half of the data bus, and write operands using only the upper half of the bus, since the Am386DXL CPU simultaneously duplicates the write data on the lower half of the data bus. For these patterns of Byte Enables and the W/$\overline{\text{R}}$ signals, $\overline{\text{BS16}}$ need not be asserted at the Am386DXL CPU allowing $\overline{\text{NA}}$ to be asserted during the bus cycle if desired.

Interrupt Acknowledge (INTA) Cycles

In response to an interrupt request on the INTR input when interrupts are enabled, the Am386DXL microprocessor performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by $\overline{\text{READY}}$ sampled asserted.

The state of A2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A31–A3 Low, A2 High, $\overline{\text{BE3}}$–$\overline{\text{BE1}}$ High, and $\overline{\text{BE0}}$ Low). The address driven during the second interrupt acknowledge cycle is 0 (A31–A2 Low, $\overline{\text{BE3}}$–$\overline{\text{BE1}}$ High, $\overline{\text{BE0}}$ Low).

The $\overline{\text{LOCK}}$ output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states, Ti, are inserted by the Am386DXL microprocessor between the two interrupt acknowledge cycles, allowing for compatibility with spec TRHRL of the 8259A Interrupt Controller.

During both interrupt acknowledge cycles, D31–D0 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the Am386DXL microprocessor will read an external interrupt vector from D7–D0 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.

Halt Indication Cycle

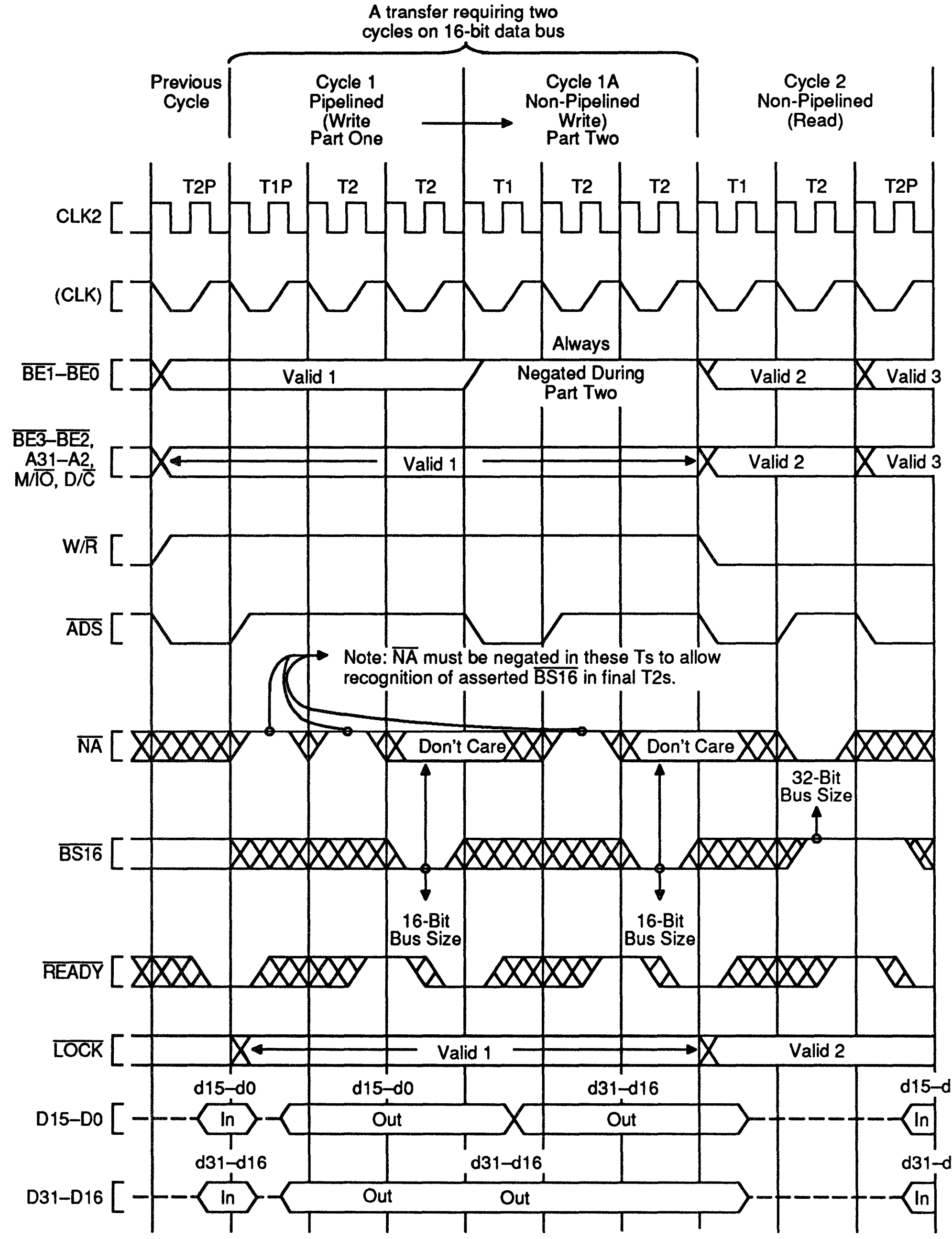
The Am386DXL microprocessor halts as a result of executing a HALT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals shown in Bus cycle Definition and a byte address of 2. $\overline{\text{BE0}}$ and $\overline{\text{BE2}}$ are the only signals distinguishing halt indication from shutdown indication, that drives an address of 0. During the halt cycle undefined data is driven on D31–D0. The halt indication cycle must be acknowledged by $\overline{\text{READY}}$ asserted.

A halted Am386DXL CPU resumes execution when INTR (if interrupts are enabled) or NMI or RESET is asserted.

Shutdown Indication Cycle

The Am386DXL microprocessor shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown in Bus Cycle Definition and a byte address of 0. $\overline{\text{BE0}}$ and $\overline{\text{BE2}}$ are the only signals distinguishing shutdown indication from halt indication, which drives an address of 2. During the shutdown cycle undefined data is driven on D31–D0. The shutdown indication cycle must be acknowledged by $\overline{\text{READY}}$ asserted.

A shutdown Am386DXL microprocessor resumes execution when NMI or RESET is asserted.

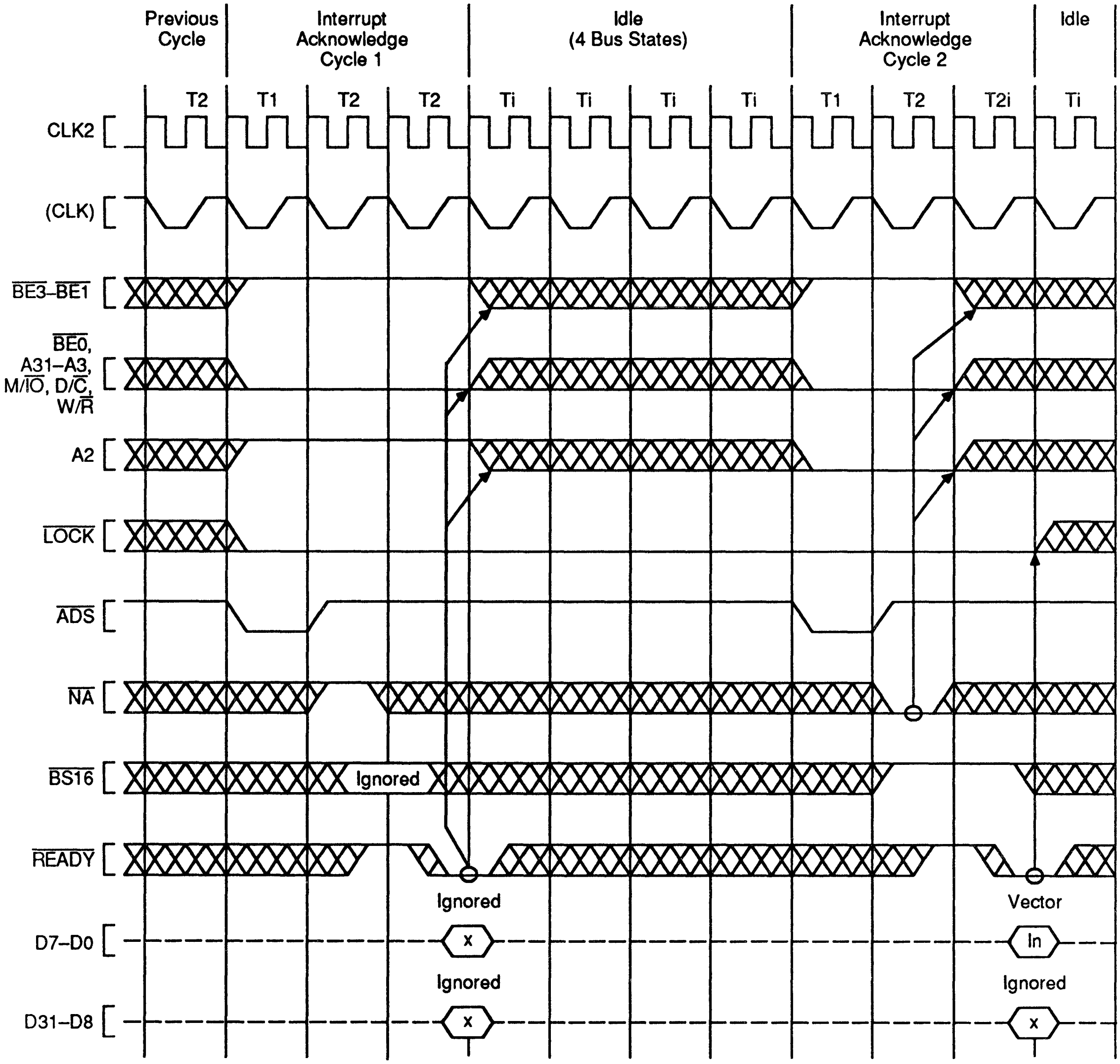

A transfer requiring two cycles on 16-bit data bus
Previous Cycle
Cycle 1 Pipelined (Write Part One
Cycle 1A Non-Pipelined Write) Part Two
Cycle 2 Non-Pipelined (Read)
T2P
T1P
T2
T2
T1
T2
T2
T1
T2
T2P
CLK2
(CLK)
BE1–BE0
Valid 1
Always Negated During Part Two
Valid 2
Valid 3
BE3–BE2, A31–A2, M/IO, D/C
Valid 1
Valid 2
Valid 3
W/R
ADS
Note: NA must be negated in these Ts to allow recognition of asserted BS16 in final T2s.
NA
Don't Care
Don't Care
32-Bit Bus Size
BS16
16-Bit Bus Size
16-Bit Bus Size
READY
LOCK
Valid 1
Valid 2
d15–d0
d15–d0
d31–d16
d15–d0
D15–D0
In
Out
Out
In
d31–d16
d31–d16
d31–d16
D31–D16
In
Out
Out
In

Key: Dn = Physical data pin n
dn = Logical data pin n

Cycle 1 is pipelined. Cycle 1A cannot be pipelined, but its address can be inferred from that of Cycle 1, to externally simulate address pipelining during Cycle 1A.

15021B–063

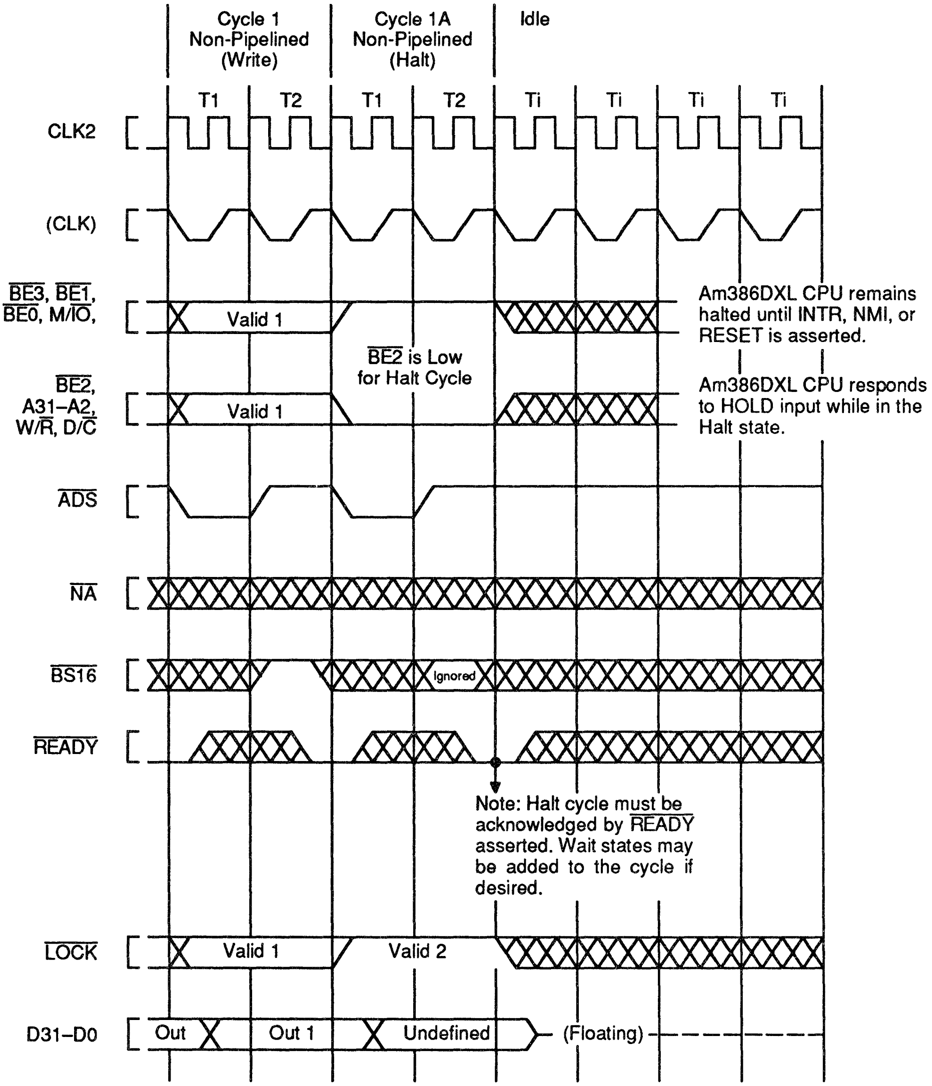
Figure 60. Using NA and BS16



Interrupt Vector (0–255) is read on D7–D0 at end of second Interrupt Acknowledge bus cycle.
Because each Interrupt Acknowledge bus cycle is followed by idle bus states, asserting NA has no practical effect. Choose the approach that is simplest for your system hardware design.

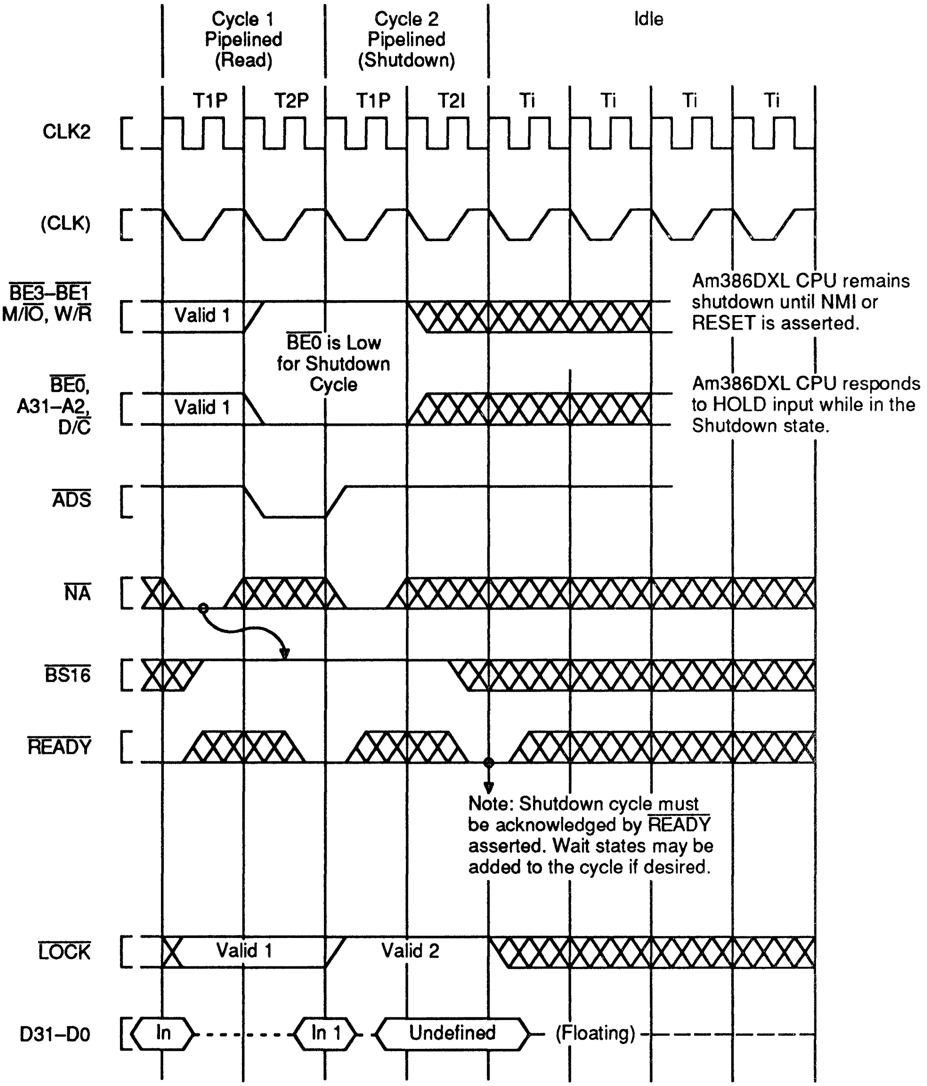
15021B–064

Figure 61. Interrupt Acknowledge Cycles



15021B–065

Figure 62. Halt Indication Cycle



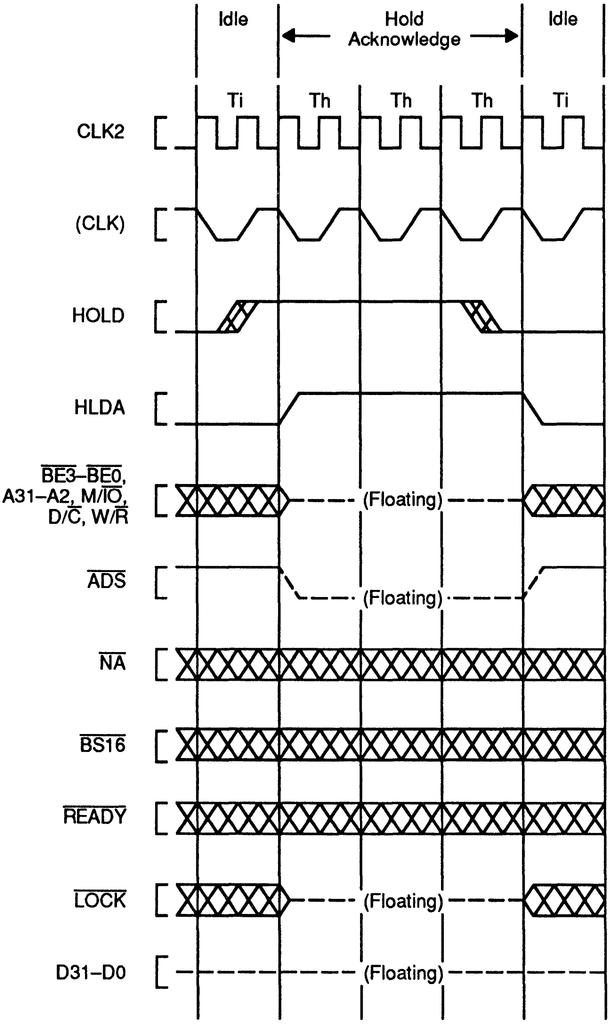
15021B–066

Figure 63. Shutdown Indication Cycle

Other Functional Descriptions

Entering and Exiting Hold Acknowledge

The Bus Hold Acknowledge State, Th, is entered in response to the HOLD input being asserted. In the Bus Hold Acknowledge state, the Am386DXL microprocessor floats all output or bi-directional signals, except for HLDA. HLDA is asserted as long as the Am386DXL CPU remains in the bus hold acknowledge state. In the Bus Hold Acknowledge state, all inputs except HOLD, $\overline{\text{FLT}}$, RESET, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$, and PEREQ are ignored (also up to one rising edge on NMI is remembered for processing when HOLD is no longer asserted).



Note: For maximum design flexibility the Am386DXL CPU has no internal pullup resistors on its outputs. Your design may require an external pullup on $\overline{\text{ADS}}$ and other Am386DXL CPU outputs to keep them negated during float periods.

15021B–067

Figure 64. Requesting Hold from Idle Bus

Th may be entered from a bus idle state, as in Figure 64, or after the acknowledgment of the current physical bus cycle if the LOCK signal is not asserted, as in Figures 65 and 66. If HOLD is asserted during a locked bus cycle, the Am386DXL microprocessor may execute one unlocked bus cycle before acknowledging HOLD. If asserting BS16 requires a second 16-bit bus cycle to complete a physical operand transfer, it is performed before HOLD is acknowledged, although the bus state diagrams in Figures 53 and 59 do not indicate that detail.

Th is exited in response to the HOLD input being negated. The following state will be Ti as in Figure 64 if no bus request is pending. The following bus state will be T1 if a bus request is internally pending, as in Figures 65 and 66.

Th is also exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in Th, the event is remembered as a non-maskable interrupt 2 and is serviced when Th is exited, unless of course, the Am386DXL microprocessor is reset before Th is exited.

RESET During HOLD Acknowledge

RESET being asserted takes priority over HOLD being asserted. Therefore, Th is exited in response to the RESET input being asserted. If RESET is asserted while HOLD remains asserted, the Am386DXL microprocessor drives its pins to defined states during reset, as in Table 15 Pin State During RESET, and performs internal reset activity as usual.

If HOLD remains asserted when RESET is negated, the Am386DXL microprocessor enters the hold acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the Am386DXL microprocessor would otherwise perform its first bus cycle. If HOLD remains asserted when RESET is negated, the BUSY input is still sampled as usual to determine whether a self test is being requested, and ERROR is still sampled as usual to determine whether a 387DX math coprocessor versus an 80287 (or none) is present.

Float

Activating the FLT input floats all Am386DXL CPU bi-directional and output signals, including HLDA. Asserting FLT isolates the Am386DXL CPU from the surrounding circuitry.

As the Am386DXL microprocessor is packaged in a surface mount PQFP, it cannot be removed from the motherboard when In-Circuit Emulation (ICE) is needed. The FLT input allows the Am386DXL CPU to be electrically isolated from the surrounding circuitry. This allows connection of an emulator to the Am386DXL microprocessor PQFP without removing it from the PCB. This method of emulation is referred to as ON-Circuit Emulation (ONCE).

Entering and Exiting Float

FLT is an asynchronous, active Low input. It is recognized on the rising edge of CLK2. When recognized, it aborts the current bus cycle and floats the outputs of the Am386DXL microprocessor (Figure 68). FLT must be held Low for a minimum of 16-CLK2 cycles. Reset should be asserted and held asserted until after FLT is deasserted. This will ensure that the Am386DXL CPU will exit Float in a valid state.

Asserting the FLT input unconditionally aborts the current bus cycle and forces the Am386DXL microprocessor into the Float mode. Since activating FLT unconditionally forces the Am386DXL CPU into Float mode, the Am386DXL CPU is not guaranteed to enter Float in a valid state. After deactivating FLT, the Am386DXL CPU is not guaraneeted to exit Float mode in a valid state. This is not a problem, as the FLT pin is meant to be used only during ONCE. After exiting Float, the Am386DXL CPU must be reset to return it to a valid state. Reset should be asserted before FLT is deasserted. This will ensure that the Am386DXL CPU will exist Float in a valid state.

FLT has an internal pull-up resistor, and if it is not used it should be unconnected.

Bus Activity During and Following Reset

RESET is the highest priority input signal capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage; or idle states or bus hold acknowledge states discontinued so that the RESET state is established.

RESET should remain asserted for at least 15-CLK2 periods to ensure it is recognized throughout the Am386DXL microprocessor, and at least 80-CLK2 periods if Am386DXL device self-test is going to be requested at the falling edge. RESET asserted pulses less than 15-CLK2 periods may not be recognized. RESET pulses less than 80-CLK2 periods followed by a self-test may cause the self-test to report a failure when no true failure exists.

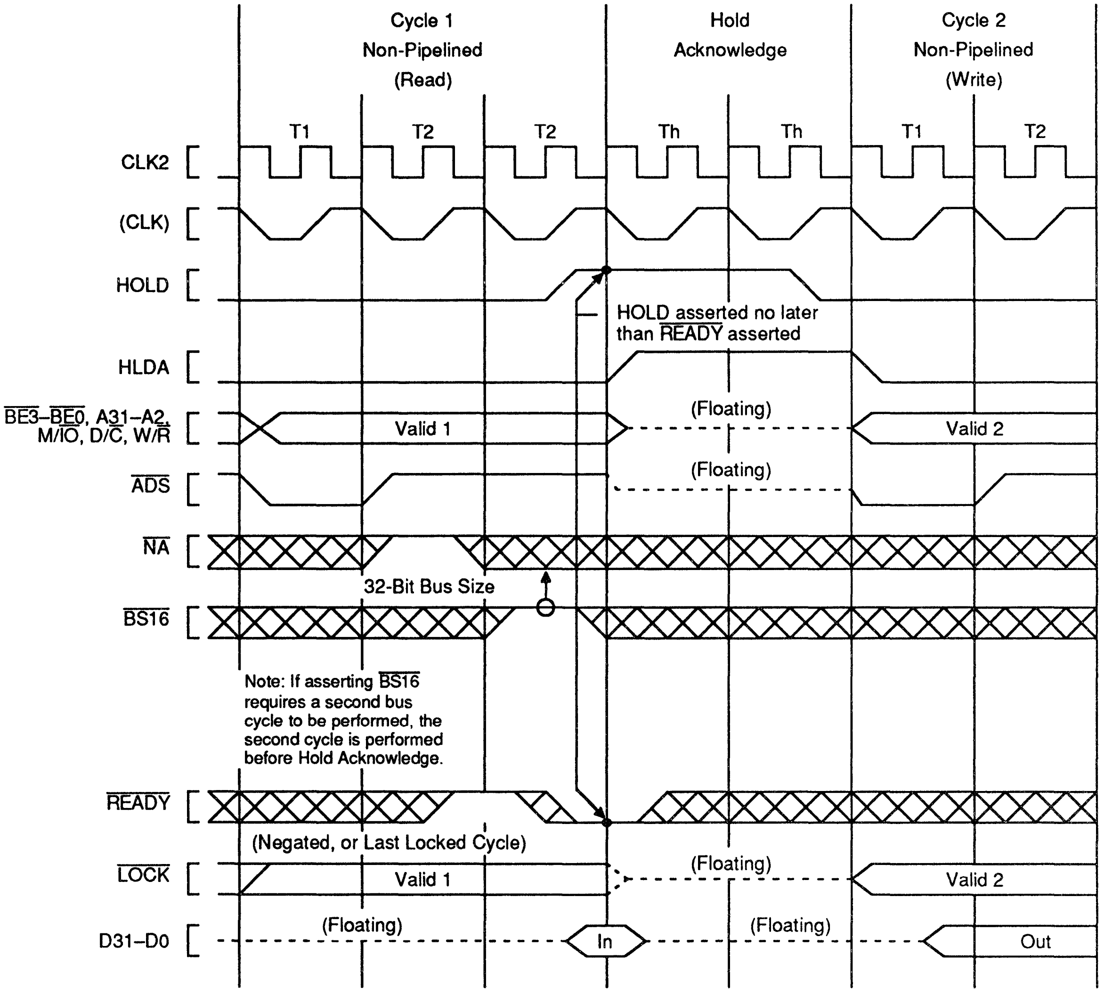
The additional RESET pulse width is required to clear additional state prior to valid self-test.

Provided the RESET falling edge meets setup and hold times, t25 and t26, the internal processor clock phase is defined at that time, as illustrated by Figure 67.

An Am386DXL microprocessor self-test may be requested at the time RESET is negated by having the $\overline{\text{BUSY}}$ input at a Low level, as shown in Figure 67. The self-test requires (2^{20}) + approximately 60-CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the Am386DXL device attempts to proceed with the reset sequence afterward.

After the RESET falling edge (and after the self-test if it was requested) the Am386DXL microprocessor performs an internal initialization sequence for approximately 350- to 450-CLK2 periods.

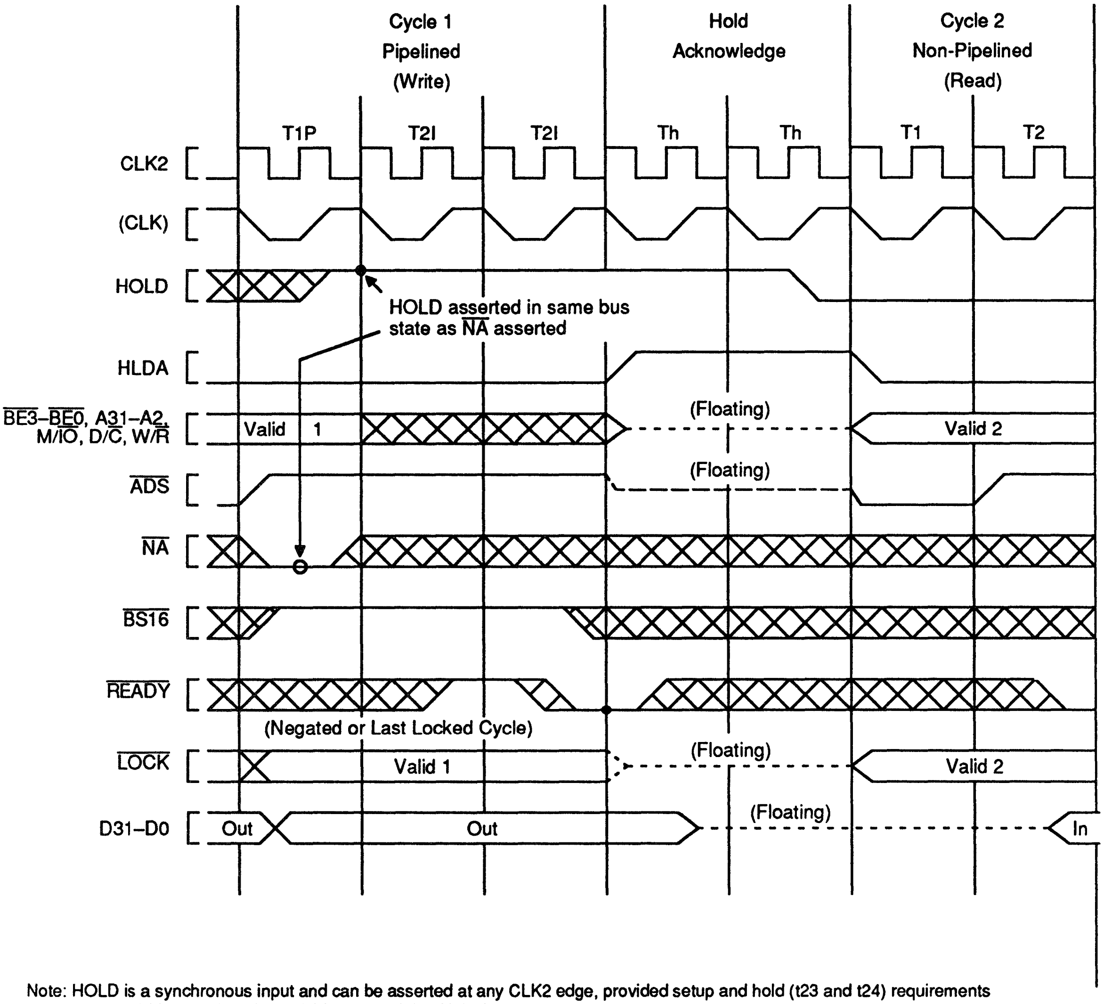
The Am386DXL microprocessor samples its $\overline{\text{ERROR}}$ input some time after the falling edge of RESET and before executing the first ESC instruction. During this sampling period $\overline{\text{BUSY}}$ must be High. If $\overline{\text{ERROR}}$ was sampled active, the Am386DXL device employs the 32-bit protocol of a 387DX math coprocessor. Even though this protocol was selected, it is still necessary to use a software recognition test to determine the presence or identity of the coprocessor and to assure compatibility with future processors.


Cycle 1
Non-Pipelined
(Read)
Hold
Acknowledge
Cycle 2
Non-Pipelined
(Write)
T1
T2
T2
Th
Th
T1
T2
CLK2
(CLK)
HOLD
HOLD asserted no later than READY asserted
HLDA
BE3–BE0, A31–A2, M/IO, D/C, W/R
Valid 1
(Floating)
Valid 2
ADS
(Floating)
NA
32-Bit Bus Size
BS16
Note: If asserting BS16 requires a second bus cycle to be performed, the second cycle is performed before Hold Acknowledge.
READY
(Negated, or Last Locked Cycle)
LOCK
Valid 1
(Floating)
Valid 2
D31–D0
(Floating)
In
(Floating)
Out

Note: HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t23 and t24) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

15021B–068

Figure 65. Requesting Hold from Active Bus (NA Negated)



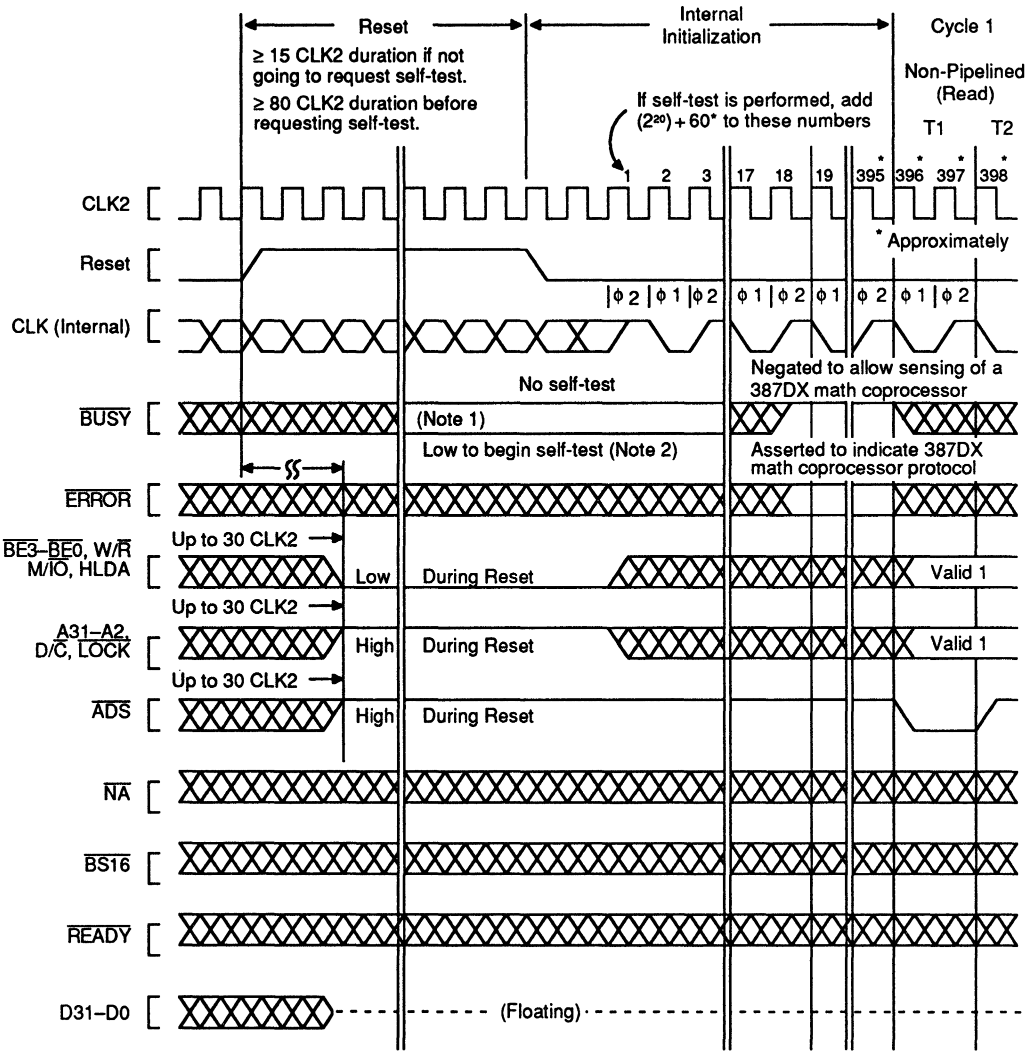
Note: HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t23 and t24) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

15021B–069

Figure 66. Requesting Hold from Active Bus (NA Asserted)

Table 21. Component and Revision Identifier History

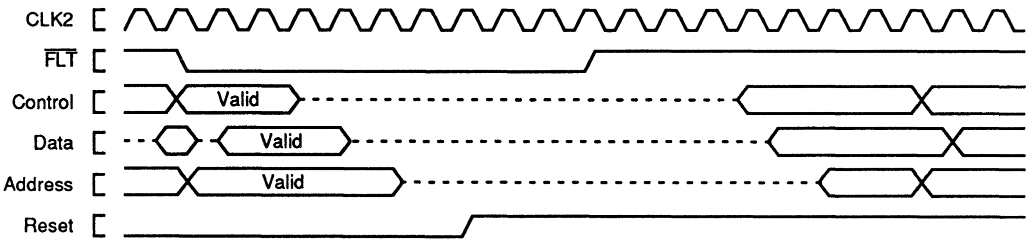
Intel i386 Stepping Name	Am386DXL Microprocessor Revision	Component Identifier	Revision Identifier
D0	B	03	05


Reset
≥ 15 CLK2 duration if not going to request self-test.
≥ 80 CLK2 duration before requesting self-test.
Internal Initialization
If self-test is performed, add (2²⁰) + 60* to these numbers
Cycle 1
Non-Pipelined (Read)
T1
T2
1
2
3
17
18
19
395
396
397
398
* Approximately
CLK2
Reset
CLK (Internal)
φ 2
φ 1
BUSY
ERROR
BE3–BE0, W/R M/IO, HLDA
A31–A2, D/C, LOCK
ADS
NA
BS16
READY
D31–D0
No self-test
(Note 1)
Low to begin self-test (Note 2)
Negated to allow sensing of a 387DX math coprocessor
Asserted to indicate 387DX math coprocessor protocol
Up to 30 CLK2
Low
High
During Reset
Valid 1
(Floating)

Notes: 1. BUSY should be held stable for 8-CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested, the Am386DXL microprocessor outputs remain in their reset state as shown here and in Table 14.

Figure 67. Bus Activity from Reset Until First Code Fetch

15021B–070



15022B–029

Figure 68. Entering and Exiting FLT

Self-Test Signature

Upon completion of self-test, (if self-test was requested by holding BUSY Low at least eight CLK2 periods before and after the falling edge of RESET), the EAX register will contain a signature of 00000000h indicating the Am386DXL CPU passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000h, applies to all Am386DXL microprocessor revision levels. Any non-zero signature indicates the Am386DXL CPU unit is faulty.

Component and Revision Identifiers

To assist Am386DXL microprocessor users, the microprocessor after reset holds a component identifier and a revision identifier in its DX register. The upper 8 bits of DX hold 03h as identification of the Am386DXL CPU component. The lower 8 bits of DX hold an 8-bit unsigned binary number related to the component revision level. The revision identifier begins chronologically with a value zero and is subject to change (typically it will be incremented) with component steppings intended to have certain improvements or distinctions from previous steppings.

These features are intended to assist Am386DXL microprocessor users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision nor to follow a completely uniform numerical sequence, depending on the type or intention of revision or manufacturing materials required to be changed.

Coprocessor Interfacing

The Am386DXL microprocessor provides an automatic interface for a 387DX floating-point math coprocessor. A 387DX math coprocessor uses an I/O-mapped interface driven automatically by the Am386DXL microprocessor and assisted by three dedicated signals: BUSY, ERROR, and PEREQ.

As the Am386DXL CPU begins supporting a coprocessor instruction, it tests the BUSY and ERROR signals to determine if the coprocessor can accept its next instruction. Thus, the BUSY and ERROR inputs eliminate the need for any preamble bus cycles for communication between processor and coprocessor. A 387DX math coprocessor can be given its command op-code immediately. The dedicated signals provide instruction synchronization, and eliminate the need of using the Am386DXL CPU WAIT op-code (9Bh) for 387DX math coprocessor instruction synchronization (the WAIT op-code was required when 8086 or 8088 was used with the 8087 coprocessor).

Custom coprocessors can be included in Am386DXL microprocessor based systems, via memory-mapped or I/O-mapped interfaces. Such coprocessor interfaces allow a completely custom protocol, and are not limited to a set of coprocessor protocol primitives. Instead, memory-mapped or I/O-mapped interfaces may use all applicable Am386DXL microprocessor instructions for high-speed coprocessor communication. The BUSY and ERROR inputs of the Am386DXL CPU may also be used for the custom coprocessor interface, if such hardware assist is desired. These signals can be tested by the Am386DXL CPU WAIT op-code (9Bh). The WAIT instruction will wait until the BUSY input is negated (interruptable by an NMI or enable INTR input), but generates an Exception 16 fault if the ERROR pin is in the asserted state when the BUSY goes (or is) negated. If the custom coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the Am386DXL microprocessor on-chip paging or segmentation mechanisms. If the custom interface is I/O-mapped, protection of the interface can be provided with the Am386DXL microprocessor IOPL (I/O Privilege Level) mechanism.

A 387DX math coprocessor interface is I/O mapped as shown in Table 22. Note that a 387DX math coprocessor interface addresses are beyond the 0hFFFFh range for programmed I/O. When the Am386DXL CPU supports a 387DX math coprocessor, the Am386DXL microprocessor automatically generates bus cycles to the coprocessor interface addresses.

Table 22. Math Coprocessor Port Addresses

Address in Am386DXL CPU I/O Space	387DX Coprocessor Register
800000F8h	Opcode Register (32-bit port)
800000FCh	Operand Register (32-bit port)

To correctly map a 387DX math coprocessor registers to the appropriate I/O addresses, connect a 387DX math coprocessor CMD0 pin directly to the A2 output of the Am386DXL microprocessor.

Software Testing for Coprocessor Presence

When software is used to test for coprocessor (387DX) presence, it should use only the following coprocessor op-codes: FINIT, FNINIT, FSTCW mem, FSTSW mem, FSTSW AX. To use other coprocessor op-codes when a coprocessor is known to be not present, first set EM = 1 in Am386DXL microprocessor CR0.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	−65°C to +150°C
Ambient Temperature Under Bias	−65°C to +125°C
Supply Voltage with Respect to Vss	−0.5 V to +7 V
Voltage on Other Pins	–0.5 V to Vcc +0.5 V

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to Absolute Maximum Ratings for extended periods may affect device reliability.

DC CHARACTERISTICS over **COMMERCIAL** operating ranges

$V_{CC} = 5$ V ±5%; $T_{CASE} = 0$°C to +85°C (PGA)
$V_{CC} = 5$ V±10%; $T_{CASE} = 0$°C to +100°C (PQFP)

Symbol	Parameter Description	Notes	Min	Max	Unit
V_{IL}	Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IH}	Input High Voltage		2.0	$V_{CC} + 0.3$	V
V_{ILC}	CLK2 Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IHC}	CLK2 Input High Voltage 20 MHz 25, 33, and 40 MHz		 $V_{CC} - 0.8$ 3.7	 $V_{CC} + 0.3$ $V_{CC} + 0.3$	 V V
V_{OL}	Output Low Voltage $I_{OL} = 4$ mA: A31–A2, D31–D0 $I_{OL} = 5$ mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, HLDA			 0.45 0.45	 V V
V_{OH}	Output High Voltage $I_{OH} = 1$ mA: A31–A2, D31–D0 $I_{OH} = 0.9$ mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, HLDA		 2.4 2.4		 V V
I_{LI}	Input Leakage Current (All pins except $\overline{\text{BS16}}$, PEREQ, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ and $\overline{\text{ERROR}}$)	$0\text{ V} \le V_{IN} \le V_{CC}$		±15	µA
I_{IH}	Input Leakage Current (PEREQ Pin)	$V_{IH} = 2.4$ V (Note 2)		200	µA
I_{IL}	Input Leakage Current ($\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, and $\overline{\text{ERROR}}$)	$V_{IL} = 0.45$ (Note 3)		–400	µA
I_{LO}	Output Leakage Current	$0.45\text{ V} \le V_{OUT} \le V_{CC}$		±15	µA
I_{CC}	Supply Current CLK2 = 40 MHz: with –20 CLK2 = 50 MHz: with –25 CLK2 = 66 MHz: with –33 CLK2 = 80 MHz: with –40	(Note 4) I_{CC} Typ = 165 I_{CC} Typ = 210 I_{CC} Typ = 275 I_{CC} Typ = 330		 200 250 330 400	 mA mA mA mA
I_{CCSB}	Standby Current	I_{CCSB} Typ = 0.02 mA (Note 5)		 150	 µA
C_{IN}	Input or I/O Capacitance	$F_C = 1$ MHz (Note 4)		10	pF
C_{OUT}	Output Capacitance	$F_C = 1$ MHz (Note 4)		12	pF
C_{CLK}	CLK2 Capacitance	$F_C = 1$ MHz (Note 4)		20	pF

Notes: 1. The Min value, –0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. $\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, and $\overline{\text{ERROR}}$ inputs each have an internal pullup resistor.
4. Not 100% tested.
5. Measurement taken with inputs at rails, outputs unloaded, $\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, and $\overline{\text{ERROR}}$ at V_{CC} voltage level, PEREQ at Gnd.

SWITCHING CHARACTERISTICS over operating range

V_{CC} = 5 V ±5%; T_{CASE} = 0°C to +85°C

No.	Parameter Description	Notes	Ref Figure	40 MHz Min	40 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		0	40	MHz
1	CLK2 Period		71	12.5		ns
2a	CLK2 High Time	at 2 V	71	5		ns
2b	CLK2 High Time	at 3.7 V	71	3.25		ns
3a	CLK2 Low Time	at 2 V	71	5		ns
3b	CLK2 Low Time	at 0.8 V	71	3.25		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	71		4	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	71		4	ns
6	A31–A2 Valid Delay	C_L = 50 pF	70, 73, 81	4	13	ns
7	A31–A2 Float Delay	(Note 1)	81	4	20	ns
8	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	13	ns
9	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	81	4	20	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	13	ns
10a	$\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	13	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	81	4	20	ns
12	D31–D0 Write Data Valid Delay	C_L = 50 pF (Note 4)	70, 74, 81	7	18	ns
12a	D31–D0 Write Data Hold Time	C_L = 50 pF	70, 75	2		ns
13	D31–D0 Float Delay	(Note 1)	81	4	17	ns
14	HLDA Valid Delay	C_L = 50 pF	70, 81	4	17	ns
15	$\overline{\text{NA}}$ Setup Time		72	5		ns
16	$\overline{\text{NA}}$ Hold Time		72	2		ns
17	$\overline{\text{BS16}}$ Setup Time		72	5		ns
18	$\overline{\text{BS16}}$ Hold Time		72	2		ns
19	$\overline{\text{READY}}$ Setup Time		72	7		ns
20	$\overline{\text{READY}}$ Hold Time		72	4		ns
21	D31–D0 Read Setup Time		72	4		ns
22	D31–D0 Read Hold Time		72	3		ns
23	HOLD Setup Time		72	4		ns
24	HOLD Hold Time		72	2		ns
25	RESET Setup Time		82	4		ns
26	RESET Hold Time		82	2		ns
27	NMI, INTR Setup Time	(Note 2)	72	5		ns
28	NMI, INTR Hold Time	(Note 2)	72	5		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$ Setup Time	(Note 2)	72	5		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$ Hold Time	(Note 2)	72	4		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific clock period.

3. Rise and fall times are not tested.

4. Min time not 100% tested.

SWITCHING CHARACTERISTICS over operating range

V_{CC} = 5 V ±5%; T_{CASE} = 0°C to +85°C

No.	Parameter Description	Notes	Ref Figures	33 MHz Min	33 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		0	33.3	MHz
1	CLK2 Period		71	15.0		ns
2a	CLK2 High Time	at 2 V	71	6.25		ns
2b	CLK2 High Time	at 3.7 V	71	4.5		ns
3a	CLK2 Low Time	at 2 V	71	6.25		ns
3b	CLK2 Low Time	at 0.8 V	71	4.5		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	71		4	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	71		4	ns
6	A31–A2 Valid Delay	C_L = 50 pF	70, 73, 81	4	15	ns
7	A31–A2 Float Delay	(Note 1)	81	4	20	ns
8	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	15	ns
9	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	81	4	20	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	15	ns
10a	$\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	14.5	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	81	4	20	ns
12	D31–D0 Write Data Valid Delay	C_L = 50 pF (Note 4)	70, 74, 81	7	24	ns
12a	D31–D0 Write Data Hold Time	C_L = 50 pF	70, 75	2		ns
13	D31–D0 Float Delay	(Note 1)	81	4	17	ns
14	HLDA Valid Delay	C_L = 50 pF	70, 81	4	20	ns
15	$\overline{\text{NA}}$ Setup Time		72	5		ns
16	$\overline{\text{NA}}$ Hold Time		72	2		ns
17	$\overline{\text{BS16}}$ Setup Time		72	5		ns
18	$\overline{\text{BS16}}$ Hold Time		72	2		ns
19	$\overline{\text{READY}}$ Setup Time		72	7		ns
20	$\overline{\text{READY}}$ Hold Time		72	4		ns
21	D31–D0 Read Setup Time		72	5		ns
22	D31–D0 Read Hold Time		72	3		ns
23	HOLD Setup Time		72	11		ns
24	HOLD Hold Time		72	2		ns
25	RESET Setup Time		82	5		ns
26	RESET Hold Time		82	2		ns
27	NMI, INTR Setup Time	(Note 2)	72	5		ns
28	NMI, INTR Hold Time	(Note 2)	72	5		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$ Setup Time	(Note 2)	72	5		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$ Hold Time	(Note 2)	72	4		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

3. Rise and fall times are not tested.

4. Min time not 100% tested.

SWITCHING CHARACTERISTICS over operating range

V_{CC} = 5 V ±5; T_{CASE} = 0°C to +85°C (PGA)
V_{CC} = 5 V ±10%; T_{CASE} = 0°C to +100°C (PQFP)

No.	Parameter Description	Notes	Ref Figures	25 MHz Min	25 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		0	25	MHz
1	CLK2 Period		71	20		ns
2a	CLK2 High Time	at 2 V	71	7		ns
2b	CLK2 High Time	at 3.7 V	71	4		ns
3a	CLK2 Low Time	at 2 V	71	7		ns
3b	CLK2 Low Time	at 0.8 V	71	5		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	71		7	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	71		7	ns
6	A31–A2 Valid Delay	C_L = 50 pF	70, 73, 81	4	21	ns
7	A31–A2 Float Delay	(Note 1)	81	4	30	ns
8	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	24	ns
8a	$\overline{\text{LOCK}}$ Valid Delay	CL = 50 pF	70, 73, 81	4	21	ns
9	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	81	4	30	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	70, 73, 81	4	21	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	81	4	30	ns
12	D31–D0 Write Data Valid Delay	C_L = 50 pF	70, 74, 81	7	27	ns
12a	D31–D0 Write Data Hold Time	C_L = 50 pF	70, 81	2		ns
13	D31–D0 Float Delay	(Note 1)	81	4	22	ns
14	HLDA Valid Delay	C_L = 50 pF	70, 81	4	22	ns
15	$\overline{\text{NA}}$ Setup Time		72	7		ns
16	$\overline{\text{NA}}$ Hold Time		72	3		ns
17	$\overline{\text{BS16}}$ Setup Time		72	7		ns
18	$\overline{\text{BS16}}$ Hold Time		72	3		ns
19	$\overline{\text{READY}}$ Setup Time		72	9		ns
20	$\overline{\text{READY}}$ Hold Time		72	4		ns
21	D31–D0 Read Setup Time		72	7		ns
22	D31–D0 Read Hold Time		72	5		ns
23	HOLD Setup Time		72	15		ns
24	HOLD Hold Time		72	3		ns
25	RESET Setup Time		82	10		ns
26	RESET Hold Time		82	3		ns
27	NMI, INTR Setup Time	(Note 2)	72	6		ns
28	NMI, INTR Hold Time	(Note 2)	72	6		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Setup Time	(Note 2)	72	6		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Hold Time	(Note 2)	72	5		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.

SWITCHING CHARACTERISTICS over operating range

$V_{CC} = 5$ V ±5%; $T_{CASE} = 0°C$ to +85°C (PGA)
$V_{CC} = 5$ V ±10%; $T_{CASE} = 0°C$ to +100°C (PQFP)

No.	Parameter Description	Notes	Ref Figures	20 MHz Min	20 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		0	20	MHz
1	CLK2 Period		71	25		ns
2a	CLK2 High Time	at 2 V	71	8		ns
2b	CLK2 High Time	at (V_{CC}–0.8 V)	71	5		ns
3a	CLK2 Low Time	at 2 V	71	8		ns
3b	CLK2 Low Time	at 0.8 V	71	6		ns
4	CLK2 Fall Time	(V_{CC}–0.8 V) to 0.8 V (Note 3)	71		8	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}–0.8 V) (Note 3)	71		8	ns
6	A31–A2 Valid Delay	$C_L = 120$ pF	70, 73, 81	4	30	ns
7	A31–A2 Float Delay	(Note 1)	81	4	32	ns
8	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$ Valid Delay	$C_L = 75$ pF	70, 73, 81	4	30	ns
9	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	81	4	32	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Valid Delay	$C_L = 75$ pF	70, 73, 81	4	28	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	81	4	30	ns
12	D31–D0 Write Data Valid Delay	$C_L = 120$ pF	70, 74, 81	4	38	ns
13	D31–D0 Float Delay	(Note 1)	81	4	27	ns
14	HLDA Valid Delay	$C_L = 75$ pF	70, 81	6	28	ns
15	$\overline{\text{NA}}$ Setup Time		72	9		ns
16	$\overline{\text{NA}}$ Hold Time		72	14		ns
17	$\overline{\text{BS16}}$ Setup Time		72	13		ns
18	$\overline{\text{BS16}}$ Hold Time		72	21		ns
19	$\overline{\text{READY}}$ Setup Time		72	12		ns
20	$\overline{\text{READY}}$ Hold Time		72	4		ns
21	D31–D0 Read Setup Time		72	11		ns
22	D31–D0 Read Hold Time		72	6		ns
23	HOLD Setup Time		72	17		ns
24	HOLD Hold Time		72	5		ns
25	RESET Setup Time		82	12		ns
26	RESET Hold Time		82	4		ns
27	NMI, INTR Setup Time	(Note 2)	72	16		ns
28	NMI, INTR Hold Time	(Note 2)	72	16		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Setup Time	(Note 2)	72	14		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Hold Time	(Note 2)	72	5		ns

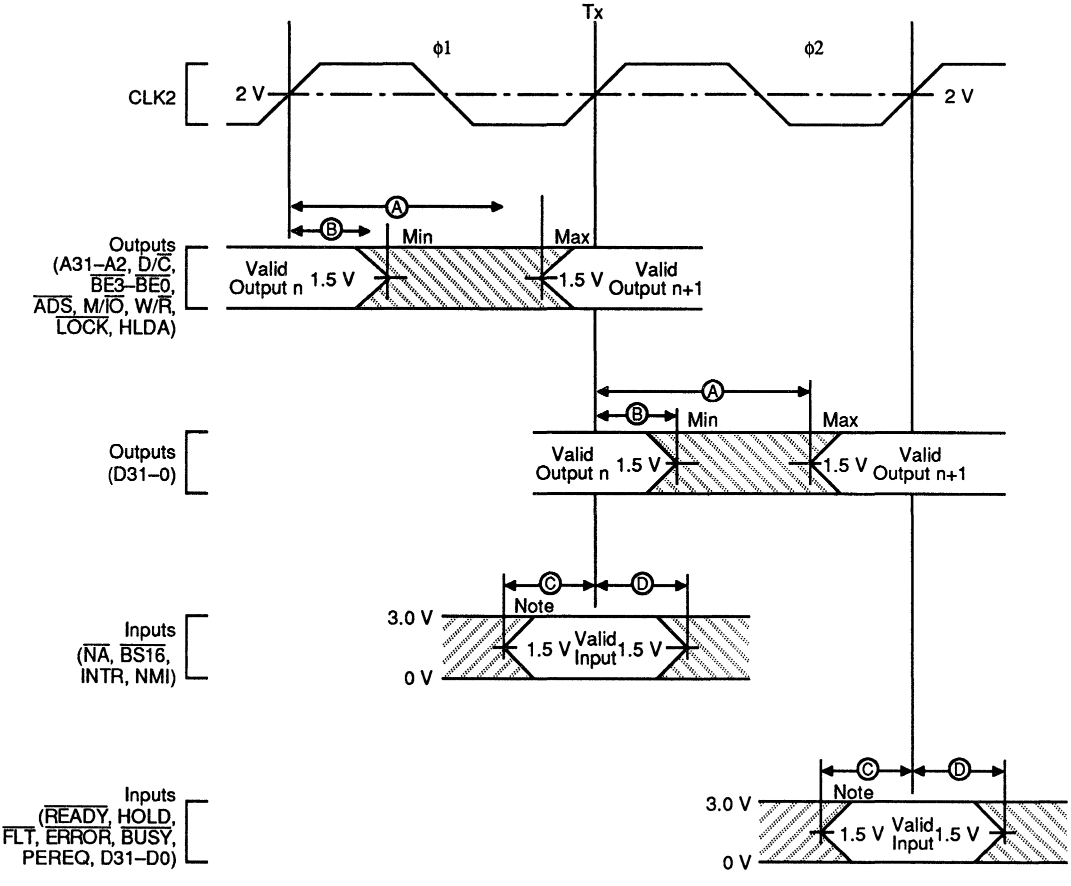
Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.

SWITCHING WAVEFORMS

The switching characteristics consist of output delays, input setup requirements, and input hold requirements. All characteristics are relative to the CLK2 rising edge crossing the 2.0 V level.

Switching characteristic measurement is defined by Figure 69. Inputs must be driven to the voltage levels indicated by this diagram. Am386DXL CPU output delays are specified with minimum and maximum limits measured as shown. The minimum Am386DXL microprocessor delay times are hold times provided to external circuitry. Am386DXL microprocessor input setup and hold time are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct Am386DXL microprocessor operation.

Outputs $\overline{ADS}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{BE3}$–$\overline{BE0}$, A31–A2, and HLDA only change at the beginning of phase one. D31–D0 (write cycles) only change at the beginning of phase two. The $\overline{READY}$, HOLD, $\overline{BUSY}$, $\overline{ERROR}$, PEREQ, $\overline{FLT}$, and D31–D0 (read cycles) inputs are sampled at the beginning of phase one. The $\overline{NA}$, $\overline{BS16}$, INTR, and NMI inputs are sampled at the beginning of phase two.



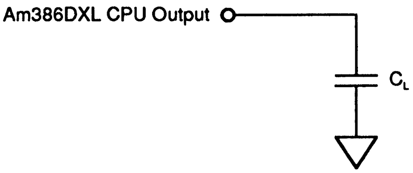
Tx
ϕ1
ϕ2
CLK2
2 V
2 V
Outputs (A31–A2, D/$\overline{C}$, $\overline{BE3}$–$\overline{BE0}$, $\overline{ADS}$, M/$\overline{IO}$, W/$\overline{R}$, $\overline{LOCK}$, HLDA)
A
B
Min
Max
Valid Output n
1.5 V
1.5 V
Valid Output n+1
Outputs (D31–0)
A
B
Min
Max
Valid Output n
1.5 V
1.5 V
Valid Output n+1
Inputs ($\overline{NA}$, $\overline{BS16}$, INTR, NMI)
C
D
Note
3.0 V
0 V
1.5 V
Valid Input
1.5 V
Inputs ($\overline{READY}$, HOLD, $\overline{FLT}$, $\overline{ERROR}$, $\overline{BUSY}$, PEREQ, D31–D0)
C
D
Note
3.0 V
0 V
1.5 V
Valid Input
1.5 V

Legend: A—Maximum Output Delay Spec
B—Minimum Output Delay Spec
C—Minimum Input Setup Spec
D—Minimum Input Hold Spec

Note: Input waveforms have tr ≤ 2.0 ns from 0.8 V to 2.0 V.

15021B–071

Figure 69. Drive Levels and Measurement Points



C_L includes all parasitic capacitances.

15021B–072

Figure 70. AC Test Load

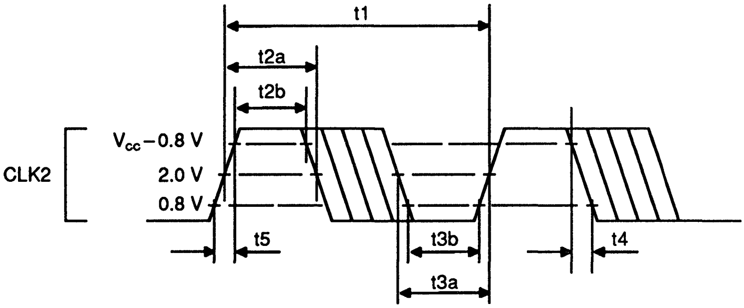
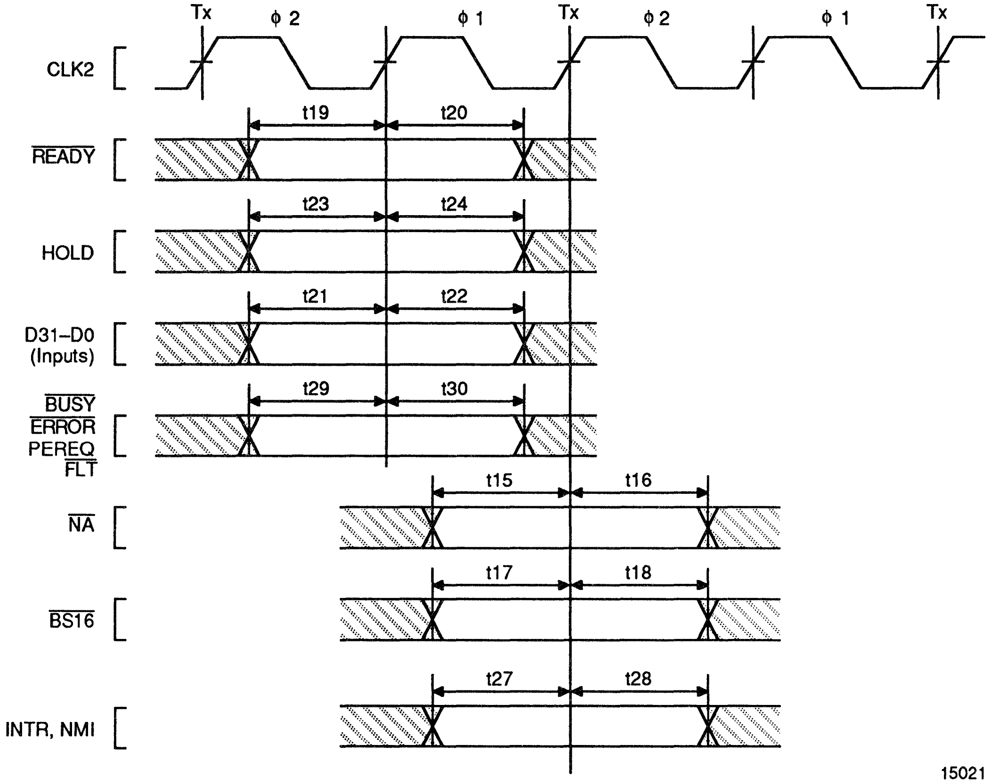


Figure 71. CLK2 Timing

15021B–073



15021B–074

Figure 72. Input Setup and Hold Timing

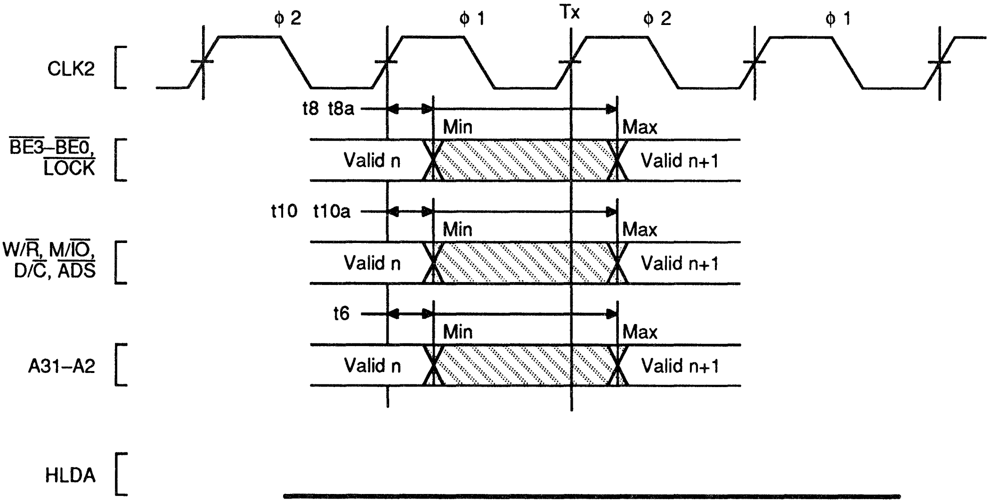


Figure 73. Output Valid Delay Timing

15021B–075

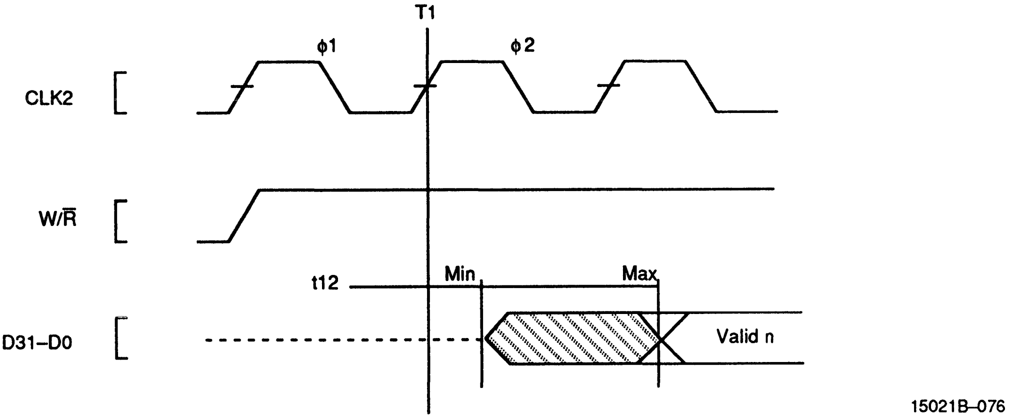


Figure 74. Write Data Valid Delay Timing (25, 33, and 40 MHz)

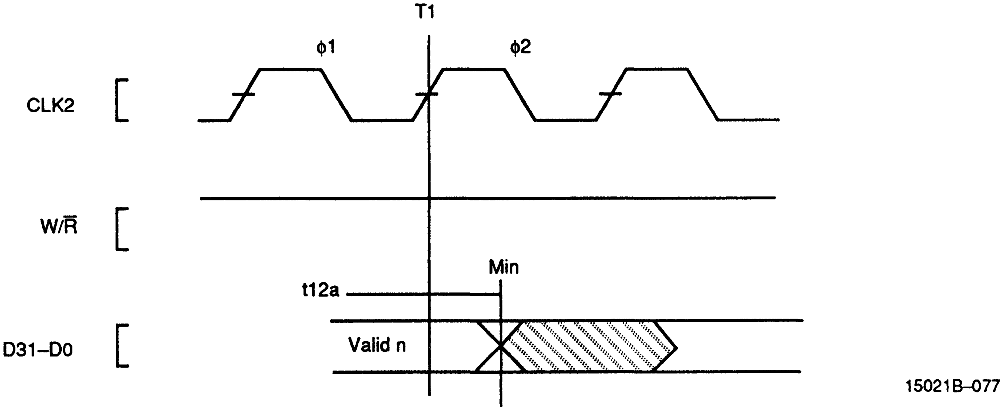


Figure 75. Write Data Hold Timing (25, 33, 40 MHz)

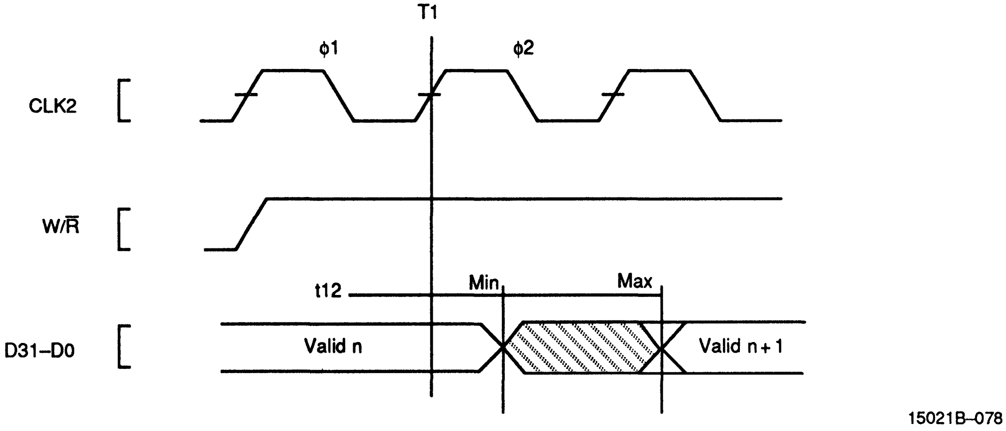
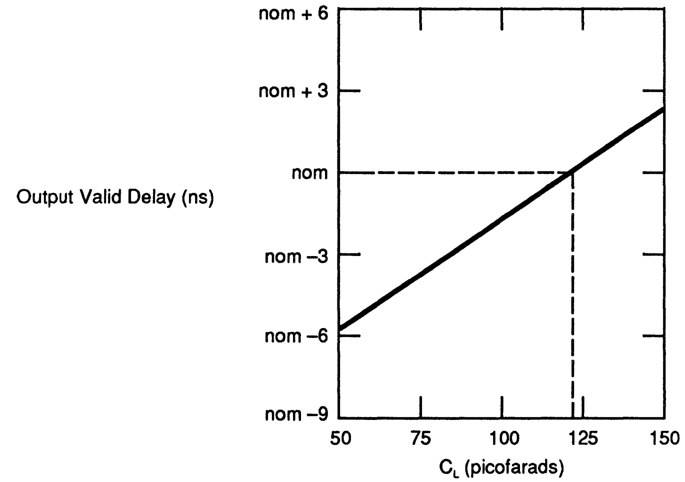


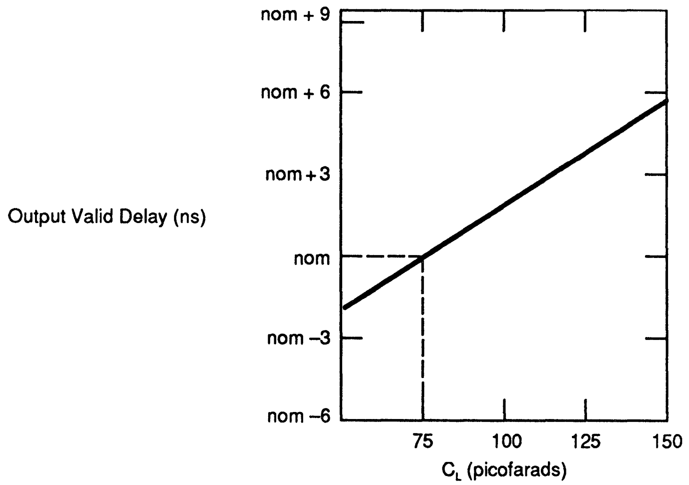
Figure 76. Write Data Valid Delay Timing (20 MHz)



Note: This graph will not be linear outside of the C_L range shown.

15021B–079

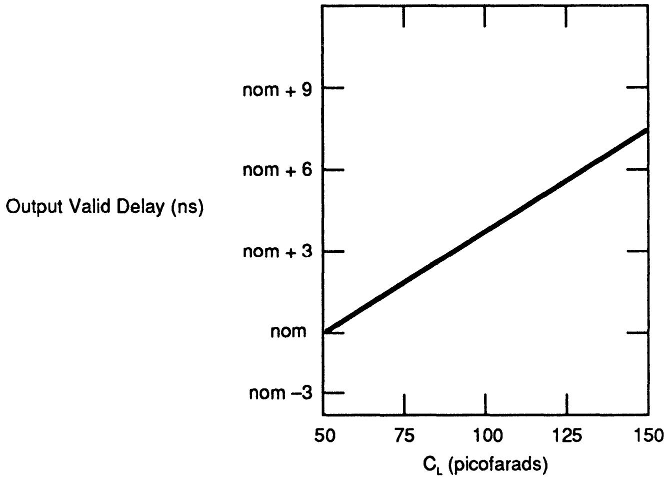
Figure 77. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L =120 pF)



Note: This graph will not be linear outside of the C_L range shown.

15021B–080

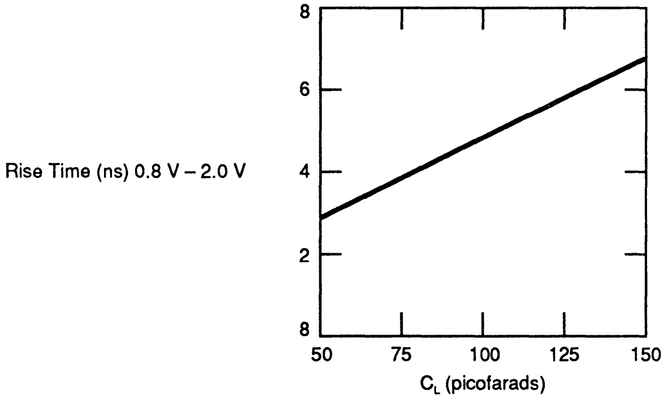
Figure 78. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 75 pF)



Note: This graph will not be linear outside of the C_L range shown.

15021B–081

Figure 79. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 50 pF)



Note: This graph will not be linear outside of the C_L range shown.

15021B–082

Figure 80. Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature

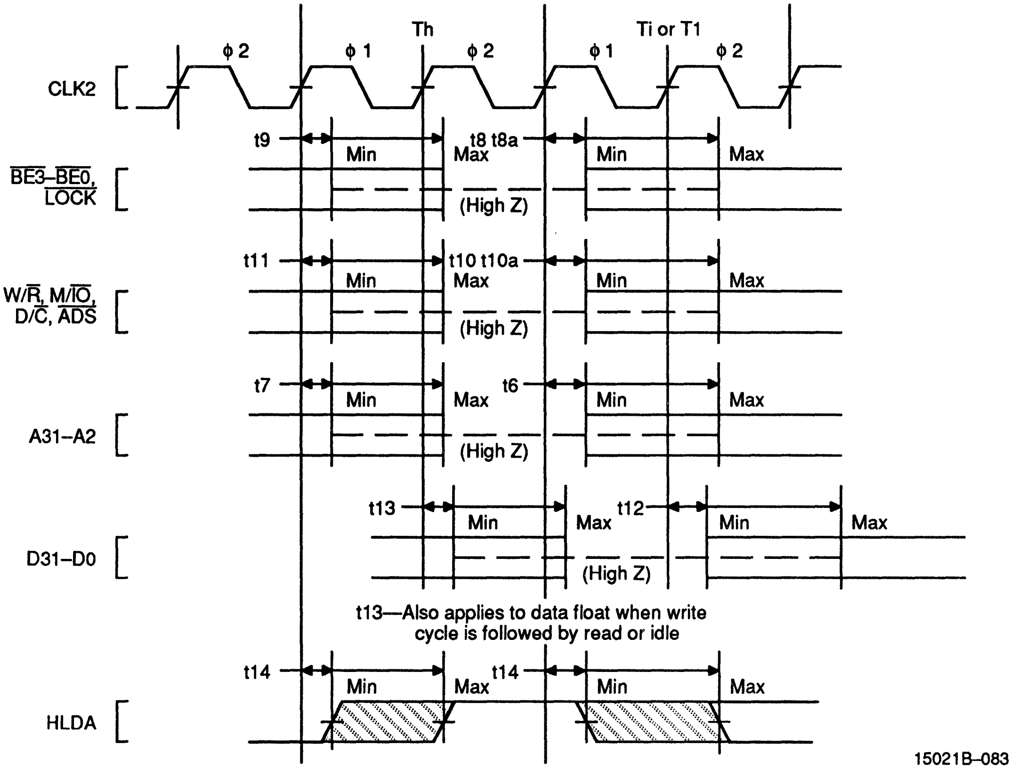
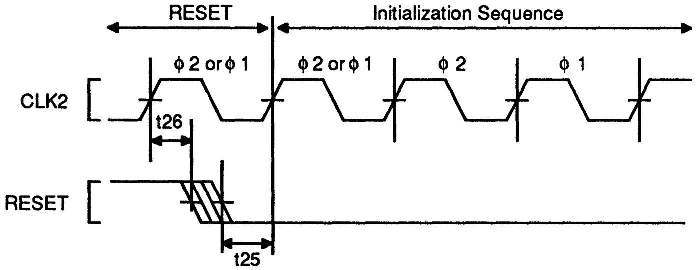


Figure 81. Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET High-to-Low transition (provided t25 and t26 are met) is φ2.

15021B–084

Figure 82. RESET Setup and Hold Timing and Internal Phase

INSTRUCTION SET

This section describes the Am386DXL microprocessor instruction set. A table lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within Am386DXL CPU instructions.

Am386DXL Microprocessor Instruction Encoding and Clock Count Summary

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table 23, by the processor clock period (e.g., 50 ns for a 20 MHz, 40 ns for a 25 MHz, 30 ns for a 33 MHz, and 25 ns for a 40 MHz Am386DXL microprocessor).

For more detailed information on the encodings of instructions refer to Section Instruction Encodings. Section Instruction Encodings explains the general structure of instruction encodings and defines exactly the encodings of all fields contained within the instruction.

Instruction Clock Count Assumptions

1. The instruction has been prefetched and decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor access to the bus.
4. No Exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling, and displacement can be used within the clock counts shown. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.

Instruction Clock Count Notation

1. If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.
2. n = number of times repeated.
3. m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component; the entire immediate data (if any) counts as one component; and each of the other bytes of the instruction and prefix(es) each count as one component.

Table 23. Am386DXL Microprocessor Instruction Set Summary

Instruction	Format			Clock Count		Comments	
				Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
GENERAL DATA TRANSFER							
MOV = Move:							
Register to Register/Memory	1 0 0 0 1 0 0 w	mod reg r/m		2/2	2/2	b	h
Register/Memory to Register	1 0 0 0 1 0 1 w	mod reg r/m		2/4	2/4	b	h
Immediate to Register/Memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	immediate data	2/2	2/2	b	h
Immediate to Register (short form)	1 0 1 1 w reg	immediate data		2	2		
Memory to Accumulator (short form)	1 0 1 0 0 0 0 w	full displacement		4	4	b	h
Accumulator to Memory (short form)	1 0 1 0 0 0 1 w	full displacement		2	2	b	h
Register/Memory to Segment Register	1 0 0 0 1 1 1 0	mod sreg3 r/m		2/5	18, 19	b	h, i, j
Segment Register to Register/Memory	1 0 0 0 1 1 0 0	mod reg r/m		2/2	2/2	b	h
MOVSX = Move with Sign Extension							
Register from Register/Memory	0 0 0 0 1 1 1 1	1 0 1 1 1 1 1 w	mod reg r/m	3/6	3/6	b	h
MOVZX = Move with Zero Extension							
Register from Register/Memory	0 0 0 0 1 1 1 1	1 0 1 1 0 1 1 w	mod reg r/m	3/6	3/6	b	h
PUSH = Push:							
Register/Memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m		5	5	b	h
Register (short form)	0 1 0 1 0 reg			2	2	b	h
Segment Register (ES,CS,SS, or DS)	0 0 0 sreg 2 1 1 0			2	2	b	h
Segment Register (FS or GS)	0 0 0 0 1 1 1 1	1 0 sreg 3 0 0 0		2	2	b	h
Immediate	0 1 1 0 1 0 s 0	immediate data		2	2	b	h
PUSHA = Push All	0 1 1 0 0 0 0 0			18	18	b	h
POP = Pop							
Register/Memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m		5	5	b	h
Register (short form)	0 1 0 1 1 reg			4	4	b	h
Segment Register (ES, SS, or DS)	0 0 0 sreg 2 1 1 1			7	21	b	h, i, j
Segment Register (FS or GS)	0 0 0 0 1 1 1 1	1 0 sreg 3 0 0 1		7	21	b	h, i, j
POPA = Pop All	0 1 1 0 0 0 0 1			24	24	b	h
XCHG = Exchange							
Register/Memory with Register	1 0 0 0 0 1 1 w	mod reg r/m		3/5	3/5	b, f	f, h
Register with Accumulator (short form)	1 0 0 1 0 reg			3	3		
IN = Input from:			**Clock Count Virtual 8086 Mode**				
Fixed Port	1 1 1 0 0 1 0 w	port number	◊26	12	6*/26**		m
Variable Port	1 1 1 0 1 1 0 w		◊27	13	7*/27**		m
OUT = Output to:							
Fixed Port	1 1 1 0 0 1 1 w	port number	◊24	10	4*/24**		m
Variable Port	1 1 1 0 1 1 1 w		◊25	11	5*/25**		m
LEA = Load EA to Register	1 0 0 0 1 1 0 1	mod reg r/m		2	2		

* If CPL ≤ IOPL ** If CPL > IOPL

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

Instruction	Format			Clock Count		Comments	
				Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
SEGMENT CONTROL							
LDS = Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		7	22	b	h, i, j
LES = Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		7	22	b	h, i, j
LFS = Load pointer to FS	0 0 0 0 1 1 1 1	1 0 1 1 0 1 0 0	mod reg r/m	7	25	b	h, i, j
LGS = Load pointer to GS	0 0 0 0 1 1 1 1	1 0 1 1 0 1 0 1	mod reg r/m	7	25	b	h, i, j
LSS = Load pointer to SS	0 0 0 0 1 1 1 1	1 0 1 1 0 0 1 0	mod reg r/m	7	22	b	h, i, j
FLAG CONTROL							
CLC = Clear Carry Flag	1 1 1 1 1 0 0 0			2	2		
CLD = Clear Direction Flag	1 1 1 1 1 1 0 0			2	2		
CLI = Clear Interrupt Enable Flag	1 1 1 1 1 0 1 0			8	8		m
CLTS = Clear Task Switched Flag	0 0 0 0 1 1 1 1	0 0 0 0 0 1 1 0		6	6	c	i
CMC = Complement Carry Flag	1 1 1 1 0 1 0 1			2	2		
LAHF = Load AH into Flag	1 0 0 1 1 1 1 1			2	2		
POPF = Pop Flag	1 0 0 1 1 1 0 1			5	5	b	h,n
PUSHF = Push Flag	1 0 0 1 1 1 0 0			4	4	b	h
SAHF = Store AH into Flag	1 0 0 1 1 1 1 0			3	3		
STC = Set Carry Flag	1 1 1 1 1 0 0 1			2	2		
STD = Set Direction Flag	1 1 1 1 1 1 0 1			2	2		
STI = Set Interrupt Enable Flag	1 1 1 1 1 0 1 1			8	8		m
ARITHMETIC							
ADD=Add							
Register to Register	0 0 0 0 0 0 d w	mod reg r/m		2	2		
Register to Memory	0 0 0 0 0 0 0 w	mod reg r/m		7	7	b	h
Memory to Register	0 0 0 0 0 0 1 w	mod reg r/m		6	6	b	h
Immediate to Register/Memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	immediate data	2/7	2/7	b	h
Immediate to Accumulator (short form)	0 0 0 0 0 1 0 w	immediate data		2	2		
ADC = Add with carry							
Register to Register	0 0 0 1 0 0 d w	mod reg r/m		2	2		
Register to Memory	0 0 0 1 0 0 0 w	mod reg r/m		7	7	b	h
Memory to Register	0 0 0 1 0 0 1 w	mod reg r/m		6	6	b	h
Immediate to Register/Memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	immediate data	2/7	2/7	b	h
Immediate to Accumulator (short form)	0 0 0 1 0 1 0 w	immediate data		2	2		
INC =Increment							
Register/Memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		2/6	2/6	b	h
Register (short form)	0 1 0 0 0 reg			2	2		
SUB =Subtract							
Register from Register	0 0 1 0 1 0 d w	mod reg r/m		2	2		
Register from Memory	0 0 1 0 1 0 0 w	mod reg r/m		7	7	b	h
Memory from Register	0 0 1 0 1 0 1 w	mod reg r/m		6	6	b	h

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

Instruction	Format				Clock Count		Comments	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC (continued)								
Immediate from Register/Memory	0 0 1 0 0 1 1 w	mod 1 0 1 r/m	immediate data		2/7	2/7	b	h
Immediate from Accumulator (short form)	0 0 0 1 1 1 0 w	immediate data			2	2		
SBB = Subtract with Borrow								
Register from Register	0 0 0 1 1 0 d w	mod reg r/m			2	2		
Register from Memory	0 0 0 1 1 0 0 w	mod reg r/m			7	7	b	h
Memory from Register	0 0 0 1 1 0 1 w	mod reg r/m			6	6	b	h
Immediate from Register/Memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	immediate data		2/7	2/7	b	h
Immediate from Accumulator	0 0 0 1 1 1 0 w	immediate data			2	2		
DEC = Decrement								
Register/Memory	1 1 1 1 1 1 1 w	reg 0 0 1 r/m			2/6	2/6	b	h
Register (short form)	0 1 0 0 1 reg				2	2		
CMP = Compare								
Register with Register	0 0 1 1 1 0 d w	mod reg r/m			2	2		
Memory with Register	0 0 1 1 1 0 0 w	mod reg r/m			5	5	b	h
Register with Memory	0 0 1 1 1 0 1 w	mod reg r/m			6	6	b	h
Immediate with Register/Memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	immediate data		2/5	2/5	b	h
Immediate with Accumulator(short form)	0 0 1 1 1 1 0 w	immediate data			2	2		
NEG = Change Sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m			2/6	2/6	b	h
AAA = ASCII Adjust for Add	0 0 1 1 0 1 1 1				4	4		
DAA = Decimal Adjust for Add	0 0 1 1 1 1 1 1				4	4		
AAS = ASCII Adjust for Subtract	0 0 1 0 0 1 1 1				4	4		
DAS = Decimal Adjust for Subtract	0 0 1 0 1 1 1 1				4	4		
MUL = Multiply (Unsigned)								
Accumulator with Register/Memory	1 1 1 1 0 1 1 w	mod 1 0 0 r/m						
Multiplier -Byte					12-17/15-20	12-17/15-20	b,d	d,h
-Word					12-25/15-28	12-25/15-28	b,d	d,h
-Doubleword					12-41/15-44	12-41/15-44	b,d	d,h
IMUL = Integer Multiply (signed)								
Accumulator with Register/Memory	1 1 1 1 0 1 1 w	mod 1 0 1 r/m						
Multiplier -Byte					12-17/15-20	12-17/15-20	b,d	d,h
-Word					12-25/15-28	12-25/15-28	b,d	d,h
-Doubleword					12-41/15-44	12-41/15-44	b,d	d,h
Register with Register/Memory	0 0 0 0 1 1 1 1	1 0 1 0 1 1 1 1	mod reg r/m					
Multiplier -Byte					12-17/15-20	12-17/15-20	b,d	d,h
-Word					12-25/15-28	12-25/15-28	b,d	d,h
-Doubleword					12-41/15-44	12-41/15-44	b,d	d,h
Register/Memory with Immediate to Register	0 1 1 0 1 0 s 1	mod reg r/m	immediate data					
-Word					13-26/14-27	13-26/14-27	b,d	d,h
-Doubleword					13-42/14-43	13-42/14-43	b,d	d,h

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

					Clock Count		Comments	
Instruction	**Format**				**Real Address Mode**	**Protected Virtual Address Mode**	**Real Address Mode**	**Protected Virtual Address Mode**
ARITHMETIC (continued)								
DIV = Divide (Unsigned)								
Accumulator by Register/Memory	1 1 1 1 0 1 1 w	mod 1 1 0 r/m						
Divisor -Byte					14/17	14/17	b,e	e, h
-Word					22/25	22/25	b,e	e, h
-Doubleword					38/41	38/41	b,e	e, h
IDIV = Integer Divide (Signed)								
Accumulator by Register/Memory	1 1 1 1 0 1 1 2	mod 1 1 1 r/m						
Divisor -Byte					19/22	19/22	b,e	e, h
-Word					27/30	27/30	b,e	e, h
-Doubleword					43/46	43/46	b,e	e, h
AAD = ASCII Adjust for Divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0			19	19		
AAM = ASCII Adjust for Multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0			17	17		
CBW = Convert Byte to Word	1 0 0 1 1 0 0 0				3	3		
CWD = Convert Word to Double Word	1 0 0 1 1 0 0 1				2	2		
LOGIC								
Shift/Rotate Instructions Not Through Carry (ROL, ROR, SAL, SAR, SHL, and SHR)								
Register/Memory by 1	1 1 0 1 0 0 0 w	mod TTT r/m			3/7	3/7	b	h
Register/Memory by CL	1 1 0 1 0 0 1 w	mod TTT r/m			3/7	3/7	b	h
Register Memory by Immediate Count	1 1 0 0 0 0 0 w	mod TTT r/m	immediate 8-bit data		3/7	3/7	b	h
Through Carry (RCL and RCR)								
Register/Memory by 1	1 1 0 1 0 0 0 w	mod TTT r/m			9/10	9/10	b	h
Register/Memory by CL	1 1 0 1 0 0 1 w	mod TTT r/m			9/10	9/10	b	h
Register/Memory by Immediate Count	1 1 0 0 0 0 0 w	mod TTT r/m	immediate 8-bit data		9/10	9/10	b	h
	TTT	**Instruction**						
	0 0 0	ROL						
	0 0 1	ROR						
	0 1 0	RCL						
	0 1 1	RCR						
	1 0 0	SHL/SAL						
	1 0 1	SHR						
	1 1 1	SAR						
SHLD = Shift Left Double								
Register/Memory by Immediate	0 0 0 0 1 1 1 1	1 0 1 0 0 1 0 0	mod reg r/m	immediate 8-bit data	3/7	3/7		
Register/Memory by CL	0 0 0 0 1 1 1 1	1 0 1 0 0 1 0 1	mod reg r/m		3/7	3/7		
SHRD = Shift Right Double								
Register/Memory by Immediate	0 0 0 0 1 1 1 1	1 0 1 0 1 1 0 0	mod reg r/m	immediate 8-bit data	3/7	3/7		
Register/Memory by CL	0 0 0 0 1 1 1 1	1 0 1 0 1 1 0 1	mod reg r/m		3/7	3/7		
AND = And								
Register to Register	0 0 1 0 0 0 d w	mod reg r/m			2	2		
Register to Memory	0 0 1 0 0 0 0 w	mod reg r/m			7	7	b	h
Memory to Register	0 0 1 0 0 0 1 w	mod reg r/m			6	6	b	h
Immediate to Register/Memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	immediate data		2/7	2/7	b	h
Immediate to Accumulator (short form)	0 0 1 0 0 1 0 w	immediate data			2	2		
TEST = And Function to Flags, no Result								
Register/Memory and Register	1 0 0 0 0 1 0 w	mod reg r/m			2/5	2/5	b	h
Immediate Data and Register/Memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	immediate data		2/5	2/5	b	h
Immediate Data and Accumulator (short form)	1 0 1 0 1 0 0 w	immediate data			2	2		

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

Instruction	Format				Clock Count		Comments	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
LOGIC (continued)								
OR = Or								
Register to Register	0 0 0 0 1 0 d w	mod reg r/m			2	2		
Register to Memory	0 0 0 0 1 0 0 w	mod reg r/m			7	7	b	h
Memory to Register	0 0 0 0 1 0 1 w	mod reg r/m			6	6	b	h
Immediate and Register/Memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	immediate data		2/7	2/7	b	h
Immediate to Accumulator (short form)	0 0 0 0 1 1 0 w	immediate data			2	2		
XOR = Exclusive or								
Register to Register	0 0 1 1 0 0 d w	mod reg r/m			2	2		
Register to Memory	0 0 1 1 0 0 0 w	mod reg r/m			7	7	b	h
Memory to Register	0 0 1 1 0 0 1 w	mod reg r/m			6	6	b	h
Immediate to Register/Memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	immediate data		2/7	2/7	b	h
Immediate to Accumulator (short form)	0 0 1 1 0 1 0 w	immediate data			2	2		
NOT = Invert Register/Memory	1 1 1 1 0 1 1 w	mod 0 1 0 r/m			2/6	2/6	b	h
STRING MANIPULATION				Clock Count Virtual 8086 Mode				
CMPS = Compare Byte/Word	1 0 1 0 0 1 1 w				10	10	b	h
INS = Input Byte/Wd from DX Port	0 1 1 0 1 1 0 w			29◊	15	9*/29**	b	h, m
LODS = Load Byte/Wd to AL/AX	1 0 1 0 1 1 0 w				5	5	b	h
MOVS = Move Byte/Word	1 0 1 0 0 1 0 w				8	8	b	h
OUTS = Output Byte/Wd to DX Port	0 1 1 0 1 1 1 w			28◊	14	8*/28**	b	h, m
SCAS = Scan Byte/Word	1 0 1 0 1 1 1 w				8	8	b	h
STOS = Store Byte/Word from AL/AX/EX	1 0 1 0 1 0 1 w				5	5	b	h
XLAT = Translate String	1 1 0 1 0 1 1 1				5	5		h
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX								
REPE CMPS = Compare string (Find Non-Match)	1 1 1 1 0 0 1 1	1 0 1 0 0 1 1 w			5+9n	5+9n	b	h
REPNE CMPS = Compare String (Find Match)	1 1 1 1 0 0 1 0	1 0 1 0 0 1 1 w		Clock Count Virtual 8086 Mode	5+9n	5+9n	b	h
REP INS = Input String	1 1 1 1 0 0 1 0	0 1 1 0 1 1 0 w		28 + 6n◊	14 + 6n	8 + 6n*/ 28 + 6n**	b	h, m
REP LODS = Load String	1 1 1 1 0 0 1 0	1 0 1 0 1 1 0 w			5 + 6n	5 + 6n	b	h
REP MOVS = Move String	1 1 1 1 0 0 1 0	1 0 1 0 0 1 0 w			8 + 4n	8 + 4n	b	h
REP OUTS = Output String	1 1 1 1 0 0 1 0	0 1 1 0 1 1 1 w		26 + 5n◊	12 + 5n	6 + 5n*/ 26 + 5n**	b	h, m
REPE SCAS = Scan String (Find Non-AL/AX/EAX)	1 1 1 1 0 0 1 1	1 0 1 0 1 1 1 w			5 + 8n	5 + 8n	b	h
REPNE SCAS = Store String (Find AL/AX/EAX)	1 1 1 1 0 0 1 0	1 0 1 0 1 1 1 w			5 + 8n	5 + 9n	b	h
REP STOS = Store String	1 1 1 1 0 0 1 0	1 0 1 0 1 0 1 w			5+5n	5+5n	b	h
BIT MANIPULATION								
BSF = Scan Bit Forward	0 0 0 0 1 1 1 1	1 0 1 1 1 1 0 0	mod reg r/m		11 + 3n	11 + 3n	b	h
BSR = Scan Bit Reverse	0 0 0 0 1 1 1 1	1 0 1 1 1 1 0 1	mod reg r/m		9 + 3n	9 + 3n	b	h

* If CPL ≤ IOPL   ** If CPL > IOPL

◊ Clock count shown applies if I/O permission allows I/O to the port in Virtual 8086 Mode. If I/O bit map denies permission, Exception 13 fault occurs; refer to clock counts for INT 3 instruction.

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

		Clock Count		Comments	
Instruction	Format	Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
BIT MANIPULATION (continued)					
BT = Test Bit					
Register/Memory, Immediate	00001111 10111010 mod 100 r/m immediate 8-bit data	3/6	3/6	b	h
Register/Memory, Register	00001111 10100011 mod reg r/m	3/12	3/12	b	h
BTC = Test Bit and Complement					
Register/Memory, Immediate	00001111 10111010 mod 111 r/m immediate 8-bit data	6/8	6/8	b	h
Register/Memory, Register	00001111 10111011 mod reg r/m	6/13	6/13	b	h
BTR = Test Bit and Reset					
Register/Memory, Immediate	00001111 10111010 mod 110 r/m immediate 8-bit data	6/8	6/8	b	h
Register/Memory, Register	00001111 10110011 mod reg r/m	6/13	6/13	b	h
BTS = Test Bit and Set					
Register/Memory, Immediate	00001111 10111010 mod 101 r/m immediate 8-bit data	6/8	6/8	b	h
Register/Memory, Register	00001111 10101011 mod reg r/m	6/13	6/13	b	h
CONTROL TRANSFER					
CALL = Call					
Direct Within Segment	11101000 full displacement	7+m	7+m	b	r
Register/Memory Indirect Within Segment	11111111 mod 010 r/m	7+m 10+m	7+m 10+m	b	h, r
Direct Intersegment	10011010 unsigned full offset, selector	17+m	34+m	b	j, k, r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			52+m		h, j, k, r
Via Call Gate to Different Privilege Level, (No Parameters)			86+m		h, j, k, r
Via Call Gate to Different Privilege Level, (x Parameters)			94+4x+m		h, j, k, r
From 80286 Task to 80286 TSS			273		h, j, k, r
From 80286 Task to Am386DXL CPU TSS			298		h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386DXL CPU TSS)			218		h, j, k, r
From Am386DXL CPU Task to 80286 TSS			273		h, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS		300		h, j, k, r	
From Am386DXL CPU Task to Virtual 8086 Task (Am386DXL CPU TSS)		218		h, j, k, r	
Indirect Intersegment	11111111 mod 011 r/m	22+m	38+m	b	h, j, k, r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			56+m		h, j, k, r
Via Call Gate to Different Privilege Level (No Parameters)			90+m		h, j, k, r
Via Call Gate to Different Privilege Level (x Parameters)			98+4x+m		h, j, k, r
From 80286 Task to 80286 TSS			278		h, j, k, r
From 80286 Task to Am386DXL CPU TSS			303		h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386DXL CPU TSS)			222		h, j, k, r
From Am386DXL CPU Task to 80286 TSS			278		h, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS		305		h, j, k, r	
From Am386DXL CPU Task to Virtual 8086 Task (Am386DXL CPU TSS)		222		h, j, k, r	
JMP = Unconditional Jump					
Short	11101011 8-bit displacement	7+m	7+m		r
Direct within Segment	11101001 full displacement	7+m	7+m		r
Register/Memory Indirect within Segment	11111111 mod 100 r/m	7+m 10+m	7+m 10+m	b	h,r
Direct Intersegment	11101010 unsigned full offset, selector	12+m	27+m		j, k, r

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

				Clock Count		Comments	
Instruction	**Format**			**Real Address Mode**	**Protected Virtual Address Mode**	**Real Address Mode**	**Protected Virtual Address Mode**
CONTROL TRANSFER (continued)							
Protected Mode Only (Direct Intersegment)							
Via Call Gate to Same Privilege Level					45 + m		h, j, k, r
From 80286 Task to 80286 TSS					274		h, j, k, r
From 80286 Task to Am386DXL CPU TSS					301		h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386DXL CPU TSS)					219		h, j, k, r
From Am386DXL CPU Task to 80286 TSS					270		h, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS				303		h, j, k, r	
From Am386DXL CPU Task to Virtual 8086 Task (Am386DXL CPU TSS)				221		h, j, k, r	
Indirect Intersegment	11111111	mod 1 0 1 r/m		17 + m	31 + m	b	h, j, k, r
Protected Mode Only (Indirect Intersegment)							
Via Call Gate to Same Privilege Level					49 + m		h, j, k, r
From 80286 Task to 80286 TSS					279		h, j, k, r
From 80286 Task to Am386DXL CPU TSS					306		h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386DXL CPU TSS)					223		h, j, k, r
From Am386DXL CPU Task to 80286 TSS					275		h, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS				308		h, j, k, r	
From Am386DXL CPU Task to Virtual 8086 Task (Am386DXL CPU TSS)				225		h, j, k, r	
RET = Return from CALL							
Within Segment	11000011			10 + m	10 + m	b	g, h, r
Within Seg. Adding Immediate to SP	11000010	16-bit displacement		10 + m	10 + m	b	g, h, r
Intersegment	11001011			18 + m	32 + m	b	g, h, j, k, r
Intersegment Adding Immediate to SP	11001010	16-bit displacement		18 + m	32 + m	b	g, h, j, k, r
Protected Mode Only (RET) to Different Prilege Level							
Intersegment					69		h, j, k, r
Intersegment Adding Immediate to SP					69		h, j, k, r
CONDITIONAL JUMPS (Note: Times are Jump "Taken or Not Taken")							
JO = Jump on Overflow							
8-bit Displacement	01110000	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000000	full displacement	7+m or 3	7+m or 3		r
JNO = Jump on Not Overflow							
8-bit Displacement	01110001	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000001	full displacement	7+m or 3	7+m or 3		r
JB/JNAE = Jump on Below/Not Above or Equal							
8-bit Displacement	01110010	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000010	full displacement	7+m or 3	7+m or 3		r
JNB/JAE = Jump on Not Below/Above or Equal							
8-bit Displacement	01110011	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000011	full displacement	7+m or 3	7+m or 3		r
JE/JZ = Jump on Equal/ Zero							
8-bit Displacement	01110100	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000100	full displacement	7+m or 3	7+m or 3		r
JNE/JNZ = Jump on Not Equal/Not Zero							
8-bit Displacement	01110 101	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000101	full displacement	7+m or 3	7+m or 3		r
JBE/JNA = Jump on Below or Equal/Not Above							
8-bit Displacement	01110110	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000110	full displacement	7+m or 3	7+m or 3		r

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

Instruction	Format			Clock Count		Comments	
				Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
CONDITIONAL JUMPS (continued)							
JNBE/JA = Jump on Not Below or Equal/Above							
8-bit Displacement	01110111	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10000111	full displacement	7+m or 3	7+m or 3		r
JS = Jump on Sign							
8-bit Displacement	01111000	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001000	full displacement	7+m or 3	7+m or 3		r
JNS = Jump on Not Sign							
8-bit Displacement	01111001	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001001	full displacement	7+m or 3	7+m or 3		r
JP/JPE = Jump on Parity/Parity Even							
8-bit Displacement	01111010	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001010	full displacement	7+m or 3	7+m or 3		r
JNP/JPO = Jump on Not Parity/Parity Odd							
8-bit Displacement	01111011	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001011	full displacement	7+m or 3	7+m or 3		r
JL/JNGE = Jump on Less/Not Greater or Equal							
8-bit Displacement	01111100	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001100	full displacement	7+m or 3	7+m or 3		r
JNL/JGE = Jump on Not Less/Greater or Equal							
8-bit Displacement	01111101	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001101	full displacement	7+m or 3	7+m or 3		r
JLE/JNG = Jump on Less or Equal/Not Greater							
8-bit Displacement	01111110	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001110	full displacement	7+m or 3	7+m or 3		r
JNLE/JG = Jump on Not Less or Equal/Greater							
8-bit Displacement	01111111	8-bit displacement		7+m or 3	7+m or 3		r
Full Displacement	00001111	10001111	full displacement	7+m or 3	7+m or 3		r
JCXZ = Jump on CX Zero *	11100011	8-bit displacement		9+m or 5	9+m or 5		r
JECXZ = Jump on ECX Zero *	11100011	8-bit displacement		9+m or 5	9+m or 5		r
LOOP = Loop CX Times	11100010	8-bit displacement		11 + m	11 + m		r
LOOPZ/LOOPE = Loop with Zero/Equal	11100001	8-bit displacement		11 + m	11 + m		r
LOOPNZ/LOOPNE = Loop while Not Zero	11100000	8-bit displacement		11 + m	11 + m		r
CONDITIONAL BYTE SET (Note: Times are Register/Memory)							
SETO = Set Byte on Overflow							
To Register/Memory	00001111	10010000	mod 000 r/m	4/5	4/5		h
SETNO = Set Byte on Not Overflow							
To Register/Memory	00001111	10010001	mod 000 r/m	4/5	4/5		h
SETB/SETNAE = Set Byte on Below/Not Above or Equal							
To Register/Memory	00001111	10010010	mod 000 r/m	4/5	4/5		h

* Address Size Prefix Differentiates JCXZ from JECXZ.

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

Instruction	Format			Clock Count		Comments	
				Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
CONDITIONAL BYTE SET (continued)							
SETNB = Set Byte on Not Below/Above or Equal							
To Register/Memory	00001111	10010011	mod 000 r/m	4/5	4/5		h
SETE/SETZ = Set Byte on Equal/Zero							
To Register/Memory	00001111	10010100	mod 000 r/m	4/5	4/5		h
SETNE/SETNZ = Set Byte on Not Equal/Not Zero							
To Register/Memory	00001111	10010101	mod 000 r/m	4/5	4/5		h
SETBE/SETNA = Set Byde on Below or Equal/Not Above							
To Register/Memory	00001111	10010110	mod 000 r/m	4/5	4/5		h
SETNBE/SETA = Set Byte on Not Below or Equal/Above							
To Register/Memory	00001111	10010111	mod 000 r/m	4/5	4/5		h
SETS = Set Byte on Sign							
To Register/Memory	00001111	10011000	mod 000 r/m	4/5	4/5		h
SETNS = Set Byte on Not Sign							
To Register/Memory	00001111	10011001	mod 000 r/m	4/5	4/5		h
SETP/SETPE = Set Byte on Parity/Parity Even							
To Register/Memory	00001111	10011010	mod 000 r/m	4/5	4/5		h
SETNP/SETPO = Set Byte on Not Parity/Parity Odd							
To Register/Memory	00001111	10011011	mod 000 r/m	4/5	4/5		h
SETL/SETNGE = Set Byte on Less/Not Greater or Equal							
To Register/Memory	00001111	10011100	mod 000 r/m	4/5	4/5		h
SETNL/SETGE = Set Byte on Not Less/Greater or Equal							
To Register/Memory	00001111	01111101	mod 000 r/m	4/5	4/5		h
SETLE/SETNG = Set Byte on Less or Equal/Not Greater							
To Register/Memory	00001111	10011110	mod 000 r/m	4/5	4/5		h
SETNLE/SETG = Set Byte on Not Less or Equal/Greater							
To Register/Memory	00001111	10011111	mod 000 r/m	4/5	4/5		h
ENTER = Enter Procedure	11001000	16-bit displacement, 8-bit level					
L = 0				10	10	b	h
L = 1				12	12	b	h
L > 1				15+4(n−1)	15+4(n−1)	b	h
LEAVE = Leave Procedure	11001001			4	4	b	h
INTERRUPT INSTRUCTIONS							
INT = Interrupt:							
Type Specified	11001101	type		37		b	
Type 3	11001100			33		b	
INTO = Interrupt 4 if Overflow Flag Set	11001110						
If OF = 1				35		b, e	
If OF = 0				3	3	b, e	

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

		Clock Count		Comments	
Instruction	Format	Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS (continued)					
Bound = Interrupt 5 if Detect Value Out of Range	01100010 | mod reg r/m				
If Out of Range		44		b, e	e, g, h, j, k, r
If in Range		10	10	b, e	e, g, h, j, k, r
Protected Mode Only (INT)					
INT: Type Specified					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			282		g, j, k, r
From 80286 Task to Am386DXL CPU TSS via Task Gate			309		g, j, k, r
From 80286 Task to Virtual 8086 Mode via Task Gate			226		g, j, k, r
From Am386DXL CPU Task to 80286 TSS via Task Gate			284		g, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS via Task Gate			311		g, j, k, r
From Am386DXL CPU Task to Virtual 8086 Mode via Task Gate			228		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			289		g, j, k, r
From Virtual 8086 Mode to Am386DXL CPU TSS via Task Gate			316		g, j, k, r
From Virtual 8086 Mode to Privilege Level 0 via Trap Gate or Interrupt Gate			119		g, j, k, r
INT: Type 3					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			278		g, j, k, r
From 80286 Task to Am386DXL CPU TSS via Task Gate			305		g, j, k, r
From 80286 Task to Virtual 8086 Mode via Task Gate			222		g, j, k, r
From Am386DXL CPU Task to 80286 TSS via Task Gate			280		g, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS via Task Gate			307		g, j, k, r
From Am386DXL CPU Task to Virtual 8086 Mode via Task Gate			224		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			285		g, j, k, r
From Virtual 8086 Mode to Am386DXL CPU TSS via Task Gate			312		g, j, k, r
From Virtual 8086 Mode to Privilege Level 0 via Trap Gate or Interrupt Gate			119		g, j, k, r
INTO					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			280		g, j, k, r
From 80286 Task to Am386DXL CPU TSS via Task Gate			307		g, j, k, r
From 80286 Task to Virtual 8086 Mode via Task Gate			224		g, j, k, r
From Am386DXL CPU Task to 80286 TSS via Task Gate			282		g, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS via Task Gate			309		g, j, k, r
From Am386DXL CPU Task to Virtual 8086 Mode via Task Gate			225		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			287		g, j, k, r
From Virtual 8086 Mode to Am386DXL CPU TSS via Task Gate			314		g, j, k, r
From Virtual 8086 Mode to Privilege Level 0 via Trap Gate or Interrupt Gate			119		g, j, k, r
BOUND					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			254		g, j, k, r
From 80286 Task to Am386DXL CPU TSS via Task Gate			284		g, j, k, r
From 80286 Task to Virtual 8086 Mode via Task Gate			231		g, j, k, r
From Am386DXL CPU Task to 80286 TSS via Task Gate			264		g, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS via Task Gate			294		g, j, k, r
From Am386DXL CPU Task to Virtual 8086 Mode via Task Gate			243		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			264		g, j, k, r
From Virtual 8086 Mode to Am386DXL CPU TSS via Task Gate			294		g, j, k, r
From Virtual 8086 Mode to Privilege Level 0 via Trap Gate or Interrupt Gate			119		g, j, k, r
INTERRUPT RETURN					
IRET = Interrupt Return	11001111	22			g, h, j, k, r
Protected Mode Only (IRET)					
To the Same Privilege Level (within Task)			38		g, h, j, k, r
To Different Privilege Level (within Task)			82		g, h, j, k, r
From 80286 Task to 80286 TSS			232		h, j, k, r
From 80286 Task to Am386DXL CPU TSS			265		h, j, k, r
From 80286 Task to Virtual 8086 Task			213		h, j, k, r
From 80286 Task to Virtual 8086 Mode (within Task)			60		
From Am386DXL CPU Task to 80286 TSS			271		h, j, k, r
From Am386DXL CPU Task to Am386DXL CPU TSS		275		h, j, k, r	
From Am386DXL CPU Task to Virtual 8086 Task		223		h, j, k, r	
From Am386DXL CPU Task to Virtual 8086 Mode (within Task)			60		

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

				Clock Count		Comments	
Instruction	**Format**			**Real Address Mode**	**Protected Virtual Address Mode**	**Real Address Mode**	**Protected Virtual Address Mode**
PROCESSOR CONTROL							
HLT = HALT	11110100			5	5		l
MOV = Move to and From Control/Debug/Test Registers							
CR0/CR2/CR3 from register	00001111	00100010	11 eee reg	11/4/5	11/4/5		l
Register From CR3–0	00001111	00100000	11 eee reg	6	6		l
DR3–0 From Register	00001111	00100011	11 eee reg	22	22		l
DR7–6 From Register	00001111	00100011	11 eee reg	16	16		l
Register from DR7–6	00001111	00100001	11 eee reg	14	14		l
Register from DR3–0	00001111	00100001	11 eee reg	22	22		l
TR7–6 from Register	00001111	00100110	11 eee reg	12	12		l
Register from TR7–6	00001111	00100100	11 eee reg	12	12		l
NOP = No Operation	10010000			3	3		
WAIT = Wait until $\overline{\text{BUSY}}$ pin is negated	10011011			7	7		
NOP = No Operation	10010000			3	3		
PROCESSOR EXTENSION INSTRUCTIONS—See coprocessor datasheets							
Processor Extension Escape	11011TTT	mod LLL r/m					h
	TTT and LLL bits are op-code information for coprocessor						
PREFIX BYTES							
Address Size Prefix	01100111			0	0		
LOCK = Bus Lock Prefix	11110000			0	0		m
Operand Size Prefix	01100110			0	0		
Segment Override Prefix							
CS:	00101110			0	0		
DS:	00111110			0	0		
ES:	00100110			0	0		
FS:	01100100			0	0		
GS:	01100101			0	0		
SS:	00110110			0	0		
PROTECTION CONTROL							
ARPL = Adjust Requested Privilege Level							
From Register/Memory	01100011	mod reg r/m		N/A	20/21	a	h
LAR = Load Access Rights							
From Register/Memory	00001111	00000010	mod reg r/m	N/A	15/16	a	g, h, j, p
LGDT = Load Global Descriptor							
Table Register	00001111	00000001	mod 010 r/m	11	11	b,c	h,l
LIDT = Load Interrupt Descriptor							
Table Register	00001111	00000001	mod 011 r/m	11	11	b, c	h, l
LLDT = Load Local Descriptor							
Table Register to Register/Memory	00001111	00000000	mod 010 r/m	N/A	20/24	a	g, h, j, l
LMSW = Load Machine Status Word							
From Register/Memory	00001111	00000001	mod 110 r/m	11/14	11/14	b.c	h, l

Table 23. Am386DXL Microprocessor Instruction Set Summary (continued)

Instruction	Format			Clock Count		Comments	
				Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
PROTECTION CONTROL (continued)							
LSL = Load Segment Limit							
From Register/Memory	00001111	00000011	mod reg r/m				
Byte-Granular Limit				N/A	21/22	a	g, h, j, p
Page-Granular Limit				N/A	25/26	a	g, h, j, p
LTR = Load Task Register							
From Register/Memory	00001111	00000000	mod 001 r/m	N/A	23/27	a	g, h, j, l
SGDT = Store Global Descriptor							
Table Register	00001111	00000001	mod 000 r/m	9	9	b, c	h
SIDT = Store Interrupt Descriptor							
Table Register	00001111	00000001	mod 001 r/m	9	9	b, c	h
SLDT = Store Local Descriptor Table Register							
To Register/Memory	00001111	00000000	mod 000 r/m	N/A	2/2	a	h
SMSW = Store Machine Status Word	00001111	00000001	mod 100 r/m	2/2	2/2	b, c	h, l
STR = Store Task Register							
To Register/Memory	00001111	00000000	mod 001 r/m	N/A	2/2	a	h
VERR = Verify Read Access							
Register/Memory	00001111	00000000	mod 100 r/m	N/A	10/11	a	g, h, j, p
VERW = Verify Write Access	00001111	00000000	mod 101 r/m	N/A	15/16	a	g, h, j, p

Instruction Notes for Table 23.

Notes a through c apply to Am386DXL CPU Real Address Mode only.

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in Exception 6 (Invalid op-code).

b. Exception 13 fault (General Protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS, or GS limit, FFFFH. Exception 12 (fault stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Notes d through g apply to Am386DXL CPU Real Address Mode and Am386DXL CPU Protected Virtual Address Mode.

d. The Am386DXL CPU uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).
Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:
Actual Clock = if $m <> 0$ then max ($[\log_2 |m|]$, 3) + b clocks: if $m = 0$ then $3 + b$ clocks
In this formula, *m* is the multiplier, and
$b = 9$ for register to register,
$b = 12$ for memory to register,
$b = 10$ for register with immediate to register,
$b = 11$ for memory with immediate to register.

e. An Exception may occur, depending on the value of the operand.

f. $\overline{\text{LOCK}}$ is automatically asserted, regardless of the presence or absence of the $\overline{\text{LOCK}}$ prefix.

g. $\overline{\text{LOCK}}$ is asserted during descriptor table accesses.

Notes h through r apply to Am386DXL CPU Protected Virtual Address Mode only.

h. Exception 13 fault (General Protection Violation) will occur if the memory operand in CS, DS, ES, FS, or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, an Exception 12 (Stack Segment Limit Violation or Not Present) occurs.

i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an Exception 13 fault (General Protection Violation). The segment's descriptor must indicate present or Exception 11 (CS, DS, ES, FS, GS Not Present). If the SS register is loaded and a stack segment not present is detected, an Exception 12 (Stack Segment Limit Violation or Not Present) occurs.

j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert $\overline{\text{LOCK}}$ to maintain descriptor integrity in multiprocessor systems.

k. JMP, CALL, INT, RET, and IRET instructions referring to another code segment will cause an Exception 13 (General Protection Violation) is an applicable privilege rule is violated.

l. An Exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).

m. An Exception 13 fault occurs if CPL is greater than IOPL.

n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.

o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 if desiring to reset the PE bit.

p. Any violation of privilege rules as applied to the selector operand does not cause a protection Exception; rather, the zero flag is cleared.

q. If the coprocessor's memory operand violates a segment limit or segment access rights, an Exception 13 fault (General Protection Exception) will occur before the ESC instruction is executed. An Exception 12 fault (Stack Segment Limit Violation or Not Present) will occur if the stack limit is violated by the operand's starting address.

r. The destination of a JMP, CALL, INT, RET, or IRET must be in the defined limit of a code segment or an Exception 13 fault (General Protection Violation) will occur.

Instruction Encoding

Overview

All instruction encodings are subsets of the general instruction format shown in Figure 83. Instructions consist of one or two primary op-code 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 op-code or op-codes, 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 op-code 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 instruction.

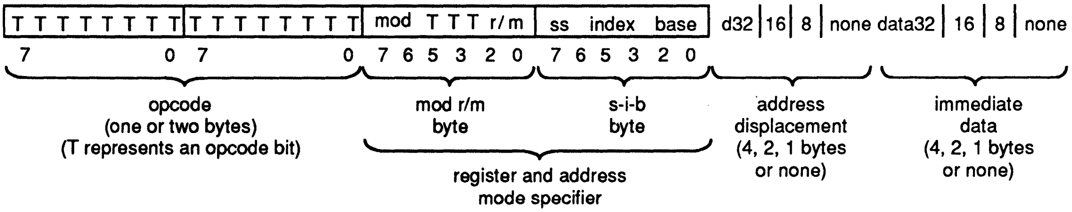
Figure 83 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the Figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the op-code bytes themselves. Table 24 is a complete list of all fields appearing in the Am386DXL microprocessor instruction set. Further ahead, following Table 24, are detailed tables for each field.

32-Bit Extensions of the Instruction Set

With the Am386DXL microprocessor, the 8086/80186/80286 instruction set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types, and 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

Whether the instruction defaults to operations of 16 or 32 bits depends on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 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 Am386DXL microprocessor when operating in those modes (for 16-bit default sizes compatible with the 8086/80C186/80286).

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 op-code bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed before the op-code 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.



15021B–085

Figure 83. General Instruction Format

Table 24. Fields within Am386DXL Microprocessor Instructions

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 bits)	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Condition Instructions, specifies a Condition Asserted or a Condition Negated	4

Note: Table 23 shows encoding of individual instructions.

These 32-bit extensions are available in all Am386DXL microprocessor modes, including the Real Address Mode or the Virtual 8086 Mode. In these modes the default is always 16 bits, so prefixes are needed to specify 32-bit operands or addresses. For instructions with more than one prefix, the order of prefixes is unimportant.

Unless specified otherwise, instructions with 8- and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on. The exact encodings of these fields are defined immediately ahead.

Encoding of Operand Length (w) Field

For any given instruction performing a data operation, the instruction is executing as a 32- or 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

Encoding of The General Register (reg) Field

The general register is specified by the reg field, which may appear in the primary op-code bytes, or as the reg field of the mod r/m byte, or as the r/m field of the mod r/m byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations		
	Function of w Field	
reg	(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		
	Function of w Field	
reg	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

Encoding of The Segment Register (sreg) Field

The sreg field in certain instructions is a 2-bit field allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the Am386DXL microprocessor FS and GS segment registers to be specified.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

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 op-code. 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 83) sometimes used as an extension of the primary op-code. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the mod r/m byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the mod r/m byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following pages define all encodings of all 16- and 32-bit addressing modes.

Encoding of 16-Bit Address Mode with mod r/m Byte

mod	r/m	Effective Address
00	000	DS:[BX + SI]
00	001	DS:[BX + DI]
00	010	SS:[BP + SI]
00	011	DS:[BP + DI]
00	100	DS:[SI]
00	101	DS:[DI]
00	110	DS:d16
00	111	DS:[BX]
01	000	DS:[BX + SI + d8]
01	001	DS:[BX + DI+d8]
01	010	SS:[BP + SI + d8]
01	011	SS:[BP + DI + d8]
01	100	DS:[SI + d8]
01	101	DS:[DI + d8]
01	110	SS:[BP + d8]
01	111	DS:[BX + d8]

mod	r/m	Effective Address
10	000	DS:[BX + SI + d16]
10	001	DS:[BX + DI + d16]
10	010	SS:[BP + SI + d16]
10	011	SS:[BP+DI+d16]
10	100	DS:[SI + d16]
10	101	DS:[DI + d16]
10	110	SS:[BP + d16]
10	111	DS:[BX + d16]
11	000	Register—See Below
11	001	Register—See Below
11	010	Register—See Below
11	011	Register—See Below
11	100	Register—See Below
11	101	Register—See Below
11	110	Register—See Below
11	111	Register—See Below

Register Specified by r/m During 16-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-Bit Address Mode with mod r/m byte (No s-i-b Byte Present)

mod	r/m	Effective Address
00	000	DS:[EAX]
00	001	DS:[ECX]
00	010	DS:[EDX]
00	011	DS:[EBX]
00	100	s-i-b is present
00	101	DS:d32
00	110	DS:[ESI]
00	111	DS:[EDI]
01	000	DS:[EAX + d8]
01	001	DS:[ECX + d8]
01	010	DS:[EDX + d8]
01	011	DS:[EBX + d8]
01	100	s-i-b is present
01	101	SS:[EBP + d8]
01	110	DS:[ESI + d8]
01	111	DS:[EDI + d8]

mod	r/m	Effective Address
10	000	DS:[EAX + d32]
10	001	DS:[ECX + d32]
10	010	DS:[EDX + d32]
10	011	DS:[EBX + d32]
10	100	s-i-b is present
10	101	SS:[EBP + d32]
10	110	DS:[ESI + d32]
10	111	DS:[EDI + d32]
11	000	Register—See Below
11	001	Register—See Below
11	010	Register—See Below
11	011	Register—See Below
11	100	Register—See Below
11	101	Register—See Below
11	110	Register—See Below
11	111	Register—See Below

Register Specified by reg or r/m During 16-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m During 32-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-Bit Address Mode (mod r/m Byte and s-i-b 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 (see note)
101	EBP
110	ESI
111	EDI

Note: When index field is 100, indicating no index register, then ss field must equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

Encoding of Operation Direction (d) Field

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory ← Register reg Field indicates Source Operand; mod r/m or mod ss index base indicates Destination Operand.

1	Register ← Register Memory
	reg Field indicates Destination Operand;
	mod r/m or mod ss index base indicates Source Operand.

Encoding of Sign-Extend (s) Field

The *s* field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16- or 32-bit destination.

s	Effect on Immediate Data 8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extended Data 8 to fill 16-Bit or 32-Bit Destination	None

Encoding of Conditional Test (tttn) Field

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n=0) or its negation (n=1), and ttt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Not Greater Than	1110
NLE/G	Not Less Than or Equal/Greater Than	1111

Encoding of Control or Debug or Test Register (eee) Field

For the loading and storing of the Control, Debug and Test registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	CR2
011	CR3
Do not use any other encoding.	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding.	

When Interpreted as Test Register Field

eee Code	Reg Name
110	TR6
111	TR7
Do not use any other encoding.	

MECHANICAL DATA

Introduction

In this section, the physical packaging and its connections are described in detail.

Package Dimensions and Mounting

The initial Am386DXL microprocessor package is a 132-pin ceramic pin grid array (PGA). Pins of this package are arranged 0.100 inch (2.54 mm) center-to-center, in a 14 x 14 matrix, three rows around.

A wide variety of available sockets allow low insertion force or zero insertion force mountings, and a choice of terminals such as soldertail, surface mount, or wire wrap.

Package Thermal Specification

The Am386DXL microprocessor is specified for operation when ambient temperature is within the range of 0°C–100°C. The ambient temperature may be measured in any environment, to determine whether the Am386DXL microprocessor is within specified operating range.

The PGA ambient temperature should be measured at the center of the top surface opposite the pins.

ELECTRICAL DATA

Introduction

The following sections describe recommended electrical connections for the Am386DXL microprocessor and its electrical specifications.

Power and Grounding

Power Connections

The Am386DXL CPU is implemented in CS21S technology and has modest power requirements. However, its high clock frequency and 72 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 20 Vcc and 21 Vss pins separately feed functional units of the Am386DXL CPU.

Power and ground connections must be made to all external Vcc and GND pins of the Am386DXL CPU. On the circuit board, all Vcc pins must be connected on a Vcc plane. All Vss pins must be likewise connected on a GND plane.

Power Decoupling Recommendations

Liberal decoupling capacitance should be placed near the Am386DXL CPU. The Am386DXL microprocessor driving its 32-bit parallel address and data buses at high frequencies can cause transient power surges, particularly when driving large capacitive loads.

Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the Am386DXL microprocessor and decoupling capacitors as much as possible. Capacitors specifically for PGA packages are also commercially available, for the lowest possible inductance.

Resistor Recommendations

The $\overline{\text{ERROR}}$, $\overline{\text{FLT}}$, and $\overline{\text{BUSY}}$ inputs have resistor pullups of approximately 20 Kohms built into the Am386DXL CPU to keep these signals negated when no 387DX math coprocessor is present in the system (or temporarily removed from its socket). The $\overline{\text{BS16}}$ input also has an internal pullup resistor of approximately 20 Kohms, and the PEREQ input has an internal pulldown resistor of approximately 20 Kohms.

In typical designs, the external pullup resistors are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pullup resistors in other ways.

Other Connection Recommendations

For reliable operation, always connect unused inputs to an appropriate signal level. NC pins should always remain unconnected.

Particularly when not using interrupts or bus hold, (as when first prototyping, perhaps) prevent any chance of spurious activity by connecting these associated inputs to GND.

Pin	Signal
B7	INTR
B8	NMI
D14	HOLD

If not using address pipelining, pullup D13 $\overline{\text{NA}}$ to Vcc.

If not using 16-bit size, pullup C14 $\overline{\text{BS16}}$ to Vcc.

Pullups in the range of 20 Kohms are recommended.

Am386™SXL

High-Performance, 32-Bit Microprocessor with 16-Bit Data Bus

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **25- and 20-MHz operating speeds**
- **True static design for long battery life in portable PCs**
  - —0 MHz (DC) minimum frequency
  - —Typical standby (DC) current < 0.02 mA
  - —Typical operating current < 165 mA at 20 MHz
  - —Wide range of chip set and BIOS support take advantage of standby mode capabilities
- **Lower heat dissipation facilitates elimination of cooling fan in desktop PCs**
- **Compatible with 386SX systems and software**
- **Pin-for-pin replacement of the Intel i386SX**
- **Supports 387SX-compatible math coprocessors**
- **100-pin PQFP package with optional protective ring for better lead coplanarity**
- **Advanced 0.8 micron CMOS technology**

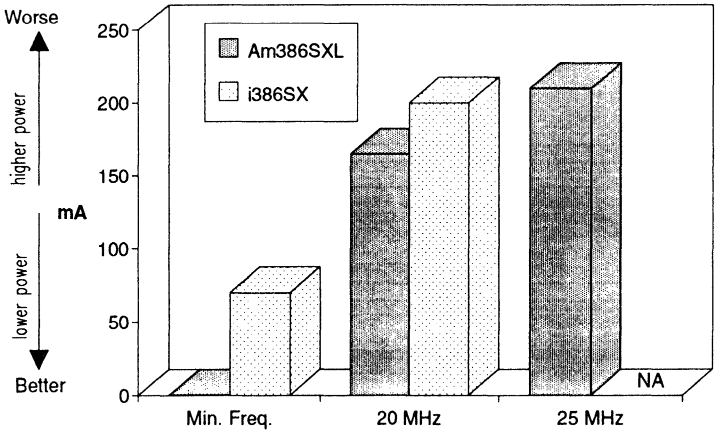
GENERAL DESCRIPTION

The Am386SXL microprocessor is a high-speed, true static implementation of the Intel i386SX. It is ideal for both desktop and battery-powered notebook personal computers. For notebooks, the Am386SXL microprocessor's true static design offers longer battery life with low operating power consumption and standby mode. At 20 MHz, this device offers a current which is 22% lower than the Intel i386SX. Standby mode allows the Am386SXL CPU to be clocked down to 0 MHz (DC) and retain full register contents. Typical current in standby mode is reduced to less than 0.02 mA—nearly a 1000x reduction in power consumption versus the Intel i386SX.

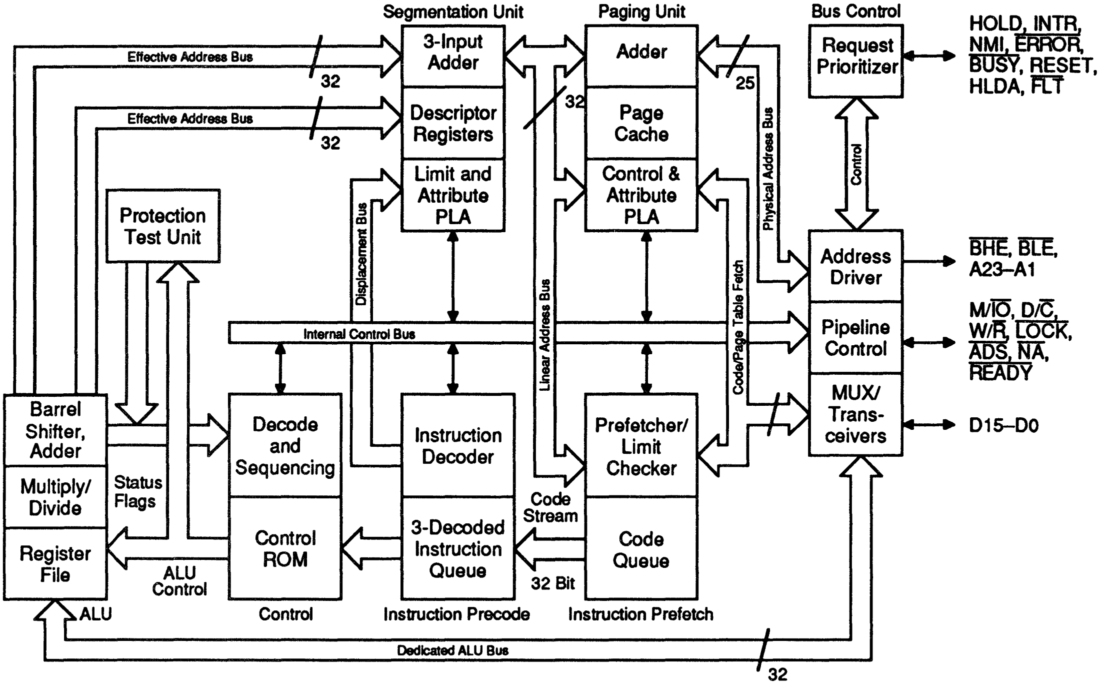
For desktop PCs, the Am386SXL microprocessor offers a 25% increase in the maximum operating speed, from 20 to 25 MHz. Also, this device offers lower heat dissipation, allowing system designers to remove or reduce the size and cost of the cooling fan.

This device will be available in a standard 100-pin plastic quad flat pack (PQFP). This package may be shipped in an optional protective ring for better lead protection during manufacturing.

Typical Icc

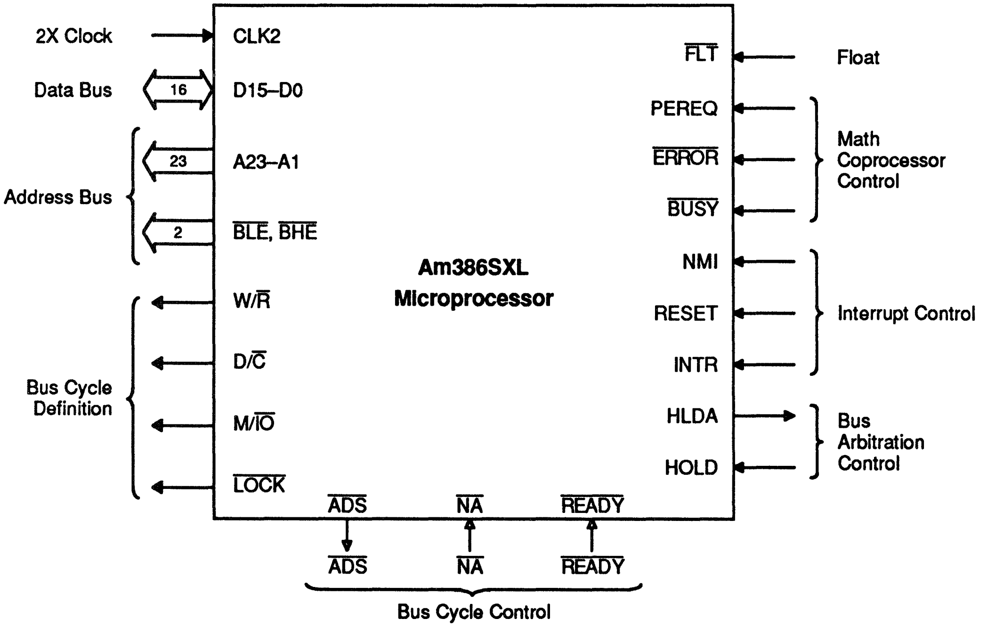


BLOCK DIAGRAM



15022B–001

LOGIC SYMBOL



15022B–003

FUNCTIONAL DESCRIPTION

True Static Operation

The Am386SXL microprocessor incorporates a true static design. Unlike dynamic circuit design, the Am386SXL CPU eliminates the minimum operating frequency restriction. It may be clocked from its maximum speed of 25 MHz all the way down to 0 MHz (DC). System designers can use this feature to design battery-powered notebook PCs with long battery life.

Standby Mode

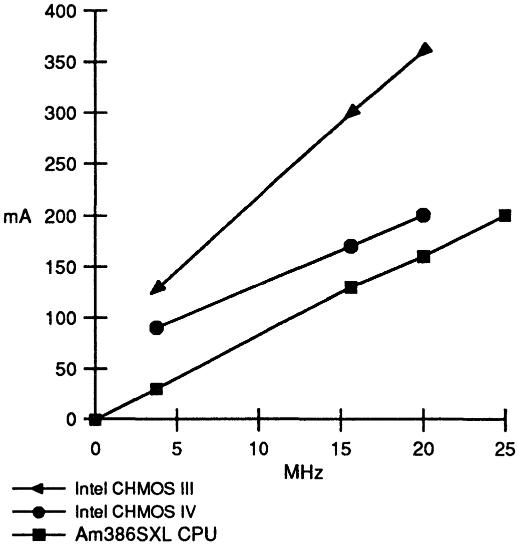
This true static design allows for a standby mode. At any operating speed (25 to 0 MHz), the Am386SXL microprocessor will retain its state (i.e., the contents of all its registers). By shutting off the clock completely, the device enters standby mode. Since power consumption is proportional to clock frequency, operating power consumption is reduced as the frequency is lowered. In standby mode, typical current draw is reduced to less than 0.02 mA at DC.

Not only does this feature improve battery life, but it also simplifies the design of power-conscious notebook computers in the following ways:

1. Eliminates the need for software in the BIOS to save and restore the contents of registers.
2. Allows simpler circuitry to control stopping of the clock since the system does not need to know what state the processor is in.

Lower Operating I_{CC}

True static design also allows lower operating I_{CC} when operating at any speed. See the following graph for typical current at operating speeds.



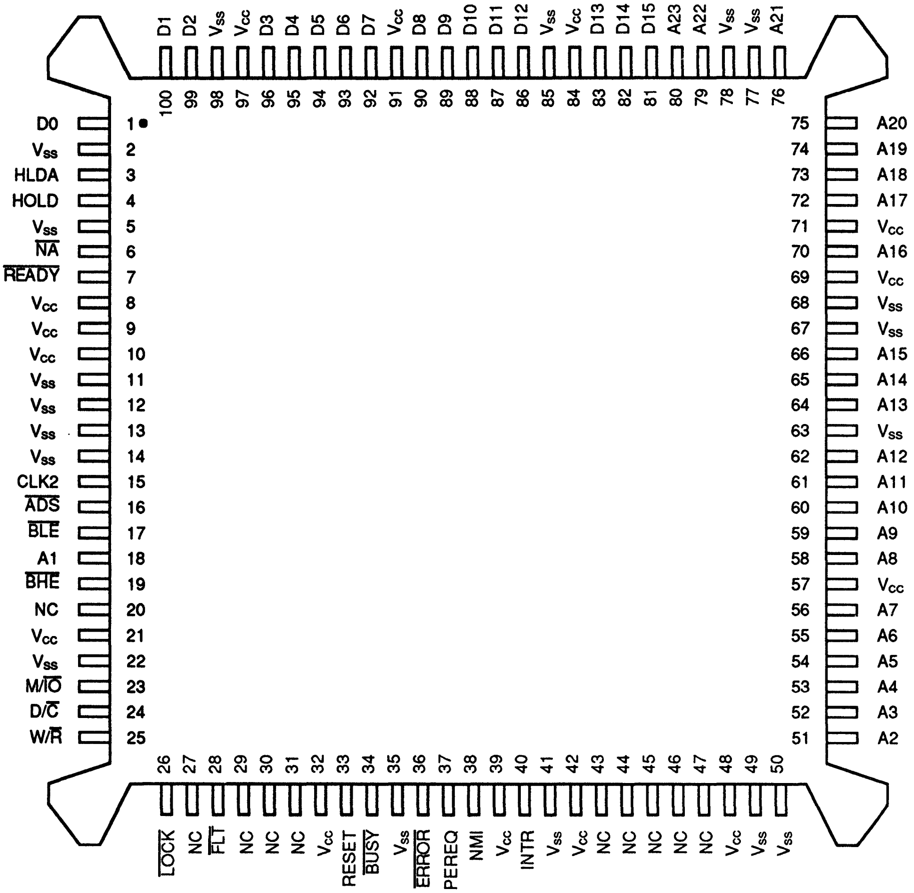
Performance On Demand

The Am386SXL microprocessor retains its state at any speed from 0 MHz (DC) to its maximum operating speed. With this feature, system designers may vary the operating speed of the system to extend the battery life in portable systems.

For example, the system could operate at low speeds during inactivity or polling operations. However, upon interrupt, the system clock can be increased up to its maximum speed. After a user-defined time-out period, the system can be returned to a low (or 0 MHz) operating speed without losing its state. This design maximizes battery life while achieving optimal performance.

CONNECTION DIAGRAM

Top View



Notes: NC = No Connect.
Pin 1 is marked for orientation.

15022B–002

PIN DESIGNATION TABLES (Sorted by Pin Name)

Address		Data		Control		NC	V_{CC}	V_{SS}
Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin No.	Pin No.	Pin No.
A1	18	D0	1	$\overline{\text{ADS}}$	16	20	8	2
A2	51	D1	100	$\overline{\text{BHE}}$	19	27	9	5
A3	52	D2	99	$\overline{\text{BLE}}$	17	29	10	11
A4	53	D3	96	$\overline{\text{BUSY}}$	34	30	21	12
A5	54	D4	95	CLK2	15	31	32	13
A6	55	D5	94	D/$\overline{\text{C}}$	24	43	39	14
A7	56	D6	93	$\overline{\text{ERROR}}$	36	44	42	22
A8	58	D7	92	$\overline{\text{FLT}}$	28	45	48	35
A9	59	D8	90	HLDA	3	46	57	41
A10	60	D9	89	HOLD	4	47	69	49
A11	61	D10	88	INTR	40		71	50
A12	62	D11	87	$\overline{\text{LOCK}}$	26		84	63
A13	64	D12	86	M/$\overline{\text{IO}}$	23		91	67
A14	65	D13	83	$\overline{\text{NA}}$	6		97	68
A15	66	D14	82	NMI	38			77
A16	70	D15	81	PEREQ	37			78
A17	72			$\overline{\text{READY}}$	7			85
A18	73			RESET	33			98
A19	74			W/$\overline{\text{R}}$	25			
A20	75							
A21	76							
A22	79							
A23	80							

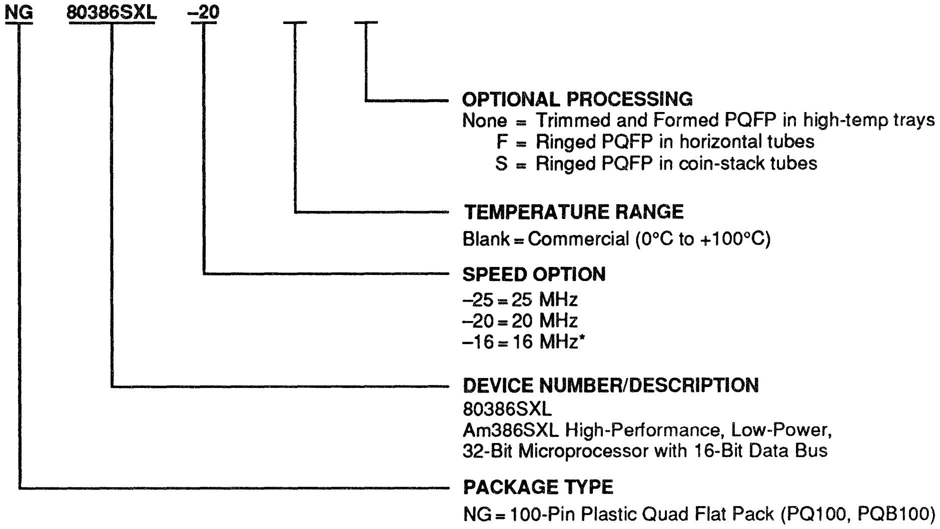
PIN DESIGNATION TABLES (Sorted by Pin Number)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	D0	21	V_{CC}	41	V_{SS}	61	A11	81	D15
2	V_{SS}	22	V_{SS}	42	V_{CC}	62	A12	82	D14
3	HLDA	23	M/$\overline{\text{IO}}$	43	NC	63	V_{SS}	83	D13
4	HOLD	24	D/$\overline{\text{C}}$	44	NC	64	A13	84	V_{CC}
5	V_{SS}	25	W/$\overline{\text{R}}$	45	NC	65	A14	85	V_{SS}
6	$\overline{\text{NA}}$	26	$\overline{\text{LOCK}}$	46	NC	66	A15	86	D12
7	$\overline{\text{READY}}$	27	NC	47	NC	67	V_{SS}	87	D11
8	V_{CC}	28	$\overline{\text{FLT}}$	48	V_{CC}	68	V_{SS}	88	D10
9	V_{CC}	29	NC	49	V_{SS}	69	V_{CC}	89	D9
10	V_{CC}	30	NC	50	V_{SS}	70	A16	90	D8
11	V_{SS}	31	NC	51	A2	71	V_{CC}	91	V_{CC}
12	V_{SS}	32	V_{CC}	52	A3	72	A17	92	D7
13	V_{SS}	33	RESET	53	A4	73	A18	93	D6
14	V_{SS}	34	$\overline{\text{BUSY}}$	54	A5	74	A19	94	D5
15	CLK2	35	V_{SS}	55	A6	75	A20	95	D4
16	$\overline{\text{ADS}}$	36	$\overline{\text{ERROR}}$	56	A7	76	A21	96	D3
17	$\overline{\text{BLE}}$	37	PEREQ	57	V_{CC}	77	V_{SS}	97	V_{CC}
18	A1	38	NMI	58	A8	78	V_{SS}	98	V_{SS}
19	$\overline{\text{BHE}}$	39	V_{CC}	59	A9	79	A22	99	D2
20	NC	40	INTR	60	A10	80	A23	100	D1

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



NG 80386SXL -20

OPTIONAL PROCESSING
None = Trimmed and Formed PQFP in high-temp trays
F = Ringed PQFP in horizontal tubes
S = Ringed PQFP in coin-stack tubes

TEMPERATURE RANGE
Blank = Commercial (0°C to +100°C)

SPEED OPTION
–25 = 25 MHz
–20 = 20 MHz
–16 = 16 MHz*

DEVICE NUMBER/DESCRIPTION
80386SXL
Am386SXL High-Performance, Low-Power,
32-Bit Microprocessor with 16-Bit Data Bus

PACKAGE TYPE
NG = 100-Pin Plastic Quad Flat Pack (PQ100, PQB100)

Valid Combinations		
NG	80386SXL	–25 –20 –16*
		–25F –20F –16F*
		–25S –20S –16S*

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations.

*Contact AMD for 16-MHz availability.

PIN DESCRIPTIONS

A23–A1

Address Bus (Outputs)

Outputs physical memory or port I/O addresses.

$\overline{\text{ADS}}$

Address Status (Active Low; Output)

Indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, and A23–A1) are being driven at the Am386SXL microprocessor pins.

$\overline{\text{BHE}}$, $\overline{\text{BLE}}$

Byte Enables (Active Low; Outputs)

Indicate which data bytes of the data bus take part in a bus cycle.

$\overline{\text{BUSY}}$

Busy (Active Low; Input)

Signals a busy condition from a processor extension.

CLK2

CLK2 (Input)

Provides the fundamental timing for the Am386SXL microprocessor.

D15–D0

Data Bus (Inputs/Outputs)

Inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles.

D/$\overline{\text{C}}$

Data/Control (Output)

A bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and code fetch.

$\overline{\text{ERROR}}$

Error (Active Low; Input)

Signals an error condition from a processor extension.

$\overline{\text{FLT}}$

Float (Active Low; Input)

An input which forces all bi-directional and output signals, including HLDA, to the three-state condition.

HLDA

Bus Hold Acknowledge (Active High; Output)

Output indicates that the Am386SXL microprocessor has surrendered control of its logical bus to another bus master.

HOLD

Bus Hold Request (Active High; Input)

Input allows another bus master to request control of the local bus.

INTR

Interrupt Request (Active High; Input)

A maskable input that signals the Am386SXL microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

$\overline{\text{LOCK}}$

Bus Lock (Active Low; Output)

A bus cycle definition pin that indicates that other system bus masters are not to gain control of the system bus while it is active.

M/$\overline{\text{IO}}$

Memory/IO (Output)

A bus cycle definition pin that distinguishes memory cycles from input/output cycles.

$\overline{\text{NA}}$

Next Address (Active Low; Input)

Used to request address pipelining.

NC

No Connect

Should always be left unconnected. Connection of a NC pin may cause the processor to malfunction or be incompatible with future steppings of the Am386SXL microprocessor.

NMI

Non-Maskable Interrupt Request (Active High; Input)

A non-maskable input that signals the Am386SXL microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

PEREQ

Processor Extension Request (Active High; Input)

Indicates that the processor has data to be transferred by the Am386SXL microprocessor.

$\overline{\text{READY}}$

Bus Ready (Active Low; Input)

Terminates the bus cycle.

RESET

Reset (Active High; Input)

Suspends any operation in progress and places the Am386SXL microprocessor in a known reset state.

V_{cc}

System Power (Active High; Input)

Provides the +5 V nominal DC supply input.

V_{ss}

System Ground (Input)

Provides the 0 V connection from which all inputs and outputs are measured.

$W/\overline{R}$

Write/Read (Output)

A bus cycle definition pin that distinguishes write cycles from read cycles.

INTRODUCTION

The Am386SXL microprocessor is 100% object-code compatible with the Am386DX, 286, and 8086 microprocessors. System manufacturers can provide Am386DX CPU based systems optimized for performance and Am386SXL CPU based systems optimized for cost, both sharing the same operating systems and application software. Systems based on the Am386SXL microprocessor can access the world's largest existing microcomputer software base. Only the Am386DX CPU architecture can run UNIX, OS/2, and MS-DOS.

Instruction pipelining, high-bus bandwidth, and a very high-performance ALU ensure short average instruction execution times and high system throughput. The Am386SXL CPU is capable of execution at sustained rates of 2.5–3.0 million instructions per second.

The integrated Memory Management Unit (MMU) includes an address translation cache, advanced multitasking hardware, and a four-level hardware-enforced protection mechanism to support operating systems. The virtual machine capability of the Am386SXL CPU allows simultaneous execution of applications from multiple operating systems such as MS-DOS and UNIX.

The Am386SXL CPU offers on-chip testability and debugging features. Four breakpoint registers allow conditional or unconditional breakpoint traps on code execution or data accesses for powerful debugging of even ROM-based systems. Other testability features include self-test, three-state of output buffers, and direct access to the page translation cache.

BASE ARCHITECTURE

The Am386SXL microprocessor consists of a central processing unit, a memory management unit, and a bus interface.

The central processing unit consists of the execution unit and the instruction unit. The execution unit contains the eight 32-bit general purpose registers which are used for both address calculation and data operations and a 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations. The instruction unit decodes the instruction op-codes and stores them in the decoded instruction queue for immediate use by the execution unit.

The MMU consists of a segmentation unit and a paging unit. Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows easy code and data relocatability, and efficient sharing. The paging mechanism operates beneath and is transparent to the segmentation process, to allow management of the physical address space.

The segmentation unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of integrity.

The Am386SXL microprocessor has two modes of operation: Real Address Mode (Real Mode) and Protected Virtual Address Mode (Protected Mode). In Real Mode the Am386SXL CPU operates as a very fast 8086, but with 32-bit extensions, if desired. Real Mode is required primarily to set up the processor for Protected Mode operation.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each such task behaves with 8086 semantics, thus allowing 8086 software (an application program or an entire operating system) to execute. The Virtual 8086 tasks can be isolated and protected from one another and the host Am386SXL microprocessor operating system by use of paging.

Finally, to facilitate high-performance system hardware designs, the Am386SXL microprocessor bus interface offers address pipelining and direct Byte Enable signals for each byte of the data bus.

Register Set

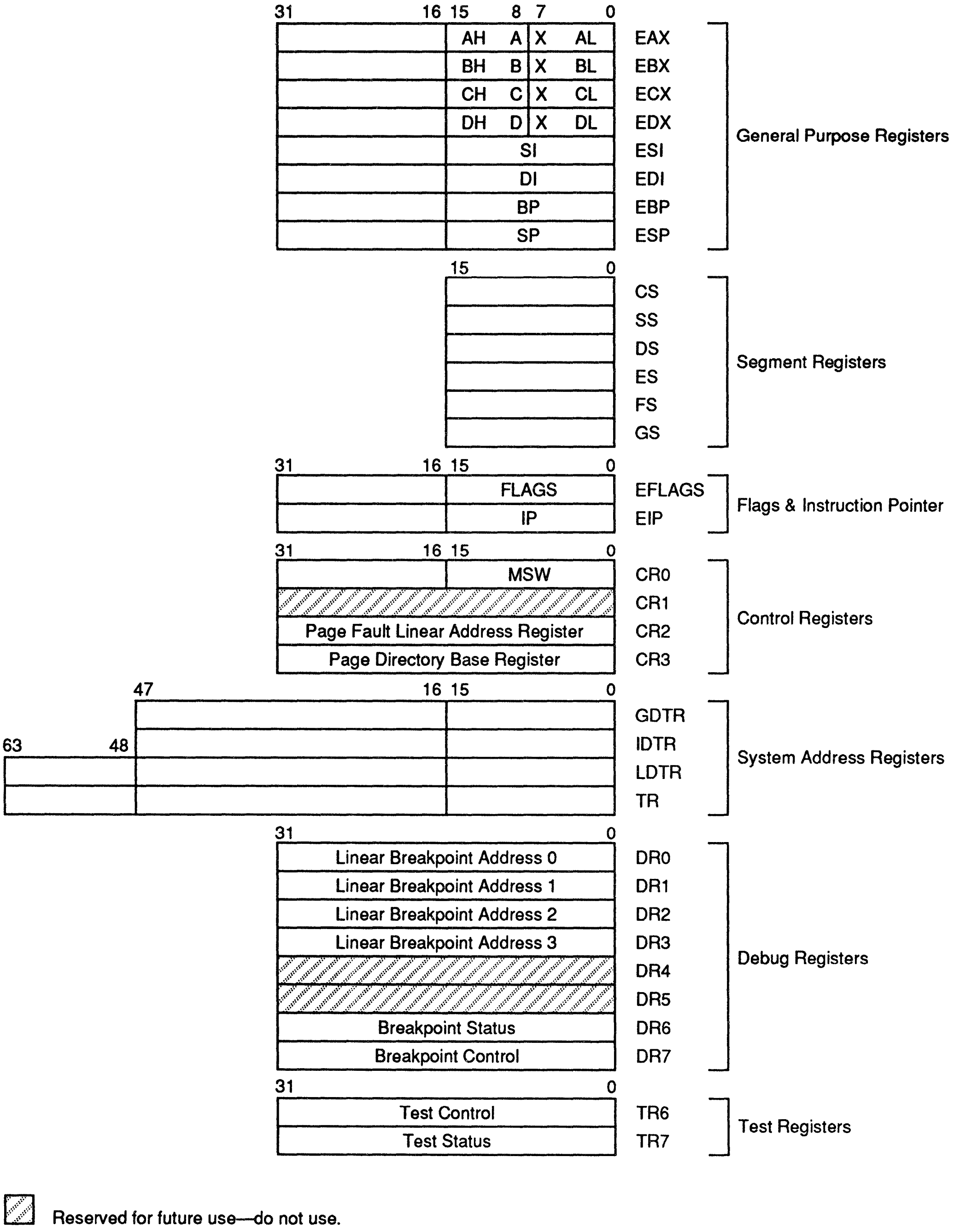
The Am386SXL microprocessor has 34 registers as shown in Figure 1. These registers are grouped into the following seven categories:

General Purpose Registers: The eight 32-bit general purpose registers are used to contain arithmetic and logical operands. Four of these (EAX, EBX, ECX, and EDX) can be used either in their entirety as 32-bit registers, as 16-bit registers, or split into pairs of separate 8-bit registers.

Segment Registers: Six 16-bit special purpose registers select, at any given time, the segments of memory that are immediately addressable for code, stack, and data.

Flags and Instruction Pointer Registers: The two 32-bit special purpose registers in Figure 1 record or control certain aspects of the Am386SXL microprocessor state. The EFLAGS register includes status and control bits that are used to reflect the outcome of many instructions and modify the semantics of some instructions. The Instruction Pointer (EIP) is 32-bits wide. The EIP controls instruction fetching, and the processor automatically increments it after executing an instruction.

Control Registers: The four 32-bit control registers are used to control the global nature of the Am386SXL microprocessor. The CR0 register contains bits that set the different processor modes (Protected, Real, Paging, and Coprocessor Emulation). CR2 and CR3 registers are used in the paging operation.



15022B–004

Figure 1. Am386SXL Microprocessor Registers

System Address Registers: These four special registers reference the tables or segments supported by the 80286/Am386SXL/Am386DX CPU's protection model. These tables or segments are:

GDTR (Global Descriptor Table Register),
IDTR (Interrupt Descriptor Table Register),
LDTR (Local Descriptor Table Register),
TR (Task State Segment Register).

Debug Registers: The six programmer accessible debug registers provide on-chip support for debugging. The use of the debug registers is described in the section Debugging Support.

Test Registers: Two registers are used to control the testing of the RAM/CAM (Content Addressable Memories) in the Translation Look-Aside Buffer portion of the Am386SXL microprocessor. Their use is discussed in the section Testability.

EFLAGS Register

The flag register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS, shown in Figure 2, control certain operations and indicate the status of the Am386SXL microprocessor. The lower 16 bits (bits 15–0) of EFLAGS contain the 16-bit flag register named FLAGS. This is the default flag register used when executing 8086, 80286, or real mode code. The functions of the flag bits are given in Table 1.

Control Registers

The Am386SXL microprocessor has three control registers of 32 bits, CR3–CR0, to hold the machine state of a global nature. These registers are shown in Figures 1 and 2. The defined CR0 bits are described in Table 2.

Instruction Set

The instruction set is divided into nine categories of operations:

Data Transfer
Arithmetic
Shift/Rotate
String Manipulation
Bit Manipulation
Control Transfer
High-Level Language Support
Operating System Support
Processor Control

These instructions are listed in the Instruction Set Clock Count Summary (pages 1-406 through 1-420).

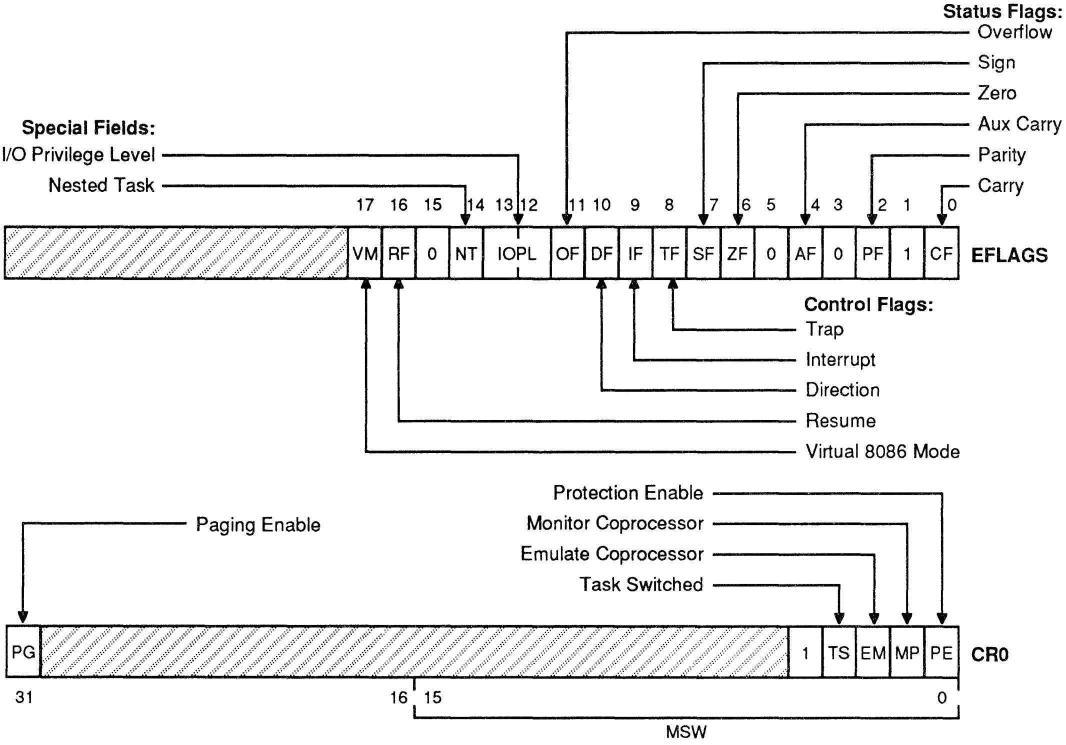


Figure 2. Status and Control Register Bit Functions

Table 1. Flag Definitions

Bit Position	Name	Function
0	CF	Carry Flag—Set on high-order bit carry or borrow; cleared otherwise.
2	PF	Parity Flag—Set if low-order 8 bits of result contain an even number of 1 bits; cleared otherwise.
4	AF	Auxiliary Carry Flag—Set on carry from or borrow to the low-order 4 bits of AL; cleared otherwise.
6	ZF	Zero Flag—Set if result is zero; cleared otherwise.
7	SF	Sign Flag—Set equal to high-order bit of result (0 if positive, 1 if negative).
8	TF	Single-Step Flag—Once set, a single-step interrupt occurs after the next instruction executes. TF is cleared by the single-step interrupt.
9	IF	Interrupt-Enable Flag—When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	Direction Flag—Causes string instructions to auto-increment (default) the appropriate index registers when cleared. Setting DF causes auto-decrement.
11	OF	Overflow Flag—Set if the operation resulted in a carry/borrow into the sign bit (high-order bit) of the result but did not result in a carry/borrow out of the high-order bit or vice-versa.
12, 13	IOPL	I/O Privilege Level—Indicates the maximum CPL permitted to execute I/O instructions without generating an Exception 13 fault or consulting the I/O permission bit map while executing in protected mode. For virtual 8086 mode it indicates the maximum CPL allowing alteration of the IF bit.
14	NT	Nested Task—Indicates that the execution of the current task is nested within another task.
16	RF	Resume Flag—Used in conjunction with debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. If set, any debug fault is ignored on the next instruction.
17	VM	Virtual 8086 Mode—If set while in protected mode, the Am386SXL microprocessor will switch to virtual 8086 operation, handling segment loads as 8086 does, but generating Exception 13 faults on privileged op-codes.

Table 2. CR0 Definitions

Bit Position	Name	Function
0	PE	Protection Mode Enable—Places the Am386SXL microprocessor into protected mode. If PE is reset, the processor operates again in Real Mode. PE may be set by loading MSW or CR0. PE can be reset only by loading CR0; it cannot be reset by the LMSW instruction.
1	MP	Monitor Coprocessor Extension—Allows WAIT instructions to cause a processor extension Not Present exception (number 7).
2	EM	Emulate Processor Extension—Causes a processor extension Not Present exception (number 7) on ESC instructions to allow emulating a processor extension.
3	TS	Task Switched—Indicates the next instruction using a processor extension will cause Exception 7, allowing software to test whether the current processor extension context belongs to the current task.
31	PG	Paging Enable Bit—Is set to enable the on-chip paging unit. It is reset to disable the on-chip paging unit.

All Am386SXL microprocessor instructions operate on either 0, 1, 2, or 3 operands; an operand resides in a register, in the instruction itself, or in memory. Most zero operand instructions (e.g., CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2 bytes long. Since the Am386SXL CPU has a 16-byte prefetch instruction queue, an average of 5 instructions will be prefetched. The use of two operands permits the following types of common instructions:

Register to Register
Memory to Register
Immediate to Register
Memory to Memory
Register to Memory
Immediate to Memory

The operands can be either 8, 16, or 32 bits long. As a general rule, when executing code written for the Am386SXL microprocessor (32-bit code), operands are 8 or 32 bits; when executing existing 8086 or 80286 code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to all instructions which override the default length of the operands (i.e., use 32-bit operands for 16-bit code, or 16-bit operands for 32-bit code).

Memory Organization

Memory on the Am386SXL microprocessor is divided into 8-bit quantities (Bytes), 16-bit quantities (Words), and 32-bit quantities (Dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address. The address of a Word or Dword is the byte address of the low-order byte.

In addition to these basic data types, the Am386SXL microprocessor supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4-Kb pages. Finally, both segmentation and paging can be combined, gaining the advantages of both systems. The Am386SXL CPU supports both pages and segmentation in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and, as such, is a tool for the application programmer, while pages are useful to the system programmer for managing the physical memory of a system.

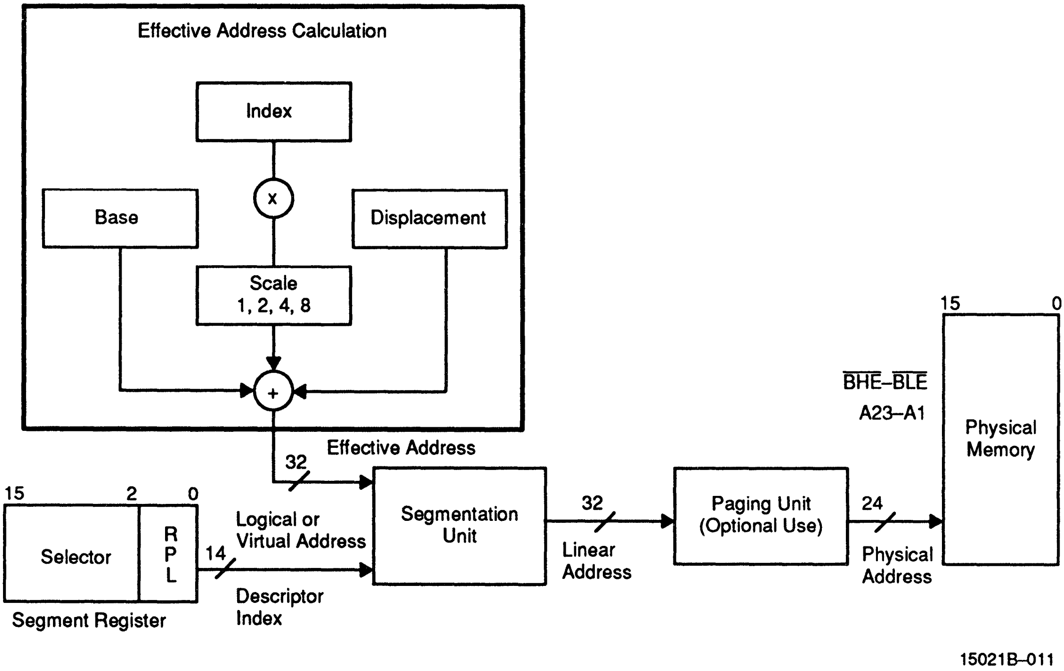


Figure 3. Address Translation

Address Spaces

The Am386SXL microprocessor has three types of address spaces: logical, linear, and physical. A logical address (also known as a virtual address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (Base, Index, Displacement), discussed in the section Addressing Modes, into an effective address. This effective address, along with the selector, is known as the logical address. Since each task on the Am386SXL CPU has a maximum of 16K (2^{14}-1) selectors, and offsets can be 4 Gb (with paging enabled), this gives a total of 2^{46} bits, or 64 tb, of logical address space per task. The programmer sees the logical address space.

The segmentation unit translates the logical address space into a 32-bit linear address space. If the paging unit is not enabled then the 32-bit linear address is truncated into a 24-bit physical address. The physical address is what appears on the address pins.

The primary differences between Real Mode and Protected Mode are how the segmentation unit performs the translation of the logical address into the linear address, size of the address space, and paging capability. In Real Mode, the segmentation unit shifts the selector left four bits and adds the result to the effective address to form the linear address. This linear address is limited to 1 Mb. In addition, Real Mode has no paging capability.

Protected Mode will see one of two different address spaces, depending on whether or not paging is enabled. Every selector has a logical base address associated with it that can be up to 32 bits in length. This 32-bit logical base address is added to the effective address to form a final 32-bit linear address. If paging is disabled, this final linear address reflects physical memory and is truncated so that only the lower 24 bits of this address are used to address the 16-Mb memory address space. If paging is enabled, this final linear address reflects a 32-bit address that is translated through the paging unit to form a 16-Mb physical address. The logical base address is stored in one of two operating system tables (i.e., the Local Descriptor Table or Global Descriptor Table).

Figure 3 shows the relationship between the various address spaces.

Segment Register Usage

The main data structure used to organize memory is the segment. On the Am386SXL CPU, 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 Gb (2^{32} bits).

In order to provide compact instruction encoding and increase processor performance, instructions do not need to explicitly specify which segment register is used. The segment register is automatically chosen according to the rules of Table 3 (Segment Register Selection Rules). In general, data references use the selector contained in the DS register; stack references use the SS register; and, instruction fetches use the CS register. The contents of the Instruction Pointer provide the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 3. The override prefixes also allow the use of the ES, FS, and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all six segments could have the base address set to zero and create a system with 4-Gb 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 the section Protected Mode Architecture.

Addressing Modes

The Am386SXL microprocessor provides a total of eight addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high-level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

Register and Immediate Modes

Two of the addressing modes provide for instructions that operate on register or immediate operands.

Register Operand Mode: The operand is located in one of the 8-, 16-, or 32-bit general registers.

Immediate Operand Mode: The operand is included in the instruction as part of the op-code.

32-Bit Memory Addressing Modes

The remaining six modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by summing any combination of the following three address elements (see Figure 3).

Displacement: an 8-,16-, or 32-bit immediate value, following the instruction.

Base: The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area.

Index: The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array or a string of characters. The index register's value can be multiplied by a scale factor, either 1, 2, 4, or 8. The scaled index is especially useful for accessing arrays or structures.

Table 3. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET Instructions	SS	None
Destination of STOS, MOVE, REP STOS, REP MOVS Instructions	ES	None
Other Data References, with Effective Address Using Base Register of:		
[EAX]	DS	CS, SS, ES, FS, GS
[EBX]	DS	CS, SS, ES, FS, GS
[ECX]	DS	CS, SS, ES, FS, GS
[EDX]	DS	CS, SS, ES, FS, GS
[ESI]	DS	CS, SS, ES, FS, GS
[EDI]	DS	CS, SS, ES, FS, GS
[EBP]	SS	CS, DS, ES, FS, GS
[ESP]	SS	CS, DS, ES, FS, GS

Combinations of these three components make up the six additional addressing modes. There is no performance penalty for using any of these addressing combinations, since the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of Base and Index components which requires one additional clock.

As shown in Figure 4, the Effective Address (EA) of an operand is calculated according to the following formula:

$EA = Base_{Register} + (Index_{Register} \times Scaling) + Displacement$

1. **Direct Mode**: The operand's offset is contained as part of the instruction as an 8-, 16-, or 32-bit displacement.
2. **Register Indirect Mode**: A Base register contains the address of the operand.
3. **Based Mode**: A Base register's contents are added to a Displacement to form the operand's offset.
4. **Scaled Index Mode**: An Index register's contents are multiplied by a Scaling factor, and the result is added to a Displacement to form the operand's offset.
5. **Based Scaled Index Mode**: The contents of an Index register are multiplied by a Scaling factor, and the result is added to the contents of a Base register to obtain the operand's offset.
6. **Based Scaled Index Mode with Displacement**: The contents of an Index register are multiplied by a Scaling factor, and the result is added to the contents of a Base register and a Displacement to form the operand's offset.

Differences Between 16- and 32-Bit Addresses

In order to provide software compatibility with the 8086 and the 80286, the Am386SXL microprocessor can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in a Segment Descriptor. If the D bit is 0, then all operand lengths and effective addresses are assumed to be 16-bits long. If the D bit is 1, then the default length for operands and addresses is 32 bits. In Real Mode the default size for operands and addresses is 16 bits.

Regardless of the default precision of the operands or addresses, the Am386SXL microprocessor is able to execute either 16- or 32-bit instructions. This is specified through the use of override prefixes. Two prefixes, the Operand Length Prefix and the Address Length Prefix, override the value of the D bit on an individual instruction basis. These prefixes are automatically added by assemblers.

The Operand Length and Address Length Prefixes can be applied separately or in combination to any instruction. The Address Length Prefix does not allow addresses over 64 Kb to be accessed in Real Mode. A memory address which exceeds 0FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional Am386SXL CPU addressing modes.

When executing 32-bit code, the Am386SXL CPU 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 registers conform to the 80286 model. Table 4 illustrates the differences.

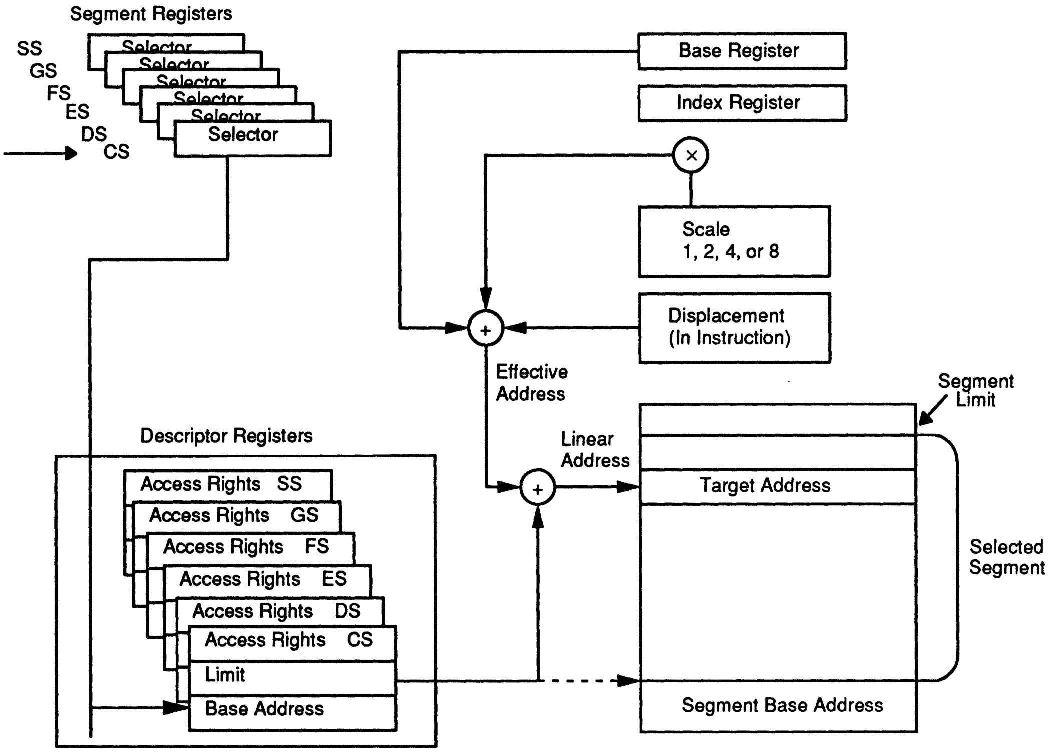


Figure 4. Addressing Mode Calculations

Data Types

The Am386SXL microprocessor supports all of the data types commonly used in high-level languages.

Bit: A single bit quantity.

Bit Field: A group of up to 32 contiguous bits, which spans a maximum of four bytes.

Bit String: A set of contiguous bits; on the Am386SXL microprocessor, bit strings can be up to 4 Gbits long.

Byte: A signed 8-bit quantity.

Unsigned Byte: An unsigned 8-bit quantity.

Integer (Word): A signed 16-bit quantity.

Long Integer (Dword): A signed 32-bit quantity. All operations assume a 2's complement representation.

Unsigned Integer (Word): An unsigned 16-bit quantity.

Unsigned Long Integer (Dword): An unsigned 32-bit quantity.

Signed Quad Word: A signed 64-bit quantity.

Unsigned Quad Word: An unsigned 64-bit quantity.

Pointer: A 16- or 32-bit offset-only quantity which indirectly references another memory location.

Long Pointer: A full pointer which consists of a 16-bit segment selector and either a 16- or 32-bit offset.

Char: A byte representation of an ASCII alphanumeric or control character.

String: A contiguous sequence of bytes, Words, or Dwords. A string may contain between 1 byte and 4 Gb.

BCD: A byte (unpacked) representation of decimal digits 0–9.

Packed BCD: A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble.

When the Am386SXL microprocessor is coupled with a 387SX math coprocessor, the following common floating point types are supported.

Floating Point: A signed 32-, 64-, or 80-bit real number representation. Floating point numbers are supported by 387SX-compatible math coprocessors.

Table 4. Base and Index Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
Base Register	BX, BP	Any 32-bit GP Register
Index Register	SI, DI	Any 32-bit GP Register Except ESP
Scale Factor	None	1, 2, 4, 8
Displacement	0, 8, 16 bits	0, 8, 32 bits

Figure 5 illustrates the data types supported by the Am386SXL microprocessor and a 387SX-compatible math coprocessor.

I/O Space

The Am386SXL CPU has two distinct physical address spaces: physical memory and I/O. Generally, peripherals are placed in I/O space, although the Am386SXL CPU also supports memory-mapped peripherals. The I/O space consists of 64 Kb which can be divided into 64K 8-bit ports or 32K 16-bit ports, or any combination of ports which add up to no more than 64 Kb. The 64K I/O address space refers to physical addresses rather than linear addresses since I/O instructions do not go through the segmentation or paging hardware. The M/$\overline{\text{IO}}$ pin acts as an additional address line, thus allowing the system designer to easily determine which address space the processor is accessing.

The I/O ports are accessed by the In and Out instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8-bit and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/$\overline{\text{IO}}$ pin to be driven Low. I/O port addresses 00F8H through 00FFH are reserved for future use.

Interrupts and Exceptions

Interrupts and exceptions alter the normal program flow in order to handle external events, report errors, or report exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT n instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately after the interrupted instruction.

Exceptions are classified as faults, traps, or aborts, depending on the way they are reported and whether or not restart of the instruction causing the exception is supported. Faults are exceptions that are detected and serviced *before* the execution of the faulting instruction. Traps are exceptions that are reported immediately *after* the execution of the instruction which caused the problem. Aborts are exceptions that do not permit the precise location of the instruction causing the exception to be determined.

Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point to the instruction causing the exception and will include any leading instruction prefixes. Table 5 summarizes the possible interrupts for the Am386SXL microprocessor and shows where the return address points to.

The Am386SXL CPU has the ability to handle up to 256 different interrupts/exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode, the vectors are 4-byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8-byte quantities which are put in an Interrupt Descriptor Table. Of the 256 possible interrupts, 32 are reserved for future use and the remaining 224 are free to be used by the system designer.

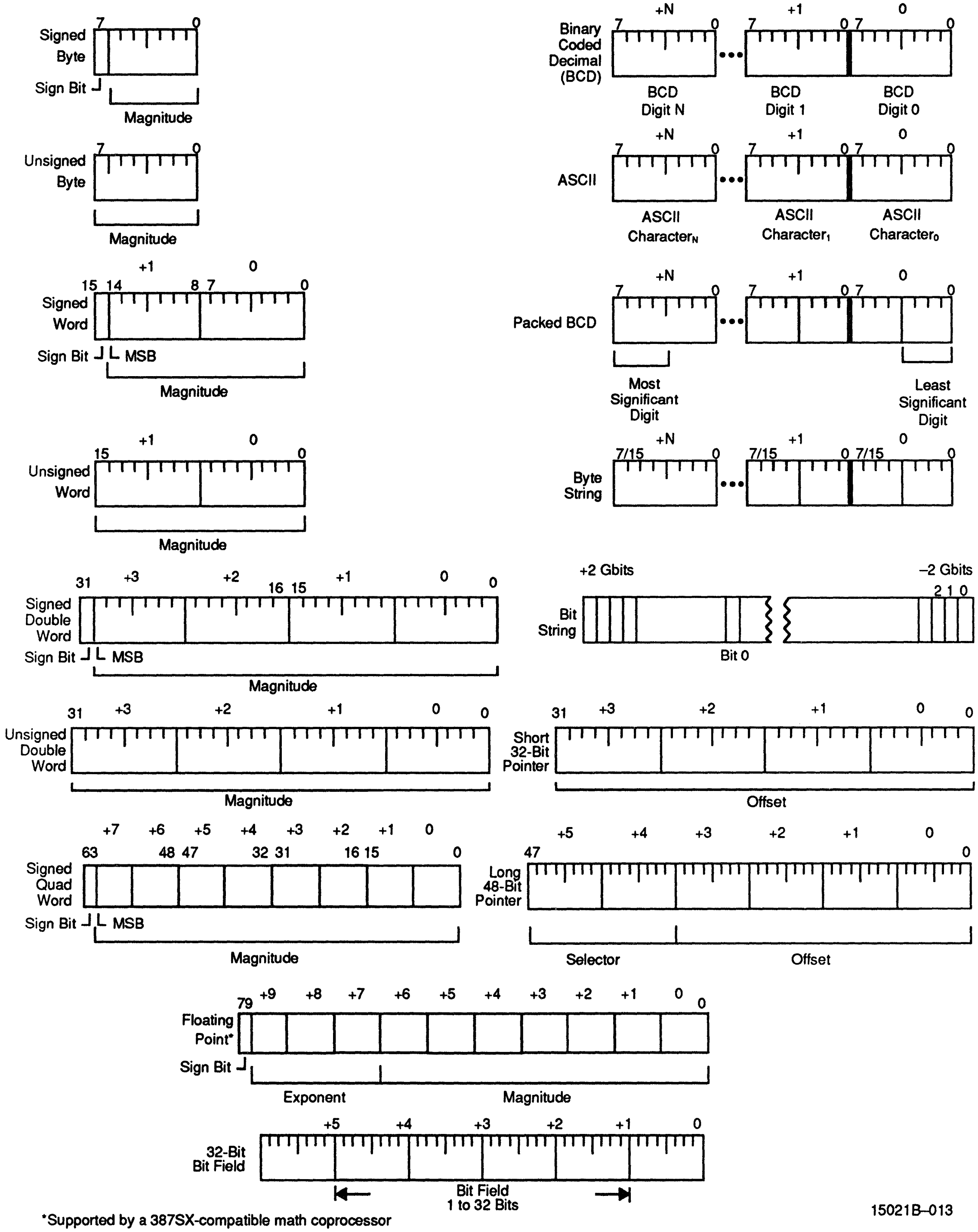
Interrupt Processing

When an interrupt occurs, the following actions happen. First, the current program address and Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the Am386SXL microprocessor which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an !RET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the Am386SXL microprocessor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

Maskable Interrupt

Maskable interrupts are the most common way to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled High and the Interrupt Flag bit (IF) is enabled. The processor only responds to interrupts between instructions (string instructions have an interrupt window between memory moves that allows interrupts during long string



*Supported by a 387SX-compatible math coprocessor

Figure 5. Am386SXL Microprocessor Supported Data Types

Table 5. Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	Yes	FAULT
Debug Exception	1	Any Instruction	Yes	TRAP*
NMI Interrupt	2	INT2 or NMI	No	NMI
One Byte Interrupt	3	INT	No	TRAP
Interrupt on Overflow	4	INTO	No	TRAP
Array Bounds Check	5	BOUND	Yes	FAULT
Invalid Op-code	6	Any Illegal Instruction	Yes	FAULT
Device Not Available	7	ESC, WAIT	Yes	FAULT
Double Fault	8	Any instruction that can generate an exception		ABORT
Coprocessor Segment Overrun	9	ESC	No	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	Yes	FAULT
Segment Not Present	11	Segment Register Instructions	Yes	FAULT
Stack Fault	12	Stack References	Yes	FAULT
General Protection Fault	13	Any Memory Reference	Yes	FAULT
Page Fault	14	Any Memory Access or Code Fetch	Yes	FAULT
Coprocessor Error	16	ESC, WAIT	Yes	FAULT
Reserved for Future Use	17–32			
Two Byte Interrupt	0–255	INT n	No	TRAP

Note: Some debug exceptions may report both traps on the previous instruction and faults on the next instruction.

moves). When an interrupt occurs the processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt (one of 224 user defined interrupts).

Interrupts through interrupt gates automatically reset IF bit, disabling INTR requests. Interrupts through Trap Gates leave the state of the IF bit unchanged. Interrupts through a Task Gate change the IF bit according to the image of the EFLAGs register in the task's Task State Segment (TSS). When an IRET instruction is executed, the original state of the IF bit is restored.

Non-Maskable Interrupt

Non-maskable interrupts provide a method of servicing very high priority interrupts. When the NMI input is pulled High it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for an NMI.

While executing the NMI servicing procedure, the Am386SXL microprocessor will not service any further NMI request or INT requests until an Interrupt Return (IRET) instruction is executed or the processor is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

Software Interrupts

A third type of interrupt/exception for the Am386SXL CPU is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the nth vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt is the single-step interrupt. It is discussed in section Single-Step Trap.

Interrupt and Exception Priorities

Interrupts are externally generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input) are recognized at instruction boundaries. When NMI and maskable INTR are both recognized at the same instruction boundary, the Am386SXL microprocessor invokes the NMI service routine first. If maskable interrupts are still enabled after the NMI service routine has been invoked, then the Am386SXL CPU will invoke the appropriate interrupt service routine.

As the Am386SXL microprocessor executes instructions, it follows a consistent cycle in checking for exceptions, as shown in Table 6. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution.

Instruction Restart

The Am386SXL microprocessor fully supports restarting all instructions after Faults. If an exception is detected in the instruction to be executed (exception categories 4 through 10 in Table 6), the Am386SXL microprocessor invokes the appropriate exception service routine. The Am386SXL microprocessor is in a state that permits restart of the instruction, for all cases by those given in Table 7. Note that all such cases will be avoided by a properly designed operating system.

Double Fault

A Double Fault (Exception 8) results when the processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12, or 13), but in the process of doing so detects an exception other than a Page Fault (Exception 14).

One other cause of generating a Double Fault is the Am386SXL CPU detecting any other exception when it is attempting to invoke the Page Fault (Exception 14) service routine (e.g., if a Page Fault is detected when the Am386SXL microprocessor attempts to invoke the Page Fault service routine). Of course, in any functional system, not only the Am386SXL CPU-based systems, the entire Page Fault service must remain present in memory.

Reset and Initialization

When the processor is initialized or Reset, the registers have the values shown in Table 8. The Am386SXL CPU will then start executing instructions near the top of physical memory, at location 0FFFFF0H. When the first intersegment Jump or Call is executed, address lines A23–A20 will drop Low for CS-relative memory cycles, and the Am386SXL CPU will only execute instructions in the lower 1 Mb of physical memory. This allows the system designer to use a shadow ROM at the top of physical memory to initialize the system and take care of Resets.

Reset forces the Am386SXL microprocessor to terminate all execution and local bus activity. No instruction execution or bus activity will occur as long as Reset is active. Between 350- and 450-CLK2 periods after Reset becomes inactive, the Am386SXL microprocessor will start executing instructions at the top of physical memory.

Table 6. Sequence of Exception Checking

Consider the case of the 386SXL microprocessor having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed.

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2. Check for external NMI and INTR.
3. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
4. Check for Segmentation Faults that prevented fetching the entire next instruction (Exceptions 11 and 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 op-code; Exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only; or Exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e., not at IOPL or at CPL = 0)).
7. If WAIT op-code, check if TS = 1 and MP = 1(Exception 7 if both are 1).
8. If ESCape op-code for math coprocessor, check if EM = 1 or TS = 1 (Exception 7 if either are 1).
9. If WAIT op-code or ESCape op-code for math coprocessor, check $\overline{\text{ERROR}}$ input signal (Exception 16 if $\overline{\text{ERROR}}$ input is asserted).
10. Check in the following order for each memory reference required by the instruction.
    a. Check for Segmentation Faults that prevent transferring the entire memory quantity (Exceptions 11, 12, and 13).
    b. Check for Page Faults that prevent transferring the entire memory quantity (Exception 14).

Note: Segmentation exceptions are generated before paging exceptions.

Table 7. Conditions Preventing Instruction Restart

1. An instruction causes a task switch to a task whose Task State Segment (TSS) is partially not present (an entire 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 Kb or less).
2. A coprocessor operand wraps around the top of a 64-Kb segment or a 4-Gb segment and spans three pages, and the page holding the middle portion of the operand is not present. This condition can be avoided by starting at a page boundary any segments containing coprocessor operands, if the segments are approximately 64K–200K bytes or larger (i.e., large enough for wraparound of the coprocessor operand to possibly occur).

Note: These conditions are avoided by using the operating system designs mentioned in this table.

Table 8. Register Values after Reset

Flag Word (EFLAGS)	uuuu0002H	Note 1
Machine Status Word (CR0)	uuuuuu10H	
Instruction Pointer (EIP)	0000FFF0H	
Code Segment (CS)	F000H	Note 2
Data Segment (DS)	0000H	Note 3
Stack Segment (SS)	0000H	
Extra Segment (ES)	0000H	Note 3
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
EAX Register	0000H	Note 4
EDX Register	Component and Stepping ID	Note 5
All Other Registers	Undefined	Note 6

Notes: 1. EFLAGS Register. The upper 14 bits of the EFLAGS register are undefined; all defined flag bits are zero.
2. The Code Segment register (CS) will have its Base Address set to 0FFFF0000H and Limit set to 0FFFFH.
3. The Data and Extra Segment registers (DS and ES) will have their Base Address set to 000000000H and Limit set to 0FFFFH.
4. If self-test is selected, the EAX register should contain a 0 value. If a value of 0 is not found, the self-test has detected a flaw in the part.
5. EDX register always holds a component and stepping identifier.
6. All undefined bits are reserved for future use and should not be used.

Testability

The Am386SXL microprocessor, like the Am386DX microprocessor, offers testability features that include a self-test and direct access to the page translation cache.

Self-Test

The Am386SXL microprocessor has the capability to perform a self-test. The self-test checks the function of all of the Control ROM and most of the non-random logic of the part. Approximately one-half of the Am386SXL CPU can be tested during self-test.

Self-Test is initiated on the Am386SXL microprocessor when the Reset pin transitions from High to Low, and the $\overline{\text{BUSY}}$ pin is Low. The self-test takes about 2^{20} clocks, or approximately 33 ms with a 16-MHz Am386SXL CPU. At the completion of self-test the processor performs reset and begins normal operation. The part has successfully passed self-test if the contents of the EAX are zero. If the results of the EAX are not zero, then the self-test has detected a flaw in the part.

TLB Testing

The Am386SXL microprocessor also provides a mechanism for testing the Translation Look-Aside Buffer (TLB), if desired. This particular mechanism may not be continued in the same way in future processors.

There are two TLB testing operations: 1) writing entries into the TLB; and, 2) performing TLB lookups. Two test registers, shown in Figure 6 are provided for the purpose of testing. TR6 is the test command register, and TR7 is the test data register.

Debugging Support

The Am386SXL microprocessor provides several features which simplify the debugging process. The three categories of on-chip debugging aids are:

1. The code execution breakpoint op-code (0CCH).
2. The single-step capability provided by the TF bit in the flag register.
3. The code and data breakpoint capability provided by the Debug Registers DR3–DR0, DR6, and DR7.

Breakpoint Instruction

A single-byte software interrupt (INT 3) breakpoint instruction is available for use by software debuggers. The breakpoint op-code is 0CCH, and generates an Exception 3 trap when executed.

Single-Step Trap

If the single-step flag (TF, bit 8) in the EFLAGS 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 1.

Debug Registers

The Debug Registers are an advanced debugging feature of the Am386SXL microprocessor. They allow data access breakpoints as well as code execution breakpoints. Since the breakpoints are indicated by on-chip registers, an instruction execution breakpoint can be placed in ROM code or in code shared by several tasks, neither of which can be supported by the INT 3 breakpoint op-code.

The Am386SXL microprocessor contains six Debug Registers, consisting of four breakpoint address

Table 9. Exceptions in Real Mode

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT vector is not within table limit	Before Instruction
CS, DS, ES, FS, GS Segment Overrun exception	13	Word memory reference with offset = 0FFFFH. An attempt to execute past the end of CS segment.	Before Instruction
SS Segment Overrun exception	12	Stack Reference beyond offset = 0FFFFH.	Before Instruction

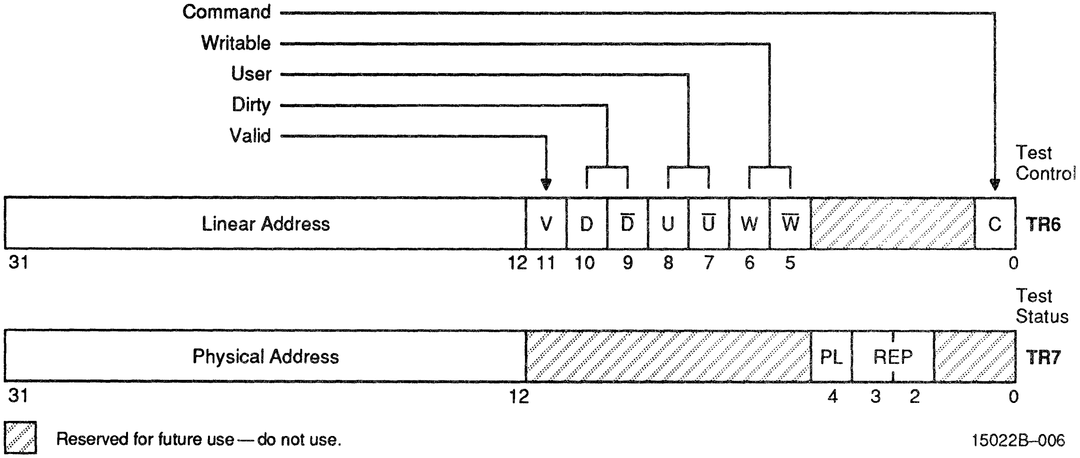


Figure 6. Test Registers

registers and two breakpoint control registers. Initially after reset, breakpoints are in the disabled state; therefore, no breakpoints will occur unless the Debug Registers are programmed. Breakpoints set up in the Debug Registers are auto-vectored to Exception 1. Figure 7 shows the breakpoint status and control registers.

REAL MODE ARCHITECTURE

When the processor is reset or powered up it is initialized in Real Mode. Real Mode has the same base architecture as the 8086, but allows access to the 32-bit register set of the Am386SXL microprocessor. The addressing mechanism, memory size, and interrupt handling are all identical to the Real Mode on the 80286.

The default operand size in Real Mode is 16 bits, as in the 8086. In order to use the 32-bit registers and addressing modes, override prefixes must be used. In addition, the segment size on the Am386SXL microprocessor in Real Mode is 64 Kb, so 32-bit addresses must have a value less than 0000FFFFH. The primary purpose of Real Mode is to set up the processor for Protected Mode operation.

Memory Addressing

In Real Mode the linear addresses are the same as physical addresses (paging is not allowed). Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register which is shifted left by four bits to an effective address. This addition results in a 20-bit physical address or a 1-Mb address space. Since segment registers are shifted left by 4 bits, Real Mode segments always start on 16-byte boundaries.

All segments in Real Mode are exactly 64-Kb long, and may be read, written, or executed. The Am386SXL microprocessor will generate an Exception 13 if a data operand or instruction fetch occurs past the end of a segment.

Reserved Locations

There are two fixed areas in memory that are reserved in Real Address Mode: the system initialization area and the interrupt table area. Locations 00000H through 003FFH are reserved for interrupt vectors. Each one of the 256 possible interrupts has a 4-byte jump vector reserved for it. Locations 0FFFFF0H through 0FFFFFFH are reserved for system initialization.

Interrupts

Many of the exceptions discussed in section Interrupts and Exceptions are not applicable to Real Mode operation; in particular, Exceptions 10, 11, and 14 do not occur in Real Mode. Other exceptions have slightly different meanings in Real Mode; Table 9 identifies these exceptions.

Shutdown and Halt

The HLT instruction stops program execution and prevents the processor from using the local bus until restarted. Either NMI, $\overline{\text{FLT}}$, INTR with interrupts enabled (IF=1), or Reset will force the Am386SXL microprocessor out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

Shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions:

1. An interrupt or an exception occurs (Exceptions 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table.
2. A Call, INT, or Push instruction attempts to wrap around the stack segment when SP is not even.

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 000FH) and the stack has enough room to contain the vector and flag information (i.e., SP is greater than 0005H). Otherwise, shutdown can only be exited by a processor reset.

LOCK Operation

The LOCK prefix on the Am386SXL microprocessor, even in Real Mode, is more restrictive than on the 80286. This is due to the addition of paging on the Am386SXL CPU in Protected Mode and Virtual 8086 Mode. The LOCK prefix is not supported during repeat string instructions.

The only instruction forms where the LOCK prefix is legal on the Am386SXL microprocessor are shown in Table 10.

Table 10. Legal Instructions for the LOCK Prefix

Op-Code	Operands (Dest, Source)
BIT Test and SET/RESET/COMPLEMENT	Mem, Reg/Immed
XCHG	Reg, Mem
XCHG	Mem, Reg
ADD, OR, ADC, SBB AND, SUB, XOR	Mem, Reg/Immed
NOT, NEG, INC, DEC	Mem

An Exception 6 will be generated if a LOCK prefix is placed before any instruction form or op-code not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above.

The LOCK prefix is not IOPL-sensitive on the Am386SXL microprocessor. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed in Table 10.

PROTECTED MODE ARCHITECTURE

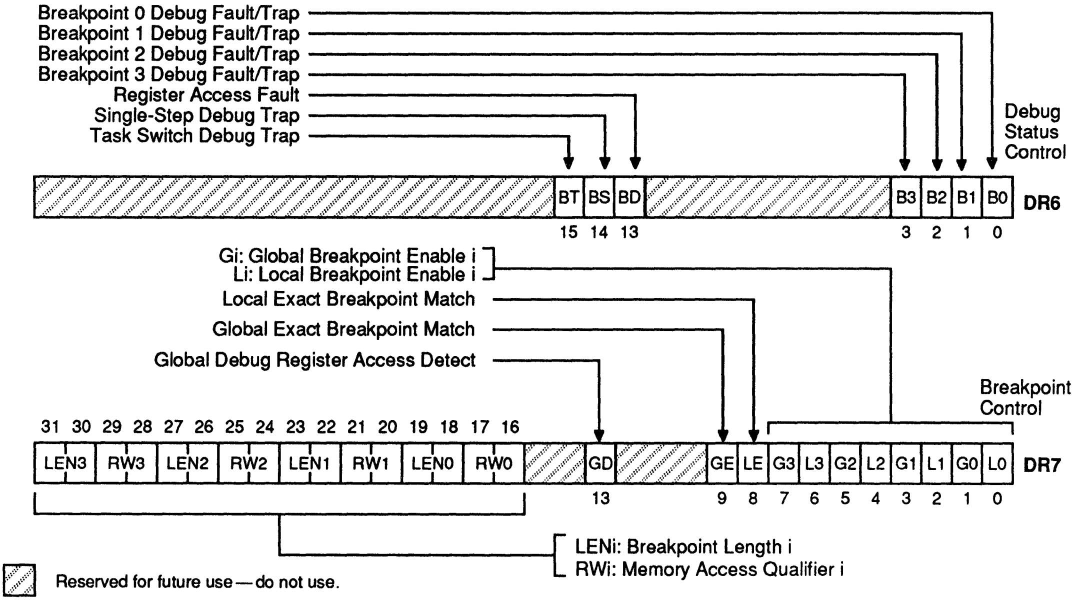
The complete capabilities of the Am386SXL microprocessor are unlocked when the processor operates in Protected Virtual Address Mode (Protected Mode). Protected Mode vastly increases the linear address space to 4 Gb (2^{32} bytes), and allows the running of virtual memory programs of almost unlimited size (64 tb (2^{46} bytes)). In addition, Protected Mode allows the Am386SXL CPU to run all of the existing Am386DX CPU (using only 16 Mb of physical memory), 80286, and 8086 CPU's software, while providing a sophisticated memory management and a hardware-assisted protection mechanism. Protected Mode allows the use of additional instructions specially optimized for supporting multitasking operating systems. The base architecture of the Am386SXL microprocessor remains the same; the registers, instructions, and addressing modes described in the previous sections are retained. The main difference between Protected Mode and Real Mode from a programmer's viewpoint is the increased address space and a different addressing mechanism.

Addressing Mechanism

Like Real Mode, Protected Mode uses two components to form the logical address: a 16-bit selector is used to determine the linear base address of a segment, the base address is added to a 32-bit effective address to form a 32-bit linear address. The linear address is then either used as a 24-bit physical address, or if paging is enabled, the paging mechanism maps the 32-bit linear address into a 24-bit physical address.

The difference between the two modes lies in calculating the base address. In Protected Mode, the selector is used to specify an index into an operating system defined table (see Figure 8). 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 Am386SXL microprocessor, as paging operates beneath segmentation. The page mechanism translates the protected linear address which comes from the segmentation unit into a physical address. Figure 9 shows the complete Am386SXL CPU addressing mechanism with paging enabled.


Breakpoint 0 Debug Fault/Trap
Breakpoint 1 Debug Fault/Trap
Breakpoint 2 Debug Fault/Trap
Breakpoint 3 Debug Fault/Trap
Register Access Fault
Single-Step Debug Trap
Task Switch Debug Trap
Debug Status Control
BT
BS
BD
B3
B2
B1
B0
DR6
15 14 13
3 2 1 0
Gi: Global Breakpoint Enable i
Li: Local Breakpoint Enable i
Local Exact Breakpoint Match
Global Exact Breakpoint Match
Global Debug Register Access Detect
Breakpoint Control
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
LEN3
RW3
LEN2
RW2
LEN1
RW1
LEN0
RW0
GD
GE
LE
G3
L3
G2
L2
G1
L1
G0
L0
DR7
13
9 8 7 6 5 4 3 2 1 0
LENi: Breakpoint Length i
RWi: Memory Access Qualifier i
Reserved for future use—do not use.

15022B-007

Figure 7. Debug Registers

Segmentation

Segmentation is one method of memory management. Segmentation provides the basis for protection. Segments are used to encapsulate regions of memory which have common attributes. For example, all of the code of a given program could be contained in a segment, or an operating system table may reside in a segment. All information about each segment is stored in an 8-byte data structure called a descriptor. All of the descriptors in a system are contained in descriptor tables which are recognized by hardware.

Terminology

The following terms are used throughout the discussion of descriptors, privilege levels, and protection:

PL: Privilege Level—One of the four hierarchical privilege levels. Level 0 is the most privileged level and level 3 is the least privileged.

RPL: Requestor Privilege Level—The privilege level of the original supplier of the selector. RPL is determined by the least two significant bits of a selector.

DPL: Descriptor Privilege Level—This is the least privileged level at which a task may access that descriptor (and the segment associated with that descriptor). Descriptor Privilege Level is determined by bits 6:5 in the Access Right Byte of a descriptor.

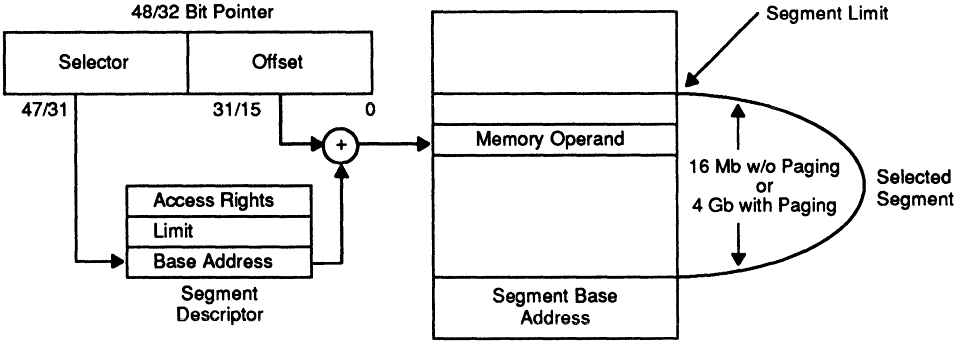
CPL: Current Privilege Level—The privilege level at which a task is currently executing, which equals the privilege level of the code segment being executed. CPL can also be determined by examining the lowest 2 bits of the CS register, except for conforming code segments.

EPL: Effective Privilege Level—The effective privilege level is the least privileged of the RPL and the DPL. EPL is the numerical maximum of RPL and DPL.

Task: One instance of the execution of a program. Tasks are also referred to as processes.

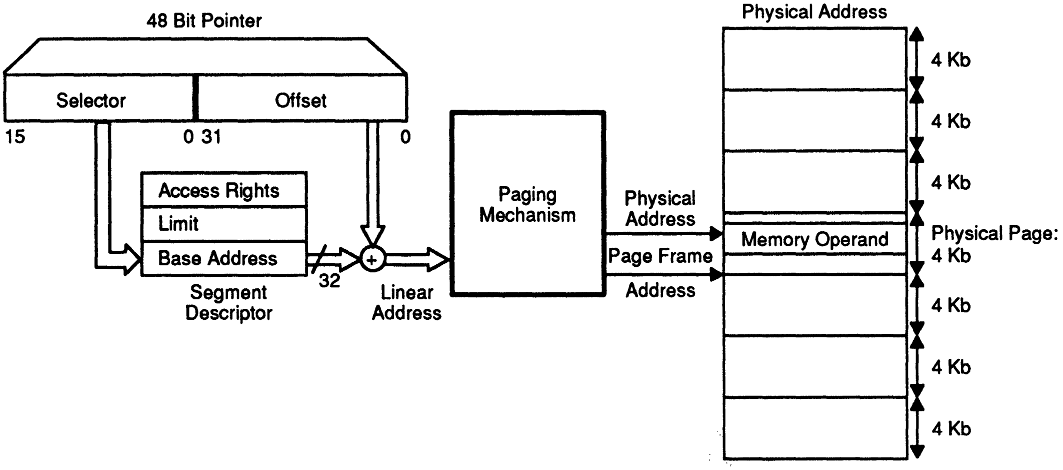
Descriptor Tables

The descriptor tables define all of the segments which are used in an Am386SXL microprocessor system. There are three types of tables which hold descriptors: the Global Descriptor Table, Local Descriptor Table, and Interrupt Descriptor Table. All of the tables are variable length memory arrays and can vary in size from 8 bytes to 64 Kb. 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.



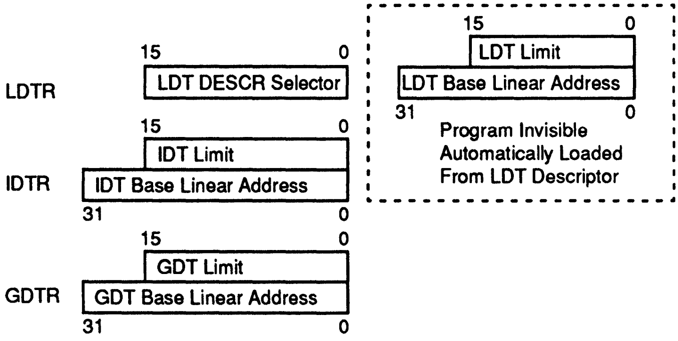
15021B–018

Figure 8. Protected Mode Addressing



15022B–005

Figure 9. Paging and Segmentation



15021B–020

Figure 10. Descriptor Table Registers

Each of the tables has a register associated with it: GDTR, LDTR, and IDTR (see Figure 1). The LGDT, LLDT, and LIDT instructions load the base and limit of the Global, Local, and Interrupt Descriptor Tables into the appropriate register. The SGDT, SLDT, and SIDT store the base and limit values. These are privileged instructions.

Global Descriptor Table

The Global Descriptor Table (GDT) contains descriptors which are available to all of the tasks in a system. The GDT can contain any type of segment descriptor except for interrupt and trap descriptors. Every Am386SXL CPU system contains a GDT.

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

Local Descriptor Table

LDTs contain descriptors which are associated with a given task. Generally, operating systems are designed so that each task has a separate LDT. The LDT may contain only code, data, stack, task gate, and call gate descriptors. LDTs provide a mechanism for isolating a given task's code and data segments from the rest of the operating system, while the GDT contains descriptors for segments which are common to all tasks. A segment cannot be accessed by a task if its segment descriptor does not exist in either the current LDT or the GDT. This provides both isolation and protection for a task's segments while still allowing global data to be shared among tasks.

Unlike the 6-byte GDT or IDT registers which contain a base address and limit, the visible portion of the LDT register contains only a 16-bit selector. This selector refers to a Local Descriptor Table descriptor in the GDT (see Figure 1).

Interrupt Descriptor Table

The third table needed for Am386SXL microprocessor systems is the Interrupt Descriptor Table. The IDT contains the descriptors which point to the location of the up to 256 interrupt service routines. The IDT may contain only task gates, interrupt gates, and trap gates. The IDT should be at least 256 bytes in size in order to hold the descriptors for the 32 interrupts reserved for future use. Every interrupt used by a system must have an entry in the IDT. The IDT entries are referenced by INT instructions, external interrupt vectors, and exceptions.

Descriptors

The object to which the segment selector points to is called a descriptor. Descriptors are eight byte quantities which contain attributes about a given region of linear address space. These attributes include the 32-bit base linear address of the segment, the 20-bit length and granularity of the segment, the protection level, read, write, or execute privileges, the default size of the operands (16 bit or 32 bit), and the type of segment. All of the attribute information about a segment is contained in 12 bits in the segment descriptor. Figure 11 shows the general format of a descriptor. All segments on the Am386SXL microprocessor have three attribute fields in common: the P bit, the DPL bit, and the S bit. The P (Present) Bit is 1 if the segment is loaded in physical memory. If P = 0, then any attempt to access this segment causes a Not Present exception (number 11). The Descriptor Privilege Level (DPL) is a two bit field which specifies the protection level, 0–3, associated with a segment.

The Am386SXL microprocessor has two main categories of segments: system segments and non-system segments (for code and data). The segment bit (S) determines if a given segment is a system segment or a code or data segment. If the S bit is 1, then the segment is either a code or data segment; if it is 0, then the segment is a system segment.

Code and Data Descriptors (S=1)

Figure 12 shows the general format of a code and data descriptor, and Table 11 illustrates how the bits in the Access Right Byte are interpreted.

Code and data segments have several descriptor fields in common. The accessed bit (A) is set whenever the processor accesses a descriptor. The granularity bit (G) specifies if a segment length is byte-granular or page-granular.

<table>
<tr><td colspan="5">31</td><td colspan="6" align="right">0</td><td>Byte Address</td></tr>
<tr><td colspan="5">Segment Base 15–0</td><td colspan="6">Segment Limit 15–0</td><td>0</td></tr>
<tr><td>Base 31–24</td><td>G</td><td>D</td><td>0</td><td>AVL</td><td>Limit 19–16</td><td>P</td><td>DPL</td><td>S</td><td>Type</td><td>A</td><td>Base 23–16</td><td>+4</td></tr>
</table>

Base	Base Address of the segment	A	Accessed Bit
Limit	The length of the segment	G	Granularity Bit (1 = Segment length is page-granular, 0 = Segment length is byte-granular)
P	Present Bit (1 = Present, 0 = Not Present)		
DPL	Descriptor Privilege Level 0–3	D	Default Operation Size (recognized in code segment descriptors only; 1 = 32-bit segment, 0 = 16-bit segment)
S	Segment Descriptor (0 = System Descriptor, 1 = Code or Data Segment Descriptor)	0	Bit must be zero for compatibility with future processors
Type	Type of Segment	AVL	Available field for user or OS

15021B–022

Figure 11. Segment Descriptors

<table>
<tr><td colspan="6">31</td><td colspan="2">0</td><td>Byte Address</td></tr>
<tr><td colspan="6">Segment Base 15–0</td><td colspan="2">Segment Limit 15–0</td><td>0</td></tr>
<tr><td>Base 31–24</td><td>G</td><td>D</td><td>0</td><td>AVL</td><td>Limit
19–16</td><td>Access Rights Bytes</td><td>Base
23–16</td><td>+4</td></tr>
</table>

D/B 1 = Default Instruction Attributes are 32 bits
0 = Default Instruction Attributes are 16 bits

AVL Available field for user or OS

G Granularity Bit 1 = Segment legnth is page-granular
0 = Segment length is byte-granular

0 Bit must be zero for compatibility with future processors

15021B–023

Figure 12. Code and Data Descriptors

<table>
<tr><td colspan="6">31</td><td colspan="5">0</td><td>Byte Address</td></tr>
<tr><td colspan="6">Segment Base 15–0</td><td colspan="5">Segment Limit 15–0</td><td>0</td></tr>
<tr><td>Base 31–24</td><td>G</td><td>D</td><td>0</td><td>0</td><td>Limit
19–16</td><td>P</td><td>DPL</td><td>0</td><td>Type</td><td>Base
23–16</td><td>+4</td></tr>
</table>

Type	Defines	Type	Defines
0	Invalid	8	Invalid
1	Available 80286 TSS	9	Available TSS
2	LDT	A	Undefined (Reserved)
3	Busy 80286 TSS	B	Busy TSS
4	80286 Call Gate	C	Am386SXL CPU Call Gate
5	Task Gate (for 80286 or Am386SXL CPU Task)	D	Undefined (Reserved)
6	80286 Interrupt Gate	E	Am386SXL CPU Interrupt Gate
7	80286 Trap Gate	F	Am386SXL CPU Trap Gate

15021B–024

Figure 13. System Descriptors

Table 11. Access Rights Byte Definition for Code and Data Descriptors

Bit Position	Name	Function		
7	Present (P)	P = 1	Segment is mapped into physical memory.	
		P = 0	No mapping to physical memory exists, Base and Limit are not used.	
6–5	Descriptor Privilege Levels (DPL)		Segment privilege attribute used in privilege tests.	
4	Segment Descriptor (S)	S = 1	Code or Data (includes stacks) Segment Descriptor.	
		S = 0	System Segment Descriptor or Gate Descriptor.	
3	Executable (E)	E = 0	Descriptor type is data segment:	If Data Segment (S = 1, E = 0)
2	Expansion Direction (ED)	ED = 0	Expand up segment, offsets must be ≤ limit.	
		ED = 1	Expand down segment, offsets must be > limit.	
1	Writeable (W)	W = 0	Data segment may not be written into.	
		W = 1	Data segment may be written into.	
3	Executable (E)	E = 1	Descriptor type is code segment:	If Code Segment (S = 1, E = 1)
2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
1	Readable (R)	R = 0	Code segment may not be read.	
		R = 1	Code segment may be read.	
0	Accessed (A)	A = 0	Segment has not been accessed.	
		A = 1	Segment selector has been loaded into segment register or used by selector test instructions.	

System Descriptor Formats (S=0)

System segments describe information about operating system tables, task, and gates. Figure 13 shows the general format of system segment descriptors, and the various types of system segments. Am386SXL CPU system descriptors (which are the same as Am386DX CPU system descriptors) contain a 32-bit base linear address and a 20-bit segment limit. 80286 system descriptors have a 24-bit base address and a 16-bit segment limit. 80286 system descriptors are identified by the upper 16 bits being all zero.

Differences Between Am386SXL Microprocessor and 80286 Descriptors

In order to provide operating system compatibility with the 80286, the Am386SXL CPU supports all of the 80286 segment descriptors. The 80286 system segment descriptors contain a 24-bit base address and 16-bit limit, while the Am386SXL CPU system segment descriptors have a 32-bit base address, a 20-bit limit field, and a granularity bit. The word count field specifies the number of 16-bit quantities to copy for 80286 call gates and 32-bit quantities for Am386SXL CPU call gates.

Selector Fields

A selector in Protected Mode has three fields: Local or Global Descriptor Table Indicator (TI), Descriptor Entry Index (Index), and Requestor (the selector's) Privilege Level (RPL), as shown in Figure 14. The TI bit selects either the Global Descriptor Table or the Local Descriptor Table. The Index selects one of 8K descriptors in the appropriate descriptor table. The RPL bits allow high speed testing of the selector's privilege attributes.

Segment Descriptor Cache

In addition to the selector value, every segment register has a segment descriptor cache register associated with it. Whenever a segment register's contents are changed, the 8-byte descriptor associated with the selector is automatically loaded (cached) on the chip. Once loaded, all references to that segment use the cached descriptor information instead of reaccessing the descriptor. The contents of the descriptor cache are not visible to the programmer. Since descriptor caches only change when a segment register is changed, programs which modify the descriptor tables must reload the appropriate segment registers after changing a descriptor's value.

Protection

The Am386SXL microprocessor has four levels of protection which are optimized to support a multitasking operating system and to isolate and protect user programs from each other and the operating system. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. The Am386SXL microprocessor also offers an additional type of protection on a page basis when paging is enabled.

The four-level hierarchical privilege system is an extension of the user/supervisor privilege mode commonly used by minicomputers. The user/supervisor mode is fully supported by the Am386SXL microprocessor paging mechanism. The Privilege Levels (PL) are numbered 0 through 3. Level 0 is the most privileged level.

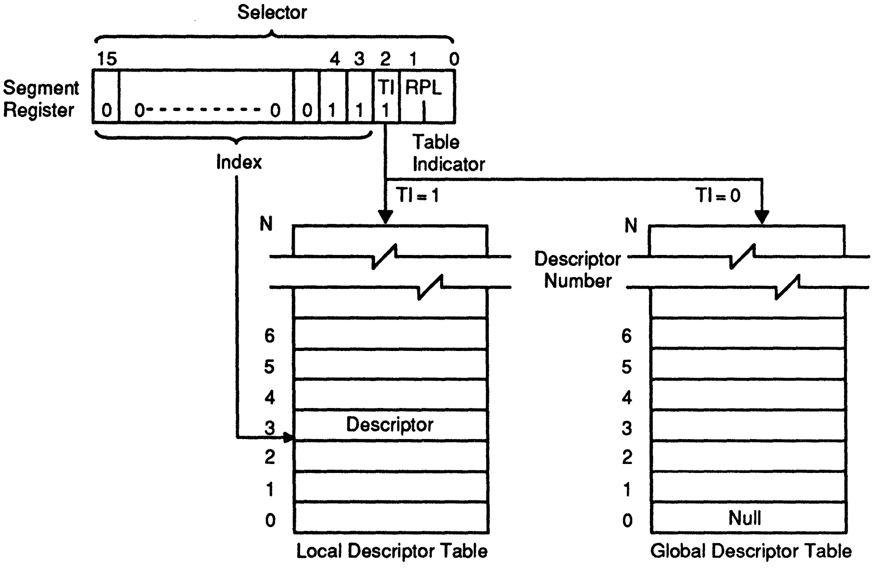


Figure 14. Example Descriptor Selection

Table 12. 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*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag register) = 0
**NT (Nested Task bit of flag register) = 1

Rules of Privilege

The Am386SXL microprocessor controls access to both data and procedures between levels of a task, according to the following rules:

—Data stored in a segment with privilege level p can be accessed only by code executing at a privilege level at least as privileged as p.

—A code segment/procedure with privilege level p can only be called by a task executing at the same or lesser privilege level than p.

Privilege Levels

At any point in time, a task on the Am386SXL microprocessor always executes at one of the four privilege levels. The Current Privilege Level (CPL) specifies what the task's privilege level is. A task's CPL may only be changed by control transfers through gate descriptors to a code segment with a different privilege level. Thus, an application program running at PL = 3 may call an operating system routine at PL = 1 (via a gate) which would cause the task's CPL to be set to 1 until the operating system routine was finished.

Selector Privilege (RPL)

The privilege level of a selector is specified by the RPL field. The selector's RPL is only used to establish a less trusted privilege level than the current privilege level of the task for the use of a segment. This level is called the task's Effective Privilege Level (EPL). The EPL is defined as being the least privileged (numerically larger) level of a task's CPL and a selector's RPL. The RPL is most commonly used to verify that pointers passed to an operating system procedure do not access data that is of higher privilege than the procedure that originated the pointer. Since the originator of a selector can specify any RPL value, the Adjust RPL (ARPL) instruction is provided to force the RPL bits to the originator's CPL.

I/O Privilege

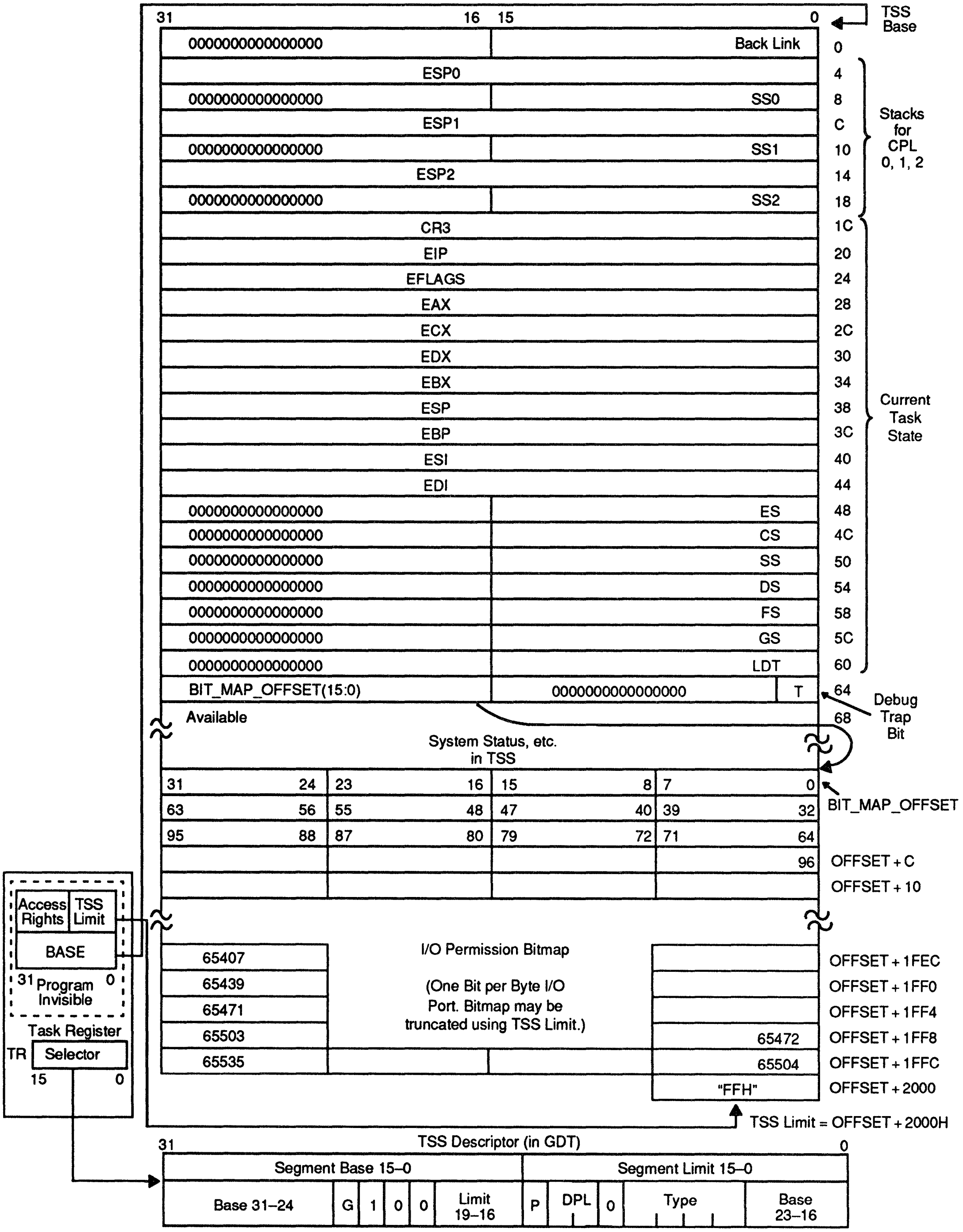
The I/O Privilege Level (IOPL) lets the operating system code executing at CPL = 0 define the least privileged level at which I/O instructions can be used. An Exception 13 (General Protection Violation) is generated if an I/O instruction is attempted when the CPL of the task is less privileged then the IOPL. The IOPL is stored in bits 13 and 14 of the EFLAGS register. The following instructions cause an Exception 13 if the CPL is greater than IOPL: IN, INS, OUT, OUTS, STI, CLI, and LOCK prefix.

Descriptor Access

There are basically two types of segment accesses: those involving code segments such as control transfers, and those involving data accesses. Determining the ability of a task to access a segment involves the type of segment to be accessed, the instruction used, the type of descriptor used, and CPL, RPL, and DPL as described above.

Any time an instruction loads a data segment register (DS, ES, FS, GS) the Am386SXL CPU makes protection validation checks. Selectors loaded in the DS, ES, FS, GS registers must refer only to data segment or readable code segments.

Finally, the privilege validation checks are performed. The CPL is compared to the EPL and if the EPL is more privileged than the CPL, an Exception 13 (General Protection Fault) is generated.



Type = 9: Available TSS.
Type = B: Busy TSS.

15022B–006

Figure 15. 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.																				

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

15022B–032b

Figure 16. Sample I/O Permission Bit Map

The rules regarding the stack segment are slightly different than those involving data segments. Instructions that load selectors into SS must refer to data segment descriptors for writeable data segments. The DPL and RPL must equal the CPL of all other descriptor types or a privilege level violation will cause an Exception 13. A stack not present fault causes an Exception 12.

Privilege Level Transfers

Inter-segment control transfers occur when a selector is loaded in the CS register. For a typical system most of these transfers are simply the result of a call or a jump to another routine. There are five types of control transfers which are summarized in Table 12. Many of these transfers result in a privilege level transfer. Changing privilege levels is done only by control transfers, using gates, task switches, and interrupt or trap gates.

Control transfers can only occur if the operation which loaded the selector references the correct desriptor type. Any violation of these descriptor usage rules will cause an Exception 13.

CALL Gates

Gates provide protected indirect CALLs. One of the major uses of gates is to provide a secure method of privilege transfers within a task. Since the operating system defines all of the gates in a system, it can ensure that all gates only allow entry into a few trusted procedures.

Task Switching

A very important attribute of any multitasking/multi-user operating system is its ability to rapidly switch between tasks or processes. The Am386SXL microprocessor directly supports this operation by providing a task switch instruction in hardware. The task switch operation saves the entire state of the machine (all of the registers, address space, and a link to the previous task), loads a new execution state, performs protection checks, and commences execution in the new task. Like transfer of control by gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS), or a task gate descriptor in the GDT or LDT. An INT n instruction, exception, trap, or external interrupt may also invoke the task switch operation if there is a task gate descriptor in the associated IDT descriptor slot.

The TSS descriptor points to a segment (see Figure 15) containing the entire execution state. A task gate descriptor contains a TSS selector. The Am386SXL microprocessor supports both 80286 and Am386SXL CPU TSSs. The limit of an Am386SXL CPU TSS must be greater than 64H (2BH for a 80286 TSS), and can be as large as 16 Mb. In the additional TSS space, the operating system is free to store additional information such as the reason the task is inactive, the time the task has spent running, or open files belonging to the task.

Each task must have a TSS associated with it. The current TSS is identified by a special register in the Am386SXL microprocessor called the Task State Segment Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TSS descriptor are loaded whenever TR is loaded with a new selector. Returning from a task is accomplished by the IRET instruction. When IRET is executed, control is returned to the task which was interrupted. The currently executing task's state is saved in the TSS and the old task state is restored from its TSS.

Several bits in the flag register and machine status word (CR0) give information about the state of a task which is useful to the operating system. The Nested Task bit (NT) controls the function of the IRET instruction. If NT = 0, the IRET instruction performs the regular return. If NT = 1, IRET performs a task switch operation base 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 Am386SXL microprocessor task state segment is marked busy by changing the descriptor type field from Type 9 to Type 0BH. An 80286 TSS is marked busy by changing the descriptor type field from Type 1 to Type 3. Use of a selector that references a busy task state segment causes an Exception 13.

The VM (Virtual Mode) bit is used to indicate if a task is a Virtual 8086 task. If VM = 1 then the tasks will use the Real Mode addressing mechanism. The Virtual 8086 environment is only entered and exited by a task switch.

The coprocessor's state is not automatically saved when a task switch occurs. The Task Switched Bit (TS) in the CR0 register helps deal with the coprocessor's state in a multitasking environment. Whenever the Am386SXL microprocessor switches tasks, it sets the TS bit. The Am386SXL CPU detects the first use of a processor extension instruction after a task switch and causes the processor extension Not Available Exception 7. The exception handler for Exception 7 may then decide whether to save the state of the coprocessor.

The T bit in the Am386SXL microprocessor TSS indicates that the processor should generate a debug exception when switching to a task. If T = 1, then upon entry to a new task a debug Exception 1 will be generated.

Initialization and Transition To Protected Mode

Since the Am386SXL microprocessor begins executing in Real Mode immediately after RESET, it is necessary to initialize the system tables and registers with the appropriate values. The GDT and IDT registers must refer to a valid GDT and IDT. The IDT should be at least 256 bytes long, and the GDT must contain descriptors for the initial code and data segments.

Protected Mode is enabled by loading CR0 with PE bit set. This can be accomplished by using the MOV CR0, R/M instruction. After enabling Protected Mode, the next instruction should execute an intersegment JMP to load the CS register and flush the instruction decode queue. The final step is to load all of the data segment registers with the initial selector values.

An alternate approach to entering Protected Mode is to use the built in task-switch to load all of the registers. In this case the GDT would contain two TSS descriptors in addition to the code and data descriptors needed for the first task. The first JMP instruction in Protected Mode would jump to the TSS causing a task switch and loading all of the registers with the values stored in the TSS. The Task State Segment Register should be initialized to point to a valid TSS descriptor.

Paging

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.

Page Organization

The Am386SXL microprocessor uses two levels of tables to translate the linear address (from the segmentation unit) into a physical address. There are three components to the paging mechanism of the Am386SXL CPU: the Page Directory, the Page Tables, and the page itself (Page Frame). All memory-resident elements of the Am386SXL microprocessor paging mechanism are the same size, namely 4 Kb. A uniform size for all of the elements simplifies memory allocation and reallocation schemes, since there is no problem with memory fragmentation. Figure 17 shows how the paging mechanism works.

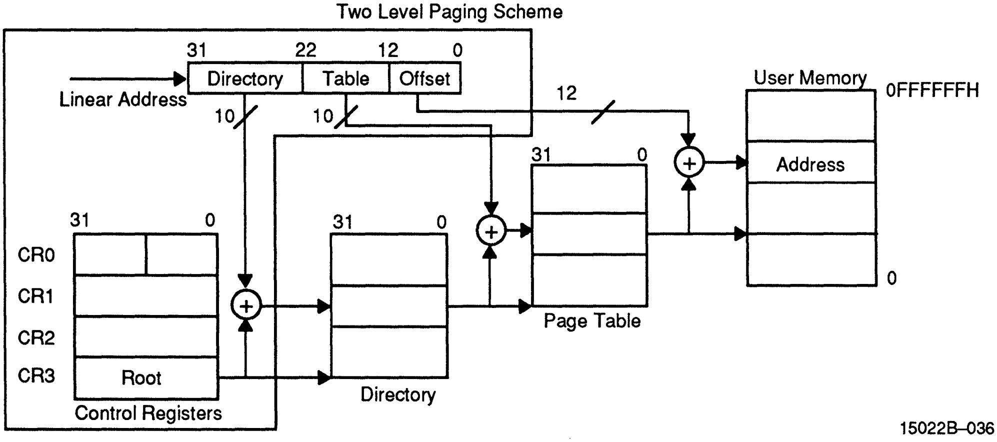


Figure 17. Paging Mechanism

31 12	11 10 9	8	7	6	5	4	3	2	1	0
Page Table Address 31–12	System Software Defineable	0	0	D	A	0	0	U – S	R – W	P

15022B–037

Figure 18. Page Directory Entry (Points to Page Table)

31 12	11 10 9	8	7	6	5	4	3	2	1	0
Page Frame Address 31–12	System Software Defineable	0	0	D	A	0	0	U – S	R – W	P

15022B–038

Figure 19. Page Table Entry (Points to Page)

Page Fault Register

CR2 is the Page Fault Linear Address register. It holds the 32-bit linear address which caused the last Page Fault detected.

Page Descriptor Base Register

CR3 is the Page Directory Physical Base Address Register. It contains the physical starting address of the Page Directory (this value is truncated to a 24-bit value associated with the Am386SXL CPU's 16-Mb physical memory limitation). The lower 12 bits of CR3 are always zero to ensure that the Page Directory is always page aligned. Loading it with a MOV CR3, reg instruction causes the Page Table entry cache to be flushed, as will a task switch through a TSS which changes the value of CR0.

Page Directory

The Page Directory is 4-Kb long and allows up to 1024 Page Directory entries. Each Page Directory entry contains information about the Page Table and the address of the next level of tables, the Page Tables. The contents of a Page Directory entry are shown in Figure 18. The upper 10 bits of the linear address (A31-A22) are used as an index to select the correct Page Directory entry.

The Page Table address contains the upper 20 bits of a 32-bit physical address that is used as the base address for the next set of tables, the Page Tables. The lower 12 bits of the Page Table addresses appear on 4-Kb boundaries. For an Am386DX CPU system, the upper 20 bits will select one of 2^{20} Page Tables, but for an Am386SXL microprocessor system, the upper 20 bits only select one of 2^{12} Page Tables. Again, this is because the Am386SXL CPU is limited to a 24-bit physical address, and the upper 8 bits (A31–A24) are truncated when the address is output on its 24 address pins.

Page Tables

Each Page Table is 4-Kb long and allows up to 1024 Page Table entries. Each Page Table entry contains information about the Page Frame and its address. The contents of a Page Table entry are shown in Figure 19. The middle 10 bits of the linear address (A21–A12) are used as an index to select the correct Page Table entry.

The Page Frame address contains the upper 20 bits of a 32-bit physical address which is used as the base address for the Page Frame. The lower 12 bits of the Page Frame address are zero so that the Page Frame addresses appear on 4-Kb boundaries. For an Am386DX CPU system, the upper 20 bits will select one of 2^{20} Page Frames, but for an Am386SXL microprocessor system, the upper 20 bits only select one of 2^{12} Page Frames. Again, this is because the Am386SXL CPU is limited to a 24-bit physical address space, and the upper 8 bits (A31–A24) are truncated when the address is output on its 24 address pins.

Page Directory/Table Entries

The lower 12 bits of the Page Table entries and Page Directory entries contain statistical information about pages and Page Tables, respectively. The P (Present) bit indicates if a Page Directory or Page Table entry can be used in address translation. If P = 1, the entry can be used for address translation. If P = 0, the entry cannot be used for translation. All of the other bits, are available for use by the software. For example, the remaining 31 bits could be used to indicate where on disk the page is stored.

The A (Accessed) bit is set by the Am386SXL CPU for both types of entries before a read or write access occurs to an address covered by the entry. The D (Dirty) bit is set to 1 before a write to an address covered by that Page Table entry occurs. The D bit is undefined for Page Directory entries. When the P, A, and D bits are updated by the Am386SXL CPU, the processor generates a Read-Modify-Write cycle which locks the bus and prevents conflicts with other processors or peripherals. Software which modifies these bits should use the LOCK prefix to ensure the integrity of the Page Tables in multi-master systems.

The 3 bits marked system software definable (in Figures 18 and 19) are software definable. System software writers are free to use these bits for whatever purpose they wish.

Page Level Protection (R/W, U/S Bits)

The Am386SXL microprocessor provides a set of protection attributes for paging systems. The paging mechanism distinguishes between two levels of protection: User, which corresponds to level 3 of the segmentation based protection; and Supervisor, which encompasses all of the other protection levels (0, 1, and 2). Programs executing at Level 0, 1, or 2 bypass the page protection, although segmentation-based protection is still enforced by the hardware.

The U/S and R/W bits are used to provide User/Supervisor and Read/Write protection for individual pages, or for all pages covered by a Page Table Directory entry. The U/S and R/W bits in the second level Page Table entry apply only to the page described by that entry. The U/S and R/W bits in the first level Page Directory Table apply to all pages described by the Page Table pointed to by that directory entry. The U/S and R/W bits for a given page are obtained by taking the most restrictive of the U/S and R/W bits from the Page Directory Table entries and using these bits to address the page.

Translation Look-Aside Buffer

The Am386SXL microprocessor paging hardware is designed to support demand paged virtual memory systems. However, performance would degrade substantially if the processor was required to access two levels of tables for every memory reference. To solve this problem, the Am386SXL CPU keeps a cache of the most recently accessed pages; this cache is called the Translation Look-Aside Buffer (TLB). The TLB is a four-way set associative 32-entry Page Table cache. It automatically keeps the most commonly used Page Table entries in the processor. The 32-entry TLB coupled with a 4K page size results in coverage of 128 Kb of memory addresses. For many common multitasking systems, the TLB will have a hit rate of greater than 98%. This means that the processor will only have to access the two-level page structure for less than 2% of all memory references.

Paging Operation

The paging hardware operates in the following fashion. The paging unit hardware receives a 32-bit linear address from the segmentation unit. The upper 20 linear address bits are compared with all 32 entries in the TLB to determine if there is a match. If there is a match (i.e., a TLB hit), then the 24-bit physical address is calculated and is placed on the address bus.

If the Page Table entry is not in the TLB, the Am386SXL microprocessor will read the appropriate Page Directory entry. If P = 1 on the Page Directory entry, indicating that the Page Table is in memory, then the Am386SXL CPU 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 Am386SXL microprocessor will update the Access and Dirty bits as needed and fetch the operand. The upper 20 bits of the linear address, read from the Page Table, will be stored in the TLB for future accesses. If P = 0 for either the Page Directory entry or the Page Table entry, then the processor will generate a Page Fault (Exception 14).

The processor will also generate a Page Fault (Exception 14) if the memory reference violated the page protection attributes. CR2 will hold the linear address which caused the Page Fault. Since Exception 14 is classified as a fault, CS:EIP will point to the instruction causing the Page Fault. The 16-bit error code, pushed as part of the Page Fault handler, will contain status bits which indicate the cause of the Page Fault.

The 16-bit error code is used by the operating system to determine how to handle the Page Fault. Figure 20 shows the format of the Page Fault error code and the interpretation of the bits. Even though the bits in the error code (U/S, W/R, and P) have similar names as the bits in the Page Directory/Table Entries, the interpretation of the error code bits is different. Figure 21 indicates what type of access caused the Page Fault.

15												3	2	1	0
U	U	U	U	U	U	U	U	U	U	U	U	U	U/$\overline{S}$	W/$\overline{R}$	P

Figure 20. Page Fault Error Code Format

U/S: The U/S bit indicates whether the access causing the fault occurred when the processor was executing in User Mode (U/S = 1) or in Supervisor mode (U/S = 0).

W/R: The W/R bit indicates whether the access causing the fault was a Read (W/R = 0) or a Write (W/R = 1).

P: The P bit indicates whether a Page Fault was caused by a not-present page (P = 0), or by a page level protection violation (P=1).

U = Undefined

U/S	W/R	Access Type
0	0	Supervisor* Read
0	1	Supervisor Write
1	0	User Read
1	1	User Write

*Descriptor table access will fault with U/S = 0, even if the program is executing at level 3.

Figure 21. Type of Access Causing Page Fault

Operating System Responsibilities

When the operating system enters or exits paging mode (by setting or resetting bit 31 in the CR0 register), a short JMP must be executed to flush the Am386SXL microprocessor's prefetch queue. This ensures that all instructions executed after the address mode change will generate correct addresses.

The Am386SXL microprocessor takes care of the page address translation process, relieving the burden from an operating system in a demand-paged system. The operating system is responsible for setting up the initial Page Tables and handling any Page Faults. The operating system is also required to invalidate (i.e., flush) the TLB when any changes are made to any of the Page Table entries. The operating system must reload CR3 to cause the TLB to be flushed.

Setting up the tables is simply a matter of loading CR3 with the address of the Page Directory, and allocating space for the Page Directory and the Page Tables. The primary responsibility of the operating system is to implement a swapping policy and handle all of the Page Faults.

A final concern of the operating system is to ensure that the TLB cache matches the information in the paging tables. In particular, any time the operating systems sets the P (Present) bit of Page Table entry to zero, the TLB must be flushed by reloading CR3. Operating systems may want to take advantage of the fact that CR3 is stored as part of a TSS, to give every task or group of tasks its own set of Page Tables.

Virtual 8086 Environment

The Am386SXL microprocessor allows the execution of 8086 application programs in both Real Mode and in Virtual 8086 Mode. The Virtual 8086 Mode allows the execution of 8086 applications, while still allowing the system designer to take full advantage of the Am386SXL CPU's protection mechanism.

Virtual 8086 Addressing Mechanism

One of the major differences between Am386SXL CPU Real and Protected modes is how the segment selectors are interpreted. When the processor is executing in Virtual 8086 Mode, the segment registers are used in a fashion identical to Real Mode. The contents of the segment register are shifted left four bits and added to the offset to form the segment base linear address.

The Am386SXL microprocessor allows the operating system to specify which programs use the 8086 address mechanism and which programs use Protected Mode addressing on a per task basis. Through the use of paging, the 1-Mb address space of the Virtual Mode task can be mapped to anywhere in the 4-Gb linear address space of the Am386SXL CPU. Like Real Mode, Virtual Mode addresses that exceed 1 Mb will cause an Exception 13. However, these restrictions should not prove to be important, because most tasks running in Virtual 8086 Mode will simply be existing 8086 application programs.

Paging in Virtual Mode

The paging hardware allows the concurrent running of multiple Virtual Mode tasks, and provides protection and operating system isolation. Although it is not strictly necessary to have the paging hardware enabled to run Virtual Mode tasks, it is needed in order to run multiple Virtual Mode tasks or to relocate the address space of a Virtual Mode task to physical address space greater than 1 Mb.

The paging hardware allows the 20-bit linear address produced by a Virtual Mode program to be divided into as many as 256 pages. Each one of the pages can be located anywhere within the maximum 16-Mb physical address space of the Am386SXL microprocessor. In addition, since CR3 (the Page Directory Base Register) is loaded by a task switch, each Virtual Mode task can use a different mapping scheme to map pages to different physical locations. Finally, the paging hardware allows the sharing of the 8086 operating system code between multiple 8086 applications.

Protection and I/O Permission Bit Map

All Virtual Mode programs execute at privilege level 3. As such, Virtual Mode programs are subject to all of the protection checks defined in Protected Mode. This is different from Real Mode, which implicitly is executing at privilege level 0. Thus, an attempt to execute a privileged instruction in Virtual Mode will cause an Exception 13 fault.

The following are privileged instructions, which may be executed only at Privilege level 0. Attempting to execute these instructions in Virtual 8086 Mode (or anytime CPL ≥ 0) causes an Exception 13 fault:

```
LIDT;    MOV DRn,REG;    MOV reg,DRn;
LGDT;    MOV TRn,reg;    MOV reg,TRn;
LMSW;    MOV CRn,reg;    MOV reg,CRn;
CLTS;
HLT;
```

Several instructions, particularly those applying to the multitasking and the protection model, are available only in Protected Mode. Therefore, attempting to execute the following instructions in Real Mode or in Virtual 8086 Mode generates an Exception 6 fault:

```
LTR;     STR;
LLDT;    SLDT;
LAR;     VERR;
LSL;     VERW;
ARPL;
```

The instructions which are IOPL sensitive in Protected Mode are:

```
IN;      STI;
OUT;     CLI
INS;
OUTS;
REP INS;
REP OUTS;
```

In Virtual 8086 Mode the following instructions are IOPL sensitive:

```
INT n;                  STI;
PUSHF;                  CLI;
POPF;                   IRET;
```

The PUSHF, POPF, and IRET instructions are IOPL sensitive in Virtual 8086 Mode only. This provision allows the IF flag to be virtualized to the virtual 8086 Mode program. The INT n software interrupt instruction is also IOPL sensitive in Virtual 8086 Mode. Note that the INT 3, INTO, and BOUND instructions are not IOPL sensitive in Virtual 8086 Mode.

The I/O instructions that directly refer to addresses in the processor's I/O space are IN, INS, OUT, and OUTS. The Am386SXL microprocessor has the ability to selectively trap references to specific I/O addresses. The structure that enables selective trapping is the I/O Permission Bit Map in the TSS segment (see Figures 15 and 16). The I/O permission map is a bit vector. The size of the map and its location in the TSS segment are variable. The processor locates the I/O permission map by means of the I/O map base field in the fixed portion of the TSS. The I/O map base field is 16-bits wide and contains the offset of the beginning of the I/O permission map.

In protected mode, when an I/O instruction (IN, INS, OUT, or OUTS) is encountered, the processor first checks whether CPL ≤ IOPL. If this condition is true, the I/O operation may proceed. If not true, the processor checks the I/O permission map (in Virtual 8086 Mode, the processor consults the map without regard for the IOPL).

Each bit in the map corresponds to an I/O port byte address; for example, the bit for port 41 is found at I/O map base +5, bit offset 1. The processor tests all the bits that correspond to the I/O addresses spanned by an I/O operation; for example, a Dword operation tests four bits corresponding to four adjacent byte addresses. If any tested bit is set, the processor signals a general protection exception. If all the tested bits are zero, the I/O operations may proceed.

It is not necessary for the I/O permission map to represent all the I/O addresses. I/O addresses not spanned by the map are treated as if they had one-bits in the map. The I/O map base should be at least one byte less than the TSS limit; the last byte beyond the I/O mapping information must contain all 1s.

Because the I/O permission map is in the TSS segment, different tasks can have different maps. Thus, the operating system can allocate ports to a task by changing the I/O permission map in the task's TSS.

Important Implementation Note: Beyond the last byte of I/O mapping information in the I/O permission bit map **must** be a byte containing all 1s. The byte of all 1s must be within the limit of the Am386SXL CPU TSS segment (see Figure 15).

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 Am386SXL microprocessor operating system. The Am386SXL CPU 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 Am386SXL microprocessor operating system in turn handles the exception or interrupt and then returns control to the 8086 program. The Am386SXL microprocessor operating system may choose to let the 8086 operating system handle the interrupt, or it may emulate the function of the interrupt handler. For example, many 8086 operating system calls are accessed by PUSHing parameters on the stack, and then executing an INT n instruction. If the IOPL is set to 0, then all INT n instructions will be intercepted by the Am386SXL CPU operating system.

An Am386SXL microprocessor operating system can provide a Virtual 8086 environment which is totally transparent to the application software by intercepting and then emulating 8086 operating system's calls, and intercepting IN and OUT instructions.

Entering and Leaving Virtual 8086 Mode

Virtual 8086 Mode is entered by executing a 32-bit IRET instruction at CPL = 0, where the stack has a 1 in the VM bit of its EFLAGS image, or a Task Switch (at any CPL) to an Am386SXL microprocessor task whose Am386SXL CPU TSS has an EFLAGS image containing a 1 in the VM bit position, while the processor is executing in the Protected Mode. POPF does not affect the VM bit, but a PUSHF always pushes a 0 in the VM bit.

The transition out of Virtual 8086 Mode to Protected Mode occurs only on receipt of an interrupt or exception. In Virtual 8086 Mode, all interrupts and exceptions vector through the Protected Mode IDT, and enter an interrupt handler in Protected Mode. As part of the interrupt processing the VM bit is cleared.

Because the matching IRET must occur from Level 0, Interrupt or Trap Gates used to field an interrupt or exception out of Virtual 8086 Mode must perform an inter-level interrupt only to Level 0. Interrupt or Trap Gates through conforming segments, or through segments with DPL > 0, will raise a GP fault with the CS selector as the error code.

Task Switches To/From Virtual 8086 Mode

Tasks which can execute in Virtual 8086 Mode must be described by a TSS with the Am386SXL CPU format (Type 9 or 11 descriptor). A task switch out of Virtual 8086 Mode will operate exactly the same as any other task switch out of a task with an Am386SXL CPU TSS. All of the programmer visible state, including the EFLAGS register with the VM bit set to 1, is stored in the TSS. The segment registers in the TSS will contain 8086 segment base values rather than selectors.

A task switch into a task described by an Am386SXL CPU 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 Am386SXL CPU TSS, the FLAGS image is loaded, so that the segment registers are loaded from the TSS image as 8086 segment base values. The task is now ready to resume in Virtual 8086 Mode.

Transitions Through Trap and Interrupt Gates, and IRET

A task switch is one way to enter or exit Virtual 8086 Mode. The other method is to exit through a Trap or Interrupt gate, as part of handling an interrupt, and to enter as part of executing an IRET instruction. The transition out must use an Am386SXL CPU Trap Gate (Type 14), or Am386SXL CPU Interrupt Gate (Type 15), which must point to a non-conforming Level 0 segment (DPL = 0) in order to permit the trap handler to IRET back to the Virtual 8086 program. The Gate must point to a non-conforming Level 0 segment to perform a level switch to Level 0 so that the matching IRET can change the VM bit. Am386SXL CPU gates must be used since 80286 gates save only the lower 16 bits of the EFLAGS register (the VM bit will not be saved). Also, the 16-bit IRET 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 Am386SXL CPU Trap or Interrupt gate, if an interrupt occurs while the task is executing in Virtual 8086 Mode, is given by the following sequence:

1. Save the FLAGS register in a temp to push later. Turn off the VM, TF, and IF bits.
2. Interrupt and Trap gates must perform a level switch from 3 (where the Virtual 8086 Mode program executes) to 0 (so IRET can return).
3. Push the 8086 segment register values onto the new stack, in this order: GS, FS, DS, and ES. These are pushed as 32-bit quantities. Then load these 4 registers with null selectors (0).
4. Push the old 8086 stack pointer onto the new stack by pushing the SS register (as 32 bits), then pushing the 32-bit ESP register saved above.
5. Push the 32-bit EFLAGS register saved in step 1.
6. Push the old 8086 instruction onto the new stack by pushing the CS register (as 32 bits), then pushing the 32-bit EIP register.
7. Load up the new CS:EIP value from the interrupt gate, and begin execution of the interrupt routine in protected mode.

The transition out of Virtual 8086 Mode performs a level change and stack switch, in addition to changing back to protected mode. Also, all of the 8086 segment register images are stored on the stack (behind the SS:ESP image), and then loaded with null (0) selectors before entering the interrupt handler. This will permit the handler to safely save and restore the DS, ES, FS, and GS registers as 80286 selectors. This is needed so that interrupt handlers, which do not care about the mode of the interrupted program, can use the same prologue and epilogue code for state saving, regardless of whether or not a native mode or Virtual 8086 Mode program was interrupted. Restoring null selectors to these registers before executing the IRET will cause a trap in the interrupt handler. Interrupt routines which expect or return values in the segment registers will have to obtain/return values from the 8086 register images pushed onto the new stack. They will need to know the mode of the interrupted program in order to know where to find/return segment registers, and also to know how to interpret segment register values.

The IRET instruction will perform the inverse of the above sequence. Only the extended IRET instruction (operand size = 32) can be used, and must be executed at Level 0, to change the VM bit to 1.

1. If the NT bit in the FLAGS register is On, an 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. Increment the ESP register by 4 to bypass the FLAGS image which was popped in step 1.
5. If VM = 1, load segment registers ES, DS, FS, and GS from memory locations SS:[ESP + 8], SS:[ESP + 12], SS:[ESP + 16], and SS:[ESP = 20], respectively, where the new value of ESP stored in step 4 is used. Since VM = 1, these are done as 8086 segment register loads.

   Else if VM = 0, check that the selectors in ES, DS, FS, and GS are valid in the interrupted routines. Null out invalid selectors to trap, if an attempt is made to access through them.
6. If RPL (CS) > CPL, pop the stack pointer SS:ESP from the stack. The ESP register is popped first, followed by 32 bits containing SS in the lower 16 bits. If VM = 0, SS is loaded as a protected mode segment register load. If VM = 1, an 8086 segment register load is used.
7. Resume execution of the interrupted routine. The VM bit in the FLAGS register (restored from the interrupt routine's stack image in step 1) determines whether the processor resumes the interrupted routine in Protected Mode or Virtual 8086 Mode.

FUNCTIONAL DATA

The Am386SXL microprocessor features a straightforward functional interface to the external hardware. The Am386SXL CPU has separate parallel buses for data and address. The data bus is 16-bits in width, and bi-directional. The address bus outputs 24-bit address values using 23 address lines and two Byte Enable signals.

The Am386SXL microprocessor has two selectable address bus cycles: address pipelined and non-address pipelined. The address pipelining option allows as much time as possible for data access by starting the pending bus cycle before the present bus cycle is finished. A non-pipelined bus cycle gives the highest bus performance by executing every bus cycle in two processor CLK cycles. For maximum design flexibility, the address pipelining option is selectable on a cycle-by-cycle basis.

The processor's bus cycle is the basic mechanism for information transfer, either from system to processor, or from processor to system. The Am386SXL microprocessor bus cycles perform data transfer in a minimum of only two clock periods. The maximum transfer bandwidth at 16 MHz is therefore 16 Mb/s. However, any bus cycle will be extended for more than two clock periods if external hardware withholds acknowledgment of the cycle.

The Am386SXL microprocessor can relinquish control of its local buses to allow mastership by other devices, such as Direct Memory Access (DMA) channels. When relinquished, HLDA is the only output pin driven by the Am386SXL microprocessor, providing near complete isolation of the processor from its system (all other output pins are in a float condition).

Signal Description Overview

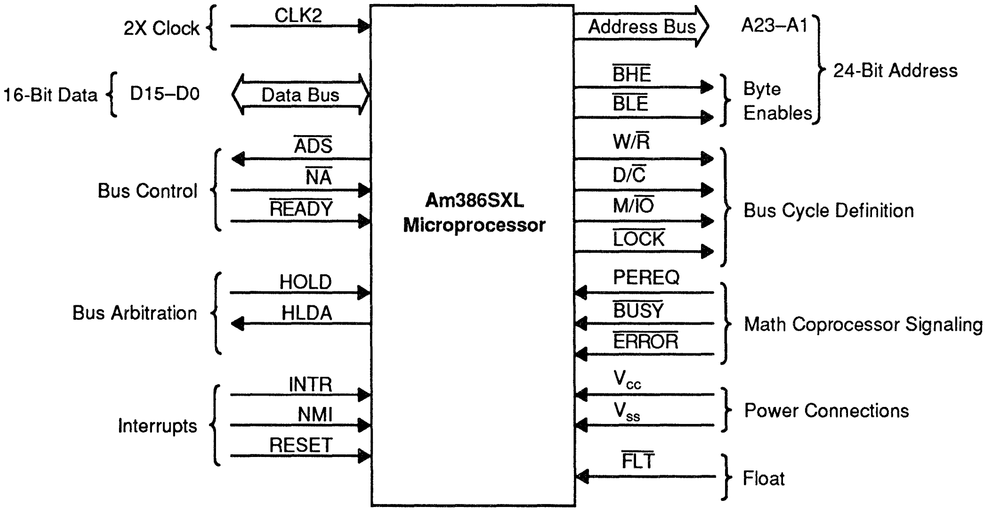
Below is a brief description of the Am386SXL microprocessor input and output signals arranged by functional groups.

Example signal: M/$\overline{\text{IO}}$—High voltage indicates memory selected
—Low voltage indicates I/O selected

The signal descriptions sometimes refer to Switching timing parameters, such as t25 Reset Setup Time and t26 Reset Hold Time. The values of these parameters can be found in the Switching Characteristics table.

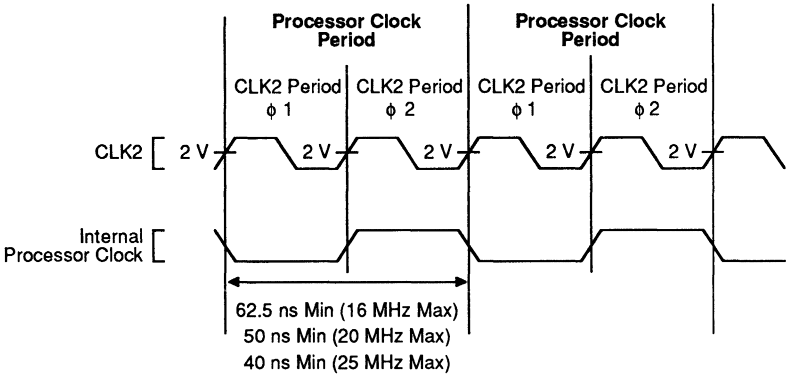
Clock (CLK2)

CLK2 provides the fundamental timing for the Am386SXL microprocessor. It is divided by two internally to generate the internal processor clock used for instruction execution. The internal clock is comprised of two phases, phase one and phase two. Each CLK2 period is a phase of the internal clock. Figure 23 illustrates the relationship. If desired, the phase of the internal processor clock can be synchronized to a known phase by ensuring the falling edge of the RESET signal meets the applicable setup and hold times, t25 and t26.



15022B–010

Figure 22. Functional Signal Groups



15022B–011

Figure 23. CLK2 Signal and Internal Processor Clock

Data Bus (D15–D0)

These three-state, bi-directional signals provide the general purpose data path between the Am386SXL microprocessor and other devices. The data bus outputs are active High and will float during Bus Hold Acknowledge. Data bus reads require that read-data setup and hold times (t21 and t22) be met relative to CLK2 for correct operation.

Address Bus (A23–A1, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$)

These three-state outputs provide physical memory addresses or I/O port addresses. A23–A16 are Low during I/O transfers, except for I/O transfers automatically generated by coprocessor instructions. During coprocessor I/O transfers, A22–A16 are driven Low and A23 is driven High, so that this address line can be used by external logic to generate the coprocessor select signal. Thus, the I/O address driven by the Am386SXL microprocessor for coprocessor commands is 8000F8H, the I/O addresses driven by the Am386SXL CPU for coprocessor data are 8000FCH or 8000FEH or cycles to a 387SX math coprocessor.

The address bus is capable of addressing 16 Mb of physical memory space (000000H through FFFFFFH), and 64 Kb of I/O address space (000000H through 00FFFFH) for programmed I/O. The address bus is active High and will float during Bus Hold Acknowledge.

The Byte Enable outputs, $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$, directly indicate which bytes of the 16-bit data bus are involved with the current transfer. $\overline{\text{BHE}}$ applies to D15–D8 and $\overline{\text{BLE}}$ applies to D7–D0. If both $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$ are asserted, then 16 bits of data are being transferred. See Table 13 for a complete decoding of these signals. The Byte Enables are active Low and will float during Bus Hold Acknowledge.

Bus Cycle Definition Signals (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$)

These three-state outputs define the type of bus cycle being performed: W/$\overline{\text{R}}$ distinguishes between write and read cycles; D/$\overline{\text{C}}$ distinguishes between data and control cycles; M/$\overline{\text{IO}}$ distinguishes between memory and I/O cycles; and, $\overline{\text{LOCK}}$ distinguishes between locked and unlocked bus cycles. All of these signals are active Low and will float during Bus Acknowledge.

The primary bus cycle definition signals are W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, and M/$\overline{\text{IO}}$, since these are the signals driven valid as $\overline{\text{ADS}}$ (Address Status output) becomes active. The $\overline{\text{LOCK}}$ is driven valid at the same time the bus cycle begins, which, due to address pipelining, could be after $\overline{\text{ADS}}$ becomes active. Exact bus cycle definitions, as a function of W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, and M/$\overline{\text{IO}}$, are given in Table 14.

Table 13. Byte Enable Definitions

$\overline{\text{BHE}}$	$\overline{\text{BLE}}$	Function
0	0	Word Transfer
0	1	Byte transfer on upper byte of the data bus, D15–D8
1	0	Byte transfer on lower byte of the data bus, D7–D0
1	1	Never occurs

$\overline{\text{LOCK}}$ indicates that other system bus masters are not to gain control of the system bus while it is active. $\overline{\text{LOCK}}$ is activated on the CLK2 edge that begins the first locked bus cycle (i.e., it is not active at the same time as the other bus cycle definition pins) and is deactivated when $\overline{\text{READY}}$ is returned at the end of the last bus cycle

Table 14. Bus Cycle Definition

M/$\overline{\text{IO}}$	D/$\overline{\text{C}}$	W/$\overline{\text{R}}$	Bus Cycle Type	Locked?
0	0	0	Interrupt Acknowledge	Yes
0	0	1	Does not occur	—
0	1	0	I/O Data Read	No
0	1	1	I/O Data Write	No
1	0	0	Memory Code Read	No
1	0	1	Halt: Address = 2 $\overline{\text{BHE}}$ = 1 $\overline{\text{BLE}}$ = 0 Shutdown: Address = 0 $\overline{\text{BHE}}$ = 1 $\overline{\text{BLE}}$ = 0	No
1	1	0	Memory Data Read	Some Cycles
1	1	1	Memory Data Write	Some Cycles

which is to be locked. The beginning of a bus cycle is determined when $\overline{\text{READY}}$ is returned in a previous bus cycle and another is pending ($\overline{\text{ADS}}$ is active), or the clock in which $\overline{\text{ADS}}$ is driven active if the bus was idle. This means that it follows more closely with the write data rules when it is valid, but may cause the bus to be locked longer than desired. The $\overline{\text{LOCK}}$ signal may be explicitly activated by the LOCK prefix on certain instructions.

$\overline{\text{LOCK}}$ is always asserted when executing the XCHG instruction, during descriptor updates, and during the interrupt acknowledge sequence.

Bus Control Signals ($\overline{\text{ADS}}$, $\overline{\text{READY}}$, $\overline{\text{NA}}$)

The following signals allow the processor to indicate when a bus cycle has begun, and allow other system hardware to control address pipelining and bus cycle termination.

Address Status ($\overline{\text{ADS}}$)

This three-state output indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, and A23–A1) are being driven at the Am386SXL microprocessor pins. $\overline{\text{ADS}}$ is an active Low output. Once $\overline{\text{ADS}}$ is driven active, valid address, Byte Enables, and definition signals will not change. In addition, $\overline{\text{ADS}}$ will remain active until its associated bus cycle begins (when $\overline{\text{READY}}$ is returned for the previous bus cycle when running pipelined bus cycles). When address pipelining is utilized, maximum throughput is achieved by initiating bus cycles when $\overline{\text{ADS}}$ and $\overline{\text{READY}}$ are active in the same clock cycle. $\overline{\text{ADS}}$ will float during Bus Hold Acknowledge. See sections Non-Pipelined Address and Pipelined Address for additional information on how $\overline{\text{ADS}}$ is asserted for different bus states.

Transfer Acknowledge ($\overline{\text{READY}}$)

This input indicates the current bus cycle is complete, and the active bytes indicated by $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$ are accepted or provided. When $\overline{\text{READY}}$ is sampled active during a read cycle or interrupt acknowledge cycle, the Am386SXL microprocessor latches the input data and terminates the cycle. When $\overline{\text{READY}}$ is sampled active during a write cycle, the processor terminates the bus cycle.

$\overline{\text{READY}}$ is ignored on the first bus state of all bus cycles, and sampled each bus state thereafter until asserted. $\overline{\text{READY}}$ must eventually be asserted to acknowledge every bus cycle, including Halt Indication and Shutdown Indication bus cycles. When being sampled, $\overline{\text{READY}}$ must always meet setup and hold times (t19 and t20) for correct operation.

Next Address Request ($\overline{\text{NA}}$)

This is used to request address pipelining. This input indicates the system is prepared to accept new values of $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, A23–A1, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, and M/$\overline{\text{IO}}$ from the Am386SXL CPU even if the end of the current cycle is not being acknowledged on $\overline{\text{READY}}$. If this input is active when sampled, the next address is driven onto the bus, provided the next bus request is already pending internally. $\overline{\text{NA}}$ is ignored in CLK cycles in which $\overline{\text{ADS}}$ or $\overline{\text{READY}}$ is activated. This signal is active Low and must satisfy setup and hold times (t15 and t16) for correct operation. See sections Read and Write Cycles and Pipelined Address for additional information.

Bus Arbitration Signals (HOLD, HLDA)

This section describes the mechanism by which the processor relinquishes control of its local buses when requested by another bus master device. See section Entering and Exiting Hold Acknowledge for additional information.

Bus Hold Request (HOLD)

This input indicates some device other than the Am386SXL microprocessor requires bus mastership. When control is granted, the Am386SXL CPU floats A23–A1, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, D15–D0, $\overline{\text{LOCK}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, W/$\overline{\text{R}}$, and $\overline{\text{ADS}}$, and then activates HLDA, thus entering the Bus Hold Acknowledge state. The local bus will remain granted to the requesting master until HOLD becomes inactive. When HOLD becomes inactive, the Am386SXL microprocessor will deactivate HLDA and drive the local bus (at the same time), thus terminating the Hold Acknowledge condition.

HOLD must remain asserted as long as any other device is a local bus master. External pull-up resistors may be required when in the Hold Acknowledge (HLDA) state, since none of the Am386SXL microprocessor floated outputs have internal pull-up resistors. See section Resistor Recommendations for additional information. HOLD is not recognized while RESET is active. If RESET is asserted while HOLD is asserted, RESET has priority and places the bus into an idle state, rather than the Hold Acknowledge (high impedance) state.

HOLD is a level-sensitive, active High, synchronous input. HOLD signals must always meet setup and hold times (t23 and t24) for correct operation.

Bus Hold Acknowledge (HLDA)

When active (High), this output indicates the Am386SXL microprocessor has relinquished control of its local bus in response to an asserted HOLD signal, and is in the Bus Hold Acknowledge state.

The Bus Hold Acknowledge state offers near complete signal isolation. In the HLDA state is the only signal being driven by the Am386SXL microprocessor. The other output or bi-directional signals (D15–D0, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, A23–A1, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, and $\overline{\text{ADS}}$) are in a high-impedance state so the requesting bus master may control them. These pins remain Off throughout the time that HLDA remains active (see Table 15). Pull-up resistors may be desired on several signals to avoid spurious activity when no bus master is driving them. See section Resistor Recommendations for additional information.

When the HOLD signal is made inactive, the Am386SXL microprocessor will deactivate HLDA and drive the bus. One rising edge on the NMI input is remembered for processing after the HOLD input is negated.

Table 15. Output Pin State During HOLD

Pin Value	Pin Names
1	HLDA
Float	$\overline{\text{LOCK}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, W/$\overline{\text{R}}$, $\overline{\text{ADS}}$, A23–A1, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, D15–D0

In addition to the normal usage of Hold Acknowledge with DMA controllers or master peripherals, the near complete isolation has particular attractiveness during system test when test equipment drives the system, and in hardware fault-tolerant applications.

HOLD Latencies

The maximum possible HOLD latency depends on the software being executed. The actual HOLD latency at any time depends on the current bus activity, the state of the $\overline{\text{LOCK}}$ signal (internal to the CPU) activated by the LOCK prefix, and interrupts. The Am386SXL microprocessor will not honor a HOLD request until the current bus operation is complete.

The Am386SXL microprocessor breaks 32-bit data or I/O accesses into 2 internally locked 16-bit bus cycles; the $\overline{\text{LOCK}}$ signal is not asserted. The Am386SXL microprocessor breaks unaligned 16-bit or 32-bit data or I/O accesses into 2 or 3 internally locked 16-bit bus cycles. Again, the $\overline{\text{LOCK}}$ signal is not asserted but a HOLD request will not be recognized until the end of the entire transfer.

Wait states affect HOLD latency. The Am386SXL microprocessor will not honor a HOLD request until the end of the current bus operation, no matter how many wait states are required. Systems with DMA where data transfer is critical must insure that $\overline{\text{READY}}$ returns promptly.

Coprocessor Interface Signals (PEREQ, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$)

In the following sections are descriptions of signals dedicated to the math coprocessor interface. In addition to the data bus, address bus, and bus cycle definition signals, the following signals control communication between the Am386SXL microprocessor and its 387SX math coprocessor extension.

Coprocessor Request (PEREQ)

When asserted (High), this input signal indicates a coprocessor request for a data operand to be transferred to/from memory by the Am386SXL Microprocessor. In response, the Am386SXL microprocessor transfers information between the math coprocessor and memory. Because the Am386SXL CPU has internally stored the math coprocessor op-code being executed, it performs the requested data transfer with the correct direction and memory address.

PEREQ is level-sensitive active High asynchronous signal. Setup and hold times (t29 and t30) relative to the CLK2 signal must be met to guarantee recognition at a particular clock edge. This signal is provided with a weak internal pull-down resistor of around 20 Kohms to Ground so that it will not float active when left unconnected.

Coprocessor Busy ($\overline{\text{BUSY}}$)

When asserted Low, this input indicates that the coprocessor is still executing an instruction, and is not yet able to accept another. When the Am386SXL microprocessor encounters any coprocessor instruction which operates on the numerics stack (e.g., load, pop, or arithmetic operation), or the WAIT instruction, this input is first automatically sampled until it is seen to be inactive. This sampling of the $\overline{\text{BUSY}}$ input prevents overrunning the execution of a previous coprocessor instruction.

The FNINIT, FNSTENV, FNSAVE, FNSTSW, FNSTCW, and FNCLEX coprocessor instructions are allowed to execute even if $\overline{\text{BUSY}}$ is active, since these instructions are used for coprocessor initialization and exception-clearing.

$\overline{\text{BUSY}}$ is an active Low, level-sensitive, asynchronous signal. Setup and hold times (t29 and t30), relative to the CLK2 signal, must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 Kohms to V_{CC} so that it will not float active when left unconnected.

$\overline{\text{BUSY}}$ serves an additional function. If $\overline{\text{BUSY}}$ is sampled Low at the falling edge of RESET, the Am386SXL microprocessor performs an internal self-test (see section Bus Activity During and Following Reset). If $\overline{\text{BUSY}}$ is sampled High, no self-test is performed.

Coprocessor Error ($\overline{\text{ERROR}}$)

When asserted Low, this input signal indicates that the previous coprocessor instruction generated a coprocessor error of a type not masked by the coprocessor's control register. This input is automatically sampled by the Am386SXL microprocessor when a coprocessor instruction is encountered, and if active, the Am386SXL CPU generates Exception 16 to access the error-handling software.

Several coprocessor instructions, generally those which clear the numeric error flags in the coprocessor or save coprocessor state, do execute without the Am386SXL CPU generating Exception 16 even if $\overline{\text{ERROR}}$ is active. These instructions are FNINIT, FNCLEX, FNSTSW, FNSTSWAX, FNSTCW, FNSTENV, and FNSAVE.

$\overline{\text{ERROR}}$ is an active Low, level-sensitive, asynchronous signal. Setup and hold times (t29 and t30), relative to the CLK2 signal, must be met to guarantee recognition at a particular clock edge. This pin is provided with a weak internal pull-up resistor of around 20 Kohms to V_{CC} so that it will not float active when left unconnected.

Interrupt Signals (INTR, NMI, RESET)

The following descriptions cover inputs that can interrupt or suspend execution of the processor's current instruction stream.

Maskable Interrupt Request (INTR)

When asserted, this input indicates a request for interrupt service, which can be masked by the Am386SXL microprocessor Flag Register IF bit. When the Am386SXL CPU responds to the INTR input, it performs two interrupt acknowledge bus cycles and, at the end of the second, latches an 8-bit interrupt vector on D7–D0 to identify the source of the interrupt.

INTR is an active High, level-sensitive, asynchronous signal. Setup and hold times (t27 and t28), relative to the CLK2 signal, must be met to guarantee recognition at a particular clock edge. To assure recognition of an INTR request, INTR should remain active until the first interrupt acknowledge bus cycle begins. INTR is sampled at the beginning of every instruction in the Am386SXL microprocessor's Execution Unit. In order to be recognized at a particular instruction boundary, INTR must be active at least eight CLK2 clock periods before the beginning of the instruction. If recognized, the Am386SXL CPU will begin execution of the interrupt.

Non-Maskable Interrupt Request (NMI)

This input indicates a request for interrupt service which cannot be masked by software. The non-maskable interrupt request is always processed according to the pointer or gate in slot 2 of the interrupt table. Because of the fixed NMI slot assignment, no interrupt acknowledge cycles are performed when processing NMI.

NMI is an active High, rising edge-sensitive, asynchronous signal. Setup and hold times (t27 and t28), relative to the CLK2 signal, must be met to guarantee recognition at a particular clock edge. To assure recognition of NMI, it must be inactive for at least eight CLK2 periods, and then be active for at least eight CLK2 periods before the beginning of the instruction boundary in the Am386SXL microprocessor's Execution Unit.

Once NMI processing has begun, no additional NMI's are processed until after the next IRET instruction, which is typically the end of the NMI service routine. If NMI is re-asserted prior to that time, however, one rising edge on NMI will be remembered for processing after executing the next IRET instruction.

Interrupt Latency

The time that elapses before an interrupt request is serviced (interrupt latency) varies according to several factors. This delay must be taken into account by the interrupt source. Any of the following factors can affect interrupt latency:

1. If interrupts are masked, an INTR request will not be recognized until interrupts are re-enabled.
2. If an NMI is currently being serviced, an incoming NMI request will not be recognized until the Am386SXL microprocessor encounters the IRET instruction.

3. An interrupt request is recognized only on an instruction boundary of the Am386SXL microprocessor's Execution Unit except for the following cases:
   - Repeat string instructions can be interrupted after each iteration.
   - If the instruction loads the Stack Segment register, an interrupt is not processed until after the following instruction, which should be an ESP. This allows the entire stack pointer to be loaded without interruption.
   - If an instruction sets the interrupt flag (enabling interrupts), an interrupt is not processed until after the next instruction.

   The longest latency occurs when the interrupt request arrives while the Am386SXL microprocessor is executing a long instruction such as multiplication, division, or a task switch in the Protected Mode.
4. Saving the Flags register and CS:EIP registers.
5. If interrupt service routine requires a task switch, time must be allowed for the task switch.
6. If the interrupt service routine saves registers that are not automatically saved by the Am386SXL microprocessor.

Reset

This input signal suspends any operation in progress and places the Am386SXL microprocessor in a known reset state. The Am386SXL CPU is reset by asserting RESET for 15 or more CLK2 periods (80 or more CLK2 periods before requesting self-test). When RESET is active, all other input pins, except $\overline{FLT}$, are ignored, and all other bus pins are driven to an Idle Bus state, as shown in Table 16. If RESET and HOLD are both active at a point in time, RESET takes priority even if the Am386SXL microprocessor was in a Hold Acknowledge state prior to RESET active.

Reset is an active High, level-sensitive, synchronous signal. Setup and hold times (t25 and t26) must be met in order to assure proper operation of the Am386SXL microprocessor.

Bus Transfer Mechanism

All data transfers occur as a result of one or more bus cycles. Logical data operands of byte and word lengths may be transferred without restrictions on physical address alignment. Any byte boundary may be used, although two physical bus cycles are performed as required for unaligned operand transfers.

The Am386SXL microprocessor address signals are designed to simplify external system hardware. Higher-order address bits are provided by A23–A1. $\overline{BHE}$ and $\overline{BLE}$ provide linear selects for the two bytes of the 16-bit data bus.

Byte Enable outputs $\overline{BHE}$ and $\overline{BLE}$ are asserted when their associated data bus bytes are involved with the present bus cycle, as listed in Table 17.

Each bus cycle is composed of at least two bus states. Each bus state requires one processor clock period. Additional bus states added to a single bus cycle are called wait states. See section Bus Functional Description.

Table 16. Pin State (Bus Idle) During Reset

Pin Name	Signal Level During Reset
$\overline{ADS}$	1
D15–D0	Float
$\overline{BHE}$, $\overline{BLE}$	0
A23–A1	1
W/$\overline{R}$	0
D/$\overline{C}$	1
M/$\overline{IO}$	0
$\overline{LOCK}$	1
HLDA	0

Table 17. Byte Enables and Associated Data and Operand Bytes

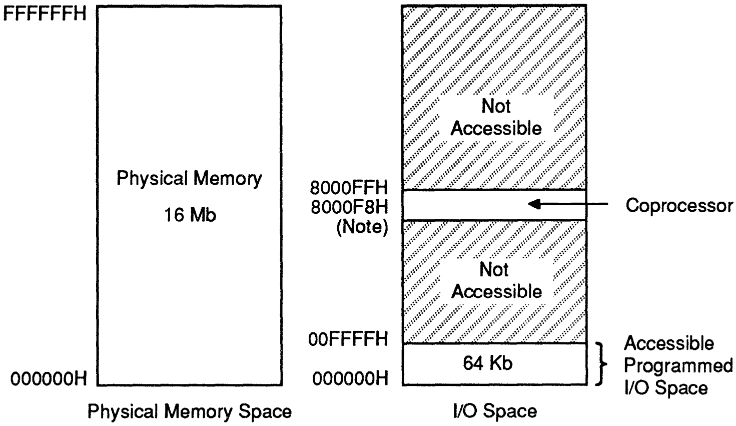
Byte Enable Signal	Associated Data Bus Signals
$\overline{BLE}$	D7–D0 (Byte 0—least significant)
$\overline{BHE}$	D15–D8 (Byte 1—most significant)

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, I/O-mapped, or both. As shown in Figure 24, physical memory addresses range from 000000H to 0FFFFFFH (16 Mb) and I/O addresses from 000000H to 00FFFFH (64 kb). Note the I/O addresses used by the automatic I/O cycles for coprocessor communication are 8000F8H to 8000FFH, beyond the address range of programmed I/O, to allow easy generation of a coprocessor chip select signal using the A23 and M/$\overline{IO}$ signals.

Bus Functional Description

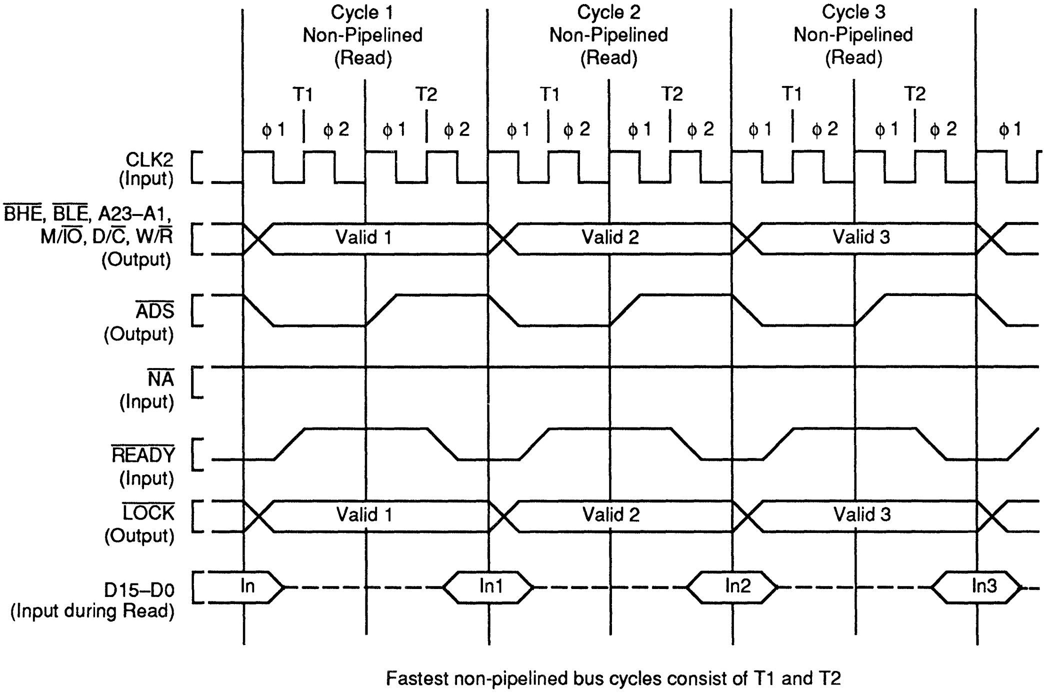
The Am386SXL microprocessor has separate, parallel buses for data and address. The data bus is 16-bits in width, and bi-directional. The address bus provides a 24-bit value using 23 signals for the 23 upper-order address bits and 23 Byte Enable signals to directly indicate the active bytes. These buses are interpreted and controlled by several definition signals.



Note: Since A23 is High during automatic communication with coprocessor, A23 High and M/$\overline{\text{IO}}$ Low can be used to easily generate a coprocessor select signal.

15022B–012

Figure 24. Physical Memory and I/O Spaces



Fastest non-pipelined bus cycles consist of T1 and T2

15022B–013

Figure 25. Fastest Read Cycles with Non-Pipelined Address Timing

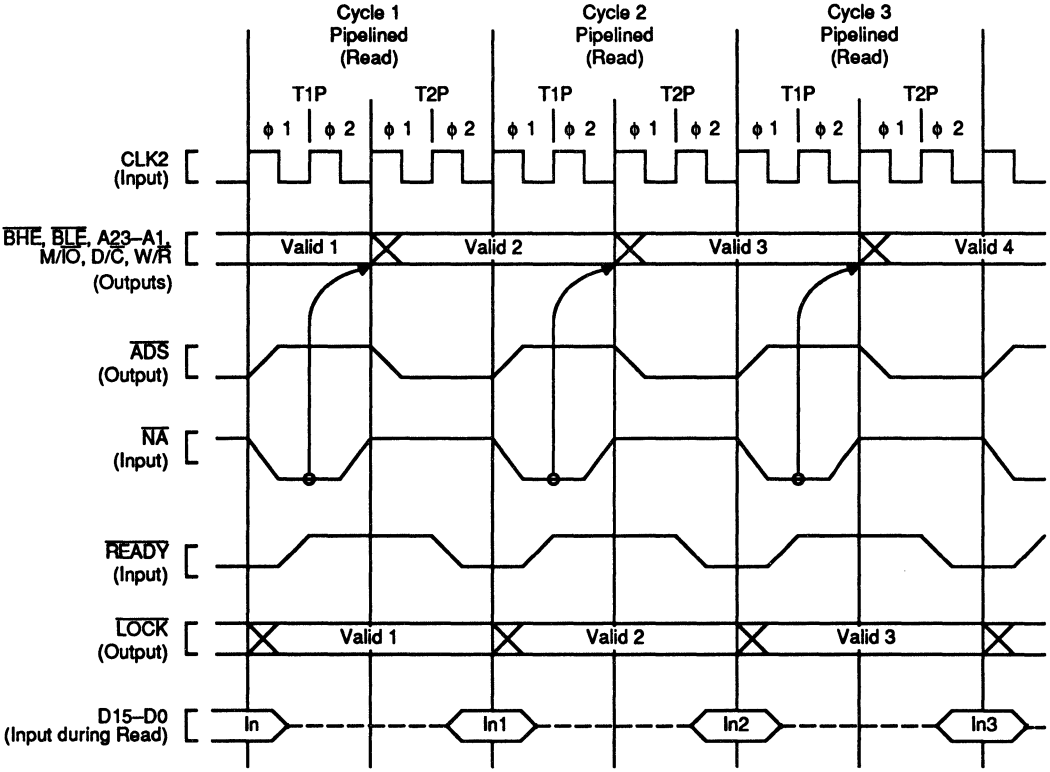
The definition of each bus cycle is given by three signals: M/$\overline{IO}$, W/$\overline{R}$, and D/$\overline{C}$. At the same time, a valid address is present on the Byte Enable signals, $\overline{BHE}$ and $\overline{BLE}$, and the other address signals, A23–A1. A status signal, $\overline{ADS}$, indicates when the Am386SXL microprocessor issues a new bus cycle definition and address.

Collectively, the address bus, data bus, and all associated control signals are referred to simply as the bus. When active, the bus performs one of the bus cycles below:

1. Read from memory space;
2. Locked read from memory space;
3. Write to memory space;
4. Locked write to memory space;
5. Read from I/O space (or math coprocessor);
6. Write to I/O space (or math coprocessor);
7. Interrupt acknowledge (always locked);
8. Indicate halt, or indicate shutdown.

Table 14 shows the encoding of the bus cycle definition signals for each bus cycle. See section Bus Cycle Definition Signals for additional information.

When the Am386SXL microprocessor bus is not performing one of the activities listed above, it is either idle or in the Hold Acknowledge state, which may be detected externally. The idle state can be identified by the Am386SXL CPU giving no further assertions on its address strobe output ($\overline{ADS}$) since the beginning of its most recent cycle, and the most recent bus cycle having been terminated. The Hold Acknowledge state is identified by the Am386SXL microprocessor asserting its Hold Acknowledge (HLDA) output.


Cycle 1 Pipelined (Read)
Cycle 2 Pipelined (Read)
Cycle 3 Pipelined (Read)
T1P
T2P
φ1
φ2
CLK2 (Input)
$\overline{BHE}$, $\overline{BLE}$, A23–A1, M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$ (Outputs)
Valid 1
Valid 2
Valid 3
Valid 4
$\overline{ADS}$ (Output)
$\overline{NA}$ (Input)
$\overline{READY}$ (Input)
$\overline{LOCK}$ (Output)
D15–D0 (Input during Read)
In
In1
In2
In3

Fastest pipelined bus cycles consist of T1P and T2P

15022B–014

Figure 26. Fastest Read Cycles with Pipelined Address Timing

The shortest time unit of bus activity is a bus state. A bus state is one processor clock period (two CLK2 periods) in duration. A complete data transfer occurs during a bus cycle, composed of two or more bus states.

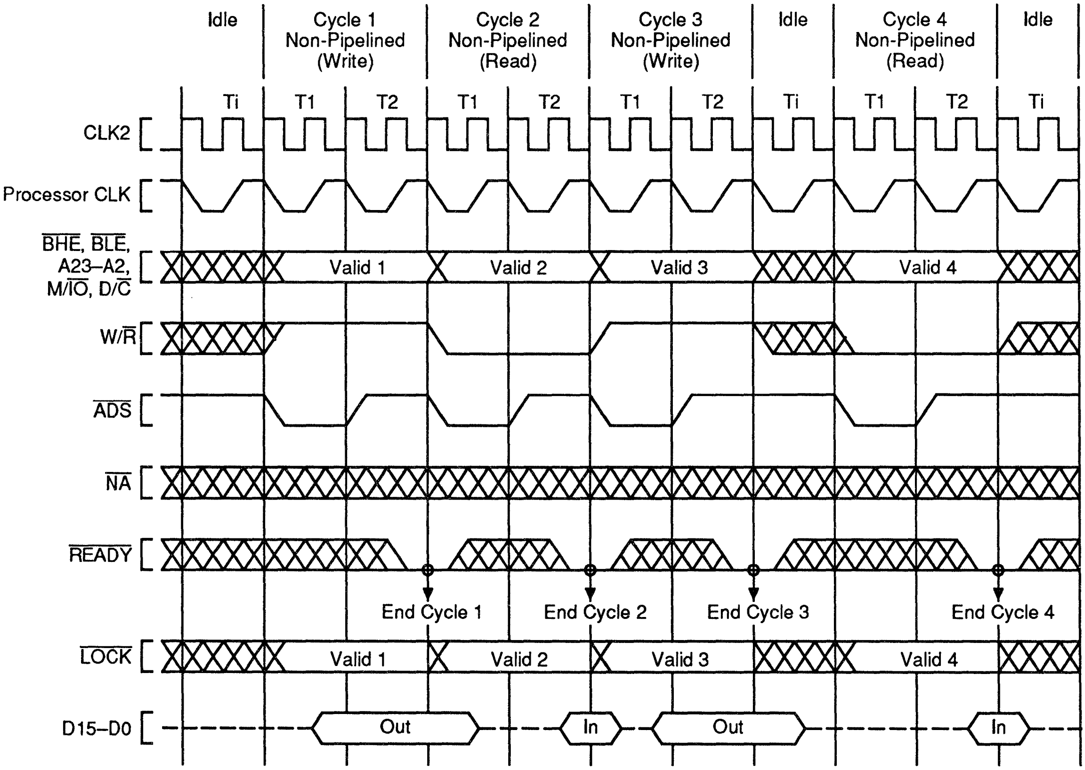
The fastest Am386SXL microprocessor bus cycle requires only two bus states. For example, three consecutive bus read cycles, each consisting of two bus states, are shown in Figure 25. The bus states in each cycle are named T1 and T2. Any memory or I/O address may be accessed by such a two-state bus cycle, if the external hardware is fast enough.

Every bus cycle continues until it is acknowledged by the external system hardware, using the Am386SXL microprocessor $\overline{\text{READY}}$ input. Acknowledging the bus cycle at the end of the first T2 results in the shortest bus cycle, requiring only T1 and T2. If $\overline{\text{READY}}$ is not immediately asserted however, T2 states are repeated indefinitely until the $\overline{\text{READY}}$ input is sampled active.

The address pipelining option provides a choice of bus cycle timings. Pipelined or non-pipelined address timing is selectable on a cycle-by-cycle basis with the Next Address ($\overline{\text{NA}}$) input.

When address pipelining is selected, the address ($\overline{\text{BHE}}$, $\overline{\text{BLE}}$, and A23–A1) and definition (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, and $\overline{\text{LOCK}}$) of the next cycle are available before the end of the current cycle. To signal their availability, the Am386SXL microprocessor address status output ($\overline{\text{ADS}}$) is asserted. Figure 26 illustrates the fastest read cycles with pipelined address timing.

Note from Figure 26 the fastest bus cycles using pipelined address require only two bus states, named T1P and T2P. Therefore, cycles with pipelined address timing allow the same data bandwidth as non-pipelined cycles, but address-to-data access time is increased by one T-state time compared to that of a non-pipelined cycle.



Note: Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state; an active bus cycle can immediately follow the write cycle.

15022B–015

Figure 27. Various Bus Cycles with Non-Pipelined Address (Zero Wait States)

Read and Write Cycles

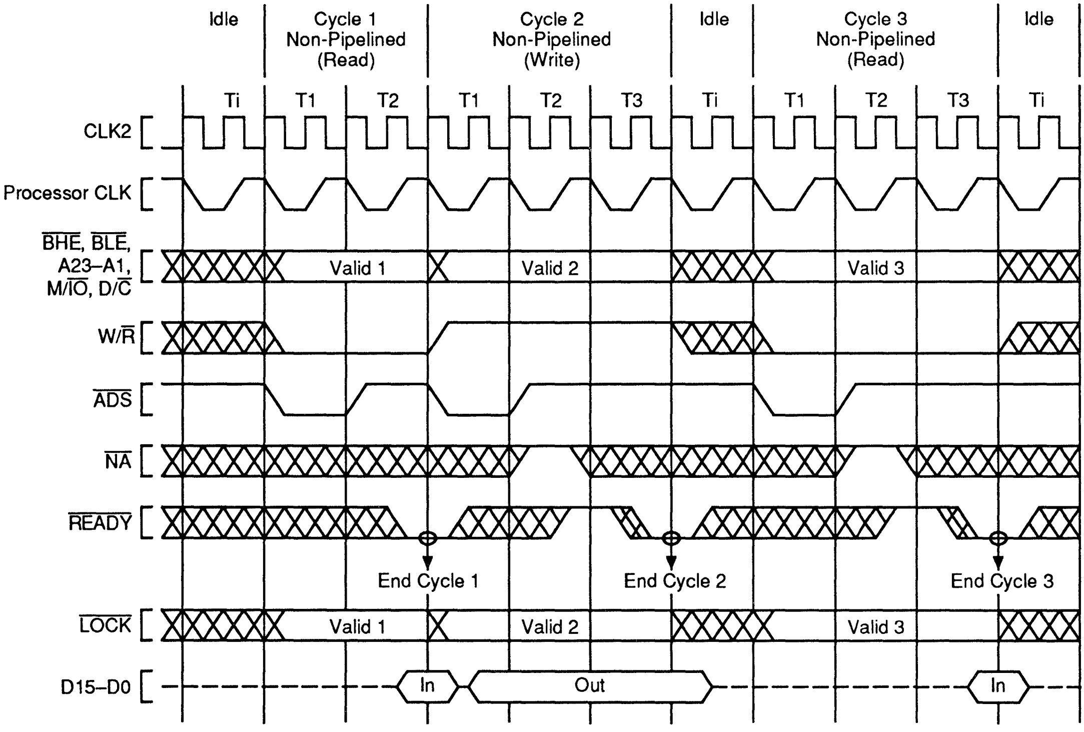
Data transfers occur as a result of bus cycles, classified as read or write cycles. During read cycles, data is transferred from an external device to the processor. During write cycles, data is transferred from the processor to an external device.

Two choices of address timing are dynamically selectable: non-pipelined or pipelined. After an idle bus state, the processor always uses non-pipelined address timing. However, the $\overline{\text{NA}}$ (Next Address) input may be asserted to select pipelined address timing for the next bus cycle. When pipelining is selected and the Am386SXL microprocessor has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by $\overline{\text{READY}}$.

Terminating a read or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the $\overline{\text{READY}}$ input. Until acknowledged, the processor inserts wait states into the bus cycle, to allow adjustment for the speed of any external device. External hardware, which has decoded the address and bus cycle type, asserts the $\overline{\text{READY}}$ input at the appropriate time.

At the end of the second bus state within the bus cycle, $\overline{\text{READY}}$ is sampled. At that time, if external hardware acknowledges the bus cycle by asserting $\overline{\text{READY}}$, the bus cycle terminates as shown in Figure 27. If $\overline{\text{READY}}$ is negated, as in Figure 28, the Am386SXL microprocessor executes another bus state (a wait state) and $\overline{\text{READY}}$ is sampled again at the end of that state. This continues indefinitely until the cycle is acknowledged by $\overline{\text{READY}}$ asserted.

When the current cycle is acknowledged, the Am386SXL microprocessor terminates it. When a read cycle is acknowledged, the Am386SXL CPU latches the infor-mation present at its data pins. When a write cycle is acknowledged, the Am386SXL microprocessor's write data remains valid throughout phase one of the next bus state, to provide write data hold time.



Note: Idle states are shown here for diagram variety only. Write cycles are not always followed by an idle state; an active bus cycle can immediately follow the write cycle.

15022B–016

Figure 28. Various Bus Cycles with Non-Pipelined Address (Various Number of Wait States)

Non-Pipelined Address

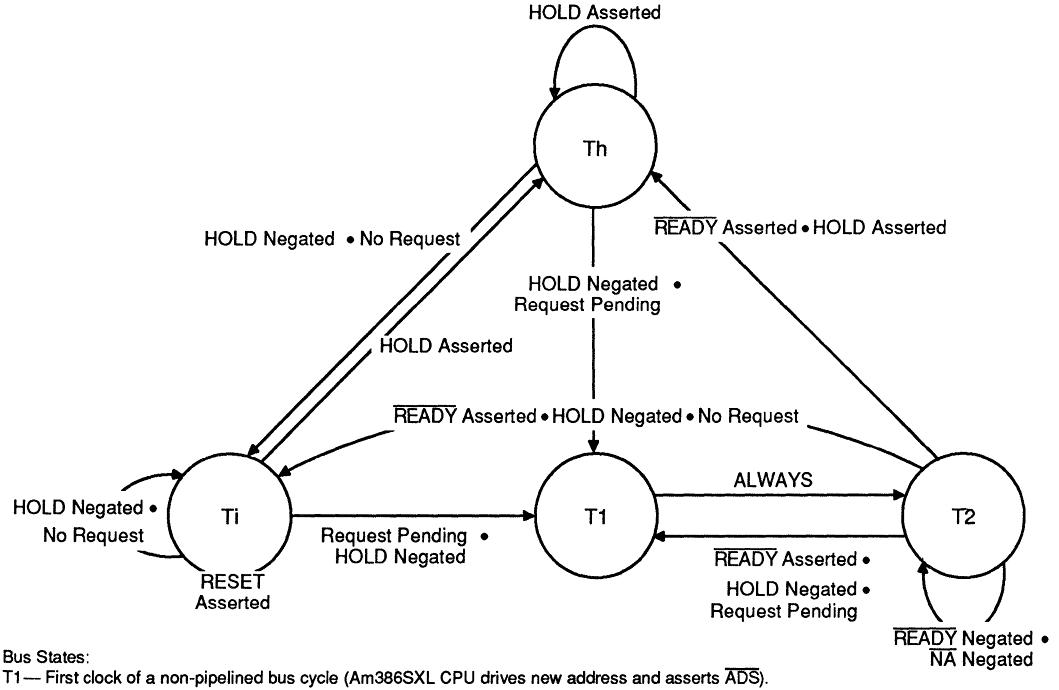
Any bus cycle may be performed with non-pipelined address timing. For example, Figure 27 shows a mixture of read and write cycles with non-pipelined address timing. Figure 27 shows that the fastest possible cycles with non-pipelined address have two bus states per bus cycle. The states are named T1 and T2. In phase one of T1, the address signals and bus cycle definition signals are driven valid and, to signal their availability, address strobe ($\overline{\text{ADS}}$) is simultaneously asserted.

During read or write cycles the data bus behaves as follows. If the cycle is a read, the Am386SXL microprocessor floats its data signal to allow driving by the external device being addressed. The Am386SXL microprocessor requires that all data bus pins be at a valid logic state (High or Low) at the end of each read cycle, when $\overline{\text{READY}}$ is asserted. The system **must** be designed to meet this requirement. If the cycle is a write, data signals are driven by the Am386SXL CPU beginning in phase two of T1 until phase one of the bus state following cycle acknowledgment.

Figure 28 illustrates non-pipelined bus cycles with one wait state added to Cycles 2 and 3. $\overline{\text{READY}}$ is sampled inactive at the end of the first T2 in Cycles 2 and 3. Therefore, Cycles 2 and 3 have T2 repeated again. At the end of the second T2, $\overline{\text{READY}}$ is sampled active.

When address pipelining is not used, the address and bus cycle definition remain valid during all wait states. When wait states are added, and it is desirable to maintain non-pipelined address timing, it is necessary to negate $\overline{\text{NA}}$ during each T2 state, except the last one, as shown in Figure 28, Cycles 2 and 3. If $\overline{\text{NA}}$ is sampled active during a T2 other than the last one, the next state would be T2I or T2P instead of another T2.

The bus states and transitions, when address pipelining is not used, are completely illustrated by Figure 29. The bus transitions between four possible states, T1, T2, Ti, and Th. Bus cycles consist of T1 and T2, with T2 being repeated for wait states. Otherwise the bus may be idle, Ti, or in the Hold Acknowledge state Th.


HOLD Asserted
Th
HOLD Negated • No Request
READY Asserted • HOLD Asserted
HOLD Negated • Request Pending
HOLD Asserted
READY Asserted • HOLD Negated • No Request
ALWAYS
HOLD Negated • No Request
Ti
T1
T2
Request Pending • HOLD Negated
READY Asserted • HOLD Negated • Request Pending
RESET Asserted
READY Negated • NA Negated

Bus States:
T1— First clock of a non-pipelined bus cycle (Am386SXL CPU drives new address and asserts $\overline{\text{ADS}}$).
T2— Subsequent clocks of a bus cycle when $\overline{\text{NA}}$ has not been sampled asserted in the current bus cycle.
Ti — Idle state.
Th— Hold Acknowledge state (Am386SXL CPU asserts HLDA).

The fastest bus cycle consists of two states: T1 and T2.

Four basic bus states describe bus operation when not using pipelined address.

15022B–017

Figure 29. Bus States (Not Using Pipelined Address)

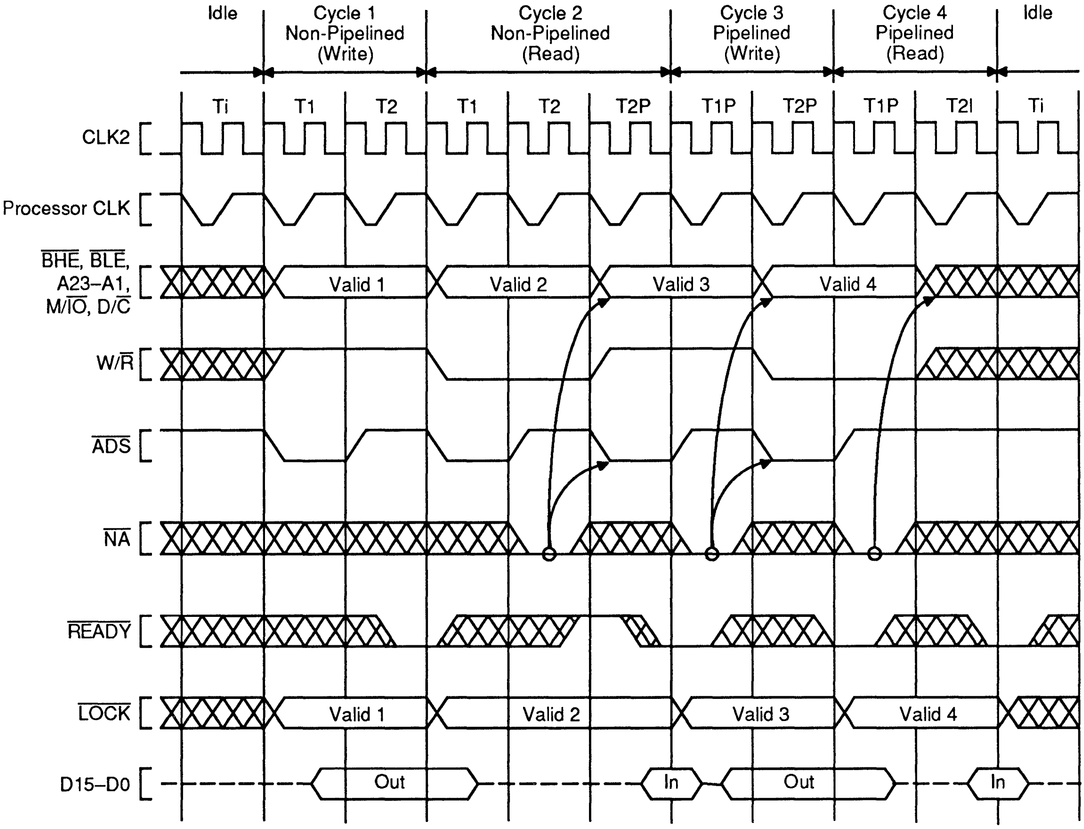
Bus cycles always begin with T1. T1 always leads to T2. If a bus cycle is not acknowledged during T2 and $\overline{NA}$ is inactive, T2 is repeated. When a cycle is acknowledged during T2, the following state will be T1 of the next bus cycle, if a bus request is pending internally, or Ti, if there is no bus request pending, or Th, if the HOLD input is being asserted.

Use of pipelined address allows the Am386SXL microprocessor to enter three additional bus states not shown in Figure 29. Figure 33 is the complete bus state diagram, including pipelined address cycles.

Pipelined Address

Address pipelining is the option of requesting the address and the bus cycle definition of the next internally pending bus cycle before the current bus cycle is acknowledged with $\overline{READY}$ asserted. $\overline{ADS}$ is asserted by the Am386SXL microprocessor when the next address is issued. The address pipelining option is controlled on a cycle-by-cycle basis with the $\overline{NA}$ input signal.

Once a bus cycle is in progress and the current address has been valid for at least one entire bus state, the $\overline{NA}$ input is sampled at the end of every phase one until the bus cycle is acknowledged. During non-pipelined bus cycles, $\overline{NA}$ is sampled at the end of phase one in every T2. An example is Cycle 2 in Figure 30, during which $\overline{NA}$ is sampled at the end of phase one of every T2 (it was asserted once during the first T2 and has no further effect during that bus cycle).



Note: Following any idle bus state (Ti), addresses are non-pipelined. Within non-pipelined bus cycles, $\overline{NA}$ is only sampled during wait states. Therefore, to begin address pipelining during a group of non-pipelined bus cycles requires a non-pipelined cycle with at least one wait state (Cycle 2 above).

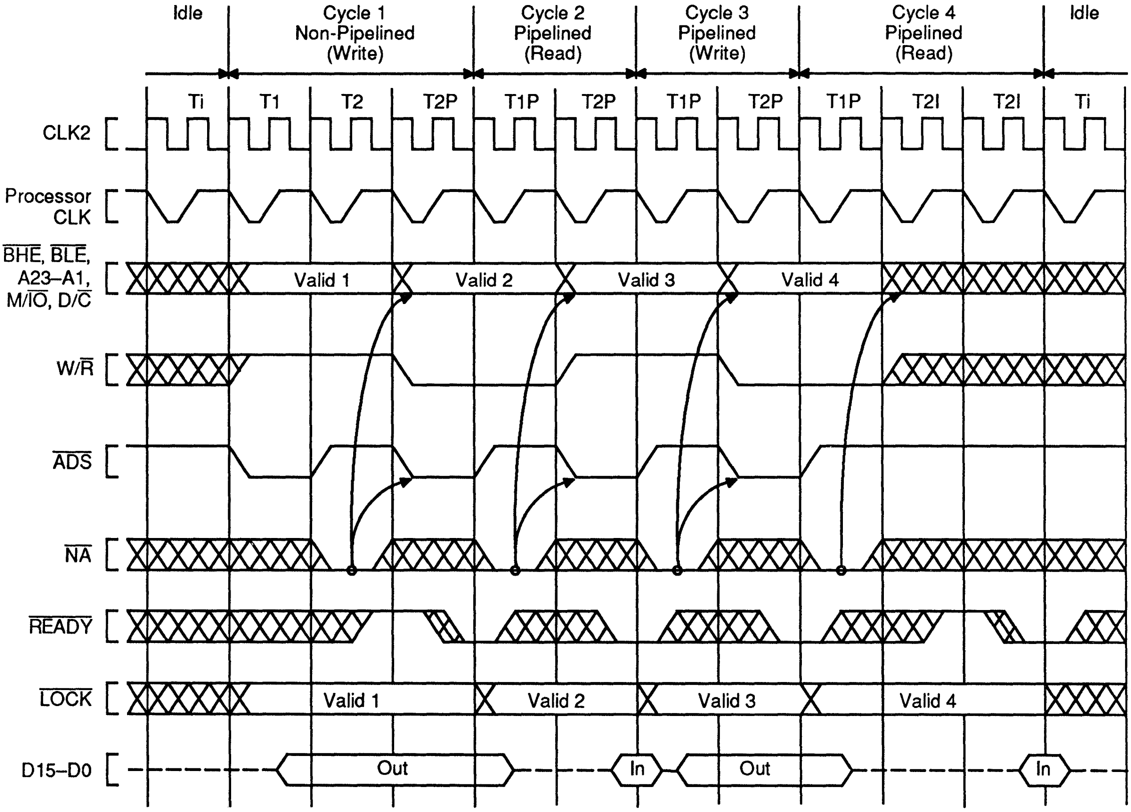
15022B–018

Figure 30. Transitioning to Pipelined Address During Burst of Bus Cycles

If NA is sampled active, the Am386SXL microprocessor is free to drive the address and bus cycle definition of the next bus cycle, and assert ADS, as soon as it has a bus request internally pending. It may drive the next address as early as the next bus state, whether the current bus cycle is acknowledged at that time or not.

Regarding the details of address pipelining, the Am386SXL CPU has the following characteristics:

1. The next address may appear as early as the bus state after NA was sampled active (see Figures 30 and 31). In that case, state T2P is entered immediately. However, when there is not an internal bus request already pending, the next address will not be available immediately after NA is asserted and T2I is entered instead of T2P (see Figure 32, Cycle 3). Provided the current bus cycle is not yet acknowledged by READY asserted, T2P will be entered as soon as the Am386SXL microprocessor does drive the next address. External hardware should therefore observe the ADS output as confirmation the next address is actually being driven on the bus.
2. Any address which is validated by a pulse on the ADS output will remain stable on the address pins for at least two processor clock periods. The Am386SXL CPU cannot produce a new address more frequently than every two processor clock periods (see Figures 30, 31, and 32).
3. Only the address and bus cycle definition of the very next bus cycle is available. The pipelining capability cannot look further than one bus cycle ahead (see Figure 32, Cycle 1).



Note: Following any idle bus state (Ti) the address is always non-pipelined and NA is only sampled during wait states. To start address pipelining after an idle state requires a non-pipelined cycle with at least one wait state (Cycle 1 above). The pipelined cycles (2, 3, and 4 above) are shown with various numbers of wait states.

15022B–019

Figure 31. Fastest Transition to Pipelined Address Following Idle Bus State

The complete bus state transition diagram, including operation with pipelined address, is given in Figure 33. Note that it is a superset of the diagram for non-pipelined address only, and the three additional bus states for pipelined address are drawn in bold.

The fastest bus cycle with pipelined address consists of just two bus states, T1P and T2P (recall for non-pipelined address it is T1 and T2). T1P is the first bus state of a pipelined cycle.

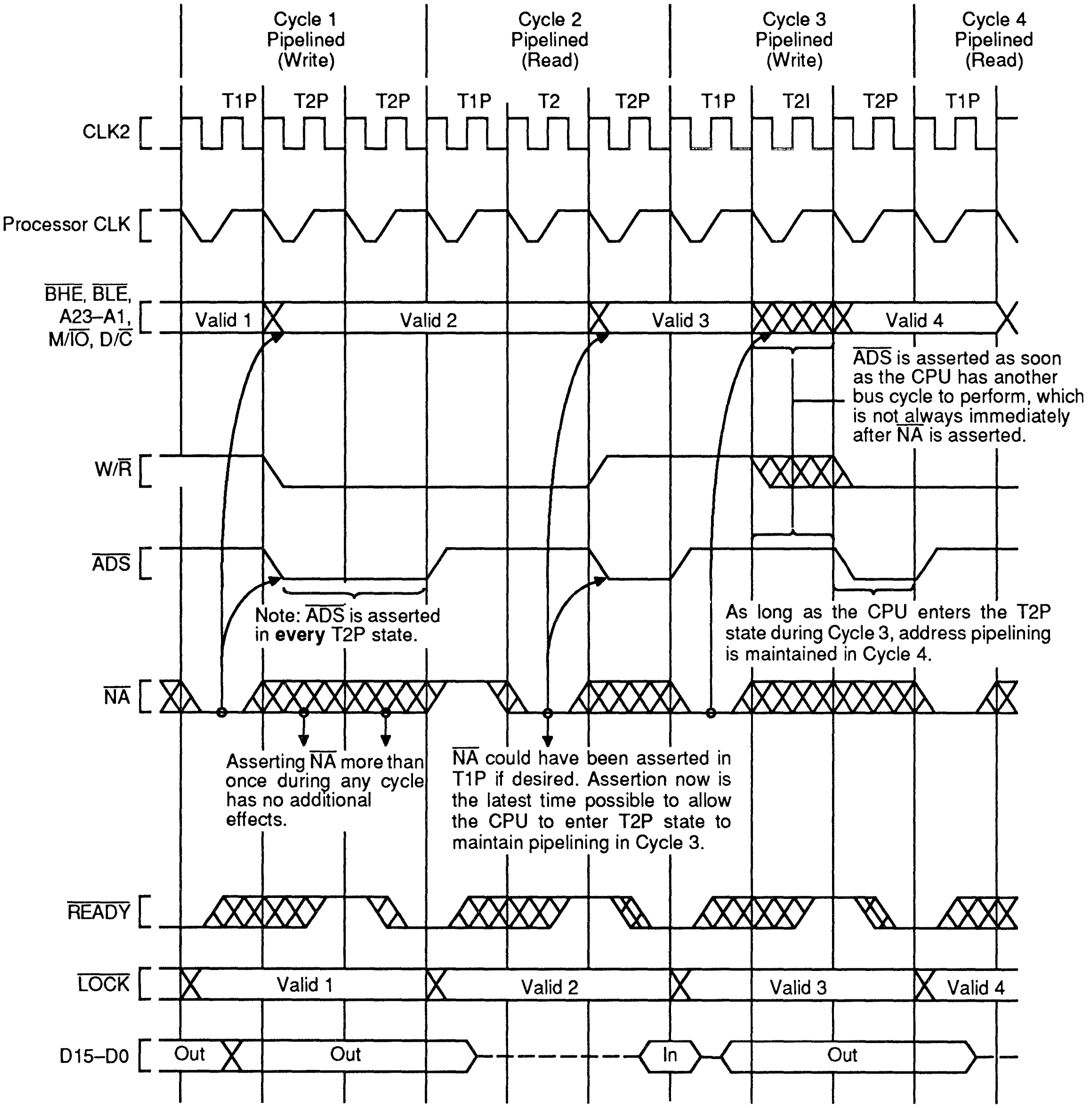
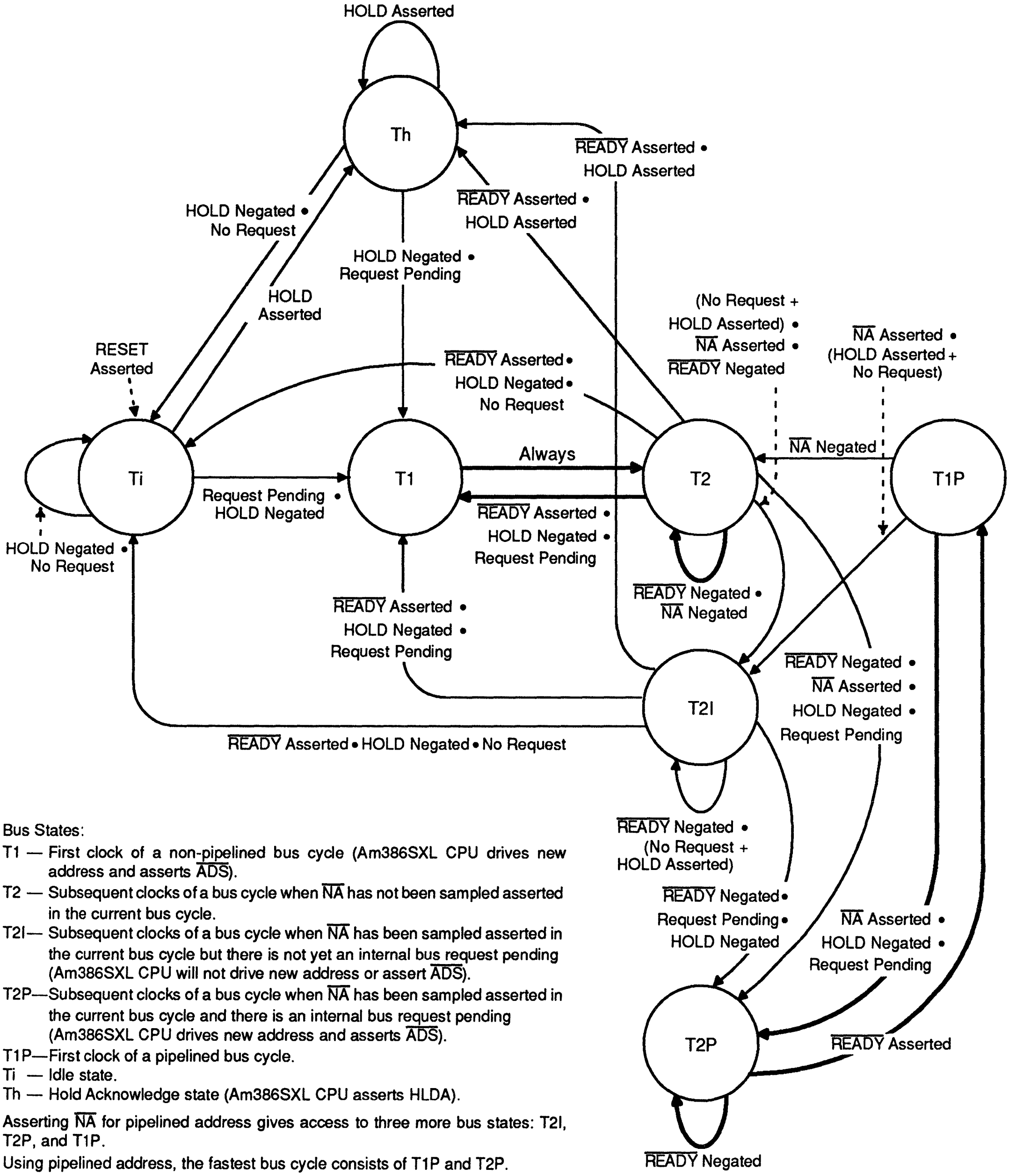

Cycle 1 Pipelined (Write)
Cycle 2 Pipelined (Read)
Cycle 3 Pipelined (Write)
Cycle 4 Pipelined (Read)
T1P
T2P
T2P
T1P
T2
T2P
T1P
T2I
T2P
T1P
CLK2
Processor CLK
BHE, BLE, A23–A1, M/IO, D/C
Valid 1
Valid 2
Valid 3
Valid 4
ADS is asserted as soon as the CPU has another bus cycle to perform, which is not always immediately after NA is asserted.
W/R
ADS
Note: ADS is asserted in **every** T2P state.
As long as the CPU enters the T2P state during Cycle 3, address pipelining is maintained in Cycle 4.
NA
Asserting NA more than once during any cycle has no additional effects.
NA could have been asserted in T1P if desired. Assertion now is the latest time possible to allow the CPU to enter T2P state to maintain pipelining in Cycle 3.
READY
LOCK
Valid 1
Valid 2
Valid 3
Valid 4
D15–D0
Out
Out
In
Out
15022A–020

Figure 32. Details of Address Pipelining During Cycles with Wait States


HOLD Asserted
Th
$\overline{\mathrm{READY}}$ Asserted • HOLD Asserted
HOLD Negated • No Request
$\overline{\mathrm{READY}}$ Asserted • HOLD Asserted
HOLD Negated • Request Pending
HOLD Asserted
(No Request + HOLD Asserted) • $\overline{\mathrm{NA}}$ Asserted • $\overline{\mathrm{READY}}$ Negated
$\overline{\mathrm{NA}}$ Asserted • (HOLD Asserted + No Request)
RESET Asserted
$\overline{\mathrm{READY}}$ Asserted • HOLD Negated • No Request
Always
$\overline{\mathrm{NA}}$ Negated
Ti
T1
T2
T1P
Request Pending • HOLD Negated
$\overline{\mathrm{READY}}$ Asserted • HOLD Negated • Request Pending
HOLD Negated • No Request
$\overline{\mathrm{READY}}$ Negated • $\overline{\mathrm{NA}}$ Negated
$\overline{\mathrm{READY}}$ Asserted • HOLD Negated • Request Pending
$\overline{\mathrm{READY}}$ Negated • $\overline{\mathrm{NA}}$ Asserted • HOLD Negated • Request Pending
T2I
$\overline{\mathrm{READY}}$ Asserted • HOLD Negated • No Request
$\overline{\mathrm{READY}}$ Negated • (No Request + HOLD Asserted)
$\overline{\mathrm{READY}}$ Negated • Request Pending • HOLD Negated
$\overline{\mathrm{NA}}$ Asserted • HOLD Negated • Request Pending
T2P
$\overline{\mathrm{READY}}$ Asserted
$\overline{\mathrm{READY}}$ Negated

Bus States:

T1 — First clock of a non-pipelined bus cycle (Am386SXL CPU drives new address and asserts $\overline{\mathrm{ADS}}$).

T2 — Subsequent clocks of a bus cycle when $\overline{\mathrm{NA}}$ has not been sampled asserted in the current bus cycle.

T2I— Subsequent clocks of a bus cycle when $\overline{\mathrm{NA}}$ has been sampled asserted in the current bus cycle but there is not yet an internal bus request pending (Am386SXL CPU will not drive new address or assert $\overline{\mathrm{ADS}}$).

T2P—Subsequent clocks of a bus cycle when $\overline{\mathrm{NA}}$ has been sampled asserted in the current bus cycle and there is an internal bus request pending (Am386SXL CPU drives new address and asserts $\overline{\mathrm{ADS}}$).

T1P—First clock of a pipelined bus cycle.

Ti — Idle state.

Th — Hold Acknowledge state (Am386SXL CPU asserts HLDA).

Asserting $\overline{\mathrm{NA}}$ for pipelined address gives access to three more bus states: T2I, T2P, and T1P.

Using pipelined address, the fastest bus cycle consists of T1P and T2P.

15022B–021

Figure 33. Complete Bus States (Including Pipelined Address)

Initiating and Maintaining Pipelined Address

Using the state diagram Figure 33, observe the transitions from an idle state (Ti) to the beginning of a pipelined bus cycle (T1P). From an idle state (Ti) the first bus cycle must begin with T1, and is therefore a non-pipelined bus cycle. The next bus cycle will be pipelined, however, provided $\overline{NA}$ is asserted and the first bus cycle ends in a T2P state (the address for the next bus cycle is driven during T2P). The fastest path from an idle state to a bus cycle with pipelined address is shown in bold below:

Ti, Ti, Ti, T1–T2–T2P, T1P–T2P,

Idle States	Non-Pipelined Cycle	Pipelined Cycle

T1-T2-T2P are the states of the bus cycle that establish address pipelining for the next bus cycle, which begins with T1P. The same is true after a bus hold state, shown below:

Th, Th, Th, T1–T2–T2P, T1P–T2P,

Hold Acknowledge States	Non-Pipelined Cycle	Pipelined Cycle

The transition to pipelined address is shown functionally by Figure 31, Cycle 1. Note that Cycle 1 is used to transition into pipelined address timing for the subsequent Cycles 2, 3, and 4, which are pipelined. The $\overline{NA}$ input is asserted at the appropriate time to select address pipelining for Cycle 2, 3, and 4.

Once a bus cycle is in progress and the current address has been valid for one entire bus state, the $\overline{NA}$ input is sampled at the end of every phase one until the bus cycle is acknowledged. Sampling begins in T2 during Cycle 1 in Figure 31. Once $\overline{NA}$ is sampled active during the current cycle, the Am386SXL microprocessor is free to drive a new address and bus cycle definition on the bus as early as the next bus state. In Figure 31, Cycle 1 for example, the next address is driven during state T2P. Thus, Cycle 1 makes the transition to pipelined address timing, since it begins with T1 but ends with T2P. Because the address for Cycle 2 is available before Cycle 2 begins, Cycle 2 is called a pipelined bus cycle, and it begins with T1P. Cycle 2 begins as soon as $\overline{READY}$ asserted terminates Cycle 1.

Examples of transition bus cycles are Figure 31, Cycle 1 and Figure 30, Cycle 2. Figure 31 shows transition during the very first cycle after an idle bus state, which is the fastest possible transition into address pipelining. Figure 30, Cycle 2 shows a transition cycle occurring during a burst of bus cycles. In any case, a transition cycle is the same whenever it occurs: it consists at least of T1, T2 ($\overline{NA}$ is asserted at that time), and T2P (provided the Am386SXL microprocessor has an internal bus request already pending, which it almost always has). T2P states are repeated if wait states are added to the cycle.

Note that only three states (T1, T2, and T2P) are required in a bus cycle performing a transition from non-pipelined address into pipelined address timing (e.g., Figure 31, Cycle 1). Figure 31, Cycles 2, 3, and 4 show that address pipelining can be maintained with two-state bus cycles consisting only of T1P and T2P.

Once a pipelined bus cycle is in progress, pipelined timing is maintained for the next cycle by asserting $\overline{NA}$ and detecting that the Am386SXL microprocessor enters T2P during the current bus cycle. The current bus cycle must end in state T2P for pipelining to be maintained in the next cycle. T2P is identified by the assertion of $\overline{ADS}$. Figures 30 and 31, however, each show pipelining ending after Cycle 4, because Cycle 4 ends in T2I. This indicates the Am386SXL CPU did not have an internal bus request prior to the acknowledgement of Cycle 4. If a cycle ends with a T2 or T2I, the next cycle will not be pipelined.

Realistically, address pipelining is almost always maintained as long as $\overline{NA}$ is sampled asserted. This is so because in the absence of any other request, a code prefetch request is always internally pending until the instruction decoder and code prefetch queue are completely full. Therefore, address pipelining is maintained for long bursts of bus cycles, if the bus is available (i.e., HOLD inactive), and $\overline{NA}$ is sampled active in each of the bus cycles.

Interrupt Acknowledge (INTA) Cycles

In response to an interrupt request on the INTR input when interrupts are enabled, the Am386SXL microprocessor performs two interrupt acknowledge cycles. These bus cycles are similar to read cycles in that bus definition signals define the type of bus activity taking place, and each cycle continues until acknowledged by $\overline{READY}$ sampled active.

The state of A2 distinguishes the first and second interrupt acknowledge cycles. The byte address driven during the first interrupt acknowledge cycle is 4 (A23–A3, A1, $\overline{BLE}$ Low, A2 and $\overline{BHE}$ High). The byte address driven during the second interrupt acknowledge cycle is 0 (A23–A1, $\overline{BLE}$ Low, and $\overline{BHE}$ High).

The $\overline{LOCK}$ output is asserted from the beginning of the first interrupt acknowledge cycle until the end of the second interrupt acknowledge cycle. Four idle bus states (Ti) are inserted by the Am386SXL microprocessor between the two interrupt acknowledge cycles for compatibility with spec TRHRL of the 8259A Interrupt Controller.

During both interrupt acknowledge cycles, D15–D0 float. No data is read at the end of the first interrupt acknowledge cycle. At the end of the second interrupt acknowledge cycle, the Am386SXL microprocessor will read an external interrupt vector from D7–D0 of the data bus. The vector indicates the specific interrupt number (from 0–255) requiring service.

Halt Indication Cycle

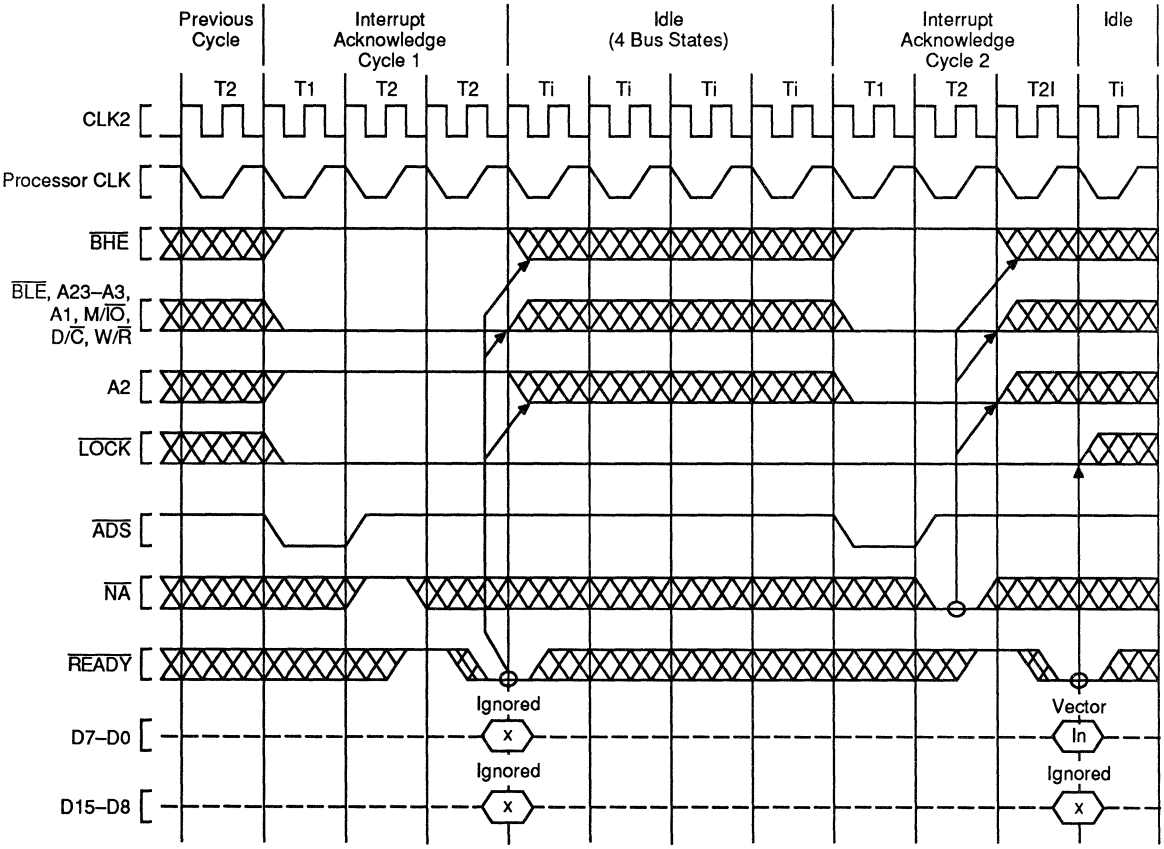
The execution unit halts as a result of executing a HLT instruction. Signaling its entrance into the halt state, a halt indication cycle is performed. The halt indication cycle is identified by the state of the bus definition signals shown on page 39, Bus Cycle Definition Signals, and an address of 2. The halt indication cycle must be acknowledged by $\overline{\text{READY}}$ asserted. A halted Am386SXL CPU resumes execution when INTR (if interrupts are enabled), NMI, or RESET is asserted.

Shutdown Indication Cycle

The Am386SXL microprocessor shuts down as a result of a protection fault while attempting to process a double fault. Signaling its entrance into the shutdown state, a shutdown indication cycle is performed. The shutdown indication cycle is identified by the state of the bus definition signals shown in section Bus Cycle Definition Signals and an address of 0. The shutdown indication cycle must be acknowledged by $\overline{\text{READY}}$ asserted. A shut-down Am386SXL microprocessor resumes execution when NMI or RESET is asserted.

Entering and Exiting Hold Acknowledge

The Bus Hold Acknowledge state (Th) is entered in response to the HOLD input being asserted. In the Bus Hold Acknowledge state, the Am386SXL microprocessor floats all outputs or bi-directional signals, except for HLDA. HLDA is asserted as long as the Am386SXL CPU remains in the Bus Hold Acknowledge state. In the Bus Hold Acknowledge state, all inputs except HOLD, $\overline{\text{FLT}}$, and RESET are ignored.



Interrupt Vector (0–255) is read on D7–D0 at end of second Interrupt Acknowledge bus cycle. Because each Interrupt Acknow-ledge bus cycle is followed by idle bus states, asserting $\overline{\text{NA}}$ has no practical effect. Choose the approach which is simplest for your system hardware design.

15022B–022

Figure 34. Interrupt Acknowledge Cycles

Th may be entered from a bus idle state, as in Figure 37, or after the acknowledgement of the current physical bus cycle, if the LOCK signal is not asserted, as in Figures 38 and 39.

Th is exited in response to the HOLD input being negated. The following state will be Ti if no bus request is pending, as in Figure 37. The following bus state will be T1 if a bus request is internally pending, as in Figures 38 and 39. Th is exited in response to RESET being asserted.

If a rising edge occurs on the edge-triggered NMI input while in Th, the event is remembered as a non-maskable interrupt 2 and is serviced when Th is exited, unless the Am386SXL microprocessor is reset before Th is exited.

Reset During Hold Acknowledge

RESET being asserted takes priority over HOLD being asserted. If RESET is asserted while HOLD remains asserted, the Am386SXL microprocessor drives its pins to defined states during reset, as in Table 16 (Pin State During Reset), and performs internal reset activity as usual.

If HOLD remains asserted when RESET is inactive, the Am386SXL CPU enters the Hold Acknowledge state before performing its first bus cycle, provided HOLD is still asserted when the Am386SXL microprocessor would otherwise perform its first bus cycle.

FLOAT

Activating the FLT input floats all Am386SXL microprocessor bi-directional and output signals, including HLDA. Asserting FLT isolates the Am386SXL microprocessor from the surrounding circuitry.

As the Am386SXL microprocessor is packaged in a surface mount PQFP, it cannot be removed from the motherboard when In-Circuit Emulation (ICE) is needed. The FLT input allows the Am386SXL CPU to be electrically isolated from the surrounding circuitry. This allows connection of an emulator to the Am386SXL microprocessor PQFP without removing it from the PCB. This method of emulation is referred to as ON-Circuit Emulation (ONCE).

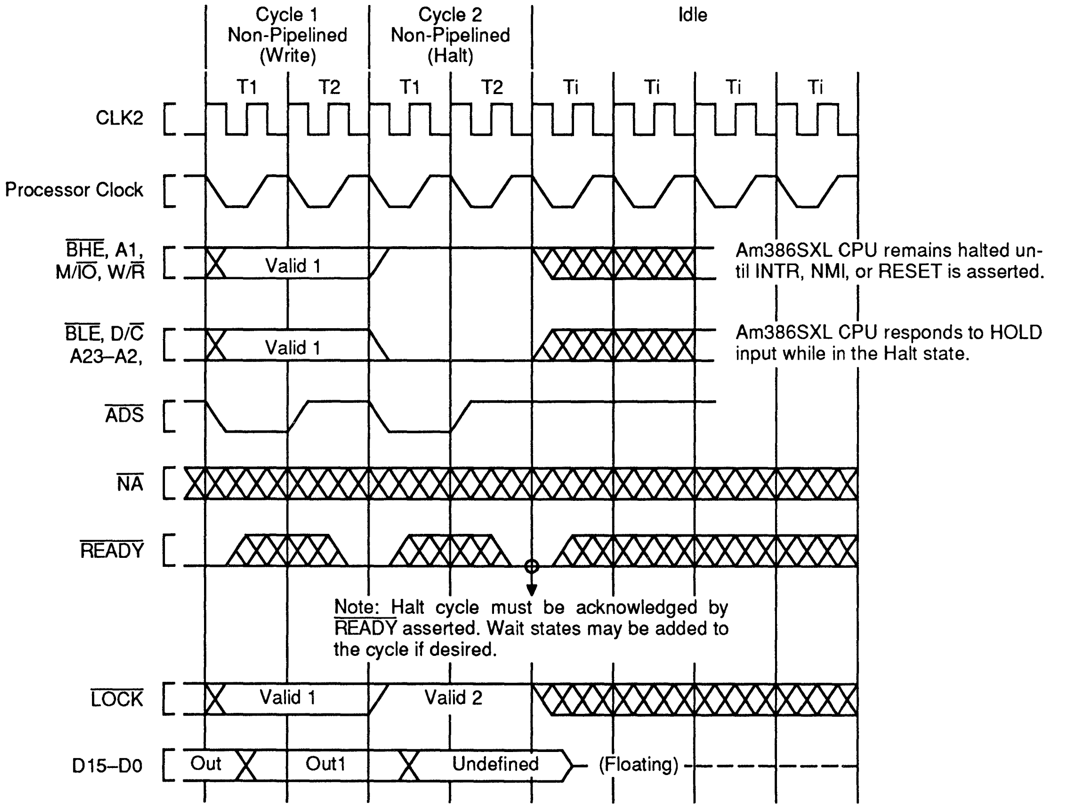

Cycle 1
Non-Pipelined
(Write)
Cycle 2
Non-Pipelined
(Halt)
Idle
T1
T2
T1
T2
Ti
Ti
Ti
Ti
CLK2
Processor Clock
BHE, A1,
M/IO, W/R
Valid 1
Am386SXL CPU remains halted until INTR, NMI, or RESET is asserted.
BLE, D/C
A23–A2,
Valid 1
Am386SXL CPU responds to HOLD input while in the Halt state.
ADS
NA
READY
Note: Halt cycle must be acknowledged by READY asserted. Wait states may be added to the cycle if desired.
LOCK
Valid 1
Valid 2
D15–D0
Out
Out1
Undefined
(Floating)
15022B–023

Figure 35. Example Halt Indication Cycle from Non-Pipelined Cycle

Entering and Exiting FLOAT

FLT is an asynchronous, active Low input. It is recognized on the rising edge of CLK2. When recognized, it aborts the current bus cycle and floats the outputs of the Am386SXL microprocessor (Figure 41). FLT must be held Low for a minimum of 16-CLK2 cycles. Reset should be asserted and held asserted until after FLT is deasserted. This will ensure that the Am386SXL CPU will exit FLOAT in a valid state.

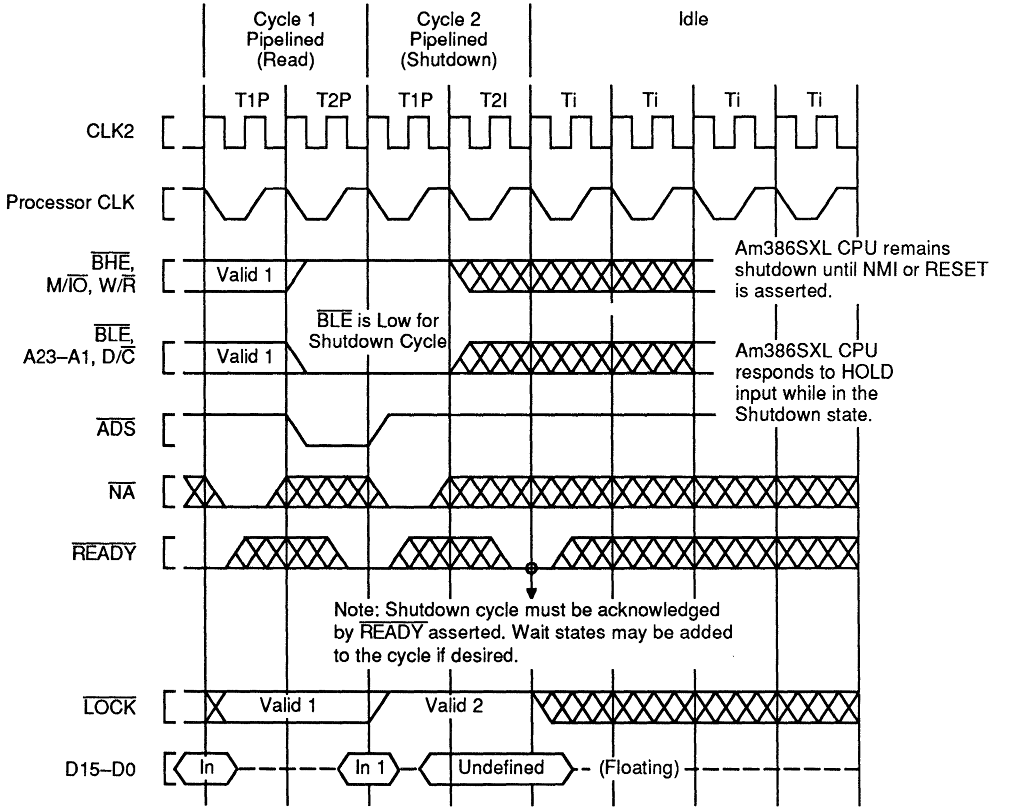
Asserting the FLT input unconditionally aborts the current bus cycle and forces the Am386SXL CPU into the FLOAT mode. Since activating FLT unconditionally forces the Am386SXL CPU into FLOAT mode, the Am386SXL microprocessor is not guaranteed to enter FLOAT in a valid state. After deactivating FLT, the Am386SXL CPU is not guaranteed to exit FLOAT mode in a valid state. This is not a problem, as the FLT pin is meant to be used only during ONCE. After exiting FLOAT, the Am386SXL microprocessor must be reset to return it to a valid state. Reset should be asserted before FLT is deasserted. This will ensure that the Am386SXL CPU will exit FLOAT in a valid state.

FLT has an internal pull-up resistor, and if it is not used it should be unconnected.

Bus Activity During and Following Reset

RESET is the highest priority input signal, capable of interrupting any processor activity when it is asserted. A bus cycle in progress can be aborted at any stage, or idle states and Bus Hold Acknowledge states discontinued, so that the reset state is established.

RESET should remain asserted for at least 15-CLK2 periods to ensure it is recognized throughout the Am386SXL microprocessor, and at least 80-CLK2 periods if self-test is going to be requested at the falling edge. RESET asserted pulses less than 15-CLK2 periods may not be recognized. RESET pulses less than 80-CLK2 periods followed by a self-test may cause the self-test to report a failure when no true failure exists.


Cycle 1 Pipelined (Read)
Cycle 2 Pipelined (Shutdown)
Idle
T1P
T2P
T1P
T2I
Ti
Ti
Ti
Ti
CLK2
Processor CLK
BHE, M/IO, W/R
Valid 1
Am386SXL CPU remains shutdown until NMI or RESET is asserted.
BLE, A23–A1, D/C
Valid 1
BLE is Low for Shutdown Cycle
Am386SXL CPU responds to HOLD input while in the Shutdown state.
ADS
NA
READY
Note: Shutdown cycle must be acknowledged by READY asserted. Wait states may be added to the cycle if desired.
LOCK
Valid 1
Valid 2
D15–D0
In
In 1
Undefined
(Floating)

15022B–024

Figure 36. Example Shutdown Indication Cycle from Non-Pipelined Cycle

Provided the RESET falling edge meets setup and hold times (t25 and t26), the internal processor clock phase is defined at that time as illustrated by Figure 40 and Figure 48.

A self-test may be requested at the time RESET goes inactive by having the BUSY input at a Low level, as shown in Figure 40. The self-test requires approximately (2^{20} + 60) CLK2 periods to complete. The self-test duration is not affected by the test results. Even if the self-test indicates a problem, the Am386SXL microprocessor attempts to proceed with the reset sequence afterwards.

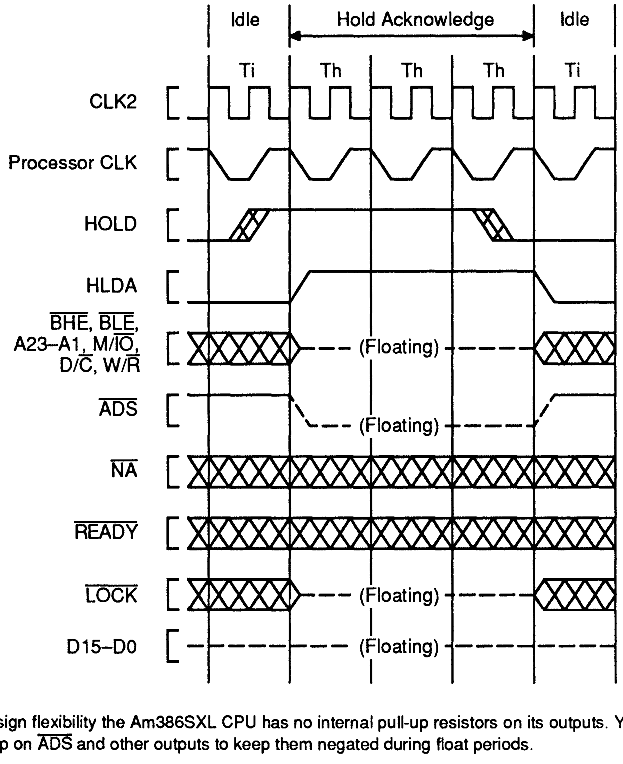
After the RESET falling edge (and after the self-test if it was requested), the Am386SXL microprocessor performs an internal initialization sequence for approximately 350- to 450-CLK2 periods.

Self-Test Signature

Upon completion of self-test (if self-test was requested by driving BUSY Low at the falling edge of RESET) the EAX register will contain a signature of 00000000H, indicating the Am386SXL microprocessor passed its self-test of microcode and major PLA contents with no problems detected. The passing signature in EAX, 00000000H, applies to all revision levels. Any non-zero signature indicates the unit is faulty.

Component and Revision Identifiers

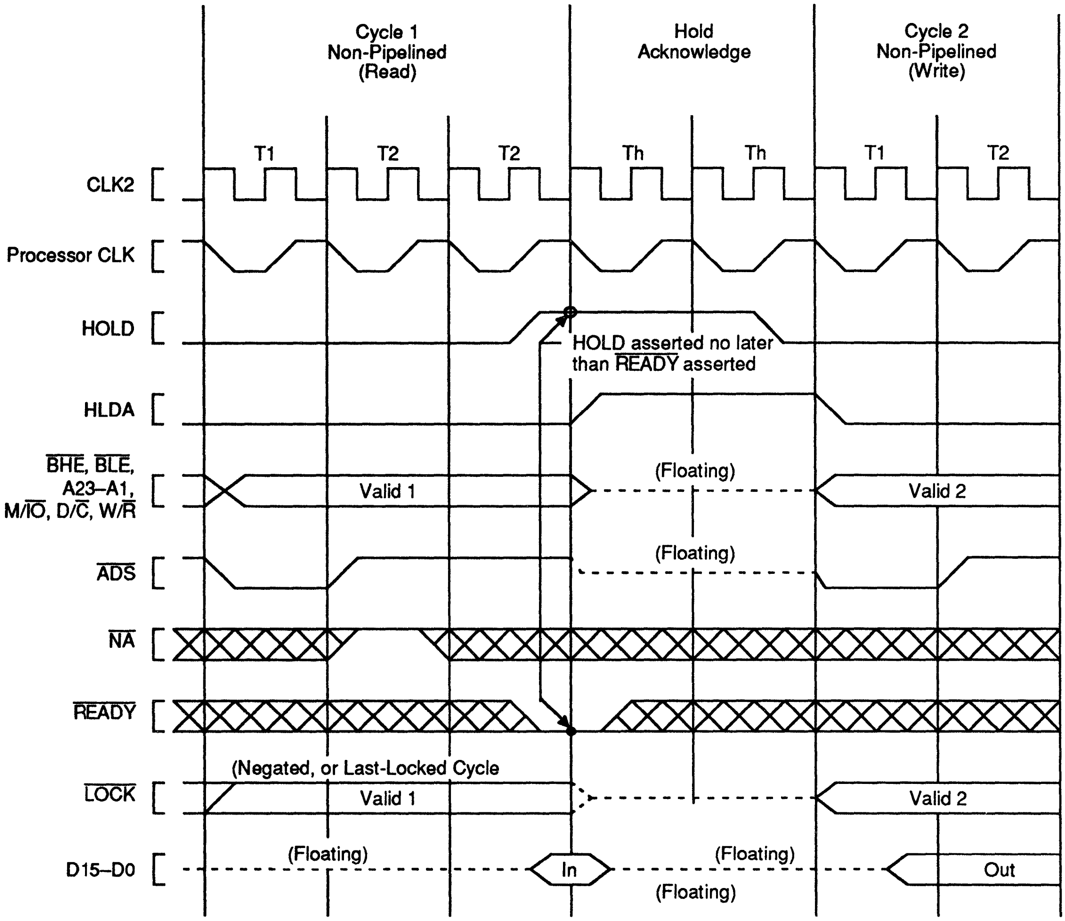
To assist users the Am386SXL microprocessor, after reset, holds a component identifier and revision identifier in its DX register. The upper 8 bits of DX hold 23H as identification of the Am386SXL CPU (the lower nibble, 03H, refers to the Am386DX microprocessor architecture. The upper nibble, 02H, refers to the second member of the Am386DX microprocessor Family). The lower 8 bits of DX hold an 8-bit unsigned binary number related to the component revision level. The revision identifier will, in general, chronologically track those component steppings which are intended to have certain improvements or distinction from previous steppings. The Am386SXL microprocessor revision identifier will track that of the Am386DX CPU where possible.



Note: For maximum design flexibility the Am386SXL CPU has no internal pull-up resistors on its outputs. Your design may require an external pull-up on ADS and other outputs to keep them negated during float periods.

15022B–025

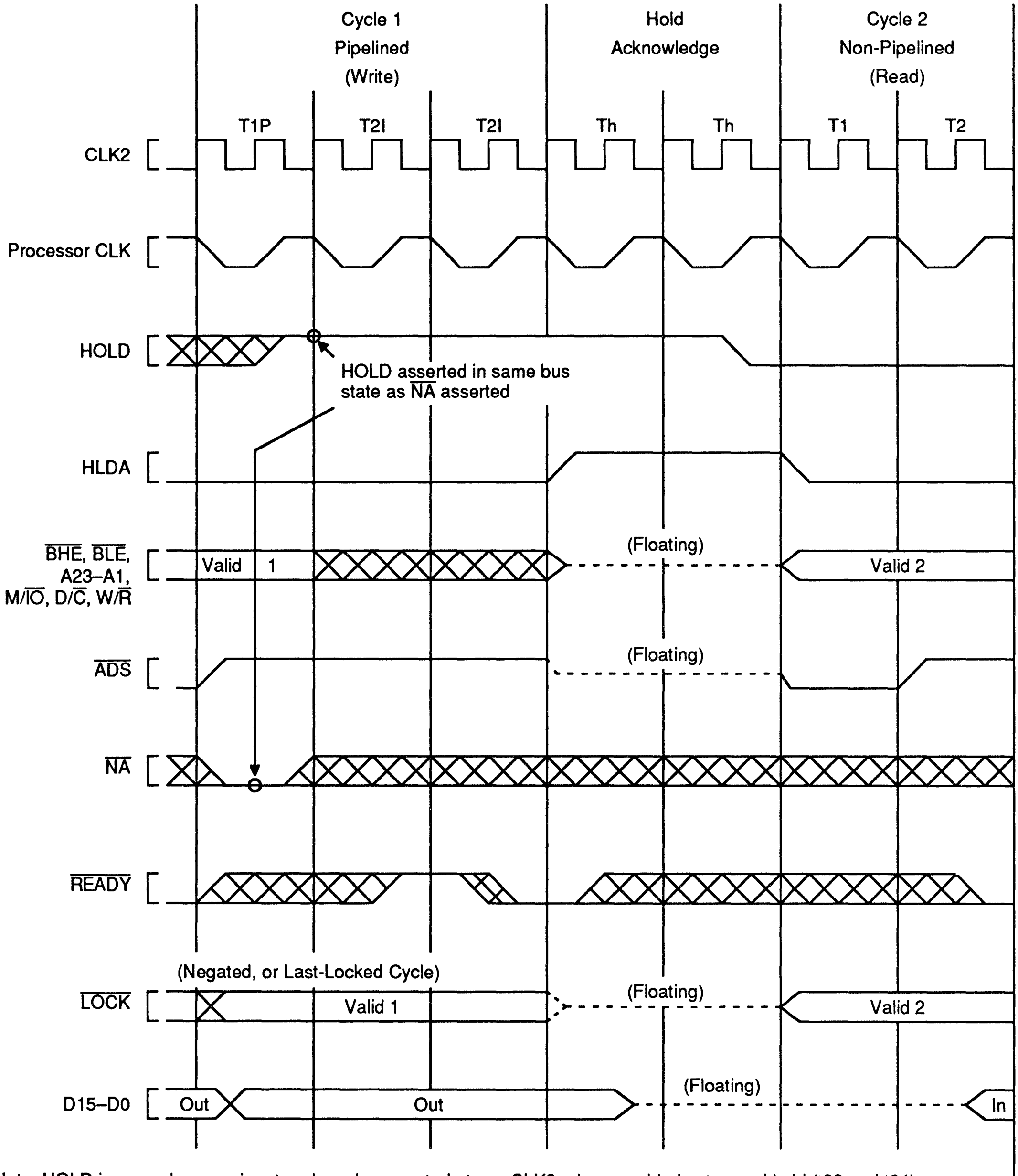
Figure 37. Requesting Hold from Idle Bus



Note: HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t23 and t24) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

15022B–026

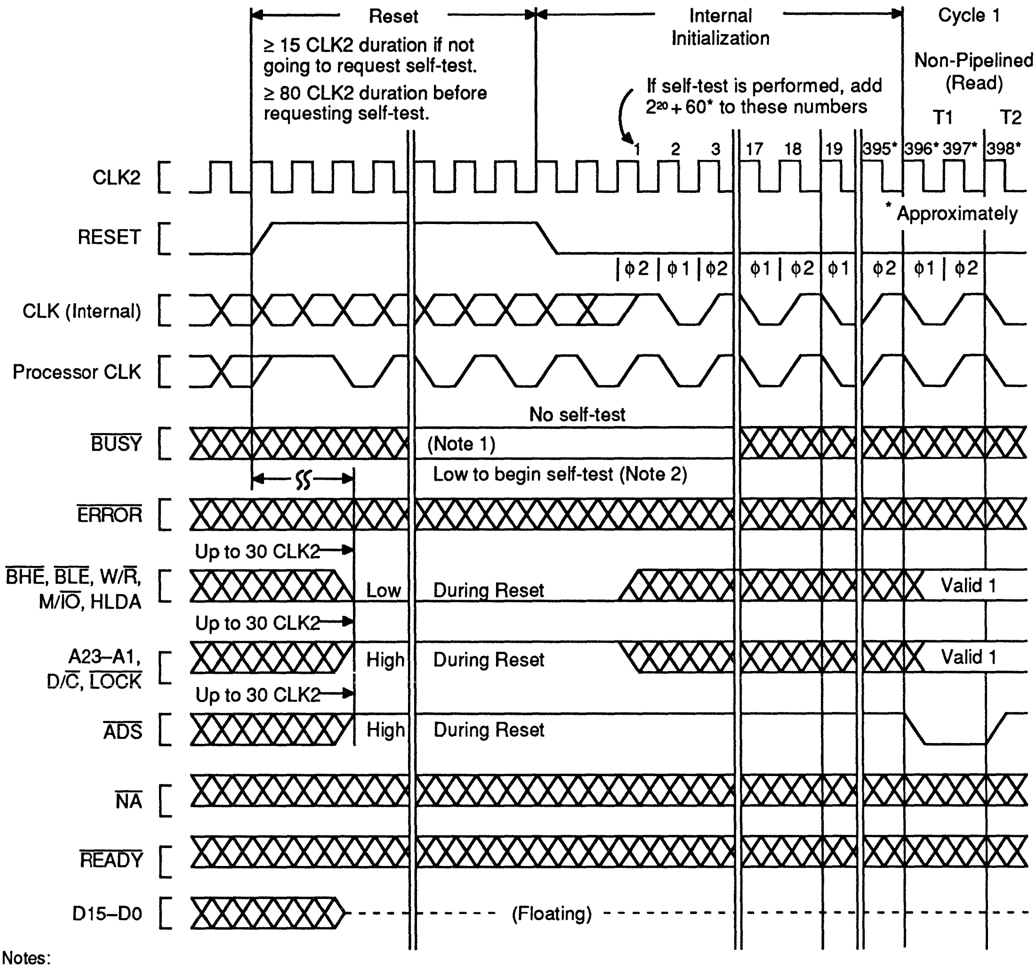
Figure 38. Requesting Hold from Active Bus ($\overline{\text{NA}}$ Inactive)



Note: HOLD is a synchronous input and can be asserted at any CLK2 edge, provided setup and hold (t23 and t24) requirements are met. This waveform is useful for determining Hold Acknowledge latency.

15022B–027

Figure 39. Requesting Hold from Idle Bus ($\overline{NA}$ Active)

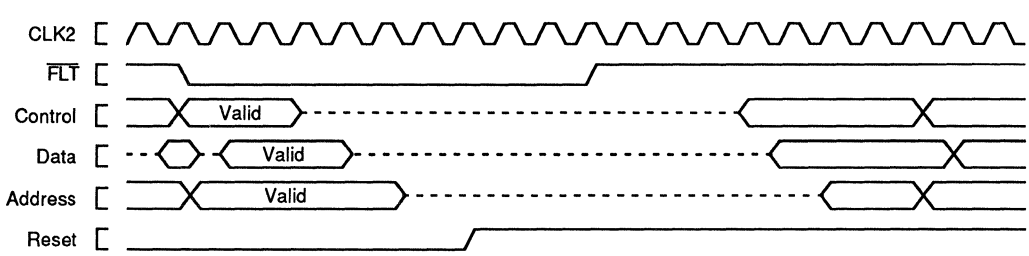


Notes:

1. BUSY should be held stable for 8-CLK2 periods before and after the CLK2 period in which RESET falling edge occurs.
2. If self-test is requested, the outputs remain in their reset state as shown here.

15022B–028

Figure 40. Bus Activity from Reset Until First Code Fetch



15022B–029

Figure 41. Entering and Exiting FLT

The revision identifier is intended to assist users to a practical extent. However, the revision identifier value is not guaranteed to change with every stepping revision, or to follow a completely uniform numerical sequence, depending on the type or intention of revision, or manufacturing materials required to be changed.

Table 18. Component and Revision Identifier History

Intel i386SX Stepping	Am386SXL Microprocessor Revision	Revision Identifier
B	A1	05H
C	B	08H

Coprocessor Interfacing

The Am386SXL microprocessor provides an automatic interface for a 387SX math coprocessor. A 387SX math coprocessor uses an I/O mapped interface driven automatically by the Am386SXL CPU and assisted by three dedicated signals: $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$, and PEREQ.

As the Am386SXL microprocessor begins supporting a math coprocessor instruction, it tests the $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ signals to determine if the coprocessor can accept its next instruction. Thus, the $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ inputs eliminate the need for any preamble bus cycles for communication between processor and math coprocessor. A 387SX math coprocessor can be given its command op-code immediately. The dedicated signals provide instruction synchronization and eliminate the need of using the WAIT op-code (9BH) for 387SX math coprocessor instruction synchronization (the WAIT op-code was required when the 8086 or 8088 was used with the 8087 math coprocessor).

Custom math coprocessors can be included in Am386SXL microprocessor based systems by memory-mapped or I/O-mapped interfaces. Such math coprocessor interfaces allow a completely custom protocol, and are not limited to a set of math coprocessor protocol primitives. Instead, memory-mapped or I/O-mapped interfaces may use all applicable instructions for high-speed math coprocessor communication. The $\overline{\text{BUSY}}$ and $\overline{\text{ERROR}}$ inputs of the Am386SXL microprocessor may also be used for the custom math coprocessor interface, if such hardware assist is desired. These signals can be tested by the WAIT op-code (9BH). The WAIT instruction will wait until the $\overline{\text{BUSY}}$ input is inactive (interruptable by an NMI or enabled INTR input), but generates an Exception 16 fault if the $\overline{\text{ERROR}}$ pin is active when the $\overline{\text{BUSY}}$ goes (or is) inactive. If the custom math coprocessor interface is memory-mapped, protection of the addresses used for the interface can be provided with the Am386SXL CPU's on-chip paging or segmentation mechanisms. If the custom interface is I/O-mapped, protection of the interface can be provided with the IOPL (I/O Privilege Level) mechanism.

A 387SX math coprocessor interface is I/O mapped as shown in Table 19. Note that a 387SX math coprocessor interface addresses are beyond the 0H–0FFFFH range for programmed I/O. When the Am386SXL microprocessor supports the 387SX math coprocessor, the Am386SXL CPU automatically generates bus cycles to the coprocessor interface addresses.

Table 19. Math Coprocessor Port Address

Address in Am386SXL CPU I/O Space	387SX-Compatible Math Coprocessor Register
8000F8H	Op-code Register
8000FCH/8000FEH*	Operand Register

*Generated as 2nd bus cycle during Dword transfer.

To correctly map a 387SX math coprocessor registers to the appropriate I/O addresses, connect the CMD0 and CMD1 lines of a 387SX math coprocessor, as listed in Table 20.

Table 20. Connections for CMD0 and CMD1 Inputs for a 387SX

Signal	Connection
CMD0	Connected directly to 386SXL CPU A2 signal.
CMD1	Connect to ground.

Software Testing for Math Coprocessor Presence

When software is used to test for math coprocessor (387SX) presence, it should use only the following math coprocessor op-codes: FINIT, FNINIT, FSTCW mem, FSTSW mem, and FSTSW AX. To use other math coprocessor op-codes when a math coprocessor is known to be not present, first set EM = 1 in the Am386SXL CPU's CR0 register.

PACKAGE THERMAL SPECIFICATIONS

The Am386SXL microprocessor is specified for operation when case temperature is within the range of 0°C–100°C. The case temperature may be measured in any environment to determine whether the Am386SXL CPU is within specified operating range. The case temperature should be measured at the center of the top surface opposite the pins.

The ambient temperature is guaranteed as long as Tc is not violated. The ambient temperature can be calculated from the θjc and θja from the following equations:

$$Tj = Tc + P \bullet \theta jc$$
$$Ta = Tj - P \bullet \theta ja$$
$$Tc = Ta + P \bullet [\theta ja - \theta jc]$$

ELECTRICAL SPECIFICATIONS

The following sections describe recommended electrical connections for the Am386SXL microprocessor, and its electrical specifications.

Power and Grounding

The Am386SXL CPU has modest power requirements. However, its high clock frequency and 47 output buffers (address, data, control, and HLDA) can cause power surges as multiple output buffers drive new signal levels simultaneously. For clean on-chip power distribution at high frequency, 14 V_{cc} and 18 V_{ss} pins separately feed functional units of the Am386SXL microprocessor.

Power and ground connections must be made to all external V_{cc} and V_{ss} pins of the Am386SXL microprocessor. On the circuit board, all V_{cc} pins should be connected on a V_{cc} plane, and V_{ss} pins should be connected on a GND plane.

Power Decoupling Recommendations

Liberal decoupling capacitors should be placed near the Am386SXL microprocessor. The Am386SXL CPU driving its 24-bit address bus and 16-bit data bus at high frequencies can cause transient power surges, particularly when driving large capacitive loads. Low inductance capacitors and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by shortening circuit board traces between the Am386SXL microprocessor and decoupling capacitors as much as possible.

Resistor Recommendations

The $\overline{\text{ERROR}}$, $\overline{\text{FLT}}$, and $\overline{\text{BUSY}}$ inputs have internal pull-up resistors of approximately 20 Kohms, and the PEREQ input has an internal pull-down resistor of approximately 20 Kohms, built into the Am386SXL microprocessor to keep these signals inactive when a 387SX-compatible math coprocessor is not present in the system (or temporarily removed from its socket).

In typical designs, the external pull-up resistors shown in Table 21 are recommended. However, a particular design may have reason to adjust the resistor values recommended here, or alter the use of pull-up resistors in other ways.

Other Connection Recommendations

For reliable operation, always connect unused inputs to an appropriate signal level. NC pins should always remain unconnected. Connection of NC pins to V_{cc} or V_{ss} will result in component malfunction or incompatibility with future steppings of the Am386SXL CPU.

Particularly when not using the interrupts or bus hold (as when first prototyping), prevent any chance of spurious activity by connecting these associated inputs to GND.

Pin	Signal
40	INTR
38	NMI
4	HOLD

If not using address pipelining, connect pin 6 ($\overline{\text{NA}}$) through a pull-up in the range of 20 Kohms to V_{cc}.

Table 21. Recommended Resistor Pull-Ups to V_{cc}

Pin	Signal	Pull-Up Value	Purpose
16	$\overline{\text{ADS}}$	20 Kohms ±10%	Lightly pull $\overline{\text{ADS}}$ inactive during Am386SXL CPU Hold Acknowledge states.
26	$\overline{\text{LOCK}}$	20 Kohms ±10%	Lightly pull $\overline{\text{LOCK}}$ inactive during Am386SXL CPU Hold Acknowledge states.

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature under bias . . . . . – 65 to 125°C
Storage Temperature . . . . – 65 to 150°C

Stresses above those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods of time may affect device reliability.

OPERATING RANGES

Supply Voltage with respect to V_{ss} . . . . –0.5 V to 7 V
Voltage on other pins . . . . –0.5 V to (V_{cc} + 0.5)V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating ranges

V_{CC} = 5 V ±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Min	Max	Unit
V_{IL}	Input Low Voltage		–0.3	+0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}+ 0.3	V
V_{ILC}	CLK2 Input Low Voltage		–0.3	+0.8	V
V_{IHC}	CLK2 Input High Voltage		V_{CC} – 0.8	V_{CC}+ 0.3	V
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A23–A1, D15–D0 I_{OL} = 5 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA			 0.45 0.45	 V V
V_{OH}	Output High Voltage I_{OH} = 1.0 mA: A23–A1, D15–D0 I_{OH} = 0.2 mA: A23–A1, D15–D0 I_{OH} = 0.9 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA I_{OH} = 0.18 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA		 2.4 V_{CC} – 0.5 2.4 V_{CC} – 0.5		 V V V
I_{LI}	Input Leakage Current (for all pins except PEREQ, $\overline{BUSY}$, $\overline{FLT}$, and $\overline{ERROR}$)	0 V ≤ V_{IN} ≤ V_{CC}		±15	μA
I_{IH}	Input Leakage Current (PEREQ pin)	V_{IH} = 2.4 V (Note 1)		200	μA
I_{IL}	Input Leakage Current ($\overline{BUSY}$, $\overline{ERROR}$, and $\overline{FLT}$ pins)	V_{IL} = 0.45 V (Note 2)		–400	μA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		±15	μA
I_{CC}	Supply Current CLK2 = 32 MHz: with –16* CLK2 = 40 MHz: with –20 CLK2 = 50 MHz: with –25	(Note 4) I_{CC} Typ = 135 mA (Note 3) I_{CC} Typ = 165 mA (Note 3) I_{CC} Typ = 210 mA (Note 3)		 160 200 250	 mA mA mA
I_{CCSB}	Standby Current	I_{CCSB} Typ = 0.02 mA (Note 5)		0.15	mA
C_{IN}	Input Capacitance	F_C = 1 MHz (Note 4)		10	pF
C_{OUT}	Output or I/O Capacitance	F_C = 1 MHz (Note 4)		12	pF
C_{CLK}	CLK2 Capacitance	F_C = 1 MHz (Note 4)		20	pF

Notes: Tested at the minimum operating frequency of the part.
*Contact AMD for 16-MHz availability.

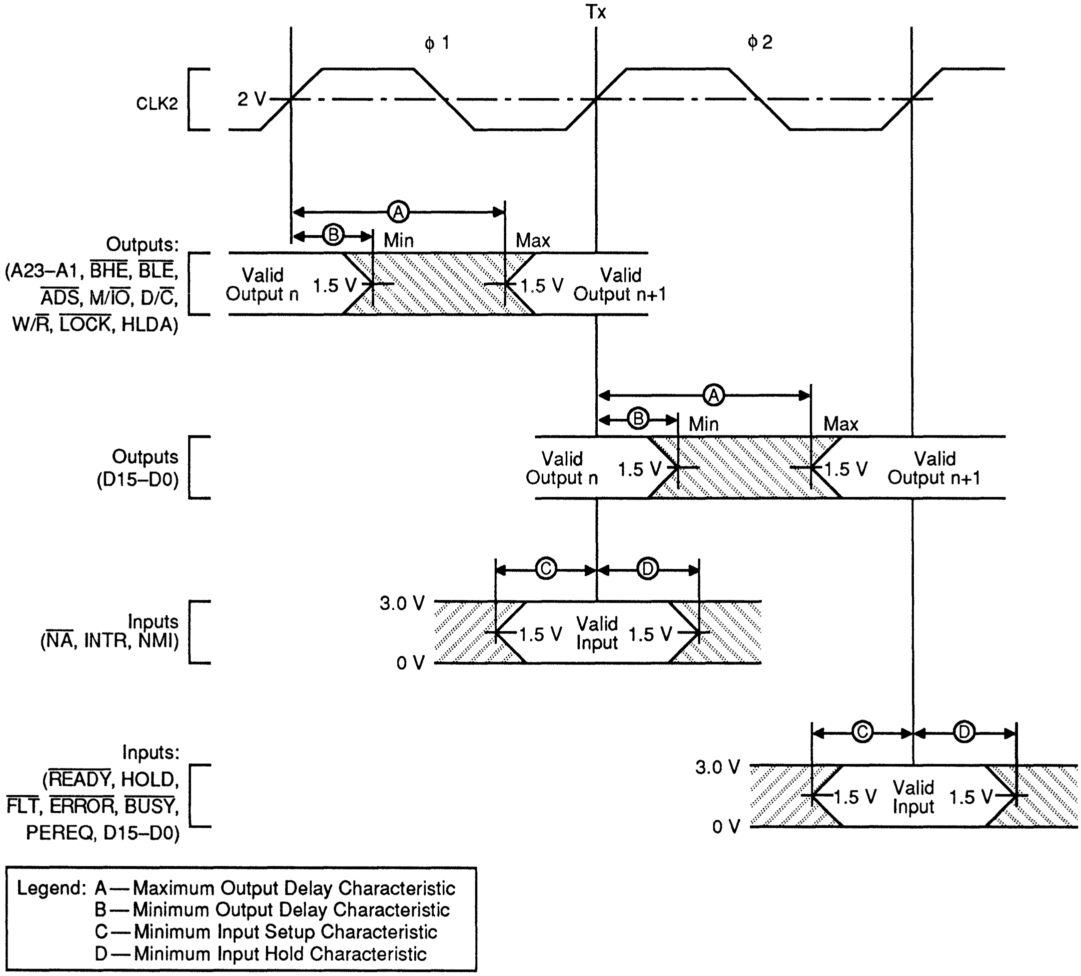
1. PEREQ input has an internal pull-down resistor.
2. $\overline{BUSY}$, $\overline{FLT}$, and $\overline{ERROR}$ inputs each have an internal pull-up resistor.
3. I_{CC} Max measurement at worst case frequency, V_{CC}, and temperature, outputs unloaded.
4. Not 100% tested.
5. Inputs at rails, outputs unloaded, PEREQ Low, $\overline{ERROR}$ High, $\overline{BUSY}$ High, and $\overline{FLT}$ High.

SWITCHING CHARACTERISTICS

The switching characteristics given consist of output delays, input setup requirements, and input hold requirements. All switching characteristics are relative to the CLK2 rising edge crossing the 2.0 V level.

Switching characteristic measurement is defined by Figure 42. Inputs must be driven to the voltage levels indicated by Figure 42 when switching characteristics are measured. Output delays are specified with minimum and maximum limits measured, as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs $\overline{\text{ADS}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, A23–A1, and HLDA only change at the beginning of phase one. D15–D0 (write cycles) only change at the beginning of phase two. The $\overline{\text{READY}}$, HOLD, $\overline{\text{BUSY}}$, $\overline{\text{ERROR}}$, PEREQ, $\overline{\text{FLT}}$, and D15–D0 (read cycles) inputs are sampled at the beginning of phase one. The $\overline{\text{NA}}$, INTR, and NMI inputs are sampled at the beginning of phase two.


Tx
ϕ 1
ϕ 2
CLK2
2 V
Outputs:
(A23–A1, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$,
$\overline{\text{ADS}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$,
W/$\overline{\text{R}}$, $\overline{\text{LOCK}}$, HLDA)
A
B
Min
Max
Valid Output n
1.5 V
1.5 V
Valid Output n+1
Outputs
(D15–D0)
A
B
Min
Max
Valid Output n
1.5 V
1.5 V
Valid Output n+1
Inputs
($\overline{\text{NA}}$, INTR, NMI)
C
D
3.0 V
0 V
1.5 V
Valid Input
1.5 V
Inputs:
($\overline{\text{READY}}$, HOLD,
$\overline{\text{FLT}}$, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$,
PEREQ, D15–D0)
C
D
3.0 V
0 V
1.5 V
Valid Input
1.5 V

Legend: A — Maximum Output Delay Characteristic
B — Minimum Output Delay Characteristic
C — Minimum Input Setup Characteristic
D — Minimum Input Hold Characteristic

15022B–030

Figure 42. Drive Levels and Measurement Points for Switching Characteristics

SWITCHING CHARACTERISTICS

Switching Characteristics at 25 MHz: V_{CC} = 5 V ±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Ref. Figure	Min	Max	Unit
	Operating Frequency	Half CLK2 freq.		0	25	MHz
1	CLK2 Period		43	20		ns
2a	CLK2 High Time	at 2 V	43	7		ns
2b	CLK2 High Time	at (V_{CC}−0.8 V)	43	4		ns
3a	CLK2 Low Time	at 2 V	43	7		ns
3b	CLK2 Low Time	at 0.8 V	43	5		ns
4	CLK2 Fall Time	(V_{CC}−0.8 V) to 0.8 V (Note 3)	43		7	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}−0.8 V) (Note 3)	43		7	ns
6	A23–A1 Valid Delay	C_L = 50 pF	46	4	17	ns
7	A23–A1 Float Delay	(Note 1)	50	4	30	ns
8	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Valid Delay	C_L = 50 pF	46	4	17	ns
9	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Float Delay	(Note 1)	50	4	30	ns
10	M/$\overline{IO}$, D/$\overline{C}$, W/$\overline{R}$, $\overline{ADS}$ Valid Delay	C_L = 50 pF	46	4	17	ns
11	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Float Delay	(Note 1)	50	4	30	ns
12	D15–D0 Write Data Valid Delay	C_L = 50 pF	46	7	23	ns
12a	D15–D0 Write Data Hold Time	C_L = 50 pF		2		ns
13	D15–D0 Write Data Float Delay	(Note 1)	47	4	22	ns
14	HLDA Valid Delay	C_L = 50 pF	46	4	22	ns
15	$\overline{NA}$ Setup Time		45	5		ns
16	$\overline{NA}$ Hold Time		45	3		ns
19	$\overline{READY}$ Setup Time		45	9		ns
20	$\overline{READY}$ Hold Time		45	4		ns
21	D15–D0 Read Data Setup Time		45	7		ns
22	D15–D0 Read Data Hold Time		45	5		ns
23	HOLD Setup Time		45	9		ns
24	HOLD Hold Time		45	3		ns
25	RESET Setup Time		51	8		ns
26	RESET Hold Time		51	3		ns
27	NMI, INTR Setup Time	(Note 2)	45	6		ns
28	NMI, INTR Hold Time	(Note 2)	45	6		ns
29	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Setup Time	(Note 2)	45	6		ns
30	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Hold Time	(Note 2)	45	5		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

3. These are not tested. They are guaranteed by design characterization.

SWITCHING CHARACTERISTICS (continued)

Switching Characteristics at 20 MHz: V_{CC} = 5 V±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Ref. Figure	Min	Max	Unit
	Operating Frequency	Half CLK2 freq.		0	20	MHz
1	CLK2 Period		43	25		ns
2a	CLK2 High Time	at 2 V	43	8		ns
2b	CLK2 High Time	at (V_{CC}−0.8 V)	43	5		ns
3a	CLK2 Low Time	at 2 V	43	8		ns
3b	CLK2 Low Time	at 0.8 V	43	6		ns
4	CLK2 Fall Time	(V_{CC}−0.8 V) to 0.8 V (Note 3)	43		8	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}−0.8 V) (Note 3)	43		8	ns
6	A23–A1 Valid Delay	C_L = 120 pF (Note 4)	46	4	30	ns
7	A23–A1 Float Delay	(Note 1)	50	4	32	ns
8	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 75 pF (Note 4)	46	4	30	ns
9	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	50	4	32	ns
10a	M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$ Valid Delay	C_L = 75 pF (Note 4)	46	4	28	ns
10b	W/$\overline{\text{R}}$, $\overline{\text{ADS}}$ Valid Delay	C_L = 75 pF (Note 4)	46	4	26	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	50	6	30	ns
12	D15–D0 Write Data Valid Delay	C_L = 120 p (Note 4)	46	4	38	ns
13	D15–D0 Write Data Float Delay	(Note 1)	50	4	27	ns
14	HLDA Valid Delay	C_L = 75 pF (Note 4)	46	4	28	ns
15	$\overline{\text{NA}}$ Setup Time		45	5		ns
16	$\overline{\text{NA}}$ Hold Time		45	12		ns
19	$\overline{\text{READY}}$ Setup Time		45	12		ns
20	$\overline{\text{READY}}$ Hold Time		45	4		ns
21	D15–D0 Read Data Setup Time		45	9		ns
22	D15–D0 Read Data Hold Time		45	6		ns
23	HOLD Setup Time		45	17		ns
24	HOLD Hold Time		45	5		ns
25	RESET Setup Time		51	12		ns
26	RESET Hold Time		51	4		ns
27	NMI, INTR Setup Time	(Note 2)	45	16		ns
28	NMI, INTR Hold Time	(Note 2)	45	16		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Setup Time	(Note 2)	45	14		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Hold Time	(Note 2)	45	5		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

3. These are not tested. They are guaranteed by design characterization.

4. Tested with C_L set at 50 pF and derated to support the indicated distributed capacitive load. See Figures 52 and 53 for the capacitive derating curve.

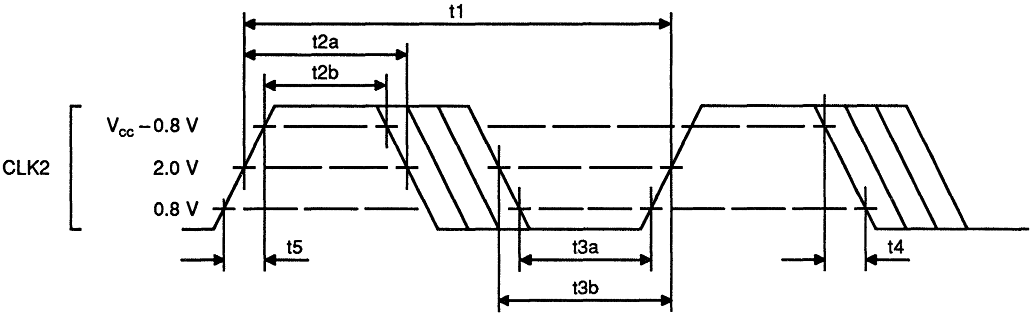
SWITCHING CHARACTERISTICS (continued)

Switching Characteristics at 16 MHz*: V_{CC} = 5 V ±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Ref. Figure	Min	Max	Unit
	Operating Frequency	Half CLK2 freq.		0	16	MHz
1	CLK2 Period		43	31		ns
2a	CLK2 High Time	at 2 V	43	9		ns
2b	CLK2 High Time	at (V_{CC} − 0.8 V)	43	5		ns
3a	CLK2 Low Time	at 2 V	43	9		ns
3b	CLK2 Low Time	at 0.8 V	43	7		ns
4	CLK2 Fall Time	(V_{CC} − 0.8 V) to 0.8 V (Note 3)	43		8	ns
5	CLK2 Rise Time	0.8 V to (V_{CC} − 0.8 V) (Note 3)	43		8	ns
6	A23–A1 Valid Delay	C_L = 120 pF (Note 4)	46	4	36	ns
7	A23–A1 Float Delay	(Note 1)	50	4	40	ns
8	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 75 pF (Note 4)	46	4	36	ns
9	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	50	4	40	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, ADS Valid Delay	C_L = 75 pF (Note 4)	46	4	33	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, ADS Float Delay	(Note 1)	50	6	35	ns
12	D15–D0 Write Data Valid Delay	C_L = 120 pF (Note 4)	46	4	40	ns
13	D15–D0 Write Data Float Delay	(Note 1)	50	4	35	ns
14	HLDA Valid Delay	C_L = 75 pF (Note 4)	46	4	33	ns
15	$\overline{\text{NA}}$ Setup Time		45	5		ns
16	$\overline{\text{NA}}$ Hold Time		45	21		ns
19	$\overline{\text{READY}}$ Setup Time		45	19		ns
20	$\overline{\text{READY}}$ Hold Time		45	4		ns
21	D15–D0 Read Data Setup Time		45	9		ns
22	D15–D0 Read Data Hold Time		45	6		ns
23	HOLD Setup Time		45	26		ns
24	HOLD Hold Time		45	5		ns
25	RESET Setup Time		51	13		ns
26	RESET Hold Time		51	4		ns
27	NMI, INTR Setup Time	(Note 2)	45	16		ns
28	NMI, INTR Hold Time	(Note 2)	45	16		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Setup Time	(Note 2)	45	16		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Hold Time	(Note 2)	45	5		ns

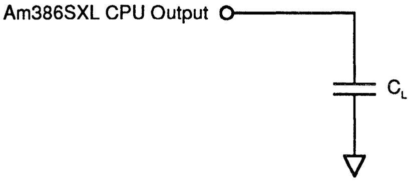
Notes: *Contact AMD for 16-MHz availability.

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with C_L set at 50 pF and derated to support the indicated distributed capacitive load. See Figures 52 and 53 for the capacitive derating curve.



15022B–031

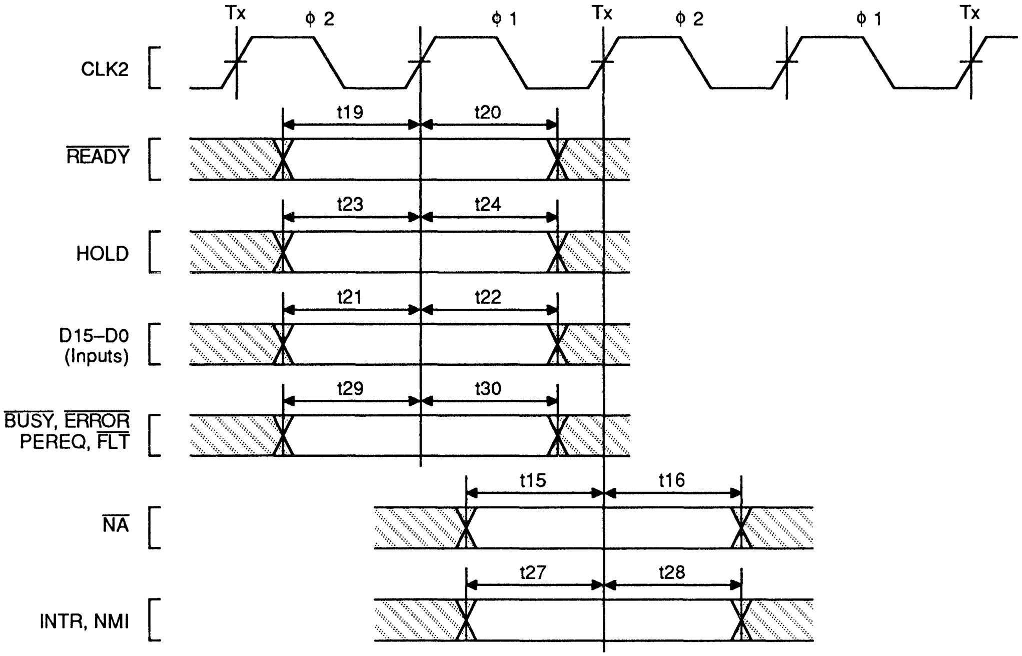
Figure 43. CLK2 Timing



15022B–032

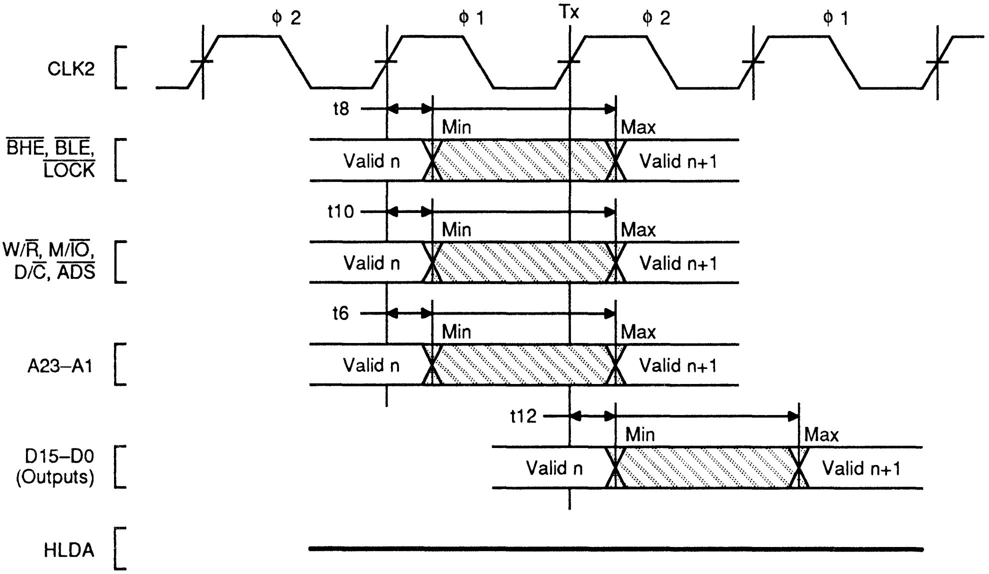
Figure 44. AC Test Circuit

SWITCHING WAVEFORMS



15022B–033

Figure 45. Input Setup and Hold Timing



15022B–034

Figure 46. Output Valid Delay Timing

SWITCHING WAVEFORMS (continued)

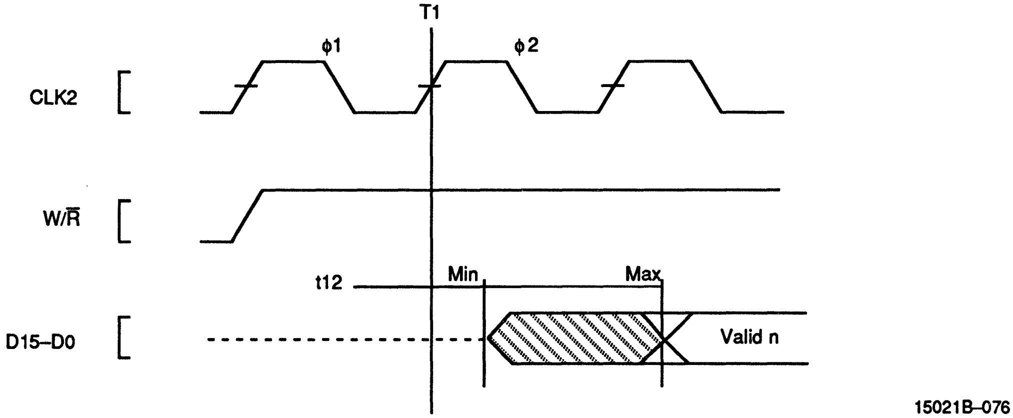


Figure 47. Write Data Valid Delay Timing (20 and 25 MHz)

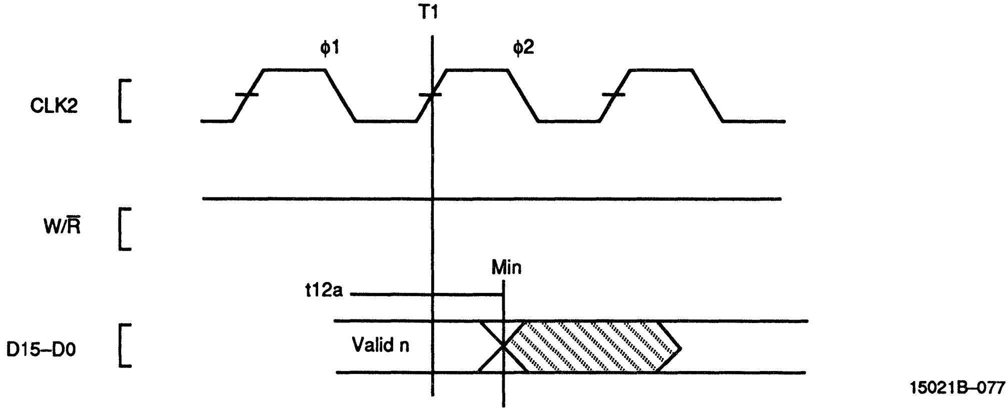


Figure 48. Write Data Hold Timing (20 and 25 MHz)

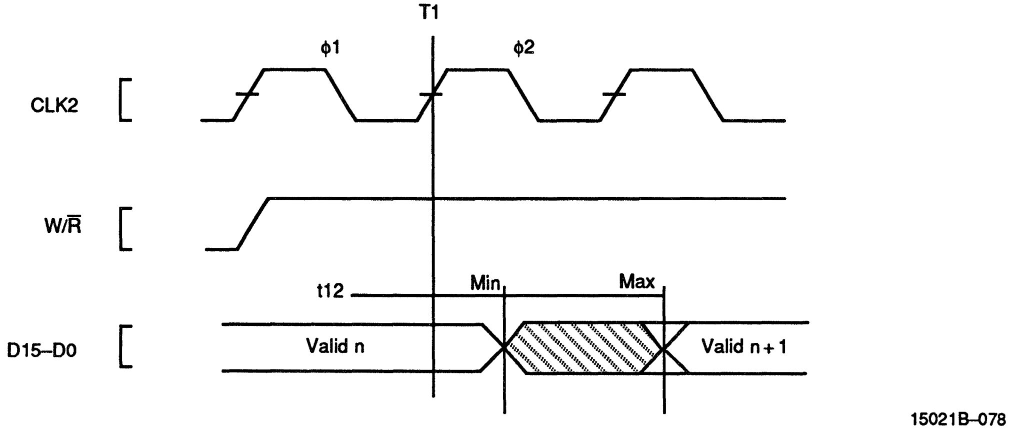
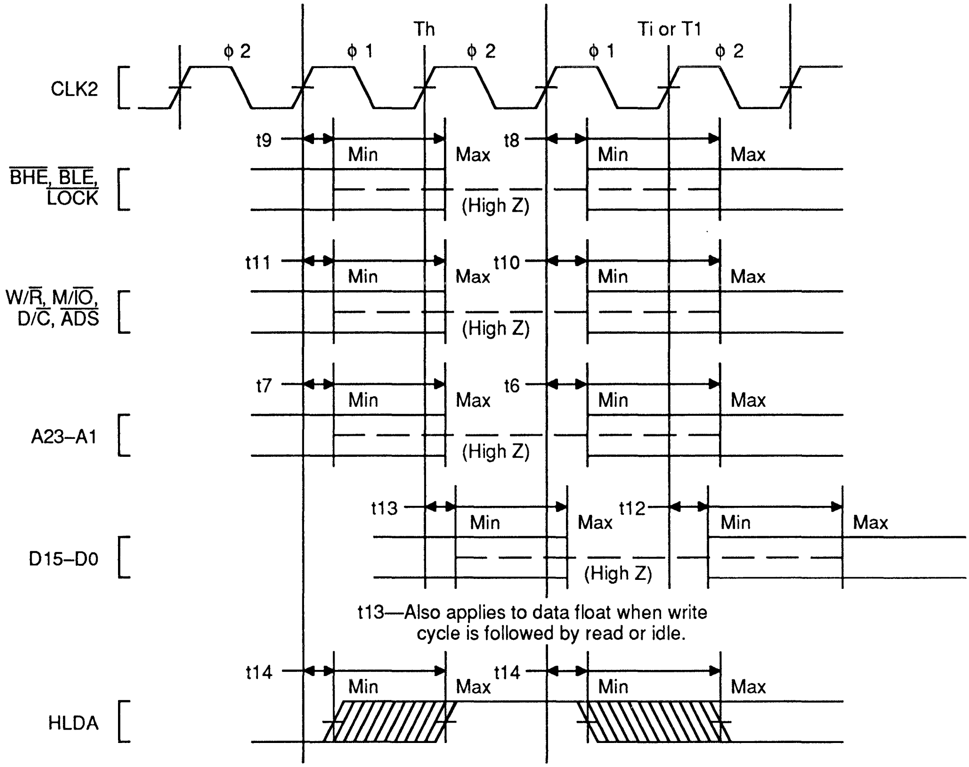
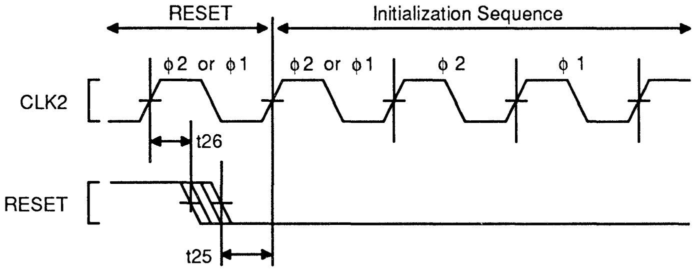


Figure 49. Write Data Valid Delay Timing (20 MHz)



15022B–035

Figure 50. Output Float Delay and HLDA Valid Delay Timing



15022B–036

Figure 51. RESET Setup and Hold Timing and Internal Phase

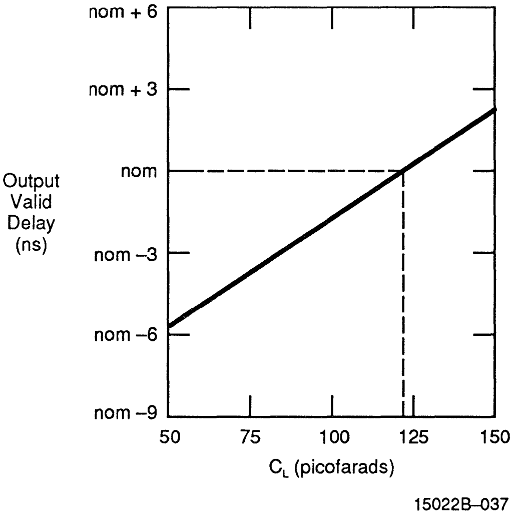


Figure 52. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 120 pF)

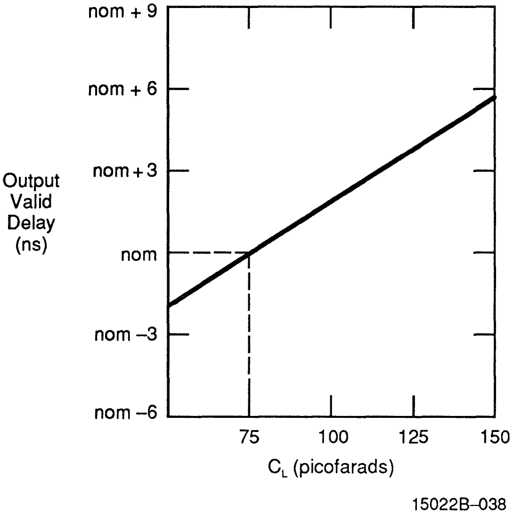


Figure 53. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 75 pF)

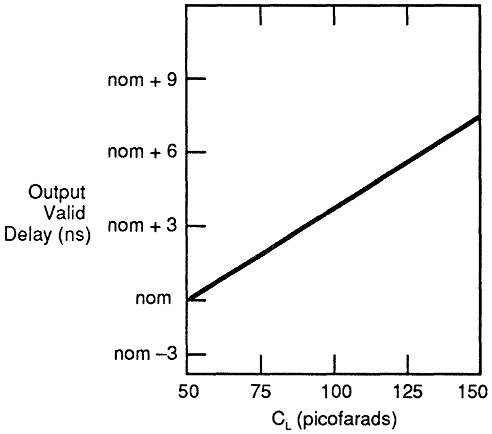


Figure 54. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (CL = 50 pF)

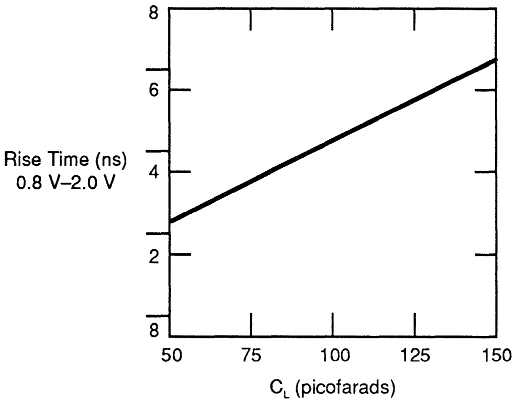


Figure 55. Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature

DIFFERENCES BETWEEN THE Am386SXL CPU AND THE Am386DX CPU

The following are the major differences between the Am386SXL CPU and the Am386DX CPU:

1. The Am386SXL CPU generates byte selects on $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$ (like the 8086 and 80286) to distinguish the upper and lower bytes on its 16-bit data bus. The Am386DX CPU uses four byte selects, $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, to distinguish between the different bytes on its 32-bit bus.
2. The Am386SXL CPU has no bus sizing option. The Am386DX CPU can select between either a 32-bit bus or a 16-bit bus by use of the $\overline{\text{BS16}}$ input. The Am386SXL CPU has a 16-bit bus size.
3. The $\overline{\text{NA}}$ pin operation in the Am386SXL CPU is identical to that of the $\overline{\text{NA}}$ pin on the Am386DX CPU

with one exception: the Am386DX CPU $\overline{\text{NA}}$ pin cannot be activated on 16-bit bus cycles (where $\overline{\text{BS16}}$ is Low in the Am386DX CPU case), whereas $\overline{\text{NA}}$ can be activated on any Am386SXL CPU bus cycle.

4. The contents of all Am386SXL CPU registers at reset are identical to the contents of the Am386DX CPU registers at reset, except the DX register. The DX register contains a component-stepping identifier at reset, that is,

   in Am386DX CPU, after reset

   DH = 3 indicates Am386DX CPU
   DI = revision number;

   in Am386SXL CPU, after reset

   DH = 23H indicates Am386SXL CPU
   DL = revision number.

5. The Am386DX CPU uses A31 and M/$\overline{\text{IO}}$ as selects for the math coprocessor. The Am386SXL CPU uses A23 and M/$\overline{\text{IO}}$ as selects.

6. The Am386DX CPU prefetch unit fetches code in four-byte units. The Am386SXL CPU prefetch unit reads two bytes as one unit (like the 80286). In $\overline{\text{BS16}}$ mode, the Am386DX CPU takes two consecutive bus cycles to complete a prefetch request. If there is a data read or write request after the prefetch starts, the Am386DX CPU will fetch all four bytes before addressing the new request.

7. Both Am386DX CPU and Am386SXL CPU have the same logical address space. The only difference is that the Am386DX CPU has a 32-bit physical address space and the Am386SXL CPU has a 24-bit physical address space. The Am386SXL CPU has a physical memory address space of up to 16 Mb instead of the 4 Gb available to the Am386DX CPU. Therefore, in Am386SXL CPU systems, the operating system must be aware of this physical memory limit and should allocate memory for applications programs within this limit. If an Am386DX CPU system uses only the lower 16 Mb of physical address, then there will be no extra effort required to migrate Am386DX CPU software to the Am386SXL CPU. Any application which uses more than 16 Mb of memory can run on the Am386SXL CPU, if the operating system utilizes the Am386SXL CPU's paging mechanism. In spite of this difference in physical address space, the Am386SXL CPU and Am386DX CPU can run the same operating systems and applications within their respective physical memory constraints.

8. The Am386SXL CPU has an input called $\overline{\text{FLT}}$ which three-states all bi-directional and output pins, including HLDA, when asserted. It is used with ON-Circuit Emulation (ONCE).

INSTRUCTION SET

This section describes the instruction set. The Instruction Set Clock Count Summary lists all instructions along with instruction encoding diagrams and clock counts. Further details of the instruction encoding are then provided in the following sections, which completely describe the encoding structure and the definition of all fields occurring within instructions.

The Am386SXL CPU Instruction Encoding and Clock Count Summary

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in the Instruction Set Clock Count Summary, by the processor clock period (e.g., 40 ns for a 25-MHz, 50 ns for a 20-MHz, and 62.5 ns for a 16-MHz Am386SXL microprocessor. The actual clock count of an Am386SXL CPU program will average 5% more than the calculated clock count due to instruction sequences which execute faster than they can be fetched from memory.

Instruction Clock Count Assumptions

1. The instruction has been prefetched, decoded, and is ready for execution.
2. Bus cycles do not require wait states.
3. There are no local bus HOLD requests delaying processor access to the bus.
4. No exceptions are detected during instruction execution.
5. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts shown. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.

Instruction Clock Count Notation

1. If two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.
2. *n* = number of times repeated.
3. *m* = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and all other bytes of the instruction and prefix(es) each count as one component.

Misaligned or 32-Bit Operand Accesses

— If instructions access a misaligned 16-bit operand or 32-bit operand on even address add:

2* clocks for read or write

4** clocks for read and write

— If instructions access a 32-bit operand on odd address add:

4* clocks for read or write

8** clocks for read and write

Wait States

Wait states add 1 clock per wait state to instruction execution for each data access.

Am386SXL Instruction Set Clock Count Summary

Instruction	Format				Clock Count		Notes	
					Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
GENERAL DATA TRANSFER								
MOV = Move:								
Register to Register/Memory	1 0 0 0 1 0 0 w	mod reg r/m			2/2	2/2*	b	h
Register/Memory to Register	1 0 0 0 1 0 1 w	mod reg r/m			2/4	2/4*	b	h
Immediate to Register/Memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	immediate data		2/2	2/2*	b	h
Immediate to Register (short form)	1 0 1 1 w reg	immediate data			2	2		
Memory to Accumulator (short form)	1 0 1 0 0 0 0 w	full displacement			4*	4*	b	h
Accumulator to Memory (short form)	1 0 1 0 0 0 1 w	full displacement			2*	2*	b	h
Register/Memory to Segment Register	1 0 0 0 1 1 1 0	mod sreg 3 r/m			2/5	22/23	b	h, i, j
Segment Register to Register/Memory	1 0 0 0 1 1 0 0	mod sreg r/m			2/2	2/2	b	h
MOVSX = Move with Sign Extension								
Register from Register/Memory	0 0 0 0 1 1 1 1	1 0 1 1 1 1 1 w	mod reg r/m		3/6*	3/6*	b	h
MOVZX = Move with Zero Extension								
Register from Register/Memory	0 0 0 0 1 1 1 1	1 0 1 1 0 1 1 w	mod reg r/m		3/6*	3/6*	b	h
PUSH = Push:								
Register/Memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m			5/7*	7/9*	b	h
Register (short form)	0 1 0 1 0 reg				2	4	b	h
Segment Register (ES,CS,SS, or DS) (short form)	000 sreg 2 110				2	4	b	h
Segment Register (ES, CS, SS, DS, FS, or GS)	0 0 0 0 1 1 1 1	10 sreg 3 000			2	4	b	h
Immediate	0 1 1 0 1 0 s 0	immediate data			2	4	b	h
PUSHA = Push All	0 1 1 0 0 0 0 0				18	34	b	h
POP = Pop								
Register/Memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m			5/7	7/9	b	h
Register (short form)	0 1 0 1 1 reg				6	6	b	h
Segment Register (ES, CS, SS, or DS)	000 sreg 2 111				7	25	b	h, i, j
Segment Register (ES, CS, SS, DS, FS, or GS)	0 0 0 0 1 1 1 1	10 sreg 3 001			7	25	b	h, i, j
POPA = Pop All	0 1 1 0 0 0 0 1				24	40	b	h
XCHG = Exchange								
Register/Memory with Register	1 0 0 0 0 1 1 w	mod reg r/m			3/5**	3/5**	b, f	f, h
Register with Accumulator (short form)	1 0 0 1 0 reg				3	3		
IN = Input From:				CLK Count Virtual 8086 Mode				
Fixed Port	1 1 1 0 0 1 0 w	port number		28***	12*	6*/26*		s/t, m
Variable Port	1 1 1 0 1 1 0 w			27***	13*	7*/27*		s/t, m

* If CPL ≤ IOPL ** If CPL > IOPL ***Clock count shown applies if I/O permission allows I/O to the port in Virtual 8086 Mode. If I/O bit map denies permission Exception 13 fault occurs; refer to clock counts for INT3 instruction.

Am386SXL Instruction Set Clock Count Summary (continued)

				Clock Count		Notes	
Instruction	Format			Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
OUT = Output To:			CLK Count Virtual 8086 Mode				
Fixed Port	1 1 1 0 0 1 1 w	port number	24***	10*	4*/24*		s/t, m
Variable Port	1 1 1 0 1 1 1 w		25***	11*	5*/25*		s/t, m
LEA = Load EA to Register	1 0 0 0 1 1 0 1	mod reg r/m		2	2		
SEGMENT CONTROL							
LDS = Load Pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		7*	26*/28*	b	h, i, j
LES = Load Pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		7*	26*/28*	b	h, i, j
LFS = Load Pointer to FS	0 0 0 0 1 1 1 1	1 0 1 1 0 1 0 0	mod reg r/m	7*	26*/28*	b	h, i, j
LGS = Load Pointer to GS	0 0 0 0 1 1 1 1	1 0 1 1 0 1 0 1	mod reg r/m	7*	26*/28*	b	h, i, j
LSS = Load Pointer to SS	0 0 0 0 1 1 1 1	1 0 1 1 0 0 1 0	mod reg r/m	7*	26*/28*	b	h, i, j
FLAG CONTROL							
CLC = Clear Carry Flag	1 1 1 1 1 0 0 0			2	2		
CLD = Clear Direction Flag	1 1 1 1 1 1 0 0			2	2		
CLI = Clear Interrupt Enable Flag	1 1 1 1 1 0 1 0			8	8		m
CLTS = Clear Task Switched Flag	0 0 0 0 1 1 1 1	0 0 0 0 0 1 1 0		5	5	c	l
CMC = Complement Carry Flag	1 1 1 1 0 1 0 1			2	2		
LAHF = Load AH into Flag	1 0 0 1 1 1 1 1			2	2		
POPF = Pop Flags	1 0 0 1 1 1 0 1			5	5	b	h, n
PUSHF = Push Flags	1 0 0 1 1 1 0 0			4	4	b	h
SAHF = Store AH into Flags	1 0 0 1 1 1 1 0			3	3		
STC = Set Carry Flag	1 1 1 1 1 0 0 1			2	2		
STD = Set Direction Flag	1 1 1 1 1 1 0 1						
STI = Set Interrupt Enable Flag	1 1 1 1 1 0 1 1			8	8		m
ARITHMETIC							
ADD = Add							
Register to Register	0 0 0 0 0 0 d w	mod reg r/m		2	2		
Register to Memory	0 0 0 0 0 0 0 w	mod reg r/m		7**	7**	b	h
Memory to Register	0 0 0 0 0 0 1 w	mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	immediate data	2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0 0 0 0 0 1 0 w	immediate data		2	2		
ADC = Add with Carry							
Register to Register	0 0 0 1 0 0 d w	mod reg r/m		2	2		

* If CPL ≤ IOPL ** If CPL > IOPL ***Clock count shown applies if I/O permission allows I/O to the port in Virtual 8086 Mode. If I/O bit map denies permission Exception 13 fault occurs; refer to clock counts for INT3 instruction.

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format			Clock Count		Notes	
				Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
ADC = Add with Carry (continued)							
Register to Memory	0 0 0 1 0 0 0 w	mod reg r/m		7**	7**	b	h
Memory to Register	0 0 0 1 0 0 1 w	mod reg r/m		6*	6*	b	h
Immediate to Register/Memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	immediate data	2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0 0 0 1 0 1 0 w	immediate data		2	2		
INC = Increment							
Register/Memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		2/6**	2/6**	b	h
Register (short form)	0 1 0 0 0 reg			2	2		
SUB = Subtract							
Register from Register	0 0 1 0 1 0 d w	mod reg r/m		2	2		
Register from Memory	0 0 1 0 1 0 0 w	mod reg r/m		7**	7**	b	h
Memory from Register	0 0 1 0 1 0 1 w	mod reg r/m		6*	6*	b	h
Immediate from Register/Memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	immediate data	2/7**	2/7**	b	h
Immediate from Accumulator (short form)	0 0 1 0 1 1 0 w	immediate data		2	2		
SBB = Subtract with Borrow							
Register from Register	0 0 0 1 1 0 d w	mod reg r/m		2	2		
Register from Memory	0 0 0 1 1 0 0 w	mod reg r/m		7**	7**	b	h
Memory from Register	0 0 0 1 1 0 1 w	mod reg r/m		6*	6*	b	h
Immediate from Register/Memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	immediate data	2/7**	2/7**	b	h
Immediate from Accumulator (short form)	0 0 0 1 1 1 0 w	immediate data		2	2		
DEC = Decrement							
Register/Memory	1 1 1 1 1 1 1 w	reg 0 0 1 r/m		2/6	2/6	b	h
Register (short form)	0 1 0 0 1 reg			2	2		
CMP = Compare							
Register with Register	0 0 1 1 1 0 d w	mod reg r/m		2	2		
Memory with Register	0 0 1 1 1 0 0 w	mod reg r/m		5*	5*	b	h
Register with Memory	0 0 1 1 1 0 1 w	mod reg r/m		6*	6*	b	h
Immediate with Register/Memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	immediate data	2/5*	2/5*	b	h
Immediate with Accumulator (short form)	0 0 1 1 1 1 0 w	immediate data		2	2		
NEG = Change Sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		2/6*	2/6*	b	h
AAA = ASCII Adjust for Add	0 0 1 1 0 1 1 1			4	4		

* If CPL ≤ IOPL   ** If CPL > IOPL

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format	Clock Count		Notes	
		Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
AAS = ASCII Adjust for Subtract	0 0 1 1 1 1 1 1	4	4		
DAA = Decimal Adjust for Add	0 0 1 0 0 1 1 1	4	4		
DAS = Decimal Adjust for Subtract	0 0 1 0 1 1 1 1	4	4		
MUL = Multiply (unsigned)					
Accumulator with Register Memory	1 1 1 1 0 1 1 w mod 1 0 0 r/m				
Multiplier – Byte		12–17/15–20*	12–17/15–20*	b, d	d, h
– Word		12–25/15–28*	12–25/15–28*	b, d	d, h
– Doubleword		12–41/17–46*	12–41/17–46*	b, d	d, h
IMUL = Integer Multiply (signed)					
Accumulator with Register Memory	1 1 1 1 0 1 1 w mod 1 0 1 r/m				
Multiplier – Byte		12–17/15–20*	12–17/15–20*	b, d	d, h
– Word		12–25/15–28*	12–25/15–28*	b, d	d, h
– Doubleword		12–41/17–46*	12–41/17–46*	b, d	d, h
Register with Register/Memory	0 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1 mod reg r/m				
Multiplier – Byte		12–17/15–20*	12–17/15–20*	b, d	d, h
– Word		12–25/15–28*	12–25/15–28*	b, d	d, h
– Doubleword		12–41/17–46*	12–41/17–46*	b, d	d, h
Register/Memory with Immediate to Register	0 1 1 0 1 0 s 1 mod reg r/m immediate data				
– Word		13–26	13–26/14–27	b, d	d, h
– Doubleword		13–42	13–42/16–45	b, d	d, h
DIV = Divide (unsigned)					
Accumulator by Register/Memory	1 1 1 1 0 1 1 w mod 1 1 0 r/m				
Divisor – Byte		14/17	14/17	b, e	e, h
– Word		22/25	22/25	b, e	e, h
– Doubleword		38/43	38/43	b, e	e, h
IDIV = Integer Divide (signed)					
Accumulator by Register/Memory	1 1 1 1 0 1 1 w mod 1 1 1 r/m				
Divisor – Byte		19/22	19/22	b, e	e, h
– Word		27/30	27/30	b, e	e, h
– Doubleword		43/48	43/48	b, e	e, h
AAD = ASCII Adjust for Divide	1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0	19	19		
AAM = ASCII Adjust for Multiply	1 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0	17	17		
CBW = Convert Byte to Word	1 0 0 1 1 0 0 0	3	3		
CWD = Convert Word to Double Word	1 0 0 1 1 0 0 1	2	2		
LOGIC					
Shift/Rotate Instruction					
Not Through Carry (**ROL, ROR, SAL, SAR, SHL, and SHR**)					
Register/Memory by 1	1 1 0 1 0 0 0 w mod TTT r/m	3/7**	3/7**	b	h

* If CPL ≤ IOPL ** If CPL > IOPL

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format	Clock Count		Notes	
		Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
LOGIC (continued)					
Not Through Carry **(ROL, ROR, SAL, SAR, SHL, and SHR) –(continued)**					
Register/Memory by CL	1 1 0 1 0 0 1 w mod TTT r/m	3/7*	3/7*	b	h
Register/Memory by Immediate Count	1 1 0 0 0 0 0 w mod TTT r/m (1)	3/7*	3/7*	b	h
Through Carry **(RCL and RCR)**					
Register/Memory by 1	1 1 0 1 0 0 0 w mod TTT r/m	9/10*	9/10*	b	h
Register/Memory by CL	1 1 0 1 0 0 1 w mod TTT r/m	9/10*	9/10*	b	h
Register/Memory by Immediate Count	1 1 0 0 0 0 0 w mod TTT r/m (1)	9/10*	9/10*	b	h

TTT	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

Instruction	Format	Clock Count		Notes	
SHLD = Shift Left Double					
Register/Memory by Immediate	0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 0 mod reg r/m (1)	3/7**	3/7**		
Register/Memory by CL	0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 1 mod reg r/m	3/7**	3/7**		
SHRD = Shift Right Double					
Register/Memory by Immediate	0 0 0 0 1 1 1 1 1 0 1 0 1 1 0 0 mod reg r/m (1)	3/7**	3/7**		
Register/Memory by CL	0 0 0 0 1 1 1 1 1 0 1 0 1 1 0 1 mod reg r/m	3/7**	3/7**		
AND = And					
Register to Register	0 0 1 0 0 0 d w mod reg r/m	2	2		
Register to Memory	0 0 1 0 0 0 0 w mod reg r/m	7**	7**	b	h
Memory to Register	0 0 1 0 0 0 1 w mod reg r/m	6*	6*	b	h
Immediate to Register/Memory	1 0 0 0 0 0 0 w mod 1 0 0 r/m immediate data	2/7*	2/7**	b	h
Immediate to Accumulator (short form)	0 0 1 0 0 1 0 w immediate data	2	2		
TEST = And Function to Flags, No Result					
Register/Memory and Register	1 0 0 0 0 1 0 w mod reg r/m	2/5*	2/5*	b	h
Immediate Data and Register/Memory	1 1 1 1 0 1 1 w mod 0 0 0 r/m immediate data	2/5*	2/5*	b	h
Immediate Data and Accumulator (short form)	1 0 1 0 1 0 0 w immediate data	2	2		
OR = Or					
Register to Register	0 0 0 0 1 0 d w mod reg r/m	2	2		
Register to Memory	0 0 0 0 1 0 0 w mod reg r/m	7**	7**	b	h
Memory to Register	0 0 0 0 1 0 1 w mod reg r/m	6*	6*	b	h

* If CPL ≤ IOPL ** If CPL > IOPL (1) Immediate 8-Bit Data

Am386SXL Instruction Set Clock Count Summary (continued)

					Clock Count		Notes	
Instruction	Format			Clock Count Virtual 8086 Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
LOGIC (continued)								
Immediate to Register/Memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	immediate data		2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0 0 0 0 1 1 0 w	immediate data			2	2		
XOR = Exclusive Or								
Register to Register	0 0 1 1 0 0 d w	mod reg r/m			2	2		
Register to Memory	0 0 1 1 0 0 0 w	mod reg r/m			7**	7**	b	h
Memory to Register	0 0 1 1 0 0 1 w	mod reg r/m			6*	6*	b	h
Immediate to Register/Memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	immediate data		2/7**	2/7**	b	h
Immediate to Accumulator (short form)	0 0 1 1 0 1 0 w	immediate data			2	2		
NOT = Invert Register/Memory	1 1 1 1 0 1 1 w	mod 0 1 0 r/m			2/6**	2/6**	b	h
STRING MANIPULATION								
CMPS = Compare Byte/Word	1 0 1 0 0 1 1 w				10*	10*	b	h
INS = Input Byte/Word from DX Port	0 1 1 0 1 1 0 w			29***	15	9*/29**	b	s/t, h, m
LODS = Load Byte/Word to AL/AX/EAX	1 0 1 0 1 1 0 w				5	5*	b	h
MOVE = Move Byte/Word	1 0 1 0 0 1 0 w				7	7**	b	h
OUTS = Output Byte/Word to DX Port	0 1 1 0 1 1 1 w			28***	14	8*/28**	b	s/t, h, m
SCAS = Scan Byte/Word	1 0 1 0 1 1 1 w				7*	7*	b	h
STOS = Store Byte/Word from AL/AX/EX	1 0 1 0 1 0 1 w				4*	4*	b	h
XLAT = Translate String	1 1 0 1 0 1 1 1				5*	5*		h
REPEATED STRING MANIPULATION								
Repeated by Count in CX or ECX								
REPE CMPS = Compare String (Find non-match)	1 1 1 1 0 0 1 1	1 0 1 0 0 1 1 w			5+9n**	5+9n**	b	h
REPNE CMPS = Compare String (Find match)	1 1 1 1 0 0 1 0	1 0 1 0 0 1 1 w			5+9n**	5+9n**	b	h
REP INS = Input String	1 1 1 1 0 0 1 0	0 1 1 0 1 1 0 w		***	13+6n*	7+6n*/ 27+6n**	6	s/t, h, m
REP LODS = Load String	1 1 1 1 0 0 1 0	1 0 1 0 1 1 0 w			5+6n*	5+6n*	b	h
REP MOVS = Move String	1 1 1 1 0 0 1 0	1 0 1 0 0 1 0 w			7+4n*	7+4n*	b	h
REP OUTS = Output String	1 1 1 1 0 0 1 0	0 1 1 0 1 1 1 w		***	12+5n*	6+5n*/ 26+5n**	b	s/t, h, m
REPE SCAS = Scan String (Find non-AL/AX/EAX)	1 1 1 1 0 0 1 1	1 0 1 0 1 1 1 w			5 + 8n*	5 + 8n*	b	h
REPNE SCAS = Scan String (Find AL/AX/EAX)	1 1 1 1 0 0 1 0	1 0 1 0 1 1 1 w			5 + 8n*	5 + 8n*	b	h
REP STOS = Store String	1 1 1 1 0 0 1 0	1 0 1 0 1 0 1 w			5 + 5n*	5 + 5n*	b	h

* If CPL≤IOPL ** If CPL > IOPL ***Clock count shown applies if I/O permission allows I/O to the port in Virtual 8086 Mode. If I/O bit map denies permission Exception 13 fault occurs; refer to clock counts for INT3 instruction.

Am386SXL Instruction Set Clock Count Summary (continued)

		Clock Count		Notes	
Instruction	Format	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
BIT MANIPULATION					
BSF = Scan Bit Forward	0 0 0 0 1 1 1 1 · 1 0 1 1 1 1 0 0 · mod reg r/m (1)	10 + 3n*	10 + 3n*	b	h
BSR = Scan Bit Reverse	0 0 0 0 1 1 1 1 · 1 0 1 1 1 1 0 1 · mod reg r/m	10 + 3n*	10 + 3n*	b	h
BT = Test Bit					
Register/Memory, Immediate	0 0 0 0 1 1 1 1 · 1 0 1 1 1 0 1 0 · mod 1 0 0 r/m (1)	3/6*	3/6*	b	h
Register/Memory, Register	0 0 0 0 1 1 1 1 · 1 0 1 0 0 0 1 1 · mod reg r/m	3/12*	3/12*	b	h
BTC = Test Bit and Complement					
Register/Memory, Immediate	0 0 0 0 1 1 1 1 · 1 0 1 1 1 0 1 0 · mod 1 1 1 r/m (1)	6/8*	6/8*	b	h
Register/Memory, Register	0 0 0 0 1 1 1 1 · 1 0 1 1 1 0 1 1 · mod reg r/m	6/13*	6/13*	b	h
BTR = Test Bit and Reset					
Register/Memory, Immediate	0 0 0 0 1 1 1 1 · 1 0 1 1 1 0 1 0 · mod 1 1 0 r/m (1)	6/8*	6/8*	b	h
Register/Memory, Register	0 0 0 0 1 1 1 1 · 1 0 1 1 0 0 1 1 · mod reg r/m	6/13*	6/13*	b	h
BTS = Test Bit and Set					
Register/Memory, Immediate	0 0 0 0 1 1 1 1 · 1 0 1 1 1 0 1 0 · mod 1 0 1 r/m (1)	6/8*	6/8*	b	h
Register/Memory, Register	0 0 0 0 1 1 1 1 · 1 0 1 0 1 0 1 1 · mod reg r/m	6/13*	6/13*	b	h
CONTROL TRANSFER					
CALL = Call					
Direct Within Segment	1 1 1 0 1 0 0 0 · full displacement	7 + m*	9 + m*	b	r
Register/Memory Indirect Within Segment	1 1 1 1 1 1 1 1 · mod 0 1 0 r/m	7 + m*/ 10 + m*	9 + m*/ 12 + m*	b	h, r
Direct Intersegment	1 0 0 1 1 0 1 0 · unsigned full offset, selector	17 + m*	42 + m*	b	j, k, r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			64 + m		h, j, k, r
Via Call Gate to Different Privilege Level (No Parameters)			98 + m		h, j, k, r
Via Call Gate to Different Privilege Level (x Parameters)			106 + 8x + m		h, j, k, r
From 80286 Task to 80286 TSS			285		h, j, k, r
From 80286 Task to Am386SXL CPU TSS			310		h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386SXL CPU TSS)			229		h, j, k, r
From Am386SXL CPU Task to 80286 TSS			285		h, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS			392		h, j, k, r
From Am386SXL CPU Task to Virtual 8086 Task (Am386SXL CPU TSS)			309		h, j, k, r
Indirect Intersegment	1 1 1 1 1 1 1 1 · mod 0 1 1 r/m	30 + m	46 + m	b	h, j, k, r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			68 + m		h, j, k, r
Via Call Gate to Different Privilege Level (No Parameters)			102 + m		h, j, k, r
Via Call Gate to Different Privilege Level (x Parameters)			110 + 8x + m		h, j, k, r
From 80286 Task to 80286 TSS					h, j, k, r
From 80286 Task to Am386SXL CPU TSS					h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386SXL CPU TSS)					h, j, k, r
From Am386SXL CPU Task to 80286 TSS					h, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS			399		h, j, k, r
From Am386SXL CPU Task to Virtual 8086 Task (Am386SXL CPU TSS)					h, j, k, r

* If CPL ≤ IOPL ** If CPL > IOPL (1) Immediate 8-bit data

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format	Clock Count Real Address Mode/ Virtual 8086 Mode	Clock Count Protected Virtual Address Mode	Notes Real Address Mode/ Virtual 8086 Mode	Notes Protected Virtual Address Mode
CONTROL TRANSFER (continued)					
JMP = Unconditional Jump					
Short	1 1 1 0 1 0 1 1 8-bit displacement	7+m	7+m		r
Direct within Segment	1 1 0 1 0 0 0 1 full displacement	7+m	7+m		r
Register/Memory Indirect Within Segment	1 1 1 1 1 1 1 1 mod 1 0 0 r/m	9+m/ 14+m	9+m/ 14+m	b	h, r
Direct Intersegment	1 1 1 0 1 0 1 0 unsigned full offset, selector	16+m	31+m		j, k, r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			53+m		h, j, k, r
From 80286 Task to 80286 TSS					h, j, k, r
From 80286 Task to Am386SXL CPU TSS					h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386SXL CPU TSS)					h, j, k, r
From Am386SXL CPU Task to 80286 TSS					h, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS					h, j, k, r
From Am386SXL CPU Task to Virtual 8086 Task (Am386SXL CPU TSS)			395		h, j, k, r
Indirect Intersegment	1 1 1 1 1 1 1 1 mod 1 0 1 r/m	17+m	31+m	b	h, j, k, r
Protected Mode Only (Indirect Intersegment)			49+m		h, j, k, r
Via Call Gate to Same Privilege Level					h, j, k, r
From 80286 Task to 80286 TSS					h, j, k, r
From 80286 Task to Am386SXL CPU TSS					h, j, k, r
From 80286 Task to Virtual 8086 Task (Am386SXL CPU TSS)					h, j, k, r
From Am386SXL CPU Task to 80286 TSS			328		h, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS					h, j, k, r
From Am386SXL CPU Task to Virtual 8086 Task (Am386SXL CPU TSS)					
RET = Return from Call					
Within Segment	1 1 0 0 0 0 1 1		12+m	b	g, h, r
Within Segment Adding Immediate to SP	1 1 0 0 0 0 1 0 16-bit displacement		12+m	b	g, h, r
Intersegment	1 1 0 0 1 0 1 1		36+m	b	g, h, j, k, r
Intersegment Adding Immediate to SP	1 1 0 0 1 0 1 0 16-bit displacement		36+m	b	g, h, j, k, r
Protected Mode Only (RET): to Different Privilege Level			72		h, j, k, r
Intersegment			72		h, j, k, r
Intersegment Adding Immediate to SP					
CONDITIONAL JUMPS (Note: Times are Jump "Taken or Not Taken")					
JO = Jump on Overflow					
8-bit Displacement	0 1 1 1 0 0 0 0 8-bit displacement	7+m or 3	7+m or 3		r
Full Displacement	0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 full displacement	7+m or 3	7+m or 3		r
JNO = Jump on Not Overflow					
8-bit Displacement	0 1 1 1 0 0 0 1 8-bit displacement	7+m or 3	7+m or 3		r
Full Displacement	0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 full displacement	7+m or 3	7+m or 3		r

* If CPL ≤ IOPL ** If CPL > IOPL

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format			Clock Count		Notes	
				Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
CONDITIONAL JUMPS (continued)							
JB/JNAE = Jump on Below/Not Above or Equal							
8-bit Displacement	0 1 1 1 0 0 1 0	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 0 1 0	full displacement	7 + m or 3	7 + m or 3		r
JNB/JAE = Jump on Not Below/Above or Equal							
8-bit Displacement	0 1 1 1 0 0 1 1	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 0 1 1	full displacement	7 + m or 3	7 + m or 3		r
JE/JZ = Jump on Equal/Zero							
8-bit Displacement	0 1 1 1 0 1 0 0	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 0 0	full displacement	7 + m or 3	7 + m or 3		r
JNE/JNZ = Jump on Not Equal/Not Zero							
8-bit Displacement	0 1 1 1 0 1 0 1	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 0 1	full displacement	7 + m or 3	7 + m or 3		r
JBE/JNA = Jump on Below or Equal/Not Above							
8-bit Displacement	0 1 1 1 0 1 1 0	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 1 0	full displacement	7 + m or 3	7 + m or 3		r
JNBE/JA = Jump on Not Below or Equal/Above							
8-bit Displacement	0 1 1 1 0 1 1 1	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 1 1	full displacement	7 + m or 3	7 + m or 3		r
JS = Jump on Sign							
8-bit Displacement	0 1 1 1 1 0 0 0	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 1 0 0 0	full displacement	7 + m or 3	7 + m or 3		r
JNS = Jump on Not Sign							
8-bit Displacement	0 1 1 1 1 0 0 1	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 1 0 0 1	full displacement	7 + m or 3	7 + m or 3		r
JP/JPE = Jump on Parity/Parity Even							
8-bit Displacement	0 1 1 1 1 0 1 0	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 1 0 1 0	full displacement	7 + m or 3	7 + m or 3		r
JNP/JPO = Jump on Not Parity/Parity Odd							
8-bit Displacement	0 1 1 1 1 0 1 1	8-bit displacement		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 1 0 1 1	full displacement	7 + m or 3	7 + m or 3		r

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format	Clock Count		Notes	
		Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
CONDITIONAL JUMPS (continued)					
JL/JNGE = Jump on Less/Not Greater or Equal					
8-bit Displacement	0 1 1 1 1 1 0 0 8-bit displacement	7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 full displacement	7 + m or 3	7 + m or 3		r
JNL/JGE = Jump on Net Less/Greater or Equal					
8-bit Displacement	0 1 1 1 1 1 0 1 8-bit displacement	7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1 1 0 0 0 1 1 0 1 full displacement	7 + m or 3	7 + m or 3		r
JLE/JNG = Jump on Less or Equal/Not Greater					
8-bit Displacement	0 1 1 1 1 1 1 0 8-bit displacement	7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 full displacement	7 + m or 3	7 + m or 3		r
JNLE/JG = Jump on Not Less or Equal/Greater					
8-bit Displacement	0 1 1 1 1 1 1 1 8-bit displacement	7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 full displacement	7 + m or 3	7 + m or 3		r
JCXZ = Jump on CX Zero*	1 1 1 0 0 0 1 1 8-bit displacement	9 + m or 5	9 + m or 5		r
JECXZ = Jump on ECX Zero	1 1 1 0 0 0 1 1 8-bit displacement	9 + m or 5	9 + m or 5		r
LOOP = Loop CX Times	1 1 1 0 0 0 1 0 8-bit displacement	11 + m	11 + m		r
LOOPZ/LOOPE = Loop with Zero/Equal	1 1 1 0 0 0 0 1 8-bit displacement	11 + m	11 + m		r
LOOPNZ/LOOPNE = Loop while Not Zero	1 1 1 0 0 0 0 0 8-bit displacement	11 + m	11 + m		r
CONDITIONAL BYTE SET (Note: Times Are Register/Memory)					
SETO = Set Byte on Overflow					
To Register/Memory	0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 0 mod 0 0 0 r/m	4/5*	4/5*		h
SETNO = Set Byte on Not Overflow					
To Register/Memory	0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 1 mod 0 0 0 r/m	4/5*	4/5*		h
SETB/SETNAE = Set Byte on Below/Not Above or Equal					
To Register/Memory	0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 mod 0 0 0 r/m	4/5*	4/5*		h
SETNB = Set Byte on Not Below/Above or Equal					
To Register/Memory	0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 mod 0 0 0 r/m	4/5*	4/5*		h
SETE/SETZ = Set Byte on Equal/Zero					
To Register/Memory	0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 mod 0 0 0 r/m	4/5*	4/5*		h
SETNE/SETNZ = Set Byte on Not Equal/Not Zero					
To Register/Memory	0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 1 mod 0 0 0 r/m	4/5*	4/5*		h

* Address Size Prefix differentiates JCXZ from JECXZ

Am386SXL Instruction Set Clock Count Summary (continued)

				Clock Count		Notes	
Instruction	Format			Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
CONDITIONAL BYTE SET (continued)							
SETBE/SETNA = Set Byte on Below or Equal/Not Above							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 0 1 1 0	mod 0 0 0 r/m	4/5*	4/5*		h
SETNBE/SETA = Set Byte on Not Below or Equal/Above							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 0 1 1 1	mod 0 0 0 r/m	4/5*	4/5*		h
SETS = Set Byte on Sign							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 1 0 0 0	mod 0 0 0 r/m	4/5*	4/5*		h
SETNS = Set Byte on Not Sign							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 1 0 0 1	mod 0 0 0 r/m	4/5*	4/5*		h
SETP/SETPE = Set Byte on Parity/Parity Even							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 1 0 1 0	mod 0 0 0 r/m	4/5*	4/5*		h
SETNP/SETPO = Set Byte on Not Parity/Parity Odd							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 1 0 1 1	mod 0 0 0 r/m	4/5*	4/5*		h
SETL/SETNGE = Set Byte on Less/Not Greater or Equal							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 1 1 0 0	mod 0 0 0 r/m	4/5*	4/5*		h
SETNL/SETGE = Set Byte on Not Less/Greater or Equal							
To Register/Memory	0 0 0 0 1 1 1 1	0 1 1 1 1 1 0 1	mod 0 0 0 r/m	4/5*	4/5*		h
SETLE/SETNG = Set Byte on Less or Equal/Not Greater							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 1 1 1 0	mod 0 0 0 r/m	4/5*	4/5*		h
SETNLE/SETG = Set Byte on Not Less or Equal/Greater							
To Register/Memory	0 0 0 0 1 1 1 1	1 0 0 1 1 1 1 1	mod 0 0 0 r/m	4/5*	4/5*		h
ENTER = Enter Procedure	1 1 0 0 1 0 0 0	16-bit displacement, 8-bit level					
L = 0				10	10	b	h
L = 1				14	14	b	h
L > 1				17 + 8 (n − 1)	17 + 8 (n − 1)	b	h
LEAVE = Leave Procedure	1 1 0 0 1 0 0 1			4	4	b	h
INTERRUPT INSTRUCTIONS							
INT = Interrupt:							
Type Specified	1 1 0 0 1 1 0 1	type		37		b	
Type 3	1 1 0 0 1 1 0 0			33		b	
INTO = Interrupt 4 if Overflow Flag Set	1 1 0 0 1 1 1 0						
If OF = 1				35		b, e	
If OF = 0				3	3	b, e	

* If CPL ≤ IOPL ** If CPL > IOPL

Am386SXL Instruction Set Clock Count Summary (continued)

		Clock Count		Notes	
Instruction	Format	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS (continued)					
INT = Interrupt:					
Type Specified					
Type 3					
Bound = Interrupt 5 if Detected Value Out of Range	0 1 1 0 0 0 1 0 | mod reg r/m				
If Out of Range		44		b, e	e,g,h,j,k,r
If In Range		10	10	b, e	e,g,h,j,k,r
Protected Mode Only (INT)					
INT: Type Specified					
Via Interrupt or Trap Gate to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			111		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			438		g, j, k, r
From 80286 Task to Am386SXL CPU TSS via Task Gate			465		g, j, k, r
From 80286 Task to Virtual 8086 Mode via Task Gate			382		g, j, k, r
From Am386SXL CPU Task to 80286 TSS via Task Gate			440		g, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS via Task Gate			467		g, j, k, r
From Am386SXL CPU Task to Virtual 8086 Mode via Task Gate			384		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			445		g, j, k, r
From Virtual 8086 Mode to Am386SXL CPU TSS via Task Gate			472		g, j, k, r
From Virtual 8086 Mode to Privilege Level 0 via Trap Gate or Interrupt Gate			275		g, j, k, r
INT: Type 3					
Via Interrupt or Trap Gate to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			111		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			382		g, j, k, r
From 80286 Task to Am386SXL CPU TSS via Task Gate			409		g, j, k, r
From 80286 Task to Virtual 8086 Mode via Task Gate			326		g, j, k, r
From Am386SXL CPU Task to 80286 TSS via Task Gate			384		g, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS via Task Gate			411		g, j, k, r
From Am386SXL CPU Task to Virtual 8086 Mode via Task Gate			328		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			389		g, j, k, r
From Virtual 8086 Mode to Am386SXL CPU TSS via Task Gate			416		g, j, k, r
From Virtual 8086 Mode to Privilege Level 0 via Trap Gate or Interrupt Gate			223		g, j, k, r
INTO					
Via Interrupt or Trap Gate to Same Privilege Level			71		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			111		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			384		g, j, k, r
From 80286 Task to Am386SXL CPU TSS via Task Gate			411		g, j, k, r
From 80286 Task to Virtual 8086 Mode via Task Gate			328		g, j, k, r
From Am386SXL CPU Task to 80286 TSS via Task Gate			Am386DX		g, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS via Task Gate			413		g, j, k, r
From Am386SXL CPU Task to Virtual 8086 Mode via Task Gate			329		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			391		g, j, k, r
From Virtual 8086 Mode to Am386SXL CPU TSS via Task Gate			418		g, j, k, r
From Virtual 8086 Mode to Privilege Level 0 via Trap Gate or Interrupt Gate			223		g, j, k, r
BOUND					
Via Interrupt or Trap Gate to Same Privilege Level			71		g, j, k, r
Via interrupt or Trap Gate to Different Privilege Level			111		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			358		g, j, k, r
From 80286 Task to Am386SXL CPU TSS via Task Gate			388		g, j, k, r

Am386SXL Instruction Set Clock Count Summary (continued)

		Clock Count		Notes	
Instruction	Format	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS (continued)					
BOUND (continued)					
From 80286 Task to Virtual 8086 Mode via Task Gate			335		g, j, k, r
From Am386SXL CPU Task to 80286 TSS via Task Gate			368		g, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS via Task Gate			398		g, j, k, r
From Am386SXL CPU Task to Virtual 8086 Mode via Task Gate			347		g, j, k, r
From Virtual 8086 Mode to 80286 TSS via Task Gate			368		g, j, k, r
From Virtual 8086 Mode to Am386SXL CPU TSS via Task Gate			398		g, j, k, r
From Virtual 8086 Mode to Privlege Level 0 via Trap Gate or Interrupt Gate			223		g, j, k, r
INTERRUPT RETURN					
IRET = Interrupt Return	1 1 0 0 1 1 1 1	24			g,h,j,k,r
Protected Mode Only (IRET)					
Via Interrupt or Trap Gate to Same Privilege Level (within Task)			42		g,h,j,k,r
Via Interrupt or Trap Gate to Different Privilege Level (within Task)			86		g,h,j,k,r
From 80286 Task to 80286 TSS			285		h, j, k, r
From 80286 Task to Am386SXL CPU TSS			318		h, j, k, r
From 80286 Task to Virtual 8086 Task			267		h, j, k, r
From 80286 Task to Virtual 8086 Mode (within Task)			113		h. j. k. r
From Am386SXL CPU Task to Virtual 8086 TSS			324		h, j, k, r
From Am386SXL CPU Task to 80286 TSS			328		h, j, k, r
From Am386SXL CPU Task to Am386SXL CPU TSS			377		h, j, k, r
From Am386SXL CPU Task to Virtual 8086 Mode (within Task)			113		h, j, k, r
PROCESSOR CONTROL					
HLT = Halt	1 1 1 1 0 1 0 0	5	5		l
MOV = Move To and From Control/Debug/Test Registers					
CR0/CR2/CR3 from Register	0 0 0 0 1 1 1 1 · 0 0 1 0 0 0 1 0 · 1 1 eee reg	10/4/5	10/4/5		l
Register from CR3–CR0	0 0 0 0 1 1 1 1 · 0 0 1 0 0 0 0 0 · 1 1 eee reg	5	6		l
DR3–DR0 from Register	0 0 0 0 1 1 1 1 · 0 0 1 0 0 0 1 1 · 1 1 eee reg	22	22		l
DR7–DR6 from Register	0 0 0 0 1 1 1 1 · 0 0 1 0 0 0 1 1 · 1 1 eee reg	16	16		l
Register from DR7–DR6	0 0 0 0 1 1 1 1 · 0 0 1 0 0 0 0 1 · 1 1 eee reg	14	14		l
Register from DR3–DR0	0 0 0 0 1 1 1 1 · 0 0 1 0 0 0 0 1 · 1 1 eee reg	22	22		l
TR7–TR6 from Register	0 0 0 0 1 1 1 1 · 0 0 1 0 0 1 1 0 · 1 1 eee reg	12	12		l
Register from TR7–TR6	0 0 0 0 1 1 1 1 · 0 0 1 0 0 1 0 0 · 1 1 eee reg	12	12		l
NOP = No Operation	1 0 0 1 0 0 0 0	3	3		l
WAIT = Wait until $\overline{\text{BUSY}}$ pin is negated	1 0 0 1 1 0 1 1	6	6		l
PROCESSOR EXTENSION INSTRUCTIONS					
Processor Extension Escape	1 1 0 1 1 T T T · mod L L L r/m TTT and LLL bits are op-code information for coprocessor.	See 387SXL Datasheet for clock counts			h
PREFIX BYTES					
Address Size Prefix	0 1 1 0 0 1 1 1	0	0		

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format	Clock Count		Notes	
		Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
PREFIX BYTES (continued)					
LOCK = Bus Lock Prefix	11110000	0	0		m
Operand Size Prefix	01100110	0	0		
Segment Override Prefix					
CS	00101110	0	0		
DS	00111110	0	0		
ES	00100110	0	0		
FS	01100100	0	0		
GS	01100101	0	0		
SS	00110110	0	0		
PROTECTION CONTROL					
ARPL = Adjust Requested Privilege Level					
From Register/Memory	01100011 mod reg r/m	N/A	20/21**	a	h
LAR = Load Access Rights					
From Register/Memory	00001111 00000010 mod reg r/m	N/A	15/16*	a	g, h, j, p
LGDT = Load Global Descriptor					
Table Register	00001111 00000001 mod 010 r/m	11*	11*	b, c	h, l
LIDT = Load Interrupt Descriptor					
Table Register	00001111 00000001 mod 011 r/m	11*	11*	b, c	h, l
LLDT = Load Local Descriptor					
Table Regiser to Register/Memory	00001111 00000000 mod 010 r/m	N/A	20/24*	a	g, h, j, l
LMSW = Load Machine Status Word					
From Register/Memory	00001111 00000001 mod 110 r/m	10/13	10/13*	b, c	h, l
LSL = Load Segment Limit					
From Register/Memory	00001111 00000011 mod reg r/m				
Byte-Granular Limit		N/A	20/21*	a	g, h, j, p
Page-Granular Limit		N/A	25/26*	a	g, h, j, p
LTR = Load Task Register					
From Register/Memory	00001111 00000000 mod 001 r/m	N/A	23/27*	a	g, h, j, l
SGDT = Store Global Descriptor					
Table Register	00001111 00000001 mod 000 r/m	9*	9*	b, c	h
SIDT = Store Interrupt Descriptor					
Table Register	00001111 00000001 mod 001 r/m	9*	9*	b, c	h

* If CPL ≤ IOPL ** If CPL > IOPL

Am386SXL Instruction Set Clock Count Summary (continued)

Instruction	Format			Clock Count		Notes	
				Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode/ Virtual 8086 Mode	Protected Virtual Address Mode
PROTECTION CONTROL (continued)							
SLDT = Store Local Descriptor Table Register							
To Register/Memory	00001111	00000000	mod 000 r/m	N/A	2/2*	a	h
SMSW = Store Machine Status Word	00001111	00000001	mod 100 r/m	2/2*	2/2*	b, c	h, l
STR = Store Task Register							
To Register/Memory	00001111	00000000	mod 001 r/m	N/A	2/2*	a	h
VERR = Verify Read Access							
Register/Memory	00001111	00000000	mod 100 r/m	N/A	10/11*	a	g, h, j, p
VERW = Verify Write Access	00001111	00000000	mod 101 r/m	N/A	15/16*	a	g, h, j, p

* If CPL ≤ IOPL ** If CPL > IOPL

Instruction Notes for Instruction Set Summary

Notes a through c apply to Real Address Mode only:

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in Exception 6 (invalid op-code).

b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS, or GS limit (FFFFH). Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Notes d through g apply to Real Address Mode and Protected Virtual Address Mode:

d. The Am386SXL CPU uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).

Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:
Actual Clock = if $m <> 0$, then max ([$\log_2 |m|$], 3) + b clocks;
= if $m = 0$, then 3 + b clocks

In this formula, m is the multiplier, and
b = 9 for register to register;
b = 12 for memory to register;
b = 10 for register with immediate to register;
b = 11 for memory with immediate to register.

e. An exception may occur, depending on the value of the operand.

f. $\overline{\text{LOCK}}$ is automatically asserted, regardless of the presence or absence of the LOCK prefix.

g. $\overline{\text{LOCK}}$ is asserted during descriptor table accesses.

Notes h through r apply to Protected Virtual Address Mode only:

h. Exception 13 fault will occur if the memory operand in CS, DS, ES, FS, or GS cannot be used due to either a segment limit violation or an access rights violation. If a stack limit is violated, an Exception 12 occurs.

i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an Exception 13 fault. The segment's descriptor must indicate "present" or Exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an Exception 12 occurs.

j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert $\overline{\text{LOCK}}$ to maintain descriptor integrity in multiprocessor systems.

k. JMP, CALL, INT, RET, and IRET instructions referring to another code segment will cause an Exception 13, if an applicable privilege rule is violated.

l. An Exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).

m. An Exception 13 fault occurs if CPL is greater than IOPL.

n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.

o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 if desiring to reset the PE bit.

p. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.

q. If the coprocessor's memory operand violates a segment limit or segment access rights, an Exception 13 fault will occur before the ESC instruction is executed. An Exception 12 fault will occur if the stack limit is violated by the operand's starting address.

r. The destination of a JMP, CALL, INT, RET, or IRET must be in the defined limit of a code segment or an Exception 13 fault will occur.

s/t. The instruction will execute in s clocks if CPL ≤ IOPL. If CPL > IOPL, the instruction will take t clock.

Instruction Encoding

Overview

All instruction encodings are subsets of the general instruction format shown in the Am386SXL Instruction Set Clock Count Summary (pages 1-406 thru 1-420). Instructions consist of one or two primary op-code 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 op-code(s), 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 op-code byte(s). This byte (mod r/m) specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte (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 56 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but Figure 56 does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the op-code bytes themselves. Table 22 is a complete list of all fields appearing in the Instruction Set. Further ahead, following Table 22, are detailed tables for each field.

32-Bit Extensions of the Instruction Set

With the Am386SXL CPU, the 8086/80186/80286 Instruction Set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types; and, 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

Whether the instruction defaults to operations of 16 bits or 32 bits depends on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 bits or 16 bits) for both operands and effective addresses, when executing that code segment. In the Real Address Mode or Virtual 8086 Mode, no code segment descriptors are used, but a D value of 0 is assumed internally by the Am386SXL CPU when operating in those modes (for 16-bit default sizes compatible with the 8086/80C186/80286).

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 op-code bytes and affect only the instruction they precede. If necessary, one or both of the prefixes may be placed

Table 22. Fields within Instructions

Field Name	Description	Number of Bits
w	Specifies if data is byte or full size (full size is either 16 or 32 bits)	1
d	Specifies direction of data operation	1
s	Specifies if an immediate data field must be sign-extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (effective address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, and ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, and GS	3
tttn	For Conditional Instructions, specifies a condition asserted or a condition negated	4

Note: Table 21 shows encoding of individual instructions.

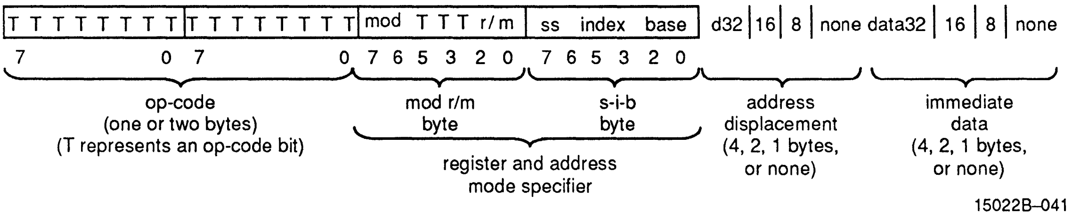


Figure 56. General Instruction Format

before the op-code bytes. The presence of the Operand Size Prefix and the Effective Address Prefix will toggle the operand size or the effective address size, respectively, to the value opposite from the Default setting. For example, if the default operand size is for 32-bit data operations, then presence of the Operand Size Prefix toggles the instruction to 16-bit data operation. As another example, if the default effective address size is 16 bits, presence of the Effective Address Size prefix toggles the instruction to use 32-bit effective address computations.

These 32-bit extensions are available in all modes, including the Real Address Mode and 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 of the extended registers.

Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode and so on. The exact encodings of these fields are defined immediately ahead.

Encoding of Operand Length (w) Field

For any given instruction performing a data operation, the instruction is executing as a 32-bit operation or a 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in the table below.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

Encoding of the General Register (reg) Field

The general register is specified by the reg field, which may appear in the primary op-code bytes, or as the reg field of the mod r/m byte, or as the r/m field of the mod r/m byte.

Encoding of reg Field When w Field is not Present in Instruction

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
101	SI	ESI
101	DI	EDI

Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-Bit Data Operations		
	Function of w Field	
reg	(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		
	Function of w Field	
reg	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

Encoding of the Segment Register (sreg) Field

The sreg field in certain instructions is a 2-bit field, allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the Am386SXL CPU FS and GS segment registers to be specified.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

Encoding of Address Mode

Except for special instructions, such as PUSH or POP, where the addressing mode is predetermined, the addressing mode for the current instruction is specified by addressing bytes following the primary op-code. 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 is specified when using 32-bit addressing mode, the mod r/m byte has r/m = 100, and mod = 00, 01, or 10. When the s-i-b 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 56) sometimes used as an extension of the primary op-code. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address, while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the mod r/m byte is interpreted as a 16-bit addressing mode specifier. When 32-bit addressing is used, the mod r/m byte is interpreted as a 32-bit addressing mode specifier.

Tables on the following pages define all encodings of all 16-bit addressing modes and 32-bit addressing modes.

Encoding of 16-Bit Address Mode with mod r/m Byte

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	DS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]

mod r/m	Effective Address
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP+SI+d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	Register—See Below
11 001	Register—See Below
11 010	Register—See Below
11 011	Register—See Below
11 100	Register—See Below
11 101	Register—See Below
11 110	Register—See Below
11 111	Register—See Below

Register Specified by r/m During 16-Bit Data Operations		
	Function of w Field	
mod r/m	(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		
	Function of w Field	
mod r/m	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-Bit Address Mode with mod r/m Byte (no s-i-b byte present)

mod r/m		Effective Address
00	000	DS:[EAX]
00	001	DS:[ECX]
00	010	DS:[EDX]
00	011	DS:[EBX]
00	100	s-i-b is present
00	101	DS:d32
00	110	DS:[ESI]
00	111	DS:[EDI]
01	000	DS:[EAX + d8]
01	001	DS:[ECX + d8]
01	010	DS:[EDX + d8]
01	011	DS:[EBX + d8]
01	100	s-i-b is present
01	101	SS:[EBP + d8]
01	110	DS:[ESI + d8]
01	111	DS:[EDI + d8]

mod r/m		Effective Address
10	000	DS:[EAX + d32]
10	001	DS:[ECX + d32]
10	010	DS:[EDX + d32]
10	011	DS:[EBX + d32]
10	100	s-i-b is present
10	101	SS:[EBP + d32]
10	110	DS:[ESI + d32]
10	111	DS:[EDI + d32]
11	000	Register—See Below
11	001	Register—See Below
11	010	Register—See Below
11	011	Register—See Below
11	100	Register—See Below
11	101	Register—See Below
11	110	Register—See Below
11	111	Register—See Below

Register Specified by reg or r/m During 16-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m During 32-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Encoding of 32-Bit Address Mode (mod r/m byte and s-i-b byte present):

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

Note: Mod field in mod r/m byte; ss, index, and 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 (see note)
101	EBP
110	ESI
111	EDI

Note: When index field is 100, indicating no index register, then ss field must equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

Encoding of Operation Direction (d) Field

In many two-operand instructions, the d field is present to indicate which operand is considered the source and which is the destination.

d	Direction of Operation
0	Register/Memory ⇐ Register reg Field indicates Source Operand; mod r/m or mod ss index base indicates Destination Operand.
1	Register ⇐ Register/Memory reg Field indicates Destination Operand; mod r/m or mod ss index base indicates Source Operand.

Encoding of Sign-Extend (s) Field

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

s	Effect on Immediate Data8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extended Data8 to Fill 16-Bit or 32-Bit Destination	None

Encoding of Conditional Test (tttn) Field

For the conditional instructions (conditional jumps and set on condition), tttn is encoded with n indicating to use the condition (n = 0), or its negation (n = 1), and ttt giving the condition to test.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	No 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/Not Greater Than	1110
NLE/G	Not Less Than or Equal/Greater Than	1111

Encoding of Control or Debug or Test Register (eee) Field

For the loading and storing of the Control, Debug, and Test registers.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	CR2
011	CR3
Do not use any other encoding	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding	

When Interpreted as Test Register Field

eee Code	Reg Name
110	TR6
111	TR7
Do not use any other encoding	

PRELIMINARY

Am386™DXLV

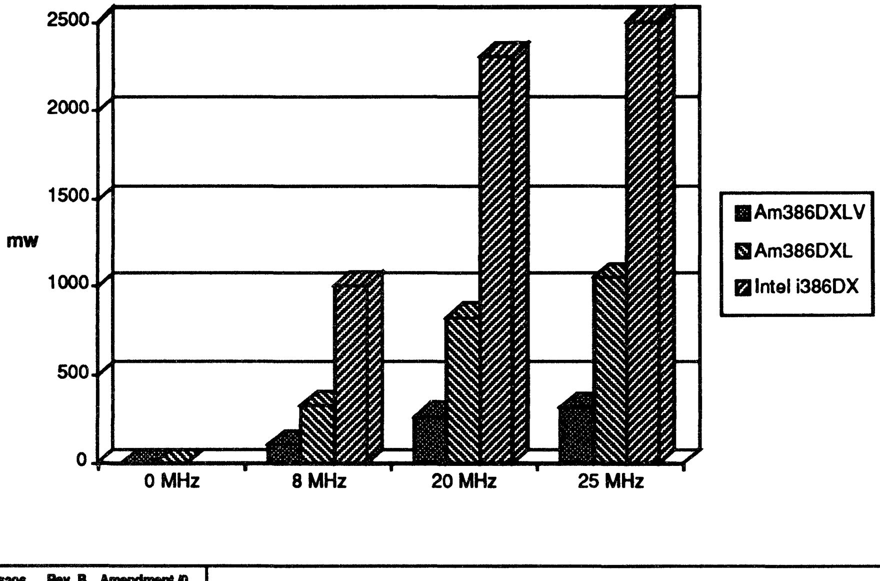
High-Performance, Low-Voltage, 32-Bit Microprocessor

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **Operating voltage range 3.0 V to 5.5 V—Ideal for portable PC applications**
  - —25 MHz operating frequency for 3.0 V to 5.5 V, 33 MHz operating frequency for 4.5 V to 5.5 V
  - —2X improvement in battery life over existing 5 V designs
  - —Wide range of chipsets and other logic available for 3 V systems with support for Standby Mode operation
  - —True static design for long battery life
  - —Power consumption 85% lower than Intel i386DX, 65% lower than Am386DXL processor
  - —Performance on demand (0 to 33 MHz)
- **SMM (System Management Mode) for system and power management**
  - —SMI (System Management Interrupt) for power management independent of processor operating mode and operating system
  - —SMI coupled with I/O instruction break feature provides transparent power off and auto resume of peripherals which may not be "power aware"
  - —SMI is non-maskable and has higher priority than NMI
  - —Automatic save and restore of the microprocessor state
  - —Wide range of chipsets supporting SMM available to allow product differentiation
- **Ideal for desktop PCs**
  - —Lower heat dissipation due to lower operating voltage allows elimination of fan
- **"Float" input to facilitate system debug and test**
- **Compatible with 386DX systems and software**
- **Supports 387DX-compatible math coprocessors**
- **132-pin PQFP package with optional protective ring for better lead coplanarity**
- **AMD® advanced 0.8 micron CMOS technology**

Typical Power Consumption



Publication # 16306 Rev. B Amendment /0
Issue Date: December 1991

GENERAL DESCRIPTION

The Am386DXLV microprocessor is a low-voltage, true static implementation of the Intel i386DX microprocessor. The operating voltage range is 3.0 V to 5.5 V. The low-voltage operation makes it ideal for both desktop and battery-powered portable personal computers. For desktop PCs, low heat dissipation allows the system designers to remove or reduce the size and cost of the system cooling fan. The Am386DXLV microprocessor operates at a maximum speed of 25 MHz from 3.0 to 5.5 V and at a maximum speed of 33 MHz from 4.5 to 5.5 V.

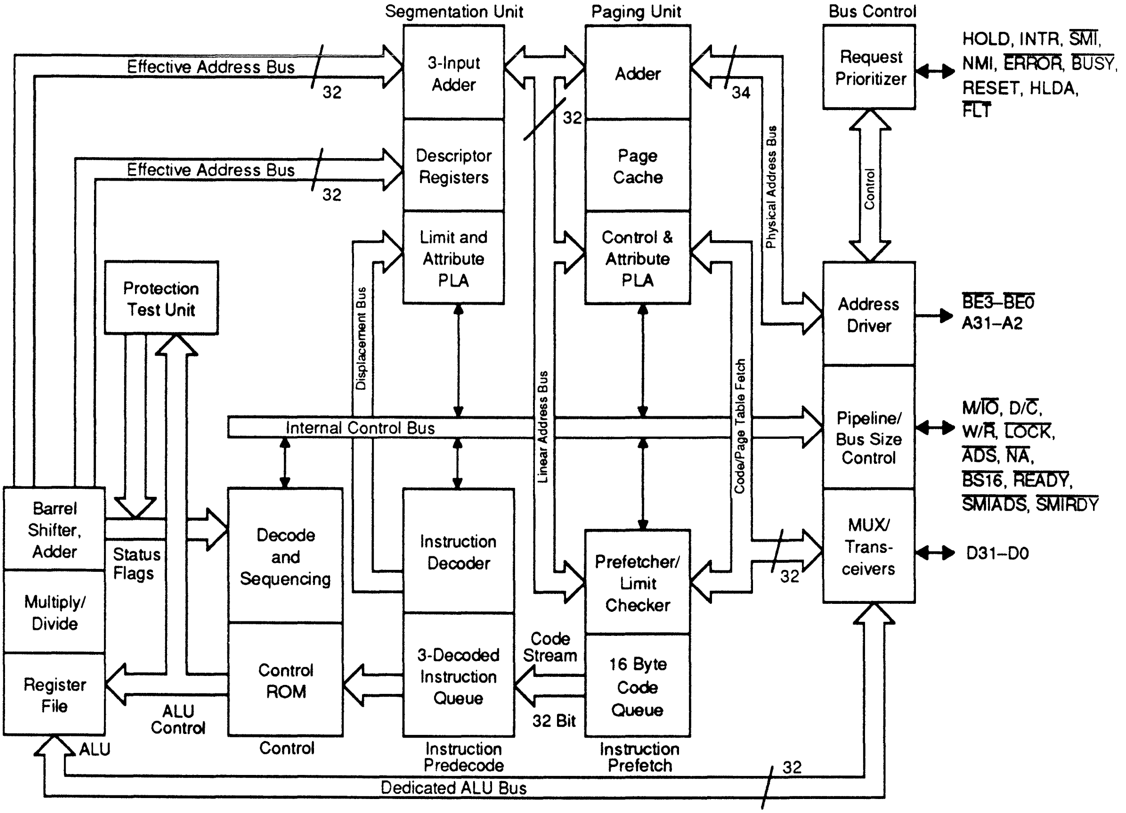
The Am386DXLV microprocessor's lower operating voltage and true static design enables longer battery life and/or lower weight for portable applications. At 25 MHz, this device has 80% lower operating Icc than the Intel i386DX. Lowering typical operating voltage from 5.0 V to 3.3 V doubles the battery life. Standby Mode allows the Am386DXLV microprocessor to be clocked down to 0 MHz (DC) and retain full register contents. In Standby Mode, typical current draw is 0.01 mA, a greater than 1000X reduction in power consumption versus the Intel i386DX or Intel i386SX.

The Am386DXLV processor is available in a small footprint 132-pin Plastic Quad Flat Pack (PQFP) package. This surface-mount package is 40% smaller than a PGA package, allowing smaller lower-cost board designs without the need for a socket.

Additionally, the Am386DXLV processors comes with SMM for system and power management. SMI is a nonmaskable, higher priority interrupt than NMI and has its own code space (1 Mb). SMI can be coupled with the I/O instruction break feature to implement transparent power management of peripherals. SMM can be used by system designers to implement system and power management code independent of the operating system or the Processor Mode.

The Am386DXLV processor incorporates a float pin that places all outputs in a three-state mode to facilitate board test and debug.

BLOCK DIAGRAM



15021B–001

FUNCTIONAL DESCRIPTION

Benefits of Lower Operating Voltage

The Am386DXLV microprocessor has an operating voltage range of 3.0 V to 5.5 V. Low voltage allows for lower operating power consumption, longer battery life, and/or smaller batteries for notebook applications.

Because power is proportional to the square of voltage, reduction of the supply voltage from 5.0 V to 3.3 V reduces power consumption by 56%. This directly translates to a doubling of battery life for portable applications. Lower power consumption can also be used to reduce the size and weight of the battery. Thus, 3 V designs facilitate a reduction in the form factor. For desktop PCs, low power consumption means elimination of the cooling fan, thus reducing the size and noise of the PC.

A lower operating voltage results in a reduction of I/O voltage swings. This reduces noise generation providing a less hostile environment for board design. It also reduces electromagnetic radiation noise making it easier to obtain FCC approval.

SMM–System Management Mode

The Am386DXLV processor has a new System Management Mode (SMM) for system and power management. This mode consists of two features: System Management Interrupt (SMI) and I/O instruction break.

SMI–System Management Interrupt

SMI is implemented through the use of special bus interface pins. This interrupt method can be used to perform system management functions such as power management independent of Processor Operating Mode (Real, Protected, or Virtual 86 Modes).

SMI can also be invoked in software. This allows system software to communicate with SMI power management code. In addition, an instruction called UMOV allows data transfers between SMI and normal system memory spaces.

Activating the $\overline{\text{SMI}}$ pin invokes a sequence which saves the operating state of the processor into a separate SMM memory space, independent of the main system memory. After the state is saved, the processor is forced into Real Mode and begins execution at address FFFFFFF0h where a far jump to the SMM code is executed. This Real Mode code can perform its system management function and then resume execution of the normal system software by executing a special opcode sequence which will reload the saved processor state and continue execution in the main system memory space. See Figure 1 for a general flowchart of an SMM operation.

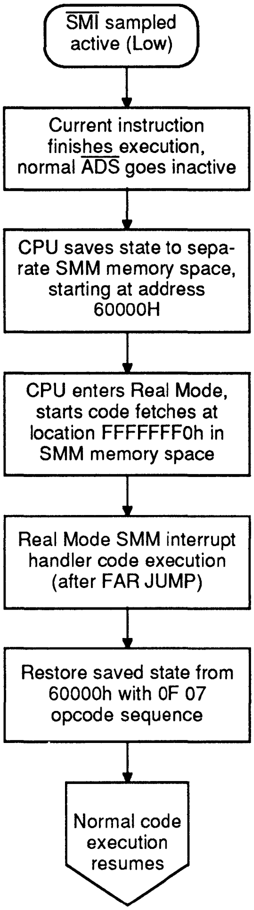

$\overline{\text{SMI}}$ sampled active (Low)
Current instruction finishes execution, normal $\overline{\text{ADS}}$ goes inactive
CPU saves state to separate SMM memory space, starting at address 60000H
CPU enters Real Mode, starts code fetches at location FFFFFFF0h in SMM memory space
Real Mode SMM interrupt handler code execution (after FAR JUMP)
Restore saved state from 60000h with 0F 07 opcode sequence
Normal code execution resumes

Figure 1. SMM Flow

CPU Interface—Pin Functions

The CPU interface for SMM consists of three pins dedicated to the SMI function. One pin, $\overline{\text{SMI}}$, is the new interrupt input. The other two pins, $\overline{\text{SMIADS}}$ and $\overline{\text{SMIRDY}}$, provide the control signals necessary for the separate SMI Mode memory space.

SMM Operation

The execution of a System Management Interrupt has four distinct phases: the initiation of the interrupt via $\overline{\text{SMI}}$, a processor state save, execution of the SMM interrupt code, and a processor state restore (to resume normal operation).

Interrupt Initiation

A System Management Interrupt is initiated by the driving of an active Low pulse on the SMI pin of at least four CLK2 periods. This pulse period will ensure recognition of the interrupt. The CPU will drive the SMI pin active after the completion of the current operation (active bus cycle, instruction execution, or both). The active drive of the pin by the CPU will be released at the end of the interrupt routine, following the last register read of the saved state.

An SMI cannot be masked off by the CPU, and it will always be recognized by the CPU, regardless of operating mode. This includes the Real, Protected, and Virtual 8086 Modes of the processor.

While the CPU is in SMI Mode, a bus hold request via the HOLD pin will be granted. The HLDA pin will go active after bus release and the SMIADS pin will float along with the other pins that normally float during a bus hold cycle.

Processor State Save

The first set of SMM bus transfer cycles after the CPU's recognition of an active SMI will be the processor saving its state to an external RAM array in a separate address space from main system memory. This is accomplished by using the SMIADS and SMIRDY pins for initiation and termination of bus cycles, instead of the ADS and READY pins. The 32-bit addresses to which the CPU saves its state are 60000H–600C8h and 60100h–60124h. These are fixed address locations for each register saved.

The value of BS16 and NA will be ignored during the state save, only full 32-bit, non-pipelined cycles are generated for the state save cycles. In a zero wait state memory implementation, it will take approximately 630 CLK2 cycles to complete the save state operation. There are 61 data transfer cycles.

SMI Code Execution

After the processor state is saved to the separate SMM memory space, the execution of the SMI interrupt routine code begins. The processor enters Real Mode, sets most of the register values to "reset" values (those values normally seen after a CPU reset), and begins fetching code from address FFFFFFF0h in the separate SMM memory space. Normally, the first thing the interrupt routine code will do is a FAR JUMP to the Real Mode entry point for the SMI interrupt routine, which is also in SMM memory space.

Any Real Mode interrupt routine code can be executed, with the obvious exception of normal interrupt routines (which are deferred). The SMM code can be located anywhere within the 1 Mb Real Mode address space, except where the processor state is saved. I/O cycles, as a result of the IN, OUT, INS, and OUTS instructions, will go to the normal address space, utilizing the normal ADS and READY bus interface signals. This facilitates power management code manipulating system hardware registers as needed through the standard I/O subsystem; a separate I/O space does not need to be implemented.

Processor State Restore (Resuming Normal Execution)

Returning to normal code execution in the main system memory, including restoring the Processor Operating Mode, is accomplished by executing a special code sequence. This code invokes a restore CPU state operation which reloads the CPU registers from the saved data in the RAM controlled by SMIADS and SMIRDY.

The ES:EDI register pair must point to physical address 60000h.Then the special opcode sequence 0Fh 07h should be executed to start the restore state operation. After completion of the restore state operation, the SMI pin will be deactivated by the CPU and normal code execution will continue at the point that it left off before the SMI occurred.

In a zero wait state memory implementation, it will take approximately 574 CLK2 cycles to complete the restore state operation. There are 61 data transfer cycles.

Software Features

There are several features of the SMI function that provide support for special operations during the execution of the system's software. These features involve the execution of reserved opcodes to induce specific SMI related operations.

Software SMI Generation

Besides hardware initiation of the System Management Interrupt via the SMI pin, there is also a software induced SMI mechanism. Generating a soft SMI involves setting a control bit in Debug Register 7 (DR7) and executing a reserved opcode (0F1H).

The functional sequence of the software-based SMI is identical to the hardware-based SMI with the exception that the SMI pin is not initially driven active by an external source. Upon execution of a soft SMI opcode, the SMI pin is driven active (Low) by the processor before the save state operation begins.

Memory Transfers to Main System Memory

While executing an SMI routine, the interrupt code can initiate memory data reads and writes to the main system memory using the normal ADS and READY pins. This is accomplished by using reserved opcodes that are special forms of the MOV instruction (called UMOV). The UMOV opcodes can move byte, word, or doubleword register operands to or from main system memory. Multiple data transfers using the normal ADS and READY pins will occur if the operands are misaligned relative to the effective address used. The UMOV opcodes are 0F10h, 0F11h, 0F12h, and 0F13h.

The UMOV instruction can use any of the 386 addressing modes, as specified in the ModR/M byte of the opcode. Note that the 16- and 32-bit versions are the same opcodes with the exception of the 066h operand size prefix. The $\overline{\text{BS16}}$ line is recognized during the normal memory space data transfer(s) initiated by these instructions.

I/O Instruction Break

The Am386DXLV processor has an I/O instruction break feature that allows the system logic to implement I/O trapping for peripheral devices. To enable the I/O instruction break feature, $\overline{\text{IIBEN}}$ must first be asserted active Low. On detecting an I/O instruction, the processor will prevent the execution unit from executing further instructions until $\overline{\text{READY}}$ is driven active Low by the system. Once $\overline{\text{READY}}$ is driven active, the execution unit will either immediately respond to any active interrupt request or continue executing instructions following the I/O instruction that caused the break.

The I/O instruction break feature can be used to allow system logic to implement I/O trapping for peripheral devices. On sensing an I/O instruction, the system can drive the $\overline{\text{SMI}}$ (or NMI or INTR) active before driving $\overline{\text{READY}}$ active. This ensures that the SMI service routine is executed immediately following the I/O instruction that caused the break. (If the I/O instruction break feature is not enabled via $\overline{\text{IIBEN}}$, several instructions after the I/O instruction that caused the break will execute before the SMI service routine is executed.) The SMI service routine can access the peripheral for which $\overline{\text{SMI}}$ was asserted and modify its state.

The SMI service routine will normally return to the instruction following the I/O instruction that caused the break. By modifying the saved state instruction pointer, the routine can choose to return to the I/O instruction that caused the break and re-execute that instruction. The default is to return to the following instruction (except for REP I/O strings). To re-execute the I/O instruction that caused the break, the SMI service routine must copy the I/O instruction pointer over the default pointer. This feature is particularly useful when an application program requests an access to a peripheral that has been powered down. The SMI service routine can restore power to the peripheral and initiate a re-execution sequence transparent to the application program. This re-execution feature should only be used if the $\overline{\text{SMI}}$ is in response to an I/O trap with $\overline{\text{IIBEN}}$ active.

Note that the I/O instruction break feature is not enabled for memory mapped I/O devices or for 80387 bus cycles even if $\overline{\text{IIBEN}}$ is active.

I/O Instruction Break Timing

The I/O Instruction Break feature requires that $\overline{\text{SMI}}$ be sampled active (Low) by the processor at least three CLK2 edges before the CLK2 edge that ends the I/O cycle with an active $\overline{\text{READY}}$ signal. This timing applies for both pipelined and non-pipelined cycles. If this timing constraint is not met, the next instruction may be executed by the internal execution unit prior to entering SMI Mode, but the $\overline{\text{SMI}}$ will be recognized eventually.

Depending on the state of the prefetch queue at the time that $\overline{\text{SMI}}$ is asserted, instruction fetch cycles may occur on the normal $\overline{\text{ADS}}$ interface before the SMI save state process begins with the assertion of $\overline{\text{SMIADS}}$. However, this fetched code will not be executed.

True Static Operation

The Am386DXLV microprocessor incorporates a true static design. Unlike dynamic circuit design, the Am386DXLV device eliminates the minimum operating frequency restriction. It may be clocked from its maximum speed of 33 MHz all the way down to 0 MHz (DC). System designers can use this feature to design true 32-bit battery-powered portable PCs with long battery life.

Standby Mode

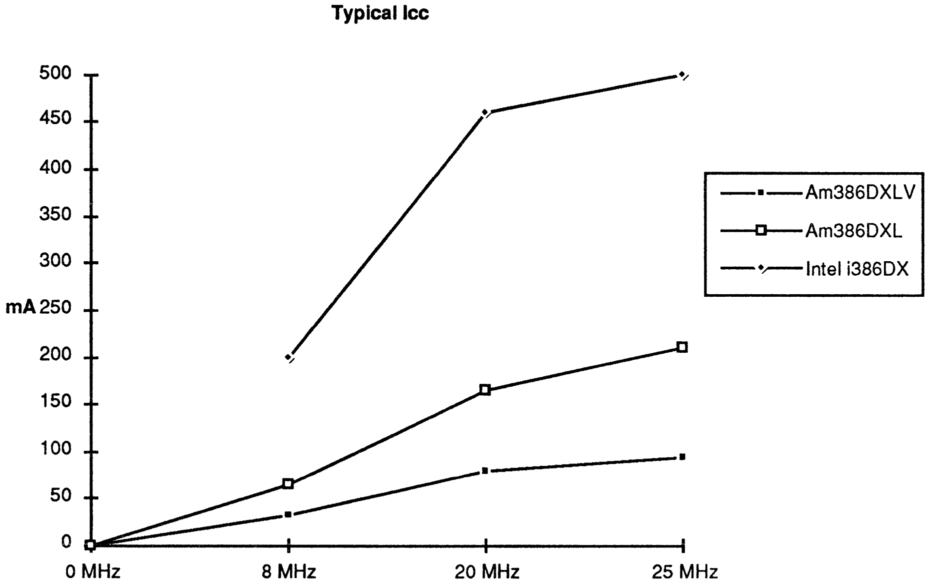
This true static design allows for a Standby Mode. At any of its operating speeds (33 MHz to 0 MHz), the Am386DXLV microprocessor will retain its state (i.e., the contents of all of its registers). By shutting off the clock completely, the device enters Standby Mode. Since power consumption is a function of clock frequency, operating power consumption is reduced as the frequency is lowered. In Standby Mode, typical current draw is reduced to less than 0.01 mA at DC.

Not only does this feature save battery life, but it also simplifies the design of power-conscious notebook computers in the following ways.

1. Eliminates the need for software in BIOS to save and restore the contents of registers.
2. Allows simpler circuitry to control stopping of the clock since the system does not need to know what state the processor is in.

Lower Operating Icc

True static design also allows lower operating Icc when operating at any speed. See the following graph for typical current at operating speeds.



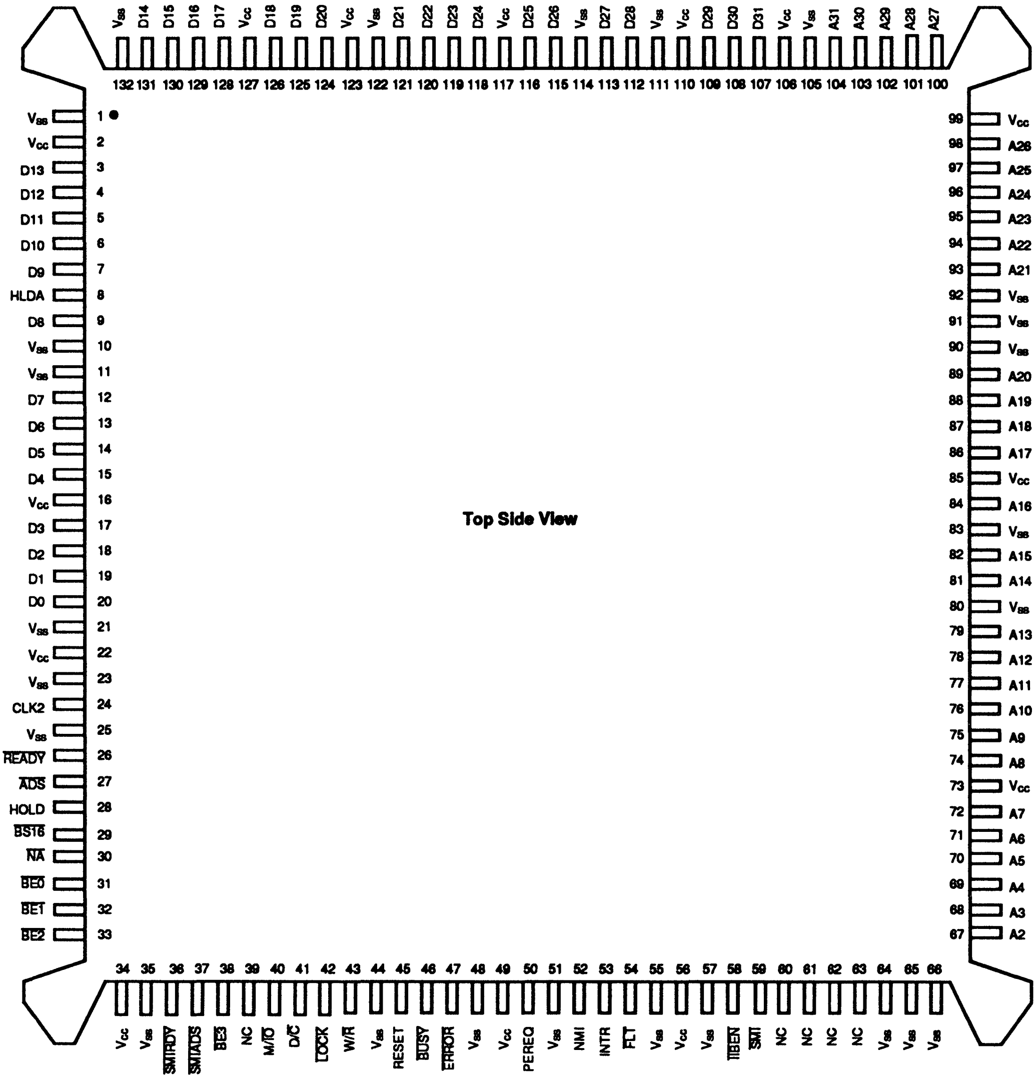
Performance on Demand

The Am386DXLV microprocessor retains its state at any speed from 0 MHz (DC) to its maximum operating speed. With this feature, system designers may vary the operating speed of the system to extend the battery life in portable systems.

For example, the system could operate at low speeds during inactivity or polling operations. However, upon interrupt, the system clock can be increased up to its maximum speed. After a user-defined time-out period, the system can be returned to a low (or a 0 MHz) operating speed without losing its state. This design maximizes battery life while achieving optimal performance.

CONNECTION DIAGRAMS

132-Lead Plastic Quad Flat Pack (PQFP) Package



15022B-002

Note: Pin 1 is marked for orientation.

PQFP Pin Designations (Functional Grouping)

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A2	67	A24	96	D6	13	D28	112	V_{CC}	16	V_{SS}	51
A3	68	A25	97	D7	12	D29	109		22		55
A4	69	A26	98	D8	9	D30	108		34		57
A5	70	A27	100	D9	7	D31	107		49		64
A6	71	A28	101	D10	6	D/$\overline{\text{C}}$	41		56		65
A7	72	A29	102	D11	5	$\overline{\text{ERROR}}$	47		73		66
A8	74	A30	103	D12	4	$\overline{\text{FLT}}$	54		85		80
A9	75	A31	104	D13	3	HLDA	8		99		83
A10	76	$\overline{\text{ADS}}$	27	D14	131	HOLD	28		106		90
A11	77	$\overline{\text{BE0}}$	31	D15	130	$\overline{\text{IIBEN}}$	58		110		91
A12	78	$\overline{\text{BE1}}$	32	D16	129	INTR	53		117		92
A13	79	$\overline{\text{BE2}}$	33	D17	128	$\overline{\text{LOCK}}$	42		123		105
A14	81	$\overline{\text{BE3}}$	38	D18	126	M/$\overline{\text{IO}}$	40		127		111
A15	82	$\overline{\text{BS16}}$	29	D19	125	$\overline{\text{NA}}$	30	V_{SS}	1		114
A16	84	$\overline{\text{BUSY}}$	46	D20	124	NMI	52		10		122
A17	86	CLK2	24	D21	121	PEREQ	50		11		132
A18	87	D0	20	D22	120	$\overline{\text{READY}}$	26		21	W/$\overline{\text{R}}$	43
A19	88	D1	19	D23	119	RESET	45		23	NC	39
A20	89	D2	18	D24	118	$\overline{\text{SMI}}$	59		25		60
A21	93	D3	17	D25	116	$\overline{\text{SMIADS}}$	37		35		61
A22	94	D4	15	D26	115	$\overline{\text{SMIRDY}}$	36		44		62
A23	95	D5	14	D27	113	V_{CC}	2		48		63

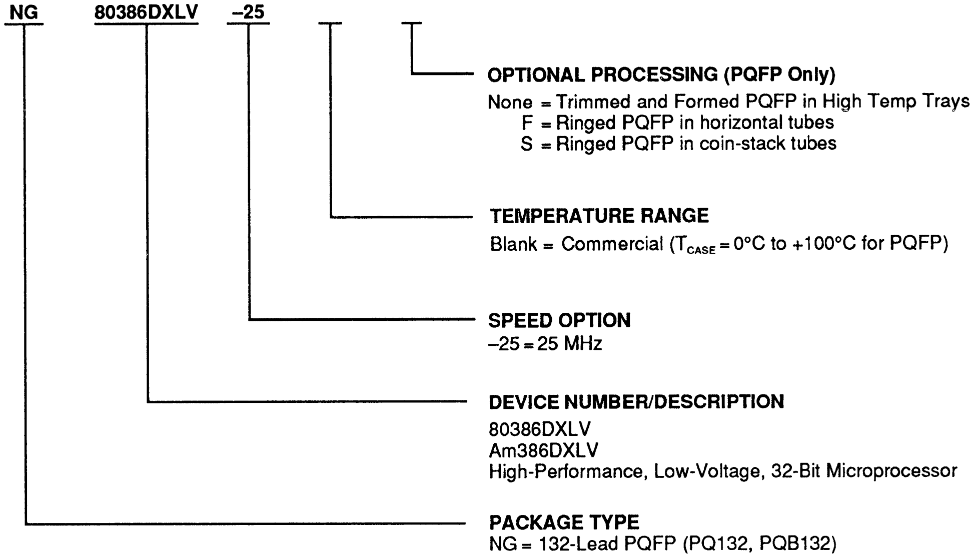
PQFP Pin Designations (Sorted by Pin No.)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	V_{SS}	23	V_{SS}	45	RESET	67	A2	89	A20	111	V_{SS}
2	V_{CC}	24	CLK2	46	$\overline{\text{BUSY}}$	68	A3	90	V_{SS}	112	D28
3	D13	25	V_{SS}	47	$\overline{\text{ERROR}}$	69	A4	91	V_{SS}	113	D27
4	D12	26	$\overline{\text{READY}}$	48	V_{SS}	70	A5	92	V_{SS}	114	VSS
5	D11	27	$\overline{\text{ADS}}$	49	V_{CC}	71	A6	93	A21	115	D26
6	D10	28	HOLD	50	PEREQ	72	A7	94	A22	116	D25
7	D9	29	$\overline{\text{BS16}}$	51	V_{SS}	73	V_{CC}	95	A23	117	V_{CC}
8	HLDA	30	$\overline{\text{NA}}$	52	NMI	74	A8	96	A24	118	D24
9	D8	31	$\overline{\text{BE0}}$	53	INTR	75	A9	97	A25	119	D23
10	V_{SS}	32	$\overline{\text{BE1}}$	54	$\overline{\text{FLT}}$	76	A10	98	A26	120	D22
11	V_{SS}	33	$\overline{\text{BE2}}$	55	V_{SS}	77	A11	99	V_{CC}	121	D21
12	D7	34	V_{CC}	56	V_{CC}	78	A12	100	A27	122	V_{SS}
13	D6	35	V_{SS}	57	V_{SS}	79	A13	101	A28	123	V_{CC}
14	A5	36	$\overline{\text{SMIRDY}}$	58	$\overline{\text{IIBEN}}$	80	V_{SS}	102	A29	124	D20
15	D4	37	$\overline{\text{SMIADS}}$	59	$\overline{\text{SMI}}$	81	A14	103	A30	125	D19
16	V_{CC}	38	$\overline{\text{BE3}}$	60	NC	82	A15	104	A31	126	D18
17	D3	39	NC	61	NC	83	V_{SS}	105	V_{SS}	127	V_{CC}
18	D2	40	M/$\overline{\text{IO}}$	62	NC	84	A16	106	V_{CC}	128	D17
19	D1	41	D/$\overline{\text{C}}$	63	NC	85	V_{CC}	107	D31	129	D16
20	D0	42	$\overline{\text{LOCK}}$	64	V_{SS}	86	A17	108	D30	130	D15
21	V_{SS}	43	W/$\overline{\text{R}}$	65	V_{SS}	87	A18	109	D29	131	D14
22	V_{CC}	44	V_{SS}	66	V_{SS}	88	A19	110	V_{CC}	132	V_{SS}

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



NG 80386DXLV –25

OPTIONAL PROCESSING (PQFP Only)
None = Trimmed and Formed PQFP in High Temp Trays
F = Ringed PQFP in horizontal tubes
S = Ringed PQFP in coin-stack tubes

TEMPERATURE RANGE
Blank = Commercial (T_{CASE} = 0°C to +100°C for PQFP)

SPEED OPTION
–25 = 25 MHz

DEVICE NUMBER/DESCRIPTION
80386DXLV
Am386DXLV
High-Performance, Low-Voltage, 32-Bit Microprocessor

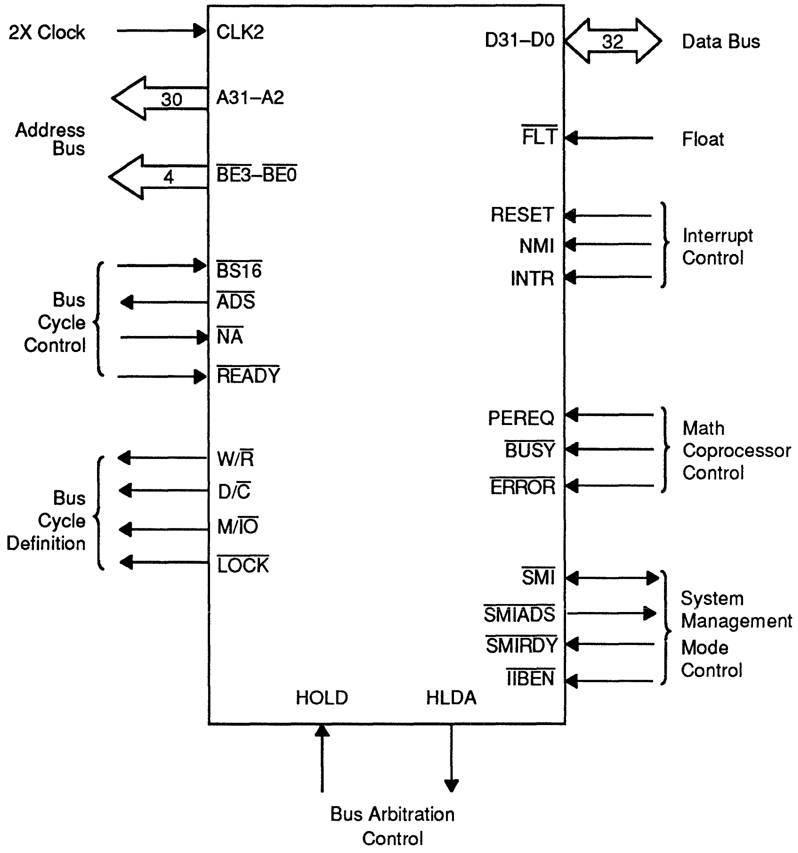
PACKAGE TYPE
NG = 132-Lead PQFP (PQ132, PQB132)

Valid Combinations		
NG	80386DXLV	–25 –25F –25S

Valid Combinations

Valid Combinations lists configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations.

LOGIC SYMBOL



15021B–003

PIN DESCRIPTION

A31–A2

Address Bus (Outputs)

Outputs physical memory or port I/O addresses.

$\overline{\text{ADS}}$

Address Status (Active Low; Output)

Indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, and A31–A2) are being driven at the Am386DXLV microprocessor pins.

$\overline{\text{BE3}}$–$\overline{\text{BE0}}$

Byte Enable (Active Low; Outputs)

Indicates which data bytes of the data bus take part in a bus cycle.

$\overline{\text{BS16}}$

Bus Size 16 (Active Low; Input)

Allows direct connection of 32-bit and 16-bit data buses. $\overline{\text{BS16}}$ has an internal pullup resistor.

$\overline{\text{BUSY}}$

Busy (Active Low; Input)

Signals a busy condition from a processor extension. $\overline{\text{BUSY}}$ has an internal pullup resistor.

CLK2

Clock (Input)

Provides the fundamental timing for the Am386DXLV microprocessor.

D31–D0

Data Bus (Inputs/Outputs)

Inputs data during memory, I/O, and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.

D/$\overline{\text{C}}$

Data/Control (Output)

A bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles (which are interrupt acknowledge, halt, and instruction fetching).

$\overline{\text{ERROR}}$

Error (Active Low; Input)

Signals an error condition from a processor extension. $\overline{\text{ERROR}}$ has an internal pullup resistor.

$\overline{\text{FLT}}$

Float (Active Low; Input)

An input signal which forces all bi-directional and output signals, including HLDA, to the three-state condition. $\overline{\text{FLT}}$ has an internal pullup resistor.

HLDA

Bus Hold Acknowledge (Active High; Output)

Indicates that the Am386DXLV microprocessor has surrendered control of its local bus to another bus master.

HOLD

Bus Hold Request (Active High; Input)

Allows another bus master to request control of the local bus.

$\overline{\text{IIBEN}}$

I/O Instruction Break Enable (Active Low; Input)

Enables the I/O instruction break feature. $\overline{\text{IIBEN}}$ has an internal pullup resistor.

INTR

Interrupt Request (Active High; Input)

A maskable input that signals the Am386DXLV microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

$\overline{\text{LOCK}}$

Bus Lock (Active Low; Output)

A bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active.

M/$\overline{\text{IO}}$

Memory I/O (Output)

A bus cycle definition pin that distinguishes memory cycles from input/output cycles.

$\overline{\text{NA}}$

Next Address (Active Low; Input)

Used to request address pipelining.

NC

No Connect

Should always remain unconnected. Connection of an NC pin may cause the processor to malfunction or be incompatible with future steppings of the Am386DXLV microprocessor.

NMI

Non-Maskable Interrupt Request (Active High; Input)

A non-maskable input that signals the Am386DXLV microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

PEREQ

Processor Extension Request (Active High; Input)

Indicates that the processor extension has data to be transferred by the Am386DXLV microprocessor. PEREQ has an internal pulldown resistor.

$\overline{\text{READY}}$

Bus Ready (Active Low; Input)

Terminates the bus cycle.

RESET

Reset (Active High; Input)

Suspends any operation in progress and places the Am386DXLV microprocessor in a known reset state.

$\overline{\text{SMI}}$

System Management Interrupt (Active Low; Input/Output)

A non-maskable interrupt pin which signals the Am386DXLV microprocessor to suspend execution and enter System Management Mode. $\overline{\text{SMI}}$ has an internal pullup resistor.

$\overline{\text{SMIADS}}$

SMI Address Status (Active Low, Three-State; Output)

Indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BE}}$3–$\overline{\text{BE}}$0, and A31–A2) are being driven at the Am386DXLV microprocessor pins while in System Management Mode.

$\overline{\text{SMIRDY}}$

SMI Ready (Active Low; Input)

This input terminates the current bus cycle to the SMM address space in the same manner the $\overline{\text{READY}}$ pin does for the Normal Mode address space. $\overline{\text{SMIRDY}}$ has an internal pullup resistor.

V_{CC}

System Power (Active High; Input)

Provides the DC supply input.

V_{SS}

System Ground (Input)

Provides 0-V connection from which all inputs and outputs are measured.

W/$\overline{\text{R}}$

Write/Read (Output)

A bus cycle definition pin that distinguishes write cycles from read cycles.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature . . . . –65°C to +150°C
Ambient Temperature Under Bias . . –65°C to +125°C

Stresses above those listed may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to ABSOLUTE MAXIMUM RATINGS for extended periods may affect device reliability.

OPERATING RANGES

Voltage on Other Pins . . . . –0.5 V to V_{CC} +0.5 V
Supply Voltage with Respect to V_{SS} . . –0.5 V to +7 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL Operating Ranges

V_{CC} = 3.0 V to 3.6 V; T_{CASE} = 0°C to +100°C

			Preliminary		
Symbol	Parameter Description	Notes	Min	Max	Unit
V_{IL}	Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.3	V
V_{ILC}	CLK2 Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IHC}	CLK2 Input High Voltage				
	25 MHz		V_{CC} – 0.6	V_{CC} + 0.3	V
V_{OL}	Output Low Voltage	(Note 5)			
	I_{OL} = 0.5 mA: A31–A2, D31–D0			0.2	V
	I_{OL} = 0.5 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, $\overline{\text{SMIADS}}$, HLDA			0.2	V
	I_{OL} = 2 mA: A31–A2, D31–D0			0.45	V
	I_{OL} = 2.5 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, $\overline{\text{SMIADS}}$, HLDA			0.45	V
V_{OH}	Output High Voltage	(Note 5)			
	I_{OH} = 0.1 mA: A31–A2, D31–D0		V_{CC} – 0.2		V
	I_{OH} = 0.1 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, $\overline{\text{SMIADS}}$, HLDA		V_{CC} – 0.2		V
	I_{OH} = 0.5 mA: A31–A2, D31–D0		V_{CC} – 0.45		V
	I_{OH} = 0.5 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, $\overline{\text{SMIADS}}$, HLDA		V_{CC} – 0.45		V
I_{LI}	Input Leakage Current (All pins except $\overline{\text{BS16}}$, PEREQ, $\overline{\text{IIBEN}}$ $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, $\overline{\text{ERROR}}$, $\overline{\text{SMI}}$, and $\overline{\text{SMIRDY}}$)	0 V ≤ V_{IN} ≤ V_{CC}		±15	µA
I_{IH}	Input Leakage Current (PEREQ Pin)	V_{IH} = V_{CC} – 0.1 V V_{IH} = 2.4 V (Note 2)		300 200	µA
I_{IL}	Input Leakage Current ($\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, $\overline{\text{ERROR}}$, $\overline{\text{SMI}}$, $\overline{\text{IIBEN}}$, and $\overline{\text{SMIRDY}}$)	V_{IL} = 0.1 V V_{IL} = 0.45 V (Note 3)		–300 –200	µA
I_{LO}	Output Leakage Current	0.1 V ≤ V_{OUT} ≤ V_{CC}		±15	µA
I_{CC}	Supply Current				
	CLK2 = 40 MHz: Oper. Freq. 20 MHz	I_{CC} Typ = 80		110	mA
	CLK2 = 50 MHz: Oper. Freq. 25 MHz	I_{CC} Typ = 95		135	mA
I_{CCSB}	Standby Current	I_{CCSB} Typ = 10		150	µA
C_{IN}	Input or I/O Capacitance	F_C = 1 MHz (Note 4)		10	pF
C_{OUT}	Output Capacitance	F_C = 1 MHz (Note 4)		12	pF
C_{CLK}	CLK2 Capacitance	F_C = 1 MHz (Note 4)		20	pF

Notes: 1. The Min value, –0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. $\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, $\overline{\text{ERROR}}$, $\overline{\text{SMI}}$, $\overline{\text{SMIRDY}}$, and $\overline{\text{IIBEN}}$ inputs each have an internal pullup resistor.
4. Not 100% tested.
5. Outputs are CMOS and will pull rail to rail if the load is not resistive.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature .... –65°C to +150°C
Ambient Temperature Under Bias .. –65°C to +125°C

Stresses above those listed may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to ABSOLUTE MAXIMUM RATINGS for extended periods may affect device reliability.

OPERATING RANGES

Voltage on Other Pins .... –0.5 V to Vcc +0.5 V
Supply Voltage with Respect
to Vss .... –0.5 V to +7 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL Operating Ranges

V_{CC} = 3.6 V to 5.5 V; T_{CASE} = 0°C to +100°C

			Preliminary		
Symbol	**Parameter Description**	**Notes**	**Min**	**Max**	**Unit**
V_{IL}	Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.3	V
V_{ILC}	CLK2 Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IHC}	CLK2 Input High Voltage 20 MHz 25, 33 MHz		 V_{CC} – 0.8 3.7	 V_{CC} + 0.3 V_{CC} + 0.3	 V V
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A31–A2, D31–D0 I_{OL} = 5 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, $\overline{\text{SMIADS}}$, HLDA	(Note 5)		 0.45 0.45	 V V
V_{OH}	Output High Voltage I_{OH} = 1 mA: A31–A2, D31–D0 I_{OH} = 0.9 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, $\overline{\text{SMIADS}}$, HLDA	(Note 5)	 2.4 2.4		 V V
I_{LI}	Input Leakage Current (All pins except $\overline{\text{BS16}}$, PEREQ, $\overline{\text{IIBEN}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, $\overline{\text{SMI}}$, $\overline{\text{SMIRDY}}$, and $\overline{\text{ERROR}}$)	0 V ≤ V_{IN} ≤ V_{CC}		±15	µA
I_{IH}	Input Leakage Current (PEREQ Pin)	V_{IH} = 2.4 V (Note 2)		200	µA
I_{IL}	Input Leakage Current ($\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, $\overline{\text{SMI}}$, $\overline{\text{SMIRDY}}$, $\overline{\text{IIBEN}}$, and $\overline{\text{ERROR}}$)	V_{IL} = 0.45 (Note 3)		–400	µA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		±15	µA
I_{CC}	Supply Current CLK2 = 40 MHz: Oper. Freq. 20 MHz CLK2 = 50 MHz: Oper. Freq. 25 MHz CLK2 = 66 MHz: Oper. Freq. 33 MHz	 I_{CC} Typ = 165 I_{CC} Typ = 210 I_{CC} Typ = 275		 200 250 330	 mA mA mA
I_{CCSB}	Standby Current	I_{CCSB} Typ = 0.02 mA		150	µA
C_{IN}	Input or I/O Capacitance	F_C = 1 MHz (Note 4)		10	pF
C_{OUT}	Output Capacitance	F_C = 1 MHz (Note 4)		12	pF
C_{CLK}	CLK2 Capacitance	F_C = 1 MHz (Note 4)		20	pF

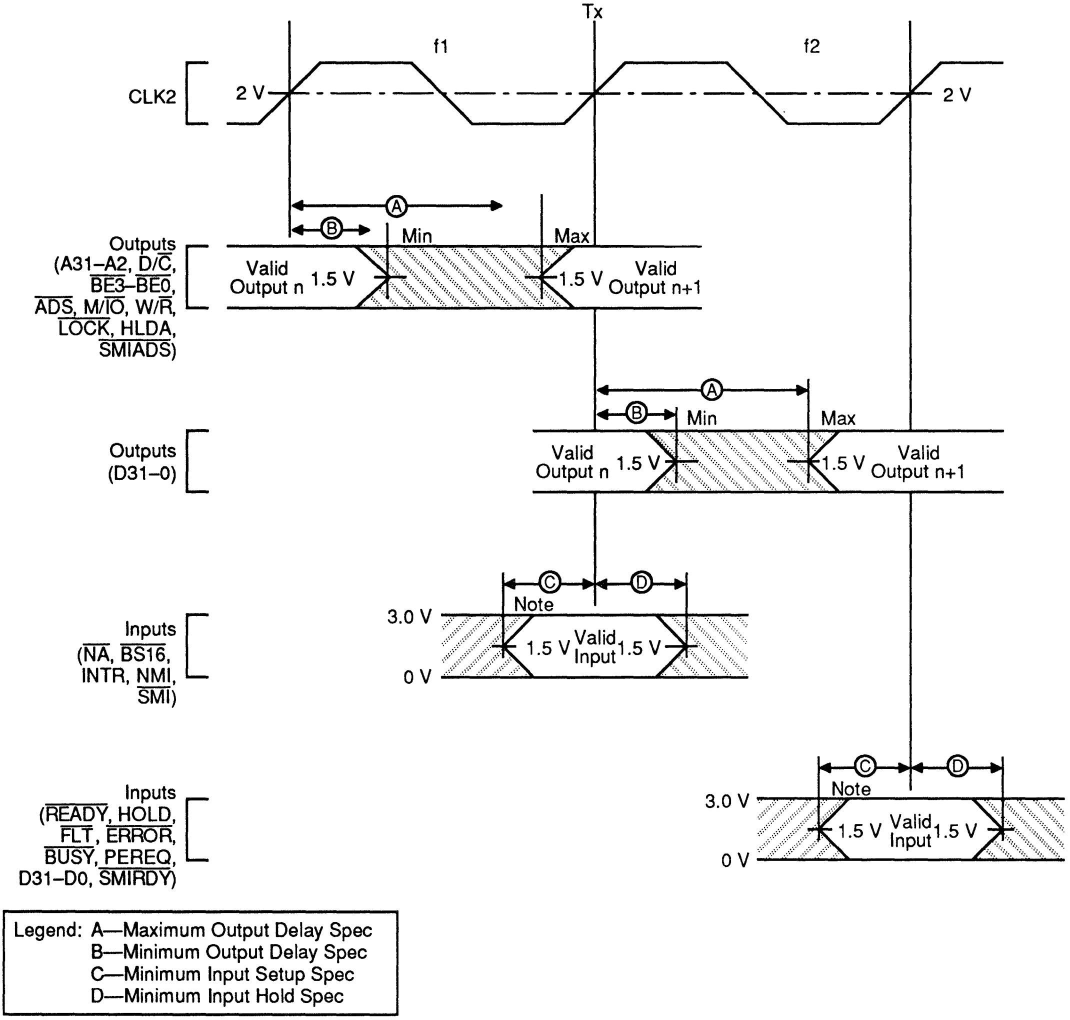
Notes: 1. The Min value, –0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. $\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$, $\overline{\text{ERROR}}$, $\overline{\text{SMI}}$, $\overline{\text{SMIRDY}}$, and $\overline{\text{IIBEN}}$ inputs each have an internal pullup resistor.
4. Not 100% tested.
5. Outputs are CMOS and will pull rail to rail if the load is not resistive.

SWITCHING CHARACTERISTICS

The switching characteristics consist of output delays, input setup requirements, and input hold requirements. All characteristics are relative to the CLK2 rising edge crossing the 2.0 V level.

Switching characteristic measurement is defined by Figure 2. Inputs must be driven to the voltage levels indicated by this diagram. Am386DXLV CPU output delays are specified with minimum and maximum limits measured as shown. The minimum Am386DXLV microprocessor delay times are hold times provided to external circuitry. Am386DXLV microprocessor input setup and hold time are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct Am386DXLV microprocessor operation.

Outputs W/$\overline{\mathrm{R}}$, D/$\overline{\mathrm{C}}$, M/$\overline{\mathrm{IO}}$, $\overline{\mathrm{LOCK}}$, $\overline{\mathrm{BE3}}$–$\overline{\mathrm{BE0}}$, $\overline{\mathrm{ADS}}$, A31–A2, HLDA, and $\overline{\mathrm{SMIADS}}$ only change at the beginning of phase one. D31–D0 (write cycles) only change at the beginning of phase two. The $\overline{\mathrm{READY}}$, HOLD, $\overline{\mathrm{BUSY}}$, $\overline{\mathrm{ERROR}}$, PEREQ, $\overline{\mathrm{FLT}}$, D31–D0, and $\overline{\mathrm{SMIRDY}}$ (read cycles) inputs are sampled at the beginning of phase one. The $\overline{\mathrm{NA}}$, $\overline{\mathrm{BS16}}$, INTR, NMI, and $\overline{\mathrm{SMI}}$ inputs are sampled at the beginning of phase two.



Tx
f1
f2
CLK2
2 V
2 V
A
B
Min
Max
Outputs (A31–A2, D/C, BE3–BE0, ADS, M/IO, W/R, LOCK, HLDA, SMIADS)
Valid Output n
1.5 V
1.5 V
Valid Output n+1
A
B
Min
Max
Outputs (D31–0)
Valid Output n
1.5 V
1.5 V
Valid Output n+1
C
D
Note
3.0 V
0 V
Inputs (NA, BS16, INTR, NMI, SMI)
1.5 V
Valid Input
1.5 V
C
D
Note
3.0 V
0 V
Inputs (READY, HOLD, FLT, ERROR, BUSY, PEREQ, D31-D0, SMIRDY)
1.5 V
Valid Input
1.5 V
Legend: A—Maximum Output Delay Spec
B—Minimum Output Delay Spec
C—Minimum Input Setup Spec
D—Minimum Input Hold Spec

Note: Input waveforms have tr ≤ 2.0 ns from 0.8 V to 2.0 V.

15021B–071

Figure 2. Drive Levels and Measurement Points

SWITCHING CHARACTERISTICS over operating range at 33 MHz

V_{CC} = 4.5 V to 5.5 V; T_{CASE} = 0°C to +100°C

				Preliminary		
No.	Parameter Description	Notes	Ref Figures	Min	Max	Unit
	Operating Frequency	Half of CLK2 Freq		0	33.3	MHz
1	CLK2 Period		4	15.0		ns
2a	CLK2 High Time	at 2 V	4	6.25		ns
2b	CLK2 High Time	at 3.7 V	4	4.5		ns
3a	CLK2 Low Time	at 2 V	4	6.25		ns
3b	CLK2 Low Time	at 0.8 V	4	4.5		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	4		4	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	4		4	ns
6	A31–A2 Valid Delay	C_L = 50 pF	3, 6	4	15	ns
7	A31–A2 Float Delay	(Note 1)	10	4	20	ns
8	$\overline{\mathrm{BE3}}$–$\overline{\mathrm{BE0}}$, $\overline{\mathrm{LOCK}}$ Valid Delay	C_L = 50 pF	3, 6	4	15	ns
9	$\overline{\mathrm{BE3}}$–$\overline{\mathrm{BE0}}$, $\overline{\mathrm{LOCK}}$ Float Delay	(Note 1)	10	4	20	ns
10	W/$\overline{\mathrm{R}}$, M/$\overline{\mathrm{IO}}$, D/$\overline{\mathrm{C}}$ Valid Delay	C_L = 50 pF	3, 6	4	15	ns
10a	$\overline{\mathrm{ADS}}$ Valid Delay	C_L = 50 pF	2, 6	4	14.5	ns
10s	$\overline{\mathrm{SMIADS}}$ Valid Delay	C_L=50 pF	2, 6	4	23	ns
11	W/$\overline{\mathrm{R}}$, M/$\overline{\mathrm{IO}}$, D/$\overline{\mathrm{C}}$, $\overline{\mathrm{ADS}}$ Float Delay	(Note 1)	10	4	20	ns
11s	$\overline{\mathrm{SMIADS}}$ Float Delay	(Note 1)	10	4	20	ns
12	D31–D0 Write Data Valid Delay	C_L = 50 pF (Note 4)	6, 7, 9	7	24	ns
12a	D31–D0 Write Data Hold Time	C_L = 50 pF	3, 8	2		ns
13	D31–D0 Float Delay	(Note 1)	10	4	17	ns
14	HLDA Valid Delay	C_L = 50 pF	3, 10	4	20	ns
15	$\overline{\mathrm{NA}}$ Setup Time		5	5		ns
16	$\overline{\mathrm{NA}}$ Hold Time		5	2		ns
17	$\overline{\mathrm{BS16}}$ Setup Time		5	5		ns
18	$\overline{\mathrm{BS16}}$ Hold Time		5	2		ns
19	$\overline{\mathrm{READY}}$ Setup Time		5	7		ns
19s	$\overline{\mathrm{SMIRDY}}$ Setup Time		5	7		ns
20	$\overline{\mathrm{READY}}$ Hold Time		5	4		ns
20s	$\overline{\mathrm{SMIRDY}}$ Hold Time		5	4		ns
21	D31–D0 Read Setup Time		5	5		ns
22	D31–D0 Read Hold Time		5	3		ns
23	HOLD Setup Time		5	11		ns
24	HOLD Hold Time		5	2		ns
25	RESET Setup Time		11	5		ns
26	RESET Hold Time		11	2		ns
27	NMI, INTR Setup Time	(Note 2)	5	5		ns
27s	$\overline{\mathrm{SMI}}$ Setup Time		5	5		ns
28	NMI, INTR Hold Time	(Note 2)	5	5		ns
28s	$\overline{\mathrm{SMI}}$ Hold Time		5	5		ns
29	PEREQ, $\overline{\mathrm{ERROR}}$, $\overline{\mathrm{BUSY}}$, $\overline{\mathrm{FLT}}$ Setup Time	(Note 2)	5	5		ns
30	PEREQ, $\overline{\mathrm{ERROR}}$, $\overline{\mathrm{BUSY}}$, $\overline{\mathrm{FLT}}$ Hold Time	(Note 2)	5	4		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.
4. Min time not 100% tested.

SWITCHING CHARACTERISTICS over operating range at 25 MHz

V_{CC} = 3.0 V to 5.5 V; T_{CASE} = 0°C to +100°C

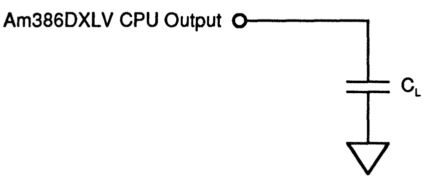
				Preliminary		
No.	**Parameter Description**	**Notes**	**Ref Figures**	**Min**	**Max**	**Unit**
	Operating Frequency	Half of CLK2 Freq		0	25	MHz
1	CLK2 Period		4	20		ns
2a	CLK2 High Time	at 2 V	4	7		ns
2b	CLK2 High Time	at 3.7 V	4	4		ns
3a	CLK2 Low Time	at 2 V	4	7		ns
3b	CLK2 Low Time	at 0.8 V	4	5		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	4		7	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	4		7	ns
6	A31–A2 Valid Delay	C_L = 50 pF	3, 6	4	21	ns
7	A31–A2 Float Delay	(Note 1)	10	4	30	ns
8	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$ Valid Delay	C_L = 50 pF	3, 6	4	24	ns
8a	$\overline{\text{LOCK}}$ Valid Delay	C_L = 50 pF	3, 6	4	21	ns
9	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	10	4	30	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	3, 6	4	21	ns
10s	$\overline{\text{SMIADS}}$ Valid Delay	C_L=50pF	3, 6	4	25	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	10	4	30	ns
11s	$\overline{\text{SMIADS}}$ Float Delay	(Note 1)	10	4	30	ns
12	D31–D0 Write Data Valid Delay	C_L = 50 pF	6, 7, 9	7	27	ns
12a	D31–D0 Write Data Hold Time	C_L = 50 pF	3, 8	2		ns
13	D31–D0 Float Delay	(Note 1)	10	4	22	ns
14	HLDA Valid Delay	C_L = 50 pF	3, 10	4	22	ns
15	$\overline{\text{NA}}$ Setup Time		5	7		ns
16	$\overline{\text{NA}}$ Hold Time		5	3		ns
17	$\overline{\text{BS16}}$ Setup Time		5	7		ns
18	$\overline{\text{BS16}}$ Hold Time		5	3		ns
19	$\overline{\text{READY}}$ Setup Time		5	9		ns
19s	$\overline{\text{SMIRDY}}$ Setup Time		5	9		ns
20	$\overline{\text{READY}}$ Hold Time		5	4		ns
20s	$\overline{\text{SMIRDY}}$ Hold Time		5	4		ns
21	D31–D0 Read Setup Time		5	7		ns
22	D31–D0 Read Hold Time		5	5		ns
23	HOLD Setup Time		5	15		ns
24	HOLD Hold Time		5	3		ns
25	RESET Setup Time		11	10		ns
26	RESET Hold Time		11	3		ns
27	NMI, INTR Setup Time	(Note 2)	5	6		ns
27s	$\overline{\text{SMI}}$ Setup Time		5	6		ns
28	NMI, INTR Hold Time	(Note 2)	5	6		ns
28s	$\overline{\text{SMI}}$ Hold Time		5	6		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Setup Time	(Note 2)	5	6		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Hold Time	(Note 2)	5	5		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

3. Rise and fall times are not tested.

SWITCHING CHARACTERISTICS (continued)



C_L includes all parasitic capacitances.

15021B–072

Figure 3. AC Test Load

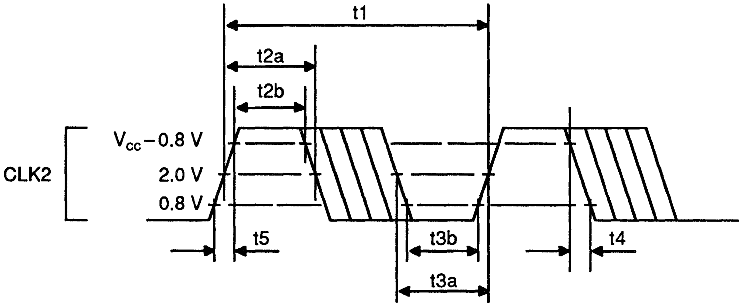


Figure 4. CLK2 Timing

15021B–073

SWITCHING WAVEFORMS

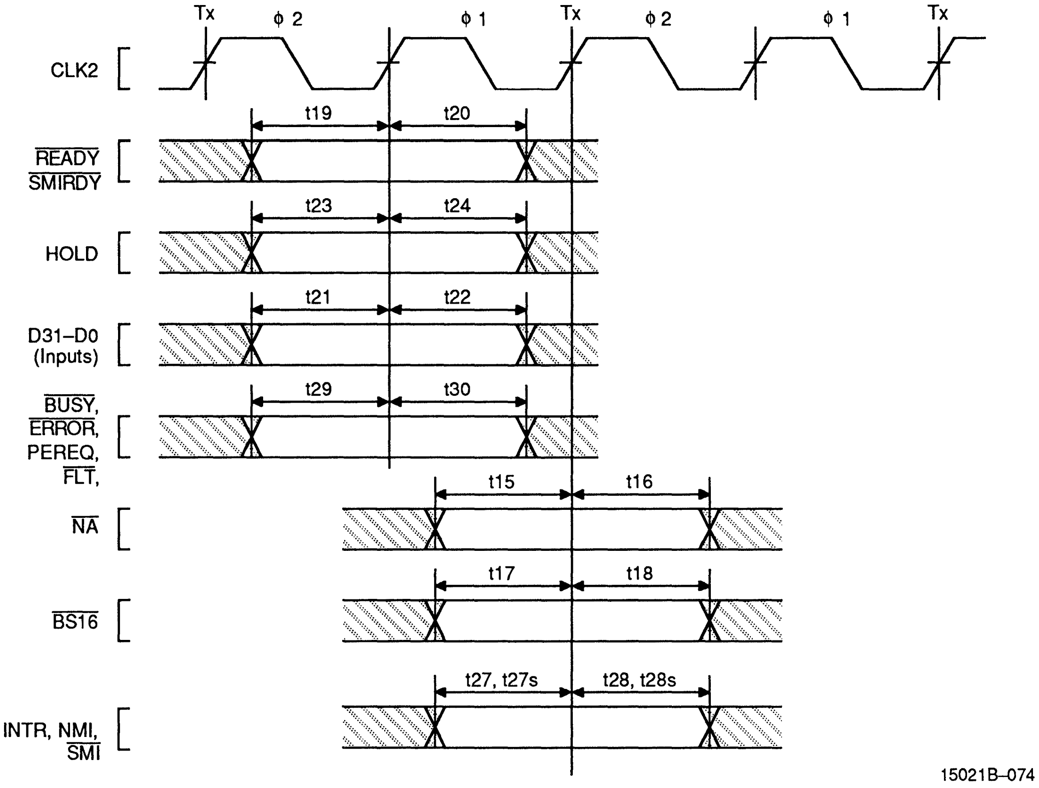


Figure 5. Input Setup and Hold Timing

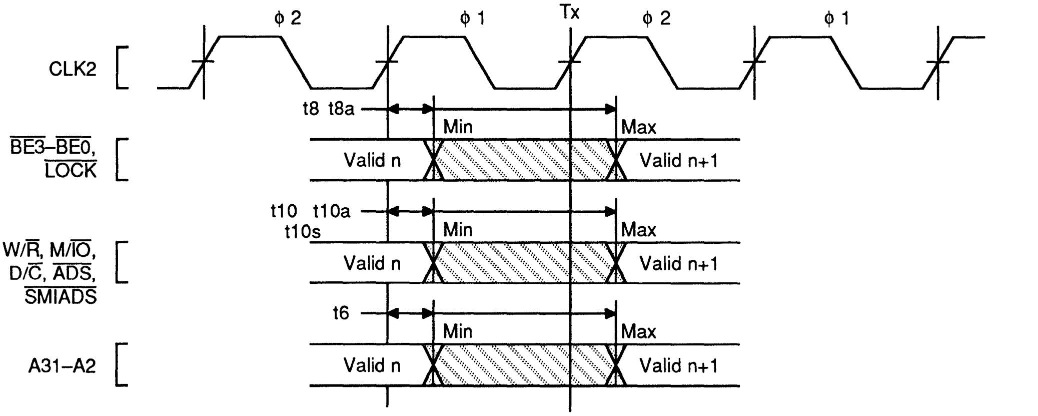


Figure 6. Output Valid Delay Timing

15021B–075

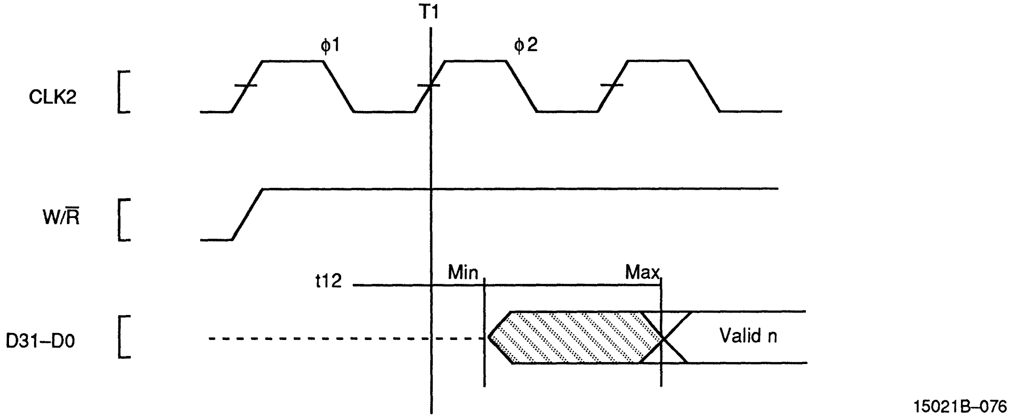


Figure 7. Write Data Valid Delay Timing (25 and 33 MHz)

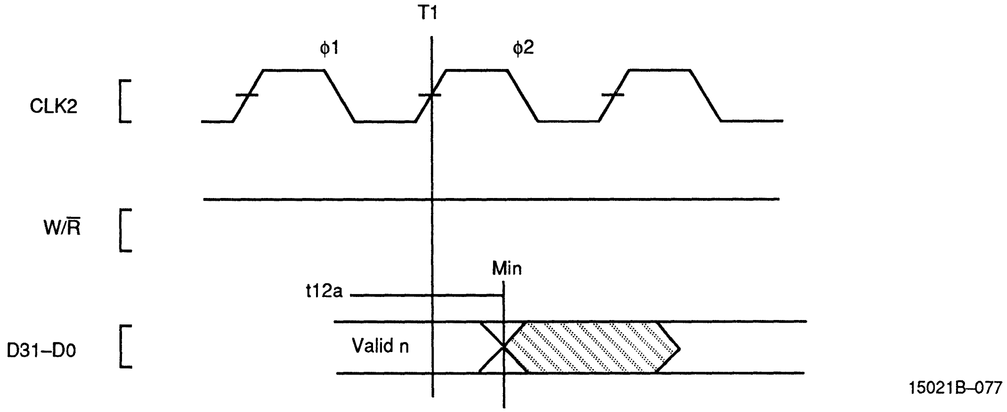


Figure 8. Write Data Hold Timing (25 and 33 MHz)

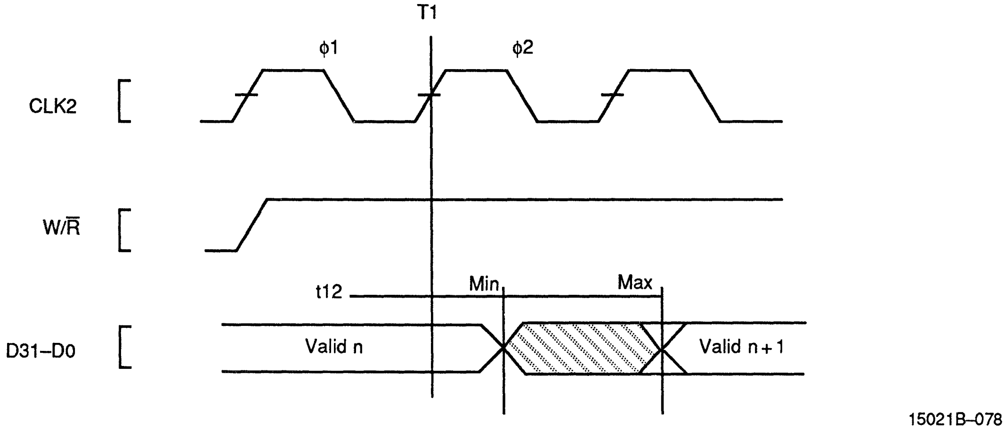


Figure 9. Write Data Valid Delay Timing (20 MHz)

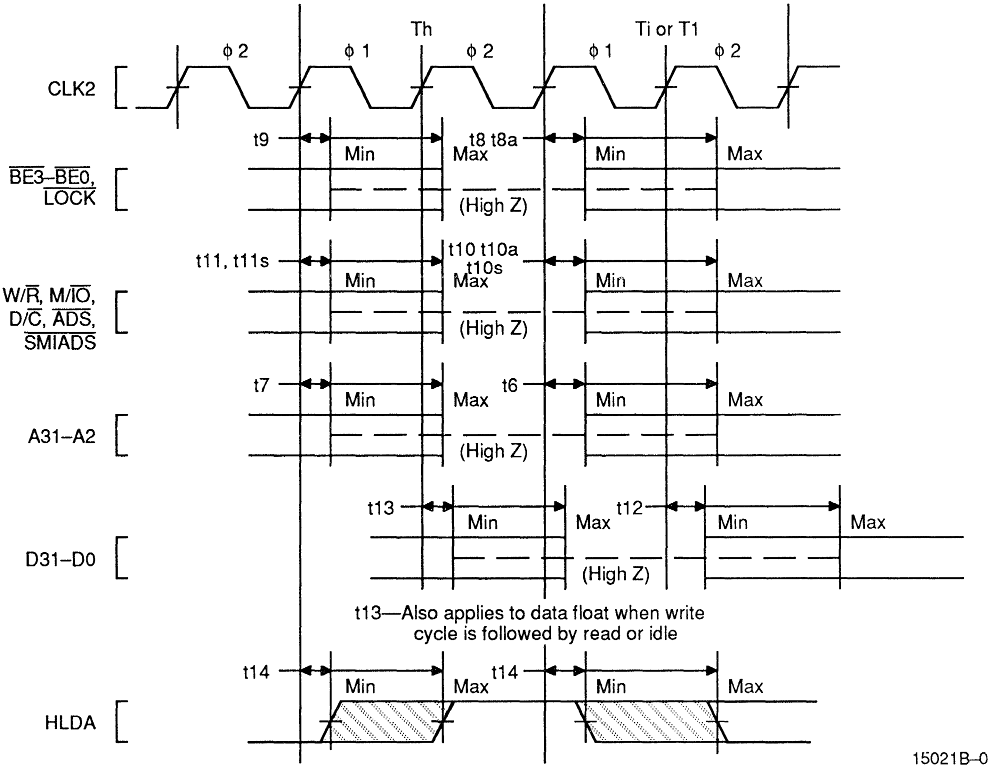
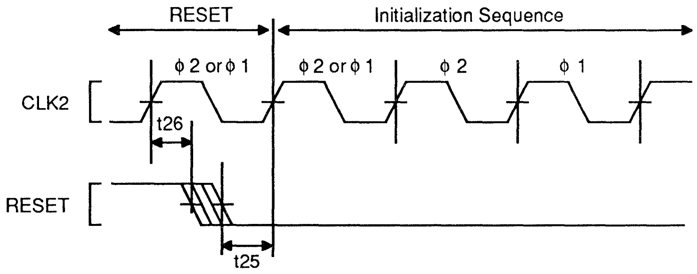


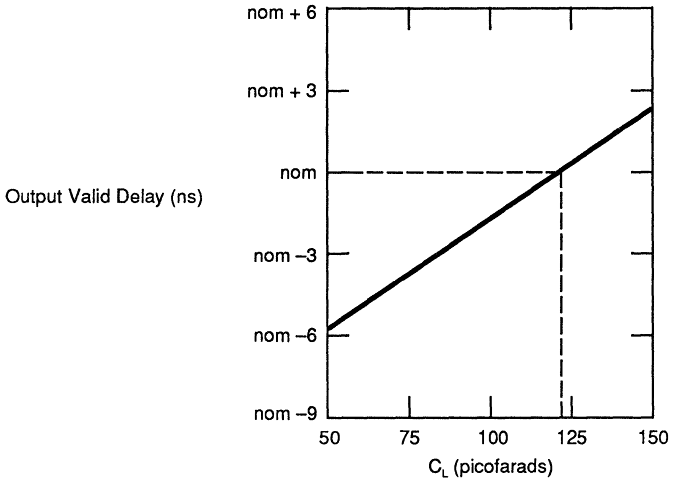
Figure 10. Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET High-to-Low transition (provided t25 and t26 are met) is ϕ2.

15021B–084

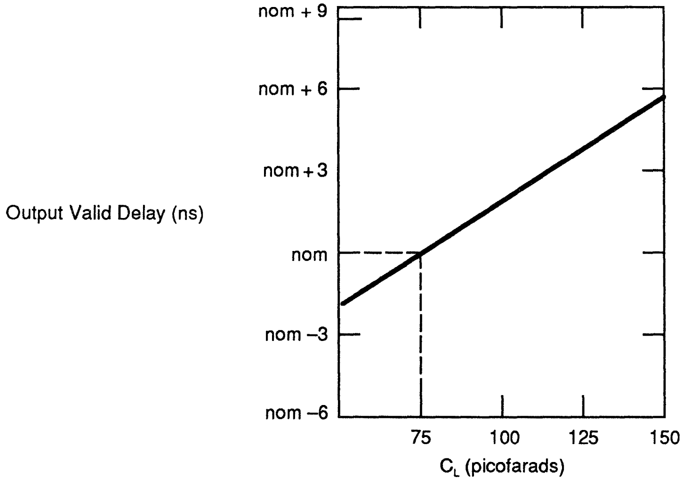
Figure 11. RESET Setup and Hold Timing and Internal Phase



Note: This graph will not be linear outside of the C_L range shown.

15021B–079

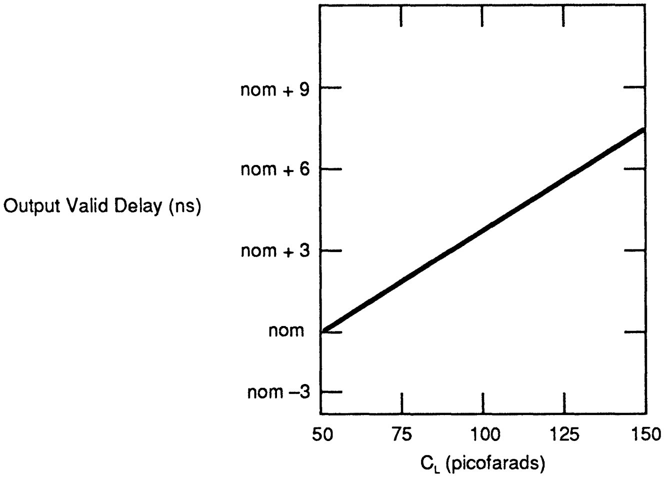
Figure 12. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L=120 pF)



Note: This graph will not be linear outside of the C_L range shown.

15021B–080

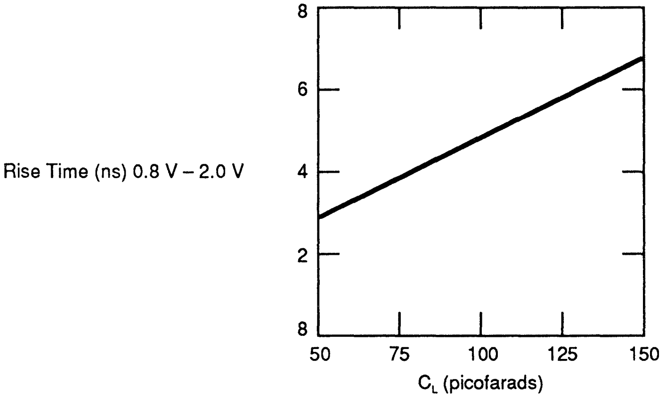
Figure 13. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 75 pF)



Note: This graph will not be linear outside of the C_L range shown.

15021B–081

Figure 14. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature ($C_L = 50$ pF)



Note: This graph will not be linear outside of the C_L range shown.

15021B–082

Figure 15. Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature

PRELIMINARY

Am386™SXLV

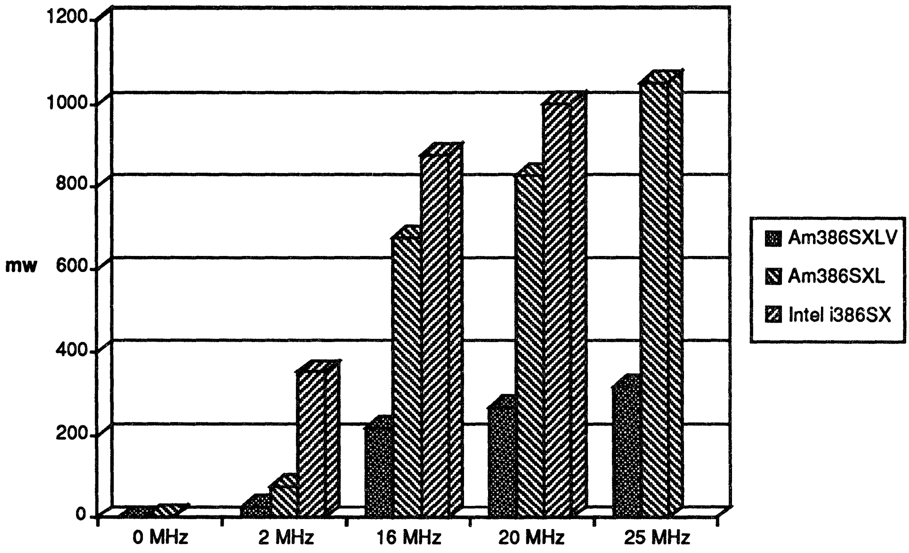
High-Performance, Low-Voltage, 32-Bit Microprocessor with 16-Bit Data Bus

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- **Operating range 3.0 V to 5.5 V—ideal for notebook PC designs**
  - —2X improvement in battery life over existing 5 V designs
  - —Wide range of chipsets and other logic available for 3 V systems with support for Standby Mode operation
  - —True static design for long battery life
  - —Power consumption 75% lower than Intel i386SX, 65% lower than Am386SXL microprocessor
  - —Performance on demand (0 to 25 MHz)
- **System Management Mode (SMM) for system and power management**
  - —System Management Interrupt (SMI) for power management independent of processor operating mode and operating system
  - —SMI coupled with I/O instruction break feature provides transparent power off and auto resume of peripherals which may not be "power aware"
  - —SMI is non-maskable and has higher priority than NMI
  - —Automatic save and restore of the microprocessor state
  - —Wide range of chipsets supporting SMM available to allow product differentiation
- **Lower heat dissipation facilitates elimination of the cooling fan in desktop PCs**
- **"Float" input to facilitate system debug and test**
- **Compatible with 386SX systems and software**
- **Supports 387SX-compatible math coprocessors**
- **100-pin PQFP package with optional protective ring for better lead coplanarity**
- **AMD® advanced 0.8 micron CMOS technology**

Typical Power Consumption



Publication # 16305 Rev. B Amendment /0
Issue Date: October 1991

GENERAL DESCRIPTION

The Am386SXLV microprocessor is a low-voltage, true static implementation of the Intel i386SX microprocessor. With the operating range of 3.0 V to 5.5 V, it is ideal for both desktop and battery-powered notebook personal computers. For desktop PCs, this device offers lower heat dissipation, allowing system designers to remove or reduce the size and cost of the cooling fan.

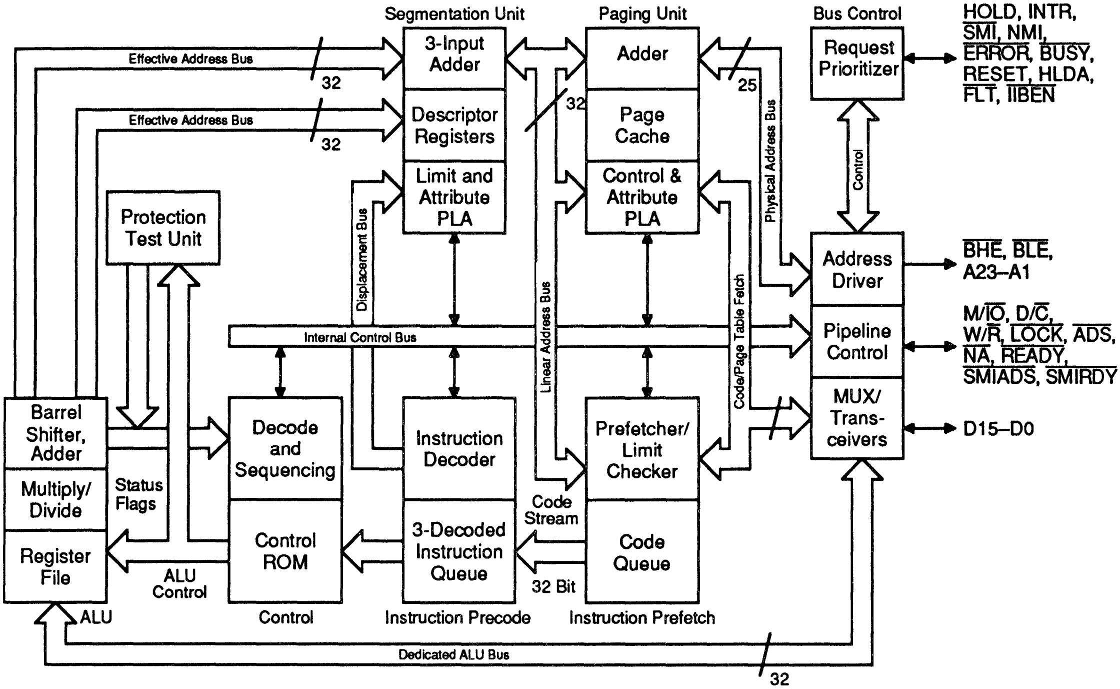
The Am386SXLV microprocessor's lower operating voltage and true static design enables longer battery life and/or lower weight for notebook applications. At 20 MHz, this device has 60% lower operating Icc than the Intel i386SX. Lowering typical operating voltage from 5.0 V to 3.3 V enables battery life to increase by a factor of two. Standby Mode allows the Am386SXLV microprocessor to be clocked down to 0 MHz (DC) and retain full register contents. In Standby Mode, typical current draw is less than 0.01 mA, a greater than 1000X reduction in power consumption versus the Intel i386SX.

The Am386SXLV microprocessor is available in a small footprint 100-pin Plastic Quad Flat Pack (PQFP) package. This package may be shipped in an optional protective ring for better lead protection during shipping.

Additionally, the Am386SXLV microprocessor comes with System Management Mode (SMM) for system and power management. SMI (System Management Interrupt) is a non-maskable, higher priority interrupt than NMI and has its own code space (1 Mb). SMI can be coupled with the I/O instruction break feature to implement transparent power managment of peripherals. SMM can be used by system designers to implement system and power management code independent of the operating system or the Processor Mode.

The Am386SXLV microprocessor incorporates a float pin that places all outputs in a three-state mode to facilitate board test and debug.

BLOCK DIAGRAM



15022B-001

FUNCTIONAL DESCRIPTION

Benefits of Lower Operating Voltage

The Am386SXLV microprocessor has an operating voltage range of 3.0 V to 5.5 V. Low voltage allows for lower operating power consumption, longer battery life, and/or smaller batteries for notebook applications.

Because power is proportional to the square of voltage, reduction of the supply voltage from 5.0 V to 3.3 V reduces power consumption by 56%. This directly translates to a doubling of battery life for portable applications. Lower power consumption can also be used to reduce the size and weight of the battery. Thus, 3.3 V designs facilitate a reduction in the form factor. For desktop PCs, low power consumption means elimination of the cooling fan, thus reducing the size and noise of the PC.

A lower operating voltage results in a reduction of I/O voltage swings. This reduces noise generation providing a less hostile environment for board design. It also reduces electromagnetic radiation noise making it easier to obtain FCC approval.

SMM—System Management Mode

The Am386SXLV microprocessor has a new System Management Mode (SMM) for system and power management. This mode consists of two features: System Management Interrupt (SMI) and I/O instruction break.

SMI—System Management Interrupt

SMI is implemented through the use of special bus interface pins. This interrupt method can be used to perform system management functions such as power management independent of Processor Operating Mode (Real, Protected, or Virtual 86 modes).

SMI can also be invoked in software. This allows system software to communicate with SMI power management code. In addition, an instruction called UMOV allows data transfers between SMI and normal system memory spaces.

Activating the $\overline{\text{SMI}}$ pin invokes a sequence that saves the operating state of the processor into a separate SMM memory space, independent of the main system memory. After the state is saved, the processor is forced into Real Mode and begins execution at address FFFFF0h in the SMM memory space where a far jump to the SMM code is executed. This Real Mode code can perform its system management function and then resume execution of the normal system software by executing a special opcode sequence which will reload the saved processor state and continue execution in the main system memory space. See Figure 1 for a general flowchart of an SMM operation.

CPU Interface—Pin Functions

The CPU interface for SMM consists of three pins dedicated to the SMI function. One pin, $\overline{\text{SMI}}$, is the new interrupt input. The other two pins, $\overline{\text{SMIADS}}$ and $\overline{\text{SMIRDY}}$, provide the control signals necessary for the separate SMM mode memory space.

Description of SMM Operation

The execution of a System Management Interrupt has four distinct phases: the initiation of the interrupt via $\overline{\text{SMI}}$, a processor state save, execution of the SMM interrupt code, and a processor state restore (to resume normal operation).

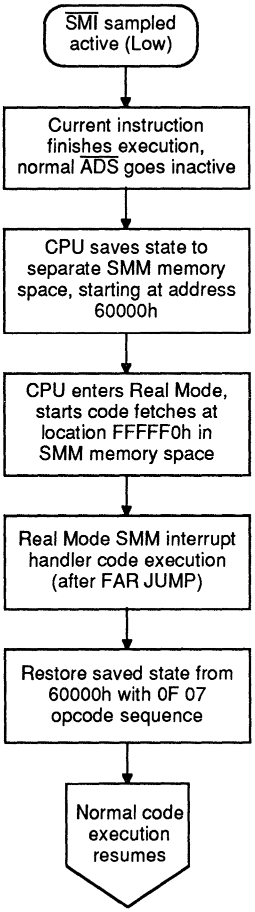

$\overline{\text{SMI}}$ sampled active (Low)
Current instruction finishes execution, normal $\overline{\text{ADS}}$ goes inactive
CPU saves state to separate SMM memory space, starting at address 60000h
CPU enters Real Mode, starts code fetches at location FFFFF0h in SMM memory space
Real Mode SMM interrupt handler code execution (after FAR JUMP)
Restore saved state from 60000h with 0F 07 opcode sequence
Normal code execution resumes

Figure 1. SMM Flow

Interrupt Initiation

A System Management Interrupt is initiated by the driving of an active Low pulse on the $\overline{\text{SMI}}$ pin of at least four CLK2 periods. This pulse period will ensure recognition of the interrupt. The CPU will drive the $\overline{\text{SMI}}$ pin active after the completion of the current operation (active bus cycle, instruction execution, or both). The active drive of the pin by the CPU will be released at the end of the interrupt routine following the last register read of the saved state.

While the CPU is in SMM, a bus hold request via the HOLD pin **will be granted.** The HLDA pin will go active after bus release and the $\overline{\text{SMIADS}}$ pin will float along with the other pins that normally float during a bus hold cycle.

Processor State Save

The first set of SMM bus transfer cycles after the CPU's recognition of an active SMI will be the processor saving its state to an external RAM array in a separate address space from main system memory. This is accomplished by using the $\overline{\text{SMIADS}}$ and $\overline{\text{SMIRDY}}$ pins for initiation and termination of bus cycles, instead of the $\overline{\text{ADS}}$ and $\overline{\text{READY}}$ pins. The 32-bit addresses to which the CPU saves its state are 60000h–600CAh and 60100h–60126h. These are fixed address locations for each register saved.

The value of $\overline{\text{NA}}$ will be ignored during the state save. Only full 16-bit, non-pipelined cycles are generated for the state save cycles. There are 114 data transfer cycles

SMI Code Execution

After the processor state is saved to the separate SMM memory space, the execution of the SMI interrupt routine code begins. The processor enters Real Mode, sets most of the register values to "reset" values (those values normally seen after a CPU reset), and begins fetching code from address FFFFF0h in the separate SMM memory space. Normally, the first thing the interrupt routine code will do is a FAR JUMP to the Real Mode entry point for the SMI interrupt routine, which is also in SMM memory space.

Any Real Mode interrupt routine code can be executed, with the obvious exception of normal interrupt routines (which are deferred). The SMM code can be located anywhere within the 1-Mb Real Mode address space, except for where the processor state is saved. I/O cycles, as a result of the IN, OUT, INS, and OUTS instructions, will go to the normal address space, utilizing the normal $\overline{\text{ADS}}$ and $\overline{\text{READY}}$ bus interface signals. This facilitates power management code manipulating system hardware registers as needed through the standard I/O subsystem; a separate I/O space does not need to be implemented.

Processor State Restore (Resuming Normal Execution)

Returning to normal code execution in the main system memory, including restoring the Processor Operating Mode, is accomplished by executing a special code sequence. This code invokes a restore CPU state operation which reloads the CPU registers from the saved data in the RAM controlled by $\overline{\text{SMIADS}}$ and $\overline{\text{SMIRDY}}$.

The ES:EDI register pair must point to physical address 60000h. Then the special opcode sequence 0Fh 07h should be executed to start the restore state operation. After completion of the restore state operation, the $\overline{\text{SMI}}$ pin will be deactivated by the CPU and normal code execution will continue at the point it left off before the SMI occurred. There are 114 data transfer cycles in the restore operation.

Software Features

There are several features of the SMI function that provide support for special operations during the execution of the system's software. These features involve the execution of reserved opcodes to induce specific SMI related operations.

Software SMI Generation

Besides hardware initiation of the System Management Interrupt via the $\overline{\text{SMI}}$ pin, there is also a software induced SMI mechanism. Generating a soft SMI involves setting a control bit in Debug Register 7 (DR7) and executing a reserved opcode (0F1H).

The functional sequence of the software based SMI is identical to the hardware based SMI with the exception that the $\overline{\text{SMI}}$ pin is not initially driven active by an external source. Upon execution of a soft SMI opcode, the $\overline{\text{SMI}}$ pin is driven active (Low) by the processor before the save state operation begins.

Memory Transfers to Main System Memory

While executing an SMI routine, the interrupt code can initiate memory data reads and writes to the main system memory using the normal $\overline{\text{ADS}}$ and $\overline{\text{READY}}$ pins. This is accomplished by using reserved opcodes that are special forms of the MOV instruction (called UMOV). The UMOV opcodes can move byte, word, or double word register operands to or from main system memory. Multiple data transfers using the normal $\overline{\text{ADS}}$ and $\overline{\text{READY}}$ pins will occur if the operands are misaligned relative to the effective address used. The UMOV opcodes are 0F10h, 0F11h, 0F12h, and 0F13h.

I/O Instruction Break

The Am386SXLV microprocessor has an I/O instruction break feature that allows the system logic to implement I/O trapping for peripheral devices. To enable the I/O Instruction break feature, $\overline{\text{IIBEN}}$ must first be asserted active Low. On detecting an I/O instruction, the processor will prevent the execution unit from executing further instructions until $\overline{\text{READY}}$ is driven active Low by the system. Once $\overline{\text{READY}}$ is driven active, the execution unit will either immediately respond to any active interrupt request or continue executing instructions following the I/O instruction that caused the break.

The I/O instruction break feature can be used to allow system logic to implement I/O trapping for peripheral devices. On sensing an I/O instruction, the system can drive the $\overline{\text{SMI}}$ pin active before driving $\overline{\text{READY}}$ active. This ensures that the SMI service routine is executed immediately following the I/O instruction that caused the break. (If the I/O instruction break feature is not enabled via $\overline{\text{IIBEN}}$, several instructions could execute before the SMI service routine is executed.)

The SMI service routine can access the peripheral for which $\overline{\text{SMI}}$ was asserted and modify its state.The SMI service routine will normally return to the instruction following the I/O instruction that caused the break. By modifying the saved state instruction pointer, the routine can choose to return to the I/O instruction that caused the break and re-execute that instruction. The default is to return to the following instruction (except for REP I/O string instruction). To re-execute the I/O instruction that caused the break, the SMI service routine must copy the I/O instruction pointer over the default pointer. This feature is particularly useful when an application program requests an access to a peripheral that has been powered down. The SMI service routine can restore power to the peripheral and initiate a re-execution sequence transparent to the application program. This re-execution feature should only be used if the SMI is in response to an I/O trap with $\overline{\text{IIBEN}}$ active. Note that the I/O instruction break feature is not enabled for memory mapped I/O devices or for coprocessor bus cycles even if $\overline{\text{IIBEN}}$ is active.

I/O Instruction Break Timing

The I/O Instruction Break feature requires that $\overline{\text{SMI}}$ be sampled active (Low) by the processor at least three CLK2 edges before the CLK2 edge that ends the I/O cycle with an active $\overline{\text{READY}}$ signal. This timing applies for both pipelined and non-pipelined cycles. If this timing constraint is not met, the next instruction may be executed by the internal execution unit prior to entering SMI Mode, but the $\overline{\text{SMI}}$ will be recognized eventually.

True Static Operation

The Am386SXLV microprocessor incorporates a true static design. Unlike dynamic circuit design, the Am386SXLV device eliminates the minimum operating frequency restriction. It may be clocked from its maximum speed of 25 MHz all the way down to 0 MHz (DC). System designers can use this feature to design battery-powered notebook PCs with long battery life.

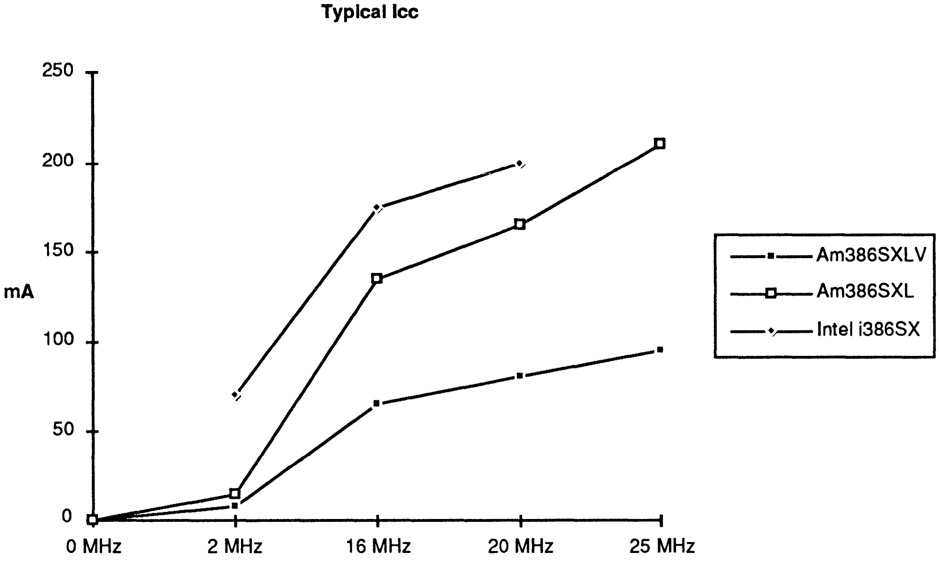
Standby Mode

This true static design allows for a Standby Mode. At any operating speed (25 MHz to 0 MHz), the Am386SXLV microprocessor will retain its state (i.e., the contents of all of its registers). By shutting off the clock completely, the device enters Standby Mode. Since power consumption is proportional to clock frequency, operating power consumption is reduced as the frequency is lowered. In Standby Mode, typical current draw is reduced to less than 0.01 mA. Not only does this feature save battery life, but it also simplifies the design of power-conscious notebook computers in the following ways.

1. Eliminates the need for software in BIOS to save and restore the contents of registers
2. Allows simpler circuitry to control stopping of the clock (since) the system does not need to know what state the processor is in

Lower Operating Icc

True static design also allows lower operating Icc when operating at any speed. See the following graph for typical current at operating speeds.



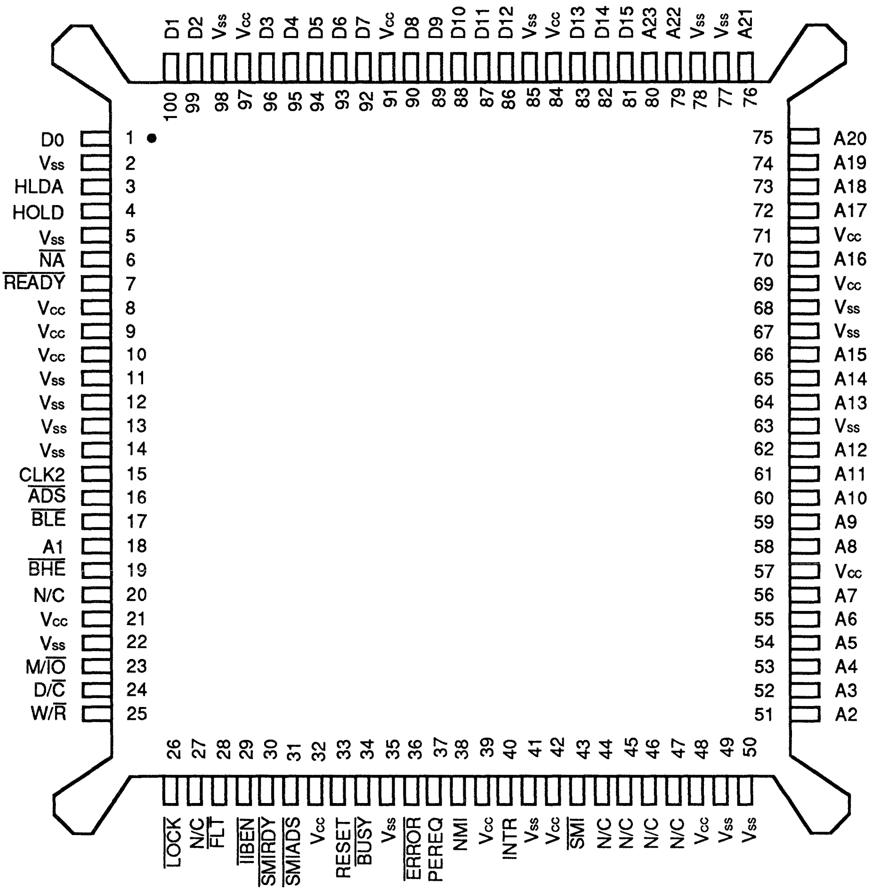
Performance on Demand

The Am386SXLV microprocessor retains its state at any speed from 0 MHz (DC) to its maximum operating speed. With this feature, system designers may vary the operating speed of the system to extend the battery life in notebook systems.

For example, the system could operate at low speeds during inactivity or polling operations. However, upon interrupt, the system clock can be increased up to its maximum speed. After a user-defined time-out period, the system can be returned to a low (or a 0 MHz) operating speed without losing its state. This design maximizes battery life while achieving optimal performance.

CONNECTION DIAGRAM
Top View

100-Pin PQFP



Notes: Pin 1 is marked for orientation.
N/C = Not connected.

PIN DESIGNATION TABLES (Sorted by Pin Name)

Address		Data		Control		NC	V_{cc}	V_{ss}
Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin No.	Pin No.	Pin No.
A1	18	D0	1	$\overline{ADS}$	16	20	8	2
A2	51	D1	100	$\overline{BHE}$	19	27	9	5
A3	52	D2	99	$\overline{BLE}$	17	44	10	11
A4	53	D3	96	$\overline{BUSY}$	34	45	21	12
A5	54	D4	95	CLK2	15	46	32	13
A6	55	D5	94	D/$\overline{C}$	24	47	39	14
A7	56	D6	93	$\overline{ERROR}$	36		42	22
A8	58	D7	92	$\overline{FLT}$	28		48	35
A9	59	D8	90	HLDA	3		57	41
A10	60	D9	89	HOLD	4		69	49
A11	61	D10	88	$\overline{IIBEN}$	29		71	50
A12	62	D11	87	INTR	40		84	63
A13	64	D12	86	$\overline{LOCK}$	26		91	67
A14	65	D13	83	M/$\overline{IO}$	23		97	68
A15	66	D14	82	$\overline{NA}$	6			77
A16	70	D15	81	NMI	38			78
A17	72			PEREQ	37			85
A18	73			$\overline{READY}$	7			98
A20	75			RESET	33			
A21	76			$\overline{SMI}$	43			
A22	79			$\overline{SMIADS}$	31			
A23	80			$\overline{SMIRDY}$	30			
				W/$\overline{R}$	25			

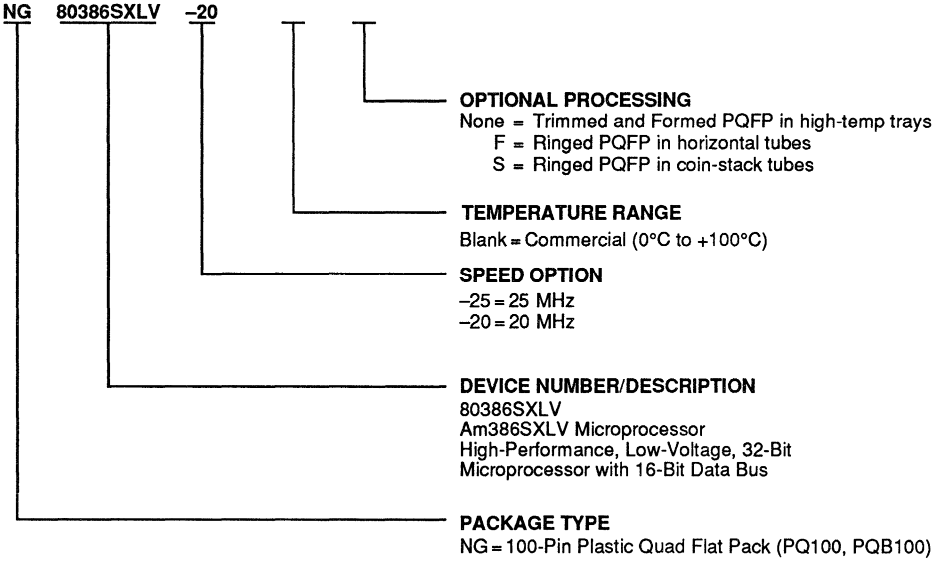
PIN DESIGNATION TABLES (Sorted by Pin Number)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	D0	21	V_{cc}	41	V_{ss}	61	A11	81	D15
2	V_{ss}	22	V_{ss}	42	V_{cc}	62	A12	82	D14
3	HLDA	23	M/$\overline{IO}$	43	$\overline{SMI}$	63	V_{ss}	83	D13
4	HOLD	24	D/$\overline{C}$	44	NC	64	A13	84	V_{cc}
5	V_{ss}	25	W/$\overline{R}$	45	NC	65	A14	85	V_{ss}
6	$\overline{NA}$	26	$\overline{LOCK}$	46	NC	66	A15	86	D12
7	$\overline{READY}$	27	NC	47	NC	67	V_{ss}	87	D11
8	V_{cc}	28	$\overline{FLT}$	48	V_{cc}	68	V_{ss}	88	D10
9	V_{cc}	29	$\overline{IIBEN}$	49	V_{ss}	69	V_{cc}	89	D9
10	V_{cc}	30	$\overline{SMIRDY}$	50	V_{ss}	70	A16	90	D8
11	V_{ss}	31	$\overline{SMIADS}$	51	A2	71	V_{cc}	91	V_{cc}
12	V_{ss}	32	V_{cc}	52	A3	72	A17	92	D7
13	V_{ss}	33	RESET	53	A4	73	A18	93	D6
14	V_{ss}	34	$\overline{BUSY}$	54	A5	74	A19	94	D5
15	CLK2	35	V_{ss}	55	A6	75	A20	95	D4
16	$\overline{ADS}$	36	$\overline{ERROR}$	56	A7	76	A21	96	D3
17	$\overline{BLE}$	37	PEREQ	57	V_{cc}	77	V_{ss}	97	V_{cc}
18	A1	38	NMI	58	A8	78	V_{ss}	98	V_{ss}
19	$\overline{BHE}$	39	V_{cc}	59	A9	79	A22	99	D2
20	NC	40	INTR	60	A10	80	A23	100	D1

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



OPTIONAL PROCESSING
None = Trimmed and Formed PQFP in high-temp trays
F = Ringed PQFP in horizontal tubes
S = Ringed PQFP in coin-stack tubes

TEMPERATURE RANGE
Blank = Commercial (0°C to +100°C)

SPEED OPTION
–25 = 25 MHz
–20 = 20 MHz

DEVICE NUMBER/DESCRIPTION
80386SXLV
Am386SXLV Microprocessor
High-Performance, Low-Voltage, 32-Bit
Microprocessor with 16-Bit Data Bus

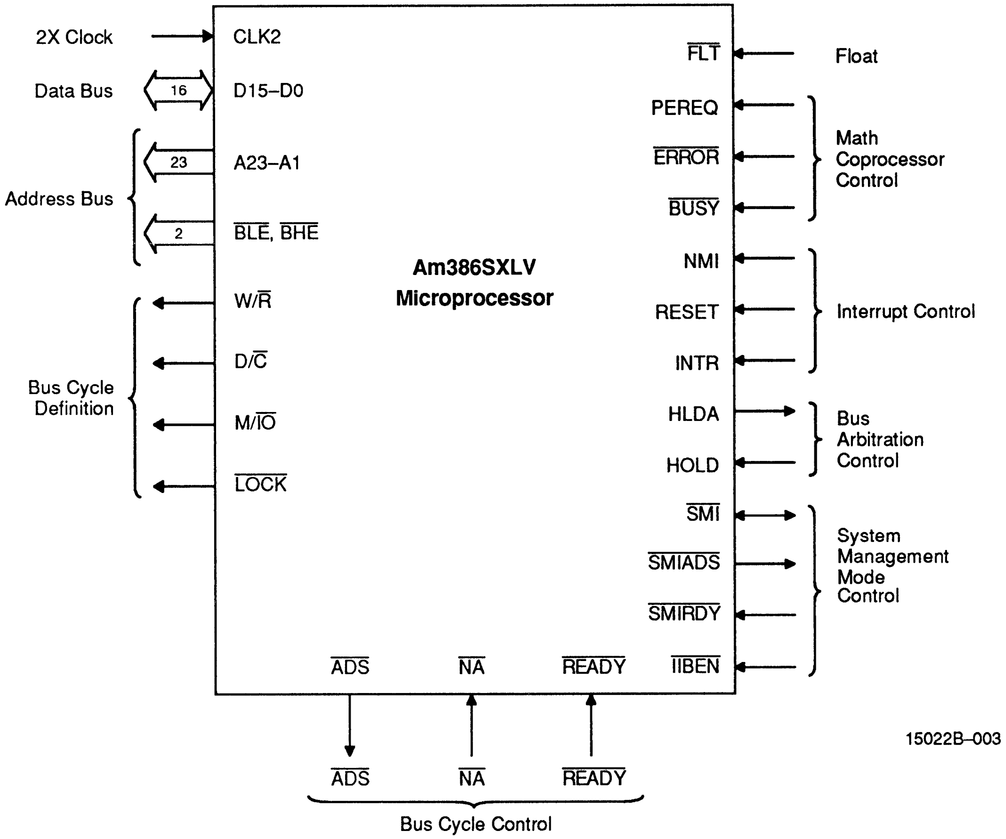
PACKAGE TYPE
NG = 100-Pin Plastic Quad Flat Pack (PQ100, PQB100)

Valid Combinations		
NG	80386SXLV	–25 –20
		–25F –20F
		–25S –20S

Valid Combinations

Valid Combinations lists configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations.

LOGIC SYMBOL



PIN DESCRIPTIONS

A23–A1

Address Bus (Outputs)

Outputs physical memory or port I/O addresses.

$\overline{\text{ADS}}$

Address Status (Active Low; Output)

Indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, and A23–A1) are being driven at the Am386SXLV microprocessor pins.

$\overline{\text{BHE}}$, $\overline{\text{BLE}}$

Byte Enables (Active Low; Outputs)

Indicate which data bytes of the data bus take part in a bus cycle.

$\overline{\text{BUSY}}$

Busy (Active Low; Input)

Signals a busy condition from a processor extension. $\overline{\text{BUSY}}$ has an internal pullup resistor.

CLK2

CLK2 (Input)

Provides the fundamental timing for the Am386SXLV microprocessor.

D15–D0

Data Bus (Inputs/Outputs)

Inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles.

D/$\overline{\text{C}}$

Data/Control (Output)

A bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and code fetch.

$\overline{\text{ERROR}}$

Error (Active Low; Input)

Signals an error condition from a processor extension. $\overline{\text{ERROR}}$ has an internal pullup resistor.

$\overline{\text{FLT}}$

Float (Active Low; Input)

An input which forces all bi-directional and output signals, including HLDA, to the three-state condition. $\overline{\text{FLT}}$ has an internal pullup resistor. The pin, if not used, should be disconnected.

HLDA

Bus Hold Acknowledge (Active High; Output)

Output indicates that the Am386SXLV microprocessor has surrendered control of its logical bus to another bus master.

HOLD

Bus Hold Request (Active High; Input)

Input allows another bus master to request control of the local bus.

$\overline{\text{IIBEN}}$

I/O Instruction Break Enable (Active Low; Input)

Enables the I/O instruction break feature. $\overline{\text{IIBEN}}$ has an internal pullup resistor.

INTR

Interrupt Request (Active High; Input)

A maskable input that signals the Am386SXLV microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

$\overline{\text{LOCK}}$

Bus Lock (Active Low; Output)

A bus cycle definition pin that indicates that other system bus masters are not to gain control of the system bus while it is active.

M/$\overline{\text{IO}}$

Memory/IO (Output)

A bus cycle definition pin that distinguishes memory cycles from input/output cycles.

$\overline{\text{NA}}$

Next Address (Active Low; Input)

Used to request address pipelining.

NC

No Connect

Should always be left unconnected. Connection of a NC pin may cause the processor to malfunction or be incompatible with future steppings of the Am386SXLV microprocessor.

NMI

Non-Maskable Interrupt Request (Active High; Input)

A non-maskable input that signals the Am386SXLV microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

PEREQ

Processor Extension Request (Active High; Input)

Indicates that the processor has data to be transferred by the Am386SXLV microprocessor. PEREQ has an internal pulldown resistor.

$\overline{\text{READY}}$

Bus Ready (Active Low; Input)

Terminates the bus cycle.

RESET

Reset (Active High; Input)

Suspends any operation in progress and places the Am386SXLV microprocessor in a known reset state.

$\overline{\text{SMI}}$

System Management Interrupt (Active Low; I/O)

A non-maskable interrupt pin which signals the Am386SXLV microprocessor to suspend execution and enter System Management Mode. $\overline{\text{SMI}}$ has an internal pullup resistor.

$\overline{\text{SMIADS}}$

SMI Address Status (Active Low Three-State; Output)

When active, this pin indicates that a valid bus cycle definition and address (W/$\overline{\text{R}}$ D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, and A23–A1) are being driven at the Am386SXLV microprocessor pins while in the System Management Mode.

$\overline{\text{SMIRDY}}$

SMI Ready (Active Low; Input)

This input terminates the current bus cycle to the SMM Mode address space in the same manner as the $\overline{\text{READY}}$ pin does for the normal mode address space. $\overline{\text{SMIRDY}}$ has an internal pullup resistor.

V_{CC}

System Power (Active High; Input)

Provides the DC supply input.

V_{SS}

System Ground (Input)

Provides the 0-V connection from which all inputs and outputs are measured.

W/$\overline{\text{R}}$

Write/Read (Output)

A bus cycle definition pin that distinguishes write cycles from read cycles.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	–65°C to +150°C
Ambient Temperature Under Bias .	–65°C to +125°C

Stresses above those listed may cause permanent damage to the device. Functionality at or above these limits is not implied. Exposure to ABSOLUTE MAXIMUM RATING conditions for extended periods of time may affect device reliability.

OPERATING RANGES

Supply Voltage with respect to V_{SS} ..	–0.5 V to +7.0 V
Voltage on Other Pins	–0.5 V to V_{CC} +0.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating ranges

V_{CC} = 3.0 V to 3.6 V; T_{CASE} = 0°C to +100°C

			Preliminary		
Symbol	**Parameter Description**	**Notes**	**Min**	**Max**	**Unit**
V_{IL}	Input Low Voltage	(Note 1)	–0.3	+0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}+ 0.3	V
V_{ILC}	CLK2 Input Low Voltage	(Note 1)	–0.3	+0.8	V
V_{IHC}	CLK2 Input High Voltage 16, 20 MHz		V_{CC} – 0.6	V_{CC}+ 0.3	V
V_{OL}	Output Low Voltage I_{OL} = 0.5 mA: A23–A1, D15–D0 I_{OL} = 0.5 mA: $\overline{BHE}$, $\overline{BLE}$,W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, $\overline{SMIADS}$, HLDA I_{OL} = 2 mA: A23–A1, D15–D0 I_{OL} = 2.5 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, $\overline{SMIADS}$, HLDA	 (Note 5)		 0.2 0.2 0.45 0.45	 V V V V
V_{OH}	Output High Voltage I_{OH} = 0.1 mA: A23–A1, D15–D0 I_{OH} = 0.1 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, $\overline{SMIADS}$, HLDA I_{OH} = 0.5 mA: A23–A1, D15–D0 I_{OH} = 0.5 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, $\overline{SMIADS}$, HLDA	 (Note 5)	 V_{CC}– 0.2 V_{CC}– 0.2 V_{CC}– 0.45 V_{CC}– 0.45		 V V V V
I_{LI}	Input Leakage Current (All pins except PEREQ, $\overline{BUSY}$, $\overline{ERROR}$, $\overline{SMI}$, $\overline{SMIRDY}$, $\overline{FLT}$, $\overline{IIBEN}$)	0 V ≤ V_{IN} ≤ V_{CC}		±15	µA
I_{IH}	Input Leakage Current (PEREQ pin)	V_{IH} = V_{CC}–0.1 V V_{IH} = 2.4 V (Note 2)		300 200	µA µA
I_{IL}	Input Leakage Current ($\overline{BUSY}$, $\overline{ERROR}$, $\overline{SMI}$, $\overline{SMIRDY}$, $\overline{FLT}$, $\overline{IIBEN}$)	V_{IL} = 0.1 V V_{IL} = 0.45 V (Note 3)		–300 –200	µA µA
I_{LO}	Output Leakage Current	0.1 V ≤ V_{OUT} ≤ V_{CC}		±15	µA
I_{CC}	Supply Current CLK2 = 32 MHz: Oper. Freq. 16 MHz CLK2 = 40 MHz: Oper. Freq. 20 MHz CLK2 = 50 MHz: Oper. Freq. 25 MHz	 I_{CC} Typ = 65 I_{CC} Typ = 80 I_{CC} Typ = 95		 95 110 125	 mA mA mA
I_{CCSB}	Standby Current	I_{CCSB} Typ=10		150	µA
C_{IN}	Input or I/O Capacitance	F_C = 1 MHz (Note 4)		10	pF
C_{OUT}	Output Capacitance	F_C = 1 MHz (Note 4)		12	pF
C_{CLK}	CLK2 Capacitance	F_C = 1 MHz (Note 4)		20	pF

Notes: 1. The min value, –0.3, is not 100% tested.
2. $\overline{PEREQ}$ input has an internal pulldown resistor.
3. $\overline{BUSY}$, $\overline{ERROR}$, $\overline{FLT}$, $\overline{SMI}$, $\overline{IIBEN}$, and $\overline{SMIRDY}$ inputs each have an internal pullup resistor.
4. Not 100% tested.
5. Outputs are CMOS and will pull rail to rail if the load is not resistive.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	–65°C to +150°C
Ambient Temperature under Bias .	–65°C to +125°C

Stresses above those listed may cause permanent damage to the device. Functionality at or above these limits is not implied. Exposure to ABSOLUTE MAXIMUM RATING conditions for extended periods of time may affect device reliability.

OPERATING RANGES

Supply Voltage with respect to V_{SS} ...	–0.5 V to +7 V
Voltage on Other Pins	–0.5 V to V_{CC} +0.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating ranges

V_{CC} = 3.6 V to 5.5 V; T_{CASE} = 0°C to +100°C

			Preliminary		
Symbol	**Parameter Description**	**Notes**	**Min**	**Max**	**Unit**
V_{IL}	Input Low Voltage	(Note 1)	–0.3	+0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}+ 0.3	V
V_{ILC}	CLK2 Input Low Voltage	(Note 1)	–0.3	+0.8	V
V_{IHC}	CLK2 Input High Voltage		V_{CC} – 0.8	V_{CC}+ 0.3	V
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A23–A1, D15–D0 I_{OL} = 5 mA: $\overline{BHE}$, $\overline{BLE}$,W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, $\overline{SMIADS}$, HLDA	 (Note 5)		 0.45 0.45	 V V
V_{OH}	Output High Voltage I_{OH} = 1.0 mA: A23–A1, D15–D0 I_{OH} = 0.2 mA: A23–A1, D15–D0 I_{OH} = 0.9 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, $\overline{SMIADS}$, HLDA I_{OH} = 0.18 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, $\overline{SMIADS}$, HLDA	 (Note 5)	 2.4 V_{CC} – 0.5 2.4 V_{CC} – 0.5		 V V V V
I_{LI}	Input Leakage Current (All pins except PEREQ, $\overline{BUSY}$, $\overline{ERROR}$, $\overline{SMI}$, $\overline{SMIRDY}$, $\overline{FLT}$, and $\overline{IIBEN}$)	0 V ≤ V_{IN} ≤ V_{CC}		±15	μA
I_{IH}	Input Leakage Current (PEREQ pin)	V_{IH} = 2.4 V (Note 2)		200	μA
I_{IL}	Input Leakage Current ($\overline{BUSY}$, $\overline{ERROR}$, $\overline{SMI}$, $\overline{SMIRDY}$, $\overline{FLT}$, $\overline{IIBEN}$)	V_{IL} = 0.45 V (Note 3)		–400	μA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		±15	μA
I_{CC}	Supply Current CLK2 = 32 MHz: Oper. Freq. 16 MHz CLK2 = 40 MHz: Oper. Freq. 20 MHz CLK2 = 50 MHz: Oper. Freq. 25 MHz	V_{CC} = 5.5 V I_{CC} Typ =135 I_{CC} Typ =165 I_{CC} Typ =210		 160 200 250	 mA mA mA
I_{CCSB}	Standby Current	I_{CCSB} Typ = 0.02 mA		0.15	mA
C_{IN}	Input or I/O Capacitance	F_C = 1 MHz (Note 4)		10	pF
C_{OUT}	Output Capacitance	F_C = 1 MHz (Note 4)		12	pF
C_{CLK}	CLK2 Capacitance	F_C = 1 MHz (Note 4)		20	pF

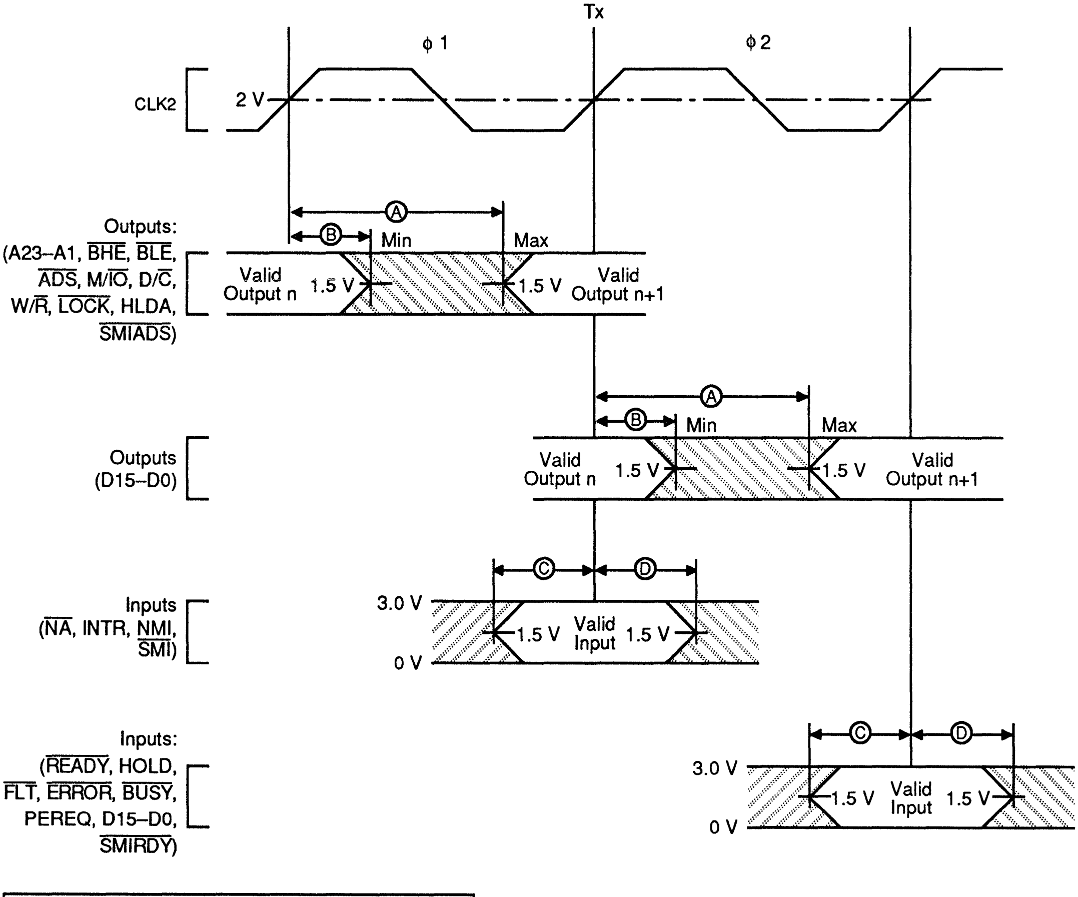
Notes: 1. The min value, –0.3, is not 100% tested.
2. $\overline{PEREQ}$ input has an internal pulldown resistor.
3. $\overline{BUSY}$, $\overline{ERROR}$, $\overline{FLT}$, $\overline{SMI}$, $\overline{IIBEN}$, and $\overline{SMIRDY}$ inputs each have an internal pullup resistor.
4. Not 100% tested.
5. Outputs are CMOS and will pull rail to rail if the load is not resistive.

SWITCHING CHARACTERISTICS

The switching characteristics given consist of output delays, input setup requirements, and input hold requirements. All switching characteristics are relative to the CLK2 rising edge crossing the 2.0 V level.

Switching characteristic measurement is defined by Figure 2. Inputs must be driven to the voltage levels indicated by Figure 2 when switching characteristics are measured. Output delays are specified with minimum and maximum limits measured, as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs $\overline{ADS}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{BHE}$, $\overline{BLE}$, $\overline{SMIADS}$, A23–A1, and HLDA only change at the beginning of phase one. D15–D0 (write cycles) only change at the beginning of phase two. The $\overline{READY}$, HOLD, $\overline{BUSY}$, $\overline{SMIRDY}$, $\overline{ERROR}$, PEREQ, $\overline{FLT}$, and D15–D0 (read cycles) inputs are sampled at the beginning of phase one. The $\overline{NA}$, INTR, NMI, and $\overline{SMI}$ inputs are sampled at the beginning of phase two.



Legend: A—Maximum Output Delay Characteristic
B—Minimum Output Delay Characteristic
C—Minimum Input Setup Characteristic
D—Minimum Input Hold Characteristic

15022B–030

Figure 2. Drive Levels and Measurement Points for Switching Characteristics

SWITCHING CHARACTERISTICS over operating ranges at 25 MHz

V_{CC} = 3.0 V –5.5 V; T_{CASE} = 0°C to 100°C

			Preliminary			
Symbol	**Parameter Description**	**Notes**	**Ref. Figures**	**Min**	**Max**	**Unit**
	Operating Frequency	Half CLK2 freq.		0	25	MHz
1	CLK2 Period		3	20		ns
2a	CLK2 High Time	at 2 V	3	7		ns
2b	CLK2 High Time	at (V_{CC}–0.8 V)	3	4		ns
3a	CLK2 Low Time	at 2 V	3	7		ns
3b	CLK2 Low Time	at 0.8 V	3	5		ns
4	CLK2 Fall Time	(V_{CC}–0.8 V) to 0.8 V (Note 3)	3		7	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}–0.8 V) (Note 3)	3		7	ns
6	A23–A1 Valid Delay	C_L = 50 pF	6	4	17	ns
7	A23–A1 Float Delay	(Note 1)	10	4	30	ns
8	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 50 pF	6	4	17	ns
9	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	10	4	30	ns
10	M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, W/$\overline{\text{R}}$, $\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	6	4	17	ns
10s	$\overline{\text{SMIADS}}$ Valid Delay	C_L = 50 pF	6	4	25	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	10	4	30	ns
11s	$\overline{\text{SMIADS}}$ Float Delay	(Note 1)	10	4	30	ns
12	D15–D0 Write Data Valid Delay	C_L = 50 pF	6, 7, 9	7	23	ns
12a	D15–D0 Write Data Hold Time	C_L = 50 pF	8	2		ns
13	D15–D0 Write Data Float Delay	(Note 1)	10	4	22	ns
14	HLDA Valid Delay	C_L = 50 pF	6	4	22	ns
15	$\overline{\text{NA}}$ Setup Time		5	5		ns
16	$\overline{\text{NA}}$ Hold Time		5	3		ns
19	$\overline{\text{READY}}$ Setup Time		5	9		ns
19s	$\overline{\text{SMIRDY}}$ Setup Time		5	9		ns
20	$\overline{\text{READY}}$ Hold Time		5	4		ns
20s	$\overline{\text{SMIRDY}}$ Hold Time		5	4		ns
21	D15–D0 Read Data Setup Time		5	7		ns
22	D15–D0 Read Data Hold Time		5	5		ns
23	HOLD Setup Time		5	9		ns
24	HOLD Hold Time		5	3		ns
25	RESET Setup Time		11	8		ns
26	RESET Hold Time		11	3		ns
27	NMI, INTR Setup Time	(Note 2)	5	6		ns
27s	$\overline{\text{SMI}}$ Setup Time	(Note 2)	5	6		ns
28	NMI, INTR Hold Time	(Note 2)	5	6		ns
28s	$\overline{\text{SMI}}$ Hold Time	(Note 2)	5	6		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Setup Time	(Note 2)	5	6		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$ Hold Time	(Note 2)	5	5		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

3. These are not tested. They are guaranteed by design characterization.

SWITCHING CHARACTERISTICS over operating ranges at 20 MHz

V_{CC} = 3.0 V to 5.5 V; T_{CASE} = 0°C to 100°C

			Preliminary			
Symbol	**Parameter Description**	**Notes**	**Ref. Figures**	**Min**	**Max**	**Unit**
	Operating Frequency	Half CLK2 freq.		0	20	MHz
1	CLK2 Period		3	25		ns
2a	CLK2 High Time	at 2 V	3	8		ns
2b	CLK2 High Time	at (V_{CC}−0.8 V)	3	5		ns
3a	CLK2 Low Time	at 2 V	3	8		ns
3b	CLK2 Low Time	at 0.8 V	3	6		ns
4	CLK2 Fall Time	(V_{CC}−0.8 V) to 0.8 V (Note 3)	3		8	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}−0.8 V) (Note 3)	3		8	ns
6	A23–A1 Valid Delay	C_L = 120 pF (Note 4)	6	4	30	ns
7	A23–A1 Float Delay	(Note 1)	10	4	32	ns
8	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Valid Delay	C_L = 75 pF (Note 4)	6	4	30	ns
9	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Float Delay	(Note 1)	10	4	32	ns
10a	M/$\overline{IO}$, D/$\overline{C}$ Valid Delay	C_L = 75 pF (Note 4)	6	4	28	ns
10b	W/$\overline{R}$, $\overline{ADS}$ Valid Delay	C_L = 75 pF (Note 4)	6	4	26	ns
10s	$\overline{SMIADS}$ Valid Delay	C_L = 75 pF (Note 4)	6	4	26	ns
11	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Float Delay	(Note 1)	10	6	30	ns
11s	$\overline{SMIADS}$ Float Delay	(Note 1)	10	4	30	ns
12	D15–D0 Write Data Valid Delay	C_L = 120 pF (Note 4)	6, 7, 9	4	38	ns
13	D15–D0 Write Data Float Delay	(Note 1)	10	4	27	ns
14	HLDA Valid Delay	C_L = 75 pF (Note 4)	6	4	28	ns
15	$\overline{NA}$ Setup Time		5	5		ns
16	$\overline{NA}$ Hold Time		5	12		ns
19	$\overline{READY}$ Setup Time		5	12		ns
19s	$\overline{SMIRDY}$ Setup Time		5	12		ns
20	$\overline{READY}$ Hold Time		5	4		ns
20s	$\overline{SMIRDY}$ Hold Tme		5	4		ns
21	D15–D0 Read Data Setup Time		5	9		ns
22	D15–D0 Read Data Hold Time		5	6		ns
23	HOLD Setup Time		5	17		ns
24	HOLD Hold Time		5	5		ns
25	RESET Setup Time		11	12		ns
26	RESET Hold Time		11	4		ns
27	NMI, INTR Setup Time	(Note 2)	5	16		ns
27s	$\overline{SMI}$ Setup Time	(Note 2)	5	16		ns
28	NMI, INTR Hold Time	(Note 2)	5	16		ns
28s	$\overline{SMI}$ Hold Time	(Note 2)	5	16		ns
29	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Setup Time	(Note 2)	5	14		ns
30	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$ Hold Time	(Note 2)	5	5		ns

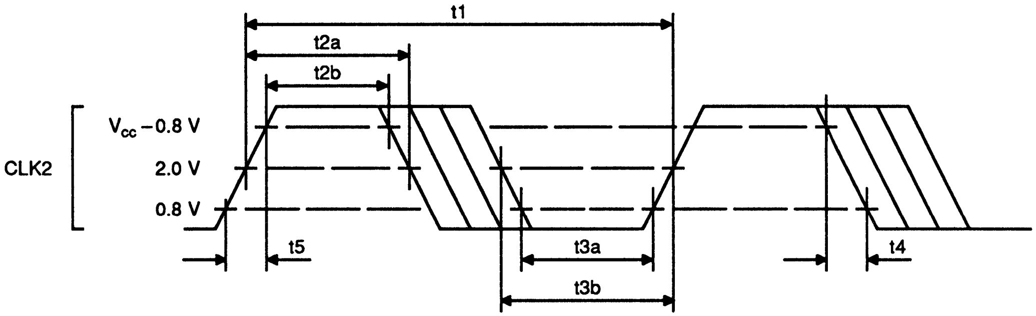
Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.

3. These are not tested. They are guaranteed by design characterization.

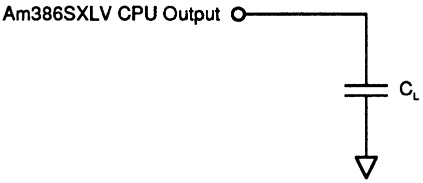
4. Tested with C_L set at 50 pF and derated to support the indicated distributed capacitive load. See Figures 12–14 for the capacitive derating curve.

SWITCHING CHARACTERISTICS (continued)



15022B–031

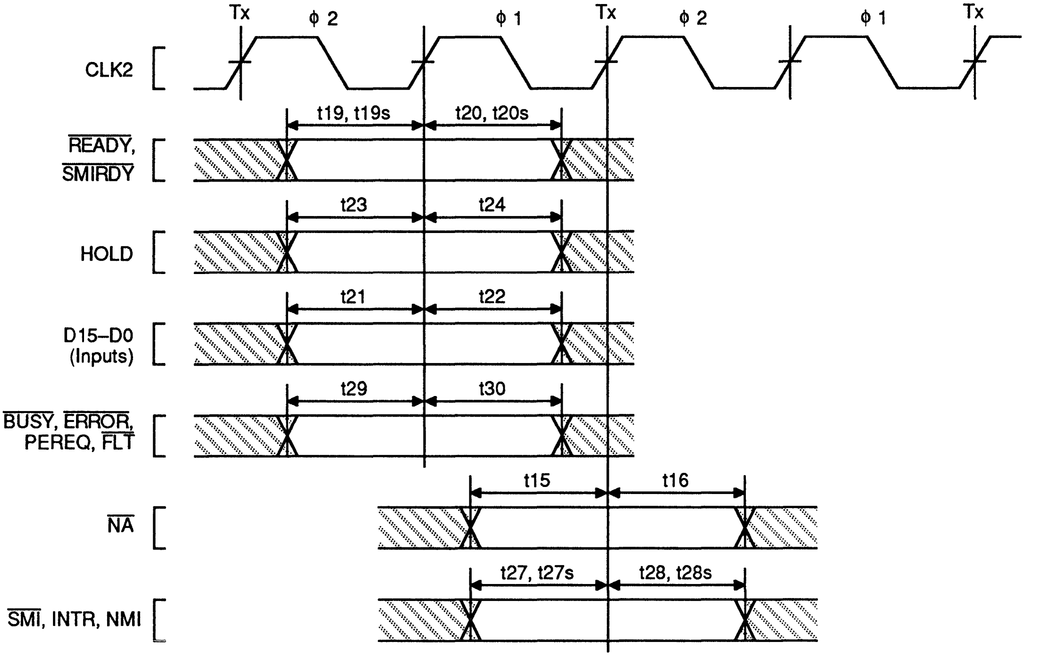
Figure 3. CLK2 Timing



15022B–032

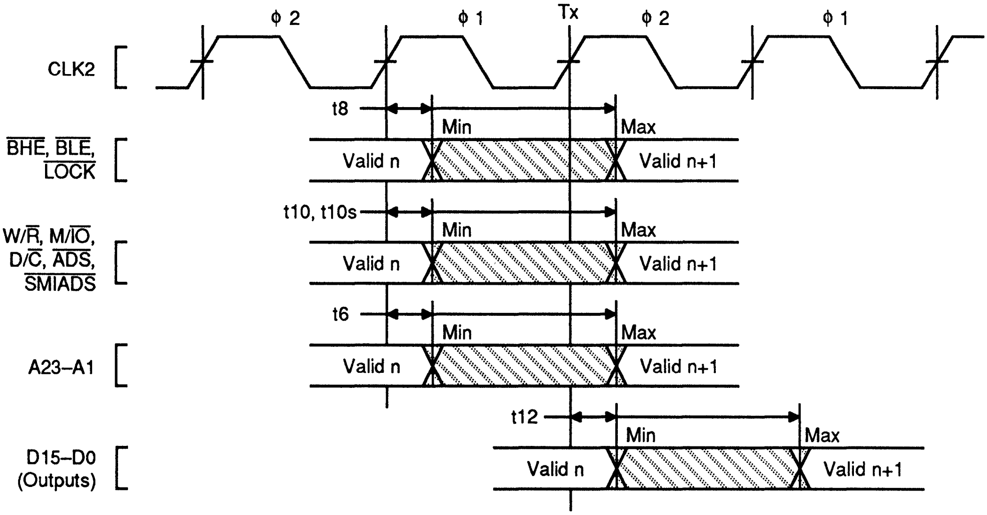
Figure 4. AC Test Circuit

SWITCHING WAVEFORMS



15022B–033

Figure 5. Input Setup and Hold Timing



15022B–034

Figure 6. Output Valid Delay Timing

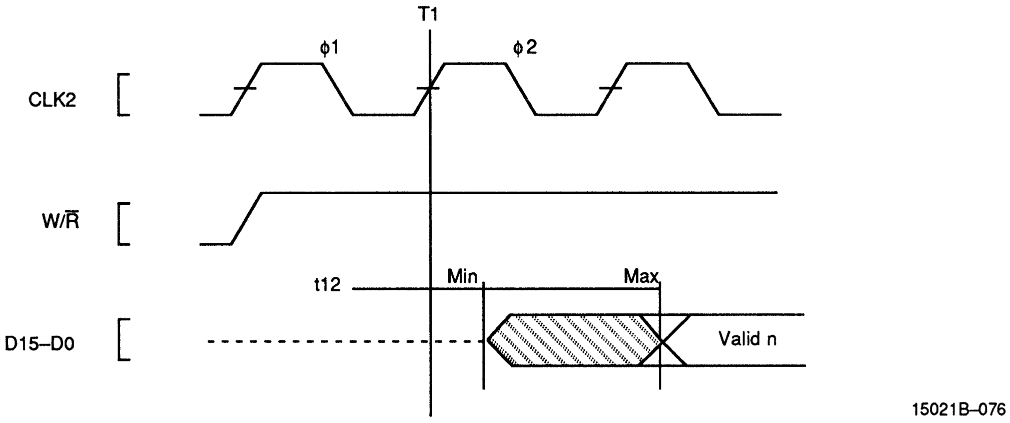


Figure 7. Write Data Valid Delay Timing (20 and 25 MHz)

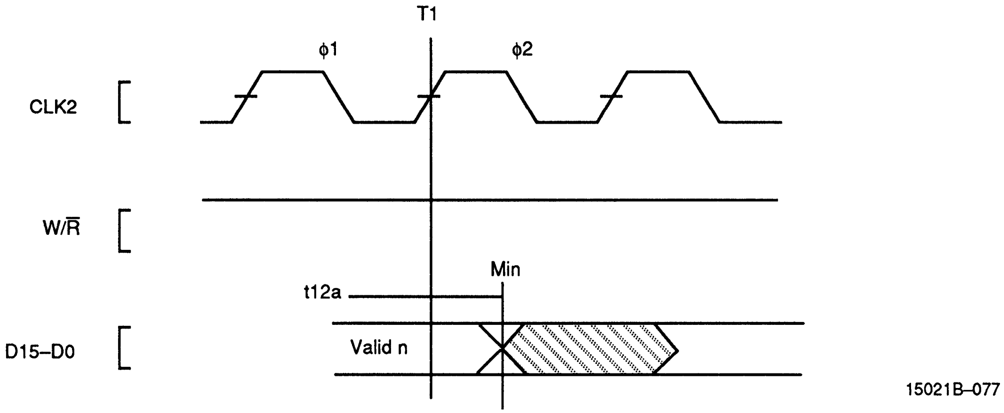


Figure 8. Write Data Hold Timing (20 and 25 MHz)

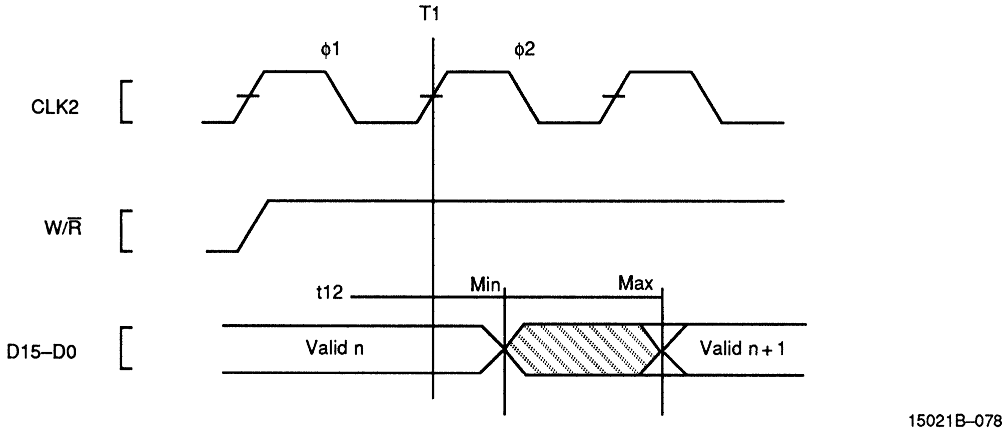
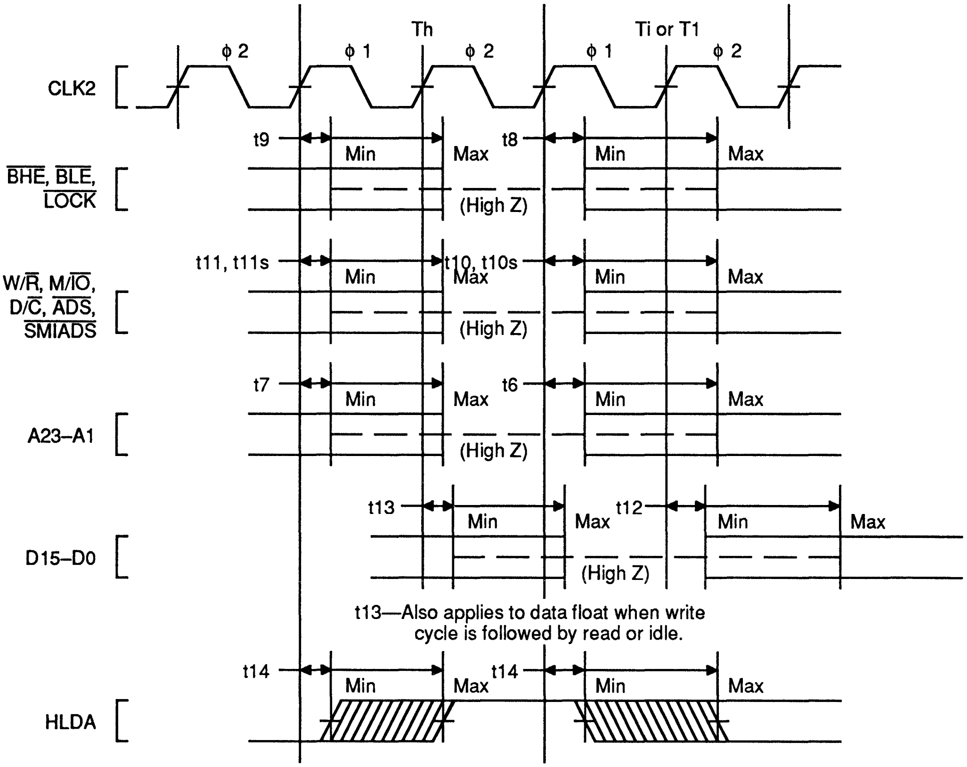


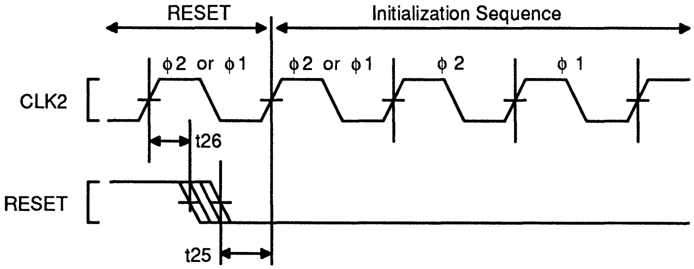
Figure 9. Write Data Valid Delay Timing (20 MHz)

SWITCHING WAVEFORMS (continued)



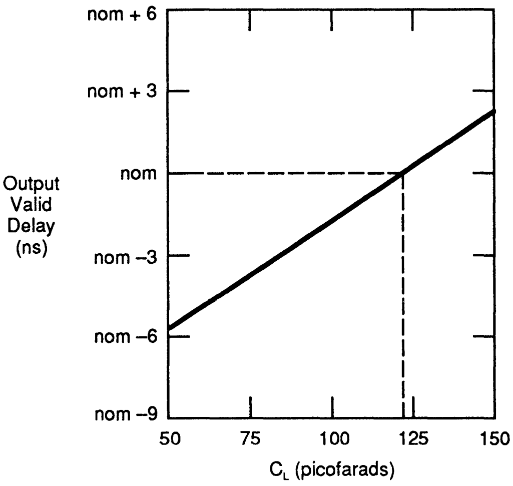
15022B–035

Figure 10. Output Float Delay and HLDA Valid Delay Timing



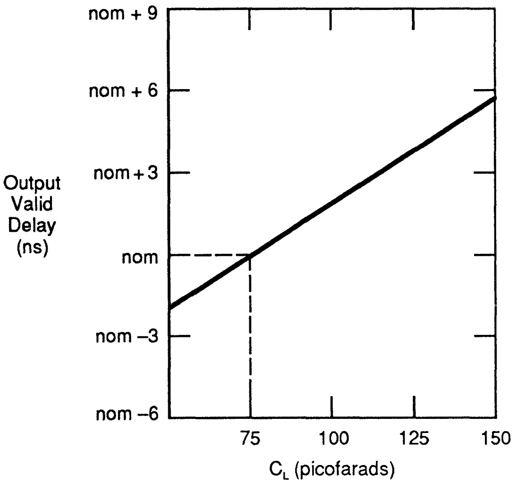
15022B–036

Figure 11. RESET Setup and Hold Timing and Internal Phase



15022B–037

Figure 12. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 120 pF)



15022B–038

Figure 13. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 75 pF)

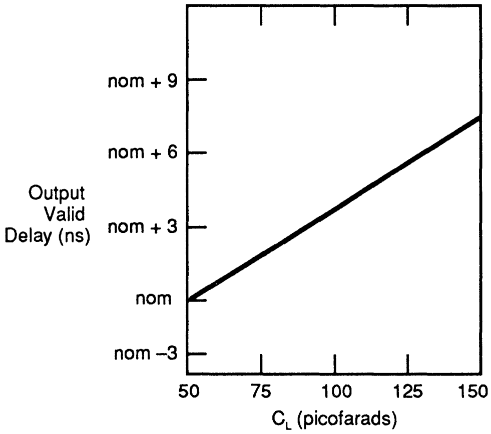


Figure 14. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 50 pF)

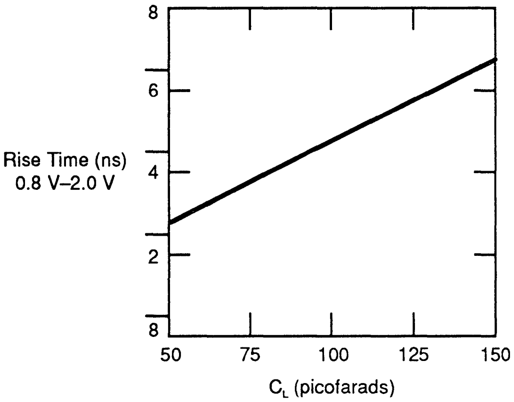


Figure 15. Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature

Am386™DX

High-Performance 32-Bit Microprocessor

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **Compatible with 386DX systems and software**
- **40-, 33-, 25-, and 20-MHz clock speeds**
- **32-bit microprocessor for personal computers and embedded systems**
- **32-bit address and data bus for high performance**
  –4-Gb physical address space
  –64-tb virtual address space
  –4-Gb maximum segment size
- **Supports 387DX-compatible math coprocessor**
- **AMD advanced 0.8 micron CMOS technology**
- **132-lead ceramic PGA package or optional 132-lead plastic quad flat pack (PQFP) package**

GENERAL DESCRIPTION

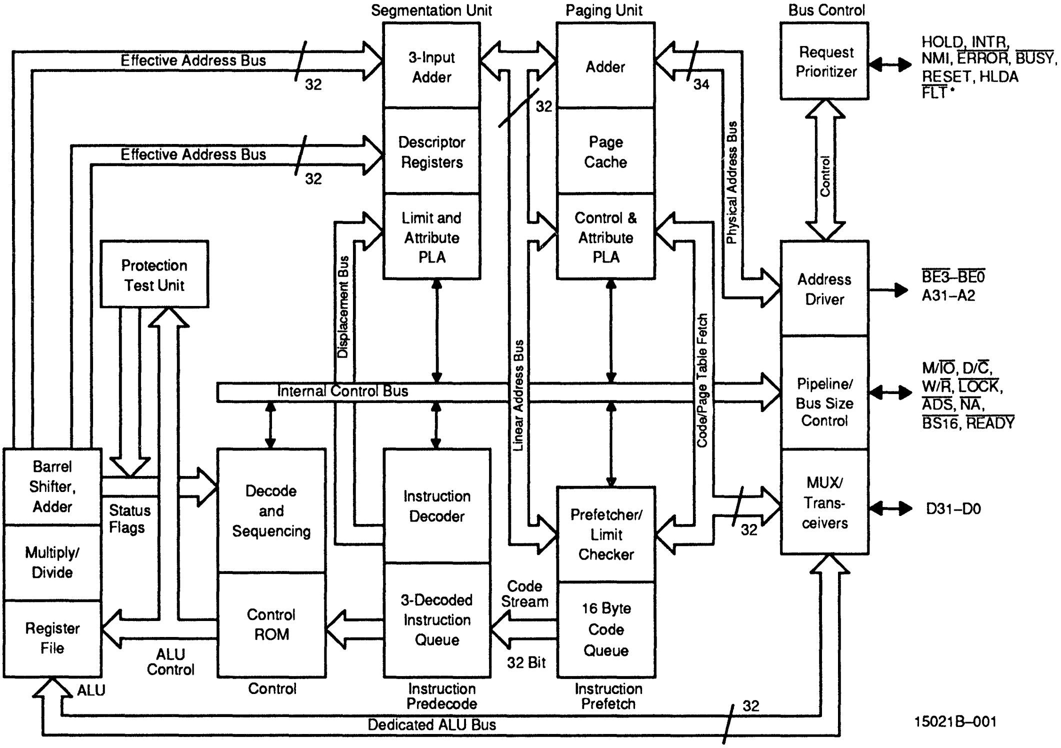
The Am386DX microprocessor is a compatible implementation of the Intel i386DX. It is engineered to meet strict requirements for compatibility. It is compatible with 386DX-based hardware, and is in fact a plug-in replacement for the Intel i386DX. It is also compatible with operating systems written for the 386 and the wide variety of commercially available software applications.

The Am386DX device is an advanced 32-bit microprocessor designed for applications needing very high performance. This device offers a 21% increase in performance from 33 to 40 MHz. The 32-bit registers and data paths support 32-bit addresses and data types. The processor addresses up to 4 Gb of physical memory and 64 tb of virtual memory. The integrated memory management and protection architecture allow high-performance execution of operating systems including DOS, Windows, OS/2, and UNIX.

The device is manufactured using the AMD advanced 0.8 micron CMOS process and is packaged in a standard 132-lead ceramic pin grid array (PGA) package. Additionally, the Am386DX microprocessor will be available in a small footprint 132-pin plastic quad flat pack (PQFP) package. This surface-mount package is 40% smaller than PGA allowing smaller, lower cost board designs without the need for a socket.

Publication # C15021 Rev. C Amendment /0
Issue Date: December 1991

BLOCK DIAGRAM



*Float feature available in Rev. C0 and later, PQFP only.

INTRODUCTION

AMD is proud to provide the Am386DX microprocessor at a time when the personal computer market requires alternatives. Alternate source manufacturers traditionally increase availability, add features, and broaden the market. AMD has focused significant engineering resources to bring you these benefits.

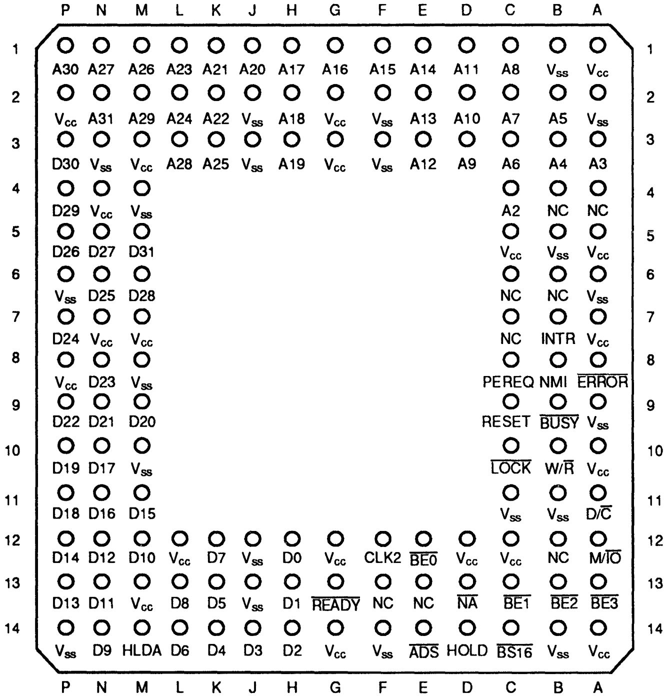
AMD's track record with the 80286 shows that we were first to raise the performance of the 80286 from 8 MHz to 10, 12, and 16 MHz. We were first to offer new packaging technology with a smaller, lower cost PLCC. Today, over 90% of all 80286s sold use PLCC packaging. AMD is committed to similar advances with the Am386DX microprocessor.

Another member of the family is the Am386DXL microprocessor. This device offers several enhancements including operating speeds up to 40 MHz, true static compatibility, and a small footprint cost-saving PQFP package. The Am386DXL microprocessor provides higher speeds and lower heat dissipation for desktop PCs, allowing the use of a smaller, or even, no cooling fan. For portables, its true static design allows for longer battery life with low power consumption and standby mode. (See Am386DXL CPU Data Sheet for more information—order #15484.)

AMD has engineered the Am386DX microprocessor to insure compatibility with the installed base of hardware and software for 386-based personal computers. The Am386DX microprocessor is your solution to meet the demand for high-performance, 32-bit personal computers.

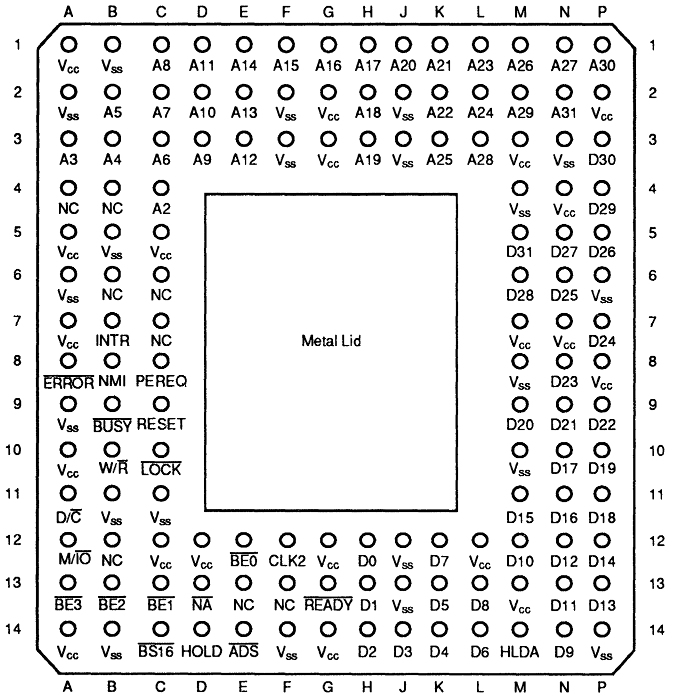
CONNECTION DIAGRAMS
132-Lead Ceramic Pin Grid Array (PGA) Package

Top Side View



	P	N	M	L	K	J	H	G	F	E	D	C	B	A
1	A30	A27	A26	A23	A21	A20	A17	A16	A15	A14	A11	A8	V_{ss}	V_{cc}
2	V_{cc}	A31	A29	A24	A22	V_{ss}	A18	V_{cc}	V_{ss}	A13	A10	A7	A5	V_{ss}
3	D30	V_{ss}	V_{cc}	A28	A25	V_{ss}	A19	V_{cc}	V_{ss}	A12	A9	A6	A4	A3
4	D29	V_{cc}	V_{ss}									A2	NC	NC
5	D26	D27	D31									V_{cc}	V_{ss}	V_{cc}
6	V_{ss}	D25	D28									NC	NC	V_{ss}
7	D24	V_{cc}	V_{cc}									NC	INTR	V_{cc}
8	V_{cc}	D23	V_{ss}									PEREQ	NMI	$\overline{ERROR}$
9	D22	D21	D20									RESET	$\overline{BUSY}$	V_{ss}
10	D19	D17	V_{ss}									$\overline{LOCK}$	W/$\overline{R}$	V_{cc}
11	D18	D16	D15									V_{ss}	V_{ss}	D/$\overline{C}$
12	D14	D12	D10	V_{cc}	D7	V_{ss}	D0	V_{cc}	CLK2	$\overline{BE0}$	V_{cc}	V_{cc}	NC	M/$\overline{IO}$
13	D13	D11	V_{cc}	D8	D5	V_{ss}	D1	$\overline{READY}$	NC	NC	$\overline{NA}$	$\overline{BE1}$	$\overline{BE2}$	$\overline{BE3}$
14	V_{ss}	D9	HLDA	D6	D4	D3	D2	V_{cc}	V_{ss}	$\overline{ADS}$	HOLD	$\overline{BS16}$	V_{ss}	V_{cc}

Pin Side View



	A	B	C	D	E	F	G	H	J	K	L	M	N	P
1	V_{cc}	V_{ss}	A8	A11	A14	A15	A16	A17	A20	A21	A23	A26	A27	A30
2	V_{ss}	A5	A7	A10	A13	V_{ss}	V_{cc}	A18	V_{ss}	A22	A24	A29	A31	V_{cc}
3	A3	A4	A6	A9	A12	V_{ss}	V_{cc}	A19	V_{ss}	A25	A28	V_{cc}	V_{ss}	D30
4	NC	NC	A2									V_{ss}	V_{cc}	D29
5	V_{cc}	V_{ss}	V_{cc}									D31	D27	D26
6	V_{ss}	NC	NC									D28	D25	V_{ss}
7	V_{cc}	INTR	NC									V_{cc}	V_{cc}	D24
8	$\overline{ERROR}$	NMI	PEREQ									V_{ss}	D23	V_{cc}
9	V_{ss}	$\overline{BUSY}$	RESET									D20	D21	D22
10	V_{cc}	W/$\overline{R}$	$\overline{LOCK}$									V_{ss}	D17	D19
11	D/$\overline{C}$	V_{ss}	V_{ss}									D15	D16	D18
12	M/$\overline{IO}$	NC	V_{cc}	V_{cc}	$\overline{BE0}$	CLK2	V_{cc}	D0	V_{ss}	D7	V_{cc}	D10	D12	D14
13	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{NA}$	NC	NC	$\overline{READY}$	D1	V_{ss}	D5	D8	V_{cc}	D11	D13
14	V_{cc}	V_{ss}	$\overline{BS16}$	HOLD	$\overline{ADS}$	V_{ss}	V_{cc}	D2	D3	D4	D6	HLDA	D9	V_{ss}

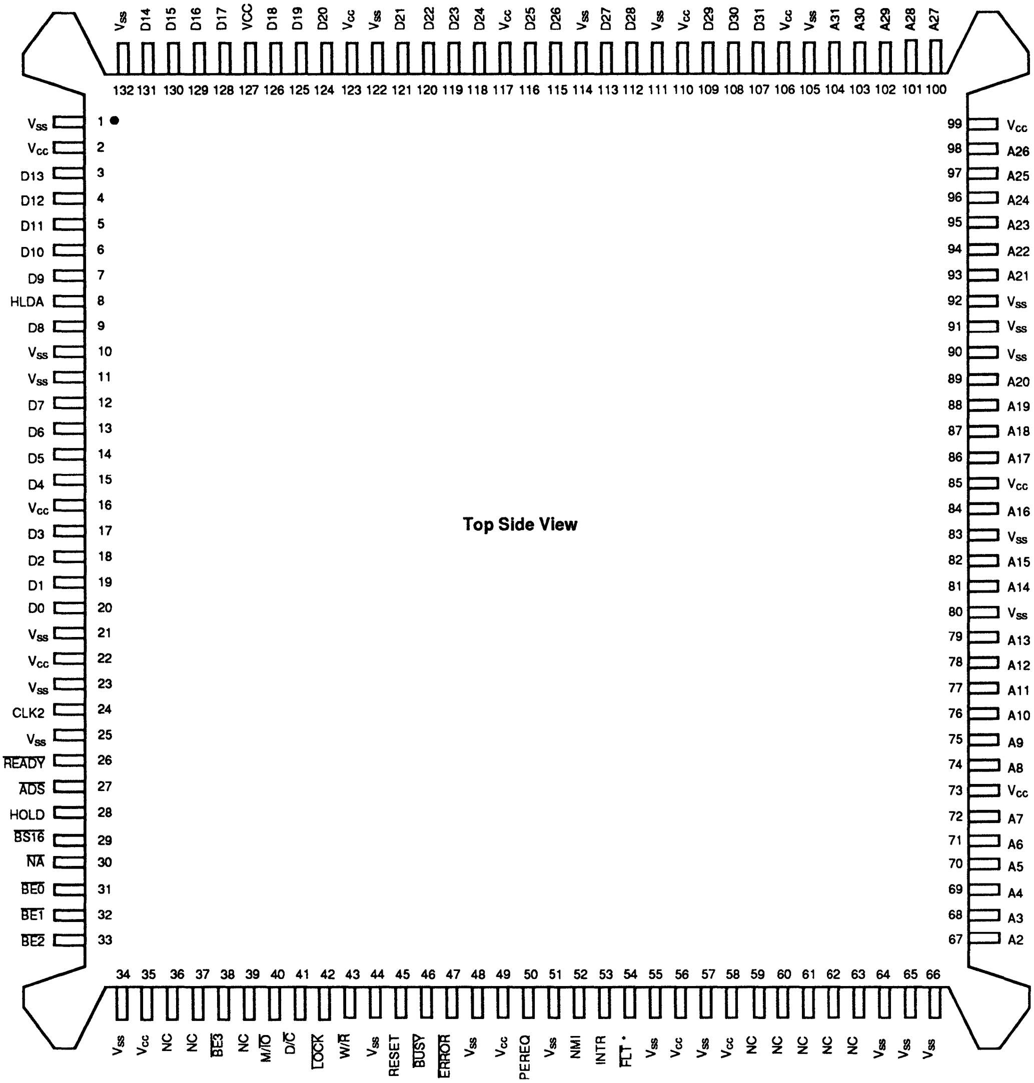
PGA Pin Designations (Functional Grouping)

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A2	C4	A24	L2	D6	L14	D28	M6	V_{cc}	C12	V_{ss}	F2
A3	A3	A25	K3	D7	K12	D29	P4		D12		F3
A4	B3	A26	M1	D8	L13	D30	P3		G2		F14
A5	B2	A27	N1	D9	N14	D31	M5		G3		J2
A6	C3	A28	L3	D10	M12	D/$\overline{\text{C}}$	A11		G12		J3
A7	C2	A29	M2	D11	N13	$\overline{\text{ERROR}}$	A8		G14		J12
A8	C1	A30	P1	D12	N12	HLDA	M14		L12		J13
A9	D3	A31	N2	D13	P13	HOLD	D14		M3		M4
A10	D2	$\overline{\text{ADS}}$	E14	D14	P12	INTR	B7		M7		M8
A11	D1	$\overline{\text{BE0}}$	E12	D15	M11	$\overline{\text{LOCK}}$	C10		M13		M10
A12	E3	$\overline{\text{BE1}}$	C13	D16	N11	M/$\overline{\text{IO}}$	A12		N4		N3
A13	E2	$\overline{\text{BE2}}$	B13	D17	N10	$\overline{\text{NA}}$	D13		N7		P6
A14	E1	$\overline{\text{BE3}}$	A13	D18	P11	NMI	B8		P2		P14
A15	F1	$\overline{\text{BS16}}$	C14	D19	P10	PEREQ	C8		P8	W/$\overline{\text{R}}$	B10
A16	G1	$\overline{\text{BUSY}}$	B9	D20	M9	$\overline{\text{READY}}$	G13	V_{ss}	A2	NC	A4
A17	H1	CLK2	F12	D21	N9	RESET	C9		A6		B4
A18	H2	D0	H12	D22	P9	V_{cc}	A1		A9		B6
A19	H3	D1	H13	D23	N8		A5		B1		B12
A20	J1	D2	H14	D24	P7		A7		B5		C6
A21	K1	D3	J14	D25	N6		A10		B11		C7
A22	K2	D4	K14	D26	P5		A14		B14		E13
A23	L1	D5	K13	D27	N5		C5		C11		F13

PGA Pin Designations (Sorted by Pin No.)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
A1	V_{cc}	B9	$\overline{\text{BUSY}}$	D3	A9	H1	A17	L13	D8	N7	V_{cc}
A2	V_{ss}	B10	W/$\overline{\text{R}}$	D12	V_{cc}	H2	A18	L14	D6	N8	D23
A3	A3	B11	V_{ss}	D13	$\overline{\text{NA}}$	H3	A19	M1	A26	N9	D21
A4	NC	B12	NC	D14	HOLD	H12	D0	M2	A29	N10	D17
A5	V_{cc}	B13	$\overline{\text{BE2}}$	E1	A14	H13	D1	M3	V_{cc}	N11	D16
A6	V_{ss}	B14	V_{ss}	E2	A13	H14	D2	M4	V_{ss}	N12	D12
A7	V_{cc}	C1	A8	E3	A12	J1	A20	M5	D31	N13	D11
A8	$\overline{\text{ERROR}}$	C2	A7	E12	$\overline{\text{BE0}}$	J2	V_{ss}	M6	D28	N14	D9
A9	V_{ss}	C3	A6	E13	NC	J3	V_{ss}	M7	V_{cc}	P1	A30
A10	V_{cc}	C4	A2	E14	$\overline{\text{ADS}}$	J12	V_{ss}	M8	V_{ss}	P2	V_{cc}
A11	D/$\overline{\text{C}}$	C5	V_{cc}	F1	A15	J13	V_{ss}	M9	D20	P3	D30
A12	M/$\overline{\text{IO}}$	C6	NC	F2	V_{ss}	J14	D3	M10	V_{ss}	P4	D29
A13	$\overline{\text{BE3}}$	C7	NC	F3	V_{ss}	K1	A21	M11	D15	P5	D26
A14	V_{cc}	C8	PEREQ	F12	CLK2	K2	A22	M12	D10	P6	V_{ss}
B1	V_{ss}	C9	RESET	F13	NC	K3	A25	M13	V_{cc}	P7	D24
B2	A5	C10	$\overline{\text{LOCK}}$	F14	V_{ss}	K12	D7	M14	HLDA	P8	V_{cc}
B3	A4	C11	V_{ss}	G1	A16	K13	D5	N1	A27	P9	D22
B4	NC	C12	V_{cc}	G2	V_{cc}	K14	D4	N2	A31	P10	D19
B5	V_{ss}	C13	$\overline{\text{BE1}}$	G3	V_{cc}	L1	A23	N3	V_{ss}	P11	D18
B6	NC	C14	$\overline{\text{BS16}}$	G12	V_{cc}	L2	A24	N4	V_{cc}	P12	D14
B7	INTR	D1	A11	G13	$\overline{\text{READY}}$	L3	A28	N5	D27	P13	D13
B8	NMI	D2	A10	G14	V_{cc}	L12	V_{cc}	N6	D25	P14	V_{ss}

CONNECTION DIAGRAMS (continued)
132-Lead Plastic Quad Flat Pack (PQFP) Package

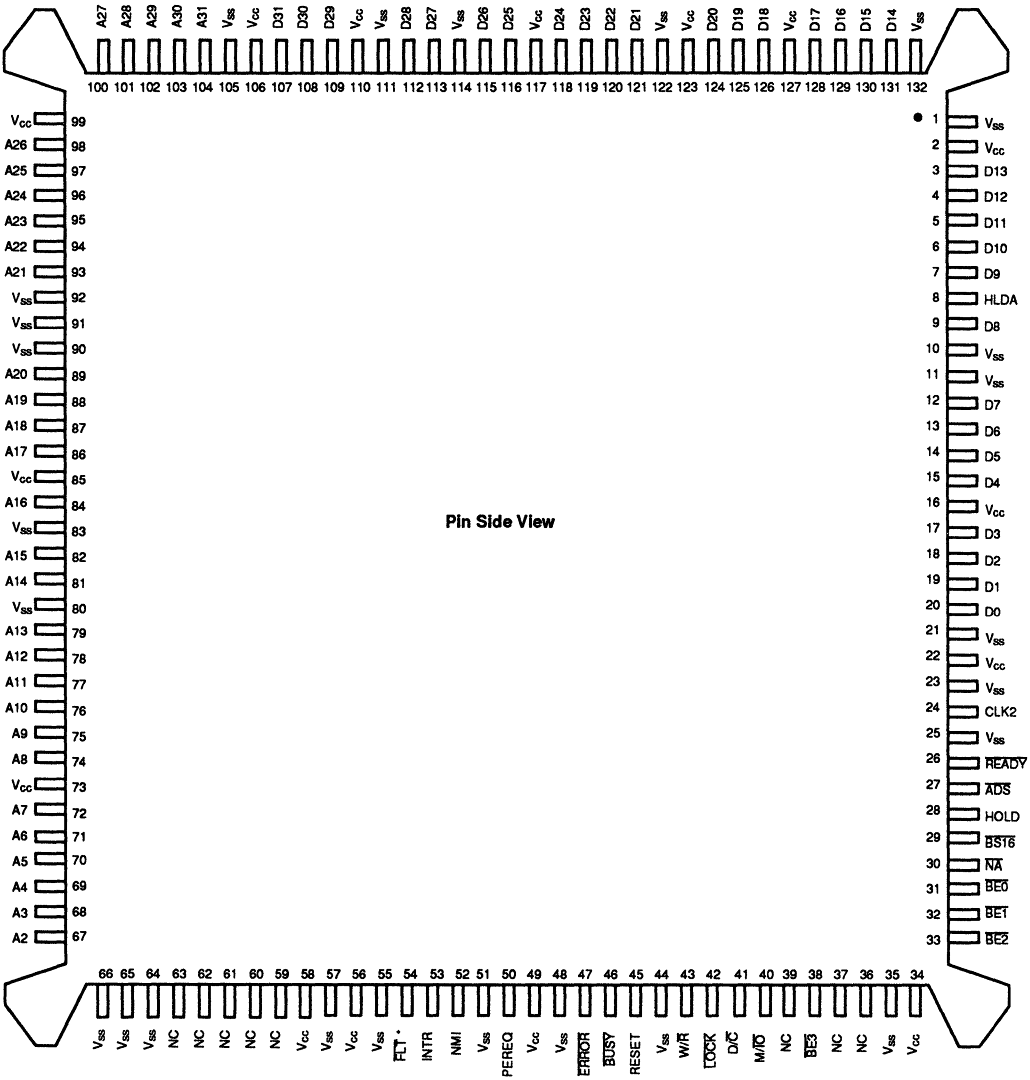


15022B–002

Notes: Pin 1 is marked for orientation.
*Float feature available in Rev. C0 and later, PQFP only.

CONNECTION DIAGRAMS (continued)

132-Lead Plastic Quad Flat Pack (PQFP) Package



15022B–002

Notes: Pin 1 is marked for orientation.
*Float feature available in Rev. C0 and later, PQFP only.

PQFP Pin Designation (Functional Grouping)

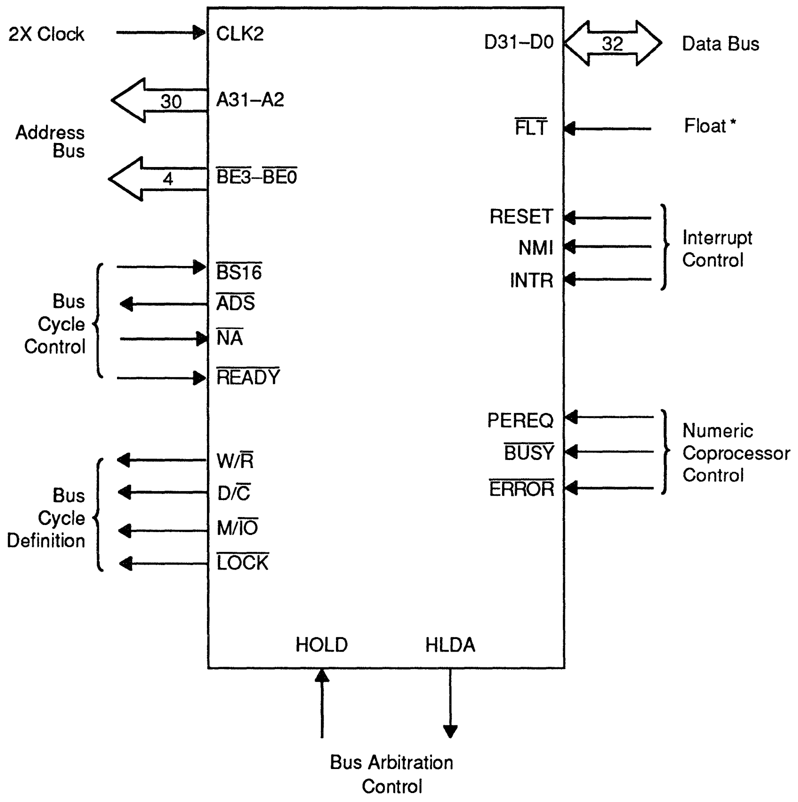
Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A2	67	A24	96	D6	13	D28	112	V_{CC}	56	V_{SS}	64
A3	68	A25	97	D7	12	D29	109		58		65
A4	69	A26	98	D8	9	D30	108		73		66
A5	70	A27	100	D9	7	D31	107		85		80
A6	71	A28	101	D10	6	D/$\overline{C}$	41		99		83
A7	72	A29	102	D11	5	$\overline{ERROR}$	47		106		90
A8	74	A30	103	D12	4	HLDA	8		110		91
A9	75	A31	104	D13	3	HOLD	28		117		92
A10	76	$\overline{ADS}$	27	D14	131	INTR	53		123		105
A11	77	$\overline{BE0}$	31	D15	130	$\overline{LOCK}$	42		127		111
A12	78	$\overline{BE1}$	32	D16	129	M/$\overline{IO}$	40	V_{SS}	1		114
A13	79	$\overline{BE2}$	33	D17	128	$\overline{NA}$	30		10		122
A14	81	$\overline{BE3}$	38	D18	126	NMI	52		11		132
A15	82	$\overline{BS16}$	29	D19	125	PEREQ	50		21	W/$\overline{R}$	43
A16	84	$\overline{BUSY}$	46	D20	124	$\overline{READY}$	26		23	NC	36
A17	86	CLK2	24	D21	121	RESET	45		25		37
A18	87	D0	20	D22	120	V_{CC}	2		35		39
A19	88	D1	19	D23	119		16		44		59
A20	89	D2	18	D24	118		22		48		60
A21	93	D3	17	D25	116		34		51		61
A22	94	D4	15	D26	115		49		55		62
A23	95	D5	14	D27	113	$\overline{FLT}$*	54		57		63

PQFP Pin Designation (Sorted by Pin No.)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	V_{SS}	23	V_{SS}	45	RESET	67	A2	89	A20	111	V_{SS}
2	V_{CC}	24	CLK2	46	$\overline{BUSY}$	68	A3	90	V_{SS}	112	D28
3	D13	25	V_{SS}	47	$\overline{ERROR}$	69	A4	91	V_{SS}	113	D27
4	D12	26	$\overline{READY}$	48	V_{SS}	70	A5	92	V_{SS}	114	VSS
5	D11	27	$\overline{ADS}$	49	V_{CC}	71	A6	93	A21	115	D26
6	D10	28	HOLD	50	PEREQ	72	A7	94	A22	116	D25
7	D9	29	$\overline{BS16}$	51	V_{SS}	73	V_{CC}	95	A23	117	V_{CC}
8	HLDA	30	$\overline{NA}$	52	NMI	74	A8	96	A24	118	D24
9	D8	31	$\overline{BE0}$	53	INTR	75	A9	97	A25	119	D23
10	V_{SS}	32	$\overline{BE1}$	54	$\overline{FLT}$*	76	A10	98	A26	120	D22
11	V_{SS}	33	$\overline{BE2}$	55	V_{SS}	77	A11	99	V_{CC}	121	D21
12	D7	34	V_{CC}	56	V_{CC}	78	A12	100	A27	122	V_{SS}
13	D6	35	V_{SS}	57	V_{SS}	79	A13	101	A28	123	V_{CC}
14	A5	36	NC	58	V_{CC}	80	V_{SS}	102	A29	124	D20
15	D4	37	NC	59	NC	81	A14	103	A30	125	D19
16	V_{CC}	38	$\overline{BE3}$	60	NC	82	A15	104	A31	126	D18
17	D3	39	NC	61	NC	83	V_{SS}	105	V_{SS}	127	V_{CC}
18	D2	40	M/$\overline{IO}$	62	NC	84	A16	106	V_{CC}	128	D17
19	D1	41	D/$\overline{C}$	63	NC	85	V_{CC}	107	D31	129	D16
20	D0	42	$\overline{LOCK}$	64	V_{SS}	86	A17	108	D30	130	D15
21	V_{SS}	43	W/$\overline{R}$	65	V_{SS}	87	A18	109	D29	131	D14
22	V_{CC}	44	V_{SS}	66	V_{SS}	88	A19	110	V_{CC}	132	V_{SS}

*Float feature available in Rev. C0 and later, PQFP only.

LOGIC SYMBOL



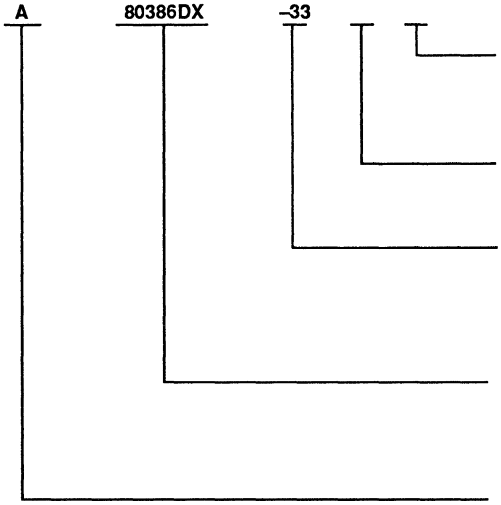
15021B–003

*Float feature available in Rev. C0 and later, PQFP only.

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



OPTIONAL PROCESSING

None = Trimmed and Formed PQFP in High Temp Trays
F = Ringed PQFP in Horizontal Tubes
S = Ringed PQFP in Coin-Stack Tubes

TEMPERATURE RANGE

Blank = Commercial (T_{CASE} = 0 to +85°C)

SPEED OPTION

–40 = 40 MHz
–33 = 33 MHz
–25 = 25 MHz
–20 = 20 MHz

DEVICE NUMBER/DESCRIPTION

80386DX
Am386DX High-Performance, 32-Bit Microprocessor

PACKAGE TYPE

A = 132-Lead Pin Grid Array (CGX 132)
NG = 132-Lead Plastic Quad Flat Pack (PQ132, PQB132)

Valid Combinations		
A	80386DX	–40 –33 –25 –20
NG	80386DX	–40F, –40S –33F, –33S –25F, –25S –20F, –20S

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations.

PIN DESCRIPTION

A31–A2
Address Bus (Outputs)

Outputs physical memory or port I/O addresses.

$\overline{\text{ADS}}$
Address Status (Active Low; Output)

Indicates that a valid bus cycle definition and address ($W/\overline{R}$, $D/\overline{C}$, $M/\overline{IO}$, $\overline{BE0}$, $\overline{BE1}$, $\overline{BE2}$, $\overline{BE3}$, and A31–A2) are being driven at the Am386DX microprocessor pins.

$\overline{\text{BE3}}$–$\overline{\text{BE0}}$
Byte Enables (Active Low; Outputs)

Indicate which data bytes of the data bus take part in a bus cycle.

$\overline{\text{BS16}}$
Bus Size 16 (Active Low; Input)

Allows direct connection of 32-bit and 16-bit data buses.

$\overline{\text{BUSY}}$
Busy (Active Low; Input)

Signals a busy condition from a processor extension.

CLK2
Clock (Input)

Provides the fundamental timing for the Am386DX microprocessor.

D31–D0
Data Bus (Inputs/Outputs)

Inputs data during memory, I/O, and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.

$D/\overline{C}$
Data/Control (Output)

A bus cycle definition pin that distinguishes data cycles, either memory or I/O from control cycles which are: interrupt acknowledge, halt, and instruction fetching.

$\overline{\text{ERROR}}$
Error (Active Low; Input)

Signals an error condition from a processor extension.

$\overline{\text{FLT}}$*
Float (Active Low; Input)

An input signal which forces all bi-directional and output signals, including HLDA, to the three-state condition. $\overline{\text{FLT}}$ has an internal pull-up resistor, and if it is not used it should be unconnected.

HLDA
Bus Hold Acknowledge (Active High; Output)

Indicates that the Am386DX microprocessor surrendered control of its local bus to another bus master.

HOLD
Bus Hold Request (Active High; Input)

Allows another bus master to request control of the local bus.

INTR
Interrupt Request (Active High; Input)

A maskable input that signals the Am386DX microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

$\overline{\text{LOCK}}$
Bus Lock (Active Low; Output)

A bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active.

$M/\overline{IO}$
Memory I/O (Output)

A bus cycle definition pin that distinguishes memory cycles from input/output cycles.

$\overline{\text{NA}}$
Next Address (Active Low; Input)

Used to request address pipelining.

NC
No Connect

Should always remain unconnected. Connection of a NC pin may cause the processor to malfunction or be incompatible with future steppings of the Am386DX microprocessor.

NMI
Non-Maskable Interrupt Request (Active High; Input)

A non-maskable input that signals the Am386DX microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

PEREQ
Processor Extension Request (Active High; Input)

Indicates that the processor extension has data to be transferred by the Am386DX microprocessor.

$\overline{\text{READY}}$
Bus Ready (Active Low; Input)

Terminates the bus cycle.

RESET
Reset (Active High; Input)

Suspends any operation in progress and places the Am386DX microprocessor in a known reset state.

V_{CC}
System Power (Active High; Input)

Provides the +5 V nominal DC supply input.

V_{SS}
System Ground (Input)

System ground provides 0 V connection from which all inputs and outputs are measured.

$W/\overline{R}$
Write/Read (Output)

A bus cycle definition pin that distinguishes write cycles from read cycles.

*Float feature available in Rev. C0 and later, PQFP only.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature . . . . –65°C to +150°C
Ambient Temperature Under Bias . . –65°C to +125°C
Supply Voltage with Respect
to Vss . . . . –0.5 V to +7 V
Voltage on Other Pins . . . . –0.5 V to Vcc +0.5 V

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to Absolute Maximum Ratings for extended periods may affect device reliability.

DC CHARACTERISTICS over COMMERCIAL operating ranges

V_{CC} = 5 V ±5%; T_{CASE} = 0°C to +85°C (PGA)
V_{CC} = 5 V ±10%; T_{CASE} = 0°C to +100°C (PQFP)

Symbol	Parameter Description	Notes	Min	Max	Unit
V_{IL}	Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.3	V
V_{ILC}	CLK2 Input Low Voltage	(Note 1)	–0.3	0.8	V
V_{IHC}	CLK2 Input High Voltage				
	20 MHz		V_{CC} – 0.8	V_{CC} + 0.3	V
	25, 33, and 40 MHz		3.7	V_{CC} + 0.3	V
V_{OL}	Output Low Voltage				
	I_{OL} = 4 mA: A31–A2, D31–D0			0.45	V
	I_{OL} = 5 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, HLDA			0.45	V
V_{OH}	Output High Voltage				
	I_{OH} = 1 mA: A31–A2, D31–D0		2.4		V
	I_{OH} = 0.9 mA: $\overline{\text{BE3}}$–$\overline{\text{BE0}}$, W/$\overline{\text{R}}$, D/$\overline{\text{C}}$, M/$\overline{\text{IO}}$, $\overline{\text{LOCK}}$, $\overline{\text{ADS}}$, HLDA		2.4		V
I_{LI}	Input Leakage Current (All pins except $\overline{\text{BS16}}$, PEREQ, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$*, and $\overline{\text{ERROR}}$)	0 V ≤ V_{IN} ≤ V_{CC}		±15	µA
I_{IH}	Input Leakage Current (PEREQ Pin)	V_{IH} = 2.4 V (Note 2)		200	µA
I_{IL}	Input Leakage Current ($\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$*, and $\overline{\text{ERROR}}$)	V_{IL} = 0.45 (Note 3)		–400	µA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		±15	µA
I_{CC}	Supply Current				
	CLK2 = 40 MHz: with –20	I_{CC} Typ = 350		500	mA
	CLK2 = 50 MHz: with –25	I_{CC} Typ = 375		550	mA
	CLK2 = 66 MHz: with –33	I_{CC} Typ = 400		550	mA
	CLK2 = 80 MHz: with –40	I_{CC} Typ = 400		550	mA
C_{IN}	Input or I/O Capacitance	F_C = 1 MHz (Note 4)		10	pF
C_{OUT}	Output Capacitance	F_C = 1 MHz (Note 4)		12	pF
C_{CLK}	CLK2 Capacitance	F_C = 1 MHz (Note 4)		20	pF

Notes: 1. The Min value, –0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. $\overline{\text{BS16}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$*, and $\overline{\text{ERROR}}$ inputs each have an internal pullup resistor.
4. Not 100% tested.
*Float feature available in Rev. Co and later, PQFP only.

SWITCHING CHARACTERISTIC over operating range

$V_{CC} = 5$ V ±5%; $T_{CASE} = 0°C$ to +85°C

No.	Parameter Description	Notes	Ref Figure	40 MHz Min	40 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		8	40	MHz
1	CLK2 Period		3	12.5	62.5	ns
2a	CLK2 High Time	at 2 V	3	5		ns
2b	CLK2 High Time	at 3.7 V	3	3.25		ns
3a	CLK2 Low Time	at 2 V	3	5		ns
3b	CLK2 Low Time	at 0.8 V	3	3.25		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	3		4	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	3		4	ns
6	A31–A2 Valid Delay	$C_L = 50$ pF	2, 5, 13	4	13	ns
7	A31–A2 Float Delay	(Note 1)	13	4	20	ns
8	$\overline{\mathrm{BE3}}$–$\overline{\mathrm{BE0}}$, $\overline{\mathrm{LOCK}}$ Valid Delay	$C_L = 50$ pF	2, 5, 13	4	13	ns
9	$\overline{\mathrm{BE3}}$–$\overline{\mathrm{BE0}}$, $\overline{\mathrm{LOCK}}$ Float Delay	(Note 1)	13	4	20	ns
10	W/$\overline{\mathrm{R}}$, M/$\overline{\mathrm{IO}}$, D/$\overline{\mathrm{C}}$ Valid Delay	$C_L = 50$ pF	2, 5, 13	4	13	ns
10a	$\overline{\mathrm{ADS}}$ Valid Delay	$C_L = 50$ pF	2, 5, 13	4	13	ns
11	W/$\overline{\mathrm{R}}$, M/$\overline{\mathrm{IO}}$, D/$\overline{\mathrm{C}}$, $\overline{\mathrm{ADS}}$ Float Delay	(Note 1)	13	4	20	ns
12	D31–D0 Write Data Valid Delay	$C_L = 50$ pF (Note 4)	2, 6, 13	7	18	ns
12a	D31–D0 Write Data Hold Time	$C_L = 50$ pF	2, 7	2		ns
13	D31–D0 Float Delay	(Note 1)	13	4	17	ns
14	HLDA Valid Delay	$C_L = 50$ pF	2, 13	4	17	ns
15	$\overline{\mathrm{NA}}$ Setup Time		4	5		ns
16	$\overline{\mathrm{NA}}$ Hold Time		4	2		ns
17	$\overline{\mathrm{BS16}}$ Setup Time		4	5		ns
18	$\overline{\mathrm{BS16}}$ Hold Time		4	2		ns
19	$\overline{\mathrm{READY}}$ Setup Time		4	7		ns
20	$\overline{\mathrm{READY}}$ Hold Time		4	4		ns
21	D31–D0 Read Setup Time		4	4		ns
22	D31–D0 Read Hold Time		4	3		ns
23	HOLD Setup Time		4	4		ns
24	HOLD Hold Time		4	2		ns
25	RESET Setup Time		14	4		ns
26	RESET Hold Time		14	2		ns
27	NMI, INTR Setup Time	(Note 2)	14	5		ns
28	NMI, INTR Hold Time	(Note 2)	14	5		ns
29	PEREQ, $\overline{\mathrm{ERROR}}$, $\overline{\mathrm{BUSY}}$ Setup Time	(Note 2)	4	5		ns
30	PEREQ, $\overline{\mathrm{ERROR}}$, $\overline{\mathrm{BUSY}}$ Hold Time	(Note 2)	4	4		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.

2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific clock period.

3. Rise and fall times are not tested.

4. Min time not 100% tested.

SWITCHING CHARACTERISTIC over operating range

V_{CC} = 5 V ±5%; T_{CASE} = 0°C to +85°C

No.	Parameter Description	Notes	Ref Figures	33 MHz Min	33 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		8	33.3	MHz
1	CLK2 Period		3	15.0	62.5	ns
2a	CLK2 High Time	at 2 V	3	6.25		ns
2b	CLK2 High Time	at 3.7 V	3	4.5		ns
3a	CLK2 Low Time	at 2 V	3	6.25		ns
3b	CLK2 Low Time	at 0.8 V	3	4.5		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	3		4	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	3		4	ns
6	A31–A2 Valid Delay	C_L = 50 pF	2, 5, 13	4	15	ns
7	A31–A2 Float Delay	(Note 1)	13	4	20	ns
8	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 50 pF	2, 5, 13	4	15	ns
9	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	13	4	20	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$ Valid Delay	C_L = 50 pF	2, 5, 13	4	15	ns
10a	$\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	2, 5, 13	4	14.5	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	13	4	20	ns
12	D31–D0 Write Data Valid Delay	C_L = 50 pF (Note 4)	2, 6, 13	7	24	ns
12a	D31–D0 Write Data Hold Time	C_L = 50 pF	2, 7	2		ns
13	D31–D0 Float Delay	(Note 1)	13	4	17	ns
14	HLDA Valid Delay	C_L = 50 pF	2, 13	4	20	ns
15	$\overline{\text{NA}}$ Setup Time		4	5		ns
16	$\overline{\text{NA}}$ Hold Time		4	2		ns
17	$\overline{\text{BS16}}$ Setup Time		4	5		ns
18	$\overline{\text{BS16}}$ Hold Time		4	2		ns
19	$\overline{\text{READY}}$ Setup Time		4	7		ns
20	$\overline{\text{READY}}$ Hold Time		4	4		ns
21	D31–D0 Read Setup Time		4	5		ns
22	D31–D0 Read Hold Time		4	3		ns
23	HOLD Setup Time		4	11		ns
24	HOLD Hold Time		4	2		ns
25	RESET Setup Time		14	5		ns
26	RESET Hold Time		14	2		ns
27	NMI, INTR Setup Time	(Note 2)	4	5		ns
28	NMI, INTR Hold Time	(Note 2)	4	5		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$ Setup Time	(Note 2)	4	5		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$ Hold Time	(Note 2)	4	4		ns

Notes:
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.
4. Min time not 100% tested.

SWITCHING CHARACTERISTIC over operating range

Vcc = 5 V ±5%; TCASE = 0°C to +85°C (PGA)
Vcc = 5 V ±10%; TCASE = 0°C to +100°C (PQFP)

No.	Parameter Description	Notes	Ref Figures	25 MHz Min	25 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		4	25	MHz
1	CLK2 Period		3	20	125	ns
2a	CLK2 High Time	at 2 V	3	7		ns
2b	CLK2 High Time	at 3.7 V	3	4		ns
3a	CLK2 Low Time	at 2 V	3	7		ns
3b	CLK2 Low Time	at 0.8 V	3	5		ns
4	CLK2 Fall Time	3.7 V to 0.8 V (Note 3)	3		7	ns
5	CLK2 Rise Time	0.8 V to 3.7 V (Note 3)	3		7	ns
6	A31–A2 Valid Delay	C_L = 50 pF	2, 5, 13	4	21	ns
7	A31–A2 Float Delay	(Note 1)	13	4	30	ns
8	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$ Valid Delay	C_L = 50 pF	2, 5, 13	4	24	ns
8a	$\overline{\text{LOCK}}$ Valid Delay	CL = 50 pF	2, 5, 13	4	21	ns
9	$\overline{\text{BE3}}$–$\overline{\text{BE0}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	13	4	30	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	2, 5, 13	4	21	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	13	4	30	ns
12	D31–D0 Write Data Valid Delay	C_L = 50 pF	2, 6, 13	7	27	ns
12a	D31–D0 Write Data Hold Time	C_L = 50 pF	2, 7	2		ns
13	D31–D0 Float Delay	(Note 1)	13	4	22	ns
14	HLDA Valid Delay	C_L = 50 pF	2, 13	4	22	ns
15	$\overline{\text{NA}}$ Setup Time		4	7		ns
16	$\overline{\text{NA}}$ Hold Time		4	3		ns
17	$\overline{\text{BS16}}$ Setup Time		4	7		ns
18	$\overline{\text{BS16}}$ Hold Time		4	3		ns
19	$\overline{\text{READY}}$ Setup Time		4	9		ns
20	$\overline{\text{READY}}$ Hold Time		4	4		ns
21	D31–D0 Read Setup Time		4	7		ns
22	D31–D0 Read Hold Time		4	5		ns
23	HOLD Setup Time		4	15		ns
24	HOLD Hold Time		4	3		ns
25	RESET Setup Time		14	10		ns
26	RESET Hold Time		14	3		ns
27	NMI, INTR Setup Time	(Note 2)	4	6		ns
28	NMI, INTR Hold Time	(Note 2)	4	6		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$* Setup Time	(Note 2)	4	6		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$* Hold Time	(Note 2)	4	5		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.
*Float feature available in Rev C0 and later, PQFP only.

SWITCHING CHARACTERISTIC over operating range

V_{CC} = 5 V ±5%; T_{CASE} = 0°C to +85°C (PGA)
V_{CC} = 5 V ±10%; T_{CASE} = 0°C to +100°C (PQFP)

No.	Parameter Description	Notes	Ref Figures	20 MHz Min	20 MHz Max	Unit
	Operating Frequency	Half of CLK2 Freq		4	20	MHz
1	CLK2 Period		3	25	125	ns
2a	CLK2 High Time	at 2 V	3	8		ns
2b	CLK2 High Time	at (V_{CC}–0.8 V)	3	5		ns
3a	CLK2 Low Time	at 2 V	3	8		ns
3b	CLK2 Low Time	at 0.8 V	3	6		ns
4	CLK2 Fall Time	(V_{CC}–0.8 V) to 0.8 V (Note 3)	3		8	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}–0.8 V) (Note 3)	3		8	ns
6	A31–A2 Valid Delay	C_L = 120 pF	2, 5, 13	4	30	ns
7	A31–A2 Float Delay	(Note 1)	13	4	32	ns
8	$\overline{BE3}$–$\overline{BE0}$, $\overline{LOCK}$ Valid Delay	C_L = 75 pF	2, 5, 13	4	30	ns
9	$\overline{BE3}$–$\overline{BE0}$, $\overline{LOCK}$ Float Delay	(Note 1)	13	4	32	ns
10	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Valid Delay	C_L = 75 pF	2, 5, 13	4	28	ns
11	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Float Delay	(Note 1)	13	4	30	ns
12	D31–D0 Write Data Valid Delay	C_L = 120 pF	2, 8, 13	4	38	ns
13	D31–D0 Float Delay	(Note 1)	13	4	27	ns
14	HLDA Valid Delay	C_L = 75 pF	2, 13	6	28	ns
15	$\overline{NA}$ Setup Time		4	9		ns
16	$\overline{NA}$ Hold Time		4	14		ns
17	$\overline{BS16}$ Setup Time		4	13		ns
18	$\overline{BS16}$ Hold Time		4	21		ns
19	$\overline{READY}$ Setup Time		4	12		ns
20	$\overline{READY}$ Hold Time		4	4		ns
21	D31–D0 Read Setup Time		4	11		ns
22	D31–D0 Read Hold Time		4	6		ns
23	HOLD Setup Time		4	17		ns
24	HOLD Hold Time		4	5		ns
25	RESET Setup Time		14	12		ns
26	RESET Hold Time		14	4		ns
27	NMI, INTR Setup Time	(Note 2)	4	16		ns
28	NMI, INTR Hold Time	(Note 2)	4	16		ns
29	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$* Setup Time	(Note 2)	4	14		ns
30	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$* Hold Time	(Note 2)	4	5		ns

Notes: 1. Float condition occurs when maximum output current becomes less than I_{LO} magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.

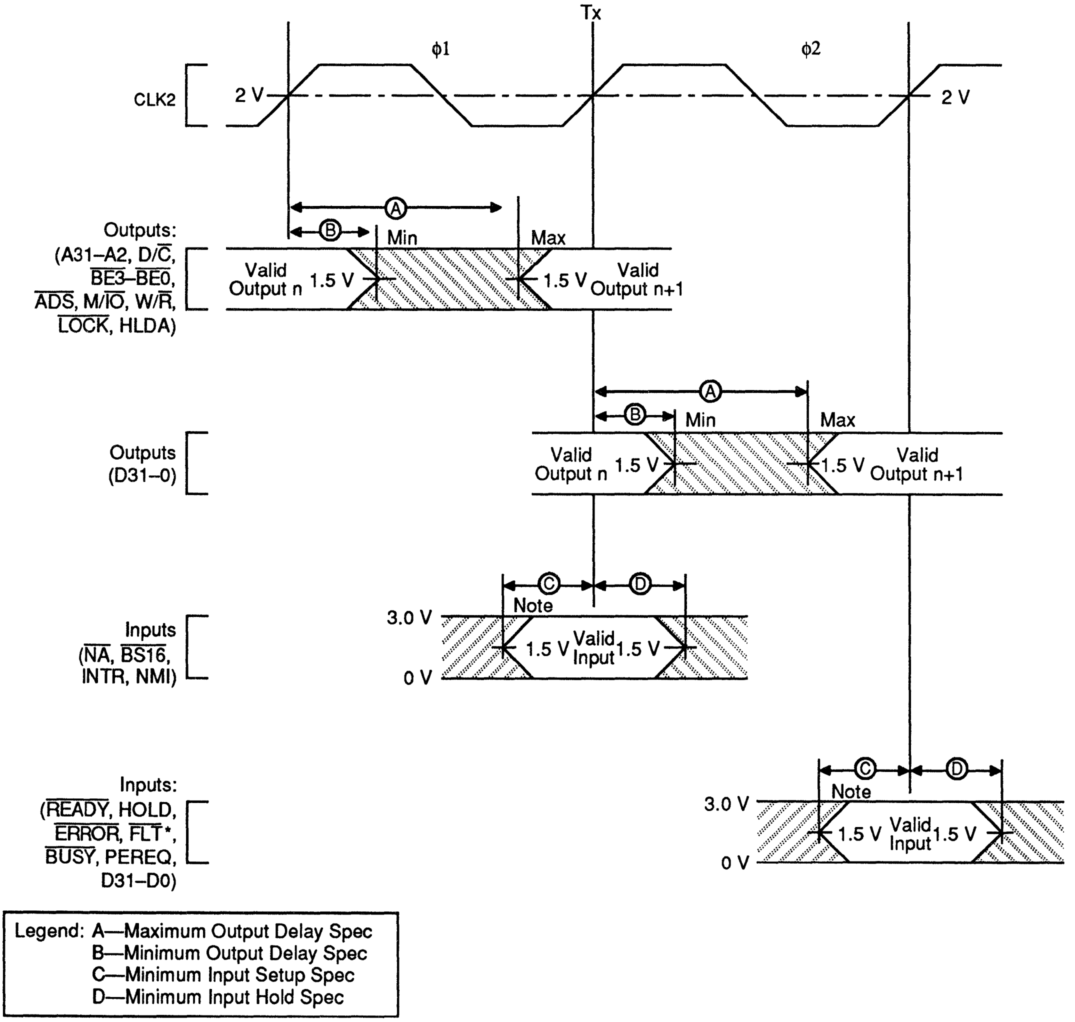
*Float feature available in Rev. C0 and later, PQFP only.

SWITCHING WAVEFORMS

The switching characteristics consist of output delays, input setup requirements, and input hold requirements. All characteristics are relative to the CLK2 rising edge crossing the 2.0 V level.

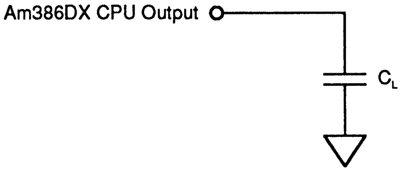
Switching characteristic measurement is defined in Figure 1. Inputs must be driven to the voltage levels indicated by this diagram. Am386DX microprocessor output delays are specified with minimum and maximum limits, measured as shown. The minimum Am386DX microprocessor delay times are hold times provided to external circuitry. Am386DX microprocessor input setup and hold time are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct Am386DX microprocessor operation.

Outputs $\overline{ADS}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{BE3}$–$\overline{BE0}$, A31–A2, and HLDA only change at the beginning of phase one. D31–D0 (write cycles) only change at the beginning of phase two. The $\overline{READY}$, HOLD, $\overline{BUSY}$, $\overline{ERROR}$, PEREQ, $\overline{FLT}$*, and D31–D0 (read cycles) inputs are sampled at the beginning of phase one. The $\overline{NA}$, $\overline{BS16}$, INTR, and NMI inputs are sampled at the beginning of phase two.


Tx
φ1
φ2
CLK2
2 V
2 V
Outputs: (A31–A2, D/C, BE3–BE0, ADS, M/IO, W/R, LOCK, HLDA)
A
B
Min
Max
Valid Output n
1.5 V
1.5 V
Valid Output n+1
Outputs (D31–0)
A
B
Min
Max
Valid Output n
1.5 V
1.5 V
Valid Output n+1
Inputs (NA, BS16, INTR, NMI)
C
D
Note
3.0 V
0 V
1.5 V
Valid Input
1.5 V
Inputs: (READY, HOLD, ERROR, FLT*, BUSY, PEREQ, D31–D0)
C
D
Note
3.0 V
0 V
1.5 V
Valid Input
1.5 V
Legend: A—Maximum Output Delay Spec
B—Minimum Output Delay Spec
C—Minimum Input Setup Spec
D—Minimum Input Hold Spec
15021B–071

Note: Input waveforms have tr ≤ 2.0 ns from 0.8 V to 2.0 V.
*Float feature available in Rev. Co and later, PQFP only.

Figure 1. Drive Levels and Measurement Points



C_L includes all parasitic capacitances.

Figure 2. AC Test Loads 15021B–072

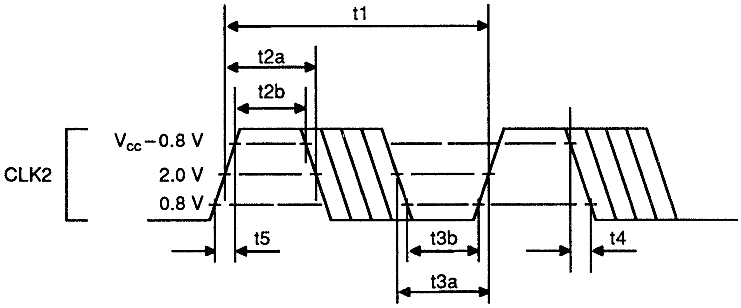
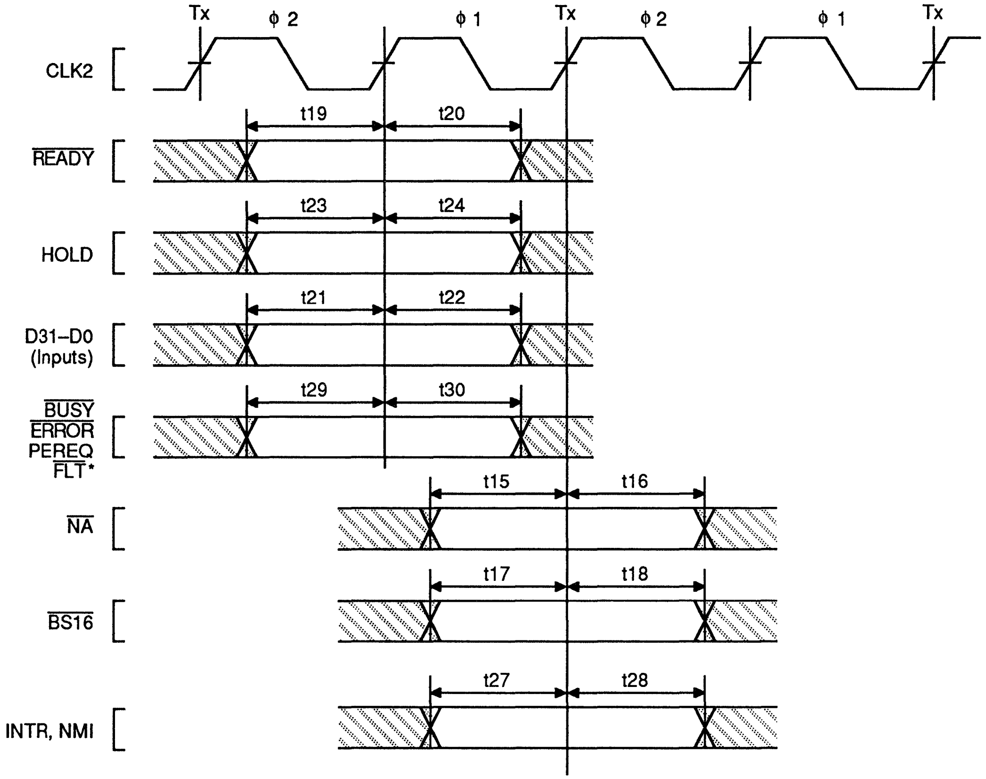


Figure 3. CLK2 Timing 15021B–073



*Float feature available in Rev. C0 and later, PQFP only.

Figure 4. Input Setup and Hold Timing

15021B–074

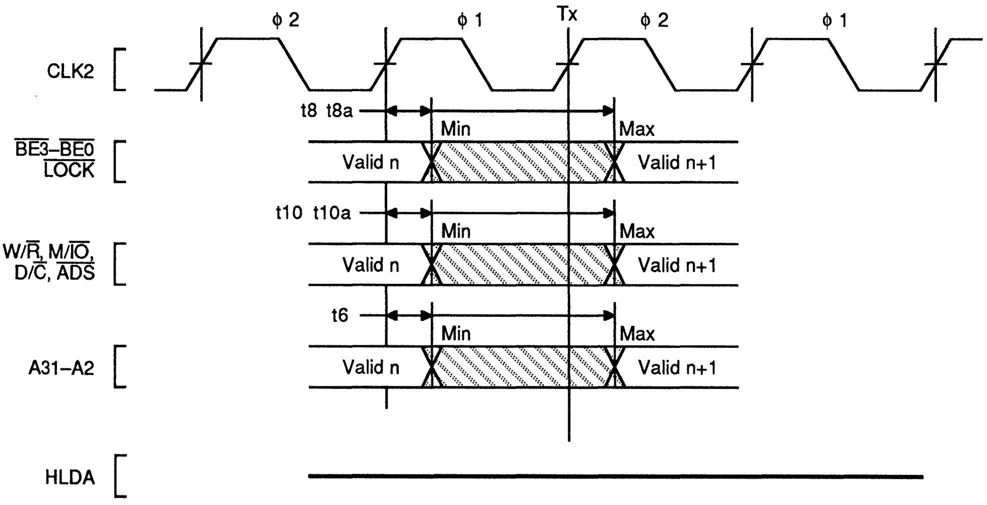


Figure 5. Output Valid Delay Timing

15021B–075

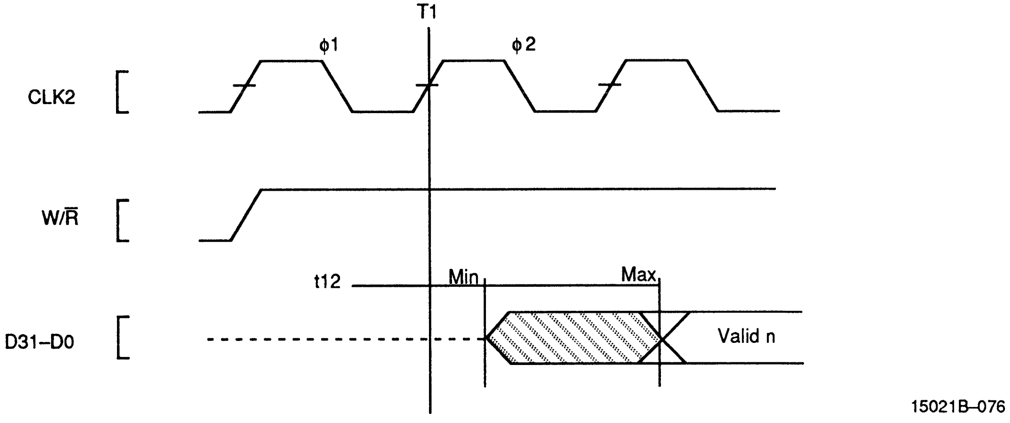


Figure 6. Write Data Valid Delay Timing (25, 33, and 40 MHz)

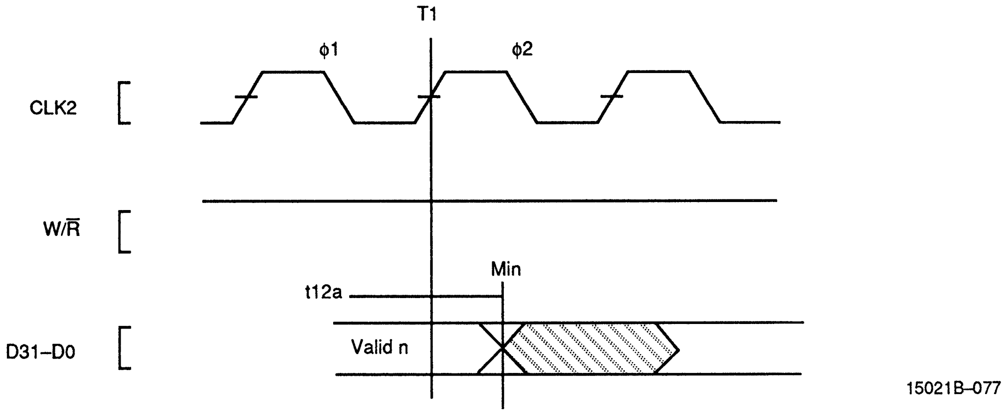


Figure 7. Write Data Hold Timing (25, 33, and 40 MHz)

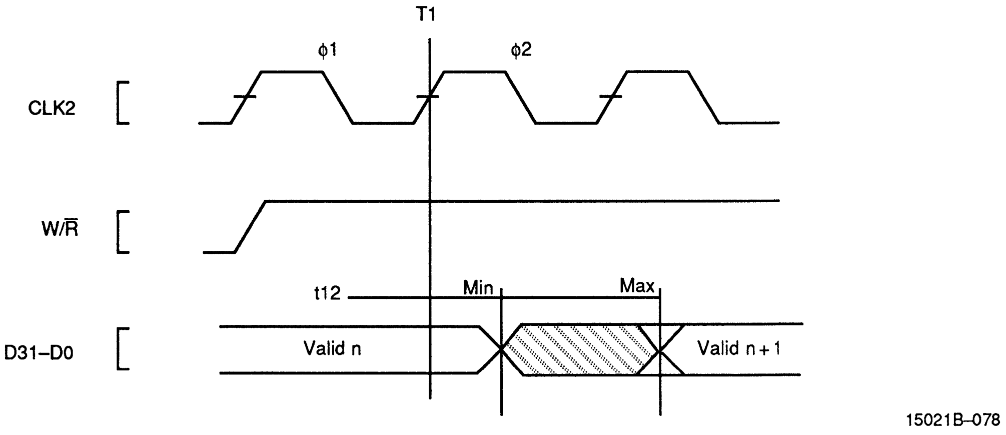
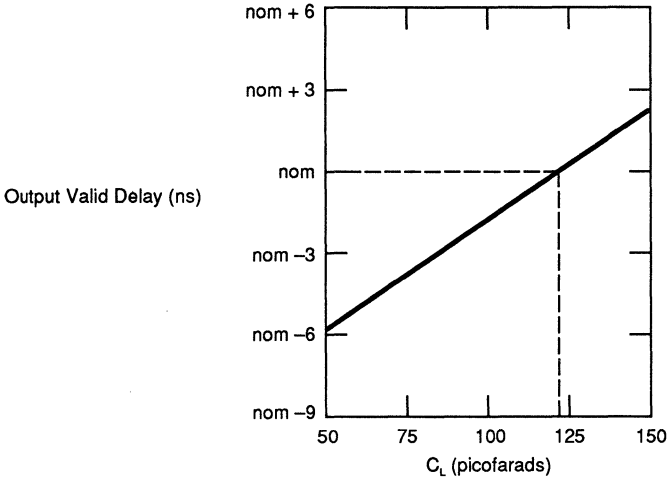


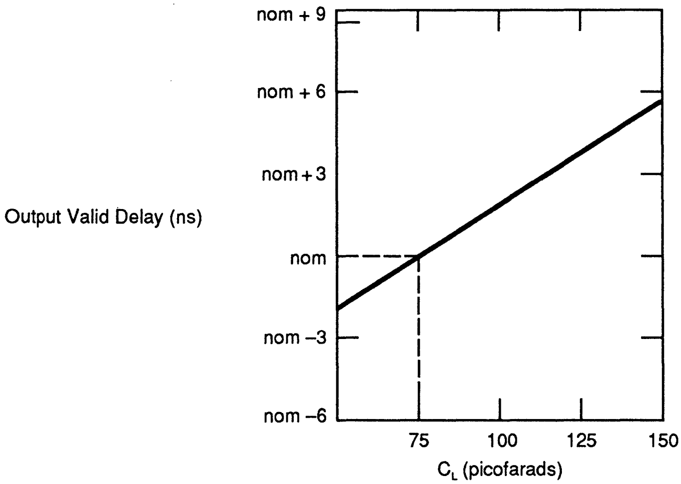
Figure 8. Write Data Valid Delay Timing (20 MHz)



Note: This graph will not be linear outside of the C_L range shown.

15021B–079

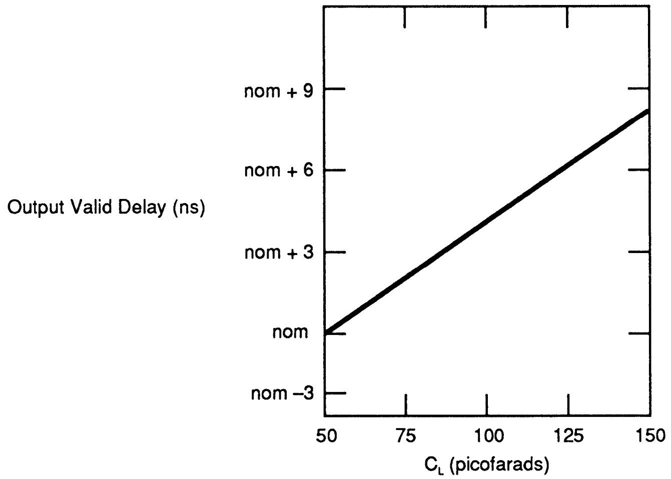
Figure 9. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L=120 pF)



15021B–080

Note: This graph will not be linear outside of the C_L range shown.

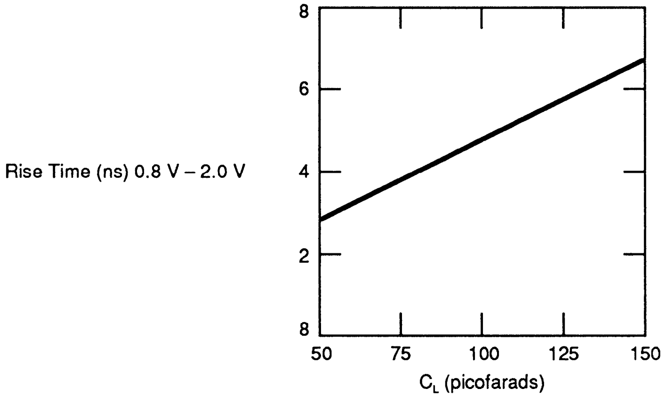
Figure 10. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L=75 pF)



Note: This graph will not be linear outside of the C_L range shown.

15021B–081

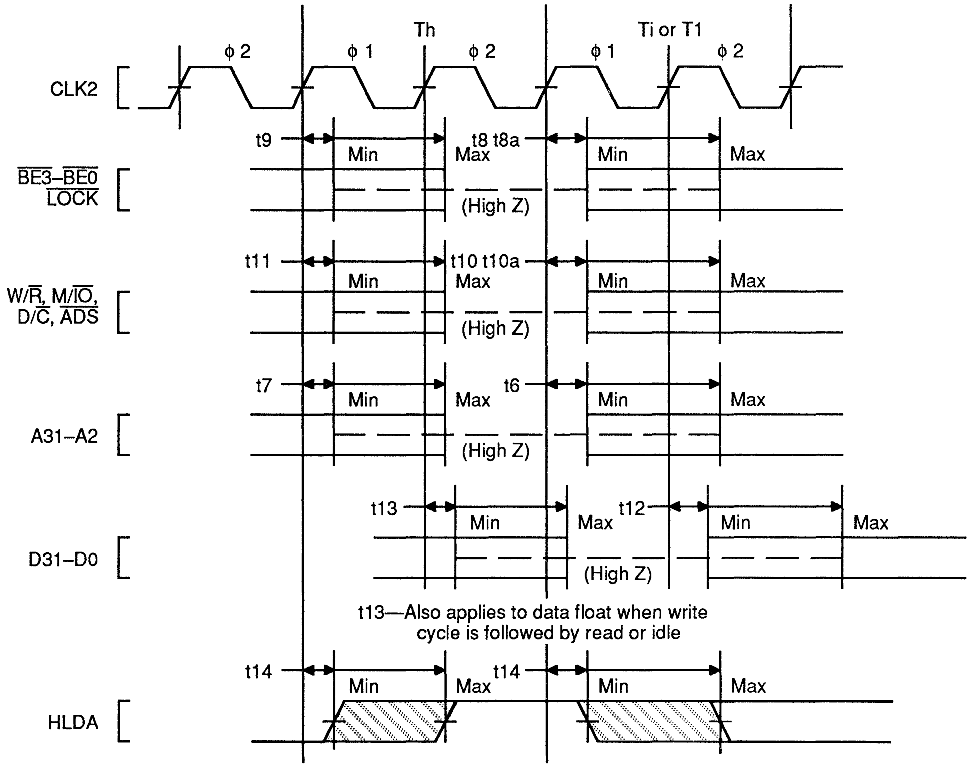
Figure 11. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature (C_L = 50 pF)



Note: This graph will not be linear outside of the C_L range shown.

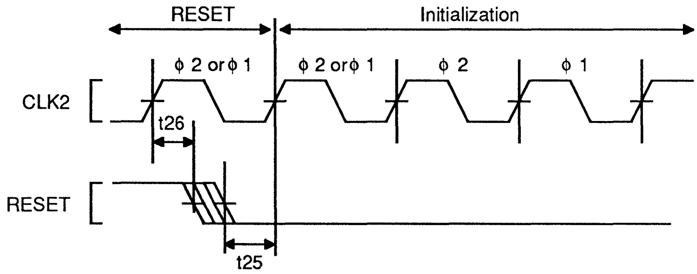
15021B–082

Figure 12. Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature



15021B–083

Figure 13. Output Float Delay and HLDA Valid Delay Timing



The second internal processor phase following RESET High-to-Low transition (provided t25 and t26 are met) is ϕ2.

15021B–084

Figure 14. RESET Setup and Hold Timing and Internal Phase

Am386™SX

High-Performance, 32-Bit Microprocessor with 16-Bit Data Bus

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- **Compatible with 386SX systems and software**
- **25- and 20-MHz operating speeds**
- **Pin-for-pin replacement of the Intel i386SX**
- **Supports 387SX-compatible math coprocessors**
- **100-lead PQFP package with optional protective ring for better lead coplanarity**
- **24-bit address bus, 16-bit data bus**
- **Advanced 0.8 micron CMOS technology**

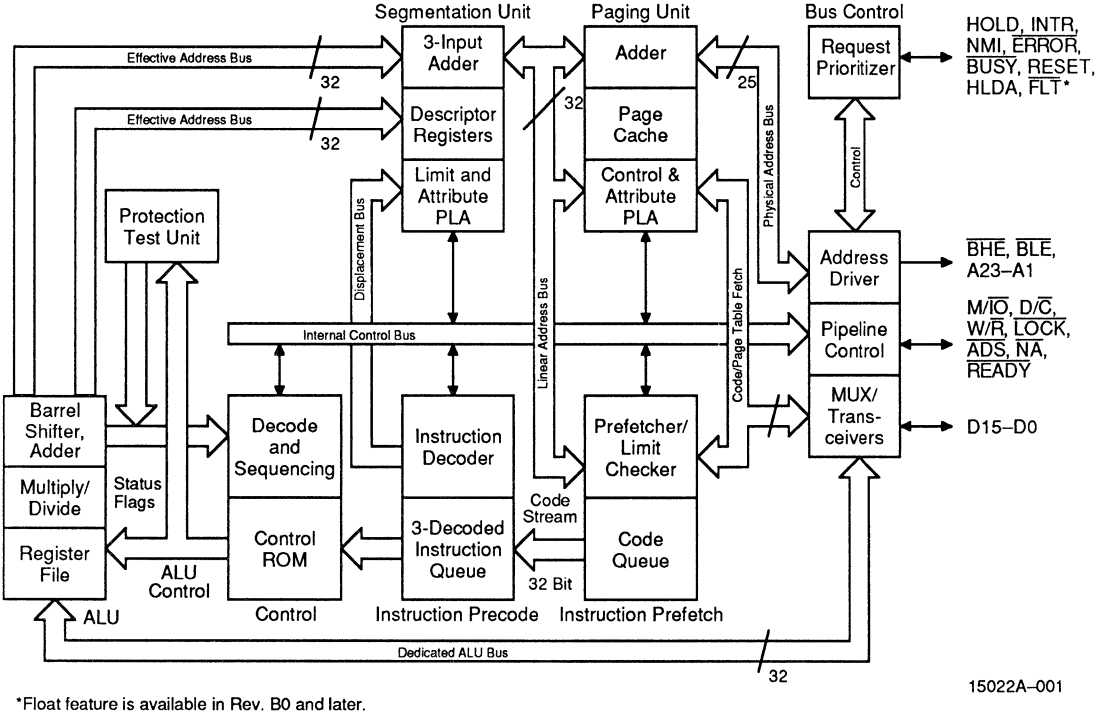
GENERAL DESCRIPTION

The Am386SX microprocessor is a compatible implementation of the Intel i386SX. It is engineered to meet strict requirements for compatibility. It is compatible with hardware designed for 386SX systems and is, in fact, a pin-for-pin replacement of the Intel i386SX. It is also compatible with operating systems written for the 386 and the wide variety of commercially available software applications.

The Am386SX microprocessor is a 32-bit CPU with a 16-bit external data bus, and a 24-bit external address bus. It provides the performance and compatibility benefits of the 386 architecture with the cost savings associated with 16-bit hardware. This device offers a 25% increase in performance from 20 to 25 MHz.

The device is manufactured using the AMD® advanced 0.8 micron CMOS process. It is packaged in a 100-pin plastic quad flat pack (PQFP). This package may be shipped in an optional protective ring for better lead protection during manufacturing.

BLOCK DIAGRAM

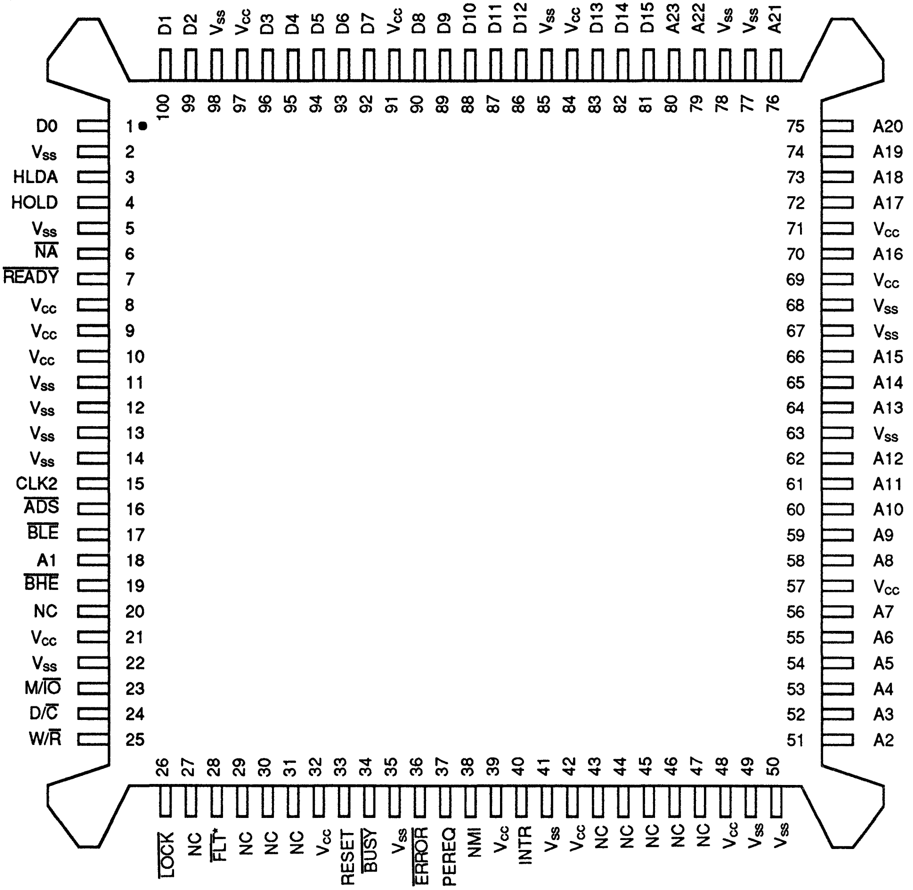


*Float feature is available in Rev. B0 and later.

Publication #: C15022 Rev. C Amendment: /0
Issue Date: December 1991

CONNECTION DIAGRAM

Top View



15022P 002

Notes: NC = No Connect
Pin 1 is marked for orientation.
*Float feature is available in Rev. B0 and later.

PIN DESIGNATIONS (Sorted by Pin Name)

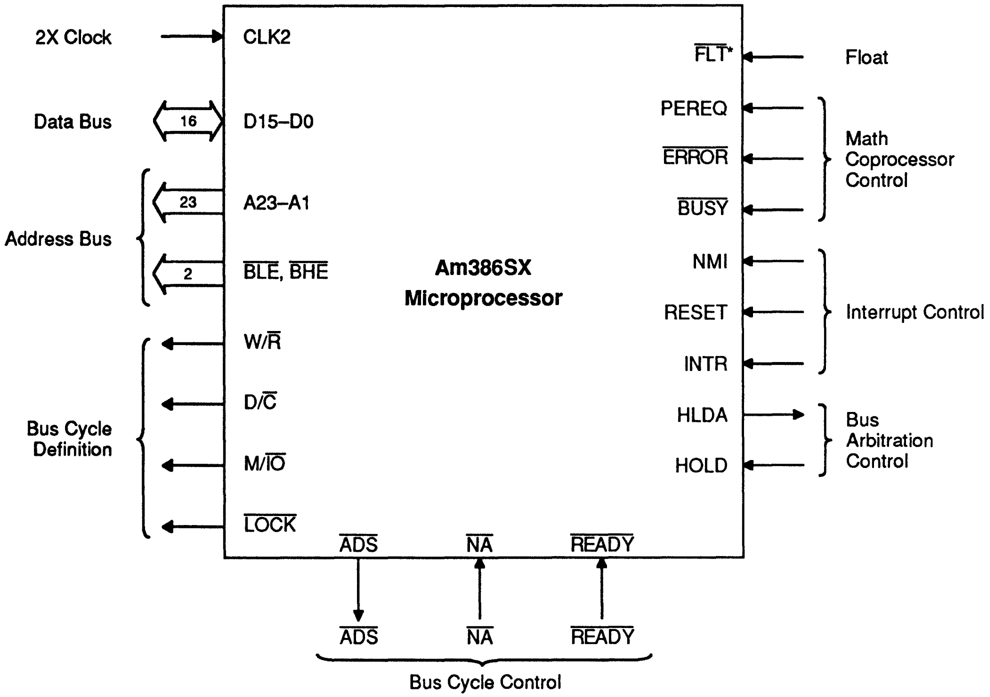
Address		Data		Control		NC	V_{cc}	V_{ss}
Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin No.	Pin No.	Pin No.
A1	18	D0	1	$\overline{\text{ADS}}$	16	20	8	2
A2	51	D1	100	$\overline{\text{BHE}}$	19	27	9	5
A3	52	D2	99	$\overline{\text{BLE}}$	17	29	10	11
A4	53	D3	96	$\overline{\text{BUSY}}$	34	30	21	12
A5	54	D4	95	CLK2	15	31	32	13
A6	55	D5	94	D/$\overline{\text{C}}$	24	43	39	14
A7	56	D6	93	$\overline{\text{ERROR}}$	36	44	42	22
A8	58	D7	92	$\overline{\text{FLT}}$*	28	45	48	35
A9	59	D8	90	HLDA	3	46	57	41
A10	60	D9	89	HOLD	4	47	69	49
A11	61	D10	88	INTR	40		71	50
A12	62	D11	87	$\overline{\text{LOCK}}$	26		84	63
A13	64	D12	86	M/$\overline{\text{IO}}$	23		91	67
A14	65	D13	83	$\overline{\text{NA}}$	6		97	68
A15	66	D14	82	NMI	38			77
A16	70	D15	81	PEREQ	37			78
A17	72			$\overline{\text{READY}}$	7			85
A18	73			RESET	33			98
A19	74			W/$\overline{\text{R}}$	25			
A20	75							
A21	76							
A22	79							
A23	80							

PIN DESIGNATIONS (Sorted by Pin Number)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	D0	21	V_{cc}	41	V_{ss}	61	A11	81	D15
2	V_{ss}	22	V_{ss}	42	V_{cc}	62	A12	82	D14
3	HLDA	23	M/$\overline{\text{IO}}$	43	NC	63	V_{ss}	83	D13
4	HOLD	24	D/$\overline{\text{C}}$	44	NC	64	A13	84	V_{cc}
5	V_{ss}	25	W/$\overline{\text{R}}$	45	NC	65	A14	85	V_{ss}
6	$\overline{\text{NA}}$	26	$\overline{\text{LOCK}}$	46	NC	66	A15	86	D12
7	$\overline{\text{READY}}$	27	NC	47	NC	67	V_{ss}	87	D11
8	V_{cc}	28	$\overline{\text{FLT}}$*	48	V_{cc}	68	V_{ss}	88	D10
9	V_{cc}	29	NC	49	V_{ss}	69	V_{cc}	89	D9
10	V_{cc}	30	NC	50	V_{ss}	70	A16	90	D8
11	V_{ss}	31	NC	51	A2	71	V_{cc}	91	V_{cc}
12	V_{ss}	32	V_{cc}	52	A3	72	A17	92	D7
13	V_{ss}	33	RESET	53	A4	73	A18	93	D6
14	V_{ss}	34	$\overline{\text{BUSY}}$	54	A5	74	A19	94	D5
15	CLK2	35	V_{ss}	55	A6	75	A20	95	D4
16	$\overline{\text{ADS}}$	36	$\overline{\text{ERROR}}$	56	A7	76	A21	96	D3
17	$\overline{\text{BLE}}$	37	PEREQ	57	V_{cc}	77	V_{ss}	97	V_{cc}
18	A1	38	NMI	58	A8	78	V_{ss}	98	V_{ss}
19	$\overline{\text{BHE}}$	39	V_{cc}	59	A9	79	A22	99	D2
20	NC	40	INTR	60	A10	80	A23	100	D1

*Float feature is available in Rev. B0 and later.

LOGIC SYMBOL



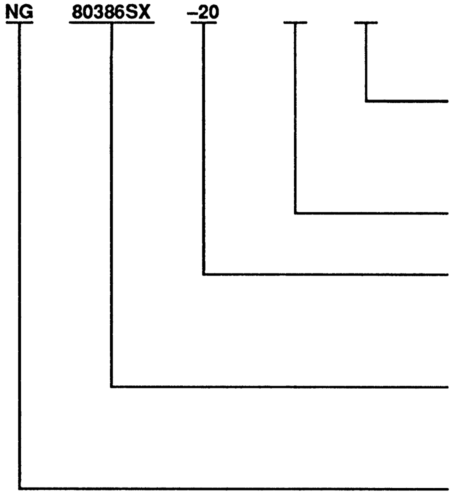
15022B–003

*Float feature is available in Rev. B0 and later.

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



OPTIONAL PROCESSING
None = Trimmed and Formed PQFP in high-temp trays
F = Ringed PQFP in horizontal tubes
S = Ringed PQFP in coin-stack tubes

TEMPERATURE RANGE
Blank = Commercial (0°C to +100°C)

SPEED OPTION
–25 = 25 MHz
–20 = 20 MHz
–16 = 16 MHz*

DEVICE NUMBER/DESCRIPTION
80386SX
Am386SX High-Performance,
32-Bit Microprocessor with 16-Bit Data Bus

PACKAGE TYPE
NG = 100-Pin Plastic Quad Flat Pack (PQ100, PQB100)

Valid Combinations		
NG	80386SX	–25 –20 –16*
		–25F –20F –16F*
		–25S –20S –16S*

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations.

*Contact AMD for 16-MHz availability.

PIN DESCRIPTION

A23–A1

Address Bus (Outputs)

Outputs physical memory or port I/O addresses.

$\overline{\text{ADS}}$

Address Status (Active Low; Output)

Indicates that a valid bus cycle definition and address ($\text{W}/\overline{\text{R}}$, $\text{D}/\overline{\text{C}}$, $\text{M}/\overline{\text{IO}}$, $\overline{\text{BHE}}$, $\overline{\text{BLE}}$, and A23–A1) are being driven at the Am386SX microprocessor pins.

$\overline{\text{BHE}}$, $\overline{\text{BLE}}$

Byte Enables (Active Low; Outputs)

Indicate which data bytes of the data bus take part in a bus cycle.

$\overline{\text{BUSY}}$

Busy (Active Low; Input)

Signals a busy condition from a processor extension.

CLK2

CLK2 (Input)

Provides the fundamental timing for the Am386SX microprocessor.

D15–D0

Data Bus (Inputs/Outputs)

Inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles.

$\text{D}/\overline{\text{C}}$

Data/Control (Output)

A bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and code fetch.

$\overline{\text{ERROR}}$

Error (Active Low; Input)

Signals an error condition from a processor extension.

$\overline{\text{FLT}}$*

Float (Active Low; Input)

An input which forces all bi-directional and output signals, including HLDA, to the three-state condition.

HLDA

Bus Hold Acknowledge (Active High; Output)

Output indicates that the Am386SX microprocessor has surrendered control of its logical bus to another bus master.

HOLD

Bus Hold Request (Active High; Input)

Input allows another bus master to request control of the local bus.

INTR

Interrupt Request (Active High; Input)

A maskable input that signals the Am386SX microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

$\overline{\text{LOCK}}$

Bus Lock (Active Low; Output)

A bus cycle definition pin that indicates that other system bus masters are not to gain control of the system bus while it is active.

$\text{M}/\overline{\text{IO}}$

Memory/IO (Output)

A bus cycle definition pin that distinguishes memory cycles from input/output cycles.

$\overline{\text{NA}}$

Next Address (Active Low; Input)

Used to request address pipelining.

NC

No Connect

Should always be left unconnected. Connection of a NC pin may cause the processor to malfunction, or be incompatible with future steppings of the Am386SX microprocessor.

NMI

Non-Maskable Interrupt Request (Actve High; Input)

A non-maskable input that signals the Am386SX microprocessor to suspend execution of the current program and execute an interrupt acknowledge function.

PEREQ

Processor Extension Request (Active High; Input)

Indicates that the processor has data to be transferred by the Am386SX microprocessor.

$\overline{\text{READY}}$

Bus Ready (Active Low; Input)

Terminates the bus cycle.

RESET

Reset (Active High; Input)

Suspends any operation in progress and places the Am386SX microprocessor in a known reset state.

V_{cc}

System Power (Active High; Input)

Provides the +5 V nominal DC supply input.

V_{ss}

System Ground (Input)

Provides the 0 V connection from which all inputs and outputs are measured.

$\text{W}/\overline{\text{R}}$

Write/Read (Output)

A bus cycle definition pin that distinguishes write cycles from read cycles.

*Float feature is available in Rev. B0 and later.

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature under bias . . . . . −65 to 125°C
Storage Temperature . . . . −65 to 150°C

Stresses above those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods of time may affect device reliability.

OPERATING RANGES

Supply Voltage with respect to V_{SS} . . . . −0.5 V to 7 V
Voltage on other pins . . . . −0.5 V to (V_{CC} + 0.5)V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating ranges

V_{CC} = 5 V ±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Min	Max	Unit
V_{IL}	Input Low Voltage		–0.3	+0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}+ 0.3	V
V_{ILC}	CLK2 Input Low Voltage		–0.3	+0.8	V
V_{IHC}	CLK2 Input High Voltage		V_{CC} – 0.8	V_{CC}+ 0.3	V
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A23–A1, D15–D0 I_{OL} = 5 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA			 0.45 0.45	 V V
V_{OH}	Output High Voltage I_{OH} = 1.0 mA: A23–A1, D15–D0 I_{OH} = 0.2 mA: A23–A1, D15–D0 I_{OH} = 0.9 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA I_{OH} = 0.18 mA: $\overline{BHE}$, $\overline{BLE}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{ADS}$, HLDA		 2.4 V_{CC} – 0.5 2.4 V_{CC} – 0.5		 V V V
I_{LI}	Input Leakage Current (for all pins except PEREQ, $\overline{BUSY}$, $\overline{FLT}$*, and $\overline{ERROR}$)	0 V ≤ V_{IN} ≤ V_{CC}		±15	µA
I_{IH}	Input Leakage Current (PEREQ pin)	V_{IH} = 2.4 V (1)		200	µA
I_{IL}	Input Leakage Current ($\overline{BUSY}$, $\overline{ERROR}$, and $\overline{FLT}$* pins)	V_{IL} = 0.45 V (2)		–400	µA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		±15	µA
I_{CC}	Supply Current CLK2 = 32 MHz: with –16** CLK2 = 40 MHz: with –20 CLK2 = 50 MHz: with –25	 I_{CC} Typ = 175 mA (3) I_{CC} Typ = 200 mA (3) I_{CC} Typ = 225 mA (3)		 275 305 335	 mA mA mA
C_{IN}	Input Capacitance	F_C = 1 MHz (4)		10	pF
C_{OUT}	Output or I/O Capacitance	F_C = 1 MHz (4)		12	pF
C_{CLK}	CLK2 Capacitance	F_C = 1 MHz (4)		20	pF

Notes: Tested at the minimum operating frequency of the part.
*Float feature is available in Rev. B0 and later.
**Contact AMD for 16-MHz availability.

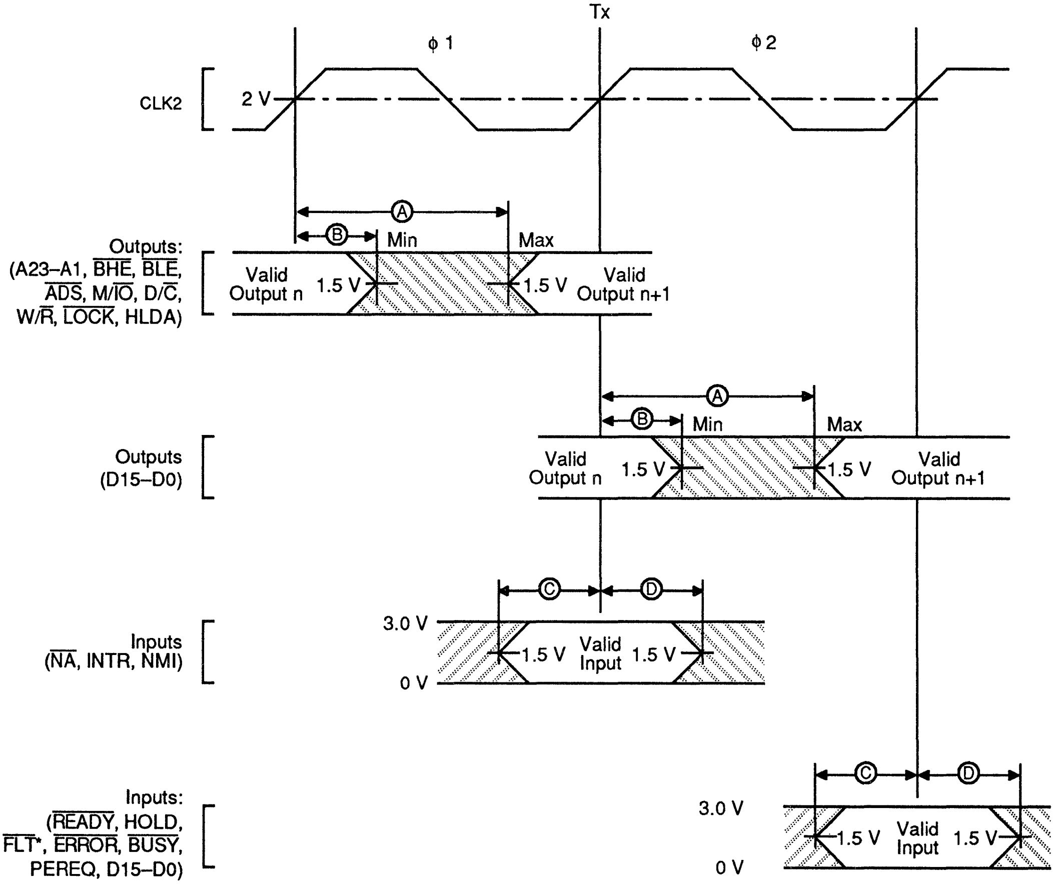
1. PEREQ input has an internal pull-down resistor.
2. $\overline{BUSY}$, $\overline{FLT}$*, and $\overline{ERROR}$ inputs each have an internal pull-up resistor.
3. I_{CC} Max measurement at worst case frequency, V_{CC}, and temperature, outputs unloaded.
4. Not 100% tested.

SWITCHING CHARACTERISTICS

The switching characteristics given consist of output delays, input setup requirements, and input hold requirements. All switching characteristics are relative to the CLK2 rising edge crossing the 2.0 V level.

Switching characteristic measurement is defined by Figure 1. Inputs must be driven to the voltage levels indicated by Figure 1 when switching characteristics are measured. Output delays are specified with minimum and maximum limits measured, as shown. The minimum delay times are hold times provided to external circuitry. Input setup and hold times are specified as minimums, defining the smallest acceptable sampling window. Within the sampling window, a synchronous input signal must be stable for correct operation.

Outputs $\overline{ADS}$, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, $\overline{BHE}$, $\overline{BLE}$, A23–A1, and HLDA only change at the beginning of phase one. D15–D0 (write cycles) only change at the beginning of phase two. The $\overline{READY}$, HOLD, $\overline{BUSY}$, $\overline{ERROR}$, PEREQ, $\overline{FLT}$*, and D15–D0 (read cycles) inputs are sampled at the beginning of phase one. The $\overline{NA}$, INTR, and NMI inputs are sampled at the beginning of phase two.



Legend: A—Maximum Output Delay Characteristic
B—Minimum Output Delay Characteristic
C—Minimum Input Setup Characteristic
D—Minimum Input Hold Characteristic

15022B–030

*Float feature is available in Rev. B0 and later.

Figure 1. Drive Levels and Measurement Points for Switching Characteristics

SWITCHING CHARACTERISTICS over operating ranges

Switching Characteristics at 25 MHz: V_{CC} = 5 V±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Ref. Figure	Min	Max	Unit
	Operating Frequency	Half CLK2 freq.		2	25	MHz
1	CLK2 Period		2	20	250	ns
2a	CLK2 High Time	at 2 V	2	7		ns
2b	CLK2 High Time	at (V_{CC} − 0.8 V)	2	4		ns
3a	CLK2 Low Time	at 2 V	2	7		ns
3b	CLK2 Low Time	at 0.8 V	2	5		ns
4	CLK2 Fall Time	(V_{CC} − 0.8 V) to 0.8 V (Note 3)	2		7	ns
5	CLK2 Rise Time	0.8 V to (V_{CC} − 0.8 V) (Note 3)	2		7	ns
6	A23–A1 Valid Delay	C_L = 50 pF	5	4	17	ns
7	A23–A1 Float Delay	(Note 1)	9	4	30	ns
8	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 50 pF	5	4	17	ns
9	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	9	4	30	ns
10	M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, W/$\overline{\text{R}}$, $\overline{\text{ADS}}$ Valid Delay	C_L = 50 pF	5	4	17	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	9	4	30	ns
12	D15–D0 Write Data Valid Delay	C_L = 50 pF	5	7	23	ns
12a	D15–D0 Write Data Hold Time	C_L = 50 pF		2		ns
13	D15–D0 Write Data Float Delay	(Note 1)	9	4	22	ns
14	HLDA Valid Delay	C_L = 50 pF	5	4	22	ns
15	$\overline{\text{NA}}$ Setup Time		4	5		ns
16	$\overline{\text{NA}}$ Hold Time		4	3		ns
19	$\overline{\text{READY}}$ Setup Time		4	9		ns
20	$\overline{\text{READY}}$ Hold Time		4	4		ns
21	D15–D0 Read Data Setup Time		4	7		ns
22	D15–D0 Read Data Hold Time		4	5		ns
23	HOLD Setup Time		4	9		ns
24	HOLD Hold Time		4	3		ns
25	RESET Setup Time		10	8		ns
26	RESET Hold Time		10	3		ns
27	NMI, INTR Setup Time	(Note 2)	4	6		ns
28	NMI, INTR Hold Time	(Note 2)	4	6		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$* Setup Time	(Note 2)	4	6		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$* Hold Time	(Note 2)	4	5		ns

Notes: *Float feature is available in Rev. B0 and later.

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.

SWITCHING CHARACTERISTICS over operating ranges (continued)

Switching Characteristics at 20 MHz: V_{CC} = 5 V±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Ref. Figure	Min	Max	Unit
	Operating Frequency	Half CLK2 freq.		2	20	MHz
1	CLK2 Period		2	25	250	ns
2a	CLK2 High Time	at 2 V	2	8		ns
2b	CLK2 High Time	at (V_{CC}−0.8 V)	2	5		ns
3a	CLK2 Low Time	at 2 V	2	8		ns
3b	CLK2 Low Time	at 0.8 V	2	6		ns
4	CLK2 Fall Time	(V_{CC}−0.8 V) to 0.8 V (Note 3)	2		8	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}−0.8 V) (Note 3)	2		8	ns
6	A23–A1 Valid Delay	C_L = 120 pF (Note 4)	9	4	30	ns
7	A23–A1 Float Delay	(Note 1)	9	4	32	ns
8	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Valid Delay	C_L = 75 pF (Note 4)	5	4	30	ns
9	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$ Float Delay	(Note 1)	9	4	32	ns
10a	M/$\overline{IO}$, D/$\overline{C}$ Valid Delay	C_L = 75 pF (Note 4)	5	4	28	ns
10b	W/$\overline{R}$, $\overline{ADS}$ Valid Delay	C_L = 75 pF (Note 4)	5	4	26	ns
11	W/$\overline{R}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{ADS}$ Float Delay	(Note 1)	9	6	30	ns
12	D15–D0 Write Data Valid Delay	C_L = 120 pF (Note 4)	5	4	38	ns
13	D15–D0 Write Data Float Delay	(Note 1)	9	4	27	ns
14	HLDA Valid Delay	C_L = 75 pF (Note 4)	5	4	28	ns
15	$\overline{NA}$ Setup Time		4	5		ns
16	$\overline{NA}$ Hold Time		4	12		ns
19	$\overline{READY}$ Setup Time		4	12		ns
20	$\overline{READY}$ Hold Time		4	4		ns
21	D15–D0 Read Data Setup Time		4	9		ns
22	D15–D0 Read Data Hold Time		4	6		ns
23	HOLD Setup Time		4	17		ns
24	HOLD Hold Time		4	5		ns
25	RESET Setup Time		10	12		ns
26	RESET Hold Time		10	4		ns
27	NMI, INTR Setup Time	(Note 2)	4	16		ns
28	NMI, INTR Hold Time	(Note 2)	4	16		ns
29	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$* Setup Time	(Note 2)	4	14		ns
30	PEREQ, $\overline{ERROR}$, $\overline{BUSY}$, $\overline{FLT}$* Hold Time	(Note 2)	4	5		ns

Notes: *Float feature is available in Rev. B0 and later.

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with C_L set at 50 pF and derated to support the indicated distributed capacitive load. See Figures 11–14 for the capacitive derating curve.

SWITCHING CHARACTERISTICS over operating ranges (continued)

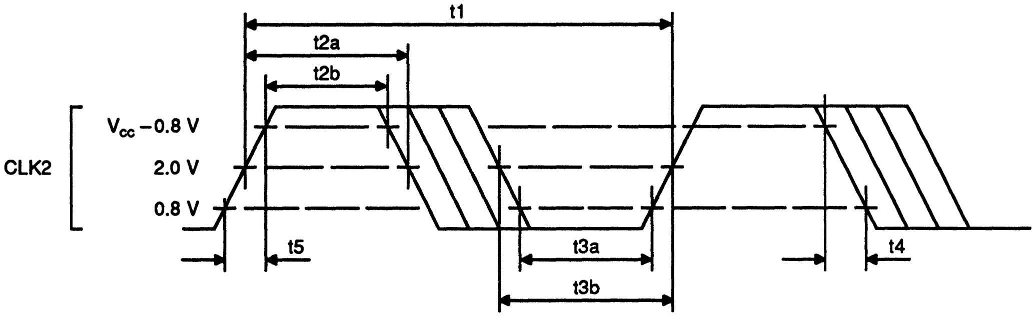
Switching Characteristics at 16 MHz**: V_{CC} = 5 V±10%; T_{CASE} = 0°C to 100°C

Symbol	Parameter Description	Notes	Ref. Figure	Min	Max	Unit
	Operating Frequency	Half CLK2 freq.		2	16	MHz
1	CLK2 Period		2	31	250	ns
2a	CLK2 High Time	at 2 V	2	9		ns
2b	CLK2 High Time	at (V_{CC}−0.8 V)	2	5		ns
3a	CLK2 Low Time	at 2 V	2	9		ns
3b	CLK2 Low Time	at 0.8 V	2	7		ns
4	CLK2 Fall Time	(V_{CC}−0.8 V) to 0.8 V (Note 3)	2		8	ns
5	CLK2 Rise Time	0.8 V to (V_{CC}−0.8 V) (Note 3)	2		8	ns
6	A23–A1 Valid Delay	C_L = 120 pF (Note 4)	5	4	36	ns
7	A23–A1 Float Delay	(Note 1)	9	4	40	ns
8	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Valid Delay	C_L = 75 pF (Note 4)	5	4	36	ns
9	$\overline{\text{BHE}}$, $\overline{\text{BLE}}$, $\overline{\text{LOCK}}$ Float Delay	(Note 1)	9	4	40	ns
10	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Valid Delay	C_L = 75 pF (Note 4)	5	4	33	ns
11	W/$\overline{\text{R}}$, M/$\overline{\text{IO}}$, D/$\overline{\text{C}}$, $\overline{\text{ADS}}$ Float Delay	(Note 1)	9	6	35	ns
12	D15–D0 Write Data Valid Delay	C_L = 120 pF (Note 4)	5	4	40	ns
13	D15–D0 Write Data Float Delay	(Note 1)	9	4	35	ns
14	HLDA Valid Delay	C_L = 75 pF (Note 4)	5	4	33	ns
15	$\overline{\text{NA}}$ Setup Time		4	5		ns
16	$\overline{\text{NA}}$ Hold Time		4	21		ns
19	$\overline{\text{READY}}$ Setup Time		4	19		ns
20	$\overline{\text{READY}}$ Hold Time		4	4		ns
21	D15–D0 Read Data Setup Time		4	9		ns
22	D15–D0 Read Data Hold Time		4	6		ns
23	HOLD Setup Time		4	26		ns
24	HOLD Hold Time		4	5		ns
25	RESET Setup Time		10	13		ns
26	RESET Hold Time		10	4		ns
27	NMI, INTR Setup Time	(Note 2)	4	16		ns
28	NMI, INTR Hold Time	(Note 2)	4	16		ns
29	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$* Setup Time	(Note 2)	4	16		ns
30	PEREQ, $\overline{\text{ERROR}}$, $\overline{\text{BUSY}}$, $\overline{\text{FLT}}$* Hold Time	(Note 2)	4	5		ns

Notes: *Float feature is available in Rev. B0 and later.

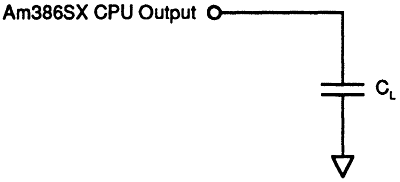
**Contact AMD for 16-MHz availability.

1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. These are not tested. They are guaranteed by design characterization.
4. Tested with C_L set at 50 pF and derated to support the indicated distributed capacitive load. See Figures 11–14 for the capacitive derating curve.



15022B–031

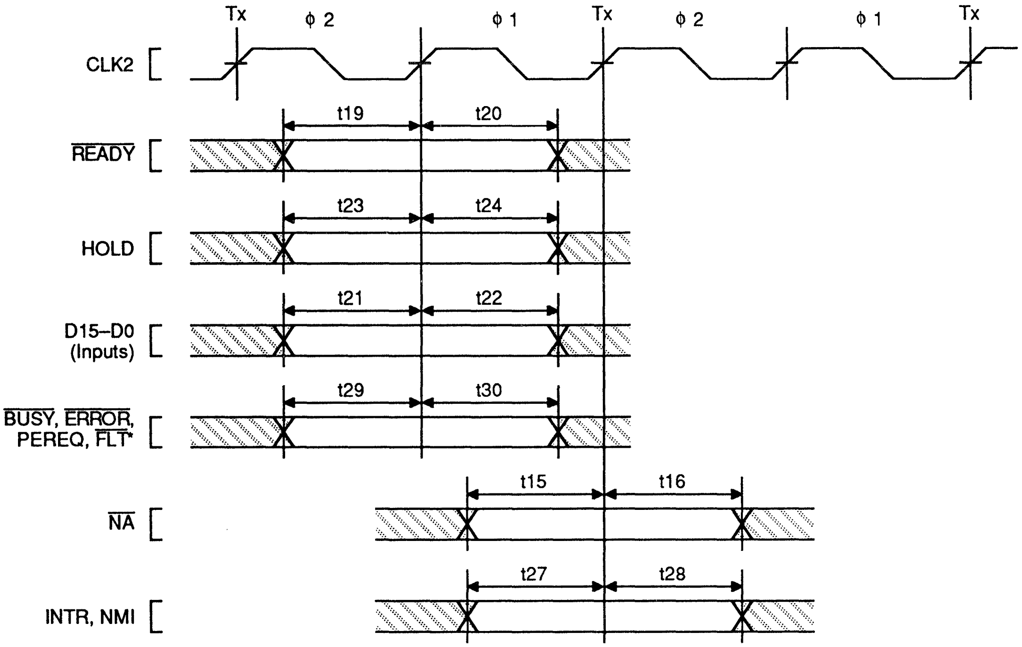
Figure 2. CLK2 Timing



15022B–032

Figure 3. AC Test Circuit

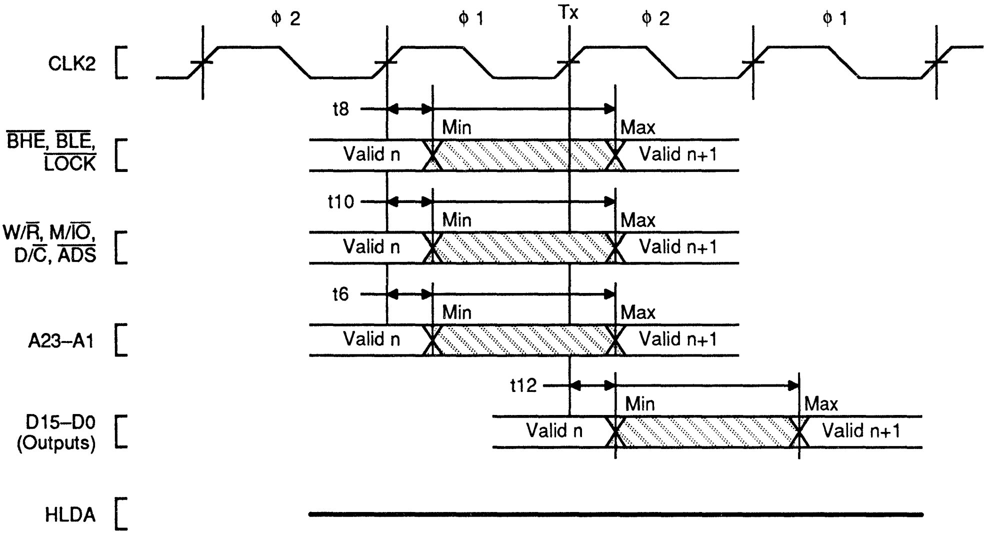
SWITCHING WAVEFORMS



*Float feature is available in Rev. B0 and later.

15022B–033

Figure 4. Input Setup and Hold Timing



15022B–034

Figure 5. Output Valid Delay Timing

SWITCHING WAVEFORMS (continued)

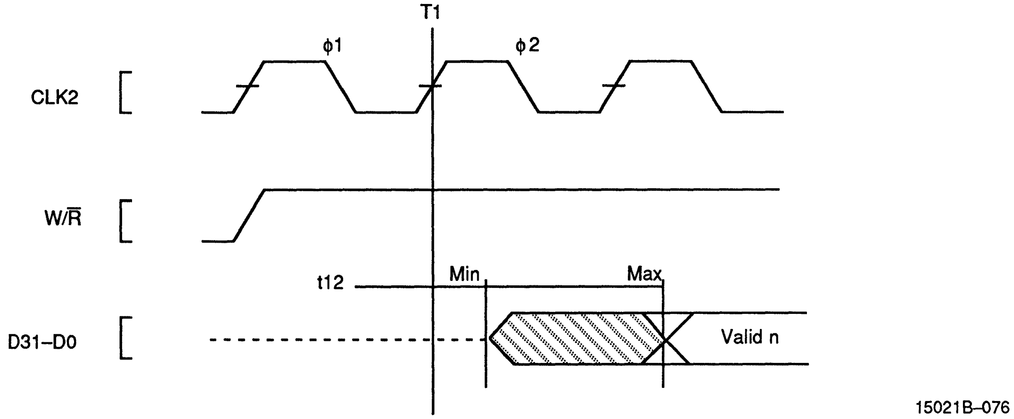


Figure 6. Write Data Valid Delay Timing (20 and 25 MHz)

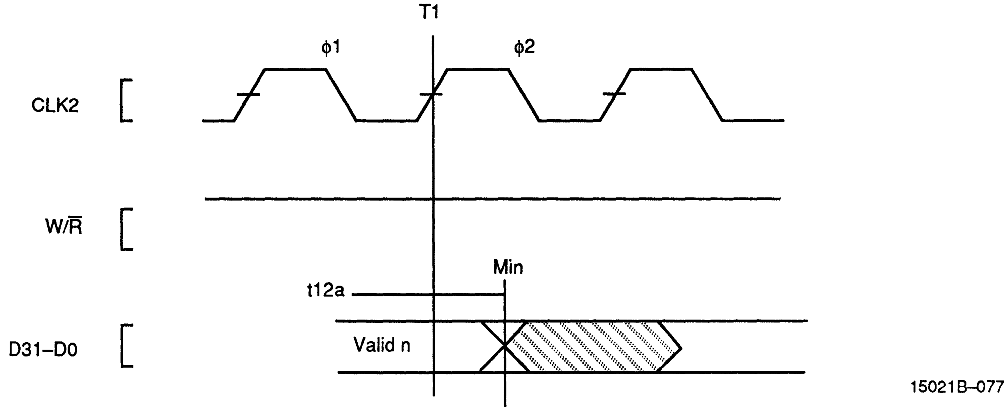


Figure 7. Write Data Hold Timing (20 and 25 MHz)

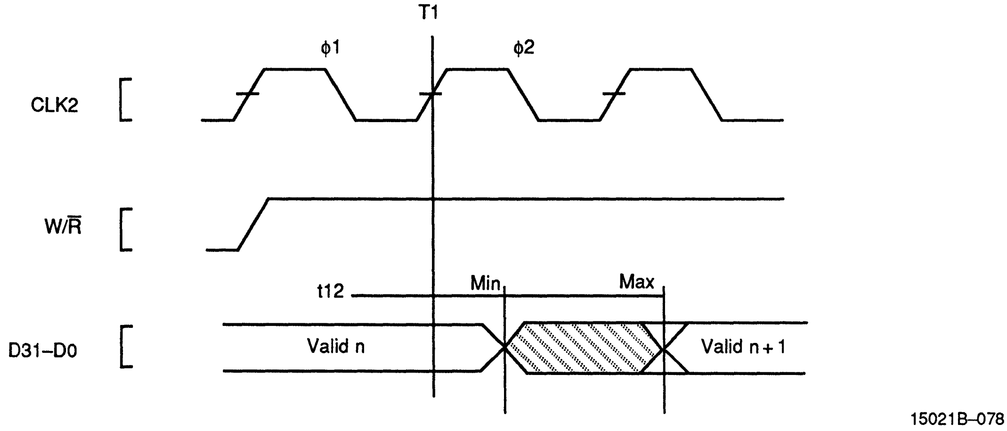
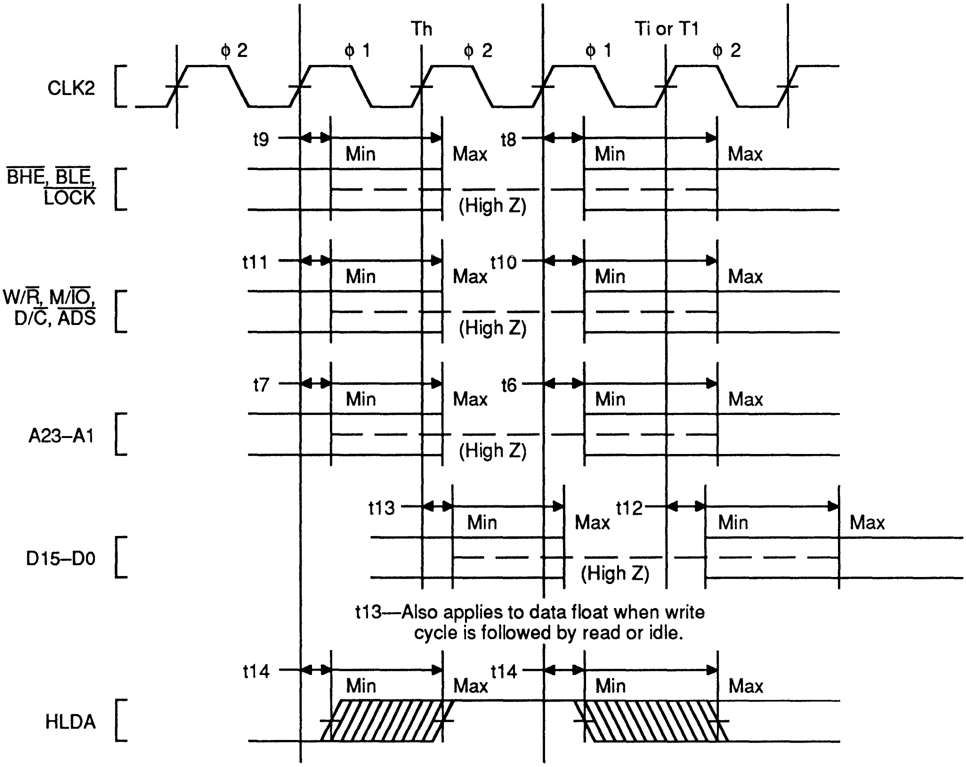


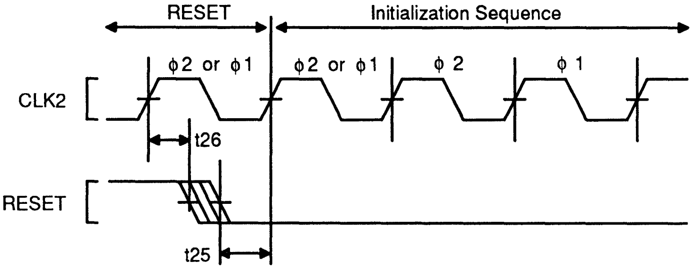
Figure 8. Write Data Valid Delay Timing (20 MHz)

SWITCHING WAVEFORMS (continued)



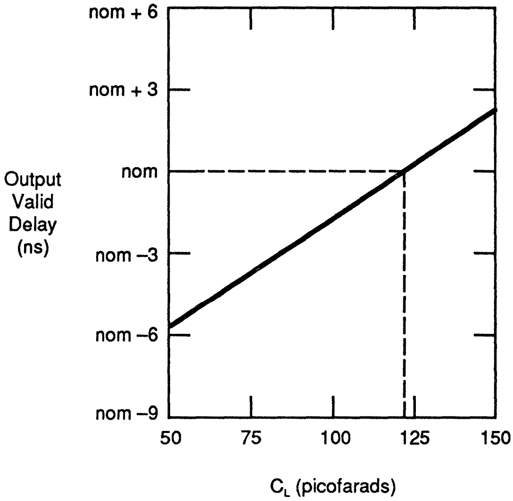
15022B–035

Figure 9. Output Float Delay and HLDA Valid Delay Timing



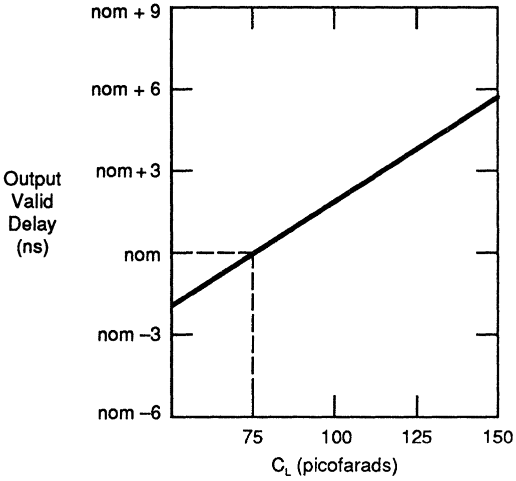
15022B–036

Figure 10. RESET Setup and Hold Timing and Internal Phase



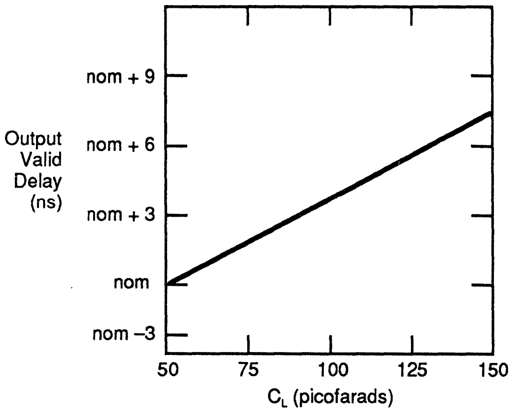
15022B–037

Figure 11. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature ($C_L = 120$ pF)



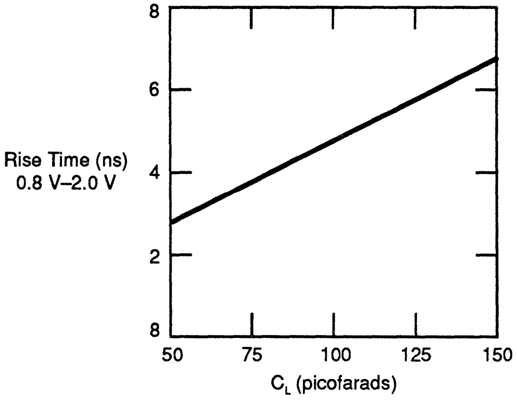
15022B–038

Figure 12. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature ($C_L = 75$ pF)



Note: This graph will not be linear outside of the C_L range shown.

Figure 13. Typical Output Valid Delay Versus Load Capacitance at Maximum Operating Temperature ($C_L = 50$ pF)



15022B–039

Figure 14. Typical Output Rise Time Versus Load Capacitance at Maximum Operating Temperature

AMD 80C287™

80-Bit CMOS Math Coprocessor

Advanced
Micro
Devices

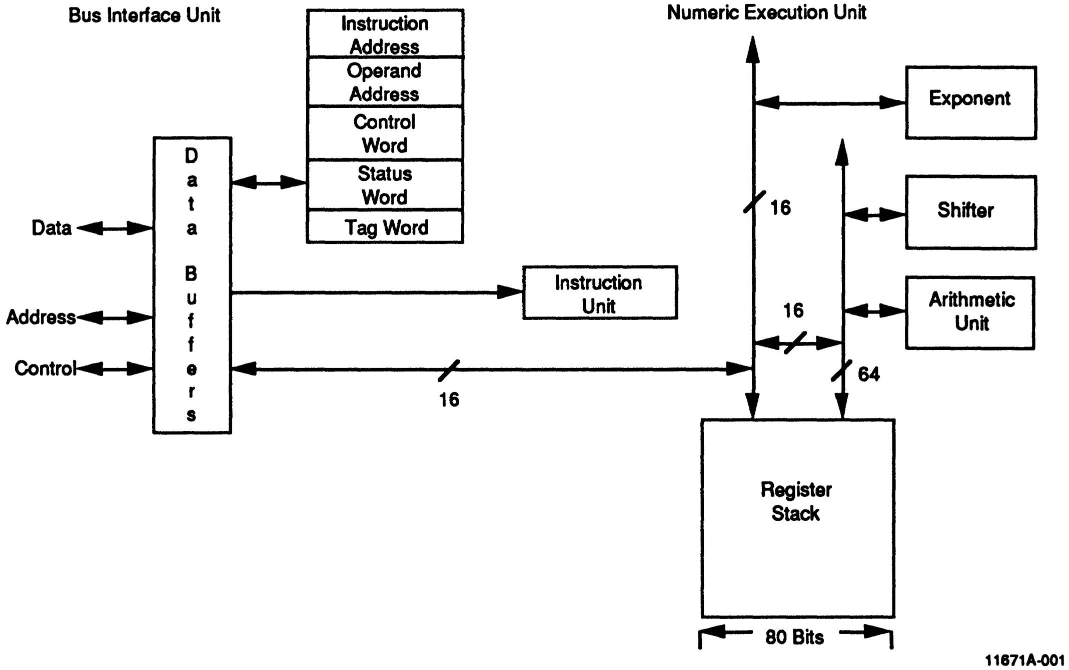
DISTINCTIVE CHARACTERISTICS

- **High-performance CMOS process yields 10-MHz and 12-MHz speed grades**
- **Available in space-saving 44-pin PLCC as well as 40-pin DIP**
- **80-bit numeric accelerator for 80C286- and 80286-based systems**
- **Compatible with IEEE floating-point standard 754**
- **Static CMOS design does not require a minimum clock rate, resulting in significantly lower power dissipation**
- **Performs single-, double-, and extended-precision floating-point, as well as word, short, and long integer and 18-digit BCD conversions**
- **Adds trigonometric, logarithmic, exponential, and arithmetic instructions to the 80C286 instruction set**

GENERAL DESCRIPTION

The AMD 80C287 math coprocessor is implemented in AMD's advanced static CMOS process that allows for significantly higher speeds at a much lower power dissipation than traditional NMOS versions or standard CMOS. The AMD 80C287 math coprocessor is a high-performance arithmetic processor that expands the 80C286 instruction set with floating-point instructions including transcendentals, and integer and BCD conversions. The floating-point operations comply with the IEEE Standard 754. The device is available in 10- and 12-MHz speed grades and is provided in 44-pin PLCC and 40-pin DIP packages. When coupled with the 80C286, the AMD 80C287 math coprocessor provides a complete solution for high-performance numeric processing applications.

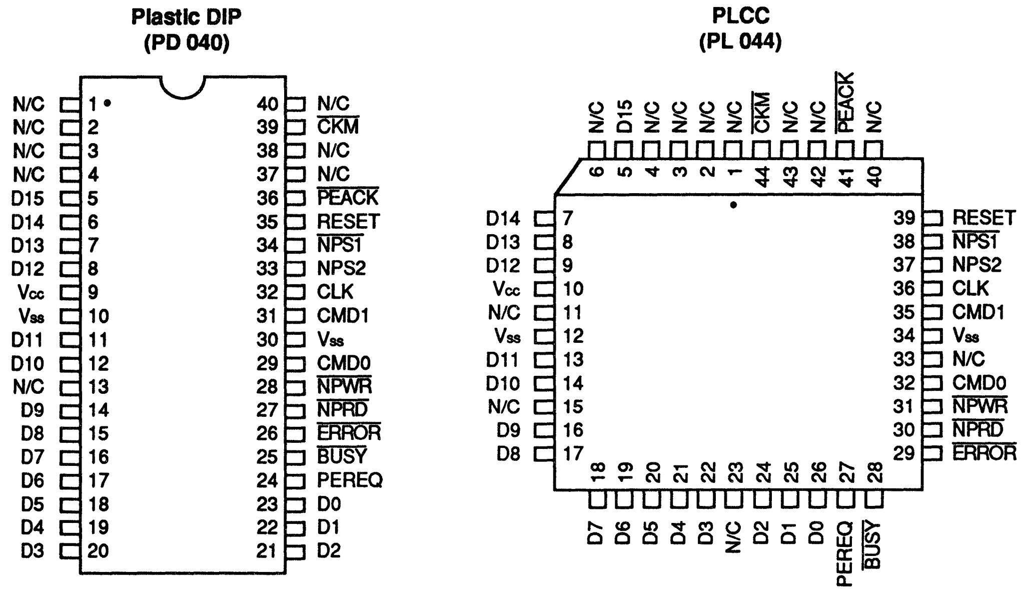
BLOCK DIAGRAM



11671A-001

Publication: 11671 Rev. D Amendment /0
Issue Date: December 1991

CONNECTION DIAGRAM



Note: N/C pins should not be connected. Pin 1 is marked for orientation.

11671A-002

PIN DESCRIPTION

$\overline{\text{BUSY}}$
Busy Status (Output; Active Low)

A Low level indicates that the AMD 80C287 math coprocessor is currently executing a command.

CKM
Clock Mode Signal (Input)

When CKM is High, the CLK is used directly. When CKM is Low, CLK is divided by three. This input must be either High or Low 20-CLK cycles before RESET goes Low.

CLK
Clock (Input)

Provides timing for AMD 80C287 math coprocessor operations.

CMD1, CMD0
Command Lines (Inputs)

CMD_1 and CMD_0, along with select inputs, allow the CPU to direct the AMD 80C287 math coprocessor operations. These inputs are timed relative to the read and write strobes.

D15–D0
Data (Inputs/Outputs)

Bi-directional data bus. These inputs are timed relative to the read and write strobes.

$\overline{\text{ERROR}}$
Error Status (Output; Active Low)

Reflects the error summary status bit of the status word. A Low level indicates that an unmasked exception condition exists.

$\overline{\text{NPRD}}$
Numeric Processor Read (Input; Active Low)

A Low level enables transfer of data from the AMD 80C287 math coprocessor. This input may be asynchronous to the AMD 80C287 clock.

$\overline{\text{NPS1}}$, NPS2
Numeric Processor Selects (Inputs)

Indicates the CPU is transferring data to and from the AMD 80C287 math coprocessor. Asserting both signals ($\overline{\text{NPS1}}$ Low and NPS2 High) enables the AMD 80C287 math coprocessor to transfer floating-point data or instructions. No data transfers involving the AMD 80C287 math coprocessor will occur unless the AMD 80C287 math coprocessor is selected via $\overline{\text{NPS1}}$ and NPS2. These inputs are timed relative to the read and write strobes.

$\overline{\text{NPWR}}$
Numeric Processor Write (Input; Active Low)

A Low level enables transfer of data to the AMD 80C287 math coprocessor. This input may be asynchronous to the AMD 80C287 clock.

$\overline{\text{PEACK}}$
Processor Extension Acknowledge (Input; Active Low)

A Low level indicates that the request signal (PEREQ) has been recognized. $\overline{\text{PEACK}}$ causes the request (PEREQ) to be withdrawn when no more transfers are required. $\overline{\text{PEACK}}$ may be asynchronous to the AMD 80C287 clock.

PEREQ
Processor Extension Request (Output)

A High level indicates that the AMD 80C287 math coprocessor is ready to transfer data. PEREQ will be disabled upon assertion of $\overline{\text{PEACK}}$ or upon actual data transfer, whichever occurs first, when no more transfers are required.

RESET
System Reset (Input)

Reset causes the AMD 80C287 math coprocessor to immediately terminate its present activity and enter a dormant state. Reset must be High for more than four CLK cycles. For proper initialization the High-Low transition must occur no sooner than 50 µs after V_{CC} and CLK meet their DC and AC specifications.

V_{cc}
+5 V Supply (Input)

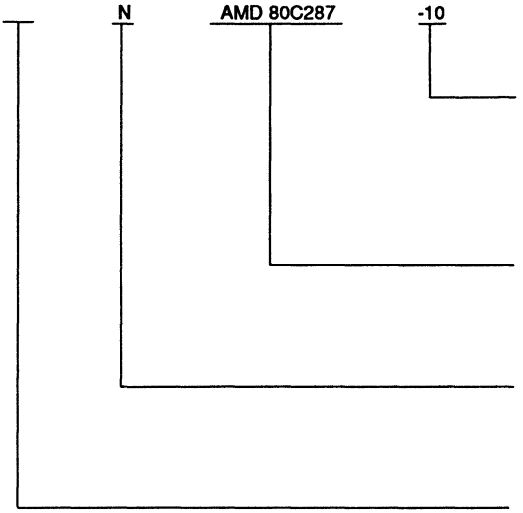
V_{ss}
System Ground (Input)

Both pins must be connected to ground.

ORDERING INFORMATION

Commodity Products

AMD products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



SPEED OPTION
–12 = 12.5 MHz
–10 = 10 MHz

DEVICE NUMBER/DESCRIPTION
AMD 80C287
80-Bit CMOS Math Coprocessor

PACKAGE TYPE
N = 44-pin Plastic Leaded Chip Carrier (PL 044)
P = 40-pin Plastic DIP (PD 040)

TEMPERATURE RANGE
Blank = Commercial (T_C = 0°C to 70°C)

Valid Combination	
N, P	AMD 80C287 –12
	AMD 80C287 –10

Valid Combination

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released valid combinations.

SIMPLIFIED FUNCTIONAL DESCRIPTION

The AMD 80C287 math coprocessor is internally divided into two basic processing units; the numeric execution unit, and the bus interface unit as shown in the block diagram. The numeric execution unit performs numeric instructions. The bus interface unit receives and decodes instructions, executes processor control instructions, and requests operands transfers to and from memory. The 80C286 may execute non-numeric instruction concurrently with numeric instruction executed on the AMD 80C287 math coprocessor. Synchronization and error recognition occurs when the next numeric instruction is decoded by the 80C286.

The Numeric Execution Unit

The numeric execution data path is 80 bits wide. All operands are converted to the internal 80-bit format before use. These instructions include arithmetic, transcendental, constant, and data transfer instructions.

The Bus Interface Unit

The bus interface unit decodes the ESC instruction executed by the 80C286. The signal $\overline{\text{BUSY}}$ is activated for the 80C286/AMD 80C287 synchronization and the signal $\overline{\text{ERROR}}$ is activated for error detection. $\overline{\text{BUSY}}$ is activated when an instruction is transferred and deactivated when the instruction completes. $\overline{\text{ERROR}}$ will be asserted if an error has occurred when $\overline{\text{BUSY}}$ is deactivated.

The signals PEREQ, $\overline{\text{PEACK}}$, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$, $\overline{\text{NPS1}}$, CMD0, CMD1, and NPS2 control data transfers between the AMD 80C287 math coprocessor and the 80C286. The 80C286 performs the actual data transfer with memory.

The Register Stack

The register stack contains eight 80-bit data registers, organized as a push down stack. Operations are performed on the stack top, between the stack top and another register, or between the stack top and memory.

System Configuration with 80C286

A simplified block diagram of the AMD 80C287 interface to a 80C286 CPU is shown in Figure 1. The AMD 80C287 math coprocessor can operate concurrently with the host CPU. The signals PEREQ, $\overline{\text{PEACK}}$, $\overline{\text{BUSY}}$, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$, CMD0, and CMD1 allow the AMD 80C287 math coprocessor to receive instructions and data from the 80C286. Detection of errors are indicated to the CPU by asserting the signal $\overline{\text{ERROR}}$. The address decode logic, bus control and timing logic are shown in this implementation using AMD PAL® devices but may also be accomplished using standard chip sets.

The AMD 80C287 math coprocessor operates either directly from the CPU clock or with a dedicated clock. The AMD 80C287 math coprocessor functions at two-thirds the frequency of the 80C286 when operating with the CPU clock (i.e., for a 16-MHz 80C286, the 32-MHz clock is divided down to 10.6 MHz).

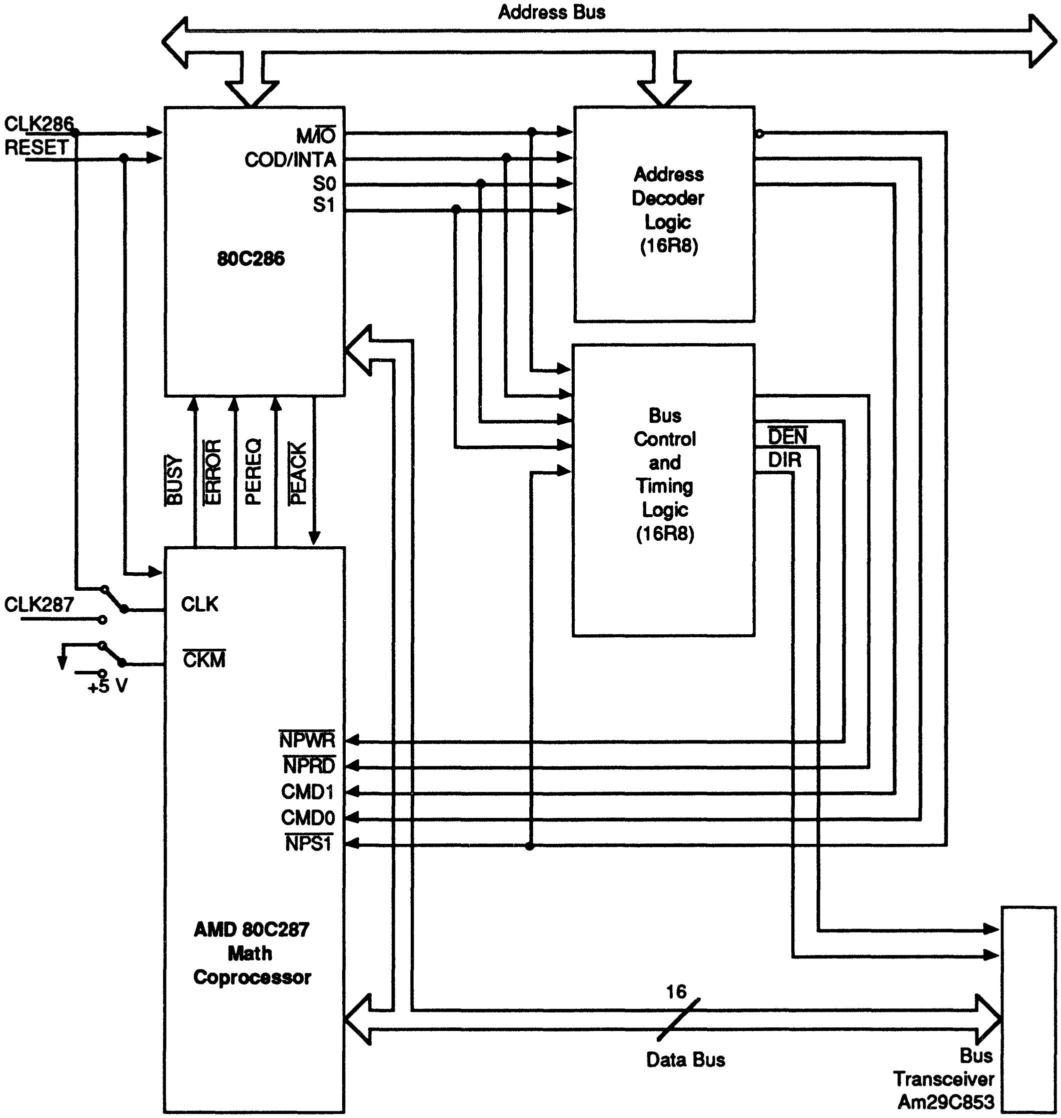


Figure 1. The 80C286/AMD 80C287 Simplified System Configuration

11671A-003

ELECTRICAL AND SWITCHING CHARACTERISTICS

Absolute Maximum Ratings

Storage Temperature	–65 to +150° C
Ambient Temperature Under Bias	–55 to +125° C
Supply Voltage to Ground Potential Continuous	–1.0 to +7.0 V
DC Voltage Applied to Outputs for High Output State	–0.3 V to + V_{CC} +0.3 V
DC Input Voltage	–0.3 to V_{CC} +0.3 V
DC Output Current, into Low Outputs	30 mA
DC Input Current	–10 to +10 mA
Power Dissipation (Max.)	0.5 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect devices reliability.

Operating Ranges

Commercial (C) Devices

Temperature, Ambient (T_A) (also meets 0 to 100°C Case Temperature (T_C) for laptop requirements)	0 to +70°C
Supply Voltage (V_{CC})	+4.5 to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating range unless otherwise specified (for APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted).

Parameter Symbol	Parameter Description	Test Conditions		Min	Max	Unit
V_{OH}	Output High Voltage	V_{CC} = Min. V_{IN} = V_{IL} or V_{IH}	I_{OH} = -0.4 mA	2.4		V
V_{OL}	Output Low Voltage	V_{CC} = Min. V_{IN} = V_{IL} or V_{IH}	I_{OL} = 3 mA		0.45	
V_{IH}	Guaranteed Input Logical High Voltage (see note below)			2.0	V_{CC} +0.5	V
V_{IL}	Guaranteed Input Logical Low Voltage (see note below)			–0.5	0.8	V
V_{IHC}	Clock Input High Voltage CKM = 1 CKM = 0			 2.0 3.8	 V_{CC} +1.0 V_{CC} +1.0	 V V
V_{ILC}	Clock Input Low Voltage $\overline{CKM}$ = 1 $\overline{CKM}$ = 0			 –0.5 –0.5	 0.8 0.6	 V V
I_{LI}	Input Leakage Current	0 V ≤ V_{IN} ≤ V_{CC}			±10	µA
I_{OZH}	Off-State (High Impedance) Output Current	V_{CC} = Max, V_O = 2.4 V			10	µA
I_{OZL}	Off-State (High Impedance) Output Current	V_{CC} = Max, V_O = 0.45 V			–10	µA
I_{CCD}	Power Supply Current, Operating	V_{CC} = Max Outputs Unloaded		10 mA + 5 mA/MHz		
I_{CCS}	Power Supply Current, Static	V_{CC} = Max, V_{IN} – V_{CC} or GND, I_O = 0 µA		20 mA		

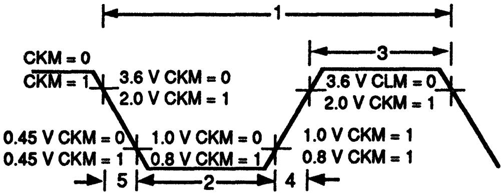
Note: These input levels provide zero-noise immunity and should only be statically tested in a noise-free environment (not functionally tested).

SWITCHING CHARACTERISTICS over commercial operating range

No.	Parameter Description	AMD 80C287-10		AMD 80C287-12		Unit
		Min	Max	Min	Max	
1	Clock Period					
	CKM = 1	100		80		ns
	CKM = 0	32		26		ns
2	Clock Low Time					
	CKM = 1	45		35		ns
	CKM = 0	11		9		ns
3	Clock High Time					
	CKM = 1	28		22		ns
	CKM = 0	11		9		ns
4	Clock Rise Time		10		8	ns
5	Clock Fall Time		10		8	ns
6	Data Setup to $\overline{NPWR}$ Inactive	75		75		ns
7	Data Hold from $\overline{NPWR}$ Inactive	18		10		ns
8	$\overline{NPWR}$, $\overline{NPRD}$ Active Time	90		70		ns
9	Command Valid Setup Time	0		0		ns
10	$\overline{PEREQ}$ Active to $\overline{NPRD}$ Active	100		80		ns
11	$\overline{PEACK}$ Active Time	60		50		ns
12	$\overline{PEACK}$ Inactive Time	200		160		ns
13	$\overline{PEACK}$ Inactive to $\overline{NPRD}$, $\overline{NPWR}$ Inactive	40		32		ns
14	$\overline{NPRD}$, $\overline{NPWR}$ Inactive to $\overline{PEACK}$ Active		–30		–30	ns
15	Command Valid Hold Time	22		18		ns
16	$\overline{PEACK}$ Active Setup to $\overline{NPRD}$, $\overline{NPWR}$	40		30		ns
17	$\overline{NPRD}$, $\overline{NPWR}$ to CLK Setup	53		40		ns
18	$\overline{NPRD}$, $\overline{NPWR}$ CLK Hold	37		29		ns
19	RESET to CLK Setup	20		20		ns
20	RESET from CLK Hold	20		20		ns
21	$\overline{NPRD}$ Inactive to Data Float		21		17	ns
22	$\overline{NPRD}$ Active to Data Valid		60		50	ns
23	$\overline{ERROR}$ Active to $\overline{BUSY}$ Inactive	100		100		ns
24	$\overline{NPWR}$, Active to $\overline{BUSY}$ Active		100		80	ns
25	$\overline{PEACK}$ Active to $\overline{PEREQ}$ Inactive		100		80	ns
26	$\overline{NPRD}$, $\overline{NPWR}$ Active to $\overline{PEREQ}$ Inactive	100		80		ns
27	Command Inactive Time					
	Write to Write	75		60		ns
	Read to Read	75		60		ns
	Write to Read	75		60		ns
	Read to Write	75		60		ns
28	Data Hold from Time $\overline{NPRD}$ Inactive	3		1		ns

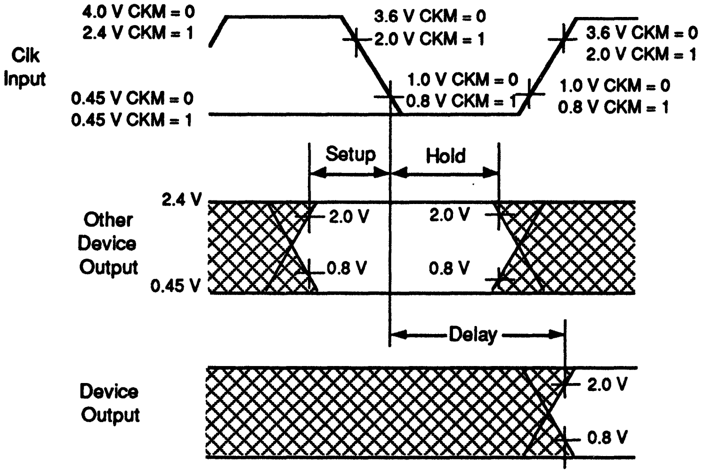
SWITCHING WAVEFORMS

AC Drive and Measurement Points—CLK Input



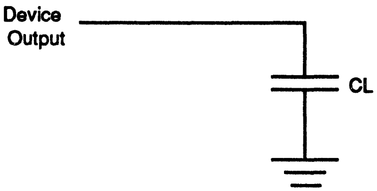
11959A-004

AC Setup, Hold and Delay Time Measurement—General



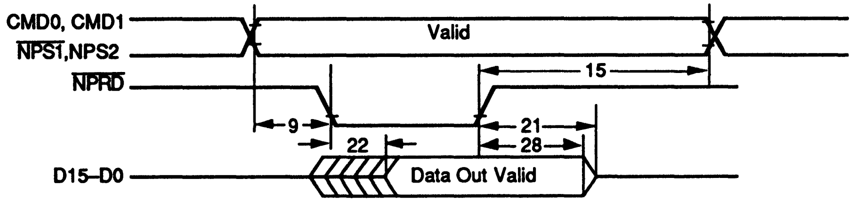
11959A-005

AC Test Loading on Outputs



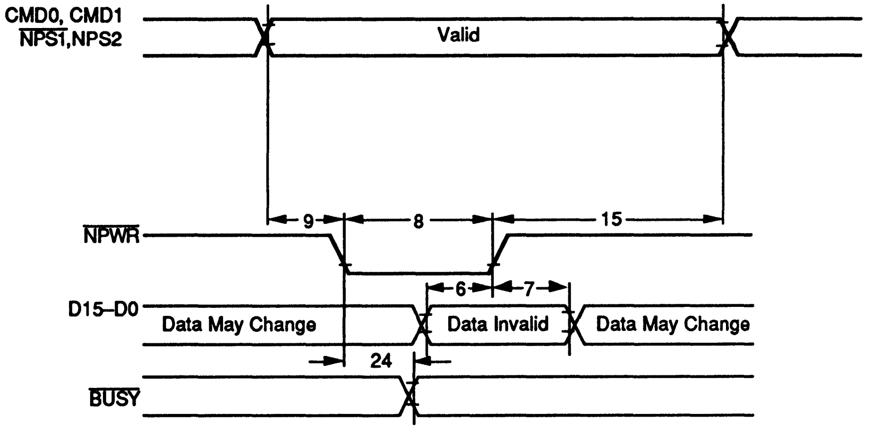
11959A-006

Read Timing from AMD 80C287 Math Coprocessor



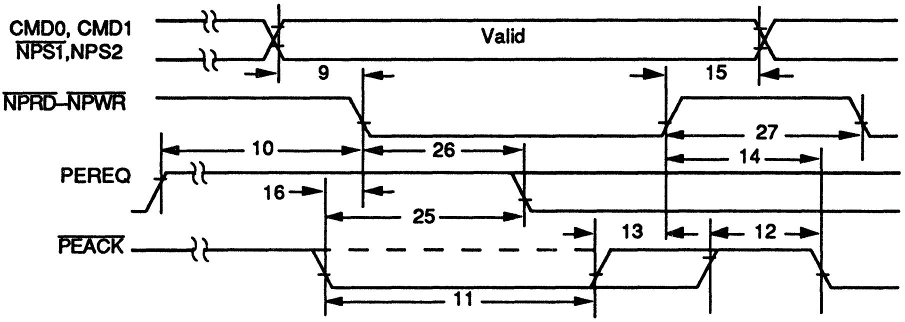
11959A-007

Write Timing from AMD 80C287 Math Coprocessor



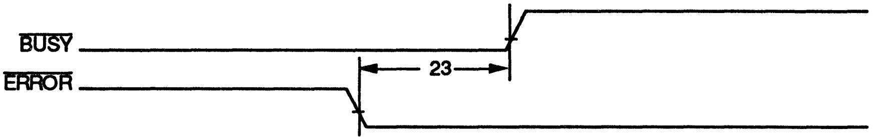
11959A-008

Data Channel Timing (Initiated by AMD 80C287 Math Coprocessor)



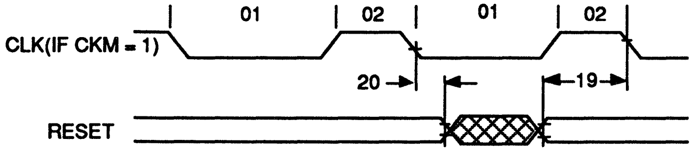
11959A-009

Error Output Timing



11959A-010

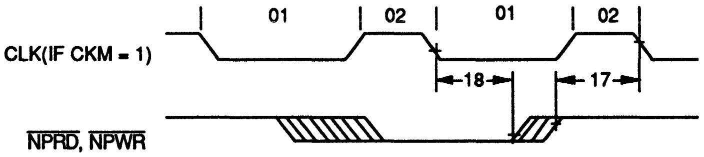
CLK, Reset Timing (CKM = 1)



NOTE: Reset, NPWR, NPRD are inputs asynchronous to CLK. Timing requirements above are given for testing purposes only, to assure recognition at a specific CLK edge.

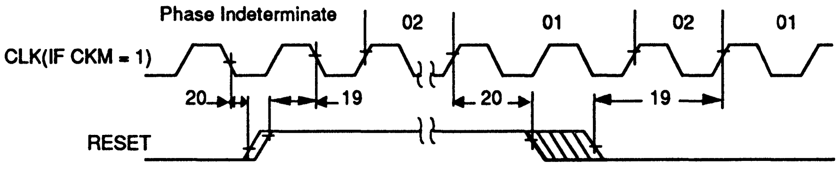
11959A-011

CLK, NPRD, NPWR TIMING (CKM = 1)



11959A-012

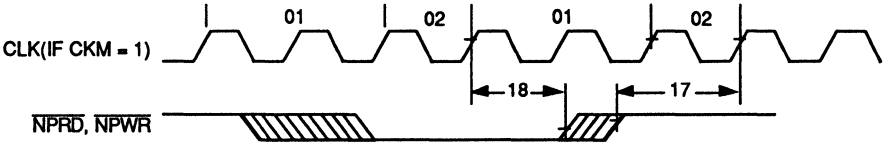
CLK, RESET TIMING (CKM = 0)



NOTE: Reset must meet timing shown to guarantee known phase of internal ÷ 3 circuit.

11959A-013

CLK, NPRD, NPWR TIMING (CKM = 0)



11959A-014

AMD 80EC287™

Advanced
Micro
Devices

Enhanced 80-Bit CMOS Math Coprocessor

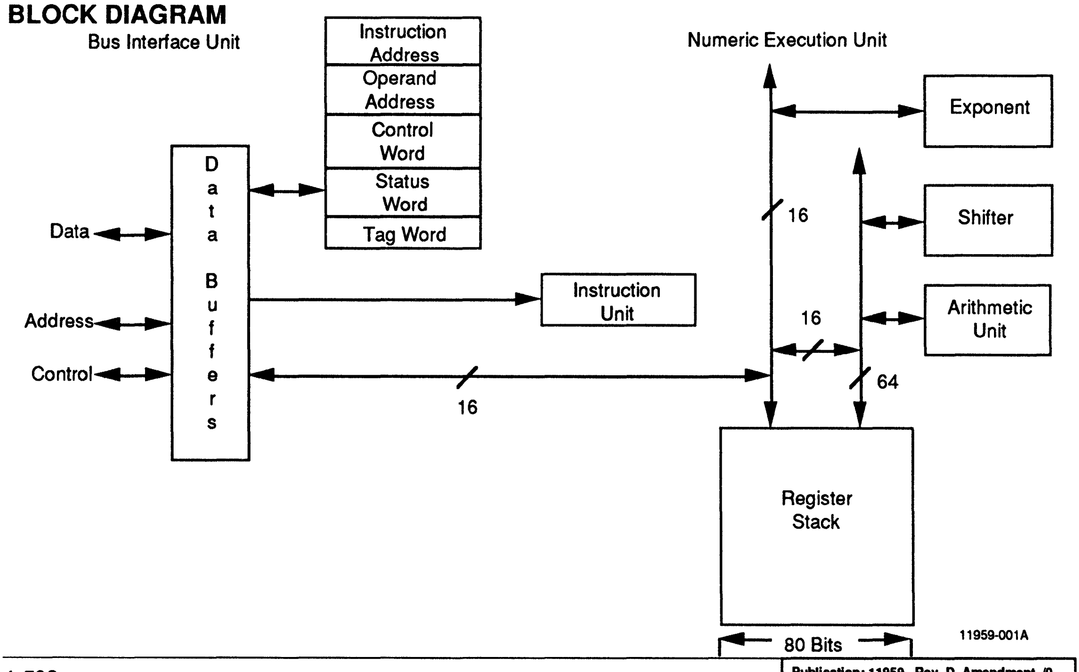
DISTINCTIVE CHARACTERISTICS

- **Pin compatible and functionally equivalent to the Intel 80287**
- **High-performance CMOS process yields 10-, and 12-MHz speed grades**
- **Enhanced sleep feature automatically shuts off the internal clock when no instruction is executing, reducing power consumption. This feature is transparent to the user**
- **Available in space-saving 44-pin PLCC as well as 40-pin DIP**
- **80-bit numeric accelerator for 80C286- and 80286-based systems**
- **Compatible with IEEE floating point Standard 754**
- **Static CMOS design does not require a minimum clock rate, resulting in significantly lower power dissipation**
- **Performs single-, double-, and extended-precision floating point, as well as word, short, and long integer and 18-digit BCD conversions**
- **Adds trigonometric, logarithmic, exponential, and arithmetic instructions to the 80C286 instruction set**

GENERAL DESCRIPTION

The AMD 80EC287 processor is implemented in AMD's advanced static CMOS process that allows for significantly higher speeds at a much lower power dissipation than traditional NMOS versions or standard CMOS. The AMD 80EC287 processor is a high-performance arithmetic processor that expands the 80C286 instruction set with floating-point instructions including transcendentals, and integer and BCD conversions. The AMD 80EC287 processor is functionally equivalent to the Intel 80287 and AMD 80C287 math coprocessor plus adds a low-power sleep feature for battery powered applications. This enhanced AMD 80EC287 processor can be a direct replacement for an AMD 80C287 processor. The sleep feature is an automatic inherent feature of the device and thus requires no external entry. The floating point operations comply with the IEEE Standard 754. The device is available in 10- and 12-MHz speed grades and is provided in 44-pin PLCC and 40-pin DIP packages. When coupled with the 80C286, the AMD 80EC287 processor provides a complete solution for high-performance numeric processing applications.

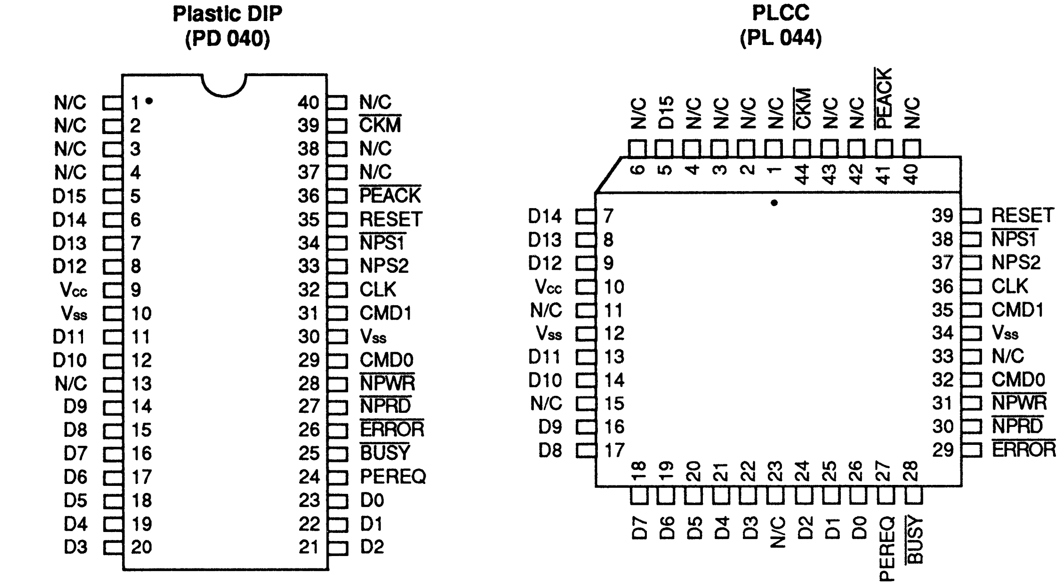
BLOCK DIAGRAM



11959-001A

Publication: 11959 Rev. D Amendment /0
Issue Date: December 1991

CONNECTION DIAGRAMS



Notes: N/C pins should not be conntected.
Pin 1 is marked for orientation

11959-002A

PIN DESCRIPTION

$\overline{\text{BUSY}}$
Busy Status (Output; Active Low)

A Low level indicates that the AMD 80EC287 math coprocessor is currently executing a command.

CKM
Clock Mode Signal (Input)

When CKM is High, the CLK is used directly. When CKM is Low, CLK is divided by three. This input must be either High or Low 20-CLK cycles before RESET goes Low.

CLK
Clock (Input)

Provides timing for AMD 80EC287 math coprocessor operations.

CMD1, CMD0
Command Lines (Inputs)

CMD1 and CMD0, along with select inputs, allow the CPU to direct the AMD 80EC287 math coprocessor operations. These inputs are timed relative to the read and write strobes.

D15–D0
Data (Inputs/Outputs)

Bi-directional data bus. These inputs are timed relative to the read and write strobes.

$\overline{\text{ERROR}}$
Error Status (Output; Active Low)

Reflects the error summary status bit of the status word. A Low level indicates that an unmasked exception condition exists.

$\overline{\text{NPRD}}$
Numeric Processor Read (Input; Active Low)

A Low level enables transfer of data from the AMD 80EC287 math coprocessor. This input may be asyn-chronous to the AMD 80EC287 clock.

$\overline{\text{NPS1}}$, NPS2
Numeric Processor Selects (Inputs)

Indicates the CPU is transferring data to and from the AMD 80EC287 math coprocessor. Asserting both signals ($\overline{\text{NPS1}}$ Low and NPS2 High) enables the AMD 80EC287 math coprocessor to transfer floating-point data or instructions. No data transfers involving the AMD 80EC287 math coprocessor will occur $\overline{\text{unless}}$ the AMD 80EC287 math coprocessor is selected via $\overline{\text{NPS1}}$ and NPS2. These inputs are timed relative to the read and write strobes.

$\overline{\text{NPWR}}$
Numeric Processor Write (Input; Active Low)

A Low level enables transfer of data to the AMD 80EC287 math coprocessor. This input may be asynchronous to the AMD 80EC287 clock.

$\overline{\text{PEACK}}$
Processor Extension Acknowledge (Input; Active Low)

A Low level indicates that the request signal (PEREQ) has been recognized. $\overline{\text{PEACK}}$ causes the request (PEREQ) to be withdrawn when no more transfers are required. $\overline{\text{PEACK}}$ may be asynchronous to the AMD 80EC287 clock.

PEREQ
Processor Extension Request (Output)

A High level indicates that the AMD 80EC287 math coprocessor is ready to transfer data. PEREQ will be disabled upon assertion of $\overline{\text{PEACK}}$ or upon actual data transfer, whichever occurs first, when no more transfers are required.

RESET
System Reset (Input)

Reset causes the AMD 80EC287 math coprocessor to immediately terminate its present activity and enter a dormant state. Reset must be High for more than four CLK cycles. For proper initialization the High-Low transition must occur no sooner than 50 µs after V_{CC} and CLK meet their DC and AC specifications.

V_{cc}
+5 V Supply (Input)

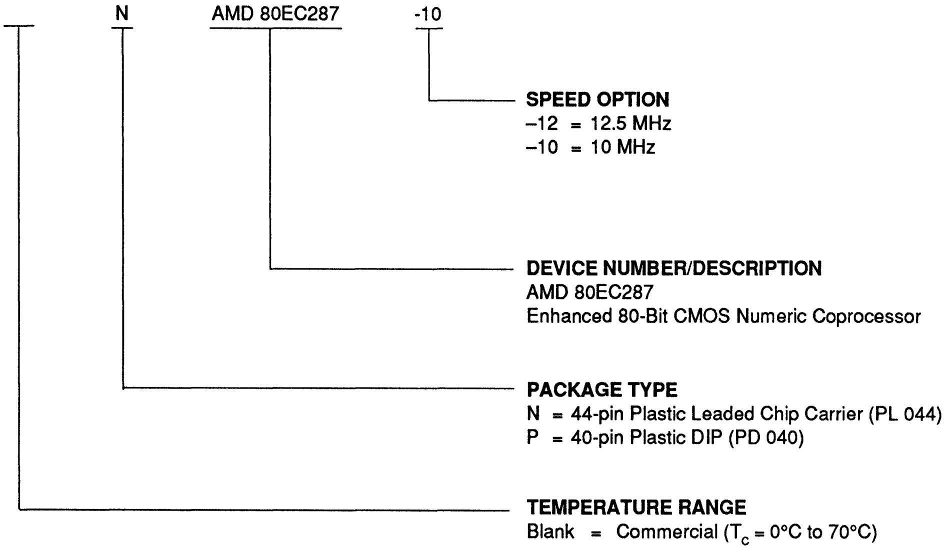
V_{ss}
System Ground (Input)

Both pins must be connected to ground.

ORDERING INFORMATION

Commodity Products

AMD products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the elements below.



N AMD 80EC287 -10

SPEED OPTION
–12 = 12.5 MHz
–10 = 10 MHz

DEVICE NUMBER/DESCRIPTION
AMD 80EC287
Enhanced 80-Bit CMOS Numeric Coprocessor

PACKAGE TYPE
N = 44-pin Plastic Leaded Chip Carrier (PL 044)
P = 40-pin Plastic DIP (PD 040)

TEMPERATURE RANGE
Blank = Commercial (T_C = 0°C to 70°C)

Valid Combination	
N, P	AMD 80EC287 –12
	AMD 80EC287 –10

Valid Combination

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released valid combinations.

SIMPLIFIED FUNCTIONAL DESCRIPTION

The AMD 80EC287 numeric processor is internally divided into two basic processing units: the numeric execution unit, and the bus interface unit as shown in the block diagram. The numeric execution unit performs numeric instructions. The bus interface unit receives and decodes instructions, executes processor control instructions, and requests operands transfers to and from memory. The 80C286 may execute non-numeric instruction concurrently with numeric instruction executed on the AMD 80EC287 processor. Synchronization and error recognition occurs when the next numeric instruction is decoded by the 80C286.

The Numeric Execution Unit

The numeric execution data path is 80-bits wide. All operands are converted to the internal 80-bit format before use. These instructions include arithmetic, transcendental, constant, and data transfer instructions.

The Bus Interface Unit

The Bus Interface Unit decodes the ESC instruction executed by the 80C286. The signal $\overline{\text{BUSY}}$ is activated for AMD 80C286/80EC287 processor synchronization and the signal $\overline{\text{ERROR}}$ is activated for error detection. $\overline{\text{BUSY}}$ is activated when an instruction is transferred and deactivated when the instruction completes. $\overline{\text{ERROR}}$ will be asserted if an error has occurred when $\overline{\text{BUSY}}$ is deactivated.

The signals PEREQ, $\overline{\text{PEACK}}$, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$, $\overline{\text{NPS1}}$, CMD0, CMD1, and NPS2 control data transfers between the AMD 80EC287 processor and the 80C286. The 80C286 performs the actual data transfer with memory.

The Register Stack

The register stack contains eight 80-bit data registers, organized as a push down stack. Operations are performed on the stack top, between the stack top and another register, or between the stack top and memory.

System Configuration with 80C286

A simplified block diagram of the AMD 80EC287 processor interface to a 80C286 CPU is shown in Figure 1. The AMD 80EC287 processor can operate concurrently with the host CPU. The signals PEREQ, $\overline{\text{PEACK}}$, $\overline{\text{BUSY}}$, $\overline{\text{NPRD}}$, $\overline{\text{NPWR}}$, CMD0, and CMD1 allow the AMD 80EC287 processor to receive instructions and data from the 80C286. Detection of errors are indicated to the CPU by asserting the signal $\overline{\text{ERROR}}$. The address decode logic, bus control and timing logic is shown in this implementation using AMD PAL® devices but may also be accomplished using standard chip sets.

The AMD 80EC287 processor operates either directly from the CPU clock or with a dedicated clock. The AMD 80EC287 processor functions at two-thirds the frequency of the 80C286 when operating with the CPU clock (i.e., for a 16-MHz 80C286, the 32-MHz clock is divided down to 10.6 MHz).

Sleep Feature

The AMD 80EC287 processor clock runs only while an instruction is executing. The internal clock shuts itself off when no instruction is executing, thus reducing power consumption. This feature is completely transparent to the user and requires no external circuitry or design interface.

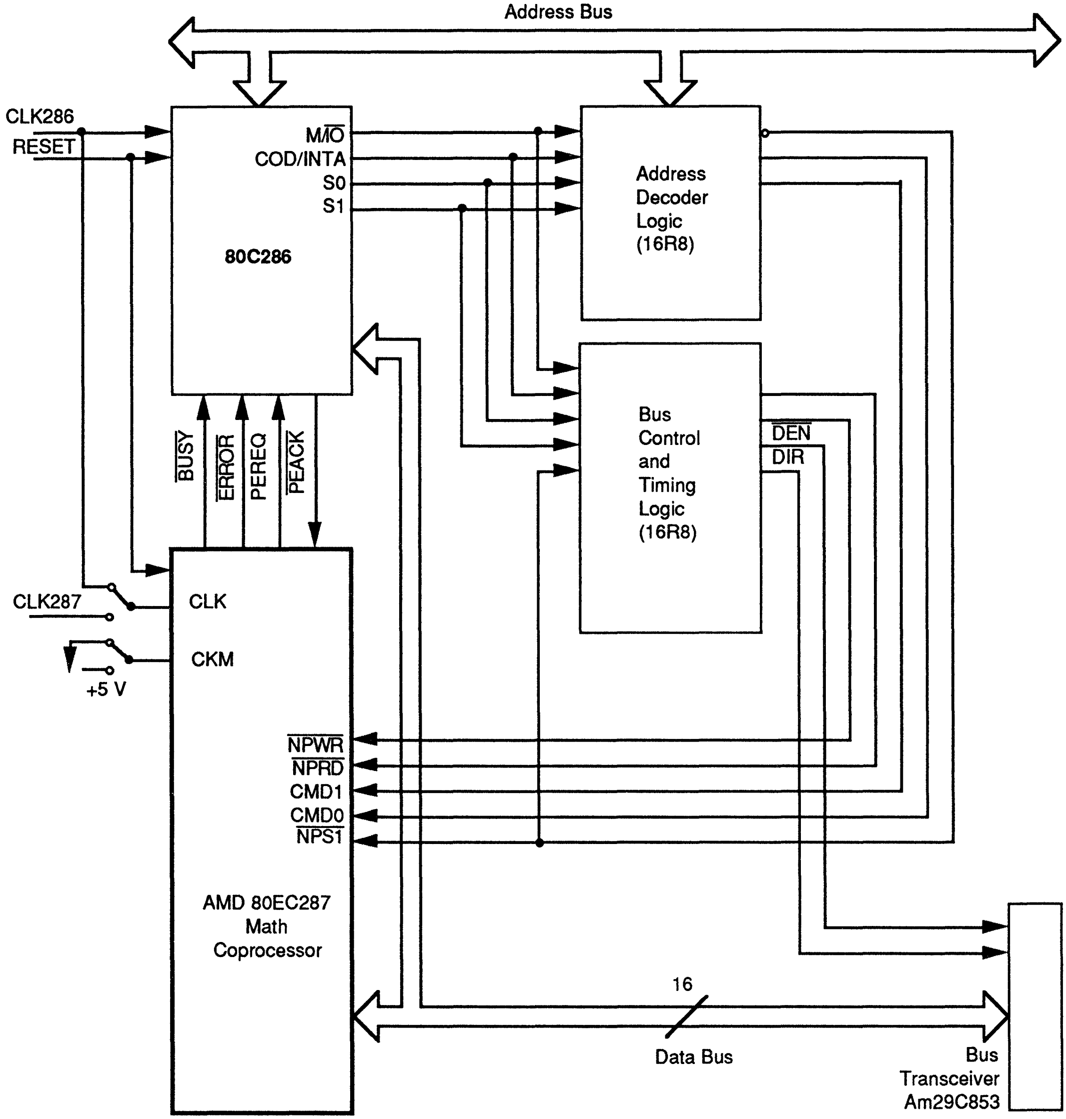


Figure 1. AMD 80C286/80EC287 Simplified System Configuration

11959-003A

ABSOLUTE MAXIMUM RATINGS

Storage Temperature .... –65°C to +150° C
Ambient Temperature Under Bias .... –55°C to +125° C
Supply Voltage to Ground Potential
Continuous .... –1.0 to +7.0 V
DC Voltage Applied to Outputs
for High Output State .... –0.3 V to + V_{CC} +0.3 V
DC Input Voltage .... –0.3 to V_{CC} +0.3 V
DC Output Current, into Low Outputs .... 30 mA
DC Input Current .... –10 to +10 mA
Power Dissipation (Max) .... 0.5 W

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect devices reliability.

OPERATING RANGES

Commercial (C) Devices
Temperature, Ambient (T_A) .... 0°C to +70°C
(also meets 0 to 100°C Case Temperature (T_C) for laptop requirements)
Supply Voltage (V_{CC}) .... + 4.5 to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating range unless otherwise specified

Parameter Symbol	Parameter Description	Test Conditions		Min	Max	Unit
V_{OH}	Output High Voltage	V_{CC} = Min V_{IN} = V_{IL} or V_{IH}	I_{OH} = –0.4 mA	2.4		V
V_{OL}	Output Low Voltage	V_{CC} = Min V_{IN} = V_{IL} or V_{IH}	I_{OL} = 3 mA		0.45	
V_{IH}	Guaranteed Input Logical High Voltage (Note 1)			2.0	V_{CC} +0.5	V
V_{IL}	Guaranteed Input Logical Low Voltage (Note 1)			–0.5	0.8	V
V_{IHC}	Clock Input High Voltage					
	CKM = 1			2.0	V_{CC} +1.0	V
	CKM = 0			3.8	V_{CC} +1.0	V
V_{ILC}	Clock Input Low Voltage					
	CKM = 1			–0.5	0.8	V
	CKM = 0			–0.5	0.6	V
I_{LI}	Input Leakage Current	0 V ≤ V_{IN} ≤ V_{CC}			±10	µA
I_{OZH}	Off-State (High Impedance) Output Current	V_{CC} = Max, V_o = 2.4 V			10	µA
I_{OZL}	Off-State (High Impedance) Output Current	V_{CC} = Max, V_o = 0.45 V			–10	µA
I_{CCS}	Power Supply Current, Static	V_{CC} = Max, V_{IN} – V_{CC} or GND, I_o= 0 µA		20 mA		
I_{CCD}	Supply Current, operating	V_{CC} = Max Outputs Unloaded		10 mA + 5 mA/MHz (Note 2)		
Icc_{SM}	Power Supply Current, Sleep Mode	V_{CC} = Max Outputs Unloaded		10 mA + 1 mA/MHz		

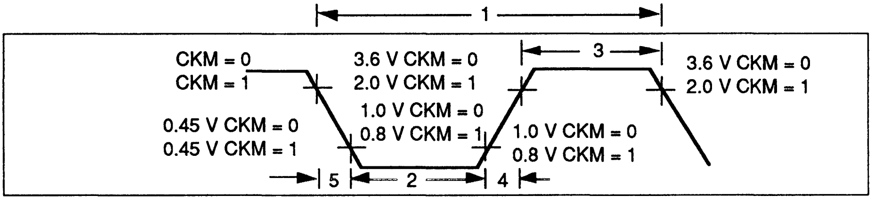
Notes: 1. These input levels provide zero-noise immunity and should only be statically tested in a noise-free environment (not functionally tested).

2. This reduces to I_{CCSM} when no instruction is executing, reducing overall power consumption.

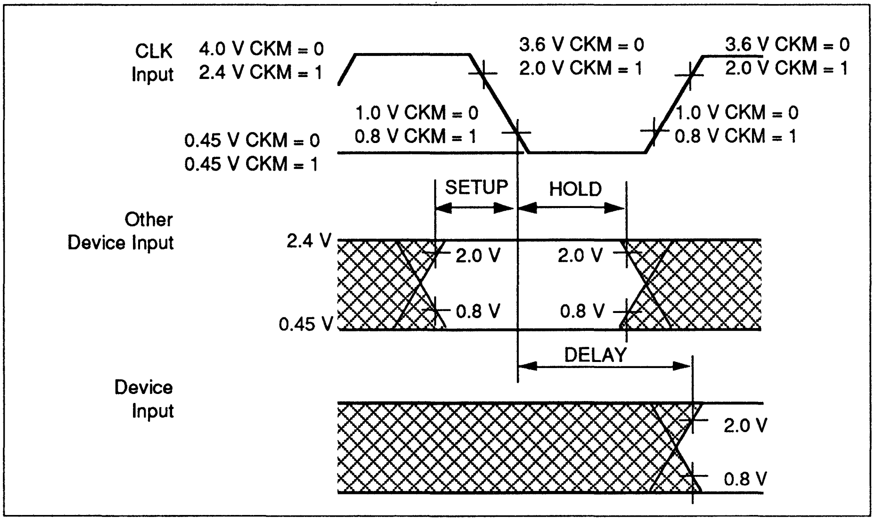
SWITCHING CHARACTERISTICS over COMMERCIAL operating range

No.	Parameter Description	Test Conditions	AMD 80EC287-10		AMD 80EC287-12		
			Min	Max	Min	Max	Unit
1	Clock Period						
	CKM = 1		100		80		ns
	CKM = 0		32		26		ns
2	Clock LowTime						
	CKM = 1		45		35		ns
	CKM = 0		11		9		ns
3	Clock High Time						
	CKM = 1		28		22		ns
	CKM = 0		11		9		ns
4	Clock Rise Time			10		8	ns
5	Clock Fall Time			10		8	ns
6	Data Setup to $\overline{NPWR}$ Inactive		75		75		ns
7	Data Hold from $\overline{NPWR}$ Inactive		18		10		ns
8	$\overline{NPWR}$, $\overline{NPRD}$ Active Time		90		70		ns
9	Command Valid Setup Time		0		0		ns
10	PEREQ Active to $\overline{NPRD}$ Active		100		80		ns
11	$\overline{PEACK}$ Active Time		60		50		ns
12	$\overline{PEACK}$ Inactive Time		200		160		ns
13	$\overline{PEACK}$ Inactive to $\overline{NPRD}$, $\overline{NPWR}$ Inactive		40		32		ns
14	$\overline{NPRD}$, $\overline{NPWR}$ Inactive to $\overline{PEACK}$ Active		–30		–30		ns
15	Command Valid Hold Time		22		18		ns
16	$\overline{PEACK}$ Active Setup to $\overline{NPRD}$, $\overline{NPWR}$		40		30		ns
17	$\overline{NPRD}$, $\overline{NPWR}$ to CLK Setup		53		40		ns
18	$\overline{NPRD}$, $\overline{NPWR}$ CLK Hold		37		29		ns
19	RESET to CLK Setup		20		20		ns
20	RESET from CLK Hold		20		20		ns
21	$\overline{NPRD}$ Inactive to Data Float			21		17	ns
22	$\overline{NPRD}$ Active to Data Valid			60		50	ns
23	$\overline{ERROR}$ Active to $\overline{BUSY}$ Inactive		100		100		ns
24	$\overline{NPWR}$, Active to $\overline{BUSY}$ Active			100		80	ns
25	$\overline{PEACK}$ Active to PEREQ Inactive			100		80	ns
26	$\overline{NPRD}$, $\overline{NPWR}$ Active to PEREQ Inactive			100		80	ns
27	Command Inactive Time						
	Write to Write		75		60		ns
	Read to Read		75		60		ns
	Write to Read		75		60		ns
	Read to Write		75		60		ns
28	Data Hold from Time $\overline{NPRD}$ Inactive		3		1		ns

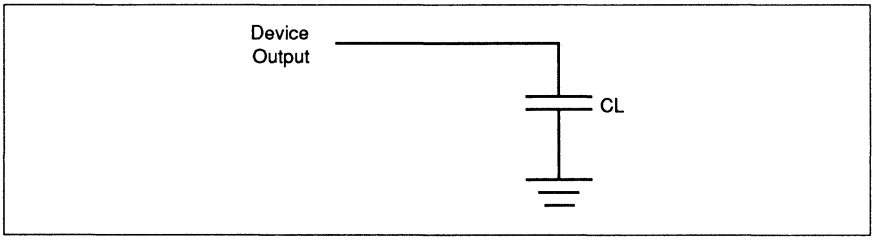
SWITCHING WAVEFORMS



AC Drive and Measurement Points—CLK Input



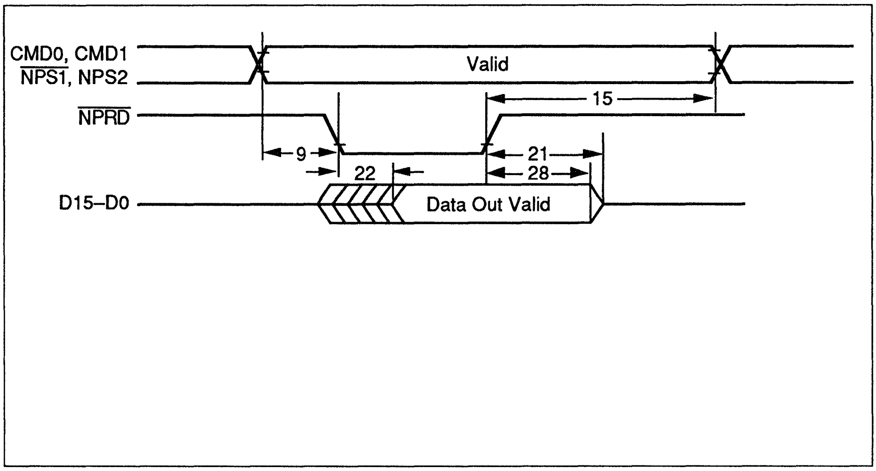
AC Setup, Hold and Delay Time Measurement—General



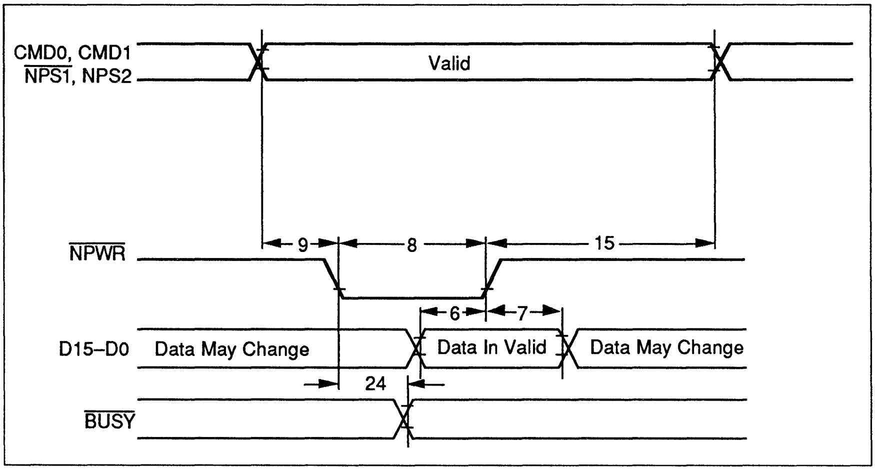
AC Test Loading on Outputs

11959-004

SWITCHING WAVEFORMS (continued)



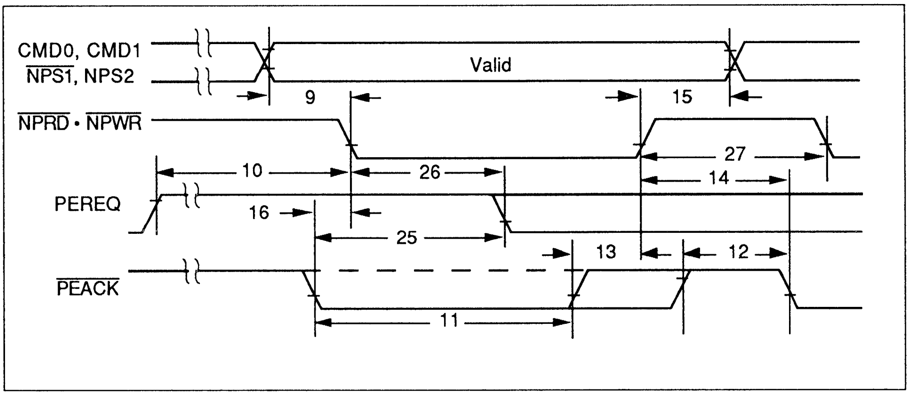
Read Timing from AMD 80EC287 Processor



Write Timing from AMD 80EC287 Processor

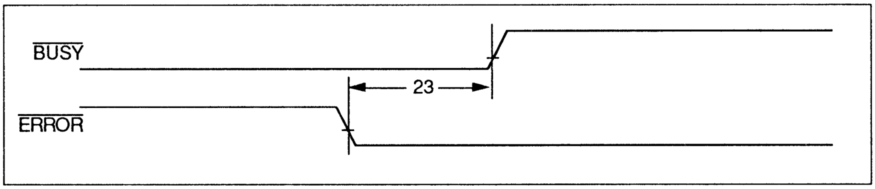
11959-005

SWITCHING WAVEFORMS (continued)

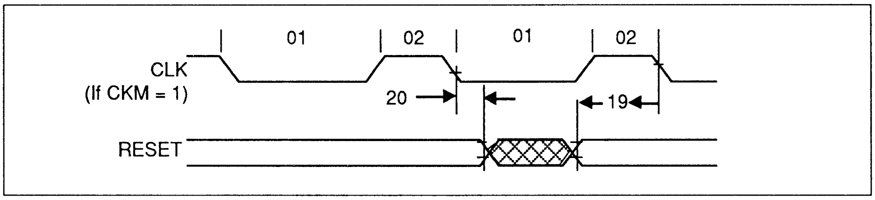


Data Channel Timing (Initiated by AMD 80EC287 Processor)

11959-006A



Error Output Timing

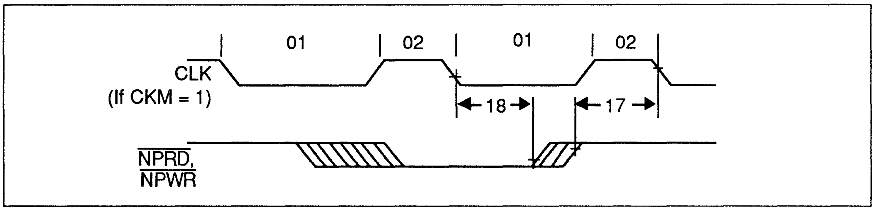


CLK, Reset Timing (CKM = 1)

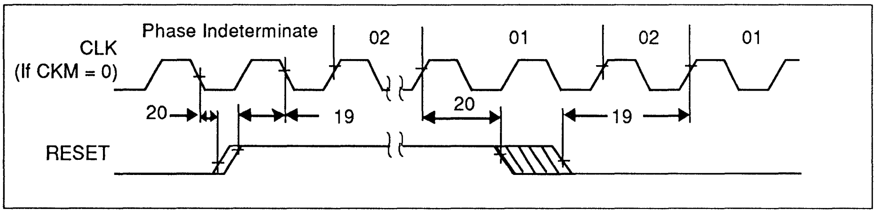
11959-007

Note: Reset, NPWR, NPRD are inputs asynchronous to CLK. Timing requirements above are given for testing purposes only to assure recognition at a specific CLK edge.

SWITCHING WAVEFORMS (continued)

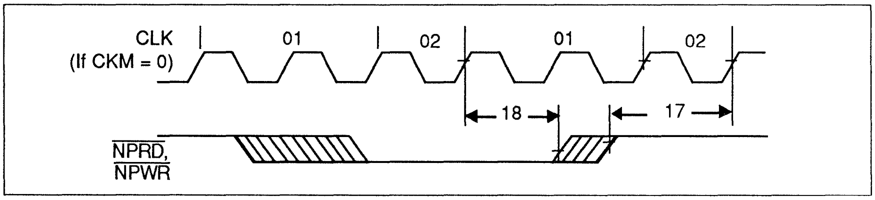


CLK, NPRD, NPWR Timing (CKM = 1)



CLK, RESET Timing (CKM = 0)

Note: Reset must meet timing shown to guarantee known phase of internal ÷ 3 circuit.



11959-008A

CLK, NPRD, NPWR Timing (CKM = 0)

Chapter 2

General Information

CHAPTER 2
General Information

THERMAL CHARACTERISTICS

Device	Package	Process	Area (KSq Mils)	Power[1, 2] (mW)	θja[3] (°C/W)	θjc[3] (°C/W)
Am386DX	PGA132	CMOS	74.0	2750	21	2
Am386SX	PQ100	CMOS	74.0	1250	45	11
	PQB100	CMOS	74.0	1250		
Am386DXL	PGA132	CMOS	74.0	2000	21	2
	PQ132	CMOS	74.0	2000	40	10
	PQB132	CMOS	74.0	2000		
Am386SXL	PQ100	CMOS	74.0	1250	45	11
	PQB100	CMOS	74.0	1250	45	11
Am386DXLV	PQ132	CMOS	74.0	446	41	9
	PQB132	CMOS	74.0	446	41	9
Am386SXLV	PQ100	CMOS	74.0	413	45	11
	PQB100	CMOS	74.0	413	45	11
Am286ZX/LX	PQ216	CMOS	247.0	1625	25	
80286	CA2068	NMOS	8.0	3000	31	8
	CGX068	NMOS	98.0	3000	37	2
80C286	CA2068	CMOS	101.4	1980	38	4
	PL 068	CMOS	101.4	1980	33	
80L286	PL 068	NMOS	98.0	2250	34	
8086	CD 040	NMOS	52.9	1700	32	7
	PD 040	NMOS	52.9	1700	44	
	CL 040	NMOS	52.9	1700	47	
8088	CD 040	NMOS	36.1	1700	34	7
	PD 040	NMOS	36.1	1700	44	
	CL 040	NMOS	36.1	1700	47	
AMD 80C287	CD 040	CMOS	82.5	600	30	5
	PL 044	CMOS	82.5	600	40	
	PD 040	CMOS	82.5	600	38	
AMD 80EC287	CD 040	CMOS	82.5	600	30	5
	PL 044	CMOS	82.5	600	40	
	PD 040	CMOS	82.5	600	38	

1. Power is the highest I_{CC} over the temperature times the nominal power supply voltage.
2. Power measured at highest possible clock frequency. Power for Am386DXLV and Am386SXLV microprocessors are at 3.3 V nominal supply voltage.
3. θja and θjc values were estimated by extrapolating measured data. Values are to be considered as typical for the device.

TapePak Application Note

Advanced
Micro
Devices

HIGH DENSITY PACKAGING

Manufacturing demands for higher board density and more cost effective systems, along with advances in micron and sub-micron technology, have made integrated circuits smaller, denser, and more complex. Consequently, more input/output (I/O) connections are required in the IC package than can be accommodated in traditional Dual-In-Line (DIP) and Plastic Leaded Chip Carrier (PLCC) packages.

AMD, in keeping pace with the increased complexity of ICs, has developed a family of fine pitch Plastic Quad Flat Pack (PQFP) packages; a reliable, high-density, low-cost, plastic package for high-lead count applications.

To facilitate the handling and testing of the fine pitch leads and preserve their formation and mechanical integrity, AMD has adopted the industry standard TapePak® package configuration and has applied it to all high pin count PQFP packages.

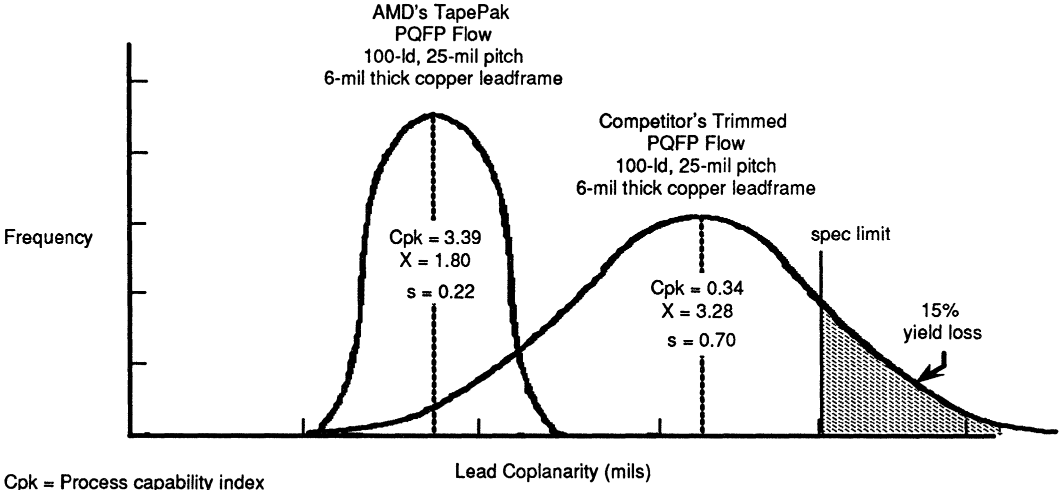
The TapePak Format

TapePak is a plastic ring formed around the perimeter of the PQFP's coplanar leads. It is molded at the same time as the package body, securing the lead frame in place and providing mechanical stability and lead protection during the manufacturing process. This configuration offers significant benefits, helping meet the challenges for testability and board assembly automation. With TapePak, board yields are increased by improved lead coplanarity, reduction in bent leads, and superior ESD protection.

TAPEPAK CONFIGURATION BENEFITS

Testability

The TapePak configuration provides extensions to the package leads, spanning them out to the perimeter of the TapePak ring. Using these as electrical contact points in test and burn-in operations preserves the actual package leads, virtually eliminating lead coplanarity and bent lead problems. Components remain in the TapePak format up to the point of board assembly. Chart number 1, based on actual data from a 60 piece sample, compares AMD's TapePak PQFP flow to that of another company's trimmed PQFP flow. AMD's lead coplanarity falls well within the industry standard specifications of 4.0 mil with a tight standard deviation. Comparatively the competition's coplanarity is higher, resulting in a 15% yield loss.



PQFP Test Through Mark Process Comparison

The TapePak lead extensions are standardized at an outside lead-pitch of 0.65 mm, which is compatible with existing automatic test equipment. With this standardization, only two socket sizes are required for AMD's entire family of TapePak PQFPs: a 36 mm ring size and a 46 mm size.

Standardized Socket Tooling

AMD has taken a leadership role, working with major socket manufacturers, to develop a TapePak compatible family of industry-standard test and burn-in sockets.

The socket is designed with an open-top, eliminating lid clearance problems during automated burn-in board loading. Curved socket leads provide the spring pressure necessary to hold the package in place, and the serrated edges of the socket leads scrub the package test-points, removing any oxidation and facilitating good electric contact. Having the test contact points along the perimeter of the ring eliminates the need for expensive pogo pins. The socket contacts splay the narrower package lead-pitch out to three rows of pins to allow connection to a through-hole pattern on the board.

Surface Mount Compatibility

The TapePak design offers all the benefits of surface-mount technology. With only the addition of a trim-and-form station, TapePak is fully compatible with existing high-speed, high-precision surface mount equipment. Trim-and-form equipment is readily available to excise the package, form the leads, and present it to a pick-and-place head for placement onto the circuit board. (For a list of pick-and-place equipment suppliers with TapePak trim-and form capability and available suppliers of trim-and-form feeders contact your local AMD sales representative). The trim-and-form specification can be based on the AMD standard 1.6 mm lead length, 1.95 mm lead length or any other specification desired. AMD's PQFP trim-and-form standards are outlined in Table 1.

Trim-and-form feeders are available for a variety of TapePak packaging methods, including coin stack tubes, flat tubes, or AMMO-Pak formats. In coin stack tubes, the PQFP/TapePak packages are stacked vertically, in flat tubes the parts are laid end -to-end. AMMO-Pak presents the parts to the trim-and-form station in the form of a taped belt, without a tube. The device quantities per carrier type and box are presented in Table 2.

Multi-tube, random access trim-and-form systems can be programmed to process several different leadcount packages within a ring size. And, with only minor modifications, automatic gravity-fed PLCC test handlers and markers can be made compatible with TapePak parts in flat tubes.

Table 1. PQFP Trim-and-Form Standards

Package Type	Lead Count	Package Stand-Off (Nominal)	Lead Length (Nominal)	Coplanarity	Bent Leads
JEDEC Metric	48-144	0.35 mm (0.0138")	1.6 mm (0.0630")	0.10 mm (0.004")	0.076 mm (0.003")
	168[1]	0.35 mm (0.0138")	1.6 mm (0.0630")	0.10 mm (0.004")	0.076 mm (0.003")
	216[1]	0.35 mm (0.0138")	1.6 mm (0.0630")	0.076 mm (0.003")	0.076 mm (0.003")
JEDEC English	100 & 132	0.0130" (0.76 mm)	0.065" (1.65 mm)	0.004" (0.10 mm)	0.003" (0.076 mm)

Note 1: Not yet JEDEC registered.

Table 2. Device Quantities Per Carrier Type and Box

Package Format	Carrier Type	Pkg. Pin Count	Devices Per Carrier	Carriers per Box[1]	Devices per Box
TapePak	Flat Tubes	36 mm ring	12	10	1202
		44 mm ring	10	12	1202
	Coin Tubes	36 mm ring	60	3	180
		46 mm ring			
		48 pin			
		80 pin	66		330
		100 pin			
Trimmed and Formed	Trays	100 pin	55		275
		120 pin	24	63	120
		132 pin	36		180
		144 pin			
		168 pin	24		120
		216 pin			

Notes: 1. In all cases, PQFPs are dry packed and sealed in a dry pack bag.

2. These represent 2K box quantities. There is double this quantity in a 4K box.

3. The top tray is empty, serving as a cover.

Improved Performance

TapePak enables very high-density ICs to be packed smaller; one third the size of a comparable plastic chip carrier and one tenth of a molded DIP. This allows board sizing to be significantly reduced, typically 50% to 80% over standard PLCC and DIP layouts. Additionally, electrical performance improves because the shorter leads to the die reduce lead capacitance, inductance, and propagation delays. Thermal characteristics are also enhanced through the use of highly conductive, copper leadframe material.

DRY PACK PROTECTION

When high pin-count PQFPs are soldered directly to a board using conventional methods, such as vapor phase of infrared reflow, there is a risk of moisture-induced package cracking. These process expose the package body to very rapid rise to high temperatures. As a result, the moisture in the package vaporizes and can crack the package as it escapes.

AMD has conducted extensive studies to show that the risk of moisture-induced package cracking does not exist until the moisture level in the package exceeds 0.10% by weight. Moisture in the package will increase or decrease with respect to the relative humidity (RH) of the environment in which it is present. Dry packing the product protects it from environmental moisture and keeps the package moisture level well within the safety zone.

Dry packing is accomplished by oven baking the product for 24 hours at 125°C to remove all package moisture. The baked packages are sealed under a partial vacuum in a moisture barrier bag with desiccant and a humidity indicator card. The bag interior remains at ≤30% RH for at least a year from the date of the dry pack seal (noted on the dry pack seal label on the bag). After a year, the product only needs to be re-baked *if* the RH level in the bag is >30%, as evidenced by the humidity indicator card. Should re-baking be necessary, a lower temperature oven bake can be done—192 hours at 40°C and ≤5 RH—if parts remain in tubes or low-temp trays. Otherwise, parts must be removed from these containers and put in metal tubes prior to the 24 hour bake at 125°C.

AMD recommends that the parts be soldered onto boards within 72 hours of breaking the dry pack seal, given a typical factory environment of 20° to 30°C and 50% to 70% RH.

TapePak and TAB

In addition to the adoption of the TapePak configuration, AMD is developing tape-automated bonding (TAB) for very high-lead count devices requiring finer pitch I/O connections (0.1 to 0.5 mm). Bond pads on the die are directly connected to copper foil traces on a polyimide-filmed polymer tape. The polyimid film holds the individual traces in place, allowing lead-pitch to be very small. The copper traces are then bonded directly to the leadframe, as in the PQFP, or to a board. This combination of TAB and TapePak provides the highest density plastic package available in the market today.

Standardization

TapePak was registered with the Joint Electronic Device Engineering Council (JEDEC) in 1987, for mechanical standardization. AMD is participating in the JEDEC and Electronic Industries Association of Japan (EIAJ) packaging committees to promote the TapePak configuration as an industry standard.

Availability

AMD is adopting the TapePak configuration for the entire family of fine pitch (0.8 mm or less) PQFPs. AMD offers both metric and English dimensioned PQFP packages from 48 to 216 leads. As TAB is implemented, higher density PQFPs will be introduced in TapePak.

Additional information on TapePak and TAB can be obtained through your local AMD sales representative.

PQFP Area Sizes in TapePak

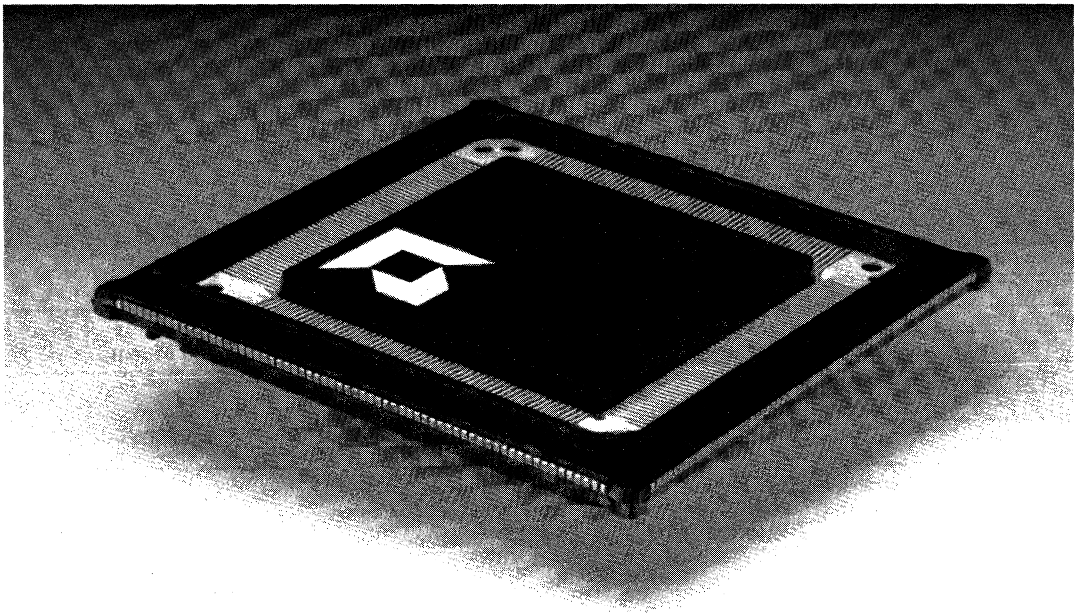
	Area Size		Package Lead Pitch	
Pin Count	**Body Package**	**TapePak Ring**	**Inside Ring**	**Outside Ring**
48 Pin	14 × 20 mm	36 × 36 mm	0.8 mm	0.65 mm
80 Pin			0.8 mm	
100 Pin			0.65 mm	
100 Pin[2]	750 × 750 mil	46 × 46 mm	25 mil	0.65 mm
120 Pin	28 × 28 mm		0.8 mm	
132 Pin[2]	950 × 950 mil		25 mil	
168 Pin[3]	28 × 28 mm		0.65 mm	
216 Pin[3]			0.5 mm	

Notes: 1. These PQFP packages are JEDEC metric version unless otherwise noted.

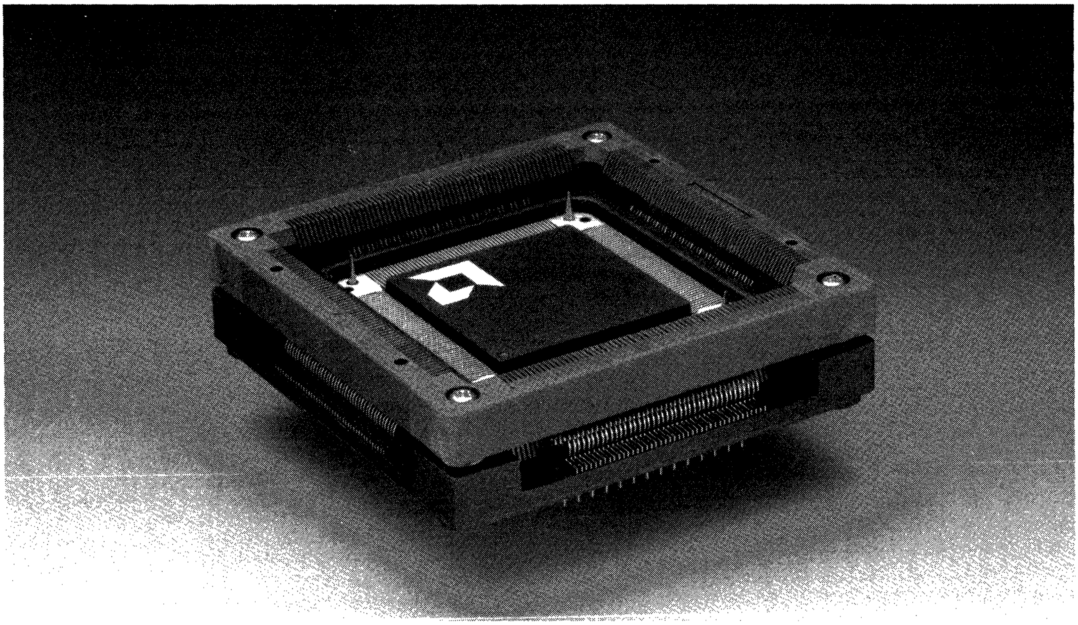
2. This PQFP package is the JEDEC English package version, which has corner bumpers.

3. These PQFP packages are not yet JEDEC registered.

PHOTOS

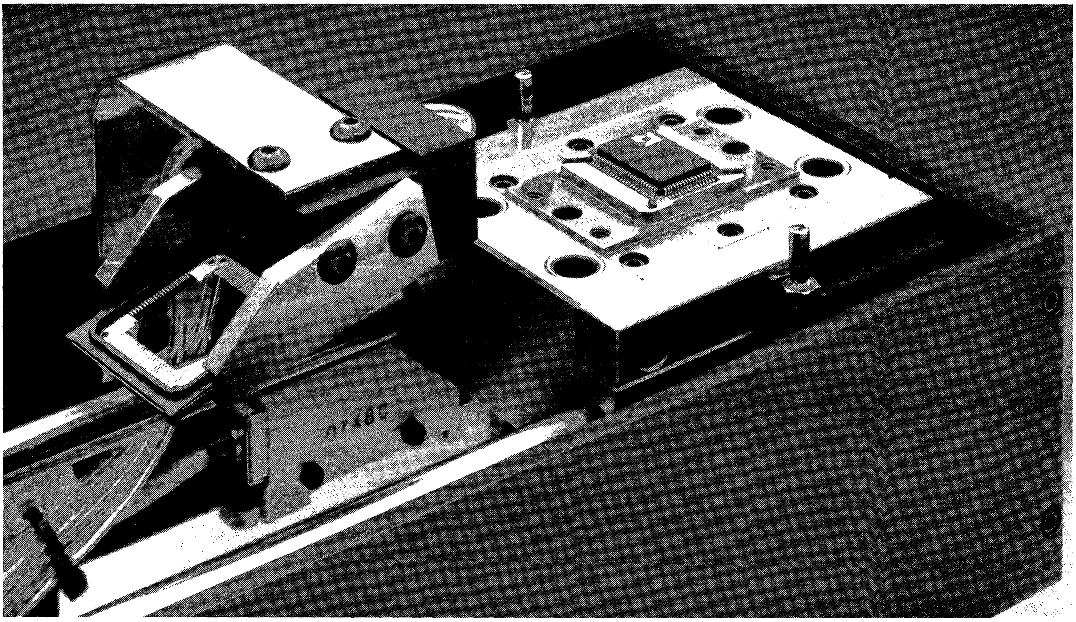


PQFP in the TapePak Format

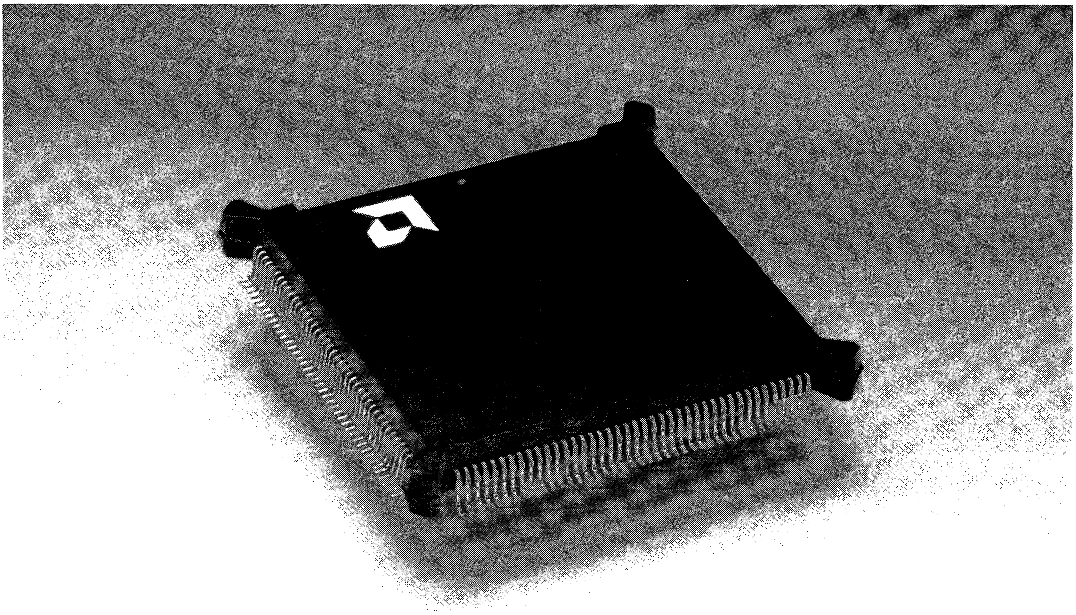


The TapePak lead extensions are used for contact points in test and burn-in socket tooling, virtually eliminating lead coplanarity problems.

PHOTOS (continued)

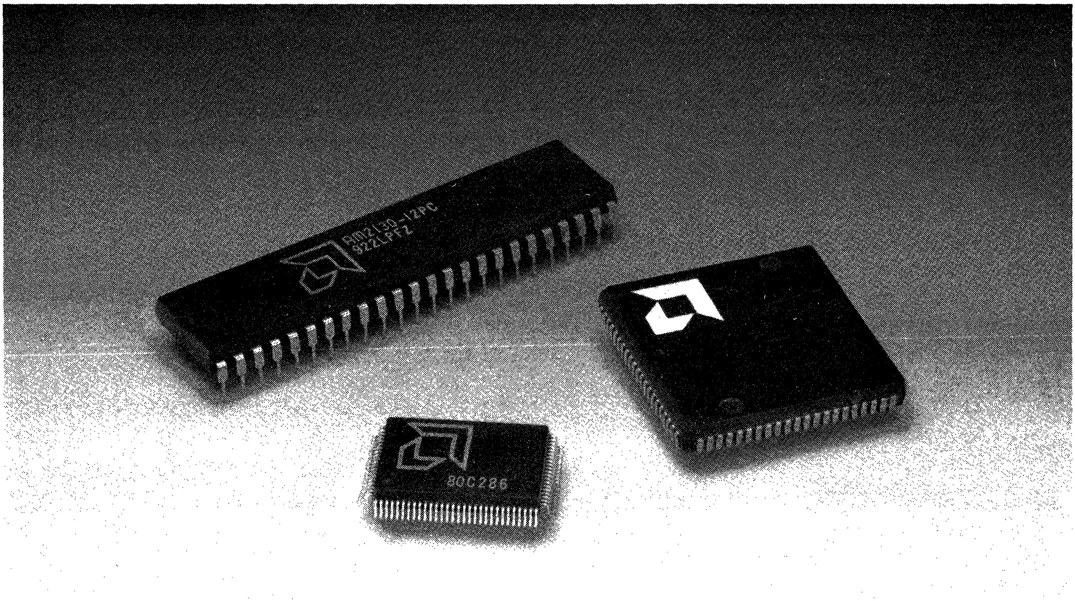


TapePak is compatible with existing pick-and-place machines with the simple addition of a trim-and-form station to excise the package and form the leads.



Am386 microprocessor in a 132-pin PQFP package.

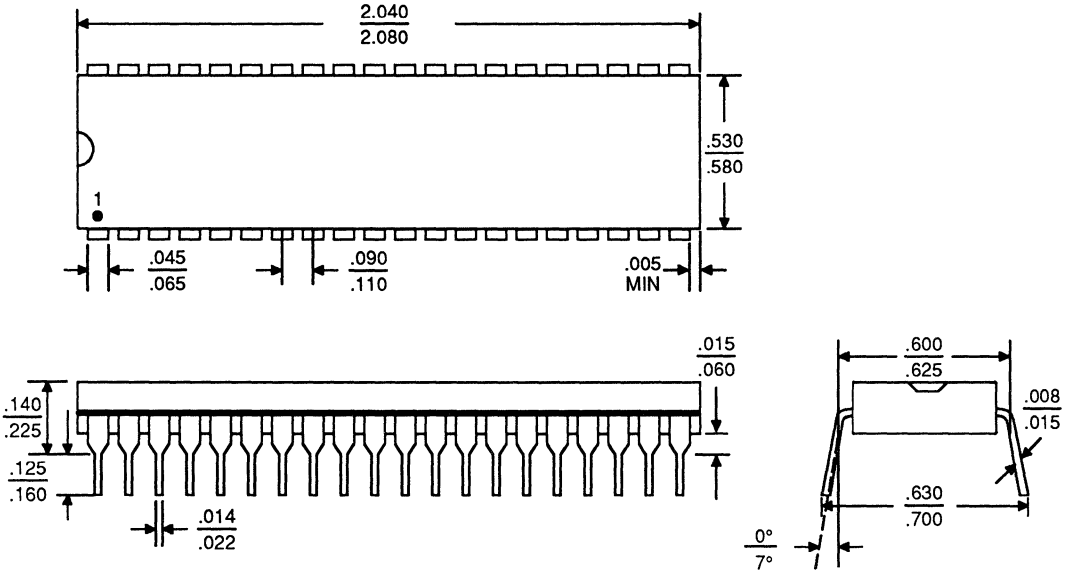
PHOTOS (continued)



TapePak enables very high-density ICs to be packaged smaller. (To left, clockwise: 48-pin plastic DIP, 84-pin PLCC, 80-pin PQFP.)

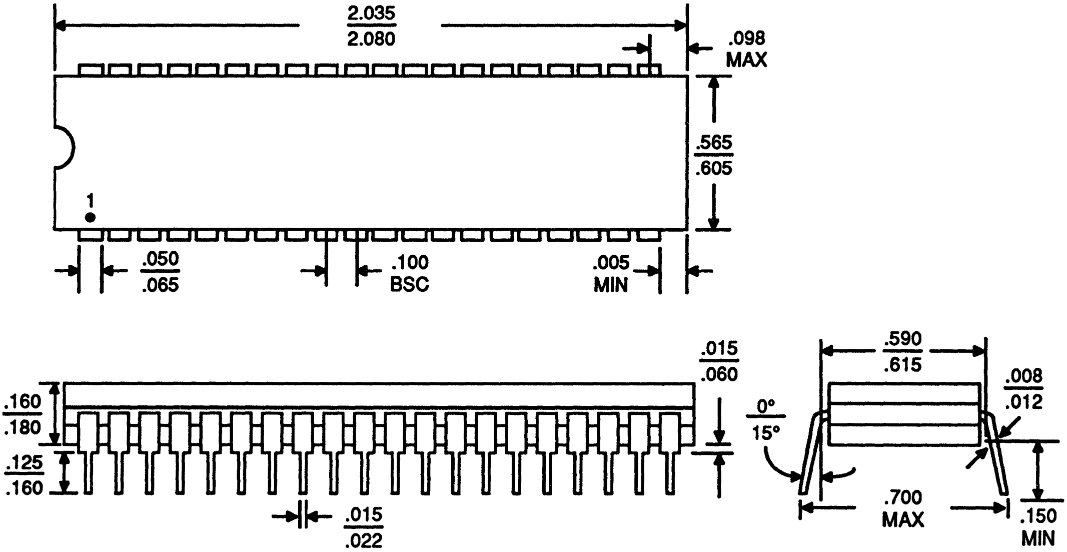
PHYSICAL DIMENSIONS

PD 040



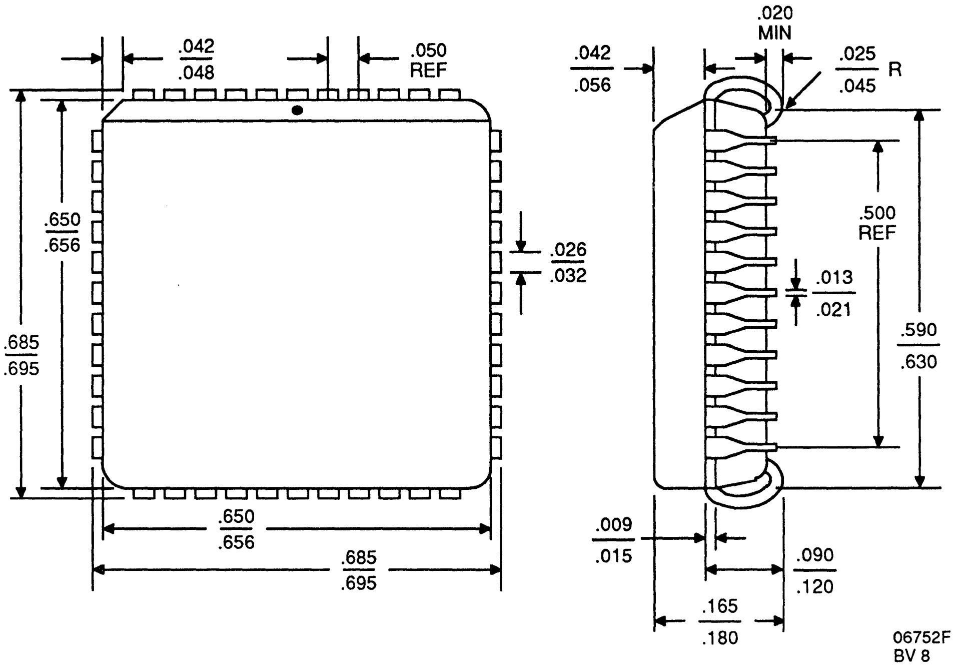
06823D
BC 6
3/29/91 CD

CD 040



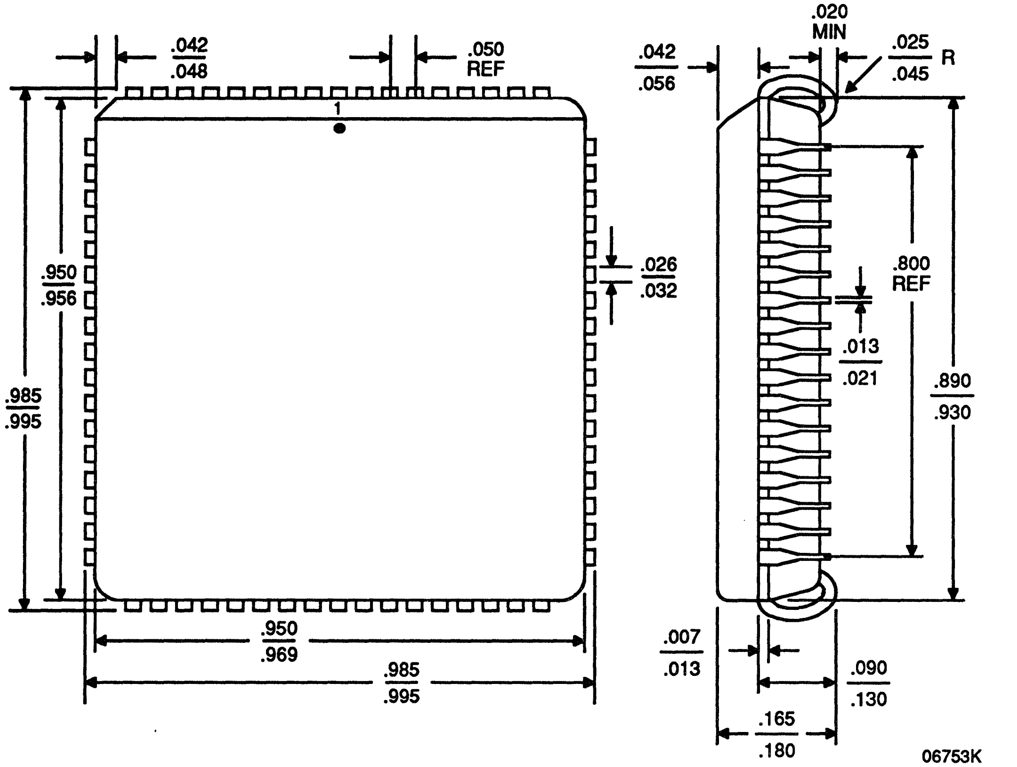
06824E
BK 13
10/25/91 c dc

PL 044



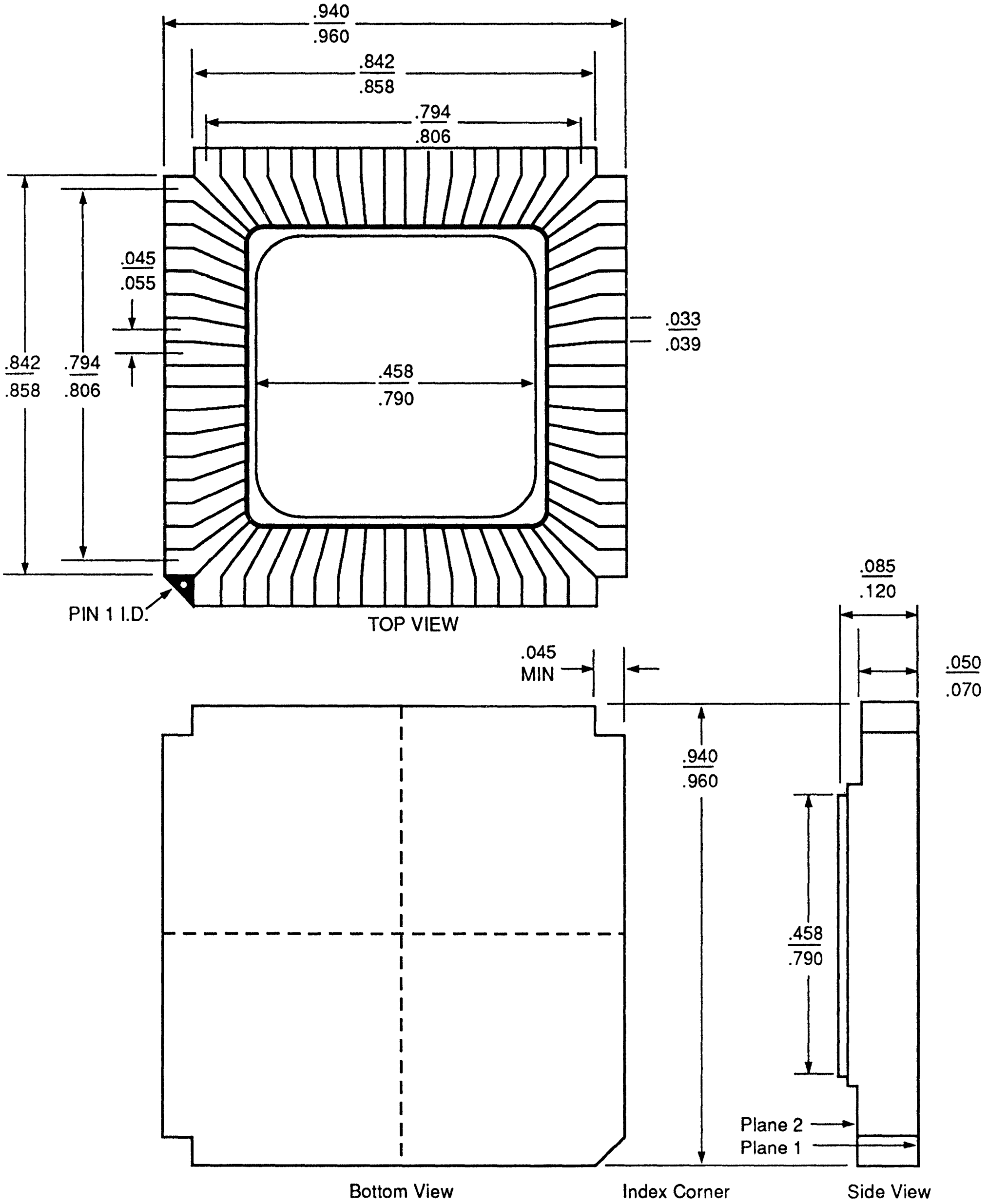
06752F
BV 8
6/9/91 c dc

PL 068



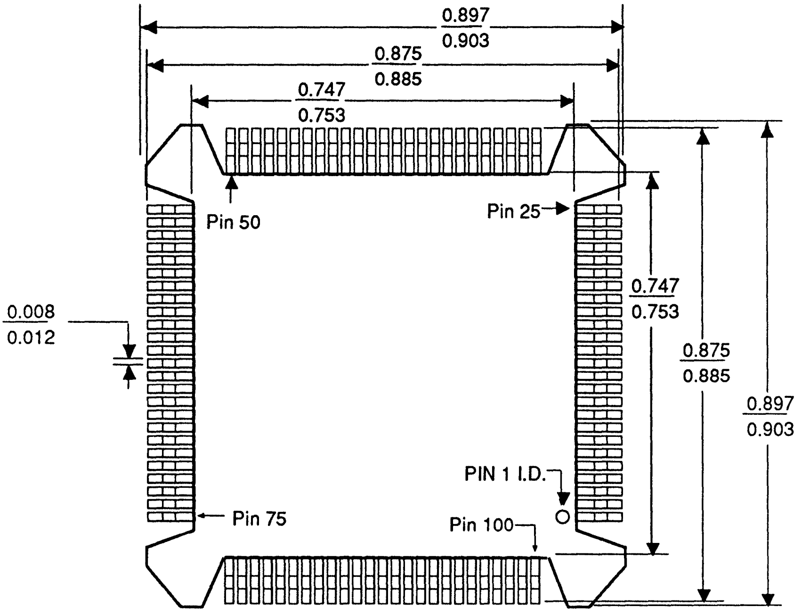
06753K
AN 7
12/5/89 CD

CA 2068

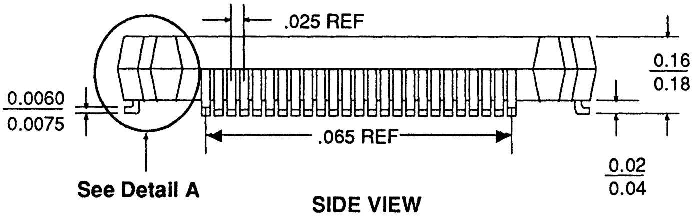


07287C
AN 31
3/89 CD

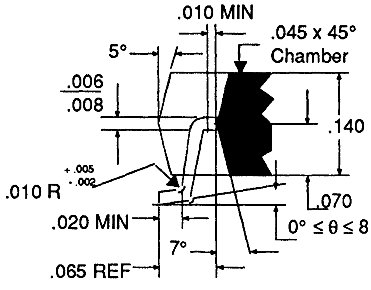
PQ 100
(measured in inches)



TOP VIEW



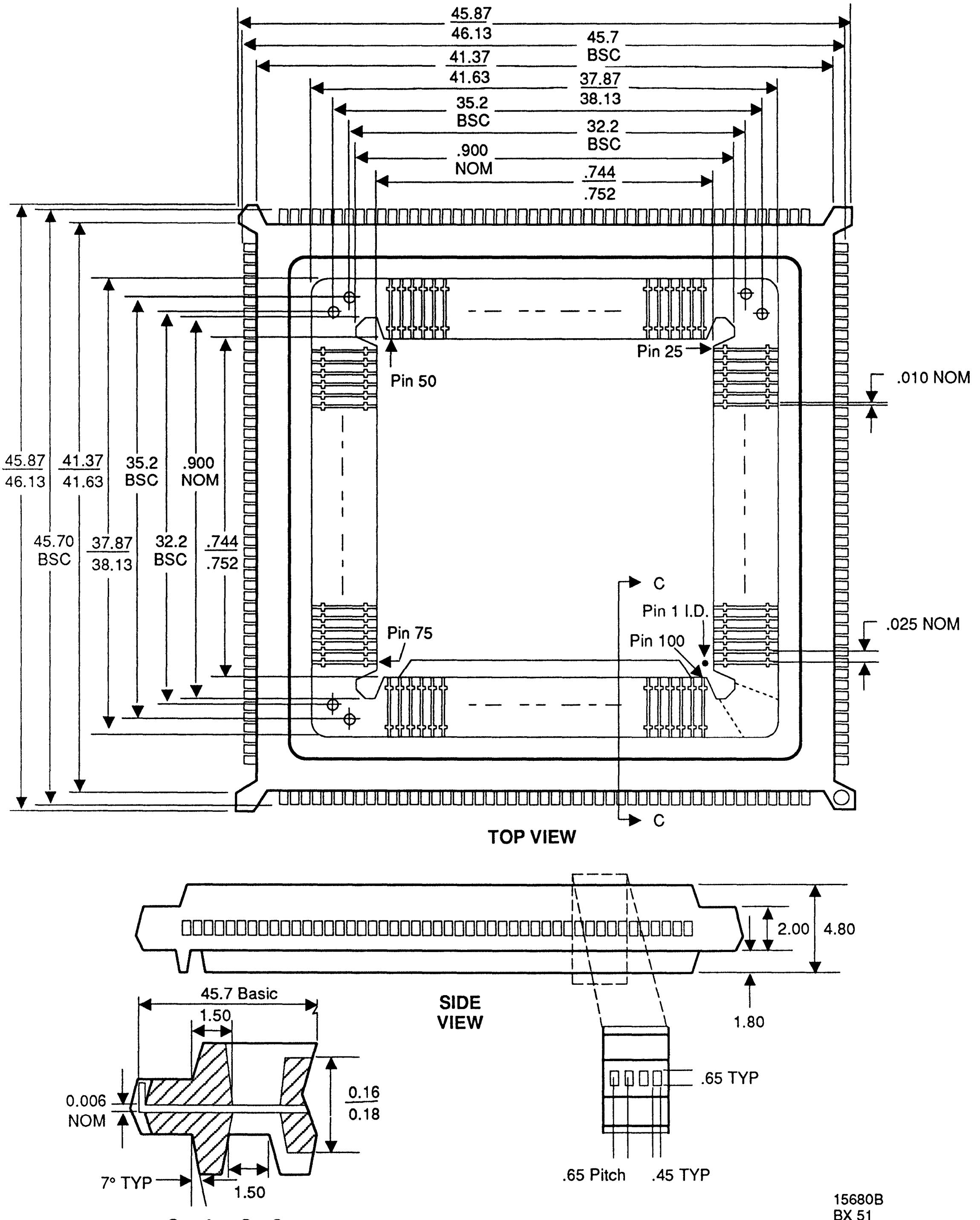
SIDE VIEW



DETAIL A

15679F
BX 43
10/29/91 SG

PQB 100



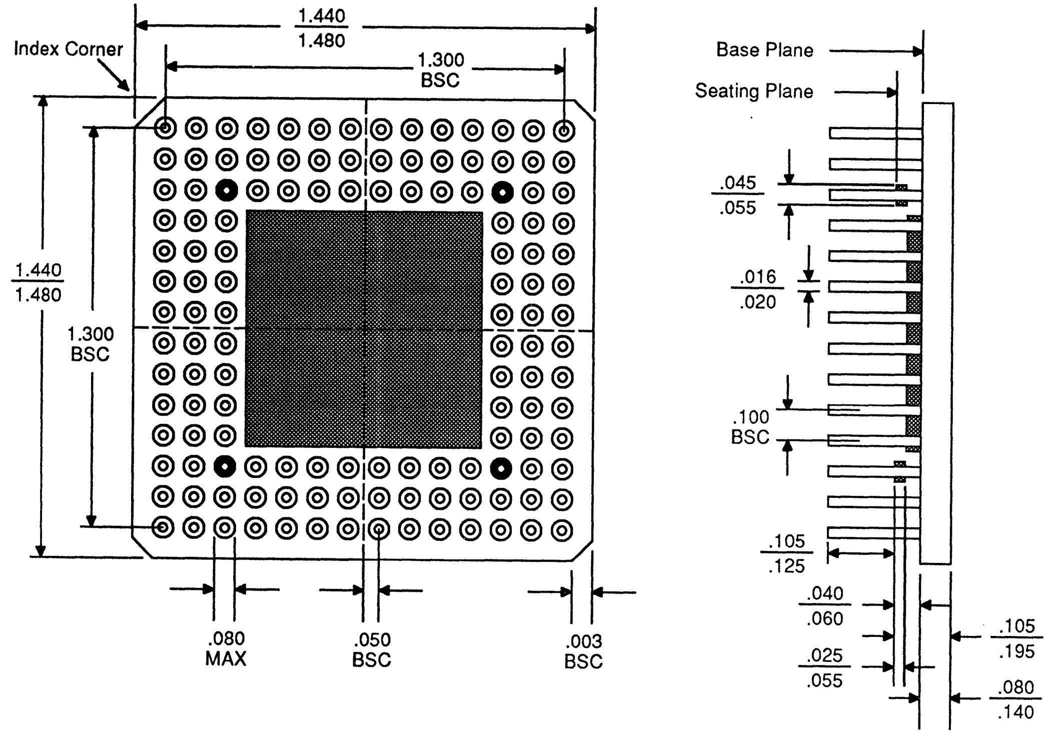
TOP VIEW

SIDE VIEW

Section C↔C

15680B
BX 51
10/29/91 SG

CGX 132

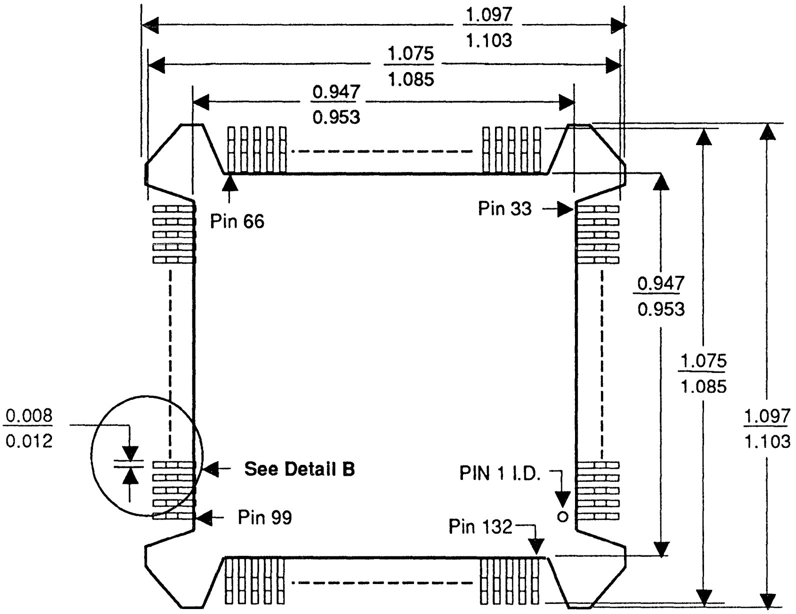


Bottom View (Pins Facing Up)

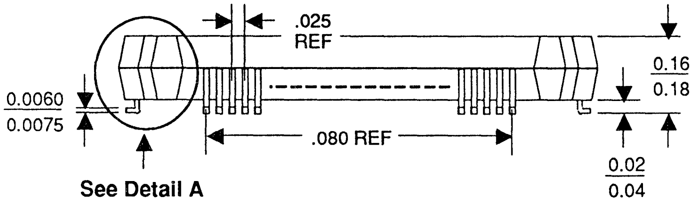
Side View

15304C
BP 36
4/30/91 CD

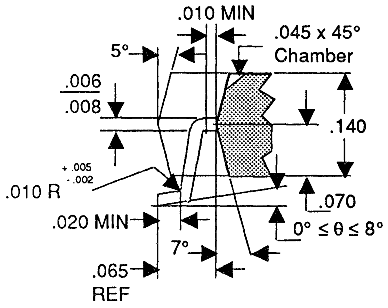
PQ 132
(measured in inches)



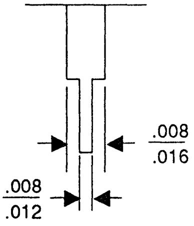
TOP VIEW



SIDE VIEW



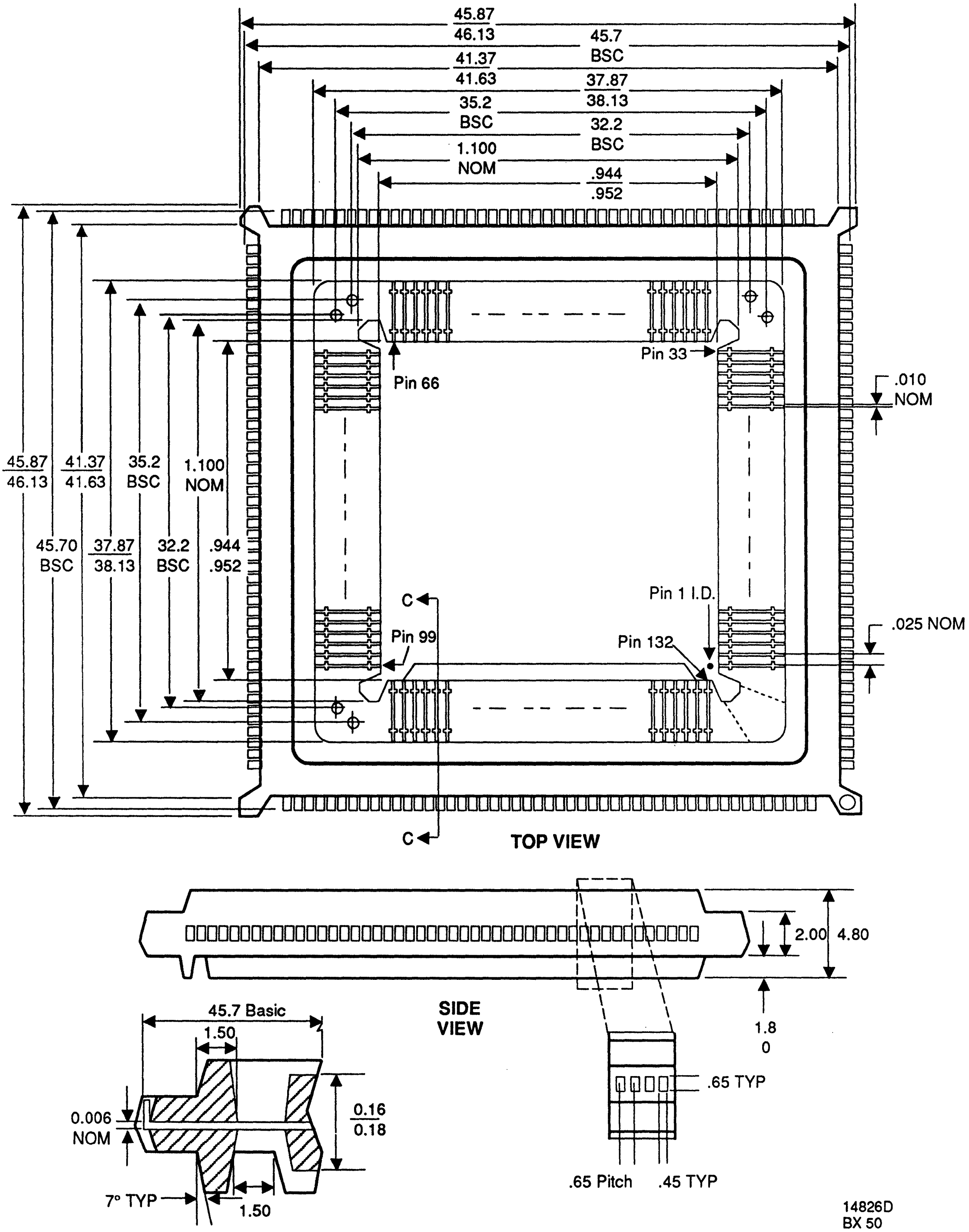
DETAIL A



DETAIL B

11772D
BX 43
10/29/91 SG

PQB 132 (measured in inches)

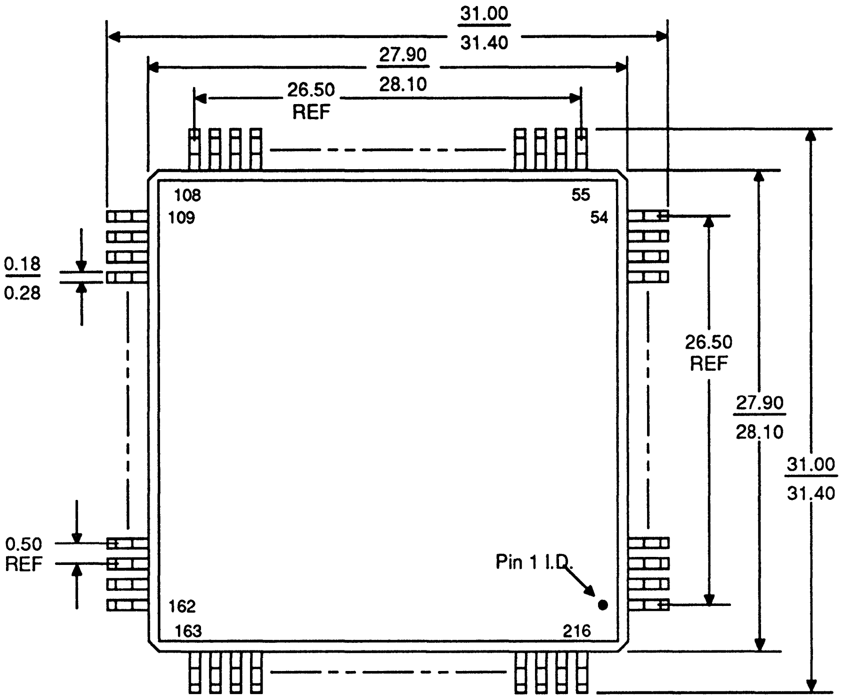


TOP VIEW

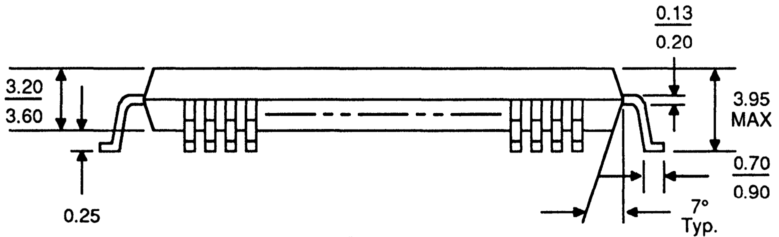
SIDE VIEW

Section C↔C

PQJ 216
(measured in millimeters)



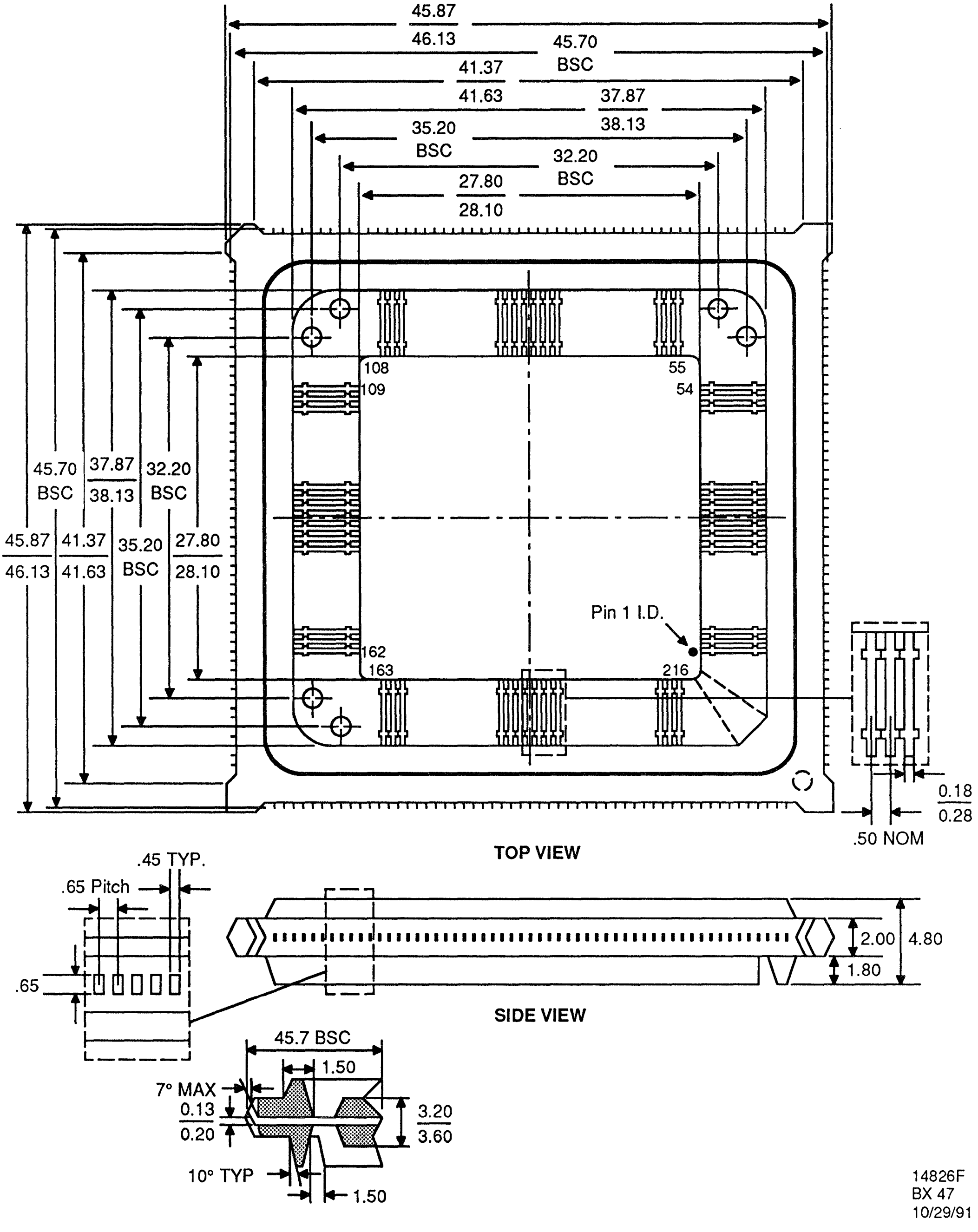
TOP VIEW



SIDE VIEW

14830G
BX 45
10/29/91 SG

PQR 216
(measured in millimeters)



14826F
BX 47
10/29/91 SG

SUPPORT LITERATURE

Am386 Family Literature

Am386 Family—Better 32-Bit Microprocessors (#15946)

OEM-targeted brochure detailing all the Am386 Family features and benefits. Includes application benchmark comparisons of performance and battery life between the Am386 and Intel 386 processors and a complete AMD versus Intel feature comparison chart.

Am386 Family—Choosing the Best 386-Based Computer (#16269)

An end-user-targeted brochure containing an overview of the Am386 Family with performance and battery life benchmark comparisons to the Intel 386 processors.

40 MHz Am386DX Microprocessor Performance Summary (#16219)

Shows the performance comparison of a 40 MHz 386 PC, 33 MHz PC, and a 20 MHz 486SX PC using both application benchmarks and industry standard DOS benchmark software. Window applications include PageMaker 4.0, Excel 3.0, and Corel Draw 2.0. DOS benchmarks include Power Meter MIPs, Dhrystones, Norton SI, BYTE, PC Magazine, and PC Week tests.

25 MHz Am386SX Microprocessor Performance Summary (#16411)

Shows the performance comparison of two 25 MHz 386 PCs using the AMD Am386SX-25 microprocessor and two 20 MHz 386SX PCs using the Intel i386SX-20. Both application benchmarks and industry standard DOS benchmarks are used.

Am386 Microprocessor Family Demo Diskette (#16274)

Graphically shows the benefits of value-added Am386 microprocessors. Emphasizes the upward growth of the 386 market, high performance and low power of the AMD Am386 Family, and complete Intel compatibility.

40-MHz Am386 Microprocessor Design Kit (#15689)

A turnkey development kit speeds to market and reduces design time of your 40 MHz Am386DX/DXL microprocessor-based computers. The kit contains complete design documentation from AMD and chip-set manufacturers to phase the 40-MHz PC into production.

Am386DXL Notebook Design Kit (#15911)

A low-power notebook design demonstrating advanced power-management techniques for maximizing battery life. This 32-bit evaluation kit is a true 32-bit Am386DXL microprocessor motherboard.

Am386SXL Notebook Design Kit (#15910)

A small-form-factor, medium-performance, non-cache, Am386SXL microprocessor-based motherboard design. This low-power design demonstrates advanced power-management techniques maximizing PC battery life.

"486SX vs. Am386-40—The 486 Falls Short" (#15964)

Am386 Family article reprint from BYTE Magazine, June 1991.

"Having the Marketing Gurus Gone to Far?" (#15965)

Am386 Family article reprint from Microprocessor Report.

80286/AMD 80C287 Literature

80C286/AMD 80C287 Programmers Reference Manual (#14554)

Includes an introduction to the 80C286 and AMD 80C287 math coprocessor features, operations, and architectures, including detailed information on memory segmentation, registers, addressing modes, and the 80C286 instruction set. This manual is aimed at system programmers and focuses on the advanced architectural features of the 80C286 when in protected mode.

AMD 80C287 Demonstration/Games Disk (1#4709)

A PC/AT diskette to aid evaluation of the AMD 80C287 math coprocessor. The disk contains benchmark performance data, fractals, and games.

Fusion286 Catalog (#13052)

Provides information on AMD and third party support tools that speed your 80286 microprocessor-based product to market. Includes products from over 20 expert suppliers of personal computer solutions, including hardware/software development tools, system support products, and manufacturing support tools.

AMD 80C287 Coprocessor Performance Benchmarks Application Note (#12640)

Benchmark analysis of the AMD 80C287 math coprocessor running PC Magazine floating-point benchmarks, HLBENCH floating-point benchmarks, and Whetstones benchmark.

Battery Life Benefits of the AMD 80EC287 Math Coprocessor Application Note (#12911)

Battery life testing benchmark comparison of the AMD 80EC287 math coprocessor vs the Intel 80C287A using the Zenith Supersport/286 laptop computer.

Am286ZX/LX Literature

Am286ZX and Am286LX Integrated Microprocessor Brochure (#15382)

An overview of the Am286ZX integrated microprocessor. Includes a description of AMD's Mercury/286ZLX demonstration board.

Am286ZX/LX Technical Manual (#15620)

Includes an introduction to the Am286ZX integrated microprocessor's features, operations, and architecture. It also includes standard peripheral, system control logic, and AT support logic details as well as bus, BIOS, and system configurations.

Mercury/286ZLX Development Kit (#15343)

A turnkey development kit to reduce design time and make Am286LX integrated microprocessor product evaluation and software development easier. The kit contains complete design documentation to help you access methodology and performance, and to phase the Am286LX microprocessor into production.

"Destination Laptop" (#15775)

Am286ZX article reporting from BTYE Magazine, February 1991.

"AMD 286ZX Combines 286 and PC System Logic (#15391)

Am286ZX article reprint from Microprocessor Report, October 3, 1990.

"The One-Chip PC Makes Building Systems a Snap" (#15385)

Am286ZX article reprint from Electronic Design, September 27, 1990.

Sales Offices

North American

ALABAMA	(205) 882-9122
ARIZONA	(602) 242-4400
CALIFORNIA,	
Culver City	(213) 645-1524
Newport Beach	(714) 752-6262
Sacramento(Roseville)	(916) 786-6700
San Diego	(619) 560-7030
San Jose	(408) 452-0500
Woodland Hills	(818) 992-4155
CANADA, Ontario,	
Kanata	(613) 592-0060
Willowdale	(416) 224-5193
COLORADO	(303) 741-2900
CONNECTICUT	(203) 264-7800
FLORIDA,	
Clearwater	(813) 530-9971
Ft. Lauderdale	(305) 776-2001
Orlando (Longwood)	(407) 862-9292
GEORGIA	(404) 449-7920
IDAHO	(208) 377-0393
ILLINOIS,	
Chicago (Itasca)	(708) 773-4422
Naperville	(708) 505-9517
MARYLAND	(301) 381-3790
MASSACHUSETTS	(617) 273-3970
MINNESOTA	(612) 938-0001
NEW JERSEY,	
Cherry Hill	(609) 662-2900
Parsippany	(201) 299-0002
NEW YORK,	
Liverpool	(315) 457-5400
Brewster	(914) 279-8323
Rochester	(716) 425-8050
NORTH CAROLINA	
Harrisburg	(704) 455-1010
Raleigh	(919) 878-8111
OHIO,	
Columbus (Westerville)	(614) 891-6455
Dayton	(513) 439-0268
OREGON	(503) 245-0080
PENNSYLVANIA	(215) 398-8006
TEXAS,	
Austin	(512) 346-7830
Dallas	(214) 934-9099
Houston	(713) 376-8084
UTAH	(801) 264-2900

International

BELGIUM,Antwerpen	TEL	(03) 248 43 00
	FAX	(03) 248 46 42
FRANCE, Paris	TEL	(1) 49-75-10-10
	FAX	(1) 49-75-10-13
	TLX	263282F
GERMANY,		
Bad Homburg	TEL	(49) 6172-24061
	FAX	(49) 6172-23195
München	TEL	(089) 4114-0
	FAX	(089) 406490
	TLX	523883
HONG KONG,	TEL	(852) 865-4525
Wanchai	FAX	(852) 865-1147
	TLX	67955AMDAPHX
ITALY, Milano	TEL	(02) 3390541
		(02) 3533241
	FAX	(02) 3498000
	TLX	843-315286
JAPAN,		
Atsugi	TEL	(0462) 29-8460
	FAX	(0462) 29-8458
Kanagawa	TEL	(0462) 47-2911
	FAX	(0462) 47-1729

International (Continued)

Tokyo	TEL	(03) 3346-7550
	FAX	(03) 3342-5196
	TLX	J24064AMDTKOJ
Osaka	TEL	(06) 243-3250
	FAX	(06) 243-3253
KOREA, Seoul	TEL	(82) 2-784-7598
	FAX	(82) 2-784-8014
LATIN AMERICA,		
Ft. Lauderdale	TEL	(305) 484-8600
	FAX	(305) 485-9736
	TLX	5109554261 AMDFTL
NORWAY, Oslo area	TEL	(02) 53 13 24
(Hövik)	FAX	(02) 58 22 62
	TLX	79079
SINGAPORE	TEL	(65) 3481188
	FAX	(65) 3480161
	TLX	55650 AMDMMI
SWEDEN,		
Stockholm area	TEL	(08) 98 61 80
(Bromma)	FAX	(08) 98 09 06
TAIWAN, Taipei	TEL	(886) 2-7213393
	FAX	(886) 2-7723422
UNITED KINGDOM,		
Manchester area	TEL	(0925) 828008
(Warrington)	FAX	(0925) 827693
	TLX	851-628524
London area	TEL	(0483) 740440
(Woking)	FAX	(0483) 756196
	TLX	851-859103

North American Representatives

CANADA	
Burnaby, B.C. - DAVETEK MARKETING	(604) 430-3680
Kanata, Ontario - VITEL ELECTRONICS	(613) 592-0060
Mississauga, Ontario - VITEL ELECTRONICS	(416) 676-9720
Lachine, Quebec - VITEL ELECTRONICS	(514) 636-5951
ILLINOIS	
Skokie -INDUSTRIAL REPRESTATIVES,INC	(708) 967-8430
INDIANA	
Huntington - ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3450
Indianapolis - ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3450
IOWA	
LORENZ SALES	(319) 377-4666
KANSAS	
Merriam – LORENZ SALES	(913) 469-1312
Wichita – LORENZ SALES	(316) 721-0500
KENTUCKY	
ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3452
MICHIGAN	
Birmingham - MIKE RAICK ASSOCIATES	(313) 644-5040
Holland – COM-TEK SALES, INC	(616) 392-7100
Novi – COM-TEK SALES, INC	(313) 227-0007
MINNESOTA	
Mel Foster Tech. Sales, Inc.	(612) 941-9790
MISSOURI	
LORENZ SALES	(314) 997-4558
NEBRASKA	
LORENZ SALES	(402) 475-4660
NEW MEXICO	
THORSON DESERT STATES	(505) 883-4343
NEW YORK	
East Syracuse – NYCOM, INC	(315) 437-8343
Woodbury – COMPONENT CONSULTANTS, INC	(516) 364-8020
OHIO	
Centerville – DOLFUSS ROOT & CO	(513) 433-6776
Columbus – DOLFUSS ROOT & CO	(614) 885-4844
Westlake – DOLFUSS ROOT & CO	(216) 899-9370
OREGON	
ELECTRA TECHNICAL SALES, INC	(503) 643-5074
PENNSYLVANIA	
RUSSELL F. CLARK CO.,INC.	(412) 242-9500
PUERTO RICO	
COMP REP ASSOC, INC	(809) 746-6550
WASHINGTON	
ELECTRA TECHNICAL SALES	(206) 821-7442
WISCONSIN	
Brookfield-INDUSTRIAL REPRESTATIVES,INC	(414) 789-9393

Advanced Micro Devices reserves the right to make changes in its product without notice in order to improve design or performance characteristics. The performance characteristics listed in this document are guaranteed by specific tests, guard banding, design and other practices common to the industry. For specific testing details, contact your local AMD sales representative. The company assumes no responsibility for the use of any circuits described herein.

Advanced Micro Devices, Inc. 901 Thompson Place, P.O. Box 3453, Sunnyvale, CA 94088, USA
Tel: (408) 732-2400 • TWX: 910-339-9280 • TELEX: 34-6306 • TOLL FREE: (800) 538-8450
APPLICATIONS HOTLINE & LITERATURE ORDERING • TOLL FREE: (800) 222-9323 • (408) 749-5703

10/24/91
Printed in USA

**ADVANCED
MICRO
DEVICES, INC.**
*901 Thompson Place
P.O. Box 3453
Sunnyvale
California 94088-3453
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL-FREE
(800) 538-8450*

***APPLICATIONS HOTLINE &
LITERATURE ORDERING
(800) 222-9323
(408) 749-5703***

RECYCLED &
RECYCLABLE

Printed in USA

CUS-44K-11/91-0

11339B