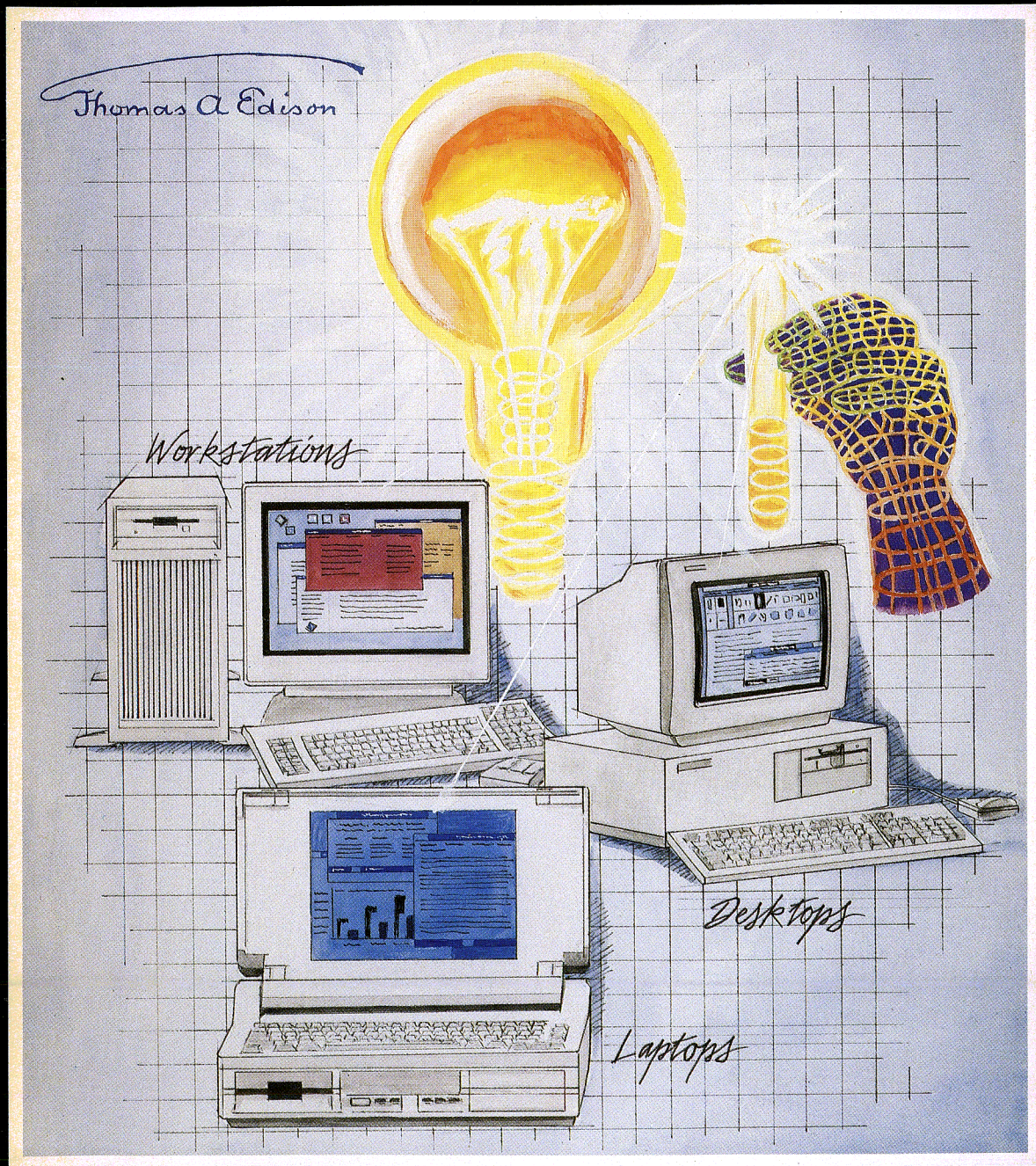




Microprocessors

Volume I





LITERATURE

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

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

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

CURRENT HANDBOOKS

Product line handbooks contain data sheets, application notes, article reprints and other design information. All handbooks can be ordered individually, and most are available in a pre-packaged set in the U.S. and Canada.

TITLE	INTEL ORDER NUMBER	ISBN
SET OF THIRTEEN HANDBOOKS (Available in U.S. and Canada)	231003	N/A
CONTENTS LISTED BELOW FOR INDIVIDUAL ORDERING:		
COMPONENTS QUALITY/RELIABILITY	210997	1-55512-132-2
EMBEDDED APPLICATIONS	270648	1-55512-123-3
8-BIT EMBEDDED CONTROLLERS	270645	1-55512-121-7
16-BIT EMBEDDED CONTROLLERS	270646	1-55512-120-9
16/32-BIT EMBEDDED PROCESSORS	270647	1-55512-122-5
MEMORY PRODUCTS	210830	1-55512-117-9
MICROCOMMUNICATIONS	231658	1-55512-119-5
MICROCOMPUTER PRODUCTS	280407	1-55512-118-7
MICROPROCESSORS	230843	1-55512-115-2
PACKAGING	240800	1-55512-128-4
PERIPHERAL COMPONENTS	296467	1-55512-127-6
PRODUCT GUIDE (Overview of Intel's complete product lines)	210846	1-55512-116-0
PROGRAMMABLE LOGIC	296083	1-55512-124-1

ADDITIONAL LITERATURE:

(Not included in handbook set)

AUTOMOTIVE HANDBOOK	231792	1-55512-125-x
INTERNATIONAL LITERATURE GUIDE (Available in Europe only)	E00029	N/A
CUSTOMER LITERATURE GUIDE	210620	N/A
MILITARY HANDBOOK (2 volume set)	210461	1-55512-126-8
SYSTEMS QUALITY/RELIABILITY	231762	1-55512-046-6



U.S. and CANADA LITERATURE ORDER FORM

NAME: _____
COMPANY: _____
ADDRESS: _____
CITY: _____ STATE: _____ ZIP: _____
COUNTRY: _____
PHONE NO.: () _____

ORDER NO.	TITLE	QTY.	PRICE	TOTAL
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____
<input type="text"/>	_____	_____ x	_____ =	_____

Subtotal _____

Must Add Your
Local Sales Tax _____

Include postage:
Must add 15% of Subtotal to cover U.S.
and Canada postage. (20% all other.)

Postage _____

Total _____

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

☐ VISA ☐ MasterCard ☐ American Express Expiration Date _____

Account No. _____

Signature _____

Mail To: Intel Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

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

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

Prices good until 12/31/91.
Source HB



INTERNATIONAL LITERATURE ORDER FORM

NAME: _____

COMPANY: _____

ADDRESS: _____

CITY: _____ STATE: _____ ZIP: _____

COUNTRY: _____

PHONE NO.: () _____

ORDER NO.	TITLE	QTY.	PRICE	TOTAL
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____
<input type="text"/>	_____	_____	x _____	= _____

Subtotal _____

Must Add Your
Local Sales Tax _____

Total _____

PAYMENT

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

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

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



Intel Corporation is a leading supplier of microcomputer components, modules and systems. When Intel invented the microprocessor in 1971, it created the era of the microcomputer. Today, Intel architectures are considered world standards. Whether used in embedded applications such as automobiles, printers and microwave ovens, or as the CPU in personal computers, client servers or supercomputers, Intel delivers leading-edge technology.

MICROPROCESSORS

VOLUME I

1991

About Our Cover:

Thinkers, inventors, and artists throughout history have breathed life into their ideas by converting them into rough working sketches, models, and products. This series of covers shows a few of these creations, along with the applications and products created by Intel customers.

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

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

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

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

287, 376, 386, 387, 486, 4-SITE, Above, ACE51, ACE96, ACE186, ACE196, ACE960, ActionMedia, BITBUS, COMMputer, CREDIT, Data Pipeline, DVI, ETOX, FaxBACK, Genius, i, i⁺, i486, i750, i860, ICE, iCEL, ICEVIEW, iCS, iDBP, iDIS, i²ICE, iLBX, iMDDX, iMMX, Inboard, Insite, Intel, intel, Intel386, intelIBOS, Intel Certified, Inteleview, intelligent Identifier, intelligent Programming, Intellec, Intellink, iOSP, iPAT, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, Library Manager, MAPNET, MCS, Megachassis, MICROMAINFRAME, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, Pro750, PROMPT, Promware, QUEST, QueX, Quick-Erase, Quick-Pulse Programming, READY LAN, RMX/80, RUPI, Seamless, SLD, SugarCube, SX, ToolTALK, UPI, VAPI, Visual Edge, VLSiCEL, and ZapCode, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix.

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

CHMOS and HMOS are patented processes of Intel Corp.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

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

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

CUSTOMER SUPPORT

INTEL'S COMPLETE SUPPORT SOLUTION WORLDWIDE

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

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

HARDWARE SUPPORT SERVICES

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

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

SOFTWARE SUPPORT SERVICES

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

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

NETWORK SERVICE AND SUPPORT

Today's broad spectrum of powerful networking capabilities are only as good as the customer support provided by the vendor. Intel offers network services and support structured to meet a wide variety of end-user computing needs. From a ground up design of your network's physical and logical design to implementation, installation and network wide maintenance. From software products to turn-key system solutions; Intel offers the customer a complete networked solution. With over 10 years of network experience in both the commercial and Government arena; network products, services and support from Intel provide you the most optimized network offering in the industry.

CONSULTING SERVICES

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

CUSTOMER TRAINING

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

DATA SHEET DESIGNATIONS

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

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

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



Overview

1

8086 Microprocessor Family

2

80286 Microprocessor Family

3

**Development Tools for the
8086, 80186, 80188, and
80286**

4

Intel386™ Family

5

i860™ Microprocessor Family

6

i750™ Video Processor Family

7

**Development Tools for the
80386 and 80486**

8

Table of Contents

Alphanumeric Index	xi
CHAPTER 1	
Overview	
Introduction	1-1
CHAPTER 2	
8086 Microprocessor Family	
DATA SHEETS	
8086 16-Bit HMOS Microprocessor 8086/8086-2/8086-1*	2-1
80C86A 16-Bit CHMOS Microprocessor	2-31
8088 8-Bit HMOS Microprocessor 8088/8088-2	2-60
80C88A 8-Bit CHMOS Microprocessor	2-90
8087 Math Coprocessor	2-122
CHAPTER 3	
80286 Microprocessor Family	
DATA SHEETS	
80C286 High Performance Microprocessor with Memory Management and Protection	3-1
80286 High Performance Microprocessor with Memory Management and Protection	3-60
80287XL/XLT CHMOS III Math CoProcessor	3-116
82C288 Bus Controller for 80286 Processors (82C288-12, 82C288-10, 82C288-8)	3-148
82C284 Clock Generator and Ready Interface for 80286 Processors (82C284-12, 82C284-10, 82C284-8)	3-169
CHAPTER 4	
Development Tools for the 8086, 80186, 80188, and 80286	
LANGUAGES AND SOFTWARE DEVELOPMENT TOOLS	
8086/80186 Software Development Packages	4-1
iC-86/286 C Compiler	4-8
AEDIT Source Code and Text Editor	4-12
iPAT Performance Analysis Tool	4-14
IN-CIRCUIT EMULATORS	
I2ICE In-Circuit Emulation System	4-18
ICE-186 and ICE-188 In-Circuit Emulators	4-22
ICE-186EB and ICE-188EB In-Circuit Emulators	4-25
ICE-286 In-Circuit Emulator	4-32
CHAPTER 5	
INTEL386™ Family	
DATA SHEETS	
i486 Microprocessor	5-1
485Turbocache Module i486 Microprocessor Cache Upgrade	5-177
82485 Second Level Cache Controller for the i486 Microprocessor	5-206
AP-447 A Memory Subsystem for the i486 CPU Including Second Level Cache	5-207
386 DX Microprocessor High Performance 32-Bit CHMOS Microprocessor with Integrated Memory Management	5-287
387 DX Math Coprocessor	5-425
82395DX High Performance 386 Smart Cache	5-466
82385 High Performance 32-Bit Cache Controller	5-547
AP-442 33 MHz 386 System Design Considerations	5-620
386 SL Microprocessor SuperSet	5-731
386 SX Microprocessor	5-864
387 SX Math Coprocessor	5-962

Table of Contents (Continued)

82395SX 386 SX Smart Cache	5-1002
82385SX High Performance Cache Controller	5-1003
82380 High Performance 32-Bit DMA Controller with Integrated System Support Peripherals	5-1080
376 High Performance 32-Bit Embedded Processor	5-1217
82370 Integrated System Peripheral	5-1312
CHAPTER 6	
i860™ Microprocessor Family	
i860 64-Bit Microprocessor	6-1
AP-434 Using i860 Microprocessor Graphics Instructions for 3-D Rendering	6-81
AP-435 Fast Fourier Transforms on the i860 Microprocessor	6-96
CHAPTER 7	
i750™ Video Processor Family	
82750PB Pixel Processor	7-1
82750DB Display Processor	7-3
CHAPTER 8	
Development Tools for the 80386 and 80486	
LANGUAGES AND SOFTWARE DEVELOPMENT TOOLS	
Intel386/i486 Family Development Support	8-1
Intel 376 Family Development Support	8-11
ICD-486 In-Circuit Debugger	8-23
IN-CIRCUIT EMULATORS	
Intel386 Family of In-Circuit Emulators	8-29
Intel i486 In-Circuit Emulator	8-55

Alphanumeric Index

376 High Performance 32-Bit Embedded Processor	5-1217
386 DX Microprocessor High Performance 32-Bit CHMOS Microprocessor with Integrated Memory Management	5-287
386 SL Microprocessor SuperSet	5-731
386 SX Microprocessor	5-864
387 DX Math Coprocessor	5-425
387 SX Math Coprocessor	5-962
485TurboCache Module i486 Microprocessor Cache Upgrade	5-177
80286 High Performance Microprocessor with Memory Management and Protection	3-60
80287XL/XLT CHMOS III Math CoProcessor	3-116
8086 16-Bit HMOS Microprocessor 8086/8086-2/8086-1*	2-1
8086/80186 Software Development Packages	4-1
8087 Math Coprocessor	2-122
8088 8-Bit HMOS Microprocessor 8088/8088-2	2-60
80C286 High Performance Microprocessor with Memory Management and Protection	3-1
80C86A 16-Bit CHMOS Microprocessor	2-31
80C88A 8-Bit CHMOS Microprocessor	2-90
82370 Integrated System Peripheral	5-1312
82380 High Performance 32-Bit DMA Controller with Integrated System Support Peripherals	5-1080
82385 High Performance 32-Bit Cache Controller	5-547
82385SX High Performance Cache Controller	5-1003
82395DX High Performance 386 Smart Cache	5-466
82395SX 386 SX Smart Cache	5-1002
82485 Second Level Cache Controller for the i486 Microprocessor	5-206
82750DB Display Processor	7-3
82750PB Pixel Processor	7-1
82C284 Clock Generator and Ready Interface for 80286 Processors (82C284-12, 82C284-10, 82C284-8)	3-169
82C288 Bus Controller for 80286 Processors (82C288-12, 82C288-10, 82C288-8)	3-148
AEDIT Source Code and Text Editor	4-12
AP-434 Using i860 Microprocessor Graphics Instructions for 3-D Rendering	6-81
AP-435 Fast Fourier Transforms on the i860 Microprocessor	6-96
AP-442 33 MHz 386 System Design Considerations	5-620
AP-447 A Memory Subsystem for the i486 CPU Including Second Level Cache	5-207
I2ICE In-Circuit Emulation System	4-18
i486 Microprocessor	5-1
i860 64-Bit Microprocessor	6-1
iC-86/286 C Compiler	4-8
ICD-486 In-Circuit Debugger	8-23
ICE-186 and ICE-188 In-Circuit Emulators	4-22
ICE-186EB and ICE-188EB In-Circuit Emulators	4-25
ICE-286 In-Circuit Emulator	4-32
Intel 376 Family Development Support	8-11
Intel386 Family of In-Circuit Emulators	8-29
Intel386/i486 Family Development Support	8-1
Intel i486 In-Circuit Emulator	8-55
iPAT Performance Analysis Tool	4-14

Overview

1

1

INTRODUCTION

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

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

REDUCED TIME TO MARKET

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

INCREASED RELIABILITY

At Intel, the rate of failure for devices is carefully tracked. Highest reliability is a tangible goal that translates to higher reliability for your product, reduced downtime, and reduced

repair costs. And as more and more functions are integrated on a single VLSI device, the resulting system requires less power, produces less heat, and requires fewer mechanical connections — again resulting in greater system reliability.

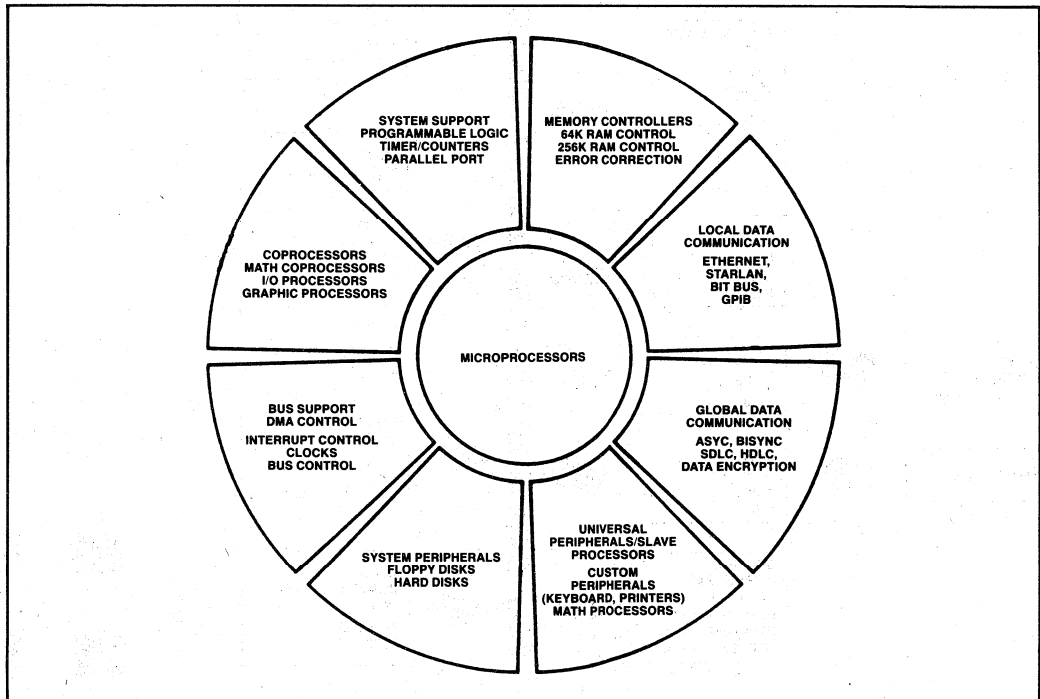
LOWER PRODUCTION COST

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

HIGHER SYSTEM PERFORMANCE

Intel microprocessors and peripherals provide the highest system performance for the demands of today's (and tomorrow's) microprocessor-based applications. For example, the Intel386/i486™ Microprocessor Family offers 32-bit performance for multitasking, multiuser systems. Intel's peripheral products have been designed with the future in mind. They support all of Intel's 8, 16 and 32-bit processors.

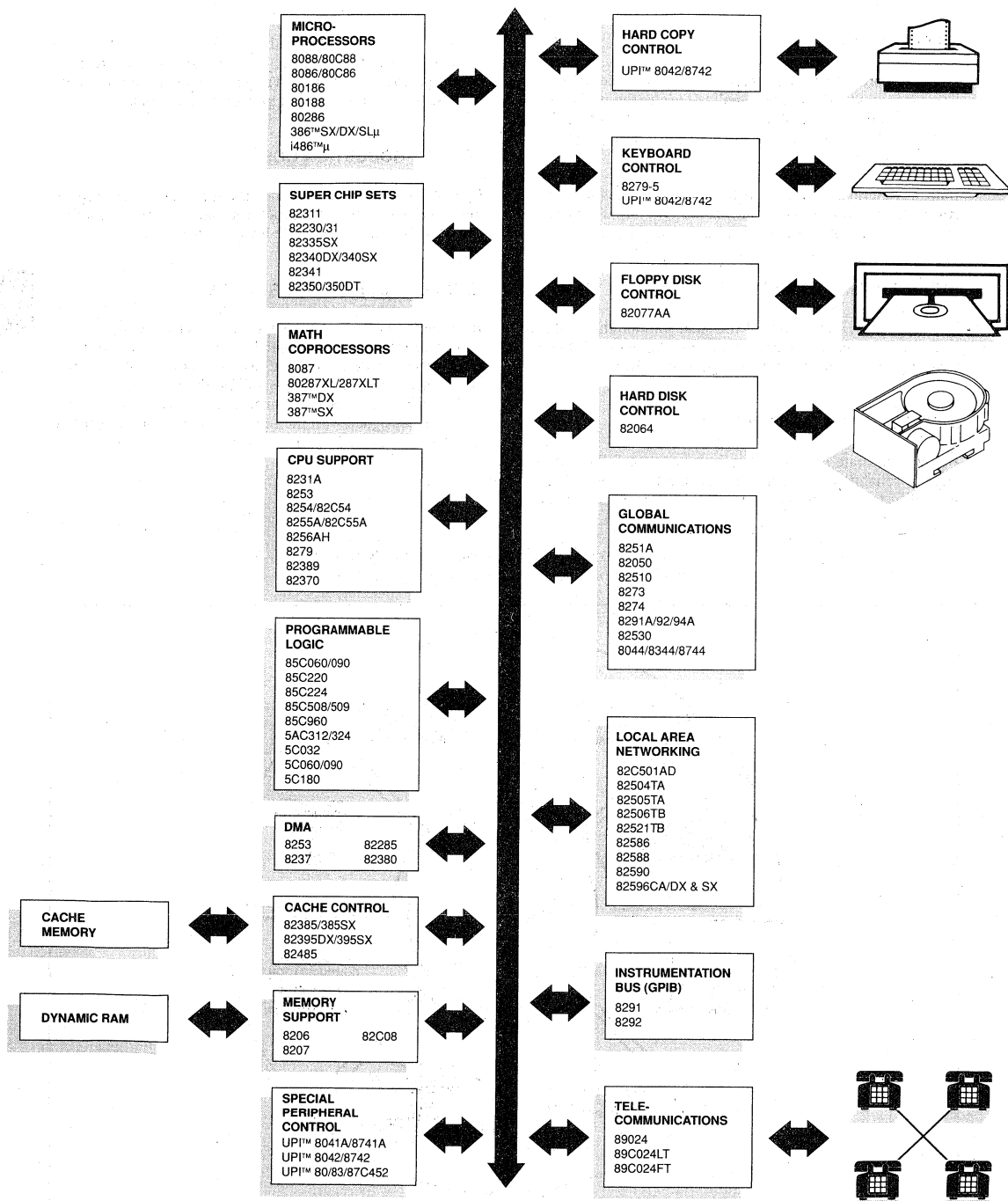
The Intel microprocessor and peripherals family provides a broad range of time-saving, high performance solutions.





Get Your Kit Together!

Intel's Microsystem Components Kit Solution



8086 Microprocessor Family

2



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

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

2

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

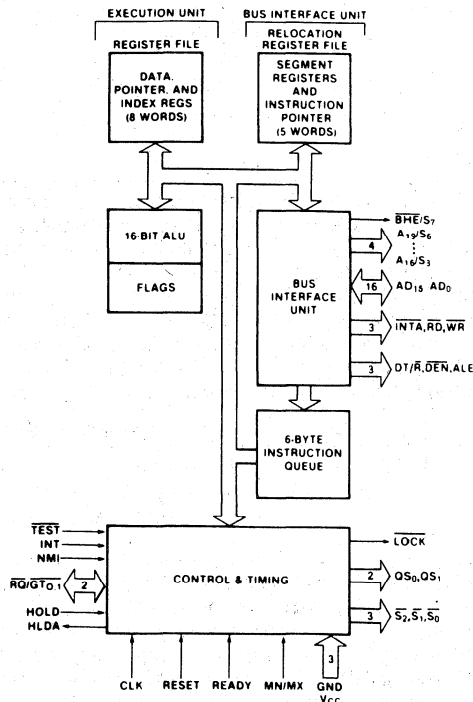
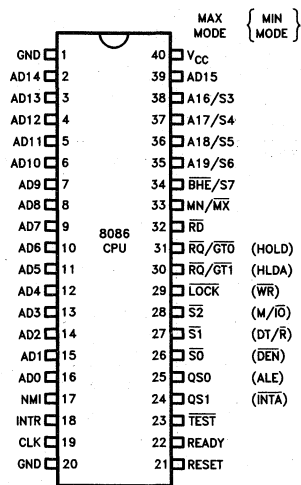


Figure 1. 8086 CPU Block Diagram

231455-1



40 Lead

Figure 2. 8086 Pin Configuration

231455-2

Table 1. Pin Description

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

Symbol	Pin No.	Type	Name and Function		
AD ₁₅ –AD ₀	2–16, 39	I/O	ADDRESS DATA BUS: These lines constitute the time multiplexed memory/I/O address (T ₁), and data (T ₂ , T ₃ , T _W , T ₄) bus. A ₀ is analogous to 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 BHE.) These lines are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge".		
A ₁₉ /S ₆ , A ₁₈ /S ₅ , A ₁₇ /S ₄ , A ₁₆ /S ₃	35–38	O	ADDRESS/STATUS: During T ₁ these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T ₂ , T ₃ , T _W , T ₄ . The status of the interrupt enable FLAG bit (S ₅) is updated at the beginning of each CLK cycle. A ₁₇ /S ₄ and A ₁₆ /S ₃ are encoded as shown. This information indicates which relocation register is presently being used for data accessing. These lines float to 3-state OFF during local bus "hold acknowledge."		
			A ₁₇ /S ₄	A ₁₆ /S ₃	Characteristics
			0 (LOW) 0 1 (HIGH) 1 S ₆ is 0 (LOW)	0 1 0 1	Alternate Data Stack Code or None Data
BHE/S ₇	34	O	BUS HIGH ENABLE/STATUS: During T ₁ the bus high enable signal (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 BHE to condition chip select functions. BHE is LOW during T ₁ for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S ₇ status information is available during T ₂ , T ₃ , and T ₄ . The signal is active LOW, and floats to 3-state OFF in "hold". It is LOW during T ₁ for the first interrupt acknowledge cycle.		
			BHE	A ₀	Characteristics
			0 0 1 1	0 1 0 1	Whole word Upper byte from/to odd address Lower byte from/to even address None
RD	32	O	READ: Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the S ₂ pin. This signal is used to read devices which reside on the 8086 local bus. RD is active LOW during T ₂ , T ₃ and T _W of any read cycle, and is guaranteed to remain HIGH in T ₂ until the 8086 local bus has floated. This signal floats to 3-state OFF in "hold acknowledge".		

Table 1. Pin Description (Continued)

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

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

$\overline{\text{S}}_2, \overline{\text{S}}_1, \overline{\text{S}}_0$	26-28	O	STATUS: active during T ₄ , T ₁ , and T ₂ and is returned to the passive state (1, 1, 1) during T ₃ or during T _W when READY is HIGH. This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals. Any change by $\overline{\text{S}}_2$, $\overline{\text{S}}_1$, or $\overline{\text{S}}_0$ during T ₄ is used to indicate the beginning of a bus cycle, and the return to the passive state in T ₃ or T _W is used to indicate the end of a bus cycle.
---	-------	---	--

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function			
$\overline{S_2}, \overline{S_1}, \overline{S_0}$ (Continued)	26–28	O	These signals float to 3-state OFF in “hold acknowledge”. These status lines are encoded as shown.			
			$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Characteristics
			0 (LOW)	0	0	Interrupt Acknowledge
			0	0	1	Read I/O Port
			0	1	0	Write I/O Port
			0	1	1	Halt
			1 (HIGH)	0	0	Code Access
			1	0	1	Read Memory
			1	1	0	Write Memory
			1	1	1	Passive
$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$	30, 31	I/O	<p>REQUEST/GRANT: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\overline{GT_1}$. $\overline{RQ}/\overline{GT}$ pins have internal pull-up resistors and may be left unconnected. The request/grant sequence is as follows (see Page 2-24):</p> <ol style="list-style-type: none">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. <p>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.</p> <p>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:</p> <ol style="list-style-type: none">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. <p>If the local bus is idle when the request is made the two possible events will follow:</p> <ol style="list-style-type: none">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.			
LOCK	29	O	<p>LOCK: output indicates that other system bus masters are not to gain control of the system bus while LOCK is active LOW. The 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”.</p>			

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function		
QS ₁ , QS ₀	24, 25	O	QUEUE STATUS: The queue status is valid during the CLK cycle after which the queue operation is performed. QS ₁ and QS ₀ provide status to allow external tracking of the internal 8086 instruction queue.		
			QS ₁	QS ₀	Characteristics
			0 (LOW)	0	No Operation
			0	1	First Byte of Op Code from Queue
			1 (HIGH)	0	Empty the Queue
			1	1	Subsequent Byte from Queue

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

M/ \overline{IO}	28	O	STATUS LINE: logically equivalent to S ₂ in the maximum mode. It is used to distinguish a memory access from an I/O access. M/ \overline{IO} becomes valid in the T ₄ preceding a bus cycle and remains valid until the final T ₄ of the cycle (M = HIGH, IO = LOW). M/ \overline{IO} floats to 3-state OFF in local bus "hold acknowledge".
\overline{WR}	29	O	WRITE: indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/ \overline{IO} signal. \overline{WR} is active for T ₂ , T ₃ and T _W of any write cycle. It is active LOW, and floats to 3-state OFF in local bus "hold acknowledge".
\overline{INTA}	24	O	\overline{INTA}: is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T ₂ , T ₃ and T _W of each interrupt acknowledge cycle.
ALE	25	O	ADDRESS LATCH ENABLE: provided by the processor to latch the address into the 8282/8283 address latch. It is a HIGH pulse active during T ₁ of any bus cycle. Note that ALE is never floated.
DT/ \overline{R}	27	O	DATA TRANSMIT/RECEIVE: needed in minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/ \overline{R} is equivalent to S ₁ in the maximum mode, and its timing is the same as for M/ \overline{IO} . (T = HIGH, R = LOW.) This signal floats to 3-state OFF in local bus "hold acknowledge".
\overline{DEN}	26	O	DATA ENABLE: provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. \overline{DEN} is active LOW during each memory and I/O access and for \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 in local bus "hold acknowledge".
HOLD, HLDA	31, 30	I/O	HOLD: indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T ₄ or T 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 the HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. Hold acknowledge (HLDA) and HOLD have internal pull-up resistors. The same rules as for $\overline{RQ}/\overline{GT}$ apply regarding when the local bus will be released. HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time.

FUNCTIONAL DESCRIPTION

General Operation

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

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

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

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

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million

bytes, addressed as 00000(H) to FFFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3a.)

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

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

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

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

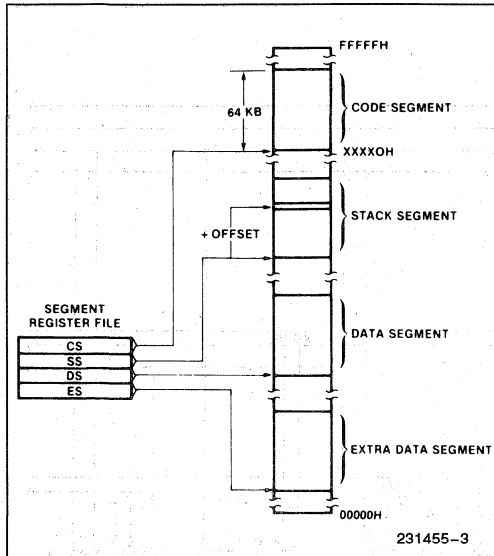


Figure 3a. Memory Organization

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

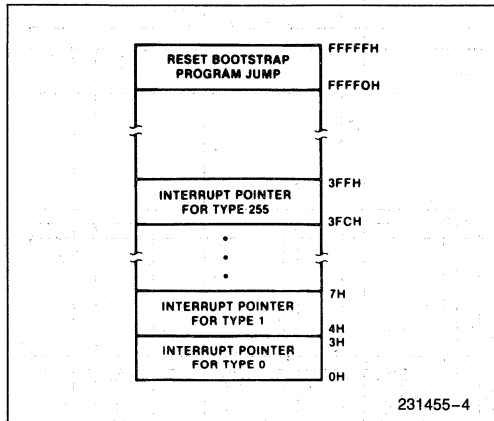


Figure 3b. Reserved Memory Locations

Certain locations in memory are reserved for specific CPU operations (see Figure 3b). Locations from

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

MINIMUM AND MAXIMUM MODES

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

BUS OPERATION

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

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

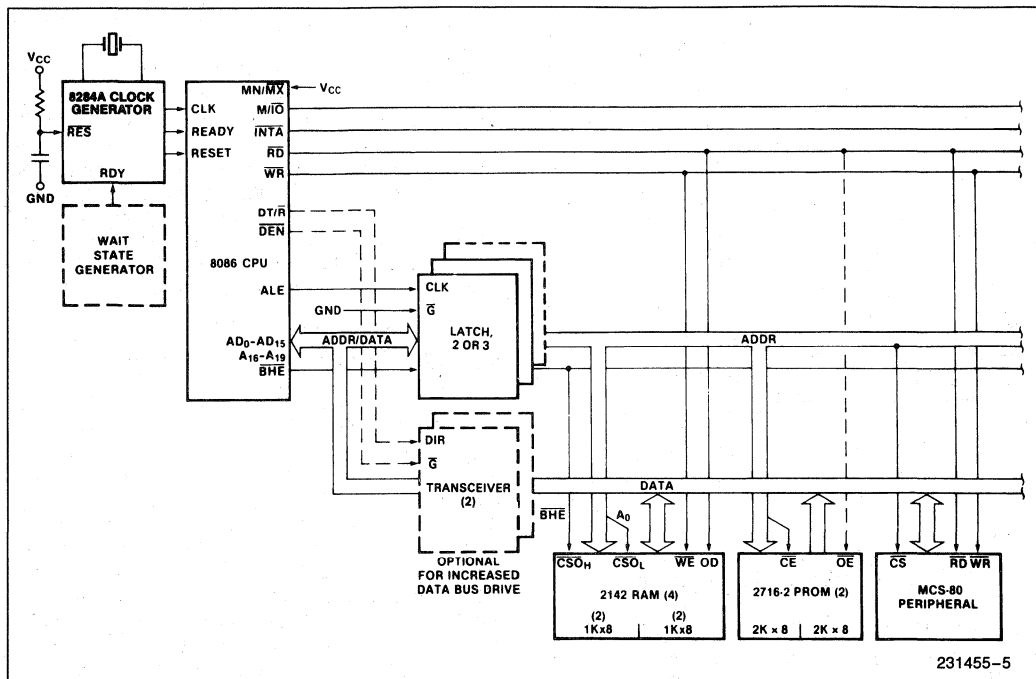


Figure 4a. Minimum Mode 8086 Typical Configuration

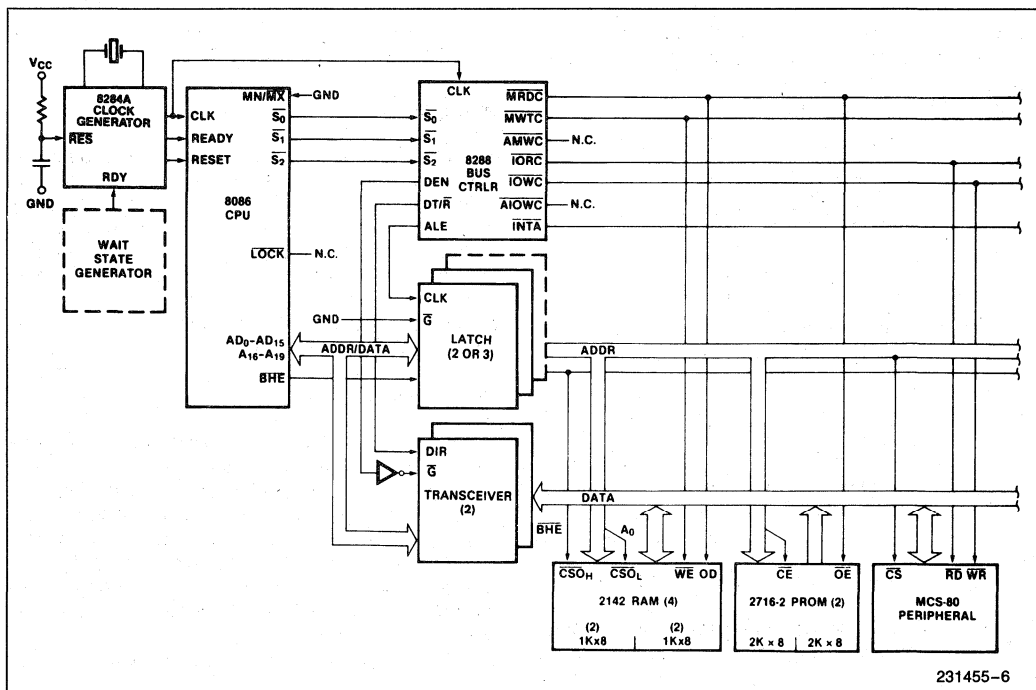


Figure 4b. Maximum Mode 8086 Typical Configuration

can occur between 8086 bus cycles. These are referred to as "Idle" states (T_1) or inactive CLK cycles. The processor uses these cycles for internal house-keeping.

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

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

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

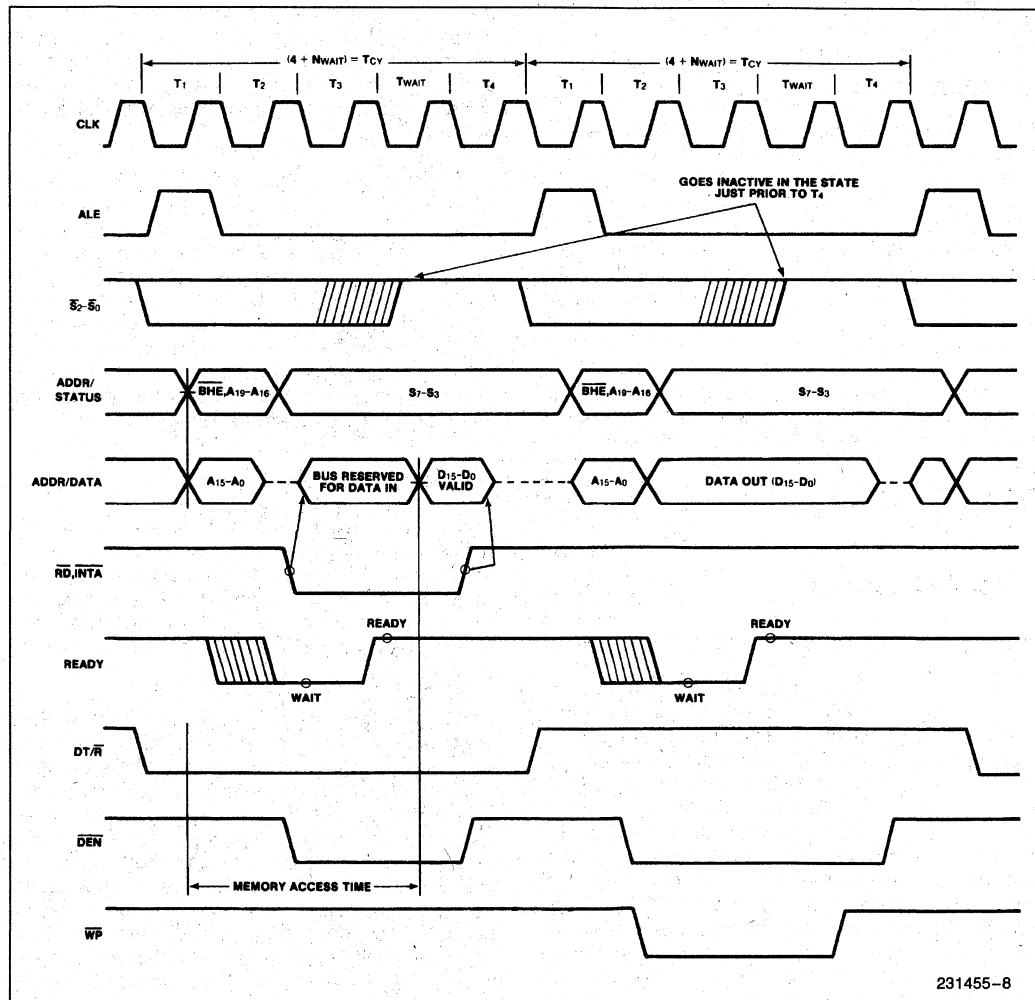


Figure 5. Basic System Timing

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

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

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

I/O ADDRESSING

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

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

External Interface

PROCESSOR RESET AND INITIALIZATION

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

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

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

INTERRUPT OPERATIONS

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

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

NON-MASKABLE INTERRUPT (NMI)

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

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

MASKABLE INTERRUPT (INTR)

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

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

HALT

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

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The LOCK status information is provided by the processor when directly consecutive bus cycles are required during the execution of an instruction. This provides the processor with the capability of performing read/modify/write operations on memory (via the Exchange Register With Memory instruction, for example) without the possibility of another system bus master receiving intervening memory cycles. This is useful in multi-processor system configurations to accomplish "test and set lock" operations. The 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 LOCK is active a request on a RQ/GT pin will be recorded and then honored at the end of the LOCK.

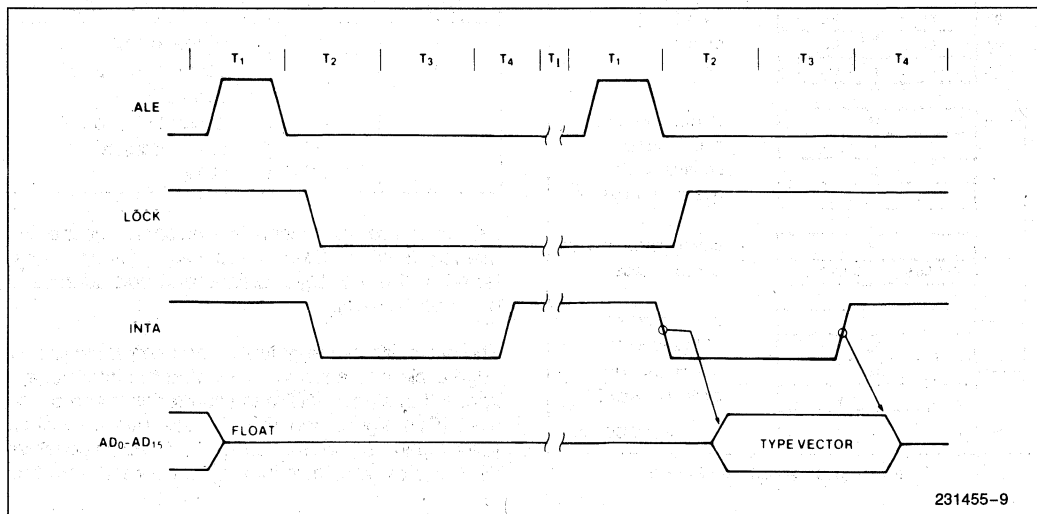


Figure 6. Interrupt Acknowledge Sequence

EXTERNAL SYNCHRONIZATION VIA TEST

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

Basic System Timing

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

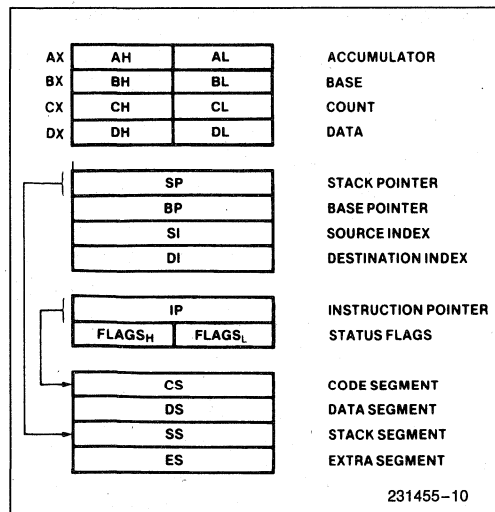


Figure 7. 8086 Register Model

SYSTEM TIMING—MINIMUM SYSTEM

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (low-going) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into the address latch. The **BHE** and **A₀** signals address the low, high, or both bytes. From T_1 to T_4 the **M/I_O** 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 (**RD**) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the **READY** line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver is required to buffer the 8086 local bus, signals **DT/R** and **DEN** are provided by the 8086.

A write cycle also begins with the assertion of **ALE** and the emission of the address. The **M/I_O** 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_4 the processor asserts the write control signal. The write (**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 **BHE** and **A₀** signals are used to select the proper byte(s) of the memory/I/O word to be read or written according to the following table:

BHE	A0	Characteristics
0	0	Whole word
0	1	Upper byte from/to odd address
1	0	Lower byte from/to even address
1	1	None

I/O ports are addressed in the same manner as memory location. Even addressed bytes are transferred on the **D₇–D₀** bus lines and odd addressed bytes on **D₁₅–D₈**.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal (**INTA**) is asserted in place of the read (**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 D₇–D₀ 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 an interrupt vector lookup table, as described earlier.

BUS TIMING—MEDIUM SIZE SYSTEMS

For medium size systems the MN/ \overline{MX} pin is connected to V_{SS} and the 8288 Bus Controller is added to the system as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 8086 is capable of handling. Signals ALE, DEN, and DT/ \overline{R} are generated by the 8288 instead of the processor in this configuration although their timing remains relatively the same. The 8086 status outputs ($\overline{S_2}$, $\overline{S_1}$, and $\overline{S_0}$) provide type-of-cycle information and become 8288 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt

acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence data isn't valid at the leading edge of write. The transceiver receives the usual DIR and \overline{G} inputs from the 8288's DT/ \overline{R} and DEN.

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

ABSOLUTE MAXIMUM RATINGS*

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

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

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

D.C. CHARACTERISTICS (8086: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5V \pm 10\%$)
 (8086-1: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5V \pm 5\%$)
 (8086-2: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5V \pm 5\%$)

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	+0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.5$	V	(Notes 1, 2)
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.5 \text{ mA}$
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -400 \mu\text{A}$
I_{CC}	Power Supply Current: 8086 8086-1 8086-2		340 360 350	mA	$T_A = 25^\circ\text{C}$
I_{LI}	Input Leakage Current		± 10	μA	$0V \leq V_{IN} \leq V_{CC}$ (Note 3)
I_{LO}	Output Leakage Current		± 10	μA	$0.45V \leq V_{OUT} \leq V_{CC}$
V_{CL}	Clock Input Low Voltage	-0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.9	$V_{CC} + 1.0$	V	
C_{IN}	Capacitance of Input Buffer (All input except \overline{AD}_0 - \overline{AD}_{15} , $\overline{RQ}/\overline{GT}$)		15	pF	$f_c = 1 \text{ MHz}$
C_{IO}	Capacitance of I/O Buffer (\overline{AD}_0 - \overline{AD}_{15} , $\overline{RQ}/\overline{GT}$)		15	pF	$f_c = 1 \text{ MHz}$

NOTES:

1. V_{IL} tested with $\overline{MN}/\overline{MX}$ Pin = 0V. V_{IH} tested with $\overline{MN}/\overline{MX}$ Pin = 5V. $\overline{MN}/\overline{MX}$ Pin is a Strap Pin.
2. Not applicable to $\overline{RQ}/\overline{GT}_0$ and $\overline{RQ}/\overline{GT}_1$ (Pins 30 and 31).
3. HOLD and HLDA $I_{LI} \text{ min} = 30 \mu\text{A}$, max = $500 \mu\text{A}$.

A.C. CHARACTERISTICS (8086: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 10\%$)
 (8086-1: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 5\%$)
 (8086-2: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 5\%$)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

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

A.C. CHARACTERISTICS (Continued)

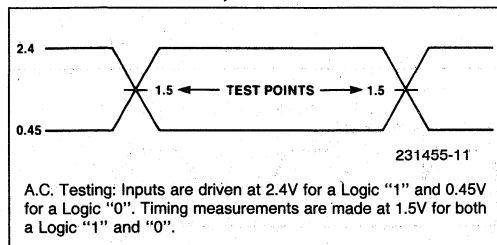
TIMING RESPONSES

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

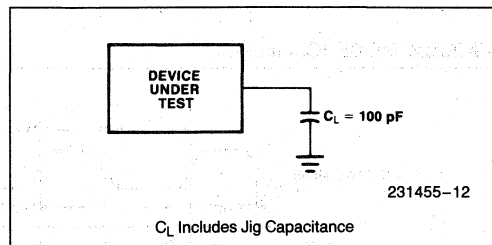
NOTES:

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

A.C. TESTING INPUT, OUTPUT WAVEFORM

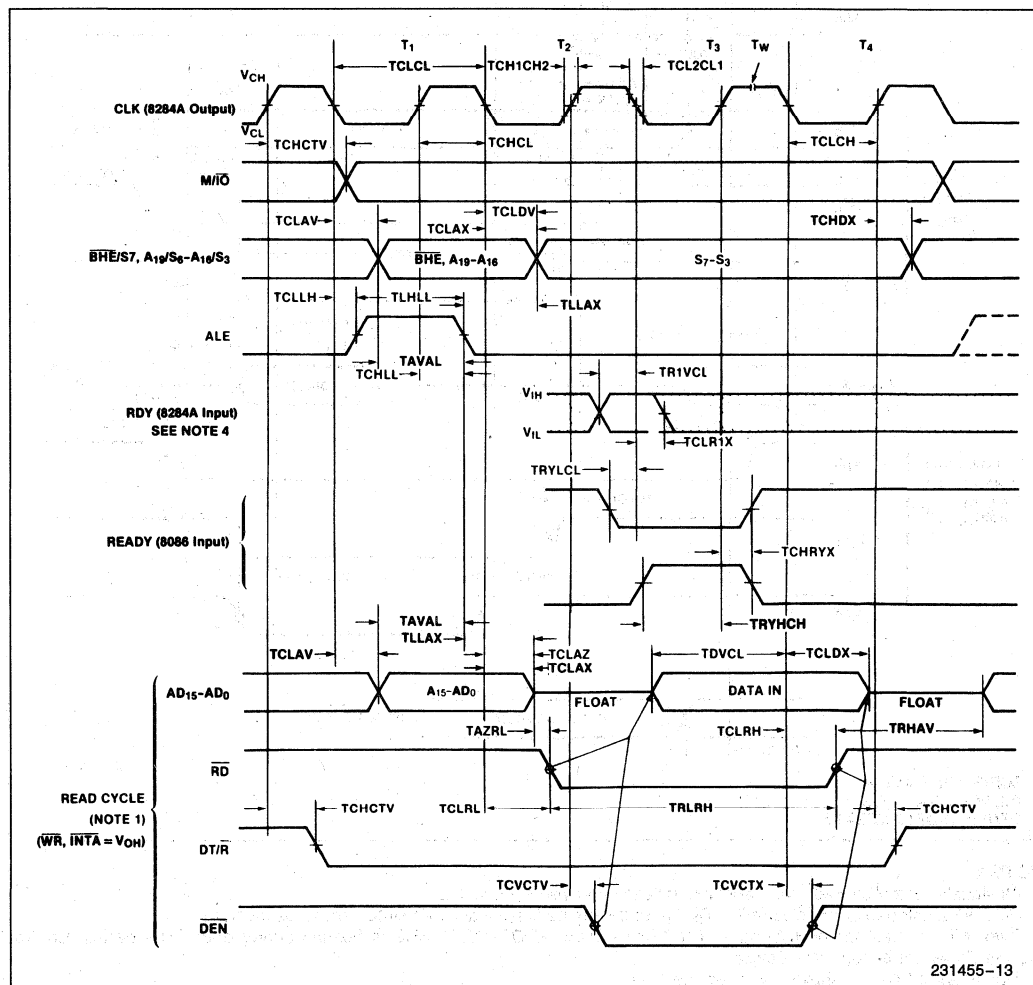


A.C. TESTING LOAD CIRCUIT



WAVEFORMS

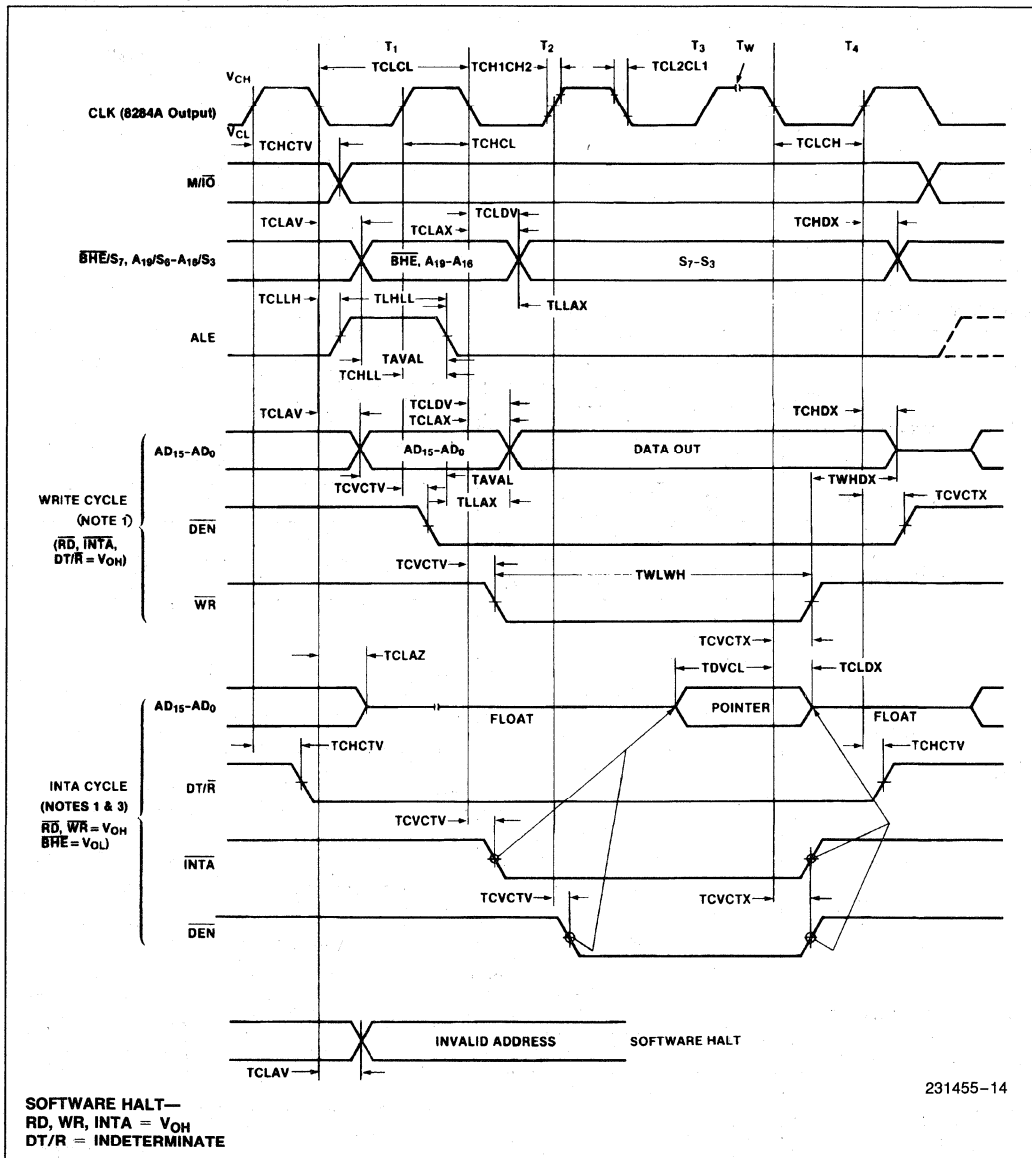
MINIMUM MODE



2

WAVEFORMS (Continued)

MINIMUM MODE (Continued)



NOTES:

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

A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER) TIMING REQUIREMENTS

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

A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

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

A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES (Continued)

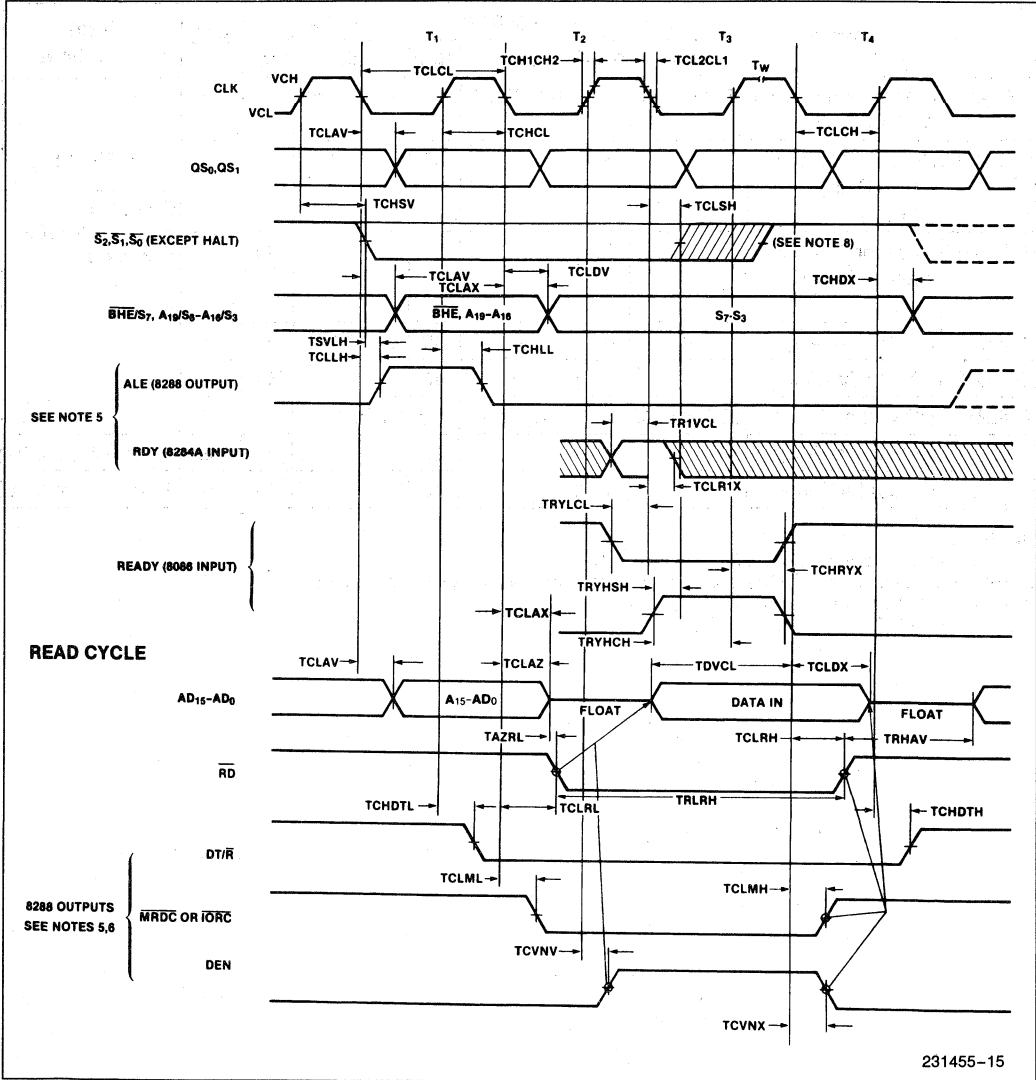
Symbol	Parameter	8086		8086-1		8086-2		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
TRHAV	RD Inactive to Next Address Active	TCLCL-45		TCLCL-35		TCLCL-40		ns	C _L = 20–100 pF for all 8086 Outputs (In addition to 8086 self-load)
TCHDTL	Direction Control Active Delay (Note 1)		50		50		50	ns	
TCHDTH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

NOTES:

- Signal at 8284A or 8288 shown for reference only.
- Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
- Applies only to T3 and wait states.
- Applies only to T2 state (8 ns into T3).

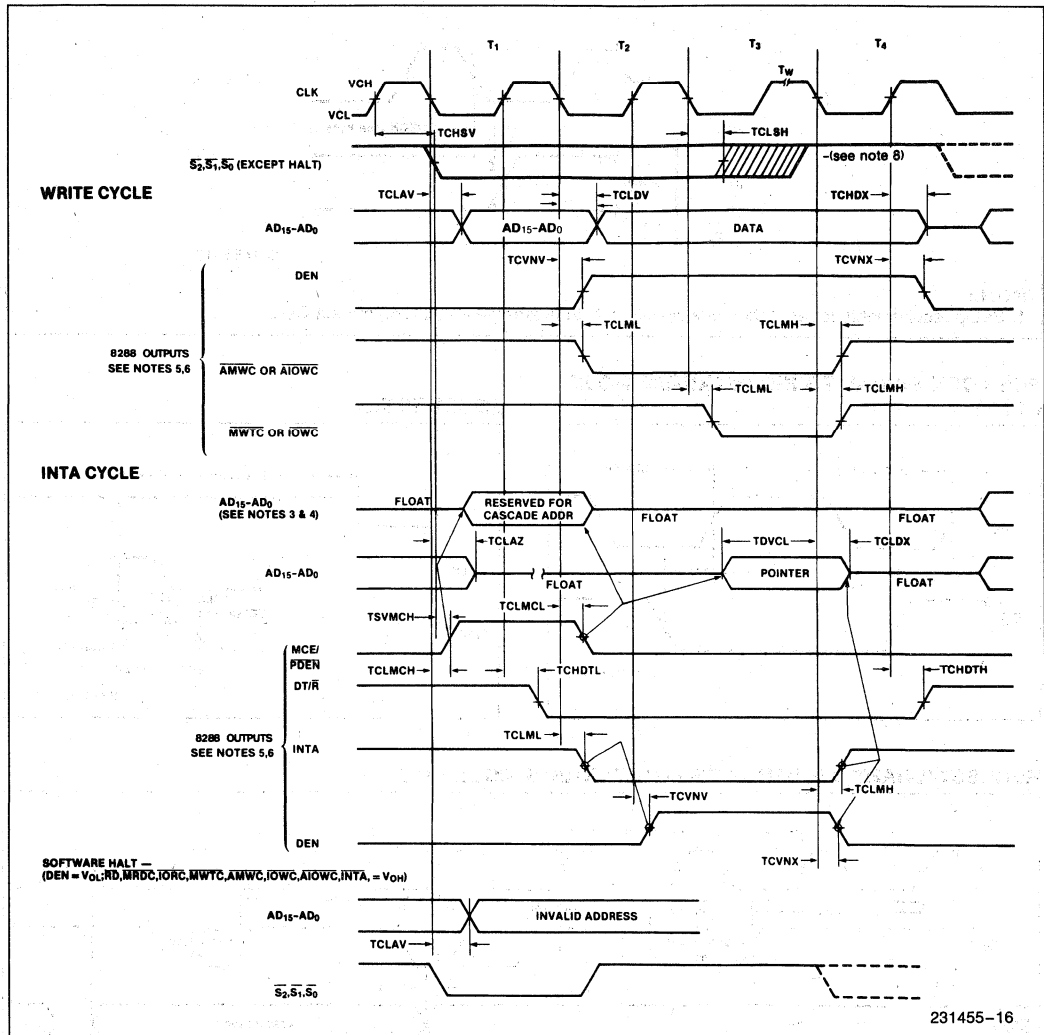
WAVEFORMS

MAXIMUM MODE



WAVEFORMS (Continued)

MAXIMUM MODE (Continued)

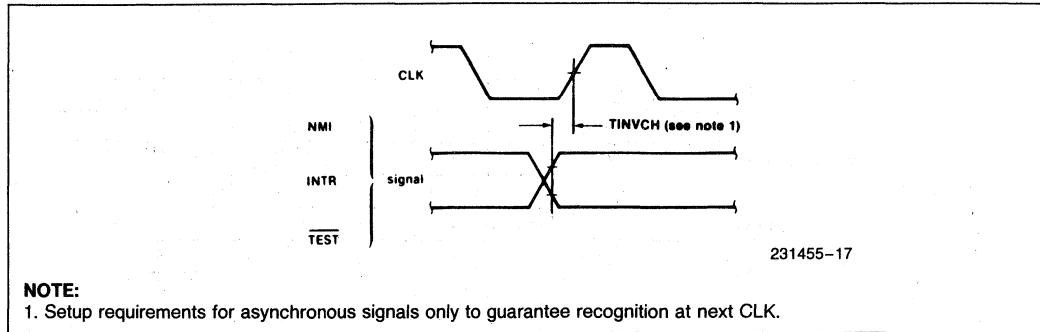


NOTES:

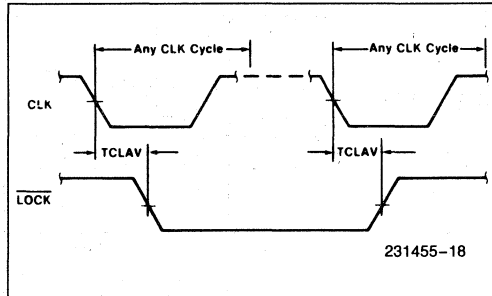
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2 , T_3 , T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycle.
4. Two INTA cycles run back-to-back. The 8086 LOCAL ADDR/DATA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

WAVEFORMS (Continued)

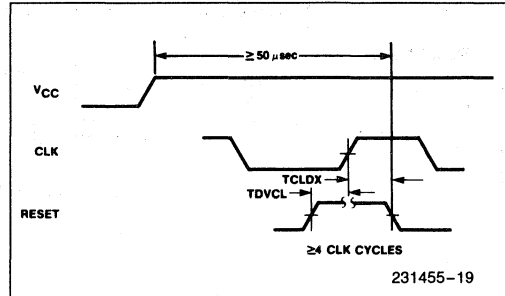
ASYNCHRONOUS SIGNAL RECOGNITION



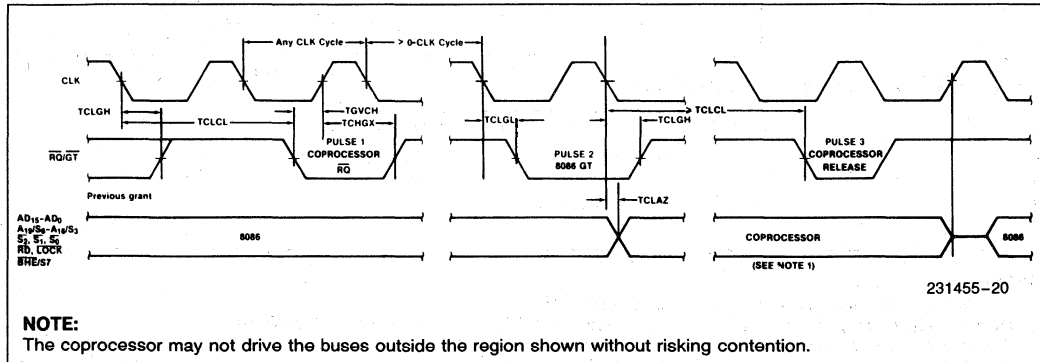
BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



RESET TIMING

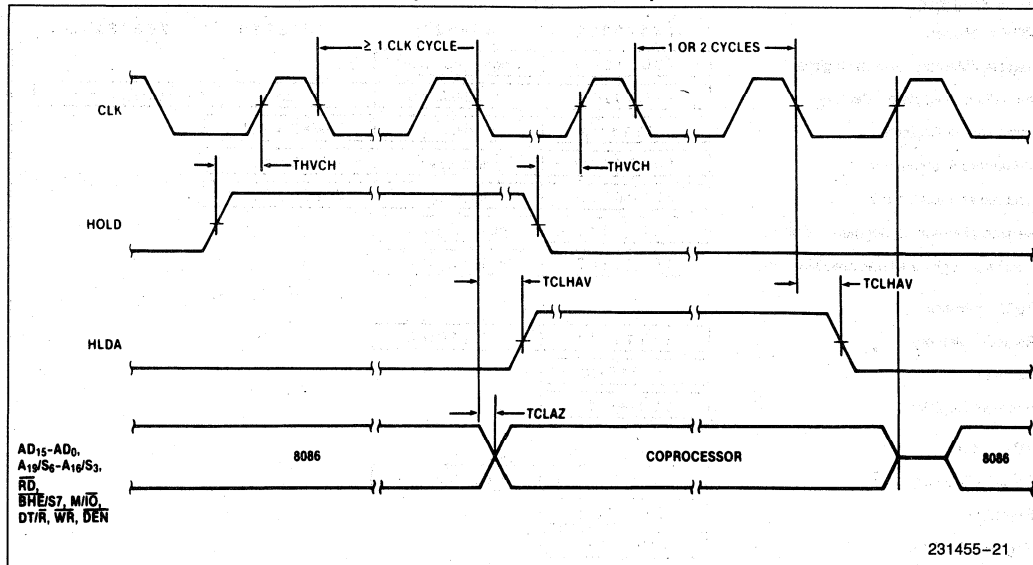


REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



WAVEFORMS (Continued)

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



2

Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
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 w	addr-low	addr-high	
Accumulator to Memory	1 0 1 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 = Translate 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		
LAHF = Load AH with Flags	1 0 0 1 1 1 1 1			
SAHF = 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			

Mnemonics © Intel, 1978

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
	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
ARITHMETIC				
ADD = Add:				
Reg./Memory with Register to Either	0 0 0 0 0 d w	mod reg r/m		
Immediate to Register/Memory	1 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 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			
BAA = 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 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	
SSB = 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 s w	mod 0 1 1 r/m	data	data if s w = 01
Immediate from Accumulator	0 0 0 1 1 1 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 and Register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with Register/Memory	1 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 = Multiply (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 = ASCII 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			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
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 = Stor Byte/Wd from AL/A	1 0 1 0 1 0 1 w			
CONTROL TRANSFER				
CALL = Call:				
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		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	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 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 100 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 101 r/m	
RET = Return from CALL:			
Within Segment	1 1 0 0 0 0 1 1		
Within Seg Adding Immed 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	disp	
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	
INT = Interrupt			
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		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
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 = Escape (to External Device)	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	

NOTES:

AL = 8-bit accumulator

AX = 16-bit accumulator

CX = Count register

DS = Data segment

ES = Extra segment

Above/below refers to unsigned value

Greater = more positive;

Less = less positive (more negative) signed values

if d = 1 then "to" reg; if d = 0 then "from" reg

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

if s w = 01 then 16 bits of immediate data form the operand

if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand

if v = 0 then "count" = 1; if v = 1 then "count" in (CL)

x = don't care

z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

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

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

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

Mnemonics © Intel, 1978

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -004 data sheet. Please review this summary carefully.

1. The Intel® 8086 implementation technology (HMOS) has been changed to (HMOS-III).
2. Delete all "changes from 1985 Handbook Specification" sentences.



80C86A 16-BIT CHMOS MICROPROCESSOR

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

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

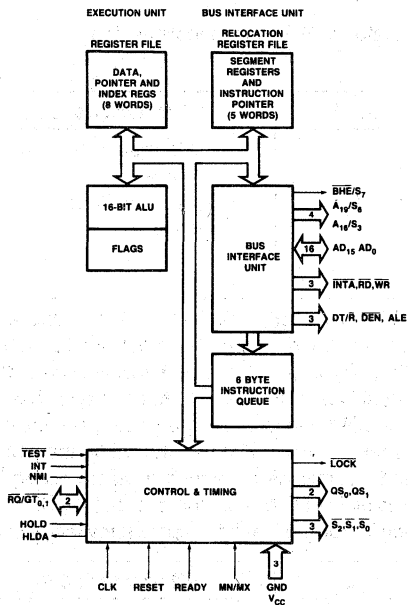


Figure 1. 80C86A
CPU Block Diagram

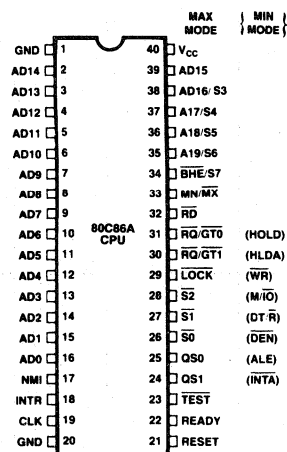


Figure 2. 80C86A
40-Lead DIP Configuration

Table 1. Pin Description

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

Symbol	Pin No.	Type	Name and Function		
AD ₁₅ –AD ₀	2–16, 39	I/O	ADDRESS DATA BUS: These lines constitute the time multiplexed memory/I/O address (T ₁) and data (T ₂ , T ₃ , T _W , T ₄) bus. A ₀ is analogous to 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 BHE.) These lines are active HIGH and float to 3-state OFF ⁽¹⁾ during interrupt acknowledge and local bus "hold acknowledge."		
A ₁₉ /S ₆ , A ₁₈ /S ₅ , A ₁₇ /S ₄ , A ₁₆ /S ₃	35–38	O	ADDRESS/STATUS: During T ₁ these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T ₂ , T ₃ , T _W , and T ₄ . The status of the interrupt enable FLAG bit (S ₅) is updated at the beginning of each CLK cycle. A ₁₇ /S ₄ and A ₁₆ /S ₃ are encoded as shown. This information indicates which relocation register is presently being used for data accessing. These lines float to 3-state OFF ⁽¹⁾ during local bus "hold acknowledge."		
			A ₁₇ /S ₄	A ₁₆ /S ₃	Characteristics
			0 (LOW) 0 1 (HIGH) 1 S ₆ is 0 (LOW)	0 1 0 1	Alternate Data Stack Code or None Data
BHE/S ₇	34	O	BUS HIGH ENABLE/STATUS: During T ₁ the bus high enable signal (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 BHE to condition chip select functions. BHE is LOW during T ₁ for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S ₇ status information is available during T ₂ , T ₃ , and T ₄ . The signal is active LOW, and floats to 3-state OFF ⁽¹⁾ in "hold." It is LOW during T ₁ for the first interrupt acknowledge cycle.		
			BHE	A ₀	Characteristics
			0 0 1 1	0 1 0 1	Whole word Upper byte from/ to odd address Lower byte from/ to even address None

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
\overline{RD}	32	O	<p>READ: Read strobe indicates that the processor is performing a memory of I/O read cycle, depending on the state of the S_2 pin. This signal is used to read devices which reside on the 80C86A local bus. \overline{RD} is active LOW during T_2, T_3 and T_W of any read cycle, and is guaranteed to remain HIGH in T_2 until the 80C86A local bus has floated.</p> <p>This floats to 3-state OFF in "hold acknowledge."</p>
READY	22	I	<p>READY: is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory/IO is synchronized by the 82C84A Clock Generator to form READY. This signal is active HIGH. The 80C86A READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.</p>
INTR	18	I	<p>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.</p>
TEST	23	I	<p>TEST: input is examined by the "Wait" instruction. If the 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.</p>
NMI	17	I	<p>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.</p>
RESET	21	I	<p>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.</p>
CLK	19	I	<p>CLOCK: provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.</p>
V_{CC}	40		<p>V_{CC}: +5V power supply pin.</p>
GND	1, 20		<p>GROUND: Both must be connected.</p>
MN/ \overline{MX}	33	I	<p>MINIMUM/MAXIMUM: indicates what mode the processor is to operate in. The two modes are discussed in the following sections.</p>

Table 1. Pin Description (Continued)

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

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

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function															
			<p>If the local bus is idle when the request is made the two possible events will follow:</p> <p>1. Local bus will be released during the next clock.</p> <p>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.</p>															
LOCK	29	O	<p>LOCK: output indicates that other system bus masters are not to gain control of the system bus while LOCK is active LOW. The LOCK signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF⁽¹⁾ in "hold acknowledge."</p>															
QS ₁ , QS ₀	24, 25	O	<p>QUEUE STATUS: The queue status is valid during the CLK cycle after which the queue operation is performed.</p> <p>QS₁ and QS₀ provide status to allow external tracking of the internal 80C86A instruction queue.</p>															
			<table><tr><th>QS₁</th><th>QS₀</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 Op Code 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>	QS ₁	QS ₀	Characteristics	0 (LOW)	0	No Operation	0	1	First Byte of Op Code from Queue	1 (HIGH)	0	Empty the Queue	1	1	Subsequent Byte from Queue
			QS ₁	QS ₀	Characteristics													
0 (LOW)	0	No Operation																
0	1	First Byte of Op Code from Queue																
1 (HIGH)	0	Empty the Queue																
1	1	Subsequent Byte from Queue																

2

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

M/ \overline{IO}	28	O	<p>STATUS LINE: logically equivalent to S₂ in the maximum mode. It is used to distinguish a memory access from an I/O access. M/\overline{IO} becomes valid in the T₄ preceding a bus cycle and remains valid until the final T₄ of the cycle (M = HIGH, IO = LOW). M/\overline{IO} floats to 3-state OFF(1) in local bus "hold acknowledge."</p>
\overline{WR}	29	O	<p>WRITE: indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/\overline{IO} signal. \overline{WR} is active for T₂, T₃ and T_W of any write cycle. It is active LOW, and floats to 3-state OFF(1) in local bus "hold acknowledge."</p>
\overline{INTA}	24	O	<p>\overline{INTA} is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T₂, T₃ and T_W of each interrupt acknowledge cycle.</p>
ALE	25	O	<p>ADDRESS LATCH ENABLE: provided by the processor to latch the address into an address latch. It is a HIGH pulse active during T₁ of any bus cycle. Note that ALE is never floated.</p>
DT/ \overline{R}	27	O	<p>DATA TRANSMIT/RECEIVE: needed in minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/\overline{R} is equivalent to S₁ in the maximum mode, and its timing is the same as for M/\overline{IO}. (T = HIGH, R = LOW.) This signal floats to 3-state OFF(1) in local bus "hold acknowledge."</p>

Table 1. Pin Description (Continued)

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

NOTE:

1. See the section on Bus Hold Circuitry.

FUNCTIONAL DESCRIPTION**STATIC OPERATION**

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

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

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

INTERNAL ARCHITECTURE

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

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

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

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

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

MEMORY ORGANIZATION

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

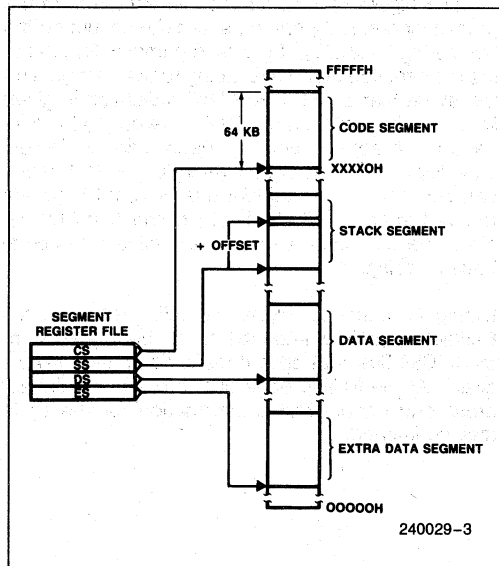


Figure 3a. Memory Organization

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

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

Physically, the memory is organized as a high bank (D₁₅-D₈) and a low bank (D₇-D₀) of 512k 8-bit bytes addressed in parallel by the processor's address lines.

A₁₉-A₁. Byte data with even addresses is transferred on the D₇-D₀ bus lines while odd addressed byte data (A₀ HIGH) is transferred on the D₁₅-D₈ bus lines. The processor provides two enable signals, BHE and A₀, to selectively allow reading from or writing into either an odd byte location, even byte location, or both. The instruction stream is fetched from memory as words and is addressed internally by the processor to the byte level as necessary.

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

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

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

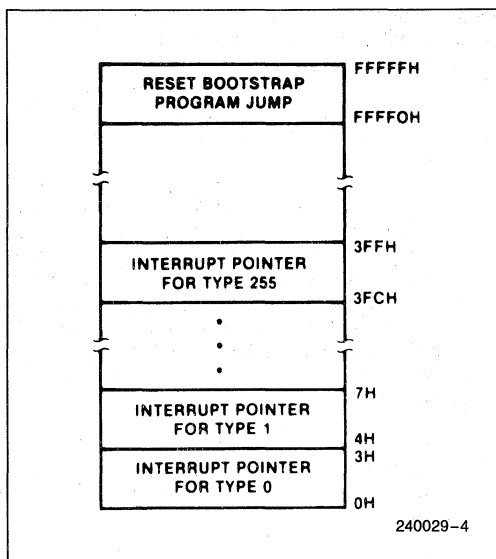


Figure 3b. Reserved Memory Locations

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 80C86A systems are sufficiently different that

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

BUS OPERATION

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

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

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



Figure 4a. Minimum Mode iAPX 80C86A Typical Configuration



Figure 4b. Maximum Mode 80C86A Typical Configuration

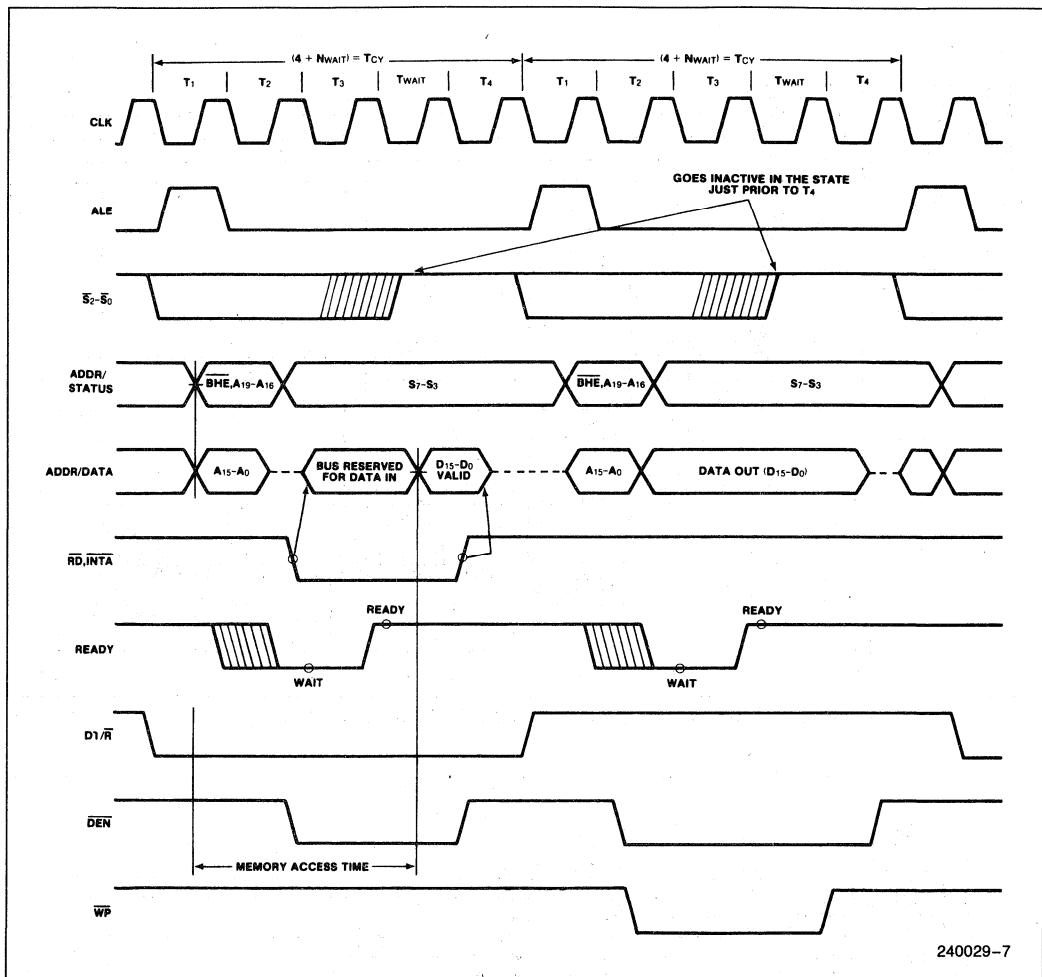


Figure 5. Basic System Timing

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

\bar{S}_2	\bar{S}_1	\bar{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)

therefore valid during T_2 through T_4 . S_3 and S_4 indicate which segment register (see Instruction Set description) was used for this bus cycle in forming the address, according to the following table:

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

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

Status bits S_3 through S_7 are multiplexed with high-order address bits and the \bar{BHE} signal, and are

I/O ADDRESSING

In the 80C86A, I/O operations can address up to a maximum of 64k I/O byte registers or 32k I/O word registers. The I/O address appears in the same format as the memory address on bus lines A₁₅–A₀. The address lines A₁₉–A₁₆ are zero in I/O operations. The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

I/O ports are addressed in the same manner as memory locations. Even addressed bytes are transferred on the D₇–D₀ bus lines and odd addressed bytes on D₁₅–D₈. Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even.

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 80C86A RESET is required to be HIGH for four or more CLK cycles. The 80C86A will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 7 CLK cycles. After this interval the 80C86A operates normally beginning with the instruction in absolute location FFFF0H (see Figure 3b). The details of this operation are specified in the Instruction Set description of the MCS®-86 Family User's Manual. The RESET input is internally synchronized to the processor clock. At

initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50 μ s after power-up, to allow complete initialization of the 80C86A.

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

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

NOTE:

1. See the section on Bus Hold Circuitry.

2

BUS HOLD CIRCUITRY

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

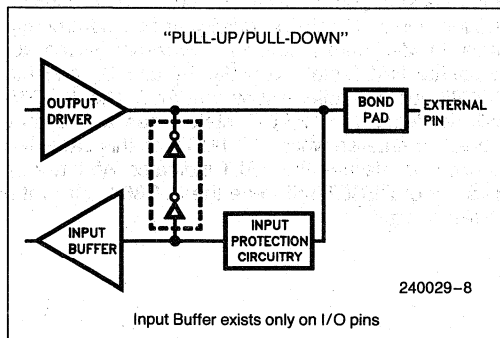


Figure 6a. Bus hold circuitry pin 2-16, 34-39.

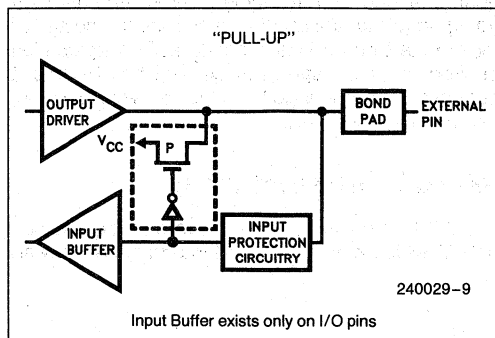


Figure 6b. Bus hold circuitry pin 26-32.

INTERRUPT OPERATIONS

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

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

NON-MASKABLE INTERRUPT (NMI)

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

MASKABLE INTERRUPT (INTR)

The 80C86A provides a single interrupt request input (INTR) which can be masked internally by software

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

During the response sequence (Figure 7) the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 80C86A emits the LOCK signal from 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., 82C59 PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop which returns the status of the original interrupt enable bit when it restores the FLAGS.

HALT

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

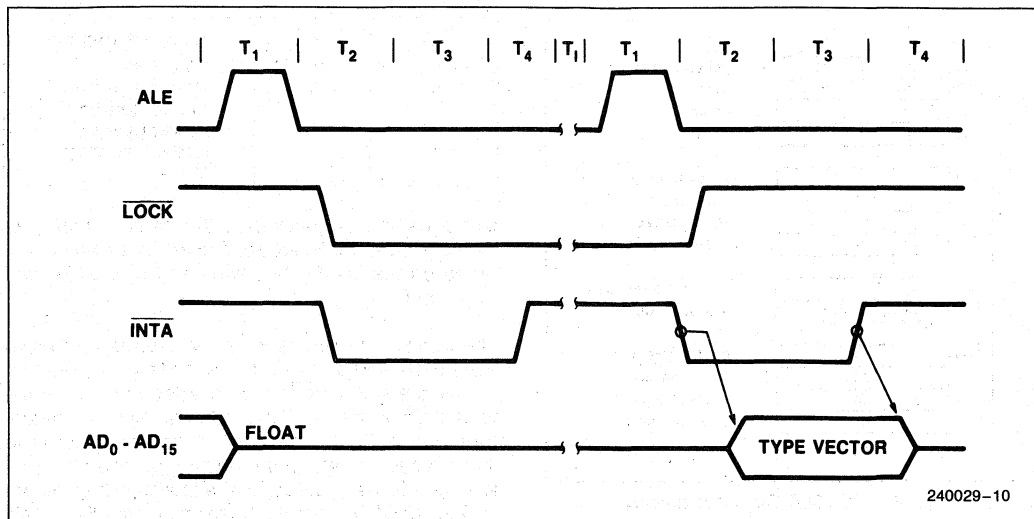


Figure 7. Interrupt Acknowledge Sequence

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The $\overline{\text{LOCK}}$ 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 80C86A 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), pro-

gram 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 80C86A drivers go to 3-state OFF if bus "Hold" is entered. If interrupts are enabled, they may occur while the processor is waiting. When this occurs the processor fetches the WAIT instruction one extra time, processes the interrupt, and then re-fetches and re-executes the WAIT instruction upon returning from the interrupt.

BASIC SYSTEM TIMING

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

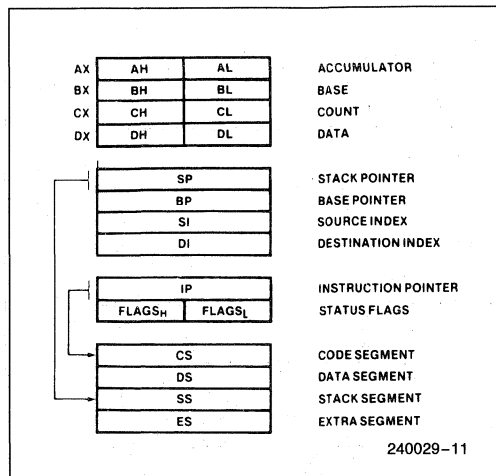


Figure 8. 80C86A Register Model

SYSTEM TIMING—MINIMUM SYSTEM

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

A write cycle also begins with the assertion of ALE and the emission of the address. The $M/\bar{I/O}$ 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_4 the processor asserts the write control signal. The write (\bar{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 BHE and A_0 signals are used to select the proper byte(s) of the memory/I/O word to be read or written according to the following table:

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

BUS TIMING—MEDIUM SIZE SYSTEMS

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

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

ABSOLUTE MAXIMUM RATINGS*

Supply Voltage
(With respect to ground) -0.5 to 7.0V

Input Voltage Applied
(w.r.t. ground) -0.5 to $V_{CC} + 0.5V$

Output Voltage Applied
(w.r.t. ground) -0.5 to $V_{CC} + 0.5V$

Power Dissipation 1.0W

Storage Temperature -65°C to 150°C

Ambient Temperature Under Bias 0°C to 70°C

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

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

D.C. CHARACTERISTICS

($T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5V \pm 5\%$)

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

NOTES:

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

A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ± 5%)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

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

A.C. CHARACTERISTICS (Continued)

($T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5V \pm 5\%$)

Timing Responses

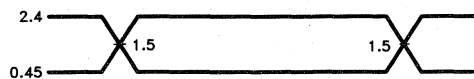
Symbol	Parameter	80C86A-2		Units	Test Conditions
		Min	Max		
TCLAV	Address Valid Delay	10	60	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	50	ns	
TLHLL	ALE Width	TCLCH – 10		ns	
TCLLH	ALE Active Delay		50	ns	
TCHLL	ALE Inactive Delay		55	ns	
TLLAX	Address Hold Time to ALE Inactive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

NOTES:

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

A.C. TESTING INPUT, OUTPUT WAVEFORM

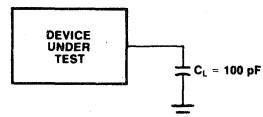
INPUT / OUTPUT



240029-12

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

A.C. TESTING LOAD CIRCUIT

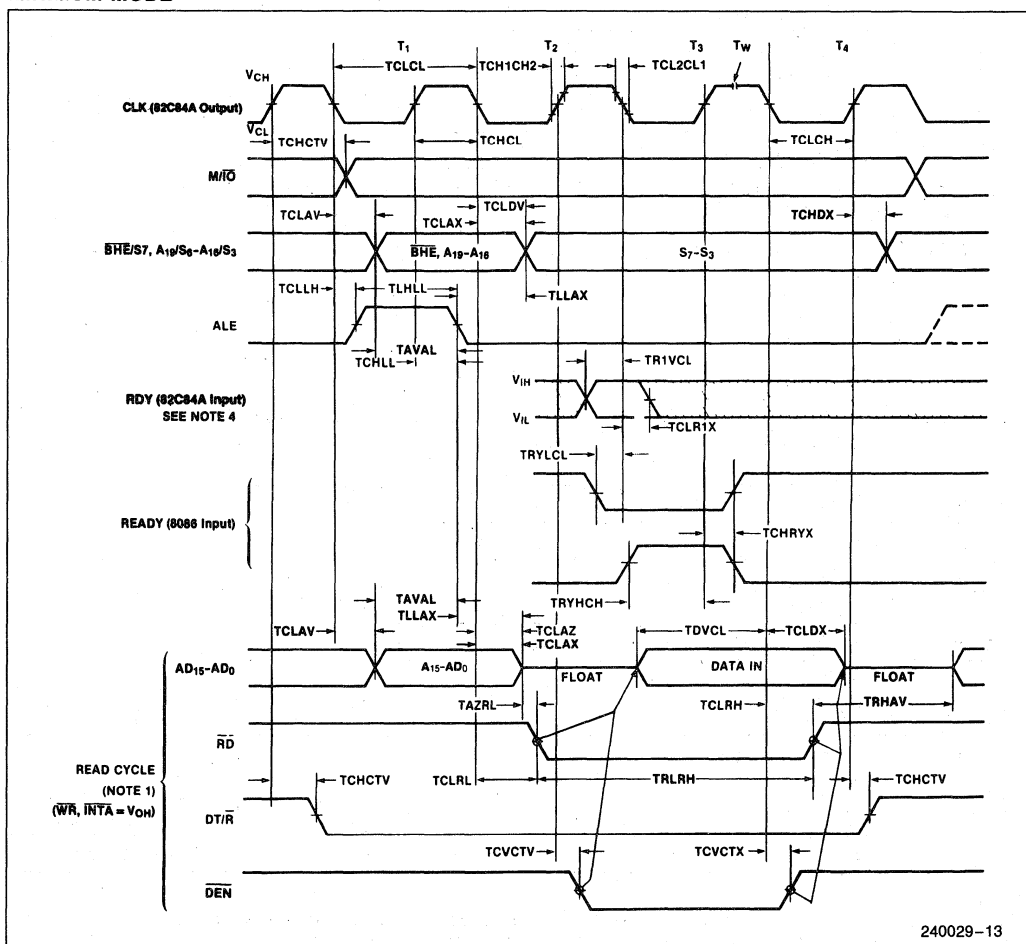


240029-14

C_L Includes Jig Capacitance

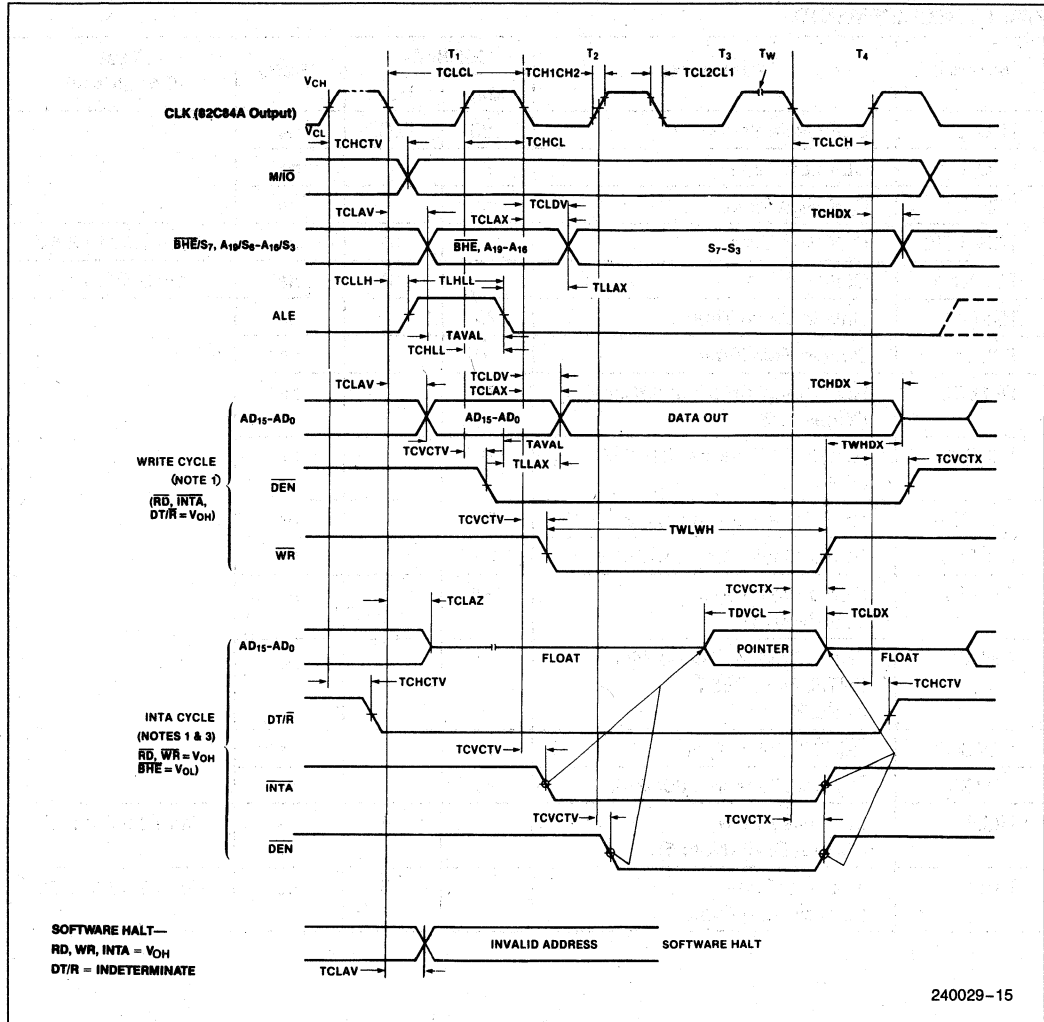
WAVEFORMS

MINIMUM MODE



WAVEFORMS (Continued)

MINIMUM MODE (Continued)



NOTES:

1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T₂, T₃, T_W to determine if T_W machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 80C86A local ADDR/DATA BUS is floating during both INTA cycles. Control signals shown for second INTA cycle.
4. Signals at 82C84A are shown for reference only.

A.C. CHARACTERISTICS

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

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

A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

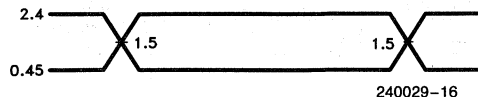
Symbol	Parameter	80C86A-2		Units	Test Conditions
		Min	Max		
TCLML	Command Active Delay (Note 1)	5	35	ns	
TCLMH	Command Inactive Delay (Note 1)	5	35	ns	
TRYHSH	READY Active to Status Passive (Note 3)		65	ns	
TCHSV	Status Active Delay	10	60	ns	
TCLSH	Status Inactive Delay	10	70	ns	
TCLAV	Address Valid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

NOTES:

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

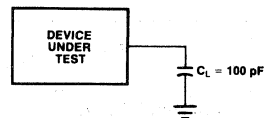
A.C. TESTING INPUT, OUTPUT WAVEFORM

INPUT / OUTPUT



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

A.C. TESTING LOAD CIRCUIT

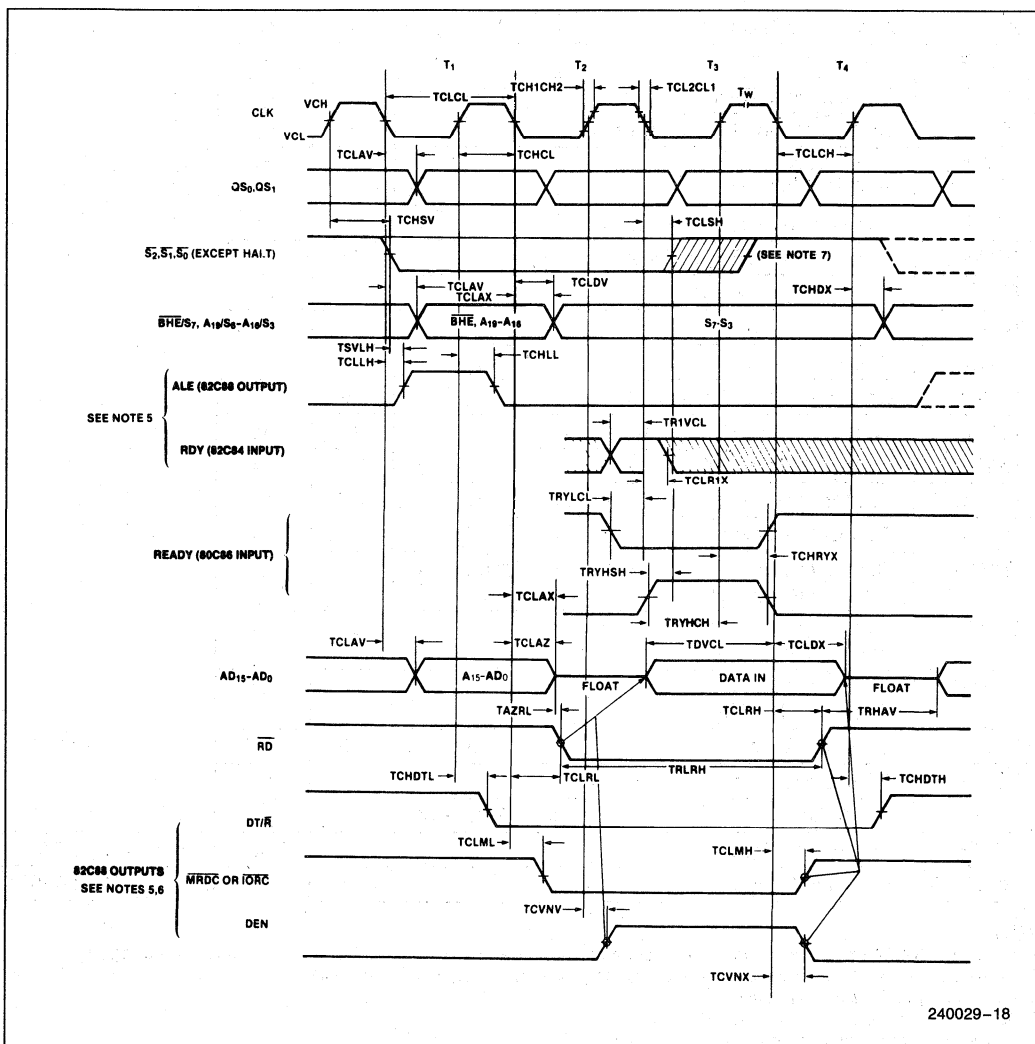


240029-17

C_L Includes Jig Capacitance

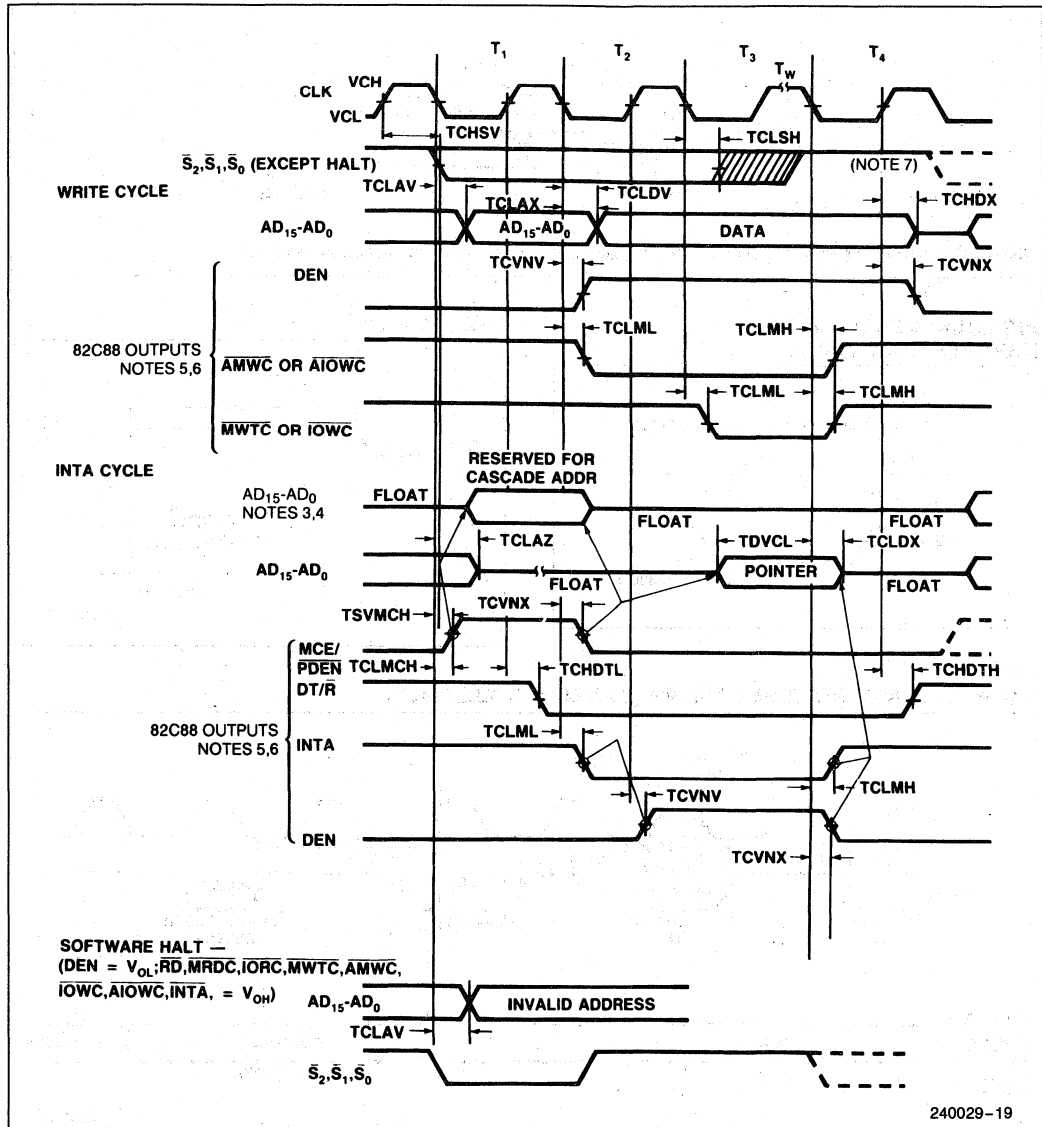
WAVEFORMS

MAXIMUM MODE



WAVEFORMS (Continued)

MAXIMUM MODE (Continued)

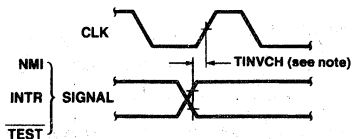


NOTES:

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

WAVEFORMS (Continued)

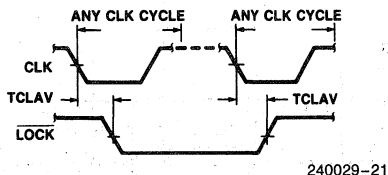
ASYNCHRONOUS SIGNAL RECOGNITION



240029-20

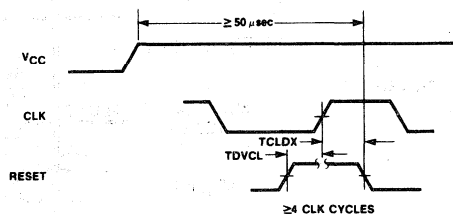
NOTE: Setup requirements for asynchronous signals only to guarantee recognition at next CLK.

BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



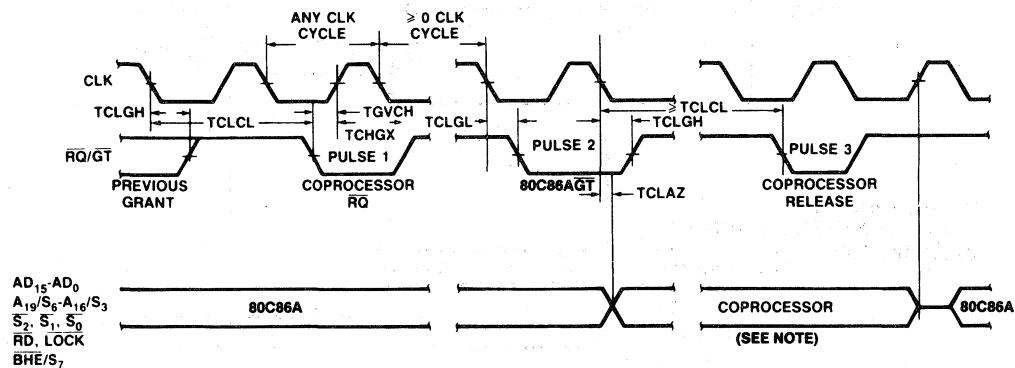
240029-21

RESET TIMING



240029-22

REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)

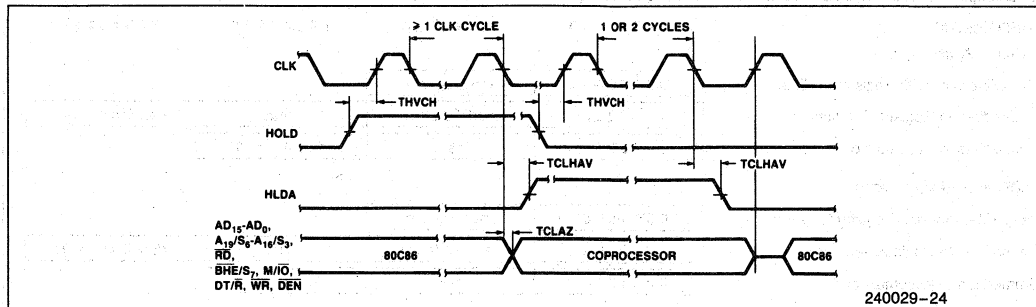


240029-23

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

WAVEFORMS (Continued)

HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



240029-24

Table 2. Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	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 = Translate 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		
LAHF = Load AH with Flags	1 0 0 1 1 1 1 1			
SAHF = 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			

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
	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
ARITHMETIC				
ADD = Add:				
Reg./Memory with Register to Either	0 0 0 0 0 d w	mod reg r/m		
Immediate to Register/Memory	1 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 1 0 w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	0 0 0 1 0 d w	mod reg r/m		
Immediate to Register/Memory	1 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 w	data	data if w = 1	
INC = Increment:				
Register/Memory	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 d w	mod reg r/m		
Immediate from Register/Memory	1 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 w	data	data if w = 1	
SBB = Subtract with Borrow				
Reg./Memory and Register to Either	0 0 0 1 1 d w	mod reg r/m		
Immediate from Register/Memory	1 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 w	data	data if w = 1	
DEC = Decrement:				
Register/Memory	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 and Register	0 0 1 1 1 d w	mod reg r/m		
Immediate with Register/Memory	1 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 = Multiply (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 = ASCII 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			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
	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
LOGIC				
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 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 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 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			
CONTROL TRANSFER				
CALL = Call:				
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		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
CONTROL TRANSFER (Continued)			
JMP = Unconditional Jump:	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 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 Seg. Adding Immed 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	disp	
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	
INT = Interrupt			
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		

Table 2. Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
PROCESSOR CONTROL	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
CLC = Clear Carry	1 1 1 1 0 0 0	
CMC = Complement Carry	1 1 1 1 0 1 0	
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 = Escape (to External Device)	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	

NOTES:

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

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

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

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

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

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

Mnemonics © Intel, 1978

DATA SHEET REVISION REVIEW

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

- In the Pin Description Table (Table 1), the description of the HLDA signal being issued has been corrected. HLDA will be issued in the middle of either the T₄ or T₁ state.



8088

8-BIT HMOS MICROPROCESSOR

8088/8088-2

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

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

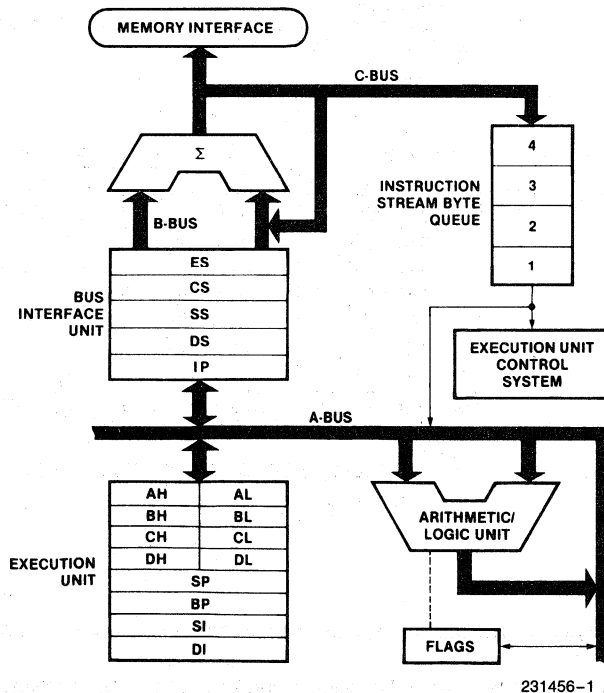


Figure 1. 8088 CPU Functional Block Diagram

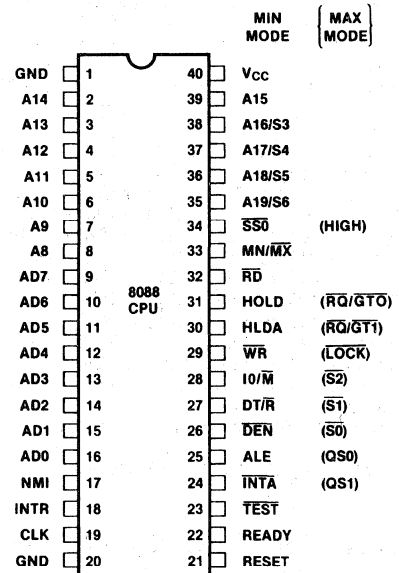


Figure 2. 8088 Pin Configuration

Table 1. Pin Description

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

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

Table 1. Pin Description (Continued)

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

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

Symbol	Pin No.	Type	Name and Function
IO/\overline{M}	28	O	STATUS LINE: is an inverted maximum mode $\overline{S_2}$. 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 ($IO = \text{HIGH}$, $M = \text{LOW}$). IO/\overline{M} floats to 3-state OFF in local bus "hold acknowledge".
\overline{WR}	29	O	WRITE: strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the IO/\overline{M} signal. \overline{WR} is active for T ₂ , T ₃ , and T _w of any write cycle. It is active LOW, and floats to 3-state OFF in local bus "hold acknowledge".
\overline{INTA}	24	O	\overline{INTA}: is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T ₂ , T ₃ , and T _w of each interrupt acknowledge cycle.
ALE	25	O	ADDRESS LATCH ENABLE: is provided by the processor to latch the address into an address latch. It is a HIGH pulse active during clock low of T ₁ of any bus cycle. Note that ALE is never floated.
DT/\overline{R}	27	O	DATA TRANSMIT/RECEIVE: is needed in a minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically, DT/\overline{R} is equivalent to $\overline{S_1}$ in the maximum mode, and its timing is the same as for IO/\overline{M} (T = HIGH, R = LOW). This signal floats to 3-state OFF in local "hold acknowledge".
\overline{DEN}	26	O	DATA ENABLE: is provided as an output enable for the data bus transceiver in a minimum system which uses the transceiver. \overline{DEN} is active LOW during each memory and I/O access, and for \overline{INTA} cycles. For a read or \overline{INTA} cycle, it is active from the middle of 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".

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function			
HOLD, HLDA	31, 30	I, O	HOLD: indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement, in the middle of a T4 or T1 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 and HLDA have internal pull-up resistors. Hold is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the set up time.			
SSO	34	O	STATUS LINE: is logically equivalent to $\overline{S0}$ in the maximum mode. The combination of SSO, IO/M and DT/R allows the system to completely decode the current bus cycle status.			
			IO/M	DT/R	SSO	Characteristics
			1(HIGH)	0	0	Interrupt Acknowledge
			1	0	1	Read I/O Port
			1	1	0	Write I/O Port
			1	1	1	Halt
			0(LOW)	0	0	Code Access
			0	0	1	Read Memory
			0	1	0	Write Memory
			0	1	1	Passive

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

Symbol	Pin No.	Type	Name and Function			
$\overline{S2}, \overline{S1}, \overline{S0}$	26-28	O	STATUS: is active during clock high of T4, T1, and T2, and is returned to the passive state (1,1,1) during T3 or during Tw when READY is HIGH. This status is used by the 8288 bus controller to generate all memory and I/O access control signals. Any change by $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 and Tw is used to indicate the end of a bus cycle. These signals float to 3-state OFF during "hold acknowledge". During the first clock cycle after RESET becomes active, these signals are active HIGH. After this first clock, they float to 3-state OFF.			
			$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Characteristics
			0(LOW)	0	0	Interrupt Acknowledge
			0	0	1	Read I/O Port
			0	1	0	Write I/O Port
			0	1	1	Halt
			1(HIGH)	0	0	Code Access
			1	0	1	Read Memory
			1	1	0	Write Memory
			1	1	1	Passive

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function															
RQ/GT0, RQ/GT1	30, 31	I/O	<p>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 RQ/GT0 having higher priority than RQ/GT1. RQ/GT has an internal pull-up resistor, so may be left unconnected. The request/grant sequence is as follows (See Figure 8):</p> <ol style="list-style-type: none">1. A pulse of one CLK wide from another local bus master indicates a local bus request ("hold") to the 8088 (pulse 1).2. During a T4 or T1 clock cycle, a pulse one clock wide from the 8088 to the requesting master (pulse 2), indicates that the 8088 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge". The same rules as for HOLD/HOLDA apply as for when the bus is released.3. A pulse one CLK wide from the requesting master indicates to the 8088 (pulse 3) that the "hold" request is about to end and that the 8088 can reclaim the local bus at the next CLK. The CPU then enters T4. <p>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.</p> <p>If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle when all the following conditions are met:</p> <ol style="list-style-type: none">1. Request occurs on or before T2.2. Current cycle is not the low bit of a word.3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.4. A locked instruction is not currently executing. <p>If the local bus is idle when the request is made the two possible events will follow:</p> <ol style="list-style-type: none">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.															
LOCK	29	O	<p>LOCK: indicates that other system bus masters are not to gain control of the system bus while LOCK is active (LOW). The 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".</p>															
QS1, QS0	24, 25	O	<p>QUEUE STATUS: provide status to allow external tracking of the internal 8088 instruction queue. The queue status is valid during the CLK cycle after which the queue operation is performed.</p> <table><tr><th>QS1</th><th>QS0</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>	QS1	QS0	Characteristics	0(LOW)	0	No Operation	0	1	First Byte of Opcode from Queue	1(HIGH)	0	Empty the Queue	1	1	Subsequent Byte from Queue
QS1	QS0	Characteristics																
0(LOW)	0	No Operation																
0	1	First Byte of Opcode from Queue																
1(HIGH)	0	Empty the Queue																
1	1	Subsequent Byte from Queue																
—	34	O	Pin 34 is always high in the maximum mode.															

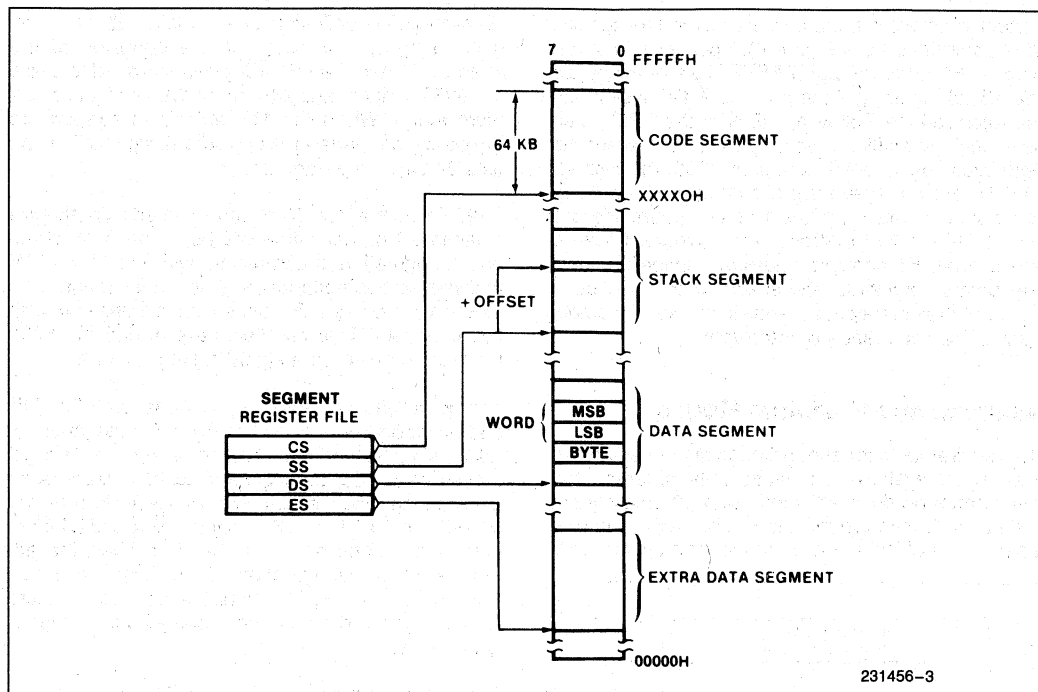


Figure 3. Memory Organization

FUNCTIONAL DESCRIPTION

Memory Organization

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

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the ad-

dressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

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

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

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

Minimum and Maximum Modes

The requirements for supporting minimum and maximum 8088 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8088 is equipped with a strap pin (MN/MX) which defines the system con-

figuration. The definition of a certain subset of the pins changes, dependent on the condition of the strap pin. When the MN/MX pin is strapped to GND, the 8088 defines pins 24 through 31 and 34 in maximum mode. When the MN/MX pin is strapped to V_{CC}, the 8088 generates bus control signals itself on pins 24 through 31 and 34.

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

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

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

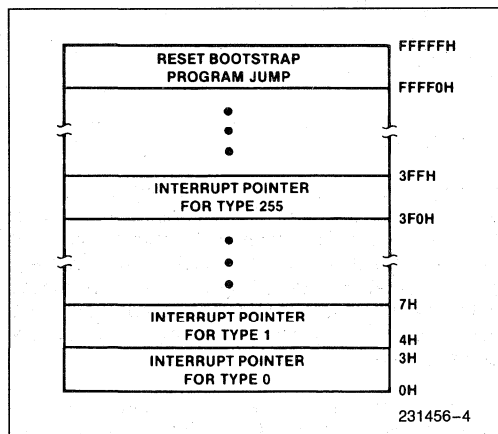


Figure 4. Reserved Memory Locations

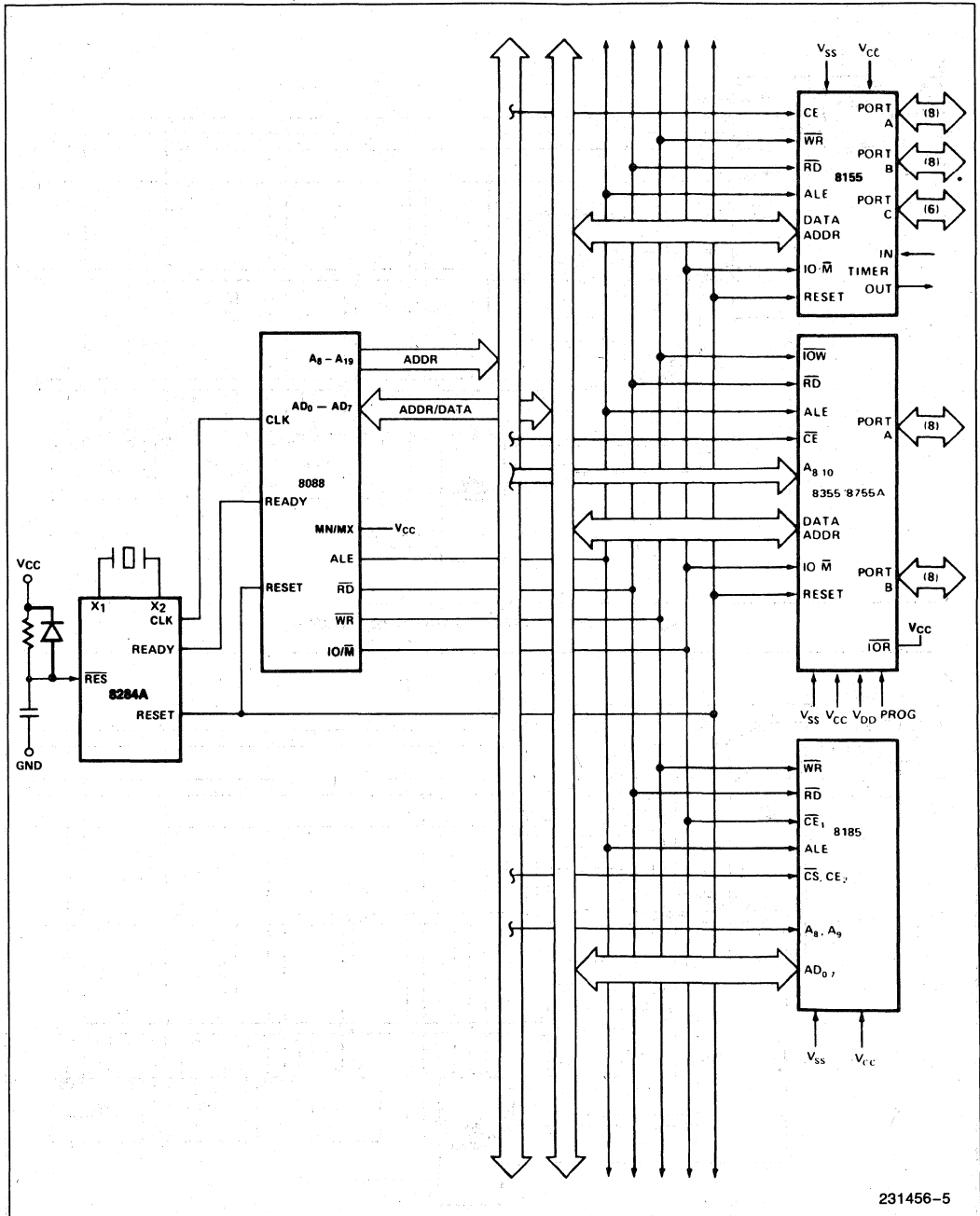


Figure 5. Multiplexed Bus Configuration

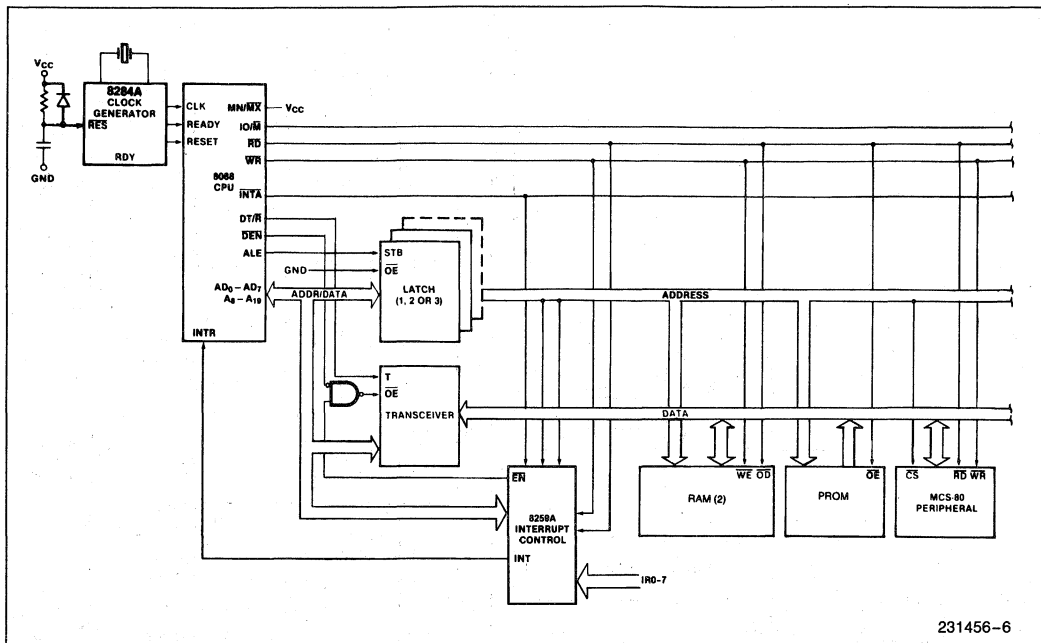


Figure 6. Demultiplexed Bus Configuration

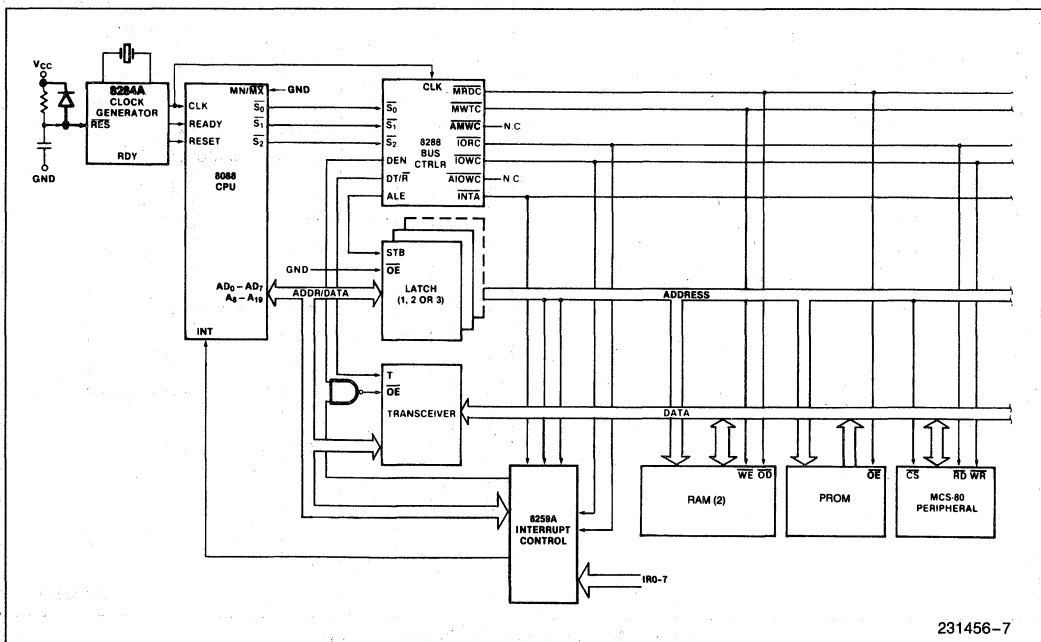


Figure 7. Fully Buffered System Using Bus Controller

Bus Operation

The 8088 address/data bus is broken into three parts—the lower eight address/data bits (AD0–AD7), the middle eight address bits (A8–A15), and the upper four address bits (A16–A19). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor, permitting the use of a standard 40 lead package. The middle eight address bits are not multiplexed, i.e. they remain val-

id throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

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

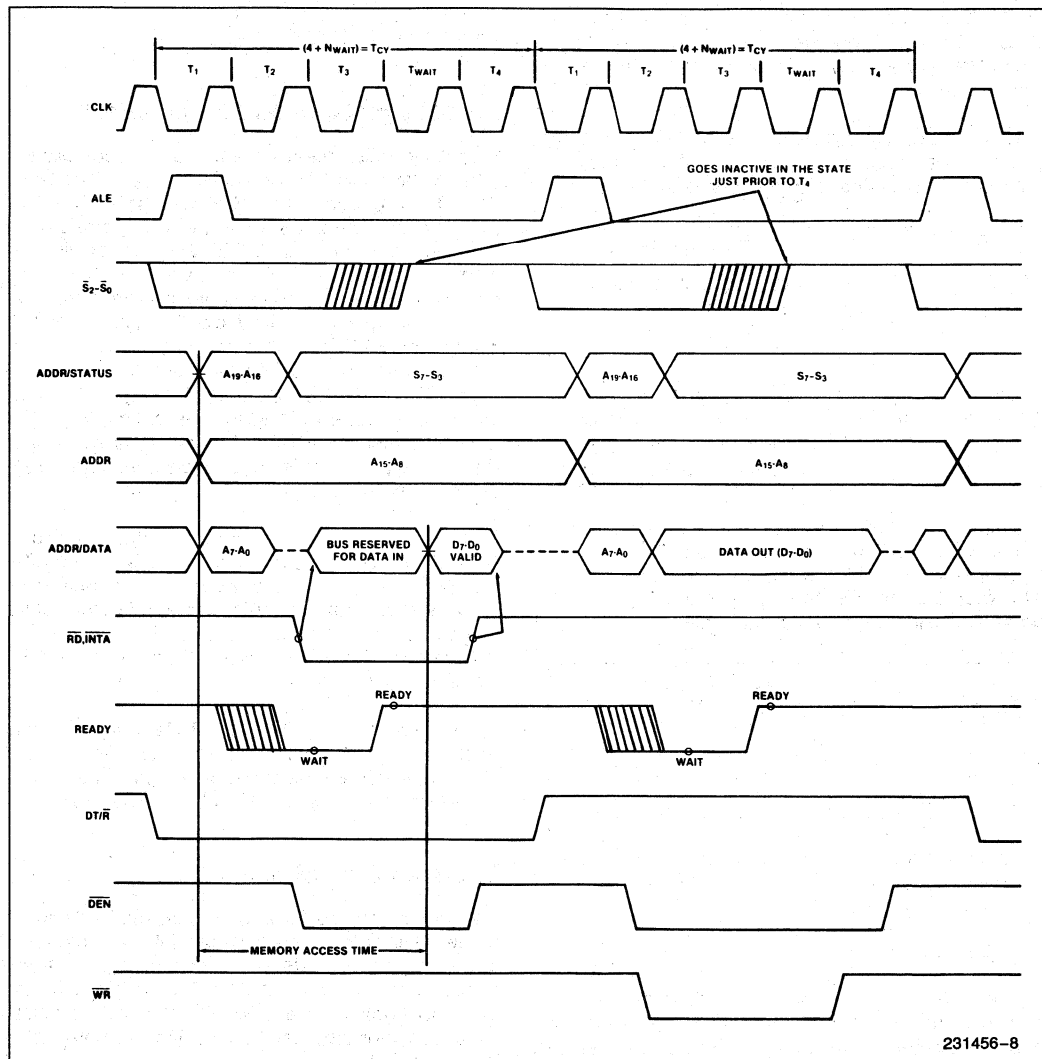


Figure 8. Basic System Timing

231456-8

ing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "wait" states (T_w) are inserted between T_3 and T_4 . Each inserted "wait" state is of the same duration as a CLK cycle. Periods can occur between 8088 driven bus cycles. These are referred to as "idle" states (T_i), or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

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

Status bits $\overline{S_0}$, $\overline{S_1}$, and $\overline{S_2}$ are used 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 S_3 through S_6 are multiplexed with high order address bits and are therefore valid during T_2 through T_4 . S_3 and S_4 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

S_5 is a reflection of the PSW interrupt enable bit. S_6 is always equal to 0.

I/O Addressing

In the 8088, I/O operations can address up to a maximum of 64K I/O registers. The I/O address appears in the same format as the memory address on bus lines $A_{15}-A_0$. 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 on its lower 16 address lines.

EXTERNAL INTERFACE

Processor Reset and Initialization

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

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

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

Interrupt Operations

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

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

Non-Maskable Interrupt (NMI)

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

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

Maskable Interrupt (INTR)

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

enable bit will be zero unless specifically set by an instruction.

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

2

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 $\overline{IO/\overline{M}}$, $\overline{DT/\overline{R}}$, 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 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 LOCK is active, a request on a $\overline{RQ/\overline{GT}}$ pin will be recorded, and then honored at the end of the LOCK.

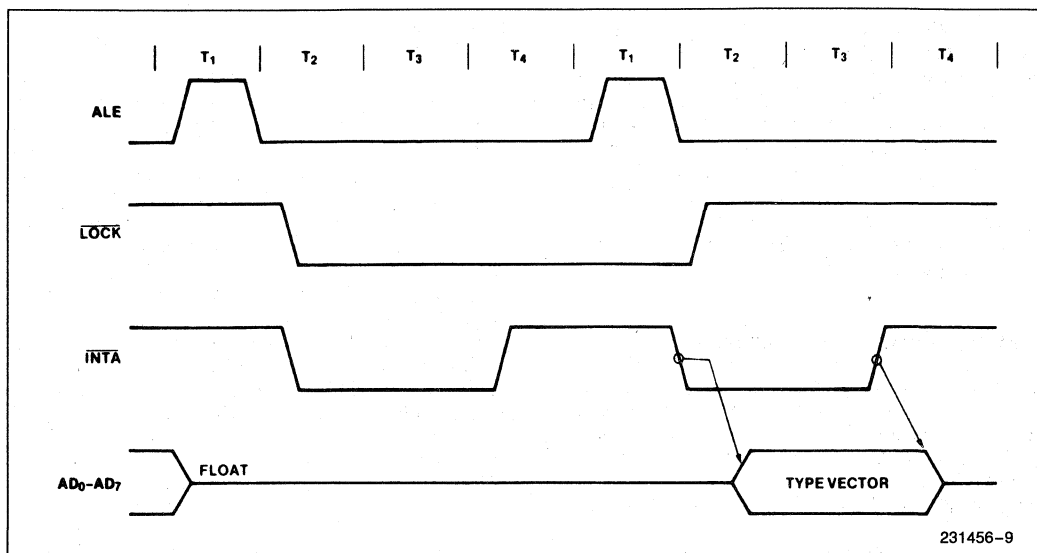


Figure 9. Interrupt Acknowledge Sequence

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

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 3-states all output drivers. If interrupts are enabled, the 8088 will recognize interrupts and process them. The WAIT instruction is then refetched, and reexecuted.

Basic System Timing

In minimum mode, the $\text{MN}/\overline{\text{MX}}$ pin is strapped to V_{CC} and the processor emits bus control signals compatible with the 8085 bus structure. In maximum mode, the $\text{MN}/\overline{\text{MX}}$ 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

(See Figure 8)

The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal. The trailing (low

going) edge of this signal is used to latch the address information, which is valid on the address/data bus (AD0-AD7) at this time, into the 8282/8283 latch. Address lines A8 through A15 do not need to be latched because they remain valid throughout the bus cycle. From T1 to T4 the $\text{IO}/\overline{\text{M}}$ 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 3-state its bus drivers. If a transceiver is required to buffer the 8088 local bus, signals $\text{DT}/\overline{\text{R}}$ 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 $\text{IO}/\overline{\text{M}}$ 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 T4, 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 (INTA) signal is asserted in place of the read (RD) signal and the address bus is floated. (See Figure 9) In the second of two successive INTA cycles, a byte of information is read from the data bus, as supplied by the interrupt system logic (i.e. 8259A priority interrupt controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into the interrupt vector lookup table, as described earlier.

Bus Timing—Medium Complexity Systems

(See Figure 10)

For medium complexity systems, the MN/ \overline{MX} pin is connected to GND and the 8288 bus controller is added to the system, as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 8088 is capable of handling. Signals ALE, \overline{DEN} , and DT/R are generated by the 8288 instead of the processor in this configuration, although their timing remains relatively the same. The 8088 status outputs ($\overline{S_2}$, $\overline{S_1}$, 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 is not valid at the leading edge of write. The transceiver receives the usual T and \overline{OE} inputs from the 8288's DT/R and \overline{DEN} outputs.

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 8289A priority interrupt controller is positioned on the local bus, a TTL gate is required to disable the transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll".

The 8088 Compared to the 8086

The 8088 CPU is an 8-bit processor designed around the 8086 internal structure. Most internal functions of the 8088 are identical to the equivalent 8086 functions. The 8088 handles the external bus

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

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

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

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

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

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

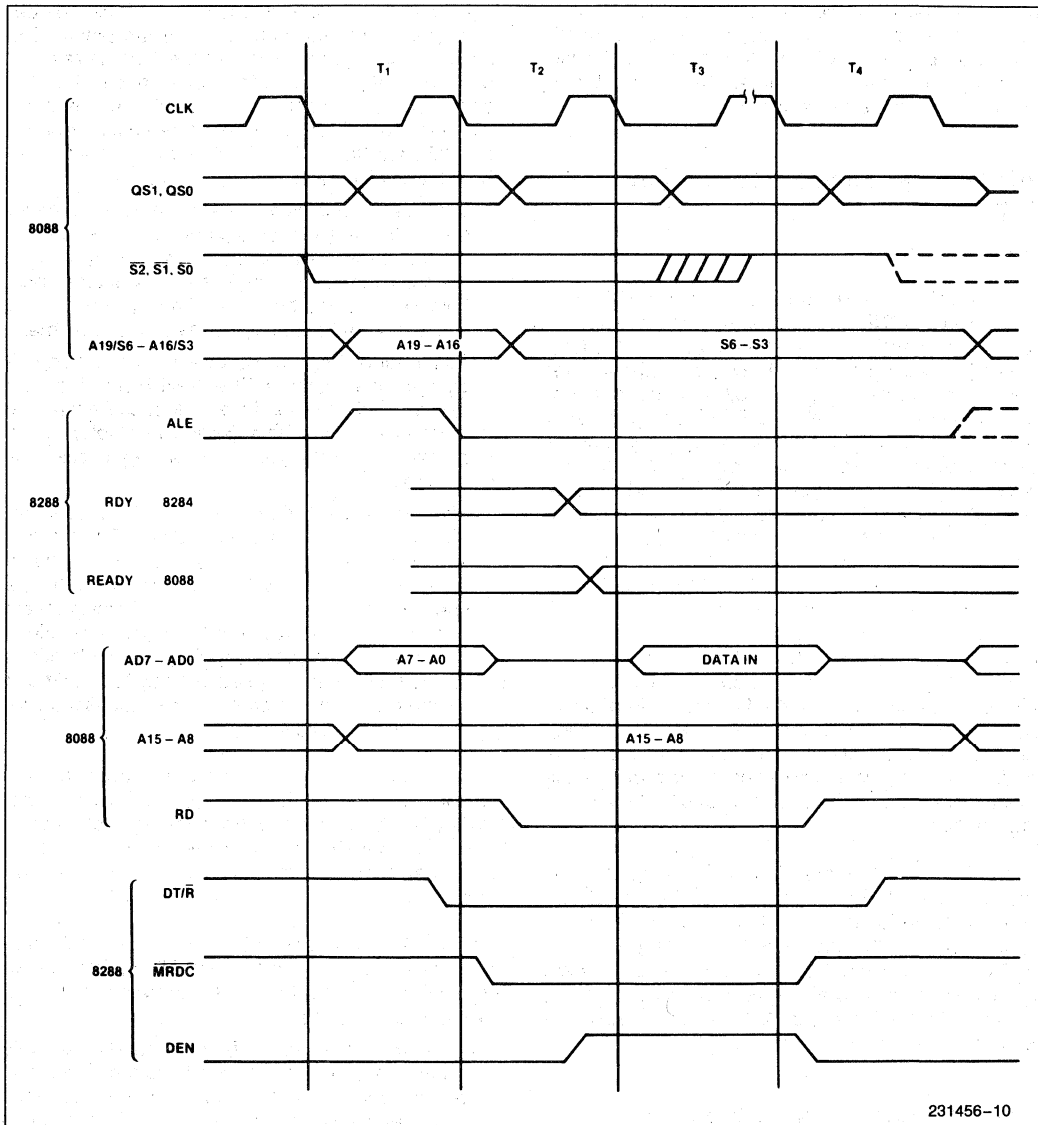


Figure 10. Medium Complexity System Timing

ABSOLUTE MAXIMUM RATINGS*

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

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

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

D.C. CHARACTERISTICS

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

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

Symbol	Parameter	Min	Max	Units	Test Conditions
V _{IL}	Input Low Voltage	-0.5	+0.8	V	(Note 1)
V _{IH}	Input High Voltage	2.0	V _{CC} + 0.5	V	(Notes 1, 2)
V _{OL}	Output Low Voltage		0.45	V	I _{OL} = 2.0 mA
V _{OH}	Output High Voltage	2.4		V	I _{OH} = -400 μA
I _{CC}	8088 Power Supply Current: 8088-2 P8088		340 350 250	mA	T _A = 25°C
I _{LI}	Input Leakage Current		± 10	μA	0V ≤ V _{IN} ≤ V _{CC} (Note 3)
I _{LO}	Output and I/O Leakage Current		± 10	μA	0.45V ≤ V _{OUT} ≤ V _{CC}
V _{CL}	Clock Input Low Voltage	-0.5	+0.6	V	
V _{CH}	Clock Input High Voltage	3.9	V _{CC} + 1.0	V	
C _{IN}	Capacitance If Input Buffer (All Input Except AD ₀ -AD ₇ , RQ/GT)		15	pF	f _c = 1 MHz
C _{IO}	Capacitance of I/O Buffer AD ₀ -AD ₇ , RQ/GT)		15	pF	f _c = 1 MHz

NOTES:

1. V_{IL} tested with MN/M \overline{X} Pin = 0V
V_{IH} tested with MN/M \overline{X} Pin = 5V
MN/M \overline{X} Pin is a strap Pin
2. Not applicable to RQ/GT₀ and RQ/GT₁ Pins (Pins 30 and 31)
3. HOLD and HLDA I_{LI} Min = 30 μA, Max = 500 μA

2

A.C. CHARACTERISTICS

($T_A = 0^\circ\text{C}$ to 70°C , T_{CASE} (Plastic) = 0°C to 95°C , T_{CASE} (CERDIP) = 0°C to 75°C ,

$T_A = 0^\circ\text{C}$ to 55°C and $T_{\text{CASE}} = 0^\circ\text{C}$ to 80°C for P8088-2 only

T_A is guaranteed as long as T_{CASE} is not exceeded)

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

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

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

A.C. CHARACTERISTICS (Continued)

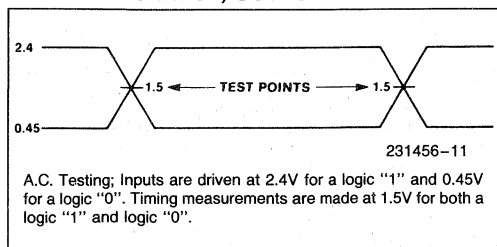
TIMING RESPONSES

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

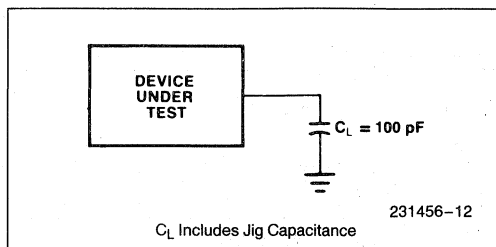
NOTES:

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

A.C. TESTING INPUT, OUTPUT WAVEFORM

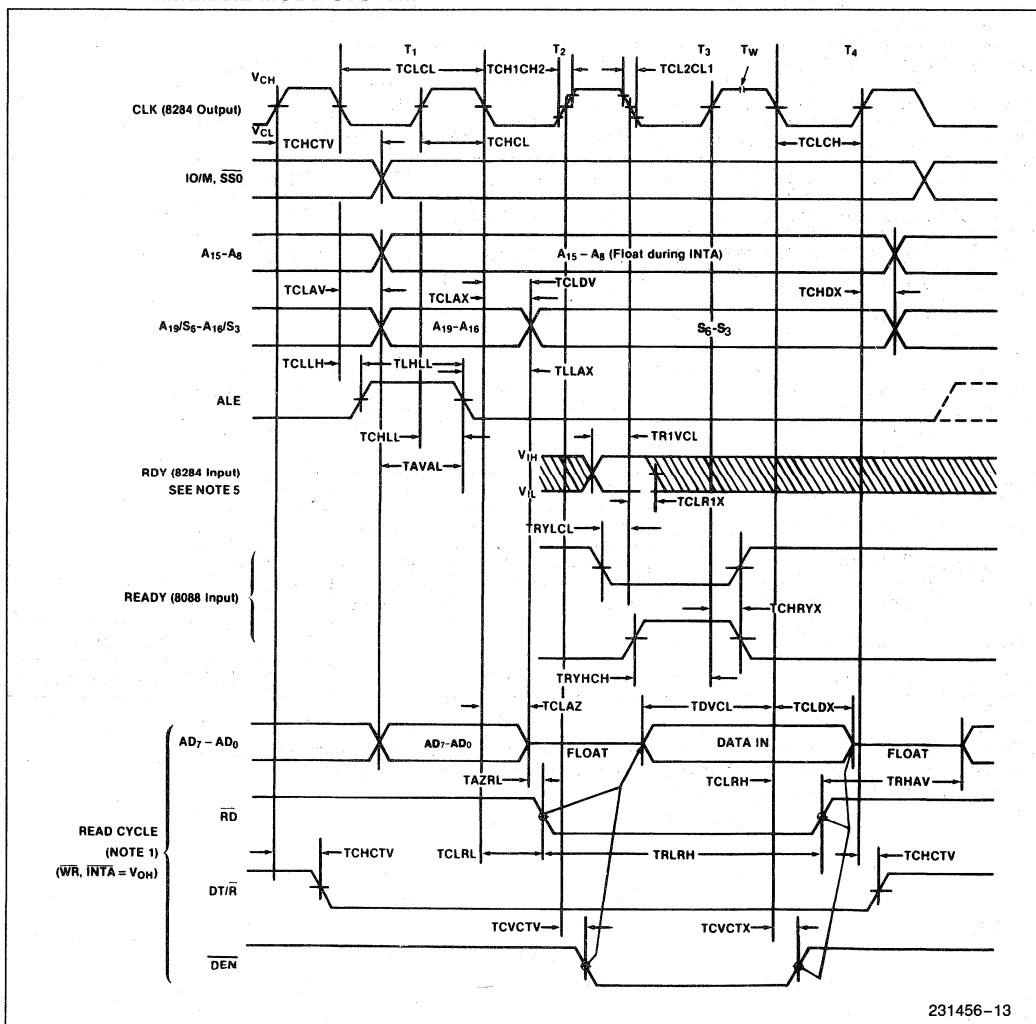


A.C. TESTING LOAD CIRCUIT

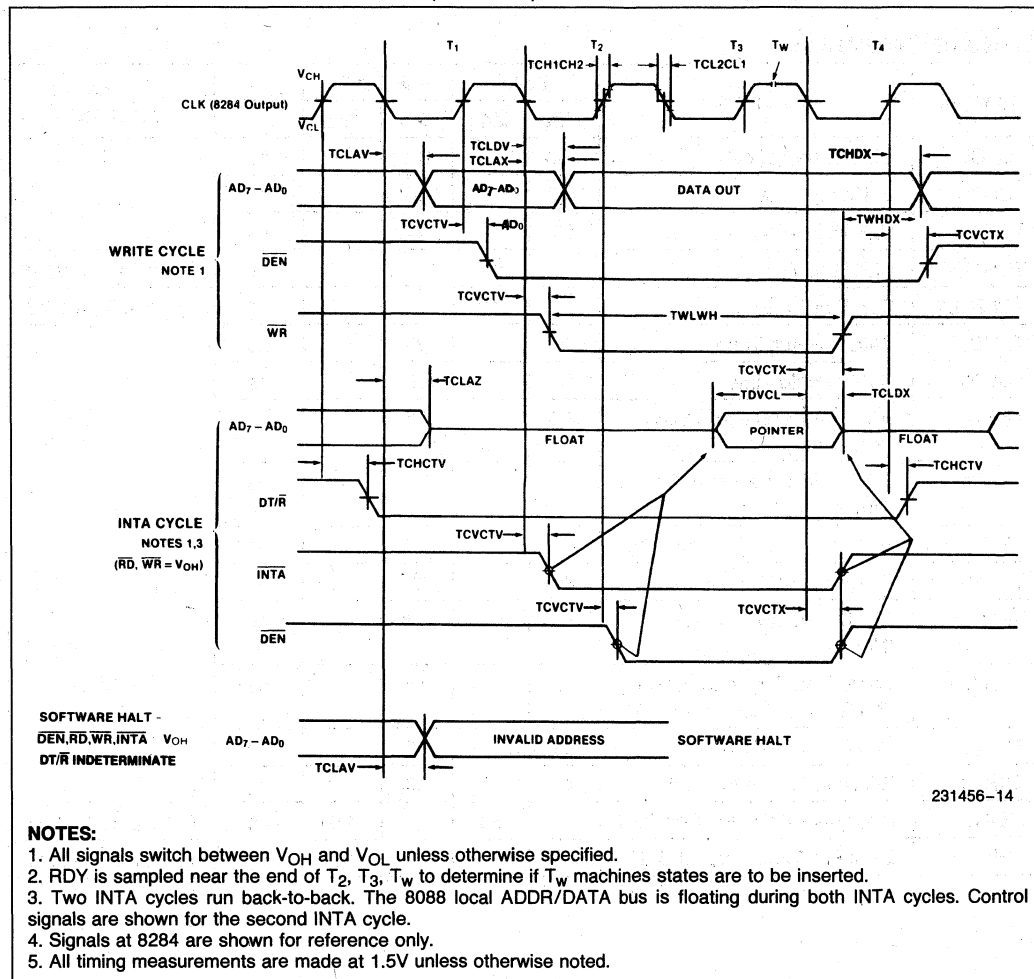


WAVEFORMS

BUS TIMING—MINIMUM MODE SYSTEM



BUS TIMING—MINIMUM MODE SYSTEM (Continued)



A.C. CHARACTERISTICS

MAX MODE SYSTEM (USING 8288 BUS CONTROLLER)

TIMING REQUIREMENTS

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

A.C. CHARACTERISTICS (Continued)

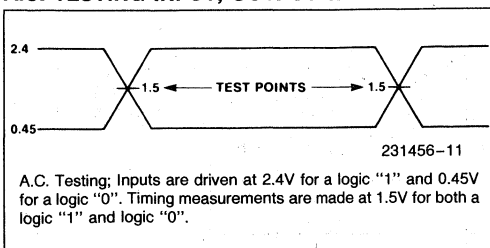
TIMING RESPONSES

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

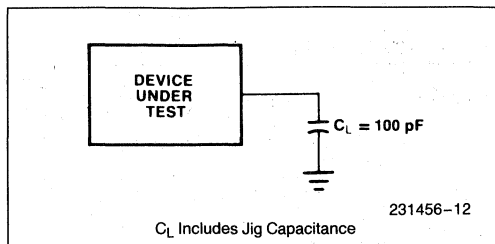
NOTES:

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

A.C. TESTING INPUT, OUTPUT WAVEFORM

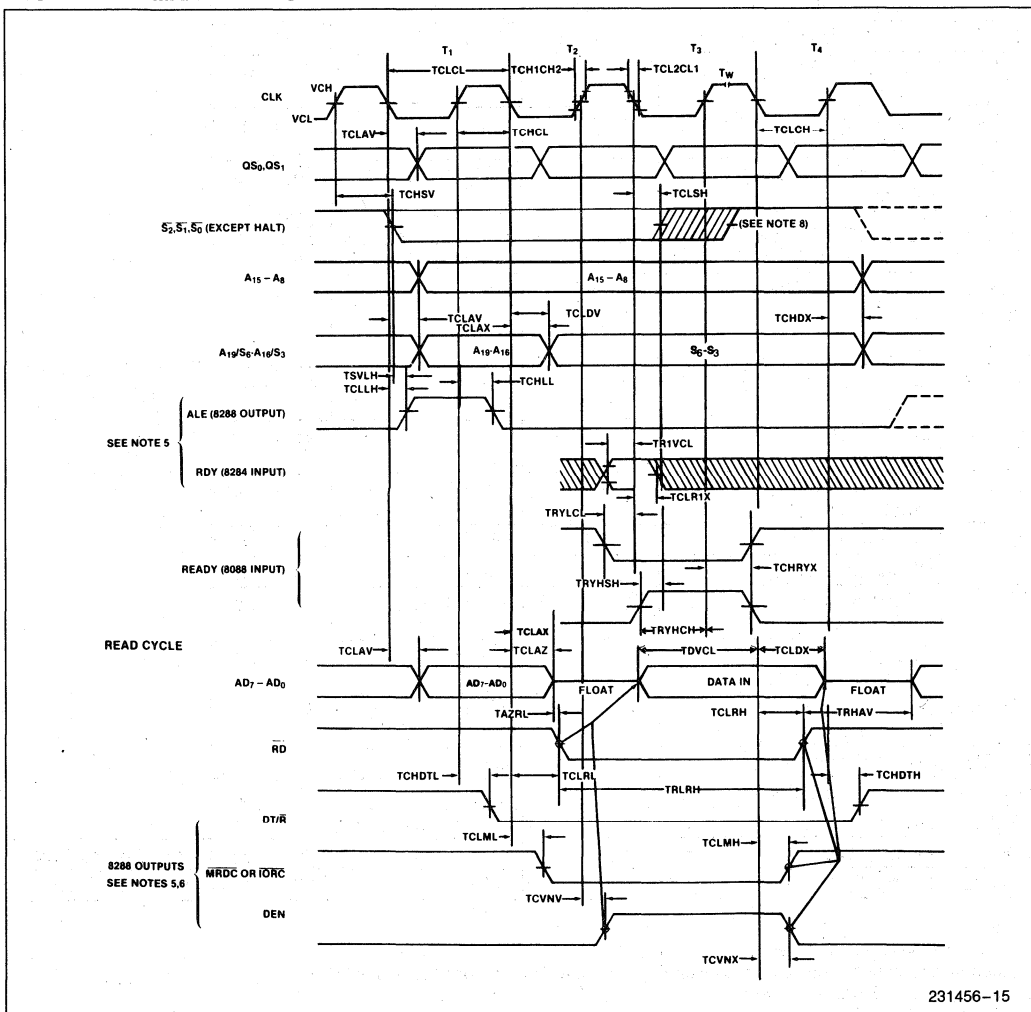


A.C. TESTING LOAD CIRCUIT



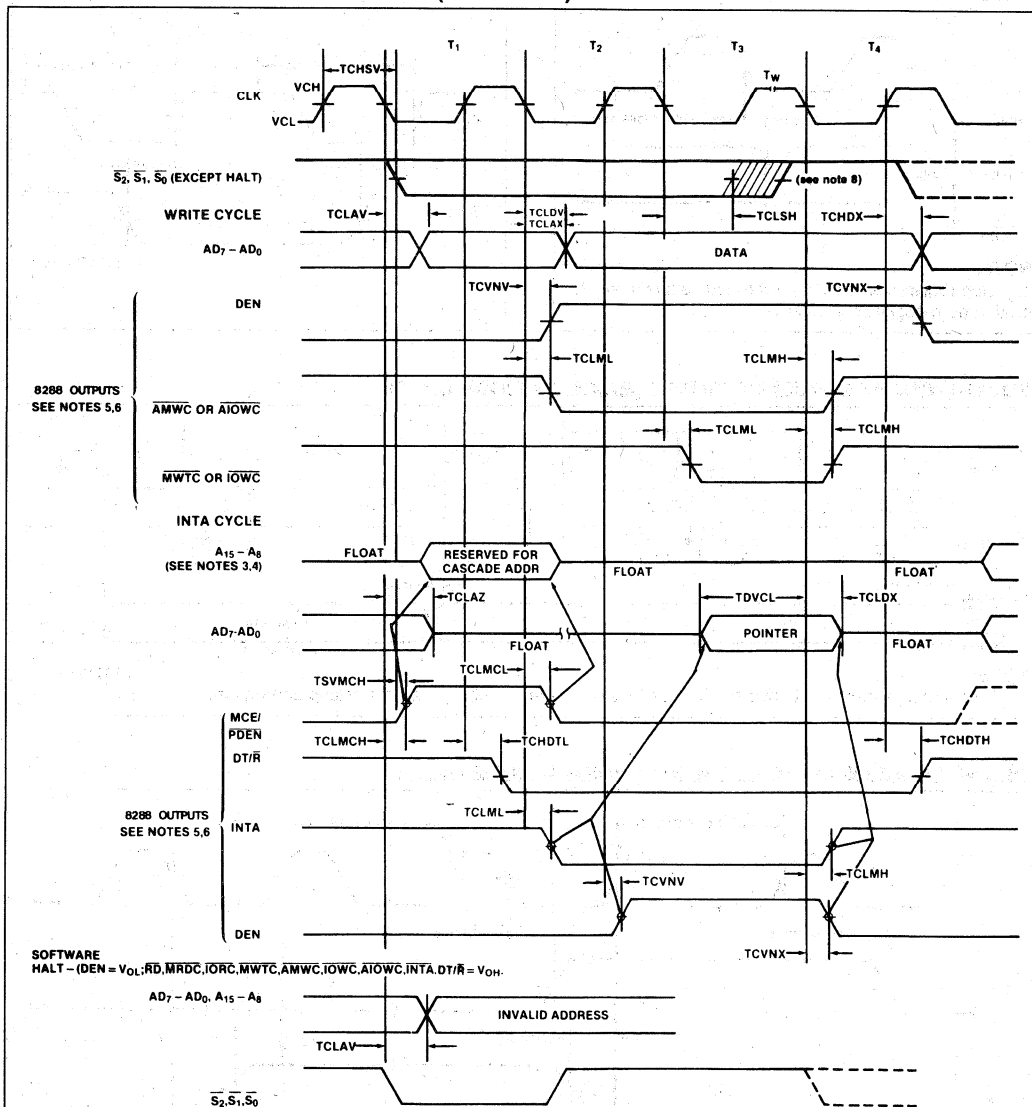
WAVEFORMS (Continued)

BUS TIMING—MAXIMUM MODE SYSTEM



WAVEFORMS (Continued)

BUS TIMING—MAXIMUM MODE SYSTEM (USING 8288)

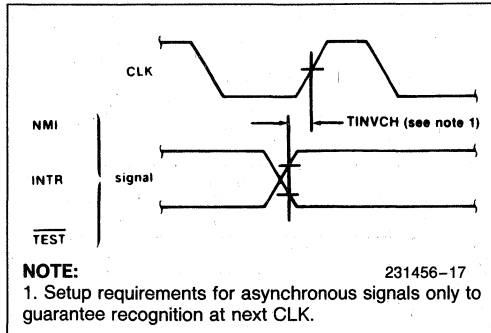


NOTES:

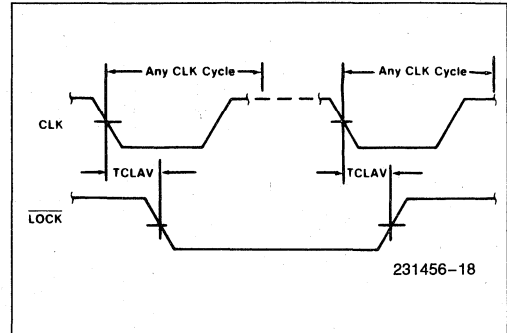
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2 , T_3 , T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycles.
4. Two INTA cycles run back-to-back. The 8088 local ADDR/DATA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

WAVEFORMS (Continued)

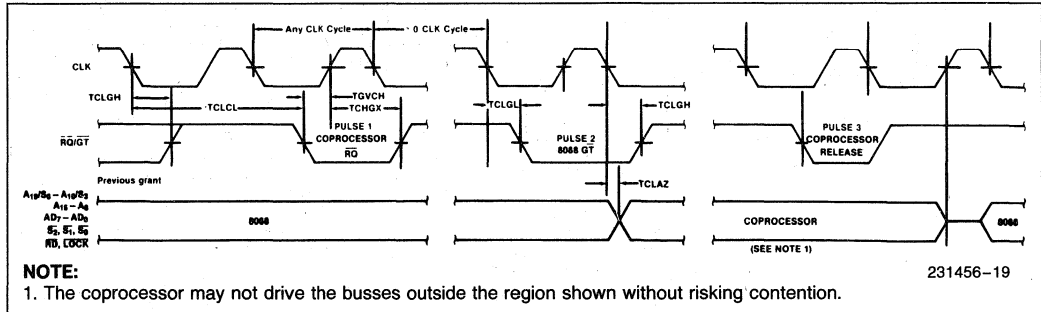
ASYNCHRONOUS SIGNAL RECOGNITION



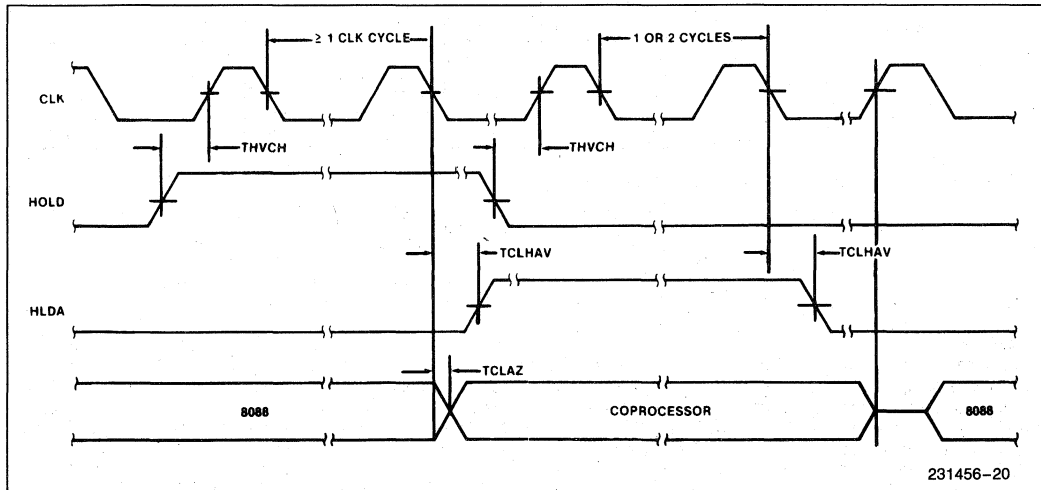
BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



8086/8088 Instruction Set Summary

Mnemonic and Description	Instruction Code			
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 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 = Translate 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		
LAHF = Load AH with Flags	1 0 0 1 1 1 1 1			
SAHF = 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			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
	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
ARITHMETIC				
ADD = Add:				
Reg./Memory with Register to Either	0 0 0 0 0 d w	mod reg r/m		
Immediate to Register/Memory	1 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 d w	mod reg r/m		
Immediate to Register/Memory	1 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			
BAA = 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 d w	mod reg r/m		
Immediate from Register/Memory	1 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	
SSB = Subtract with Borrow				
Reg./Memory and Register to Either	0 0 0 1 1 d w	mod reg r/m		
Immediate from Register/Memory	1 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 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 and Register	0 0 1 1 1 d w	mod reg r/m		
Immediate with Register/Memory	1 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 = Multiply (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 = ASCII 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			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
	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
LOGIC				
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 = Stor Byte/Wd from AL/A	1 0 1 0 1 0 1 w			
CONTROL TRANSFER				
CALL = Call:				
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		

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	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 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 Seg Adding Immed 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	disp	
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	
INT = Interrupt			
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		

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
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 = Escape (to External Device)	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	

NOTES:

AL = 8-bit accumulator

AX = 16-bit accumulator

CX = Count register

DS = Data segment

ES = Extra segment

Above/below refers to unsigned value

Greater = more positive:

Less = less positive (more negative) signed values

if d = 1 then "to" reg; if d = 0 then "from" reg

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

if s:w = 01 then 16 bits of immediate data form the operand

if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand

if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register

x = don't care

z is used for string primitives for comparison with ZF FLAG SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

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

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

FLAGS =

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

Mnemonics © Intel, 1978

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -005 data sheet. Please review this summary carefully.

1. The Intel® 8088 implementation technology (HMOS) has been changed to (HMOS-II).



80C88A 8-BIT CHMOS MICROPROCESSOR

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

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

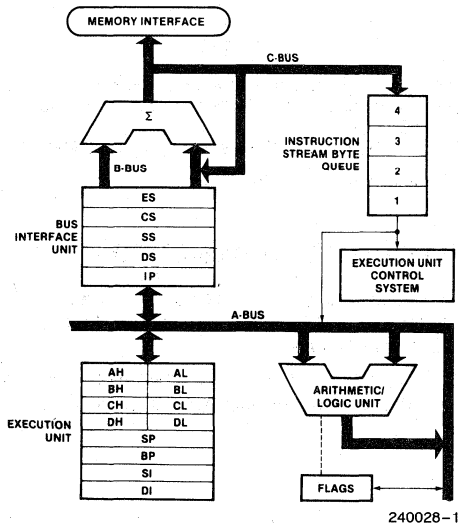


Figure 1. 80C88A CPU
Functional Block Diagram

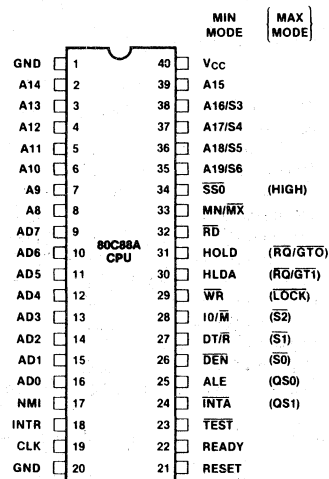


Figure 2. 80C88A 40-Lead
DIP Configuration

Table 1. Pin Description

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

Symbol	Pin No.	Type	Name and Function																				
AD7-AD0	9-16	I/O	ADDRESS DATA BUS: These lines constitute the time multiplexed memory/I/O address (T1) and data (T2, T3, Tw, and T4) bus. These lines are active HIGH and float to 3-state OFF ⁽¹⁾ during interrupt acknowledge and local bus "hold acknowledge".																				
A15-A8	2-8, 39	O	ADDRESS BUS: These lines provide address bits 8 through 15 for the entire bus cycle (T1-T4). These lines do not have to be latched by ALE to remain valid. A15-A8 are active HIGH and float to 3-state OFF ⁽¹⁾ during interrupt acknowledge and local bus "hold acknowledge".																				
A19/S6, A18/S5, A17/S4, A16/S3	35-38	O	ADDRESS/STATUS: During T1, these are the four most significant address lines for memory operations. During I/O operations, these lines are LOW. During memory and I/O operations, status information is available on these lines during T2, T3, Tw, and T4. S6 is always low. The status of the interrupt enable flag bit (S5) is updated at the beginning of each clock cycle. S4 and S3 are encoded as shown. This information indicates which segment register is presently being used for data accessing. These lines float to 3-state OFF ⁽¹⁾ during local bus "hold acknowledge". <table><tr><th>S4</th><th>S3</th><th>CHARACTERISTICS</th></tr><tr><td>0 (LOW)</td><td>0</td><td>Alternate Data</td></tr><tr><td>0</td><td>1</td><td>Stack</td></tr><tr><td>1 (HIGH)</td><td>0</td><td>Code or None</td></tr><tr><td>1</td><td>1</td><td>Data</td></tr><tr><td colspan="3">S6 is 0 (LOW)</td></tr></table>			S4	S3	CHARACTERISTICS	0 (LOW)	0	Alternate Data	0	1	Stack	1 (HIGH)	0	Code or None	1	1	Data	S6 is 0 (LOW)		
S4	S3	CHARACTERISTICS																					
0 (LOW)	0	Alternate Data																					
0	1	Stack																					
1 (HIGH)	0	Code or None																					
1	1	Data																					
S6 is 0 (LOW)																							
\overline{RD}	32	O	READ: Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the IO/M pin or S2. This signal is used to read devices which reside on the 80C88A local bus. \overline{RD} is active LOW during T2, T3 and Tw of any read cycle, and is guaranteed to remain HIGH in T2 until the 80C88A local bus has floated. This signal floats to 3-state OFF ⁽¹⁾ in "hold acknowledge".																				
READY	22	I	READY: is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory or I/O is synchronized by the 82C84A clock generator to form READY. This signal is active HIGH. The 80C88A READY input is not synchronized. Correct operation is not guaranteed if the set up and hold times are not met.																				

Table 1. Pin Description (Continued)

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

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

IO/ \overline{M}	28	O	STATUS LINE: is an inverted maximum mode $\overline{S_2}$. It is used to distinguish a memory access from an I/O access. IO/ \overline{M} becomes valid in the T4 preceding a bus cycle and remains valid until the final T4 of the cycle (I/O = HIGH, M = LOW). IO/ \overline{M} floats to 3-state OFF ⁽¹⁾ in local bus "hold acknowledge".
WR	29	O	WRITE: strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the IO/ \overline{M} signal. WR is active for T2, T3, and Tw of any write cycle. It is active LOW, and floats to 3-state OFF ⁽¹⁾ in local bus "hold acknowledge".
\overline{INTA}	24	O	INTA: is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and Tw of each interrupt acknowledge cycle.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function																																				
ALE	25	O	ADDRESS LATCH ENABLE: is provided by the processor to latch the address into an address latch. It is a HIGH pulse active during clock low of T1 of any bus cycle. Note that ALE is never floated.																																				
DT/ \bar{R}	27	O	DATA TRANSMIT/RECEIVE: is needed in a minimum system that desires to use a data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically, DT/ \bar{R} is equivalent to $\bar{S}1$ in the maximum mode, and its timing is the same as for IO/ \bar{M} (T = HIGH, R = LOW). This signal floats to 3-state OFF ⁽¹⁾ in local "hold acknowledge".																																				
\overline{DEN}	26	O	DATA ENABLE: is provided as an output enable for the transceiver in a minimum system which uses the transceiver. \overline{DEN} is active LOW during each memory and I/O access, and for \overline{INTA} cycles. For a read or \overline{INTA} cycle, it is active from the middle of T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. \overline{DEN} floats to 3-state OFF ⁽¹⁾ during local bus "hold acknowledge".																																				
HOLD, HLDA	30, 31	I, O	HOLD: indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement, in the middle of a T4 or T1 clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor lowers HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. Hold is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the set up time.																																				
$\overline{SS}0$	34	O	<p>STATUS LINE: is logically equivalent to $\overline{S}0$ in the maximum mode. The combination of $\overline{SS}0$, IO/\bar{M} and DT/\bar{R} allows the system to completely decode the current bus cycle status.</p> <table> <tr> <th>IO/\bar{M}</th><th>DT/\bar{R}</th><th>$\overline{SS}0$</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>	IO/ \bar{M}	DT/ \bar{R}	$\overline{SS}0$	CHARACTERISTICS	1(HIGH)	0	0	Interrupt Acknowledge	1	0	1	Read I/O port	1	1	0	Write I/O port	1	1	1	Halt	0(LOW)	0	0	Code access	0	0	1	Read memory	0	1	0	Write memory	0	1	1	Passive
IO/ \bar{M}	DT/ \bar{R}	$\overline{SS}0$	CHARACTERISTICS																																				
1(HIGH)	0	0	Interrupt Acknowledge																																				
1	0	1	Read I/O port																																				
1	1	0	Write I/O port																																				
1	1	1	Halt																																				
0(LOW)	0	0	Code access																																				
0	0	1	Read memory																																				
0	1	0	Write memory																																				
0	1	1	Passive																																				

Table 1. Pin Description (Continued)

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

Symbol	Pin No.	Type	Name and Function			
$\overline{S2}, \overline{S1}, \overline{S0}$	26-28	O	<p>STATUS: is active during clock high of T4, T1, and T2, and is returned to the passive state (1,1,1) during T3 or during Tw when READY is HIGH. This status is used by the 82C88 bus controller to generate all memory and I/O access control signals. Any change by $\overline{S2}, \overline{S1}$, or $\overline{S0}$ during T4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T3 or Tw is used to indicate the end of a bus cycle.</p> <p>These signals float to 3-state OFF(1) during "hold acknowledge". During the first clock cycle after RESET becomes active, these signals are active HIGH. After this first clock, they float to 3-state OFF.</p>			
			$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	CHARACTERISTICS
			0 (LOW)	0	0	Interrupt Acknowledge
			0	0	1	Read I/O port
			0	1	0	Write I/O port
			0	1	1	Halt
			1 (HIGH)	0	0	Code access
			1	0	1	Read memory
$\overline{RQ}/\overline{GT0}, \overline{RQ}/\overline{GT1}$	30, 31	I/O	1	1	0	Write memory
			1	1	1	Passive
			<p>REQUEST/GRANT: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT0}$ having higher priority than $\overline{RQ}/\overline{GT1}$. $\overline{RQ}/\overline{GT}$ has an internal pull-up resistor, so may be left unconnected. The request/grant sequence is as follows (see timing diagram):</p> <ol style="list-style-type: none"> 1. A pulse of one CLK wide from another local bus master indicates a local bus request ("hold") to the 80C88A (pulse 1). 2. During a T4 or T1 clock cycle, a pulse one clock wide from the 80C88A to the requesting master (pulse 2), indicates that the 80C88A has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge". The same rules as for HOLD/HOLDA apply as for when the bus is released. 3. A pulse one CLK wide from the requesting master indicates to the 80C88A (pulse 3) that the "hold" request is about to end and that the 80C88A can reclaim the local bus at the next CLK. The CPU then enters T4. 			

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function															
$\overline{RQ/GT0}$, $\overline{RQ/GT1}$	30, 31	I/O	<p>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.</p> <p>If the request is made while the CPU is performing a memory cycle, it will release the local bus during T4 of the cycle when all the following conditions are met:</p> <ol style="list-style-type: none">1. Request occurs on or before T2.2. Current cycle is not the low bit of a word.3. Current cycle is not the first acknowledge of an interrupt acknowledge sequence.4. A locked instruction is not currently executing. <p>If the local bus is idle when the request is made the two possible events will follow:</p> <ol style="list-style-type: none">1. Local bus will be released during the next clock.2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied.															
\overline{LOCK}	29	O	<p>LOCK: indicates that other system bus masters are not to gain control of the system bus while \overline{LOCK} is active (LOW). The \overline{LOCK} signal is activated by the “LOCK” prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF(1) in “hold acknowledge”.</p>															
QS1, QS0	24, 25	O	<p>QUEUE STATUS: provide status to allow external tracking of the internal 80C88A instruction queue.</p> <p>The queue status is valid during the CLK cycle after which the queue operation is performed.</p> <table><tr><th>QS1</th><th>QS0</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>	QS1	QS0	CHARACTERISTICS	0(LOW)	0	No operation	0	1	First byte of opcode from queue	1(HIGH)	0	Empty the queue	1	1	Subsequent byte from queue
QS1	QS0	CHARACTERISTICS																
0(LOW)	0	No operation																
0	1	First byte of opcode from queue																
1(HIGH)	0	Empty the queue																
1	1	Subsequent byte from queue																
—	34	O	Pin 34 is always high in the maximum mode.															

NOTE:

1. See the section on Bus Hold Circuitry.

FUNCTIONAL DESCRIPTION

STATIC OPERATION

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

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

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

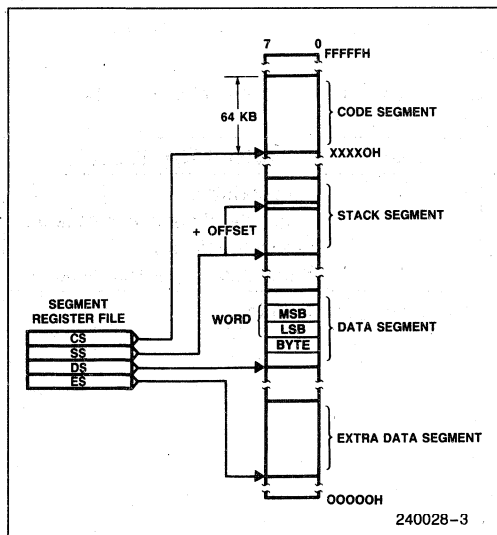


Figure 3. Memory Organization

MEMORY ORGANIZATION

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

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

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

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

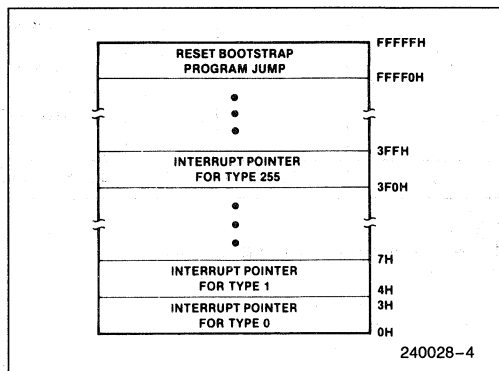


Figure 4. Reserved Memory Locations

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

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

MINIMUM AND MAXIMUM MODES

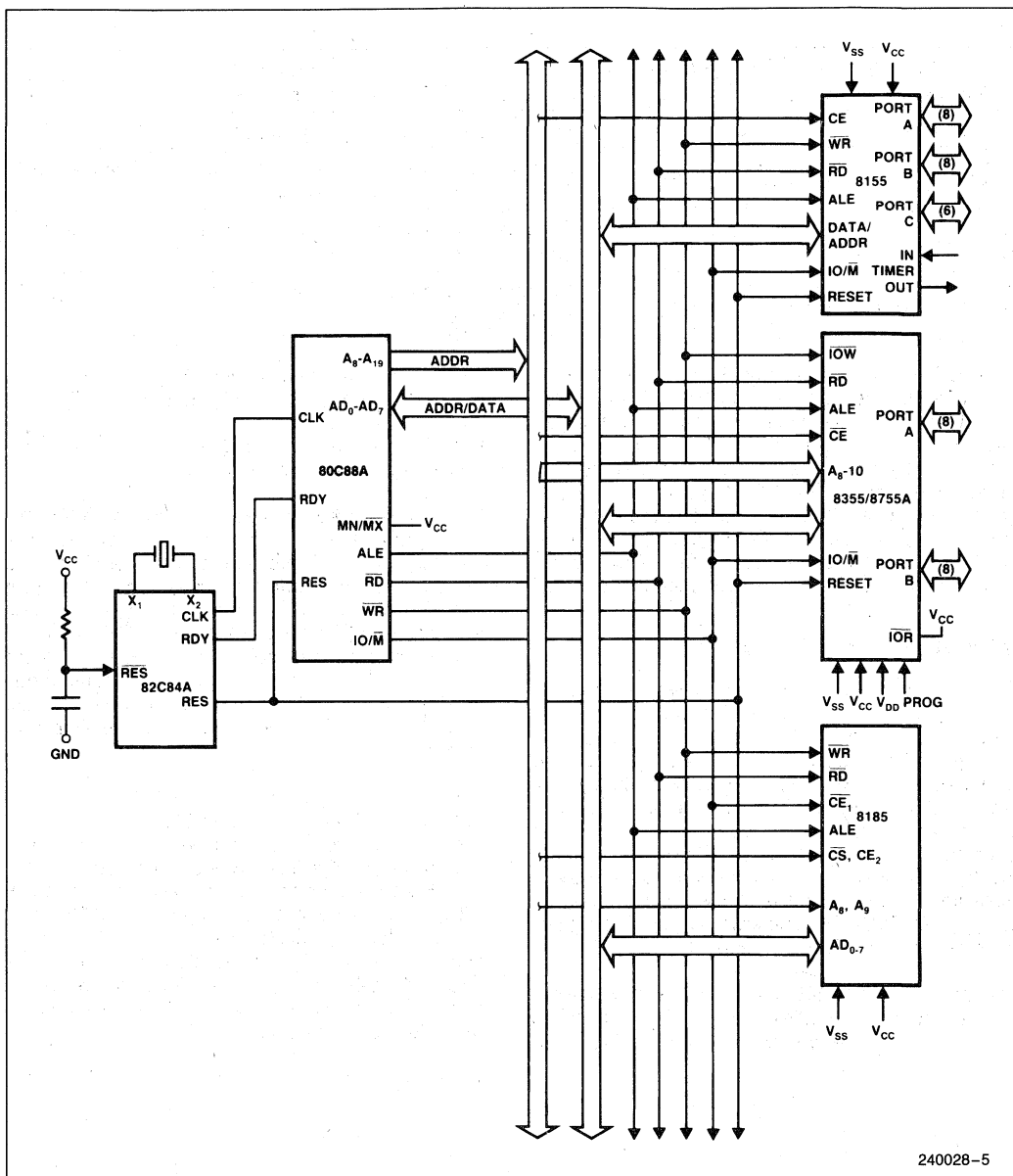
The requirements for supporting minimum and maximum 80C88A systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 80C88A is equipped with a strap pin (MN/MX) which defines the system configuration. The definition of a certain subset of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

The minimum mode 80C88A can be used with either a multiplexed or demultiplexed bus. The multiplexed bus configuration is compatible with the MCS®-85

multiplexed bus peripherals (8155, 8156, 8355, 8755A, and 8185). This configuration (See Figure 5) provides the user with a minimum chip count system. This architecture provides the 80C88A processing power in a highly integrated form.

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

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



240028-5

Figure 5. Multiplexed Bus Configuration

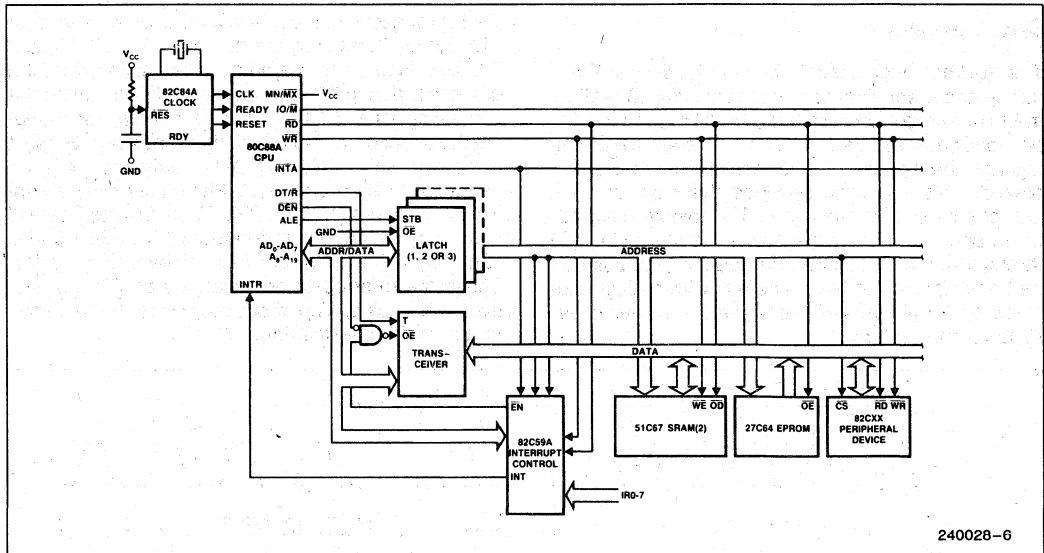


Figure 6. Demultiplexed Bus Configuration

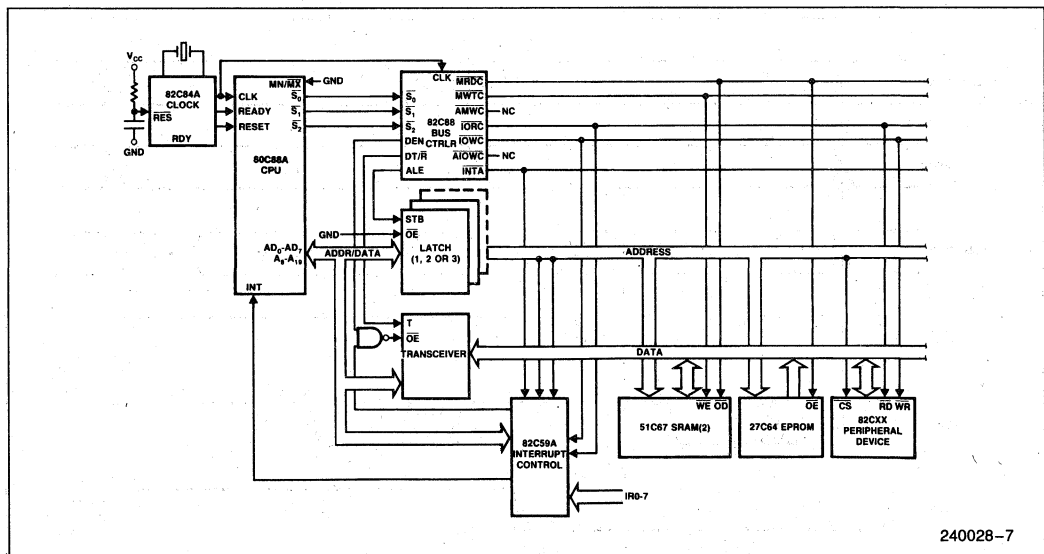


Figure 7. Fully Buffered System Using Bus Controller

Bus Operation

The 80C88A address/data bus is broken into three parts—the lower eight address/data bits (A0–A7), the middle eight address bits (A8–A15), and the upper four address bits (A16–A19). The address/data bits and the highest four address bits are time multiplexed. This technique provides the most efficient use of pins on the processor. The middle eight address bits are not multiplexed, i.e. they remain valid throughout each bus cycle. In addition, the bus can be demultiplexed at the processor with a single address latch if a standard, non-multiplexed bus is desired for the system.

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

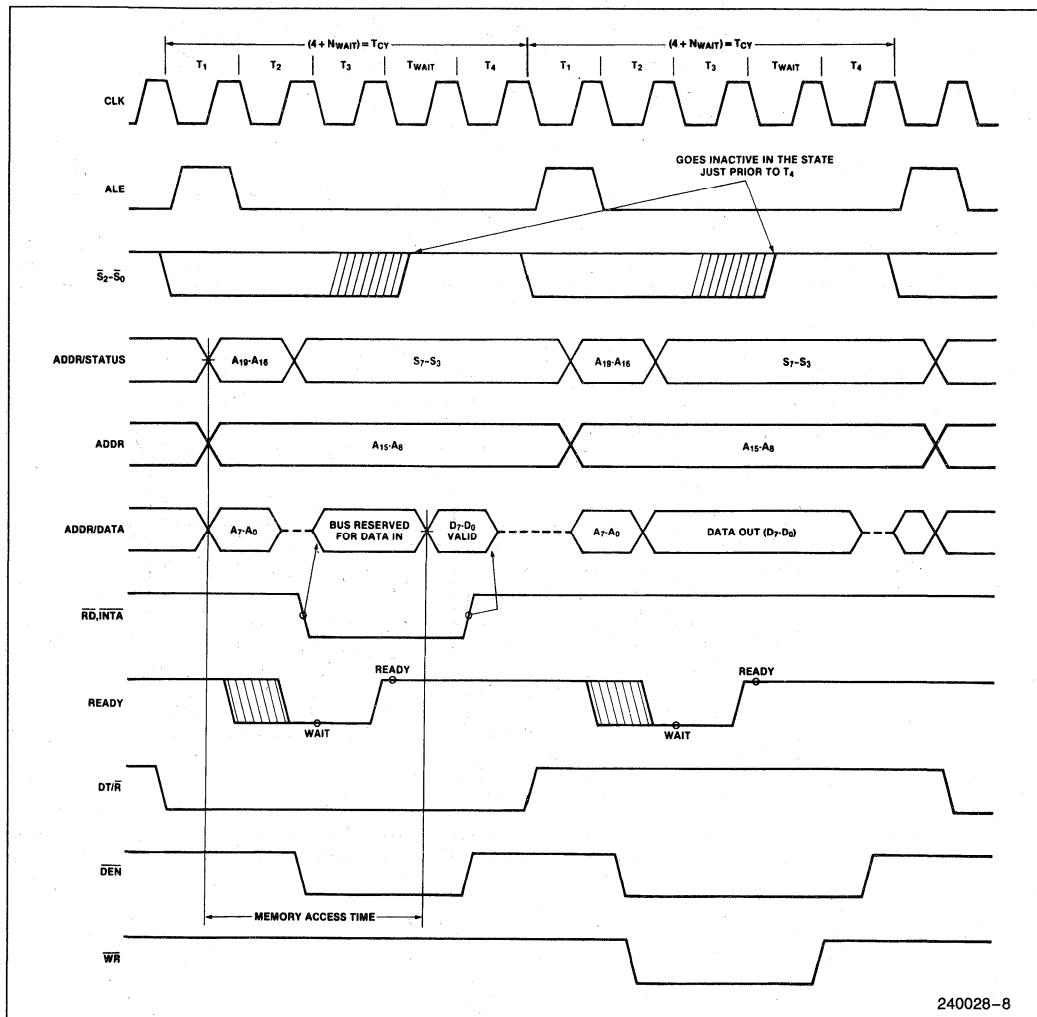


Figure 8. Basic System Timing

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

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

\bar{S}_2	\bar{S}_1	\bar{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:

S4	S3	CHARACTERISTICS
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

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

I/O ADDRESSING

In the 80C88A, I/O operations can address up to a maximum of 64k I/O registers. The I/O address appears in the same format as the memory address on bus lines A15–A0. The address lines A19–A16 are zero in I/O operations. The variable I/O instructions, which use register DX as a pointer, have full address

capability, while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space. I/O ports are addressed in the same manner as memory locations.

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

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

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

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

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

NOTE:

1. See the section on Bus Hold Circuitry.

BUS HOLD CIRCUITRY

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

current is negligible and power dissipation is significantly reduced when compared to the use of passive pull-up resistors.

INTERRUPT OPERATIONS

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

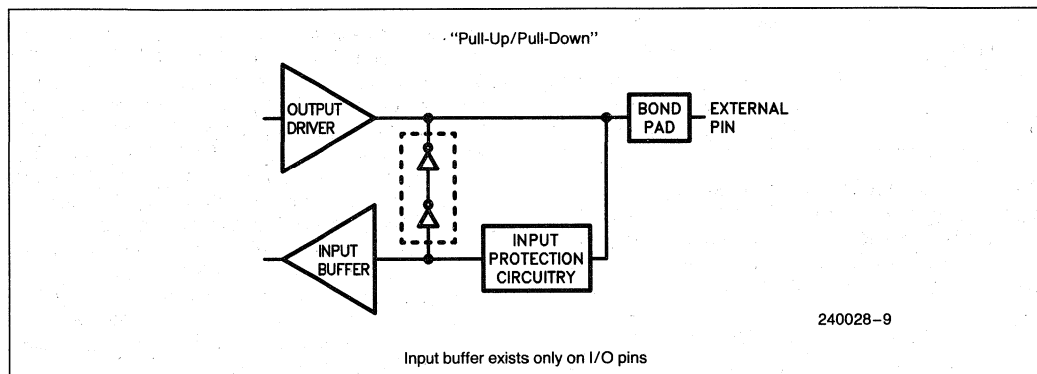


Figure 9a. Bus hold circuitry pin 2-16, 35-39.

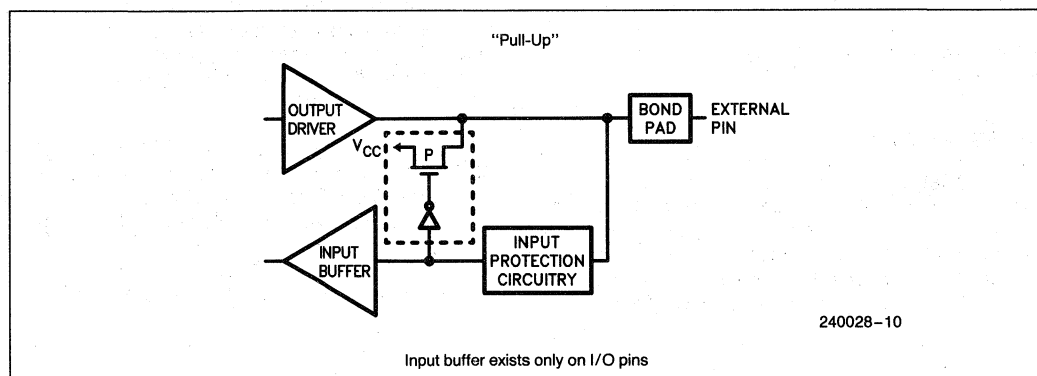


Figure 9b. Bus hold circuitry pin 26-32, 34.

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

NON-MASKABLE INTERRUPT (NMI)

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

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

be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

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

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

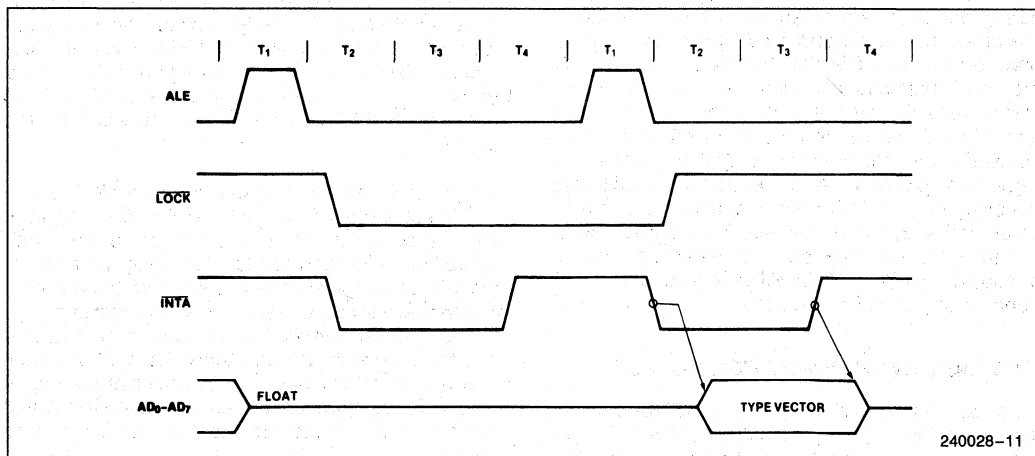


Figure 10. Interrupt Acknowledge Sequence

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

HALT

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

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

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

EXTERNAL SYNCHRONIZATION VIA \overline{TEST}

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

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

BASIC SYSTEM TIMING

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

System Timing — Minimum System

(See Figure 8.)

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

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

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

BUS TIMING — MEDIUM COMPLEXITY SYSTEMS

(See Figure 11.)

For medium complexity systems, the MN/M \overline{X} pin is connected to GND and the 82C88 bus controller is added to the system, as well as a latch for latching the system address, and a transceiver to allow for bus loading greater than the 80C88A is capable of handling. Signals ALE, DEN, and DT/R are generated by the 82C88 instead of the processor in this configuration, although their timing remains relatively the same. The 80C88A status outputs (S $\overline{2}$, S $\overline{1}$, and S $\overline{0}$) provide type of cycle information and become 82C88 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 82C88 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 82C88 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence, data is not valid at the leading edge of write. The transceiver receives the usual T and OE inputs from the 82C88's DT/R and DEN outputs.

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

THE 80C88A COMPARED TO THE 80C86

The 80C88A CPU is an 8-bit processor designed around the 80C86 internal structure. Most internal functions of the 80C88A are identical to the equivalent

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

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

The 80C88A and 80C86 are completely software compatible by 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 80C88A or an 80C86.

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

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

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

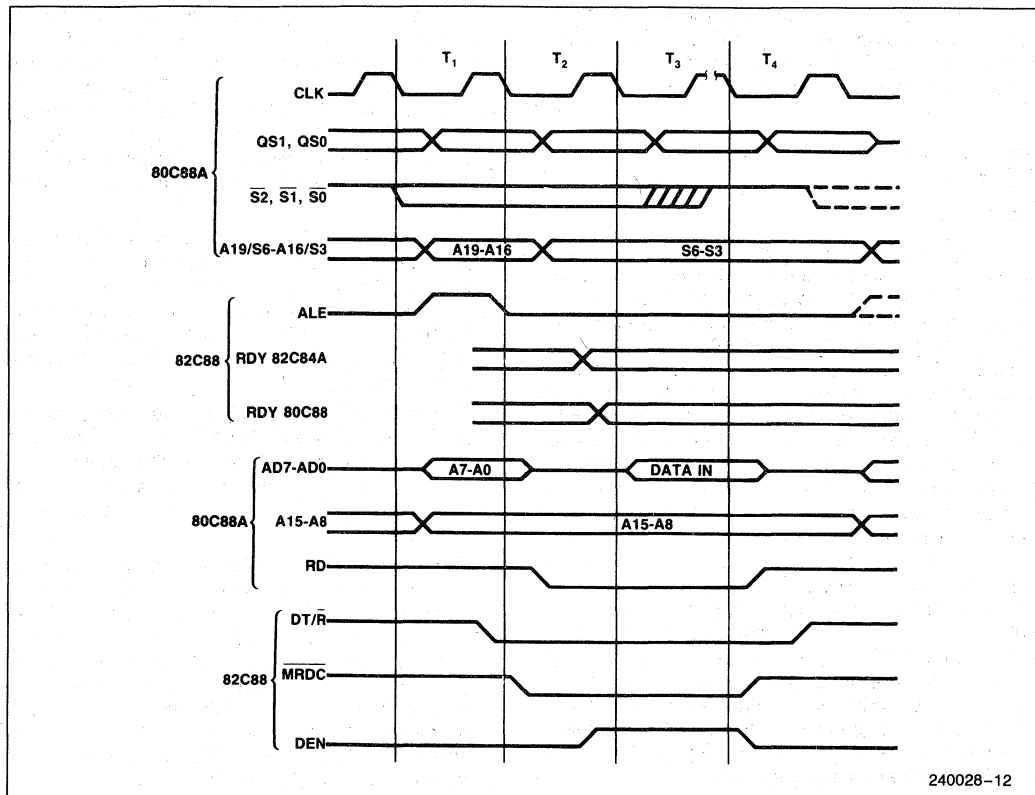


Figure 11. Medium Complexity System Timing

240028-12

ABSOLUTE MAXIMUM RATINGS*

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

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

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

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

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

NOTES:

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

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

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

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

A.C. CHARACTERISTICS (Continued)

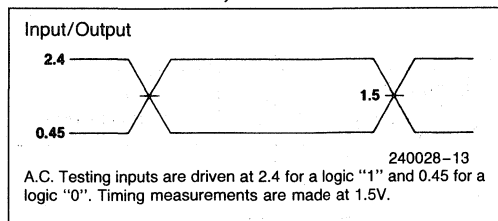
TIMING RESPONSES

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

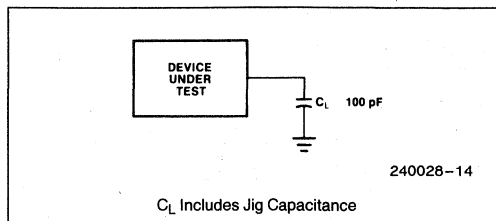
NOTES:

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

A.C. TESTING INPUT, OUTPUT WAVEFORM

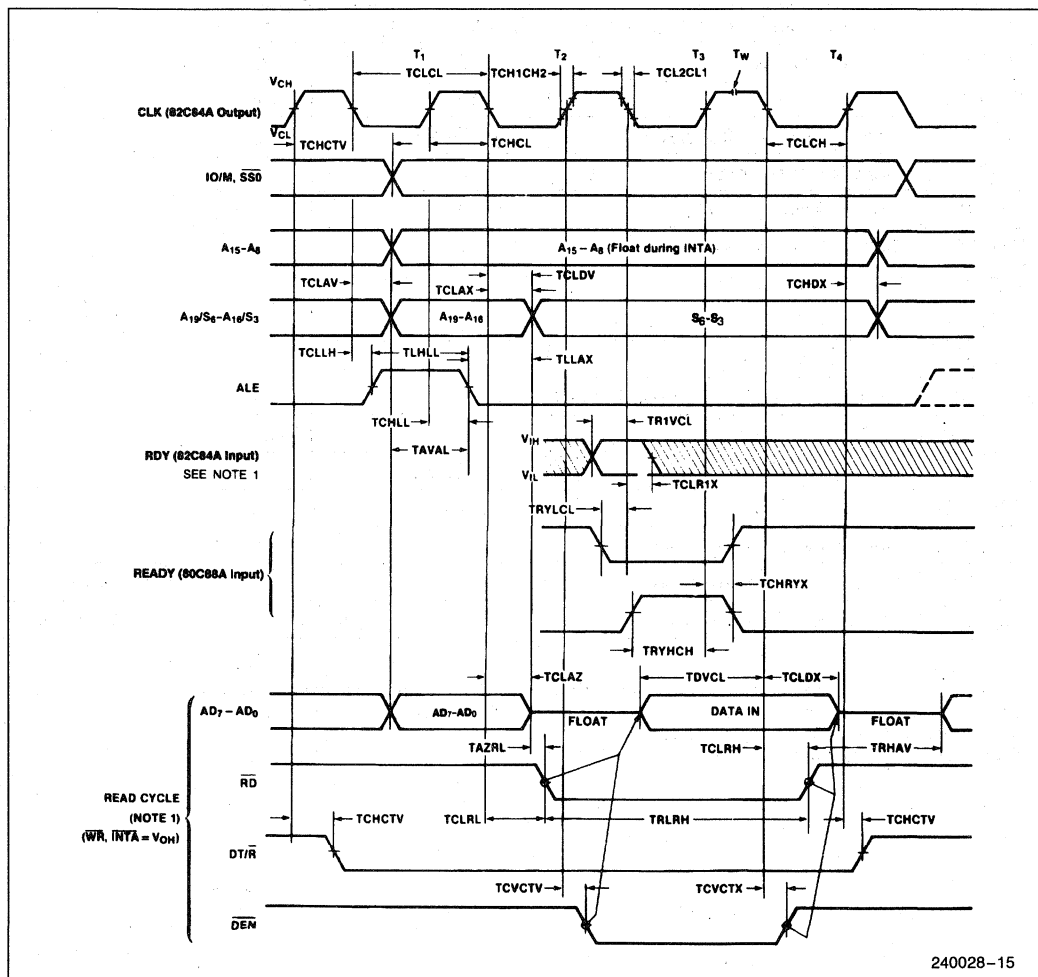


A.C. TESTING LOAD CIRCUIT



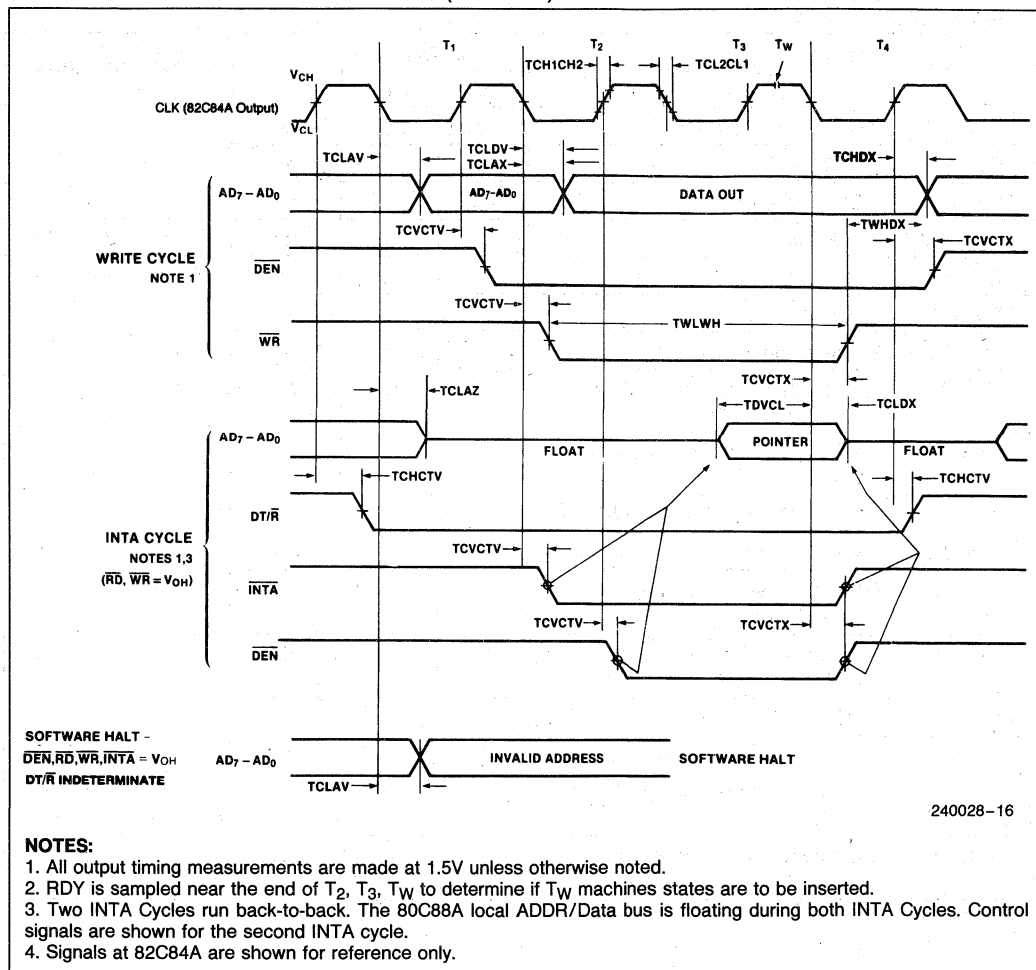
WAVEFORMS

BUS TIMING — MINIMUM MODE SYSTEM



WAVEFORMS (Continued)

BUS TIMING — MINIMUM MODE SYSTEM (Continued)



A.C. CHARACTERISTICS

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

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

A.C. CHARACTERISTICS

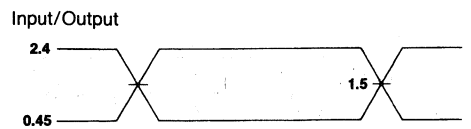
TIMING RESPONSES

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

NOTES:

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

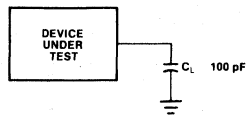
A.C. TESTING INPUT, OUTPUT WAVEFORM



240028-17

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

A.C. TESTING LOAD CIRCUIT

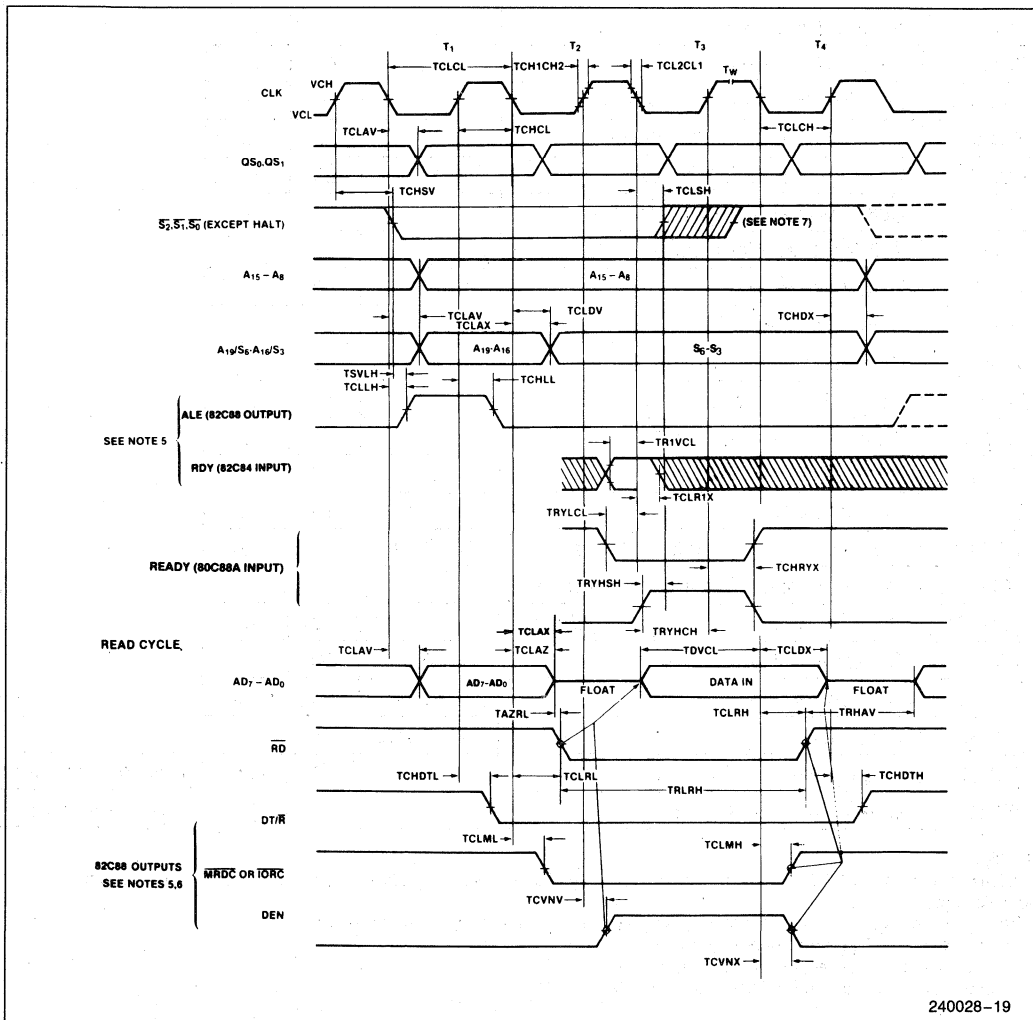


240028-18

C_L Includes Jig Capacitance

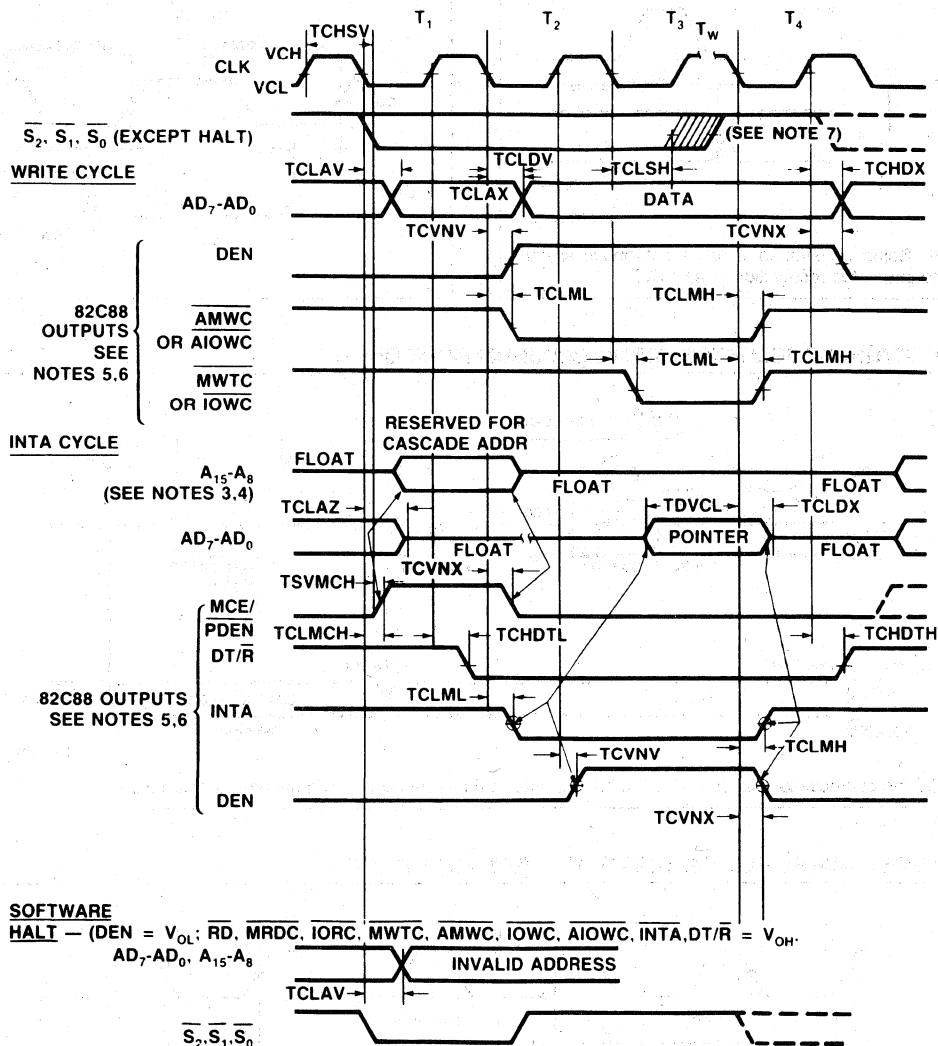
WAVEFORMS

BUS TIMING—MAXIMUM MODE



WAVEFORMS (Continued)

BUS TIMING — MAXIMUM MODE SYSTEM (USING 82C88)



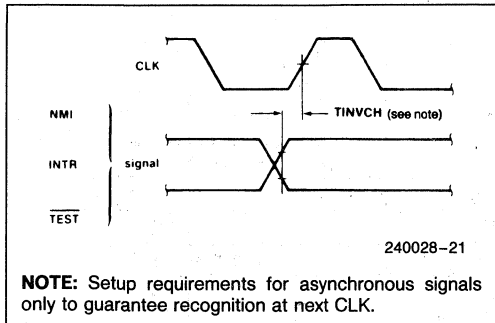
240028-20

NOTES:

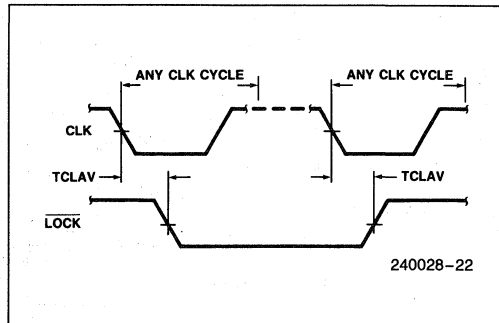
1. All output timing measurements are made at 1.5V unless otherwise noted.
2. RDY is sampled near the end of T_2 , T_3 , T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycles.
4. Two INTA cycles run back-to-back. The 80C88A local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

WAVEFORMS (Continued)

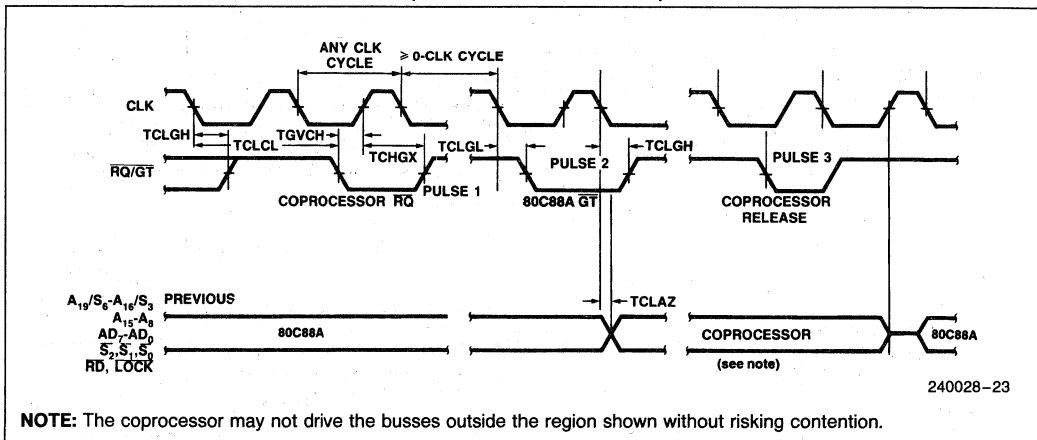
ASYNCHRONOUS SIGNAL RECOGNITION



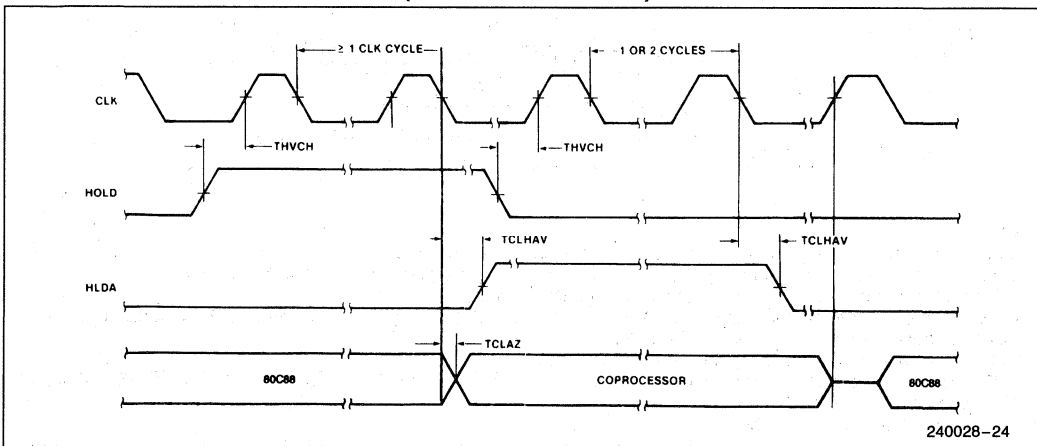
BUS LOCK SIGNAL TIMING (MAXIMUM MODE ONLY)



REQUEST/GRANT SEQUENCE TIMING (MAXIMUM MODE ONLY)



HOLD/HOLD ACKNOWLEDGE TIMING (MINIMUM MODE ONLY)



80C86A/80C88A INSTRUCTION SET SUMMARY

Mnemonic and Description	Instruction Code			
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	add-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 = Translate 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		
LAHF = Load AH with Flags	1 0 0 1 1 1 1 1			
SAHF = 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			

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code			
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 d w	mod reg r/m		
Immediate to Register/Memory	1 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 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 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 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 and Register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with Register/Memory	1 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 = Multiply (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 = ASCII 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			

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code			
	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
LOGIC				
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 = Stor Byte/Wd from AL/A	1 0 1 0 1 0 1 w			
CONTROL TRANSFER				
CALL = Call:				
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		

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	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 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 Seg Adding Immed 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	disp	
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	
INT = Interrupt			
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		

80C86A/80C88A INSTRUCTION SET SUMMARY (Continued)

Mnemonic and Description	Instruction Code	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
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 = Escape (to External Device)	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	

NOTES:

AL = 8-bit accumulator

AX = 16-bit accumulator

CX = Count register

DS = Data segment

ES = Extra segment

Above/below refers to unsigned value.

Greater = more positive;

Less = less positive (more negative) signed values

if d = 1 then "to" reg; if d = 0 then "from" reg

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**MOV CS, REG/MEMORY not allowed.

DATA SHEET REVISION REVIEW

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

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

if s:w = 01 then 16 bits of immediate data form the operand.

if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.

if v = 0 then "count" = 1; if v = 1 then "count" in (CL) x = don't care

z is used for string primitives for comparison with ZF FLAG.

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

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

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

FLAGS =

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

Mnemonics © Intel, 1978



8087 MATH COPROCESSOR

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

The Intel 8087 Math CoProcessor is an extension to the Intel 8086/8088 microprocessor architecture. When combined with the 8086/8088 microprocessor, the 8087 dramatically increases the processing speed of computer applications which utilize mathematical operations such as CAM, numeric controllers, CAD or graphics.

The 8087 Math CoProcessor adds 68 mnemonics to the 8086 microprocessor instruction set. Specific 8087 math operations include logarithmic, arithmetic, exponential, and trigonometric functions. The 8087 supports integer, floating point and BCD data formats, and fully conforms to the ANSI/IEEE floating point standard.

The 8087 is fabricated with HMOS III technology and packaged in a 40-pin cerdip package.

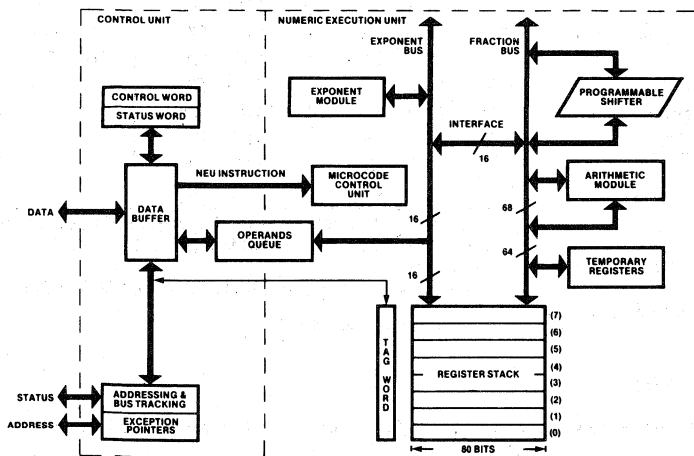


Figure 1. 8087 Block Diagram

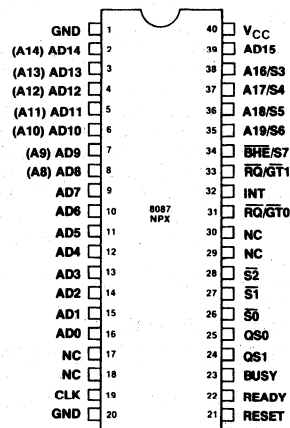


Figure 2. 8087 Pin Configuration

Table 1. 8087 Pin Description

Symbol	Type	Name and Function																								
AD15–AD0	I/O	ADDRESS DATA: These lines constitute the time multiplexed memory address (T_1) and data (T_2 , T_3 , T_W , T_4) bus. A0 is analogous to the \overline{BHE} for the lower byte of the data bus, pins D7–D0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory operations. Eight-bit oriented devices tied to the lower half of the bus would normally use A0 to condition chip select functions. These lines are active HIGH. They are input/output lines for 8087-driven bus cycles and are inputs which the 8087 monitors when the CPU is in control of the bus. A15–A8 do not require an address latch in an 8088/8087 or 80188/8087. The 8087 will supply an address for the T_1 – T_4 period.																								
A19/S6, A18/S5, A17/S4, A16/S3	I/O	ADDRESS MEMORY: During T_1 these are the four most significant address lines for memory operations. During memory operations, status information is available on these lines during T_2 , T_3 , T_W , and T_4 . For 8087-controlled bus cycles, S6, S4, and S3 are reserved and currently one (HIGH), while S5 is always LOW. These lines are inputs which the 8087 monitors when the CPU is in control of the bus.																								
$\overline{BHE}/S7$	I/O	BUS HIGH ENABLE: During T_1 the bus high enable signed (\overline{BHE}) should be used to enable data onto the most significant half of the data bus, pins D15–D8. Eight-bit-oriented devices tied to the upper half of the bus would normally use \overline{BHE} to condition chip select functions. \overline{BHE} is LOW during T_1 for read and write cycles when a byte is to be transferred on the high portion of the bus. The S7 status information is available during T_2 , T_3 , T_W , and T_4 . The signal is active LOW. S7 is an input which the 8087 monitors during the CPU-controlled bus cycles.																								
$\overline{S2}$, $\overline{S1}$, $\overline{S0}$	I/O	STATUS: For 8087-driven, these status lines are encoded as follows: <table><tr><th>$\overline{S2}$</th><th>$\overline{S1}$</th><th>$\overline{S0}$</th><th></th></tr><tr><td>0 (LOW)</td><td>X</td><td>X</td><td>Unused</td></tr><tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>Unused</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> <p>Status is driven active during T_4, remains valid during T_1 and T_2, and is returned to the passive state (1, 1, 1) during T_3 or during T_W when READY is HIGH. This status is used by the 8288 Bus Controller (or the 82188 Integrated Bus Controller with an 80186/80188 CPU) to generate all memory access control signals. Any change in $\overline{S2}$, $\overline{S1}$, or $\overline{S0}$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle. These signals are monitored by the 8087 when the CPU is in control of the bus.</p>	$\overline{S2}$	$\overline{S1}$	$\overline{S0}$		0 (LOW)	X	X	Unused	1 (HIGH)	0	0	Unused	1	0	1	Read Memory	1	1	0	Write Memory	1	1	1	Passive
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$																								
0 (LOW)	X	X	Unused																							
1 (HIGH)	0	0	Unused																							
1	0	1	Read Memory																							
1	1	0	Write Memory																							
1	1	1	Passive																							
$\overline{RQ}/\overline{GT0}$	I/O	REQUEST/GRANT: This request/grant pin is used by the 8087 to gain control of the local bus from the CPU for operand transfers or on behalf of another bus master. It must be connected to one of the two processor request/grant pins. The request/grant sequence on this pin is as follows: <ol style="list-style-type: none">1. A pulse one clock wide is passed to the CPU to indicate a local bus request by either the 8087 or the master connected to the 8087 $\overline{RQ}/\overline{GT1}$ pin.2. The 8087 waits for the grant pulse and when it is received will either initiate bus transfer activity in the clock cycle following the grant or pass the grant out on the $\overline{RQ}/\overline{GT1}$ pin in this clock if the initial request was for another bus master.3. The 8087 will generate a release pulse to the CPU one clock cycle after the completion of the last 8087 bus cycle or on receipt of the release pulse from the bus master on $\overline{RQ}/\overline{GT1}$. <p>For 80186/80188 systems the same sequence applies except $\overline{RQ}/\overline{GT}$ signals are converted to appropriate HOLD, HLDA signals by the 82188 Integrated Bus Controller. This is to conform with 80186/80188's HOLD, HLDA bus exchange protocol. Refer to the 82188 data sheet for further information.</p>																								

Table 1. 8087 Pin Description (Continued)

Symbol	Type	Name and Function															
RQ/GT1	I/O	<p>REQUEST/GRANT: This request/grant pin is used by another local bus master to force the 8087 to request the local bus. If the 8087 is not in control of the bus when the request is made the request/grant sequence is passed through the 8087 on the RQ/GT0 pin one cycle later. Subsequent grant and release pulses are also passed through the 8087 with a two and one clock delay, respectively, for resynchronization. RQ/GT1 has an internal pullup resistor, and so may be left unconnected. If the 8087 has control of the bus the request/grant sequence is as follows:</p> <ol style="list-style-type: none"> 1. A pulse 1 CLK wide from another local bus master indicates a local bus request to the 8087 (pulse 1). 2. During the 8087's next T_4 or T_1 a pulse 1 CLK wide from the 8087 to the requesting master (pulse 2) indicates that the 8087 has allowed the local bus to float and that it will enter the "RQ/GT acknowledge" state at the next CLK. The 8087's control unit is disconnected logically from the local bus during "RQ/GT acknowledge." 3. A pulse 1 CLK wide from the requesting master indicates to the 8087 (pulse 3) that the "RQ/GT" request is about to end and that the 8087 can reclaim the local bus at the next CLK. <p>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.</p> <p>For 80186/80188 system, the RQ/GT1 line may be connected to the 82188 Integrated Bus Controller. In this case, a third processor with a HOLD, HLDA bus exchange system may acquire the bus from the 8087. For this configuration, RQ/GT1 will only be used if the 8087 is the bus master. Refer to 82188 data sheet for further information.</p>															
QS1, QS0	I	<p>QS1, QS0: QS1 and QS0 provide the 8087 with status to allow tracking of the CPU instruction queue.</p> <table> <tr> <th>QS1</th><th>QS0</th><th></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 Op Code 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>	QS1	QS0		0 (LOW)	0	No Operation	0	1	First Byte of Op Code from Queue	1 (HIGH)	0	Empty the Queue	1	1	Subsequent Byte from Queue
QS1	QS0																
0 (LOW)	0	No Operation															
0	1	First Byte of Op Code from Queue															
1 (HIGH)	0	Empty the Queue															
1	1	Subsequent Byte from Queue															
INT	O	<p>INTERRUPT: This line is used to indicate that an unmasked exception has occurred during numeric instruction execution when 8087 interrupts are enabled. This signal is typically routed to an 8259A for 8086/8088 systems and to INTO for 80186/80188 systems. INT is active HIGH.</p>															
BUSY	O	<p>BUSY: This signal indicates that the 8087 NEU is executing a numeric instruction. It is connected to the CPU's TEST pin to provide synchronization. In the case of an unmasked exception BUSY remains active until the exception is cleared. BUSY is active HIGH.</p>															
READY	I	<p>READY: READY is the acknowledgement from the addressed memory device that it will complete the data transfer. The RDY signal from memory is synchronized by the 8284A Clock Generator to form READY for 8086 systems. For 80186/80188 systems, RDY is synchronized by the 82188 Integrated Bus Controller to form READY. This signal is active HIGH.</p>															
RESET	I	<p>RESET: RESET causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. RESET is internally synchronized.</p>															
CLK	I	<p>CLOCK: The clock provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.</p>															
V _{CC}		<p>POWER: V_{CC} is the +5V power supply pin.</p>															
GND		<p>GROUND: GND are the ground pins.</p>															

NOTE:

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

APPLICATION AREAS

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

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

PROGRAMMING LANGUAGE SUPPORT

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

RELATED INFORMATION

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

FUNCTIONAL DESCRIPTION

The 8087 Math CoProcessor's architecture is designed for high performance numeric computing in conjunction with general purpose processing.

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

System Configuration

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

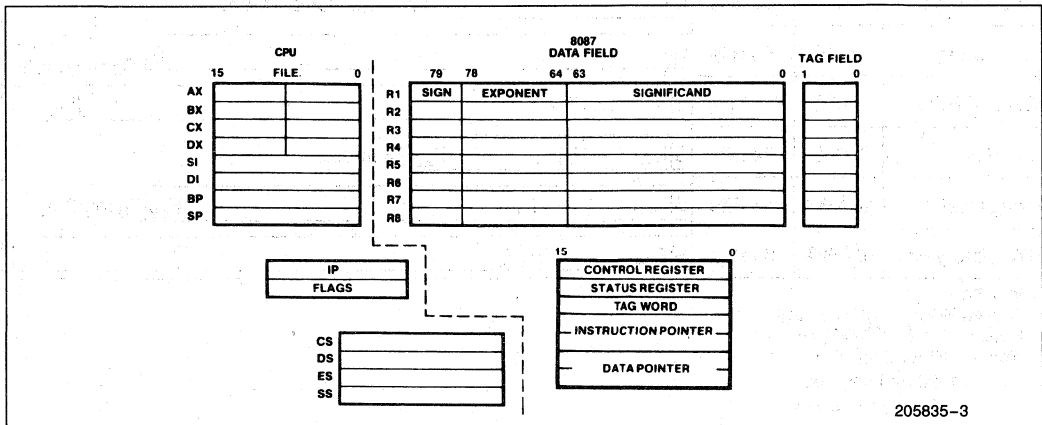


Figure 3. CPU + 8087 Architecture

The 8087 uses one of the request/grant lines of the 8086/8088 architecture (typically $\overline{RQ}/\overline{GT0}$) to obtain control of the local bus for data transfers. The other request/grant line is available for general system use (for instance by an I/O processor in LOCAL mode). A bus master can also be connected to the 8087's $\overline{RQ}/\overline{GT1}$ line. In this configuration the 8087 will pass the request/grant handshake signals between the CPU and the attached master when the 8087 is not in control of the bus and will relinquish the bus to the master directly when the 8087 is in control. In this way two additional masters can be configured in an 8086/8088 system; one will share the 8086/8088 bus with the 8087 on a first-come first-served basis, and the second will be guaranteed to be higher in priority than the 8087.

For 80186/80188 systems, $\overline{RQ}/\overline{GT0}$ and $\overline{RQ}/\overline{GT1}$ are connected to the corresponding inputs of the 82188 Integrated Bus Controller. Because the 80186/80188 has a HOLD, HLDA bus exchange protocol, an interface is needed which will translate $\overline{RQ}/\overline{GT}$ signals to corresponding HOLD, HLDA signals and vice versa. One of the functions of the 82188 IBC is to provide this translation. $\overline{RQ}/\overline{GT0}$ is translated to HOLD, HLDA signals which are then directly connected to the 80186/80188. The $\overline{RQ}/\overline{GT1}$ line is also translated into HOLD, HLDA signals (referred to as SYSHOLD, SYSHLDA signals) by the 82188 IBC. This allows a third processor (using a HOLD, HLDA bus exchange protocol) to gain control of the bus.

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

Bus Operation

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

Table 2. 8087 Data Types

Data Formats	Range	Precision	Most Significant Byte													
			7	0	7	0	7	0	7	0	7	0	7	0	7	0
Word Integer	10^4	16 Bits	I ₁₅ I ₀ Two's Complement													
Short Integer	10^9	32 Bits	I ₃₁ I ₀ Two's Complement													
Long Integer	10^{18}	64 Bits	I ₆₃ I ₀ Two's Complement													
Packed BCD	10^{18}	18 Digits	S ₁₇ D ₁₇ D ₁₆ D ₁ D ₀													
Short Real	$10^{\pm 38}$	24 Bits	S ₁ E ₇ E ₀ F ₁ F ₂₃ F ₀ Implicit													
Long Real	$10^{\pm 308}$	53 Bits	S ₁ E ₁₀ E ₀ F ₁ F ₅₂ F ₀ Implicit													
Temporary Real	$10^{\pm 4932}$	64 Bits	S ₁ E ₁₄ E ₀ F ₀ F ₆₃													

Integer: I

Packed BCD: $(-1)^{S(D_{17}...D_0)}$

Real: $(-1)^{S(2E-Bias)}(F_0 \bullet F_{1...})$

bias = 127 for Short Real

1023 for Long Real

16383 for Temp Real

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

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

Programming Interface

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

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

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

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

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

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

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

NUMERIC PROCESSOR EXTENSION ARCHITECTURE

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

Control Unit

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

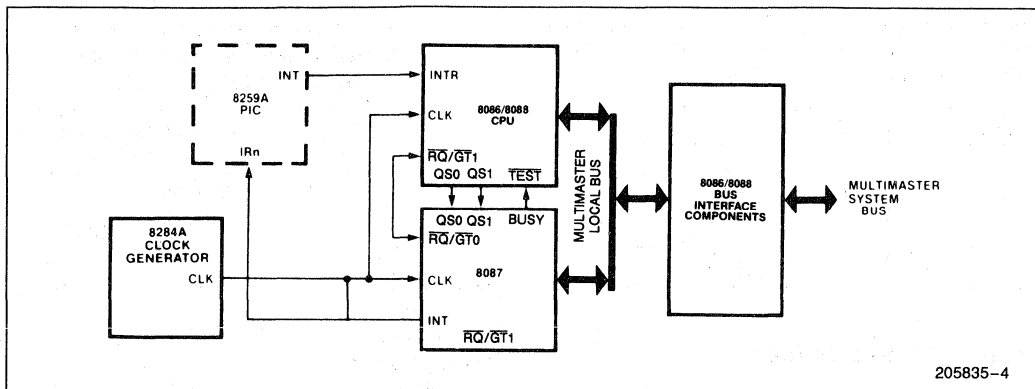


Figure 4. 8086/8087, 8088/8087 System Configuration

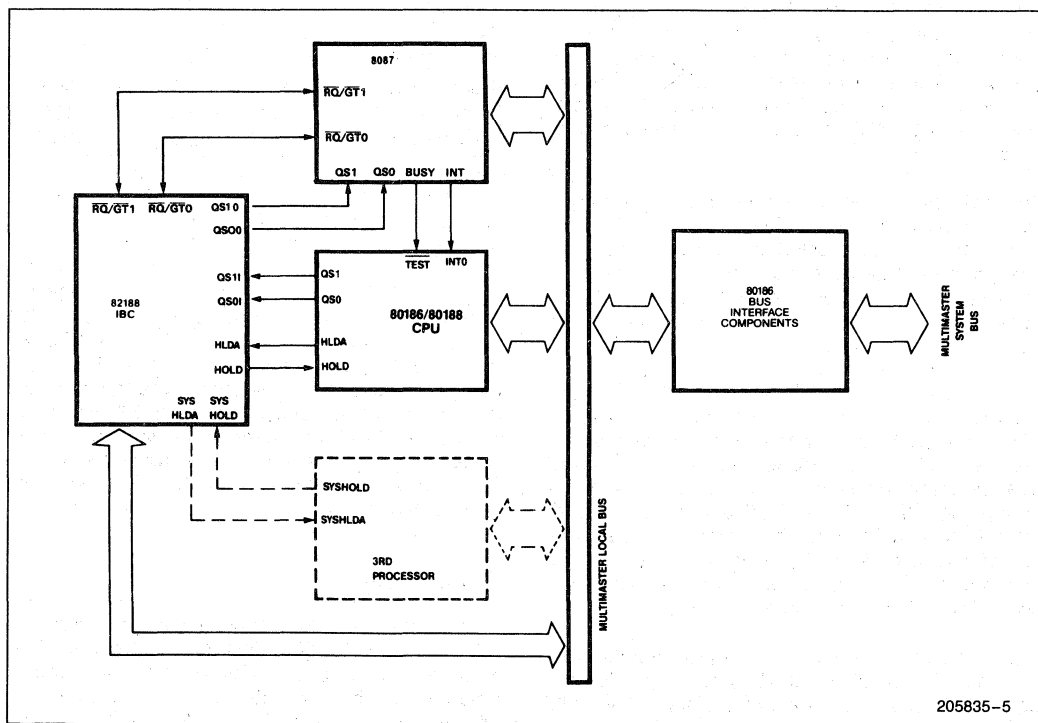


Figure 5. 80186/8087, 80188/8087 System Configuration

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

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

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

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

Numeric Execution Unit

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

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

Register Set

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

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

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

Status Word

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

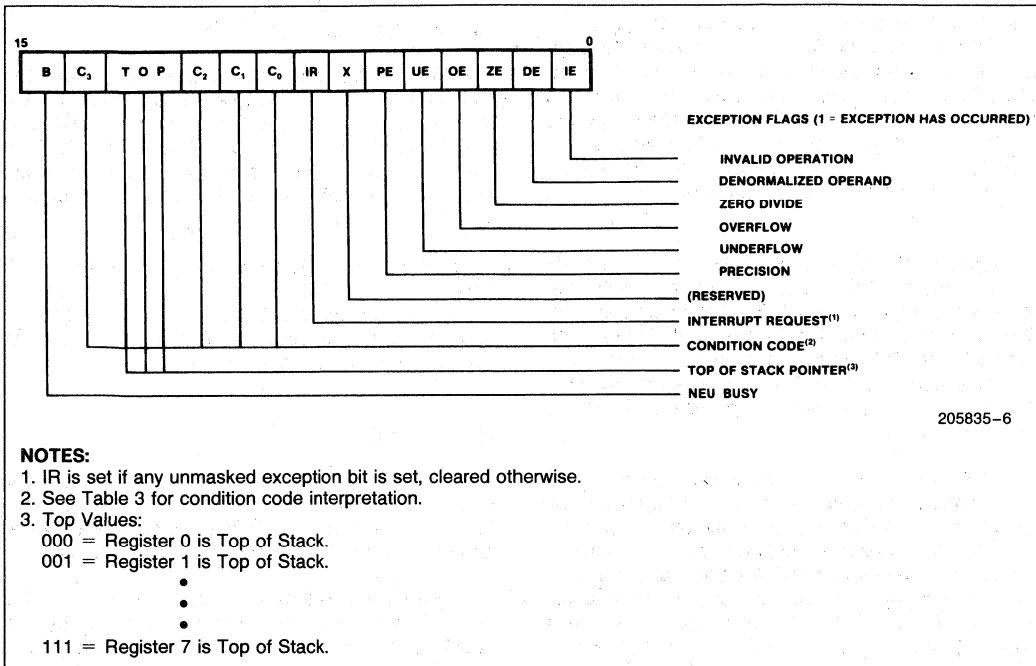


Figure 6. 8087 Status Word

The four numeric condition code bits (C₀–C₃) are similar to flags in a CPU: various instructions update these bits to reflect the outcome of the 8087 operations. The effect of these instructions on the condition code bits is summarized in Table 4.

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

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

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

Tag Word

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

Instruction and Data Pointers

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

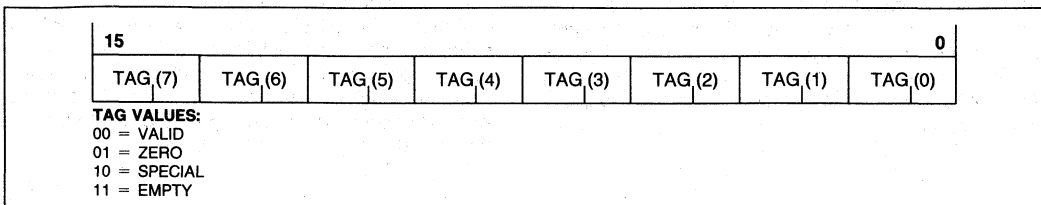


Figure 7. 8087 Tag Word

Table 4a. Condition Code Interpretation

Instruction Type	C ₃	C ₂	C ₁	C ₀	Interpretation
Compare, Test	0	0	X	0	ST > Source or 0 (FTST)
	0	0	X	1	ST < Source or 0 (FTST)
	1	0	X	0	ST = Source or 0 (FTST)
	1	1	X	1	ST is not comparable
Remainder	Q ₁	0	Q ₀	Q ₂	Complete reduction with three low bits of quotient (See Table 4b)
	U	1	U	U	Incomplete Reduction
Examine	0	0	0	0	Valid, positive unnormalized
	0	0	0	1	Invalid, positive, exponent = 0
	0	0	1	0	Valid, negative, unnormalized
	0	0	1	1	Invalid, negative, exponent = 0
	0	1	0	0	Valid, positive, normalized
	0	1	0	1	Infinity, positive
	0	1	1	0	Valid, negative, normalized
	0	1	1	1	Infinity, negative
	1	0	0	0	Zero, positive
	1	0	0	1	Empty
	1	0	1	0	Zero, negative
	1	0	1	1	Empty
	1	1	0	0	Invalid, positive, exponent = 0
	1	1	0	1	Empty
	1	1	1	0	Invalid, negative, exponent = 0
	1	1	1	1	Empty

NOTES:

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

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

Dividend Range	Q ₂	Q ₁	Q ₀
Dividend < 2 * Modulus	C ₃ ¹	C ₁ ¹	Q ₀
Dividend < 4 * Modulus	C ₃ ¹	Q ₁	Q ₀
Dividend ≥ 4 * Modulus	Q ₂	Q ₁	Q ₀

NOTE:

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

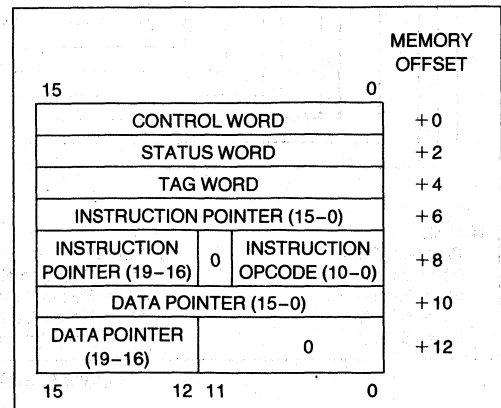


Figure 8. 8087 Instruction and Data Pointer Image in Memory

Control Word

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

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

Exception Handling

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

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

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

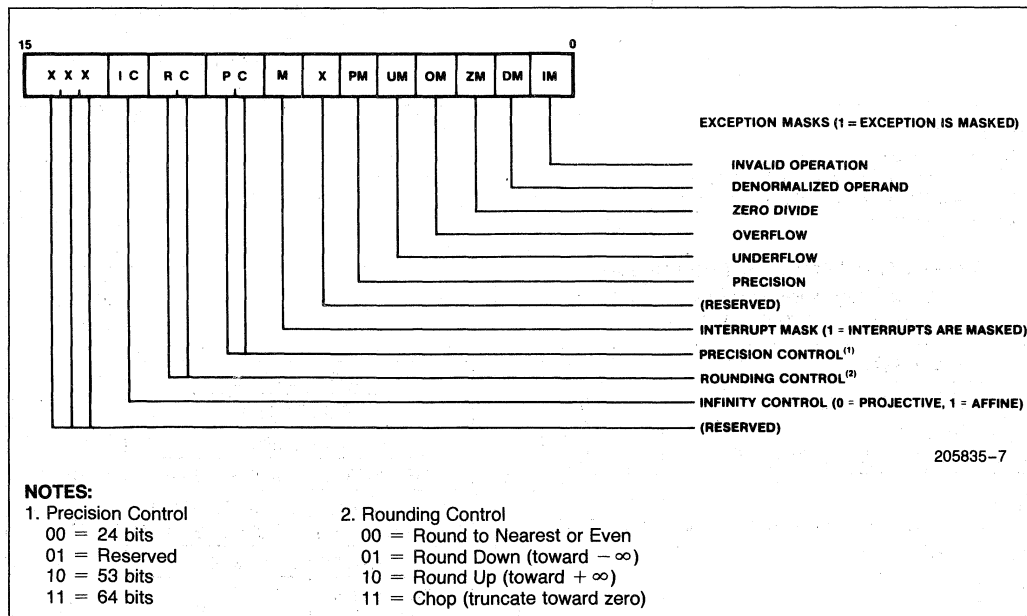


Figure 9. 8087 Control Word

2. **OVERFLOW:** The result is too large in magnitude to fit the specified format. The 8087 will generate an encoding for infinity if this exception is masked.
3. **ZERO DIVISOR:** The divisor is zero while the dividend is a non-infinite, non-zero number. Again, the 8087 will generate an encoding for infinity if this exception is masked.
4. **UNDERFLOW:** The result is non-zero but too small in magnitude to fit in the specified format. If this exception is masked the 8087 will denormal-

ize (shift right) the fraction until the exponent is in range. This process is called gradual underflow.

5. **DENORMALIZED OPERAND:** At least one of the operands or the result is denormalized; it has the smallest exponent but a non-zero significand. Normal processing continues if this exception is masked off.
6. **INEXACT RESULT:** If the true result is not exactly representable in the specified format, the result is rounded according to the rounding mode, and this flag is set. If this exception is masked, processing will simply continue.

ABSOLUTE MAXIMUM RATINGS*

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

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

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

2

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}, V_{CC} = 5V \pm 5\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage (Note 8)		0.45	V	$I_{OL} = 2.5 \text{ mA}$
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -400 \mu\text{A}$
I_{CC}	Power Supply Current		475	mA	$T_A = 25^\circ\text{C}$
I_{LI}	Input Leakage Current		± 10	μA	$0V \leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current		± 10	μA	$T_A = 25^\circ\text{C}$
V_{CL}	Clock Input Low Voltage	-0.5	0.6	V	
V_{CH}	Clock Input High Voltage	3.9	$V_{CC} + 1.0$	V	
C_{IN}	Capacitance of Inputs		10	pF	$f_c = 1 \text{ MHz}$
C_{IO}	Capacitance of I/O Buffer (AD0-15, A16-A19, BHE, S2-S0, RQ/GT) and CLK		15	pF	$f_c = 1 \text{ MHz}$
C_{OUT}	Capacitance of Outputs BUSY INT		10	pF	$f_c = 1 \text{ MHz}$

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

TIMING REQUIREMENTS

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

TIMING RESPONSES

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

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = 5V \pm 5\%$ (Continued)

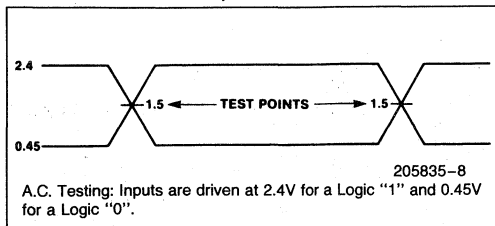
TIMING RESPONSES (Continued)

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

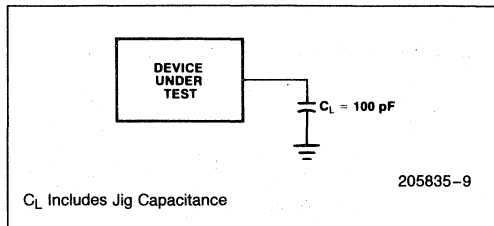
NOTES:

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

A.C. TESTING INPUT, OUTPUT WAVEFORM

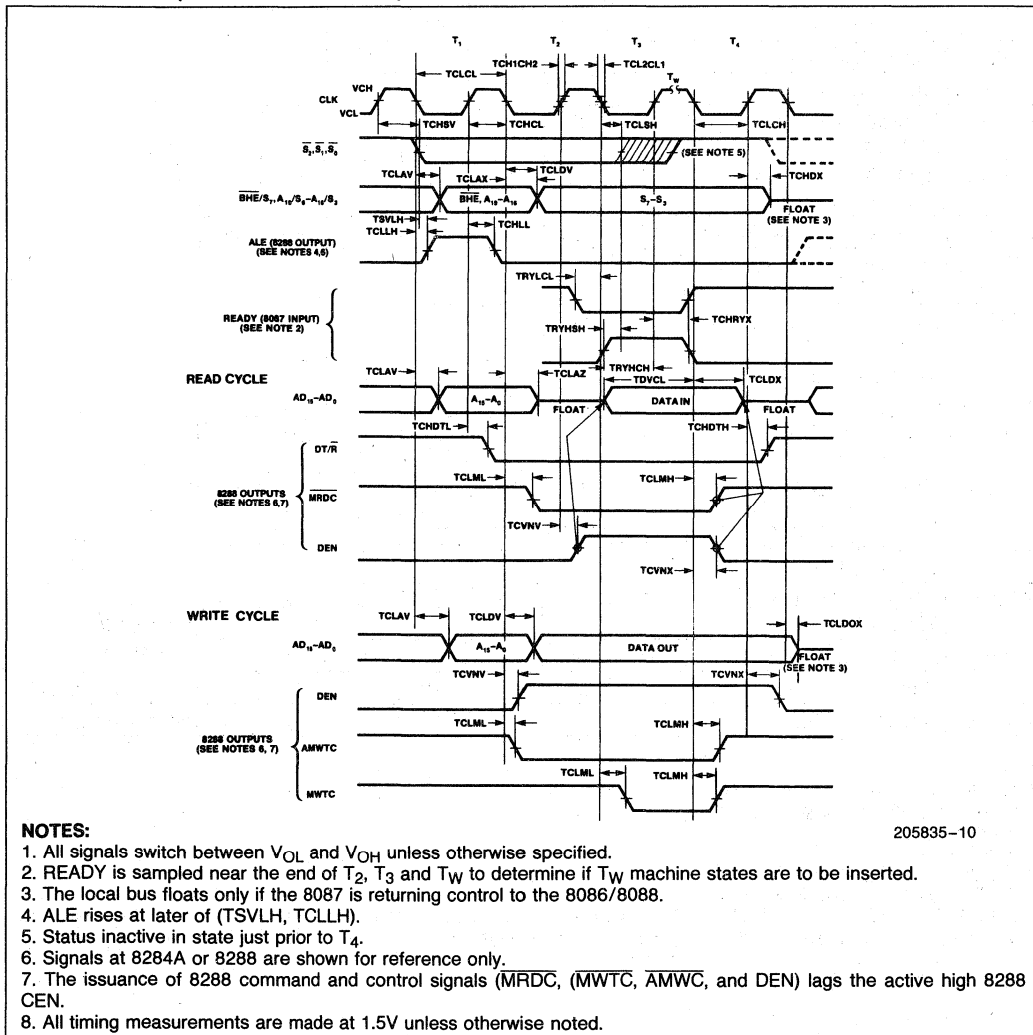


A.C. TESTING LOAD CIRCUIT



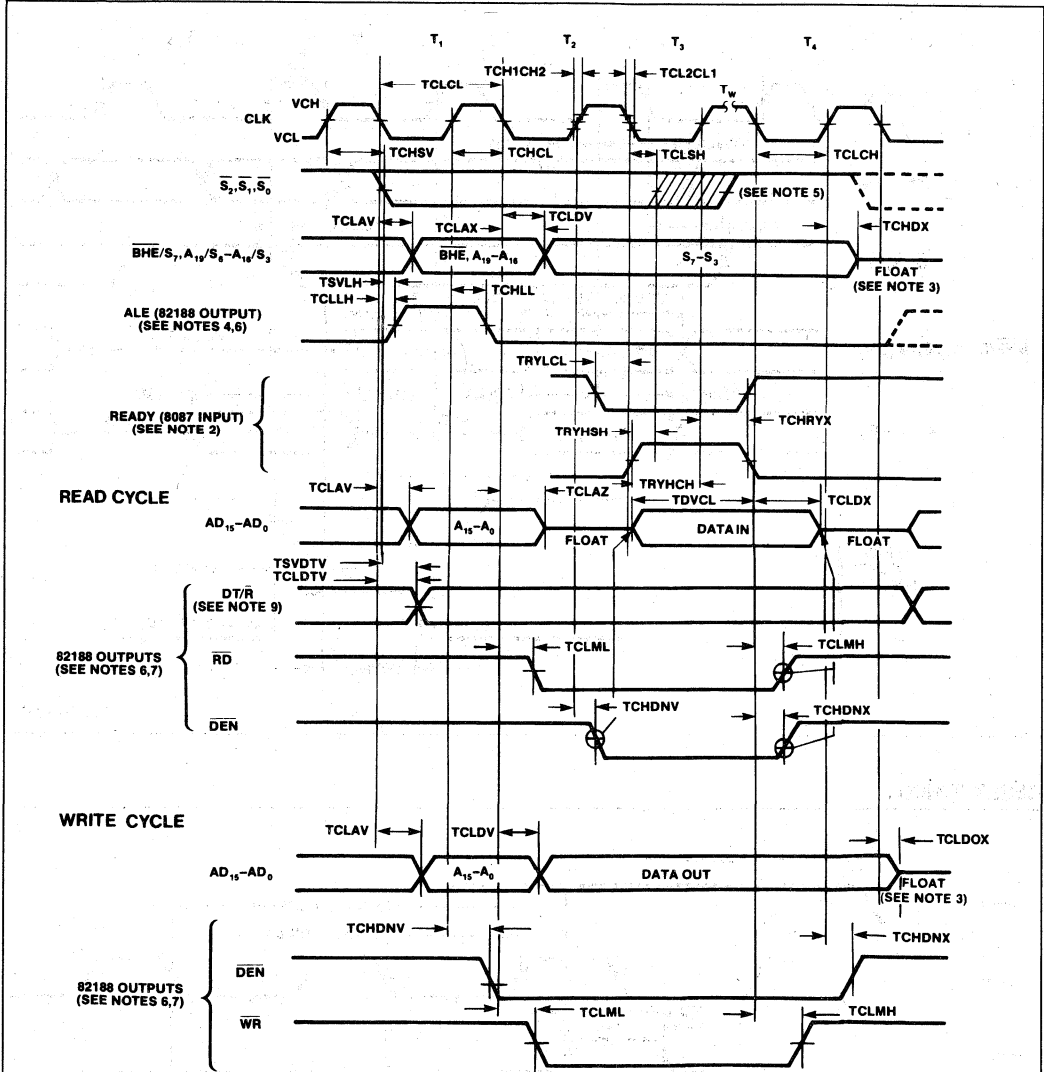
WAVEFORMS

MASTER MODE (with 8288 references)



WAVEFORMS (Continued)

MASTER MODE (with 82188 references)



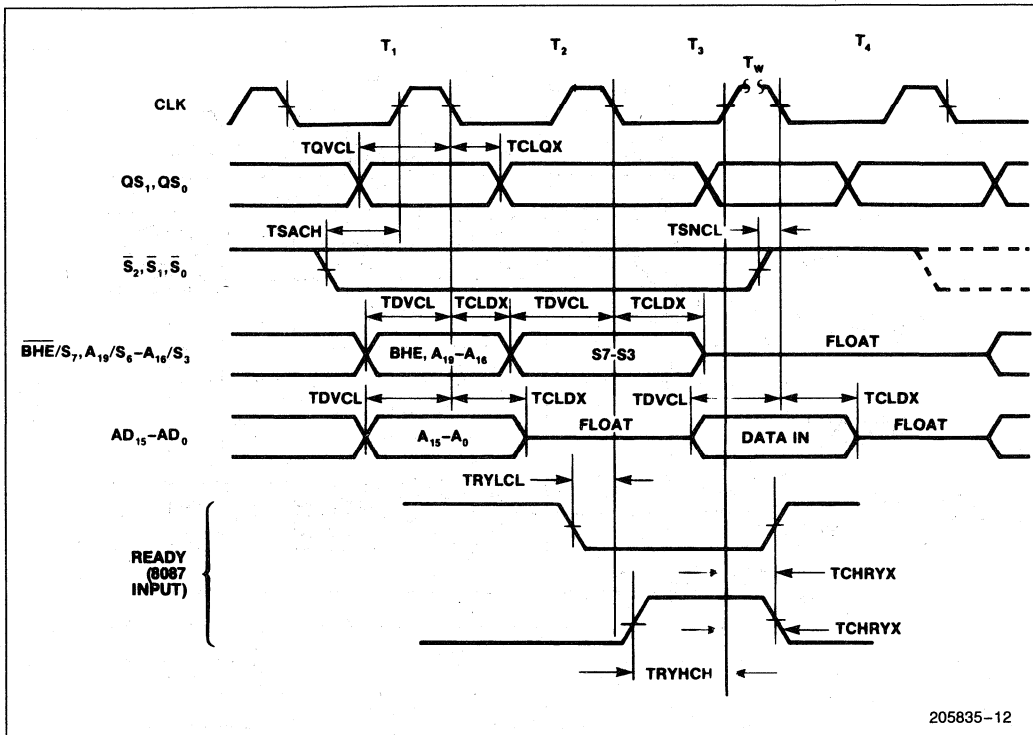
NOTES:

1. All signals switch between V_{OL} and V_{OH} unless otherwise specified.
2. READY is sampled near the end of T₂, T₃ and T_W to determine if T_W machine states are to be inserted.
3. The local bus floats only if the 8087 is returning control to the 80186/80188.
4. ALE rises at later of (TSVLH, TCLLH).
5. Status inactive in state just prior to T₄.
6. Signals at 8284A or 82188 are shown for reference only.
7. The issuance of 8288 command and control signals (MRDC, MWTC, AMWC, and DEN) lags the active high 8288 CEN.
8. All timing measurements are made at 1.5V unless otherwise noted.
9. DT/R becomes valid at the later of (TSVDTV, TCLDTV).

205835-11

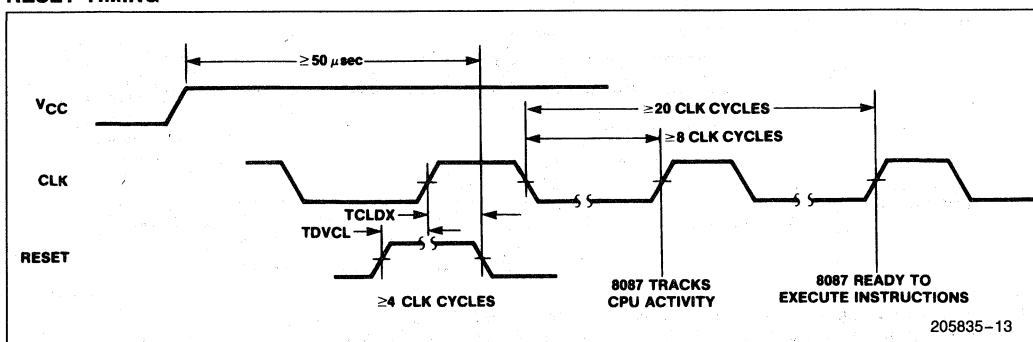
WAVEFORMS (Continued)

PASSIVE MODE



205835-12

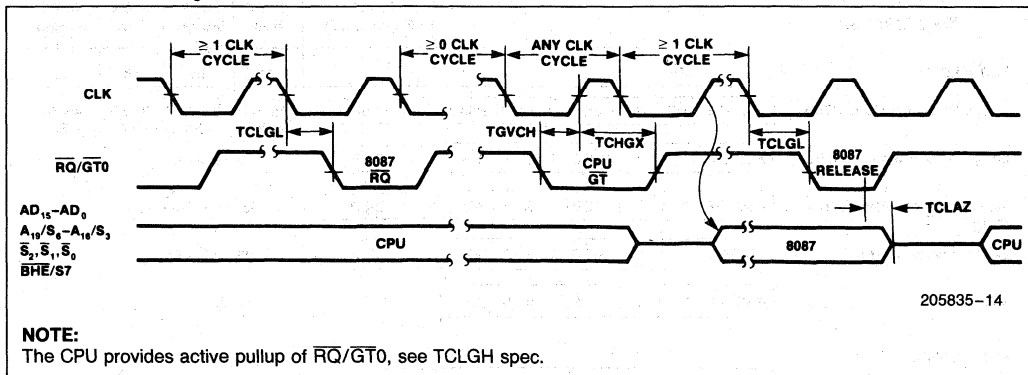
RESET TIMING



205835-13

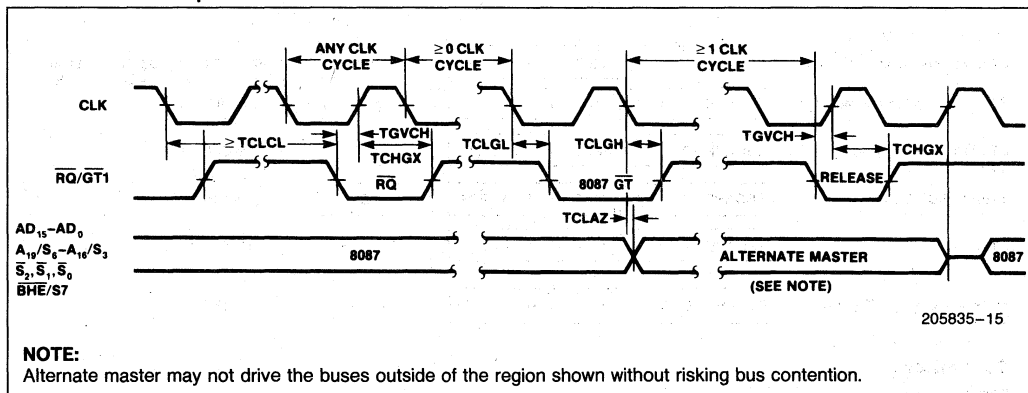
WAVEFORMS (Continued)

REQUEST/GRANT₀ TIMING



2

REQUEST/GRANT₁ TIMING



BUSY AND INTERRUPT TIMING

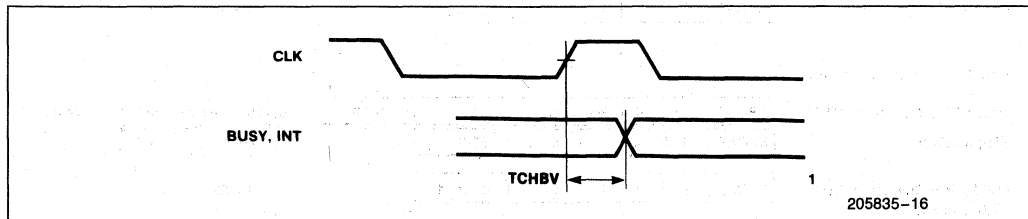


Table 5. 8087 Extensions to the 86/186 Instructions Sets

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

205835-17

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

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

205835-18

NOTE:

1. If P = 1 then add 5 clocks.

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

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

205835-19

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

		Clock Count Range
FFREE = Free ST(i)	ESCAPE 1 0 1 1 1 0 0 0 ST(i)	9-16
FNOP = No Operation	ESCAPE 0 0 1 1 1 0 1 0 0 0 0	10-16
FWAIT = CPU Wait for 8087	1 0 0 1 1 0 1 1	3 + 5n*
		205835-20

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

NOTES:

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

2

80286 Microprocessor Family **3**



80C286

HIGH PERFORMANCE CHMOS MICROPROCESSOR WITH MEMORY MANAGEMENT AND PROTECTION

- High Speed CHMOS III Technology
- Pin for Pin, Clock for Clock, and Functionally Compatible with the HMOS 80286

(See 80C286 Data Sheet, Order #210253)

- Stop Clock Capability
 - Uses Less Power (see I_{CCS} Specification)

- 12.5 MHz Clock Rate

- Available in a Variety of Packages:
 - 68 Pin PLCC (Plastic Leaded Chip Carrier)
 - 68 Pin PGA (Pin Grid Array)

(See Packaging Spec., Order #231369)

INTRODUCTION

The 80C286 is an advanced 16 bit CHMOS III microprocessor designed for multi-user and multi-tasking applications that require low power and high performance. The 80C286 is fully compatible with its predecessor the HMOS 80286 and object-code compatible with the 8086 and 80386 family of products. In addition, the 80C286 has a power down mode which uses less power, making it ideal for mobile applications. The 80C286 has built-in memory protection that maintains a four level protection mechanism for task isolation, a hardware task switching facility and memory management capabilities that map 2^{30} bytes (one gigabyte) of virtual address space per task (per user) into 2^{24} bytes (16 megabytes) of physical memory.

The 80C286 is upward compatible with 8086 and 8088 software. Using 8086 real address mode, the 80C286 is object code compatible with existing 8086, 8088 software. In protected virtual address mode, the 80C286 is source code compatible with 8086, 8088 software which may require upgrading to use virtual addresses supported by the 80C286's integrated memory management and protection mechanism. Both modes operate at full 80C286 performance and execute a superset of the 8086 and 8088 instructions.

The 80C286 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 80C286 also supports virtual memory systems by providing a segment-not-present exception and restartable instructions.

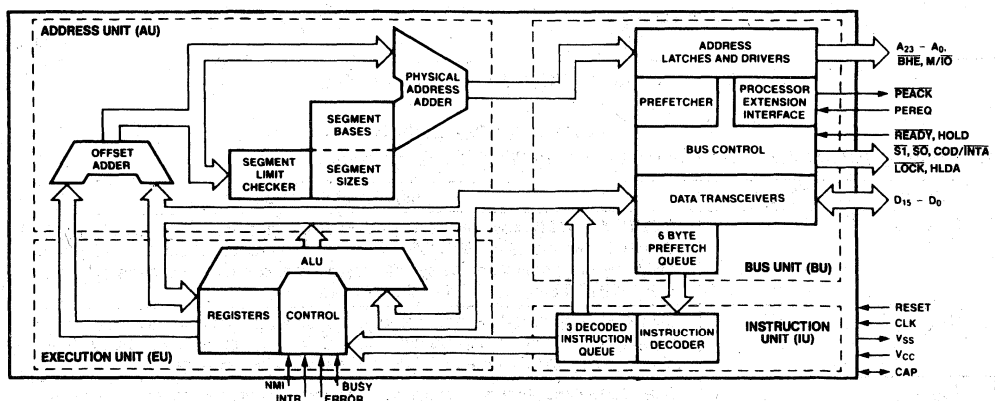
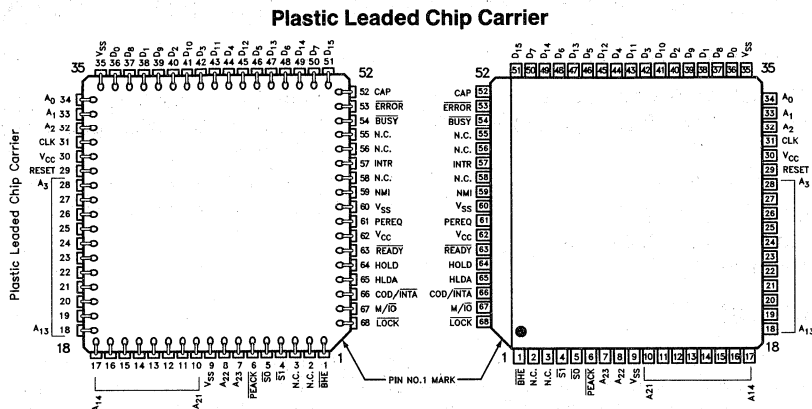


Figure 1. 80C286 Internal Block Diagram

231923-1

Component Pad Views—As viewed from underside of component when mounted on the board.

P.C. Board Views—As viewed from the component side of the P.C. board.

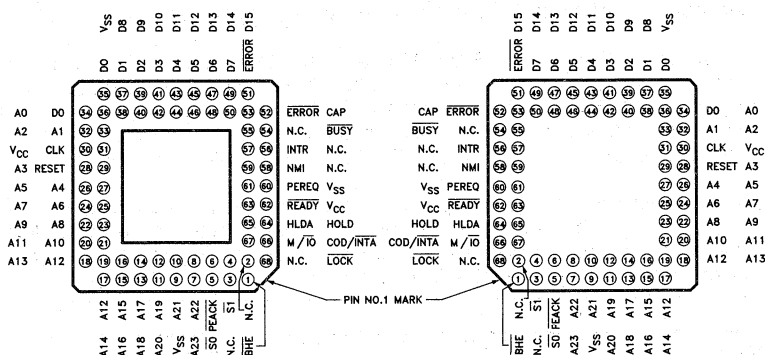


231923-2

NOTE:

N.C. signals must not be connected

Pin Grid Array



231923-3

Figure 2. 80C286 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for the 80C286 microprocessor :

Symbol	Type	Name and Function
CLK	I	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 a LOW to HIGH transition on the RESET input.
D ₁₅ -D ₀	I/O	DATA BUS inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles. The data bus is active HIGH and floats to 3-state OFF* during bus hold acknowledge.
A ₂₃ -A ₀	O	ADDRESS BUS outputs physical memory and I/O port addresses. A ₀ is LOW when data is to be transferred on pins D ₇₋₀ . A ₂₃ -A ₁₆ are LOW during I/O transfers. The address bus is active HIGH and floats to 3-state OFF* during bus hold acknowledge.
BHE	O	BUS HIGH ENABLE indicates transfer of data on the upper byte of the data bus. D ₁₅₋₈ . Eight-bit oriented devices assigned to the upper byte of the data bus would normally use BHE to condition chip select functions. BHE is active LOW and floats to 3-state OFF* during bus hold acknowledge.

*See bus hold circuitry section.

Table I. Pin Description (Continued)

Symbol	Type	Name and Function				
BHE (Continued)		BHE and A0 Encodings				
		BHE Value	A0 Value	Function		
		0	0	Word transfer		
		0	1	Transfer on upper half of data bus (D ₁₅ –D ₈)		
		1	0	Byte transfer on lower half of data bus (D ₇ –D ₀)		
1	1	Will never occur				
S ₁ , S ₀	O	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, S ₁ and S ₀ are active LOW and float to 3-state OFF* during bus hold acknowledge.				
		80C286 Bus Cycle Status Definition				
		COD/ $\overline{\text{INTA}}$	M/ $\overline{\text{IO}}$	S ₁	S ₀	Bus Cycle Initiated
		0 (LOW)	0	0	0	Interrupt acknowledge
		0	0	0	1	Will not occur
		0	0	1	0	Will not occur
		0	0	1	1	None; not a status cycle
		0	1	0	0	IF A1 = 1 then halt; else shutdown
		0	1	0	1	Memory data read
		0	1	1	0	Memory data write
		0	1	1	1	None; not a status cycle
		1 (HIGH)	0	0	0	Will not occur
		1	0	0	1	I/O read
		1	0	1	0	I/O write
		1	0	1	1	None; not a status cycle
		1	1	0	0	Will not occur
		1	1	0	1	Memory instruction read
		1	1	1	0	Will not occur
		1	1	1	1	None; not a status cycle
		M/ $\overline{\text{IO}}$	O	MEMORY I/O SELECT distinguishes memory access from I/O access. If HIGH during T _s , a memory cycle or a halt/shutdown cycle is in progress. If LOW, an I/O cycle or an interrupt acknowledge cycle is in progress. M/ $\overline{\text{IO}}$ floats to 3-state OFF* during bus hold acknowledge.		
COD/ $\overline{\text{INTA}}$	O	CODE/INTERRUPT ACKNOWLEDGE distinguishes instruction fetch cycles from memory data read cycles. Also distinguishes interrupt acknowledge cycles from I/O cycles. COD/ $\overline{\text{INTA}}$ floats to 3-state OFF* during bus hold acknowledge. Its timing is the same as M/ $\overline{\text{IO}}$.				
LOCK	O	BUS LOCK indicates that other system bus masters are not to gain control of the system bus for the current and the following bus cycle. The 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. LOCK is active LOW and floats to 3-state OFF* during bus hold acknowledge.				
READY	I	BUS READY terminates a bus cycle. Bus cycles are extended without limit until terminated by READY LOW. READY is an active LOW synchronous input requiring setup and hold times relative to the system clock be met for correct operation. READY is ignored during bus hold acknowledge.				
HOLD HLDA	I O	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 3-state OFF* and then activate HLDA, thus entering the bus hold acknowledge condition. The local bus will remain granted to the requesting master until HOLD becomes inactive which results in the 80C286 deactivating HLDA and regaining control of the local bus. This terminates the bus hold acknowledge condition and regaining control of the local bus. This terminates the bus hold acknowledge condition. HOLD may be asynchronous to the system clock. These signals are active HIGH.				
INTR	I	INTERRUPT REQUEST requests the 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.				

*See bus hold circuitry section.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function										
NMI	I	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 PEACK	I O	PROCESSOR EXTENSION OPERAND REQUEST AND ACKNOWLEDGE extend 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 PEACK output signals the processor extension when the requested operand is being transferred. PEREQ is active HIGH and floats to 3-state OFF* during bus hold acknowledge. PEACK may be asynchronous to the system clock. PEACK is active LOW.										
BUSY ERROR	I I	PROCESSOR EXTENSION BUSY AND ERROR indicate the operating condition of a processor extension to the 80C286. An active BUSY input stops 80C286 program execution on WAIT and some ESC instructions until BUSY becomes inactive (HIGH). The 80C286 may be interrupted while waiting for BUSY to become inactive. An active 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. These inputs have internal pull-up resistors.										
RESET	I	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: <table><tr><th colspan="2">80C286 Pin State During Reset</th></tr><tr><th>Pin Value</th><th>Pin Names</th></tr><tr><td>1 (HIGH)</td><td>S0, S1, PEACK, A23-A0, BHE, LOCK</td></tr><tr><td>0 (LOW)</td><td>M/IO, COD/INTA, HLDA (Note 1)</td></tr><tr><td>3-state OFF*</td><td>D15-D0</td></tr></table> <p>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 38 CLK cycles from the trailing edge of RESET are required by the 80C286 for internal initialization before the first bus cycle, to fetch code from the power-on execution address, occurs.</p> <p>A LOW to HIGH transition of RESET synchronous to the system clock will end a processor cycle at the second HIGH to LOW transition of the system clock. The LOW to HIGH transition of RESET may be asynchronous to the system clock; however, in this case it cannot be predetermined which phase of the processor clock will occur during the next system clock period. Synchronous LOW to HIGH transitions of RESET are required only for systems where the processor clock must be phase synchronous to another clock.</p>	80C286 Pin State During Reset		Pin Value	Pin Names	1 (HIGH)	S0, S1, PEACK, A23-A0, BHE, LOCK	0 (LOW)	M/IO, COD/INTA, HLDA (Note 1)	3-state OFF*	D15-D0
80C286 Pin State During Reset												
Pin Value	Pin Names											
1 (HIGH)	S0, S1, PEACK, A23-A0, BHE, LOCK											
0 (LOW)	M/IO, COD/INTA, HLDA (Note 1)											
3-state OFF*	D15-D0											
V _{SS}	I	SYSTEM GROUND: 0 Volts.										
V _{CC}	I	SYSTEM POWER: + 5 Volt Power Supply.										
CAP	I	SUBSTRATE FILTER CAPACITOR: a 0.047 μ F \pm 20% 12V capacitor can be connected between this pin and ground for compatibility with the HMOS 80286. For systems using only an 80C286, this pin can be left floating.										

*See bus hold circuitry section.

NOTE:

1. HLDA is only Low if HOLD is inactive (Low).

FUNCTIONAL DESCRIPTION

Introduction

The 80C286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multi-tasking systems. Depending on the application, a 12 MHz 80C286's performance is up to ten times faster than the standard 5 MHz 8086's, while providing complete upward software compatibility with Intel's 8086, 88, and 186 family of CPU's.

The 80C286 operates in two modes: 8086 real address mode and protected virtual address mode. Both modes execute a superset of the 8086 and 88 instruction set.

In 8086 real address mode programs use real addresses with up to one megabyte of address space. Programs use virtual addresses in protected virtual address mode, also called protected mode. In protected mode, the 80C286 CPU automatically maps 1 gigabyte of virtual addresses per task into a 16 megabyte real address space. This mode also provides memory protection to isolate the operating system and ensure privacy of each tasks' programs and data. Both modes provide the same base instruction set, registers, and addressing modes.

The following Functional Description describes first, the base 80C286 architecture common to both modes, second, 8086 real address mode, and third, protected mode.

80C286 BASE ARCHITECTURE

The 8086, 88, 186, and 286 CPU family all contain the same basic set of registers, instructions, and

addressing modes. The 80C286 processor is upward compatible with the 8086, 8088, and 80186 CPU's and fully compatible with the HMOS 80286.

Register Set

The 80C286 base architecture has fifteen registers as shown in Figure 3. These registers are grouped into the following four categories:

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

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

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

Status and Control Registers: The 3 16-bit special purpose registers in figure 3A record or control certain aspects of the 80C286 processor state including the Instruction Pointer, which contains the offset address of the next sequential instruction to be executed.

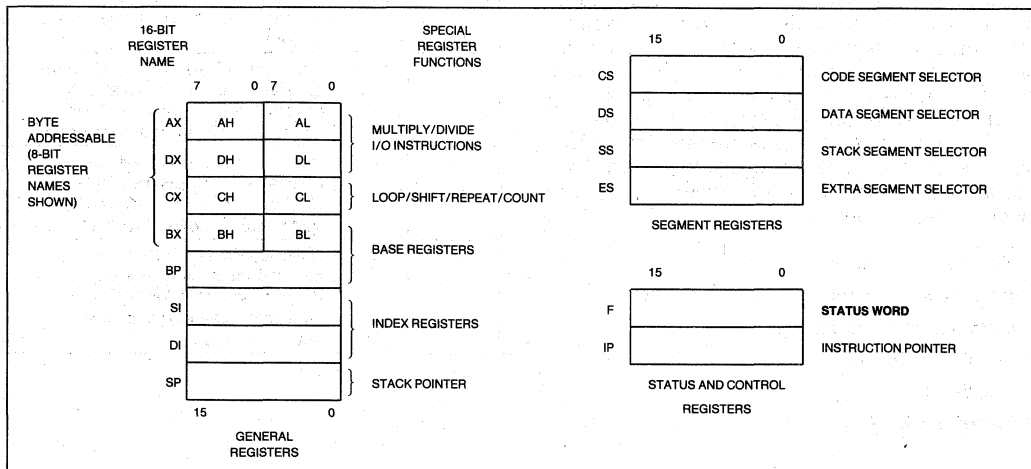


Figure 3. Register Set

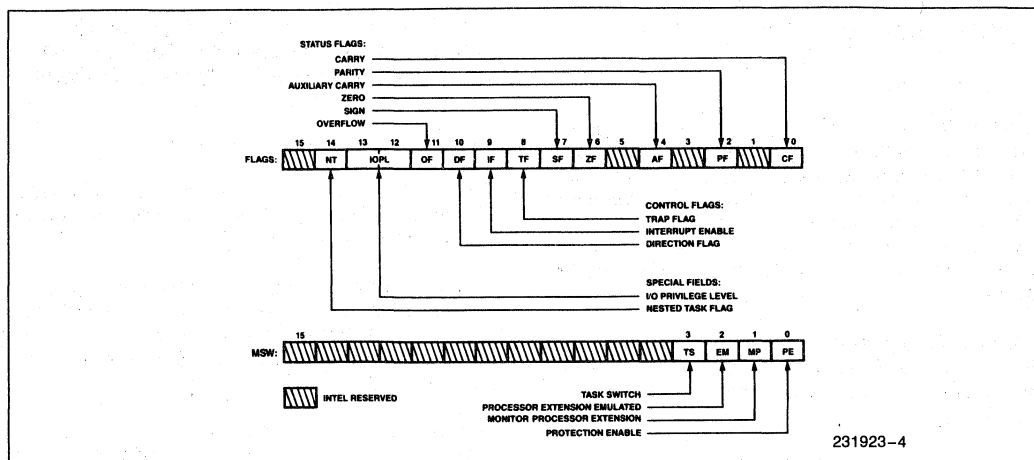


Figure 3a. Status and Control Register Bit Functions

Flags Word Description

The Flags word (Flags) records specific characteristics of the result of logical and arithmetic instructions (bits 0, 2, 4, 6, 7, and 11) and controls the operation of the 80C286 within a given operating mode (bits 8 and 9). Flags is a 16-bit register. The function of the flag bits is given in Table 2.

Instruction Set

The instruction set is divided into seven categories: data transfer, arithmetic, shift/rotate/logical, string manipulation, control transfer, high level instructions, and processor control. These categories are summarized in Figure 4.

An 80C286 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

Table 2. Flags Word Bit Functions

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

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

For detailed operation and usage of each instruction, see Appendix B of the 80286/80287 Programmer's Reference Manual (Order No. 210498).

GENERAL PURPOSE	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push all registers on stack
POPA	Pop all registers from stack
XCHG	Exchange byte or word
XLAT	Translate byte
INPUT/OUTPUT	
IN	Input byte or word
OUT	Output byte or word
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
FLAG TRANSFER	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack

Figure 4a. Data Transfer Instructions

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

Figure 4c. String Instructions

ADDITION	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
MULTIPLICATION	
MUL	Multiple byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply
DIVISION	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to doubleword

Figure 4b. Arithmetic Instructions

LOGICALS	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word
SHIFTS	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word
ROTATES	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

Figure 4d. Shift/Rotate Logical Instructions

CONDITIONAL TRANSFERS		UNCONDITIONAL TRANSFERS	
JA/JNBE	Jump if above/not below nor equal	CALL	Call procedure
JAE/JNB	Jump if above or equal/not below	RET	Return from procedure
JB/JNAE	Jump if below/not above nor equal	JMP	Jump
JBE/JNA	Jump if below or equal/not above		
JC	Jump if carry	ITERATION CONTROLS	
JE/JZ	Jump if equal/zero	LOOP	Loop
JG/JNLE	Jump if greater/not less nor equal		
JGE/JNL	Jump if greater or equal/not less	LOOPE/LOOPZ	Loop if equal/zero
JL/JNGE	Jump if less/not greater nor equal	LOOPNE/LOOPNZ	Loop if not equal/not zero
JLE/JNG	Jump if less or equal/not greater	JCXZ	Jump if register CX = 0
JNC	Jump if not carry		
JNE/JNZ	Jump if not equal/not zero	INTERRUPTS	
JNO	Jump if not overflow	INT	Interrupt
JNP/JPO	Jump if not parity/parity odd		
JNS	Jump if not sign	INTO	Interrupt if overflow
JO	Jump if overflow	IRET	Interrupt return
JP/JPE	Jump if parity/parity even		
JS	Jump if sign		

Figure 4e. Program Transfer Instructions

FLAG OPERATIONS	
STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag
EXTERNAL SYNCHRONIZATION	
HLT	Halt until interrupt or reset
WAIT	Wait for $\overline{\text{BUSY}}$ not active
ESC	Escape to extension processor
LOCK	Lock bus during next instruction
NO OPERATION	
NOP	No operation
EXECUTION ENVIRONMENT CONTROL	
LMSW	Load machine status word
SMSW	Store machine status word

Figure 4f. Processor Control Instructions

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

Figure 4g. High Level Instructions

Memory Organization

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

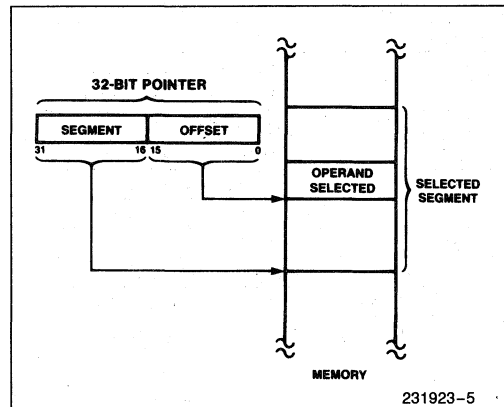


Figure 5. Two Component Address

Table 3. Segment Register Selection Rules

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

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

Most instructions need not explicitly specify which segment register is used. The correct segment register is automatically chosen according to the rules of Table 3. These rules follow the way programs are written (see Figure 6) as independent modules that require areas for code and data, a stack, and access to external data areas.

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

Addressing Modes

The 80C286 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)

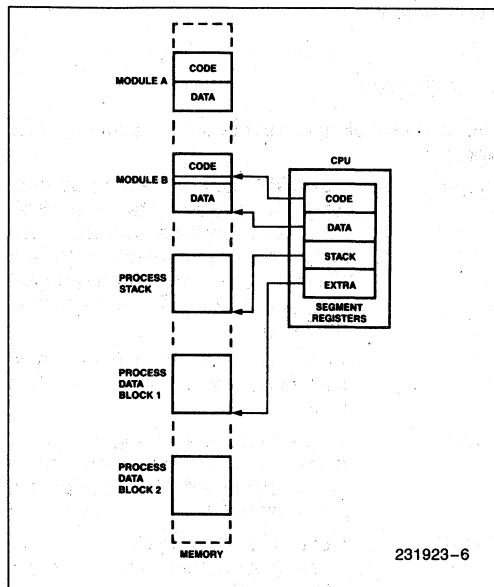


Figure 6. Segmented Memory Helps Structure Software

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

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

Combinations of these three address elements define the six memory addressing modes, described below.

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

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

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

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

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

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

Data Types

The 80C286 directly supports the following data types:

- Integer:** A signed binary numeric value contained in an 8-bit byte or a 16-bit word. All operations assume a 2's complement representation. Signed 32 and 64-bit integers are supported using the Numeric Data Processor, the 80287.
- Ordinal:** An unsigned binary numeric value contained in an 8-bit byte or 16-bit word.
- Pointer:** A 32-bit quantity, composed of a segment selector component and an offset component. Each component is a 16-bit word.
- String:** A contiguous sequence of bytes or words. A string may contain from 1 byte to 64K bytes.
- ASCII:** A byte representation of alphanumeric and control characters using the ASCII standard of character representation.
- BCD:** A byte (unpacked) representation of the decimal digits 0–9.
- Packed BCD:** A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble of the byte.
- Floating Point:** A signed 32, 64, or 80-bit real number representation. (Floating point operands are supported using the 80287 Numeric Processor).

Figure 7 graphically represents the data types supported by the 80C286.

either an 8-bit port address, specified in the instruction, or a 16-bit port address in the DX register. 8-bit port addresses are zero extended such that A₁₅–A₈ are LOW. I/O port addresses 00F8(H) through 00FF(H) are reserved.

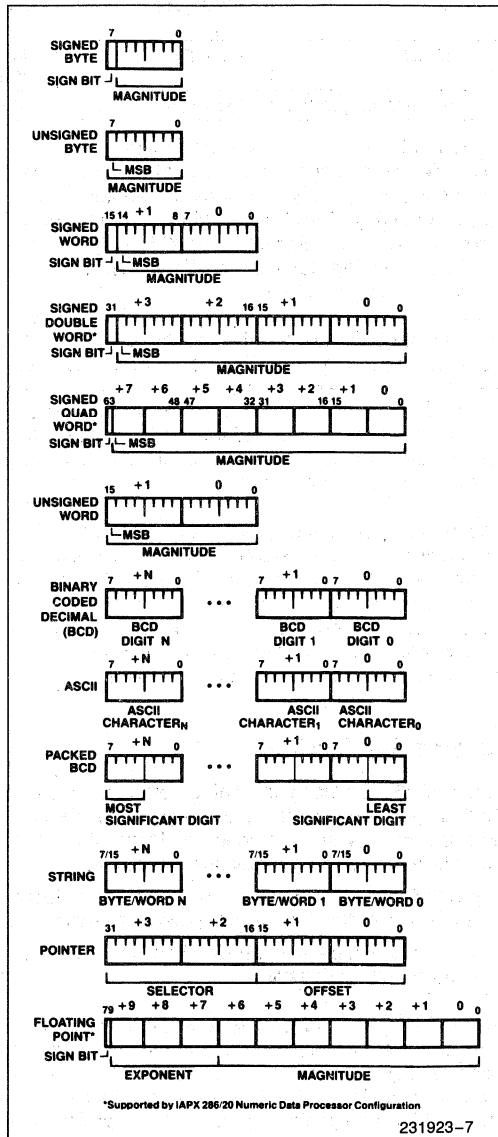


Figure 7. 80C286 Supported Data Types

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

Table 4. Interrupt Vector Assignments

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

Interrupts

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

A table containing up to 256 pointers defines the proper interrupt service routine for each interrupt. Interrupts 0–31, some of which are used for instruction exceptions, are reserved. For each interrupt, an 8-bit vector must be supplied to the 80C286 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 80C286 provides a maskable hardware interrupt request pin, INTR. Software enables this input by

setting the interrupt flag bit (IF) in the flag word. All 224 user-defined interrupt sources can share this input, yet they can retain separate interrupt handlers. An 8-bit vector read by the CPU during the interrupt acknowledge sequence (discussed in System Interface section) identifies the source of the interrupt.

Further maskable interrupts are disabled while servicing an interrupt by resetting the IF 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 80C286 will service neither further NMI requests, INTR requests, nor the processor extension segment overrun interrupt until an interrupt return (IRET) instruction is executed or the CPU is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. IF is cleared at the beginning of an NMI interrupt to inhibit INTR interrupts.

SINGLE STEP INTERRUPT

The 80C286 has an internal interrupt that allows programs to execute one instruction at a time. It is called the single step interrupt and is controlled by the single step flag bit (TF) in the flag word. Once this bit is set, an internal single step interrupt will occur after the next instruction has been executed. The interrupt clears the TF bit and uses an internally supplied vector of 1. The IRET instruction is used to set the TF bit and transfer control to the next instruction to be single stepped.

Interrupt Priorities

When simultaneous interrupt requests occur, they are processed in a fixed order as shown in Table 5. Interrupt processing involves saving the flags, return address, and setting CS:IP to point at the first instruction of the interrupt handler. If other interrupts remain enabled they are processed before the first instruction of the current interrupt handler is executed. The last interrupt processed is therefore the first one serviced.

Table 5. Interrupt Processing Order

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

Initialization and Processor Reset

Processor initialization or start up is accomplished by driving the RESET input pin HIGH. RESET forces the 80C286 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 80C286 begins execution in real address mode with the instruction at physical location FFFFFFF0(H). RESET also sets some registers to predefined values as shown in Table 6.

Table 6. 80C286 Initial Register State after RESET

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

HOLD must not be active during the time from the leading edge of RESET to 34 CLKs after the trailing edge of RESET.

Machine Status Word Description

The machine status word (MSW) records when a task switch takes place and controls the operating mode of the 80C286. It is a 16-bit register of which the lower four bits are used. One bit places the CPU into protected mode, while the other three bits, as shown in Table 7, control the processor extension interface. After RESET, this register contains FFF0(H) which places the 80C286 in 8086 real address mode.

Table 7. MSW Bit Functions

Bit Position	Name	Function
0	PE	Protected mode enable places the 80C286 into protected mode and cannot be cleared except by RESET.
1	MP	Monitor processor extension allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	Emulate processor extension causes a processor extension not present exception (number 7) on ESC instructions to allow emulating a processor extension.
3	TS	Task switched indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task.

The LMSW and SMSW instructions can load and store the MSW in real address mode. The recommended use of TS, EM, and MP is shown in Table 8.

Table 8. Recommended MSW Encodings For Processor Extension Control

TS	MP	EM	Recommended Use	Instructions Causing Exception 7
0	0	0	Initial encoding after RESET. 80C286 operation is identical to 8086, 88.	None
0	0	1	No processor extension is available. Software will emulate its function.	ESC
1	0	1	No processor extension is available. Software will emulate its function. The current processor extension context may belong to another task.	ESC
0	1	0	A processor extension exists.	None
1	1	0	A processor extension exists. The current processor extension context may belong to another task. The Exception 7 on WAIT allows software to test for an error pending from a previous processor extension operation.	ESC or WAIT

Halt

The HLT instruction stops program execution and prevents the CPU from using the local bus until restarted. Either NMI, INTR with IF = 1, or RESET will force the 80C286 out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

8086 REAL ADDRESS MODE

The 80C286 executes a fully upward-compatible superset of the 8086 instruction set in real address mode. In real address mode the 80C286 is object code compatible with 8086 and 8088 software. The real address mode architecture (registers and addressing modes) is exactly as described in the 80C286 Base Architecture section of this Functional Description.

Memory Size

Physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A₀ through A₁₉ and BHE. A₂₀ through A₂₃ should be ignored.

Memory Addressing

In real address mode physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A₀ through A₁₉ and BHE. Address bits A₂₀–A₂₃ may not always be zero in real mode. A₂₀–A₂₃ should not be used by the system while the 80C286 is operating in Real Mode.

The selector portion of a pointer is interpreted as the upper 16 bits of a 20-bit segment address. The lower four bits of the 20-bit segment address are always zero. Segment addresses, therefore, begin on multiples of 16 bytes. See Figure 8 for a graphic representation of address information.

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

Reserved Memory Locations

The 80C286 reserves two fixed areas of memory in real address mode (see Figure 9); system initialization

area and interrupt table area. Locations from addresses FFFF0(H) through FFFFF(H) are reserved for system initialization. Initial execution begins at location FFFF0(H). Locations 00000(H) through 003FF(H) are reserved for interrupt vectors.

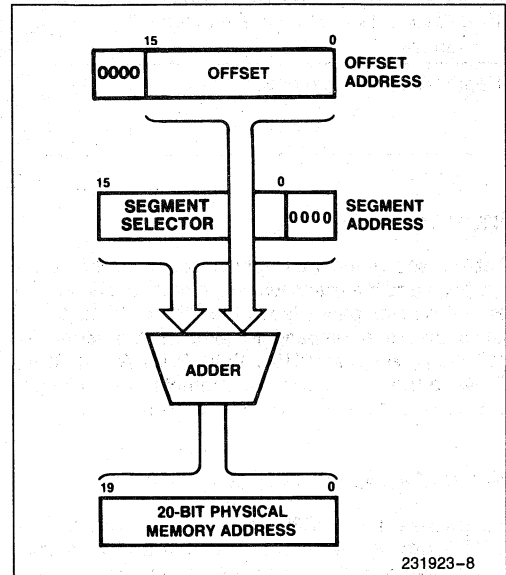


Figure 8. 8086 Real Address Mode Address Calculation

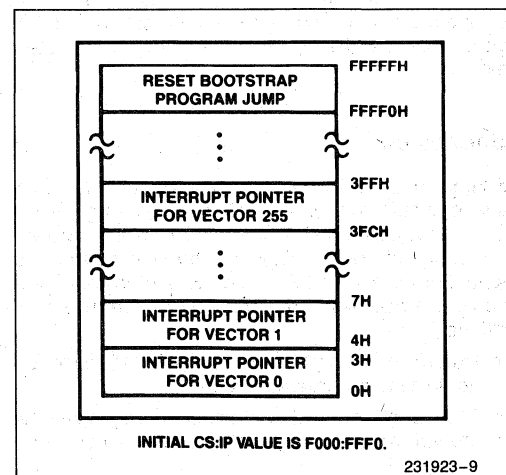


Figure 9. 8086 Real Address Mode Initially Reserved Memory Locations

Table 9. Real Address Mode Addressing Interrupts

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

Interrupts

Table 9 shows the interrupt vectors reserved for exceptions and interrupts which indicate an addressing error. The exceptions leave the CPU in the state existing before attempting to execute the failing instruction (except for PUSH, POP, PUSHA, or POPA). Refer to the next section on protected mode initialization for a discussion on exception 8.

Protected Mode Initialization

To prepare the 80C286 for protected mode, the LIDT instruction is used to load the 24-bit interrupt table base and 16-bit limit for the protected mode interrupt table. This instruction can also set a base and limit for the interrupt vector table in real address mode. After reset, the interrupt table base is initialized to 000000(H) and its size set to 03FF(H). These values are compatible with 8086, 88 software. LIDT should only be executed in preparation for protected mode.

Shutdown

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

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

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

PROTECTED VIRTUAL ADDRESS MODE

The 80C286 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 80C286 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 80C286 Base Architecture section of this Functional Description remain the same. Programs for the 8086, 88, 186, and real address mode 80C286 can be run in protected mode; however, embedded constants for segment selectors are different.

Memory Size

The protected mode 80C286 provides a 1 gigabyte virtual address space per task mapped into a 16 megabyte physical address space defined by the address pin A_{23-A_0} and \overline{BHE} . The virtual address space may be larger than the physical address space since any use of an address that does not map to a physical memory location will cause a restartable exception.

Memory Addressing

As in real address mode, protected mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector, however, specifies an index into a memory resident table rather than the upper 16-bits of a real memory address. The 24-bit

base address of the desired segment is obtained from the tables in memory. The 16-bit offset is added to the segment base address to form the physical address as shown in Figure 10. The tables are automatically referenced by the CPU whenever a segment register is loaded with a selector. All 80C286 instructions which load a segment register will reference the memory based tables without additional software. The memory based tables contain 8 byte values called descriptors.

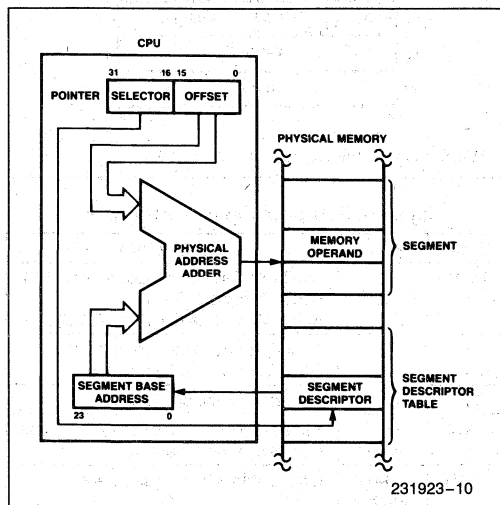


Figure 10. Protected Mode Memory Addressing

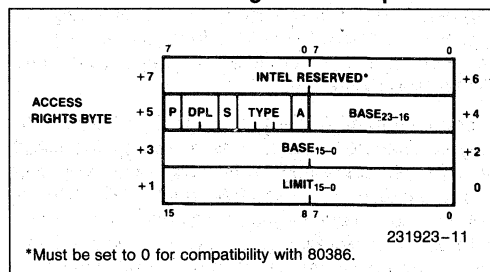
DESCRIPTORS

Descriptors define the use of memory. Special types of descriptors also define new functions for transfer of control and task switching. The 80C286 has segment descriptors for code, stack and data segments, and system control descriptors for special system data segments and control transfer operations. Descriptor accesses are performed as locked bus operations to assure descriptor integrity in multi-processor systems.

CODE AND DATA SEGMENT DESCRIPTORS (S = 1)

Besides segment base addresses, code and data descriptors contain other segment attributes including segment size (1 to 64K bytes), access rights (read only, read/write, execute only, and execute/read), and presence in memory (for virtual memory systems) (See Figure 11). Any segment usage violating a segment attribute indicated by the segment descriptor will prevent the memory cycle and cause an exception or interrupt.

Code or Data Segment Descriptor



*Must be set to 0 for compatibility with 80386.

Access Rights Byte Definition

Bit Position	Name	Function
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 Level (DPL)	Segment privilege attribute used in privilege tests.
4	Segment Descriptor (S)	S = 1 Code or Data (includes stacks) segment descriptor S = 0 System Segment Descriptor or Gate Descriptor
3	Executable (E)	Data segment descriptor type is: Expand up segment, offsets must be ≤ limit. Expand down segment, offsets must be > limit. Data segment may not be written into. Data segment may be written into.
2	Expansion Direction (ED)	
1	Writable (W)	
3	Executable (E)	Code Segment Descriptor type is: Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged. Code segment may not be read Code segment may be read.
2	Conforming (C)	
1	Readable (R)	
0	Accessed (A)	A = 0 Segment has not been accessed. A = 1 Segment selector has been loaded into segment register or used by selector test instructions.

Type
Field
Definition

Figure 11. Code and Data Segment Descriptor Formats

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

Data segments ($S = 1, E = 0$) may be either read-only or read-write as controlled by the W bit of the access rights byte. Read-only ($W = 0$) data segments may not be written into. Data segments may grow in two directions, as determined by the Expansion Direction (ED) bit: upwards ($ED = 0$) for data segments, and downwards ($ED = 1$) for a segment containing a stack. The limit field for a data segment descriptor is interpreted differently depending on the ED bit (see Figure 11).

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

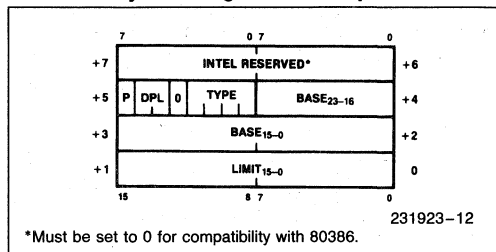
SYSTEM SEGMENT DESCRIPTORS ($S = 0$, $TYPE = 1-3$)

In addition to code and data segment descriptors, the protected mode 80C286 defines System Segment Descriptors. These descriptors define special system data segments which contain a table of descriptors (Local Descriptor Table Descriptor) or segments which contain the execution state of a task (Task State Segment Descriptor).

Figure 12 gives the formats for the special system data segment descriptors. The descriptors contain a 24-bit base address of the segment and a 16-bit limit. The access byte defines the type of descriptor, its state and privilege level. The descriptor contents are valid and the segment is in physical memory if $P = 1$. If $P = 0$, the segment is not valid. The DPL field is only used in Task State Segment descriptors and indicates the privilege level at which the descrip-

tor may be used (see Privilege). Since the Local Descriptor Table descriptor may only be used by a special privileged instruction, the DPL field is not used. Bit 4 of the access byte is 0 to indicate that it is a system control descriptor. The type field specifies the descriptor type as indicated in Figure 12.

System Segment Descriptor



System Segment Descriptor Fields

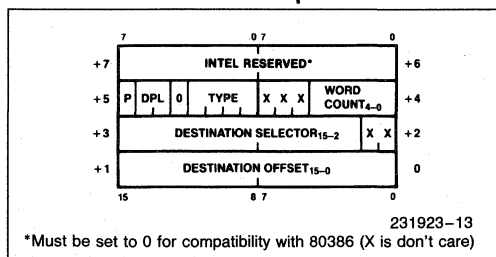
Name	Value	Description
TYPE	1	Available Task State Segment (TSS)
	2	Local Descriptor Table
	3	Busy Task State Segment (TSS)
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid
DPL	0-3	Descriptor Privilege Level
BASE	24-bit number	Base Address of special system data segment in real memory
LIMIT	16-bit number	Offset of last byte in segment

Figure 12. System Segment Descriptor Format

GATE DESCRIPTORS ($S = 0$, $TYPE = 4-7$)

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

Gate Descriptor



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 SELECTOR	16-bit selector	Selector to the target code segment (Call, Interrupt or Trap Gate)
		Selector to the target task state segment (Task Gate)
DESTINATION OFFSET	16-bit offset	Entry point within the target code segment

Figure 13. Gate Descriptor Format

Figure 13 shows the format of the gate descriptors. The descriptor contains a destination pointer that points to the descriptor of the target segment and the entry point offset. The destination selector in an interrupt gate, trap gate, and call gate must refer to a code segment descriptor. These gate descriptors contain the entry point to prevent a program from constructing and using an illegal entry point. Task gates may only refer to a task state segment. Since task gates invoke a task switch, the destination offset is not used in the task gate.

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

The access byte format is the same for all gate descriptors. P = 1 indicates that the gate contents are valid. P = 0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the de-

scriptor privilege level and specifies when this descriptor may be used by a task (refer to privilege discussion below). Bit 4 must equal 0 to indicate a system control descriptor. The type field specifies the descriptor type as indicated in Figure 13.

SEGMENT DESCRIPTOR CACHE REGISTERS

A segment descriptor cache register is assigned to each of the four segment registers (CS, SS, DS, ES). Segment descriptors are automatically loaded (cached) into a segment descriptor cache register (Figure 14) whenever the associated segment register is loaded with a selector. Only segment descriptors may be loaded into segment descriptor cache registers. Once loaded, all references to that segment of memory use the cached descriptor information instead of reaccessing the descriptor. The descriptor cache registers are not visible to programs. No instructions exist to store their contents. They only change when a segment register is loaded.

SELECTOR FIELDS

A protected mode selector has three fields: descriptor entry index, local or global descriptor table indicator (TI), and selector privilege (RPL) as shown in Figure 15. These fields select one of two memory based tables of descriptors, select the appropriate table entry and allow highspeed testing of the selector's privilege attribute (refer to privilege discussion below).

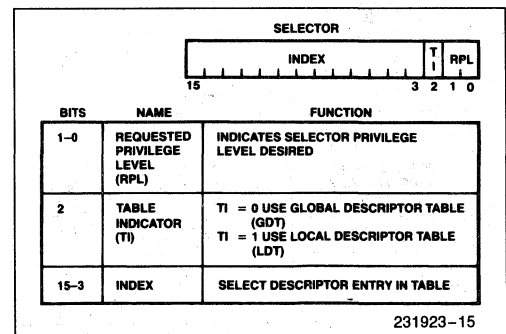


Figure 15. Selector Fields

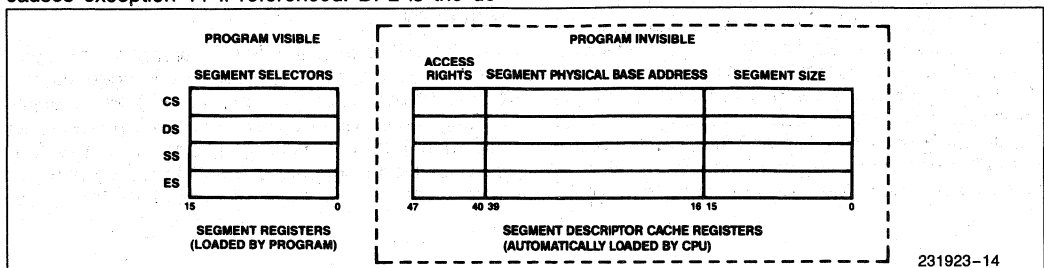


Figure 14. Descriptor Cache Registers

LOCAL AND GLOBAL DESCRIPTOR TABLES

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

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

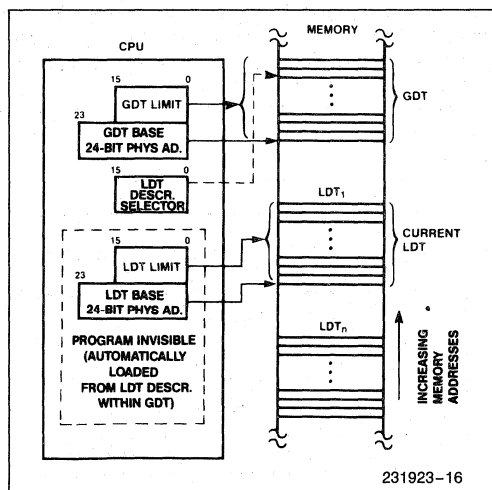


Figure 16. Local and Global Descriptor Table Definition

The LGDT and LLDT instructions load the base and limit of the global and local descriptor tables. LGDT and LLDT are privileged, i.e. they may only be executed by trusted programs operating at level 0. The LGDT instruction loads a six byte field containing the 16-bit table limit and 24-bit physical base address of the Global Descriptor Table as shown in Figure 17. The LDT instruction loads a selector which refers to a Local Descriptor Table descriptor containing the

base address and limit for an LDT, as shown in Figure 12.

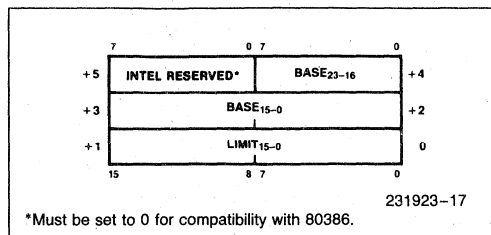


Figure 17. Global Descriptor Table and Interrupt Descriptor Table Data Type

INTERRUPT DESCRIPTOR TABLE

The protected mode 80C286 has a third descriptor table, called the Interrupt Descriptor Table (IDT) (see Figure 18), used to define up to 256 interrupts. It may contain only task gates, interrupt gates and trap gates. The IDT (Interrupt Descriptor Table) has a 24-bit physical base and 16-bit limit register in the CPU. The privileged LGDT instruction loads these registers with a six byte value of identical form to that of the LGDT instruction (see Figure 17 and Protected Mode Initialization).

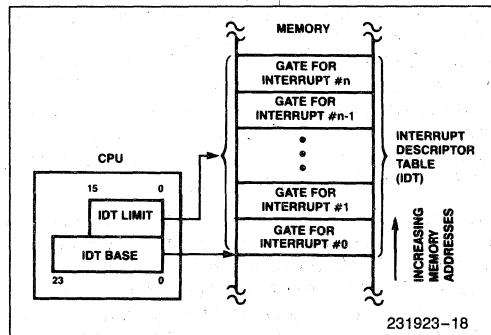
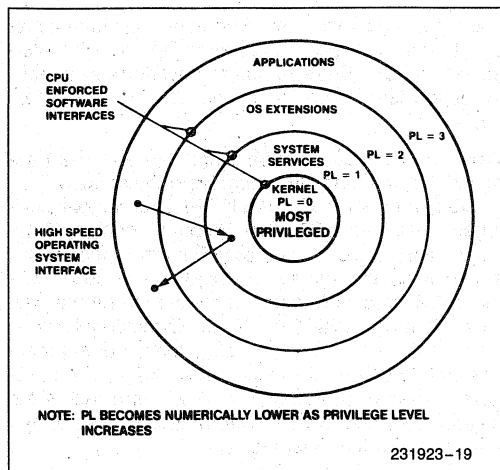


Figure 18. Interrupt Descriptor Table Definition

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

Privilege

The 80C286 has a four-level hierarchical privilege system which controls the use of privileged instructions and access to descriptors (and their associated segments) within a task. Four-level privilege, as shown in Figure 19, is an extension of the user/supervisor mode commonly found in minicomputers. The privilege levels are numbered 0 through 3.



Level 0 is the most privileged level. Privilege levels provide protection within a task. (Tasks are isolated by providing private LDT's for each task.) Operating system routines, interrupt handlers, and other system software can be included and protected within the virtual address space of each task using the four levels of privilege. Each task in the system has a separate stack for each of its privilege levels.

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

TASK PRIVILEGE

A task always executes at one of the four privilege levels. The task privilege level at any specific instant is called the Current Privilege Level (CPL) and is defined by the lower two bits of the CS register. CPL cannot change during execution in a single code segment. A task's CPL may only be changed by control transfers through gate descriptors to a new code segment (See Control Transfer). Tasks begin executing at the CPL value specified by the code segment selector within TSS when the task is initiated via a task switch operation (See Figure 20). A task executing at Level 0 can access all data segments defined in the GDT and the task's LDT and is considered the most trusted level. A task executing a Level 3 has the most restricted access to data and is considered the least trusted level.

DESCRIPTOR PRIVILEGE

Descriptor privilege is specified by the Descriptor Privilege Level (DPL) field of the descriptor access byte. DPL specifies the least trusted task privilege

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

SELECTOR PRIVILEGE

Selector privilege is specified by the Requested Privilege Level (RPL) field in the least significant two bits of a selector. Selector RPL may establish a less trusted privilege level than the current privilege level for the use of a selector. This level is called the task's effective privilege level (EPL). RPL can only reduce the scope of a task's access to data with this selector. A task's effective privilege is the numeric maximum of RPL and CPL. A selector with RPL = 0 imposes no additional restriction on its use while a selector with RPL = 3 can only refer to segments at privilege Level 3 regardless of the task's CPL. RPL is generally used to verify that pointer parameters passed to a more trusted procedure are not allowed to use data at a more privileged level than the caller (refer to pointer testing instructions).

Descriptor Access and Privilege Validation

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

DATA SEGMENT ACCESS

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

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

If the privilege checks fail (e.g. DPL is numerically less than the maximum of CPL and RPL) or an incorrect type of descriptor is referenced (e.g. gate de-

scriptor or execute only code segment) exception 13 occurs. If the segment is not present, exception 11 is generated.

Instructions that load selectors into SS must refer to data segment descriptors for writable data segments. The descriptor privilege (DPL) and RPL must equal CPL. All other descriptor types or a privilege level violation will cause exception 13. A not present fault causes exception 12.

CONTROL TRANSFER

Four types of control transfer can occur when a selector is loaded into CS by a control transfer operation (see Table 10). Each transfer type can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules (e.g. JMP through a call gate or RET to a Task State Segment) will cause exception 13.

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

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

When a JMP or CALL references a Task State Segment descriptor, the descriptor DPL must be the same or less privileged than the task's CPL. Refer-

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

The privilege rules on control transfer require:

- JMP or CALL direct to a code segment (code segment descriptor) can only be to a conforming segment with DPL of equal or greater privilege than CPL or a non-conforming segment at the same privilege level.
- interrupts within the task or calls that may change privilege levels, can only transfer control through a gate at the same or a less privileged level than CPL to a code segment at the same or more privileged level than CPL.
- return instructions that don't switch tasks can only return control to a code segment at the same or less privileged level.
- task switch can be performed by a call, jump or interrupt which references either a task gate or task state segment at the same or less privileged level.

Table 10. Descriptor Types Used for Control Transfer

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

PRIVILEGE LEVEL CHANGES

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

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

Protection

The 80C286 includes mechanisms to protect critical instructions that affect the CPU execution state (e.g. HLT) and code or data segments from improper usage. These protection mechanisms are grouped into three forms:

Restricted *usage* of segments (e.g. no write allowed to read-only data segments). The only segments available for use are defined by descriptors in the Local Descriptor Table (LDT) and Global Descriptor Table (GDT).

Restricted *access* to segments via the rules of privilege and descriptor usage.

Privileged instructions or operations that may only be executed at certain privilege levels as determined by the CPL and I/O Privilege Level (IOPL). The IOPL is defined by bits 14 and 13 of the flag word.

These checks are performed for all instructions and can be split into three categories: segment load checks (Table 11), operand reference checks (Table 12), and privileged instruction checks (Table 13). Any violation of the rules shown will result in an exception. A not-present exception related to the stack segment causes exception 12.

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

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

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

Table 11. Segment Register Load Checks

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

Table 12. Operand Reference Checks

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

NOTE:

Carry out in offset calculations is ignored.

Table 13. Privileged Instruction Checks

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

EXCEPTIONS

The 80C286 detects several types of exceptions and interrupts, in protected mode (see Table 14). Most are restartable after the exceptional condition is removed. Interrupt handlers for most exceptions can read an error code, pushed on the stack after the return address, that identifies the selector involved (0 if none). The return address normally points to the failing instruction, including all leading prefixes. For a processor extension segment overrun exception, the return address will not point at the ESC instruction that caused the exception; however, the processor extension registers may contain the address of the failing instruction.

Table 14. Protected Mode Exceptions

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

NOTE:

1. When a PUSH or POP instruction attempts to wrap around the stack segment, the machine state after the exception will not be restartable because stack segment wrap around is not permitted. This condition is identified by the value of the saved SP being either 0000(H), 0001(H), FFFF(H), or FFFF(H).

2. These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.

These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.

All these checks are performed for all instructions and can be split into three categories: segment load checks (Table 11), operand reference checks (Table 12), and privileged instruction checks (Table 13). Any violation of the rules shown will result in an exception. A not-present exception causes exception 11 or 12 and is restartable.

Special Operations

TASK SWITCH OPERATION

The 80C286 provides a built-in task switch operation which saves the entire 80C286 execution state (registers, address space, and a link to the previous task), loads a new execution state, and commences execution in the new task. Like gates, the task switch operation is invoked by executing an inter-segment JMP or CALL instruction which refers to a Task State Segment (TSS) or task gate descriptor in the GDT or LDT. An INT n instruction, exception, or external interrupt may also invoke the task switch operation by selecting a task gate descriptor in the associated IDT descriptor entry.

The TSS descriptor points at a segment (see Figure 20) containing the entire 80C286 execution state while a task gate descriptor contains a TSS selector. The limit field of the descriptor must be > 002B(H).

Each task must have a TSS associated with it. The current TSS is identified by a special register in the 80C286 called the Task Register (TR). This register contains a selector referring to the task state segment descriptor that defines the current TSS. A hidden base and limit register associated with TR are loaded whenever TR is loaded with a new selector.

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

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

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

PROCESSOR EXTENSION CONTEXT SWITCHING

The context of a processor extension (such as the 80287 numerics processor) is not changed by the task switch operation. A processor extension context need only be changed when a different task attempts to use the processor extension (which still contains the context of a previous task). The 80C286 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 80C286 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 80C286 provides several instructions to speed pointer testing and consistency checks for maintaining system integrity (see Table 15). These instructions use the memory management hardware to verify that a selector value refers to an appropriate segment without risking an exception. A condition flag (ZF) indicates whether use of the selector or segment will cause an exception.

tions use the memory management hardware to verify that a selector value refers to an appropriate segment without risking an exception. A condition flag (ZF) indicates whether use of the selector or segment will cause an exception.

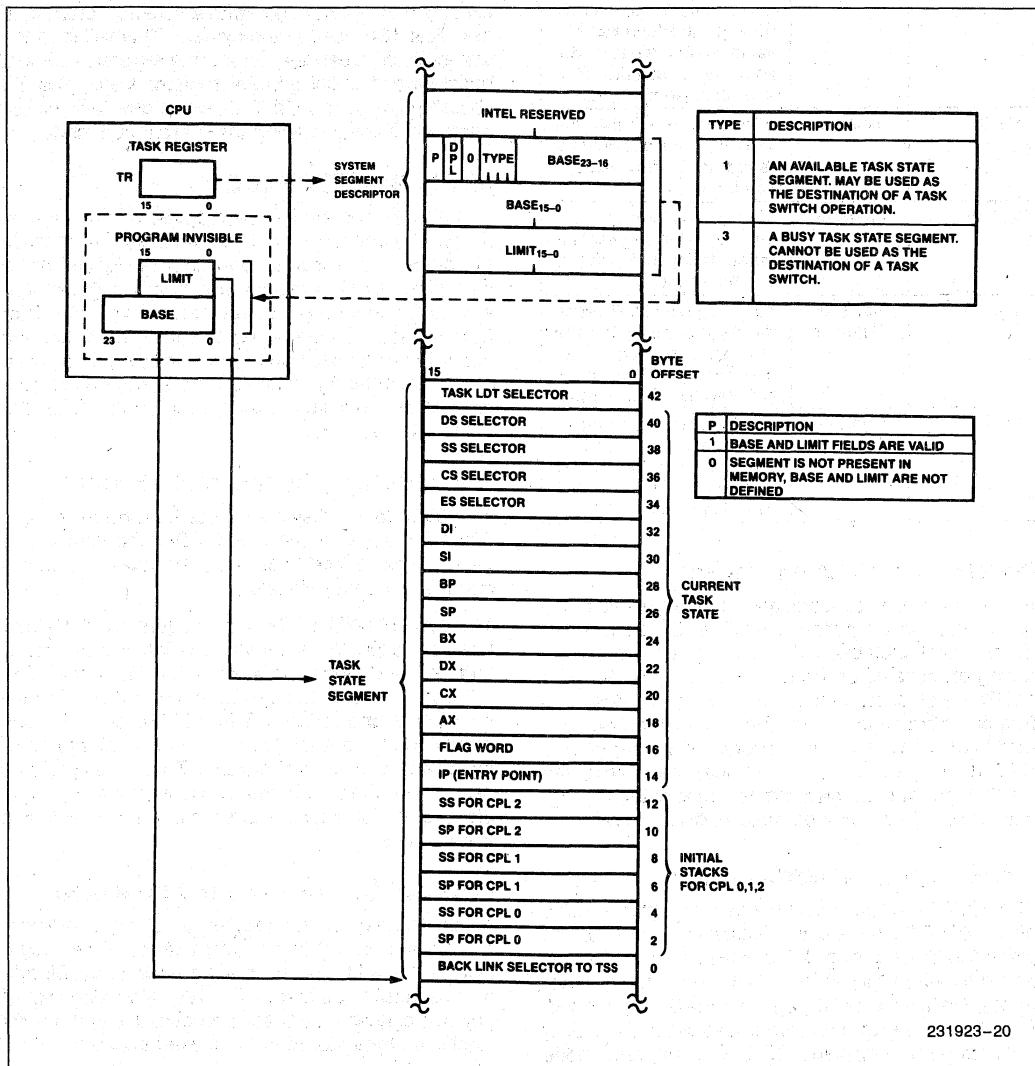


Figure 20. Task State Segment and TSS Registers

Table 15. 80C286 Pointer Test Instructions

Instruction	Operands	Function
ARPL	Selector, Register	Adjust Requested Privilege Level: adjusts the RPL of the selector to the numeric maximum of current selector RPL value and the RPL value in the register. Set zero flag if selector RPL was changed by ARPL.
VERR	Selector	VERify for Read: sets the zero flag if the segment referred to by the selector can be read.
VERW	Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL	Register, Selector	Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

DOUBLE FAULT AND SHUTDOWN

If two separate exceptions are detected during a single instruction execution, the 80C286 performs the double fault exception (8). If an execution occurs during processing of the double fault exception, the 80C286 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 80C286 out of shutdown. Shutdown is externally signalled via a HALT bus operation with A₁ LOW.

PROTECTED MODE INITIALIZATION

The 80C286 initially executes in real address mode after RESET. To allow initialization code to be placed at the top of physical memory, A₂₃–A₂₀ will be HIGH when the 80C286 performs memory references relative to the CS register until CS is changed. A₂₃–A₂₀ will be zero for references to the DS, ES, or SS segments. Changing CS in real address mode will force A₂₃–A₂₀ LOW whenever CS is used again. The initial CS:IP value of F000:FFF0 provides 64K bytes of code space for initialization code without changing CS.

Protected mode operation requires several registers to be initialized. The GDT and IDT base registers must refer to a valid GDT and IDT. After executing the LMSW instruction to set PE, the 80C286 must

immediately execute an intra-segment JMP instruction to clear the instruction queue of instructions decoded in real address mode.

To force the 80C286 CPU registers to match the initial protected mode state assumed by software, execute a JMP instruction with a selector referring to the initial TSS used in the system. This will load the task register, local descriptor table register, segment registers and initial general register state. The TR should point at a valid TSS since any task switch operation involves saving the current task state.

SYSTEM INTERFACE

The 80C286 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 80C286 family includes several devices to generate standard system buses such as the IEEE 796 standard MULTIBUS.

Bus Interface Signals and Timing

The 80C286 microsystem local bus interfaces the 80C286 to local memory and I/O components. The interface has 24 address lines, 16 data lines, and 8 status and control signals.

The 80C286 CPU, 82C284 clock generator, 82C288 bus controller, transceivers, and latches provide a buffered and decoded system bus interface. The 82C284 generates the system clock and synchronizes READY and RESET. The 82C288 converts bus operation status encoded by the 80C286 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 D₇–D₀ while odd bytes are transferred over D₁₅–D₈. Even-addressed words are transferred over D₁₅–D₀ in one bus cycle, while odd-addressed word require two bus operations. The first transfers data on D₁₅–D₈, and the second transfers data on D₇–D₀. 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 ($D_{15}-D_8$) are accessed with odd I/O addresses. Devices on the lower data byte are accessed with even I/O addresses. An interrupt controller such as Intel's 82C59A-2 must be connected to the lower data byte (D_7-D_0) for proper return of the interrupt vector.

Bus Operation

The 80C286 uses a double frequency system clock (CLK input) to control bus timing. All signals on the local bus are measured relative to the system CLK input. The CPU divides the system clock by 2 to produce the internal processor clock, which determines bus state. Each processor clock is composed of two system clock cycles named phase 1 and phase 2. The 82C284 clock generator output (PCLK) identifies the next phase of the processor clock. (See Figure 21.)

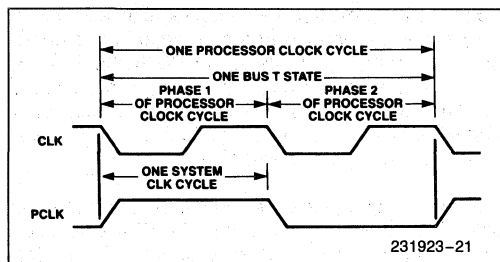


Figure 21. System and Processor Clock Relationships

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

The 80C286 bus has three basic states: idle (T_i), send status (T_s), and perform command (T_c). The 80C286 CPU also has a fourth local bus state called hold (T_h). T_h indicates that the 80C286 has surrendered control of the local bus to another bus master in response to a HOLD request.

Each bus state is one processor clock long. Figure 22 shows the four 80C286 local bus states and allowed transitions.

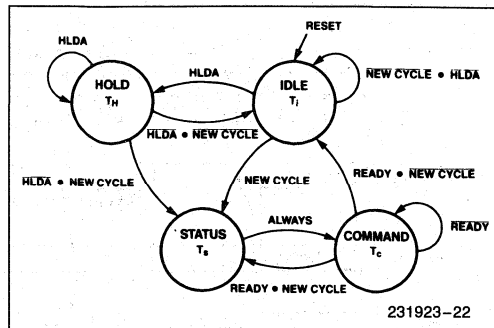


Figure 22. 80C286 Bus States

Bus States

The idle (T_i) state indicates that no data transfers are in progress or requested. The first active state T_s is signaled by status line $\overline{S1}$ or $\overline{S0}$ going LOW and identifying phase 1 of the processor clock. During T_s , the command encoding, the address, and data (for a write operation) are available on the 80C286 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 80C286 will float* all address, data, and status output pins enabling another bus master to use the local bus. The 80C286 HOLD input signal is used to place the 80C286 into the T_h state. The 80C286 HLDA output signal indicates that the CPU has entered T_h .

Pipelined Addressing

The 80C286 uses a local bus interface with pipelined timing to allow as much time as possible for data access. Pipelined timing allows a new bus operation to be initiated every two processor cycles, while allowing each individual bus operation to last for three processor cycles.

The timing of the address outputs is pipelined such that the address of the next bus operation becomes available during the current bus operation. Or in other words, the first clock of the next bus operation is overlapped with the last clock of the current bus operation. Therefore, address decode and routing logic can operate in advance of the next bus operation.

*NOTE: See section on bus hold circuitry.

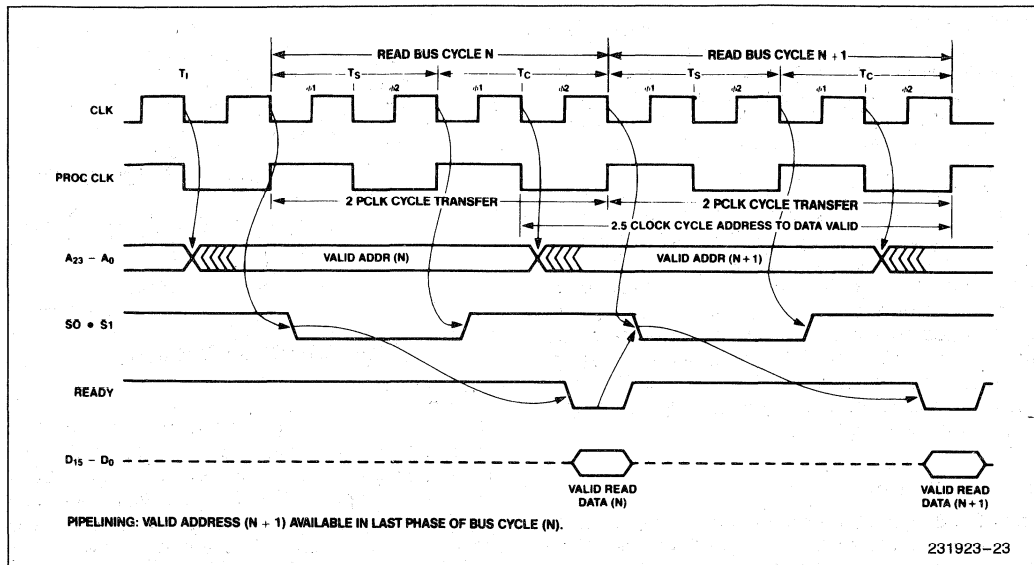


Figure 23. Basic Bus Cycle

External address latches may hold the address stable for the entire bus operation, and provide additional AC and DC buffering.

The 80C286 does not maintain the address of the current bus operation during all T_C states. Instead, the address for the next bus operation may be emitted during phase 2 of any T_C . The address remains valid during phase 1 of the first T_C to guarantee hold time, relative to ALE, for the address latch inputs.

Bus Control Signals

The 82C288 bus controller provides control signals; address latch enable (ALE), Read/Write commands, data transmit/receive (DT/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/R). DEN enables the data transceivers; while DT/R controls transceiver direction. DEN and DT/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 80C286 local bus.

Command extension allows additional time for external devices to respond to a command and is analogous to inserting wait states on the 8086. External logic can control the duration of any bus operation such that the operation is only as long as necessary. The READY input signal can extend any bus operation for as long as necessary.

Command delay allows an increase of address or write data setup time to system bus command active for any bus operation by delaying when the system bus command becomes active. Command delay is controlled by the 82C288 CMDLY input. After T_S , the bus controller samples CMDLY at each falling 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/R.

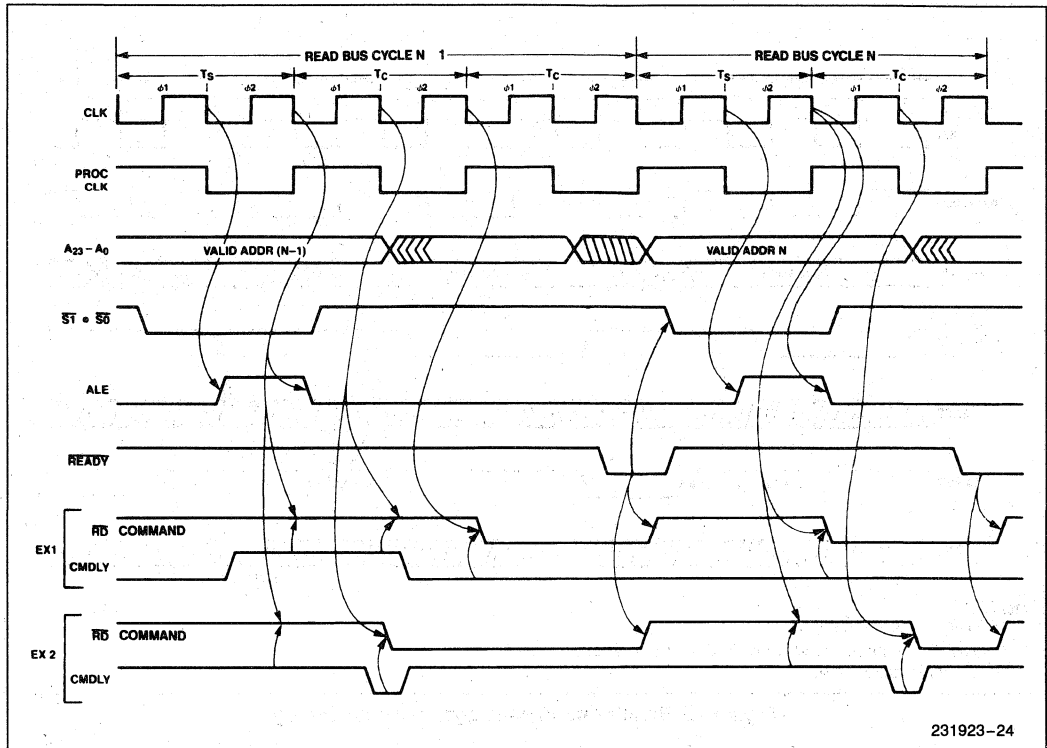


Figure 24. CMDLY Controls the Leading Edge of Command Signal

Figure 24 illustrates four uses of CMDLY. Example 1 shows delaying the read command two system CLKs for cycle N-1 and no delay for cycle N, and example 2 shows delaying the read command one system CLK for cycle N-1 and one system CLK delay for cycle N.

Bus Cycle Termination

At maximum transfer rates, the 80C286 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 80C286 local bus. The bus master and bus controller enter T_c directly after T_s and continue executing T_c cycles until terminated by **READY**.

READY Operation

The current bus master and 82C288 bus controller terminate each bus operation simultaneously to achieve maximum bus operation bandwidth. Both are informed in advance by **READY** active (open-collector output from 82C284) which identifies the last T_c cycle of the current bus operation. The bus master and bus controller must see the same sense

of the **READY** signal, thereby requiring **READY** be synchronous to the system clock.

Synchronous Ready

The 82C284 clock generator provides **READY** synchronization from both synchronous and asynchronous sources (see Figure 25). 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 T_c . 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 82C284 SRDY setup and hold time requirements. But the 82C284 asynchronous ready input (ARDY) is designed to accept such signals. The **ARDY** input is sampled at the beginning of each T_c cycle by 82C284 synchronization logic. This provides one system CLK cycle time to resolve its value before broadcasting it to the bus master and bus controller.

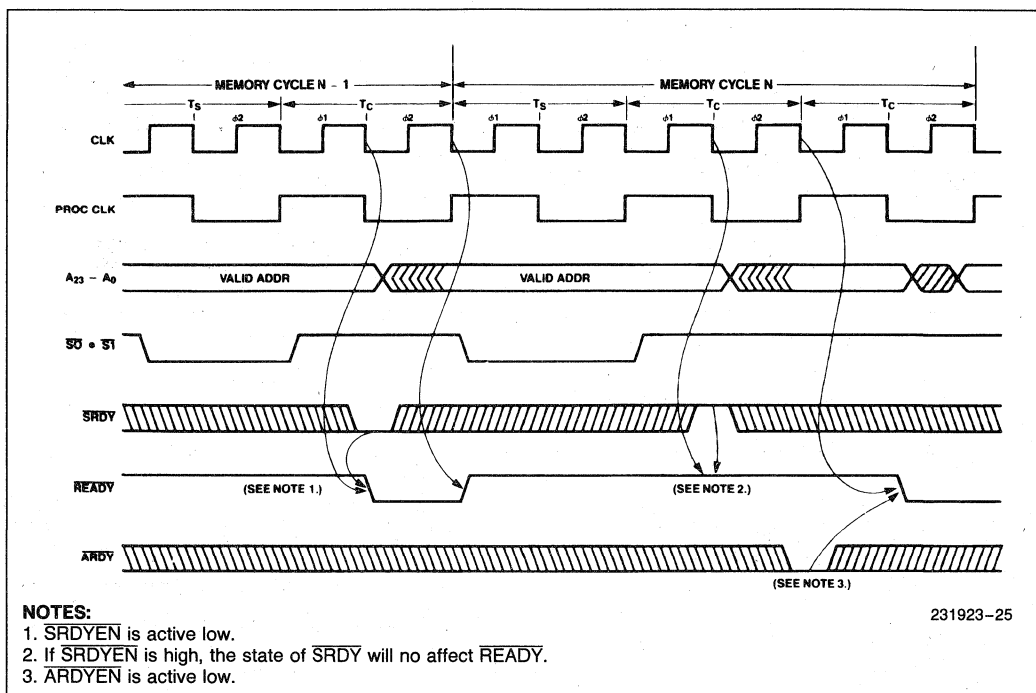


Figure 25. Synchronous and Asynchronous Ready

$\overline{\text{ARDY}}$ or $\overline{\text{ARDYEN}}$ must be HIGH at the end of T_S . $\overline{\text{ARDY}}$ cannot be used to terminate bus cycle with no wait states.

Each ready input of the 82C284 has an enable pin ($\overline{\text{SRDYEN}}$ and $\overline{\text{ARDYEN}}$) to select whether the current bus operation will be terminated by the synchronous or asynchronous ready. Either of the ready inputs may terminate a bus operation. These enable inputs are active low and have the same timing as their respective ready inputs. Address decode logic usually selects whether the current bus operation should be terminated by $\overline{\text{ARDY}}$ or $\overline{\text{SRDY}}$.

Data Bus Control

Figures 26, 27, and 28 show how the $\text{DT}/\overline{\text{R}}$, DEN , data bus, and address signals operate for different combinations of read, write, and idle bus operations. $\text{DT}/\overline{\text{R}}$ goes active (LOW) for a read operation. $\text{DT}/\overline{\text{R}}$ remains HIGH before, during, and between write operations.

The data bus is driven with write data during the second phase of T_S . The delay in write data timing allows the read data drivers, from a previous read cycle, sufficient time to enter 3-state OFF* before the 80C286 CPU begins driving the local data bus for write operations. Write data will always remain valid for one system clock past the last T_C to provide sufficient hold time for Multibus or other similar memory or I/O systems. During write-read or write-idle sequences the data bus enters 3-state OFF* during the second phase of the processor cycle after the last T_C . In a write-write sequence the data bus does not enter 3-state OFF* between T_C and T_S .

Bus Usage

The 80C286 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.

*NOTE: See section on bus hold circuitry.

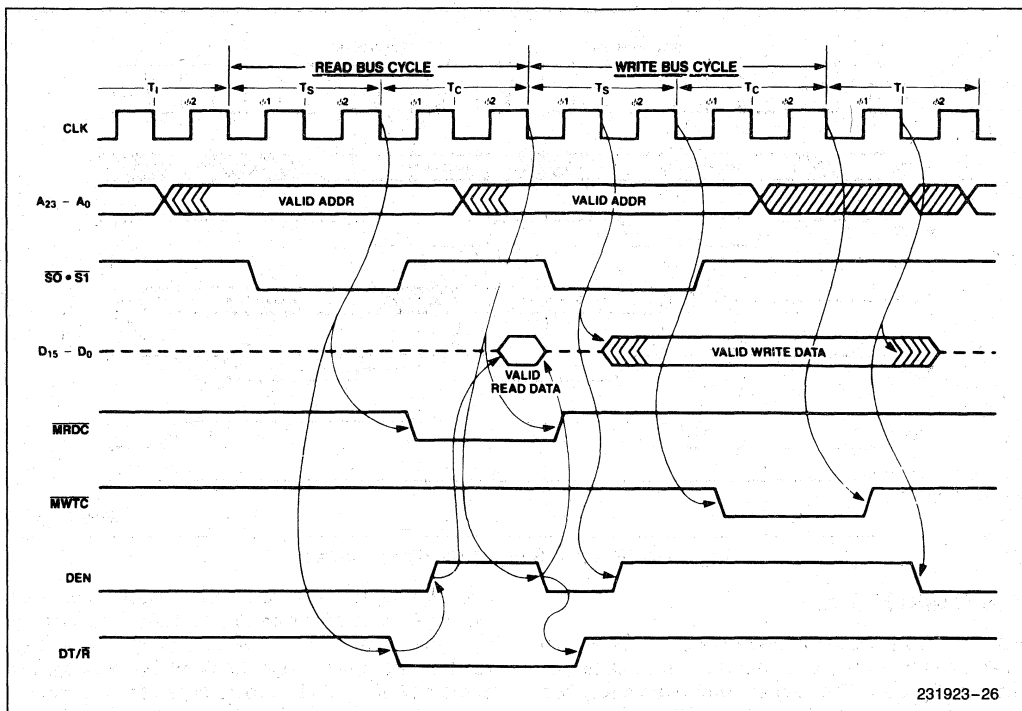


Figure 26. Back to Back Read-Write Cycles

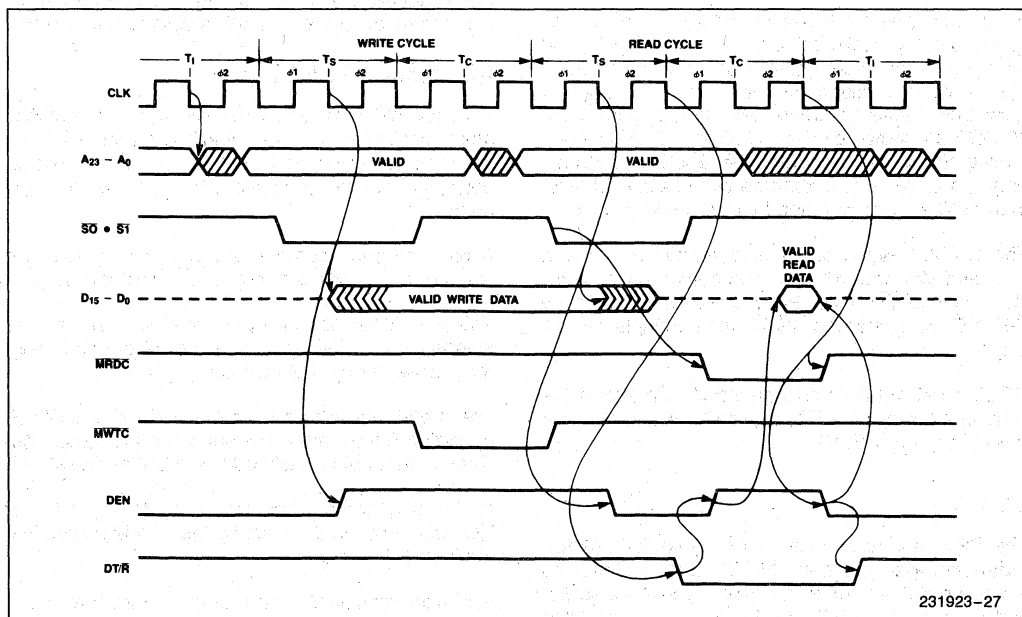


Figure 27. Back to Back Write-Read Cycles

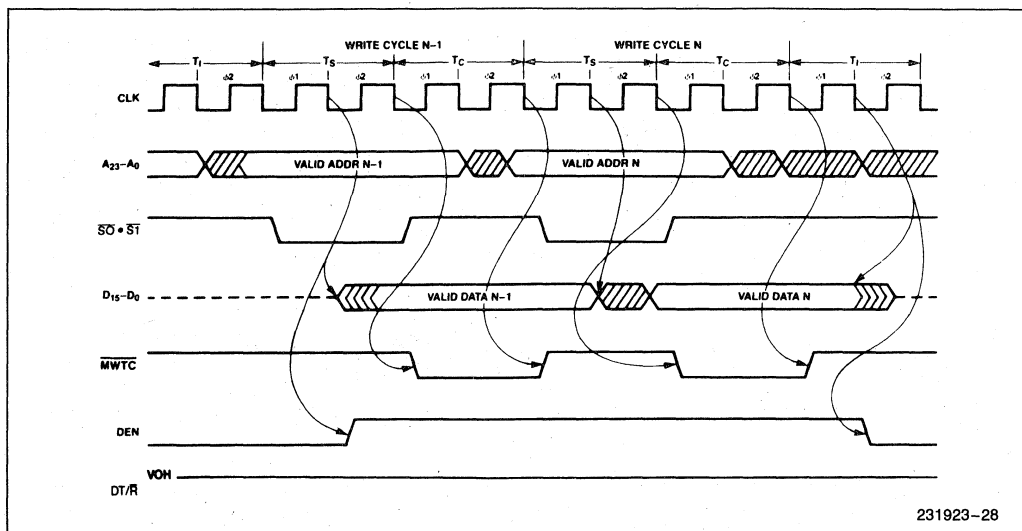


Figure 28. Back to Back Write-Write Cycles

HOLD and HLDA

HOLD AND HLDA allow another bus master to gain control of the local bus by placing the 80C286 bus into the T_h state. The sequence of events required to pass control between the 80C286 and another local bus master are shown in Figure 29.

In this example, the 80C286 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 80C286 as shown by the HOLD signal. After completing the write operation, the 80C286 performs one T_1 bus cycle, to guarantee write data hold time, then enters T_h as signaled by HLDA going active.

The \overline{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.

Lock

The CPU asserts an active lock signal during Interrupt-Acknowledge cycles, the XCHG instruction, and during some descriptor accesses. Lock is also asserted when the LOCK prefix is used. The LOCK prefix may be used with the following ASM-286 assembly instructions; MOVS, INS, and OUTS. For bus cycles other than Interrupt-Acknowledge cycles,

Lock will be active for the first and subsequent cycles of a series of cycles to be locked. Lock will not be shown active during the last cycle to be locked. For the next-to-last cycle, Lock will become inactive at the end of the first T_c regardless of the number of wait-states inserted. For Interrupt-Acknowledge cycles, Lock will be active for each cycle, and will become inactive at the end of the first T_c for each cycle regardless of the number of wait-states inserted.

Instruction Fetching

The 80C286 Bus Unit (BU) will fetch instructions ahead of the current instruction being executed. This activity is called prefetching. It occurs when the local bus would otherwise be idle and obeys the following rules:

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

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

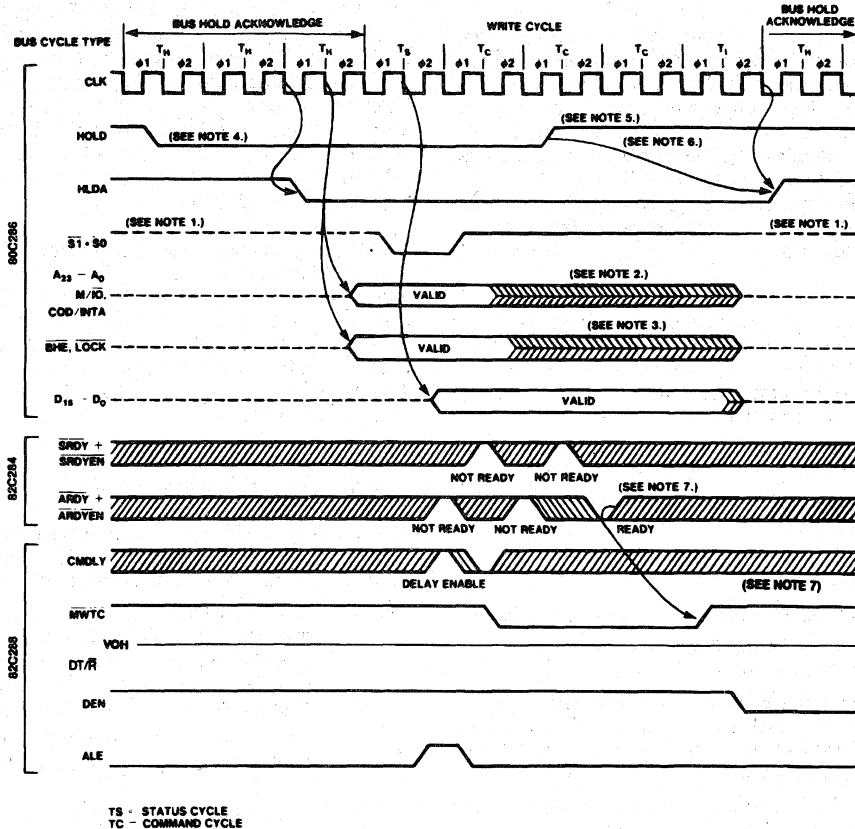
The prefetcher will perform only a byte code fetch operation for control transfers to an instruction beginning on a numerically odd physical address.

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

In real address mode, the prefetcher may fetch up to 6 bytes beyond the last control transfer or HLT instruction in a code segment.

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

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



NOTES:

1. Status lines are not driven by 80C286, yet remain high due to internal pullup resistors during HOLD state. See section on bus hold circuitry.
2. Address, M/IO and COD/INTA may start floating during any T_C depending on when internal 80C286 bus arbiter decides to release bus to external HOLD. The float starts in ϕ_2 of T_C . See section on bus hold circuitry.
3. BHE and LOCK may start floating after the end of any T_C depending on when internal 80C286 bus arbiter decides to release bus to external HOLD. The float starts in ϕ_1 of T_C . See section on bus hold circuitry.
4. The minimum HOLD to HLDA time is shown. Maximum is one T_H longer.
5. The earliest HOLD time is shown. It will always allow a subsequent memory cycle if pending is shown.
6. The minimum HOLD to HLDA time is shown. Maximum is a function of the instruction, type of bus cycle and other machine state (i.e., Interrupts, Waits, Lock, etc.).
7. Asynchronous ready allows termination of the cycle. Synchronous ready does not signal ready in this example. Synchronous ready state is ignored after ready is signaled via the asynchronous input.

Figure 29. MULTIBUS® Write Terminated by Asynchronous Ready with Bus Hold

Processor Extension Transfers

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

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

NOTE:

Odd-aligned numerics instructions should be avoided when using an 80C286 system running six or more memory-write wait-states. The 80C286 can generate an incorrect numerics address if all the following conditions are met:

- Two floating point (FP) instructions are fetched and in the 80C286 queue.
- The first FP instruction is any floating point store except FSTSW AX.
- The second FP instruction is any floating point store except FSTSW AX.
- The second FP instruction accesses memory.
- The operand of the first instruction is aligned on an odd memory address.
- More than five wait-states are inserted during either of the last two memory write transfers (transferred as two bytes for odd aligned operands) of the first instruction.

The second FP instruction operand address will be incremented by one if these conditions are met. These conditions are most likely to occur in a multi-master system. For a hardware solution, contact your local Intel representative.

Ten or more command delays should not be used when accessing the numerics coprocessor. Excessive command delays can cause the 80C286 and 80287 to lose synchronization.

Interrupt Acknowledge Sequence

Figure 30 illustrates an interrupt acknowledge sequence performed by the 80C286 in response to an

INTR input. An interrupt acknowledge sequence consists of two INTA bus operations. The first allows a master 82C59A-2 Programmable Interrupt Controller (PIC) to determine which if any of its slaves should return the interrupt vector. An eight bit vector is read on D0–D7 of the 80C286 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 30), onto the local address bus for distribution to slave interrupt controllers via the system address bus. The 80C286 emits the 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 80C286 between INTA bus operations to allow for the minimum INTA to INTA time and CAS (cascade address) out delay of the 82C59A-2. The second INTA bus operation must always have at least one extra T_c state added via logic controlling READY. This is needed to meet the 82C59A-2 minimum INTA pulse width.

Local Bus Usage Priorities

The 80C286 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 LOCK either explicitly (via the LOCK instruction prefix) or implicitly (i.e. some segment descriptor accesses, interrupt acknowledge sequence, or an XCHG with memory).

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

The second or third cycle of a processor extension data transfer.

Local bus request via HOLD input.

Processor extension data operand transfer via PEREQ input.

Data transfer performed by EU as part of an instruction.

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

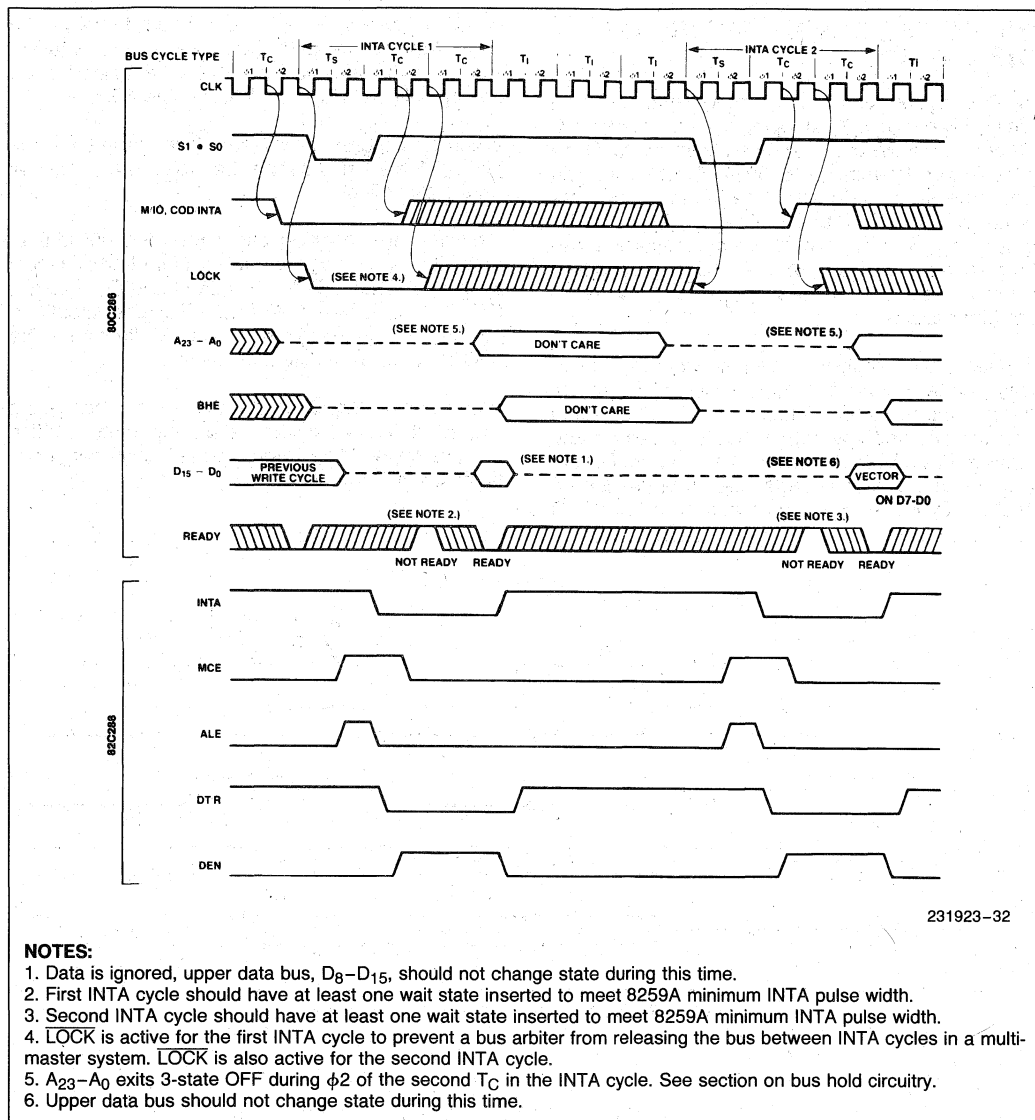


Figure 30. Interrupt Acknowledge Sequence

Halt or Shutdown Cycles

The 80C286 externally indicates halt or shutdown conditions as a bus operation. These conditions occur due to a HLT instruction or multiple protection exceptions while attempting to execute one instruction. A halt or shutdown bus operation is signalled when $\overline{S1}$, $\overline{S0}$ and COD/ \overline{INTA} are LOW and M/ \overline{IO} is HIGH. A₁ HIGH indicates halt, and A₁ LOW indicates shutdown. The 82C288 bus controller does not issue ALE, nor is READY required to terminate a halt or shutdown bus operation.

During halt or shutdown, the 80C286 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 80C286 out of either halt or shutdown. An INTR, if interrupts are enabled, or a processor extension segment overrun exception will also force the 80C286 out of halt.

THE POWER-DOWN FEATURE OF THE 80C286

The 80C286, unlike the HMOS part, can enter into a power-down mode. By stopping the processor CLK, the processor will enter a power-down mode. Once in the power-down mode, all 80C286 outputs remain static (the same state as before the mode was entered). The 80C286 D.C. specification I_{CCS} rates the amount of current drawn by the processor when in the power-down mode. When the CLK is reapplied to the processor, it will resume execution where it was interrupted.

In order to obtain maximum benefits from the power-down mode, certain precautions should be taken. When in the power-down mode, all 80C286 outputs remain static and any output that is turned on and remains in a HIGH condition will source current when loaded. Best low-power performance can be obtained by first putting the processor in the HOLD

condition (turning off all of the output buffers), and then stopping the processor CLK in the phase 2 state. In this condition, any output that is loaded will source only the "Bus Hold Sustaining Current".

When stopping the processor clock, minimum clock high and low times cannot be violated (no glitches on the clock line).

Violating this condition can cause the 80C286 to erase its internal register states. Note that all inputs to the 80C286 (CLK, HOLD, PEREQ, RESET, READY, INTR, NMI, BUSY, and ERROR) should be at V_{CC} or V_{SS} ; any other value will cause the 80C286 to draw additional current.

When coming out of power-down mode, the system CLK must be started with the same polarity in which it was stopped. An example power down sequence is shown in Figure 31.

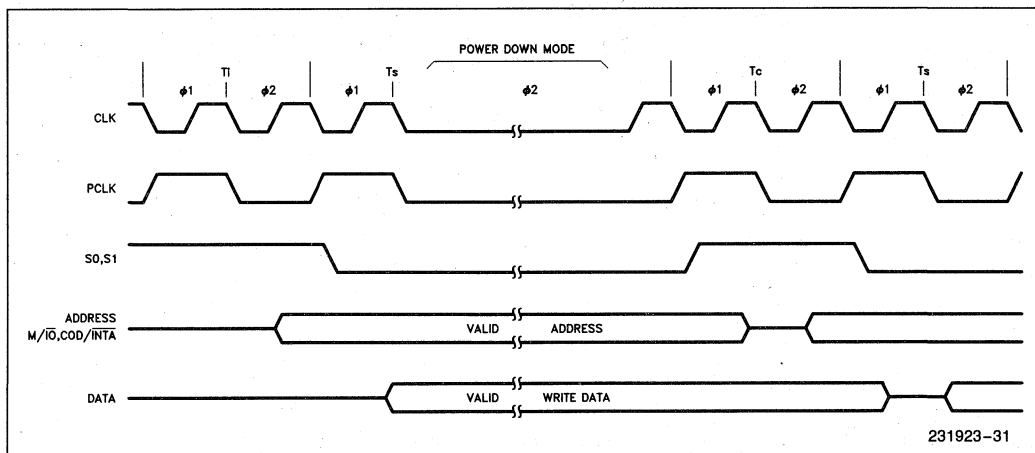


Figure 31. Example Power-Down Sequence

BUS HOLD CIRCUITRY

To avoid high current conditions caused by floating inputs to peripheral CMOS devices and eliminate the need for pull-up/down resistors, "bus-hold" circuitry has been used on all tri-state 80C286 outputs. See Table A for a list of these pins and Figures Ba and Bb for a complete description of which pins have bus hold circuitry. These circuits will maintain the last valid logic state if no driving source is present (i.e., an unconnected pin or a driving source which goes to a high impedance state). To overdrive the "bus hold" circuits, an external driver must be capable of supplying the maximum "Bus Hold Overdrive" sink or source current at valid input voltage levels. Since this "bus hold" circuitry is active and not a

"resistive" type element, the associated power supply current is negligible and power dissipation is significantly reduced when compared to the use of passive pull-up resistors.

Bus Hold Circuitry on the 80C286

Signal	Pin Location	Polarity Pulled to when tri-stated
$\overline{S}1, \overline{S}0, \overline{PEACK}, \overline{LOCK}$	4-6, 68	Hi, See Figure Bb
Data Bus (D_0-D_{15})	36-51	Hi/Low, See Figure Ba
$\overline{COD}/\overline{INTA}, M/\overline{IO}$	66-67	Hi/Low, See Figure Ba

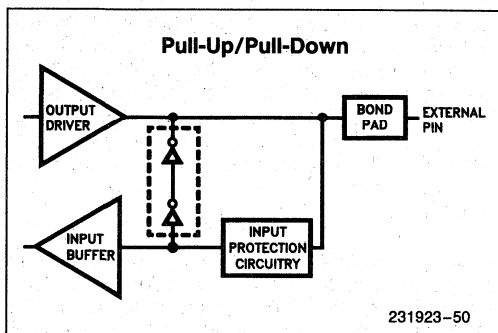


Figure Ba. Bus Hold Circuitry Pins 36-51, 66-67

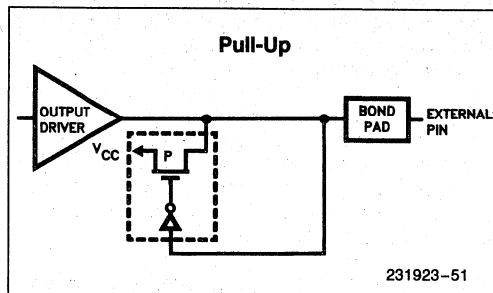


Figure Bb. Bus Hold Circuitry Pins 4-6, 68

SYSTEM CONFIGURATIONS

The versatile bus structure of the 80C286 microsystem, with a full complement of support chips, allows flexible configuration of a wide range of systems. The basic configuration, shown in Figure 32, is similar to an 8086 maximum mode system. It includes the CPU plus an 82C59A-2 interrupt controller, 82C284 clock generator, and the 82C288 Bus Controller.

As indicated by the dashed lines in Figure 32, the ability to add processor extensions is an integral feature of 80C286 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 80C286 supervises all data transfers and instruction execution for the processor extension.

The 80287 has all the instructions and data types of an 8087. The 80287 NPX can perform numeric calculations and data transfers concurrently with CPU program execution. Numerics code and data have the same integrity as all other information protected by the 80C286 protection mechanism.

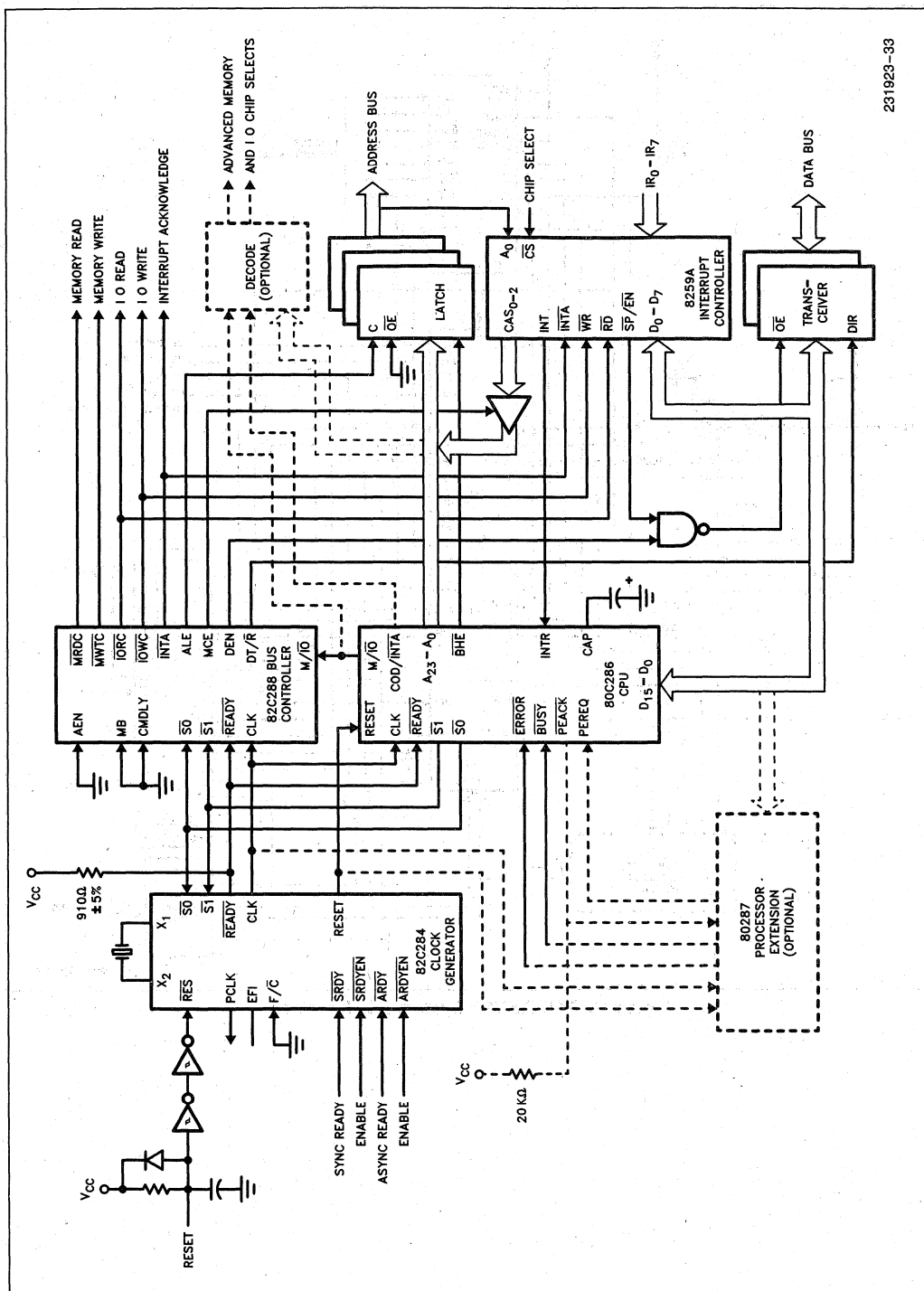
The 80C286 can overlap chip select decoding and address propagation during the data transfer for the previous bus operation. This information is latched by ALE during the middle of a T_s cycle. The latched chip select and address information remains stable during the bus operation while the next cycle's address

is being decoded and propagated into the system. Decode logic can be implemented with a high speed PROM or PAL.

The optional decode logic shown in Figure 32 takes advantage of the overlap between address and data of the 80C286 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/\overline{INTA} and M/\overline{IO} signals are applied to the decode logic to distinguish between interrupt, I/O, code and data bus cycles.

By adding a bus arbiter, the 80C286 provides a MULTIBUS system bus interface as shown in Figure 33. 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 setup times. This arrangement will add at least one extra T_c 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 33. This configuration allows the 80C286 to support an on-board bus for local memory and peripherals, and the MULTIBUS for system bus interfacing.



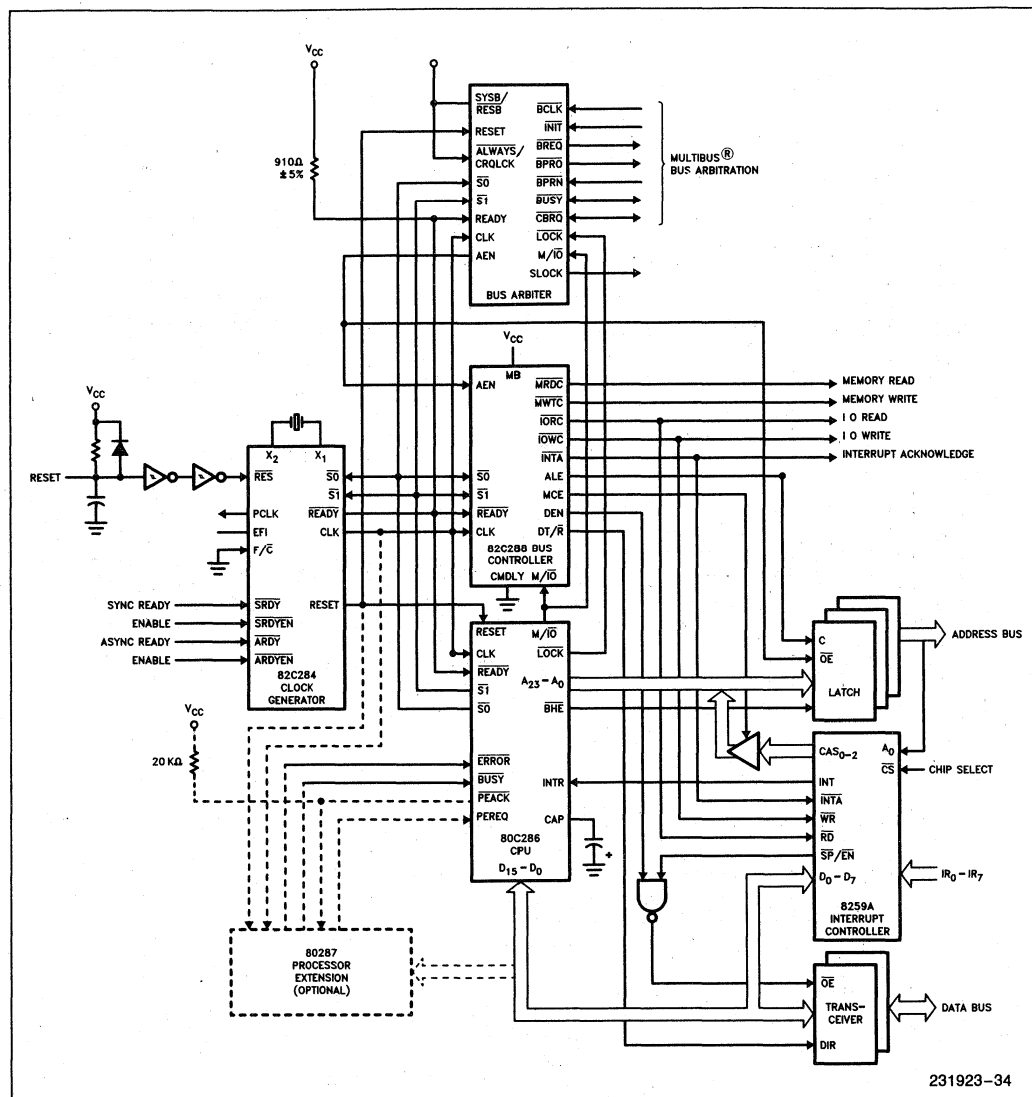


Figure 33. MULTIBUS® System Bus Interface

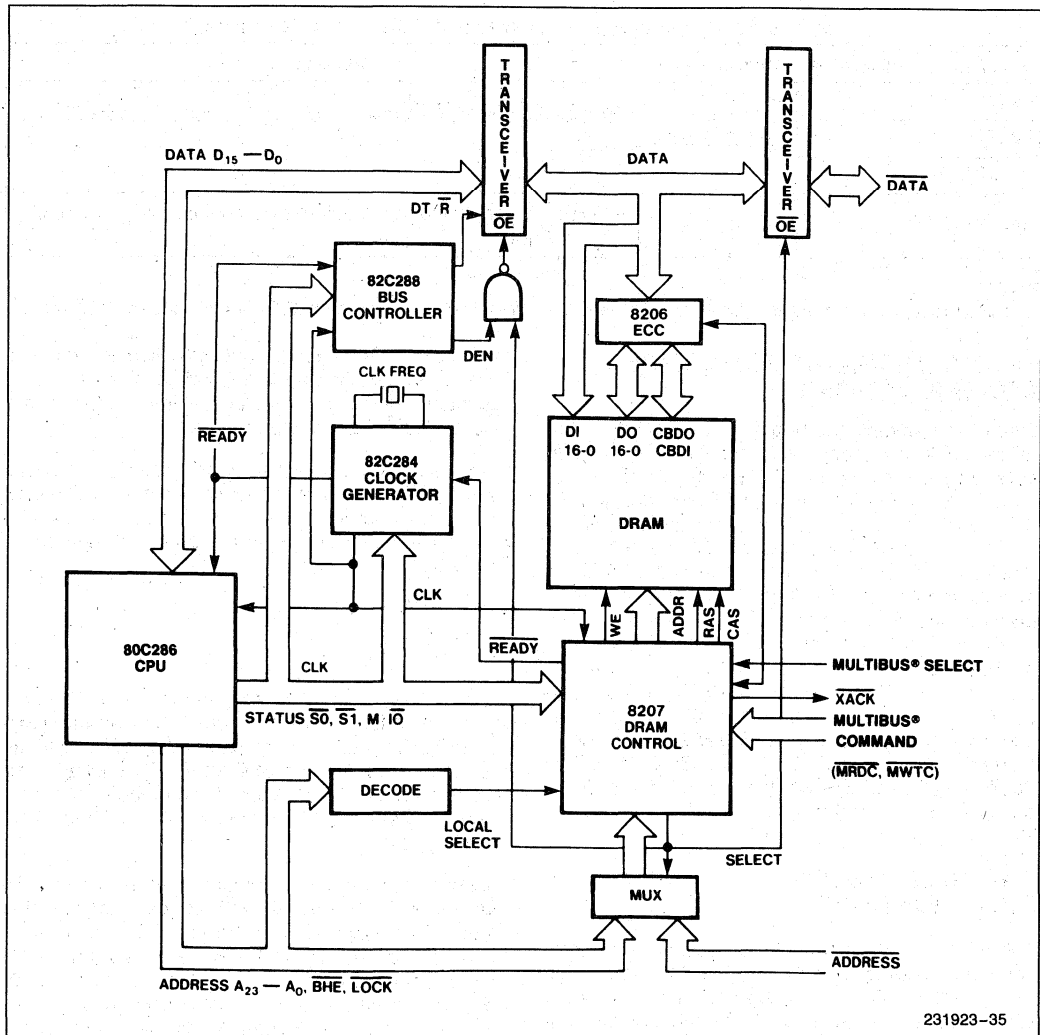


Figure 34. 80C286 System Configuration with Dual-Ported Memory

Figure 34 shows the addition of dual ported dynamic memory between the MULTIBUS system bus and the 80C286 local bus. The dual port interface is provided by the 8207 Dual Port DRAM Controller. The 8207 runs synchronously with the CPU to maximize throughput for local memory references. It also arbitrates between requests from the local and system buses and performs functions such as refresh.

initialization of RAM, and read/modify/write cycles. The 8207 combined with the 8206 Error Checking and Correction memory controller provide for single bit error correction. The dual-ported memory can be combined with a standard MULTIBUS system bus interface to maximize performance and protection in multiprocessor system configurations.

Table 16. 80C286 Systems Recommended Pull Up Resistor Values

80C286 Pin and Name	Pullup Value	Purpose
4— \overline{ST}	20 K Ω \pm 10%	Pull $\overline{S0}$, \overline{ST} , and \overline{PEACK} inactive during 80C286 hold periods (Note 1)
5— $\overline{S0}$		
6— \overline{PEACK}		
63—READY	910 Ω \pm 5%	Pull READY inactive within required minimum time (C_L = 150 pF, $I_R \leq 7$ mA)

NOTE:

1. Pullup resistors are not required for $\overline{S0}$ and \overline{ST} when the corresponding pins on the 82C284 are connected to $\overline{S0}$ and \overline{ST} .

80C286 IN-CIRCUIT EMULATION CONSIDERATIONS

One of the advantages of using the 80C286 is that full in-circuit emulation development support is available through either the I²ICE 80286 probe for 8 MHz/10 MHz or ICE286 for 12.5 MHz designs. To utilize these powerful tools it is necessary that the designer be aware of a few minor parametric and functional differences between the 80C286 and the in-circuit emulators. The I²ICE datasheet (I²ICE Integrated Instrumentation and In-Circuit Emulation System, order #210469) contains a detailed description of these design considerations. The ICE286 Fact Sheet (#280718) and User's Guide (#452317) contain design considerations for the 80C286 12.5 MHz microprocessor. It is recommended that the appropriate document be reviewed by the 80C286 system designer to determine whether or not these differences affect the design.

PACKAGE THERMAL SPECIFICATIONS

The 80C286 Microprocessor is specified for operation when case temperature (T_C) is within the range of 0°C–85°C. Case temperature, unlike ambient temperature, is easily measured in any environment

to determine whether the 80C286 Microprocessor is within the specified operating range. The case temperature should be measured at the center of the top surface of the component.

The maximum ambient temperature (T_A) allowable without violating T_C specifications can be calculated from the equations shown below. T_J is the 80C286 junction temperature. P is the power dissipated by the 80C286.

$$\begin{aligned} T_J &= T_C + P \cdot \theta_{JC} \\ T_A &= T_J - P \cdot \theta_{JA} \\ T_C &= T_A + P \cdot [\theta_{JA} - \theta_{JC}] \end{aligned}$$

Values for θ_{JA} and θ_{JC} are given in Table 17. θ_{JA} is given at various airflows. Table 18 shows the maximum T_A allowable (without exceeding T_C) at various airflows. Note that the 80C286 PLCC package has an internal heat spreader. T_A can be further improved by attaching "fins" or an external "heat sink" to the package.

Junction temperature calculations should use an I_{CC} value that is measured without external resistive loads. The external resistive loads dissipate additional power external to the 80C286 and not on the die. This increases the resistor temperature, not the die temperature. The full capacitive load (C_L = 100 pF) should be applied during the I_{CC} measurement.

Table 17. Thermal Resistances
(°C/Watt) θ_{JC} and θ_{JA}

Package	θ_{JC}	θ_{JA} versus Airflow ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
68-Lead PGA	5.5	29	22	16	15	14	13
68-Lead PLCC w/Internal Heat Spreader	8	29	23	21	18	16	15

NOTE:

The numbers in Table 18 were calculated using a V_{CC} of 5.0V, and an I_{CC} of 150 mA, which is representative of the worst case I_{CC} at T_C = 85°C with the outputs unloaded.

Table 18. Maximum T_A at Various Airflows

Package	T_A (°C) versus Airflow ft/min (m/sec)					
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
68-Lead PGA	68	73	78	78	79	80
68 Lead-PLCC w/Internal Heat Spreader	70	74	76	78	79	80

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature under Bias 0°C to +70°C
 Storage Temperature -65°C to +150°C
 Voltage on Any Pin with
 Respect to Ground -1.0V to +7V
 Power Dissipation 1.1W

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

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

D.C. CHARACTERISTICS ($V_{CC} = 5V \pm 10\%$, $T_{CASE} = 0^\circ C$ to $+85^\circ C$)

Symbol	Parameter	Min	Max	Typ	Unit	Test Conditions
I_{CC}	Supply Current		200	125	mA	$C_L = 100$ pF (Note 1)
I_{CCS}	Supply Current (Static)		5	0.5	mA	(Note 2)
C_{CLK}	CLK Input Capacitance		20		pF	FREQ = 1 MHz (Note 3)
C_{IN}	Other Input Capacitance		10		pF	FREQ = 1 MHz (Note 3)
C_O	Input/Output Capacitance		20		pF	FREQ = 1 MHz (Note 3)

NOTES:

1. Tested at maximum frequency with no resistive loads on the outputs.
2. Tested while clock stopped in phase 2 and inputs at V_{CC} or V_{SS} with the outputs unloaded.
3. These are not tested but are guaranteed by design characterization.

D.C. CHARACTERISTICS ($V_{CC} = 5V \pm 10\%$, $T_{CASE} = 0^\circ C$ to $+85^\circ C$)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input LOW Voltage	-0.5	0.8	V	FREQ = 2 MHz
V_{IH}	Input HIGH Voltage	2.0	$V_{CC} + 0.5$	V	FREQ = 2 MHz
V_{ILC}	CLK Input LOW Voltage	-0.5	0.8	V	FREQ = 2 MHz
V_{IHC}	CLK Input HIGH Voltage	3.8	$V_{CC} + 0.5$	V	FREQ = 2 MHz
V_{OL}	Output LOW Voltage		0.45	V	$I_{OL} = 2.0$ mA, FREQ = 2 MHz
V_{OH}	Output HIGH Voltage	3.0 $V_{CC} - 0.5$		V V	$I_{OH} = -2.0$ mA, FREQ = 2 MHz $I_{OH} = -100$ μ A, FREQ = 2 MHz
I_{LI}	Input Leakage Current		± 10	μ A	$V_{IN} = GND$ or V_{CC} (Note 1)
I_{LO}	Output Leakage Current		± 10	μ A	$V_O = GND$ or V_{CC} (Note 1)
I_{IL}	Input Sustaining Current on BUSY# and ERROR# Pins	-30	-500	μ A	$V_{IN} = 0V$ (Note 1)
I_{BHL}	Input Sustaining Current (Bus Hold LOW)	38	150	μ A	$V_{IN} = 1.0V$ (Notes 1, 2)
I_{BHH}	Input Sustaining Current (Bus Hold HIGH)	-50	-350	μ A	$V_{IN} = 3.0V$ (Notes 1, 3)
I_{BHLO}	Bus Hold LOW Overdrive	200		μ A	(Notes 1, 4)
I_{BHHO}	Bus Hold HIGH Overdrive	-400		μ A	(Notes 1, 5)

NOTES:

1. Tested with the clock stopped.
2. I_{BHL} should be measured after lowering V_{IN} to GND and then raising to 1.0V on the following pins: 36-51, 66, 67.
3. I_{BHH} should be measured after raising V_{IN} to V_{CC} and then lowering to 3.0V on the following pins: 4-6, 36-51, 66-68.
4. An external driver must source at least I_{BHLO} to switch this node from LOW to HIGH.
5. An external driver must sink at least I_{BHHO} to switch this node from HIGH to LOW.

A.C. CHARACTERISTICS (V_{CC} = 5V ± 10%, T_{CASE} = 0°C to +85°C)

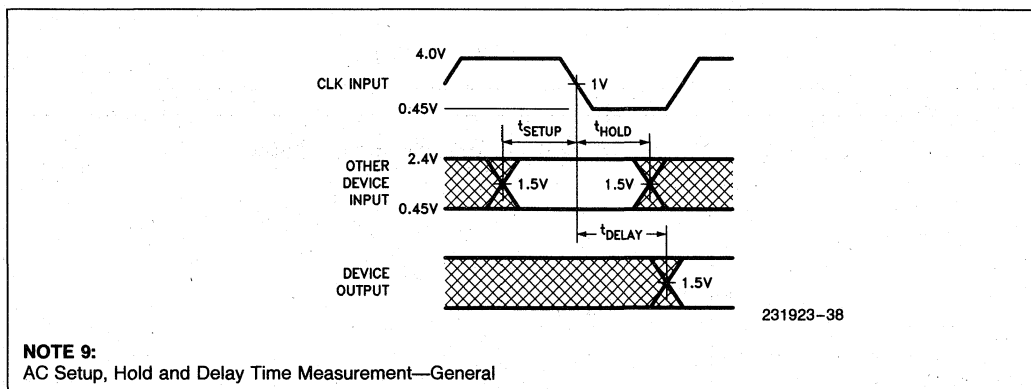
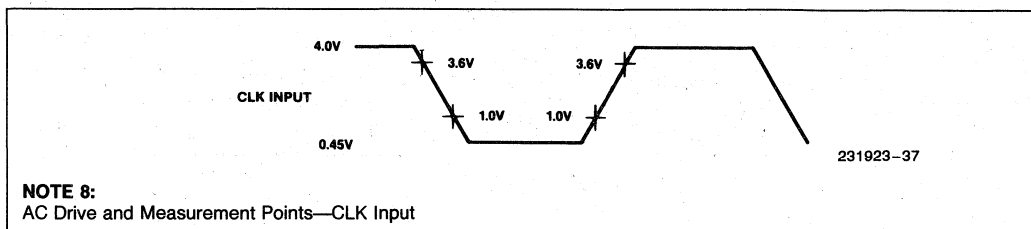
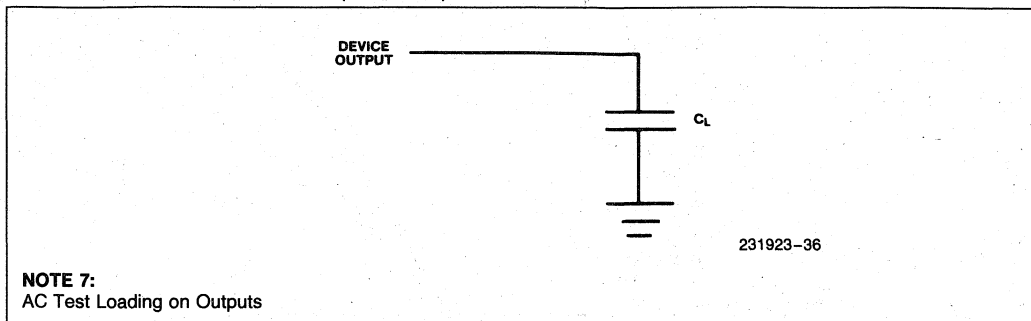
A.C. timings are referenced to 1.5V points of signals as illustrated in datasheet waveforms, unless otherwise noted.

Symbol	Parameter	12.5 MHz		Unit	Test Conditions
		Min	Max		
1	System Clock (CLK) Period	40	DC	ns	(Note 1)
2	System Clock (CLK) LOW Time	11		ns	at 1.0V
3	System Clock (CLK) HIGH Time	13		ns	at 3.6V
17	System Clock (CLK) Rise Time		8	ns	1.0V to 3.6V (Note 2)
18	System Clock (CLK) Fall Time		8	ns	3.6V to 1.0V (Note 2)
4	Asynchronous Inputs Setup Time	16		ns	(Note 3)
5	Asynchronous Inputs Hold Time	16		ns	(Note 3)
6	RESET Setup Time	19		ns	
7	RESET Hold Time	6		ns	
8	Read Data Setup Time	6		ns	
9	Read Data Hold Time	7		ns	
10	READY Setup Time	23		ns	
11	READY Hold Time	21		ns	
12a1	Status Active Delay	5	16	ns	(Notes 4, 5, 7)
12a2	PEACK Active Delay	5	18	ns	(Notes 4, 5, 7)
12b	Status/PEACK Inactive Delay	5	20	ns	(Notes 4, 5, 7)
13	Address Valid Delay	4	29	ns	(Notes 4, 5, 7)
14	Write Data Valid Delay	3	27	ns	(Notes 4, 5, 7)
15	Address/Status/Data Float Delay	2	32	ns	(Notes 2, 4, 6)
16	HLDA Valid Delay	3	24	ns	(Notes 4, 5, 7)
19	Address Valid To Status Valid Setup Time	23		ns	(Notes 2, 4, 5)

NOTES:

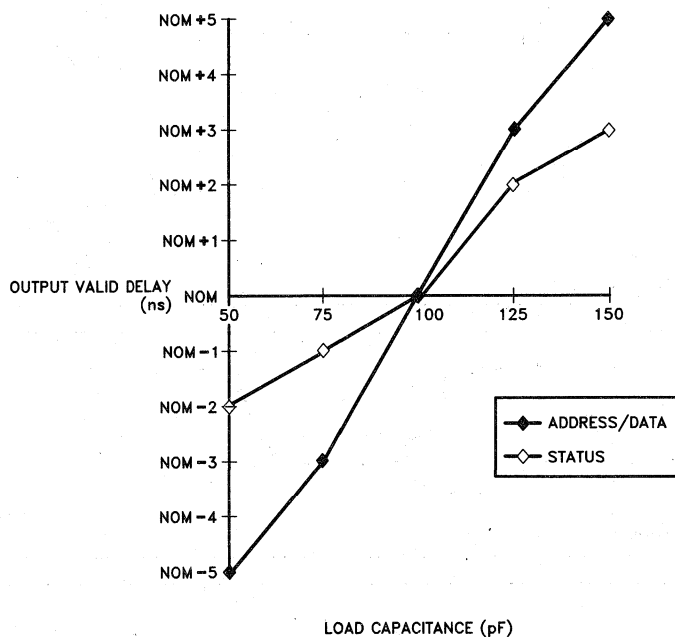
- Functionality at frequencies less than 2 MHz is not tested, but is guaranteed by design characterization.
- These are not tested but are guaranteed by design characterization.
- Asynchronous inputs are INTR, NMI, HOLD, PEREQ, ERROR, and BUSY. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.
- Delay from 1.0V on the CLK, to 1.5V or float on the output as appropriate for valid or floating condition.
- Output load: C_L = 100 pF.
- Float condition occurs when output current is less than I_{LO} in magnitude.
- Minimum output delay timings are not tested, but are guaranteed by design characterization.

A.C. CHARACTERISTICS (Continued)



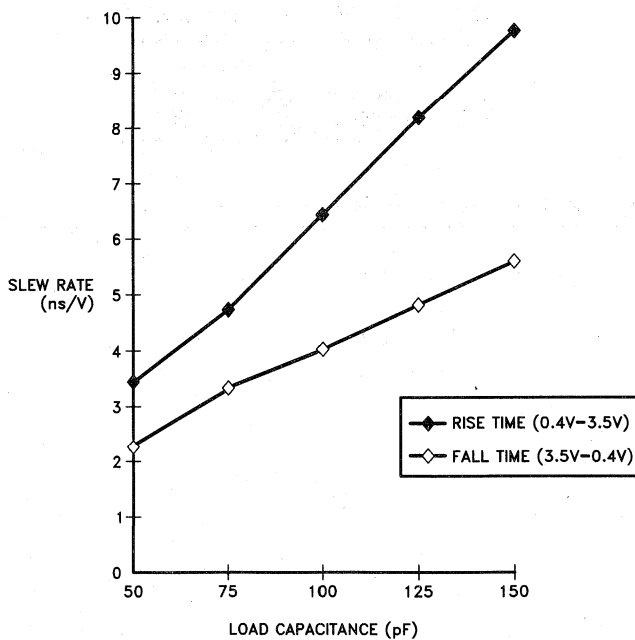
3

Typical Capacitive Derating Curves



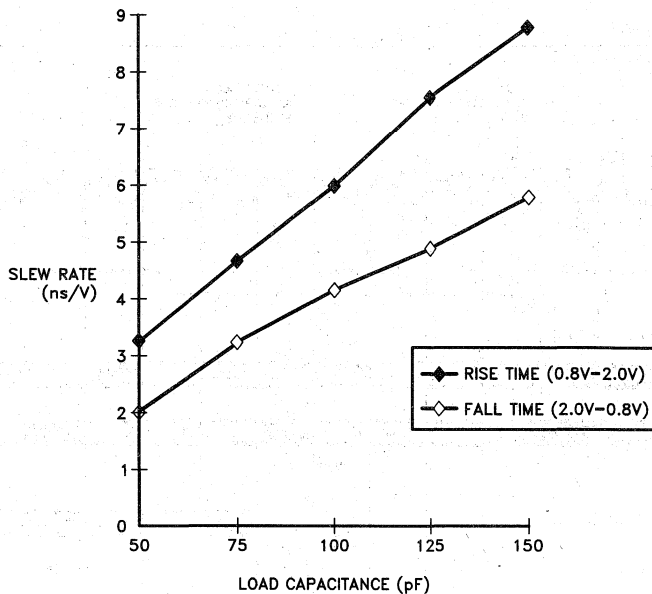
231923-46

Typical CMOS Level Slew Rates for Address/Data Buffers



231923-47

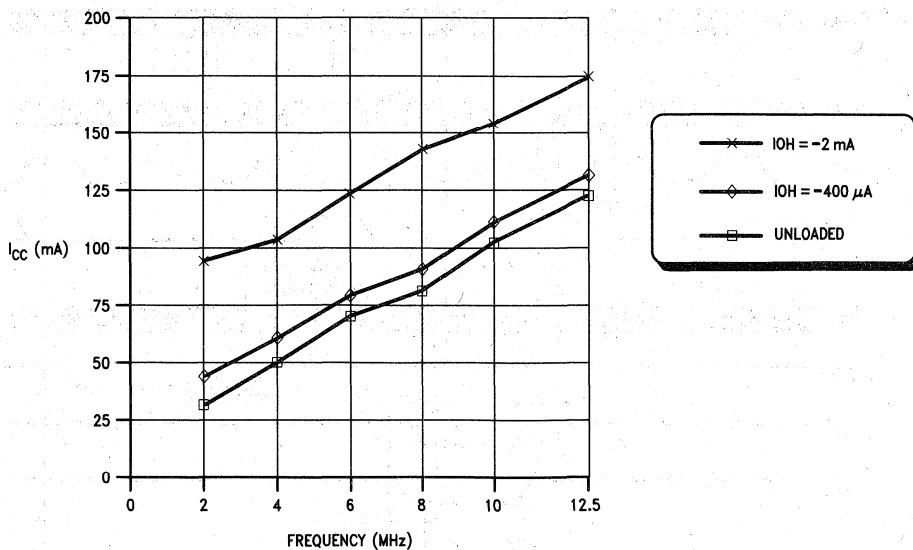
Typical TTL Level Slew Rates for Address/Data Buffers



231923-48

3

Typical I_{CC} vs Frequency for Different Output Loads



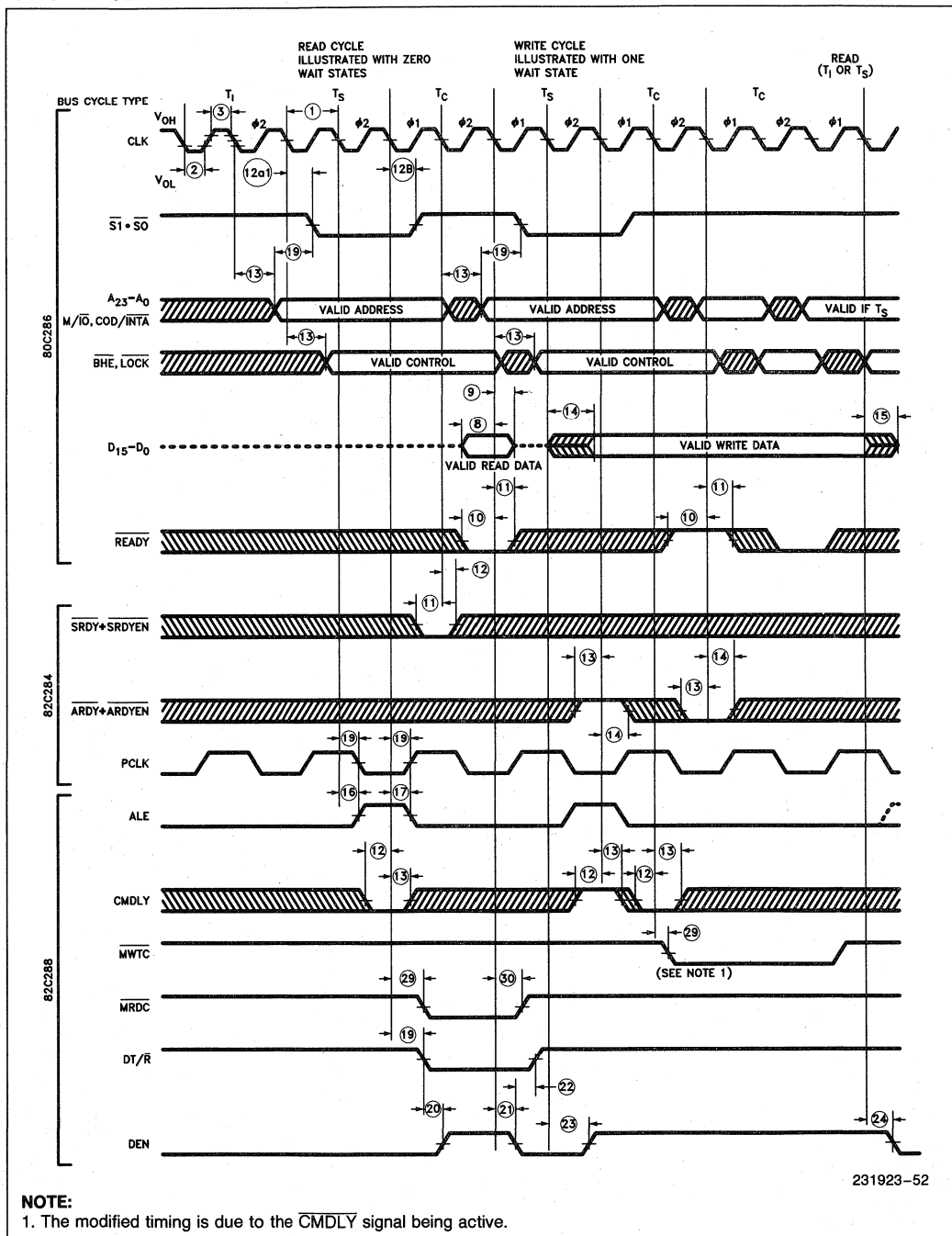
231923-53

NOTES:

1. $V_{CC} = 5.0\text{ V}$
2. Loaded: $I_{OL} = 2.0\text{ mA}$, I_{OH} as shown, $C_L = 100\text{ pF}$
Unloaded: $C_L = 100\text{ pF}$

WAVEFORMS

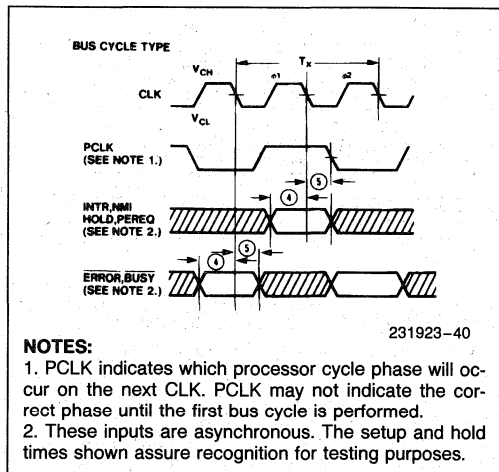
MAJOR CYCLE TIMING



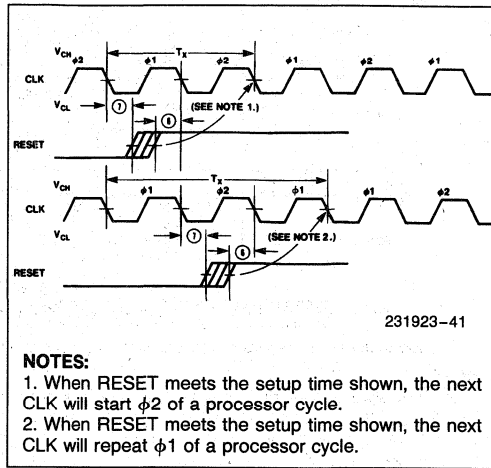
231923-52

WAVEFORMS (Continued)

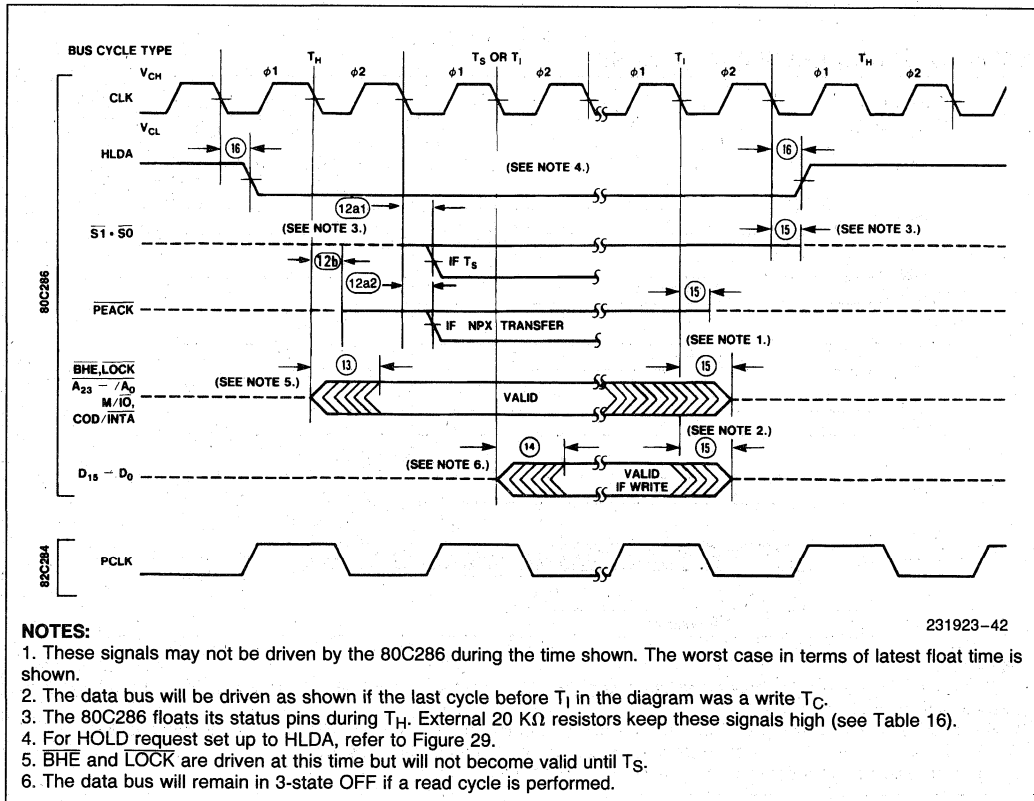
80C286 ASYNCHRONOUS INPUT SIGNAL TIMING



80C286 RESET INPUT TIMING AND SUBSEQUENT PROCESSOR CYCLE PHASE

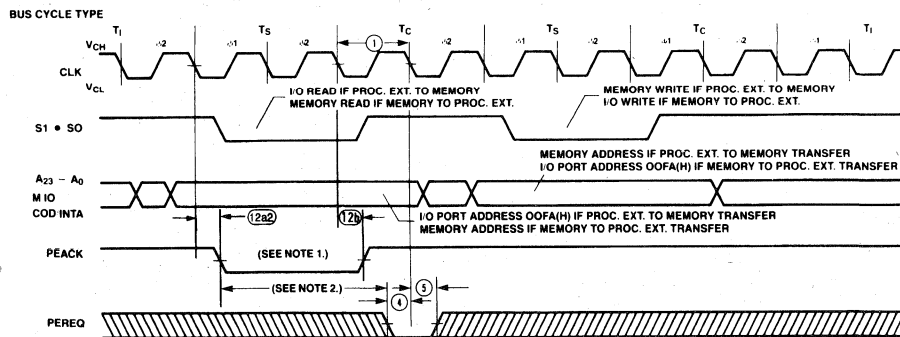


EXITING AND ENTERING HOLD



WAVEFORMS (Continued)

80C286 PEREQ/PEACK TIMING FOR ONE TRANSFER ONLY



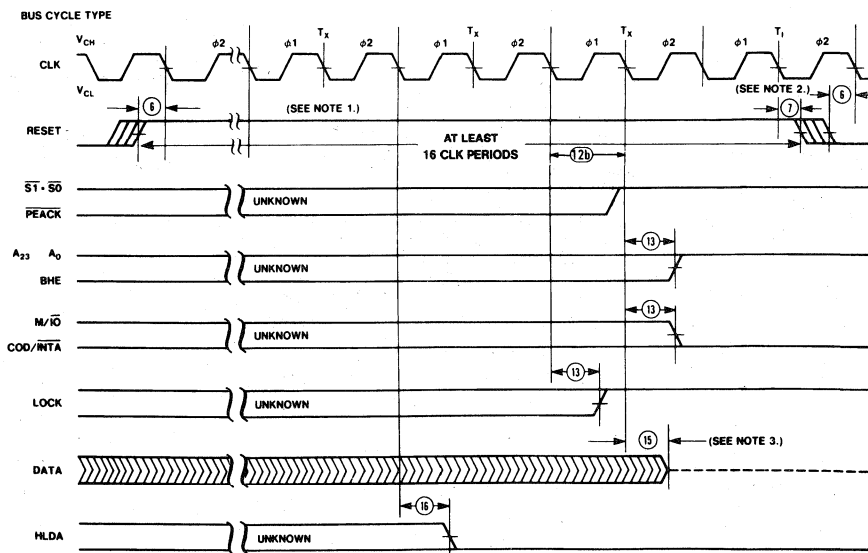
ASSUMING WORD-ALIGNED MEMORY OPERAND. IF ODD ALIGNED, 80286 TRANSFERS TO/FROM MEMORY BYTE-AT-A-TIME WITH TWO MEMORY CYCLES.

NOTES:

1. PEACK always goes active during the first bus operation of a processor extension data operand transfer sequence. The first bus operation will be either a memory read at operand address or I/O read at port address OOFa(H).
 2. To prevent a second processor extension data operand transfer, the worst case maximum time (Shown above) is: $3 \times \phi - 12a2_{max} - \phi_{min}$. The actual, configuration dependent, maximum time is: $3 \times \phi - 12a2_{max} - \phi_{min} + A \times 2 \times \phi$.
- 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.

231923-43

INITIAL 80C286 PIN STATE DURING RESET



NOTES:

1. Setup time for RESET ↑ may be violated with the consideration that φ₁ of the processor clock may begin one system CLK period later.
2. Setup and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during φ₁ or φ₂. If RESET ↓ occurs in φ₁, the reference clock edge can be φ₂ of the previous bus cycle.
3. The data bus is only guaranteed to be in 3-state OFF at the time shown.

231923-44

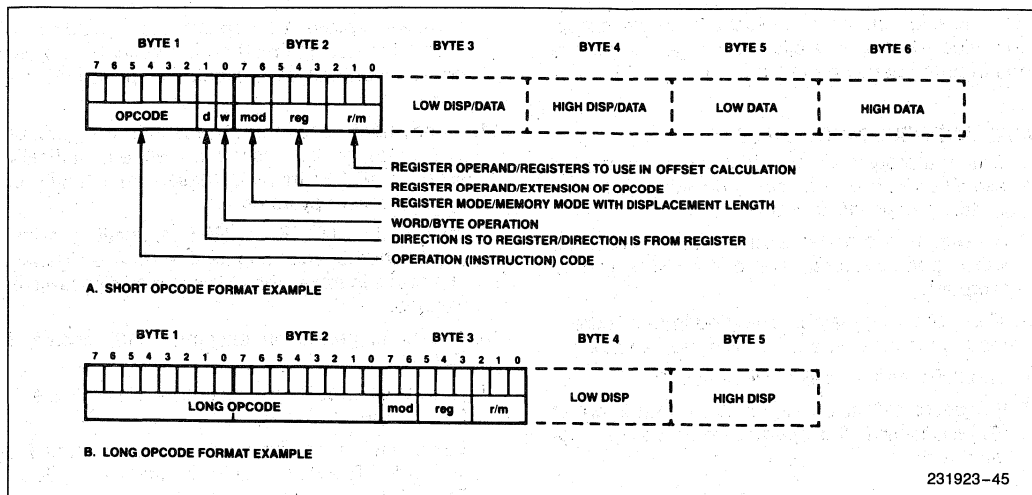


Figure 35. 80C286 Instruction Format Examples

80C286 INSTRUCTION SET SUMMARY

Instruction Timing Notes

The instruction clock counts listed below establish the maximum execution rate of the 80C286. With no delays in bus cycles, the actual clock count of an 80C286 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. A 12 MHz processor clock has a clock period of 83 nanoseconds and requires an 80C286 system clock (CLK input) of 24 MHz.

Instruction Clock Count Assumptions

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

Instruction Set Summary Notes

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

Above/below refers to unsigned value

Greater refers to positive signed value

Less refers to less positive (more negative) signed values

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

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

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

if s = 1 then an immediate data byte is sign-extended to form the 16-bit operand

x don't care

z used for string primitives for comparison with ZF FLAG

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

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

n = number of times repeated

m = number of bytes of code in next instruction

Level (L)—Lexical nesting level of the procedure

3

The following comments describe possible exceptions, side effects, and allowed usage for instructions in both operating modes of the 80C286.

REAL ADDRESS MODE ONLY

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

EITHER MODE

6. An exception may occur, depending on the value of the operand.
7. `LOCK` is automatically asserted regardless of the presence or absence of the `LOCK` instruction prefix.
8. `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 `LOCK` to maintain descriptor integrity in multiprocessor systems.
12. `JMP`, `CALL`, `INT`, `RET`, `IRET` instructions referring to another code segment will cause a general protection exception (13) if any privilege rule is violated.
13. A general protection exception (13) occurs if $CPL \neq 0$.
14. A general protection exception (13) occurs if $CPL > IOPL$.
15. The IF field of the flag word is not updated if $CPL > IOPL$. The IOPL field is updated only if $CPL = 0$.
16. Any violation of privilege rules as applied to the selector operand do not cause a protection exception; rather, the instruction does not return a result and the zero flag is cleared.
17. If the starting address of the memory operand violates a segment limit, or an invalid access is attempted, a general protection exception (13) will occur before the `ESC` instruction is executed. A stack segment overrun exception (12) will occur if the stack limit is violated by the operand's starting address. If a segment limit is violated during an attempted data transfer then a processor extension segment overrun exception (9) occurs.
18. The destination of an `INT`, `JMP`, `CALL`, `RET` or `IRET` instruction must be in the defined limit of a code segment or a general protection exception (13) will occur.

80C286 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	<table><tr><td>1 0 0 0 1 0 0 w</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 1 0 0 w	mod reg	r/m	2,3*	2,3*	2	9	
1 0 0 0 1 0 0 w	mod reg	r/m							
Register/memory to register	<table><tr><td>1 0 0 0 1 0 1 w</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 1 0 1 w	mod reg	r/m	2,5*	2,5*	2	9	
1 0 0 0 1 0 1 w	mod reg	r/m							
Immediate to register/memory	<table><tr><td>1 1 0 0 0 1 1 w</td><td>mod 0 0 0 r/m</td><td>data</td><td>data if w = 1</td></tr></table>	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1	2,3*	2,3*	2	9
1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1						
Immediate to register	<table><tr><td>1 0 1 1 w</td><td>reg</td><td>data</td><td>data if w = 1</td></tr></table>	1 0 1 1 w	reg	data	data if w = 1	2	2		
1 0 1 1 w	reg	data	data if w = 1						
Memory to accumulator	<table><tr><td>1 0 1 0 0 0 0 w</td><td>addr-low</td><td>addr-high</td></tr></table>	1 0 1 0 0 0 0 w	addr-low	addr-high	5	5	2	9	
1 0 1 0 0 0 0 w	addr-low	addr-high							
Accumulator to memory	<table><tr><td>1 0 1 0 0 0 1 w</td><td>addr-low</td><td>addr-high</td></tr></table>	1 0 1 0 0 0 1 w	addr-low	addr-high	3	3	2	9	
1 0 1 0 0 0 1 w	addr-low	addr-high							
Register/memory to segment register	<table><tr><td>1 0 0 0 1 1 1 0</td><td>mod 0 reg</td><td>r/m</td></tr></table>	1 0 0 0 1 1 1 0	mod 0 reg	r/m	2,5*	17,19*	2	9,10,11	
1 0 0 0 1 1 1 0	mod 0 reg	r/m							
Segment register to register/memory	<table><tr><td>1 0 0 0 1 1 0 0</td><td>mod 0 reg</td><td>r/m</td></tr></table>	1 0 0 0 1 1 0 0	mod 0 reg	r/m	2,3*	2,3*	2	9	
1 0 0 0 1 1 0 0	mod 0 reg	r/m							
PUSH = Push:									
Memory	<table><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 1 0 r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 1 1 0 r/m	5*	5*	2	9		
1 1 1 1 1 1 1 1	mod 1 1 0 r/m								
Register	<table><tr><td>0 1 0 1 0</td><td>reg</td></tr></table>	0 1 0 1 0	reg	3	3	2	9		
0 1 0 1 0	reg								
Segment register	<table><tr><td>0 0 0 reg</td><td>1 1 0</td></tr></table>	0 0 0 reg	1 1 0	3	3	2	9		
0 0 0 reg	1 1 0								
Immediate	<table><tr><td>0 1 1 0 1 0 s 0</td><td>data</td><td>data if s = 0</td></tr></table>	0 1 1 0 1 0 s 0	data	data if s = 0	3	3	2	9	
0 1 1 0 1 0 s 0	data	data if s = 0							
PUSHA = Push All	<table><tr><td>0 1 1 0 0 0 0 0</td></tr></table>	0 1 1 0 0 0 0 0	17	17	2	9			
0 1 1 0 0 0 0 0									
POP = Pop:									
Memory	<table><tr><td>1 0 0 0 1 1 1 1</td><td>mod 0 0 0 r/m</td></tr></table>	1 0 0 0 1 1 1 1	mod 0 0 0 r/m	5*	5*	2	9		
1 0 0 0 1 1 1 1	mod 0 0 0 r/m								
Register	<table><tr><td>0 1 0 1 1</td><td>reg</td></tr></table>	0 1 0 1 1	reg	5	5	2	9		
0 1 0 1 1	reg								
Segment register	<table><tr><td>0 0 0 reg</td><td>1 1 1 (reg ≠ 01)</td></tr></table>	0 0 0 reg	1 1 1 (reg ≠ 01)	5	20	2	9,10,11		
0 0 0 reg	1 1 1 (reg ≠ 01)								
POPA = Pop All	<table><tr><td>0 1 1 0 0 0 0 1</td></tr></table>	0 1 1 0 0 0 0 1	19	19	2	9			
0 1 1 0 0 0 0 1									
XCHG = Exchange:									
Register/memory with register	<table><tr><td>1 0 0 0 0 1 1 w</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 0 1 1 w	mod reg	r/m	3,5*	3,5*	2,7	7,9	
1 0 0 0 0 1 1 w	mod reg	r/m							
Register with accumulator	<table><tr><td>1 0 0 1 0</td><td>reg</td></tr></table>	1 0 0 1 0	reg	3	3				
1 0 0 1 0	reg								
IN = Input from:									
Fixed port	<table><tr><td>1 1 1 0 0 1 0 w</td><td>port</td></tr></table>	1 1 1 0 0 1 0 w	port	5	5		14		
1 1 1 0 0 1 0 w	port								
Variable port	<table><tr><td>1 1 1 0 1 1 0 w</td></tr></table>	1 1 1 0 1 1 0 w	5	5		14			
1 1 1 0 1 1 0 w									
OUT = Output to:									
Fixed port	<table><tr><td>1 1 1 0 0 1 1 w</td><td>port</td></tr></table>	1 1 1 0 0 1 1 w	port	3	3		14		
1 1 1 0 0 1 1 w	port								
Variable port	<table><tr><td>1 1 1 0 1 1 1 w</td></tr></table>	1 1 1 0 1 1 1 w	3	3		14			
1 1 1 0 1 1 1 w									
XLAT = Translate byte to AL	<table><tr><td>1 1 0 1 0 1 1 1</td></tr></table>	1 1 0 1 0 1 1 1	5	5		9			
1 1 0 1 0 1 1 1									
LEA = Load EA to register	<table><tr><td>1 0 0 0 1 1 0 1</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 1 1 0 1	mod reg	r/m	3*	3*			
1 0 0 0 1 1 0 1	mod reg	r/m							
LDS = Load pointer to DS	<table><tr><td>1 1 0 0 0 1 0 1</td><td>mod reg</td><td>r/m (mod ≠ 11)</td></tr></table>	1 1 0 0 0 1 0 1	mod reg	r/m (mod ≠ 11)	7*	21*	2	9,10,11	
1 1 0 0 0 1 0 1	mod reg	r/m (mod ≠ 11)							
LES = Load pointer to ES	<table><tr><td>1 1 0 0 0 1 0 0</td><td>mod reg</td><td>r/m (mod ≠ 1)</td></tr></table>	1 1 0 0 0 1 0 0	mod reg	r/m (mod ≠ 1)	7*	21*	2	9,10,11	
1 1 0 0 0 1 0 0	mod reg	r/m (mod ≠ 1)							

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80C286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
DATA TRANSFER (Continued)					
LAHF Load AH with flags	10011111	2	2		
SAHF = Store AH into flags	10011110	2	2		
PUSHF = Push flags	10011100	3	3	2	9
POPF = Pop flags	10011101	5	5	2,4	9,15
ARITHMETIC					
ADD = Add:					
Reg/memory with register to either	000000dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	100000sw mod 000 r/m data data if sw = 01	3,7*	3,7*	2	9
Immediate to accumulator	0000010w data data if w = 1	3	3		
ADC = Add with carry:					
Reg/memory with register to either	000100dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	100000sw mod 010 r/m data data if sw = 01	3,7*	3,7*	2	9
Immediate to accumulator	0001010w data data if w = 1	3	3		
INC = Increment:					
Register/memory	1111111w mod 000 r/m	2,7*	2,7*	2	9
Register	01000reg	2	2		
SUB = Subtract:					
Reg/memory and register to either	001010dw mod reg r/m	2,7*	2,7*	2	9
Immediate from register/memory	100000sw mod 101 r/m data data if sw = 01	3,7*	3,7*	2	9
Immediate from accumulator	0010110w data data if w = 1	3	3		
SBB = Subtract with borrow:					
Reg/memory and register to either	000110dw mod reg r/m	2,7*	2,7*	2	9
Immediate from register/memory	100000sw mod 011 r/m data data if sw = 01	3,7*	3,7*	2	9
Immediate from accumulator	0001110w data data if w = 1	3	3		
DEC = Decrement					
Register/memory	1111111w mod 001 r/m	2,7*	2,7*	2	9
Register	01001reg	2	2		
CMP = Compare					
Register/memory with register	0011101w mod reg r/m	2,6*	2,6*	2	9
Register with register/memory	0011100w mod reg r/m	2,7*	2,7*	2	9
Immediate with register/memory	100000sw mod 111 r/m data data if sw = 01	3,6*	3,6*	2	9
Immediate with accumulator	0011110w data data if w = 1	3	3		
NEG = Change sign	1111011w mod 011 r/m	2	7*	2	9
AAA = ASCII adjust for add	00110111	3	3		
DAA = Decimal adjust for add	00100111	3	3		

80C286 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)					
AAS = ASCII adjust for subtract	<div>00111111</div>	3	3		
DAS = Decimal adjust for subtract	<div>00101111</div>	3	3		
MUL = Multiply (unsigned):	<div>1111011w</div> <div>mod 100</div> <div>r/m</div>				
Register-Byte		13	13		
Register-Word		21	21		
Memory-Byte		16*	16*	2	9
Memory-Word		24*	24*	2	9
IMUL = Integer multiply (signed):	<div>1111011w</div> <div>mod 101</div> <div>r/m</div>				
Register-Byte		13	13		
Register-Word		21	21		
Memory-Byte		16*	16*	2	9
Memory-Word		24*	24*	2	9
IMUL = Integer immediate multiply (signed)	<div>011010s1</div> <div>mod reg</div> <div>r/m</div> <div>data</div> <div>data if s = 0</div>	21,24*	21,24*	2	9
DIV = Divide (unsigned)					
	<div>1111011w</div> <div>mod 110</div> <div>r/m</div>				
Register-Byte		14	14	6	6
Register-Word		22	22	6	6
Memory-Byte		17*	17*	2,6	6,9
Memory-Word		25*	25*	2,6	6,9
IDIV = Integer divide (signed)	<div>1111011w</div> <div>mod 111</div> <div>r/m</div>				
Register-Byte		17	17	6	6
Register-Word		25	25	6	6
Memory-Byte		20*	20*	2,6	6,9
Memory-Word		28*	28*	2,6	6,9
AAM = ASCII adjust for multiply	<div>11010100</div> <div>00001010</div>	16	16		
AAD = ASCII adjust for divide	<div>11010101</div> <div>00001010</div>	14	14		
CBW = Convert byte to word	<div>10011000</div>	2	2		
CWD = Convert word to double word	<div>10011001</div>	2	2		
LOGIC					
Shift/Rotate Instructions:					
Register/Memory by 1	<div>1101000w</div> <div>mod TTT</div> <div>r/m</div>	2,7*	2,7*	2	9
Register/Memory by CL	<div>1101001w</div> <div>mod TTT</div> <div>r/m</div>	5+n,8+n*	5+n,8+n*	2	9
Register/Memory by Count	<div>1100000w</div> <div>mod TTT</div> <div>r/m</div> <div>count</div>	5+n,8+n*	5+n,8+n*	2	9
TTT Instruction					
000 ROL					
001 ROR					
010 RCL					
011 RCR					
100 SHL/SAL					
101 SHR					
111 SAR					

Shaded areas indicate instructions not available in 8086, 68 microsystems.

80C286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC (Continued)					
AND = And:					
Reg/memory and register to either	001000dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w mod 100 r/m data data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0010010w data data if w = 1	3	3		
TEST = And function to flags, no result:					
Register/memory and register	1000010w mod reg r/m	2,6*	2,6*	2	9
Immediate data and register/memory	1111011w mod 000 r/m data data if w = 1	3,6*	3,6*	2	9
Immediate data and accumulator	1010100w data data if w = 1	3	3		
OR = Or:					
Reg/memory and register to either	000010dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w mod 001 r/m data data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0000110w data data if w = 1	3	3		
XOR = Exclusive or:					
Reg/memory and register to either	001100dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w mod 110 r/m data data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0011010w data data if w = 1	3	3		
NOT = Invert register/memory	1111011w mod 010 r/m	2,7*	2,7*	2	9
STRING MANIPULATION:					
MOVS = Move byte/word	1010010w	5	5	2	9
CMPS = Compare byte/word	1010011w	8	8	2	9
SCAS = Scan byte/word	1010111w	7	7	2	9
LDS = Load byte/wd to AL/AX	1010110w	5	5	2	9
STOS = Stor byte/wd from AL/A	1010101w	3	3	2	9
INS = Input byte/wd from DX port	0110110w	5	5	2	9,14
OUTS = Output byte/wd to DX port	0110111w	5	5	2	9,14
Repeated by count in CX					
MOV _s = Move string	11110011 1010010w	5+4n	5+4n	2	9
CMPS = Compare string	1111001z 1010011w	5+9n	5+9n	2,8	8,9
SCAS = Scan string	1111001z 1010111w	5+8n	5+8n	2,8	8,9
LDS = Load string	11110011 1010110w	5+4n	5+4n	2,8	8,9
STOS = Store string	11110011 1010101w	4+3n	4+3n	2,8	8,9
INS = Input string	11110011 0110110w	5+4n	5+4n	2	9,14
OUTS = Output string	11110011 0110111w	5+4n	5+4n	2	9,14

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80C286 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								
CALL = Call:								
Direct within segment	<table><tr><td>11101000</td><td>disp-low</td><td>disp-high</td></tr></table>	11101000	disp-low	disp-high	7 + m	7 + m	2	18
11101000	disp-low	disp-high						
Register/memory indirect within segment	<table><tr><td>11111111</td><td>mod 010</td><td>r/m</td></tr></table>	11111111	mod 010	r/m	7 + m, 11 + m*	7 + m, 11 + m*	2,8	8,9,18
11111111	mod 010	r/m						
Direct intersegment	<table><tr><td>10011010</td><td colspan="2">segment offset</td></tr></table>	10011010	segment offset		13 + m	26 + m	2	11,12,18
10011010	segment offset							
Protected Mode Only (Direct intersegment):								
Via call gate to same privilege level			41 + m		8,11,12,18			
Via call gate to different privilege level, no parameters			82 + m		8,11,12,18			
Via call gate to different privilege level, x parameters			86 + 4x + m		8,11,12,18			
Via TSS			177 + m		8,11,12,18			
Via task gate			182 + m		8,11,12,18			
Indirect intersegment	<table><tr><td>11111111</td><td>mod 011</td><td>r/m</td></tr></table> (mod ≠ 11)	11111111	mod 011	r/m	16 + m	29 + m*	2	8,9,11,12,18
11111111	mod 011	r/m						
Protected Mode Only (Indirect intersegment):								
Via call gate to same privilege level			44 + m*		8,9,11,12,18			
Via call gate to different privilege level, no parameters			83 + m*		8,9,11,12,18			
Via call gate to different privilege level, x parameters			90 + 4x + m*		8,9,11,12,18			
Via TSS			180 + m*		8,9,11,12,18			
Via task gate			185 + m*		8,9,11,12,18			
JMP = Unconditional jump:								
Short/long	<table><tr><td>11101011</td><td>disp-low</td></tr></table>	11101011	disp-low	7 + m	7 + m		18	
11101011	disp-low							
Direct within segment	<table><tr><td>11101001</td><td>disp-low</td><td>disp-high</td></tr></table>	11101001	disp-low	disp-high	7 + m	7 + m		18
11101001	disp-low	disp-high						
Register/memory indirect within segment	<table><tr><td>11111111</td><td>mod 100</td><td>r/m</td></tr></table>	11111111	mod 100	r/m	7 + m, 11 + m*	7 + m, 11 + m*	2	9,18
11111111	mod 100	r/m						
Direct intersegment	<table><tr><td>11101010</td><td colspan="2">segment offset</td></tr></table>	11101010	segment offset		11 + m	23 + m		11,12,18
11101010	segment offset							
Protected Mode Only (Direct intersegment):								
Via call gate to same privilege level			38 + m		8,11,12,18			
Via TSS			175 + m		8,11,12,18			
Via task gate			180 + m		8,11,12,18			
Indirect intersegment	<table><tr><td>11111111</td><td>mod 101</td><td>r/m</td></tr></table> (mod ≠ 11)	11111111	mod 101	r/m	15 + m*	26 + m*	2	8,9,11,12,18
11111111	mod 101	r/m						
Protected Mode Only (Indirect intersegment):								
Via call gate to same privilege level			41 + m*		8,9,11,12,18			
Via TSS			178 + m*		8,9,11,12,18			
Via task gate			183 + m*		8,9,11,12,18			
RET = Return from CALL:								
Within segment	<table><tr><td>11000011</td></tr></table>	11000011	11 + m	11 + m	2	8,9,18		
11000011								
Within seg adding immed to SP	<table><tr><td>11000010</td><td>data-low</td><td>data-high</td></tr></table>	11000010	data-low	data-high	11 + m	11 + m	2	8,9,18
11000010	data-low	data-high						
Intersegment	<table><tr><td>11001011</td></tr></table>	11001011	15 + m	25 + m	2	8,9,11,12,18		
11001011								
Intersegment adding immediate to SP	<table><tr><td>11001010</td><td>data-low</td><td>data-high</td></tr></table>	11001010	data-low	data-high	15 + m		2	8,9,11,12,18
11001010	data-low	data-high						
Protected Mode Only (RET):								
To different privilege level			55 + m		9,11,12,18			

80C286 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)									
JE/JZ = Jump on equal zero	<table><tr><td>0 1 1 1 0 1 0 0</td><td>disp</td></tr></table>	0 1 1 1 0 1 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 0 0	disp								
JL/JNGE = Jump on less/not greater or equal	<table><tr><td>0 1 1 1 1 1 0 0</td><td>disp</td></tr></table>	0 1 1 1 1 1 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 0 0	disp								
JLE/JNG = Jump on less or equal/not greater	<table><tr><td>0 1 1 1 1 1 1 0</td><td>disp</td></tr></table>	0 1 1 1 1 1 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 1 0	disp								
JB/JNAE = Jump on below/not above or equal	<table><tr><td>0 1 1 1 0 0 1 0</td><td>disp</td></tr></table>	0 1 1 1 0 0 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 1 0	disp								
JBE/JNA = Jump on below or equal/not above	<table><tr><td>0 1 1 1 0 1 1 0</td><td>disp</td></tr></table>	0 1 1 1 0 1 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 1 0	disp								
JP/JPE = Jump on parity/parity even	<table><tr><td>0 1 1 1 1 0 1 0</td><td>disp</td></tr></table>	0 1 1 1 1 0 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 1 0	disp								
JO = Jump on overflow	<table><tr><td>0 1 1 1 0 0 0 0</td><td>disp</td></tr></table>	0 1 1 1 0 0 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 0 0	disp								
JS = Jump on sign	<table><tr><td>0 1 1 1 1 0 0 0</td><td>disp</td></tr></table>	0 1 1 1 1 0 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 0 0	disp								
JNE/JNZ = Jump on not equal/not zero	<table><tr><td>0 1 1 1 0 1 0 1</td><td>disp</td></tr></table>	0 1 1 1 0 1 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 0 1	disp								
JNL/JGE = Jump on not less/greater or equal	<table><tr><td>0 1 1 1 1 1 0 1</td><td>disp</td></tr></table>	0 1 1 1 1 1 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 0 1	disp								
JNLE/JG = Jump on not less or equal/greater	<table><tr><td>0 1 1 1 1 1 1 1</td><td>disp</td></tr></table>	0 1 1 1 1 1 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 1 1	disp								
JNB/JAE = Jump on not below/above or equal	<table><tr><td>0 1 1 1 0 0 1 1</td><td>disp</td></tr></table>	0 1 1 1 0 0 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 1 1	disp								
JNBE/JA = Jump on not below or equal/above	<table><tr><td>0 1 1 1 0 1 1 1</td><td>disp</td></tr></table>	0 1 1 1 0 1 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 1 1	disp								
JNP/JPO = Jump on not par/par odd	<table><tr><td>0 1 1 1 1 0 1 1</td><td>disp</td></tr></table>	0 1 1 1 1 0 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 1 1	disp								
JNO = Jump on not overflow	<table><tr><td>0 1 1 1 0 0 0 1</td><td>disp</td></tr></table>	0 1 1 1 0 0 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 0 1	disp								
JNS = Jump on not sign	<table><tr><td>0 1 1 1 1 0 0 1</td><td>disp</td></tr></table>	0 1 1 1 1 0 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 0 1	disp								
LOOP = Loop CX times	<table><tr><td>1 1 1 0 0 0 1 0</td><td>disp</td></tr></table>	1 1 1 0 0 0 1 0	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 1 0	disp								
LOOPZ/LOOPE = Loop while zero/equal	<table><tr><td>1 1 1 0 0 0 0 1</td><td>disp</td></tr></table>	1 1 1 0 0 0 0 1	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 0 1	disp								
LOOPNZ/LOOPNE = Loop while not zero/equal	<table><tr><td>1 1 1 0 0 0 0 0</td><td>disp</td></tr></table>	1 1 1 0 0 0 0 0	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 0 0	disp								
JCXZ = Jump on CX zero	<table><tr><td>1 1 1 0 0 0 1 1</td><td>disp</td></tr></table>	1 1 1 0 0 0 1 1	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 1 1	disp								
ENTER = Enter Procedure	<table><tr><td>1 1 0 0 1 0 0 0</td><td>data-low</td><td>data-high</td><td>L</td></tr></table>	1 1 0 0 1 0 0 0	data-low	data-high	L			2,8	8,9
1 1 0 0 1 0 0 0	data-low	data-high	L						
L = 0		11	11	2,8	8,9				
L = 1		15	15	2,8	8,9				
L > 1		16 + 4(L - 1)	16 + 4(L - 1)	2,8	8,9				
LEAVE = Leave Procedure	<table><tr><td>1 1 0 0 1 0 0 1</td></tr></table>	1 1 0 0 1 0 0 1	5	5					
1 1 0 0 1 0 0 1									
INT = Interrupt:									
Type specified	<table><tr><td>1 1 0 0 1 1 0 1</td><td>type</td></tr></table>	1 1 0 0 1 1 0 1	type	23 + m		2,7,8			
1 1 0 0 1 1 0 1	type								
Type 3	<table><tr><td>1 1 0 0 1 1 0 0</td></tr></table>	1 1 0 0 1 1 0 0	23 + m		2,7,8				
1 1 0 0 1 1 0 0									
INTO = Interrupt on overflow	<table><tr><td>1 1 0 0 1 1 1 0</td></tr></table>	1 1 0 0 1 1 1 0	24 + m or 3 (3 if no interrupt)	(3 if no interrupt)	2,6,8				
1 1 0 0 1 1 1 0									

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80C286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only: Via interrupt or trap gate to same privilege level Via interrupt or trap gate to fit different privilege level Via Task Gate			40 + m 78 + m 167 + m		7,8,11,12,18 7,8,11,12,18 7,8,11,12,18
IRET = Interrupt return	11001111	17 + m	31 + m	2,4	8,9,11,12,15,18
Protected Mode Only: To different privilege level To different task (NT = 1)			55 + m 169 + m		8,9,11,12,15,18 8,9,11,12,18
BOUND = Detect value out of range	01100010 mod reg r/m	13*	13* (Use INT clock count if exception 5)	2,6	6,8,9,11,12,18
PROCESSOR CONTROL					
CLC = Clear carry	11111000	2	2		
CMC = Complement carry	11110101	2	2		
STC = Set carry	11111001	2	2		
CLD = Clear direction	11111100	2	2		
STD = Set direction	11111101	2	2		
CLI = Clear interrupt	11111010	3	3		14
STI = Set interrupt	11111011	2	2		14
HLT = Halt	11110100	2	2		13
WAIT = Wait	10011011	3	3		
LOCK = Bus lock prefix	11110000	0	0		14
CTS = Clear task switched flag	00001111 00000110	2	2	3	13
ESC = Processor Extension Escape	11011TTT mod LLL r/m (TTT LLL are opcode to processor extension)	9–20*	9–20*	5,8	8,17
SEG = Segment Override Prefix	001 reg 110	0	0		
PROTECTION CONTROL					
LGDT = Load global descriptor table register	00001111 00000001 mod 010 r/m	11*	11*	2,3	9,13
SGDT = Store global descriptor table register	00001111 00000001 mod 000 r/m	11*	11*	2,3	9
LIDT = Load interrupt descriptor table register	00001111 00000001 mod 011 r/m	12*	12*	2,3	9,13
SIDT = Store interrupt descriptor table register	00001111 00000001 mod 001 r/m	12*	12*	2,3	9
LLDT = Load local descriptor table register from register memory	00001111 00000000 mod 010 r/m		17,19*	1	9,11,13
SLDT = Store local descriptor table register to register/memory	00001111 00000000 mod 000 r/m		2,3*	1	9

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80C286 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 (Continued)								
LTR = Local task register from register/memory	<table><tr><td>00001111</td><td>00000000</td><td>mod 011 r/m</td></tr></table>	00001111	00000000	mod 011 r/m		17,19*	1	9,11,13
00001111	00000000	mod 011 r/m						
STR = Store task register to register/memory	<table><tr><td>00001111</td><td>00000000</td><td>mod 001 r/m</td></tr></table>	00001111	00000000	mod 001 r/m		2,3*	1	9
00001111	00000000	mod 001 r/m						
LMSW = Load machine status word from register/memory	<table><tr><td>00001111</td><td>00000001</td><td>mod 110 r/m</td></tr></table>	00001111	00000001	mod 110 r/m	3,6*	3,6*	2,3	9,13
00001111	00000001	mod 110 r/m						
SMSW = Store machine status word	<table><tr><td>00001111</td><td>00000001</td><td>mod 100 r/m</td></tr></table>	00001111	00000001	mod 100 r/m	2,3*	2,3*	2,3	9
00001111	00000001	mod 100 r/m						
LAR = Load access rights from register/memory	<table><tr><td>00001111</td><td>00000010</td><td>mod reg r/m</td></tr></table>	00001111	00000010	mod reg r/m		14,16*	1	9,11,16
00001111	00000010	mod reg r/m						
LSL = Load segment limit from register/memory	<table><tr><td>00001111</td><td>00000011</td><td>mod reg r/m</td></tr></table>	00001111	00000011	mod reg r/m		14,16*	1	9,11,16
00001111	00000011	mod reg r/m						
ARPL = Adjust requested privilege level: from register/memory	<table><tr><td></td><td>01100011</td><td>mod reg r/m</td></tr></table>		01100011	mod reg r/m		10*,11*	2	8,9
	01100011	mod reg r/m						
VERR = Verify read access: register/memory	<table><tr><td>00001111</td><td>00000000</td><td>mod 100 r/m</td></tr></table>	00001111	00000000	mod 100 r/m		14,16*	1	9,11,16
00001111	00000000	mod 100 r/m						
VERR = Verify write access:	<table><tr><td>00001111</td><td>00000000</td><td>mod 101 r/m</td></tr></table>	00001111	00000000	mod 101 r/m		14,16*	1	9,11,16
00001111	00000000	mod 101 r/m						

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Footnotes

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

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

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

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

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

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DC

REG is assigned according to the following table:

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

DATA SHEET REVISION REVIEW

3

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

1. The test conditions in the A.C. Characteristics table has been changed.
2. The "Typical I_{CC} vs Frequency for Different Output Loads" graph has been modified.
3. The maximum ambient temperature (T_A) vs. various airflows has been updated.
4. Deleted the 82C284 and 82C288 A.C. Characteristics tables.
5. "PRELIMINARY" status was removed from the datasheet.



80286

High Performance Microprocessor with Memory Management and Protection

(80286-12, 80286-10, 80286-8)

- High Performance HMOS III Technology
 - Large Address Space:
 - 16 Megabytes Physical
 - 1 Gigabyte Virtual per Task
 - Integrated Memory Management, Four-Level Memory Protection and Support for Virtual Memory and Operating Systems
 - High Bandwidth Bus Interface (12.5 Megabyte/Sec)
 - Industry Standard O.S. Support:
 - MS-DOS*, UNIX*, XENIX*, iRMX®
 - Optional Processor Extension:
 - 80287 High Performance 80-bit Numeric Data Processor
 - Two 8086 Upward Compatible Operating Modes:
 - 8086 Real Address Mode
 - Protected Virtual Address Mode
 - Complete System Development Support:
 - Assembler, PL/M, Pascal, FORTRAN, and In-Circuit-Emulator (ICETM-286)
 - Available in:
 - 68-Pin PLCC (Plastic Leaded Chip Carrier)
 - 68-Pin LCC (Leadless Chip Carrier)
 - 68-Pin PGA (Pin Grid Array)
- (See Packaging Spec., Order #231369)

The 80286 is an advanced, high-performance microprocessor with specially optimized capabilities for multiple user and multi-tasking systems. The 80286 has built-in memory protection that supports operating system and task isolation as well as program and data privacy within tasks. A 12.5 MHz 80286 provides six times or more throughput than the standard 5 MHz 8086. The 80286 includes memory management capabilities that map 2^{30} (one gigabyte) of virtual address space per task into 2^{24} bytes (16 megabytes) of physical memory.

The 80286 is upward compatible with 8086 and 88 software. Using 8086 real address mode, the 80286 is object code compatible with existing 8086, 88 software. In protected virtual address mode, the 80286 is source code compatible with 8086, 88 software and may require upgrading to use virtual addresses supported by the 80286's integrated memory management and protection mechanism. Both modes operate at full 80286 performance and execute a superset of the 8086 and 88 instructions.

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

*XENIX and MS-DOS are trademarks of Microsoft Corp.

*UNIX is a trademark of Bell Labs or AT&T

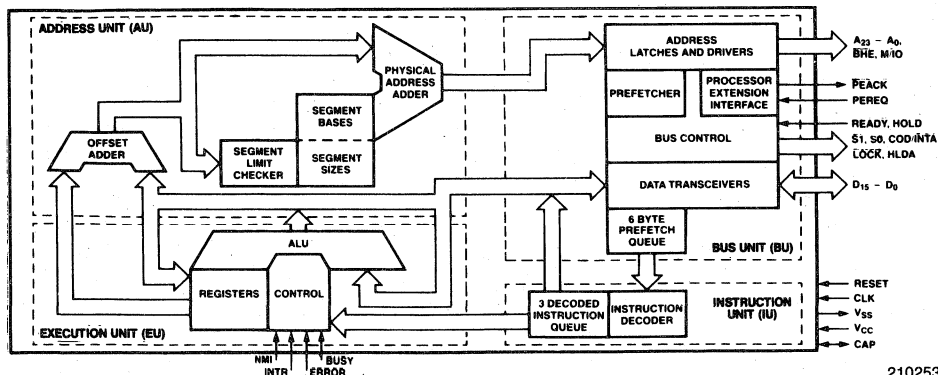


Figure 1. 80286 Internal Block Diagram

210253-1

Component Pad Views—As viewed from underside of component when mounted on the board.

P.C. Board Views—As viewed from the component side of the P.C. board.

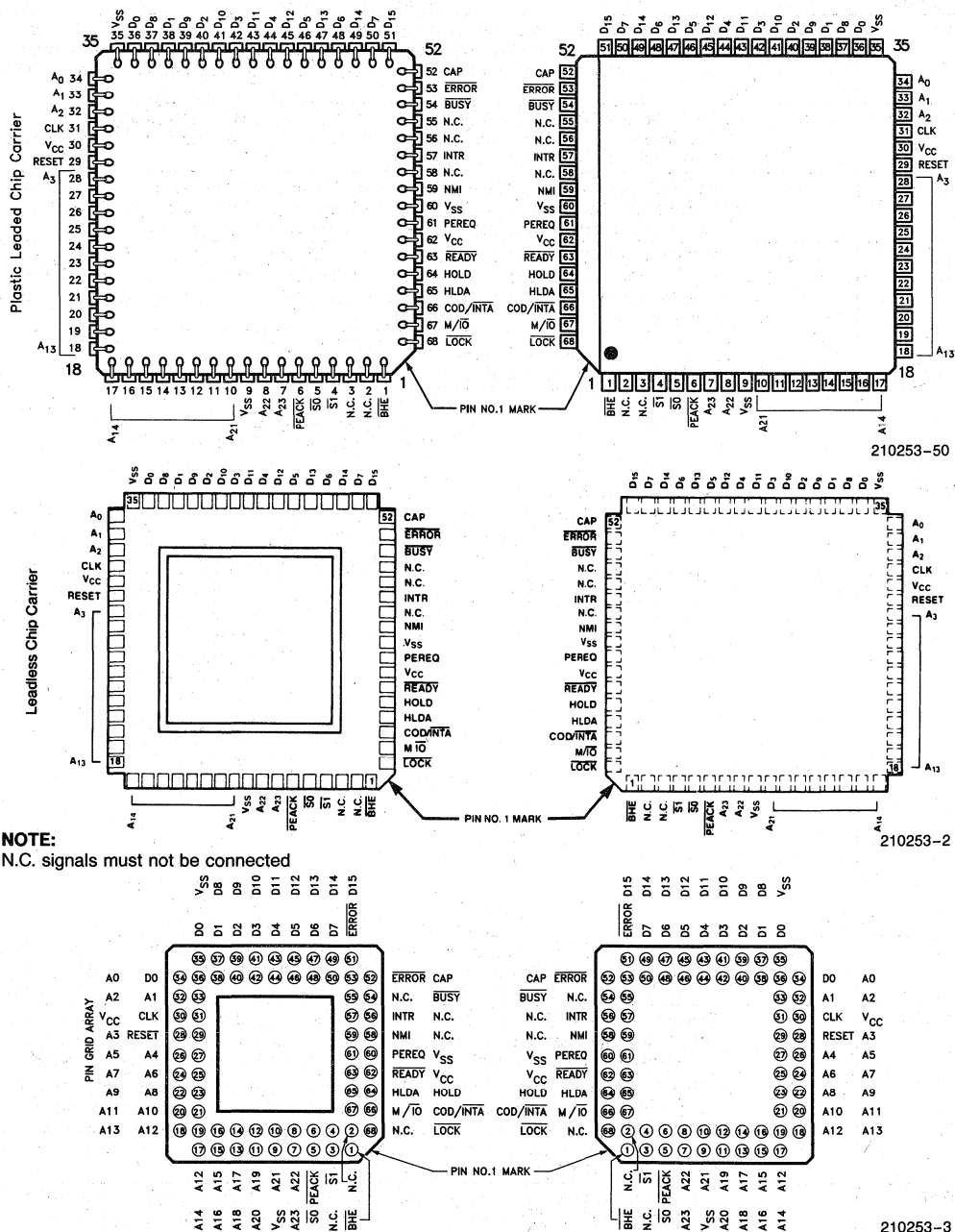


Figure 2. 80286 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for the 80286 microprocessor:

Symbol	Type	Name and Function																																																																																										
CLK	I	SYSTEM CLOCK provides the fundamental timing for 80286 systems. It is divided by two inside the 80286 to generate the processor clock. The internal divide-by-two circuitry can be synchronized to an external clock generator by a LOW to HIGH transition on the RESET input.																																																																																										
D ₁₅ –D ₀	I/O	DATA BUS inputs data during memory, I/O, and interrupt acknowledge read cycles; outputs data during memory and I/O write cycles. The data bus is active HIGH and floats to 3-state OFF during bus hold acknowledge.																																																																																										
A ₂₃ –A ₀	O	ADDRESS BUS outputs physical memory and I/O port addresses. A ₀ is LOW when data is to be transferred on pins D ₇ – ₀ . A ₂₃ –A ₁₆ are LOW during I/O transfers. The address bus is active HIGH and floats to 3-state OFF during bus hold acknowledge.																																																																																										
BHE	O	BUS HIGH ENABLE indicates transfer of data on the upper byte of the data bus. D ₁₅ – ₈ . Eight-bit oriented devices assigned to the upper byte of the data bus would normally use BHE to condition chip select functions. BHE is active LOW and floats to 3-state OFF during bus hold acknowledge.																																																																																										
		<table><tr><th colspan="3">BHE and A0 Encodings</th></tr><tr><th>BHE Value</th><th>A0 Value</th><th>Function</th></tr><tr><td>0</td><td>0</td><td>Word transfer</td></tr><tr><td>0</td><td>1</td><td>Byte transfer on upper half of data bus (D₁₅–D₈)</td></tr><tr><td>1</td><td>0</td><td>Byte transfer on lower half of data bus (D₇–₀)</td></tr><tr><td>1</td><td>1</td><td>Will never occur</td></tr></table>	BHE and A0 Encodings			BHE Value	A0 Value	Function	0	0	Word transfer	0	1	Byte transfer on upper half of data bus (D ₁₅ –D ₈)	1	0	Byte transfer on lower half of data bus (D ₇ – ₀)	1	1	Will never occur																																																																								
BHE and A0 Encodings																																																																																												
BHE Value	A0 Value	Function																																																																																										
0	0	Word transfer																																																																																										
0	1	Byte transfer on upper half of data bus (D ₁₅ –D ₈)																																																																																										
1	0	Byte transfer on lower half of data bus (D ₇ – ₀)																																																																																										
1	1	Will never occur																																																																																										
S ₁ , S ₀	O	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, S ₁ and S ₀ are active LOW and float to 3-state OFF during bus hold acknowledge.																																																																																										
		<table><tr><th colspan="5">80286 Bus Cycle Status Definition</th></tr><tr><th>COD/$\overline{\text{INTA}}$</th><th>M/$\overline{\text{IO}}$</th><th>S₁</th><th>S₀</th><th>Bus Cycle Initiated</th></tr><tr><td>0 (LOW)</td><td>0</td><td>0</td><td>0</td><td>Interrupt acknowledge</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Will not occur</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>Will not occur</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>IF A₁ = 1 then halt; else shutdown</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>Memory data read</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>Memory data write</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr><tr><td>1 (HIGH)</td><td>0</td><td>0</td><td>0</td><td>Will not occur</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>I/O read</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>I/O write</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>Will not occur</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>Memory instruction read</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>Will not occur</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>None; not a status cycle</td></tr></table>	80286 Bus Cycle Status Definition					COD/ $\overline{\text{INTA}}$	M/ $\overline{\text{IO}}$	S ₁	S ₀	Bus Cycle Initiated	0 (LOW)	0	0	0	Interrupt acknowledge	0	0	0	1	Will not occur	0	0	1	0	Will not occur	0	0	1	1	None; not a status cycle	0	1	0	0	IF 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	Will not occur	1	0	0	1	I/O read	1	0	1	0	I/O write	1	0	1	1	None; not a status cycle	1	1	0	0	Will not occur	1	1	0	1	Memory instruction read	1	1	1	0	Will not occur	1	1	1	1	None; not a status cycle
80286 Bus Cycle Status Definition																																																																																												
COD/ $\overline{\text{INTA}}$	M/ $\overline{\text{IO}}$	S ₁	S ₀	Bus Cycle Initiated																																																																																								
0 (LOW)	0	0	0	Interrupt acknowledge																																																																																								
0	0	0	1	Will not occur																																																																																								
0	0	1	0	Will not occur																																																																																								
0	0	1	1	None; not a status cycle																																																																																								
0	1	0	0	IF 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	Will not occur																																																																																								
1	0	0	1	I/O read																																																																																								
1	0	1	0	I/O write																																																																																								
1	0	1	1	None; not a status cycle																																																																																								
1	1	0	0	Will not occur																																																																																								
1	1	0	1	Memory instruction read																																																																																								
1	1	1	0	Will not occur																																																																																								
1	1	1	1	None; not a status cycle																																																																																								
M/ $\overline{\text{IO}}$	O	MEMORY I/O SELECT distinguishes memory access from I/O access. If HIGH during T _s , a memory cycle or a halt/shutdown cycle is in progress. If LOW, an I/O cycle or an interrupt acknowledge cycle is in progress. M/ $\overline{\text{IO}}$ floats to 3-state OFF during bus hold acknowledge.																																																																																										
COD/ $\overline{\text{INTA}}$	O	CODE/INTERRUPT ACKNOWLEDGE distinguishes instruction fetch cycles from memory data read cycles. Also distinguishes interrupt acknowledge cycles from I/O cycles. COD/ $\overline{\text{INTA}}$ floats to 3-state OFF during bus hold acknowledge. Its timing is the same as M/ $\overline{\text{IO}}$.																																																																																										
LOCK	O	BUS LOCK indicates that other system bus masters are not to gain control of the system bus for the current and the following bus cycle. The 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. LOCK is active LOW and floats to 3-state OFF during bus hold acknowledge.																																																																																										
READY	I	BUS READY terminates a bus cycle. Bus cycles are extended without limit until terminated by READY LOW. READY is an active LOW synchronous input requiring setup and hold times relative to the system clock be met for correct operation. READY is ignored during bus hold acknowledge.																																																																																										

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
HOLD HLDA	I O	BUS HOLD REQUEST AND HOLD ACKNOWLEDGE control ownership of the 80286 local bus. The HOLD input allows another local bus master to request control of the local bus. When control is granted, the 80286 will float its bus drivers to 3-state OFF and then activate HLDA, thus entering the bus hold acknowledge condition. The local bus will remain granted to the requesting master until HOLD becomes inactive which results in the 80286 deactivating HLDA and regaining control of the local bus. This terminates the bus hold acknowledge condition. HOLD may be asynchronous to the system clock. These signals are active HIGH.
INTR	I	INTERRUPT REQUEST requests the 80286 to suspend its current program execution and service a pending external request. Interrupt requests are masked whenever the interrupt enable bit in the flag word is cleared. When the 80286 responds to an interrupt request, it performs two interrupt acknowledge bus cycles to read an 8-bit interrupt vector that identifies the source of the interrupt. To assure program interruption, INTR must remain active until the first interrupt acknowledge cycle is completed. INTR is sampled at the beginning of each processor cycle and must be active HIGH at least two processor cycles before the current instruction ends in order to interrupt before the next instruction. INTR is level sensitive, active HIGH, and may be asynchronous to the system clock.
NMI	I	NON-MASKABLE INTERRUPT REQUEST interrupts the 80286 with an internally supplied vector value of 2. No interrupt acknowledge cycles are performed. The interrupt enable bit in the 80286 flag word does not affect this input. The NMI input is active HIGH, may be asynchronous to the system clock, and is edge triggered after internal synchronization. For proper recognition, the input must have been previously LOW for at least four system clock cycles and remain HIGH for at least four system clock cycles.
PEREQ PEACK	I O	PROCESSOR EXTENSION OPERAND REQUEST AND ACKNOWLEDGE extend the memory management and protection capabilities of the 80286 to processor extensions. The PEREQ input requests the 80286 to perform a data operand transfer for a processor extension. The PEACK output signals the processor extension when the requested operand is being transferred. PEREQ is active HIGH and floats to 3-state OFF during bus hold acknowledge. PEACK may be asynchronous to the system clock. PEACK is active LOW.
BUSY ERROR	I I	PROCESSOR EXTENSION BUSY AND ERROR indicate the operating condition of a processor extension to the 80286. An active BUSY input stops 80286 program execution on WAIT and some ESC instructions until BUSY becomes inactive (HIGH). The 80286 may be interrupted while waiting for BUSY to become inactive. An active ERROR input causes the 80286 to perform a processor extension interrupt when executing WAIT or some ESC instructions. These inputs are active LOW and may be asynchronous to the system clock. These inputs have internal pull-up resistors.

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function	
RESET	I	SYSTEM RESET clears the internal logic of the 80286 and is active HIGH. The 80286 may be reinitialized at any time with a LOW to HIGH transition on RESET which remains active for more than 16 system clock cycles. During RESET active, the output pins of the 80286 enter the state shown below:	
		80286 Pin State During Reset	
		Pin Value	Pin Names
		1 (HIGH) 0 (LOW) 3-state OFF	S0, S1, PEACK, A23–A0, BHE, LOCK M/IO, COD/INTA, HLDA (Note 1) D15–D0
		Operation of the 80286 begins after a HIGH to LOW transition on RESET. The HIGH to LOW transition of RESET must be synchronous to the system clock. Approximately 38 CLK cycles from the trailing edge of RESET are required by the 80286 for internal initialization before the first bus cycle, to fetch code from the power-on execution address, occurs. A LOW to HIGH transition of RESET synchronous to the system clock will end a processor cycle at the second HIGH to LOW transition of the system clock. The LOW to HIGH transition of RESET may be asynchronous to the system clock; however, in this case it cannot be predetermined which phase of the processor clock will occur during the next system clock period. Synchronous LOW to HIGH transitions of RESET are required only for systems where the processor clock must be phase synchronous to another clock.	
VSS	I	SYSTEM GROUND: 0 Volts.	
VCC	I	SYSTEM POWER: + 5 Volt Power Supply.	
CAP	I	SUBSTRATE FILTER CAPACITOR: a $0.047\ \mu\text{F} \pm 20\%$ 12V capacitor must be connected between this pin and ground. This capacitor filters the output of the internal substrate bias generator. A maximum DC leakage current of $1\ \mu\text{A}$ is allowed through the capacitor. For correct operation of the 80286, the substrate bias generator must charge this capacitor to its operating voltage. The capacitor chargeup time is 5 milliseconds (max.) after VCC and CLK reach their specified AC and DC parameters. RESET may be applied to prevent spurious activity by the CPU during this time. After this time, the 80286 processor clock can be synchronized to another clock by pulsing RESET LOW synchronous to the system clock.	

NOTE:

1. HLDA is only Low if HOLD is inactive (Low).

FUNCTIONAL DESCRIPTION

Introduction

The 80286 is an advanced, high-performance micro-processor with specially optimized capabilities for multiple user and multi-tasking systems. Depending on the application, a 12.5 MHz 80286's performance is up to six times faster than the standard 5 MHz 8086's, while providing complete upward software compatibility with Intel's 8086, 88, and 186 family of CPU's.

The 80286 operates in two modes: 8086 real address mode and protected virtual address mode. Both modes execute a superset of the 8086 and 88 instruction set.

In 8086 real address mode programs use real addresses with up to one megabyte of address space. Programs use virtual addresses in protected virtual address mode, also called protected mode. In protected mode, the 80286 CPU automatically maps 1 gigabyte of virtual addresses per task into a 16 megabyte real address space. This mode also provides memory protection to isolate the operating system and ensure privacy of each tasks' programs and data. Both modes provide the same base instruction set, registers, and addressing modes.

The following Functional Description describes first, the base 80286 architecture common to both modes, second, 8086 real address mode, and third, protected mode.

80286 BASE ARCHITECTURE

The 8086, 88, 186, and 286 CPU family all contain the same basic set of registers, instructions, and

addressing modes. The 80286 processor is upward compatible with the 8086, 8088, and 80186 CPU's.

Register Set

The 80286 base architecture has fifteen registers as shown in Figure 3. These registers are grouped into the following four categories:

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

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

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

Status and Control Registers: The 3 16-bit special purpose registers in figure 3A record or control certain aspects of the 80286 processor state including the Instruction Pointer, which contains the offset address of the next sequential instruction to be executed.

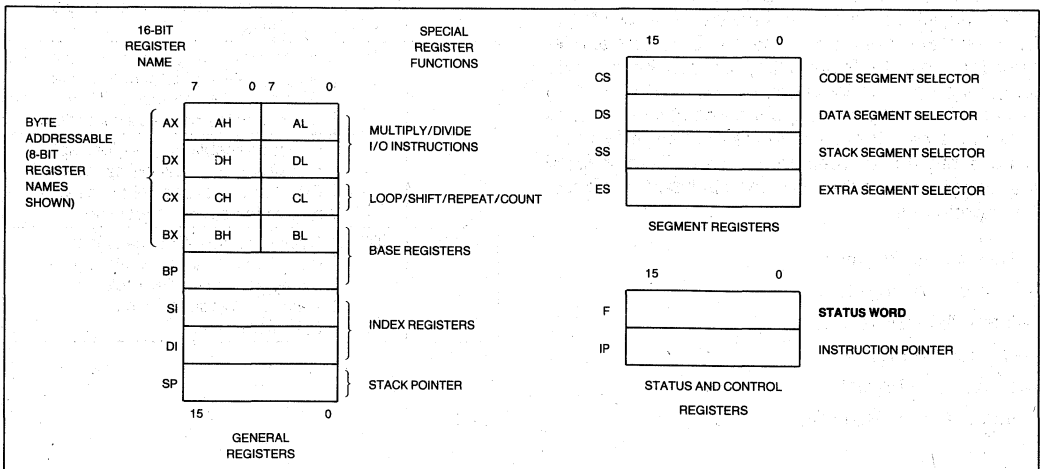


Figure 3. Register Set

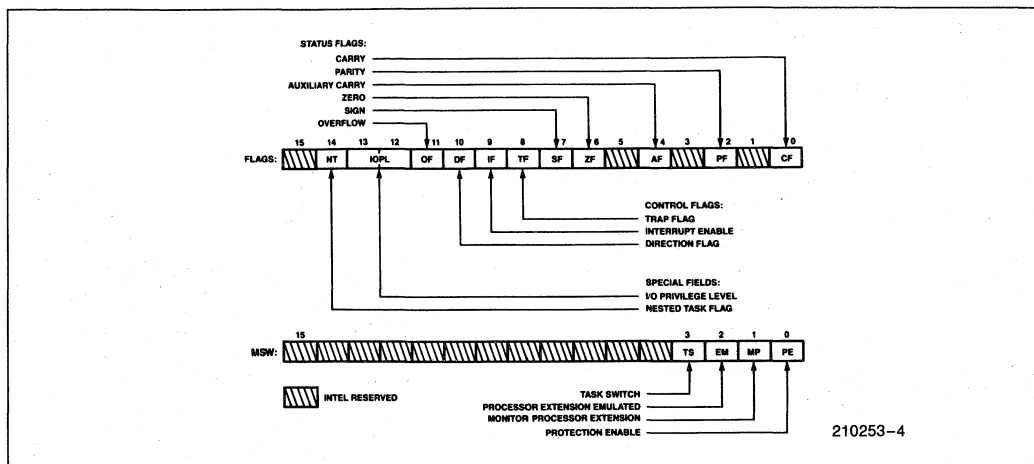


Figure 3a. Status and Control Register Bit Functions

Flags Word Description

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

Instruction Set

The instruction set is divided into seven categories: data transfer, arithmetic, shift/rotate/logical, string manipulation, control transfer, high level instructions, and processor control. These categories are summarized in Figure 4.

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

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

Table 2. Flags Word Bit Functions

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

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

For detailed operation and usage of each instruction, see Appendix of 80286 Programmer's Reference Manual (Order No. 210498)

GENERAL PURPOSE	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push all registers on stack
POPA	Pop all registers from stack
XCHG	Exchange byte or word
XLAT	Translate byte
INPUT/OUTPUT	
IN	Input byte or word
OUT	Output byte or word
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
FLAG TRANSFER	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack

Figure 4a. Data Transfer Instructions

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

Figure 4c. String Instructions

ADDITION	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
MULTIPLICATION	
MUL	Multiple byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply
DIVISION	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to doubleword

Figure 4b. Arithmetic Instructions

LOGICALS	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word
SHIFTS	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word
ROTATES	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

Figure 4d. Shift/Rotate Logical Instructions

CONDITIONAL TRANSFERS		UNCONDITIONAL TRANSFERS	
JA/JNBE	Jump if above/not below nor equal	CALL	Call procedure
JAE/JNB	Jump if above or equal/not below	RET	Return from procedure
JB/JNAE	Jump if below/not above nor equal	JMP	Jump
JBE/JNA	Jump if below or equal/not above		
JC	Jump if carry	ITERATION CONTROLS	
JE/JZ	Jump if equal/zero	LOOP	Loop
JG/JNLE	Jump if greater/not less nor equal		
JGE/JNL	Jump if greater or equal/not less	LOOPE/LOOPZ	Loop if equal/zero
JL/JNGE	Jump if less/not greater nor equal	LOOPNE/LOOPNZ	Loop if not equal/not zero
JLE/JNG	Jump if less or equal/not greater	JCXZ	Jump if register CX = 0
JNC	Jump if not carry		
JNE/JNZ	Jump if not equal/not zero	INTERRUPTS	
JNO	Jump if not overflow	INT	Interrupt
JNP/JPO	Jump if not parity/parity odd		
JNS	Jump if not sign	INTO	Interrupt if overflow
JO	Jump if overflow	IRET	Interrupt return
JP/JPE	Jump if parity/parity even		
JS	Jump if sign		

Figure 4e. Program Transfer Instructions

FLAG OPERATIONS	
STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag
EXTERNAL SYNCHRONIZATION	
HLT	Halt until interrupt or reset
WAIT	Wait for BUSY not active
ESC	Escape to extension processor
LOCK	Lock bus during next instruction
NO OPERATION	
NOP	No operation
EXECUTION ENVIRONMENT CONTROL	
LMSW	Load machine status word
SMSW	Store machine status word

Figure 4f. Processor Control Instructions

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

Figure 4g. High Level Instructions

Memory Organization

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

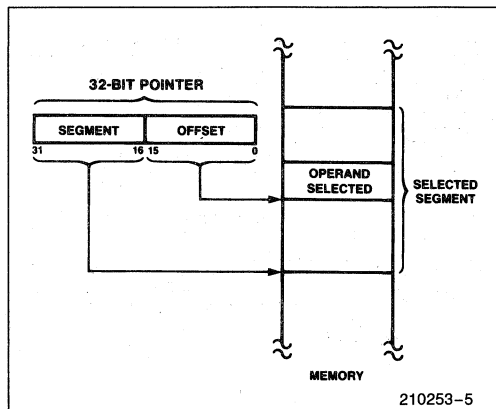


Figure 5. Two Component Address

Table 3. Segment Register Selection Rules

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

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

Most instructions need not explicitly specify which segment register is used. The correct segment register is automatically chosen according to the rules of Table 3. These rules follow the way programs are written (see Figure 6) as independent modules that require areas for code and data, a stack, and access to external data areas.

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

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)

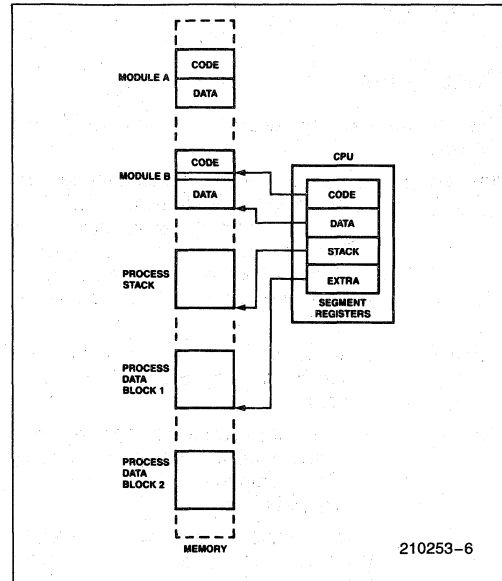


Figure 6. Segmented Memory Helps Structure Software

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

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

Combinations of these three address elements define the six memory addressing modes, described below.

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

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

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

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

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

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

Data Types

The 80286 directly supports the following data types:

- Integer:** A signed binary numeric value contained in an 8-bit byte or a 16-bit word. All operations assume a 2's complement representation. Signed 32 and 64-bit integers are supported using the Numeric Data Processor, the 80287.
- Ordinal:** An unsigned binary numeric value contained in an 8-bit byte or 16-bit word.
- Pointer:** A 32-bit quantity, composed of a segment selector component and an offset component. Each component is a 16-bit word.
- String:** A contiguous sequence of bytes or words. A string may contain from 1 byte to 64K bytes.
- ASCII:** A byte representation of alphanumeric and control characters using the ASCII standard of character representation.
- BCD:** A byte (unpacked) representation of the decimal digits 0–9.
- Packed BCD:** A byte (packed) representation of two decimal digits 0–9 storing one digit in each nibble of the byte.
- Floating Point:** A signed 32, 64, or 80-bit real number representation. (Floating point operands are supported using the 80287 Numeric Processor).

Figure 7 graphically represents the data types supported by the 80286.

either an 8-bit port address, specified in the instruction, or a 16-bit port address in the DX register. 8-bit port addresses are zero extended such that A₁₅–A₈ are LOW. I/O port addresses 00F8(H) through 00FF(H) are reserved.

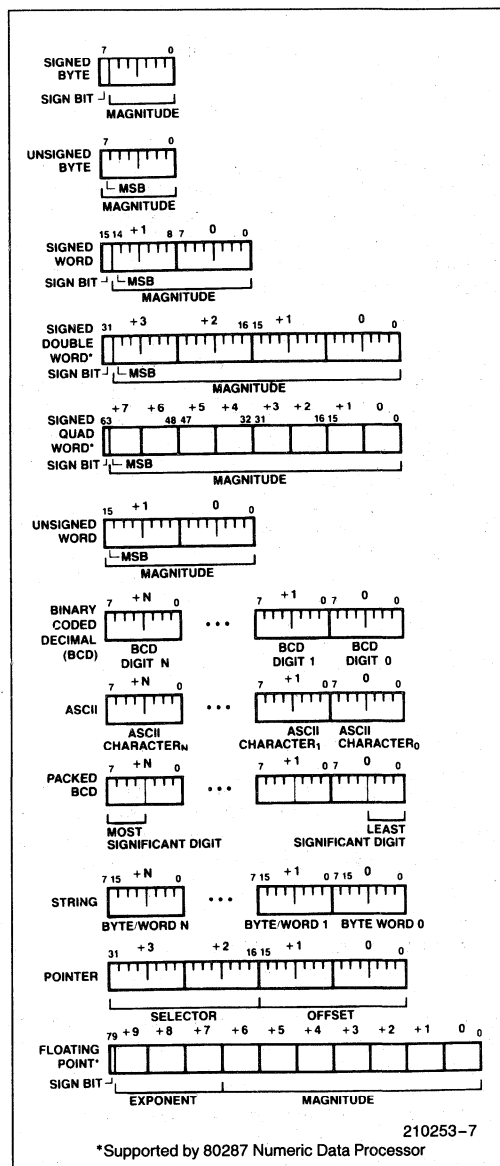


Figure 7. 80286 Supported Data Types

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

Table 4. Interrupt Vector Assignments

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

Interrupts

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

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

MASKABLE INTERRUPT (INTR)

The 80286 provides a maskable hardware interrupt request pin, INTR. Software enables this input by

setting the interrupt flag bit (IF) in the flag word. All 224 user-defined interrupt sources can share this input, yet they can retain separate interrupt handlers. An 8-bit vector read by the CPU during the interrupt acknowledge sequence (discussed in System Interface section) identifies the source of the interrupt.

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

NON-MASKABLE INTERRUPT REQUEST (NMI)

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

While executing the NMI servicing procedure, the 80286 will service neither further NMI requests, INTR requests, nor the processor extension segment overrun interrupt until an interrupt return (IRET) instruction is executed or the CPU is reset. If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. IF is cleared at the beginning of an NMI interrupt to inhibit INTR interrupts.

SINGLE STEP INTERRUPT

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

Interrupt Priorities

When simultaneous interrupt requests occur, they are processed in a fixed order as shown in Table 5. Interrupt processing involves saving the flags, return address, and setting CS:IP to point at the first instruction of the interrupt handler. If other interrupts remain enabled they are processed before the first instruction of the current interrupt handler is executed. The last interrupt processed is therefore the first one serviced.

Table 5. Interrupt Processing Order

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

Initialization and Processor Reset

Processor initialization or start up is accomplished by driving the RESET input pin HIGH. RESET forces the 80286 to terminate all execution and local bus activity. No instruction or bus activity will occur as long as RESET is active. After RESET becomes inactive and an internal processing interval elapses, the 80286 begins execution in real address mode with the instruction at physical location FFFF0(H). RESET also sets some registers to predefined values as shown in Table 6.

Table 6. 80286 Initial Register State after RESET

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

HOLD must not be active during the time from the leading edge of RESET to 34 CLKs after the trailing edge of RESET.

Machine Status Word Description

The machine status word (MSW) records when a task switch takes place and controls the operating mode of the 80286. It is a 16-bit register of which the lower four bits are used. One bit places the CPU into protected mode, while the other three bits, as shown in Table 7, control the processor extension interface. After RESET, this register contains FFF0(H) which places the 80286 in 8086 real address mode.

Table 7. MSW Bit Functions

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

The LMSW and SMSW instructions can load and store the MSW in real address mode. The recommended use of TS, EM, and MP is shown in Table 8.

Table 8. Recommended MSW Encodings For Processor Extension Control

TS	MP	EM	Recommended Use	Instructions Causing Exception 7
0	0	0	Initial encoding after RESET. 80286 operation is identical to 8086, 88.	None
0	0	1	No processor extension is available. Software will emulate its function.	ESC
1	0	1	No processor extension is available. Software will emulate its function. The current processor extension context may belong to another task.	ESC
0	1	0	A processor extension exists.	None
1	1	0	A processor extension exists. The current processor extension context may belong to another task. The Exception 7 on WAIT allows software to test for an error pending from a previous processor extension operation.	ESC or WAIT

Halt

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

8086 REAL ADDRESS MODE

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

Memory Size

Physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A₀ through A₁₉ and BHE. A₂₀ through A₂₃ should be ignored.

Memory Addressing

In real address mode physical memory is a contiguous array of up to 1,048,576 bytes (one megabyte) addressed by pins A₀ through A₁₉ and BHE. Address bits A₂₀–A₂₃ may not always be zero in real mode. A₂₀–A₂₃ should not be used by the system while the 80286 is operating in Real Mode.

The selector portion of a pointer is interpreted as the upper 16 bits of a 20-bit segment address. The lower four bits of the 20-bit segment address are always zero. Segment addresses, therefore, begin on multiples of 16 bytes. See Figure 8 for a graphic representation of address information.

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

Reserved Memory Locations

The 80286 reserves two fixed areas of memory in real address mode (see Figure 9); system initializa-

tion area and interrupt table area. Locations from addresses FFFF0(H) through FFFFF(H) are reserved for system initialization. Initial execution begins at location FFFF0(H). Locations 00000(H) through 003FF(H) are reserved for interrupt vectors.

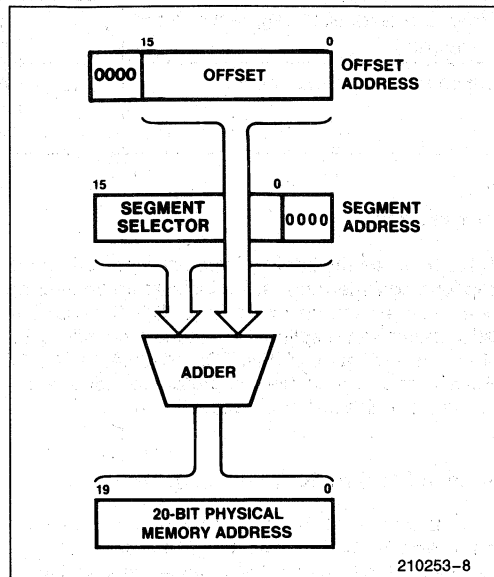


Figure 8. 8086 Real Address Mode Address Calculation

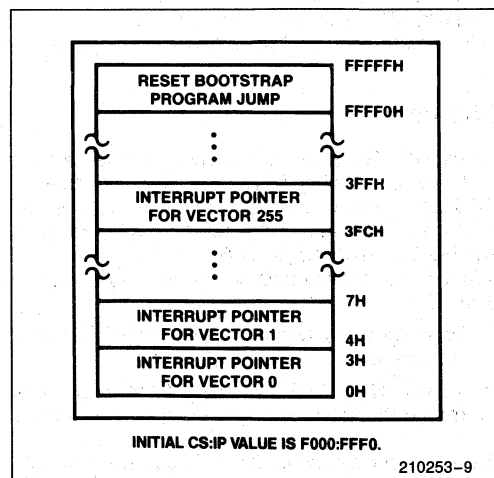


Figure 9. 8086 Real Address Mode Initially Reserved Memory Locations

Table 9. Real Address Mode Addressing Interrupts

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

Interrupts

Table 9 shows the interrupt vectors reserved for exceptions and interrupts which indicate an addressing error. The exceptions leave the CPU in the state existing before attempting to execute the failing instruction (except for PUSH, POP, PUSHA, or POPA). Refer to the next section on protected mode initialization for a discussion on exception 8.

Protected Mode Initialization

To prepare the 80286 for protected mode, the LIDT instruction is used to load the 24-bit interrupt table base and 16-bit limit for the protected mode interrupt table. This instruction can also set a base and limit for the interrupt vector table in real address mode. After reset, the interrupt table base is initialized to 000000(H) and its size set to 03FF(H). These values are compatible with 8086, 88 software. LIDT should only be executed in preparation for protected mode.

Shutdown

Shutdown occurs when a severe error is detected that prevents further instruction processing by the CPU. Shutdown and halt are externally signalled via a halt bus operation. They can be distinguished by A₁ HIGH for halt and A₁ LOW for shutdown. In real address mode, shutdown can occur under two conditions:

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

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

PROTECTED VIRTUAL ADDRESS MODE

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

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

All registers, instructions, and addressing modes described in the 80286 Base Architecture section of this Functional Description remain the same. Programs for the 8086, 88, 186, and real address mode 80286 can be run in protected mode; however, embedded constants for segment selectors are different.

Memory Size

The protected mode 80286 provides a 1 gigabyte virtual address space per task mapped into a 16 megabyte physical address space defined by the address pin A₂₃-A₀ and BHE. The virtual address space may be larger than the physical address space since any use of an address that does not map to a physical memory location will cause a restartable exception.

Memory Addressing

As in real address mode, protected mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector, however, specifies an index into a memory resident table rather than the upper 16-bits of a real memory address. The 24-bit

base address of the desired segment is obtained from the tables in memory. The 16-bit offset is added to the segment base address to form the physical address as shown in Figure 10. The tables are automatically referenced by the CPU whenever a segment register is loaded with a selector. All 80286 instructions which load a segment register will reference the memory based tables without additional software. The memory based tables contain 8 byte values called descriptors.

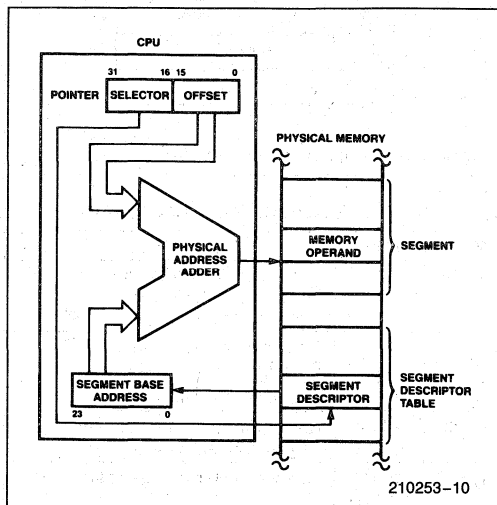


Figure 10. Protected Mode Memory Addressing

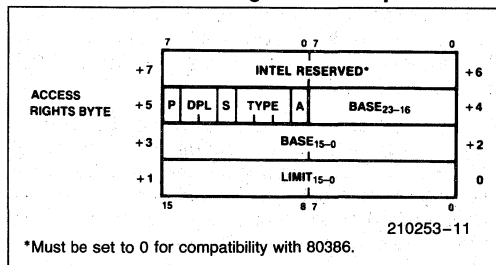
DESCRIPTORS

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

CODE AND DATA SEGMENT DESCRIPTORS (S = 1)

Besides segment base addresses, code and data descriptors contain other segment attributes including segment size (1 to 64K bytes), access rights (read only, read/write, execute only, and execute/read), and presence in memory (for virtual memory systems) (See Figure 11). Any segment usage violating a segment attribute indicated by the segment descriptor will prevent the memory cycle and cause an exception or interrupt.

Code or Data Segment Descriptor



*Must be set to 0 for compatibility with 80386.

Access Rights Byte Definition

Bit Position	Name	Function
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 Level (DPL)	Segment privilege attribute used in privilege tests.
4	Segment Descriptor (S)	S = 1 Code or Data (includes stacks) segment descriptor S = 0 System Segment Descriptor or Gate Descriptor
3	Executable (E)	E = 0 Data segment descriptor type is:
2	Expansion Direction (ED)	ED = 0 Expand up segment, offsets must be ≤ limit.
1	Writable (W)	ED = 1 Expand down segment, offsets must be > limit. 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:
2	Conforming (C)	C = 1 Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.
1	Readable (R)	R = 0 Code segment may not be read R = 1 Code segment may be read.
0	Accessed (A)	A = 0 Segment has not been accessed. A = 1 Segment selector has been loaded into segment register or used by selector test instructions.

Type Field Definition

Figure 11. Code and Data Segment Descriptor Formats

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

Data segments ($S = 1, E = 0$) may be either read-only or read-write as controlled by the W bit of the access rights byte. Read-only ($W = 0$) data segments may not be written into. Data segments may grow in two directions, as determined by the Expansion Direction (ED) bit: upwards ($ED = 0$) for data segments, and downwards ($ED = 1$) for a segment containing a stack. The limit field for a data segment descriptor is interpreted differently depending on the ED bit (see Figure 11).

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

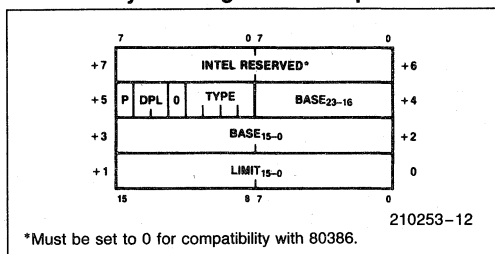
SYSTEM SEGMENT DESCRIPTORS ($S = 0$, $TYPE = 1-3$)

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

Figure 12 gives the formats for the special system data segment descriptors. The descriptors contain a 24-bit base address of the segment and a 16-bit limit. The access byte defines the type of descriptor, its state and privilege level. The descriptor contents are valid and the segment is in physical memory if $P = 1$. If $P = 0$, the segment is not valid. The DPL field is only used in Task State Segment descriptors and indicates the privilege level at which the descrip-

tor may be used (see Privilege). Since the Local Descriptor Table descriptor may only be used by a special privileged instruction, the DPL field is not used. Bit 4 of the access byte is 0 to indicate that it is a system control descriptor. The type field specifies the descriptor type as indicated in Figure 12.

System Segment Descriptor



System Segment Descriptor Fields

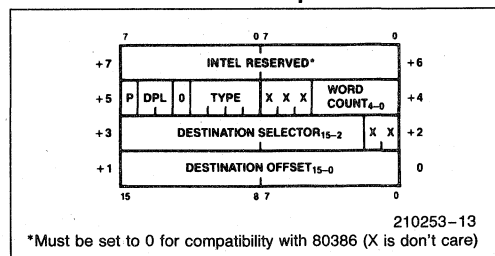
Name	Value	Description
TYPE	1	Available Task State Segment (TSS)
	2	Local Descriptor Table
	3	Busy Task State Segment (TSS)
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid
DPL	0-3	Descriptor Privilege Level
BASE	24-bit number	Base Address of special system data segment in real memory
LIMIT	16-bit number	Offset of last byte in segment

Figure 12. System Segment Descriptor Format

GATE DESCRIPTORS ($S = 0$, $TYPE = 4-7$)

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

Gate Descriptor



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 SELECTOR	16-bit selector	Selector to the target code segment (Call, Interrupt or Trap Gate) Selector to the target task state segment (Task Gate)
DESTINATION OFFSET	16-bit offset	Entry point within the target code segment

Figure 13. Gate Descriptor Format

Figure 13 shows the format of the gate descriptors. The descriptor contains a destination pointer that points to the descriptor of the target segment and the entry point offset. The destination selector in an interrupt gate, trap gate, and call gate must refer to a code segment descriptor. These gate descriptors contain the entry point to prevent a program from constructing and using an illegal entry point. Task gates may only refer to a task state segment. Since task gates invoke a task switch, the destination offset is not used in the task gate.

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

The access byte format is the same for all gate descriptors. P = 1 indicates that the gate contents are valid. P = 0 indicates the contents are not valid and causes exception 11 if referenced. DPL is the de-

scriptor privilege level and specifies when this descriptor may be used by a task (refer to privilege discussion below). Bit 4 must equal 0 to indicate a system control descriptor. The type field specifies the descriptor type as indicated in Figure 13.

SEGMENT DESCRIPTOR CACHE REGISTERS

A segment descriptor cache register is assigned to each of the four segment registers (CS, SS, DS, ES). Segment descriptors are automatically loaded (cached) into a segment descriptor cache register (Figure 14) whenever the associated segment register is loaded with a selector. Only segment descriptors may be loaded into segment descriptor cache registers. Once loaded, all references to that segment of memory use the cached descriptor information instead of reaccessing the descriptor. The descriptor cache registers are not visible to programs. No instructions exist to store their contents. They only change when a segment register is loaded.

SELECTOR FIELDS

A protected mode selector has three fields: descriptor entry index, local or global descriptor table indicator (TI), and selector privilege (RPL) as shown in Figure 15. These fields select one of two memory based tables of descriptors, select the appropriate table entry and allow highspeed testing of the selector's privilege attribute (refer to privilege discussion below).

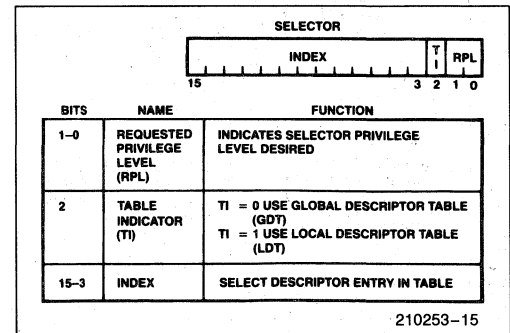


Figure 15. Selector Fields

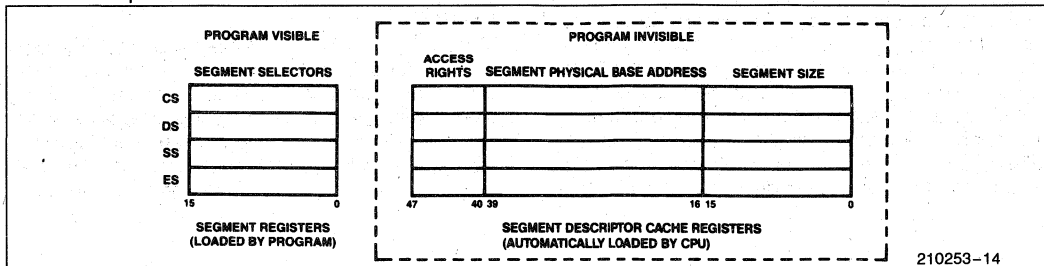


Figure 14. Descriptor Cache Registers

LOCAL AND GLOBAL DESCRIPTOR TABLES

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

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

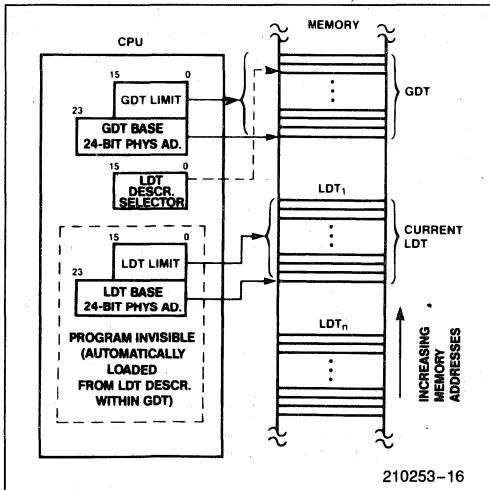


Figure 16. Local and Global Descriptor Table Definition

The LGDT and LLDT instructions load the base and limit of the global and local descriptor tables. LGDT and LLDT are privileged, i.e. they may only be executed by trusted programs operating at level 0. The LGDT instruction loads a six byte field containing the 16-bit table limit and 24-bit physical base address of the Global Descriptor Table as shown in Figure 17. The LLDT instruction loads a selector which refers to a Local Descriptor Table descriptor containing the

base address and limit for an LDT, as shown in Figure 12.

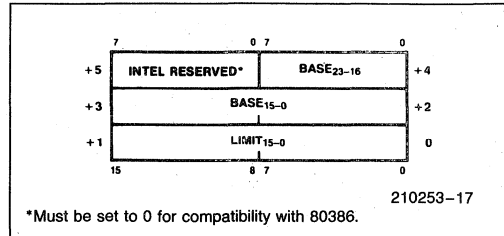


Figure 17. Global Descriptor Table and Interrupt Descriptor Table Data Type

INTERRUPT DESCRIPTOR TABLE

The protected mode 80286 has a third descriptor table, called the Interrupt Descriptor Table (IDT) (see Figure 18), used to define up to 256 interrupts. It may contain only task gates, interrupt gates and trap gates. The IDT (Interrupt Descriptor Table) has a 24-bit physical base and 16-bit limit register in the CPU. The privileged LIDT instruction loads these registers with a six byte value of identical form to that of the LGDT instruction (see Figure 17 and Protected Mode Initialization).

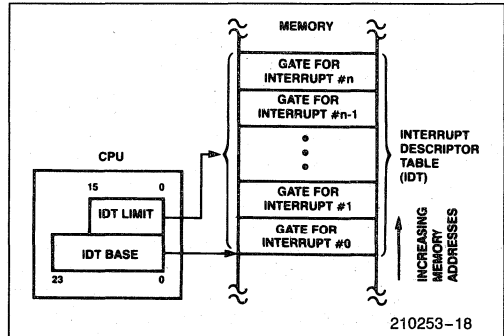
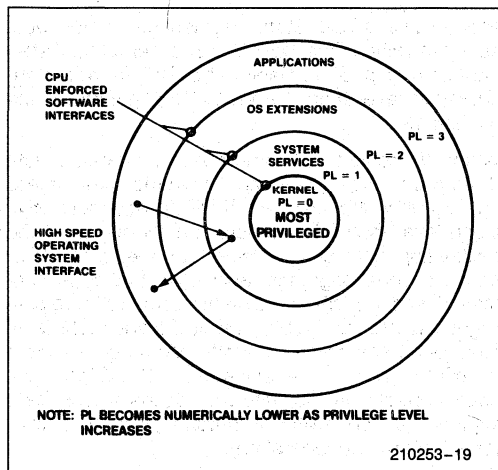


Figure 18. Interrupt Descriptor Table Definition

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

Privilege

The 80286 has a four-level hierarchical privilege system which controls the use of privileged instructions and access to descriptors (and their associated segments) within a task. Four-level privilege, as shown in Figure 19, is an extension of the user/supervisor mode commonly found in minicomputers. The privilege levels are numbered 0 through 3. Level 0 is the



most privileged level. Privilege levels provide protection within a task. (Tasks are isolated by providing private LDT's for each task.) Operating system routines, interrupt handlers, and other system software can be included and protected within the virtual address space of each task using the four levels of privilege. Each task in the system has a separate stack for each of its privilege levels.

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

TASK PRIVILEGE

A task always executes at one of the four privilege levels. The task privilege level at any specific instant is called the Current Privilege Level (CPL) and is defined by the lower two bits of the CS register. CPL cannot change during execution in a single code segment. A task's CPL may only be changed by control transfers through gate descriptors to a new code segment (See Control Transfer). Tasks begin executing at the CPL value specified by the code segment selector within TSS when the task is initiated via a task switch operation (See Figure 20). A task executing at Level 0 can access all data segments defined in the GDT and the task's LDT and is considered the most trusted level. A task executing a Level 3 has the most restricted access to data and is considered the least trusted level.

DESCRIPTOR PRIVILEGE

Descriptor privilege is specified by the Descriptor Privilege Level (DPL) field of the descriptor access byte. DPL specifies the least trusted task privilege level (CPL) at which a task may access the descrip-

tor. Descriptors with DPL = 0 are the most protected. Only tasks executing at privilege level 0 (CPL = 0) may access them. Descriptors with DPL = 3 are the least protected (i.e. have the least restricted access) since tasks can access them when CPL = 0, 1, 2, or 3. This rule applies to all descriptors, except LDT descriptors.

SELECTOR PRIVILEGE

Selector privilege is specified by the Requested Privilege Level (RPL) field in the least significant two bits of a selector. Selector RPL may establish a less trusted privilege level than the current privilege level for the use of a selector. This level is called the task's effective privilege level (EPL). RPL can only reduce the scope of a task's access to data with this selector. A task's effective privilege is the numeric maximum of RPL and CPL. A selector with RPL = 0 imposes no additional restriction on its use while a selector with RPL = 3 can only refer to segments at privilege Level 3 regardless of the task's CPL. RPL is generally used to verify that pointer parameters passed to a more trusted procedure are not allowed to use data at a more privileged level than the caller (refer to pointer testing instructions).

Descriptor Access and Privilege Validation

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

DATA SEGMENT ACCESS

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

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

If the privilege checks fail (e.g. DPL is numerically less than the maximum of CPL and RPL) or an incorrect type of descriptor is referenced (e.g. gate de-

scriptor or execute only code segment) exception 13 occurs. If the segment is not present, exception 11 is generated.

Instructions that load selectors into SS must refer to data segment descriptors for writable data segments. The descriptor privilege (DPL) and RPL must equal CPL. All other descriptor types or a privilege level violation will cause exception 13. A not present fault causes exception 12.

CONTROL TRANSFER

Four types of control transfer can occur when a selector is loaded into CS by a control transfer operation (see Table 10). Each transfer type can only occur if the operation which loaded the selector references the correct descriptor type. Any violation of these descriptor usage rules (e.g. JMP through a call gate or RET to a Task State Segment) will cause exception 13.

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

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

When a JMP or CALL references a Task State Segment descriptor, the descriptor DPL must be the same or less privileged than the task's CPL. Refer-

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

The privilege rules on control transfer require:

- JMP or CALL direct to a code segment (code segment descriptor) can only be to a conforming segment with DPL of equal or greater privilege than CPL or a non-conforming segment at the same privilege level.
- interrupts within the task or calls that may change privilege levels, can only transfer control through a gate at the same or a less privileged level than CPL to a code segment at the same or more privileged level than CPL.
- return instructions that don't switch tasks can only return control to a code segment at the same or less privileged level.
- task switch can be performed by a call, jump or interrupt which references either a task gate or task state segment at the same or less privileged level.

Table 10. Descriptor Types Used for Control Transfer

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

PRIVILEGE LEVEL CHANGES

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

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

Protection

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

Restricted *usage* of segments (e.g. no write allowed to read-only data segments). The only segments available for use are defined by descriptors in the Local Descriptor Table (LDT) and Global Descriptor Table (GDT).

Restricted *access* to segments via the rules of privilege and descriptor usage.

Privileged instructions or operations that may only be executed at certain privilege levels as determined by the CPL and I/O Privilege Level (IOPL). The IOPL is defined by bits 14 and 13 of the flag word.

These checks are performed for all instructions and can be split into three categories: segment load checks (Table 11), operand reference checks (Table 12), and privileged instruction checks (Table 13). Any violation of the rules shown will result in an exception. A not-present exception related to the stack segment causes exception 12.

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

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

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

Table 11
Segment Register Load Checks

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

Table 12. Operand Reference Checks

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

NOTE:

Carry out in offset calculations is ignored.

Table 13. Privileged Instruction Checks

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

EXCEPTIONS

The 80286 detects several types of exceptions and interrupts, in protected mode (see Table 14). Most are restartable after the exceptional condition is removed. Interrupt handlers for most exceptions can read an error code, pushed on the stack after the return address, that identifies the selector involved (0 if none). The return address normally points to the failing instruction, including all leading prefixes. For a processor extension segment overrun exception, the return address will not point at the ESC instruction that caused the exception; however, the processor extension registers may contain the address of the failing instruction.

Table 14. Protected Mode Exceptions

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

NOTE:

1. When a PUSH or POP instruction attempts to wrap around the stack segment, the machine state after the exception will not be restartable because stack segment wrap around is not permitted. This condition is identified by the value of the saved SP being either 0000(H), 0001(H), FFFE(H), or FFFF(H).

2. These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.

These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.

All these checks are performed for all instructions and can be split into three categories: segment load checks (Table 11), operand reference checks (Table 12), and privileged instruction checks (Table 13). Any violation of the rules shown will result in an exception. A not-present exception causes exception 11 or 12 and is restartable.

Special Operations

TASK SWITCH OPERATION

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

The TSS descriptor points at a segment (see Figure 20) containing the entire 80286 execution state while a task gate descriptor contains a TSS selector. The limit field of the descriptor must be > 002B(H).

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

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

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

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

PROCESSOR EXTENSION CONTEXT SWITCHING

The context of a processor extension (such as the 80287 numerics processor) is not changed by the task switch operation. A processor extension context need only be changed when a different task attempts to use the processor extension (which still contains the context of a previous task). The 80286 detects the first use of a processor extension after a task switch by causing the processor extension not present exception (7). The interrupt handler may then decide whether a context change is necessary.

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

POINTER TESTING INSTRUCTIONS

The 80286 provides several instructions to speed pointer testing and consistency checks for maintaining system integrity (see Table 15). These instruc-

tions use the memory management hardware to verify that a selector value refers to an appropriate segment without risking an exception. A condition flag (ZF) indicates whether use of the selector or segment will cause an exception.

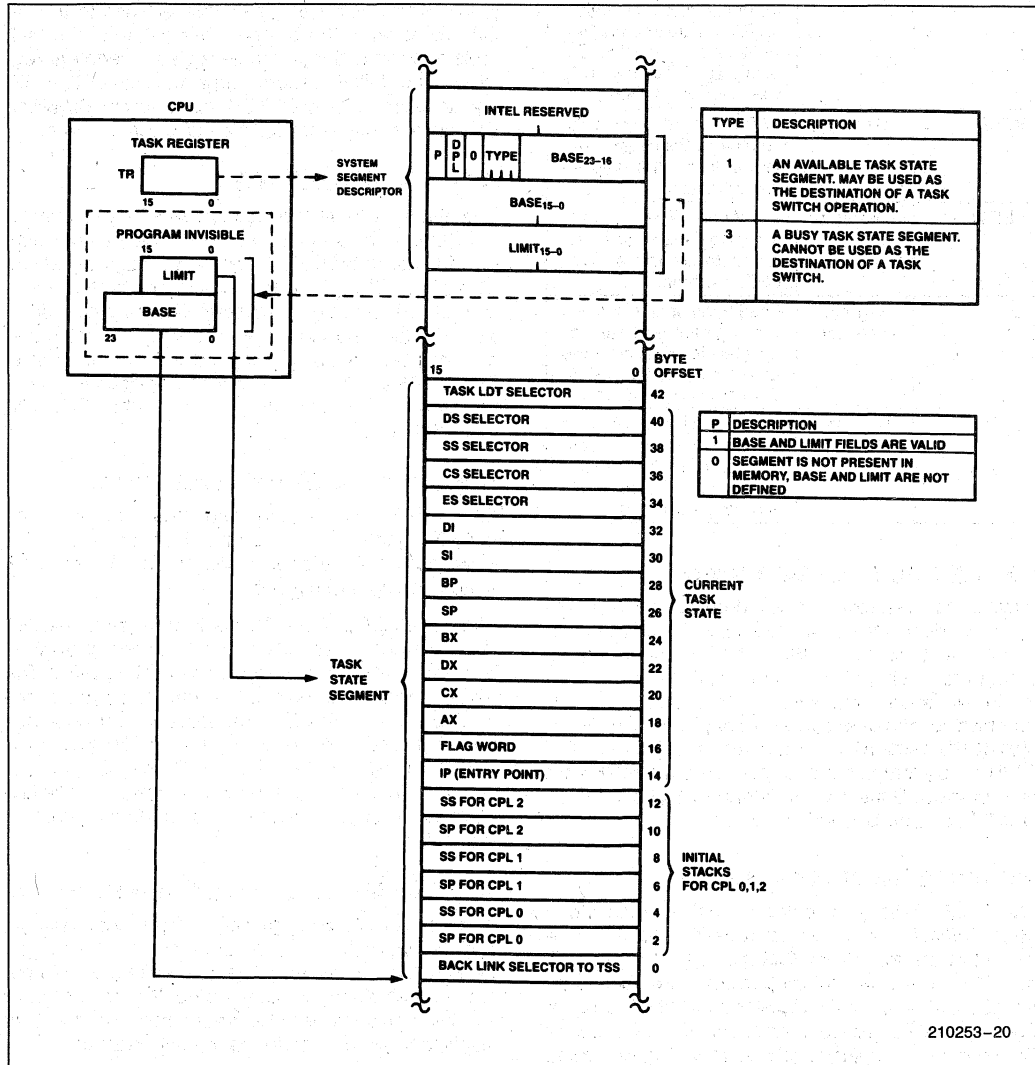


Figure 20. Task State Segment and TSS Registers

Table 15. 80286 Pointer Test Instructions

Instruction	Operands	Function
ARPL	Selector, Register	Adjust Requested Privilege Level: adjusts the RPL of the selector to the numeric maximum of current selector RPL value and the RPL value in the register. Set zero flag if selector RPL was changed by ARPL.
VERR	Selector	VERify for Read: sets the zero flag if the segment referred to by the selector can be read.
VERW	Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL	Register, Selector	Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

DOUBLE FAULT AND SHUTDOWN

If two separate exceptions are detected during a single instruction execution, the 80286 performs the double fault exception (8). If an execution occurs during processing of the double fault exception, the 80286 will enter shutdown. During shutdown no further instructions or exceptions are processed. Either NMI (CPU remains in protected mode) or RESET (CPU exits protected mode) can force the 80286 out of shutdown. Shutdown is externally signalled via a HALT bus operation with A₁ LOW.

PROTECTED MODE INITIALIZATION

The 80286 initially executes in real address mode after RESET. To allow initialization code to be placed at the top of physical memory, A₂₃–A₂₀ will be HIGH when the 80286 performs memory references relative to the CS register until CS is changed. A₂₃–A₂₀ will be zero for references to the DS, ES, or SS segments. Changing CS in real address mode will force A₂₃–A₂₀ LOW whenever CS is used again. The initial CS:IP value of F000:FFF0 provides 64K bytes of code space for initialization code without changing CS.

Protected mode operation requires several registers to be initialized. The GDT and IDT base registers must refer to a valid GDT and IDT. After executing the LMSW instruction to set PE, the 80286 must im-

mediately execute an intra-segment JMP instruction to clear the instruction queue of instructions decoded in real address mode.

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

SYSTEM INTERFACE

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

Bus Interface Signals and Timing

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

The 80286 CPU, 82C284 clock generator, 82C288 bus controller, trancivers, and latches provide a buffered and decoded system bus interface. The 82C284 generates the system clock and synchronizes 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 D₇–₀ while odd bytes are transferred over D₁₅–₈. Even-addressed words are transferred over D₁₅–₀ in one bus cycle, while odd-addressed word require *two* bus operations. The first transfers data on D₁₅–₈, and the second transfers data on D₇–₀. Both byte data transfers occur automatically, transparent to software.

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

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

Bus Operation

The 80286 uses a double frequency system clock (CLK input) to control bus timing. All signals on the local bus are measured relative to the system CLK input. The CPU divides the system clock by 2 to produce the internal processor clock, which determines bus state. Each processor clock is composed of two system clock cycles named phase 1 and phase 2. The 82C284 clock generator output (PCLK) identifies the next phase of the processor clock. (See Figure 21.)

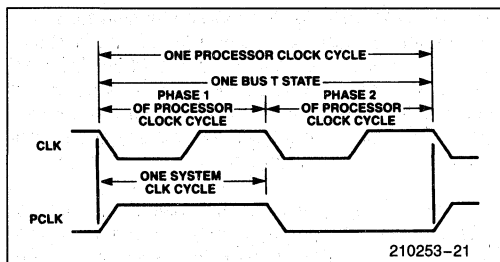


Figure 21. System and Processor Clock Relationships

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

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

Each bus state is one processor clock long. Figure 22 shows the four 80286 local bus states and allowed transitions.

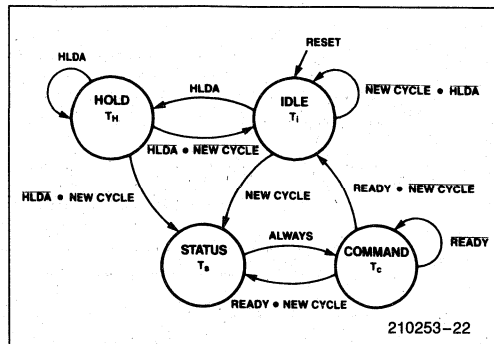


Figure 22. 80286 Bus States

Bus States

The idle (T_i) state indicates that no data transfers are in progress or requested. The first active state T_s is signaled by status line $S\overline{1}$ or $S\overline{0}$ going LOW and identifying phase 1 of the processor clock. During T_s , the command encoding, the address, and data (for a write operation) are available on the 80286 output pins. The 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. Pipelined timing allows a new bus operation to be initiated every two processor cycles, while allowing each individual bus operation to last for three processor cycles.

The timing of the address outputs is pipelined such that the address of the next bus operation becomes available during the current bus operation. Or in other words, the first clock of the next bus operation is overlapped with the last clock of the current bus operation. Therefore, address decode and routing logic can operate in advance of the next bus operation.

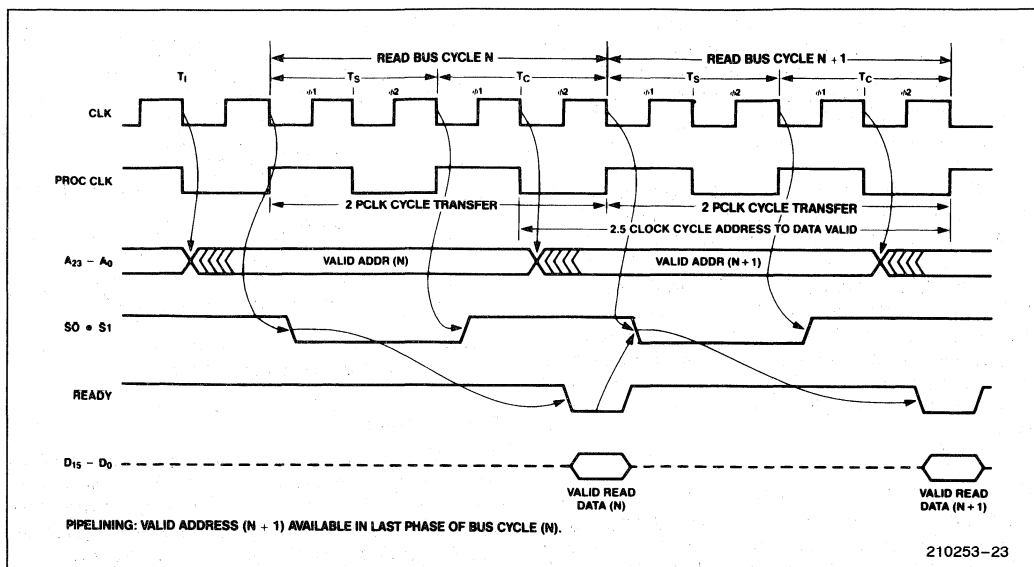


Figure 23. Basic Bus Cycle

External address latches may hold the address stable for the entire bus operation, and provide additional AC and DC buffering.

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

Bus Control Signals

The 82C288 bus controller provides control signals; address latch enable (ALE), Read/Write commands, data transmit/receive (DT/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/R). DEN enables the data transceivers; while DT/R controls transceiver direction. DEN and DT/R are timed to prevent bus contention between the bus master, data bus transceivers, and system data bus transceivers.

Command Timing Controls

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

Command extension allows additional time for external devices to respond to a command and is analogous to inserting wait states on the 8086. External logic can control the duration of any bus operation such that the operation is only as long as necessary. The READY input signal can extend any bus operation for as long as necessary.

Command delay allows an increase of address or write data setup time to system bus command active for any bus operation by delaying when the system bus command becomes active. Command delay is controlled by the 82C288 CMDLY input. After T_S , the bus controller samples CMDLY at each falling 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/R.

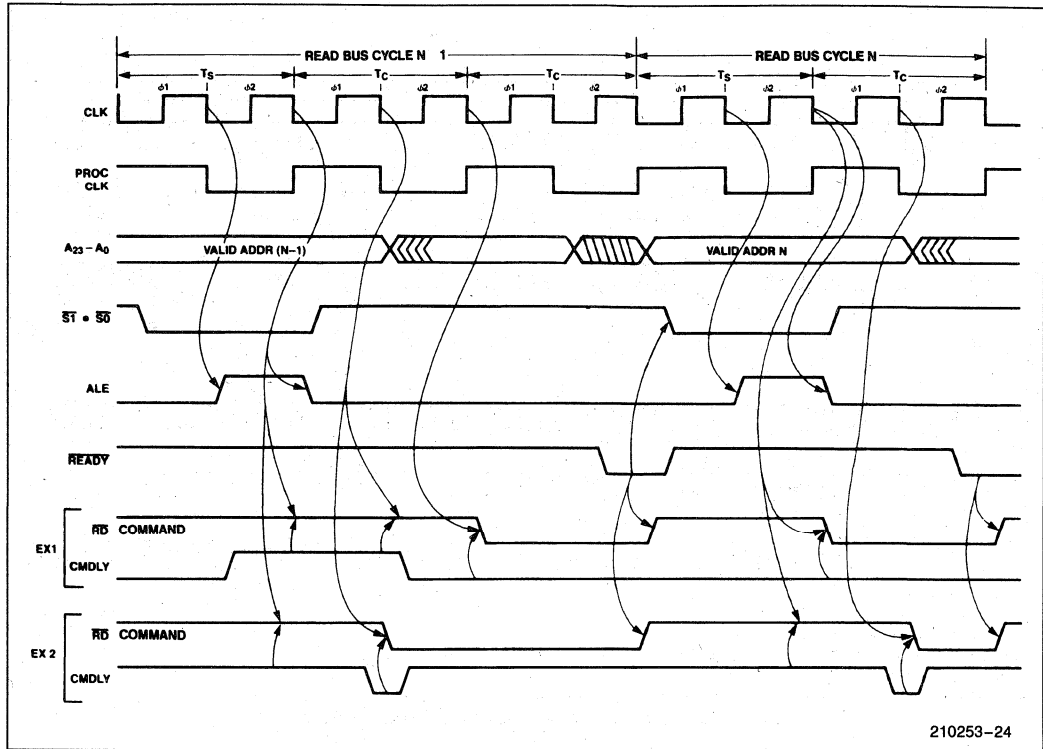


Figure 24. CMDLY Controls the Leading Edge of Command Signal

Figure 24 illustrates four uses of CMDLY. Example 1 shows delaying the read command two system CLKs for cycle N-1 and no delay for cycle N, and example 2 shows delaying the read command one system CLK for cycle N-1 and one system CLK delay for cycle N.

Bus Cycle Termination

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

READY Operation

The current bus master and 82C288 bus controller terminate each bus operation simultaneously to achieve maximum bus operation bandwidth. Both are informed in advance by **READY** active (open-collector output from 82C284) which identifies the last T_c cycle of the current bus operation. The bus master and bus controller must see the same sense

of the **READY** signal, thereby requiring **READY** be synchronous to the system clock.

Synchronous Ready

The 82C284 clock generator provides **READY** synchronization from both synchronous and asynchronous sources (see Figure 25). 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 T_c . 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 82C284 **SRDY** setup and hold time requirements. But the 82C284 asynchronous ready input (**ARDY**) is designed to accept such signals. The **ARDY** input is sampled at the beginning of each T_c cycle by 82C284 synchronization logic. This provides one system CLK cycle time to resolve its value before broadcasting it to the bus master and bus controller.

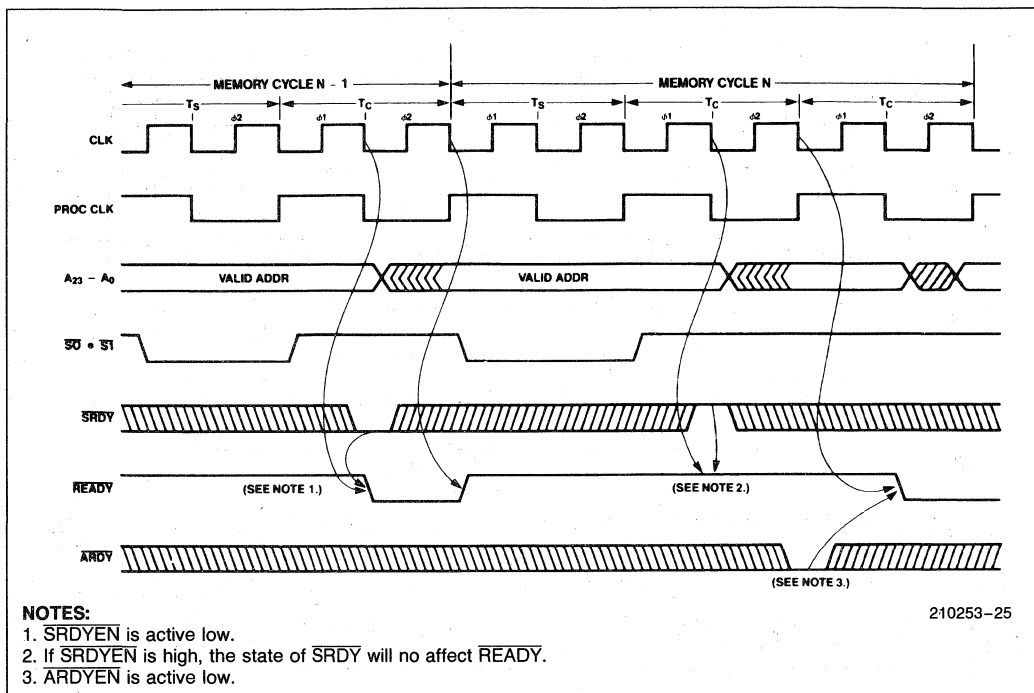


Figure 25. Synchronous and Asynchronous Ready

\overline{ARDY} or \overline{ARDYEN} must be HIGH at the end of T_S . \overline{ARDY} cannot be used to terminate bus cycle with no wait states.

Each ready input of the 82C284 has an enable pin (\overline{SRDYEN} and \overline{ARDYEN}) to select whether the current bus operation will be terminated by the synchronous or asynchronous ready. Either of the ready inputs may terminate a bus operation. These enable inputs are active low and have the same timing as their respective ready inputs. Address decode logic usually selects whether the current bus operation should be terminated by \overline{ARDY} or \overline{SRDY} .

Data Bus Control

Figures 26, 27, and 28 show how the $\overline{DT/\overline{R}}$, \overline{DEN} , data bus, and address signals operate for different combinations of read, write, and idle bus operations. $\overline{DT/\overline{R}}$ goes active (LOW) for a read operation. $\overline{DT/\overline{R}}$ 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 3-state OFF before the 80286 CPU begins driving the local data bus for write operations. Write data will always remain valid for one system clock past the last T_C to provide sufficient hold time for Multibus or other similar memory or I/O systems. During write-read or write-idle sequences the data bus enters 3-state OFF during the second phase of the processor cycle after the last T_C . In a write-write sequence the data bus does not enter 3-state OFF between T_C and T_S .

Bus Usage

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

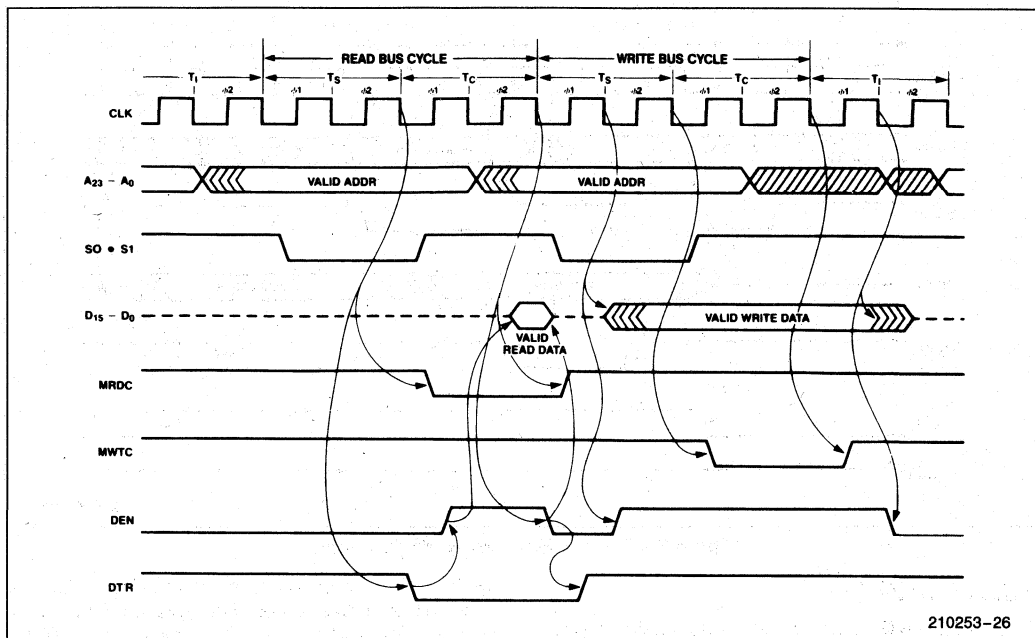


Figure 26. Back to Back Read-Write Cycles

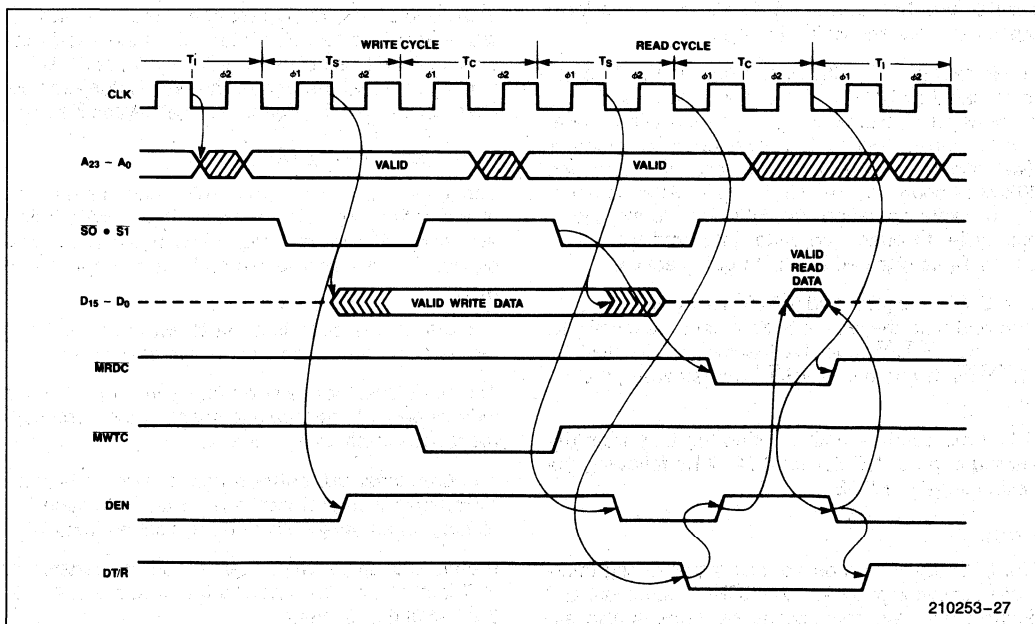


Figure 27. Back to Back Write-Read Cycles

3

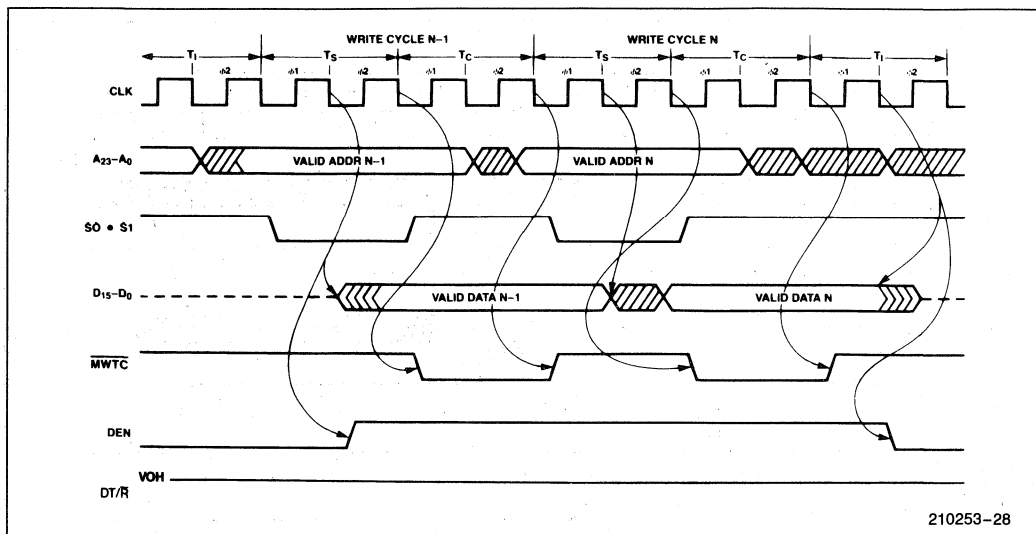


Figure 28. Back to Back Write-Write Cycles

HOLD and HLDA

HOLD AND HLDA allow another bus master to gain control of the local bus by placing the 80286 bus into the T_h state. The sequence of events required to pass control between the 80286 and another local bus master are shown in Figure 29.

In this example, the 80286 is initially in the T_h state as signaled by HLDA being active. Upon leaving T_h , as signaled by HLDA going inactive, a write operation is started. During the write operation another local bus master requests the local bus from the 80286 as shown by the HOLD signal. After completing the write operation, the 80286 performs one T_1 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.

Lock

The CPU asserts an active lock signal during Interrupt-Acknowledge cycles, the XCHG instruction, and during some descriptor accesses. Lock is also asserted when the LOCK prefix is used. The LOCK prefix may be used with the following ASM-286 assembly instructions; MOVS, INS, and OUTS. For bus

cycles other than Interrupt-Acknowledge cycles, Lock will be active for the first and subsequent cycles of a series of cycles to be locked. Lock will not be shown active during the last cycle to be locked. For the next-to-last cycle, Lock will become inactive at the end of the first T_C regardless of the number of wait-states inserted. For Interrupt-Acknowledge cycles, Lock will be active for each cycle, and will become inactive at the end of the first T_C for each cycle regardless of the number of wait-states inserted.

Instruction Fetching

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

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

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

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

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

In real address mode, the prefetcher may fetch up to 6 bytes beyond the last control transfer or HLT instruction in a code segment.

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

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

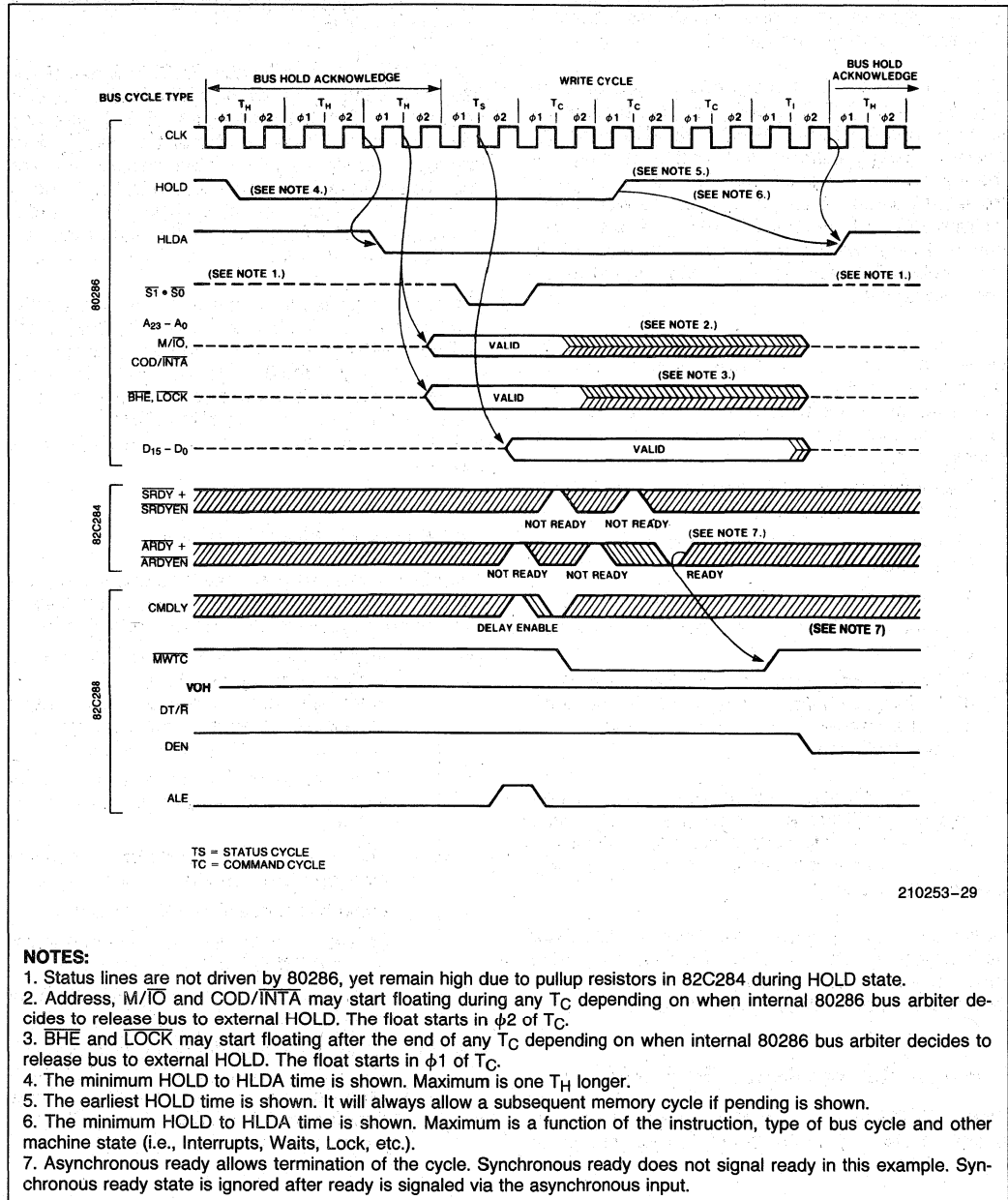


Figure 29. MULTIBUS® Write Terminated by Asynchronous Ready with Bus Hold

Processor Extension Transfers

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

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

NOTE:

Odd-aligned numerics operands should be avoided when using an 80286 system running six or more memory-write wait states. The 80286 can generate an incorrect numerics address if all the following conditions are met:

- Two floating point (FP) instructions are fetched and in the 80286 queue.
- The first FP instruction is any floating point store except FSTSW AX.
- The second FP instruction accesses memory.
- The operand of the first instruction is aligned on an odd memory address.
- Six or more wait states are inserted during either of the last two memory write (odd aligned operands are transferred as two bytes) transfers of the first instruction.

The second FP operand's address will be incremented by one if these conditions are met. These conditions are most likely to occur in a multi-master system. For a hardware solution, contact your local Intel representative.

Commands to the numerics coprocessor should not be delayed by nine or more T-states. Excessive (nine or more) command-delays can cause the 80286 and 80287 to lose synchronization.

Interrupt Acknowledge Sequence

Figure 30 illustrates an interrupt acknowledge sequence performed by the 80286 in response to an

INTR input. An interrupt acknowledge sequence consists of two INTA bus operations. The first allows a master 8259A Programmable Interrupt Controller (PIC) to determine which if any of its slaves should return the interrupt vector. An eight bit vector is read on D0-D7 of the 80286 during the second INTA bus operation to select an interrupt handler routine from the interrupt table.

The Master Cascade Enable (MCE) signal of the 82C288 is used to enable the cascade address drivers, during INTA bus operations (See Figure 30), onto the local address bus for distribution to slave interrupt controllers via the system address bus. The 80286 emits the LOCK signal (active LOW) during T_3 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 READY. This is needed to meet the 8259A minimum INTA pulse width.

Local Bus Usage Priorities

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

(Highest) Any transfers which assert LOCK either explicitly (via the LOCK instruction prefix) or implicitly (i.e. some segment descriptor accesses, interrupt acknowledge sequence, or an XCHG with memory).

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

The second or third cycle of a processor extension data transfer.

Local bus request via HOLD input.

Processor extension data operand transfer via PEREQ input.

Data transfer performed by EU as part of an instruction.

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

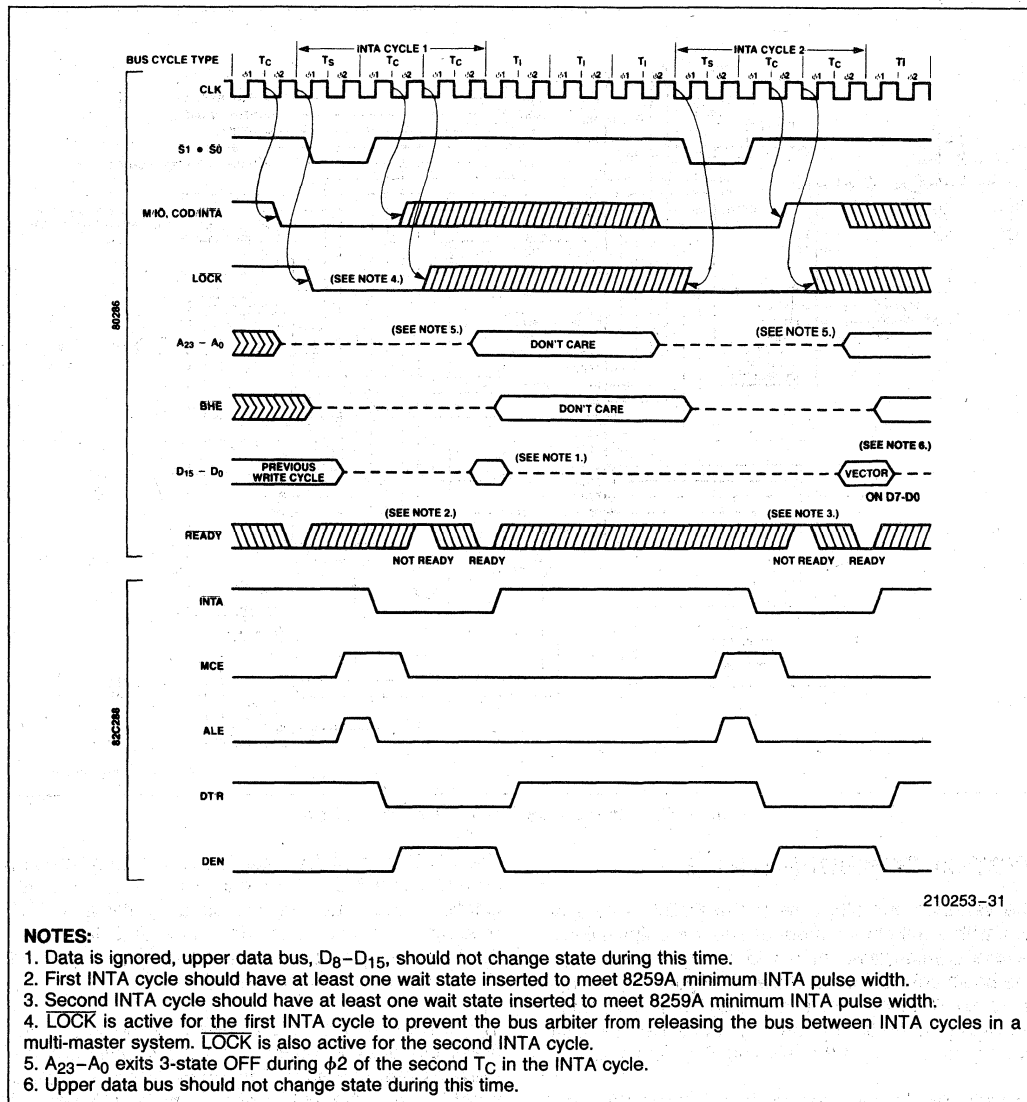


Figure 30. Interrupt Acknowledge Sequence

Halt or Shutdown Cycles

The 80286 externally indicates halt or shutdown conditions as a bus operation. These conditions occur due to a HLT instruction or multiple protection exceptions while attempting to execute one instruction. A halt or shutdown bus operation is signalled when $\overline{S1}$, $\overline{S0}$ and $\overline{COD/INTA}$ are LOW and $\overline{M/IO}$ is HIGH. A_1 HIGH indicates halt, and A_1 LOW indicates shutdown. The 82C288 bus controller does

not issue ALE, nor is \overline{READY} required to terminate a halt or shutdown bus operation.

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

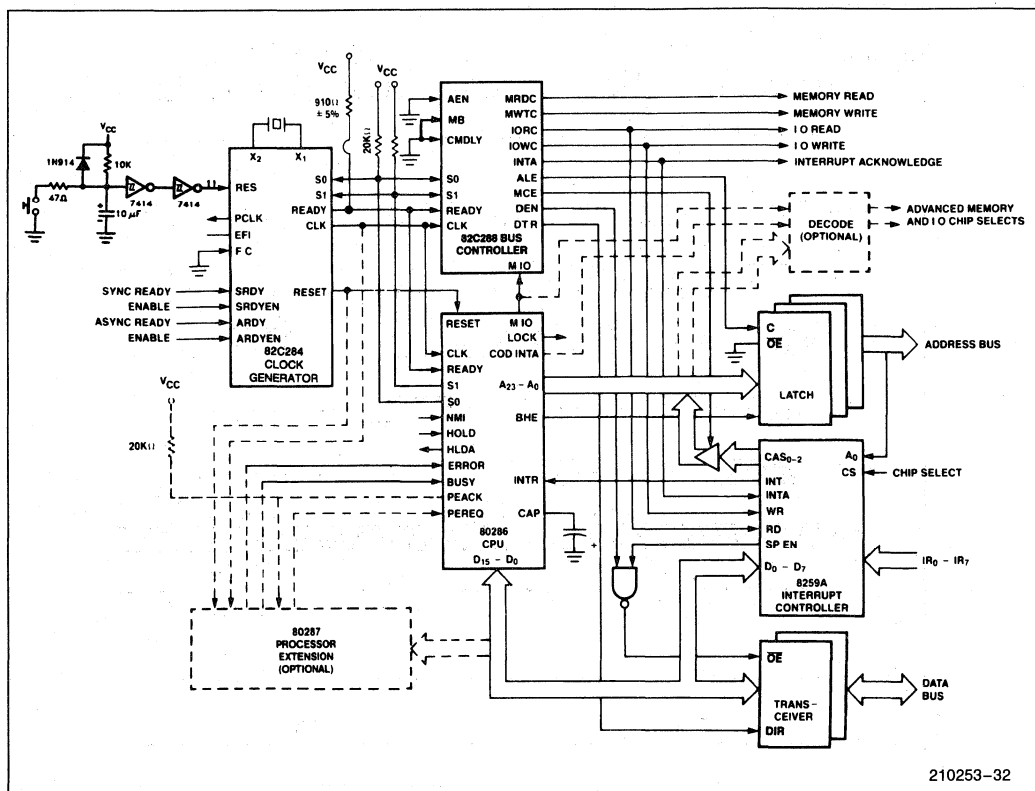


Figure 31. Basic 80286 System Configuration

SYSTEM CONFIGURATIONS

The versatile bus structure of the 80286 microsystem, with a full complement of support chips, allows flexible configuration of a wide range of systems. The basic configuration, shown in Figure 31, is similar to an 8086 maximum mode system. It includes the CPU plus an 8259A interrupt controller, 82C284 clock generator, and the 82C288 Bus Controller.

As indicated by the dashed lines in Figure 31, the ability to add processor extensions is an integral feature of 80286 microsystems. The processor extension interface allows external hardware to perform special functions and transfer data concurrent with CPU execution of other instructions. Full system integrity is maintained because the 80286 supervises all data transfers and instruction execution for the processor extension.

The 80287 has all the instructions and data types of an 8087. The 80287 NPX can perform numeric calculations and data transfers concurrently with CPU program execution. Numerics code and data have the same integrity as all other information protected by the 80286 protection mechanism.

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

The optional decode logic shown in Figure 31 takes advantage of the overlap between address and data of the 80286 bus cycle to generate advanced memory and I/O-select signals. This minimizes system

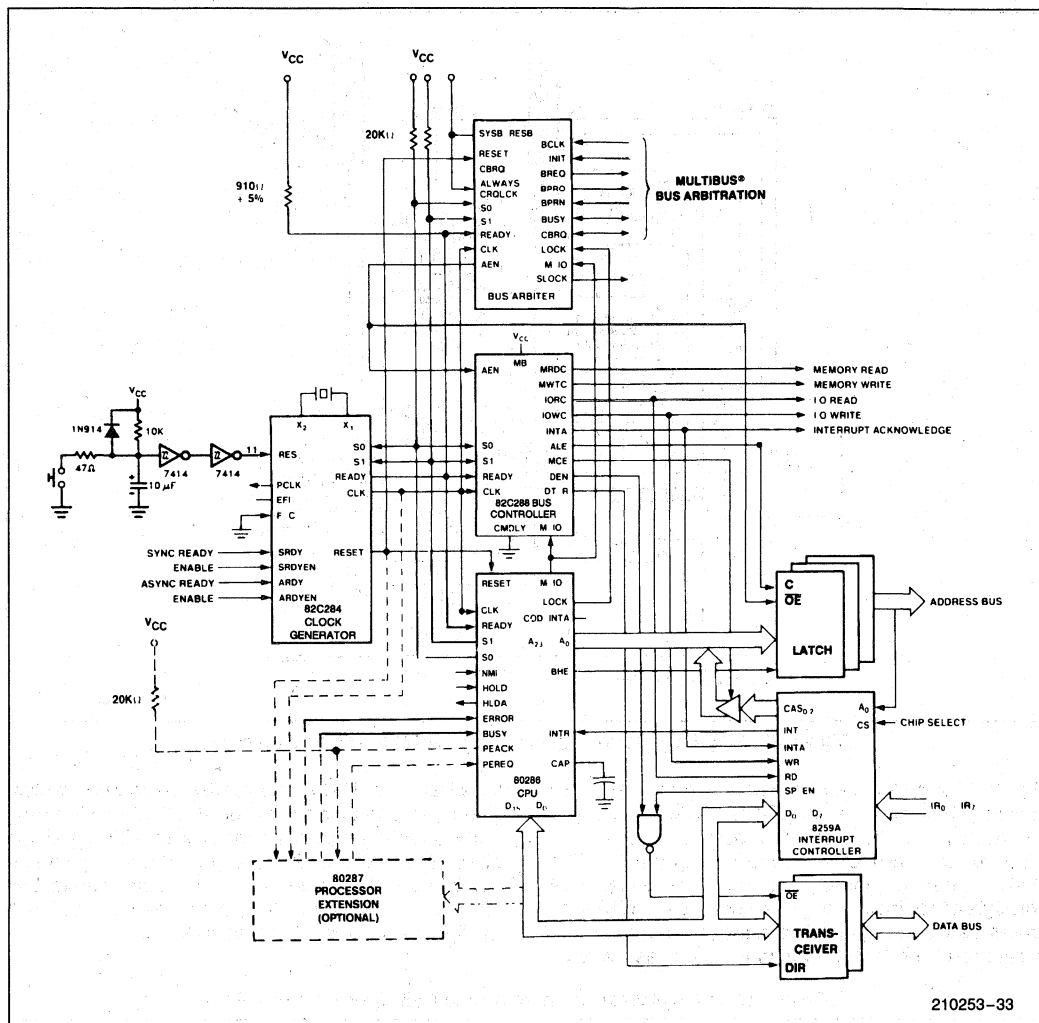


Figure 32. MULTIBUS® System Bus Interface

performance degradation caused by address propagation and decode delays. In addition to selecting memory and I/O, the advanced selects may be used with configurations supporting local and system buses to enable the appropriate bus interface for each bus cycle. The COD/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 32. 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 setup times. This arrangement will add at least one extra T_c 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 32. This configuration allows the 80286 to support an on-board bus for local memory and peripherals, and the MULTIBUS for system bus interfacing.

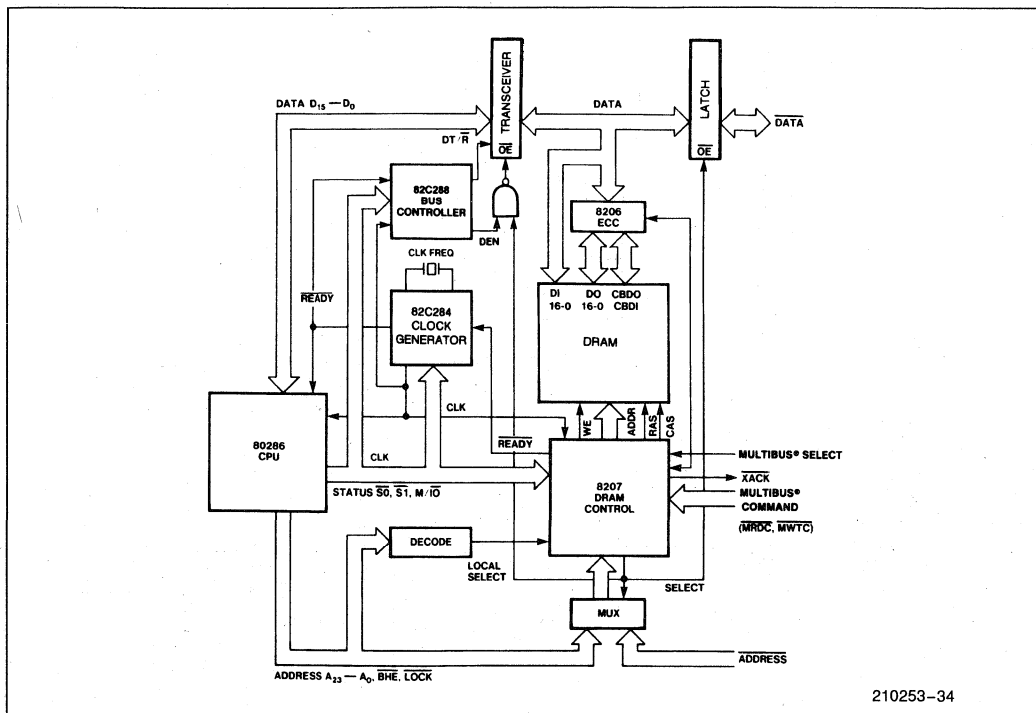


Figure 33. 80286 System Configuration with Dual-Ported Memory

Figure 33 shows the addition of dual ported dynamic memory between the MULTIBUS system bus and the 80286 local bus. The dual port interface is provided by the 8207 Dual Port DRAM Controller. The 8207 runs synchronously with the CPU to maximize throughput for local memory references. It also arbitrates between requests from the local and system buses and performs functions such as refresh,

initialization of RAM, and read/modify/write cycles. The 8207 combined with the 8206 Error Checking and Correction memory controller provide for single bit error correction. The dual-port memory can be combined with a standard MULTIBUS system bus interface to maximize performance and protection in multiprocessor system configurations.

Table 16. 80286 Systems Recommended Pull Up Resistor Values

80286 Pin and Name	Pullup Value	Purpose
4— $\overline{S1}$	20 K Ω \pm 10%	Pull $\overline{S0}$, $\overline{S1}$, and \overline{PEACK} inactive during 80286 hold periods ⁽¹⁾
5— $\overline{S0}$		
6— \overline{PEACK}		
63— \overline{READY}	910 Ω \pm 5%	Pull \overline{READY} inactive within required minimum time ($C_L = 150$ pF, $I_R \leq 7$ mA)

NOTE:

1. Pull-up resistors are not required on $\overline{S0}$ and $\overline{S1}$ when the corresponding pins of the 82C284 are connected to $\overline{S0}$ and $\overline{S1}$.

I²C[™]-286 System Design Considerations

One of the advantages of using the 80286 is that full in-circuit emulation debugging support is provided through the I²C system 80286 probe. To utilize this powerful tool it is necessary that the system designer be aware of a few minor parametric and

functional differences between the 80286 and I²C system 80286 probe. The I²C data sheet (I²C Integrated Instrumentation and In-Circuit Emulation System, order #210469) contains a detailed description of these design considerations. It is recommended that this document be reviewed by the 80286 system designer to determine whether or not these differences affect his design.

PACKAGE THERMAL SPECIFICATIONS

The 80286 Microprocessor is specified for operation when case temperature (T_C) is within the range 0°C–85°C. Case temperature, unlike ambient temperature, is easily measured in any environment to determine whether the 80286 Microprocessor is within the specified operating range. The case temperature should be measured at the center of the top surface of the component.

The maximum ambient temperature (T_A) allowable without violating T_C specifications can be calculated from the equations shown below. T_J is the 80286 junction temperature. P is the power dissipated by the 80286.

$$T_J = T_C + P \cdot \theta_{JC}$$

$$T_A = T_J - P \cdot \theta_{JA}$$

$$T_C = T_A + P \cdot [\theta_{JA} - \theta_{JC}]$$

Values for θ_{JA} and θ_{JC} are given in Table 17. θ_{JA} is given at various airflows. Table 18 shows the maximum T_A allowable (without exceeding T_C) at various airflows. Note that the 80286 PLCC package has an internal heat spreader. T_A can be further improved by attaching "fins" or an external "heat sink" to the package.

Junction temperature calculations should use an I_{CC} value that is measured without external resistive loads. The external resistive loads dissipate additional power external to the 80286 and not on the die. This increases the resistor temperature, not the die temperature. The full capacitive load ($C_L = 100$ pF) should be applied during the I_{CC} measurement.

Table 17. Thermal Resistances (°C/Watt) θ_{JC} and θ_{JA}

Package	θ_{JC}	θ_{JA} versus Airflow — ft/min (m/sec)					
		0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
68-Lead LCC	8	28	22	16	13	12	11
68-Lead PGA	5.5	28	22	16	15	14	13
68-Lead PLCC w/ Internal Heat Spreader	8	28	23	21	18	16	15

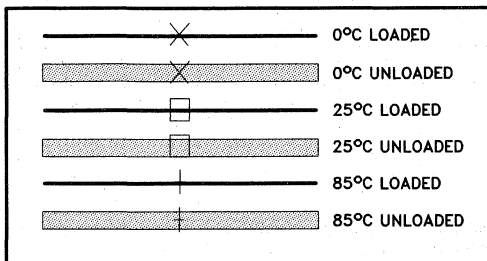
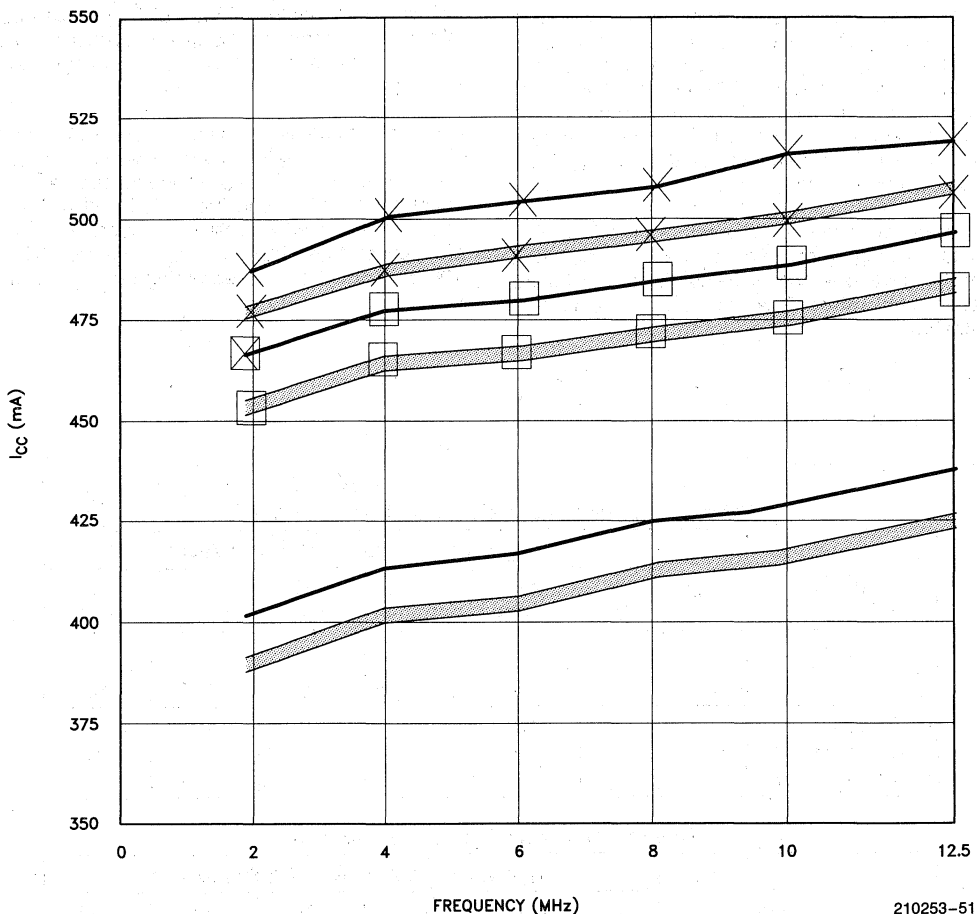
Table 18. Maximum T_A at Various Airflows

Package	T_A (°C) versus Airflow — ft/min (m/sec)					
	0 (0)	200 (1.01)	400 (2.03)	600 (3.04)	800 (4.06)	1000 (5.07)
68-Lead LCC	40	54	67	74	76	78
68-Lead PGA	34	48	61	64	66	68
68-Lead PLCC w/Internal Heat Spreader	40	51	56	63	67	69

NOTE:

The numbers in Table 18 were calculated using a V_{CC} of 5.0V, and an I_{CC} of 450 mA, which is representative of the worst case I_{CC} at $T_C = 85^\circ\text{C}$ with the outputs unloaded.

Typical I_{CC} vs Frequency for Different Output Loads and Case Temperatures



NOTES:

1. $V_{CC} = 5.0V$
2. Loaded: $I_{OL} = 2.0 mA$, $I_{OH} = -400 \mu A$, $C_L = 100 pF$.
Unloaded: $C_L = 100 pF$.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to + 70°C
 Storage Temperature - 65°C to + 150°C
 Voltage on Any Pin with
 Respect to Ground - 1.0V to + 7V
 Power Dissipation 3.3W

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

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

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

Symbol	Parameter	Min	Max	Unit	Test Condition
I _{CC}	Supply Current (0°C Turn On)		600	mA	(Note 1)
C _{CLK}	CLK Input Capacitance		20	pF	(Note 2)
C _{IN}	Other Input Capacitance		10	pF	(Note 2)
C _O	Input/Output Capacitance		20	pF	(Note 2)

NOTES:

1. C_L = 100 pF. Tested at maximum frequency without resistive loads on the outputs.
2. These are not tested. They are guaranteed by design characterization.

D.C. CHARACTERISTICS

(V_{CC} = 5V ± 5%, T_{CASE} = 0°C to + 85°C)* Tested at the minimum operating frequency of the part.

Symbol	Parameter	Min	Max	Unit	Test Condition
V _{IL}	Input LOW Voltage	-0.5	0.8	V	
V _{IH}	Input HIGH Voltage	2.0	V _{CC} + 0.5	V	
V _{ILC}	CLK Input LOW Voltage	-0.5	0.6	V	
V _{IHC}	CLK Input HIGH Voltage	3.8	V _{CC} + 0.5	V	
V _{OL}	Output LOW Voltage		0.45	V	I _{OL} = 2.0 mA
V _{OH}	Output HIGH Voltage	2.4		V	I _{OH} = -400 μA
I _{LI}	Input Leakage Current		± 10	μA	0V ≤ V _{IN} ≤ V _{CC}
I _{LCR}	Input CLK, RESET Leakage Current		± 10	μA	0.45V ≤ V _{IN} ≤ V _{CC}
I _{LCR}	Input CLK, RESET Leakage Current		± 1	mA	0V ≤ V _{IN} ≤ 0.45V
I _{IL}	Input Sustaining Current on <u>BUSY</u> and <u>ERROR</u> Pins	-30	-500	μA	V _{IN} = 0V
I _{LO}	Output Leakage Current		± 10	μA	0.45V ≤ V _{OUT} ≤ V _{CC} 25°C ≤ T _{CASE} ≤ 85°C
I _{LO}	Output Leakage Current		± 20	μA	0.45V ≤ V _{OUT} ≤ V _{CC} 0°C ≤ T _{CASE} ≤ 25°C
I _{LO}	Output Leakage Current		± 1	mA	0V ≤ V _{OUT} ≤ 0.45V

NOTE:

*T_A is guaranteed from 0°C to + 55°C as long as T_{CASE} is not exceeded.

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

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

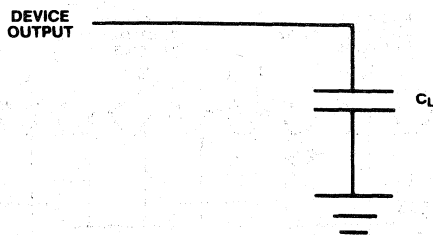
Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Max	-12 Min	-12 Max		
1	System Clock (CLK) Period	62	250	50	250	40	250	ns	
2	System Clock (CLK) LOW Time	15		12		11		ns	at 1.0V
3	System Clock (CLK) HIGH Time	25		16		13		ns	at 3.6V
17	System Clock (CLK) Rise Time		10		8	—	8	ns	1.0V to 3.6V, (Note 7)
18	System Clock (CLK) Fall Time		10		8	—	8	ns	3.6V to 1.0V, (Note 7)
4	Asynch. Inputs Setup Time	20		20		15		ns	(Note 1)
5	Asynch. Inputs Hold Time	20		20		15		ns	(Note 1)
6	RESET Setup Time	28		23		18		ns	
7	RESET Hold Time	5		5		5		ns	
8	Read Data Setup Time	10		8		5		ns	
9	Read Data Hold Time	8		8		6		ns	
10	READY Setup Time	38		26		22		ns	
11	READY Hold Time	25		25		20		ns	
12	Status/PEACK Valid Delay	1	40	—	—	—	—	ns	(Notes 2, 3, 8)
12a1	Status Active Delay	—	—	1	22	3	18	ns	(Notes 2, 3, 8)
12a2	PEACK Active Delay	—	—	1	22	3	20	ns	(Notes 2, 3, 8)
12b	Status/PEACK Inactive Delay	—	—	1	30	3	22	ns	(Notes 2, 3, 8)
13	Address Valid Delay	1	60	1	35	1	32	ns	(Notes 2, 3, 8)
14	Write Data Valid Delay	0	50	0	30	0	30	ns	(Notes 2, 3, 8)
15	Address/Status/Data Float Delay	0	50	0	47	0	32	ns	(Notes 2, 4, 7)
16	HLDA Valid Delay	0	50	0	47	0	27	ns	(Notes 2, 3, 8)
19	Address Valid To Status Valid Setup Time	38		27		22		ns	(Notes 3, 5, 6, 7)

*T_A is guaranteed from 0°C to +55°C as long as T_{CASE} is not exceeded.

NOTES:

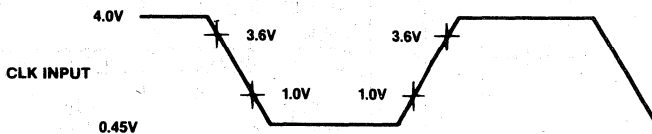
- Asynchronous inputs are INTR, NMI, HOLD, PEREQ, ERROR, and BUSY. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.
- Delay from 1.0V on the CLK, to 0.8V or 2.0V or float on the output as appropriate for valid or floating condition.
- Output load: C_L = 100 pF.
- Float condition occurs when output current is less than I_{LO} in magnitude.
- Delay measured from address either reaching 0.8V or 2.0V (valid) to status going active reaching 2.0V or status going inactive reaching 0.8V.
- For load capacitance of 10 pF or more on STATUS/PEACK lines, subtract typically 7 ns.
- These are not tested. They are guaranteed by design characterization.
- Minimum output delay timings are not tested, but are guaranteed by design characterization.

A.C. CHARACTERISTICS (Continued)



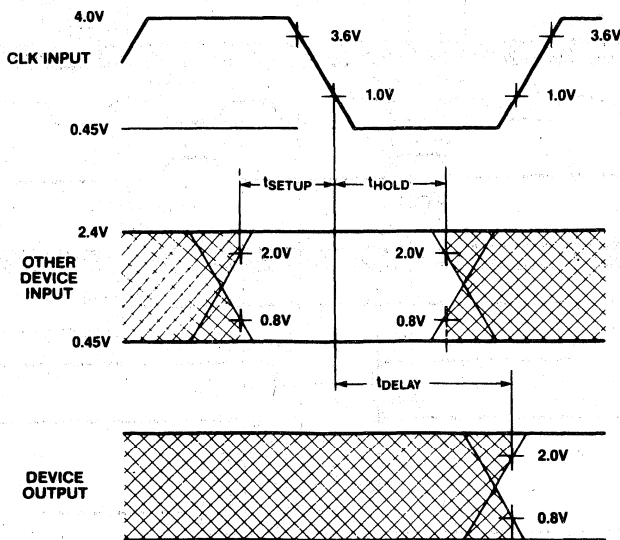
210253-37

NOTE 8:
AC Test Loading on Outputs



210253-38

NOTE 9:
AC Drive and Measurement Points—CLK Input

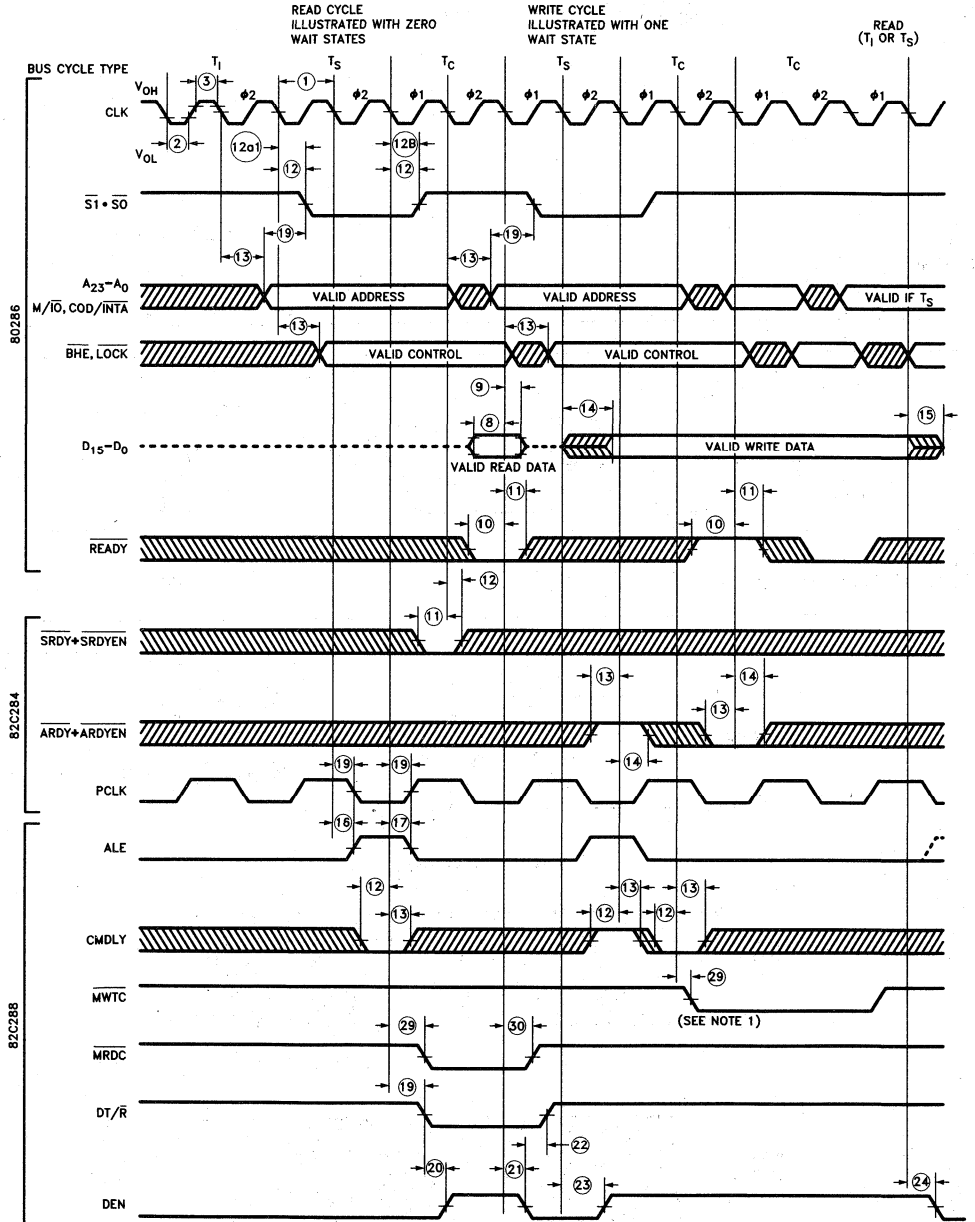


210253-39

NOTE 10:
AC Setup, Hold and Delay Time Measurement—General

WAVEFORMS

MAJOR CYCLE TIMING



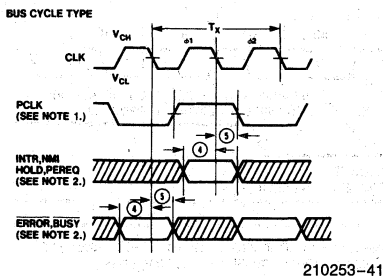
210253-40

NOTE:

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

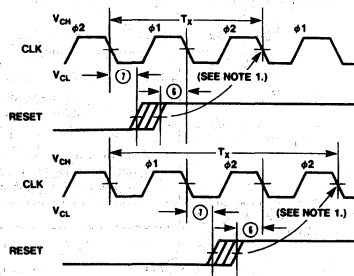
WAVEFORMS (Continued)

80286 ASYNCHRONOUS INPUT SIGNAL TIMING



210253-41

80286 RESET INPUT TIMING AND SUBSEQUENT PROCESSOR CYCLE PHASE



210253-42

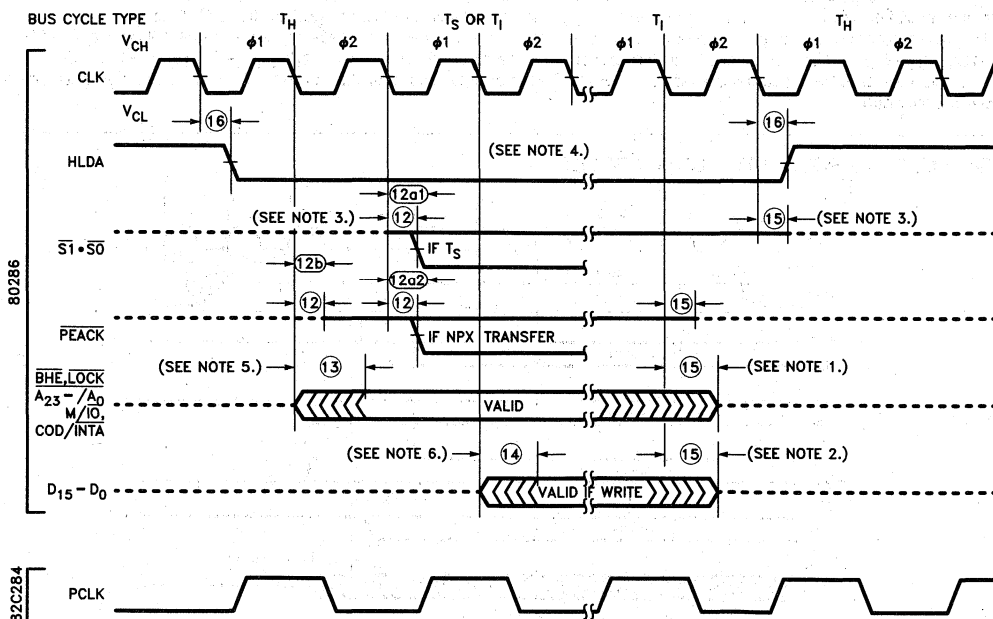
NOTES:

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.

NOTE:

When RESET meets the setup time shown, the next CLK will start or repeat $\phi 2$ of a processor cycle.

EXITING AND ENTERING HOLD



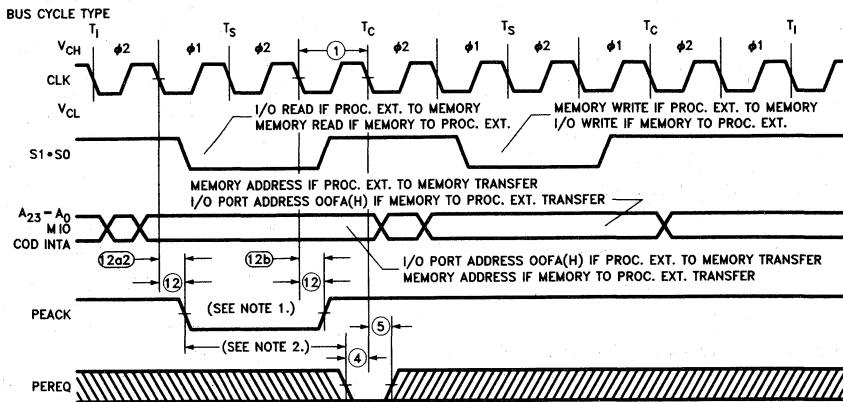
210253-43

NOTES:

1. These signals may not be driven by the 80286 during the time shown. The worst case in terms of latest float time is shown.
2. The data bus will be driven as shown if the last cycle before T_1 in the diagram was a write T_C .
3. The 80286 floats its status pins during T_H . External 20 K Ω resistors keep these signals high (see Table 16).
4. For HOLD request set up to HLDA, refer to Figure 29.
5. BHE and LOCK are driven at this time but will not become valid until T_S .
6. The data bus will remain in 3-state OFF if a read cycle is performed.

WAVEFORMS (Continued)

80286 PEREQ/PEACK TIMING FOR ONE TRANSFER ONLY

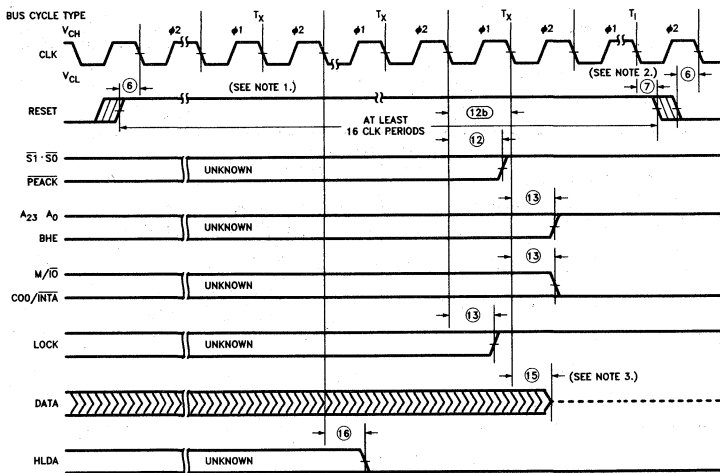


210253-44

NOTES:

1. 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 OOF(A(H)).
2. To prevent a second processor extension data operand transfer, the worst case maximum time (Shown above) is: $3 \times \text{①} - 12a_{2\text{max}} - \text{④ min.}$. The actual, configuration dependent, maximum time is: $3 \times \text{①} - 12a_{2\text{max}} - \text{④ min.} + A \times 2 \times \text{①}$. A is the number of extra T_c states added to either the first or second bus operation of the processor extension data operand transfer sequence.

INITIAL 80286 PIN STATE DURING RESET



210253-45

NOTES:

1. Setup time for RESET \uparrow may be violated with the consideration that ϕ_1 of the processor clock may begin one system CLK period later.
2. Setup and hold times for RESET \downarrow must be met for proper operation, but RESET \downarrow may occur during ϕ_1 or ϕ_2 . If RESET \downarrow occurs in ϕ_1 , the reference clock edge can be ϕ_2 of the previous bus cycle.
3. The data bus is only guaranteed to be in 3-state OFF at the time shown.

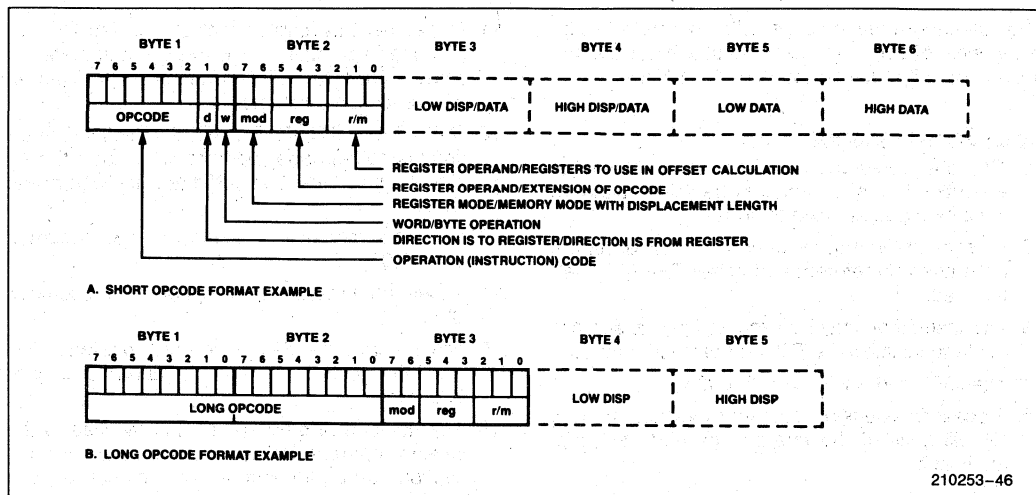


Figure 35. 80286 Instruction Format Examples

80286 INSTRUCTION SET SUMMARY

Instruction Timing Notes

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

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

Instruction Clock Count Assumptions

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

Instruction Set Summary Notes

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

Above/below refers to unsigned value

Greater refers to positive signed value

Less refers to less positive (more negative) signed values

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

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

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

if s = 1 then an immediate data byte is sign-extended to form the 16-bit operand

x don't care

z used for string primitives for comparison with ZF FLAG

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

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

n = number of times repeated

m = number of bytes of code in next instruction

Level (L)—Lexical nesting level of the procedure

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

REAL ADDRESS MODE ONLY

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

EITHER MODE

6. An exception may occur, depending on the value of the operand.
7. **LOCK** is automatically asserted regardless of the presence or absence of the **LOCK** instruction prefix.
8. **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 **LOCK** to maintain descriptor integrity in multiprocessor systems.
12. **JMP**, **CALL**, **INT**, **RET**, **IRET** instructions referring to another code segment will cause a general protection exception (13) if any privilege rule is violated.
13. A general protection exception (13) occurs if $CPL \neq 0$.
14. A general protection exception (13) occurs if $CPL > IOPL$.
15. The IF field of the flag word is not updated if $CPL > IOPL$. The IOPL field is updated only if $CPL = 0$.
16. Any violation of privilege rules as applied to the selector operand do not cause a protection exception; rather, the instruction does not return a result and the zero flag is cleared.
17. If the starting address of the memory operand violates a segment limit, or an invalid access is attempted, a general protection exception (13) will occur before the **ESC** instruction is executed. A stack segment overrun exception (12) will occur if the stack limit is violated by the operand's starting address. If a segment limit is violated during an attempted data transfer then a processor extension segment overrun exception (9) occurs.
18. The destination of an **INT**, **JMP**, **CALL**, **RET** or **IRET** instruction must be in the defined limit of a code segment or a general protection exception (13) will occur.

80286 INSTRUCTION SET SUMMARY

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS					
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode				
DATA TRANSFER									
MOV = Move:									
Register to Register/Memory	<table><tr><td>1 0 0 0 1 0 0 w</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 1 0 0 w	mod reg	r/m	2,3*	2,3*	2	9	
1 0 0 0 1 0 0 w	mod reg	r/m							
Register/memory to register	<table><tr><td>1 0 0 0 1 0 1 w</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 1 0 1 w	mod reg	r/m	2,5*	2,5*	2	9	
1 0 0 0 1 0 1 w	mod reg	r/m							
Immediate to register/memory	<table><tr><td>1 1 0 0 0 1 1 w</td><td>mod 0 0 0 r/m</td><td>data</td><td>data if w = 1</td></tr></table>	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1	2,3*	2,3*	2	9
1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1						
Immediate to register	<table><tr><td>1 0 1 1 w</td><td>reg</td><td>data</td><td>data if w = 1</td></tr></table>	1 0 1 1 w	reg	data	data if w = 1	2	2		
1 0 1 1 w	reg	data	data if w = 1						
Memory to accumulator	<table><tr><td>1 0 1 0 0 0 0 w</td><td>addr-low</td><td>addr-high</td></tr></table>	1 0 1 0 0 0 0 w	addr-low	addr-high	5	5	2	9	
1 0 1 0 0 0 0 w	addr-low	addr-high							
Accumulator to memory	<table><tr><td>1 0 1 0 0 0 1 w</td><td>addr-low</td><td>addr-high</td></tr></table>	1 0 1 0 0 0 1 w	addr-low	addr-high	3	3	2	9	
1 0 1 0 0 0 1 w	addr-low	addr-high							
Register/memory to segment register	<table><tr><td>1 0 0 0 1 1 1 0</td><td>mod 0 reg</td><td>r/m</td></tr></table>	1 0 0 0 1 1 1 0	mod 0 reg	r/m	2,5*	17,19*	2	9,10,11	
1 0 0 0 1 1 1 0	mod 0 reg	r/m							
Segment register to register/memory	<table><tr><td>1 0 0 0 1 1 0 0</td><td>mod 0 reg</td><td>r/m</td></tr></table>	1 0 0 0 1 1 0 0	mod 0 reg	r/m	2,3*	2,3*	2	9	
1 0 0 0 1 1 0 0	mod 0 reg	r/m							
PUSH = Push:									
Memory	<table><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 1 0 r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 1 1 0 r/m	5*	5*	2	9		
1 1 1 1 1 1 1 1	mod 1 1 0 r/m								
Register	<table><tr><td>0 1 0 1 0</td><td>reg</td></tr></table>	0 1 0 1 0	reg	3	3	2	9		
0 1 0 1 0	reg								
Segment register	<table><tr><td>0 0 0 reg</td><td>1 1 0</td></tr></table>	0 0 0 reg	1 1 0	3	3	2	9		
0 0 0 reg	1 1 0								
Immediate	<table><tr><td>0 1 1 0 1 0 s 0</td><td>data</td><td>data if s = 0</td></tr></table>	0 1 1 0 1 0 s 0	data	data if s = 0	3	3	2	9	
0 1 1 0 1 0 s 0	data	data if s = 0							
PUSHA = Push All	<table><tr><td>0 1 1 0 0 0 0 0</td></tr></table>	0 1 1 0 0 0 0 0	17	17	2	9			
0 1 1 0 0 0 0 0									
POP = Pop:									
Memory	<table><tr><td>1 0 0 0 1 1 1 1</td><td>mod 0 0 0 r/m</td></tr></table>	1 0 0 0 1 1 1 1	mod 0 0 0 r/m	5*	5*	2	9		
1 0 0 0 1 1 1 1	mod 0 0 0 r/m								
Register	<table><tr><td>0 1 0 1 1</td><td>reg</td></tr></table>	0 1 0 1 1	reg	5	5	2	9		
0 1 0 1 1	reg								
Segment register	<table><tr><td>0 0 0 reg</td><td>1 1 1 (reg≠01)</td></tr></table>	0 0 0 reg	1 1 1 (reg≠01)	5	20	2	9,10,11		
0 0 0 reg	1 1 1 (reg≠01)								
POPA = Pop All	<table><tr><td>0 1 1 0 0 0 0 1</td></tr></table>	0 1 1 0 0 0 0 1	19	19	2	9			
0 1 1 0 0 0 0 1									
XCHG = Exhcange:									
Register/memory with register	<table><tr><td>1 0 0 0 0 1 1 w</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 0 1 1 w	mod reg	r/m	3,5*	3,5*	2,7	7,9	
1 0 0 0 0 1 1 w	mod reg	r/m							
Register with accumulator	<table><tr><td>1 0 0 1 0</td><td>reg</td></tr></table>	1 0 0 1 0	reg	3	3				
1 0 0 1 0	reg								
IN = Input from:									
Fixed port	<table><tr><td>1 1 1 0 0 1 0 w</td><td>port</td></tr></table>	1 1 1 0 0 1 0 w	port	5	5		14		
1 1 1 0 0 1 0 w	port								
Variable port	<table><tr><td>1 1 1 0 1 1 0 w</td></tr></table>	1 1 1 0 1 1 0 w	5	5		14			
1 1 1 0 1 1 0 w									
OUT = Output to:									
Fixed port	<table><tr><td>1 1 1 0 0 1 1 w</td><td>port</td></tr></table>	1 1 1 0 0 1 1 w	port	3	3		14		
1 1 1 0 0 1 1 w	port								
Variable port	<table><tr><td>1 1 1 0 1 1 1 w</td></tr></table>	1 1 1 0 1 1 1 w	3	3		14			
1 1 1 0 1 1 1 w									
XLAT = Translate byte to AL	<table><tr><td>1 1 0 1 0 1 1 1</td></tr></table>	1 1 0 1 0 1 1 1	5	5		9			
1 1 0 1 0 1 1 1									
LEA = Load EA to register	<table><tr><td>1 0 0 0 1 1 0 1</td><td>mod reg</td><td>r/m</td></tr></table>	1 0 0 0 1 1 0 1	mod reg	r/m	3*	3*			
1 0 0 0 1 1 0 1	mod reg	r/m							
LDS = Load pointer to DS	<table><tr><td>1 1 0 0 0 1 0 1</td><td>mod reg</td><td>r/m (mod≠11)</td></tr></table>	1 1 0 0 0 1 0 1	mod reg	r/m (mod≠11)	7*	21*	2	9,10,11	
1 1 0 0 0 1 0 1	mod reg	r/m (mod≠11)							
LES = Load pointer to ES	<table><tr><td>1 1 0 0 0 1 0 0</td><td>mod reg</td><td>r/m (mod≠1)</td></tr></table>	1 1 0 0 0 1 0 0	mod reg	r/m (mod≠1)	7*	21*	2	9,10,11	
1 1 0 0 0 1 0 0	mod reg	r/m (mod≠1)							

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION		FORMAT	CLOCK COUNT		COMMENTS	
			Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
DATA TRANSFER (Continued)						
LAHF Load AH with flags	10011111		2	2		
SAHF = Store AH into flags	10011110		2	2		
PUSHF = Push flags	10011100		3	3	2	9
POPF = Pop flags	10011101		5	5	2,4	9,15
ARITHMETIC						
ADD = Add:						
Reg/memory with register to either	000000dw mod reg r/m		2,7*	2,7*	2	9
Immediate to register/memory	100000sw mod 000 r/m data data if sw = 01		3,7*	3,7*	2	9
Immediate to accumulator	0000010w data data if w = 1		3	3		
ADC = Add with carry:						
Reg/memory with register to either	000100dw mod reg r/m		2,7*	2,7*	2	9
Immediate to register/memory	100000sw mod 010 r/m data data if sw = 01		3,7*	3,7*	2	9
Immediate to accumulator	0001010w data data if w = 1		3	3		
INC = Increment:						
Register/memory	1111111w mod 000 r/m		2,7*	2,7*	2	9
Register	01000reg		2	2		
SUB = Subtract:						
Reg/memory and register to either	001010dw mod reg r/m		2,7*	2,7*	2	9
Immediate from register/memory	100000sw mod 101 r/m data data if sw = 01		3,7*	3,7*	2	9
Immediate from accumulator	0010110w data data if w = 1		3	3		
SBB = Subtract with borrow:						
Reg/memory and register to either	000110dw mod reg r/m		2,7*	2,7*	2	9
Immediate from register/memory	100000sw mod 011 r/m data data if sw = 01		3,7*	3,7*	2	9
Immediate from accumulator	0001110w data data if w = 1		3	3		
DEC = Decrement						
Register/memory	1111111w mod 001 r/m		2,7*	2,7*	2	9
Register	01001reg		2	2		
CMP = Compare						
Register/memory with register	0011101w mod reg r/m		2,6*	2,6*	2	9
Register with register/memory	0011100w mod reg r/m		2,7*	2,7*	2	9
Immediate with register/memory	100000sw mod 111 r/m data data if sw = 01		3,6*	3,6*	2	9
Immediate with accumulator	0011110w data data if w = 1		3	3		
NEG = Change sign	1111011w mod 011 r/m		2	7*	2	9
AAA = ASCII adjust for add	00110111		3	3		
DAA = Decimal adjust for add	00100111		3	3		

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION		FORMAT	CLOCK COUNT		COMMENTS	
			Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC (Continued)						
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
IMUL = Integer immediate multiply (signed)	0 1 1 0 1 0 s 1	mod reg r/m data data if s = 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	6	6
Register-Word			22	22	6	6
Memory-Byte			17*	17*	2,6	6,9
Memory-Word			25*	25*	2,6	6,9
IDIV = Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m				
Register-Byte			17	17	6	6
Register-Word			25	25	6	6
Memory-Byte			20*	20*	2,6	6,9
Memory-Word			28*	28*	2,6	6,9
AAM = ASCII adjust for multiply	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 count	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						

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC (Continued)					
AND = And:					
Reg/memory and register to either	001000dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w mod 100 r/m data data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0010010w data data if w = 1	3	3		
TEST = And function to flags, no result:					
Register/memory and register	1000010w mod reg r/m	2,6*	2,6*	2	9
Immediate data and register/memory	1111011w mod 000 r/m data data if w = 1	3,6*	3,6*	2	9
Immediate data and accumulator	1010100w data data if w = 1	3	3		
OR = Or:					
Reg/memory and register to either	000010dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w mod 001 r/m data data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0000110w data data if w = 1	3	3		
XOR = Exclusive or:					
Reg/memory and register to either	001100dw mod reg r/m	2,7*	2,7*	2	9
Immediate to register/memory	1000000w mod 110 r/m data data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0011010w data data if w = 1	3	3		
NOT = Invert register/memory	1111011w mod 010 r/m	2,7*	2,7*	2	9
STRING MANIPULATION:					
MOVS = Move byte/word	1010010w	5	5	2	9
CMPS = Compare byte/word	1010011w	8	8	2	9
SCAS = Scan byte/word	1010111w	7	7	2	9
LDS = Load byte/wd to AL/AX	1010110w	5	5	2	9
STOS = Stor byte/wd from AL/A	1010101w	3	3	2	9
INS = Input byte/wd from DX port	0110110w	5	5	2	9,14
OUTS = Output byte/wd to DX port	0110111w	5	5	2	9,14
Repeated by count in CX					
MOV _s = Move string	11110011 1010010w	5+4n	5+4n	2	9
CMPS = Compare string	1111001z 1010011w	5+9n	5+9n	2,8	8,9
SCAS = Scan string	1111001z 1010111w	5+8n	5+8n	2,8	8,9
LDS = Load string	11110011 1010110w	5+4n	5+4n	2,8	8,9
STOS = Store string	11110011 1010101w	4+3n	4+3n	2,8	8,9
INS = Input string	11110011 0110110w	5+4n	5+4n	2	9,14
OUTS = Output string	11110011 0110111w	5+4n	5+4n	2	9,14

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS				
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode			
CONTROL TRANSFER								
CALL = Call:								
Direct within segment	<table><tr><td>1 1 1 0 1 0 0 0</td><td>disp-low</td><td>disp-high</td></tr></table>	1 1 1 0 1 0 0 0	disp-low	disp-high	7 + m	7 + m	2	18
1 1 1 0 1 0 0 0	disp-low	disp-high						
Register/memory indirect within segment	<table><tr><td>1 1 1 1 1 1 1 1</td><td>mod 0 1 0</td><td>r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 0 1 0	r/m	7 + m, 11 + m*	7 + m, 11 + m*	2,8	8,9,18
1 1 1 1 1 1 1 1	mod 0 1 0	r/m						
Direct intersegment	<table><tr><td>1 0 0 1 1 0 1 0</td><td colspan="2">segment offset</td></tr></table>	1 0 0 1 1 0 1 0	segment offset		13 + m	26 + m	2	11,12,18
1 0 0 1 1 0 1 0	segment offset							
Protected Mode Only (Direct intersegment):								
Via call gate to same privilege level			41 + m		8,11,12,18			
Via call gate to different privilege level, no parameters			82 + m		8,11,12,18			
Via call gate to different privilege level, x parameters			86 + 4x + m		8,11,12,18			
Via TSS			177 + m		8,11,12,18			
Via task gate			182 + m		8,11,12,18			
Indirect intersegment	<table><tr><td>1 1 1 1 1 1 1 1</td><td>mod 0 1 1</td><td>r/m</td></tr></table> (mod≠11)	1 1 1 1 1 1 1 1	mod 0 1 1	r/m	16 + m	29 + m*	2	8,9,11,12,18
1 1 1 1 1 1 1 1	mod 0 1 1	r/m						
Protected Mode Only (Indirect intersegment):								
Via call gate to same privilege level			44 + m*		8,9,11,12,18			
Via call gate to different privilege level, no parameters			83 + m*		8,9,11,12,18			
Via call gate to different privilege level, x parameters			90 + 4x + m*		8,9,11,12,18			
Via TSS			180 + m*		8,9,11,12,18			
Via task gate			185 + m*		8,9,11,12,18			
JMP = Unconditional jump:								
Short/long	<table><tr><td>1 1 1 0 1 0 1 1</td><td>disp-low</td></tr></table>	1 1 1 0 1 0 1 1	disp-low	7 + m	7 + m		18	
1 1 1 0 1 0 1 1	disp-low							
Direct within segment	<table><tr><td>1 1 1 0 1 0 0 1</td><td>disp-low</td><td>disp-high</td></tr></table>	1 1 1 0 1 0 0 1	disp-low	disp-high	7 + m	7 + m		18
1 1 1 0 1 0 0 1	disp-low	disp-high						
Register/memory indirect within segment	<table><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 0 0</td><td>r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 1 0 0	r/m	7 + m, 11 + m*	7 + m, 11 + m*	2	9,18
1 1 1 1 1 1 1 1	mod 1 0 0	r/m						
Direct intersegment	<table><tr><td>1 1 1 0 1 0 1 0</td><td colspan="2">segment offset</td></tr></table>	1 1 1 0 1 0 1 0	segment offset		11 + m	23 + m		11,12,18
1 1 1 0 1 0 1 0	segment offset							
Protected Mode Only (Direct intersegment):								
Via call gate to same privilege level			38 + m		8,11,12,18			
Via TSS			175 + m		8,11,12,18			
Via task gate			180 + m		8,11,12,18			
Indirect intersegment	<table><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 0 1</td><td>r/m</td></tr></table> (mod≠11)	1 1 1 1 1 1 1 1	mod 1 0 1	r/m	15 + m*	26 + m*	2	8,9,11,12,18
1 1 1 1 1 1 1 1	mod 1 0 1	r/m						
Protected Mode Only (Indirect intersegment):								
Via call gate to same privilege level			41 + m*		8,9,11,12,18			
Via TSS			178 + m*		8,9,11,12,18			
Via task gate			183 + m*		8,9,11,12,18			
RET = Return from CALL:								
Within segment	<table><tr><td>1 1 0 0 0 0 1 1</td></tr></table>	1 1 0 0 0 0 1 1	11 + m	11 + m	2	8,9,18		
1 1 0 0 0 0 1 1								
Within seg adding immed to SP	<table><tr><td>1 1 0 0 0 0 1 0</td><td>data-low</td><td>data-high</td></tr></table>	1 1 0 0 0 0 1 0	data-low	data-high	11 + m	11 + m	2	8,9,18
1 1 0 0 0 0 1 0	data-low	data-high						
Intersegment	<table><tr><td>1 1 0 0 1 0 1 1</td></tr></table>	1 1 0 0 1 0 1 1	15 + m	25 + m	2	8,9,11,12,18		
1 1 0 0 1 0 1 1								
Intersegment adding immediate to SP	<table><tr><td>1 1 0 0 1 0 1 0</td><td>data-low</td><td>data-high</td></tr></table>	1 1 0 0 1 0 1 0	data-low	data-high	15 + m		2	8,9,11,12,18
1 1 0 0 1 0 1 0	data-low	data-high						
Protected Mode Only (RET):								
To different privilege level			55 + m		9,11,12,18			

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)									
JE/JZ = Jump on equal zero	<table><tr><td>0 1 1 1 0 1 0 0</td><td>disp</td></tr></table>	0 1 1 1 0 1 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 0 0	disp								
JL/JNGE = Jump on less/not greater or equal	<table><tr><td>0 1 1 1 1 1 0 0</td><td>disp</td></tr></table>	0 1 1 1 1 1 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 0 0	disp								
JLE/JNG = Jump on less or equal/not greater	<table><tr><td>0 1 1 1 1 1 1 0</td><td>disp</td></tr></table>	0 1 1 1 1 1 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 1 0	disp								
JB/JNAE = Jump on below/not above or equal	<table><tr><td>0 1 1 1 0 0 1 0</td><td>disp</td></tr></table>	0 1 1 1 0 0 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 1 0	disp								
JBE/JNA = Jump on below or equal/not above	<table><tr><td>0 1 1 1 0 1 1 0</td><td>disp</td></tr></table>	0 1 1 1 0 1 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 1 0	disp								
JP/JPE = Jump on parity/parity even	<table><tr><td>0 1 1 1 1 0 1 0</td><td>disp</td></tr></table>	0 1 1 1 1 0 1 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 1 0	disp								
JO = Jump on overflow	<table><tr><td>0 1 1 1 0 0 0 0</td><td>disp</td></tr></table>	0 1 1 1 0 0 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 0 0	disp								
JS = Jump on sign	<table><tr><td>0 1 1 1 1 0 0 0</td><td>disp</td></tr></table>	0 1 1 1 1 0 0 0	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 0 0	disp								
JNE/JNZ = Jump on not equal/not zero	<table><tr><td>0 1 1 1 0 1 0 1</td><td>disp</td></tr></table>	0 1 1 1 0 1 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 0 1	disp								
JNL/JGE = Jump on not less/greater or equal	<table><tr><td>0 1 1 1 1 1 0 1</td><td>disp</td></tr></table>	0 1 1 1 1 1 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 0 1	disp								
JNLE/JG = Jump on not less or equal/greater	<table><tr><td>0 1 1 1 1 1 1 1</td><td>disp</td></tr></table>	0 1 1 1 1 1 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 1 1 1	disp								
JNB/JAE = Jump on not below/above or equal	<table><tr><td>0 1 1 1 0 0 1 1</td><td>disp</td></tr></table>	0 1 1 1 0 0 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 1 1	disp								
JNBE/JA = Jump on not below or equal/above	<table><tr><td>0 1 1 1 0 1 1 1</td><td>disp</td></tr></table>	0 1 1 1 0 1 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 1 1 1	disp								
JNP/JPO = Jump on not par/par odd	<table><tr><td>0 1 1 1 1 0 1 1</td><td>disp</td></tr></table>	0 1 1 1 1 0 1 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 1 1	disp								
JNO = Jump on not overflow	<table><tr><td>0 1 1 1 0 0 0 1</td><td>disp</td></tr></table>	0 1 1 1 0 0 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 0 0 0 1	disp								
JNS = Jump on not sign	<table><tr><td>0 1 1 1 1 0 0 1</td><td>disp</td></tr></table>	0 1 1 1 1 0 0 1	disp	7 + m or 3	7 + m or 3		18		
0 1 1 1 1 0 0 1	disp								
LOOP = Loop CX times	<table><tr><td>1 1 1 0 0 0 1 0</td><td>disp</td></tr></table>	1 1 1 0 0 0 1 0	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 1 0	disp								
LOOPZ/LOOPE = Loop while zero/equal	<table><tr><td>1 1 1 0 0 0 0 1</td><td>disp</td></tr></table>	1 1 1 0 0 0 0 1	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 0 1	disp								
LOOPNZ/LOOPNE = Loop while not zero/equal	<table><tr><td>1 1 1 0 0 0 0 0</td><td>disp</td></tr></table>	1 1 1 0 0 0 0 0	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 0 0	disp								
JCXZ = Jump on CX zero	<table><tr><td>1 1 1 0 0 0 1 1</td><td>disp</td></tr></table>	1 1 1 0 0 0 1 1	disp	8 + m or 4	8 + m or 4		18		
1 1 1 0 0 0 1 1	disp								
ENTER = Enter Procedure	<table><tr><td>1 1 0 0 1 0 0 0</td><td>data-low</td><td>data-high</td><td>L</td></tr></table>	1 1 0 0 1 0 0 0	data-low	data-high	L			2,8	8,9
1 1 0 0 1 0 0 0	data-low	data-high	L						
L = 0		11	11						
L = 1		15	15	2,8	8,9				
L > 1		16 + 4(L - 1)	16 + 4(L - 1)	2,8	8,9				
LEAVE = Leave Procedure	<table><tr><td>1 1 0 0 1 0 0 1</td></tr></table>	1 1 0 0 1 0 0 1	5	5					
1 1 0 0 1 0 0 1									
INT = Interrupt:									
Type specified	<table><tr><td>1 1 0 0 1 1 0 1</td><td>type</td></tr></table>	1 1 0 0 1 1 0 1	type	23 + m		2,7,8			
1 1 0 0 1 1 0 1	type								
Type 3	<table><tr><td>1 1 0 0 1 1 0 0</td></tr></table>	1 1 0 0 1 1 0 0	23 + m		2,7,8				
1 1 0 0 1 1 0 0									
INTO = Interrupt on overflow	<table><tr><td>1 1 0 0 1 1 1 0</td></tr></table>	1 1 0 0 1 1 1 0	24 + m or 3 (3 if no interrupt)	(3 if no interrupt)	2,6,8				
1 1 0 0 1 1 1 0									

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only: Via interrupt or trap gate to same privilege level Via interrupt or trap gate to fit different privilege level Via Task Gate			40 + m 78 + m 167 + m		7,8,11,12,18 7,8,11,12,18 7,8,11,12,18
IRET = Interrupt return	11001111	17 + m	31 + m	2,4	8,9,11,12,15,18
Protected Mode Only: To different privilege level To different task (NT = 1)			55 + m 169 + m		8,9,11,12,15,18 8,9,11,12,18
BOUND = Detect value out of range	01100010 mod reg r/m	13*	13* (Use INT clock count if exception 5)	2,6	8,8,9,11,12,18
PROCESSOR CONTROL					
CLC = Clear carry	11111000	2	2		
CMC = Complement carry	11110101	2	2		
STC = Set carry	11111001	2	2		
CLD = Clear direction	11111100	2	2		
STD = Set direction	11111101	2	2		
CLI = Clear interrupt	11111010	3	3		14
STI = Set interrupt	11111011	2	2		14
HLT = Halt	11110100	2	2		13
WAIT = Wait	10011011	3	3		
LOCK = Bus lock prefix	11110000	0	0		14
CTS = Clear task switched flag	00001111 00000110	2	2	3	13
ESC = Processor Extension Escape	11011TTT mod LLL r/m (TTT LLL are opcode to processor extension)	9–20*	9–20*	5,8	8,17
SEG = Segment Override Prefix	001 reg 110	0	0		
PROTECTION CONTROL					
LGDT = Load global descriptor table register	00001111 00000001 mod 010 r/m	11*	11*	2,3	9,13
SGDT = Store global descriptor table register	00001111 00000001 mod 000 r/m	11*	11*	2,3	9
LIDT = Load interrupt descriptor table register	00001111 00000001 mod 011 r/m	12*	12*	2,3	9,13
SIDT = Store interrupt descriptor table register	00001111 00000001 mod 001 r/m	12*	12*	2,3	9
LLDT = Load local descriptor table register from register memory	00001111 00000000 mod 010 r/m		17,19*	1	9,11,13
SLDT = Store local descriptor table register to register/memory	00001111 00000000 mod 000 r/m		2,3*	1	9

Shaded areas indicate instructions not available in 8086, 88 microsystems.

80286 INSTRUCTION SET SUMMARY (Continued)

FUNCTION	FORMAT	CLOCK COUNT		COMMENTS	
		Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
PROTECTION CONTROL (Continued)					
LTR = Local task register from register/memory	00001111 00000000 mod 011 r/m		17,19*	1	9,11,13
STR = Store task register to register/memory	00001111 00000000 mod 001 r/m		2,3*	1	9
LMSW = Load machine status word from register/memory	00001111 00000001 mod 110 r/m	3,6*	3,6*	2,3	9,13
SMSW = Store machine status word	00001111 00000001 mod 100 r/m	2,3*	2,3*	2,3	9
LAR = Load access rights from register/memory	00001111 00000010 mod reg r/m		14,16*	1	9,11,16
LSL = Load segment limit from register/memory	00001111 00000011 mod reg r/m		14,16*	1	9,11,16
ARPL = Adjust requested privilege level: from register/memory	01100011 mod reg r/m		10*,11*	2	8,9
VERR = Verify read access: register/memory	00001111 00000000 mod 100 r/m		14,16*	1	9,11,16
VERR = Verify write access:	00001111 00000000 mod 101 r/m		14,16*	1	9,11,16

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Footnotes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DC

REG is assigned according to the following table:

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

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -014 data sheet. Please review this summary carefully.

1. Removed the Range of Clock Rates bullet.
2. The maximum ambient temperature (T_A) vs Various Airflows Table has been updated.
3. Removed the maximum values of System Clock (CLK) LOW period (t_2) of 8 MHz, 10 MHz, and 12.5 MHz parts in the A.C. Characteristics table.
4. Removed the maximum values of System Clock (CLK) HIGH period (t_3) of 8 MHz, 10 MHz, and 12.5 MHz parts in the A.C. Characteristics table.
5. Deleted the 82C284 and 82C288 A.C. Characteristics tables.

80287XL/XLT CHMOS III MATH COPROCESSOR

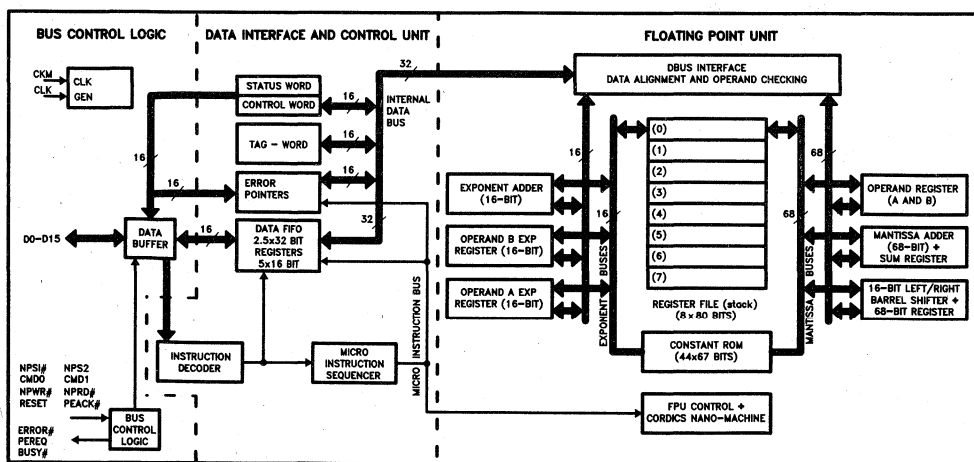
- Interfaces with 80286 and 80C286 CPUs
- Operates in Any Socket Designed for Intel 80287 or 80287XL up to 12.5 MHz Clock Speeds
- Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic
- 50% Higher Performance than Intel 80287
- Low Power CHMOS III Technology
- Upward Object Code Compatible from Intel 80287 and 8087
- Expands Data Types to Include 32-, 64-, 80-Bit Floating Point, or Integers, and 18 Digit BCD Operands
- Extends CPU Instruction Set to Include Trigonometric, Logarithmic, Exponential, and Arithmetic Instruction
- Implements 387™ Transcendental Operations for SINE, COSINE, TANGENT ARCTANGENT and LOGARITHM
- Eight 80-Bit Numeric Registers; for Stack use or Individual Access
- Available in 40-pin DIP as 80287XL and 44-pin PLCC as 80287XLT

(See Packaging Outlined and Dimensions, order #231369)

The Intel 80287XL Math CoProcessor is an extension to the Intel 80286 microprocessor architecture. When combined with an 80286 microprocessor, the 80287XL dramatically increases the processing speed of computer application software which utilize floating point mathematical operations. This makes an ideal addition to a computer workstation platform for applications such as financial modeling and spreadsheets, CAD/CAM, or business graphics.

The 80287XL Math CoProcessor adds over seventy mnemonics to the Intel 80286 microprocessor instruction set. The 80287XL is compatible with the Intel 80287 and 8087 Math CoProcessors. The 80287XL increases performance by over 50% in typical floating-point tests, such as a Whetstone test, compared to the Intel 80287. The 80287XL supports integer, floating point and BCD data formats and fully conforms to the ANSI/IEEE 754-1985 Floating Point Standard.

There are two versions of 80287XL: 80287XL in a 40-pin DIP package and the 80287XLT in a 44-pin PLCC package for small footprint applications such as portable personal computers. Each supports a clock speed up to 12.5 MHz which enables operation in any Math CoProcessor socket designed for the Intel 80287-6/8/10 or Intel 80C287A-12. Both versions are manufactured with low-power, CHMOS III technology.



290376-1

Figure 0.1. 80287XL Block Diagram

CONTENTS

PAGE

1.0 Functional Description	3-120
2.0 Programming Interface	3-120
2.1 Data Types	3-121
2.2 Numeric Operands	3-121
2.3 Register Set	3-121
2.3.1 Data Registers	3-121
2.3.2 Tag Word	3-121
2.3.3 Status Word	3-121
2.3.4 Control Word	3-125
2.3.5 Instruction and Data Pointers	3-126
2.4 Interrupt Description	3-127
2.5 Exception Handling	3-127
2.6 Initialization	3-128
2.7 8087 and 80287 Compatibility	3-128
2.7.1 General Differences	3-129
2.7.2 Exceptions	3-129
3.0 Hardware Interface	3-130
3.1 Signal Description	3-130
3.1.1 Clock (CLK)	3-132
3.1.2 Clocking Mode (CKM)	3-132
3.1.3 System Reset (RESET)	3-132
3.1.4 Processor Extension Request (PEREQ)	3-132
3.1.5 Busy Status (BUSY #)	3-132
3.1.6 Error Status (ERROR #)	3-133
3.1.7 Processor Extension Acknowledge (PEACK #)	3-133
3.1.8 Data Pins (D ₁₅ -D ₀)	3-133
3.1.9 Numeric Processor Write (NPWR #)	3-133
3.1.10 Numeric Processor Read (NPRD #)	3-133
3.1.11 Numeric Processor Selects (NPS1 # and NPS2)	3-133
3.1.12 Command Selects (CMD0 and CMD1)	3-133
3.1.13 System Power (V _{CC})	3-133
3.1.14 System Ground (V _{SS})	3-133
3.2 Processor Architecture	3-133
3.2.1 Bus Control Logic	3-133
3.2.2 Data Interface and Control Unit	3-133
3.2.3 Floating-Point Unit	3-134

CONTENTS	PAGE
3.3 Bus Cycles	3-134
3.3.1 80287XL Addressing	3-134
3.3.2 CPU/NPX Synchronization	3-134
3.4 Bus Operation	3-135
3.5 80286/80287XL, 80C286/80287XL Interface and Socket Compatibility	3-135
4.0 Electrical Data	3-137
4.1 Absolute Maximum Ratings	3-137
4.2 Power and Frequency Requirements	3-137
4.3 DC Characteristics	3-137
4.4 AC Characteristics	3-138
5.0 80287XL Extensions to the CPU's Instruction Set	3-144
 FIGURES	 PAGE
Figure 0.1 80287XL Block Diagram	3-116
Figure 1.1 80287XL Register Set	3-120
Figure 2.1 80287XL Tag Word	3-122
Figure 2.2 Status Word	3-123
Figure 2.3 Control Word	3-126
Figure 2.4 Protected Mode Instruction and Data Pointer Image in Memory	3-127
Figure 2.5 Real Mode Instruction and Data Pointer Image in Memory	3-127
Figure 3.1 DIP Pin Configurations	3-131
Figure 3.2 PLCC Pin Configuration	3-131
Figure 3.3 80286/80287XL System Configuration	3-136
Figure 4.1 AC Drive and Measurement Points—CLK Input	3-140
Figure 4.2 AC Setup, Hold, and Delay Time Measurements—General	3-140
Figure 4.3 AC Test Loading on Outputs	3-140
Figure 4.4 Data Transfer Timing (Initiated by CPU)	3-141
Figure 4.5 Data Channel Timing (Initiated by 80287XL)	3-141
Figure 4.6 ERROR# Output Timing	3-142
Figure 4.7 CLK, RESET Timing (CKM = 1)	3-142
Figure 4.8 CLK, NPRD#, NPWR# Timing (CKM = 1)	3-142
Figure 4.9 CLK, RESET Timing (CKM = 0)	3-142
Figure 4.10 CLK, NPRD#, NPWR# Timing (CKM = 0)	3-143
Figure 4.11 RESET, PEACK# Setup and Hold Timing	3-143

TABLES

PAGE

Table 2.1	80287XL Data Type Representation in Memory	3-122
Table 2.2	Condition Code Interpretation	3-124
Table 2.3	Condition Code Interpretation after FPREM and FPREM1 Instructions	3-125
Table 2.4	Condition Code Resulting from Comparison	3-125
Table 2.5	Condition Code Defining Operand Class	3-125
Table 2.6	CPU Interrupt Vectors Reserved for NPX	3-127
Table 2.7	Exceptions	3-128
Table 3.1	Pin Summary	3-130
Table 3.2	PLCC Pin Cross-Reference	3-132
Table 3.3	Output Pin Status during Reset	3-132
Table 3.4	Bus Cycles Definition	3-134
Table 3.5	I/O Address Decoding	3-134
Table 4.1	DC Specifications	3-137
Table 4.2	Timing Requirements	3-138
Table 4.3	Timing Responses	3-139
Table 4.4	Clock Timings	3-139
Table 5.1	Instruction Formats	3-144

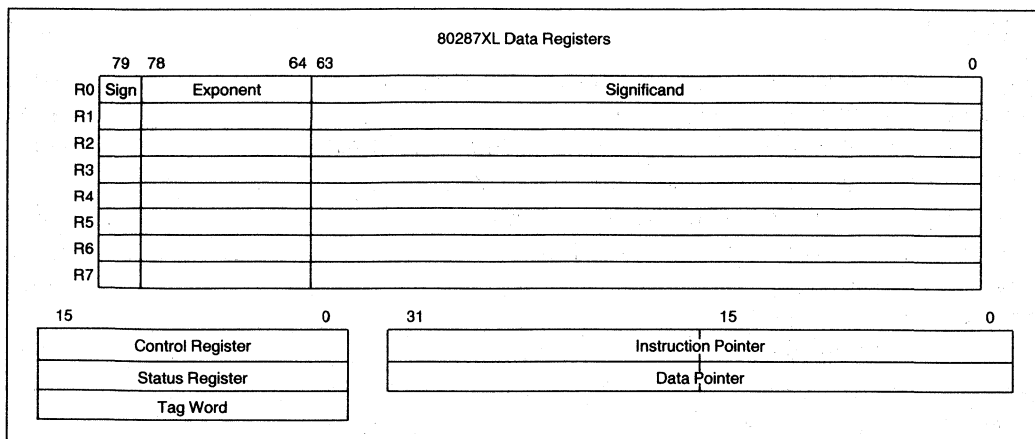


Figure 1.1. 80287XL Register Set

1.0 FUNCTIONAL DESCRIPTION

The 80287XL Math CoProcessor provides arithmetic instructions for a variety of numeric data types. It also executes numerous built-in transcendental functions (e.g. tangent, sine, cosine, and log functions). The 80287XL effectively extends the register and instruction set of its CPU for existing data types and adds several new data types as well. Figure 1.1 shows the additional registers visible to programs in a system that includes the 80287XL. Essentially, the 80287XL can be treated as an additional resource or an extension to the CPU. The CPU together with an 80287XL Math CoProcessor can be used as a single unified system.

The 80287XL has two operating modes. After reset, the 80287XL is in the real-address mode. It can be placed into protected mode by executing the FSETPM instruction. It can be switched back to real-address mode by executing the FRSTPM instruction (note that this feature is useful only with CPU's that can also switch back to real-address mode). These instructions control the format of the administrative instructions FLDENV, FSTENV, FRSTOR, and FSAVE. Regardless of operating mode, all references to memory for numerics data or status information are performed by the CPU, and therefore obey the memory-management and protection rules of the CPU.

In real-address mode, a system that includes the 80287XL is completely upward compatible with software for the 8086/8087 and for 80286/80287 or 80C287A real-address mode.

In protected mode, a system that includes the 80287XL is completely upward compatible with software for 80286/80287 or 80C287A protected mode systems. The only differences of operation that may

appear when 8086/8087 programs are ported to a protected-mode 80287XL system are in the format of operands for the administrative instructions FLDENV, FSTENV, FRSTOR, and FSAVE. These instructions are normally used only by exception handlers and operating systems, not by applications programs.

2.0 PROGRAMMING INTERFACE

The 80287XL adds to the CPU additional data types, registers, instructions, and interrupts specifically designed to facilitate high-speed numerics processing. To use the 80287XL requires no special programming tools, because all new instructions and data types are directly supported by the assembler and compilers for high-level languages. All 8086/8088 development tools that support the 8087 can also be used to develop software for the 80286/80287XL in real-address mode. All 80286 development tools that support the 80287/80C287A can also be used to develop software for the 80286/80287XL and 80C286/80287XL. The 80287XL supports all 80387 instructions, producing the same binary results.

All communication between the CPU and the 80287XL is transparent to applications software. The CPU automatically controls the 80287XL whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the 80287XL. All memory addressing modes are available for addressing numerics operands.

Section 6 at the end of this data sheet lists the instructions that the 80287XL adds to the 80286 instruction set.

2.1 Data Types

Table 2.1 lists the seven data types that the 80287XL supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at physical-memory addresses that correspond to the word size of the CPU; operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

Internally, the 80287XL holds all numbers in the extended-precision real format. Instructions that load operands from memory automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating-point numbers, or 18-digit packed BCD numbers into extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

2.2 Numeric Operands

A typical NPX (Numeric Processor Extension) instruction accepts one or two operands and produces one (or sometimes two) results. In two-operand instructions, one operand is the contents of an NPX register, while the other may be a memory location. The operands of some instructions are predefined; for example, FSQRT always takes the square root of the number in the top stack element.

2.3 Register Set

Figure 1.1 shows the 80287XL register set. When an 80287XL is present in a system, programmers may use these registers in addition to the registers normally available on the CPU.

2.3.1 DATA REGISTERS

80287XL computations use the 80287XL's data registers. These eight 80-bit registers provide the equivalent capacity of 20 32-bit registers. Each of the eight data registers in the 80287XL is 80 bits wide and is divided into "fields" corresponding to the NPX's extended-precision real data type.

The 80287XL register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as individually addressable registers. The TOP field in the status word identifies the current top-of-stack register. A "push" operation decrements TOP by one and loads a value into the new top register. A "pop" operation stores the value from the current top register and then increments

TOP by one. The 80287XL register stack grows "down" toward lower-addressed registers.

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

2.3.2 TAG WORD

The tag word marks the content of each numeric data register, as Figure 2.1 shows. Each two-bit tag represents one of the eight data registers. The principal function of the tag word is to optimize the NPX's performance and stack handling by making it possible to distinguish between empty and nonempty register locations. It also enables exception handlers to identify special values (e.g. NaNs or denormals) in the contents of a stack location without the need to perform complex decoding of the actual data.

2.3.3 STATUS WORD

The 16-bit status word (in the status register) shown in Figure 2.2 reflects the overall state of the 80287XL. It may be read and inspected by programs.

Bit 15, the B-bit (busy bit) is included for 8087 compatibility only. It always has the same value as the ES bit (bit 7 of the status word); it does **not** indicate the status of the BUSY# output of 80287XL.

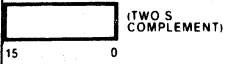
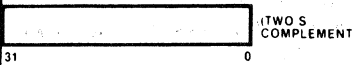
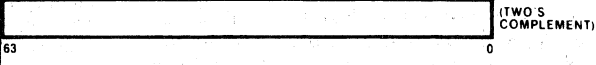
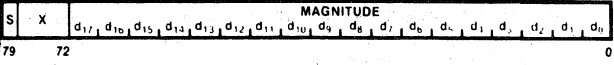
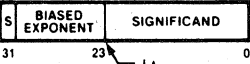
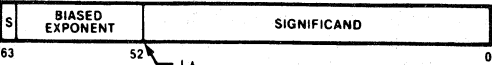
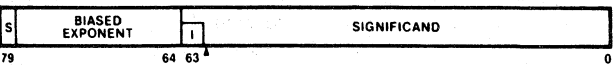
Bits 13–11 (TOP) point to the 80287XL register that is the current top-of-stack.

The four numeric condition code bits (C_3 – C_0) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of these instructions on the condition code are summarized in Tables 2.2 through 2.5.

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the ERROR# signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C_1) distinguishes between stack overflow ($C_1 = 1$) and underflow ($C_1 = 0$).

Table 2.1. 80287XL Data Type Representation in Memory

Data Formats	Range	Precision	Most Significant Byte HIGHEST ADDRESSED BYTE													
			7	0	7	0	7	0	7	0	7	0	7	0	7	0
Word Integer	$\pm 10^4$	16 Bits														
Short Integer	$\pm 10^9$	32 Bits														
Long Integer	$\pm 10^{18}$	64 Bits														
Packed BCD	$\pm 10^{18}$	18 Digits														
Single Precision	$\pm 10^{\pm 38}$	24 Bits														
Double Precision	$\pm 10^{\pm 308}$	53 Bits														
Extended Precision	$\pm 10^{\pm 4932}$	64 Bits														

290376-2

NOTES:

1. S = Sign bit (0 = positive, 1 = negative)
2. d_n = Decimal digit (two per byte)
3. X = Bits have no significance: 80287XL ignores when loading, zeroes when storing
4. Δ = Position of implicit binary point
5. I = Integer bit of significand; stored in temporary real, implicit in single and double precision
6. Exponent Bias (normalized values):
Single: 127 (7FH)
Double: 1023 (3FFH)
Extended Real: 16383 (3FFFH)
7. Packed BCD: $(-1)^S (D_{17} \dots D_0)$
8. Real: $(-1)^S (2E\text{-BIAS}) (F_0 F_1 \dots)$

15							0
TAG (7)	TAG (6)	TAG (5)	TAG (4)	TAG (3)	TAG (2)	TAG (1)	TAG (0)

NOTE:

The index i of tag(i) is not top-relative. A program typically uses the "top" field of Status Word to determine which tag(i) field refers to logical top of stack.

TAG VALUES:

- 00 = Valid
- 01 = Zero
- 10 = QNaN, SNaN, Infinity, Denormal and Unsupported Formats
- 11 = Empty

Figure 2.1. 80287XL Tag Word

Figure 2.2 shows the six exception flags in bits 5–0 of the status word. Bits 5–0 are set to indicate that the 80287XL has detected an exception while executing an instruction. A later section entitled “Exception Handling” explains how they are set and used.

Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the

value of ES (bit 7) and its reflection in the B-bit (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the ERROR# output of the 80287XL is activated immediately.

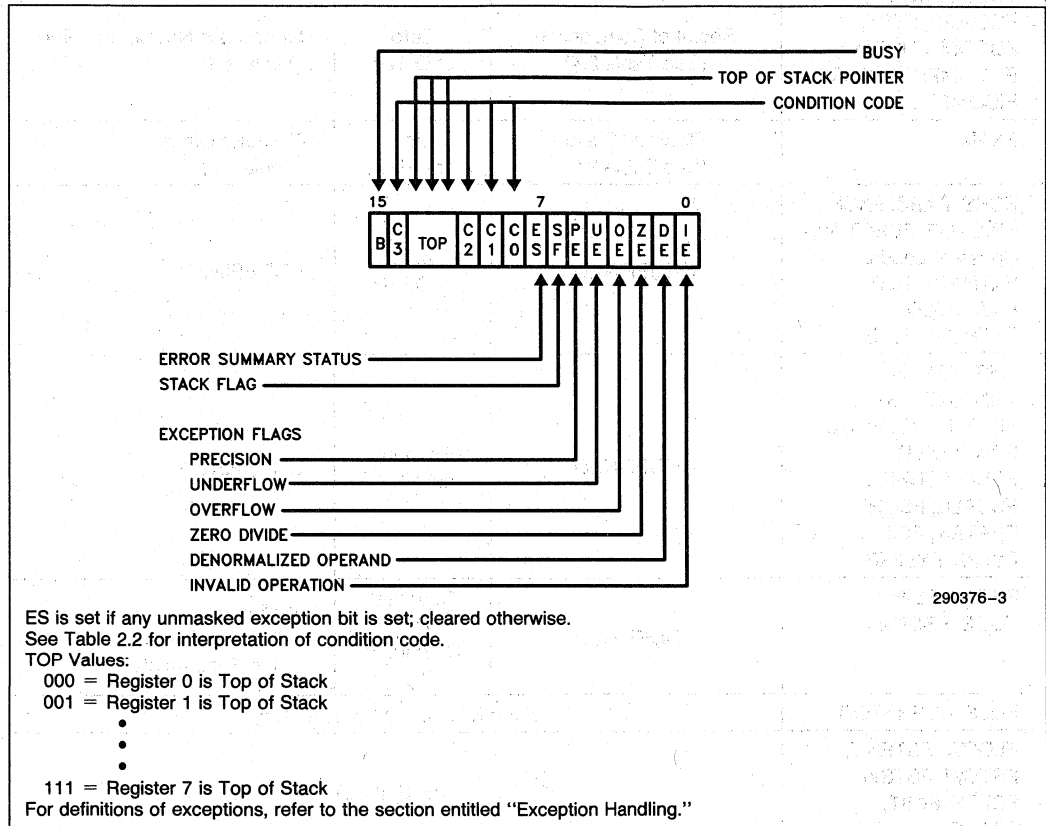


Figure 2.2. Status Word

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1 (See Table 2.3)	Three Least Significant Bits of Quotient Q2 Q0			Reduction 0 = Complete 1 = Incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of Comparison (See Table 2.4)		Zero or O/U #	Operand is Not Comparable (Table 2.4)
FXAM	Operand Class (See Table 2.5)		Sign or O/U #	Operand Class (Table 2.5)
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant Loads, FEXTRACT, FLD, FILD, FBLD, FSTP (Ext Real)	UNDEFINED		Zero or O/U #	UNDEFINED
FIST, FBSTP, FRNDINT, FST FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U #	UNDEFINED
FPTAN, FSIN, FCOS, FSINCOS	UNDEFINED		Roundup or O/U # Undefined if C2 = 1	Reduction 0 = Complete 1 = Incomplete
FLDENV, FRSTOR	Each Bit Loaded from Memory			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FINIT, FSAVE	UNDEFINED			

3-124

Table 2.3. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: Further iteration required for complete reduction.	
0	Q1	Q0	Q2	Q MOD 8	Complete Reduction: C0, C3, C1 contain three least significant bits of quotient.
	0	0	0	0	
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 2.4. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 2.5. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	- Unsupported
0	0	1	1	- Nan
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	- Normal
0	1	1	1	- Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	- 0
1	0	1	1	- Empty
1	1	0	0	+ Denormal
1	1	1	0	- Denormal

2.3.4 CONTROL WORD

The NPX provides several processing options that are selected by loading a control word from memory into the control register. Figure 2.3 shows the format and encoding of fields in the control word.

The low-order byte of this control word configures exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the 80287XL recognizes.

The high-order byte of the control word configures the 80287XL operating mode, including precision, rounding, and infinity control.

- The “infinity control bit” (bit 12) is not meaningful to the 80287XL, and programs must ignore its value. To maintain compatibility with the 8087 and 80287, this bit can be programmed; however, regardless of its value, the 80287XL always treats infinity in the affine sense ($-\infty < +\infty$). This bit is initialized to zero both after a hardware reset and after the FINIT instruction.
- The rounding control (RC) bits (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FEXTRACT, FABS, and FCHS), and all transcendental instructions.
- The precision control (PC) bits (bits 9–8) can be used to set the 80287XL internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

2.3.5 INSTRUCTION AND DATA POINTERS

Because the NPX operates in parallel with the CPU, any exceptions detected by the NPX may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the failing numeric instruction, the 80287XL contains registers that aid in diagnosis. These registers supply the opcode of the failing numeric instruction, the address of the instruction, and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written exception handlers. Whenever the 80287XL executes a new ESC instruction, it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if

present), and the opcode. CPUs with 32-bit internal architectures contain 32-bit versions of these registers and do not use the contents of the NPX registers. This difference is not apparent to programmers, however.

The instruction and data pointers appear in one of four formats depending on the operating mode of the system (protected mode or real-address mode) and (for CPUs with 32-bit internal architectures) depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). (See Figures 2.4 and 2.5.) The ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR are used to transfer these values between the registers and memory. Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

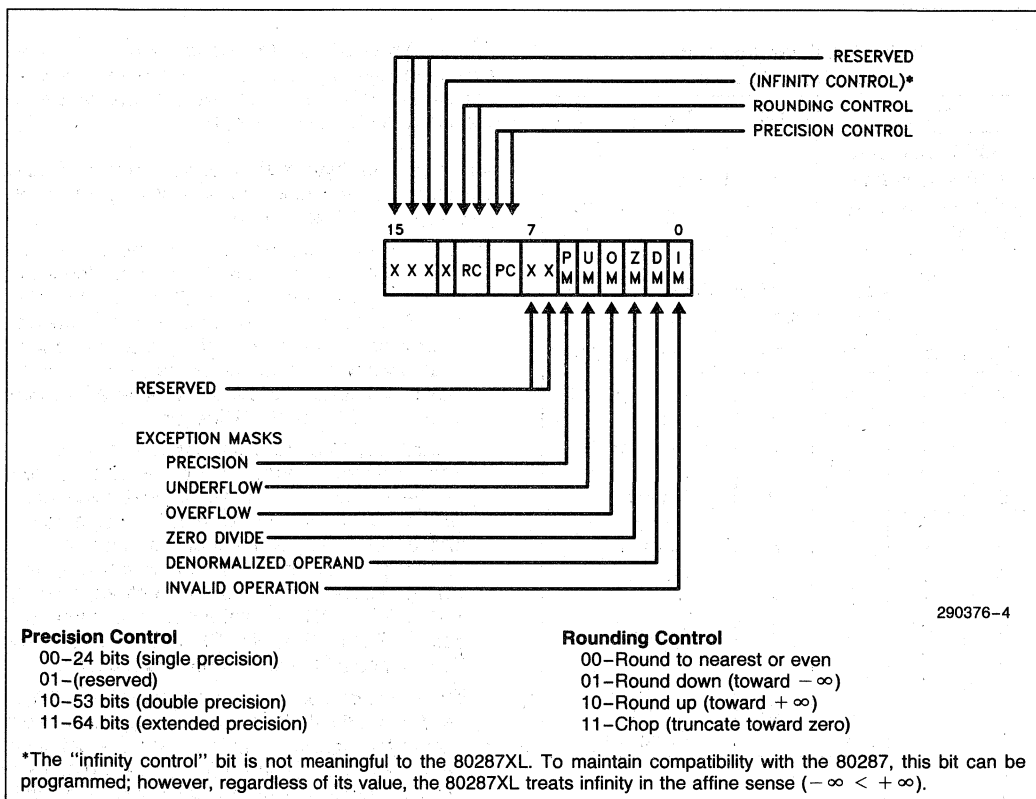


Figure 2.3. Control Word

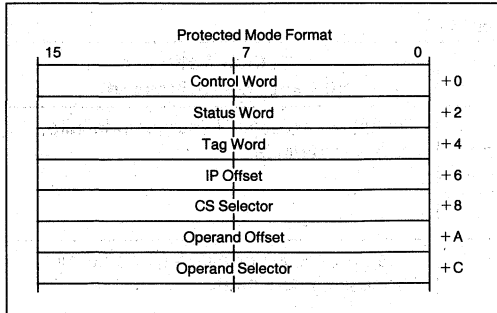


Figure 2.4. Protected Mode Instruction and Data Pointer Image in Memory

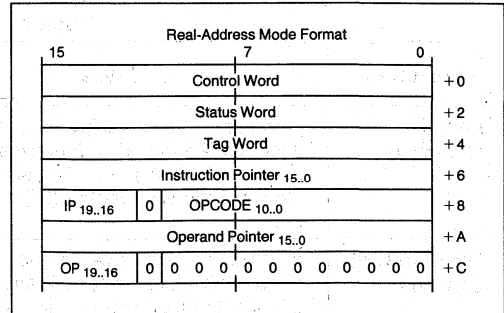


Figure 2.5. Real Mode Instruction and Data Pointer Image in Memory

Table 2.6. CPU Interrupt Vectors Reserved for NPX

Interrupt Number	Cause of Interrupt
7	In a system with a CPU that has control registers, an ESC instruction was encountered when EM or TS of CPU control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction causes interrupt 7. This indicates that the current NPX context may not belong to the current task.
9	In a protected-mode system, an operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for expand-up segments, zero for expand-down segments) and spanned inaccessible addresses ^a . The failing numerics instruction is not restartable. The address of the failing numerics instruction and data operand may be lost; an FSTENV does not return reliable addresses. The segment overrun exception should be handled by executing an FNINIT instruction (i.e., an FINIT without a preceding WAIT). The exception can be avoided by never allowing numerics operands to cross the end of a segment.
13	In a protected-mode system, the first word of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the ESC instruction that caused the exception, including any prefixes. The 80287XL has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC and WAIT instructions can cause this interrupt. The CPU return address pushed onto the stack of the exception handler points to a WAIT or ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the NPX. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.

a. An operand may wrap around an addressing limit when the segment limit is near an addressing limit and the operand is near the largest valid address in the segment. Because of the wrap-around, the beginning and ending addresses of such an operand will be at opposite ends of the segment. There are two ways that such an operand may also span inaccessible addresses: 1) if the segment limit is not equal to the addressing limit (e.g. addressing limit is FFFFH and segment limit is FFFDH) the operand will span addresses that are not within the segment (e.g. an 8-byte operand that starts at valid offset FFFCH will span addresses FFFC–FFFFH and 0000–0003H; however addresses FFFE and FFFFH are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible segments but intermediate bytes of the operand fall in a not-present segment or page or in a segment or page to which the procedure does not have access rights.

2.4 Interrupt Description

CPU interrupts are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 2.6 shows these interrupts and their functions.

2.5 Exception Handling

The 80287XL detects six different exception conditions that can occur during instruction execution. Table 2.7 lists the exception conditions in order of precedence, showing for each the cause and the

Table 2.7. Exceptions

Exception	Cause	Default Action (If Exception is Masked)
Invalid Operation	Operation on a signalling NaN, unsupported format, indeterminate form ($0 \times \infty$, $0/0$, $(+\infty) + (-\infty)$, etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinite.
Denormalized Operand	At least one of the operands is denormalized, i.e., it has the smallest exponent but a nonzero significand.	The operand is normalized, and normal processing continues.
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number.	Result is ∞ .
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞ .
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy.	Result is denormalized or zero.
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g. $1/3$); the result is rounded according to the rounding mode.	Normal processing continues.

default action taken by the 80287XL if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the ERROR# signal. When the CPU attempts to execute another ESC instruction or WAIT, exception 16 occurs. The exception condition must be resolved via an interrupt service routine. The return address pushed onto the CPU stack upon entry to the service routine does not necessarily point to the failing instruction nor to the following instruction. The 80287XL saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.

2.6 Initialization

After FNINIT or RESET, the control word contains the value 037FH (all exceptions masked, precision control 64 bits, rounding to nearest) the same values as in an 80287 after RESET. For compatibility with the 8087 and 80287, the bit that used to indicate infinity control (bit 12) is set to zero; however, regardless of its setting, infinity is treated in the affine

sense. After FNINIT or RESET, the status word is initialized as follows:

- All exceptions are set to zero.
- Stack TOP is zero, so that after the first push the stack top will be register seven (111B).
- The condition code C_3-C_0 is **undefined**.
- The B-bit is zero.

The tag word contains FFFFH (all stack locations are empty).

80286/80287XL initialization software should execute an FNINIT instruction (i.e. an FINIT without a preceding WAIT) after RESET. The FNINIT is not strictly required for either 80287, 80C287A or 80287XL software, but Intel recommends its use to help ensure upward compatibility with other processors.

2.7 8087 and 80287 Compatibility

This section summarizes the differences between the 80287XL and the 80287. Any migration from the 8087 directly to the 80287XL must also take into account the differences between the 8087 and the 80287 as listed in the 80286 and 80287 Programmer's Reference Manual. There are no compatibility differences between the 80287XL and 80C287A except the pinout configuration.

Many changes have been designed into the 80287XL to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

2.7.1 GENERAL DIFFERENCES

The 80287XL supports only affine closure for infinity arithmetic, not projective closure.

Operands for FSCALE and FPATAN are no longer restricted in range (except for $\pm\infty$); F2XM1 and FPTAN accept a wider range of operands.

Rounding control is in effect for FLD *constant*.

Software cannot change entries of the tag word to values (other than empty) that differ from actual register contents.

After reset, FINIT, and incomplete FPREM, the 80287XL resets to zero the condition code bits C₃–C₀ of the status word.

In conformance with the IEEE standard, the 80287XL does not support the special data formats pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

The denormal exception has a different purpose on the 80287XL. A system that uses the denormal-exception handler solely to normalize the denormal operands, would better mask the denormal exception on the 80287XL. The 80287XL automatically normalizes denormal operands when the denormal exception is masked.

2.7.2 EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the 80287XL:

1. When the overflow or underflow exception is masked, the 80287XL differs from the 80287 in rounding when overflow or underflow occurs. The 80287XL produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the 80287XL sets its underflow flag only if there is also a loss of accuracy during denormalization.
3. Fewer invalid-operation exceptions due to denormal operands, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.

4. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
5. The denormal exception can occur during the transcendental instructions and the FEXTRACT instruction.
6. The denormal exception no longer takes precedence over all other exceptions.
7. When the denormal exception is masked, the 80287XL automatically normalizes denormal operands. The 8087/80287 performs unnormal arithmetic, which might produce an unnormal result.
8. When the operand is zero, the FEXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
9. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
10. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
11. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized operand exception. When loading a signalling NaN, FLD *single/double precision* signals an invalid-operation exception.
12. The 80287XL only generates quiet NaNs (as on the 80287); however, the 80287XL distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
13. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 8087/80287 leaves the original operand in ST(1) intact.
14. When the scaling factor is $\pm\infty$, the FSCALE (ST(0), ST(1)) instruction behaves as follows (ST(0) and ST(1) contain the scaled and scaling operands respectively):
 - FSCALE(0, ∞) generates the invalid operation exception.
 - FSCALE(finite, $-\infty$) generates zero with the same sign as the scaled operand.
 - FSCALE(finite, $+\infty$) generates $-\infty$ with the same sign as the scaled operand.

The 8087/80287 returns zero in the first case and raises the invalid-operation exception in the other cases.

15. The 80287XL returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 and 80287 support a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

3.0 HARDWARE INTERFACE

In the following description of hardware interface, the # symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

3.1 Signal Description

In the following signal descriptions, the 80287XL pins are grouped by function as follows:

1. Execution control—CLK, CKM, RESET
2. NPX handshake—PEREQ, PEACK#, BUSY#, ERROR#
3. Bus interface pins—D₁₅–D₀, NPWR#, NPRD#
4. Chip/Port Select—NPS1#, NPS2, CMD0, CMD1
5. Power supplies—V_{CC}, V_{SS}

Table 3.1 lists every pin by its identifier, gives a brief description of its function, and lists some of its characteristics. Figure 3.1 shows the locations of pins on the Ceramic package, while Figure 3.2 shows the locations of pins on the PLCC package. Table 3.2 helps to locate pin identifiers in Figures 3.1 and 3.2.

Table 3.1. Pin Summary

Pin Name	Function	Active State	Input/Output
CLK CKM RESET	CLock Clocking Mode System reset	High	I I I
PEREQ PEACK# BUSY# ERROR#	Processor Extension REQuest Processor Extension ACKnowledge Busy status Error status	High Low Low Low	O I O O
D ₁₅ –D ₀ NPRD# NPWR#	Data pins Numeric Processor ReaD Numeric Processor WRite	High Low Low	I/O I I
NPS1# NPS2 CMD0 CMD1	NPX select #1 NPX select #2 CoMmanD 0 CoMmanD 1	Low High High High	I I I I
V _{CC} V _{SS}	System power System ground		I I

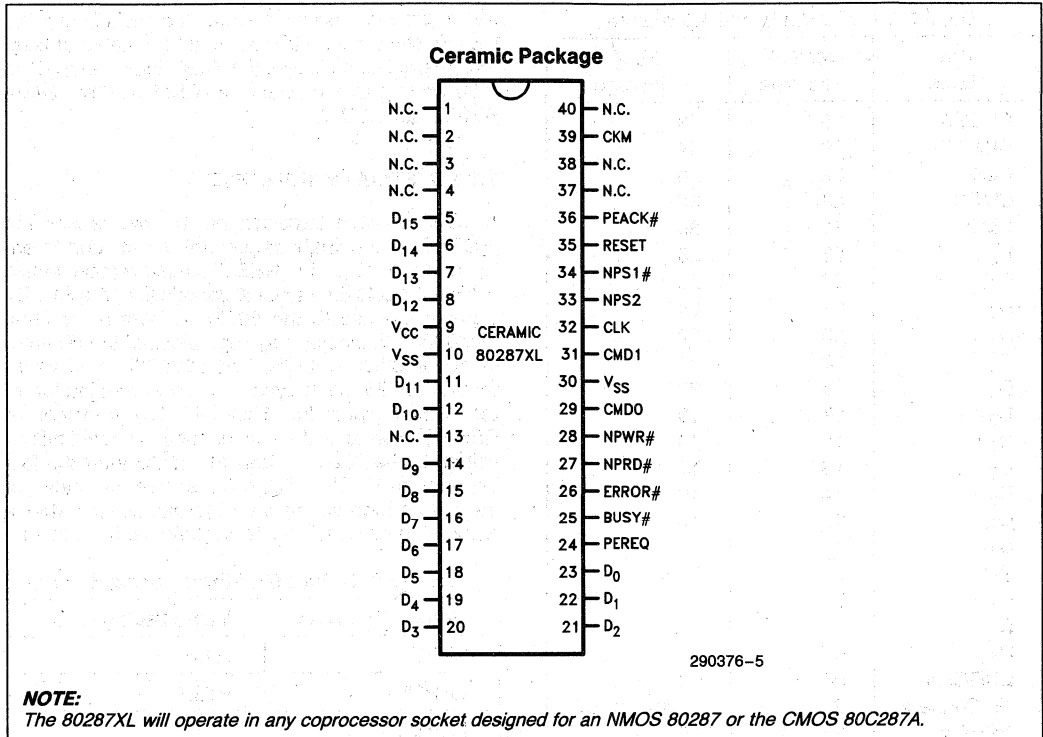


Figure 3.1. DIP Pin Configuration

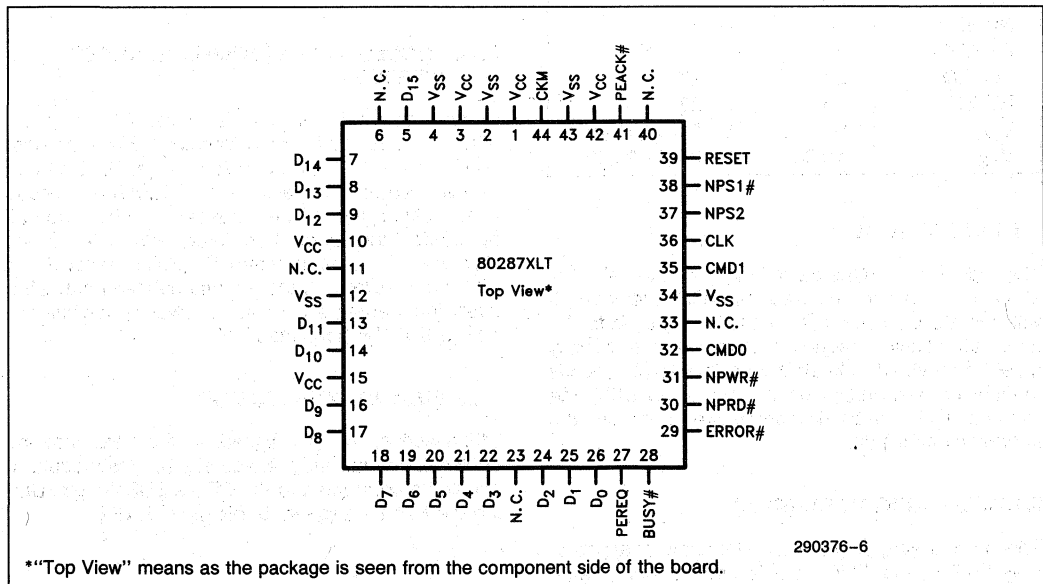


Figure 3.2. PLCC Pin Configuration

Table 3.2. PLCC Pin Cross-Reference

Pin Name	Ceramic Package	PLCC Package
BUSY #	25	28
CKM	39	44
CLK	32	36
CMD0	29	32
CMD1	31	35
D ₀	23	26
D ₁	22	25
D ₂	21	24
D ₃	20	22
D ₄	19	21
D ₅	18	20
D ₆	17	19
D ₇	16	18
D ₈	15	17
D ₉	14	16
D ₁₀	12	14
D ₁₁	11	13
D ₁₂	8	9
D ₁₃	7	8
D ₁₄	6	7
D ₁₅	5	5
ERROR #	26	29
No Connect	2	6,11,23,33,40
NPRD #	27	30
NPS1 #	34	38
NPS2	33	37
NPWR #	28	31
PEACK #	36	41
PEREQ	24	27
RESET	35	39
V _{CC}	9	1,3,10,15,42
V _{SS}	10,30	2,4,12,34,43

3.1.1 CLOCK (CLK)

This input provides the basic timing for internal operation. This pin does not require MOS-level input; it will operate at either TTL or MOS levels up to the maximum allowed frequency. A minimum frequency must be provided to keep the internal logic properly functioning. Depending on the signal on CKM, the signal on CLK can be divided by two to produce the internal clock signal.

3.1.2 CLOCKING MODE (CKM)

This pin is a strapping option. When it is strapped to V_{CC} (HIGH), the CLK input is used directly; when strapped to V_{SS} (LOW), the CLK input is divided by

two to produce the internal clock signal. During the RESET sequence, this input must be stable at least four internal clock cycles (i.e. CLK clocks when CKM is HIGH; $2 \times$ CLK clocks when CKM is LOW) before RESET goes LOW.

3.1.3 SYSTEM RESET (RESET)

A LOW to HIGH transition on this pin causes the 80287XL to terminate its present activity and to enter a dormant state. RESET must remain active (HIGH) for at least four CLK periods (i.e., the RESET signal presented to the 80287XL must be at least four 80287XL clocks long, regardless of the frequency of the CPU). Note that the 80287XL is active internally for 25 clock cycles after the termination of the RESET signal (the HIGH to LOW transition of RESET); therefore, the first instruction should not be written to the 80287XL until 25 clocks after the falling edge of RESET. Table 3.3 shows the status of the output pins during the reset sequence. After a reset, all output pins return to their inactive states.

Table 3.3. Output Pin Status during Reset

Output Pin Name	Value During Reset
BUSY #	HIGH
ERROR #	HIGH
PEREQ	LOW
D ₁₅ -D ₀	Tristate OFF

3.1.4 PROCESSOR EXTENSION REQUEST (PEREQ)

When active, this pin signals to the CPU that the 80287XL is ready for data transfer to/from its data FIFO. With 80286 and 80C286 CPUs, PEREQ can be deactivated after assertion of PEACK#. These CPUs rely on the NPX to deassert PEREQ when all operands have been transferred. When there are more than five data transfers, PEREQ is deactivated after the first three transfers and subsequently after every four transfers. This signal always goes inactive before BUSY # goes inactive.

3.1.5 BUSY STATUS (BUSY#)

When active, this pin signals to the CPU that the 80287XL is currently executing an instruction. It should be connected to the CPU's BUSY # pin. During the RESET sequence this pin is HIGH.

3.1.6 ERROR STATUS (ERROR#)

This pin reflects the ES bit of the status register. When active, it indicates that an unmasked exception has occurred. This signal can be changed to inactive state only by the following instructions (without a preceding WAIT): FNINIT, FNCLEX, FNSTENV, FNSAVE, FLDCW, FLDENV, and FRSTOR. This pin should be connected to the ERROR# pin of the CPU. ERROR# can change state only when BUSY# is active.

3.1.7 PROCESSOR EXTENSION ACKNOWLEDGE (PEACK#)

During execution of escape instructions, an 80286 or 80C286 CPU asserts PEACK# to acknowledge that the request signal (PEREQ) has been recognized and that data transfer is in progress. The 80286/80C286 also drives this signal HIGH during RESET.

This input may be asynchronous with respect to the 80287XL clock except during a RESET sequence, when it must satisfy setup and hold requirements relative to RESET.

3.1.8 DATA PINS (D₁₅-D₀)

These bidirectional pins are used to transfer data and opcodes between the CPU and 80287XL. They are normally connected directly to the corresponding CPU data pins. Other buffers/drivers driving the local data bus must be disabled when the CPU reads from the NPX. HIGH state indicates a value of one. D₀ is the least significant data bit.

3.1.9 NUMERIC PROCESSOR WRITE (NPWR#)

A signal on this pin enables transfers of data from the CPU to the NPX. This input is valid only when NPS1# and NPS2 are both active.

3.1.10 NUMERIC PROCESSOR READ (NPRD#)

A signal on this pin enables transfers of data from the NPX to the CPU. This input is valid only when NPS1# and NPS2 are both active.

3.1.11 NUMERIC PROCESSOR SELECTS (NPS1# and NPS2)

Concurrent assertion of these signals indicates that the CPU is performing an escape instruction and enables the 80287XL to execute that instruction. No data transfer involving the 80287XL occurs unless the device is selected by these lines.

3.1.12 COMMAND SELECTS (CMD0 AND CMD1)

These pins along with the select pins allow the CPU to direct the operation of the 80287XL.

3.1.13 SYSTEM POWER (V_{CC})

System power provides the $+5V \pm 10\%$ DC supply input. All V_{CC} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

3.1.14 SYSTEM GROUND (V_{SS})

All V_{SS} pins should be tied together on the circuit board and local decoupling capacitors should be used between V_{CC} and V_{SS}.

3.2 Processor Architecture

As shown by the block diagram on the front page, the 80287XL NPX is internally divided into three sections: the bus control logic (BCL), the data interface and control unit, and the floating point unit (FPU). The FPU (with the support of the control unit which contains the sequencer and other support units) executes all numerics instructions. The data interface and control unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, and sequencing the microinstructions, and for handling some of the administrative instructions. The BCL is responsible for CPU bus tracking and interface.

3.2.1 BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from memory to the 80287XL and transferring outputs from the 80287XL to memory. A dedicated communication protocol makes possible high-speed transfer of opcodes and operands between the CPU and 80287XL.

3.2.2 DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the FIFO or the instruction decoder. The instruction de-

Table 3.4. Bus Cycles Definition

NPS1 #	NPS2	CMD0	CMD1	NPRD #	NPWR #	Bus Cycle Type
x	0	x	x	x	x	80287XL not selected
1	x	x	x	x	x	80287XL not selected
0	1	0	0	1	0	Opcode write to 80287XL
0	1	0	0	0	1	CW or SW read from 80287XL
0	1	1	0	0	1	Read data from 80287XL
0	1	1	0	1	0	Write data to 80287XL
0	1	0	1	1	0	Write exception pointers
0	1	0	1	0	1	Reserved
0	1	1	1	0	1	Reserved
0	1	1	1	1	0	Reserved

coder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, FSTCW, FSETPM, or FRSTPM, the control executes it independently of the FPU and the sequencer. The data interface and control unit is the one that generates the BUSY#, PEREQ, and ERROR# signals that synchronize 80287XL activities with the CPU.

3.2.3 FLOATING-POINT UNIT

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

3.3 Bus Cycles

The pins NPS1#, NPS2, CMD0, CMD1, NPRD#, and NPWR# identify bus cycles for the NPX. Table 3.4 defines the types of 80287XL bus cycles.

3.3.1 80287XL ADDRESSING

The NPS1#, NPS2, CMD0, and CMD1 signals allow the NPX to identify which bus cycles are intended for the NPX. The NPX responds to I/O cycles when the I/O address is 00F8H, 00FAH, 00FCH. The correspondence between I/O addresses and control signals is defined by Table 3.5. To guarantee correct operation of the NPX, programs must not perform any I/O operations to these reserved port addresses.

Table 3.5. I/O Address Decoding

I/O Address (Hexadecimal)	80287XL Select and Command Inputs			
	NPS2	NPS1 #	CMD1	CMD0
00F8	1	0	0	0
00FA	1	0	0	1
00FC	1	0	1	0

3.3.2 CPU/NPX SYNCHRONIZATION

The pins BUSY#, PEREQ, and ERROR# are used for various aspects of synchronization between the CPU and the NPX.

BUSY# is used to synchronize instruction transfer from the CPU to the 80287XL. When the 80287XL recognizes an ESC instruction, it asserts BUSY#. For most ESC instructions, the CPU waits for the 80287XL to deassert BUSY# before sending the new opcode.

The NPX uses the PEREQ pin of the CPU to signal that the NPX is ready for data transfer to or from its data FIFO. The NPX does not directly access memory; rather, the CPU provides memory access services for the NPX. Thus, memory access on behalf of the NPX always obeys the rules applicable to the mode of the CPU, whether the CPU be in real-address mode or protected mode.

Once the CPU initiates an 80287XL instruction that has operands, the CPU waits for PEREQ signals that indicate when the 80287XL is ready for operand transfer. Once all operands have been transferred

(or if the instruction has no operands) the CPU continues program execution while the 80287XL executes the ESC instruction.

In 8086/8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In 80287XL systems, however, WAIT instructions are required only for operand synchronization; namely, after NPX stores to memory (except FSTSW and FSTCW) or load from memory. (In 80286/80287XL systems, WAIT is required before FLDENV and FRSTOR; with other CPU's, WAIT is not required in these cases.) Used this way, WAIT ensures that the value has already been written or read by the NPX before the CPU reads or changes the value.

Once it has started to execute a numerics instruction and has transferred the operands from the CPU, the 80287XL can process the instruction in parallel with and independent of the host CPU. When the NPX detects an exception, it asserts the ERROR# signal, which causes a CPU interrupt.

3.4 Bus Operation

With respect to bus interface, the 80287XL is fully asynchronous with the CPU, even when it operates from the same clock source as the CPU. The CPU initiates a bus cycle for the NPX by activating both NPS1# and NPS2, the NPX select signals. During the CLK period in which NPS1# and NPS2 are activated, the 80287XL also examines the NPRD# and NPWR# input signals to determine whether the cycle is a read or a write cycle and examines the CMD0 and CMD1 inputs to determine whether an opcode, operand, or control/status register transfer is to occur. The 80287XL activates its BUSY# output some time after the leading edge of the NPRD# or NPWR# signal. Input and output data are referenced to the trailing edges of the NPRD# and NPWR# signals.

The 80287XL activates the PEREQ signal when it is ready for data transfer. In 80286/80C286 systems, the CPU activates PEACK# when no more data transfers are required, which causes the 80287XL to deactivate PEREQ, halting the data transfer.

3.5 80286/80287XL, 80C286/80287XL Interface and Socket Compatibility

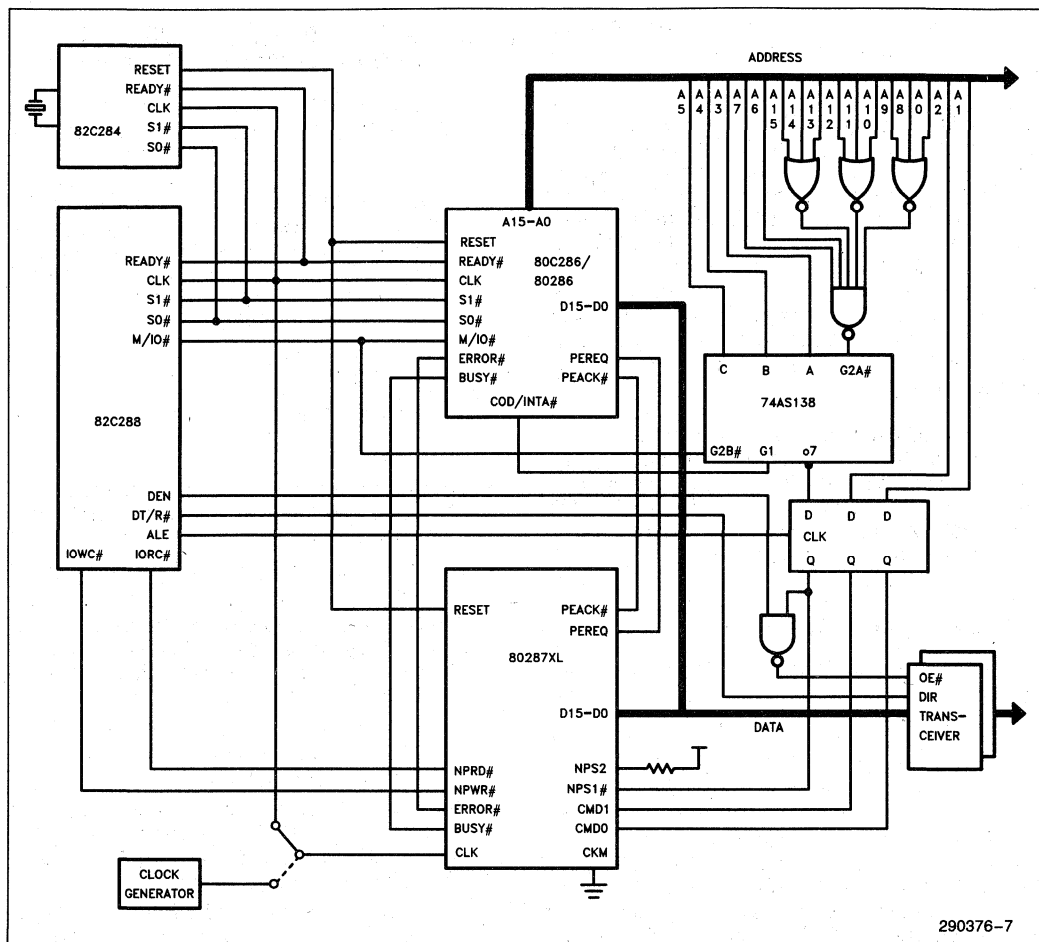
The ceramic 80287XL device can fit into existing 80287 sockets since the pin configuration is identical.

The Cerdip 80C287A utilizes a different pin configuration with extra power and ground pins. However, the 80287XL operates in 80C287A sockets also. The extra power and ground pins are not connected inside the 80287XL and not used. Refer to 80C287A data sheet (Order #240347).

Note that when the clock selection is CKM = 0, the 80287XL divides the clock input by two, not by three as on the 80287. In this case, the 80287XL will operate faster.

The interface between the 80287XL and the 80286/80C286 CPU (illustrated in Figure 3.3) has these characteristics:

- The 80287XL resides on the local data bus of the CPU.
- The CPU and 80287XL share the same RESET signals. They may also share the same clock input; however, for greatest performance, an external oscillator may be needed.
- The corresponding BUSY#, ERROR#, PEREQ, and PEACK# pins are connected together.
- NPS2 is tied HIGH permanently, while NPS1#, CMD1, and CMD0 come from the latched address pins. The 80286 generates I/O addresses 00F8H, 00FAH, and 00FCH during NPX bus cycles. Address 00FEH is reserved.
- The 80287XL NPRD# and NPWR# inputs are connected to I/O read and write signals from local bus control logic.



290376-7

Figure 3.3. 80286/80287XL System Configuration

4.0 ELECTRICAL DATA

4.1 Absolute Maximum Ratings

NOTE

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

Case temperature T_C under bias 0°C to 85°C

Storage temperature -65°C to $+150^{\circ}\text{C}$

Voltage on any pin
with respect to ground -0.5 to $V_{CC} + 0.5\text{V}$

Power dissipation 1.5 Watt

4.2 Power and Frequency Requirements

The typical relationship between I_{CC} and the frequency of operation F is as follows:

$$I_{CC\text{typ}} = 55 + 5 \cdot F \text{ mA, where } F \text{ is in MHz.}$$

When the frequency is reduced below the minimum operating frequency specified in the AC Characteristics table, the internal states of the 80287XL may become indeterminate. The 80287XL clock cannot be stopped; otherwise, I_{CC} would increase significantly beyond what the equation above indicates. Power dissipation decreases with frequency for frequencies ≥ 4 MHz.

4.3 D.C. Characteristics

Table 4.1. D.C. Specifications $T_C = 0$ to 85°C , $V_{CC} = 5\text{V} \pm 10\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LOW Voltage	-0.5	$+0.8$	V	
V_{IH}	Input HIGH Voltage	2.0	$V_{CC} + 0.5$	V	
V_{ICL}	Clock Input LOW Voltage	-0.5	$+0.8$	V	
V_{ICH}	Clock Input HIGH Voltage	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output LOW Voltage		0.45	V	$I_{OL} = 3.0 \text{ mA}$
V_{OH}	Output HIGH Voltage	2.4		V	$I_{OH} = -0.4 \text{ mA}$
I_{CC}	Power Supply Current		135	mA	Note 1
I_{LI}	Input Leakage Current		± 10	μA	Note 2
I_{LO}	I/O Leakage Current		± 10	μA	Note 3
C_{IN}	Input Capacitance		10	pF	Note 4
C_O	I/O or Output Capacitance		12	pF	Note 4
C_{CLK}	Clock Capacitance		20	pF	Note 4

NOTES:

1. 12.5 MHz operation, output load = 100 pF

2. $0\text{V} \leq V_{IN} \leq V_{CC}$

3. $0.45\text{V} \leq V_{OUT} \leq V_{CC} - 0.45$

4. $F_C = 1\text{MHz}$

4.4 A.C. Characteristics

Table 4.2. Timing Requirements $T_C = 0$ to 85 deg C, $V_{CC} = 5V \pm 10\%$
All timings are measured at 1.5V unless otherwise specified

Symbol	Parameter	12.5 MHz		Test Conditions
		Min (ns)	Max (ns)	
Tdwh (t6)	Data setup to NPWR #	43		
Twhdx (t7)	Data hold from NPWR #	14		
Trlrh (t8)	NPRD # active time	59		
Twlwh (t9)	NPWR # active time	59		
Tavwl (t10)	Command valid to NPWR #	0		
Tavrl (t11)	Command valid to NPRD #	0		
Tmhr1 (t12)	Min delay from PEREQ active to NPRD # active	40		
Tklkh (t33)	PEACK # active time	55		
Tkhkl (t34)	PEACK # inactive time	60		
Tkhch (t35)	PEACK # inactive to NPRD #, NPWR # inactive	30		
Tklcl (t36)	PEACK # active setup to NPRD #, NPWR # active	30		
Tchkl (t37)	NPRD #, NPWR # inactive to PEACK # active	-30		
Twhax (t18)	Command hold from NPWR #	12		
Trhax (t19)	Command hold from NPRD #	12		
Tivcl (t20)	NPRD #, NPWR #, RESET to CLK setup time	46		Note 1
Tclih (t21)	NPRD #, NPWR #, RESET from CLK hold time	26		Note 1
Tpaksu (t38)	PEACK # setup to RESET falling edge	80		
Tpakhd (t39)	PEACK # hold from RESET falling edge	80		
Trscl (t24)	RESET to CLK setup	21		Note 1
Tclrs (t25)	RESET from CLK hold	14		Note 1
Tcmdi (t26)	Command inactive time			
	Write to write	69		
	Read to read	69		
	Read to write	69		
	Write to read	69		

NOTE:

1. This is an asynchronous input. This specification is given for testing purposes only, to assure recognition at a specific CLK edge (not tested).

Table 4.3. Timing Responses

Symbol	Parameter	12.5 MHz		Test Conditions
		Min (ns)	Max (ns)	
Trhqz (t27)	NPRD # inactive to data float*		18	Note 2
Trlqv (t28)	NPRD # active to data valid		50	Note 3
Tilbh (t29)	ERROR # active to BUSY # inactive	104		Note 4
Twlbv (t30)	NPWR # active to BUSY # active		80	Note 4
Tklml (t31)	NPRD #, NPWR # or PEACK # active to PEREQ inactive		80	Note 5
Trhgh (t32)	Data hold from NPRD # inactive	2		Note 3

NOTES:

- * The data float delay is not tested.
2. The float condition occurs when the measured output current is less than I_{OL} on $D_{15}-D_0$.
3. $D_{15}-D_0$ loading: $C_1 = 100\text{pf}$.
4. BUSY # loading: $C_1 = 100\text{pf}$.
5. On last data transfer of numeric instruction.

Table 4.4. Clock Timings

Symbol	Parameter	12.5 MHz		Test Conditions
		Min (ns)	Max (ns)	
Tclcl (t1a)	CLK period	CKM = 1	80	Note 6, 10 $V_{CC} = \pm 10\%$ $V_{CC} = \pm 5\%$, Note 11 Note 7, 10 Note 8 Note 9
(t1b)		CKM = 0	40	
Tclch (t2a)	CLK low time	CKM = 1	35	
(t2b)		CKM = 0	9	
Tchcl (t3a)	CLK high time	CKM = 1	35	
		CKM = 1	28	
(t3b)		CKM = 0	13	
Tch1ch2 (t4)			10	
Tch2ch1 (t5)			10	

NOTES:

6. At 0.8V.
7. At 2.0V.
8. CKM = 1: 3.5V to 1.0V
9. CKM = 1: 1.0V to 3.5V
10. Proper operation can also be achieved by meeting the CPU specification
11. Provides compatibility for sockets designed for Intel 80287-6/8/10 MHz Math CoProcessors.

3

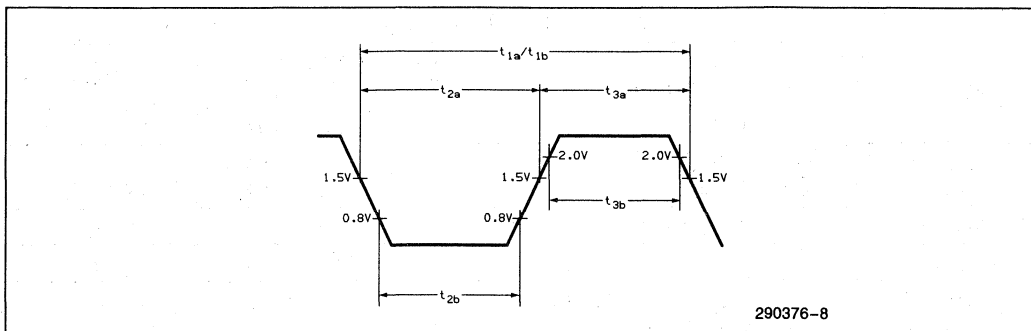


Figure 4.1. AC Drive and Measurement Points—CLK Input

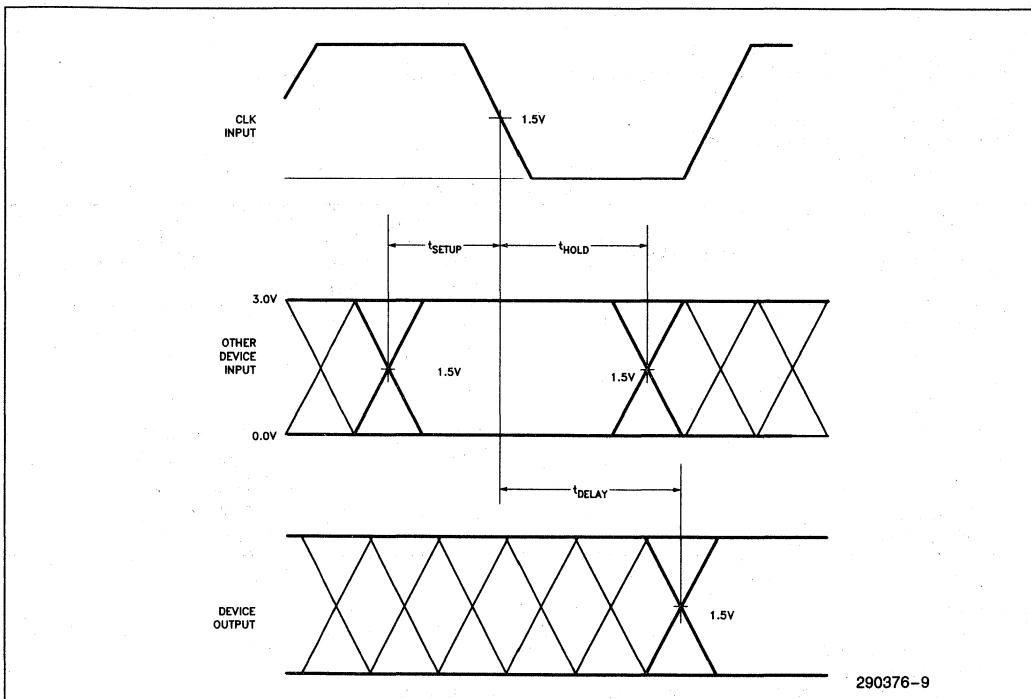


Figure 4.2. AC Setup, Hold, and Delay Time Measurements—General

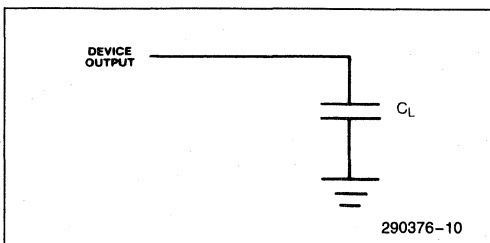
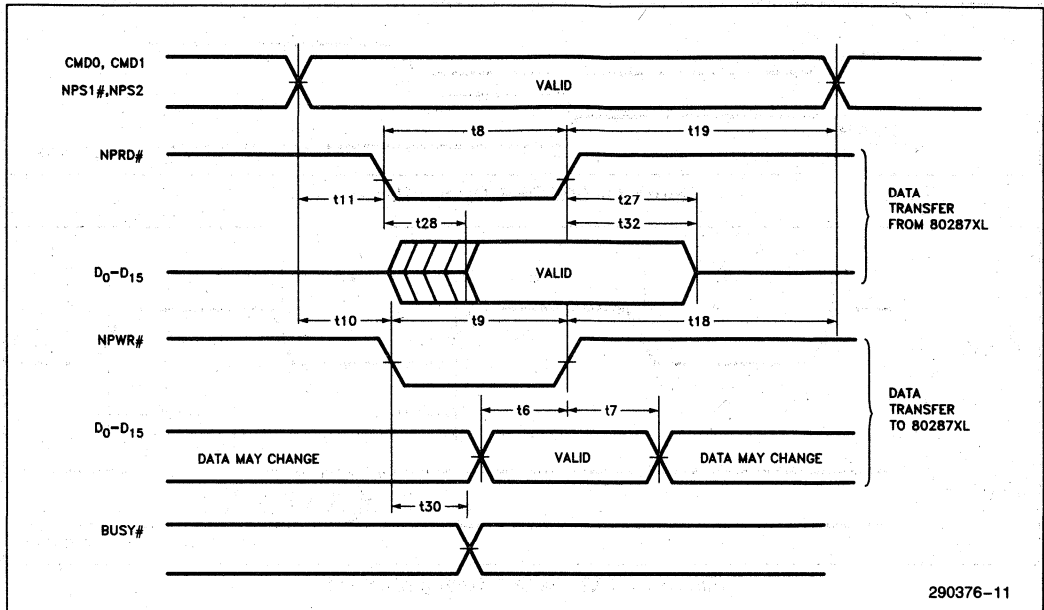
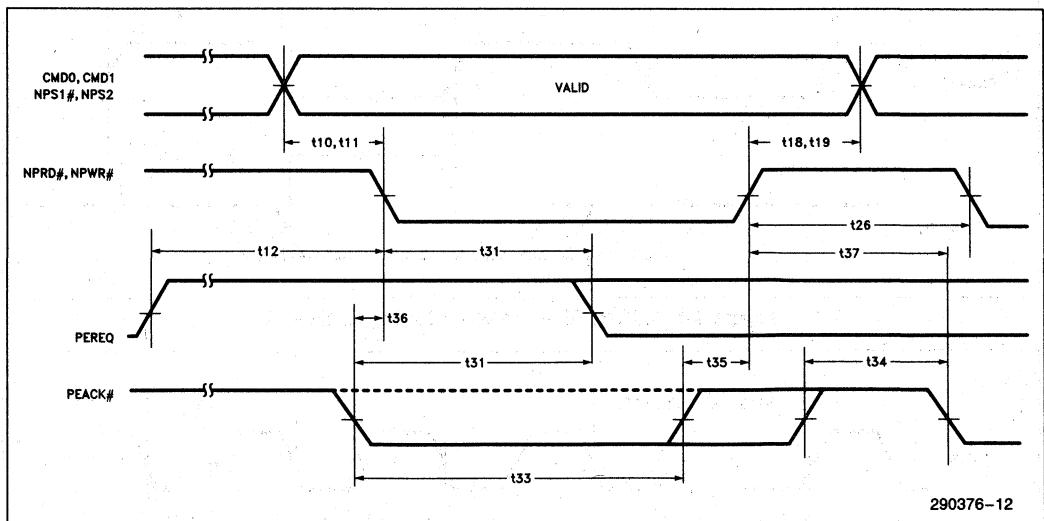


Figure 4.3. AC Test Loading on Outputs

RESET, NPWR#, NPRD# inputs are asynchronous to CLK. Timing requirements in Figures 4.7 through 4.10 are given for testing purposes only, to assure recognition at a specific CLK edge.



3



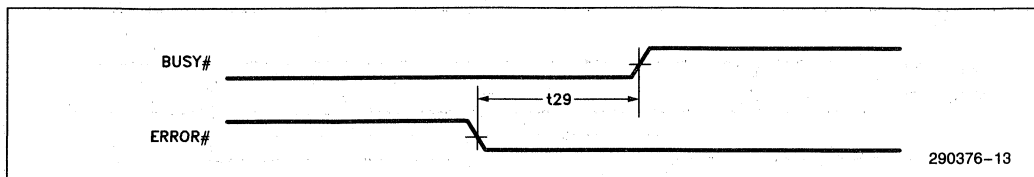


Figure 4.6. ERROR# Output Timing

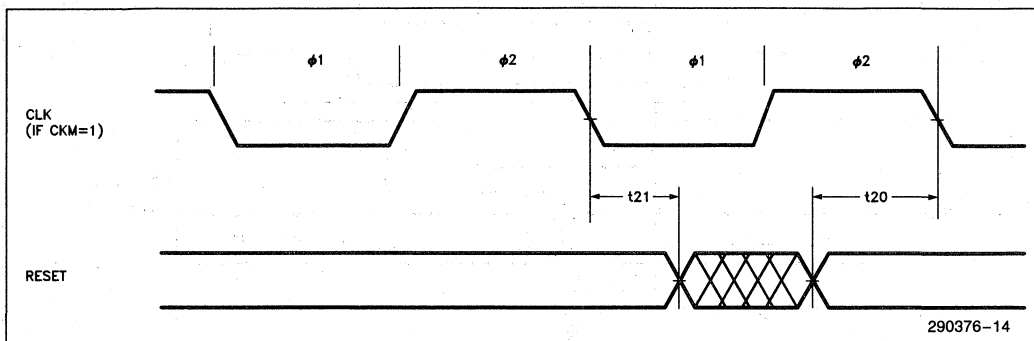


Figure 4.7. CLK, RESET Timing (CKM = 1)

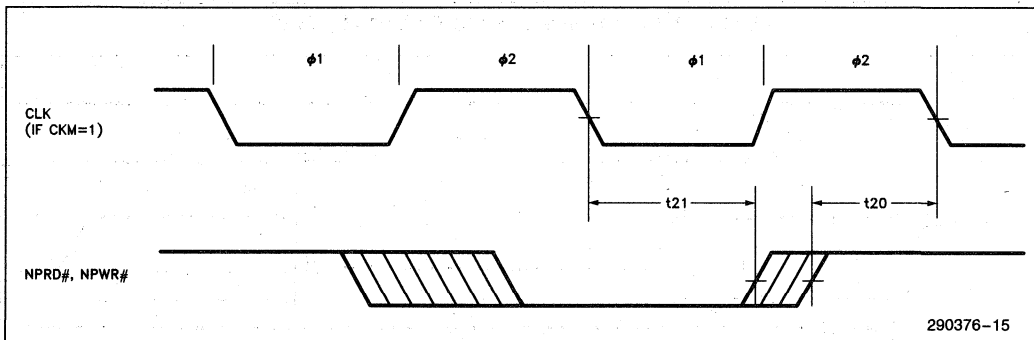
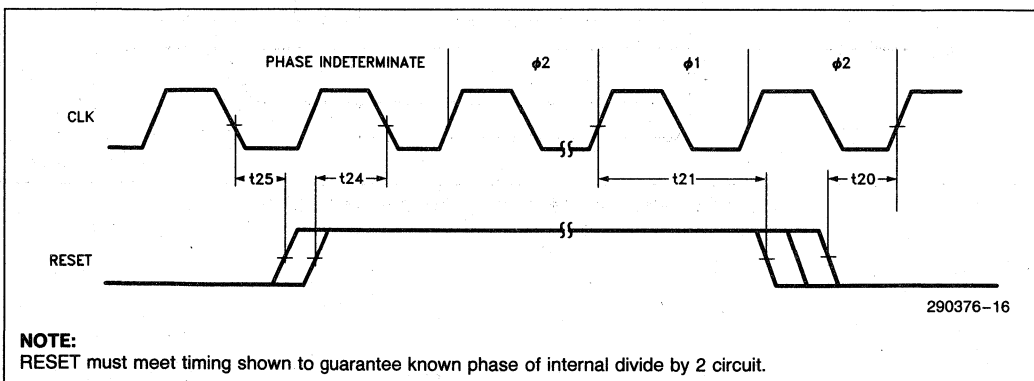


Figure 4.8. CLK, NPRD#, NPWR# Timing (CKM = 1)



NOTE:

RESET must meet timing shown to guarantee known phase of internal divide by 2 circuit.

Figure 4.9. CLK, RESET Timing (CKM = 0)

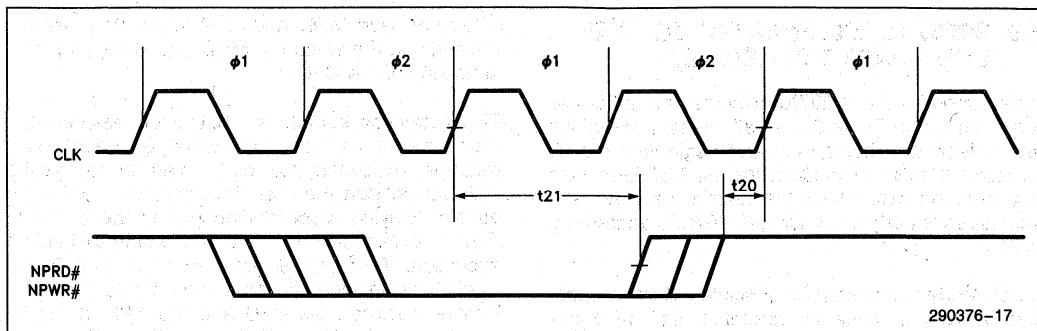


Figure 4.10. CLK, NPRD#, NPWR# Timing (CKM = 0)

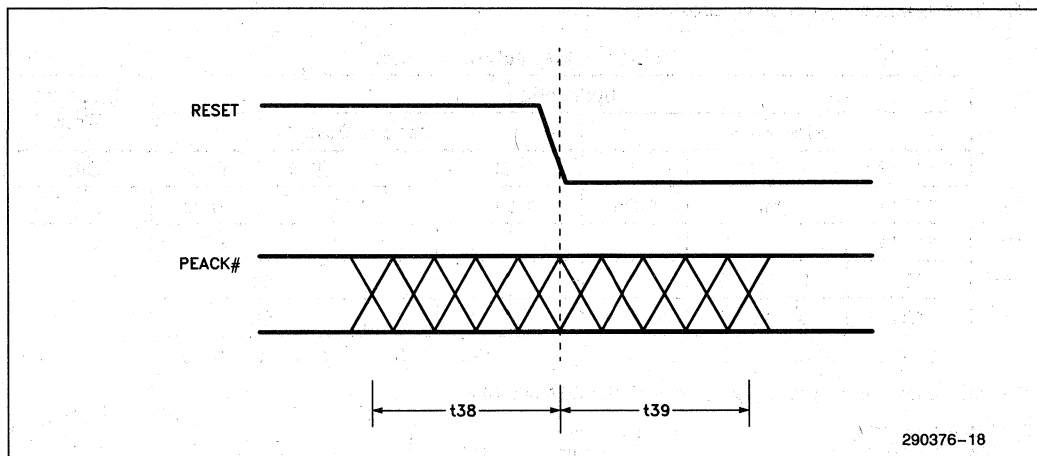


Figure 4.11. RESET, PEACK# Setup and Hold Timing

5.0 80287XL EXTENSIONS TO THE CPU'S INSTRUCTION SET

Instructions for the 80287XL assume one of the five forms shown in Table 5-1. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the CPU's addressing modes.

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of CPU instructions (refer to Programmer's Reference Manual for the CPU). The DISP (displacement) is optionally present in instruc-

tions that have MOD and R/M fields. Its presence depends on the values of MOD and R/M, as for instructions of the CPU.

The instruction summaries that follow assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD requests delaying processor access to the bus; and that no exceptions are detected during instruction execution. Timings are given in internal 80287XL clocks and include the time for opcode and data transfer between the CPU and the NPX. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

Table 5.1. Instruction Formats

Instruction										Optional Field	
First Byte					Second Byte						
1	11011		OPA		1	MOD		1	OPB	R/M	DISP
2	11011		MF		OPA	MOD		OPB*		R/M	DISP
3	11011		d	P	OPA	1	1	OPB*		ST(i)	
4	11011		0	0	1	1	1	1	OP		
5	11011		0	1	1	1	1	1	OP		
	15-11	10	9	8	7	6	5	4	3	2	1 0

OP = Instruction opcode, possibly split into two fields OPA and OPB

MF = Memory Format
 00-32-bit real
 01-32-bit integer
 10-64-bit real
 11-16-bit integer

d = Destination
 0-Destination is ST(0)
 1-Destination is ST(i)
 R XOR d = 0-Destination (Op) Source
 R XOR d = 1-Source (Op) Destination

*In FSUB and FDIV, the low-order bit of the OPB is the R (reversed) bit

P = Pop
 0-Do not pop stack
 1-Pop stack after operation

ST(i) = Register stack element i
 000 = Stack top
 001 = Second stack element

ESC = 11011

•
 •
 •
 111 = Eighth stack element

80287XL Extension to the CPU's Instruction Set

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2-3	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
DATA TRANSFER							
FLD = Load ^a							
Integer/real memory to ST(0)	ESC MF 1	MOD 000 R/M	SIB/DISP	36	61-68	45	61-65
Long integer memory to ST(0)	ESC 111	MOD 101 R/M	SIB/DISP		76-87		
Extended real memory to ST(0)	ESC 011	MOD 101 R/M	SIB/DISP		48		
BCD memory to ST(0)	ESC 111	MOD 100 R/M	SIB/DISP		270-279		
ST(i) to ST(0)	ESC 001	11000 ST(i)			21		
FST = Store							
ST(0) to integer/real memory	ESC MF 1	MOD 010 R/M	SIB/DISP	51	86-100	56	88-101
ST(0) to ST(i)	ESC 101	11010 ST(i)			18		
FSTP = Store and Pop							
ST(0) to integer/real memory	ESC MF 1	MOD 011 R/M	SIB/DISP	51	86-100	56	88-101
ST(0) to long integer memory	ESC 111	MOD 111 R/M	SIB/DISP		91-108		
ST(0) to extended real	ESC 011	MOD 111 R/M	SIB/DISP		61		
ST(0) to BCD memory	ESC 111	MOD 110 R/M	SIB/DISP		520-542		
ST(0) to ST(i)	ESC 101	11001 ST(i)			19		
FXCH = Exchange							
ST(i) and ST(0)	ESC 001	11001 ST(i)			25		
COMPARISON							
FCOM = Compare							
Integer/real memory to ST(0)	ESC MF 0	MOD 010 R/M	SIB/DISP	42	72-79	51	71-75
ST(i) to ST(0)	ESC 000	11010 ST(i)			31		
FCOMP = Compare and pop							
Integer/real memory to ST	ESC MF 0	MOD 011 R/M	SIB/DISP	42	72-79	51	71-77
ST(i) to ST(0)	ESC 000	11011 ST(i)			33		
FCOMPP = Compare and pop twice							
ST(1) to ST(0)	ESC 110	1101 1001			33		
FTST = Test ST(0)							
	ESC 001	1110 0100			35		
FUCOM = Unordered compare							
	ESC 101	11100 ST(i)			31		
FUCOMP = Unordered compare and pop							
	ESC 101	11101 ST(i)			33		
FUCOMPP = Unordered compare and pop twice							
	ESC 010	1110 1001			33		
FXAM = Examine ST(0)							
	ESC 001	11100101			37-45		
CONSTANTS							
FLDZ = Load +0.0 into ST(0)	ESC 001	1110 1110			27		
FLD1 = Load +1.0 into ST(0)	ESC 001	1110 1000			31		
FLDPI = Load pi into ST(0)	ESC 001	1110 1011			47		
FLDL2T = Load log ₂ (10) into ST(0)	ESC 001	1110 1001			47		

Shaded areas indicate instructions not available in 8087/80287, but available on 80C287A and 80287XL.

NOTE:

a. When loading single- or double-precision zero from memory, add 5 clocks.

80287XL Extension to the CPU's Instruction Set (Continued)

Instruction	Encoding			Clock Count Range			
	Byte 0	Byte 1	Optional Bytes 2-3	32-Bit Real	32-Bit Integer	64-Bit Real	16-Bit Integer
CONSTANTS (Continued)							
FLDL2E = Load $\log_2(e)$ into ST(0)	ESC 001	1110 1010			47		
FLDLG2 = Load $\log_{10}(2)$ into ST(0)	ESC 001	1110 1100			48		
FLDLN2 = Load $\log_e(2)$ into ST(0)	ESC 001	1110 1101			48		
ARITHMETIC							
FADD = Add							
Integer/real memory with ST(0)	ESC MF 0	MOD 000 R/M	SIB/DISP	40-48	73-78	49-79	71-85
ST(i) and ST(0)	ESC d P 0	11000 ST(i)			30-38 ^b		
FSUB = Subtract							
Integer/real memory with ST(0)	ESC MF 0	MOD 10 R R/M	SIB/DISP	40-48	73-98	49-77	71-83 ^c
ST(i) and ST(0)	ESC d P 0	1110 R R/M			33-41 ^d		
FMUL = Multiply							
Integer/real memory with ST(0)	ESC MF 0	MOD 001 R/M	SIB/DISP	43-51	77-88	52-77	76-87
ST(i) and ST(0)	ESC d P 0	1100 1 R/M			25-53 ^e		
FDIV = Divide							
Integer/real memory with ST(0)	ESC MF 0	MOD 11 R R/M	SIB/DISP	105	136-143 ^f	114	136-140 ^g
ST(i) and ST(0)	ESC d P 0	1111 R R/M			95 ^h		
FSQRT = Square root	ESC 001	1111 1010			129-136		
FSCALE = Scale ST(0) by ST(1)	ESC 001	1111 1101			74-93		
FPPREM = Partial remainder of ST(0) ÷ ST(1)	ESC 001	1111 1000			81-162		
FPPREM1 = Partial remainder (IEEE)	ESC 001	1111 0101			102-192		
FRNDINT = Round ST(0) to Integer	ESC 001	1111 1100			73-87		
FXTRACT = Extract components of ST(0)	ESC 001	1111 0100			75-83		
FABS = Absolute value of ST(0)	ESC 001	1110 0001			29		
FCHS = Change sign of ST(0)	ESC 001	1110 0000			31-37		

Shaded areas indicate instructions not available in 8087/80287, but available on 80287XL and 80C287A.

NOTES:

- Add 3 clocks to the range when $d = 1$.
- Add 1 clock to **each** range when $R = 1$.
- Add 3 clocks to the range when $d = 0$.
- Typical = 48 (When $d = 0$, 42-50, typical = 45).
- Add 1 clock to the range when $R = 1$.
- 135-141 when $R = 1$.
- Add 3 clocks to the range when $d = 1$.
- $-0 \leq ST(0) \leq +\infty$.

80287XL Extension to the CPU's Instruction Set (Continued)

Instruction	Encoding			Clock Count Range
	Byte 0	Byte 1	Optional Bytes 2-3	
TRANSCENDENTAL				
FCOS = Cosine of ST(0)	ESC 001	1111 1111		130-779i
FPTAN ^k = Partial tangent of ST(0)	ESC 001	1111 0010		198-504i
FPATAN = Partial arctangent	ESC 001	1111 0011		321-494
FSIN = Sine of ST(0)	ESC 001	1111 1110		129-778i
FSINCOS = Sine and cosine of ST(0)	ESC 001	1111 1011		201-816i
F2XM1 ^l = 2 ^{ST(0)} - 1	ESC 001	1111 0000		215-483
FYL2XM ^m = ST(1) * log ₂ (ST(0))	ESC 001	1111 0001		127-545
FYL2XP1 ⁿ = ST(1) * log ₂ (ST(0) + 1.0)	ESC 001	1111 1001		264-554
PROCESSOR CONTROL				
FINIT = Initialize NPX	ESC 011	1110 0011		25
FSETPM = Set protected mode	ESC 011	1110 0100		12
FRSTPM = Reset protected mode	ESC 011	1111 0100		12
FSTSW AX = Store status word	ESC 111	1110 0000		18
FLDCW = Load control word	ESC 001	MOD 101 R/M	SIB/DISP	33
FSTCW = Store control word	ESC 101	MOD 111 R/M	SIB/DISP	18
FSTSW = Store status word	ESC 101	MOD 111 R/M	SIB/DISP	18
FCLEX = Clear exceptions	ESC 011	1110 0010		8
FSTENV = Store environment	ESC 001	MOD 110 R/M	SIB/DISP	192-193
FLDENV = Load environment	ESC 001	MOD 100 R/M	SIB/DISP	85
FSAVE = Save state	ESC 101	MOD 110 R/M	SIB/DISP	521-522
FRSTOR = Restore state	ESC 101	MOD 100 R/M	SIB/DISP	396
FINCSTP = Increment stack pointer	ESC 001	1111 0111		28
FDECSTP = Decrement stack pointer	ESC 001	1111 0110		29
FFREE = Free ST(i)	ESC 101	1100 0 ST(i)		25
FNOP = No operations	ESC 001	1101 0000		19

Shaded areas indicate instructions not available in 8087/80287, but available on 80287XL and 80C287A.

NOTES:

j. These timings hold for operands in the range $|x| < \pi/4$. For operands not in this range, up to 78 additional clocks may be needed to reduce the operand.

k. $0 \leq |ST(0)| < 2^{63}$.

l. $-1.0 \leq ST(0) \leq 1.0$.

m. $0 \leq ST(0) < \infty$, $-\infty < ST(1) < +\infty$.

n. $0 \leq |ST(0)| < (2 - \text{SQRT}(2))/2$, $-\infty < ST(1) < +\infty$.



82C288

BUS CONTROLLER FOR 80286 PROCESSORS (82C288-12, 82C288-10, 82C288-8)

- Provides Commands and Controls for Local and System Bus
 - Wide Flexibility in System Configurations
 - High Speed CHMOS III Technology
 - Fully Compatible with the HMOS 82288
 - Fully Static Device
 - Available in 20 Pin PLCC (Plastic Leaded Chip Carrier) and 20 Pin Cerdip Packages
- (See Packaging Spec, Order #231369)

The Intel 82C288 Bus Controller is a 20-pin CHMOS III component for use in 80286 microsystems. The 82C288 is fully compatible with its predecessor the HMOS 82288. The bus controller is fully static and supports a low power mode. The bus controller provides command and control outputs with flexible timing options. Separate command outputs are used for memory and I/O devices. The data bus is controlled with separate data enable and direction control signals.

Two modes of operation are possible via a strapping option: MULTIBUS® I compatible bus cycles, and high speed bus cycles.

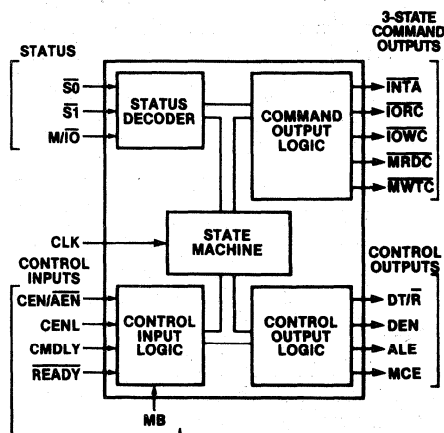
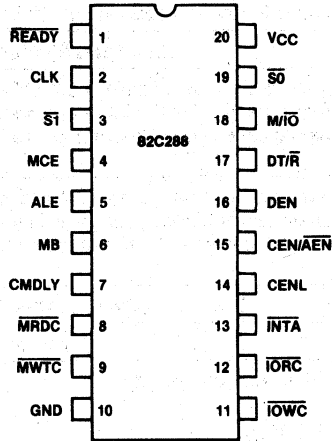


Figure 1. 82C288 Block Diagram

240042-1

20 Pin Cerdip Package

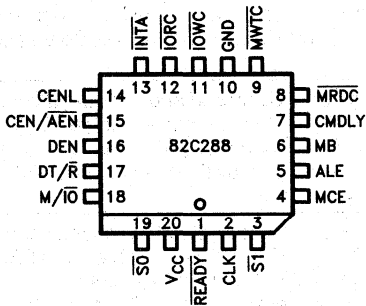


240042-2

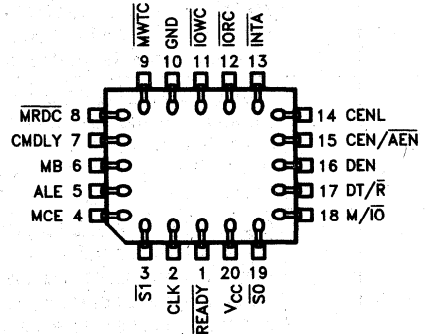
P.C. Board Views—As viewed from the component side of the P.C. board.

Component Pad Views—As viewed from under-side of component when mounted on the board.

20 Pin PLCC Package



240042-3



240042-4

Figure 2. 82C288 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for the 82C288 bus controller.

Symbol	Type	Name and Function			
CLK	I	SYSTEM CLOCK provides the basic timing control for the 82C288 in an 80286 microsystem. Its frequency is twice the internal processor clock frequency. The falling edge of this input signal establishes when inputs are sampled and command and control outputs change.			
$\overline{S0}, \overline{S1}$	I	BUS CYCLE STATUS starts a bus cycle and, along with M/\overline{IO} , defines the type of bus cycle. These inputs are active LOW. A bus cycle is started when either $\overline{S1}$ or $\overline{S0}$ is sampled LOW at the falling edge of CLK. Setup and hold times must be met for proper operation.			
80286 Bus Cycle Status Definition					
		M/\overline{IO}	$\overline{S1}$	$\overline{S0}$	Type of Bus Cycle
		0	0	0	Interrupt Acknowledge
		0	0	1	I/O Read
		0	1	0	I/O Write
		0	1	1	None; Idle
		1	0	0	Halt or Shutdown
		1	0	1	Memory Read
		1	1	0	Memory Write
		1	1	1	None; Idle
M/\overline{IO}	I	MEMORY OR I/O SELECT determines whether the current bus cycle is in the memory space or I/O space. When LOW, the current bus cycle is in the I/O space. Setup and hold times must be met for proper operation.			
MB	I	MULTIBUS MODE SELECT determines timing of the command and control outputs. When HIGH, the bus controller operates with MULTIBUS I compatible timings. When LOW, the bus controller optimizes the command and control output timing for short bus cycles. The function of the CEN/AEN input pin is selected by this signal. This input is typically a strapping option and not dynamically changed.			
CENL	I	COMMAND ENABLE LATCHED is a bus controller select signal which enables the bus controller to respond to the current bus cycle being initiated. CENL is an active HIGH input latched internally at the end of each T_S cycle. CENL is used to select the appropriate bus controller for each bus cycle in a system where the CPU has more than one bus it can use. This input may be connected to V_{CC} to select this 82C288 for all transfers. No control inputs affect CENL. Setup and hold times must be met for proper operation.			
CMDLY	I	COMMAND DELAY allows delaying the start of a command. CMDLY is an active HIGH input. If sampled HIGH, the command output is not activated and CMDLY is again sampled at the next CLK cycle. When sampled LOW the selected command is enabled. If \overline{READY} is detected LOW before the command output is activated, the 82C288 will terminate the bus cycle, even if no command was issued. Setup and hold times must be satisfied for proper operation. This input may be connected to GND if no delays are required before starting a command. This input has no effect on 82C288 control outputs.			
READY	I	READY indicates the end of the current bus cycle. \overline{READY} is an active LOW input. MULTIBUS I mode requires at least one wait state to allow the command outputs to become active. \overline{READY} must be LOW during reset, to force the 82C288 into the idle state. Setup and hold times must be met for proper operation. The 82C284 drives \overline{READY} LOW during RESET.			

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
CEN/AEN	I	<p>COMMAND ENABLE/ADDRESS ENABLE controls the command and DEN outputs of the bus controller. CEN/AEN inputs may be asynchronous to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs. This input may be connected to V_{CC} or GND.</p> <p>When MB is HIGH this pin has the AEN function. AEN is an active LOW input which indicates that the CPU has been granted use of a shared bus and the bus controller command outputs may exit 3-state OFF and become inactive (HIGH). AEN HIGH indicates that the CPU does not have control of the shared bus and forces the command outputs into 3-state OFF and DEN inactive (LOW).</p> <p>When MB is LOW this pin has the CEN function. CEN is an unlatched active HIGH input which allows the bus controller to activate its command and DEN outputs. With MB LOW, CEN LOW forces the command and DEN outputs inactive but does not tristate them.</p>
ALE	O	<p>ADDRESS LATCH ENABLE controls the address latches used to hold an address stable during a bus cycle. This control output is active HIGH. ALE will not be issued for the halt bus cycle and is not affected by any of the control inputs.</p>
MCE	O	<p>MASTER CASCADE ENABLE signals that a cascade address from a master 8259A interrupt controller may be placed onto the CPU address bus for latching by the address latches under ALE control. The CPU's address bus may then be used to broadcast the cascade address to slave interrupt controllers so only one of them will respond to the interrupt acknowledge cycle. This control output is active HIGH. MCE is only active during interrupt acknowledge cycles and is not affected by any control input. Using MCE to enable cascade address drivers requires latches which save the cascade address on the falling edge of ALE.</p>
DEN	O	<p>DATA ENABLE controls when data transceivers connected to the local data bus should be enabled. DEN is an active HIGH control output. DEN is delayed for write cycles in the MULTIBUS I mode.</p>
DT/R	O	<p>DATA TRANSMIT/RECEIVE establishes the direction of data flow to or from the local data bus. When HIGH, this control output indicates that a write bus cycle is being performed. A LOW indicates a read bus cycle. DEN is always inactive when DT/R changes states. This output is HIGH when no bus cycle is active. DT/R is not affected by any of the control inputs.</p>
OWC	O	<p>I/O WRITE COMMAND instructs an I/O device to read the data on the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. READY controls when it becomes inactive.</p>
ORC	O	<p>I/O READ COMMAND instructs an I/O device to place data onto the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. READY controls when it becomes inactive.</p>
MWTC	O	<p>MEMORY WRITE COMMAND instructs a memory device to read the data on the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. READY controls when it becomes inactive.</p>
MRDC	O	<p>MEMORY READ COMMAND instructs the memory device to place data onto the data bus. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. READY controls when it becomes inactive.</p>

Table 1. Pin Description (Continued)

Symbol	Type	Name and Function
INTA	O	INTERRUPT ACKNOWLEDGE tells an interrupting device that its interrupt request is being acknowledged. This command output is active LOW. The MB and CMDLY inputs control when this output becomes active. READY controls when it becomes inactive.
V _{CC}		System Power: +5V Power Supply
GND		System Ground: 0V

Table 2. Command and Control Outputs for Each Type of Bus Cycle

Type of Bus Cycle	M/ \overline{IO}	$\overline{S1}$	$\overline{S0}$	Command Activated	DT/ \overline{R} State	ALE, DEN Issued?	MCE Issued?
Interrupt Acknowledge	0	0	0	INTA	LOW	YES	YES
I/O Read	0	0	1	\overline{IORC}	LOW	YES	NO
I/O Write	0	1	0	\overline{IOWC}	HIGH	YES	NO
None; Idle	0	1	1	None	HIGH	NO	NO
Halt/Shutdown	1	0	0	None	HIGH	NO	NO
Memory Read	1	0	1	\overline{MRDC}	LOW	YES	NO
Memory Write	1	1	0	\overline{MWTC}	HIGH	YES	NO
None; Idle	1	1	1	None	HIGH	NO	NO

Operating Modes

Two types of buses are supported by the 82C288: MULTIBUS I and non-MULTIBUS I. When the MB input is strapped HIGH, MULTIBUS I timing is used. In MULTIBUS I mode, the 82C288 delays command and data activation to meet IEEE-796 requirements on address to command active and write data to command active setup timing. MULTIBUS I mode requires at least one wait state in the bus cycle since the command outputs are delayed. The non-MULTIBUS I mode does not delay any outputs and does not require wait states. The MB input affects the timing of the command and DEN outputs.

Command and Control Outputs

The type of bus cycle performed by the local bus master is encoded in the M/ \overline{IO} , $\overline{S1}$, and $\overline{S0}$ inputs. Different command and control outputs are activated depending on the type of bus cycle. Table 2 indicates the cycle decode done by the 82C288 and the effect on command, DT/ \overline{R} , ALE, DEN, and MCE outputs.

Bus cycles come in three forms: read, write, and halt. Read bus cycles include memory read, I/O read, and interrupt acknowledge. The timing of the associated read command outputs (\overline{MRDC} , \overline{IORC} ,

and INTA), control outputs (ALE, DEN, DT/ \overline{R}) and control inputs (CEN/ \overline{AEN} , CENL, CMDLY, MB, and READY) are identical for all read bus cycles. Read cycles differ only in which command output is activated. The MCE control output is only asserted during interrupt acknowledge cycles.

Write bus cycles activate different control and command outputs with different timing than read bus cycles. Memory write and I/O write are write bus cycles whose timing for command outputs (\overline{MWTC} and \overline{IOWC}), control outputs (ALE, DEN, DT/ \overline{R}) and control inputs (CEN/ \overline{AEN} , CENL, CMDLY, MB, and READY) are identical. They differ only in which command output is activated.

Halt bus cycles are different because no command or control output is activated. All control inputs are ignored until the next bus cycle is started via $\overline{S1}$ and $\overline{S0}$.

Static Operation

All 82C288 circuitry is of static design. Internal registers and logic are static and require no refresh as with dynamic circuit design. This eliminates the minimum operating frequency restriction placed on the HMOS 82288. The CHMOS III 82C288 can operate from DC to the appropriate upper frequency limit.

The clock may be stopped in either state (HIGH/LOW) and held there indefinitely.

Power dissipation is directly related to operating frequency. As the system frequency is reduced, so is the operating power. When the clock is stopped to the 82C288, power dissipation is at a minimum. This is useful for low-power and portable applications.

FUNCTIONAL DESCRIPTION

Introduction

The 82C288 bus controller is used in 80286 systems to provide address latch control, data transceiver control, and standard level-type command outputs. The command outputs are timed and have sufficient drive capabilities for large TTL buses and meet all IEEE-796 requirements for MULTIBUS I. A special MULTIBUS I mode is provided to satisfy all address/data setup and hold time requirements. Command timing may be tailored to special needs via a CMDLY input to determine the start of a command and READY to determine the end of a command.

Connection to multiple buses are supported with a latched enable input (CENL). An address decoder can determine which, if any, bus controller should be enabled for the bus cycle. This input is latched to allow an address decoder to take full advantage of the pipelined timing on the 80286 local bus.

Bus sharing by several bus controllers is supported. An $\overline{\text{AEN}}$ input prevents the bus controller from driving the shared bus command and data signals except when enabled by an external MULTIBUS I type bus arbiter.

Separate DEN and DT/ $\overline{\text{R}}$ outputs control the data transceivers for all buses. Bus contention is eliminated by disabling DEN before changing DT/ $\overline{\text{R}}$. The DEN timing allows sufficient time for tristate bus drivers to enter 3-state OFF before enabling other drivers onto the same bus.

The term CPU refers to any 80286 processor or 80286 support component which may become an 80286 local bus master and thereby drive the 82C288 status inputs.

Processor Cycle Definition

Any CPU which drives the local bus uses an internal clock which is one half the frequency of the system clock (CLK) (see Figure 3). Knowledge of the phase of the local bus master internal clock is required for proper operation of the 80286 local bus. The local bus master informs the bus controller of its internal clock phase when it asserts the status signals. Status signals are always asserted beginning in Phase 1 of the local bus master's internal clock.

3

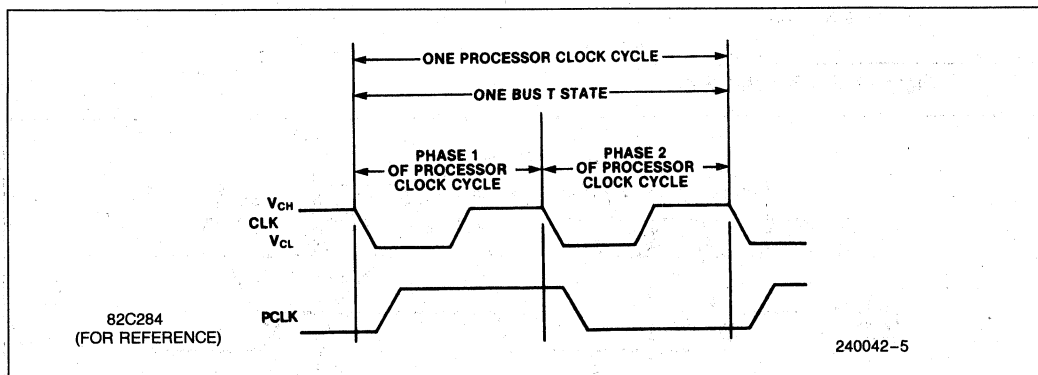


Figure 3. CLK Relationship to the Processor Clock and Bus T-States

Bus State Definition

The 82C288 bus controller has three bus states (see Figure 4): Idle (T_I) Status (T_S) and Command (T_C). Each bus state is two CLK cycles long. Bus state phases correspond to the internal CPU processor clock phases.

The T_I bus state occurs when no bus cycle is currently active on the 80286 local bus. This state may be repeated indefinitely. When control of the local bus is being passed between masters, the bus remains in the T_I state.

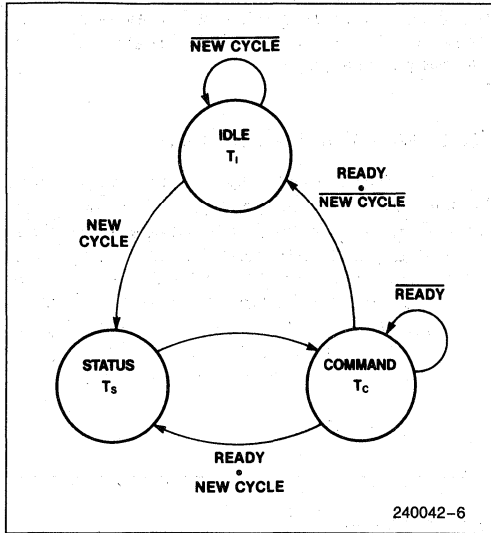


Figure 4. 82C288 Bus States

Bus Cycle Definition

The $\overline{S1}$ and $\overline{S0}$ inputs signal the start of a bus cycle. When either input becomes LOW, a bus cycle is started. The T_S bus state is defined to be the two CLK cycles during which either $\overline{S1}$ or $\overline{S0}$ are active (see Figure 5). These inputs are sampled by the 82C288 at every falling edge of CLK. When either $\overline{S1}$ or $\overline{S0}$ are sampled LOW, the next CLK cycle is considered the second phase of the internal CPU clock cycle.

The local bus enters the T_C bus state after the T_S state. The shortest bus cycle may have one T_S state and one T_C state. Longer bus cycles are formed by repeating T_C state. A repeated T_C bus state is called a wait state.

The \overline{READY} input determines whether the current T_C bus state is to be repeated. The \overline{READY} input has the same timing and effect for all bus cycles. \overline{READY} is sampled at the end of each T_C bus state to see if it is active. If sampled HIGH, the T_C bus state is repeated. This is called inserting a wait state. The control and command outputs do not change during wait states.

When \overline{READY} is sampled LOW, the current bus cycle is terminated. Note that the bus controller may enter the T_S bus state directly from T_C if the status lines are sampled active at the next falling edge of CLK.

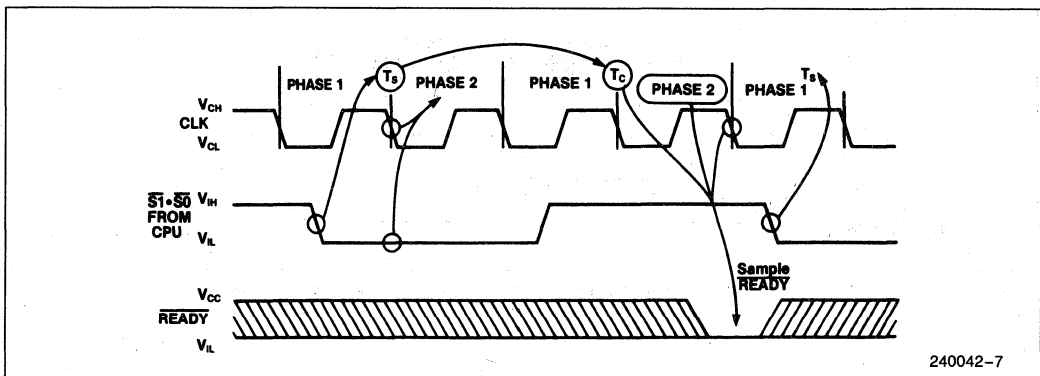


Figure 5. Bus Cycle Definition

Figures 6 through 10 show the basic command and control output timing for read and write bus cycles. Halt bus cycles are not shown since they activate no outputs. The basic idle-read-idle and idle-write-idle bus cycles are shown. The signal label CMD represents the appropriate command output for the bus cycle. For Figures 6 through 10, the CMDLY input is connected to GND and CENL to V_{CC}. The effects of CENL and CMDLY are described later in the section on control inputs.

Figures 6, 7 and 8 show non-MULTIBUS I cycles. MB is connected to GND while CEN is connected to V_{CC}. Figure 6 shows a read cycle with no wait states while Figure 7 shows a write cycle with one wait state. The READY input is shown to illustrate how wait states are added.

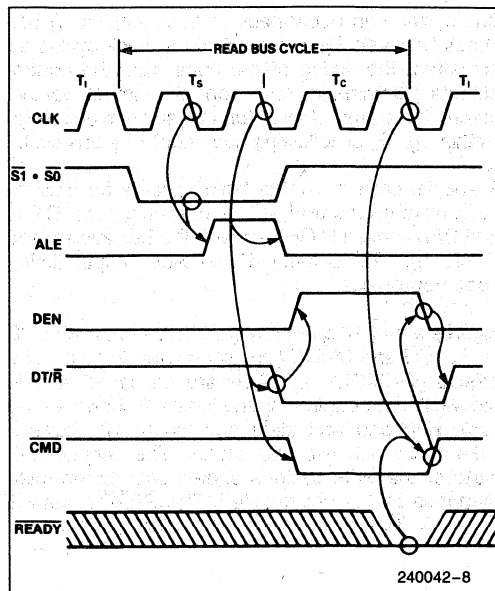


Figure 6. Idle-Read-Idle Bus Cycles with MB = 0

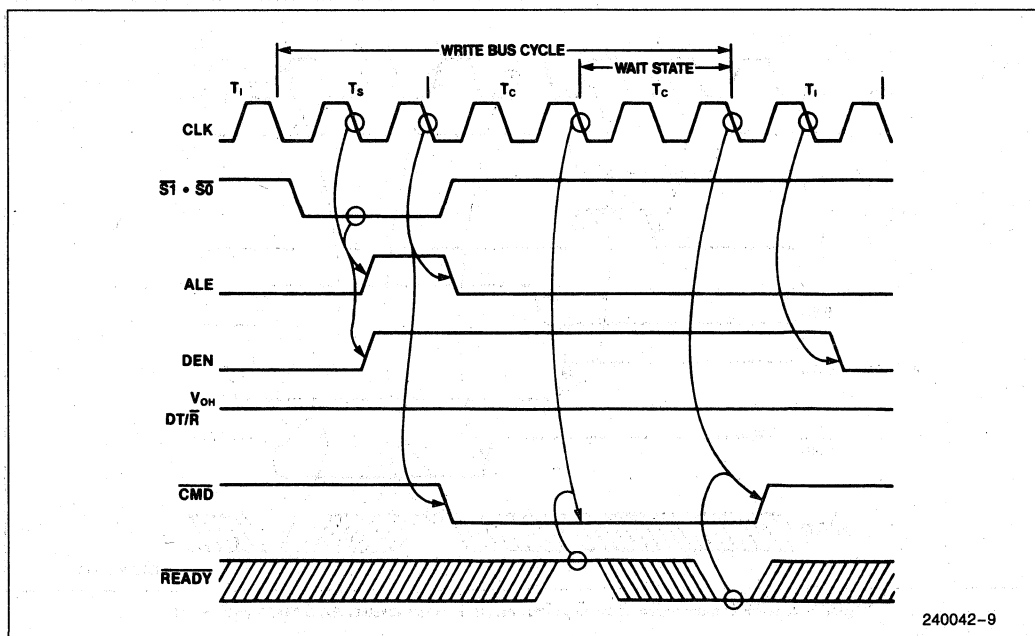


Figure 7. Idle-Write-Idle Bus Cycles with MB = 0

Bus cycles can occur back to back with no T_1 bus states between T_C and T_S . Back to back cycles do not affect the timing of the command and control outputs. Command and control outputs always reach the states shown for the same clock edge (within T_S , T_C or following bus state) of a bus cycle.

A special case in control timing occurs for back to back write cycles with $MB = 0$. In this case, DT/R and DEN remain HIGH between the bus cycles (see Figure 8). The command and ALE output timing does not change.

Figures 9 and 10 show a MULTIBUS I cycle with $MB = 1$. \overline{AEN} and $CMDLY$ are connected to GND. The effects of $CMDLY$ and \overline{AEN} are described later in the section on control inputs. Figure 9 shows a read cycle with one wait state and Figure 10 shows a write cycle with two wait states. The second wait state of the write cycle is shown only for example purposes and is not required. The $READY$ input is shown to illustrate how wait states are added.

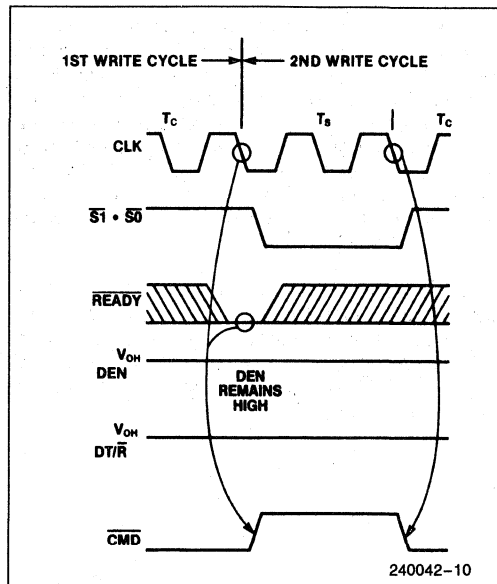


Figure 8. Write-Write Bus Cycles with $MB = 0$

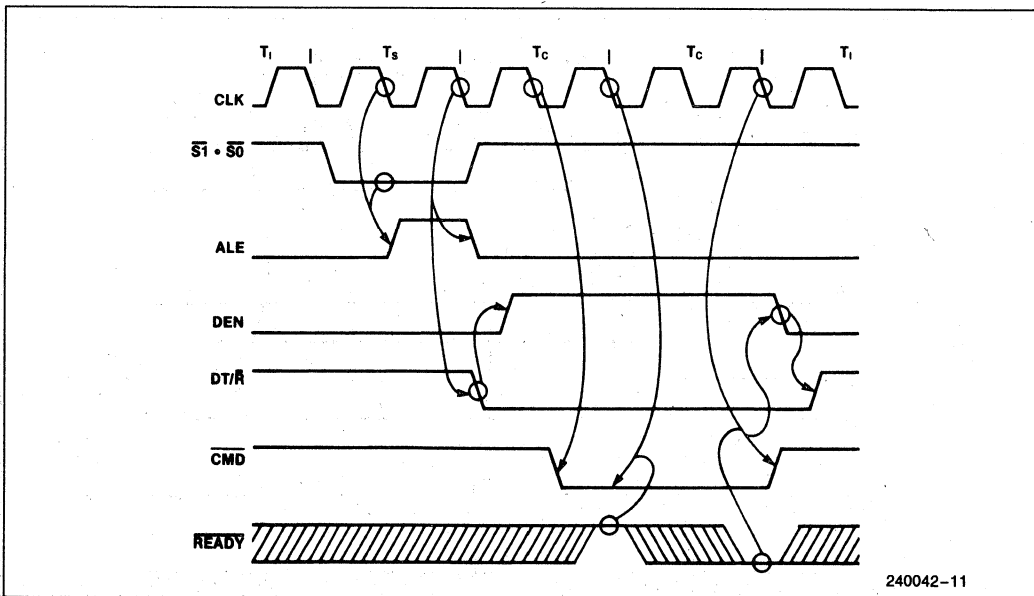


Figure 9. Idle-Read-Idle Bus Cycles with 1 Wait State and with $MB = 1$

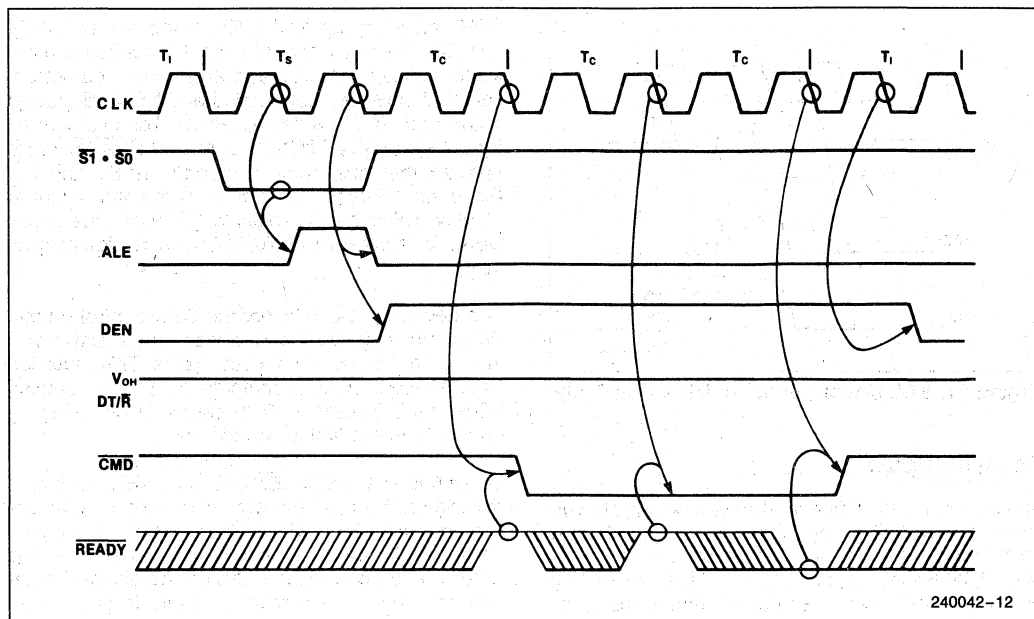


Figure 10. Idle-Write-Idle Bus Cycles with 2 Wait States and with MB = 1

The MB control input affects the timing of the command and DEN outputs. These outputs are automatically delayed in MULTIBUS I mode to satisfy three requirements:

- 1) 50 ns minimum setup time for valid address before any command output becomes active.
- 2) 50 ns minimum setup time for valid write data before any write command output becomes active.
- 3) 65 ns maximum time from when any read command becomes inactive until the slave's read data drivers reach 3-state OFF.

Three signal transitions are delayed by MB = 1 as compared to MB = 0:

- 1) The HIGH to LOW transition of the read command outputs (IORC, MRDC, and INTA) are delayed one CLK cycle.
- 2) The HIGH to LOW transition of the write command outputs (IOWC and MWTC) are delayed two CLK cycles.
- 3) The LOW to HIGH transition of DEN for write cycles is delayed one CLK cycle.

Back to back bus cycles with MB = 1 do not change the timing of any of the command or control outputs. DEN always becomes inactive between bus cycles with MB = 1.

Except for a halt or shutdown bus cycle, ALE will be issued during the second half of T_S for any bus cycle. ALE becomes inactive at the end of the T_S to allow latching the address to keep it stable during the entire bus cycle. The address outputs may change during Phase 2 of any T_C bus state. ALE is not affected by any control input.

Figure 11 shows how MCE is timed during interrupt acknowledge (INTA) bus cycles. MCE is one CLK cycle longer than ALE to hold the cascade address from a master 8259A valid after the falling edge of ALE. With the exception of the MCE control output, an INTA bus cycle is identical in timing to a read bus cycle. MCE is not affected by any control input.

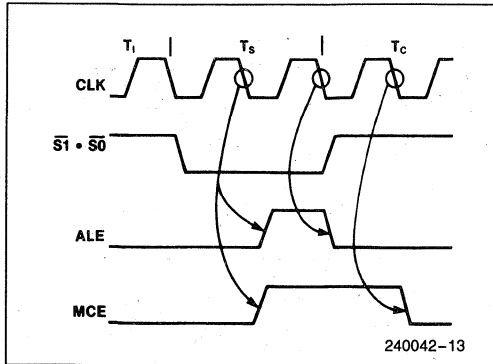


Figure 11. MCE Operation for an INTA Bus Cycle

Control Inputs

The control inputs can alter the basic timing of command outputs, allow interfacing to multiple buses, and share a bus between different masters. For many 80286 systems, each CPU will have more than one bus which may be used to perform a bus cycle. Normally, a CPU will only have one bus controller active for each bus cycle. Some buses may be shared by more than one CPU (i.e. MULTIBUS) requiring only one of them use the bus at a time.

Systems with multiple and shared buses use two control input signals of the 82C288 bus controller, CENL and \overline{AEN} (see Figure 12). CENL enables the bus controller to control the current bus cycle. The \overline{AEN} input prevents a bus controller from driving its command outputs. \overline{AEN} HIGH means that another bus controller may be driving the shared bus.

In Figure 12, two buses are shown: a local bus and a MULTIBUS I. Only one bus is used for each CPU bus cycle. The CENL inputs of the bus controller select which bus controller is to perform the bus cycle. An address decoder determines which bus to use for each bus cycle. The 82C288 connected to the shared MULTIBUS I must be selected by CENL and be given access to the MULTIBUS I by \overline{AEN} before it will begin a MULTIBUS I operation.

CENL must be sampled HIGH at the end of the T_S bus state (see waveforms) to enable the bus controller to activate its command and control outputs. If sampled LOW the commands and DEN will not go active and DT/\overline{R} will remain HIGH. The bus controller will ignore the \overline{CMDLY} , CEN, and \overline{READY} inputs until another bus cycle is started via $\overline{S1}$ and $\overline{S0}$. Since an address decoder is commonly used to identify which bus is required for each bus cycle, CENL is latched to avoid the need for latching its input.

The CENL input can affect the DEN control output. When $MB = 0$, DEN normally becomes active during Phase 2 of T_S in write bus cycles. This transition occurs before CENL is sampled. If CENL is sampled LOW, the DEN output will be forced LOW during T_C as shown in the timing waveforms.

When $MB = 1$, CEN/\overline{AEN} becomes \overline{AEN} . \overline{AEN} controls when the bus controller command outputs enter and exit 3-state OFF. \overline{AEN} is intended to be driven by a MULTIBUS I type bus arbiter, which assures only one bus controller is driving the shared bus at any time. When \overline{AEN} makes a LOW to HIGH transition, the command outputs immediately enter 3-state OFF and DEN is forced inactive. An inactive DEN should force the local data transceivers connected to the shared data bus into 3-state OFF (see Figure 12). The LOW to HIGH transition of \overline{AEN} should only occur during T_1 or T_S bus states.

The HIGH to LOW transition of \overline{AEN} signals that the bus controller may now drive the shared bus command signals. Since a bus cycle may be active or be in the process of starting, \overline{AEN} can become active during any T-state. \overline{AEN} LOW immediately allows DEN to go to the appropriate state. Three CLK edges later, the command outputs will go active (see timing waveforms). The MULTIBUS I requires this delay for the address and data to be valid on the bus before the command becomes active.

When $MB = 0$, CEN/\overline{AEN} becomes CEN. CEN is an asynchronous input which immediately affects the command and DEN outputs. When CEN makes a HIGH to LOW transition, the commands and DEN

are immediately forced inactive. When CEN makes a LOW to HIGH transition, the commands and DEN outputs immediately go to the appropriate state (see timing waveforms). READY must still become active to terminate a bus cycle if CEN remains LOW for a selected bus controller (CENL was latched HIGH).

Some memory or I/O systems may require more address or write data setup time to command active than provided by the basic command output timing. To provide flexible command timing, the CMDLY input can delay the activation of command outputs. The CMDLY input must be sampled LOW to activate the command outputs. CMDLY does not affect the control outputs ALE, MCE, DEN, and DT/R.

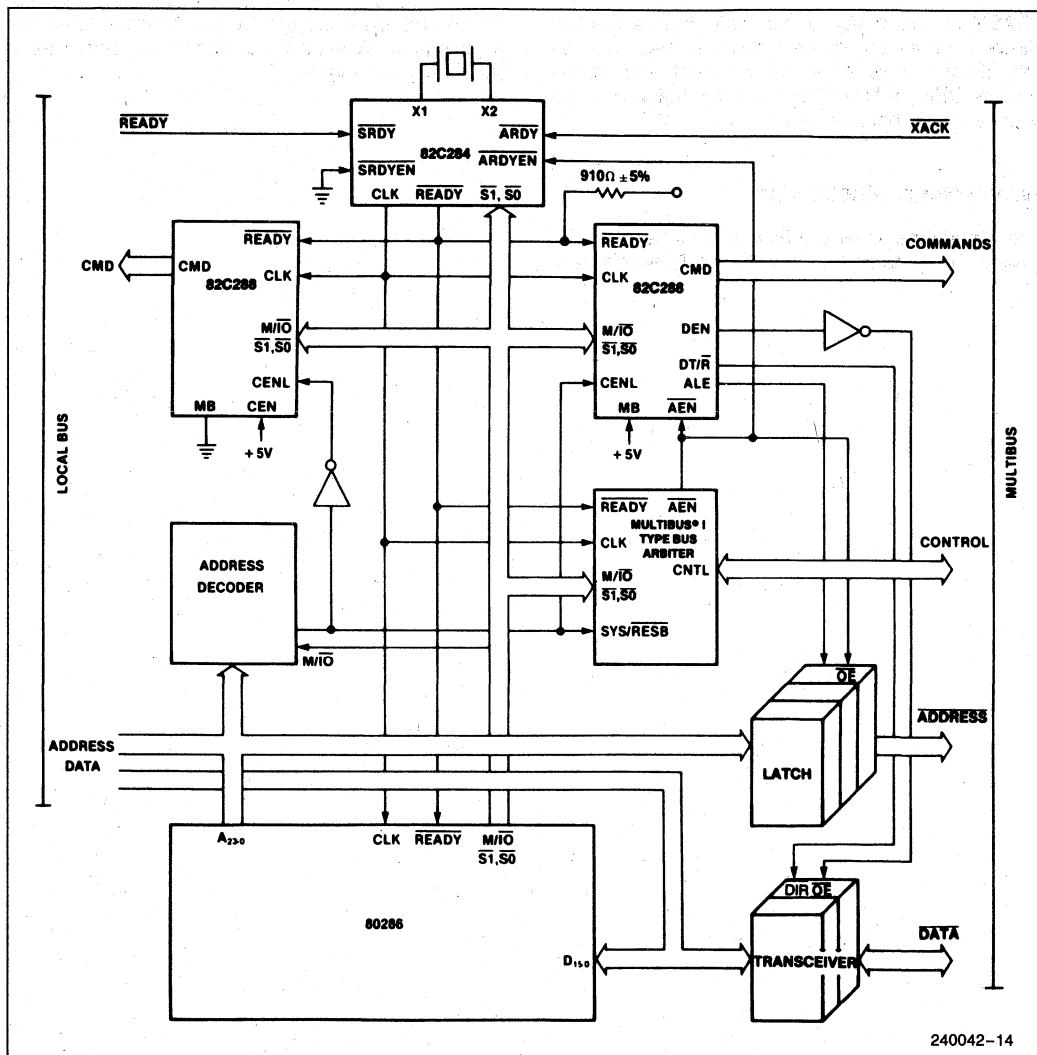


Figure 12. System Use of AEN and CEN

CMDLY is first sampled on the falling edge of the CLK ending T_S . If sampled HIGH, the command output is not activated, and CMDLY is again sampled on the next falling edge of CLK. Once sampled LOW, the proper command output becomes active immediately if $MB = 0$. If $MB = 1$, the proper command goes active no earlier than shown in Figures 9 and 10.

\overline{READY} can terminate a bus cycle before CMDLY allows a command to be issued. In this case no commands are issued and the bus controller will deactivate DEN and DT/\overline{R} in the same manner as if a command had been issued.

Waveforms Discussion

The waveforms show the timing relationships of inputs and outputs and do not show all possible tran-

sitions of all signals in all modes. Instead, all signal timing relationships are shown via the general cases. Special cases are shown when needed. The waveforms provide some functional descriptions of the 82C288; however, most functional descriptions are provided in Figures 5 through 11.

To find the timing specification for a signal transition in a particular mode, first look for a special case in the waveforms. If no special case applies, then use a timing specification for the same or related function in another mode.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to GND	−0.5V to +7V
Power Dissipation	1 Watt

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

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

D.C. CHARACTERISTICS $V_{CC} = 5V \pm 5\%$, $T_{CASE} = 0^{\circ}C$ to $85^{\circ}C$ *

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input LOW Voltage	−0.5	0.8	V	
V_{IH}	Input HIGH Voltage	2.0	$V_{CC} + 0.5$	V	
V_{ILC}	CLK Input LOW Voltage	−0.5	0.6	V	
V_{IHC}	CLK Input HIGH Voltage	3.8	$V_{CC} + 0.5$	V	
V_{OL}	Output LOW Voltage Command Outputs Control Outputs		0.45 0.45	V V	$I_{OL} = 32$ mA (Note 1) $I_{OL} = 16$ mA (Note 2)
V_{OH}	Output HIGH Voltage Command Outputs Control Outputs	2.4 $V_{CC} - 0.5$ 2.4 $V_{CC} - 0.5$		V V V V	$I_{OH} = -5$ mA (Note 1) $I_{OH} = -1$ mA (Note 1) $I_{OH} = -1$ mA (Note 2) $I_{OH} = -0.2$ mA (Note 2)
I_{IL}	Input Leakage Current		± 10	μA	$0V \leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current		± 10	μA	$0.45V \leq V_{OUT} \leq V_{CC}$
I_{CC}	Power Supply Current		75	mA	
I_{CCS}	Power Supply Current (Static)		3	mA	(Note 3)
C_{CLK}	CLK Input Capacitance		12	pF	$F_C = 1$ MHz
C_I	Input Capacitance		10	pF	$F_C = 1$ MHz
C_O	Input/Output Capacitance		20	pF	$F_C = 1$ MHz

* T_A is guaranteed from $0^{\circ}C$ to $+70^{\circ}C$ as long as T_{CASE} is not exceeded.

NOTES:

1. Command Outputs are \overline{INTA} , \overline{IORC} , \overline{IOWC} , \overline{MRDC} and \overline{MWRC} .
2. Control Outputs are $\overline{DT/R}$, \overline{DEN} , \overline{ALE} and \overline{MCE} .
3. Tested while outputs are unloaded, and inputs at V_{CC} or V_{SS} .

A.C. CHARACTERISTICS

$V_{CC} = 5V, \pm 5\%$, $T_{CASE} = 0^{\circ}C$ to $+85^{\circ}C$. * AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted.

Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Max	-12 Min	-12 Max		
1	CLK Period	62	250	50	250	40	250	ns	
2	CLK HIGH Time	20		16		13		ns	at 3.6V
3	CLK LOW Time	15		12		11		ns	at 1.0V
4	CLK Rise Time		10		8		8	ns	1.0V to 3.6V
5	CLK Fall Time		10		8		8	ns	3.6V to 1.0V
6	M/ \overline{IO} and Status Setup Time	22		18		15		ns	
7	M/ \overline{IO} and Status Hold Time	1		1		1		ns	
8	CENL Setup Time	20		15		15		ns	
9	CENL Hold Time	1		1		1		ns	
10	\overline{READY} Setup Time	38		26		18		ns	
11	\overline{READY} Hold Time	25		25		20		ns	
12	CMDLY Setup Time	20		15		15		ns	
13	CMDLY Hold Time	1		1		1		ns	
14	\overline{AEN} Setup Time	20		15		15		ns	(Note 3)
15	\overline{AEN} Hold Time	0		0		0		ns	(Note 3)
16	ALE, MCE Active Delay from CLK	3	20	3	16	3	16	ns	(Note 4)
17	ALE, MCE Inactive Delay from CLK		25		19		19	ns	(Note 4)
18	DEN (Write) Inactive from CENL		35		23		23	ns	(Note 4)
19	DT/ \overline{R} LOW from CLK		25		23		23	ns	(Note 4)
20	DEN (Read) Active \overline{R} from DT/	5	35	5	21	5	21	ns	(Note 4)
21	DEN (Read) Inactive Dly from CLK	3	35	3	21	3	19	ns	(Note 4)
22	DT/ \overline{R} HIGH from DEN Inactive	5	35	5	20	5	18	ns	(Note 4)
23	DEN (Write) Active Delay from CLK		30		23		23	ns	(Note 4)
24	DEN (Write) Inactive Dly from CLK	3	30	3	19	3	19	ns	(Note 4)

* T_A is guaranteed from $0^{\circ}C$ to $+70^{\circ}C$ as long as T_{CASE} is not exceeded.

A.C. CHARACTERISTICS

$V_{CC} = 5V, \pm 5\%$, $T_{CASE} = 0^{\circ}C \text{ to } +85^{\circ}C$. * AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in data sheet waveforms, unless otherwise noted. (Continued)

Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		Unit	Test Condition
		-8 Min	-8 Max	-10 Min	-10 Max	-12 Min	-12 Max		
25	DEN Inactive from CEN		30		25		25	ns	(Note 4)
26	DEN Active from CEN		30		24		24	ns	(Note 4)
27	DT/ \overline{R} HIGH from CLK (when CEN = LOW)		35		25		25	ns	(Note 4)
28	DEN Active from \overline{AEN}		30		26		26	ns	(Note 4)
29	\overline{CMD} Active Delay from CLK	3	25	3	21	3	21	ns	(Note 5)
30	\overline{CMD} Inactive Delay from CLK	5	20	5	20	5	20	ns	(Note 5)
31	\overline{CMD} Active from CEN		25		25		25	ns	(Note 5)
32	\overline{CMD} Inactive from CEN		25		25		25	ns	(Note 5)
33	\overline{CMD} Inactive Enable from \overline{AEN}		40		40		40	ns	(Note 5)
34	\overline{CMD} Float Delay from \overline{AEN}		40		40		40	ns	(Note 6)
35	MB Setup Time	20		20		20		ns	
36	MB Hold Time	0		0		0		ns	
37	Command Inactive Enable from MB \downarrow		40		40		40	ns	(Note 5)
38	Command Float Time from MB \uparrow		40		40		40	ns	(Note 6)
39	DEN Inactive from MB \uparrow		30		26		26	ns	(Note 4)
40	DEN Active from MB \downarrow		30		30		30	ns	(Note 4)

* T_A is guaranteed from $0^{\circ}C$ to $+70^{\circ}C$ as long as T_{CASE} is not exceeded.

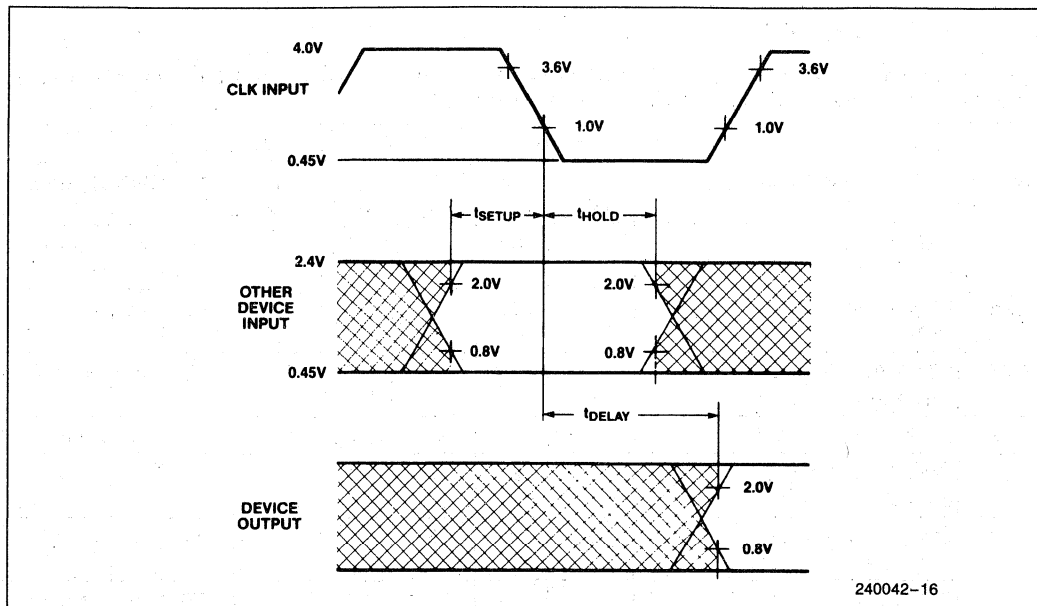
NOTES:

3. \overline{AEN} is an asynchronous input. This specification is for testing purposes only, to assure recognition at a specific CLK edge.

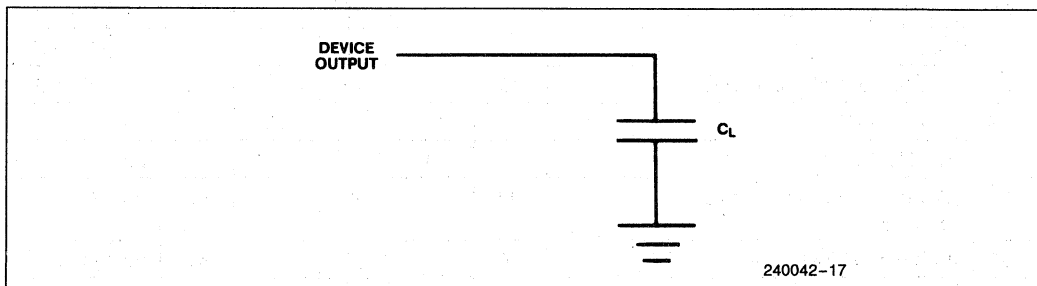
4. Control output load: $C_L = 150 \text{ pF}$.

5. Command output load: $C_L = 300 \text{ pF}$.

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



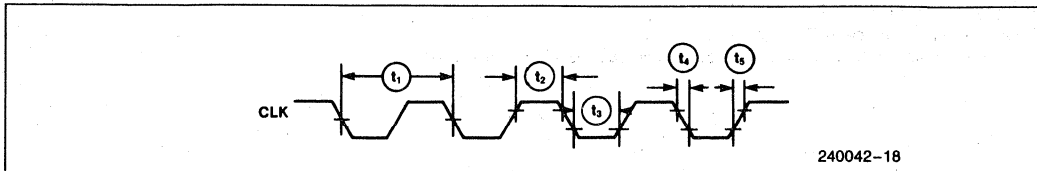
Note 7: AC Setup, Hold and Delay Time Measurement—General



Note 8: AC Test Loading on Outputs

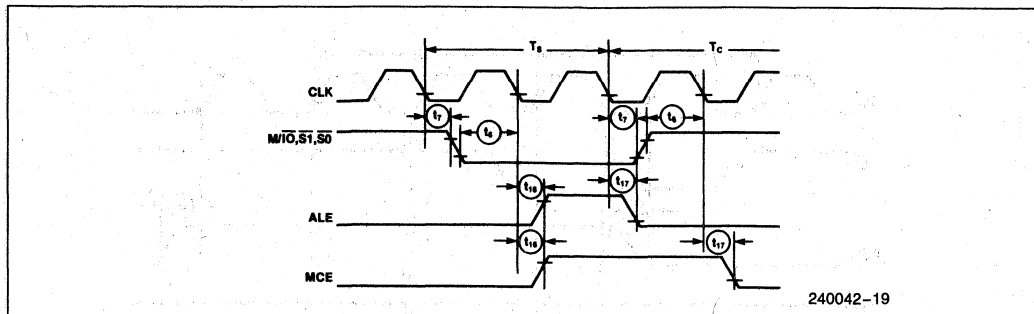
WAVEFORMS

CLK CHARACTERISTICS

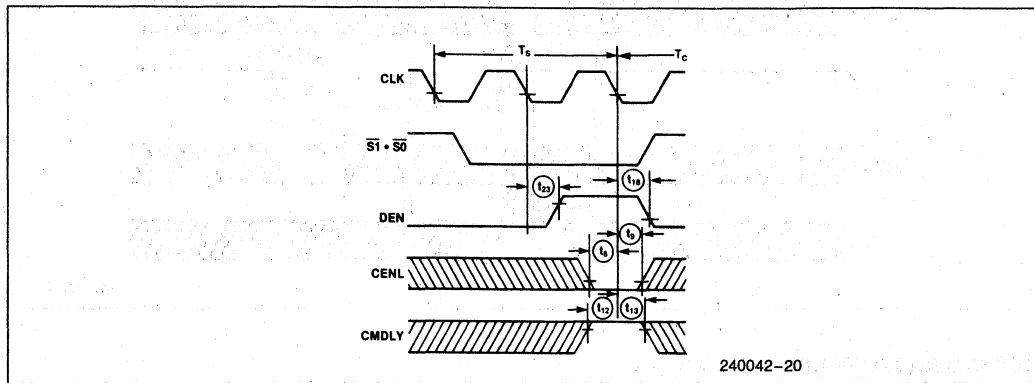


WAVEFORMS (Continued)

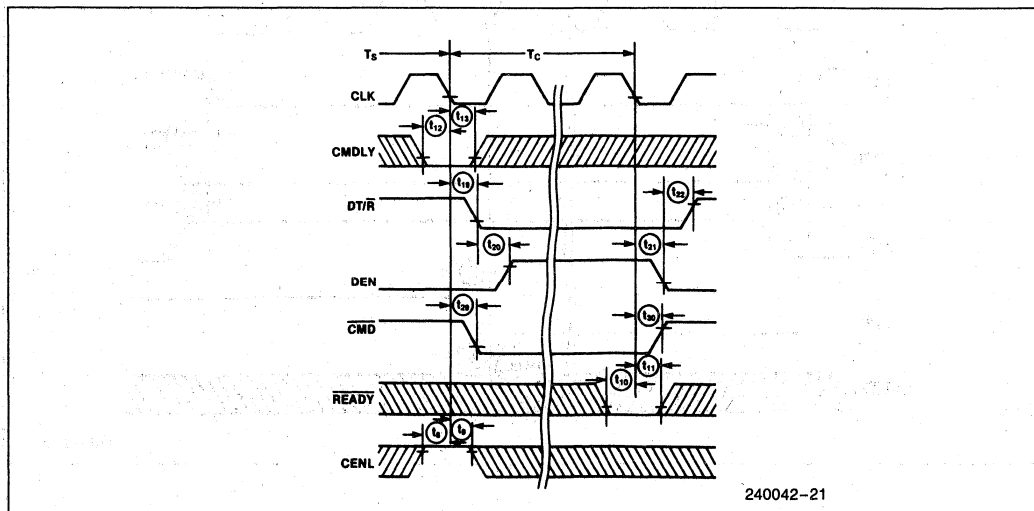
STATUS, ALE, MCE, CHARACTERISTICS



CENL, CMDLY, DEN CHARACTERISTICS WITH MB = 0 AND CEN = 1 DURING WRITE CYCLE

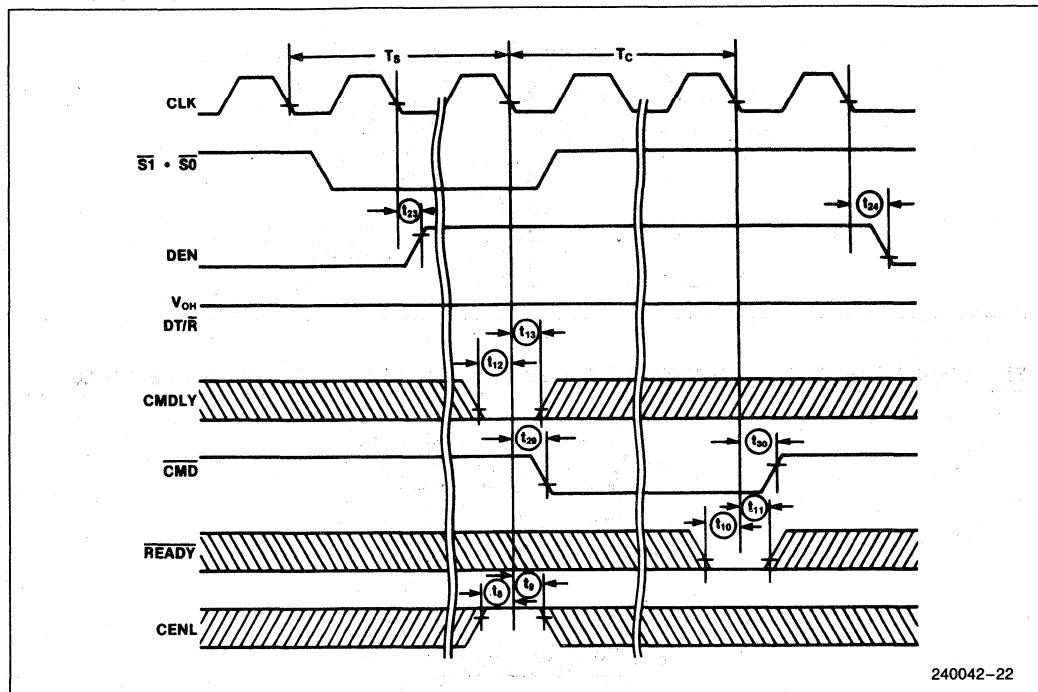


READ CYCLE CHARACTERISTICS WITH MB = 0 AND CEN = 1

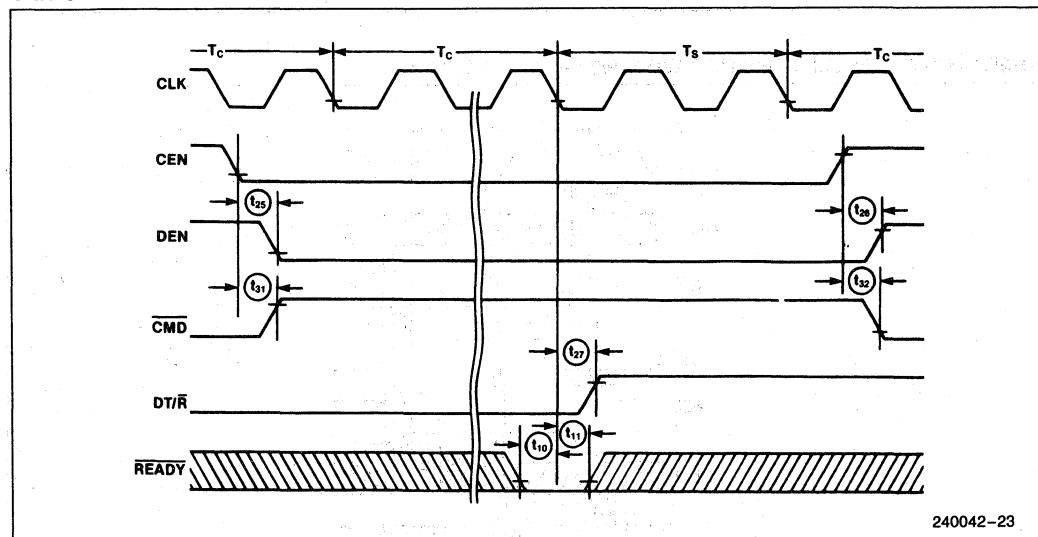


WAVEFORMS (Continued)

WRITE CYCLE CHARACTERISTIC WITH MB = 0 AND CEN = 1

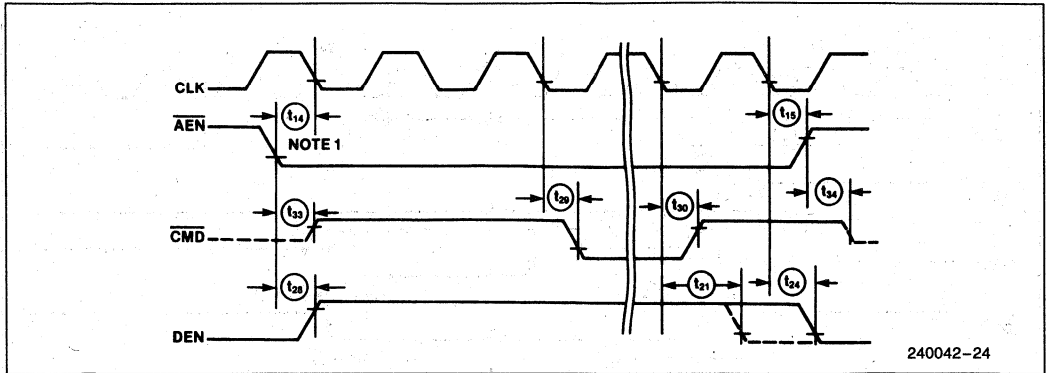


CEN CHARACTERISTICS WITH MB = 0



WAVEFORMS (Continued)

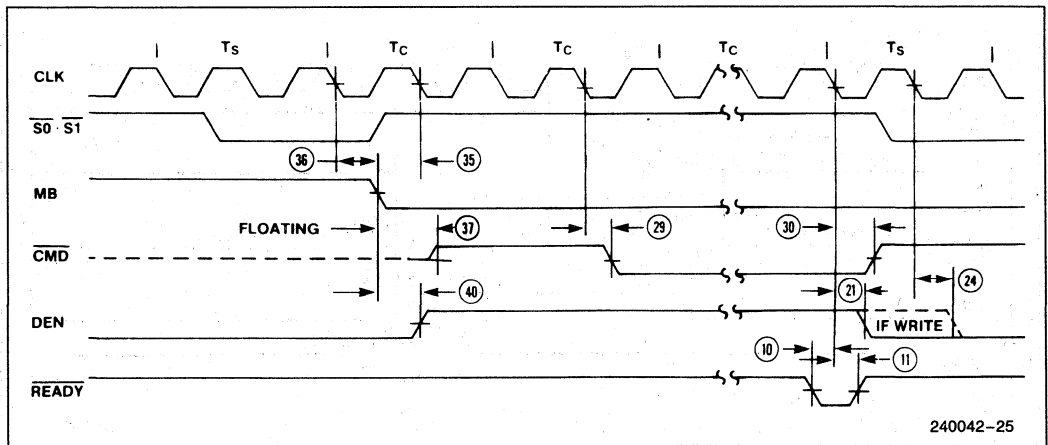
$\overline{\text{AEN}}$ CHARACTERISTICS WITH $\text{MB} = 1$



NOTE:

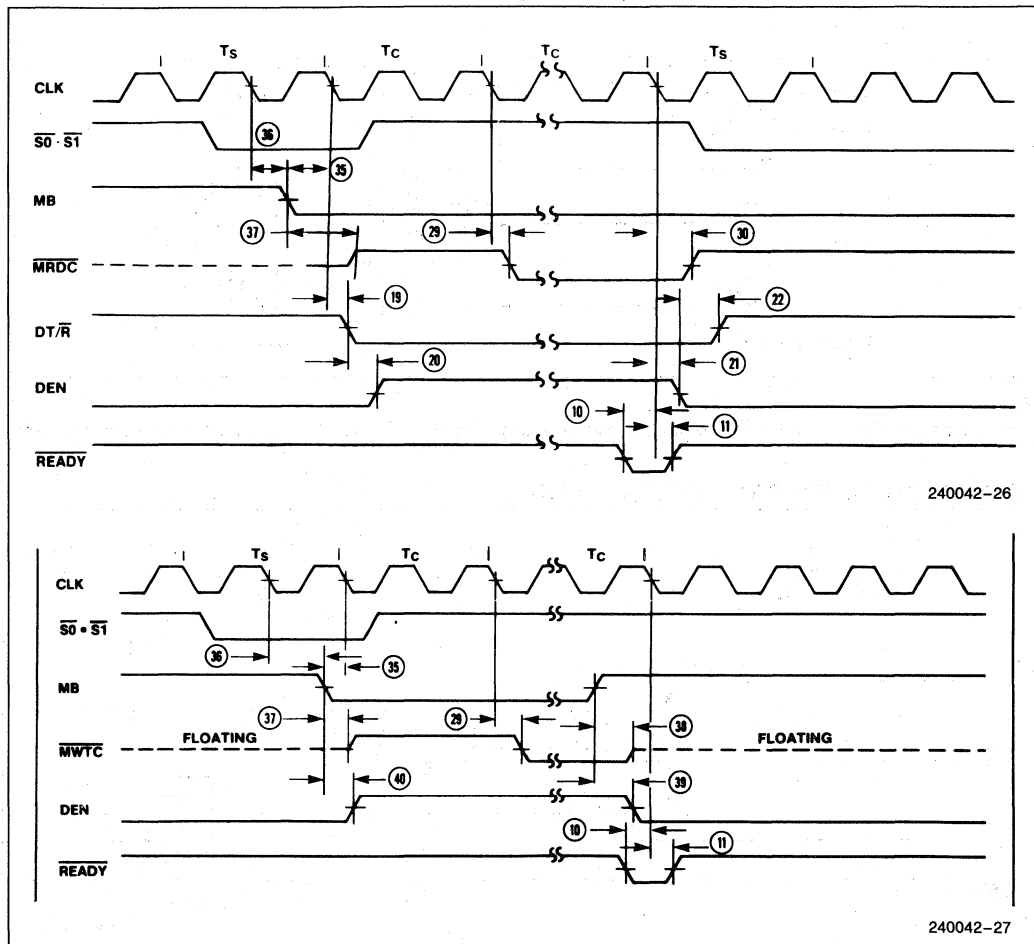
1. $\overline{\text{AEN}}$ is an asynchronous input. $\overline{\text{AEN}}$ setup and hold time is specified to guarantee the response shown in the waveforms.

MB CHARACTERISTICS WITH $\overline{\text{AEN}}/\text{CEN} = \text{HIGH}$



WAVEFORMS (Continued)

MB CHARACTERISTICS WITH $\overline{\text{AEN}}/\text{CEN} = \text{HIGH}$ (Continued)



NOTES:

1. MB is an asynchronous input. MB setup and hold times specified to guarantee the response shown in the waveforms.
2. If the setup time, t_{s35} , is met two clock cycles will occur before CMD becomes active after the falling edge of MB.

DATA SHEET REVISION REVIEW

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

1. The I_{CCS} specification was changed from 1 mA to 3 mA maximum.
2. The "PRELIMINARY" markings have been removed from the data sheet.



82C284 CLOCK GENERATOR AND READY INTERFACE FOR 80286 PROCESSORS (82C284-12, 82C284-10, 82C284-8)

- Generates System Clock for 80286 Processors
 - Uses Crystal or TTL Signal for Frequency Source
 - Provides Local READY and MULTIBUS® I READY Synchronization
 - High Speed CHMOS III Technology
 - Generates System Reset Output
 - Available in 18-Lead Cerdip and 20-Pin PLCC (Plastic Leaded Chip Carrier) Packages
- (See Packaging Spec, Order #231369)

The 82C284 is a clock generator/driver which provides clock signals for 80286 processors and support components. It also contains logic to supply $\overline{\text{READY}}$ to the CPU from either asynchronous or synchronous sources and synchronous RESET from an asynchronous input.

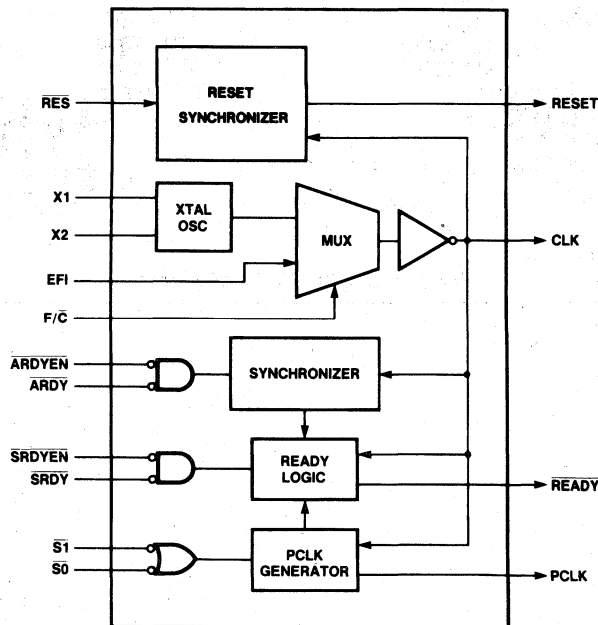
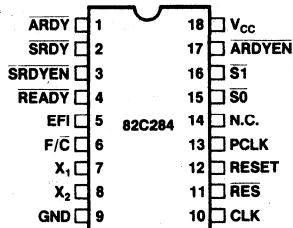


Figure 1. 82C284 Block Diagram

210453-1

18-Lead Cerdip

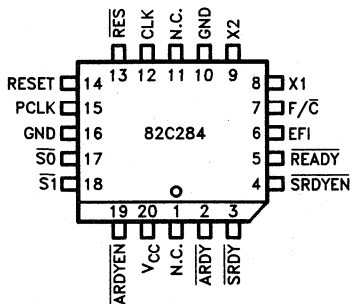


210453-2

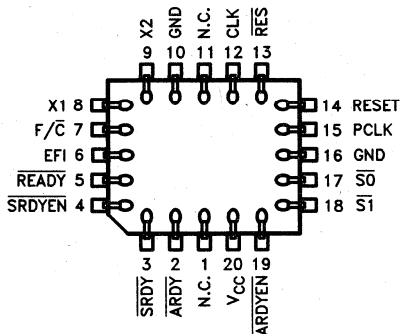
P.C. Board Views—As viewed from the component side of the P.C. Board.

Component Pad Views—As viewed from under-side of component when mounted on the board.

20 Pin PLCC



210453-18



210453-19

NOTE:

1. N.C. Signals must not be connected.

Figure 2. 82C284 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for the 82C284 clock generator.

Symbol	Type	Name and Function
CLK	O	SYSTEM CLOCK is the signal used by the processor and support devices which must be synchronous with the processor. The frequency of the CLK output has twice the desired internal processor clock frequency. CLK can drive both TTL and MOS level inputs.
F/ \overline{C}	I	FREQUENCY/CRYSTAL SELECT is a strapping option to select the source for the CLK output. When F/ \overline{C} is strapped LOW, the internal crystal oscillator drives CLK. When F/ \overline{C} is strapped HIGH, the EFI input drives the CLK output.
X1, X2	I	CRYSTAL IN are the pins to which a parallel resonant fundamental mode crystal is attached for the internal oscillator. When F/ \overline{C} is LOW, the internal oscillator will drive the CLK output at the crystal frequency. The crystal frequency must be twice the desired internal processor clock frequency.
EFI	I	EXTERNAL FREQUENCY IN drives CLK when the F/ \overline{C} input is strapped HIGH. The EFI input frequency must be twice the desired internal processor clock frequency.
PCLK	O	PERIPHERAL CLOCK is an output which provides a 50% duty cycle clock with 1/2 the frequency of CLK. PCLK will be in phase with the internal processor clock following the first bus cycle after the processor has been reset.
ARDYEN	I	ASYNCHRONOUS READY ENABLE is an active LOW input which qualifies the ARDY input. ARDYEN selects ARDY as the source of ready for the current bus cycle. Inputs to ARDYEN may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs.
ARDY	I	ASYNCHRONOUS READY is an active LOW input used to terminate the current bus cycle. The ARDY input is qualified by ARDYEN. Inputs to ARDY may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous outputs.
SRDYEN	I	SYNCHRONOUS READY ENABLE is an active LOW input which qualifies SRDY. SRDYEN selects SRDY as the source for READY to the CPU for the current bus cycle. Setup and hold times must be satisfied for proper operation.
SRDY	I	SYNCHRONOUS READY is an active LOW input used to terminate the current bus cycle. The SRDY input is qualified by the SRDYEN input. Setup and hold times must be satisfied for proper operation.
READY	O	READY is an active LOW output which signals the current bus cycle is to be completed. The SRDY, SRDYEN, ARDY, ARDYEN, S1, S0 and RES inputs control READY as explained later in the READY generator section. READY is an open drain output requiring an external pull-up resistor.

Table 1. Pin Description (Continued)

The following pin function descriptions are for the 82C284 clock generator.

Symbol	Type	Name and Function
$\overline{S0}, \overline{S1}$	I	STATUS input prepare the 82C284 for a subsequent bus cycle. $\overline{S0}$ and $\overline{S1}$ synchronize PCLK to the internal processor clock and control \overline{READY} . These inputs have internal pull-up resistors to keep them HIGH if nothing is driving them. Setup and hold times must be satisfied for proper operation.
RESET	O	RESET is an active HIGH output which is derived from the \overline{RES} input. RESET is used to force the system into an initial state. When RESET is active, \overline{READY} will be active (LOW).
\overline{RES}	I	RESET IN is an active LOW input which generates the system reset signal, RESET. Signals to \overline{RES} may be applied asynchronously to CLK. Setup and hold times are given to assure a guaranteed response to synchronous inputs.
V_{CC}		SYSTEM POWER: +5V Power Supply
GND		SYSTEM GROUND: 0V

FUNCTIONAL DESCRIPTION

Introduction

The 82C284 generates the clock, ready, and reset signals required for 80286 processors and support components. The 82C284 contains a crystal controlled oscillator, clock generator, peripheral clock generator, Multibus ready synchronization logic and system reset generation logic.

Clock Generator

The CLK output provides the basic timing control for an 80286 system. CLK has output characteristics sufficient to drive MOS devices. CLK is generated by either an internal crystal oscillator or an external source as selected by the F/C strapping option. When F/C is LOW, the crystal oscillator drives the CLK output. When F/C is HIGH, the EFI input drives the CLK output.

The 82C284 provides a second clock output, PCLK, for peripheral devices. PCLK is CLK divided by two. PCLK has a duty cycle of 50% and MOS output drive characteristics. PCLK is normally synchronized to the internal processor clock.

After reset, the PCLK signal may be out of phase with the internal processor clock. The $\overline{S1}$ and $\overline{S0}$ signals of the first bus cycle are used to synchronize

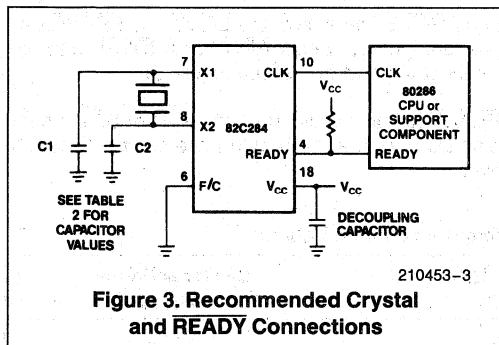
PCLK to the internal processor clock. The phase of the PCLK output changes by extending its HIGH time beyond one system clock (see waveforms). PCLK is forced HIGH whenever either $\overline{S0}$ or $\overline{S1}$ were active (LOW) for the two previous CLK cycles. PCLK continues to oscillate when both $\overline{S0}$ and $\overline{S1}$ are HIGH.

Since the phase of the internal processor clock will not change except during reset, the phase of PCLK will not change except during the first bus cycle after reset.

Oscillator

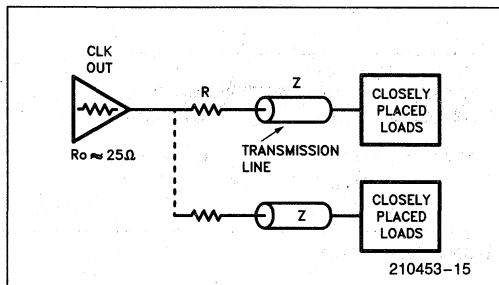
The oscillator circuit of the 82C284 is a linear Pierce oscillator which requires an external parallel resonant, fundamental mode, crystal. The output of the oscillator is internally buffered. The crystal frequency chosen should be twice the required internal processor clock frequency. The crystal should have a typical load capacitance of 32 pF.

X1 and X2 are the oscillator crystal connections. For stable operation of the oscillator, two loading capacitors are recommended, as shown in Table 2. The sum of the board capacitance and loading capacitance should equal the values shown. It is advisable to limit stray board capacitances (not including the effect of the loading capacitors or crystal capacitance) to less than 10 pF between the X1 and X2 pins. Decouple V_{CC} and GND as close to the 82C284 as possible.



CLK Termination

Due to the CLK output having a very fast rise and fall time, it is recommended to properly terminate the CLK line at frequencies above 10 MHz to avoid signal reflections and ringing. Termination is accomplished by inserting a small resistor (typically 10 Ω –74 Ω) in series with the output, as shown in Figure 4. This is known as series termination. The resistor value plus the circuit output impedance should be made equal to the impedance of the transmission line.

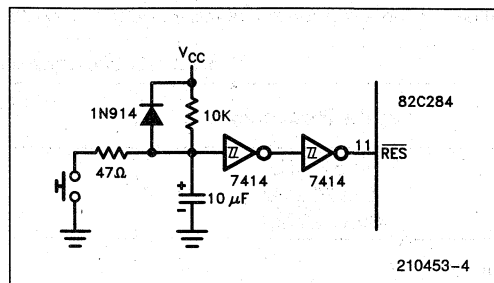


Reset Operation

The reset logic provides the RESET output to force the system into a known, initial state. When the RES input is active (LOW), the RESET output becomes active (HIGH). RES is synchronized internally at the falling edge of CLK before generating the RESET output (see waveforms). Synchronization of the RES input introduces a one or two CLK delay before affecting the RESET output.

At power up, a system does not have a stable V_{CC} and CLK. To prevent spurious activity, RES should

be asserted until V_{CC} and CLK stabilize at their operating values. 80286 processors and support components also require their RESET inputs be HIGH a minimum of 16 CLK cycles. A network such as shown in Figure 5 will keep RES LOW long enough to satisfy both needs.



Ready Operation

The 82C284 accepts two ready sources for the system ready signal which terminates the current bus cycle. Either a synchronous (SRDY) or asynchronous ready (ARDY) source may be used. Each ready input has an enable (SRDYEN and ARDYEN) for selecting the type of ready source required to terminate the current bus cycle. An address decoder would normally select one of the enable inputs.

READY is enabled (LOW), if either SRDY + SRDYEN = 0 or ARDY + ARDYEN = 0 when sampled by the 82C284 READY generation logic. READY will remain active for at least two CLK cycles.

The READY output has an open-drain driver allowing other ready circuits to be wire or'ed with it, as shown in Figure 3. The READY signal of an 80286 system requires an external pull-up resistor. To force the READY signal inactive (HIGH) at the start of a bus cycle, the READY output floats when either S1 or S0 are sampled LOW at the falling edge of CLK. Two system clock periods are allowed for the pull-up resistor to pull the READY signal to V_{IH}. When RESET is active, READY is forced active one CLK later (see waveforms).

Figure 6 illustrates the operation of SRDY and SRDYEN. These inputs are sampled on the falling edge of CLK when S1 and S0 are inactive and PCLK

is HIGH. $\overline{\text{READY}}$ is forced active when both $\overline{\text{SRDY}}$ and $\overline{\text{SRDYEN}}$ are sampled as LOW.

Figure 7 shows the operation of $\overline{\text{ARDY}}$ and $\overline{\text{ARDYEN}}$. These inputs are sampled by an internal synchronizer at each falling edge of CLK. The output of the synchronizer is then sampled when PCLK is HIGH. If the synchronizer resolved both the $\overline{\text{ARDY}}$

and $\overline{\text{ARDYEN}}$ as active, the $\overline{\text{SRDY}}$ and $\overline{\text{SRDYEN}}$ inputs are ignored. Either $\overline{\text{ARDY}}$ or $\overline{\text{ARDYEN}}$ must be HIGH at the end of T_S (see Figure 7).

$\overline{\text{READY}}$ remains active until either $\overline{\text{S1}}$ or $\overline{\text{S0}}$ are sampled LOW, or the ready inputs are sampled as inactive.

Table 2. 82C284 Crystal Loading Capacitance Values

Crystal Frequency	C1 Capacitance (Pin 7)	C2 Capacitance (Pin 8)
1 to 8 MHz	60 pF	40 pF
8 to 20 MHz	25 pF	15 pF
Above 20 MHz	15 pF	15 pF

NOTE:

Capacitance values must include stray board capacitance.

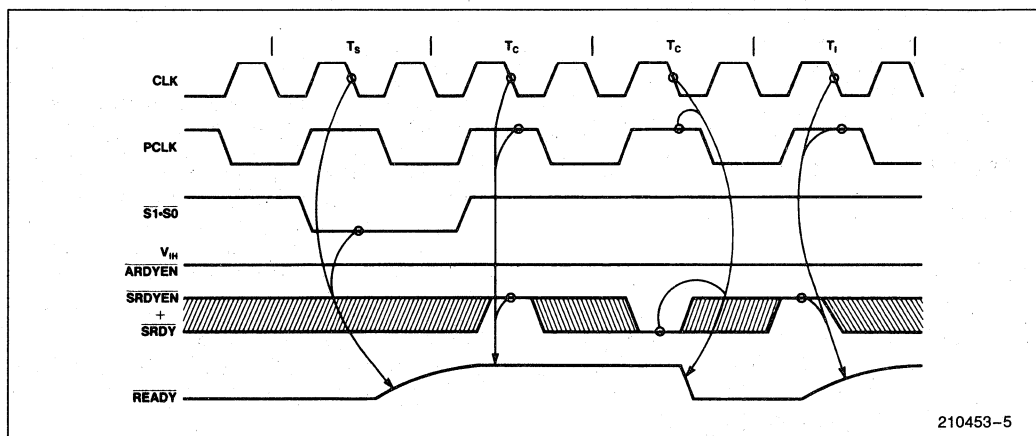


Figure 6. Synchronous Ready Operation

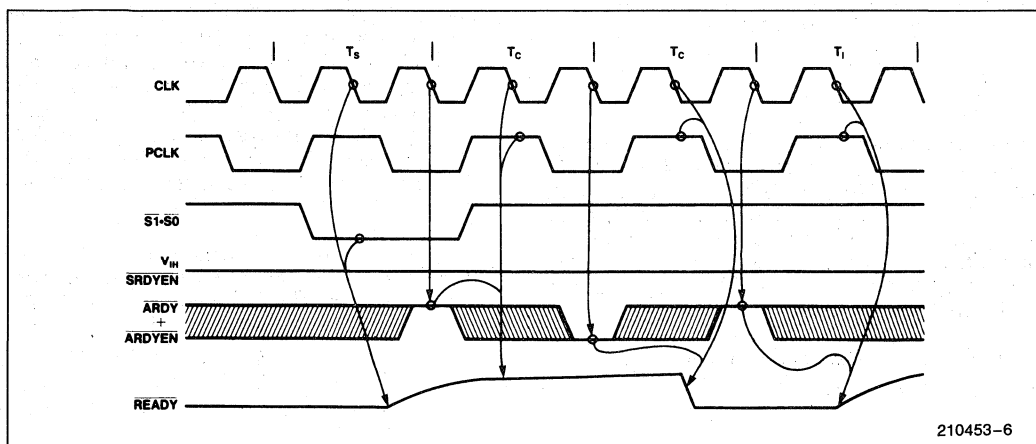


Figure 7. Asynchronous Ready Operation

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias 0°C to +70°C
 Storage Temperature -65°C to +150°C
 All Output and Supply Voltages -0.5V to +7V
 All Input Voltages -1.0V to +5.5V
 Power Dissipation 1 Watt

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

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

D.C. CHARACTERISTICS $T_{CASE} = 0^{\circ}\text{C}$ to $+85^{\circ}\text{C}$, * $V_{CC} = 5\text{V} \pm 5\%$

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input LOW Voltage		0.8	V	
V_{IH}	Input HIGH Voltage	2.0		V	
V_{IHR}	RES and EFI Input HIGH Voltage	2.6		V	
V_{OL}	RESET, PCLK Output LOW Voltage		0.45	V	$I_{OL} = 5\text{ mA}$
V_{OH}	RESET, PCLK Output HIGH Voltage	2.4		V	$I_{OH} = -1\text{ mA}$
		$V_{CC} - 0.5$		V	$I_{OH} = -0.2\text{ mA}$
V_{OLR}	READY, Output LOW Voltage		0.45	V	$I_{OL} = 9\text{ mA}$
V_{OLC}	CLK Output LOW Voltage		0.45	V	$I_{OL} = 5\text{ mA}$
V_{OHC}	CLK Output HIGH Voltage	4.0		V	$I_{OH} = -800\text{ }\mu\text{A}$
I_{IL}	Input Sustaining Current on S0 and S1 Pins	-60	-500	μA	$V_{IN} = 0\text{V}$
I_{LI}	Input Leakage Current		± 10	μA	$0 \leq V_{IN} \leq V_{CC}^{(1)}$
I_{CC}	Power Supply Current		75	mA	at 25 MHz Output CLK Frequency
C_I	Input Capacitance		10	pF	$F_C = 1\text{ MHz}$

* T_A is guaranteed from 0°C to +70°C as long as T_{CASE} is not exceeded.

NOTE:

- Status lines S0 and S1 excluded because they have internal pull-up resistors.

A.C. CHARACTERISTICS $V_{CC} = 5V \pm 5\%$, $T_{CASE} = 0^{\circ}C$ to $+85^{\circ}C$.*

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

82C284 A.C. Timing Parameters

Symbol	Parameter	8 MHz		10 MHz		12.5 MHz		Units	Test Conditions
		Min	Max	Min	Max	Min	Max		
1	EFI to CLK Delay		25		25		25	ns	At 1.5V (1)
2	EFI LOW Time	28		22.5		13		ns	At 1.5V (1, 7)
3	EFI HIGH Time	28		22.5		22		ns	At 1.5V (1, 7)
4	CLK Period	62	500	50	500	40	500	ns	
5	CLK LOW Time	15		12		11		ns	At 1.0V (1, 2, 7, 8, 9, 10)
6	CLK HIGH Time	25		16		13		ns	At 3.6V (1, 2, 7, 8, 9, 10)
7	CLK Rise Time		10		8		8	ns	1.0V to 3.6V (1, 2, 10, 11)
8	CLK Fall Time		10		8		8	ns	3.6V to 1.0V (1, 9, 10, 11)
9	Status Setup Time	22		—		—		ns	(Note 1)
9a	Status Setup Time for Status Going Active	—		20		22		ns	(Note 1)
9b	Status Setup Time for Status Going Inactive	—		20		18		ns	(Note 1)
10	Status Hold Time	1		1		3		ns	(Note 1)
11	SRDY or SRDYEN Setup Time	20		17.5		17		ns	(Note 1)
12	SRDY or SRDYEN Hold Time	0		2		2		ns	(Notes 1, 11)
13	ARDY or ARDYEN Setup Time	0		0		0		ns	(Notes 1, 3)
14	ARDY or ARDYEN Hold Time	30		30		25		ns	(Notes 1, 3)
15	RES Setup Time	20		20		18		ns	(Notes 1, 3)
16	RES Hold Time	10		10		8		ns	(Notes 1, 3)
17	READY Inactive Delay	5		5		5		ns	At 0.8V (4)
18	READY Active Delay	0	24	0	24	0	18	ns	At 0.8V (4)
19	PCLK Delay	0	45	0	35	0	23	ns	(Note 5)
20	RESET Delay	5	34	5	27	3	22	ns	(Note 5)
21	PCLK LOW Time	t4–20		t4–20		t4–20		ns	(Notes 5, 6)
22	PCLK HIGH Time	t4–20		t4–20		t4–20		ns	(Notes 5, 6)

* T_A is guaranteed from $0^{\circ}C$ to $70^{\circ}C$ as long as T_{CASE} is not exceeded.

NOTES:

1. CLK loading: $C_L = 100$ pF. The 82C284's X1 and X2 inputs are designed primarily for parallel-resonant crystals. Serial-resonant crystals may also be used, however, they may oscillate up to 0.01% faster than their nominal frequencies when used with the 82C284. For either type of crystal, capacitive loading should be as specified by Table 2.
2. With the internal crystal oscillator using recommended crystal and capacitive loading; or with the EFI input meeting specifications t2 and t3. The recommended crystal loading for CLK frequencies of 8 MHz–20 MHz are 25 pF from pin X1 to ground, and 15 pF from pin X2 to ground; for CLK frequencies above 20 MHz 15 pF from pin X1 to ground, and 15 pF from pin X2 to ground. These recommended values are ± 5 pF and include all stray capacitance. Decouple V_{CC} and GND as close to the 82C284 as possible.
3. This is an asynchronous input. This specification is given for testing purposes only, to assure recognition at specific CLK edge.

NOTES:

4. Pull-up Resistor values for READY Pin:

CPU Frequency	8 MHz	10 MHz	12.5 MHz
Resistor	910Ω	700Ω	600Ω
CL	150 pF	150 pF	150 pF
I _{OL}	7 mA	7 mA	9 mA

5. PCLK and RESET loading: C_L = 75 pF.

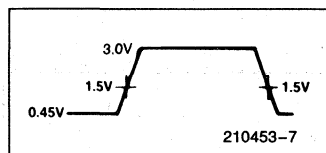
6. t₄ refers to any allowable CLK period.

7. When driving the 82C284 with EFI, provide minimum EFI HIGH and LOW times as follows:

CLK Output Frequency	16 MHz	20 MHz	25 MHz
Min. Required EFI HIGH Time	28 ns	22.5 ns	22 ns
Min. Required EFI LOW Time	28 ns	22.5 ns	13 ns

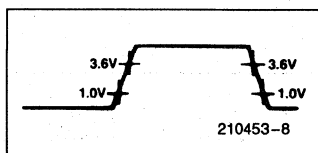
8. When using a crystal (with recommended capacitive loading per Table 2) appropriate for the speed of the 80286, CLK output HIGH and LOW times guaranteed to meet the 80286 requirements.

Reset Drive EFI Drive and Measurement Points



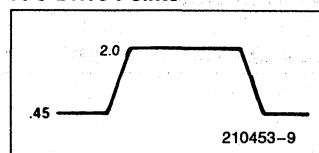
Note 9

CLK Output Measurement Points

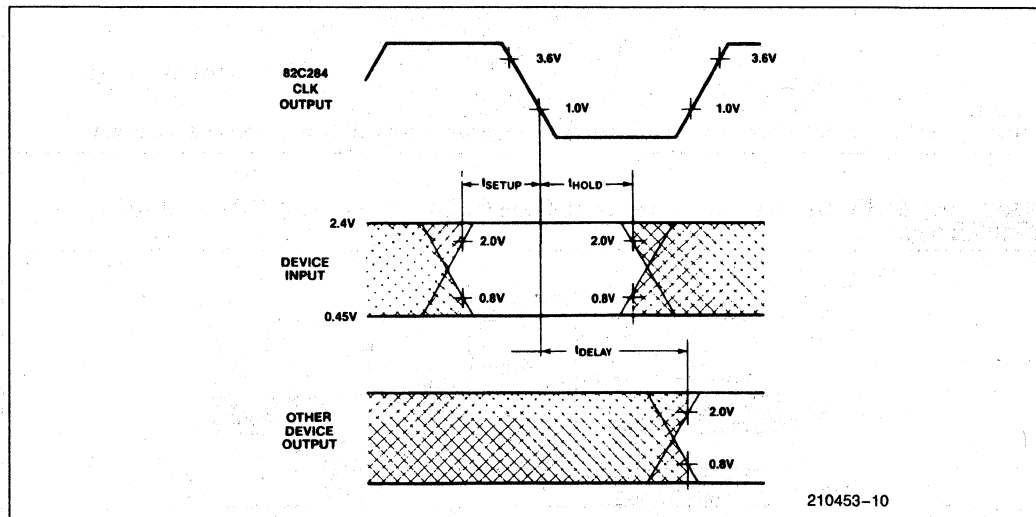


Note 10

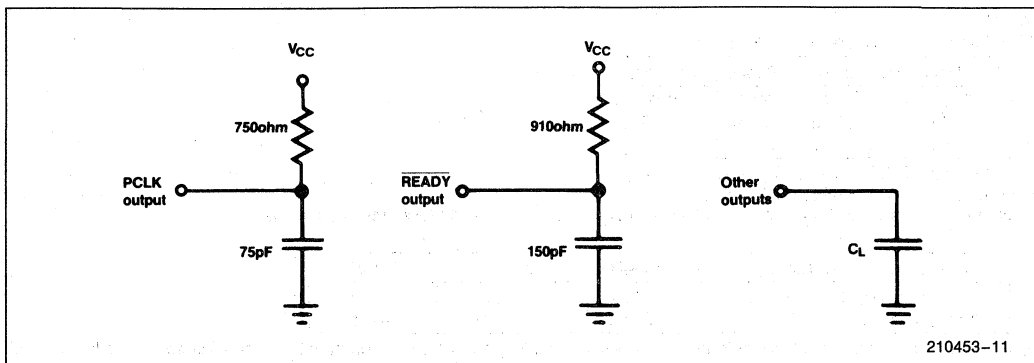
F/C Drive Points



Note 11



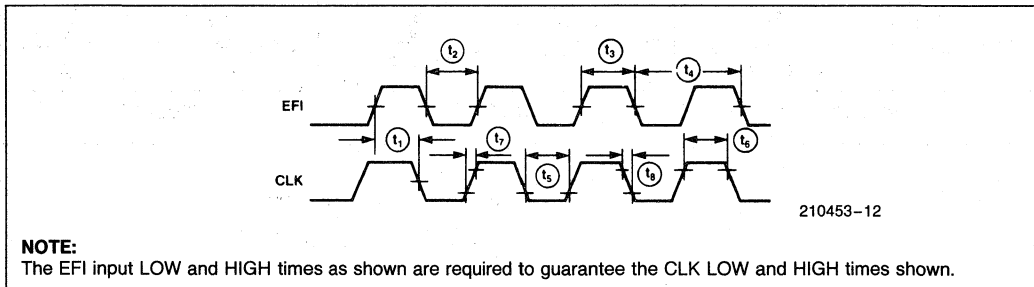
Note 12. AC Setup, Hold and Delay Time Measurement—General



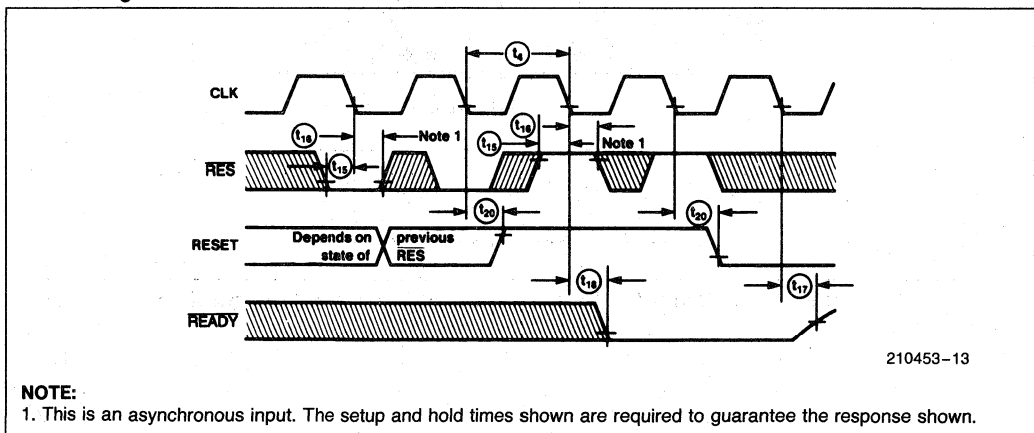
Note 13. AC Test Loading on Outputs

WAVEFORMS

CLK as a Function of EFI

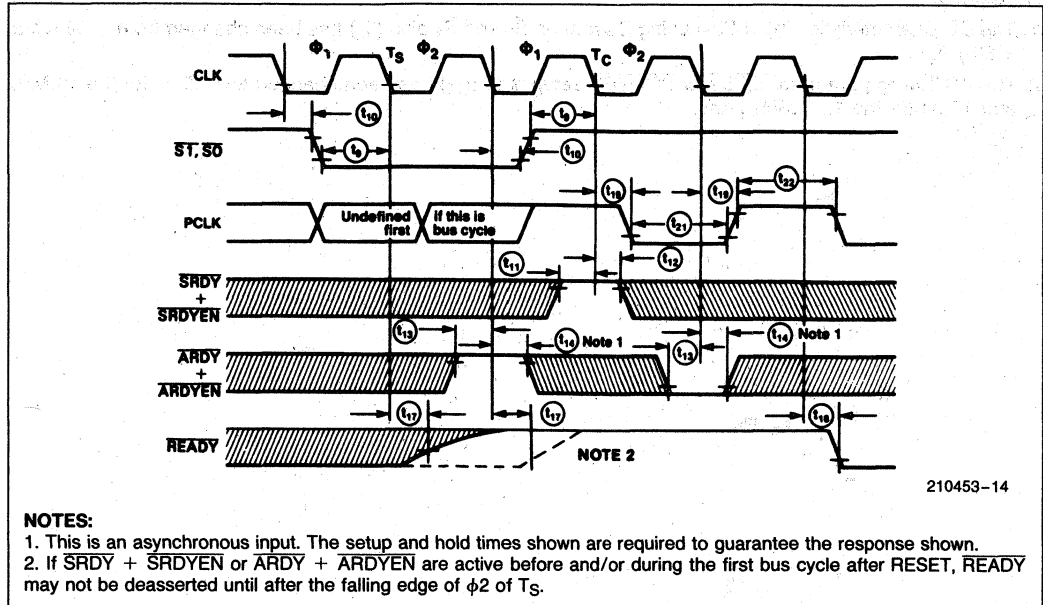


RESET and $\overline{\text{READY}}$ Timing as a Function of $\overline{\text{RES}}$ with $\overline{\text{S1}}$, $\overline{\text{S0}}$, $\overline{\text{ARDY}} + \overline{\text{ARDYEN}}$, and $\overline{\text{SRDY}} + \overline{\text{SRDYEN}}$ High



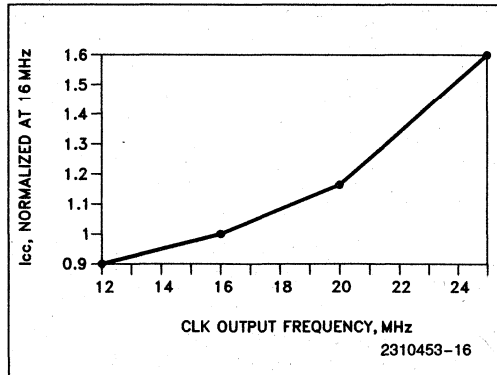
WAVEFORMS (Continued)

READY and PCLK Timing with RES High

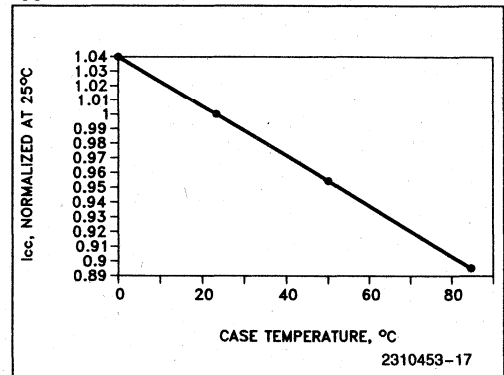


3

ICC vs Frequency @ Nominal Conditions



ICC vs Case Temperature @ 25 MHz



DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -010 data sheet. Please review this summary carefully.

1. The DC Characteristics Input Sustaining Current on \overline{S}_0 and \overline{S}_1 pins (I_{IL}) has been changed from $-30\ \mu\text{A}$ to $-60\ \mu\text{A}$.
2. The AC Timing parameter $\overline{\text{SRDY}}$ or $\overline{\text{SRDYEN}}$ setup time (t_{11}) has been changed to 17.5 ns for the 10 MHz and 17 ns for the 12.5 MHz parts.

Development Tools for the 8086, 80186, 80188, and 80286

4



8086/80186 SOFTWARE DEVELOPMENT PACKAGES

```
Debug Window Go Set View Browse Help
j : 0006
index : 0008
char c : 0005H:0004H 'Y'

168 3      }
169 2      while
170 2      p
171 2      printf(" ");
172 2      for (j = index; ((j < index + 0x10) && (j
173 2      { c = file_buf[j];

175 3      c = ' ';
176 3      putchar(c);
177 3      }
178 2      printf("\n");
179 2      }

*go til #172
( Break at :0D#172 )
*ua0 = j
*ua1 = index
*ua2 = char c
*

Mod: 0D Proc: DUMP_REC Line: #174 PSTEP
```

AX: 0054
BX: 3E68
CX: 0001
DX: 0000
SI: 0006
DI: 0000
BP: 3E50
SP: 3E46
SS: 6667
DS: 6667
ES: 6667
CS: 7377
IP: 05F8
odisz0Pc
DB86
X832

280809-1

COMPLETE SOFTWARE DEVELOPMENT SUPPORT FOR THE 8086/80186 FAMILY OF MICROPROCESSORS

Intel supports application development for the 8086/80186 family of microprocessors (8086/88, 80186/188, 80C186/C188, 80C186EB/C188EB and real mode 80286 and 80386 designs) with a complete set of development languages and utilities. These tools include a macro assembler and compilers for C, PL/M, FORTRAN and Pascal. A linker/relocator program, library manager, numerics support libraries, and object-to-hex utility are also available. Intel software tools generate fast and efficient code. They are designed to give maximum control over the processor. Most importantly, they are designed to get your application up and running in an embedded system fast and with maximum design productivity.

FEATURES

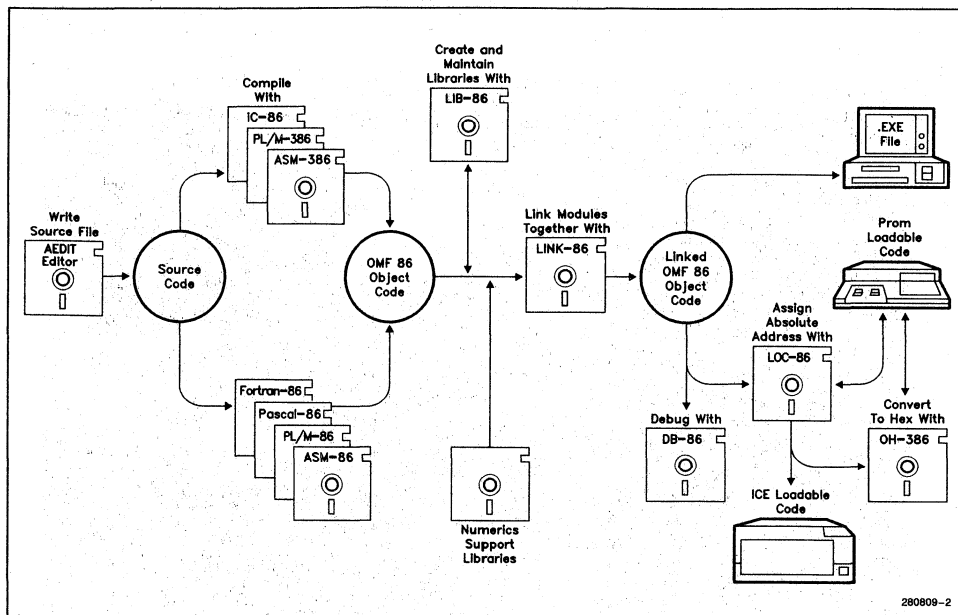


Figure 1. The Application Development Process

FEATURES

- Macro assembler for speed-critical code
- NEW windowed, interactive source level debugger works with all Intel languages
- ANSI standard iC-86 package for structured C programming, with many processor specific extensions
- PL/M compiler for high-level language programs with support for many low-level hardware functions
- FORTRAN for ANSI-compatible, numeric intensive applications
- Pascal for developing modular, portable applications that are easy to maintain
- Linker program for linking modules generated by Intel compilers and assemblers
- Locator for generating programs with absolute addresses for execution from ROM based systems
- AEDIT Source Code and text editor
- Library manager for creating and maintaining object module libraries
- Complete 8087/80C187 numeric libraries including software emulator support

- Object-to-hex conversion utility for burning code into (E)PROMS
- Hosted on IBM PC XT AT* or compatibles running DOS, DEC VAX* or MicroVAX* systems running VMS, and Intel Systems 86/3XX or 286/3XX running iRMX® Operating System

ASM-86 MACRO ASSEMBLER

ASM-86 is the macro assembler for the 8086/80186 family of components. It is used to translate symbolic assembly language source into relocatable object code where utmost speed, small code size and hardware control are critical. Intel's exclusive macro facility in ASM-86 saves development and maintenance time, since common code sequences need only be developed once. The assembler's simplified instruction set makes program development easier. This assembler also saves development time by performing extensive checks on consistent usage of variables and labels. Inconsistencies are detected when the program is assembled, before linking or debugging is started.

FEATURES

DB86 SOURCE LEVEL DEBUGGER

DB-86 is an on-host software execution environment with source level debug capabilities for object modules produced by iC-86, ASM-86, PL/M-86, Pascal-86 and FORTRAN-86. Its powerful, source-oriented interface allows users to focus their efforts on finding bugs rather than spending time learning and manipulating the debug environment.

- **Ease of learning.** Drop-down menus make the tool easy to learn for new or casual users. A command line interface is also provided for more complex problems.
- **Extensive debug modes.** Watch windows, conditional breakpoints (breakpoints triggered by program conditions), trace points, and fixed and temporary breakpoints can be set and modified as needed.
- **See into your program.** You can browse source and call stack, observe processor registers, output screen, and watch window variables accessed by either the pull down menu or by a single keystroke using function keys.
- **Full debug symbolics for maximum productivity.** The user need not know whether a variable is an unsigned integer, a real, or a structure; the debugger utilizes the wealth of variable typing information available in Intel languages to display program variables in their respective type formats.
- **Support for overlaid programs and the numeric coprocessor.**

iC-86 SOFTWARE PACKAGE

Intel's iC-86 brings the full power of the C programming language to 8086/88, 80C186/C188 and 80C186EB/C188EB-based microprocessor systems. It can also be used to develop real mode programs for execution on the 80286 or 80386. iC-86 has been developed specifically for embedded microprocessor-based applications. iC-86 meets the ANSI C standard. Key features of the iC-86 compiler include:

- **Highly Optimized.** Four levels of optimization are available. Important optimization features include a jump optimizer and improved register manipulation via register history.
- **ROMable Code and Libraries.** The iC-86 compiler produces ROMable code which can be loaded directly into embedded target systems. Libraries are also completely ROMable, retargetable and reentrant.
- **Supports Small, Medium, Compact, and Large memory segmentation models.**
- **Symbolics.** The iC-86 compiler boosts programming productivity by providing extensive debug information, including type information and symbols. The symbolics information can be used to debug using Intel ICE™ emulators and the new DB-86 Source level debugger.
- **Built-in functions.** iC-86 is loaded with built-in functions. The flags register, I/O ports, interrupts, and numerics chip can be controlled directly, without the need for assembly language coding. You spend more of your productive time programming in C and less with Assembler. Built-in functions also improve compile-time and run-time performance since the compiler generates in-line code instructions instead of function calls to assembly instructions.
- **Standard Language.** iC-86 conforms to the ANSI standard for the C language. iC-86 code is fully linkable with other modules written in other Intel 8086/186 languages, allowing programmers to use the optimal language for any task.

FEATURES

PL/M-86 SOFTWARE PACKAGE

PL/M-86 is a high-level programming language designed to support the software requirements of advanced 16-bit microprocessors. PL/M-86 provides the productivity advantages of a high-level language while providing the low-level hardware access features of assembly language. Key features of PL/M-86 include:

- **Structured programming.** PL/M-86 supports modular and structured programming, making programs easier to understand, maintain and debug.
- **Built-in functions.** PL/M-86 includes an extensive list of functions, including TYPE CONVERSION functions, STRING manipulations, and functions for interrogating 8086/186 hardware flags.
- **Interrupt handling.** The INTERRUPT attribute allows you to define interrupt handling procedures. The compiler generates code to save and restore all registers for INTERRUPT procedures.
- **Compiler controls.** Compile-time options increase the flexibility of the PL/M-86 compiler. They include: optimization, conditional compilation, the inclusion of common PL/M source files from disk, cross-reference of symbols, and optional assembly language code in the listing file.
- **Data types.** PL/M-86 supports seven data types, allowing the compiler to perform three different kinds of arithmetic: signed, unsigned and floating point.
- **Language compatibility.** PL/M-86 object modules are compatible with all other object modules generated by Intel 8086/186 languages.

FORTRAN-86 SOFTWARE PACKAGE

FORTRAN-86 meets the ANSI FORTRAN 77 Language Subset Specification and includes almost all of the features of the full standard. This compatibility assures portability of existing FORTRAN programs and shortens the development process, since programmers are immediately productive without retraining.

FORTRAN-86 provides extensive support for numeric processing tasks and applications, with features such as:

- Support for single, double, double extended precision, complex, and double complex floating-point data types

- Support for proposed REALMATH IEEE floating point standard
- Full support for all other data types: integer, logical and character
- Optional hardware (8087 numeric data processor) or software (simulator) floating-point support at link time

PASCAL-86 SOFTWARE PACKAGE

Pascal-86 conforms to the ISO Pascal standard, facilitating application portability, training and maintenance. It has also been enhanced with microcomputer support features such as interrupt handling, direct port I/O and separate compilation.

A well-defined and documented run-time operating system interface allows the user to execute applications under user-designed operating systems as an alternate to the development system environment. Program modules compiled under Pascal-86 are compatible and linkable with modules written in other Intel 8086/186 languages, so developers can implement each module in the language most appropriate for the task at hand.

Pascal-86 object modules contain symbol and type information for program debugging using Intel ICE™ emulators and the DB-86 debugger.

LINK-86 LINKER

Intel's LINK-86 utility is used to combine multiple object modules into a single program and resolve references between independently compiled modules. The resulting linked module can be either a bound load-time-locatable module or simply a relocatable module. A .EXE option allows modules to be generated which can be executed directly on a DOS system.

LINK-86 greatly increases productivity by allowing you to use modular programming. The incremental link capability allows new modules to be easily added to existing software. Because applications can be broken into separate modules, they're easier to design, test and maintain. Standard modules can be reused in different applications, saving software development time.

FEATURES

LOC-86 LOCATOR

The LOC-86 utility changes relocatable 8086/186 object modules into absolute object modules. Its default address assignment algorithm will automatically assign absolute addresses to the object modules prior to loading of the code into the target system. This frees you from concern about the final arrangement of the object code in memory. You still have the power to override the control and specify absolute addresses for various Segments, Classes, and Groups in memory. You may also reserve various parts of memory.

LOC-86 is a powerful tool for embedded development because it simplifies set up of the bootstrap loader and initialization code for execution from ROM based systems. The locator will also optionally generate a print file containing diagnostic information to assist in program debugging.

NUMERICS SUPPORT LIBRARIES

The 8087 and 80C187 Support Libraries greatly facilitate the use of floating-point calculations from programs written in Assembler, PL/M, and C. It adds to these languages many of the functions that are built into applications programming languages, such as Pascal and FORTRAN. Numerics functions can be processed by either the 8087 or 80C187 numeric coprocessors, or by the corresponding software emulators. The decimal conversion library aids the translation between decimal and binary formats. A Common Elementary Function library provides support for transcendental, rounding and other common functions, not directly handled by the numeric processor. An Error Handler Module makes it easy to write interrupt routines that recover from floating-point error conditions.

LIB-86 LIBRARIAN

The Intel LIB-86 utility creates and maintains libraries of software object modules. Standard modules can be placed in a library and linked to your application using the LINK-86 utility.

AEDIT SOURCE CODE AND TEXT EDITOR

AEDIT is a full-screen text editing system designed specifically for software engineers and technical writers. With the facilities for automatic program block indentation, HEX display and input, and full macro support, AEDIT is an essential tool for any programming environment. And with AEDIT, the output file is the pure ASCII text (or HEX code) you input—no special characters or proprietary formats.

Dual file editing means you can create source code and its supporting documents at the same time. Keep your program listing with its errors in the background for easy reference while correcting the source in the foreground. Using the split-screen windowing capability, it is easy to compare two files, or copy text from one to the other. The DOS system-escape command eliminates the need to leave the editor to compile a program, get a directory listing, or execute any other program executable at the DOS system level.

OH-86 OBJECT-TO-HEXADECIMAL CONVERTER

The OH-86 utility converts Intel 8086/186 object modules into standard hexadecimal format, allowing the code to be loaded directly into PROM using industry standard PROM programmers.

WORLDWIDE SERVICE, SUPPORT, AND TRAINING

To augment its development tools, Intel offers a full array of seminars, classes, and workshops, field application engineering expertise, hotline technical support and on-site service.

Intel also offers a Software Support package which includes technical software information, telephone support, automatic distribution of software and documentation updates, access to the "ToolTalk" electronic bulletin board, *iComments* publication, remote diagnostic software, and a development tools troubleshooting guide.

Intel's Hardware Support package includes technical hardware information, telephone support, warranty on parts, labor, material, and on-site hardware support.



ORDERING INFORMATION

LIB-86 LIBRARIAN

R286ASM86EU FORTRAN-86

D86ASM86KIT ASM-86

Assembler
for PC XT or
AT system (or
compatible)
running DOS
3.0 or higher

VVSASM86 ASM-86

Assembler
for VAX/
VMS

MVVSASM86 ASM-86

Assembler
for
MicroVAX/
VMS

R86ASM86SU ASM-86

Assembler
for Intel 86/
3XX systems
running
iRMX 86
operating
system

R286ASM86EU ASM-86

Assembler
for Intel 286/
3XX systems
running
iRMX® II
operating
system

Note: ASM-86 includes Macro Assembler, Link-86,
Loc-86, Lib-86, Cross-Reference utility, OH-86,
Numerics Support, and DB-86 Source Level
Debugger. (DB-86 available in DOS version
only.)

D86C86NL iC-86

Software
Package for
IBM PC XT/
AT running
PC DOS 3.0
or higher

VVSC86 iC-86

Software
Package for
VAX/VMS

MVVSC86 iC-86

Software
Package for
MicroVAX/
VMS

VVSPLM86 PL/M-86

Software
Package for
VAX/VMS

MVVSPLM86 PL/M-86

Software
Package for
MicroVAX/
VMS

R86PLM86SU PL/M-86

Software
Package for
Intel System
8086/3XX
running
iRMX 86
operating
system

D86FOR86NL FORTRAN-86

Software
Package for
PC XT AT (or
compatible)
running PC-
DOS 3.0 or
higher

VVSFORT86 FORTRAN-86

Software
Package for
VAX/VMS
4.3 and later

MVVSFORT86 FORTRAN-86

Software
Package for
MicroVAX/
VMS

R86FOR86SU FORTRAN-86

Software
Package for
Intel System
86/3XX
running
iRMX 86
operating
system

D86PAS86NL PASCAL-86

Software
Package for
IBM PC XT
AT running
PC DOS 3.0 or
higher

VVSPAS86 PASCAL-86

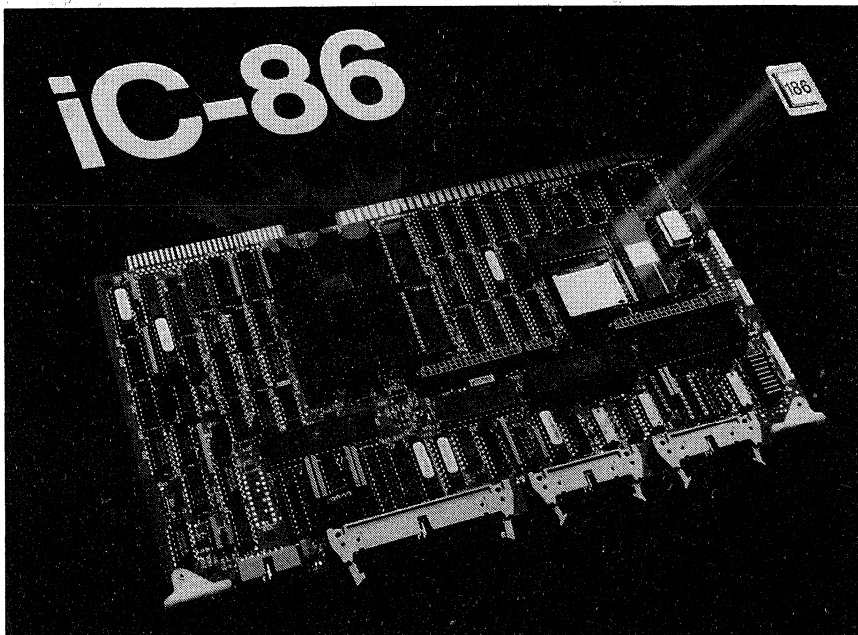
Software
Package for
VAX/VMS



ORDERING INFORMATION

R86C86SU	iC-86	Software Package for Intel System 8086/3XX running iRMX 86 operating system	MVVPAS86	PASCAL-86	Software Package for MicroVAX/ VMS
D86PLM86NL	PL/M-86	Software Package for IBM PC XT AT running PC DOS 3.0 or higher	R86PAS86SU	PASCAL-86	Software Package for Intel System 86/3XX running iRMX 86
			D86EDNL		AEDIT Source Code Editor for IBM PC XT/AT running PC DOS 3.0 or higher

INTEL iC-86/286 C COMPILER



280787-1

INTEL iC-86/286 COMPILER

The Intel iC-86/286 compiler is the C compiler to use for 8086/186/286 embedded microprocessor designs. In addition to outstanding execution speed, Intel's iC-86/286 compiler generates compact, efficient code which can be easily loaded into ROM-based systems. The iC-86/286 compiler is also fully supported by the Intel DB86 windowed source-level software debugger and in-circuit emulation tools.

iC-86/286 COMPILER FEATURES

- Optimized for embedded systems
- Built-in functions for automatic machine code generation
- ROMable code and libraries
- Integrated debugging with Intel ICET[™] and I²ICET[™]
- Compliance with draft ANSI standard
- Supports Small, Medium, Compact, and Large memory models
- PL/M compatible subsystems
- Selector data type support
- Linkable with other Intel 8086/286 languages such as ASM and PL/M
- ROMable and reentrant libraries
- Ability to mix memory models with "near" and "far" pointers
- C and PL/M calling conventions for compatibility with PL/M and other C programs
- iRMX[®] interface libraries included

ICET[™], iRMX[™], 386[™], iPAT[™], and I²ICET[™] are trademarks of Intel Corporation.

FEATURES

BUILT-IN FUNCTIONS

The iC-86/286 compiler features more than 35 processor-specific functions that directly generate machine code within the C language.

Built-in functions eliminate the need for inline assembly language coding or making calls to assembly functions. This increases code performance and reduces programming time.

With built-ins you can enable or disable interrupts and directly control hardware I/O without having to exit C for assembler. This means you can write high performance software for real time applications without having to keep track of every architectural detail, as you would in assembly language. For example, to generate an INT instruction, you simply type:

```
causeinterrupt (number)
```

Or, the following iC-86 instruction will cause the processor to come to a halt with interrupts enabled:

```
halt ( )
```

EMBEDDED COMPONENT SUPPORT

iC-86/286 compiler was designed specifically for embedded microprocessor applications. It produces ROMable code which can be loaded directly into target systems via Intel ICE emulators and debugged *without modification* for fast, easy, development and debugging.

HIGHLY OPTIMIZED

The iC-86/286 compiler has four levels of optimization for tailoring performance to your application. Important optimization features include a jump optimizer and improved register manipulation using register history.

RUN-TIME SUPPORT

Run-time libraries for the iC-86/286 compiler are designed for use in many environments. Both DOS and iRMX interface libraries are included so programs executing on those systems can take advantage of operating system features. The interface libraries conform to the ANSI standard. They also meet the IEEE standard POSIX interface so you can easily retarget the libraries for use in applications that do not run on DOS or iRMX.

The libraries are completely ROMable and re-entrant making it easy to adapt them for embedded, multi-tasking or real-time applications.

Both the DOS and iRMX-I operating system interface libraries are provided with iC-86 hosted on DOS. The iRMX-I hosted version of iC-86 includes the iRMX interface libraries only.

iC-286 compiler includes iRMX-II interface libraries only.

INTEGRATED DEBUG TOOLS

The iC-86/286 compiler is part of a completely integrated set of development tools from Intel (Figure 1).

Code output from the compiler can be easily linked with modules written in assembler and high-level languages, such as PL/M, Fortran, and Pascal.

Linked modules and programs can be debugged using Intel's DB86 windowed source-level software debugger. The debugger uses an advanced interface with windows and pull-down menus for the ultimate in debug productivity. Watch windows can be opened to observe changing program variables and processor registers. You can readily switch between program modules and view the calling sequence and call stack.

Naturally code generated by iC-86/286 works completely with Intel's I2ICE, ICE-186, ICE-286 and ICE-386 family of in-circuit emulators as well as the iPAT(tm) performance analysis tool. This complete set of tools gives you the power to quickly debug, test, integrate and optimize your application code for the target system.

FEATURES

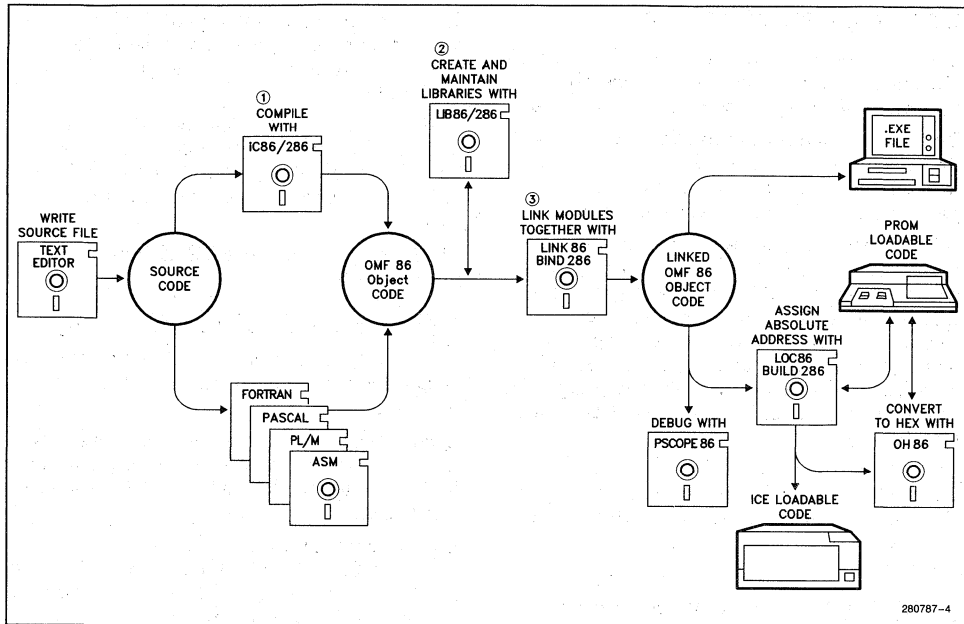


Figure 1: The Application Development Process

SERVICE AND SUPPORT

Intel's development tools are backed by our worldwide service and support organization dedicated to solving any problems encountered by our customers. Several hardware and

software service programs are available which include hotline support, consulting, training, technical newsletters, bulletin boards and other services. The iC-86/286 compiler includes 90 days of software support under warranty.

SPECIFICATIONS

ENVIRONMENT

Hardware Requirements	DOS Version:	IBM PC XT or AT (or 100% Compatible) running DOS 3.1 or greater.
	iRMX Version:	iRMX-I system for iC-86 iRMX-II system for iC-286
Memory Requirements	DOS Version:	256 KB
	iRMX Version:	374 KB
Media	DOS Version:	5¼" DS/DD Diskettes
		3½" DS/DD Diskettes
	iRMX Version:	DS/DD iRMX Standard Format

STANDARDS

iC-86/286 conforms to the X3J11 ANSI draft proposal for the C programming language.

SPECIFICATIONS

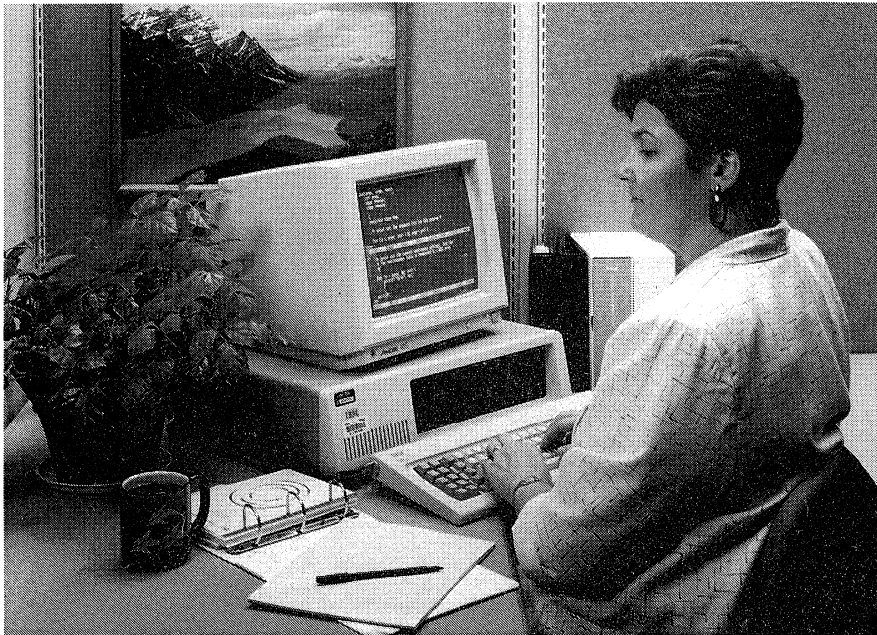
ORDERING INFORMATION

Order Code	Host Environment	Target Code
D86C86NL	DOS	8086/186
D86C286NL	DOS	80286
R86C86	iRMX-I	8086/186
R286C286	iRMX-II	80286

Other programming tools:

Order Code	Host Environment	Description
D86PAK86NL	DOS	8086/186 Assembler, DB86 Debugger, Utilities
		AEDIT text editor
D86ASM86KIT	DOS	8086/186 Assembler, DB86 Debugger, Utilities
D86ASM286NL	DOS	80286 Assembler
R86ASM86	iRMX-I	8086/186 Assembler, Utilities
R286ASM286	iRMX-II	80286 Assembler, Utilities
RMXIISFTSCP	iRMX-II	80286 Soft-Scope Debugger

AEDIT SOURCE CODE AND TEXT EDITOR



280804-1

PROGRAMMER SUPPORT

AEDIT is a full-screen text editing system designed specifically for software engineers and technical writers. With the facilities for automatic program block indentation, HEX display and input, and full macro support, AEDIT is an essential tool for any programming environment. And with AEDIT, the output file is the pure ASCII text (or HEX code) you input—no special characters or proprietary formats.

Dual file editing means you can create source code and its supporting documents at the same time. Keep your program listing with its errors in the background for easy reference while correcting the source in the foreground. Using the split-screen windowing capability, it is easy to compare two files, or copy text from one to the other. The DOS system-escape command eliminates the need to leave the editor to compile a program, get a directory listing, or execute any other program executable at the DOS system level.

There are no limits placed on the size of the file or the length of the lines processed with AEDIT. It even has a batch mode for those times when you need to make automatic string substitutions or insertions in a number of separate text files.

AEDIT FEATURES

- Complete range of editing support—from document processing to HEX code entry and modification
- Supports system escape for quick execution of PC-DOS System level commands
- Full macro support for complex or repetitive editing tasks
- Hosted on PC-DOS and RMX operating systems
- Dual file support with optional split-screen windowing
- No limit to file size or line length

FEATURES

- Quick response with an easy to use menu driven interface
- Configurable and extensible for complete control of the editing process

POWERFUL TEXT EDITOR

As a text editor, AEDIT is versatile and complete. In addition to simple character insertion and cursor positioning commands, AEDIT supports a number of text block processing commands. Using these commands you can easily move, copy, or delete both small and large blocks of text. AEDIT also provides facilities for forward or reverse string searches, string replacement and query replace.

AEDIT removes the restriction of only inserting characters when adding or modifying text. When adding text with AEDIT you may choose to either insert characters at the current cursor location, or over-write the existing text as you type. This flexibility simplifies the creation and editing of tables and charts.

USER INTERFACE

The menu-driven interface AEDIT provides makes it unnecessary to memorize long lists of commands and their syntax. Instead, a complete list of the commands or options available at any point is always displayed at the bottom of the screen. This makes AEDIT both easy to learn and easy to use.

FULL FLEXIBILITY

In addition to the standard PC terminal support provided with AEDIT, you are able to

configure AEDIT to work with almost any terminal. This along with user-definable macros and full adjustable tabs, margins, and case sensitivity combine to make AEDIT one of the most flexible editors available today.

MACRO SUPPORT

AEDIT will create macros by simply keeping track of the command and text that you type, "learning" the function the macro is to perform. The editor remembers your actions for later execution, or you may store them in a file to use in a later editing session.

Alternatively, you can design a macro using AEDIT's powerful macro language. Included with the editor is an extensive library of useful macros which you may use or modify to meet your individual editing needs.

TEXT PROCESSING

For your documentation needs, paragraph filling or justification simplifies the chore of document formatting. Automatic carriage return insertion means you can focus on the content of what you are typing instead of how close you are to the edge of the screen.

SERVICE, SUPPORT, AND TRAINING

Intel augments its development tools with a full array of seminars, classes, and workshops; on-site consulting services; field application engineering expertise; telephone hot-line support; and software and hardware maintenance contracts. This full line of services will ensure your design success.

SPECIFICATIONS

HOST SYSTEM

AEDIT for PC-DOS has been designed to run on the IBM* PC XT, IBM PC AT, and compatibles. It has been tested and evaluated for the PC-DOS 3.0 or greater operating system.

Versions of AEDIT are available for the iRMX™-86 and iRMX II Operating System.

ORDERING INFORMATION

D86EDINL AEDIT Source Code Editor Release 2.2 for PC-DOS with supporting documentation

122716	AEDIT-DOS Users Guide
122721	AEDIT-DOS Pocket Reference
RMX864WSU	AEDIT for iRMX I Operating System
R286EDI286EU	AEDIT for iRMX II/III Operating System

iPAT™ PERFORMANCE ANALYSIS TOOL



REAL-TIME SOFTWARE ANALYSIS FOR THE 8086/88, 80186/188, 80286, and 80386

280786-1

Intel's iPAT™ Performance Analysis Tool enables OEMs developing applications based on the 8086/88, 80186/188, 80286, or 80386 microprocessors to analyze real-time software execution in their prototype systems at speeds up to 20 MHz. Through such analysis, it is possible to speed-tune applications with real-time data, optimize use of operating systems (such as Intel's iRMX® II Real-Time Multitasking Executive for the 80286 and 80386, and iRMK™ Real-Time Multitasking Kernel for the 80386), characterize response characteristics, and determine code execution coverage by real-time test suites. Analysis is performed symbolically, non-intrusively, and in real-time with 100% sampling in the microprocessor prototype environment. iPAT supports analysis of OEM-developed software built using 8086, 80286, and 80386 assemblers and compilers supplied by Intel and other vendors.

All iPAT Performance Analysis Tool products are serially linked to DOS computer systems (such as IBM® PC AT, PC XT, and PS/2® Model 80) to host iPAT control and graphic display software. Several means of access to the user's prototype microprocessor system are supported. For the 80286 (real and protected mode), a 12.5 MHz iPAT-286 probe can be used with the iPATCORE system. For the 8086/88 (MAX MODE designs only), a 10 MHz iPAT-88 probe can be used with the iPATCORE system. iPATCORE systems also can be connected to sockets provided on the ICE™-286 and ICE-186 in-circuit emulators, or interfaced to I2ICE™ in-circuit emulators with probes supporting the 8086/88, 80186/188, or 80286. The 20 MHz iPAT-386 probe, also supported by the common iPATCORE system, can be operated either in "piggyback" fashion connected to an Intel ICE in-circuit emulator for the Intel386, or directly connected to a prototype system independent of an ICE. iPAT-386 supports all models of 80386 applications anywhere in the lowest 16 Megabytes of the 80386 linear address space.

FEATURES

iPAT™ FEATURES

- Up to 20 MHz real-time analysis
- Histograms and analysis tables
- Performance profiles of up to 125 partitions
- Code execution coverage over up to 252K
- Hardware or software interrupt analysis
- Simple use with function keys and graphics
- Use with or without Intel ICEs

MOST COMPLETE REAL-TIME ANALYSIS AVAILABLE TODAY

iPAT Performance Analysis Tools use in-circuit probes containing proprietary chip technology to achieve full sampling in real-time non-intrusively.

MEETS THE REAL-TIME DESIGNER'S NEEDS

The iPAT products include support for interactions between real-time software and hardware interrupts, real-time operating systems, "idle time," and full analysis of real-time process control systems.

SPEED-TUNING YOUR SOFTWARE

By examining iPAT histogram and tabular information about procedure usage (including or not including their interaction with other procedures, hardware, operating systems, or interrupt service routines) for critical functions, the software engineer can quickly pinpoint trouble spots. Armed with this information, bottlenecks can be eliminated by means such as changes to algorithms, recoding in assembler, or adjusting system interrupt priorities. Finally, iPAT can be used to prove the acceptability of the developer's results.

EFFICIENCY AND EFFECTIVENESS IN TESTING

With iPAT code execution coverage information, product evaluation with test suites can be performed more effectively and in less time. The evaluation team can quickly pinpoint areas of code that are executed or not executed under real-time conditions. By this means, the evaluation team can substantially remove the "black box" aspect of testing and assure 100% hits on the software under test. Coverage information can be used to document testing at the module, procedure, and line

level. iPAT utilities also support generation of instruction-level code coverage information.

ANALYSIS WITH OR WITHOUT SYMBOLICS

If your application is developed with "debug" symbolics generated by Intel 8086, 80286, or 80386 assemblers and compilers, iPAT can use them—automatically. Symbolic names also can be defined within the iPAT environment, or conversion tools supplied with the iPAT products can be used to create symbolic information from virtually any vendor's map files for 8086, 80286, and 80386 software tools.

REAL OR PROTECTED MODE

iPAT supports 80286 and 80386 protected mode symbolic information generated by Intel 80286 and 80386 software tools. It can work with absolute addresses, as well as base-offset or selector-offset references to partitions in the prototype system's execution address space.

FROM ROM-LOADED TO OPERATING SYSTEM LOADED APPLICATIONS

The software analysis provided by iPAT watches absolute execution addresses in-circuit in real time, but also supports use of various iPAT utilities to determine the load locations for load-time located software, such as applications running under iRMXII, DOS, Microsoft Windows*, or MS*-OS/2.

USE STANDALONE OR WITH ICE

The iPAT-386, iPAT-286, and iPAT-86/88 probes, together with an iPATCORE system, provide standalone software analysis independent of an ICE (in-circuit emulator) system. The iPATCORE system and DOS-hosted software also can be used together with ICE-386, ICE-286, and I²ICE-86/88, 186/188, or 286 in-circuit emulators and DOS-hosted software. Under the latter scenario, the user can examine prototype software characteristics in real-time on one DOS host while another DOS host is used to supply input or test conditions to the prototype through an ICE.

FEATURES

UTILITIES FOR YOUR NEEDS

Various utilities supplied with iPAT products support generation of symbolic information from map files associated with 3rd-party software tools, extended analysis of iPAT code execution coverage analysis data, and convenience in the working environment. For example, symbolics can be generated for maps produced by most software tools, instruction-level code execution information can be produced, and iRMXII-format disks can be

read/written in DOS floppy drives to facilitate file transfer.

WORLDWIDE SERVICE AND SUPPORT

All iPAT Performance Analysis Tool products are supported by Intel's worldwide service and support. Total hardware and software support is available, including a hotline number when the need is there.

SPECIFICATIONS

HOST COMPUTER REQUIREMENTS

All iPAT Performance Analysis Tool products are hosted on IBM PC AT, PC XT, or PS/2 Model 80 personal computers, or 100% compatibles, and use a serial link for host-to-iPAT communications. At least a PC AT class system is recommended. The DOS host system must meet the following minimum requirements:

- 640K Bytes of Memory
- 360K Byte or 1.2M Byte floppy disk drive
- Fixed disk drive
- A serial port (COM1 or COM2) supporting 9600 baud data transfer
- DOS 3.0 or later
- IBM or 100% compatible BIOS

PHYSICAL DESCRIPTIONS

Unit	Width		Height		Length	
	In.	Cm.	In.	Cm.	In.	Cm.
iPATCORE	8.25	21.0	1.75	4.5	13.75	35.0
Power Supply	7.75	20.0	4.25	11.0	11.0	28.0
iPAT-386 probe	3.0	7.6	0.50	1.3	4.0	10.1
iPAT-286 probe	4.0	10.2	1.12	2.8	6.0	15.3
iPAT-86 probe	4.0	10.2	1.12	2.8	6.0	15.3
iPATCABLE (to ICE-186/286)	4.0	10.2	.25	.6	36.0	91.4
IIIPATB,C,D (I ² ICE board)	12.0	30.5	12.0	30.5	.5	1.3
Serial cables PC AT/XT PS/2					144.0	370.0

ELECTRICAL CONSIDERATIONS

The iPATCORE system power supply uses an AC power source at 100V, 120V, 220V, or 240V over 47Hz to 63Hz. 2 amps (AC) at 100V or 120V; 1 amp at 220V or 240V.

iPAT-386, iPAT-286 and iPAT-86/88 probes are externally powered, impose no power demands on the user's prototype, and can thus be used to analyze software activity through power down and power up of a prototype system. For ICE-386, ICE-286, ICE-186, and I²ICE microprocessor probes, see the appropriate in-circuit emulator factsheets.

ENVIRONMENTAL SPECIFICATIONS

Operating Temperature: 10°C to 40°C (50°F to 104°F) ambient

Operating Humidity: Maximum of 85% relative humidity, non-condensing

SPECIFICATIONS

CONFIGURATION GUIDE

For all of the following application requirements, the iPAT system is supported with iPAT 2.0 (or greater) or iPAT/I²C 1.2 (or greater) host software, as footnoted.

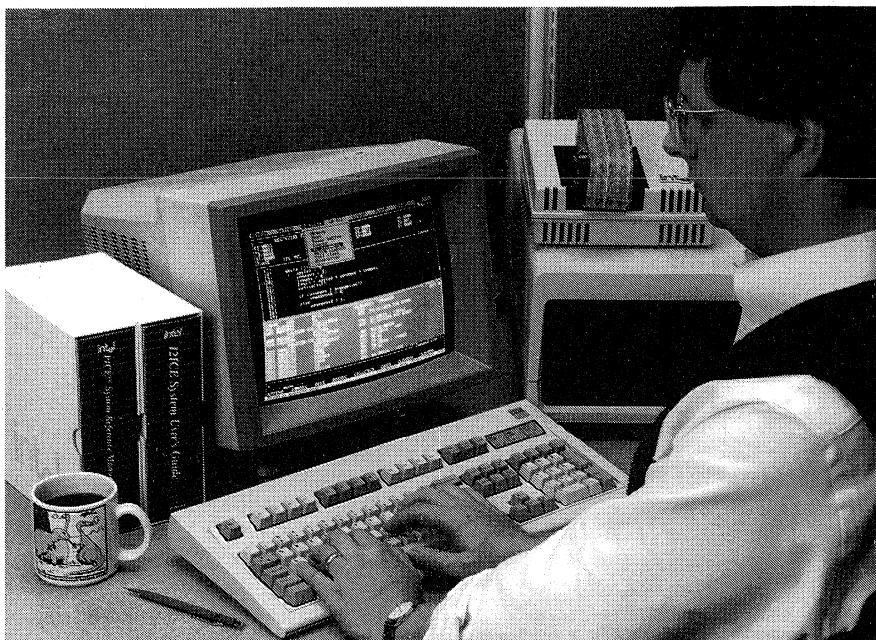
Application Software	Option	iPAT Order Codes	Host System
80386 Embedded	#1	iPAT386DOS ¹ , iPATCORE	DOS ⁴
iRMK on 80386	#1	iPAT386DOS, iPATCORE	DOS
iRMXII OS-Loaded or Embedded on 386	#1	iPAT386DOS, iPATCORE	DOS
OS/2-Loaded on 386	#1	iPAT386DOS, iPATCORE	DOS
iRMXII OS-Loaded or Embedded	#1	iPAT286DOS, iPATCORE	DOS
80286 Embedded	#1	iPAT286DOS, iPATCORE	DOS
	#2	ICEPATKIT ²	DOS
	#3	I ² CIPATKIT ³	DOS
	#4	IIIPATD, iPATCORE ³	DOS ⁴
DOS OS-Loaded 80286	#1	iPAT286DOS, iPATCORE	DOS
OS/2 OS-Loaded 80286	#1	iPAT286DOS, iPATCORE	DOS
80186/188 Embedded	#1	ICEPATKIT ²	DOS
	#2	I ² CIPATKIT ³	DOS
	#3	IIIPATD, iPATCORE ³	DOS ⁴
DOS OS-Loaded 8086/88	#1	iPAT88DOS, iPATCORE	DOS
8086/88 Embedded	#1	iPAT88DOS, iPATCORE	DOS
	#2	I ² CIPATKIT ³	DOS
	#3	IIIPATD, iPATCORE ³	DOS ⁴

Notes:

1. Operable standalone or with ICE-386 (separate product; separate host). iPAT-386 probe connects directly to prototype system socket, or to optional 4 probe-to-socket hinge cable (order code TA386A), or to ICE-386 probe socket.
2. Requires ICE-186 or ICE-286 in-circuit emulator system.
3. Requires I²C in-circuit emulator system.
4. Includes iPAT/I²C integrated software (iPAT/ICE 1.2 or greater), which only supports sequential iPAT and ICE operation on one host, rather than in parallel on two hosts (iPAT 2.0 or greater).



I²ICE™ IN-CIRCUIT EMULATION SYSTEM



280800-1

IN-CIRCUIT EMULATOR FOR THE 8086/80186/80286 FAMILY OF MICROPROCESSORS

The I²ICE™ In-circuit Emulator is a high-performance, cost-effective debug environment for developing systems with the Intel 8086/80186/80286 family of microprocessors. With 10 MHz emulation, a window-oriented user interface, and compatibility with Intel's iPAT™ Performance Analysis Tool, the I²ICE Emulator gives you unmatched speed and control over all phases of hardware/software debug.

FEATURES

- Emulation speeds up to 10 MHz with 8086/88, 80186/188 and 80286 microprocessors
- 8087 and 80287 numeric coprocessor support
- Hosted on IBM PC AT* or AT BIOS compatibles
- ICEVIEW™ window-oriented user interface with pull-down menus and context-sensitive help
- Source and symbol display using all Intel languages
- 1K frame bus and execution trace buffer
- Symbolic debugging for flexible access to memory location and program variables
- Flexible breakpointing for quick problem isolation
- Memory expandable to 288K with zero wait states
- Worldwide service and support
- iPAT option for software speed tuning

I²ICE, ICEVIEW, and iPAT are trademarks of Intel Corporation.

*IBM is a trademark of International Business Machines Corporation.

FEATURES

ONE TOOL FOR THE ENTIRE DEVELOPMENT PROCESS

The I²ICE Emulator allows hardware and software design to proceed simultaneously, so you can develop software even before prototype hardware is available. With 32K of zero wait-state mappable memory (and an additional 256K with optional memory boards), you can use the I²ICE Emulator to debug at any stage of the development cycle: hardware development, software development, system integration or system test.

HIGH-SPEED, REAL-TIME EMULATION

The I²ICE Emulator delivers full-speed, real-time emulation at speeds up to 10 MHz. Based on Intel's exclusive microprocessor technology, the I²ICE Emulator matches each chip's electrical and timing characteristics without memory or interrupt intrusions, ensuring design accuracy and eliminating surprises. The performance of your prototype is the performance you can expect from your final product.

EASY-TO-USE ICEVIEW™ INTERFACE

The ICEVIEW interface makes the I²ICE Emulator easy to learn and use by providing easy access to application information and ICE functions. Pull-down menus and windows boost productivity for both new and experienced users. Multiple on-screen windows allow you to access the source display, execution trace, register, and other important information, all at the same time. You can watch the information change as you modify and step through your program. You can even customize window size and screen positions. A command line interface is also available with syntax checking and context-sensitive prompts. ICEVIEW works with monochrome, CGA and the latest EGA color displays.

SYMBOLIC DEBUG SPEEDS DEVELOPMENT

The extensive debug symbolics generated by the Intel 8086 and 80286 assemblers and compilers can increase your development productivity. Symbolics with automatic formatting are available for all primitive types, regardless of whether the variables are globals, locals (stack-resident) or pointers. The virtual symbol table supports all symbolics, even in very large programs. Aliasing can be used to reduce keystrokes and save time.

POWERFUL BREAK AND TRACE CAPABILITY FOR FAST PROBLEM ISOLATION

The I²ICE Emulator allows up to eight simultaneous break/trace conditions to be set (four execution, four bus), a timesaver when solving hardware/software integration problems. Break and trace points can be set on specified line numbers, on procedures, or on symbolic data events, such as writing a variable to a value or range of values. You can break or trace on specific hardware events, such as a read or write to a specific address, data or I/O port, or on a combination of events.

MULTIPROCESSOR, PROTECTED MODE, AND COPROCESSOR SUPPORT

Up to four I²ICE systems can be linked and controlled simultaneously from one PC host, enabling you to debug multiprocessor systems. The I²ICE Emulator with an 80286 probe supports all 80286 protected mode capabilities. It also supports the 8087 and 80287 numeric coprocessors.

FEATURES

iPAT™ FOR SOFTWARE PERFORMANCE AND CODE COVERAGE ANALYSIS

The I²ICE Emulator interfaces to Intel's iPAT Performance Analysis Tool for examining software execution speeds and code coverage in real time. iPAT displays critical performance data about your code in easy-to-understand histograms and tables. Elusive bottlenecks are readily seen, allowing you to focus your attention to get the most performance out of your product.

iPAT also performs code execution coverage, letting you perform product evaluations faster and more effectively. iPAT pinpoints areas in your code either executed or not executed according to specific conditions, taking the guesswork out of software evaluations.

EASY INTERFACE TO EXTERNAL INSTRUMENTS

The I²ICE system includes external emulation clips and software support for setting breakpoints, tracepoints and arm/disarm conditions on external events, making it easy to connect external logic analyzers and signal generators. You can debug complex hardware/software interactions with a high level of productivity.

WORLDWIDE SERVICE, SUPPORT, AND TRAINING

To augment its development tools, Intel offers a full array of seminars, classes, and workshops, field application engineering expertise, hotline technical support and on-site service.

Intel also offers a Software Support package which includes technical software information, telephone support, automatic distribution of software and documentation updates, access to the "ToolTalk" electronic bulletin board, *iComments* publication, remote diagnostic software, and a development tools troubleshooting guide.

Intel's Hardware Support package includes technical hardware information, telephone support, warranty on parts, labor, material, and on-site hardware support.

SPECIFICATIONS

Host Requirements

IBM PC/AT or 100% PC AT BIOS compatible
DOS 3.XX
640 Kbytes of memory
360 Kbytes or 1.2 MB floppy disk drive
Hard disk drive
Monochrome, CGA or EGA monitor (EGA recommended)

Physical Description

Unit	Width		Height		Length	
	cm	in	cm	in	cm	in
I ² ICE chassis	43.2	17.0	21.0	8.25	61.3	24.13
Probe base	21.6	8.5	7.6	3.0	25.4	10.0

Host/chassis cable 15 ft. (4.6 m)

Electrical Characteristics

90-132 V or 180-264 V (selectable)
47-63 Hz
12 amps (AC)

Environmental Specifications

Operating temperature: 0-40°C (32-104°F) ambient
Operating humidity: Maximum of 85% relative humidity, non-condensing

FEATURES

ORDERING INFORMATION

Kit Code	Contents
-----------------	-----------------

pIII010KITD	I ² ICE system 10 MHz 8086/8088 support kit for IBM PC host. Includes probe, chassis, and host interface module and software.
pIII111KITD	I ² ICE system 10 MHz 80186 support kit for IBM PC host. Includes probe, chassis, host interface module and software. Note: For 80188 support, the III198 option below must also be ordered.

III198

8 MHz 80188 support conversion kit to convert 80186 probe to 80188 probe.

pIII212KITD

I²ICE system, 10 MHz 80286 support kit for IBM PC AT host. Includes probe, chassis, host interface module and software.

954D

I²ICE PC AT host software. Includes ICEVIEW™ windowed human interface.

Note: I²ICE probes, chassis, software, cables and iPAT options are available separately.

INTEL ICETM-186 AND ICE-188 IN-CIRCUIT EMULATORS



280726-1

HIGH PERFORMANCE REAL-TIME EMULATION

The Intel ICETM-186 and ICETM-188 emulators deliver reliable 16 MHz real-time emulation for the 80C186 and 80C188 microprocessors. The 80C186 and 80C188 embedded processors are based on the new 80C186 modular CPU core with fully-static design. The in-circuit emulators are a versatile and efficient tool for developing, debugging and testing products designed with Intel microprocessors. The ICE-186/188 has many productivity boosting features to help you get your products to market as quickly as possible. The contemporary windowed human interface, symbolic source-level debug, 3K frames of dynamic trace display and the powerful conditional break capabilities are standard on the ICE-186/188 emulator. Intel, the inventor of the 80C186 and 80C188 microprocessors, has the most complete line of development tools from a single vendor to meet all of your development needs. Our language development tools portfolio includes the assembler, software debugger DB-86, C, PL/M, PASCAL and FORTRAN, all designed to work with the ICE-186/188 emulator to meet your embedded design needs.

FEATURES

ICETM-186/188 FEATURES

- Reliable Full Speed Emulation up to 16 MHz
- 3K Frames Dynamic Trace Buffer can be displayed without stopping emulation
- Optional 128K, 512K or 1 Mbyte of Zero Wait-State Mapped Memory
- 80C187 Numeric Coprocessor Support
- High Speed RS-232-C and GPIB Communication Link
- Crystal Power Accessory for Software Development
- Interface for Intel Performance Analysis Tool (iPAT™) and Logic Analyzer
- Emulation and ONCE™ Mode Support for QFP and PLCC Packages
- Windowed Human Interface with Mouse Support
- Source Level Debug with effective Source Window and Watch Window Operations
- Powerful Go Command with two-level breakpoints, event counters and single stepping capability
- High-Level Language Symbolic Debug
- Tutorial Software to Speed Up Learning Curve
- Complete Intel Service and Support

SOFTWARE PERFORMANCE ANALYSIS

Intel's Performance Analysis Tool (iPAT) is designed to increase the team productivity with features such as interrupt latency measurement, code coverage analysis and software module timing analysis. The emulator provides an external connector for the iPAT tool. These features enable the user to design reliable, high performance embedded control products.

BUILT-IN SUPPORT FOR LOGIC ANALYZERS

General purpose logic analyzers can be used with the ICE-186/188 emulator to provide detailed timing of specific events. The emulator has an external sync signal to trigger the logic analyzer, making complex event triggering easy. An additional 60 pin connector is included to support the logic analyzer.

SOFTWARE CARROUSEL

To address the need of switching among different development tools, such as editing, recompilation and linking with minimum interruptions, the Software Carrousel package is included. It provides an environment which allows up to eleven other programs to be accessible while running the emulator software. By pressing a pair of keys, the current program environment is preserved and a different application environment is ready.

WORLDWIDE SERVICE, SUPPORT, AND TRAINING

To augment its development tools, Intel offers a full array of seminars, classes, and works, field application engineering expertise, hotline technical support and on-site service.

Intel also offers a Software Support package which includes technical software information, telephone support, automatic distribution of software and documentation updates, access to the "ToolTalk" electronic bulletin board, *iComments* publication, remote diagnostic software, and a development tools troubleshooting guide.

Intel's Hardware Support package includes technical hardware information, telephone support, warranty on parts, labor, material, and on-site support.

SPECIFICATIONS

PERSONAL COMPUTER REQUIREMENTS

The ICE-186/188 emulator is hosted on an IBM PC AT platform or IBM PS/2. The emulator has been tested and evaluated on an IBM PC AT. The PC AT must meet the following minimum requirements:

- 640 KBytes of Memory
- An additional 1 MByte of expanded memory, recommending an Above Board managed by emm.sys driver. Other memory managers conforming to the Lotus/Intel/Microsoft Expanded Memory Specification Version 3.2 or later are available.
- One 20 MByte Hard Disk
- PC DOS 3.2 or Later
- A serial Port (COM1 or COM2) supporting minimally at 9600 Baud Data Transfers, or a National Instruments GPIB-PC2A board
- Math coprocessor

ORDERING INFORMATION

ICE186LP ICE-186 16 MHz system, includes control unit, PLCC probe, power supply, emulator s/w, Software Carousel, CPA adaptor and ONCE adaptor

ICE186LQ ICE-186 16MHz system, includes control unit, QFP probe, power supply, emulator s/w, Software Carousel, CPA adaptor

ICE188LP ICE-188 16 MHz System, includes control unit, PLCC probe, power supply, emulator s/w, Software Carousel, CPA adaptor and ONCE adaptor

ICE188LQ ICE-188 16MHz system, includes control unit, QFP probe, power supply, emulator s/w, Software Carousel, CPA adaptor

The above order codes must order a memory option

MEM128 128 Kbytes map memory

MEM512 512 Kbytes map memory

MEM1MB 1 Mbytes map memory

UP186LP ICE-186 PLCC Probe

UP186LQ ICE-186 QFP Probe

UP188LP ICE-188 PLCC Probe

UP188LQ ICE-188 QFP Probe

HC18XLP PLCC hinge cable

HC18XLQ QFP hinge cable

D86ASM86KIT Assembler, Link-86, Loc-86, Lib 86, cross reference utility, OH-86, numerics support, software debugger, hosted on PC DOS 3.0 or higher

D86C86NL C-86 compiler for PC DOS 3.0 or higher

D86PLM86NL PL/M-86 compiler for PC DOS 3.0 or higher

D86PAS86NL PASCAL-86 compiler for PC DOS 3.0 or higher

D86FOR86NL FORTRAN-86 compiler for PC DOS 3.0 or higher

***D86ASM86NL** ASM-86 Macro Assembler



INTEL ICETTM-186EB AND ICE-188EB IN-CIRCUIT EMULATORS

HIGH PERFORMANCE REAL-TIME EMULATION

The Intel ICETTM-186EB and ICETTM-188EB emulators deliver reliable 16 MHz real-time emulation for the 80C186EB and 80C188EB microprocessors. The in-circuit emulators are a versatile and efficient tool for developing, debugging and testing products designed with Intel microprocessors. The ICE-186EB/188EB has many productivity boosting features to help you get your products to market as quickly as possible. The contemporary windowed human interface, symbolic source-level debug, 3K frames of dynamic trace display and the powerful conditional break capabilities are standard on the ICE-186EB/188EB emulator. Intel, the inventor of the 80C186EB and 80C188EB microprocessors, has the most complete line of development tools from a single vendor to meet all of your development needs. Our language development tools portfolio includes the assembler, software debugger DB-86, C, PL/M, PASCAL and FORTRAN, all designed to work with the ICE-186EB/188EB emulator to meet your embedded design needs.

ICETTM-186EB/188EB FEATURES

- Reliable Full Speed Emulation up to 16 MHz
- 3K Frames Dynamic Trace Buffer can be displayed without stopping emulation
- Optional 128K, 512K or 1 Mbytes of Zero Wait-State Mapped Memory
- 80C187 Numeric Coprocessor Support
- High Speed RS-232-C and GPIB Communication Link
- Crystal Power Accessory for Software Development
- Interface for Intel Performance Analysis Tool (iPATTM) and Logic Analyzer
- Emulation and ONCETM Mode Support for QFP and PLCC Packages
- Windowed Human Interface with Mouse Support
- Source Level Debug with effective Source Window and Watch Window Operations
- Powerful Go Command with two-level breakpoints, event counters and single stepping capability
- High-Level Language Symbolic Debug
- Tutorial Software to Speed Up Learning Curve
- Complete Intel Service and Support

FEATURES

RELIABLE HIGH SPEED EMULATION

The ICE-186EB/188EB emulator uses superior Intel component bondout and advanced cable technology to ensure accuracy of emulation. Accurate timing is a critical factor in today's 16 MHz designs.

The emulator is designed to support low power application needs. The probe supports true CMOS voltage inputs/outputs and has low power current draw. There is no power consumption difference between using the emulator or the actual component. The ICE-186EB/188EB also supports the debugging of target systems with 80C187 numeric coprocessor.

WINDOWED HUMAN INTERFACE

The windowed interface increases productivity for both expert and casual users. Multiple windows can simultaneously provide source code, watch variables, memory, registers and trace information. Breakpoints can be set directly on the source code. The source window is automatically updated when the emulator completes a breakpoint or a single stepping operation. The user can program the watch windows to track up to ten program variables. The data displayed in the windows is automatically updated while executing and debugging code. Pull down menus offer a set of common emulator commands, making it easier to configure the debug environment.

PRODUCTIVE EVENT MONITOR CAPABILITY

With higher integration of the GO command, event recognition, and emulator action, complex logic debug is easy and efficient. Breakpoint events can be detected on execution addresses and/or bus addresses and/or bus access types such as memory or I/O reads or writes. Address and data specification can be based on single value, range and don't cares. A flexible single step command allows the user to execute one machine-level or one high-level instruction at a time.

All the emulator commands can be programmed with a set of macro commands and can be included in a procedure for repeated debug sessions. The combined use of the event monitor and the macro programming feature can speed up the application debug.

This is also a useful tool for automated manufacturing testing and debug.

PROBLEM TRACKING THROUGH DYNAMIC TRACE BUFFER

The emulator can display up to 3,072 frames of processor activity, including both the execution and data bus activity. The information collected in real time provides valuable reference for tracking down problems. Users can view the trace buffer or modify the trace conditions at any time without stopping the emulation. The trace buffer can be displayed in either instruction or the cycle mode.

SOFTWARE DEBUG WITHOUT A PROTOTYPE

Software developers often are required to debug application code before the functional prototypes are available. The Crystal Power Accessory (CPA) with the emulator map memory provides a complete target environment for debugging software modules that do not require timer and hardware interrupt support. The CPA is also used to run the emulator confidence tests.

The ICE-186EB/188EB provides up to 1 MB of zero wait-state memory. This memory can be used instead of target memory for code debugging. It is addressable in 32 Kbyte increments. The emulator can also be used to support simulated I/O ports, addressable in 4 Kbyte increments.

INTEGRATED HIGH LEVEL LANGUAGE SUPPORT

The ICE-186EB/188EB emulator is designed to support all INTEL software products. Intel's comprehensive language development tools include assembler, ANSI C, PL/M, PASCAL and FORTRAN compilers. These software tools were designed to take advantage of the component architecture and specifically support embedded applications. Symbols from the source code can be directly used in your ICE debug sessions. The ICE source window can display the source code in the original language used to produce the object code. This integrated emulator and language development environment allows designers to focus on the development process.

FEATURES

FAST BREAK SUPPORT

The fastbreak feature is used when there is a need to interrogate the current execution status without halting the emulator for extended periods of time. The fastbreak has a maximum break period of 5625 clock cycles. The requested information can be stored in the trace buffer for later reference.

PRESERVE DRAM CONTENTS

The ICE-186EB/188EB continues DRAM refresh signals even when the emulator has been halted, thus ensuring DRAM memory is not lost or corrupted. During the interrogation mode the emulator will keep the timer functioning and will correctly respond to interrupts in real-time.

MULTIPLE HIGH SPEED COMMUNICATION LINKS

Two communication links are available for use in conjunction with the IBM PC AT host. The emulator can use an RS-232-C serial link with a transfer rate up to 57.6 Kbyte per second. A user supplied National Instruments (IEEE-488) GPIB communication board will provide parallel transfers at rates up to 300 Kbytes per second.

SOFTWARE PERFORMANCE ANALYSIS

Intel's Performance Analysis Tool (iPAT) is designed to increase the team productivity with features such as interrupt latency measurement, code coverage analysis and software module timing analysis. The emulator provides an external connector for

the iPAT tool. These features enable the user to design reliable, high performance embedded control products.

BUILT-IN SUPPORT FOR LOGIC ANALYZERS

General purpose logic analyzers can be used with the ICE-186EB/188EB emulator to provide detailed timing of specific events. The emulator has an external sync signal to trigger the logic analyzer, making complex event triggering easy. An additional 60 pin connector is included to support the logic analyzer.

SOFTWARE CAROUSEL

To address the need of switching among different development tools, such as editing, recompilation and linking with minimum interruptions, the Software Carousel package is included. It provides an environment which allows up to eleven other programs to be accessible while running the emulator software. By pressing a pair of keys, the current program environment is preserved and a different application environment is ready.

WORLDWIDE SERVICE, SUPPORT, AND TRAINING

To augment its development tools, Intel offers a full array of seminars, classes, and workshops. In addition, on-site consulting services, field application engineering expertise, telephone hotline support and software and hardware maintenance contracts are available to help assure your design success, as well as electronic bulletin boards and other services.

SPECIFICATIONS

PERSONAL COMPUTER REQUIREMENTS

The ICE-186EB/188EB emulator is hosted on an IBM PC AT platform or IBM PS/2. The emulator has been tested and evaluated on an IBM PC AT. The PC AT must meet the following minimum requirements:

- 640 KBytes of Memory

- An additional 1 MBytes of expanded memory, recommending an Above Board managed by emm.sys driver. Other memory managers conforming to the Lotus/Intel/Microsoft Expanded Memory Specification Version 3.2 or later are available.
- One 20 MBytes Hard Disk
- PC DOS 3.2 or Later
- A serial Port (COM1 or COM2) supporting minimally at 9600 Baud Data Transfers, or a National Instruments GPIB-PC2A board
- Math coprocessor

SPECIFICATIONS

PHYSICAL DESCRIPTION AND CHARACTERISTICS

The ICE-186EB/188EB Emulator consists of the following components:

Table 1 Emulator's Physical Characteristics

Unit	Width		Height		Length	
	In.	Cm	In.	Cm	In.	Cm
Emulator Control Unit	10.4	26.4	1.7	4.3	20.7	52.6
Power Supply	7.7	19.6	4.1	10.4	11.0	27.9
User Probe	3.7	9.4	.65	1.6	7.0	17.8
User Probe Adapter Cable					3.4	8.6
Crystal Power Accessory	4.30	10.9	.60	1.5	6.7	17.0
Serial Cable					144.0	366.0

ICETM-186EB/188EB AC SPECIFICATIONS

Table 2 ICETM-186EB/188EB AC Specifications

Timing Requirements				
Symbol	Parameter	Component		ICE
		Min	Max	Typical
T _{DVCL}	Data in Setup (A/D)	10nS		19nS
T _{CLDX}	Data in Hold (A/D)	0		-2
T _{SRYCL}	Synchronous Ready (READY) Transition Setup Time	10		10
T _{CLSRV}	READY Transition Hold Time	0		0
T _{HVCL}	HOLD Setup	10		24
T _{CLHV}	HOLD Hold	0		0
T _{INVCH}	NMI TEST #	10		15
	INTR, TIMERIN Setup Time	10		10
T _{CHINV}	NMI TEST #	0		0
	INTR, TIMERIN Hold Time	0		0

SPECIFICATIONS

ICE™-186EB/188EB AC SPECIFICATIONS

Table 2 (Continued)

Output Delays					
Symbol	Parameter	CPU		Emulator	
		Min	Max	Min	Max
TCHOV1	ALE, S2:0#, DEN#, DT/R#	3nS	20nS	3nS	20nS
TCHOV2	GCS7:0#, LCS#, UCS#, NCS#	3	25	3	25
TCLOV1	BHE#, DEN#, LOCK#, RESOUT, HLDA T0OUT, T1OUT A19:16	3	20	3	20
TCLOV2	RD#, WR#, GCS7:0#, LCS#, UCS#, NCS#, INTA1:0#	3	25	3	25
TCLADV	AD15:0	3	25	6	34
TCHOF	RD#, WR#, BHE#, DT/R#, LOCK#, S2:0#, A19:16	0	25	0	25
TCLOF	DEN#	0	25	0	25
TCLADF	AD15:0	0	25	3	34
Clkin Requirements					
Symbol	Parameter	CPU		Emulator	
		Min	Max	Min	Max
T _C	CLKIN Period	31.25nS	∞	31.25nS	∞ *
T _{CKHL}	CLKIN Fall Time	0	8	0	8
T _{CKLH}	CLKIN Rise Time	0	8	0	8
T _{CLCK}	CLKIN Low Time	20	∞	20	∞ *
T _{CHCK}	CLKIN High Time	20	∞	20	∞ *
Clkout Timing					
TCICO	CLKIN to CLKOUT Skew	0nS	15nS	0nS	15nS
TCLCL	CLKOUT Period	$2 \cdot T_C$		$2 \cdot T_C$	
TCLCH	CLKOUT Low Time	$(T/2) - 5$	$(T/2) + 5$	$(T/2) - 5$	$(T/2) + 5$
TCHCL	CLKOUT High Time	$(T/2) - 5$	$(T/2) + 5$	$(T/2) - 5$	$(T/2) + 5$
TCH1CH2	CLKOUT Rise Time	1	6	1	6
TCL2CL1	CLKOUT Fall Time	1	6	1	6

*Emulator will acknowledge condition with a "clock lost" error message

SPECIFICATIONS

ICETM-186EB/188EB DC SPECIFICATIONS

Table 3 lists the DC input specifications at the ICE-186EB user probe.

Table 3 ICETM-186EB/188EB DC Input Specifications

Parameter	Signals	CPU		Emulator	
		Max	Min	Max	Min
V _{IL}	All (except CLKIN, HOLD, RESIN# and NMI)	.3*V _{CC}	-.5	.3*V _{CC}	-.5
V _{IL}	HOLD, RESIN# and NMI	.3*V _{CC}	-.5	.8	-.5
V _{IL}	CLKIN	.2*V _{CC}	-.3	.2*V _{CC}	-.3
V _{IH}	All (except CLKIN, HOLD, RESIN# and NMI)	V _{CC} + .5	.7*V _{CC}	V _{CC} + .5	.7*V _{CC}
V _{IH}	HOLD, RESIN# and NMI	V _{CC} + .5	.7*V _{CC}	7.0	2.0
V _{IH}	CLKIN	V _{CC} + .3	.8*V _{CC}	V _{CC} + .3	.8*V _{CC}
I _{IL}	All (except HOLD, RESIN#, NMI, A19:16 and LOCK#)	±15uA		±15uA	
I _{IL}	HOLD, RESIN#, NMI	±15uA		-.6mA	
I _{IL}	A19:16 and LOCK#	-2mA		-2mA	

Table 4 lists the DC output specifications at the ICE-186EB user probe.

Table 4 ICETM-186EB/188EB DC Output Specifications

Parameter	Signals	CPU		Emulator	
		Max	Min	Max	Min
V _{OH}	All (except AD15:0)		V _{CC} - .5		V _{CC} - .5
V _{OH}	AD15:0		V _{CC} - .5		3.94
V _{OL}	All (except AD15:0)	.45		.45	
V _{OL}	AD15:0	.45		.44	
I _{OH}	All (except AD15:0)	-2mA		-2mA	
I _{OH}	AD15:0	-2mA		-24mA	
I _{OL}	All (except AD15:0)	5mA		5mA	
I _{OL}	AD15:0	5mA		24mA	

SPECIFICATIONS

Table 5 lists the DC specifications for the ISYNCH and OSYNCH lines.

Table 5 DC Specifications for ISYNCH and OSYNCH

Symbol	Parameter	Min	Max	Notes
ISYNCH I_{IL}	Input Low current		2.6mA	$V_i = 0.5V$
ISYNCH I_{IH}	Input High current		-100uA	$V_i = 2.7V$
OSYNCH V_{OL}	Output Low voltage		0.5V	$I_{oi} = 15mA$
OSYNCH V_{OH}	Output High voltage	2.7V	$I_{oh} = -0.25mA$	

Table D-6 lists the DC power requirements at the ICE-186EB user probe.

Table 6 Probe Power Specifications

Symbol	Parameter	Min	Max
V_{CC}	Supply Voltage	4.75	5.25
I_{CC}	Supply Current		90mA

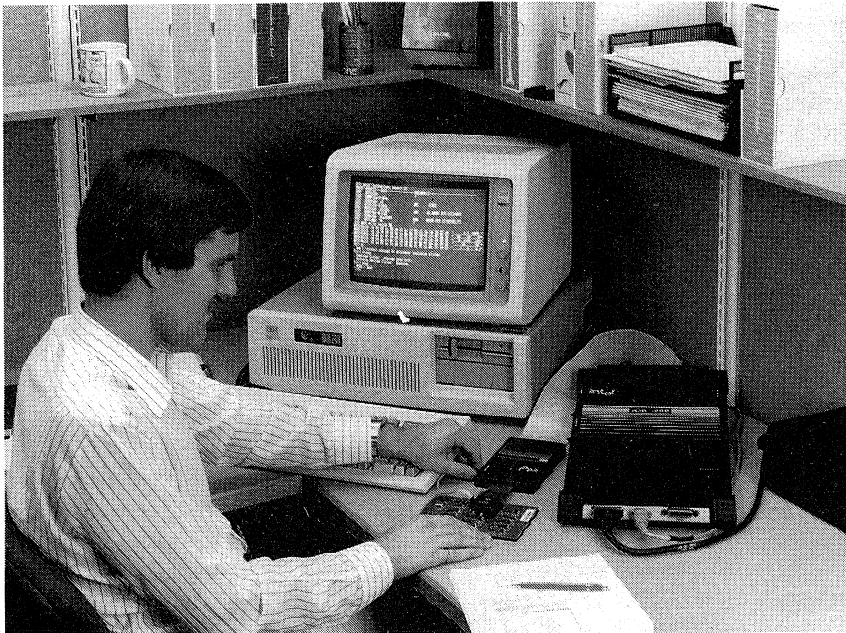
ORDERING INFORMATION

ICE186EBP	ICE-186EB 16 MHz system, includes control unit, PLCC probe, power supply, emulator s/w, Software Carousel, CPA adaptor and ONCE adaptor	MEM128	128 Kbytes map memory
ICE186EBQ	ICE-186EB 16 MHz system, includes control unit, QFP probe, power supply, emulator s/w, Software Carousel, CPA adaptor	MEM512	512 Kbytes map memory
ICE188EBP	ICE-188EB 16 MHz System, includes control unit, PLCC probe, power supply, emulator s/w, Software Carousel, CPA adaptor and ONCE adaptor	MEM1MB	1 Mbytes map memory
ICE188EBQ	ICE-188EB 16 MHz system, includes control unit, QFP probe, power supply, emulator s/w, Software Carousel, CPA adaptor	D86ASM86KIT	Assembler, Link-86, Loc-86, Lib 86, cross reference utility, OH-86, numerics support, software debugger, hosted on PC DOS 3.0 or higher
		D86C86NL	C-86 compiler for PC DOS 3.0 or higher
		D86PLM86NL	PL/M-86 compiler for PC DOS 3.0 or higher
		D86PAS86NL	PASCAL-86 compiler for PC DOS 3.0 or higher
		D86FOR86NL	FORTTRAN-86 compiler for PC DOS 3.0 or higher
		*D86ASM86NL	ASM-86 Macro Assembler

*European customers only

The above order codes must order a memory option

ICETM-286 IN-CIRCUIT EMULATOR



280728-1

HIGH PERFORMANCE REAL-TIME EMULATION

Intel's ICETM-286 emulator delivers real-time emulation for the 80286 microprocessor at speeds up to 12.5 MHz. The in-circuit emulator is a versatile and efficient tool for developing, debugging and testing products designed with the Intel 80286 microprocessor. The ICE-286 emulator provides real time, full speed emulation in a users system. Popular features such as symbolic debug, 2 Kbytes trace memory, and single-step program execution are standard on the ICE-286 emulator. Intel provides a complete development environment using assembler (ASM-286) as well as high-level languages such as Intel's iC286, PL/M-286 or Fortran 286 to accelerate development schedules.

Intel's ICE-286 emulator is hosted on IBM's Personal Computer AT, already available as a standard development solution in most of today's engineering environments. The ICE-286 emulator operates in prototype or standalone mode allowing software development and debug before a prototype system is available. The ICE-286 emulator is ideally suited for developing real time applications such as process control, machine control, communications, or other applications requiring the full power of the 12.5 MHz 80286 microprocessor.

ICE-286 FEATURES

- Full 12.5 MHz Emulation Speed
- 2 Kbytes Deep Trace Memory
- Two-Level Breakpoints with Occurrence Counters
- Single-Step Capability
- 128 Kbytes Zero Wait-State Mapped Memory
- Support For Protected and Real Modes
- High-Level Language Support
- Symbolic Debug

FEATURES

- Numeric Processor Extension Support
- RS-232-C and GPIB Communication Links
- Crystal Power Accessory
- Interface for Intel Performance Analysis Tool (iPAT™)
- Interface for Optional General Purpose Logic Analyzer
- Tutorial Software
- Complete Intel Service and Support

HIGHEST EMULATION SPEED AVAILABLE TODAY

The ICE-286 emulator supports development and debug of time-critical hardware and software using Intel's 12.5 MHz 80286 microprocessor.

RETRACE SOFTWARE TRACKS

This emulator captures up to 2048 frames of processor activity, including both execution and data bus activity. With this trace memory, large blocks of program code can be traced in real time and viewed for program flow and behavior characteristics.

HARDWARE BREAKPOINTS FOR COMPLEX DEBUG

User-defined "TIL-THEN" breakpoint statements stop emulation at specific execution addresses or bus events. During the hardware and software integration phase, breakpoint statements can be defined as execution addresses and/or bus addresses and/or bus access types, such as memory and I/O reads or writes. Additionally, event counters provide another level of breakpoint control for sophisticated state machine constructs used to specify emulation breakpoints/tracepoints.

SMALL OR LARGE STEPS

A stepping command can be used to view program execution one frame at a time or in preset frame blocks. When used in conjunction with symbolic debug, code execution can be monitored quickly and precisely.

DEBUG CODE WITHOUT A PROTOTYPE

Even before prototype hardware is available, the ICE-286 emulator working in conjunction with the Crystal Power Accessory (CPA) creates a "virtual" application environment. 128 Kbytes of zero wait-state memory is

available for mapped memory and I/O resource addressing in 4K increments. The CPA provides emulator diagnostics as well as the ability to use the emulator without a prototype.

PROTECTED AND REAL MODES

The ICE-286 emulator has full access to all protected-mode registers and permits modification of register contents. Protected mode of execution is beneficial for secure, multitasking applications.

HIGH-LEVEL LANGUAGE SUPPORT OPTIMIZED FOR INTEL TOOLS

The ICE-286 supports emulation for programs written in Intel's ASM 286 and ASM 86 or any of the Intel high-level languages:

PL/M-286/86	Fortran-286/86
Pascal-286/86	C-286/86

These languages are optimized for Intel component architectures to deliver a tightly integrated, high performance development environment.

USER-FRIENDLY SYMBOLICS AID IN DEBUG

Symbolics allow access to program symbols by name rather than cumbersome physical addresses. Symbolic debug speeds the debugging process by reducing reliance on memory maps. In a dynamic development process, user variables can be used as parameters for ICE-286 commands resulting in a consistent debug environment.

80287 NUMERICS SUPPORT

The ICE-286 emulator provides emulation support for the 80287 numerics processor. 80287 registers can be displayed and modified allowing full debug support for numerics.

FEATURES

MULTIPLE HIGH-SPEED COMMUNICATION LINKS

Two communication links are available for use in conjunction with the host IBM PC AT. The ICE-286 emulator uses either serial (RS-232-C) or a parallel (GPIB) link. A user supplied National Instruments (IEEE-488) GPIB communication board provides parallel transfers at rates up to 300 Kbytes per second.

SOFTWARE ANALYSIS (iPAT™)

Intel's Performance Analysis Tool (iPAT™) is designed to increase team productivity with features like interrupt latency measurement, code coverage analysis and software module performance analysis. These features enable the user to design reliable, high performance embedded control products. The ICE-286 emulator has an external 60 pin connector for iPAT.

BUILT-IN SUPPORT FOR LOGIC ANALYSIS

General-purpose logic analyzers can be used in conjunction with the ICE-286 to provide detailed timing of specific events. The ICE-286 emulator provides an external sync signal for triggering logic analysis, making complex trigger sequence programming easy. An additional 60 pin connector is included for the logic analyzer.

WORLDWIDE SERVICE AND SUPPORT

The ICE-286 emulator is supported by Intel's worldwide service and support organization. Total hardware and software support is available including a hotline number when the need is there.

SPECIFICATIONS

PERSONAL COMPUTER REQUIREMENTS

The ICE-286 emulator is hosted on an IBM PC AT. The emulator has been tested and evaluated on an IBM PC AT. The PC AT must meet the following minimum requirements:

- 640 Kbytes of Memory
- Intel Above Board with at Least 1 Mbyte of Expansion Memory
- One 360 Kbytes or One 1.2 Mbytes Floppy Disk Drive
- One 20 Mbytes Fixed-Disk Drive
- PC-DOS 3.2 or Later
- A Serial Port (COM1 or COM2) Supporting Minimally at 9600 Baud Data Transfers, or a National Instruments GPIB-PC2A Board.
- IBM PC AT BIOS

ELECTRICAL CONSIDERATIONS

Icc 1050 mA

ENVIRONMENTAL SPECIFICATIONS

Temperature 10°C to 40°C Ambient
Storage Temperature - 40°C to 70°C

PHYSICAL DESCRIPTION AND CHARACTERISTICS

The ICE-286 Emulator consists of the following components:

Unit	Width		Height		Length	
	Inches	Cm.	Inches	Cm.	Inches	Cm.
Emulator Control Unit	10.40	26.40	1.70	4.30	20.70	52.60
Power Supply	7.60	19.00	4.15	10.70	11.00	27.90
User Probe	3.70	9.40	.65	1.60	7.00	17.80
User Cable/Plcc					22.00	55.90
Hinge Cable					3.40	8.60
Crystal Power Accessory	4.30	10.90	.60	1.50	6.70	17.00
CPA Power Cable					9.00	22.90



SPECIFICATIONS

TIMING/DC CONSIDERATIONS

ICETM-286 USER PIN DIFFERENCES				
PARAMETER	COMPONENT SPEC.		ICETM-286 SPEC.	
	Min	Max	Min	Max
2 System (CLK) low time	11	237	14	236
3 System (CLK) high time	13	239	14	236
8 Read Data Setup	5		7	
10 /Ready Setup	22		24	
12a Status/peak # active delay	3	18	3	20
12b Status/peak # inactive delay	3	20	3	22
13 Address valid delay	1	32	1	34
14 Write data valid delay	0	30	0	33
15 Address/status/data float	0	32	0	34
Consult User Guide for additional specifications.				

ORDERING INFORMATION

ICE286	ICE-286 NMOS System including ICE S/W packages (Requires DOS 3.XX PC AT with Above Board)	D86ASM286NL	286 macro assembler 286 builder/binder/mapper utilities for DOS 3.XX.
ICE286AB	ICE-286 NMOS System including ICE S/W packages and Intel's 2 MByte Above Board (PCMB 4125) (Requires DOS 3.XX PC-AT)	D86C286NL	286 C compiler and run time libraries for DOS 3.XX.
ICE286PAT	ICE-286 NMOS System including ICE S/W Packages and the iPAT system (Requires DOS 3.XX PC AT with Above Board)	D86PLM286NL	286 PL/M compiler for DOS 3.XX.



DOMESTIC SALES OFFICES

ALABAMA

Intel Corp.
5015 Bradford Dr., #2
Huntsville 35805
Tel: (205) 830-4010
FAX: (205) 837-2640

ARIZONA

Intel Corp.
410 North 44th Street
Suite 500
Phoenix 85008
Tel: (602) 231-0386
FAX: (602) 244-0446

Intel Corp.
7225 N. Mona Lisa Rd.
Suite 215
Tucson 85741
Tel: (602) 544-0227
FAX: (602) 544-0232

CALIFORNIA

Intel Corp.
21515 Vanowen Street
Suite 116
Canoga Park 91303
Tel: (818) 704-8500
FAX: (818) 340-1144

Intel Corp.
300 N. Continental Blvd.
Suite 100
El Segundo 90245
Tel: (213) 640-6040
FAX: (213) 640-7133

Intel Corp.
1 Sierra Gate Plaza
Suite 280C
Roseville 95678
Tel: (916) 782-8086
FAX: (916) 782-8153

Intel Corp.
9665 Chesapeake Dr.
Suite 325
San Diego 92123
Tel: (619) 292-8086
FAX: (619) 292-0628

Intel Corp.*
400 N. Tustin Avenue
Suite 450
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114
FAX: (714) 541-9157

Intel Corp.*
San Tomas 4
2700 San Tomas Expressway
2nd Floor
Santa Clara 95051
Tel: (408) 986-8086
TWX: 910-338-0255
FAX: (408) 727-2620

COLORADO

Intel Corp.
4445 Northpark Drive
Suite 100
Colorado Springs 80907
Tel: (719) 594-6622
FAX: (303) 594-0720

Intel Corp.*
600 S. Cherry St.
Suite 700
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289
FAX: (303) 322-8670

CONNECTICUT

Intel Corp.
301 Lee Farm Corporate Park
83 Wooster Heights Rd.
Danbury 06810
Tel: (203) 748-3130
FAX: (203) 794-0339

FLORIDA

Intel Corp.
800 Fairway Drive
Suite 160
Deerfield Beach 33441
Tel: (305) 421-0506
FAX: (305) 421-2444

Intel Corp.
5850 T.G. Lee Blvd.
Suite 340
Orlando 32822
Tel: (407) 240-8000
FAX: (407) 240-8097

Intel Corp.
11300 4th Street North
Suite 170
St. Petersburg 33716
Tel: (813) 577-2413
FAX: (813) 578-1607

GEORGIA

Intel Corp.
20 Technology Parkway
Suite 150
Norcross 30092
Tel: (404) 449-0541
FAX: (404) 605-9762

ILLINOIS

Intel Corp.*
Woodfield Corp. Center III
300 N. Martingale Road
Suite 400
Schaumburg 60173
Tel: (708) 605-8031
FAX: (708) 706-9762

INDIANA

Intel Corp.
8910 Purdue Road
Suite 350
Indianapolis 46268
Tel: (317) 875-0623
FAX: (317) 875-8938

IOWA

Intel Corp.
1930 St. Andrews Drive N.E.
2nd Floor
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS

Intel Corp.
10985 Cody St.
Suite 140
Overland Park 66210
Tel: (913) 345-2727
FAX: (913) 345-2076

MARYLAND

Intel Corp.*
10010 Junction Dr.
Suite 200
Annapolis Junction 20701
Tel: (301) 206-2860
FAX: (301) 206-3677
(301) 206-3678

MASSACHUSETTS

Intel Corp.*
Westford Corp. Center
3 Carlisle Road
2nd Floor
Westford 01886
Tel: (508) 692-0960
TWX: 710-343-6333
FAX: (508) 692-7867

MICHIGAN

Intel Corp.
7071 Orchard Lake Road
Suite 100
West Bloomfield 48322
Tel: (313) 851-8096
FAX: (313) 851-8770

MINNESOTA

Intel Corp.
3500 W. 80th St.
Suite 360
Bloomington 55431
Tel: (612) 835-6722
TWX: 910-576-2867
FAX: (612) 831-6497

MISSOURI

Intel Corp.
4203 Earth City Expressway
Suite 131
Earth City 63045
Tel: (314) 291-1990
FAX: (314) 291-4341

NEW JERSEY

Intel Corp.*
Lincroft Office Center
125 Half Mile Road
Red Bank 07701
Tel: (908) 747-2233
FAX: (908) 747-0993

Intel Corp.
280 Corporate Center
75 Livingston Avenue
First Floor
Roseland 07068
Tel: (201) 740-0111
FAX: (201) 740-0626

NEW YORK

Intel Corp.*
850 Crosskeys Office Park
Fairport 14450
Tel: (716) 425-2750
TWX: 510-253-7391
FAX: (716) 223-2561

Intel Corp.*
2950 Express Dr., South
Suite 130
Islandia 11722
Tel: (516) 231-3300
TWX: 510-227-6236
FAX: (516) 348-7939

Intel Corp.
300 Westage Business Center
Suite 230
Fishkill 12524
Tel: (914) 897-3860
FAX: (914) 897-3125

Intel Corp.
Seventeen State Street
14th Floor
New York 10004
Tel: (212) 248-8086
FAX: (212) 248-0886

NORTH CAROLINA

Intel Corp.
5800 Executive Center Dr.
Suite 105
Charlotte 28212
Tel: (704) 568-8966
FAX: (704) 535-2236

Intel Corp.
5540 Centerview Dr.
Suite 215
Raleigh 27606
Tel: (919) 851-9537
FAX: (919) 851-8974

OHIO

Intel Corp.*
3401 Park Center Drive
Suite 220
Dayton 45414
Tel: (513) 890-5350
TWX: 810-450-2528
FAX: (513) 890-8658

Intel Corp.*
25700 Science Park Dr.
Suite 100
Beachwood 44122
Tel: (216) 464-2736
TWX: 810-427-9298
FAX: (804) 282-0673

OKLAHOMA

Intel Corp.
6801 N. Broadway
Suite 115
Oklahoma City 73162
Tel: (405) 848-8086
FAX: (405) 840-9819

OREGON

Intel Corp.
15254 N.W. Greenbrier Pkwy.
Building B
Beaverton 97006
Tel: (503) 645-8051
TWX: 910-467-8741
FAX: (503) 645-8181

PENNSYLVANIA

Intel Corp.*
925 Harvest Drive
Suite 200
Blue Bell 19422
Tel: (215) 641-1000
FAX: (215) 641-0785

Intel Corp.*
400 Penn Center Blvd.
Suite 610
Pittsburgh 15235
Tel: (412) 823-4970
FAX: (412) 828-7578

PUERTO RICO

Intel Corp.
South Industrial Park
P.O. Box 910
Las Piedras 00671
Tel: (809) 733-8616

TEXAS

Intel Corp.
8911 N. Capital of Texas Hwy.
Suite 4230
Austin 78759
Tel: (512) 794-8086
FAX: (512) 338-9335

Intel Corp.*
12000 Ford Road
Suite 400
Dallas 75234
Tel: (214) 241-8087
FAX: (214) 484-1180

Intel Corp.*
7322 S.W. Freeway
Suite 1490
Houston 77074
Tel: (713) 988-8086
TWX: 910-881-2490
FAX: (713) 988-3660

UTAH

Intel Corp.
428 East 6400 South
Suite 104
Murray 84107
Tel: (801) 263-8051
FAX: (801) 268-1457

VIRGINIA

Intel Corp.
9030 Stony Point Pkwy.
Suite 360
Richmond 23235
Tel: (804) 330-9393
FAX: (804) 330-3019

WASHINGTON

Intel Corp.
155 108th Avenue N.E.
Suite 386
Bellevue 98004
Tel: (206) 453-8086
TWX: 910-443-3002
FAX: (206) 451-9556

Intel Corp.
408 N. Mullan Road
Suite 102
Spokane 99206
Tel: (509) 928-8086
FAX: (509) 928-9467

WISCONSIN

Intel Corp.
330 S. Executive Dr.
Suite 102
Brookfield 53005
Tel: (414) 784-8087
FAX: (414) 796-2115

CANADA

BRITISH COLUMBIA

Intel Semiconductor of
Canada, Ltd.
4585 Canada Way
Suite 202
Burnaby V5G 4L6
Tel: (604) 298-0387
FAX: (604) 298-8234

ONTARIO

Intel Semiconductor of
Canada, Ltd.
2650 Queensview Drive
Suite 250
Ottawa K2B 8H6
Tel: (613) 829-9714
FAX: (613) 820-5936

Intel Semiconductor of
Canada, Ltd.
190 Attwell Drive
Suite 500
Rexdale M9W 6H8
Tel: (416) 675-2105
FAX: (416) 675-2438

QUEBEC

Intel Semiconductor of
Canada, Ltd.
1 Rue Holiday
Suite 115
Tour East
Pl. Claire H9R 5N3
Tel: (514) 694-9130
FAX: 514-694-0064

*Sales and Service Office
*Field Application Location



INTERNATIONAL SALES OFFICES

AUSTRALIA

Intel Australia Pty. Ltd.
Unit 13
Allambie Grove Business Park
25 Frenchs Forest Road East
Frenchs Forest, NSW, 2086
Tel: 61-2975-3300
FAX: 61-2975-3375

BRAZIL

Intel Semicondutores do Brazil LTDA
Av. Paulista, 1159-CJS 404/405
01311 - Sao Paulo - S.P.
Tel: 55-11-287-5809
TLX: 3911153146 ISDB
FAX: 55-11-287-5119

CHINA/HONG KONG

Intel PRC Corporation
15/F, Office 1, Citic Bldg.
Jian Guo Men Wai Street
Beijing, PRC
Tel: (1) 500-4850
TLX: 22947 INTEL CN
FAX: (1) 500-2953

Intel Semiconductor Ltd.*
10/F East Tower
Bond Center
Queensway, Central
Hong Kong
Tel: (852) 844-4555
FAX: (852) 868-1989

INDIA

Intel Asia Electronics, Inc.
4/2, Samrah Plaza
St. Mark's Road
Bangalore 560001
Tel: 011-91-812-215065
TLX: 953-845-2646 INTEL IN
FAX: 091-812-215067

JAPAN

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26
Tel: 0298-47-8511
TLX: 3656-160
FAX: 0298-47-8450

Intel Japan K.K.*
Daichi Mitsugi Bldg.
1-889 Fuchu-cho
Fuchu-shi, Tokyo 183
Tel: 0423-60-7871
FAX: 0423-60-0315

Intel Japan K.K.*
Bldg. Kumagaya
2-69 Hon-cho
Kumagaya-shi, Saitama 360
Tel: 0485-24-6871
FAX: 0485-24-7518

Intel Japan K.K.*

Kawa-asa Bldg.
2-11-5 Shin-Yokohama
Kohoku-ku, Yokohama-shi
Kanagawa, 222
Tel: 045-474-7661
FAX: 045-471-4394

Intel Japan K.K.*
Ryokuchi-Eki Bldg.
2-4-1 Terauchi
Toyonaka-shi, Osaka 560
Tel: 06-863-1091
FAX: 06-863-1084

Intel Japan K.K.
Shinmaru Bldg.
1-5-1 Marunouchi
Chiyoda-ku, Tokyo 100
Tel: 03-201-3621
FAX: 03-201-6850

Intel Japan K.K.
Green Bldg.
1-16-20 Nishiki
Naka-ku, Nagoya-shi
Aichi 450
Tel: 052-204-1261
FAX: 052-204-1285

KOREA

Intel Korea, Ltd.
18th Floor, Life Bldg.
61 Yoido-dong, Youngdeungpo-Ku
Seoul 150-010
Tel: (2) 784-8186, 8286, 8386
TLX: K29312 INTELKO
FAX: (2) 784-8086

SINGAPORE

Intel Singapore Technology, Ltd.
101 Thomson Road #21-05/06
United Square
Singapore 1130
Tel: 250-7811
TLX: 39921 INTEL
FAX: 250-9256

TAIWAN

Intel Technology Far East Ltd.
8th Floor, No. 205
Bank Tower Bldg.
Tung Hua N. Road
Taipei
Tel: 886-2-716-9660
FAX: 886-2-717-2455

INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

ARGENTINA

Dafsys S.R.L.
Chacabuco, 90-6 Piso
1069-Buenos Aires
Tel: 54-1-334-7726
FAX: 54-1-334-1871

AUSTRALIA

Email Electronics
15-17 Hume Street
Huntingdale, 3166
Tel: 011-61-3-544-8244
TLX: AA 30895
FAX: 011-61-3-543-8179

NSD-Australia
205 Middleborough Rd.
Box Hill, Victoria 3128
Tel: 03 8900970
FAX: 03 8990819

BRAZIL

Elebra Componentes
Rua Geraldo Flausina Gomes, 78
7 Andar
04575 - Sao Paulo - S.P.
Tel: 55-11-534-9641
TLX: 55-11-54593/54591
FAX: 55-11-534-9424

CHINA/HONG KONG

Novel Precision Machinery Co., Ltd.
Room 728 Trade Square
681 Cheung Sha Wan Road
Kowloon, Hong Kong
Tel: (852) 360-8999
TWX: 32032 NVTNL HX
FAX: (852) 725-3695

INDIA

Micronic Devices
Arun Complex
No. 65 D.V.G. Road
Basavanagudi
Bangalore 560 004
Tel: 011-91-812-600-631
011-91-812-611-365
TLX: 9538458332 MDBBG

Micronic Devices
No. 516 5th Floor
Swastik Chambers
Sion, Trombay Road
Chembur
Bombay 400 071
TLX: 9531 171447 MDEV

Micronic Devices
25/8, 1st Floor
Bada Bazaar Marg
Old Rajinder Nagar
New Delhi 110 060
Tel: 011-91-11-5723509
011-91-11-589771
TLX: 031-63253 MDND IN

Micronic Devices
6-3-348/12A Dwarakapuri Colony
Hyderabad 500 482
Tel: 011-91-842-226748

S&S Corporation
1587 Kooser Road
San Jose, CA 95118
Tel: (408) 978-6216
TLX: 820281
FAX: (408) 978-8635

JAPAN

Asahi Electronics Co. Ltd.
KMM Bldg. 2-14-1 Asano
Kokurakita-ku
Kitakyushu-shi 802
Tel: 093-511-6471
FAX: 093-551-7861

CTC Components Systems Co., Ltd.
4-8-1 Dobashi, Miyamae-ku
Kawasaki-shi, Kanagawa 213
Tel: 044-852-5121
FAX: 044-877-4268

Dia Semicon Systems, Inc.
Flower Hill Shinmachi Higashi-kan
1-23-9 Shinmachi, Setagaya-ku
Tokyo 154
Tel: 03-439-1600
FAX: 03-439-1601

Okaya Koki
2-4-18 Sakae
Naka-ku, Nagoya-shi 460
Tel: 052-204-2916
FAX: 052-204-2901

Ryoyo Electro Corp.
Konwa Bldg.
1-12-22 Tsukiji
Chuo-ku, Tokyo 104
Tel: 03-546-5011
FAX: 03-546-5044

KOREA

J-Tek Corporation
Dong Sung Bldg. 9/F
158-24, Samsung-Dong, Kangnam-Ku
Seoul 135-090
Tel: (822) 557-8039
FAX: (822) 557-8304

Samsung Electronics
Samsung Main Bldg.
150 Taebyeung-Ro-2KA, Chung-Ku
Seoul 100-102
C.P.O. Box 8780
Tel: (822) 751-3680
TWX: KORSST K 27970
FAX: (822) 753-9065

MEXICO

SSB Electronics, Inc.
675 Palomar Street, Bldg. 4, Suite A
Chula Vista, CA 92011
Tel: (619) 585-3253
TLX: 287751 CBALL UR
FAX: (619) 585-8322

Dicopel S.A.
Tochtitl 368 Fracc. Ind. San Antonio
Azcapotzalco
C.P. 02760-Mexico, D.F.
Tel: 52-5-561-3211
TLX: 177 3790 Dicome
FAX: 52-5-561-1279
Phi S.A. de C.V.
Fco. Villa esq. Ajusco s/n
Cuernavaca - Morelos
Tel: 52-73-13-9412
FAX: 52-73-17-5333

NEW ZEALAND

Email Electronics
36 Olive Road
Penrose, Auckland
Tel: 011-64-9-591-155
FAX: 011-64-9-592-681

SINGAPORE

Electronic Resources Pte, Ltd.
17 Harvey Road
#03-01 Singapore 1336
Tel: (65) 283-0888
TWX: RS 56541 ERS
FAX: (65) 289-5327

SOUTH AFRICA

Electronic Building Elements
178 Erasmus St. (off Watermeyer St.)
Meyerspark, Pretoria, 0184
Tel: 011-2712-803-7680
FAX: 011-2712-803-8294

TAIWAN

Micro Electronics Corporation
12th Floor, Section 3
285 Nanking East Road
Taipei, R.O.C.
Tel: (886) 2-7198419
FAX: (886) 2-7197916

Acer Sertek Inc.
15th Floor, Section 2
Chien Kuo North Rd.
Taipei 18479 R.O.C.
Tel: 886-2-501-0055
TWX: 23756 SERTEK
FAX: (886) 2-5012521

*Field Application Location



UNITED STATES

Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

JAPAN

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26

FRANCE

Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-Yvelines Cedex

UNITED KINGDOM

Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ

GERMANY

Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen

HONG KONG

Intel Semiconductor Ltd.
10/F East Tower
Bond Center
Queensway, Central

CANADA

Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8

Microprocessors

This year marks the 20-year anniversary of the invention of the microprocessor. This invention resulted in the mass proliferation of computing technology creating the microcomputer revolution of the 1980's. Intel has continued its technological lead with faster and more capable products for microcomputing.

Intel offers an architecture that provides both the performance and the compatibility needed to take progress from one generation of products to the next.

This handbook contains extensive information on Intel's microprocessor families, numeric coprocessors, cache and memory controllers, and floppy and hard disk controllers. A development tools section is also included for the 8051, 8096, 8086/186/188, 286, 386™ and 486™ processors.

The data sheets and application notes contained in this handbook offer comprehensive charts, diagrams, instructions and hardware information for leading-edge 32-bit-system development.