



Microprocessor and Peripheral Handbook

Volume II—Peripheral





LITERATURE

To order Intel literature write or call:

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

Intel Literature:
(800) 548-4725

Use the order blank on the facing page or call our Toll Free Number listed above to order literature. Remember to add your local sales tax and a 10% postage charge for U.S. and Canada customers, 20% for outside U.S. customers.

1987 HANDBOOKS

Product line handbooks contain data sheets, application notes, article reprints and other design information.

NAME	ORDER NUMBER	*PRICE IN U.S. DOLLARS
COMPLETE SET OF 9 HANDBOOKS Save \$50.00 off the retail price of \$175.00	231003	\$125.00
MEMORY COMPONENTS HANDBOOK	210830	\$18.00
MICROCOMMUNICATIONS HANDBOOK	231658	\$20.00
EMBEDDED CONTROLLER HANDBOOK (includes Microcontrollers and 8085, 80186, 80188)	210918	\$18.00
MICROPROCESSOR AND PERIPHERAL HANDBOOK (2 Volume Set)	230843	\$25.00
DEVELOPMENT TOOLS HANDBOOK	210940	\$18.00
DOS DEVELOPMENT SOFTWARE CATALOG	280199	N/C
OEM BOARDS AND SYSTEMS HANDBOOK	280407	\$18.00
MILITARY HANDBOOK	210461	\$18.00
COMPONENTS QUALITY/RELIABILITY HANDBOOK	210997	\$20.00
SYSTEMS QUALITY/RELIABILITY HANDBOOK	231762	\$20.00
PRODUCT GUIDE Overview of Intel's complete product lines	210846	N/C
LITERATURE PRICE LIST List of Intel Literature	210620	N/C
INTEL PACKAGING OUTLINES AND DIMENSIONS Packaging types, number of leads, etc.	231369	N/C

*These prices are for the U.S. and Canada only. In Europe and other international locations, please contact your local Intel Sales Office or Distributor for literature prices.



MICROPROCESSOR AND PERIPHERAL HANDBOOK

VOLUME II PERIPHERAL

1987

About Our Cover:

Intel's family of peripheral products is large, powerful and diverse, offering many performance options to the designer of microprocessor-based systems. In keeping with our "building block" approach to systems design, the cover concept uses many cylinders representing peripherals in a variety of colors and sizes surrounding a microprocessor represented by a large cube.

Concept/Design: Hall Kelley, Concept/Photography: R. J. Muna



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

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

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

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

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, FASTPATH, GENIUS, i, i, ICE, iCEL, iCS, iDBP, iDIS, i2ICE, iLBX, Inboard, i_m, iMDDX, iMMX, Insite, Intel, int_el, int_elBOS, Intelevison, int_el_igent Identifier, int_el_igent Programming, Intellec, Intellink, iOSP, iPDS, iPSC, iRMX, iSBC, iSBX, iSDM, iSXM, KEPPROM, Library Manager, MAP-NET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, PC-BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, UPI, and VLSiCEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix 4-SITE.

Ethernet is a trademark of Xerox.

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

*MULTIBUS is a patented Intel bus.

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

Intel Literature Department
SC6-59
P.O. Box 58065
Santa Clara, CA 95052-8065

Table of Contents

Alphanumeric Index	vii
CHAPTER 1	
Memory Controllers	
DATA SHEETS	
✓8203 64K Dynamic RAM Controller	1-1
8206 Error Detection and Correction Unit	1-17
8207 Dual-Port Dynamic RAM Controller	1-39
82C08 CHMOS Dynamic RAM Controller	1-86
APPLICATION NOTES	
Interfacing the 8207 Dynamic RAM Controller to the 80186 AP-167	1-115
Interfacing the 8207 Advanced Dynamic RAM Controller to the 80286 AP-168	1-121
USERS MANUAL	
8207 Users Manual	1-128
82C08 User's Manual	1-152
CHAPTER 2	
Support Peripherals	
DATA SHEETS	
8231A Arithmetic Processing Unit	2-1
8253/8253-5 Programmable Interval Timer	2-14
8254 Programmable Interval Timer	2-25
82C54 CHMOS Programmable Interval Timer	2-46
8255A/8255A-5 Programmable Peripheral Interface	2-63
82C55A CHMOS Programmable Peripheral Interface	2-87
8256AH Multifunction Microprocessor Support Controller	2-110
8279/8279-5 Programmable Keyboard/Display Interface	2-134
APPLICATION NOTES	
Designing with the 8256 AP-153	2-151
8256AH Application Note AP-183	2-222
CHAPTER 3	
Floppy Disk Controllers	
DATA SHEETS	
8272A Single/Double Density Floppy Disk Controller	3-1
82072 CHMOS High Integration Floppy Disk	3-32
APPLICATION NOTES	
An Intelligent Data Base System Using the 8272 AP-116	3-75
✓Software Design and Implementation of Floppy Disk Systems AP-121	3-116
Designing with the 82072 CHMOS Single-Chip Floppy Disk Controller AP-289	3-164
CHAPTER 4	
Hard Disk Controllers	
DATA SHEETS	
82064 Winchester Disk Controller with On-Chip Error Detection and Correction	4-1
APPLICATION NOTES	
Multimode Winchester Controller Using the 82064 AP-402	4-33
CHAPTER 5	
Universal Peripheral Interface Slave Microcontrollers	
DATA SHEETS	
UPI™-452 Slave Microcontroller (80/83/87452)	5-1
UPI™-41/42 8-Bit Slave Microcontroller (80/8741/42AH)	5-54
8243 MCS-48 Input/Output Expander	5-73

Table of Contents (Continued)

Microprocessor Peripherals UPITM User's Manual	5-79
APPLICATION NOTES	
Applications Using the 8042 UPITM Microcontroller	5-143
Complex Peripheral Control with the UPITM-42 AP-161	5-147
An 8741A/8041A Digital Cassette Controller AP-90	5-202
UPITM-452 Accelerates iAPX 286 Bus Performance AP-281	5-209
SYSTEM SUPPORT	
ICE-42 8042 In-Circuit Emulator	5-229
iUP-200/iUP-201 Universal PROM Programmers	5-237

CHAPTER 6

Alphanumeric Terminal Controllers

DATA SHEETS

8275H Programmable CRT Controller	6-1
8276H Small System CRT Controller	6-32

APPLICATION NOTES

A Low Cost CRT Terminal Using the 8275 AP-62	6-49
--	------

CHAPTER 7

Graphics Coprocessors

DATA SHEETS

82716 Video Storage Display Device	7-1
82730 Text Coprocessor	7-34
82786 CHMOS Graphics Coprocessor	7-76

APPLICATION NOTES

A Low Cost and High Integration System Using 82716 AP-268	7-119
82786 Hardware Configuration AP-270	7-171

ARCHITECTURAL OVERVIEWS

The 82786 CHMOS Graphics Coprocessor AP-259	7-232
---	-------

Alphanumeric Index

82C08 CHMOS Dynamic RAM Controller	1-86
82C08 User's Manual	1-152
82C54 CHMOS Programmable Interval Timer	2-46
82C55A CHMOS Programmable Peripheral Interface	2-87
8203 64K Dynamic RAM Controller	1-1
8206 Error Detection and Correction Unit	1-17
82064 Winchester Disk Controller with On-Chip Error Detection and Correction	4-1
8207 Dual-Port Dynamic RAM Controller	1-39
8207 Users Manual	1-128
82072 CHMOS High Integration Floppy Disk Controller	3-32
8231A Arithmetic Processing Unit	2-1
8243 MCS-48 Input/Output Expander	5-73
8253/8253-5 Programmable Interval Timer	2-14
8254 Programmable Interval Timer	2-25
8255A /8255A-5 Programmable Peripheral Interface	2-63
8256AH Application Note AP-183	2-222
8256AH Multifunction Microprocessor Support Controller	2-110
82716 Video Storage Display Device	7-1
✓8272A Single/Double Density Floppy Disk Controller	3-1
82730 Text Coprocessor	7-34
✓8275H Programmable CRT Controller	6-1
8276H Small System CRT Controller	6-32
82786 CHMOS Graphics Coprocessor	7-76
82786 Hardware Configuration AP-270	7-171
8279/8279-5 Programmable Keyboard/Display Interface	2-134
A Low Cost and High Integration System Using 82716 AP-268	7-119
A Low-Cost CRT Terminal Using the 8275 AP-62	6-49
An Intelligent Data Base System Using the 8272 AP-116	3-75
An 8741A/8041A Digital Cassette Controller AP-90	5-202
Applications Using the 8042 UPI™ Microcontroller	5-143
Complex Peripheral Control with the UPI™-42 AP-161	5-147
Designing with the 82072 CHMOS Single-Chip Floppy Disk Controller AP-289	3-164
Designing with the 8256 AP-153	2-151
iUP-200/iUP-201 Universal PROM Programmers	5-237
Interfacing the 8207 Advanced Dynamic RAM Controller to the 80286 AP-168	1-121
Interfacing the 8207 Dynamic RAM Controller to the 80186 AP-167	1-115
ICE-42 8042 In-Circuit Emulator	5-229
Microprocessor Peripherals UPI™ User's Manual	5-79
Multimode Winchester Controller Using the 82064 AP-402	4-33
Software Design and Implementation of Floppy Disk Systems AP-121	3-116
The 82786 CHMOS Graphics Coprocessor AP-259	7-232
UPI™-41/42 8-Bit Slave Microcontroller (80/8741/42AH)	5-54
UPI™-452 Accelerates iAPX 286 Bus Performance AP-281	5-209
UPI™-452 Slave Microcontroller (80/83/87452)	5-1

CUSTOMER SUPPORT

CUSTOMER SUPPORT

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

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Such support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is quite extensive. It includes factory repair services and worldwide field service offices providing hardware repair services, software support services, customer training classes, and consulting services.

HARDWARE SUPPORT SERVICES

Intel is committed to providing an international service support package through a wide variety of service offerings available from Intel Hardware Support.

SOFTWARE SUPPORT SERVICES

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 includes updates and the subscription service. Contracts are sold in environments which represent product groupings (i.e., iRMX environment).

CONSULTING SERVICES

Intel provides field systems engineering services for any phase of your development or support effort. You can use our systems engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training, and customizing or tailoring 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.

INTRODUCTION

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

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

REDUCED TIME TO MARKET

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

INCREASED RELIABILITY

At Intel, the rate of failure for devices is carefully tracked. Highest reliability is a tangible goal that translates to higher reliability for your product, reduced downtime, and reduced repair costs. And as more and more functions are intergrated on a single VLSI device, the resulting system requires less power, produces less heat, and requires fewer mechanical connections—again resulting in greater system reliability.

LOWER PRODUCTION COST

By minimizing design time, increasing reliability, and

replacing numerous parts, microprocessor and peripheral solutions can contribute dramatically to lower product costs.

HIGHER SYSTEM PERFORMANCE

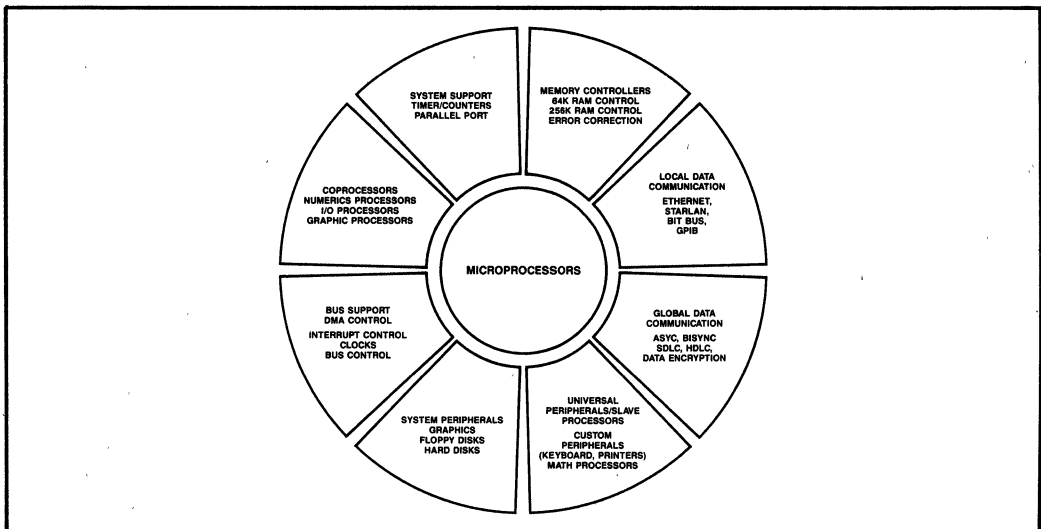
Intel microprocessors and peripherals provide the highest system performance for the demands of today's (and tomorrow's) microprocessor-based applications. For example, the 80386 32 bit offers the highest performance for multitasking, multiuser systems. Intel's peripheral products have been designed with the future in mind. They support all of Intel's 8, 16 and 32 bit processors.

HOW TO USE THE GUIDE

The following application guide illustrates the range of microprocessors and peripherals that can be used for the applications in the vertical column of the left. The peripherals are grouped by the I/O function they control. CRT datacommunication, universal (user programmable), mass storage dynamic RAM controllers, and CPU/bus support.

An "X" in a horizontal application row indicates a potential peripheral or CPU, depending upon the features desired. For example, a conversational terminal could use either of the three display controllers, depending upon features like the number of characters per row or font capability. A "Y" indicates a likely candidate, for example, the 8272A Floppy Disk Controller in a small business computer.

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



APPLICATION CHART

POTENTIAL APPLICATION X — TYPICAL APPLICATION Y

APPLICATION	MICROPROCESSORS						NUMERIC PROCESSORS			DISPLAY				DATA COMMUNICATIONS								
	8088	8086	80188	80186	80286	80386	8087	80287	80387	8275/76	82716	82730	82786	8251A	82510	8273	8274	8291A/92/94	82530	82588	82586/501/502	
PERIPHERALS																						
Printers	X		X	Y		X								X	X							
Plotters	X	X	X	Y		X								X	X							
Keyboards			X	Y										X	X							
MASS STORAGE																						
Hard Disk	X	X	Y	Y																		
Mini Winchester	X		Y	Y																		
Tape			X	Y																		
Cassette			Y	Y																		
Floppy/Mini			Y	Y																		
COMMUNICATIONS																						
Digital Telephone																						
ISDN																						
PBX			X	X	Y	Y										X	X		X	X	X	Y
LANs	X	X	X	Y	Y														X	X	X	
Modems														X	X				X	X	X	
Bisync																			X	X	X	
SDLC/HDLC																			X	X	X	
Serial Backplane																			X	X	X	
Central Office		X		X	Y	Y									X	X			X	X	X	
Network Control		Y		X	Y	Y										X			X	X	X	
OFFICE/BUS																						
Copier/FAX	X		X	Y																		X
Wordprocessor	X	X	X	X	Y	Y					X	Y		X								X
Typewriter			X	X							X	X										
Electronic Mail			X	X	X	X					X	Y										X
Transaction System			Y	X	X	X					X	X										X
Data Entry	X	X	X	X	X	X				X	X	X		X	X							X
COMPUTERS																						
SM Bus Computer			X	Y	X	Y	Y	Y	Y	X	Y	Y		X	X				X	X	X	X
PC	Y	X	X	X	X	X					X	X		X	X				X	X	X	X
Portable PC			X	X	X	X					X	X		X	X				X	X	X	X
Home Computer	X	X	Y	X	X	X					X	X		X	X				X	X	X	X
TERMINALS																						
Conversational			Y																			
Graphics CRT		Y		Y	Y	X	Y	Y	Y	X	Y	Y		X	X				X	X	X	
Editing	X	X	X	X	X	X					Y	Y		X	X				X	X	X	
Intelligent	X	X	Y	Y							Y	Y		X	X				X	X	X	
Integrated Voice/Data	X	X	X	X							Y	Y		X	X				X	X	X	
Videotex	X	X	X	X							Y	Y		X	X				X	X	X	
Printing: Laser, Impact	X	X	X	X							Y	Y		X	X				X	X	X	
Portable	X	X	Y	X							Y	Y		X	X				X	X	X	
INDUSTRIAL/AUTO																						
Robotics			Y	Y	X	Y	Y	Y	Y		X	X										
Network	X	X	X	X	X	Y	Y	Y	Y		X	X		X	X				X	X	X	X
Numeric Control			X	X	X	Y	Y	Y	Y		X	X							X	X	X	X
Process Control	X	X	X	X	X	Y	Y	Y	Y		X	X				X			X	X	X	X
Instrumentation	X	X	X	X	X	X	X	X	X		X	X							X	X	X	X
Aviation/Navigation			X	X	X	X					Y	Y		X	X				X	X	X	X
INDUSTRIAL/DATA ACQUISITION																						
Laboratory Instrumentation	X	X	Y	X	X	X					Y	Y		X	X			Y				
Source Data	X	X	Y	Y			X	X	X		X	Y		X	X							
Auto Test	X	X	Y	X	Y	X					Y	Y		X	X							X
Medical	X	X	Y	X	X	X	Y	Y	Y		Y	Y		X	X							X
Test Instruments	X	X	Y	X	X	Y					Y	Y		X	X				X	X	X	X
Security			X	Y							Y	Y		X	X							X
COMMERCIAL DATA PROCESSING																						
POS Terminal		X	X	Y		X				X	Y	X		X	X				X	X	X	X
Financial Transfer		X	X	Y		X	Y	Y	Y		X	X		X	X				X	X	X	X
Automatic Teller		X	X	Y		X					X	X		X	X				X	X	X	X
Document Processing	X	X	X	Y	X	X				X	Y	X		X	X				X	X	X	X
WORKSTATIONS																						
Office	X	X	X	Y	X	Y	Y	Y	Y		Y	Y		X	X				X	X	X	X
Engineering	X	X		Y	X	Y	Y	Y	Y		Y	Y		X	X				X	X	X	X
CAD		X		Y	Y	Y					Y	Y		X	X				X	X	X	X
MINI MAINFRAME																						
Processor & Control Store		X		Y	Y	Y				Y	Y								X	X	X	X
Database Subsystems		X		Y	X	Y	Y	Y	Y		Y	Y							X	X	X	X
I/O Subsystems			Y	Y	Y	Y													X	X	X	X
Communication Subsystem		X		Y	Y	Y								X	X				X	X	X	X

APPLICATION CHART

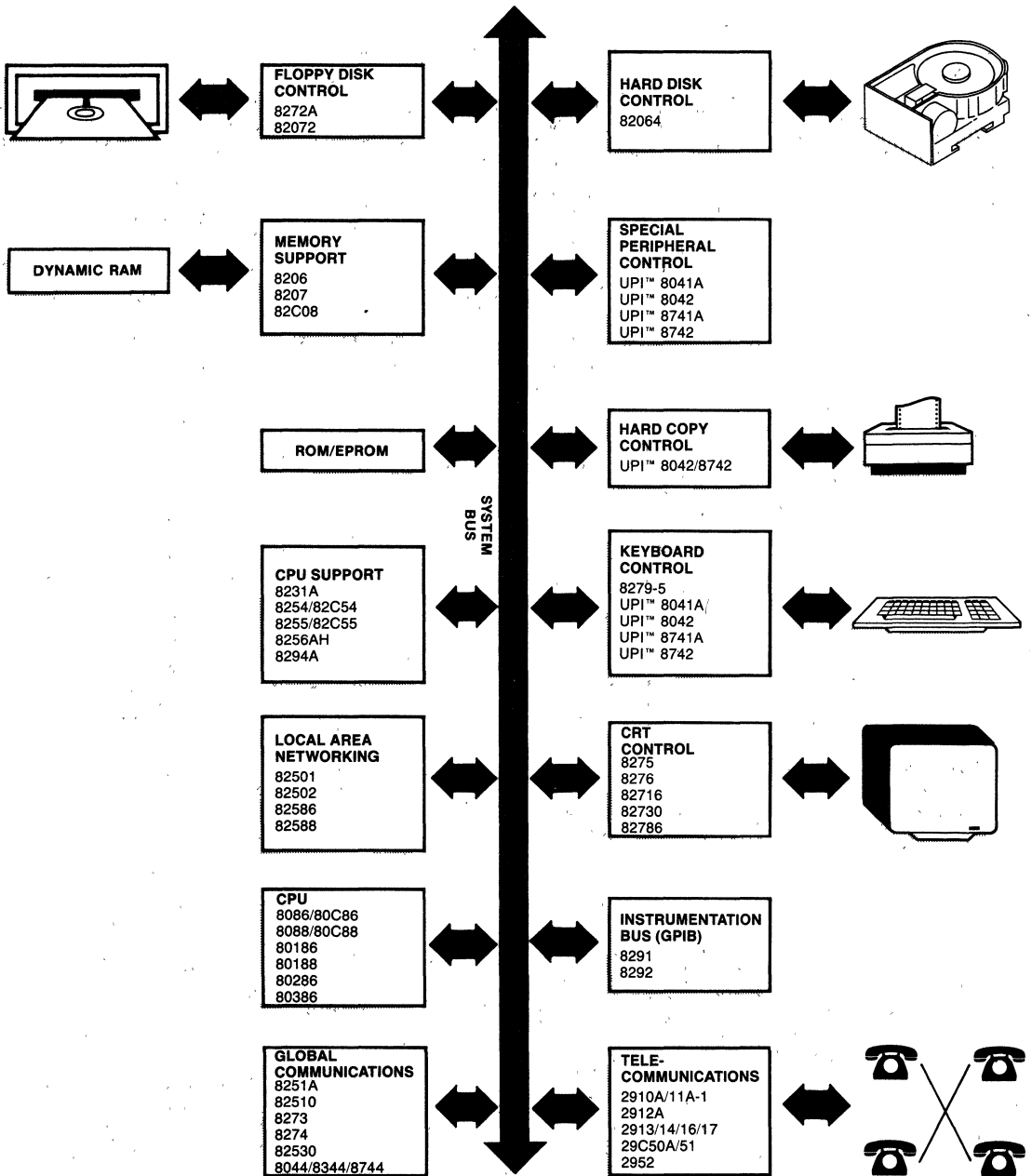
POTENTIAL APPLICATION X — TYPICAL APPLICATION Y

IPI/RUPI	DISK CONTROL			DRAM CONTROL			SUPPORT					TELECOMMUNICATION					APPLICATION					
	8041/A/2 8741	80/87C452	8272A	82072	82064	8286	8207	82C08	82258	8237	825A/C54	8255/C55	8256AH	8259A	8279	2910/11		2912	2913/14/16/17	29C50A	29C51	29C2
Y X Y	X X	X	X			X X	X X	X X	X X	Y X	X	X X X	X	Y Y								PERIPHERALS Printers Plotters Keyboards
X X	X			Y Y				Y Y	Y Y		X X											MASS STORAGE Hard Disk Mini Winchester Tape Cassette Floppy/Mini
X	X X	X X	X X			X X	X X	X X	Y X	X X	X X	X X	X X	Y Y		Y	Y	Y	X Y Y	X X	Y	COMMUNICATIONS Digital Telephone ISDN PBX LANs Modems Bisync SDLC/HDL Serial Backplane Central Office Network Control
X	X	X	X	X Y	X Y	X X	X X	X X	X X	Y Y	Y Y	X X	X X	Y Y	X		Y	Y	Y	X Y	Y	OFFICE/BUS Copier/FAX Wordprocessor Typewriter Electronic Mail Transaction System Data Entry
X X X X	X X	X Y Y Y	X Y Y Y	X Y	X X	X X	X X	X X	X X	Y Y	Y Y	X X	X X	Y Y	X							COMPUTERS SM Bus Computer PC Portable PC Home Computer
		X X	X X	X X	X X	X X	X X	X X	X X	Y Y	Y Y	X X	X X	Y Y								TERMINALS Conversational Graphics CRT Editing Intelligent Integrated Voice/Data Videotex Printing: Laser, Impact Portable
X X X X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	Y Y	Y Y	X X	X X	Y Y	X				X X	X	Y	INDUSTRIAL/AUTO Robotics Network Numeric Control Process Control Instrumentation Aviation/Navigation
X X X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	Y Y	Y Y	X X	X X	Y Y	X							INDUSTRIAL/DATA ACQUISITION Laboratory Instrumentation Source Data Auto Test Medical Test Instruments
X X	X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	Y Y	X							COMMERCIAL DATA PROCESSING POS Terminal Financial Transfer Automatic Teller Document Processing
X X		Y Y Y	Y Y Y	Y Y Y	X X X	X X X	X X X	X Y Y	X Y Y	X X X	Y Y Y	X X X	X X X	Y Y Y								WORKSTATIONS Office Engineering CAD
X X	X	X	X	X	X X X	X X X	X X X	Y Y Y	X X	X X	X X	X X	X X	Y Y Y								MINI MAINFRAME Processor & Control Store Database Subsystems I/O Subsystems Communication Subsystem



Get Your Kit Together!

Intel's Microsystem Components Kit Solution



Memory Controllers

1

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
DEPARTMENT OF CHEMISTRY
5800 S. UNIVERSITY AVENUE
CHICAGO, ILLINOIS 60637
TEL: 773-936-3700
FAX: 773-936-3701
WWW: WWW.CHEM.UCHICAGO.EDU



8203 64K DYNAMIC RAM CONTROLLER

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

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

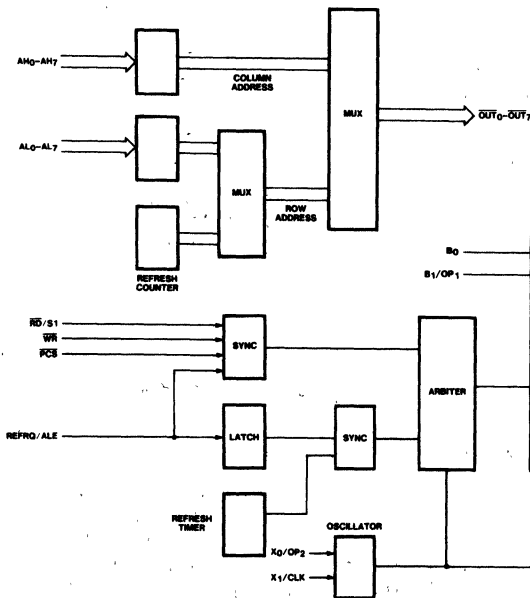


Figure 1. 8203 Block Diagram

210444-1

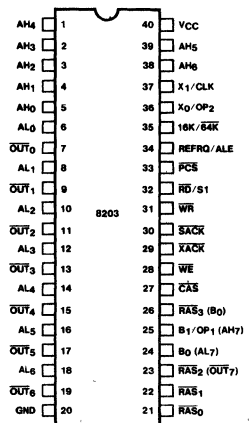


Figure 2.
Pin Configuration

210444-2

Table 1. Pin Descriptions

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

Table 1. Pin Descriptions (Continued)

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

FUNCTIONAL DESCRIPTION

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

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

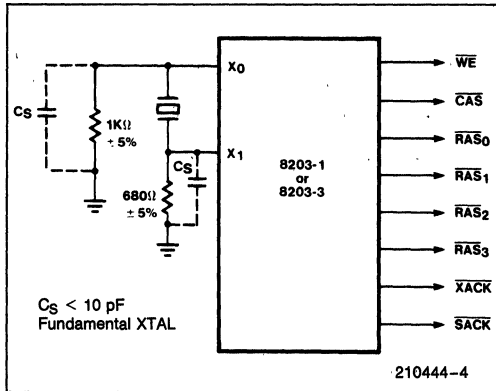


Figure 3. Crystal Operation for the 8203-1 and 8203-3

All 8203 timing is generated from a single reference clock. This clock is provided via an external oscillator or an on-chip crystal oscillator. All output signal

transitions are synchronous with respect to this clock reference, except for the trailing edges of the CPU handshake signals \overline{SACK} and \overline{XACK} .

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

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

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

16K/64 Option Selection

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

Other Option Selections

The 8203 has two strapping options. When OP₁ is selected (16K mode only), pin 32 changes from a RD input to an S1 input, and pin 34 changes from a REFRQ input to an ALE input. See "Refresh Cycles" and "Read Cycles" for more detail. OP₁ is selected by tying pin 25 to +12V through a 5.1 KΩ resistor on the 8203-1 or 8203-3 only.

When OP₂ is selected, the internal oscillator is disabled and pin 37 changes from a crystal input (X₁) to a CLK input for an external TTL clock. OP₂ is selected by shorting pin 36 (X₀/OP₂) directly to pin 40 (V_{CC}). No current limiting resistor should be used. OP₂ may also be selected by tying pin 36 to +12V through a 1 KΩ resistor.

Refresh Timer

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

Refresh Counter

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

Pin #	16K Function	64K Function
23	RAS ₂	Address Output (OUT ₇)
24	Bank Select (B ₀)	Address Input (AL ₇)
25	Bank Select (B ₁)	Address Input (AH ₇)
26	RAS ₃	Bank Select (B ₀)

Figure 4. 16K/64K Mode Selection

	Inputs		Outputs			
	B ₁	B ₀	RAS ₀	RAS ₁	RAS ₂	RAS ₃
16K Mode	0	0	0	1	1	1
	0	1	1	0	1	1
	1	0	1	1	0	1
	1	1	1	1	1	0
64K Mode	—	0	0	1	—	—
	—	1	1	0	—	—

Figure 5. Bank Selection

Description	Pin #	Normal Function	Option Function
B ₁ /OP ₁ (16K only)/AH ₇	25	Bank (RAS) Select	Advanced-Read Mode (8203-1, -3)
X ₀ /OP ₂	36	Crystal Oscillator (8203-1 and 8203-3)	External Oscillator

Figure 6. 8203 Option Selection

Address Multiplexer

The address multiplexer takes the address inputs and the refresh counter outputs, and gates them onto the address outputs at the appropriate time. The address outputs, in conjunction with the RAS and CAS outputs, determine the address used by the dynamic RAMs for read, write, and refresh cycles. During the first part of a read or write cycle, AL₀-AL₇ are gated to OUT₀-OUT₇, then AH₀-AH₇ are gated to the address outputs.

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

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

OUT₀-OUT₇ do not need inverters or buffers unless additional drive is required.

Synchronizer/Arbiter

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

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

System Operation

The 8203 is always in one of the following states:

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

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

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

Test Cycle

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

Refresh Cycles

The 8203 has two ways of providing dynamic RAM refresh:

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

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

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

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

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

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

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

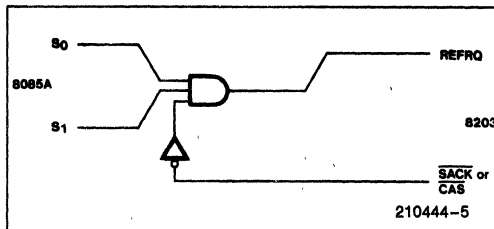


Figure 7. Hidden Refresh

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

Read Cycles

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

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

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

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

Figure 8. 8203 Read Options

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

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

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

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

Write Cycles

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

General System Considerations

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

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

The B₀-B₁ inputs are similar to the address inputs in that they are not latched. B₀ and B₁ should not be changed during a memory cycle, since they directly control which RAS output is activated.

The 8203 uses a two-stage synchronizer for the memory request inputs (RD, WR, ALE), and a separate two stage synchronizer for the external refresh input (REFRQ). As with any synchronizer, there is always a finite probability of metastable states inducing system errors. The 8203 synchronizer was

designed to have a system error rate less than 1 memory cycle every three years based on the full operating range of the 8203.

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

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

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

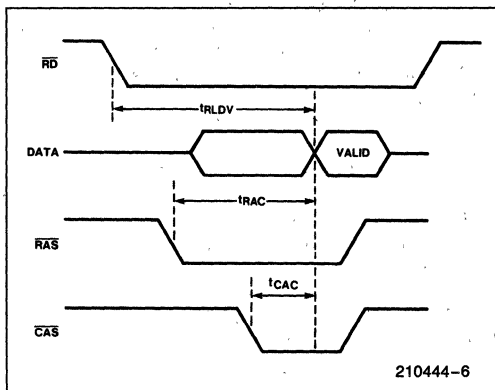


Figure 9. Read Access Time

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

The RAS₀₋₃, CAS, OUT₀₋₇, and WE outputs contain on-chip series damping resistors (typically 20Ω) to minimize overshoot.

Some dynamic RAMs require more than 2.4V V_{IH} . Noise immunity may be improved for these RAMs by

adding pull-up resistors to the 8203's outputs. Intel RAMs do not require pull-up resistors.

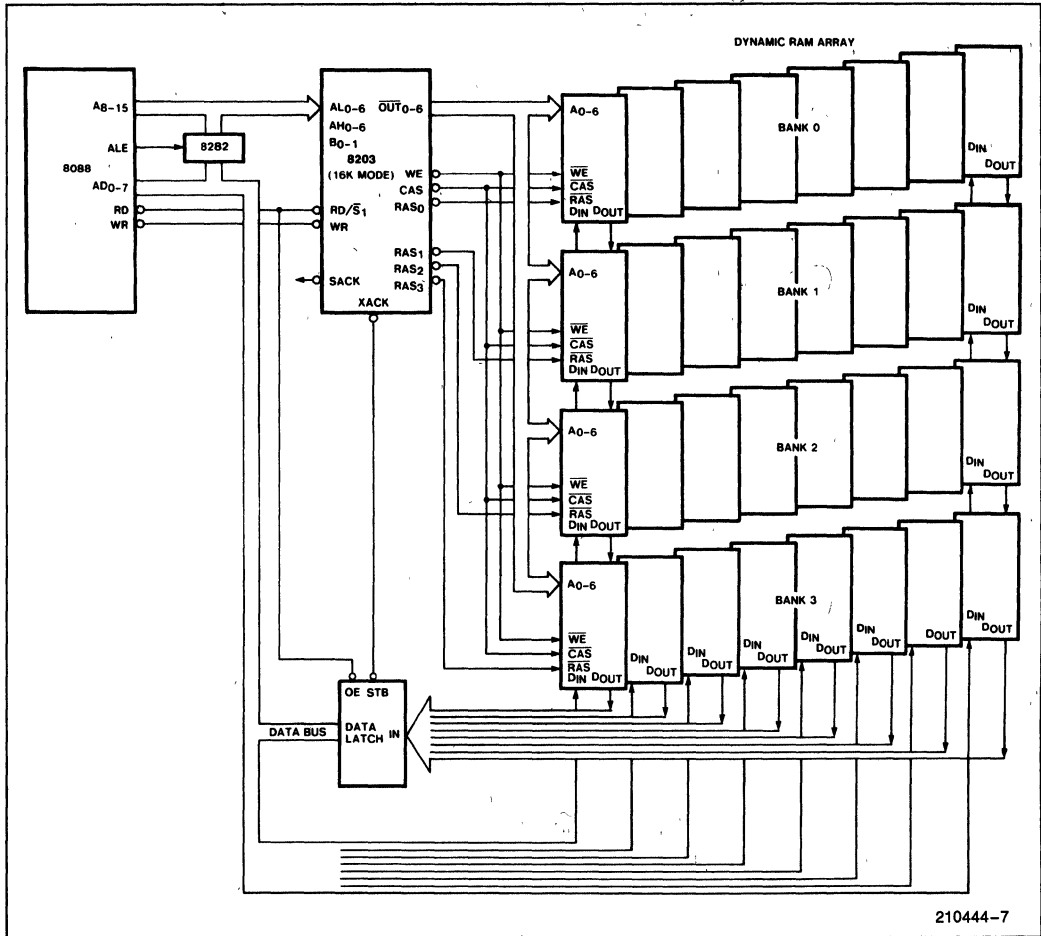
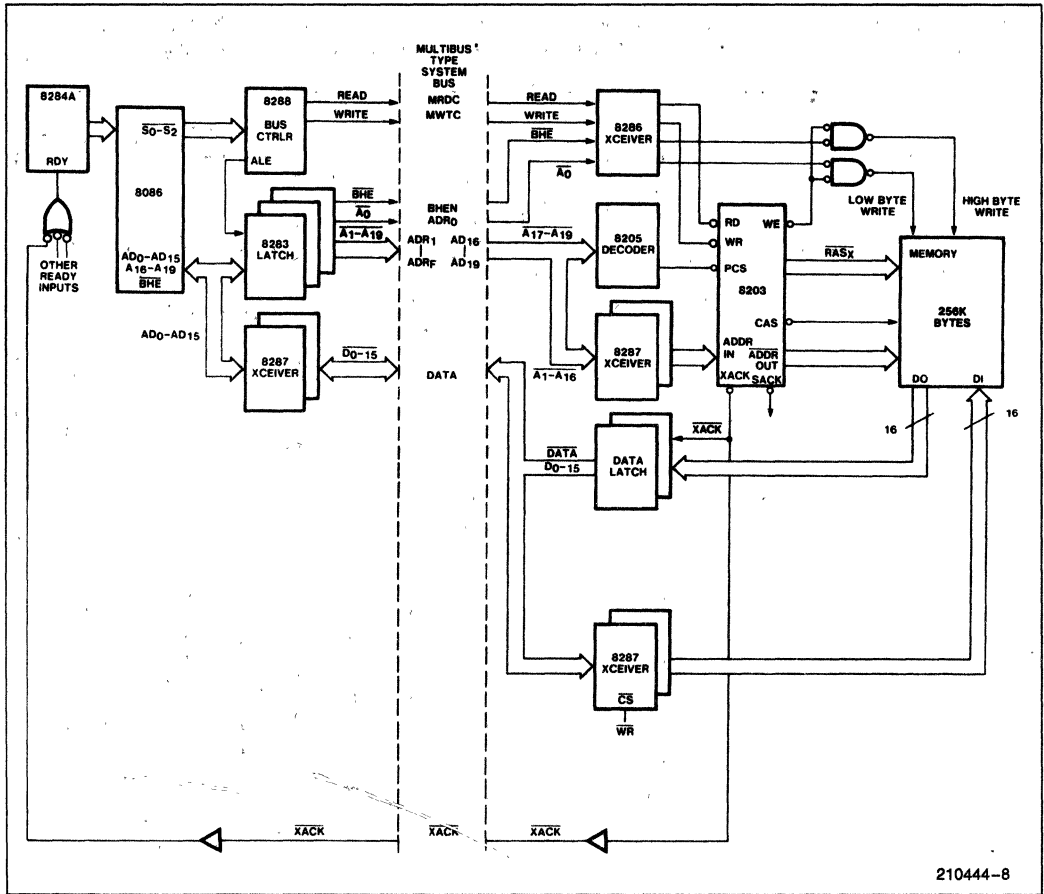


Figure 10. Typical 8088 System



210444-8

Figure 11. 8086/256K Byte System

ABSOLUTE MAXIMUM RATINGS*

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

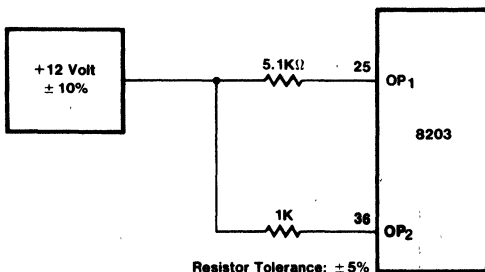
**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.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

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$; $V_{CC} = 5.0\text{V} \pm 10\%$ (5.0V $\pm 5\%$ for 8203-3); $GND = 0\text{V}$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{CC}	Input Clamp Voltage		-1.0	V	$I_C = -5 \text{ mA}$
I_{CC}	Power Supply Current		290	mA	
I_F	Forward Input Current CLK, 64K/16K Mode Select All Other Inputs ⁽³⁾		-2.0 -320	mA μA	$V_F = 0.45\text{V}$ $V_F = 0.45\text{V}$
I_R	Reverse Input Current ⁽³⁾		40	μA	$V_R = V_{CC}^{(1,5)}$
V_{OL}	Output Low Voltage SACK, XACK All Other Outputs		0.45 0.45	V V	$I_{OL} = 5 \text{ mA}$ $I_{OL} = 3 \text{ mA}$
V_{OH}	Output High Voltage SACK, XACK All Other Outputs	2.4 2.6		V V	$V_{IL} = 0.65\text{V}$ $I_{OH} = -1 \text{ mA}$ $I_{OH} = -1 \text{ mA}$
V_{IL}	Input Low Voltage		0.8	V	$V_{CC} = 5.0\text{V}^{(2)}$
V_{IH1}	Input High Voltage	2.0	V_{CC}	V	$V_{CC} = 5.0\text{V}$
V_{IH2}	Option Voltage		V_{CC}	V	(4)
C_{IN}	Input Capacitance		30	pF	$F = 1 \text{ MHz}$ $V_{BIAS} = 2.5\text{V}$, $V_{CC} = 5\text{V}$ $T_A = 25^\circ\text{C}^{(6)}$

NOTES:

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



210444-3

- $I_R = 150 \mu\text{A}$ for pin 35 (Mode Select 16K/64K).
- Sampled not 100% tested.

A.C. CHARACTERISTICS
 $T_J = 0^\circ\text{C to } 70^\circ\text{C}; V_{CC} = 5V \pm 10\% (5.0V \pm 5\% \text{ for } 8203\text{-}3); GND = 0V$

 Measurements made with respect to $\overline{RAS}_0\text{--}\overline{RAS}_3$, \overline{CAS} , \overline{WE} , $\overline{OUT}_0\text{--}\overline{OUT}_6$ are at 2.4V and 0.8V. All other pins are measured at 1.5V. All times are in ns.

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

A.C. CHARACTERISTICS (Continued)

$T_J = 0^{\circ}\text{C}$ to 70°C ; $V_{CC} = 5\text{V} \pm 10\%$ ($5.0\text{V} \pm 5\%$ for 8203-3); $GND = 0\text{V}$

Measurements made with respect to $\text{RAS}_0\text{-RAS}_3$, CAS , WE , $\text{OUT}_0\text{-OUT}_6$ are at 2.4V and 0.8V. All other pins are measured at 1.5V. All times are in ns.

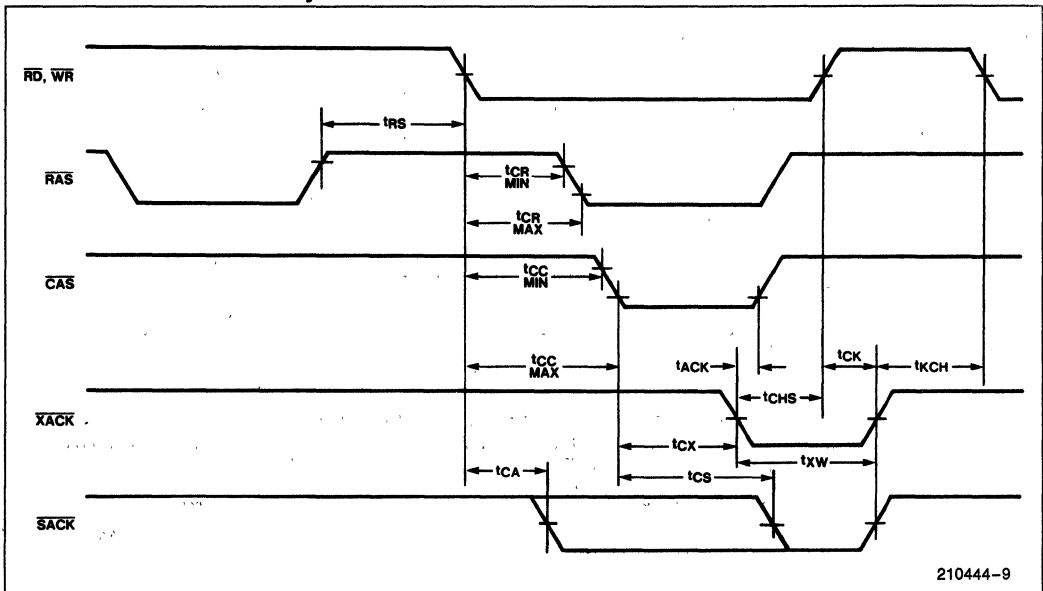
Symbol	Parameter	Min	Max	Notes
t_{LL}	REFRQ Pulse Width	20		
t_{CHS}	CMD Hold Time	30		11
t_{RFR}	REFRQ to $\overline{\text{RAS}}$ Delay		$4t_p + 100$	6
t_{WW}	$\overline{\text{WR}}$ to $\overline{\text{WE}}$ Delay	0	50	8
t_{AD}	CPU Address Delay	0	40	3

NOTES:

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

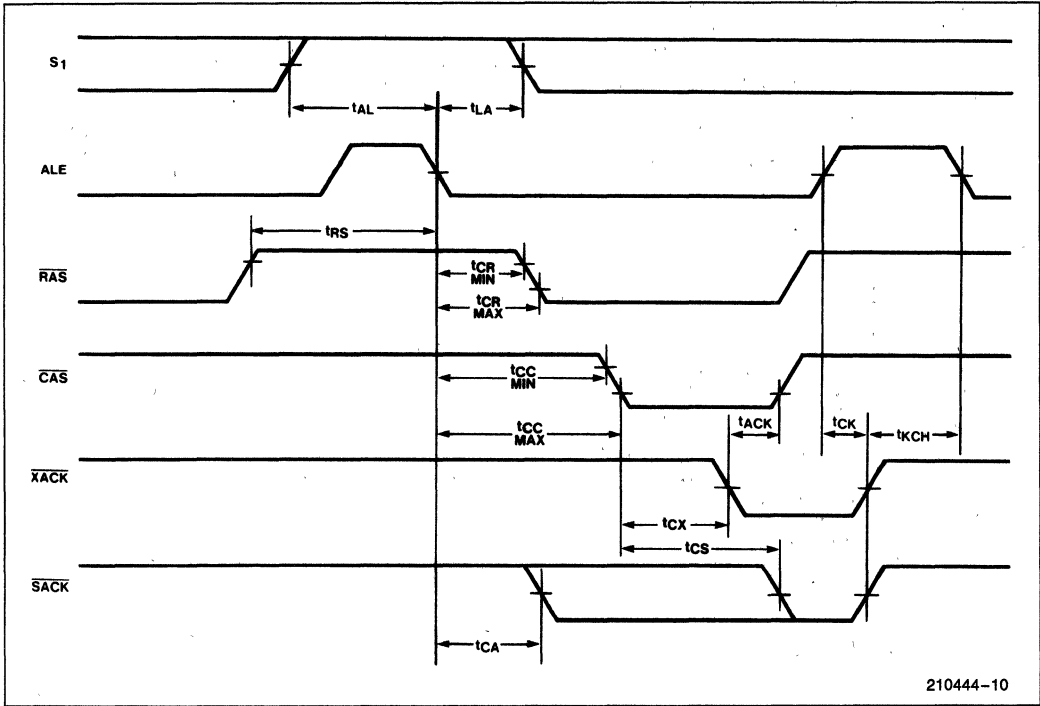
WAVEFORMS

Normal Read or Write Cycle

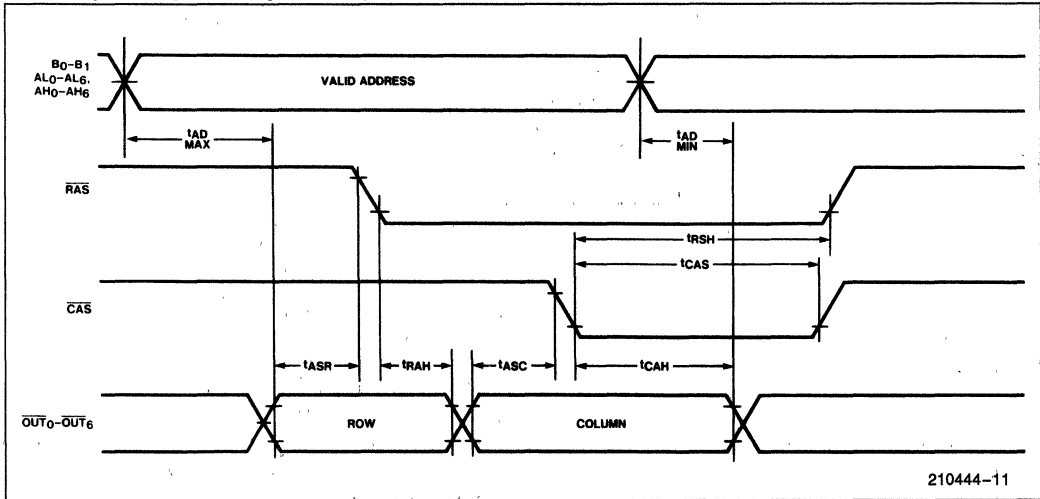


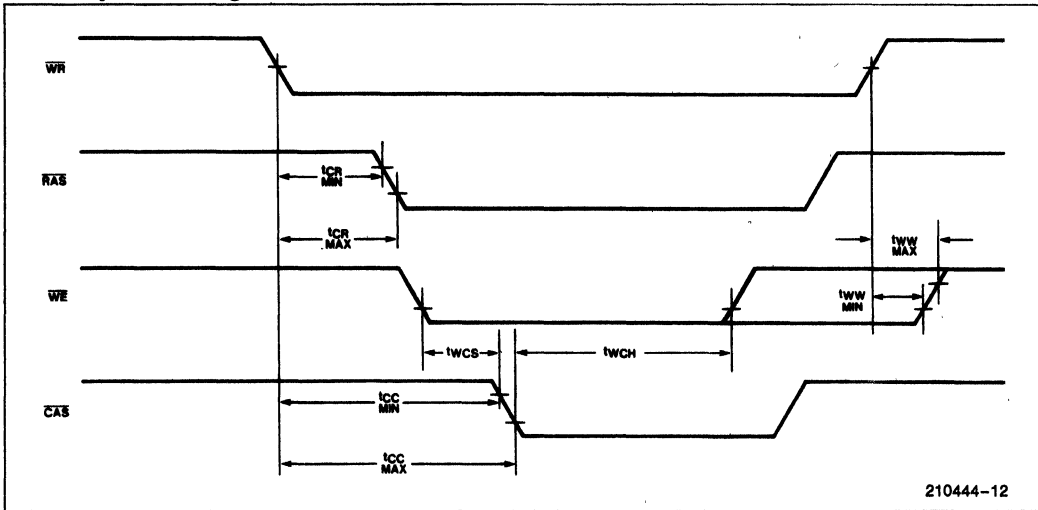
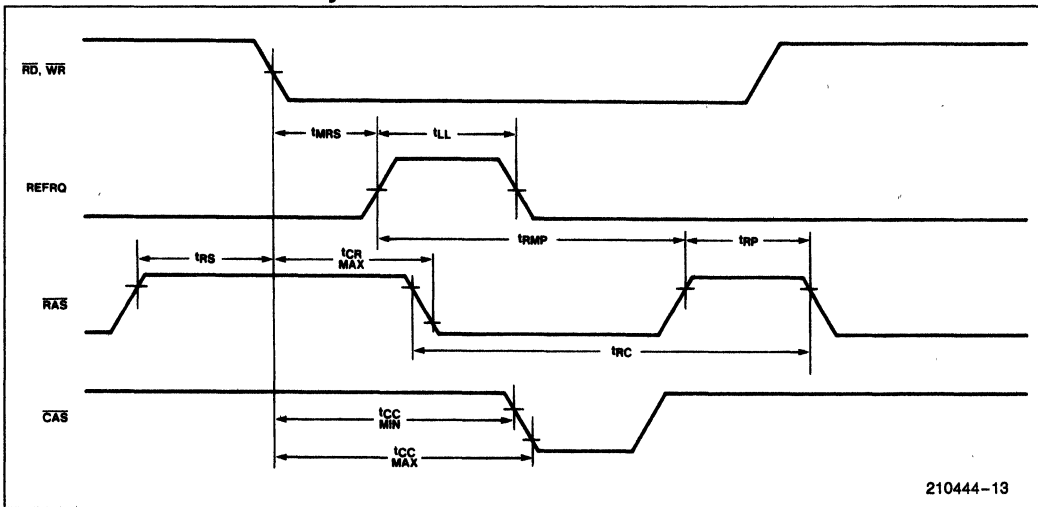
WAVEFORMS (Continued)

Advanced Read Mode



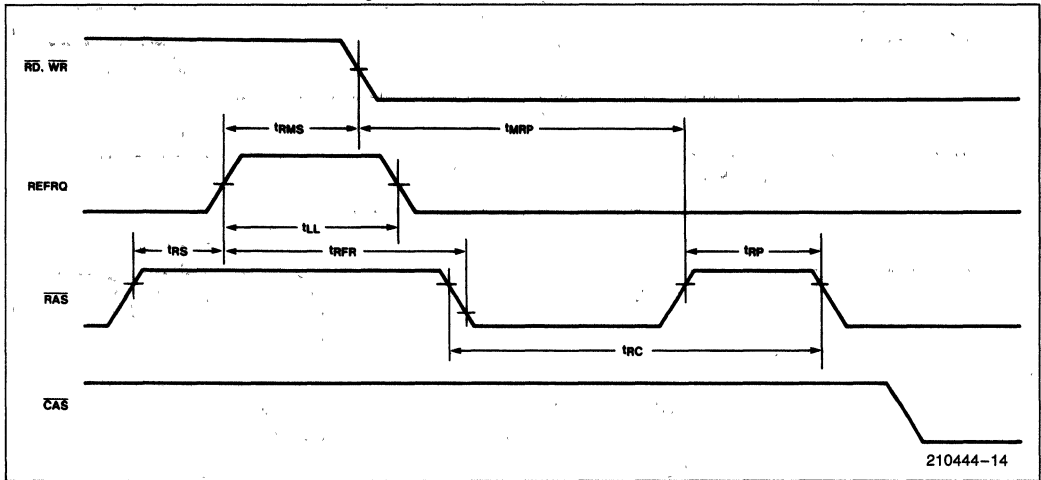
Memory Compatibility Timing



WAVEFORMS (Continued)**Write Cycle Timing****Read or Write Followed By External Refresh**

WAVEFORMS (Continued)

External Refresh Followed By Read or Write



Clock and System Timing

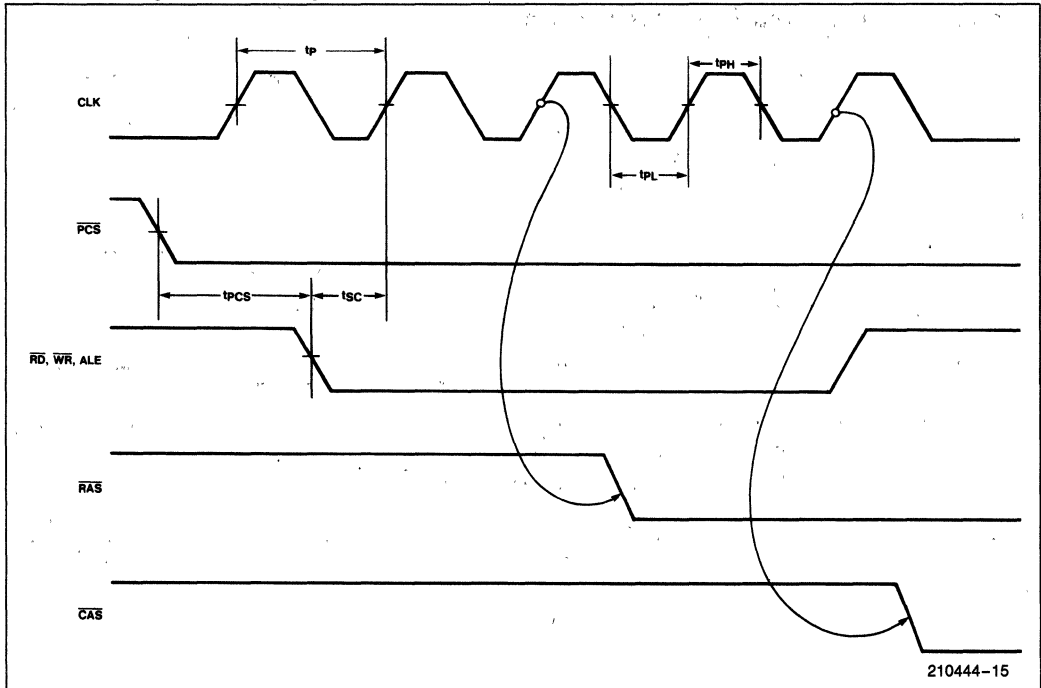
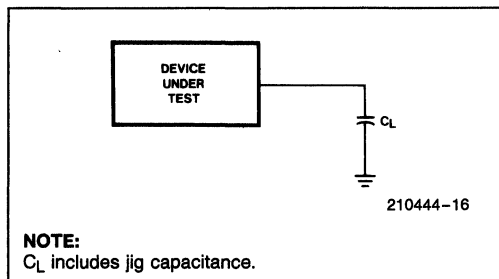


Table 2. 8203 Output Loading. All specifications are for the Test Load unless otherwise noted.

Pin	Test Load
\overline{SACK} , \overline{XACK}	$C_L = 30 \text{ pF}$
\overline{OUT}_0 – \overline{OUT}_6	$C_L = 160 \text{ pF}$
\overline{RAS}_0 – \overline{RAS}_3	$C_L = 60 \text{ pF}$
\overline{WE}	$C_L = 224 \text{ pF}$
\overline{CAS}	$C_L = 320 \text{ pF}$

A.C. TESTING LOAD CIRCUIT



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

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

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

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

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

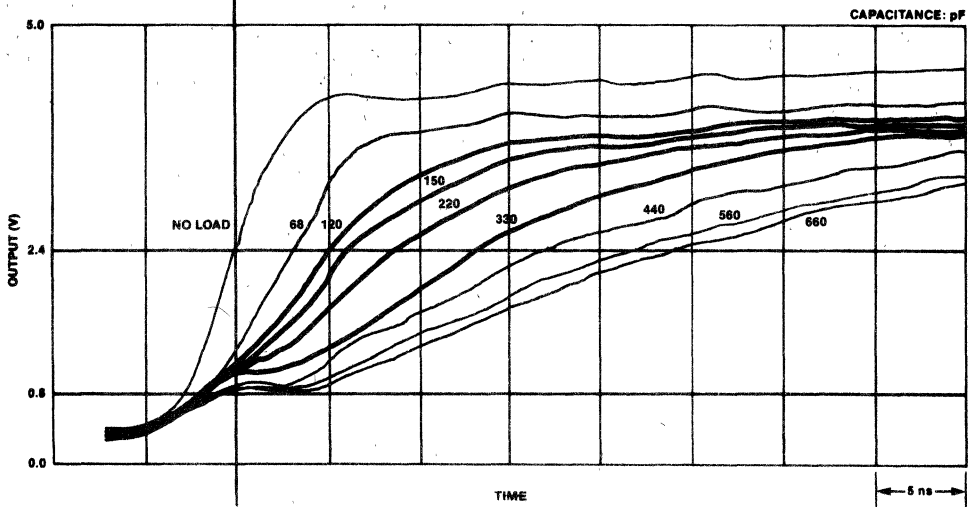
\overline{CAS} load = 160 pF + board capacitance.

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

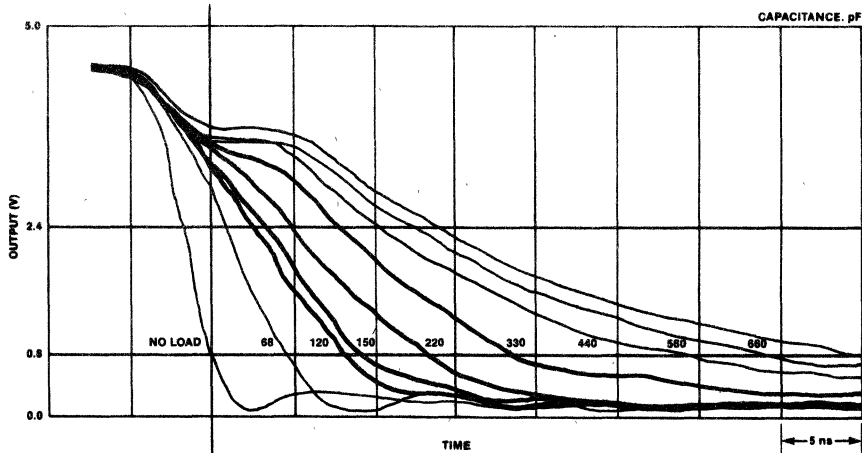
2) From the waveform diagrams, we determine that the falling edge timing is needed for t_{CR} and t_{CC} . Next find the curve that *best* approximates the test load; i.e., 68 pF for \overline{RAS} and 330 pF for \overline{CAS} .

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

A.C. CHARACTERISTICS FOR DIFFERENT CAPACITIVE LOADS



210444-17



210444-18

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

MEASUREMENT CONDITIONS

Pins not measured are loaded with the Test Load capacitance

$T_A = 25^\circ\text{C}$ $V_{CC} = +5\text{V}$ $t_p = 50 \text{ ns}$



8206 ERROR DETECTION AND CORRECTION UNIT

- Detects All Single Bit, and Double Bit and Most Multiple Bit Errors
- Corrects All Single Bit Errors
- | 3 Selections | 8206-1 | 8206 |
|--------------|--------|-------|
| Detection | 35 ns | 42 ns |
| Correction | 55 ns | 67 ns |
- Syndrome Outputs for Error Logging
- Automatic Error Scrubbing with 8207
- Expandable to Handle 80 Bit Memories
- Separate Input and Output Busses—No Timing Strokes Required
- Supports Read With and Without Correction, Writes, Partial (Byte) Writes, and Read-Modify-Writes
- HMOS III Technology for Low Power
- 68 Pin Leadless JEDEC Package
- 68 Pin Grid Array Package

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

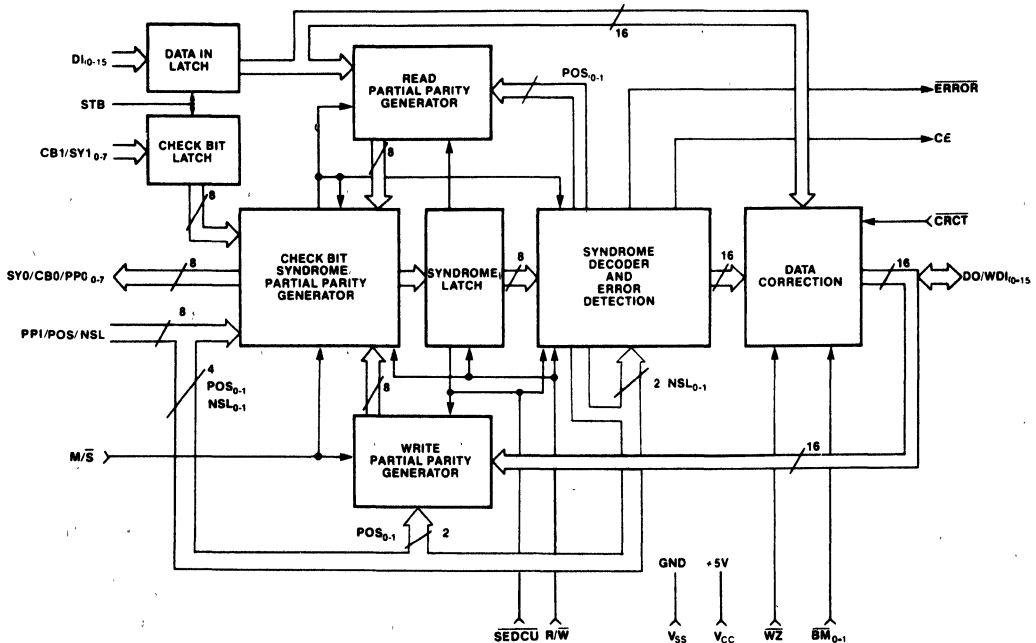


Figure 1. 8206 Block Diagram

205220-1

Table 1. 8206 Pin Description

Symbol	Pin No.	Type	Name and Function
D ₁₀₋₁₅	1, 68-61, 59-53	I	DATA IN: These inputs accept a 16 bit data word from RAM for error detection and/or correction.
CBI/SY ₀ CBI/SY ₁ CBI/SY ₂ CBI/SY ₃ CBI/SY ₄ CBI/SY ₅ CBI/SY ₆ CBI/SY ₇	5 6 7 8 9 10 11 12	I I I I I I I I	CHECK BITS IN/SYNDROME IN: In a single 8206 system, or in the master in a multi-8206 system, these inputs accept the check bits (5 to 8) from the RAM. In a single 8206 16 bit system, CBI ₀₋₅ are used. In slave 8206's these inputs accept the syndrome from the master.
DO/WDI ₀ DO/WDI ₁ DO/WDI ₂ DO/WDI ₃ DO/WDI ₄ DO/WDI ₅ DO/WDI ₆ DO/WDI ₇ DO/WDI ₈ DO/WDI ₉ DO/WDI ₁₀ DO/WDI ₁₁ DO/WDI ₁₂ DO/WDI ₁₃ DO/WDI ₁₄ DO/WDI ₁₅	51 50 49 48 47 46 45 44 42 41 40 39 38 37 36 35	I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O	DATA OUT/WRITE DATA IN: In a read cycle, data accepted by D ₁₀₋₁₅ appears at these outputs corrected if CRCT is low, or uncorrected if CRCT is high. The BM inputs must be high to enable the output buffers during the read cycle. In a write cycle, data to be written into the RAM is accepted by these inputs for computing the write check bits. In a partial-write cycle, the byte not to be modified appears at either DO ₀₋₇ if BM ₀ is high, or DO ₈₋₁₅ if BM ₁ is high, for writing to the RAM. When WZ is active, it causes the 8206 to output all zeros at DO ₀₋₁₅ , with the proper write check bits on CBO.
SYO/CBO/PPO ₀ SYO/CBO/PPO ₁ SYO/CBO/PPO ₂ SYO/CBO/PPO ₃ SYO/CBO/PPO ₄ SYO/CBO/PPO ₅ SYO/CBO/PPO ₆ SYO/CBO/PPO ₇	23 24 25 27 28 29 30 31	O O O O O O O O	SYNDROME OUT/CHECK BITS OUT/PARTIAL PARITY OUT: In a single 8206 system, or in the master in a multi-8206 system, the syndrome appears at these outputs during a read. During a write, the write check bits appear. In slave 8206's the partial parity bits used by the master appear at these outputs. The syndrome is latched (during read-modify-writes) by R/W going low.
PPI ₀ /POS ₀ PPI ₁ /POS ₁	13 14	I I	PARTIAL PARITY IN/POSITION: In the master in a multi-8206 system, these inputs accept partial parity bits 0 and 1 from the slaves. In a slave 8206 these inputs inform it of its position within the system (1 to 4). Not used in a single 8206 system.
PPI ₂ /NSL ₀ PPI ₃ /NSL ₁	15 16	I I	PARTIAL PARITY IN/NUMBER OF SLAVES: In the master in a multi-8206 system, these inputs accept partial parity bits 2 and 3 from the slaves. In a multi-8206 system these inputs are used in slave number 1 to tell it the total number of slaves in the system (1 to 4). Not used in other slaves or in a single 8206 system.
PPI ₄ CE	17	I/O	PARTIAL PARITY IN/CORRECTABLE ERROR: In the master in a multi-8206 system this pin accepts partial parity bit 4. In slave number 1 only, or in a single 8206 system, this pin outputs the correctable error flag. CE is latched by R/W going low. Not used in other slaves.

Table 1. 8206 Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
PPI ₅ PPI ₆ PPI ₇	18 19 20	 	PARTIAL PARITY IN: In the master in a multi-8206 system these pins accept partial parity bits 5 to 7. The number of partial parity bits equals the number of check bits. Not used in single 8206 systems or in slaves.
ERROR	22	O	ERROR: This pin outputs the error flag in a single 8206 system or in the master of a multi-8206 system. It is latched by R/W going low. Not used in slaves.
CRCT	52		CORRECT: When low this pin causes data correction during a read or read-modify-write cycle. When high, it causes error correction to be disabled, although error checking is still enabled.
STB	2		STROBE: STB is an input control used to strobe data at the DI inputs and check-bits at the CBI/SYI inputs. The signal is active high to admit the inputs. The signals are latched by the high-to-low transition of STB.
BM ₀ BM ₁	33 32	 	BYTE MARKS: When high, the Data Out pins are enabled for a read cycle. When low, the Data Out buffers are tristated for a write cycle. BM ₀ controls DO ₀₋₇ , while BM ₁ controls DO ₈₋₁₅ . In partial (byte) writes, the byte mark input is low for the new byte to be written.
R/W	21		READ/WRITE: When high this pin causes the 8206 to perform detection and correction (if CRCT is low). When low, it causes the 8206 to generate check-bits. On the high-to-low transition the syndrome is latched internally for read-modify-write cycles.
WZ	34		WRITE ZERO: When low this input overrides the BM ₀₋₁ and R/W inputs to cause the 8206 to output all zeros at DO ₀₋₁₅ with the corresponding check-bits at CBO ₀₋₇ . Used for memory initialization.
M/S	4		MASTER/SLAVE: Input tells the 8206 whether it is a master (high) or a slave (low).
SEDCU	3		SINGLE EDC UNIT: Input tells the master whether it is operating as a single 8206 (low) or as the master in a multi-8206 system (high). Not used in slaves.
V _{CC}	60		POWER SUPPLY: +5V
V _{SS}	26		LOGIC GROUND
V _{SS}	43		OUTPUT DRIVER GROUND

FUNCTIONAL DESCRIPTION

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

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

A single 8206 handles 8 or 16 bits of data, and up to 5 8206's can be cascaded in order to handle data paths of 80 bits. For a single 8206 8 bit system, the DI_{8-15} , DO/WDI_{8-15} and BM_1 inputs are grounded. See the Multi-Chip systems section for information on 24-80 bit systems.

The 8206 has a "flow through" architecture. It supports two kinds of error correction architecture: 1) Flow-through, or correct-always; and 2) Parallel, or check-only. These are two separate 16-pin busses,

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

Figure 3. Number of Check Bits Used by 8206

one to accept data from the RAM (DI) and the other to deliver corrected data to the system bus (DO/WDI). The logic is entirely combinatorial during a read cycle. This is in contrast to an architecture with only one bus, with bidirectional bus drivers that must first read the data and then be turned around to output the corrected data. The latter architecture typically requires additional hardware (latches and/or transceivers) and may be slower in a system due to timing skews of control signals.

READ CYCLE

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

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

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

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

WRITE CYCLE

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

In a partial write, part of the data word is overwritten, and part is retained in memory. This is accomplished by performing a read-modify-write cycle. The complete old word is read into the 8206 and corrected, with the syndrome internally latched by R/\bar{W} going low. Only that part of the word not to be modified is output onto the DO pins, as controlled by the Byte Mark inputs. That portion of the word to be overwrit-

ten is supplied by the system bus. The 8206 then calculates check bits for the new word, using the byte from the previous read and the new byte from the system bus, and writes them to the memory.

READ-MODIFY-WRITE CYCLES

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

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

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

INITIALIZATION

A memory system operating with ECC requires some form of initialization at system power-up in or-

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

MULTI-CHIP SYSTEMS

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

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

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

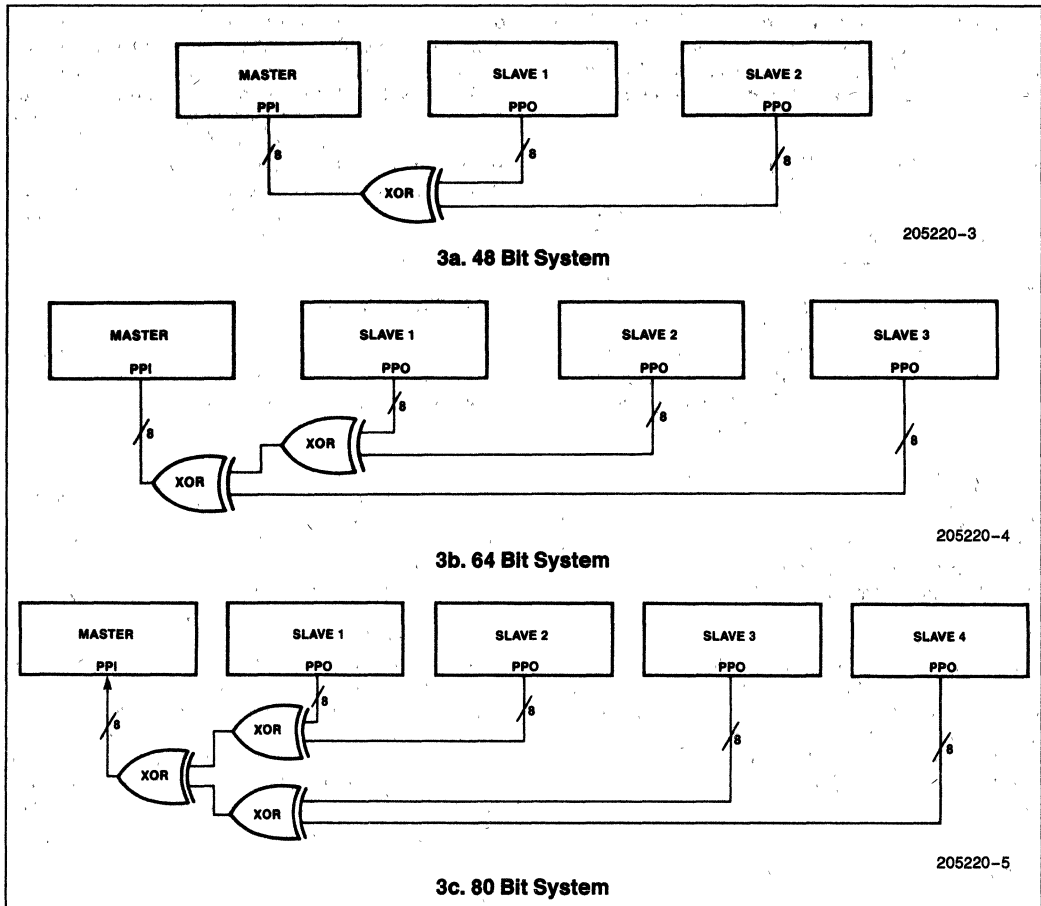


Figure 4. External Logic for Multi-Chip Systems

Table 3. Master/Slave Pin Assignments

Pin No.	Pin Name	Master	Slave 1	Slave 2	Slave 3	Slave 4
4	M/ \bar{S}	+5V	gnd	gnd	gnd	gnd
3	\overline{SEDCU}	+5V	+5V	+5V	+5V	+5V
13	PPI ₀ /POS ₀	PPI	gnd	+5V	gnd	+5V
14	PPI ₁ /POS ₁	PPI	gnd	gnd	+5V	+5V
15	PPI ₂ /NSL ₀	PPI	*	+5V	+5V	+5V
16	PPI ₃ /NSL ₁	PPI	*	+5V	+5V	+5V

NOTE:

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

Table 4. NSL Pin Assignments for Slave 1

Pin	Number of Slaves			
	1	2	3	4
PPI ₂ /NSL ₀	GND	+5V	GND	+5V
PPI ₃ /NSL ₁	GND	GND	+5V	+5V

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

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

$$TDVSV + TPVSV + TSVQV + nt_{XOR}$$

Data-in to error flag (read cycle) =

$$TDVSV + TPVEV + nt_{XOR}$$

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

$$TDVSV + TPVSV + TSVCV + nt_{XOR}$$

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

$$TQVQV + TPVSV + nt_{XOR}$$

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

$$TDVSV + TPVSV + TSVQV + TQVQV + TPVSV + 2nt_{XOR}$$

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

$$TDVQV + TQVQV + TPVSV + nt_{XOR}$$

HAMMING CODE

The 8206 uses a modified Hamming code which was optimized for multi-chip EDCU systems. The code is

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

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

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

Table 5. Modified Hamming Code Check Bit Generation

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

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

1-24

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

Table 6. 8206 Syndrome Decoding

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

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

SYSTEM ENVIRONMENT

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

The 8206 is designed for direct connection to the Intel 8207 Advanced Dynamic RAM Controller.

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

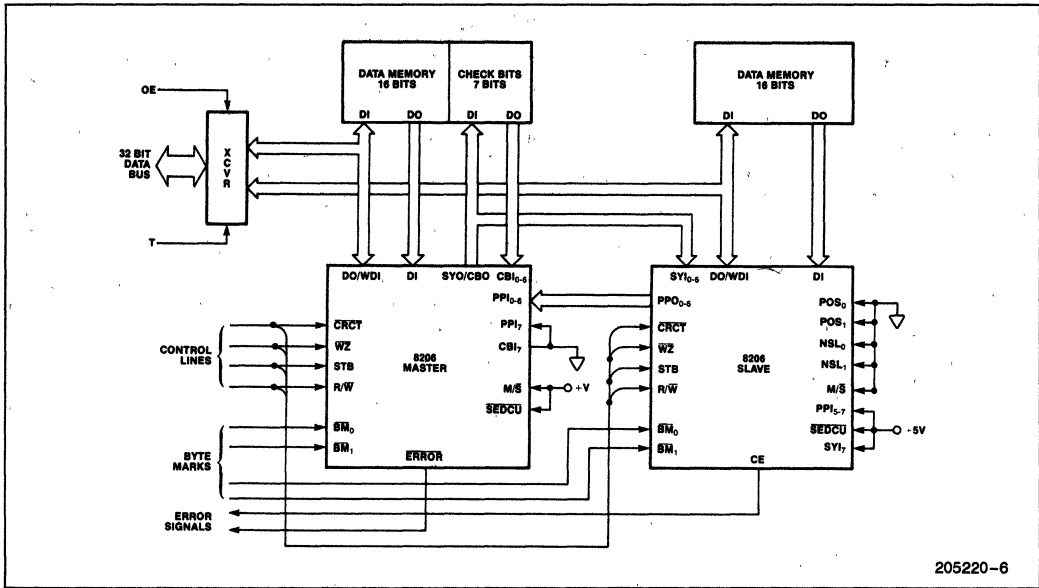


Figure 5. 32-Bit 8206 System Interface

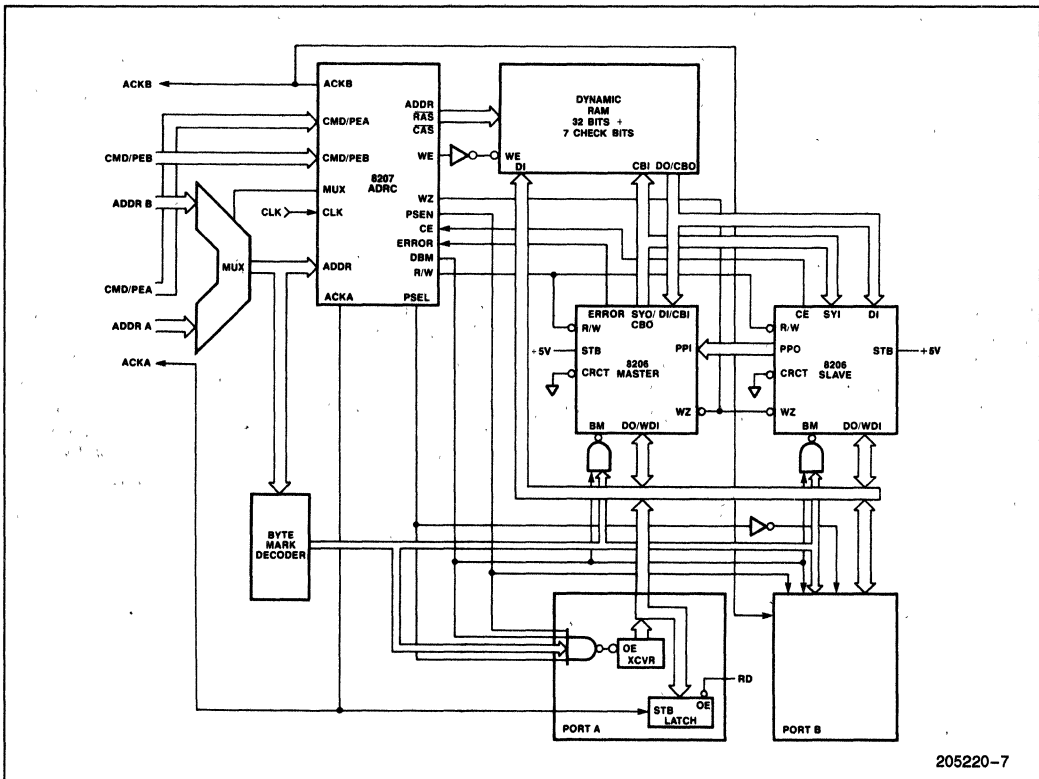


Figure 6. Dual Port RAM Subsystem with 8206/8207 (32-bit bus)

MEMORY BOARD TESTING

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

Mode 0—Read and write with error correction.

Implementation: This mode is the normal 8206 operating mode.

Mode 1—Read and write data with error correction disabled to allow test of data memory.

Implementation: This mode is performed with \overline{CRCT} deactivated.

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

Implementation: Any pattern may be written into the check bits memory by judiciously choosing the proper data word to

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

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

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

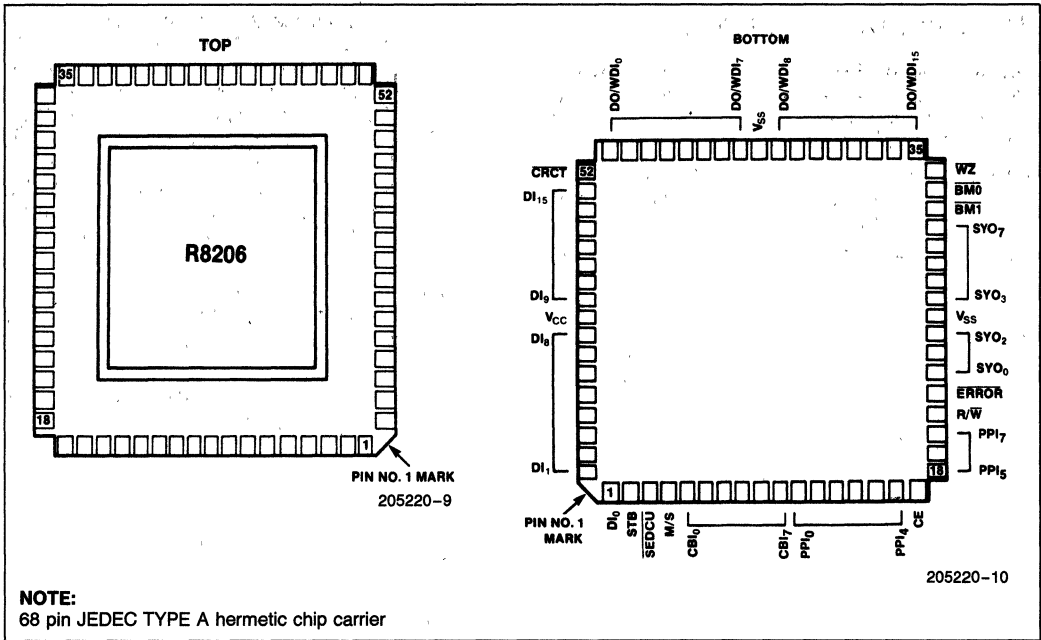


Figure 8a. 8206 Leadless Chip Carrier (LCC) Pinout Diagram

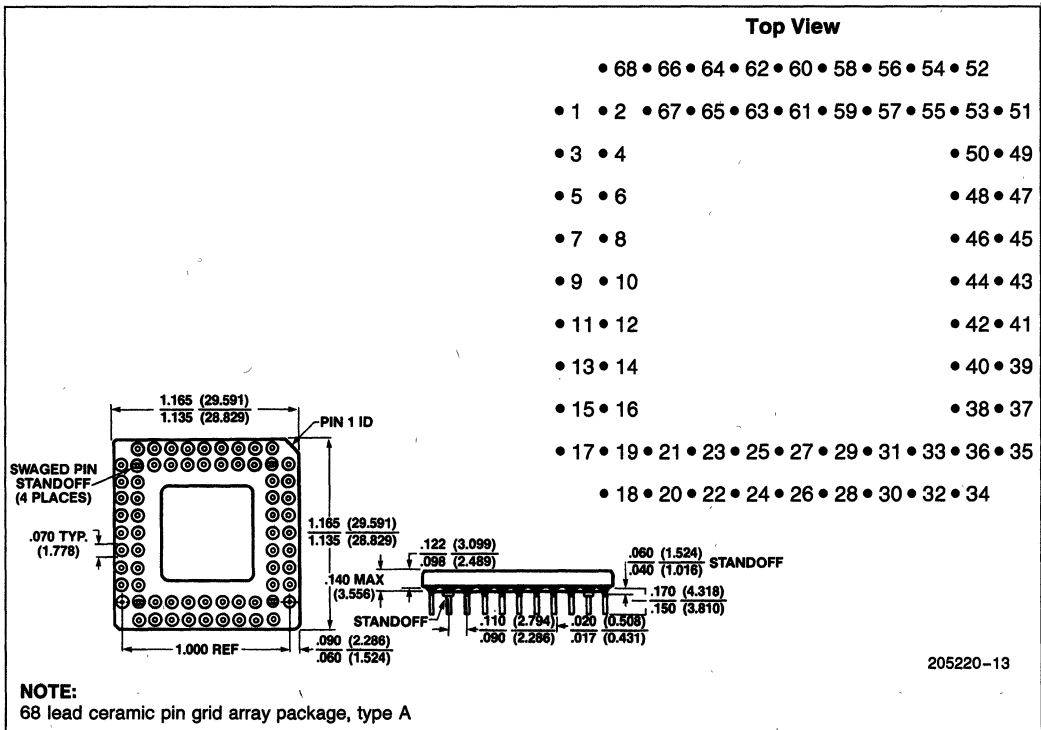


Figure 8b. 8206 Pin Grid Array (PGA) Package and Pinout Diagram

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage On Any Pin
 with Respect to Ground..... -0.5V to +7V
 Power Dissipation..... 1.5W

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

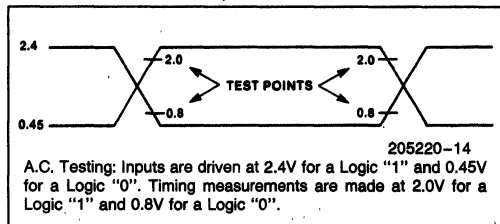
D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = 5.0\text{V} \pm 10\%$, $V_{SS} = \text{GND}$

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

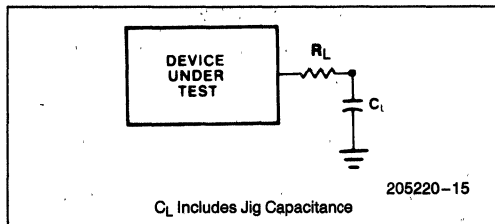
NOTES:

1. SEDCU (pin 3) and M/S (pin 4) are device strapping options and should be tied to V_{CC} or GND. $V_{IH\ min} = V_{CC} - 0.5\text{V}$ and $V_{IL\ max} = 0.5\text{V}$.
2. PPI₀₋₇ (pins 13-20) and CBI₆₋₇ (pins 11, 12) have internal pull-up resistors and if left unconnected will be pulled to V_{CC} .

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



A.C. CHARACTERISTICS
 $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 10\%$, $V_{SS} = 0\text{V}$, $R_L = 22\Omega$, $C_L = 50\text{ pF}$; all times are in ns

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

NOTES:

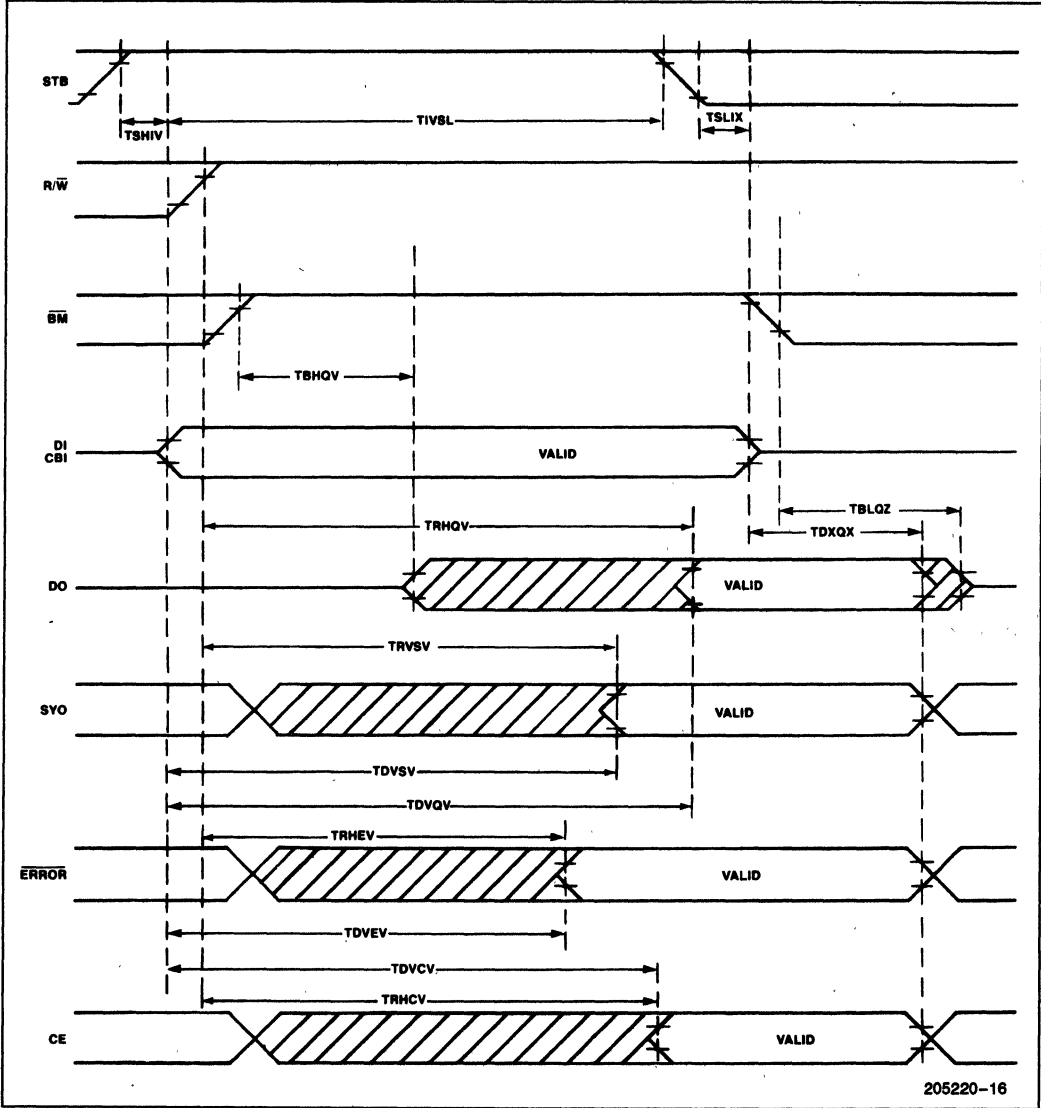
1. A.C. Test Levels for CBO and DO are 2.4V and 0.8V.

 2. T_{SHIV} is required to guarantee output delay timings: T_{DVEV} , T_{DVCV} , T_{DVQV} , T_{DVS} , $T_{SHIV} + T_{IVSL}$ guarantees a min STB pulse width of 35 ns.

3. Not required for 8/16 bit systems.

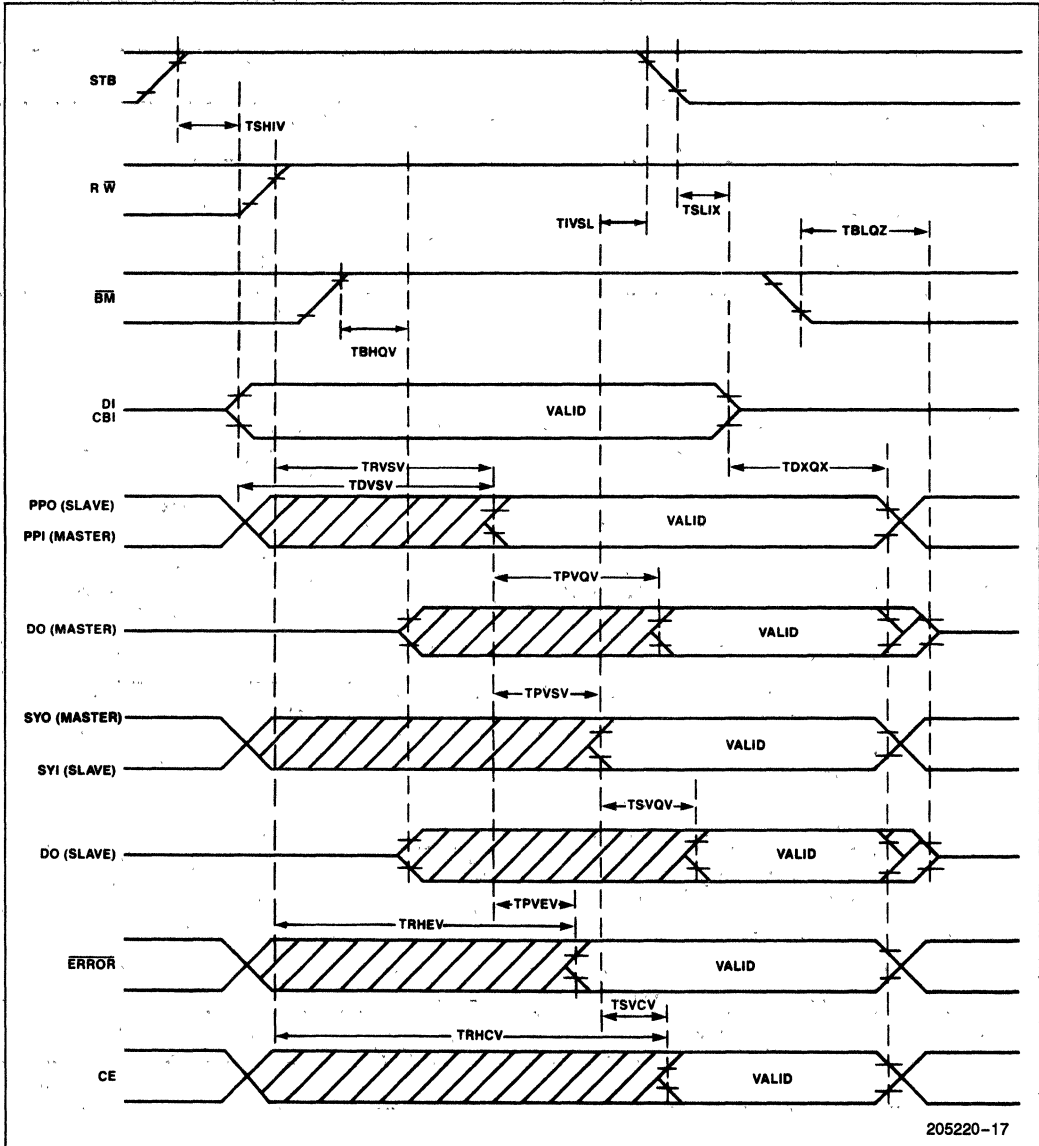
WAVEFORMS

READ



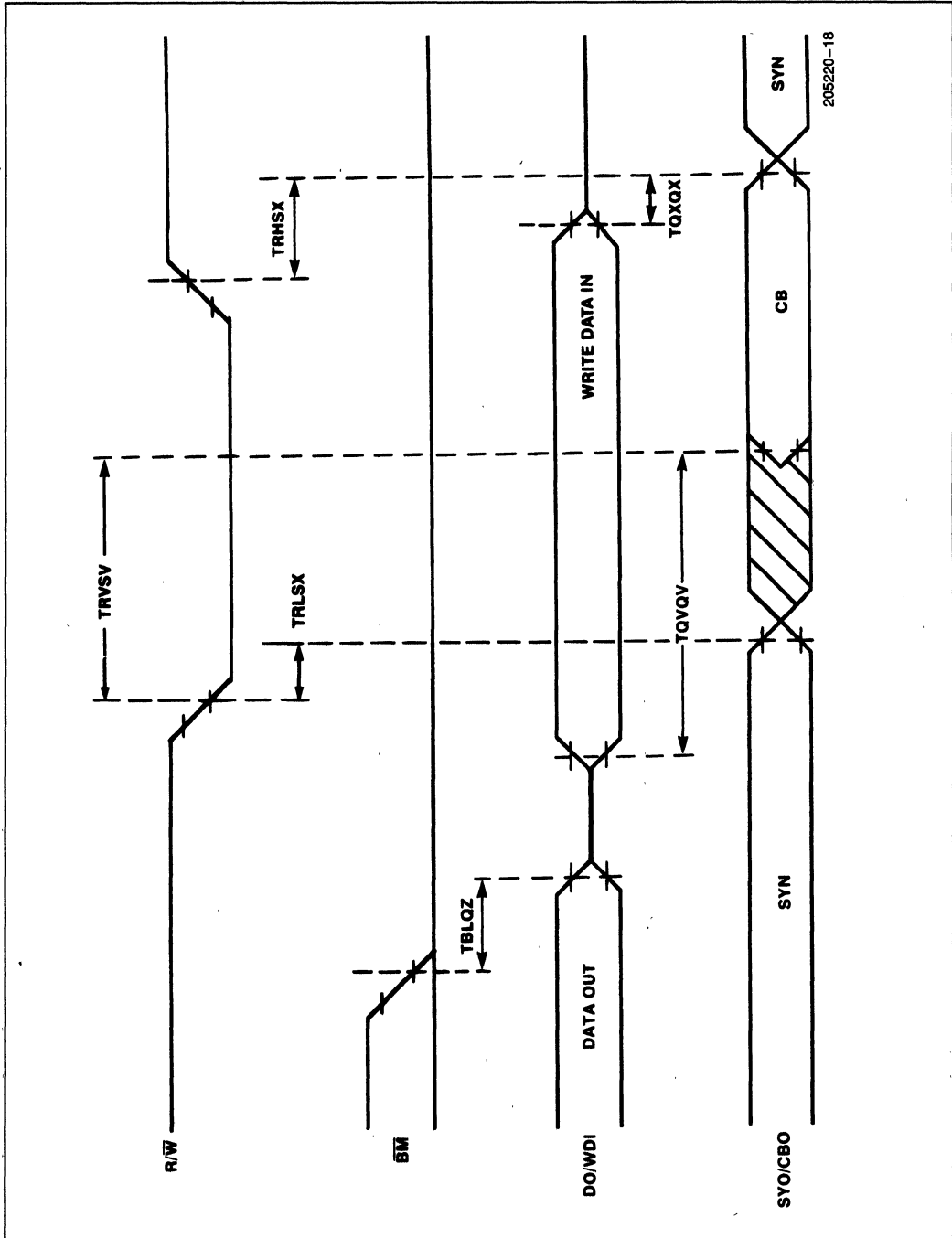
WAVEFORMS (Continued)

READ—MASTER/SLAVE



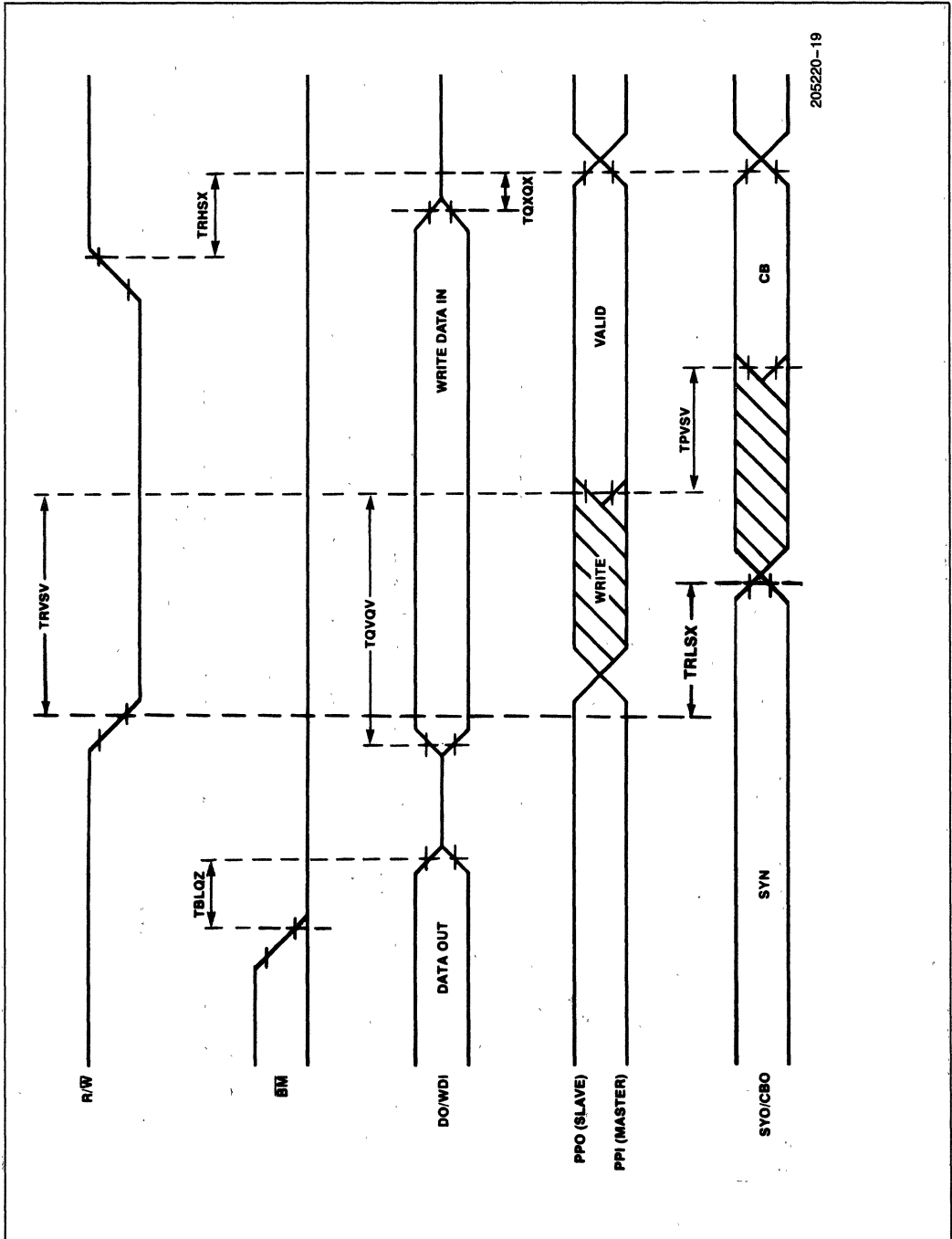
WAVEFORMS (Continued)

FULL WRITE



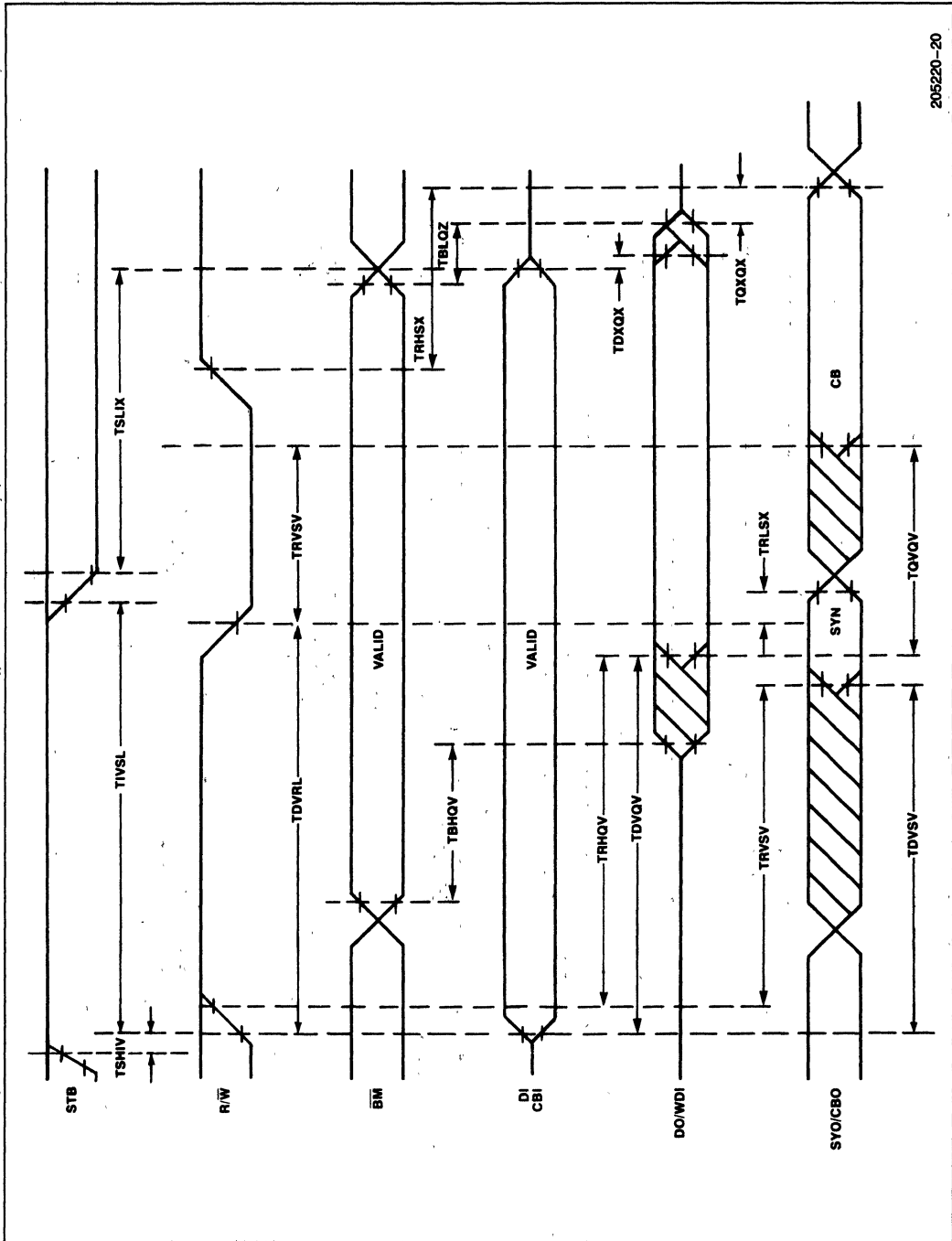
WAVEFORMS (Continued)

FULL WRITE—MASTER/SLAVE



WAVEFORMS (Continued)

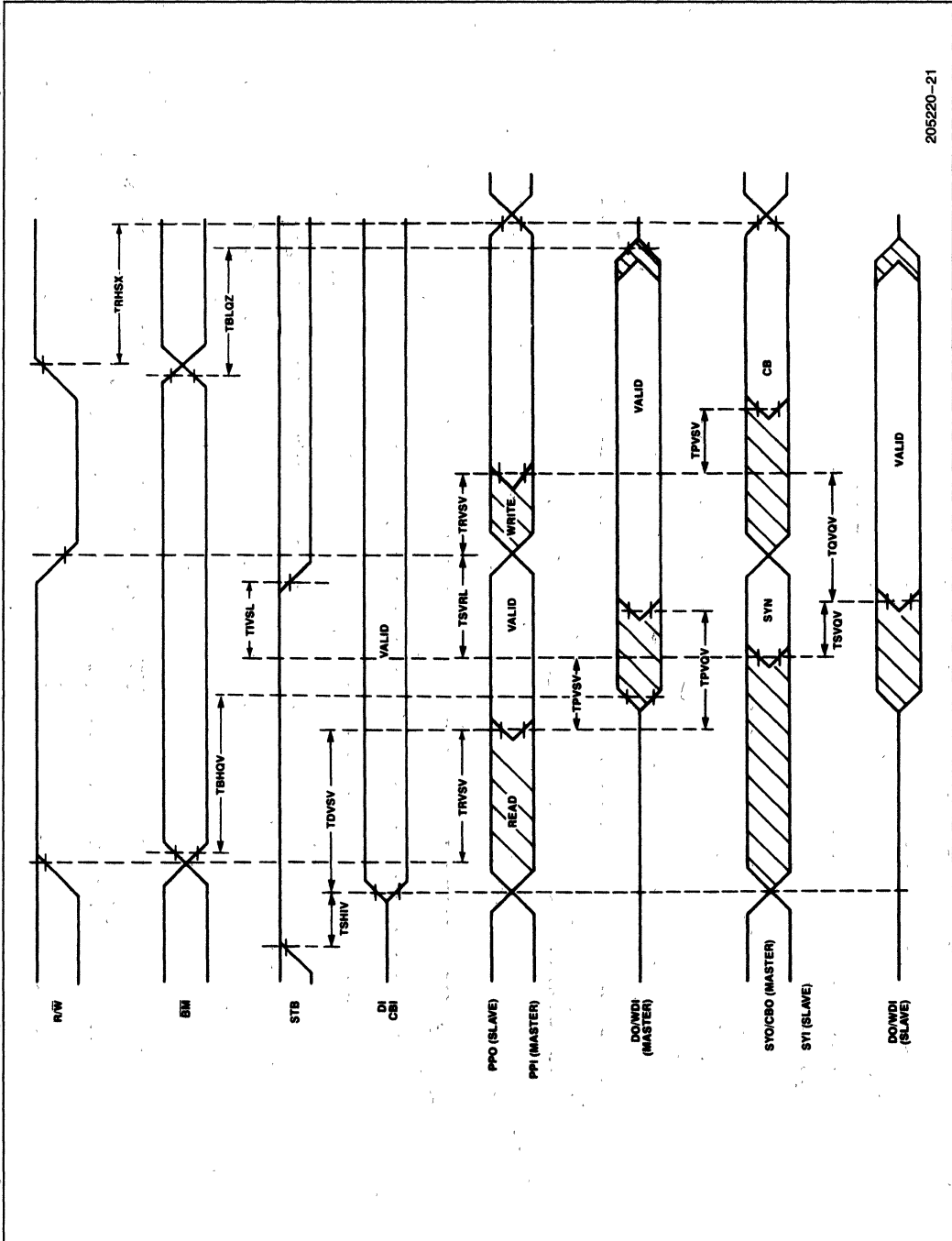
READ MODIFY WRITE



205220-20

WAVEFORMS (Continued)

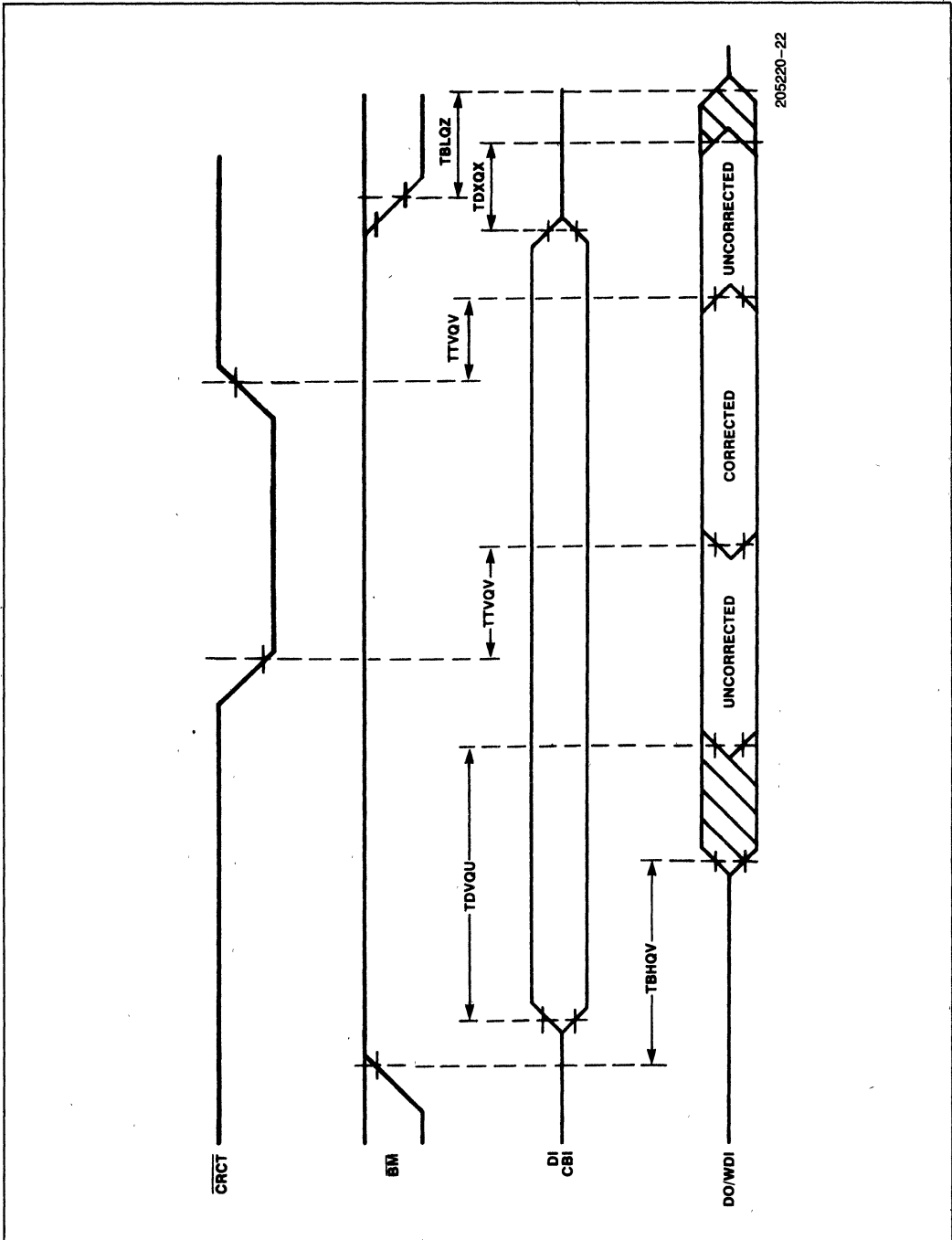
READ MODIFY WRITE—MASTER/SLAVE



205220-21

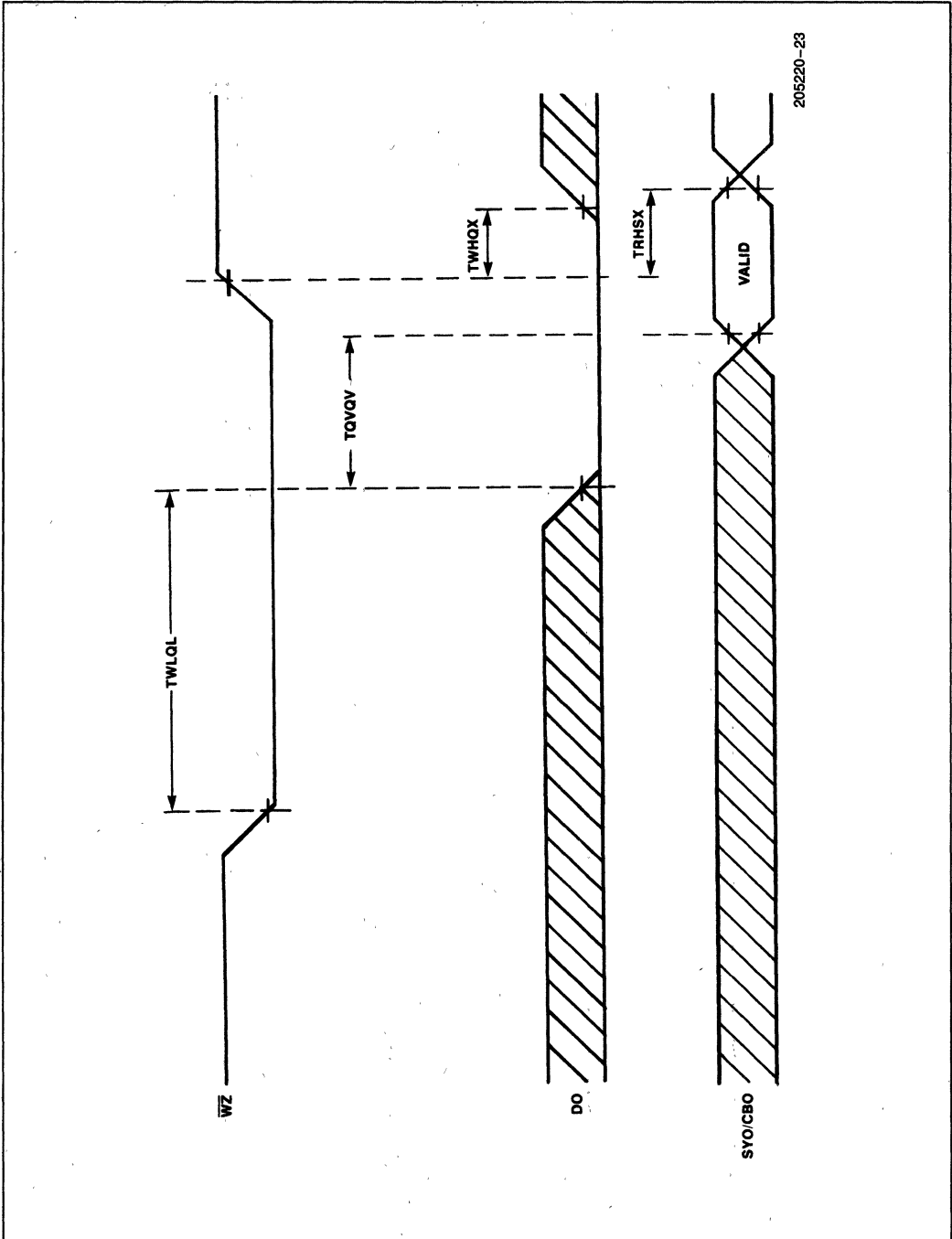
WAVEFORMS (Continued)

NON-CORRECTING READ



WAVEFORMS (Continued)

WRITE ZERO

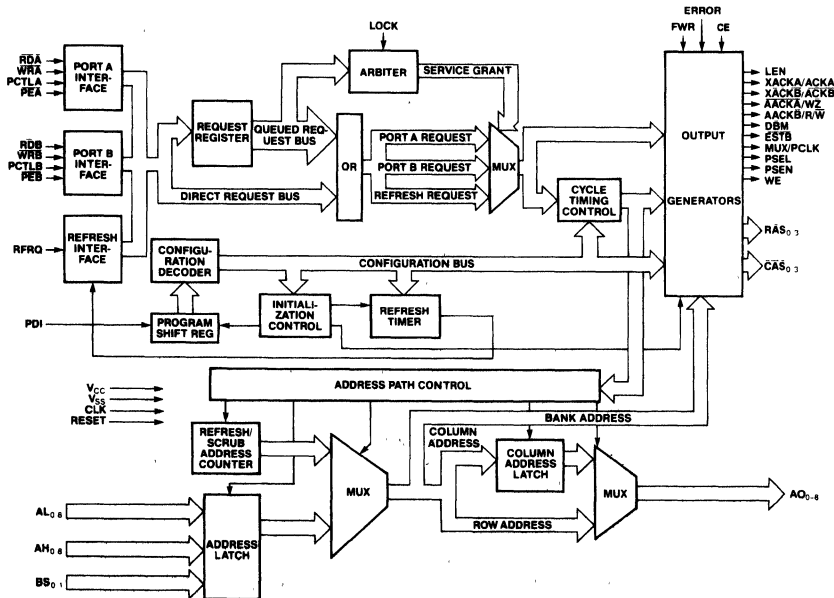




8207 DUAL-PORT DYNAMIC RAM CONTROLLER

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

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



210463-1

Figure 1. 8207 Block Diagram

Table 1. Pin Description

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

Table 1. Pin Description (Continued)

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

Table 1. Pin Description (Continued)

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

GENERAL DESCRIPTION

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

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

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

FUNCTIONAL DESCRIPTION

Processor Interface

The 8207 has control circuitry for two ports each capable of supporting one of several possible bus

structures. The ports are independently configurable allowing the dynamic RAM to serve as an interface between two different bus structures.

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

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

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

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

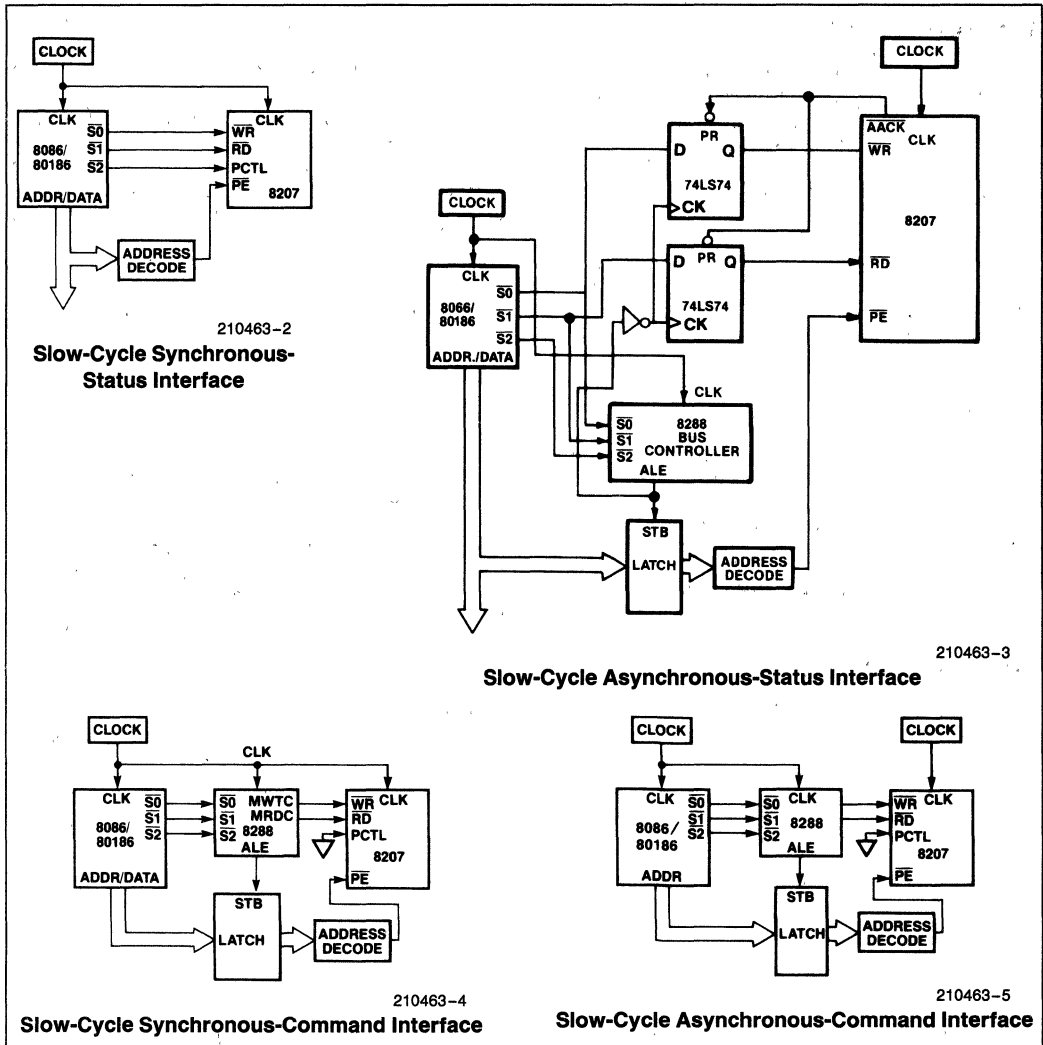


Figure 2A. Slow-Cycle (CFS = 0) Port Interfaces Supported by the 8207

Single-Port Operation

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

Dual-Port Operation

The 8207 provides for two-port operation. Two independent processors may access memory controlled

by the 8207. The 8207 arbitrates between each of the processor requests and directs data to or from the appropriate port. Selection is done on a priority concept that reassigns priorities based upon past history. Processor requests are internally queued.

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

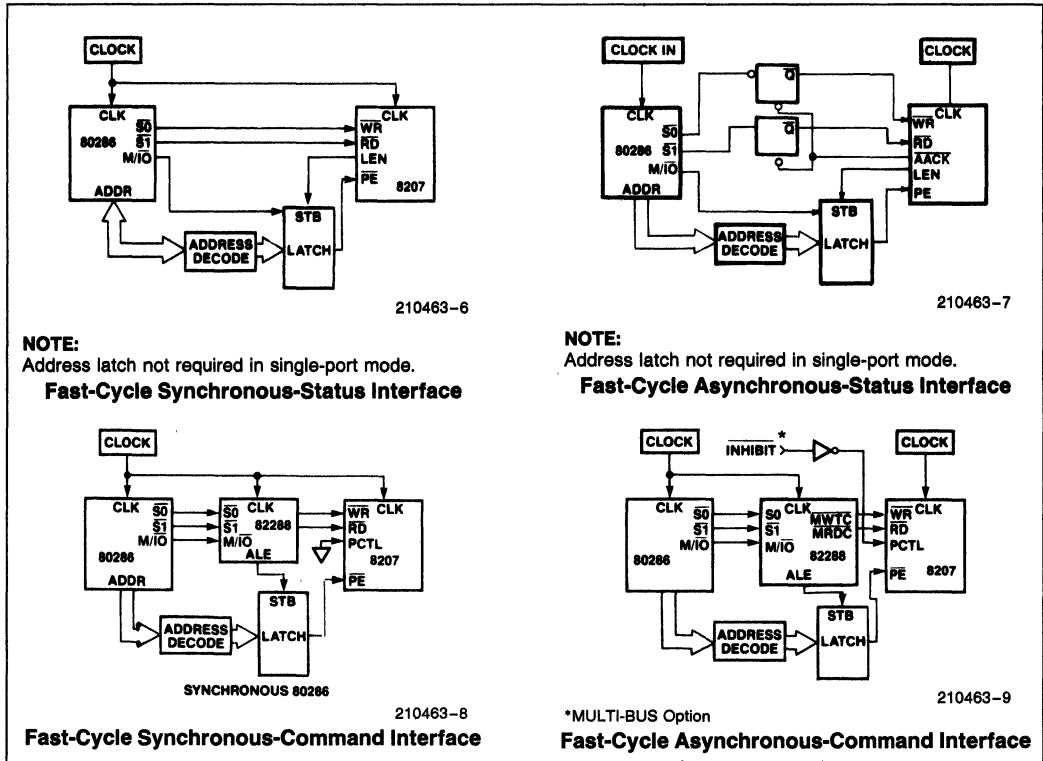


Figure 2B. Fast-Cycle (CFS = 1) Port Interfaces Supported by the 8207

Dynamic RAM Interface

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

The 8207 divides memory into as many as four banks, each bank having its own Row (RAS) and Column (CAS) Address Strobe pair. This organization permits RAM cycle interleaving and permits er-

ror scrubbing during ECC refresh cycles. RAM cycle interleaving overlaps the start of the next RAM cycle with the RAM Precharge period of the previous cycle. Hiding the precharge period of one RAM cycle behind the data access period of the next RAM cycle optimizes memory bandwidth and is effective as long as successive RAM cycles occur in alternate banks.

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

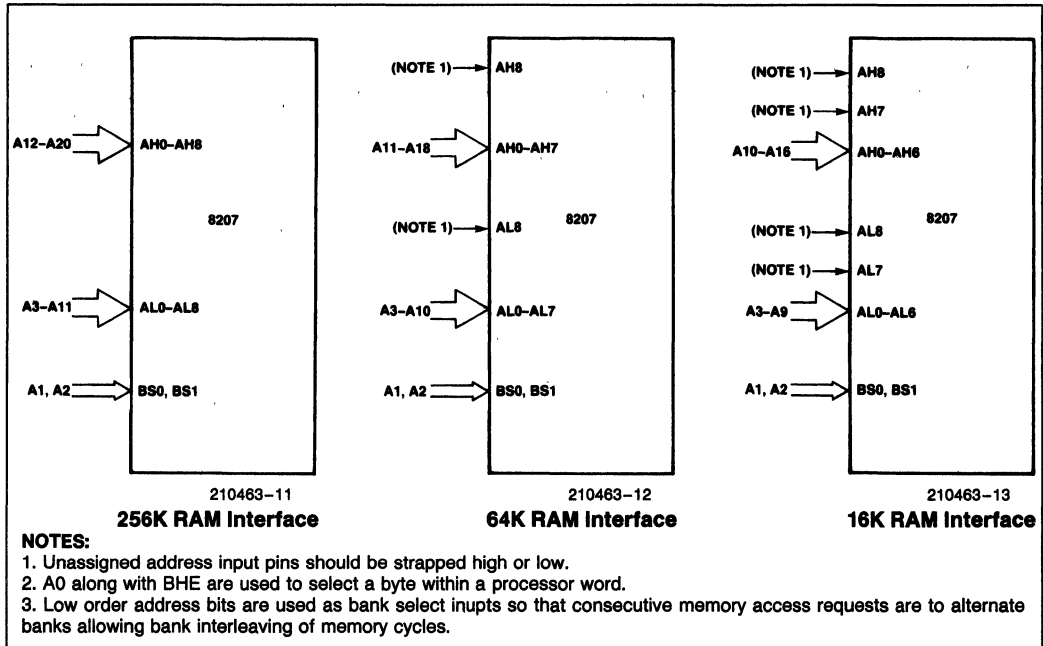


Figure 4. Processor Address Interface to the 8207 Using 16K, 64K, and 256K RAMs

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

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

Memory Initialization

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

Table 2. Bank Selection Decoding and Word Expansion

Program Bits		Bank Input		RAS/CAS Pair Allocation
RB1	RB0	BS1	BS0	
0	0	0	0	RAS ₀₋₃ , CAS ₀₋₃ to Bank 0
0	0	0	1	Illegal
0	0	1	0	Illegal
0	0	1	1	Illegal
0	1	0	0	RAS _{0,1} , CAS _{0,1} to Bank 0
0	1	0	1	RAS _{2,3} , CAS _{2,3} to Bank 1
0	1	1	0	Illegal
0	1	1	1	Illegal
1	0	0	0	RAS ₀ , CAS ₀ to Bank 0
1	0	0	1	RAS ₁ , CAS ₁ to Bank 1
1	0	1	0	RAS ₂ , CAS ₂ to Bank 2
1	0	1	1	Illegal
1	1	0	0	RAS ₀ , CAS ₀ to Bank 0
1	1	0	1	RAS ₁ , CAS ₁ to Bank 1
1	1	1	0	RAS ₂ , CAS ₂ to Bank 2
1	1	1	1	RAS ₃ , CAS ₃ to Bank 3

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

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

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

8206 ECC Interface

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

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

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

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

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

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

Error Scrubbing

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

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

Refresh

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

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

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

External Refresh Requests after RESET

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

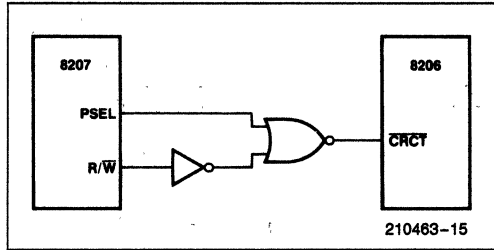


Figure 6. Interface to 8206 CRCT Input when Port A Receives AACK and Port B Receives XACK

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

Non-ECC Systems:

$$T_{RESP} = T_{PROG} + T_{PREP} \quad (2)$$

where:

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

which is programming time

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

which is the RAM warm-up time

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

ECC Systems:

$$T_{RESP} = T_{PROG} + T_{PREP} + T_{INIT} \quad (5)$$

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

RESET

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

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

nous). Differentiated reset is unnecessary when the default port synchronization programming is used.

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

Within four clocks after RESET goes active, all the 8207 outputs will go high, except for PSEN, WE, and A00-2, which will go low.

OPERATIONAL DESCRIPTION

Programming the 8207

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

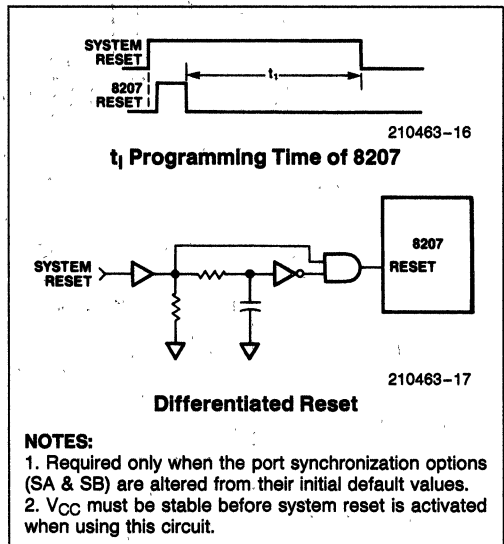


Figure 7. 8207 Differentiated Reset Circuit

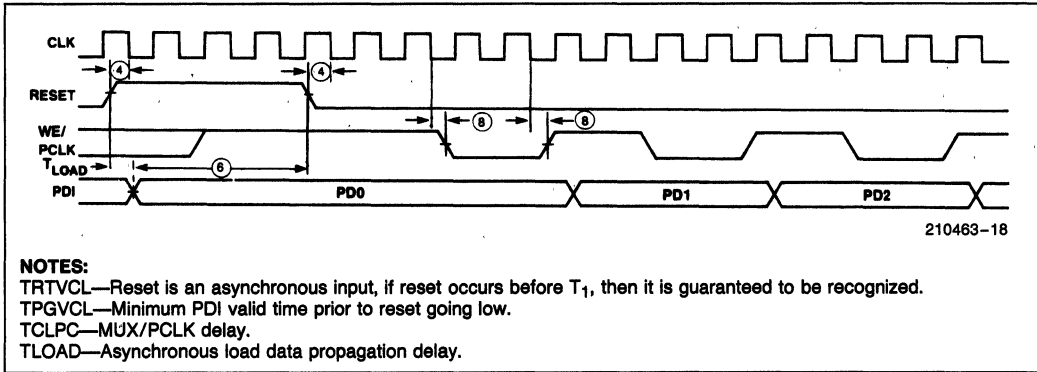


Figure 8. Timing Illustrating External Shift Register Requirements for Programming the 8207

Status/Command Mode

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

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

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

Refresh Options

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

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

Status Code			Function	
S ₂	S ₁	S ₀	8086/80186	80286
0	0	0	Interrupt	Interrupt
0	0	1	I/O Read	I/O Read
0	1	0	I/O Write	I/O Write
0	1	1	Halt	Idle
1	0	0	Instruction Fetch	Halt
1	0	1	Memory Read	Memory Read
1	1	0	Memory Write	Memory Write
1	1	1	Idle	Idle

Table 3B. 8207 Response

8207 Command			Function	
PCTL	R _D	W _R	8086/80186 Status Interface	80286 Status or Command Interface
0	0	0	Ignore	Ignore*
0	0	1	Ignore	Read
0	1	0	Ignore	Write
0	1	1	Ignore	Ignore*
1	0	0	Read	Ignore
1	0	1	Read	Inhibit
1	1	0	Write	Inhibit
1	1	1	Ignore	Ignore

*Illegal with CFS = 0

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

Internal Refresh Only

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

External Refresh with Failsafe

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

External Refresh without Failsafe

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

Burst Refresh

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

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

No Refresh

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

Option Program Data Word

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

Using an External Shift Register

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

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

ECC Mode (ECC Program Bit)

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

Default Programming Options

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

PD15				PD8 PD7								PD0			
0	0	TM1	PPR	FFS	EXT	PLS	CI0	CI1	RB1	RB0	RFS	CFS	SB	SA	0

Program Data Bit	Name	Polarity/Function
PD0	ECC	ECC = 0 For Non-ECC Mode
PD1	\overline{SA}	\overline{SA} = 0 Port A is Synchronous \overline{SA} = 1 Port A is Asynchronous
PD2	SB	SB = 0 Port B is Asynchronous SB = 1 Port B is Synchronous
PD3	\overline{CFS}	\overline{CFS} = 0 Fast-Cycle iAPX 286 Mode \overline{CFS} = 1 Slow-Cycle iAPX 86 Mode
PD4	\overline{RFS}	\overline{RFS} = 0 Fast RAM \overline{RFS} = 1 Slow RAM
PD5 PD6	$\overline{RB0}$ $\overline{RB1}$	RAM Bank Occupancy See Table 2
PD7 PD8	CI1 CI0	Count Interval Bit 1; see Table 6 Count Interval Bit 0; see Table 6
PD9	\overline{PLS}	\overline{PLS} = 0 Long Refresh Period \overline{PLS} = 1 Short Refresh Period
PD10	EXT	EXT = 0 Not Extended EXT = 1 Extended
PD11	FFS	FFS = 0 Fast CPU Frequency FFS = 1 Slow CPU Frequency
PD12	PPR	PPR = 0 Most Recently Used Port Priority PPR = 1 Port A Preferred Priority
PD13	TM1	TM1 = 0 Test Mode 1 Off TM1 = 1 Test Mode 1 Enabled
PD14	0	Reserved, Must be Zero
PD15	0	Reserved, Must be Zero

Figure 9. Non-ECC Mode Program Data Word

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

Program Data Bit	Name	Polarity/Function
PD0	ECC	ECC = 1 ECC Mode
PD1	SA	SA = 0 Port A Asynchronous SA = 1 Port A Synchronous
PD2	\overline{SB}	\overline{SB} = 0 Port B Synchronous \overline{SB} = 1 Port B Asynchronous
PD3	CFS	CFS = 0 Slow-Cycle iAPX 86 Mode CFS = 1 Fast-Cycle iAPX 286 Mode
PD4	RFS	RFS = 0 Slow RAM RFS = 1 Fast RAM
PD5	\overline{XA}	\overline{XA} = 0 MULTIBUS-Compatible ACKA \overline{XA} = 1 Advanced ACKA Not Multibus-Compatible
PD6	XB	Advanced ACKB Not Multibus-Compatible XB = 1 Multibus-Compatible ACKB
PD7	$\overline{CI1}$	Count Interval Bit 1; see Table 6
PD8	$\overline{CI0}$	Count Interval Bit 0; see Table 6
PD9	PLS	PLS = 0 Short Refresh Period PLS = 1 Long Refresh Period
PD10	\overline{EXT}	\overline{EXT} = 0 Master and Slave EDCU \overline{EXT} = 1 Master EDCU Only
PD11	FFS	FFS = 0 Slow CPU Frequency FFS = 1 Fast CPU Frequency
PD12	PPR	PPR = 0 Port A Preferred Priority PPR = 1 Most Recently Used Port Priority
PD13	RB0	RAM Bank Occupancy
PD14	RB1	See Table 2
PD15	$\overline{TM2}$	$\overline{TM2}$ = 0 Test Mode 2 Enabled $\overline{TM1}$ = 1 Test Mode 2 Off

Figure 10. ECC Mode Program Data Word

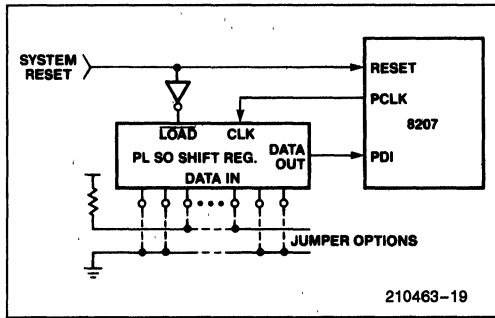


Figure 11. External Shift Register Interface

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

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

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

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

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

Synchronous/Asynchronous Mode (SA and SB Program Bits)

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

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

Microprocessor Clock Frequency Option (CFS and FFS Program Bits)

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

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

Table 5. Microprocessor Clock Frequency Options

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

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

RAM Speed Option (RFS Program Bit)

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

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

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

In addition to PLS program option, two other programming bits for refresh exist: Count Interval 0 (CI0) and Count Interval 1 (CI1). These two programming bits allow the rate at which refresh requests are generated to be increased in order to permit refresh requests to be generated close to the same 15.6 or 7.8 microsecond period when the 8207 is operating at reduced frequencies. The interval be-

tween refreshes is decreased by 0%, 10%, 20%, or 30% as a function of how the count interval bits are programmed. A 5% guardband is built-in to allow for any clock frequency variations. Table 6 shows the refresh period options available.

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

Extend Option (EXT Program Bit)

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

Port Priority Option and Arbitration (PPR Program Bit)

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

Table 6. Refresh Count Interval Table

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

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

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

A port is selected if the arbiter has given the selected port direct access to the timing generators. The front-end logic, which includes the arbiter, is designed to operate in parallel with the selected port. Thus a request on the selected port is serviced immediately. In contrast, an unselected port only has access to the timing generators through the front-end logic. Before a RAM cycle can start for an unselected port, that port must first become selected (i.e., the MUX output now gates that port's address into the 8207 in the case of Port A or B). Also, in order to allow its address to stabilize, a newly selected port's first RAM cycle is started by the front-end logic. Therefore, the selected port has direct access to the timing generators. What all this means is that a request on a selected port is started immediately, while a request on an unselected port is started two to three clock periods after the request, assuming

that the other two ports are idle. Under normal operating conditions, this arbitration time is hidden behind the RAM cycle of the selected port so that as soon as the present cycle is over a new cycle is started. Table 7 lists the arbitration rules for both options.

Port LOCK Function

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

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

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

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

NOTE:

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

Dual-Port Considerations

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

Figure 3 illustrates an example of dual-port operation using the processors in the slow cycle group. Note the use of cross-coupled NAND gates at the MUX output for minimizing contention between the

two latches, and the use of flip flops on the status lines of the asynchronous processor for delaying the status and thereby guaranteeing RAS will not be issued, even in the worst case, until address is valid.

Processor Timing

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

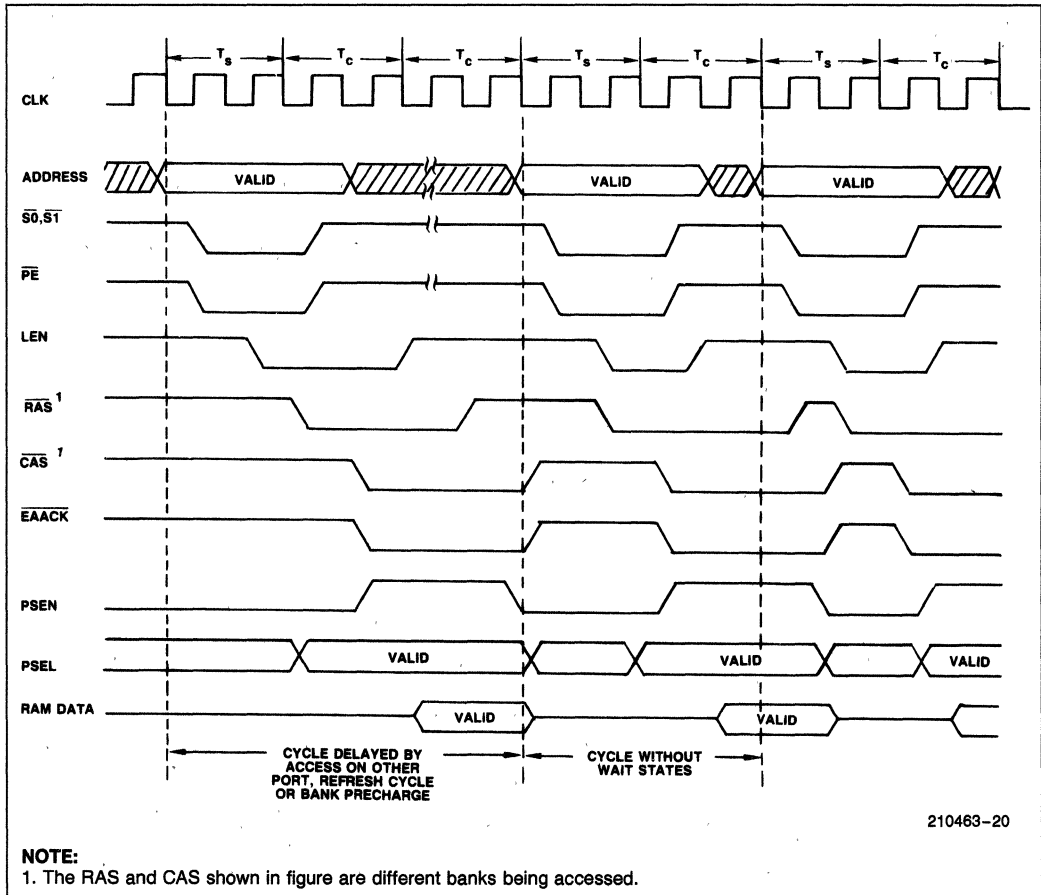


Figure 14. iAPX 286/8207 Synchronous-Status Timing Programmed in non-ECC Mode, C0 Configuration (Read Cycle)

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

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

the internal synchronization delay to be added to the status (or command)-to- \overline{PE} delay time, thus allowing for more external decode time that is available in synchronous operation.

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

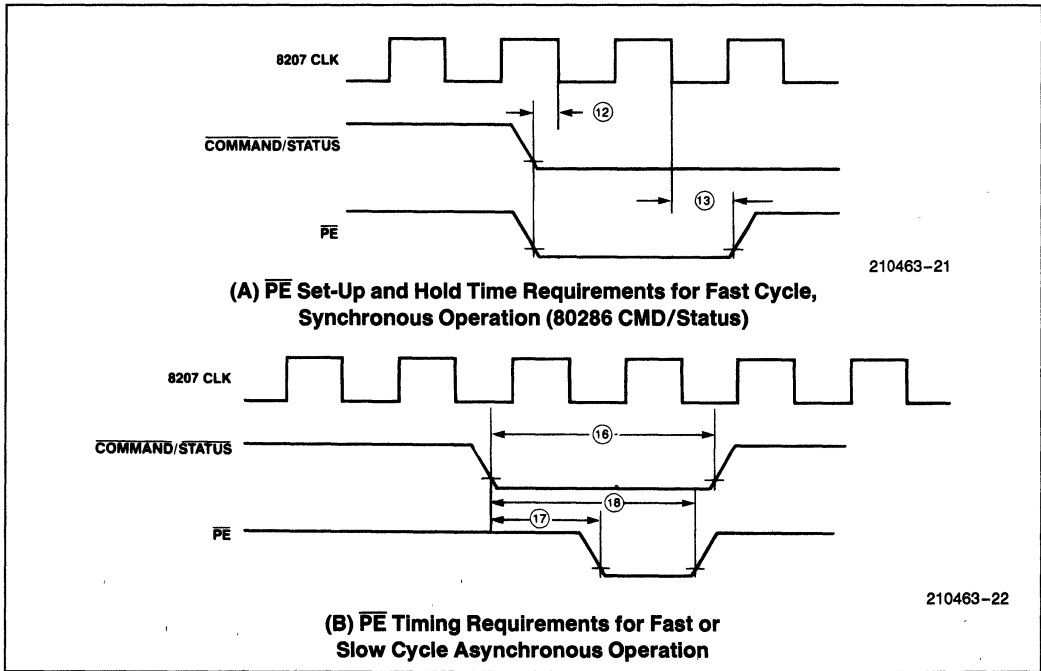
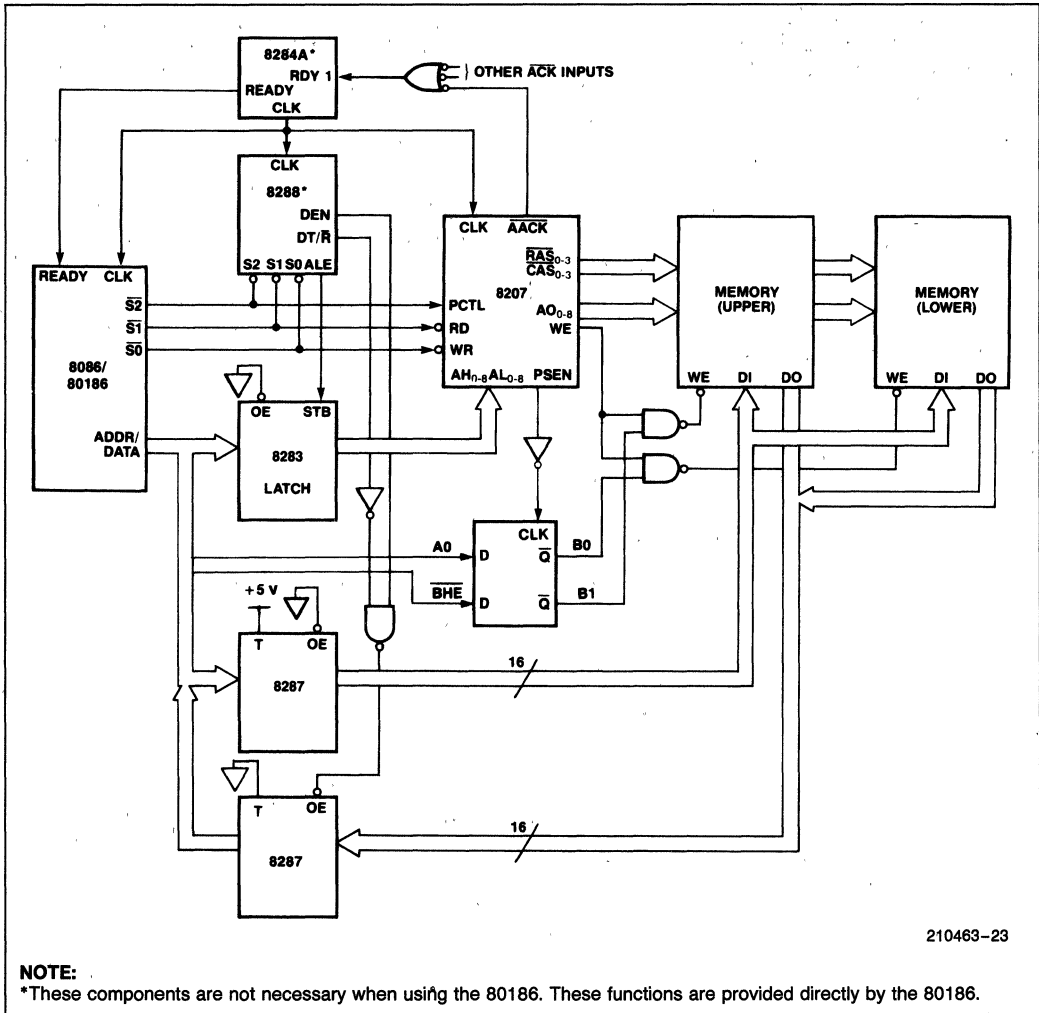


Figure 15



210463-23

NOTE:

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

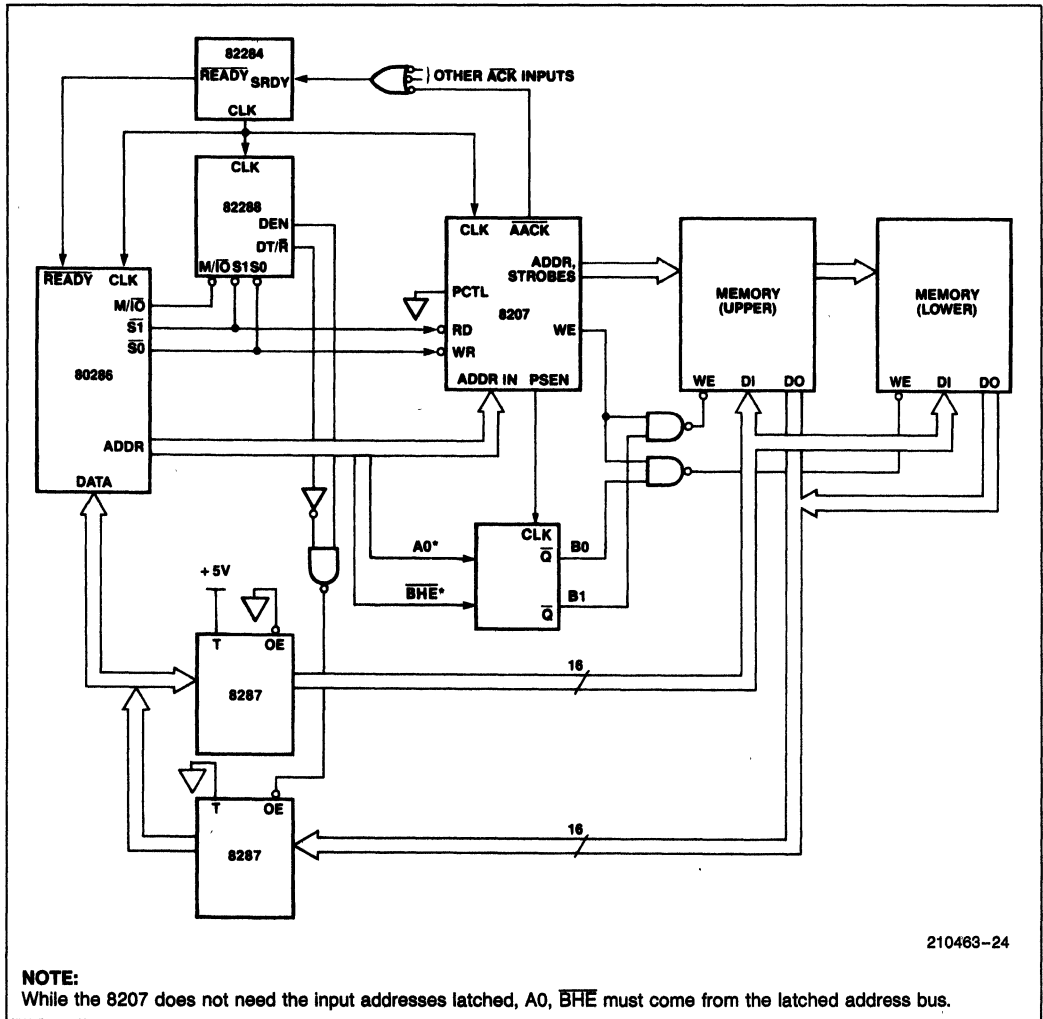
Figure 16. 8086/80186, 8207 Single Port Non-ECC Synchronous Systems

Memory Acknowledge (AACK, XACK)

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

(CFS = 0), while the SA and SB programming bits optimize AACK for synchronous operation ("early" AACK) or asynchronous operation ("late" AACK).

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



210463-24

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

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

In ECC-based systems there is one memory acknowledgement (\overline{XACK} or \overline{AACK}) per port and a program-

ming bit associated with each acknowledge. If the X programming bit is active, the strobe is configured as \overline{XACK} , while if the bit is inactive, the strobe is configured as \overline{AACK} . As in non-ECC, the SA and SB programming bits determine whether the \overline{AACK} strobe is early or late (\overline{EAACK} or \overline{LAACK}).

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

Table 8. Processor Interface/Acknowledge Summary

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

Table 9. Memory Acknowledge Option Summary

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

Test Modes

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

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

Test Mode 2 sets the internal refresh counter to a value slightly less than rollover. During functional

testing other than that covered in Test Mode 1, the 8207 will normally be set in Test Mode 2. Test Mode 2 eliminates memory initialization in ECC mode. This allows quick examination of the circuitry which brings the 8207 out of memory initialization and into normal operation.

General System Considerations

The RAS₀₋₃, CAS₀₋₃, AO₀₋₈, output buffers were designed to directly drive the heavy capacitive loads associated with dynamic RAM arrays. To keep the RAM driver outputs from ringing excessively in the system environment and causing noise in other output pins it is necessary to match the output impedance of the RAM output buffers with the RAM array by using series resistors and to add series resistors to other control outputs for noise reduction if necessary. Each application may have different impedance characteristics and may require different series resistance values. The series resistance values should be determined for each application. In non-ECC systems unused ECC input pins should be tied high or low to improve noise immunity.

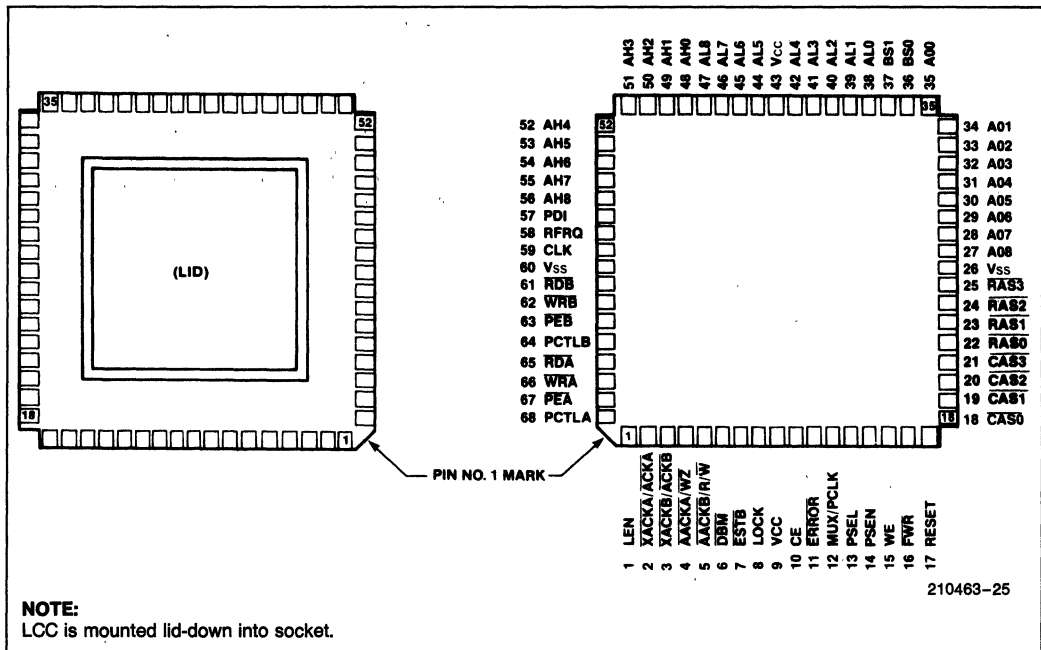
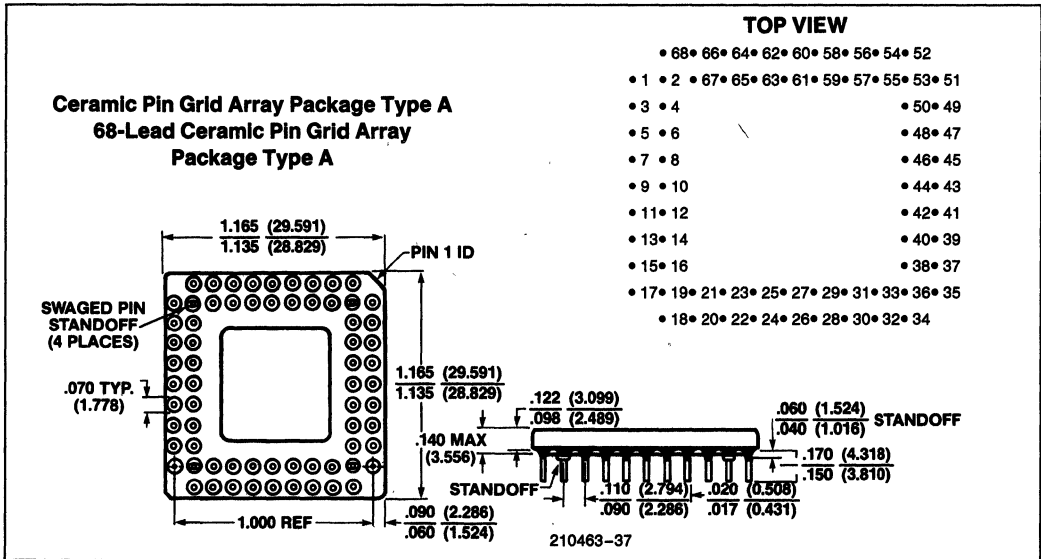


Figure 19. 8207 Pinout Diagram



8207 Pin Grid Array (PGA) Pin-Out

Packaging

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

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

NOTE:

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

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature
 Under Bias -0°C to +70°C
 Storage Temperature -65°C to +150°C
 Voltage on Any Pin with
 Respect to Ground -0.5V to +7V
 Power Dissipation (Note 2) 2.5W

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

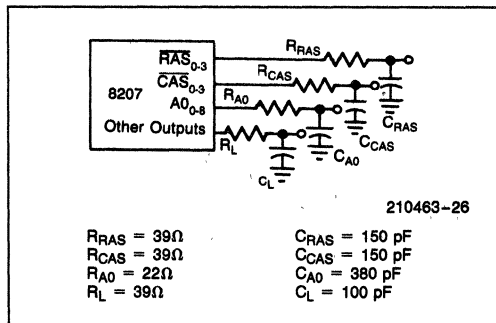
D.C. CHARACTERISTICS $V_{CC} = 5.0V \pm 10\%$ for 8207-10, 8207-8;
 $T_A = 0^\circ C$ to $70^\circ C$; $V_{SS} = GND$; $V_{CC} = 5.0V \pm 5\%$ for 8207-16 (Note 2)

Symbol	Parameter	Min	Max	Units	Comments
V_{IL}	Input Low Voltage	-0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage		0.45	V	(Note 1)
V_{OH}	Output High Voltage	2.4		V	(Note 1)
V_{ROL}	RAM Output Low Voltage		0.45	V	(Note 1)
V_{ROH}	RAM Output High Voltage	2.6		V	(Note 1)
I_{CC}	Supply Current		455	mA	$T_A = 0^\circ C^{(2)}$
I_{LI}	Input Leakage Current		± 10	μA	$0V \leq V_{IN} \leq V_{CC}$
V_{CL}	Clock Input Low Voltage	-0.5	± 0.6	V	
V_{CH}	Clock Input High Voltage	3.8	$V_{CC} + 0.5$	V	
C_{IN}	Input Capacitance		20	pF	$f_c = 1 \text{ MHz}^{(3)}$

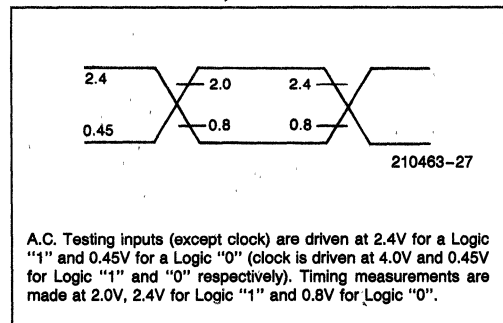
NOTE:

- $I_{OL} = 5 \text{ mA}$ and $I_{OH} = -0.2 \text{ mA}$ (Typically $I_{OL} = 10 \text{ mA}$ and $I_{OH} = -0.88 \text{ mA}$). WE: $I_{OL} = 8 \text{ mA}$.
- These values are expected to improve with conversion to the HMOS III process in 1987.
- Sampled, not 100% tested.

A.C. TESTING LOAD CIRCUIT(2)



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. CHARACTERISTICS
 $V_{CC} = 5V \pm 10\%$ for 8207-8; $T_A = 0^\circ C$ to $70^\circ C$; $V_{CC} = +5V \pm 5\%$ for 8207-16

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

Ref	Symbol	Parameter	8207-16, -8		8207-10		Units	Notes	
			Min	Max	Min	Max			
CLOCK AND PROGRAMMING									
—	tF	Clock Fall Time		10		10	ns	3	
—	tR	Clock Rise Time		10		10	ns	3	
1	TCLCL	Clock Period	8207-16	62.5	200	50.0	250	ns	1
			8207-10					ns	2
			8207-8	125	500			ns	2
2	TCL	Clock Low Time	8207-16	15	180	TCLCL/2-12		ns	1
			8207-10					ns	2
			8207-8	TCLCL/2-12				ns	2
3	TCH	Clock High Time	8207-16	20	180	TCLCL/3-3		ns	1
			8207-10					ns	2
			8207-8	TCLCL/3-3				ns	2
4	TRTVCL	Reset to CLK ↓ Setup		20		40	ns	4	
5	TRTH	Reset Pulse Width		4TCLCL		4TCLCL	ns		
6	TPGVRTL	PCTL, PDI, RFRQ to RESET ↓ Setup		125		125	ns	5	
7	TRTLPGX	PCTL, RFRQ to RESET ↓ Hold		10		10	ns		
8	TCLPC	PCLK from CLK ↓ Delay			45	45	ns		
9	TPDVCL	PDin to CLK ↓ Setup		60		60	ns		
10	TCLPDX	PDin to CLK ↓ Hold		40		40	ns	6	
RAM WARM-UP AND INITIALIZATION									
64	TCLWZL	\overline{WZ} from CLK ↓ Delay			40	40	ns	7	
SYNCHRONOUS μP PORT INTERFACE									
11	TPEVCL	PE to CLK ↓ Setup		27		27	ns	2	
12	TKVCL	\overline{RD} , \overline{WR} , \overline{PE} , PCTL to CLK ↓ Setup		20			ns	1	
13	TCLKX	\overline{RD} , \overline{WR} , \overline{PE} , PCTL to CLK ↓ Hold		0		0	ns		
14	TKVCH	\overline{RD} , \overline{WR} , PCTL to CLK ↑ Setup		20		20	ns	2	

A.C. CHARACTERISTICS (Continued)

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

Measurements made with respect to RAS₀₋₃, CAS₀₋₃, AO₀₋₈, are a +2.4V and 0.8V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

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

A.C. CHARACTERISTICS (Continued)

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

Measurements made with respect to RAS₀₋₃, CAS₀₋₃, AO₀₋₈, are a +2.4V and 0.8V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

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

A.C. CHARACTERISTICS (Continued)

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

Measurements made with respect to RAS₀₋₃, CAS₀₋₃, AO₀₋₈, are a +2.4V and 0.8V. All other pins are measured at 2.0V and 0.8V. All times are ns unless otherwise indicated. Testing done with specified test load.

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

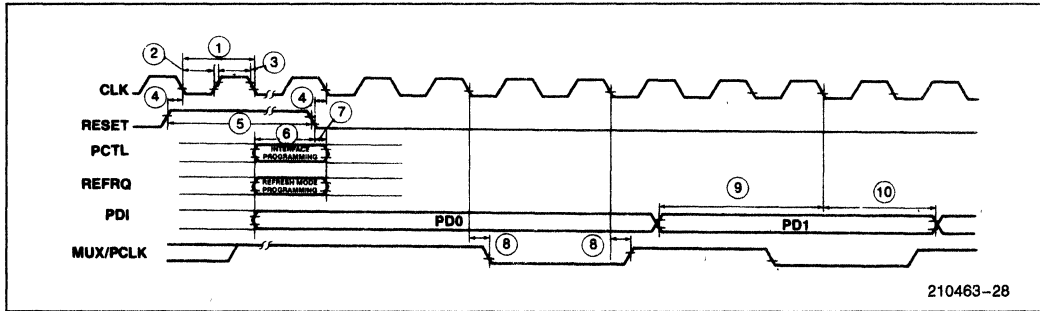
NOTES:

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

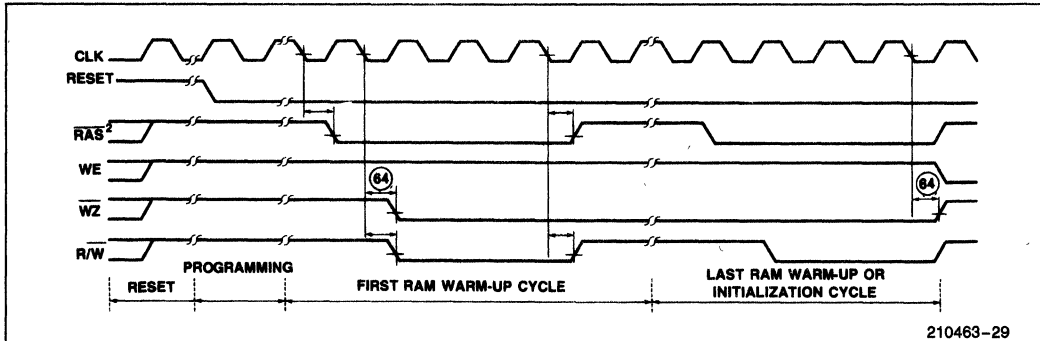
20. tASC is a function of clock frequency and thus varies with changes in frequency. A minimum value is specified.
21. See 8207 DRAM Interface Tables 14–18.
22. TWRLFV is defined for both synchronous and asynchronous \overline{FWR} . In systems in which \overline{FWR} is decoded directly from the address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

WAVEFORMS

CLOCK AND PROGRAMMING TIMINGS



RAM WARM-UP AND MEMORY INITIALIZATION CYCLES

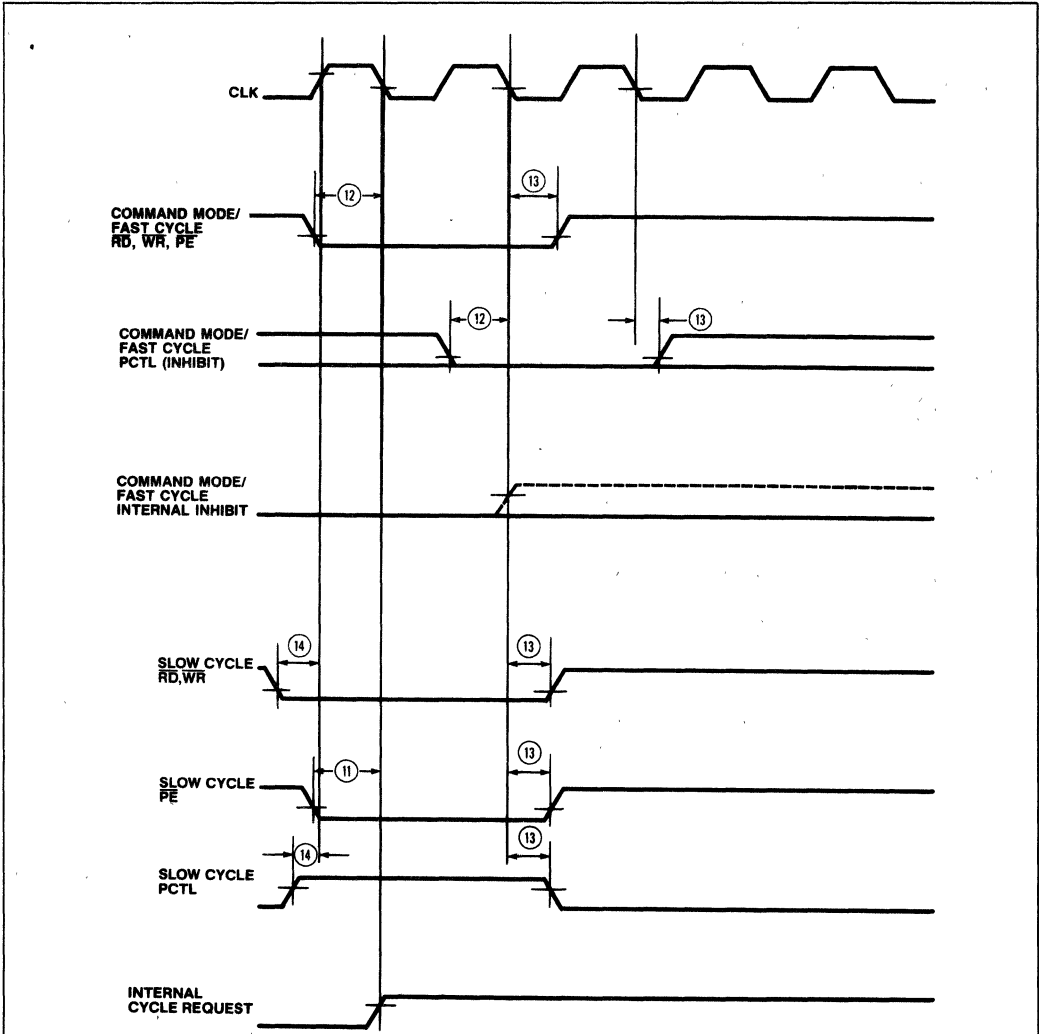


NOTES:

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

WAVEFORMS (Continued)

SYNCHRONOUS PORT INTERFACE

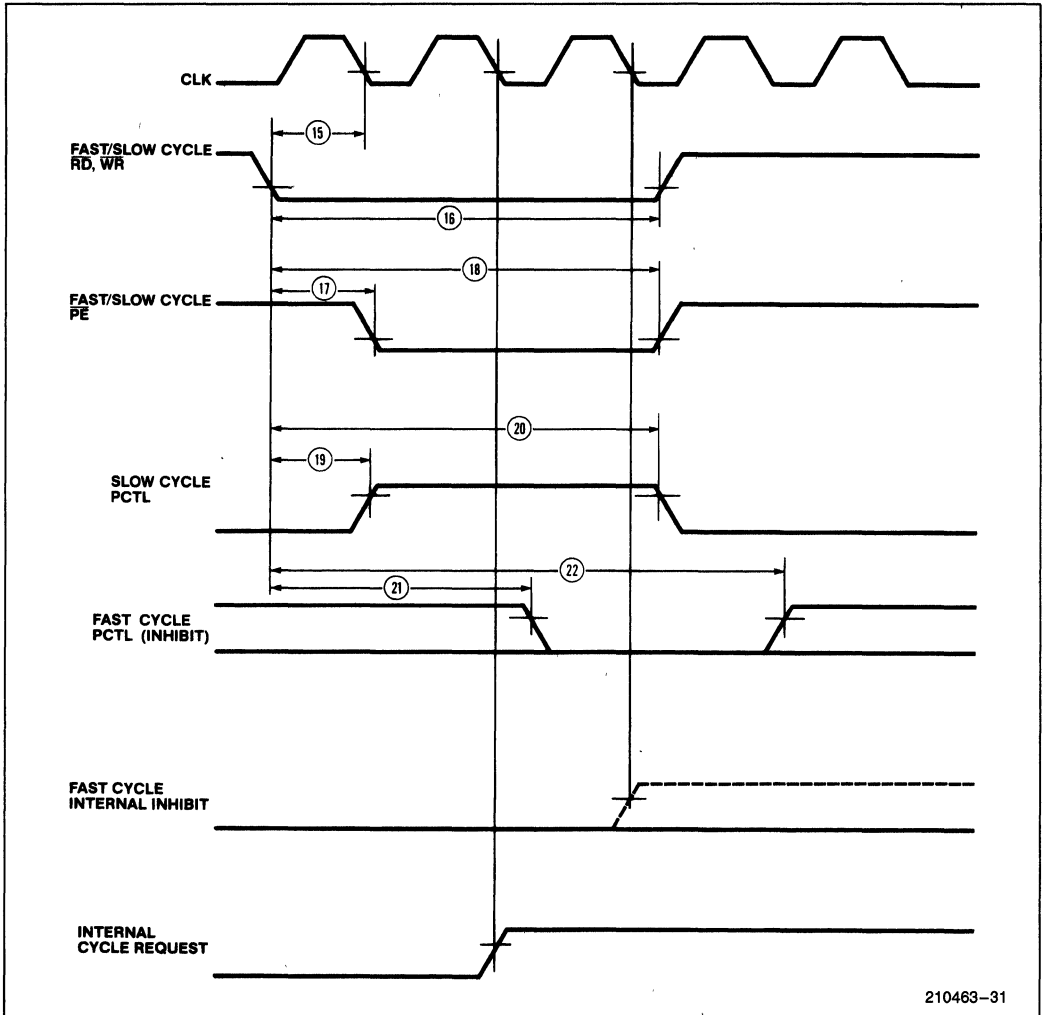


210463-30

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

WAVEFORMS (Continued)

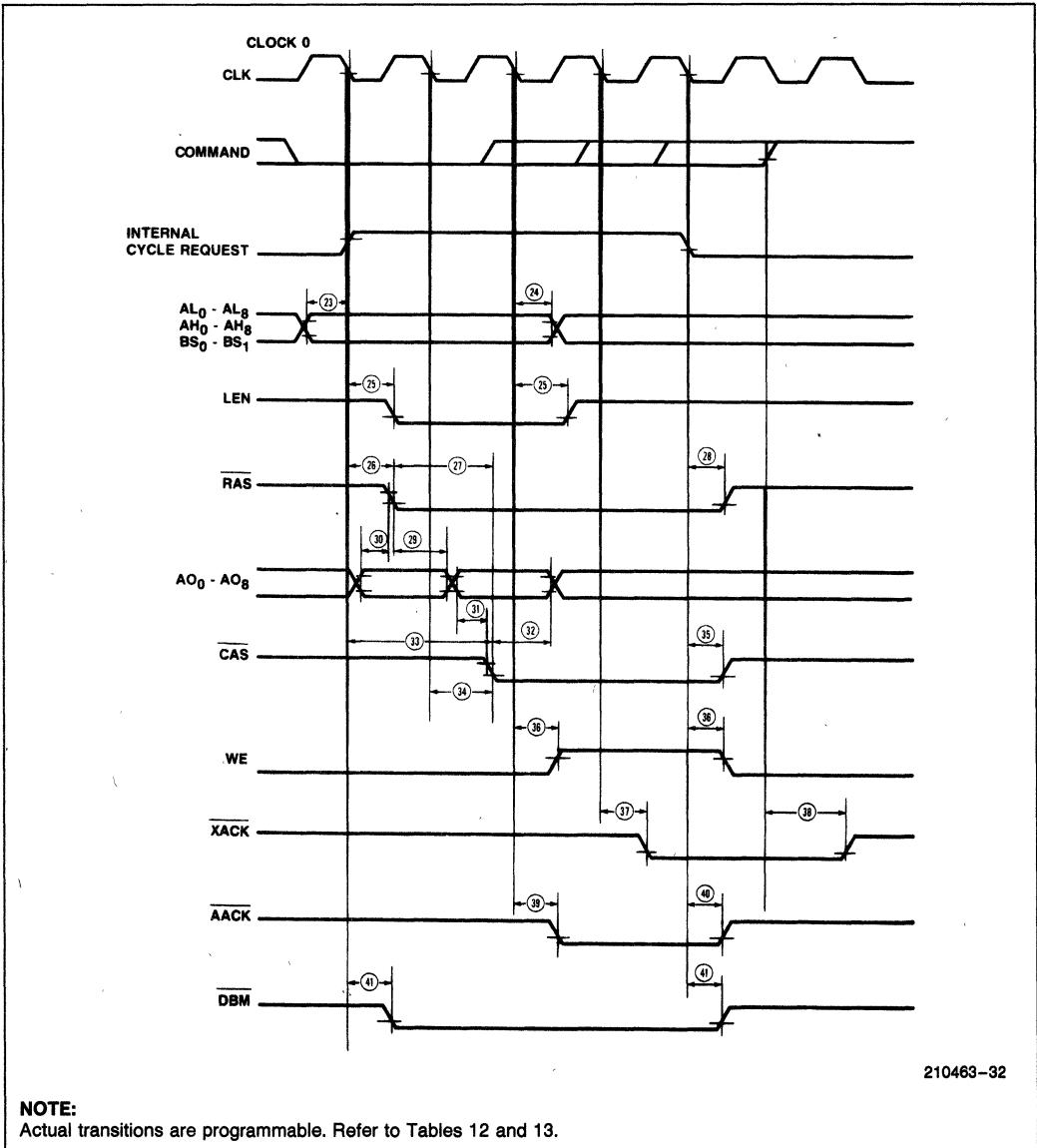
ASYNCHRONOUS PORT INTERFACE



210463-31

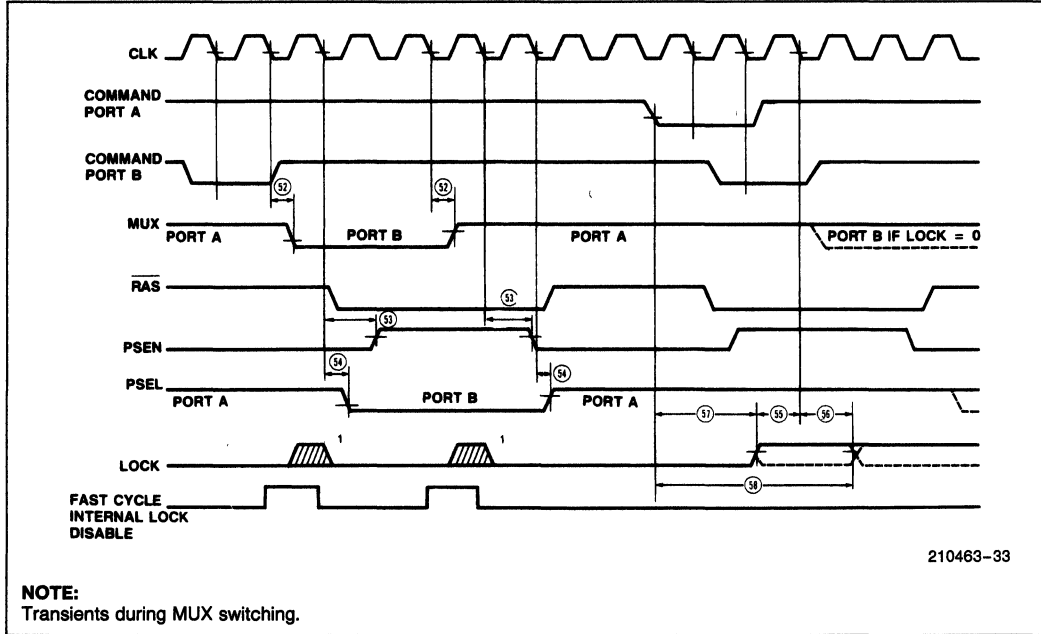
WAVEFORMS (Continued)

RAM INTERFACE TIMING
ECC AND NON-ECC MODE

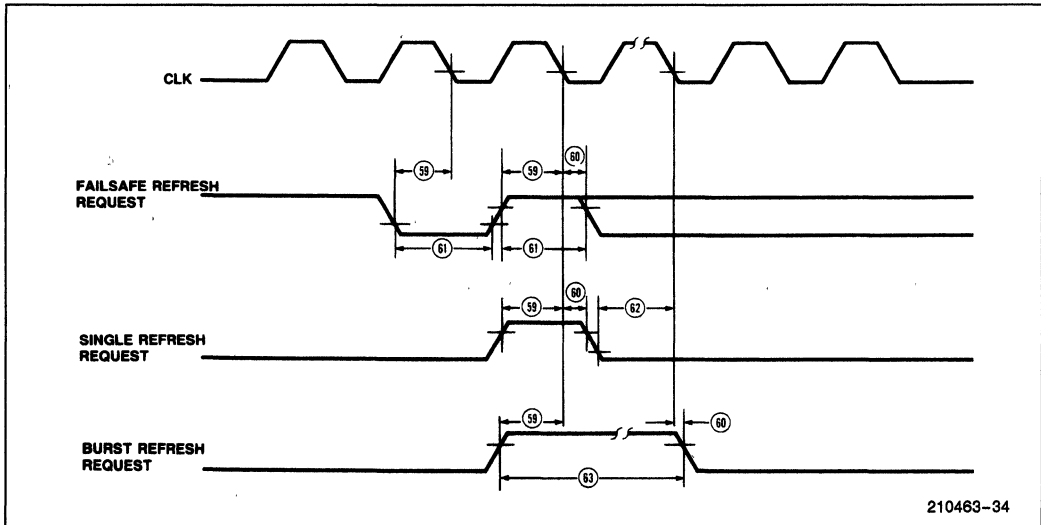


WAVEFORMS (Continued)

PORT SWITCHING AND LOCK TIMING

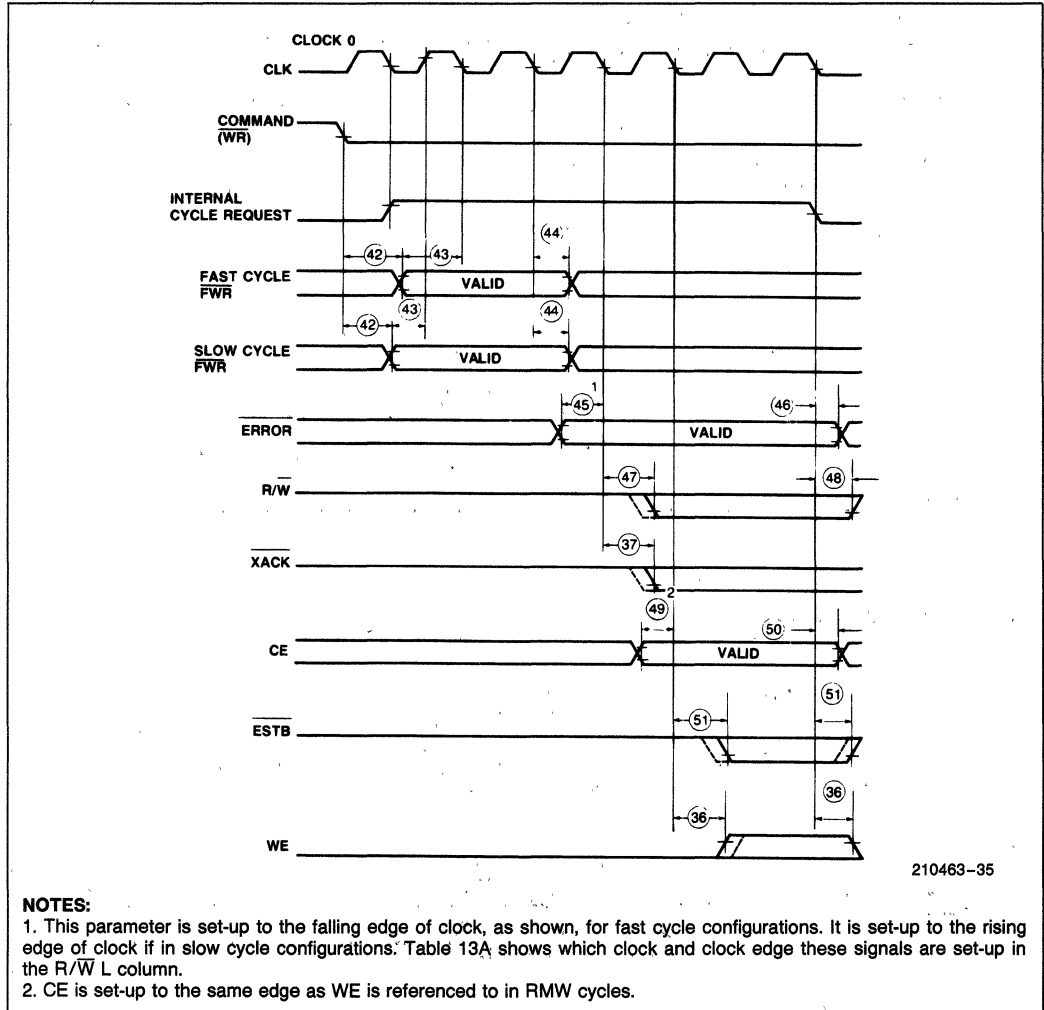


REFRESH REQUEST TIMING



WAVEFORMS (Continued)

ECC INTERFACE TIMING



210463-35

CONFIGURATION TIMING CHARTS

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

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

Table 10. Non-ECC System Configurations

Non-ECC Mode: PD0 = 0

Timing Conf.	CFS(PD3)	RFS(PD4)	EXT(PD10)	FFS(PD11)
C ₀	iAPX286(0)	Fast RAM(0)	Not EXT(0)	12 MHz(1)
C ₀	iAPX286(0)	Fast RAM(0)	EXT(1)	12 MHz(1)
C ₀	iAPX286(0)	Slow RAM(1)	Not EXT(0)	12 MHz(1)
C ₀	iAPX286(0)	Slow RAM(1)	EXT(1)	12 MHz(1)
C ₀	iAPX286(0)	Fast RAM(0)	Not EXT(0)	16 MHz(0)
C ₁	iAPX286(0)	Slow RAM(1)	Not EXT(0)	16 MHz(0)
C ₁	iAPX286(0)	Fast RAM(0)	EXT(1)	16 MHz(0)
C ₂	iAPX286(0)	Slow RAM(1)	EXT(1)	16 MHz(0)
C ₃	iAPX186(1)	Fast RAM(0)	Not EXT(0)	10, 8 MHz(0)
C ₃	iAPX186(1)	Slow RAM(1)	Not EXT(0)	10, 8 MHz(0)
C ₃	iAPX186(1)	Fast RAM(0)	EXT(1)	10, 8 MHz(0)
C ₃	iAPX186(1)	Fast RAM(0)	Not EXT(0)	6 MHz(1)
C ₃	iAPX186(1)	Fast RAM(0)	EXT(1)	6 MHz(1)
C ₃	iAPX186(1)	Slow RAM(1)	Not EXT(0)	6 MHz(1)
C ₃	iAPX186(1)	Slow RAM(1)	EXT(1)	6 MHz(1)
C ₄	iAPX186(1)	Slow RAM(1)	EXT(1)	10, 8 MHz(0)

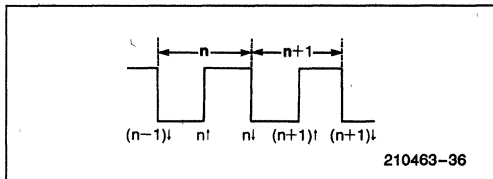
Table 11. ECC System Configurations

ECC Mode: PD0 = 1

Timing Conf.	CFS(PD3)	RFS(PD4)	EXT(PD10)	FFS(PD11)
C ₀	iAPX286(1)	Slow RAM(0)	M/S EDCU(0)	10 MHz(0)
C ₀	iAPX286(1)	Slow RAM(0)	M EDCU(1)	10 MHz(0)
C ₀	iAPX286(1)	Fast RAM(1)	M/S EDCU(0)	10 MHz(0)
C ₀	iAPX286(1)	Fast RAM(1)	M EDCU(1)	10 MHz(0)
C ₀	iAPX286(1)	Fast RAM(1)	M EDCU(1)	16 MHz(1)
C ₁	iAPX286(1)	Slow RAM(0)	M EDCU(1)	16 MHz(1)
C ₂	iAPX286(1)	Fast RAM(1)	M/S EDCU(0)	16 MHz(1)
C ₃	iAPX286(1)	Slow RAM(0)	M/S EDCU(0)	16 MHz(1)
C ₄	iAPX186(0)	Slow RAM(0)	M/S EDCU(0)	6 MHz(0)
C ₄	iAPX186(0)	Fast RAM(1)	M/S EDCU(0)	6 MHz(0)
C ₄	iAPX186(0)	Slow RAM(0)	M EDCU(1)	10, 8 MHz(1)
C ₄	iAPX186(0)	Fast RAM(1)	M EDCU(1)	10, 8 MHz(1)
C ₅	iAPX186(0)	Slow RAM(0)	M/S EDCU(0)	10, 8 MHz(1)
C ₅	iAPX186(0)	Fast RAM(1)	M/S EDCU(0)	10, 8 MHz(1)
C ₆	iAPX186(0)	Slow RAM(0)	M EDCU(1)	6 MHz(0)
C ₆	iAPX186(0)	Fast RAM(1)	M EDCU(1)	6 MHz(0)

Using the Timing Charts

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



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

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

Notes for interpreting the timing charts:

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

Table 12A. Timing Chart—Non-ECC Mode

C _n	Cycle	PSEN		PSEL		DBM		LEN		RAS		CAS		WE	
		H	L	V	\bar{V}	L	H	L	H	L	H	L	H	L	
C ₀	RD, RF	0↓	3↓	0↓	4↓	0↓	4↓	0↓	2↓	0↓	3↓	1↓	4↓		
	WR	0↓	4↓	0↓	5↓			0↓	2↓	0↓	5↓	1↓	5↓	2↓	5↓
C ₁	RD, RF	0↓	5↓	0↓	6↓	0↓	6↓	0↓	2↓	0↓	4↓	1↓	6↓		
	WR	0↓	4↓	0↓	5↓			0↓	2↓	0↓	5↓	1↓	5↓	2↓	5↓
C ₂	RD, RF	0↓	5↓	0↓	6↓	0↓	6↓	0↓	2↓	0↓	4↓	1↓	6↓		
	WR	0↓	4↓	0↓	5↓			0↓	2↓	0↓	5↓	1↓	5↓	2↓	5↓
C ₃	RD, RF	0↓	2↓	0↓	3↓	0↓	3↓			0↓	3↓	0↓	3↓		
	WR	0↓	3↓	0↓	4↓					0↓	4↓	0↓	4↓	2↑	4↓
C ₄	RD, RF	0↓	3↓	0↓	4↓	0↓	4↓			0↓	4↓	0↓	4↓		
	WR	0↓	3↓	0↓	4↓					0↓	4↓	0↓	4↓	2↑	4↓

Table 12B. Timing Chart—Non-ECC Mode

C _n	Cycle	Col Addr		EAACK		LAACK		XACK		MUX	
		V	\bar{V}	L	H	L	H	L	H	V	\bar{V}
C ₀	RD, RF	0↓	2↓	1↓	4↓	2↓	5↓	3↓	\bar{RD}	-2↓	2↓
	WR	0↓	2↓	1↓	4↓	1↓	4↓	3↓	\bar{WR}	-2↓	2↓
C ₁	RD, RF	0↓	3↓	2↓	5↓	2↓	5↓	4↓	\bar{RD}	-2↓	2↓
	WR	0↓	3↓	1↓	4↓	1↓	4↓	3↓	\bar{WR}	-2↓	2↓
C ₂	RD, RF	0↓	3↓	2↓	5↓	3↓	6↓	4↓	\bar{RD}	-2↓	2↓
	WR	0↓	3↓	1↓	4↓	1↓	4↓	3↓	\bar{WR}	-2↓	2↓
C ₃	RD, RF	0↓	2↓	0↓	2↓	1↓	3↓	2↓	\bar{RD}	-1↓	2↓
	WR	0↓	2↓	0↓	2↓	1↑	3↑	2↓	\bar{WR}	-1↓	2↓
C ₄	RD, RF	0↓	2↓	1↓	3↓	1↓	3↓	3↑	\bar{RD}	-1↓	2↓
	WR	0↓	2↓	0↓	2↓	1↑	3↑	2↓	\bar{WR}	-1↓	2↓

C _n	Cycle	H		L		V		V		PSEL		DBM		LEN		RAS		CAS		R/W		WE			
		RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR	RD, RF	WR
C ₀	RD, RF	0	↑	5	↓	6	↑	6	↑	6	↑	6	↑	6	↑	2	↑	4	↑	1	↑	6	↑	6	↑
	WR	0	↑	5	↓	6	↑	6	↑	6	↑	6	↑	6	↑	2	↑	4	↑	1	↑	6	↑	3	↓
C ₁	RD, RF	0	↑	5	↓	6	↑	6	↑	6	↑	6	↑	6	↑	2	↑	4	↑	1	↑	6	↑	3	↓
	WR	0	↑	5	↓	6	↑	6	↑	6	↑	6	↑	6	↑	2	↑	4	↑	1	↑	6	↑	3	↓
C ₂	RD, RF	0	↑	6	↑	6	↑	6	↑	6	↑	7	↑	7	↑	2	↑	5	↑	1	↑	7	↑	4	↑
	WR	0	↑	6	↑	6	↑	6	↑	6	↑	7	↑	7	↑	2	↑	5	↑	1	↑	7	↑	4	↑
C ₃	RD, RF	0	↑	6	↑	6	↑	6	↑	6	↑	7	↑	7	↑	2	↑	5	↑	1	↑	7	↑	4	↑
	WR	0	↑	6	↑	6	↑	6	↑	6	↑	7	↑	7	↑	2	↑	5	↑	1	↑	7	↑	4	↑
C ₄	RD, RF	0	↑	3	↑	0	↑	4	↑	0	↑	4	↑	0	↑	3	↑	0	↑	4	↑	0	↑	3	↓
	WR	0	↑	4	↑	0	↑	4	↑	0	↑	5	↑	0	↑	5	↑	0	↑	5	↑	1	↓	5	↑
C ₅	RD, RF	0	↑	3	↑	0	↑	4	↑	0	↑	4	↑	0	↑	3	↑	0	↑	4	↑	0	↑	3	↓
	WR	0	↑	4	↑	0	↑	4	↑	0	↑	5	↑	0	↑	5	↑	0	↑	5	↑	1	↓	5	↑
C ₆	RD, RF	0	↑	3	↑	0	↑	4	↑	0	↑	4	↑	0	↑	3	↑	0	↑	4	↑	0	↑	3	↓
	WR	0	↑	3	↑	0	↑	4	↑	0	↑	4	↑	0	↑	3	↑	0	↑	4	↑	0	↑	2	↓

Table 13A. Timing Chart—ECC Mode

Table 13B. Timing Chart—ECC Mode

		Col Addr		ESTB		EAACK		LAACK		XACK		MUX	
C _n	Cycle	V	\bar{V}	L	H	L	H	L	H	L	H	V	\bar{V}
C ₀	RD, RF	0↓	2↓			2↓	5↓	3↓	6↓	4↓	RD	-2↓	2↓
	WR	0↓	2↓			2↓	5↓	2↓	5↓	4↓	WR	-2↓	2↓
	RMW	0↓	2↓	6↓	8↓	5↓	8↓	5↓	8↓	7↓	WR	-2↓	2↓
C ₁	RD, RF	0↓	3↓			3↓	6↓	3↓	6↓	4↓	RD	-2↓	2↓
	WR	0↓	3↓			2↓	5↓	2↓	5↓	4↓	WR	-2↓	2↓
	RMW	0↓	3↓	6↓	8↓	5↓	8↓	5↓	8↓	7↓	WR	-2↓	2↓
C ₂	RD, RF	0↓	3↓			4↓	7↓	4↓	7↓	5↓	RD	-2↓	2↓
	WR	0↓	3↓			3↓	6↓	3↓	6↓	5↓	WR	-2↓	2↓
	RMW	0↓	3↓	8↓	10↓	7↓	10↓	7↓	10↓	9↓	WR	-2↓	2↓
C ₃	RD, RF	0↓	3↓			4↓	7↓	5↓	8↓	5↓	RD	-2↓	2↓
	WR	0↓	3↓			3↓	6↓	3↓	6↓	5↓	WR	-2↓	2↓
	RMW	0↓	3↓	8↓	10↓	7↓	10↓	7↓	10↓	9↓	WR	-2↓	2↓
C ₄	RD, RF	0↓	2↓			1↓	3↓	2↑	4↑	3↑	RD	-1↓	2↓
	WR	0↓	2↓			1↓	3↓	2↑	4↑	3↓	WR	-1↓	2↓
	RMW	0↓	2↓	5↑	6↑	3↓	5↓	4↑	6↑	5↓	WR	-1↓	2↓
C ₅	RD, RF	0↓	2↓			2↓	4↓	3↑	5↑	3↑	RD	-1↓	2↓
	WR	0↓	2↓			1↓	3↓	2↑	4↑	3↓	WR	-1↓	2↓
	RMW	0↓	2↓	5↑	6↑	3↓	5↓	4↑	6↑	5↓	WR	-1↓	2↓
C ₂	RD, RF	0↓	2↓			1↓	3↓	1↑	3↑	2↑	RD	-1↓	2↓
	WR	0↓	2↓			1↓	3↓	1↑	3↑	2↓	WR	-1↓	2↓
	RMW	0↓	2↓	3↑	4↑	1↓	3↓	2↑	4↑	3↓	WR	-1↓	2↓

8207—DRAM Interface Parameter Equations

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

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

- tRAC: response parameter.
- tCAC: response parameter.
- tREF: See "Refresh Period Options"
- tCRP: must be met only if $\overline{\text{CAS}}$ -only cycles, which do not occur with 8207, exist.
- tRAH: See "A.C. Characteristics"
- tRCD: See "A.C. Characteristics"
- tASC: See "A.C. Characteristics"
- tASR: See "A.C. Characteristics"
- tOFF: response parameter.

READ & REFRESH CYCLES

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

WRITE CYCLE

- tRC: guaranteed by tRWC.
- tRAS: guaranteed by tRRW.
- tCAS: guaranteed by tCRW.
- tWCS: $\overline{\text{WE}}$ always activated after $\overline{\text{CAS}}$ is activated, except in memory initialization, hence tWCS is always negative (this is important for RMW only) except in memory initialization; in memory initialization tWCS is positive and has several clocks of margin.
- tDS: system-dependent parameter.
- tDH: system-dependent parameter.
- tDHR: system-dependent parameter.

READ-MODIFY-WRITE CYCLE

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

Table 14. Non-ECC Mode—RD, RF Cycles

Parameter	Fast Cycle Configurations			Slow Cycle Configurations		Notes
	C ₀	C ₁	C ₂	C ₃	C ₄	
tRP	3TCLCL - T26	4TCLCL - T26	4TCLCL - T26	2TCLCL - T26	2TCLCL - T26	1
tCPN	3TCLCL - T35	3TCLCL - T35	3TCLCL - T35	2.5TCLCL - T35	2.5TCLCL - T35	1
tRSH	2TCLCL - T34	3TCLCL - T34	3TCLCL - T34	3TCLCL - T34	4TCLCL - T34	1
tCSH	4TCLCL - T26	6TCLCL - T26	6TCLCL - T26	3TCLCL - T26	4TCLCL - T26	1
tCAH	TCLCL - T34	2TCLCL - T34	2TCLCL - T34	2TCLCL - T34	2TCLCL - T34	1
tAR	2TCLCL - T26	3TCLCL - T26	3TCLCL - T26	2TCLCL - T26	2TCLCL - T26	1
tT	3/30	3/30	3/30	3/30	3/30	2
tRC	6TCLCL	8TCLCL	8TCLCL	5TCLCL	6TCLCL	1
tRAS	3TCLCL - T26	4TCLCL - T26	4TCLCL - T26	3TCLCL - T26	4TCLCL - T26	1
tCAS	3TCLCL - T34	5TCLCL - T34	5TCLCL - T34	3TCLCL - T34	4TCLCL - T34	1
tRCS	2TCLCL - TCL - T36 - TBUF	2TCLCL - TCL - T36 - TBUF	2TCLCL - TCL - T36 - TBUF	1.5TCLCL - TCL - T36 - TBUF	1.5TCLCL - TCL - T36 - TBUF	1

Table 15. Non-ECC Mode—WR Cycle

Parameter	Fast Cycle Configurations			Slow Cycle Configurations		Notes
	C ₀	C ₁	C ₂	C ₃	C ₄	
tRP	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	2TCLCL – T26	2TCLCL – T26	1
tCPN	4TCLCL – T35	4TCLCL – T35	4TCLCL – T35	2.5TCLCL – T35	2.5TCLCL – T35	1
tRSH	4TCLCL – T34	4TCLCL – T34	4TCLCL – T34	4TCLCL – T34	4TCLCL – T34	1
tCSH	5TCLCL – T26	5TCLCL – T26	5TCLCL – T26	4TCLCL – T26	4TCLCL – T26	1
tCAH	TCLCL – T34	2TCLCL – T34	2TCLCL – T34	2TCLCL – T34	2TCLCL – T34	1
tAR	2TCLCL – T26	3TCLCL – T26	3TCLCL – T26	2TCLCL – T26	2TCLCL – T26	1
tT	3/30	3/30	3/30	3/30	3/30	2
tRWC	8TCLCL	8TCLCL	8TCLCL	6TCLCL	6TCLCL	1
tRRW	5TCLCL – T26	5TCLCL – T26	5TCLCL – T26	4TCLCL – T26	4TCLCL – T26	1
tCRW	4TCLCL – T34	4TCLCL – T34	4TCLCL – T34	4TCLCL – T34	4TCLCL – T34	1
tWCH	3TCLCL + TCL – T34	3TCLCL + TCL – T34	3TCLCL + TCL – T34	3TCLCL + TCL – T34	3TCLCL + TCL – T34	1, 3
tWCR	4TCLCL + TCL – T26	4TCLCL + TCL – T26	4TCLCL + TCL – T26	3TCLCL + TCL – T26	3TCLCL + TCL – T26	1, 3
tWP	2TCLCL + TCL – T36 – TBUF	2TCLCL + TCL – T36 – TBUF	2TCLCL + TCL – T36 – TBUF	2TCLCL – T36 – TBUF	2TCLCL – T36 – TBUF	1
tRWL	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	1
tCWL	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	1

Table 16A. ECC Mode—RD, RF Cycles

Parameter	Fast Cycle Mode				Notes
	C ₀	C ₁	C ₂	C ₃	
tRP	4TCLCL – T26	4TCLCL – T26	4TCLCL – T26	4TCLCL – T26	1
tCPN	3TCLCL – T35	3TCLCL – T35	3TCLCL – T35	3TCLCL – T35	1
tRSH	3TCLCL – T34	3TCLCL – T34	4TCLCL – T34	4TCLCL – T34	1
tCSH	6TCLCL – T26	6TCLCL – T26	7TCLCL – T26	7TCLCL – T26	1
tCAH	TCLCL – T34	2TCLCL – T34	2TCLCL – T34	2TCLCL – T34	1
tAR	2TCLCL – T26	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	1
tT	3/30	3/30	3/30	3/30	2
tRC	8TCLCL	8TCLCL	9TCLCL	9TCLCL	1
tRAS	4TCLCL – T26	4TCLCL – T26	5TCLCL – T26	5TCLCL – T26	1
tCAS	5TCLCL – T34	5TCLCL – T34	6TCLCL – T34	6TCLCL – T34	1
tRCS	TCLCL – T36 – TBUF	TCLCL – T36 – TBUF	TCLCL – T36 – TBUF	TCLCL – T36 – TBUF	1

Table 16B. ECC Mode—RD, RF Cycles

Parameter	Slow Cycle Mode			Notes
	C ₄	C ₅	C ₆	
tRP	2TCLCL – T26	2TCLCL – T26	2TCLCL – T26	1
tCPN	1.5TCLCL – T35	1.5TCLCL – T35	1.5TCLCL – T35	1
tRSH	3TCLCL – T34	3TCLCL – T34	3TCLCL – T34	1
tCSH	4TCLCL – T26	4TCLCL – T26	4TCLCL – T26	1
tCAH	2TCLCL – T34	2TCLCL – T34	2TCLCL – T34	1
tAR	2TCLCL – T26	2TCLCL – T26	2TCLCL – T26	1
tT	3/30	3/30	3/30	2
tRC	5TCLCL	5TCLCL	5TCLCL	1
tRAS	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	1
tCAS	4TCLCL – T34	4TCLCL – T34	4TCLCL – T34	1
tRCS	0.5TCLCL – T36 – TBUF	0.5TCLCL – T36 – TBUF	0.5TCLCL – T36 – TBUF	1

Table 17A. ECC Mode—WR Cycle

Parameter	Fast Cycle Mode				Notes
	C ₀	C ₁	C ₂	C ₃	
tRP	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	1
tCPN	4TCLCL – T35	4TCLCL – T35	4TCLCL – T35	4TCLCL – T35	1
tRSH	5TCLCL – T34	5TCLCL – T34	6TCLCL – T34	6TCLCL – T34	1
tCSH	6TCLCL – T26	6TCLCL – T26	7TCLCL – T26	7TCLCL – T26	1
tCAH	TCLCL – T34	2TCLCL – T34	2TCLCL – T34	2TCLCL – T34	1
tAR	2TCLCL – T26	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	1
tT	3/30	3/30	3/30	3/30	2
tRWC	9TCLCL	9TCLCL	10TCLCL	10TCLCL	1
tRRW	6TCLCL – T26	6TCLCL – T26	7TCLCL – T26	7TCLCL – T26	1
tCRW	5TCLCL – T34	5TCLCL – T34	6TCLCL – T34	6TCLCL – T34	1
tWCH	5TCLCL – T34	5TCLCL – T34	6TCLCL – T34	6TCLCL – T34	1, 4
tWCR	6TCLCL – T26	6TCLCL – T26	7TCLCL – T26	7TCLCL – T26	1, 4
tWP	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	1
tRWL	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	1
tCWL	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	1

Table 17B. ECC Mode—WR Cycle

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

Table 18A. ECC Mode—RMW

Parameter	Fast Cycle Mode				Notes
	C ₀	C ₁	C ₂	C ₃	
tRP	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	1
tCPN	4TCLCL – T35	4TCLCL – T35	4TCLCL – T35	4TCLCL – T35	1
tRSH	8TCLCL – T34	8TCLCL – T34	10TCLCL – T34	10TCLCL – T34	1
tCSH	9TCLCL – T26	9TCLCL – T26	11TCLCL – T26	11TCLCL – T26	1
tCAH	TCLCL – T34	2TCLCL – T34	2TCLCL – T34	2TCLCL – T34	1
tAR	2TCLCL – T26	3TCLCL – T26	3TCLCL – T26	3TCLCL – T26	1
tT	3/30	3/30	3/30	3/30	2
tRWC	12TCLCL	12TCLCL	14TCLCL	14TCLCL	1
tRRW	9TCLCL – T26	9TCLCL – T26	11TCLCL – T26	11TCLCL – T26	1
tCRW	8TCLCL – T34	8TCLCL – T34	10TCLCL – T34	10TCLCL – T34	1
tRCS	TCLCL – T36 – TBUF	TCLCL – T36 – TBUF	TCLCL – T36 – TBUF	TCLCL – T36 – TBUF	1
tRWD	6TCLCL – T26	6TCLCL – T26	8TCLCL – T26	8TCLCL – T26	1, 4
tCWD	5TCLCL – T34	5TCLCL – T34	7TCLCL – T34	7TCLCL – T34	1
tWP	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	1
tRWL	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	1
tCWL	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	3TCLCL – T36 – TBUF	1

Table 18B. ECC Mode—RMW

Parameter	Slow Cycle Mode			Notes
	C ₄	C ₅	C ₆	
tRP	2TCLCL – T26	2TCLCL – T26	2TCLCL – T26	1
tCPN	2.5TCLCL – T35	2.5TCLCL – T35	2.5TCLCL – T35	1
tRSH	7TCLCL – T34	7TCLCL – T34	5TCLCL – T34	1
tCSH	7TCLCL – T26	7TCLCL – T26	5TCLCL – T26	1
tCAH	2TCLCL – T34	2TCLCL – T34	2TCLCL – T34	1
tAR	2TCLCL – T26	2TCLCL – T26	2TCLCL – T26	1
tT	3/30	3/30	3/30	2
tRWC	9TCLCL	9TCLCL	7TCLCL	1
tRRW	7TCLCL – T26	7TCLCL – T26	5TCLCL – T26	1
tCRW	7TCLCL – T34	7TCLCL – T34	5TCLCL – T34	1
tRCS	0.5TCLCL – T36 – TBUF	0.5TCLCL – T36 – TBUF	0.5TCLCL – T36 – TBUF	1
tRWD	4TCLCL + TCL – T26	4TCLCL + TCL – T26	2TCLCL + TCL – T26	1
tCWD	4TCLCL + TCL – T34	4TCLCL + TCL – T34	2TCLCL + TCL – T34	1
tWP	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	1
tRWL	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	1
tCWL	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	3TCLCL – TCL – T36 – TBUF	1

NOTES:

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



82C08 CHMOS DYNAMIC RAM CONTROLLER

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

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

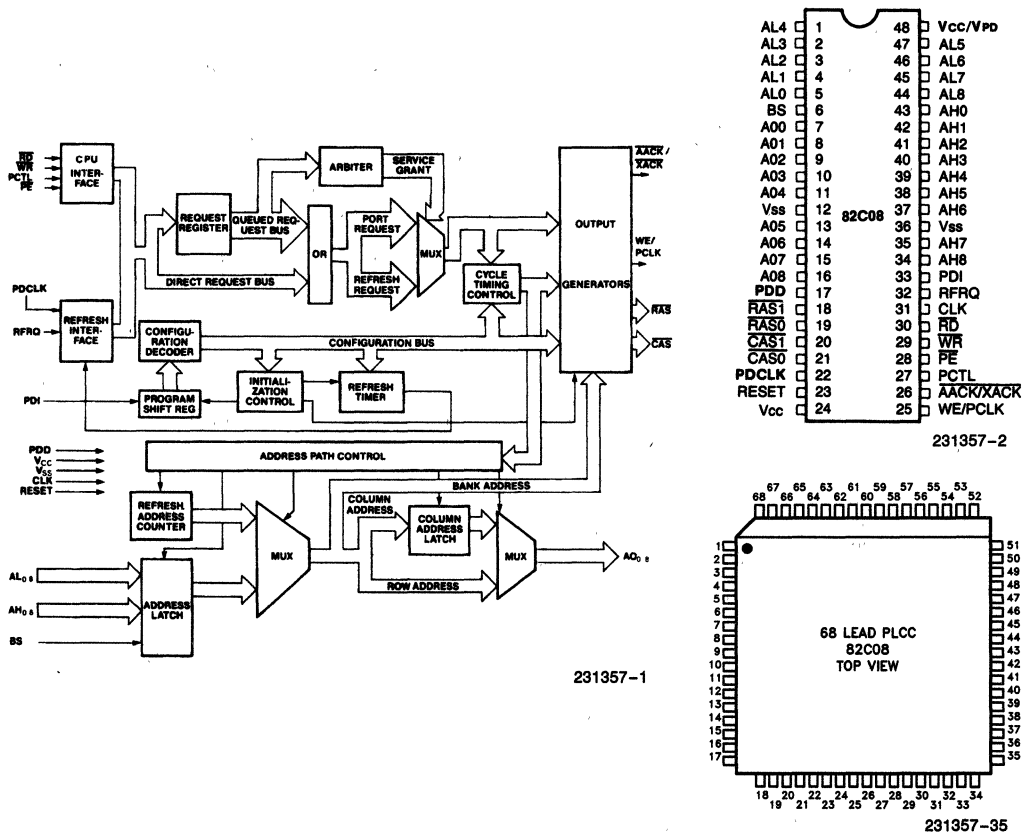


Figure 1. Block Diagram and Pinout Diagrams

Table 1. Pin Description

Symbol	DIP Pin	PLCC	Type	Name and Function
AL0 AL1 AL2 AL3 AL4 AL5 AL6 AL7 AL8	5 4 3 2 1 47 46 45 44	55 56 57 58 59 63 64 66 67	 	ADDRESS LOW: These lower order address inputs are used to generate the column address for the internal address multiplexer. In iAPX 286 mode (CFS = 1), these addresses are latched internally.
AH0 AH1 AH2 AH3 AH4 AH5 AH6 AH7 AH8	43 42 41 40 39 38 37 35 34	2 3 4 5 6 7 8 12 13	 	ADDRESS HIGH: These higher order address inputs are used to generate the row address for the internal address multiplexer. In iAPX 286 mode, these addresses are latched internally.
BS	6	50		BANK SELECT: This input is used to select one of the two banks of the dynamic RAM array.
AO0 AO1 AO2 AO3 AO4 AO5 AO6 AO7 AO8	7 8 9 10 11 13 14 15 16	49 48 47 46 45 41 40 39 38	O O O O O O O O O	ADDRESS OUTPUTS: These outputs are designed to provide the row and column addresses, of either the CPU or the refresh counter, to the dynamic RAM array. These outputs drive the dynamic RAM array directly and need no external drivers. However, they typically need series resistors to match impedances.
$\overline{\text{RAS}}_0$ $\overline{\text{RAS}}_1$	19 18	33 36	O O	ROW ADDRESS STROBE: These outputs are used by the dynamic RAM array to latch the row address, present on the AO0–8 pins. These outputs are selected by the BS pin. These outputs drive the dynamic RAM array directly and need no external drivers.
$\overline{\text{CAS}}_0$ $\overline{\text{CAS}}_1$	21 20	30 31	O O	COLUMN ADDRESS STROBE: These outputs are used by the dynamic RAM array to latch the column address, present on the AO0–8 pins. These outputs are selected by the BS pin. These outputs drive the dynamic RAM array directly and need no external drivers.
RESET	23	28		RESET: This active high signal causes all internal counters to be reset. Upon release of RESET, data appearing at the PDI pin is clocked-in by the PCLK output. The states of the PDI, PCTL, and RFRQ pins are sampled by RESET going inactive and are used to program the 82C08. An 8-cycle dynamic RAM warm-up is performed after clocking PDI bits into the 82C08.
WE/ PCLK	25	24	O	WRITE ENABLE/PROGRAMMING CLOCK: Immediately after a RESET this pin becomes PCLK and is used to clock serial programming data into the PDI pin. After the 82C08 is programmed this active high signal provides the dynamic RAM array the write enable input for a write operation.

Table 1. Pin Description (Continued)

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

Table 1. Pin Description (Continued)

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

*Different function than the HMOS 8208.

GENERAL DESCRIPTION

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

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

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

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

FUNCTIONAL DESCRIPTION

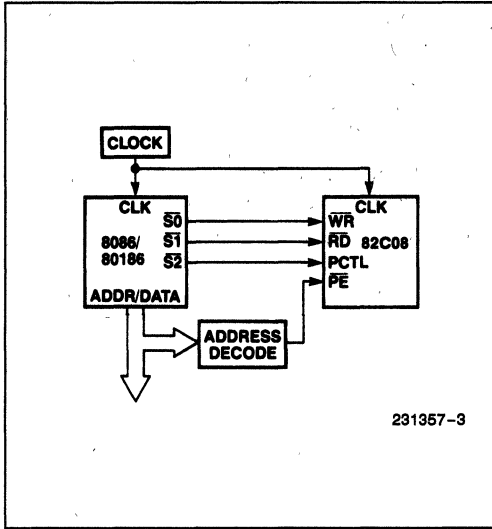
Processor Interface

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

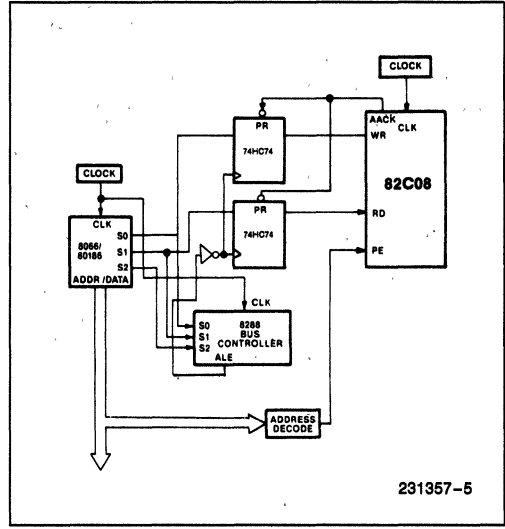
The 82C08 achieves high performance (i.e. no wait states) by decoding the status lines directly from the processor. The 82C08 can also be programmed to receive read or write MULTIBUS commands or commands from a bus controller.

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

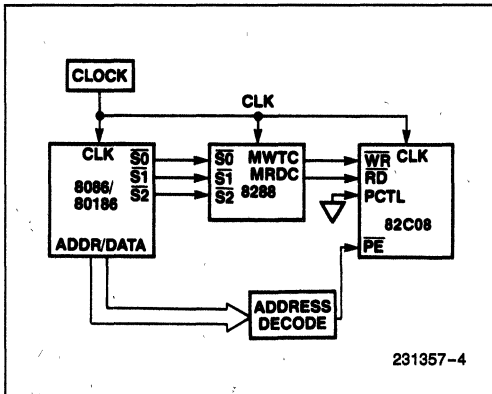
Figure 2 shows the different processor interfaces to the 82C08 using the synchronous or asynchronous mode and status or command interface. Figure 3 shows detailed interfaces to the iAPX 186 and iAPX 286 processors.



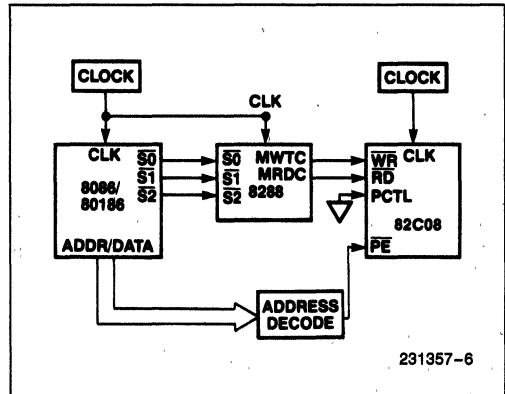
Slow-Cycle Synchronous-Status Interface



Slow-Cycle Asynchronous-Status Interface

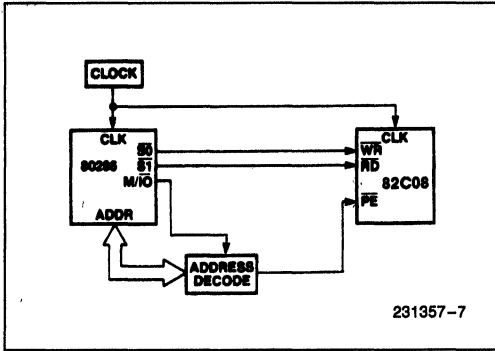


Slow-Cycle Synchronous-Command Interface

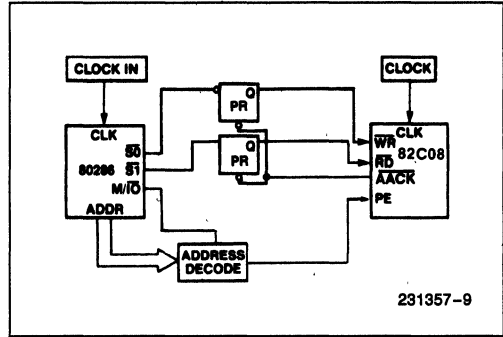


Slow-Cycle Asynchronous-Command Interface

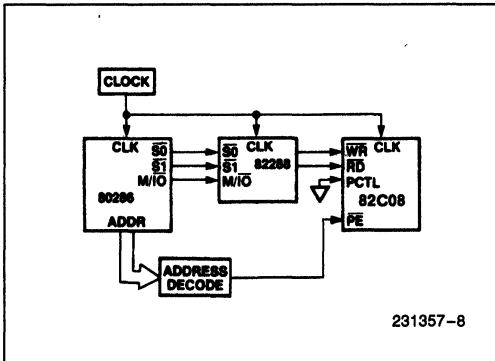
Figure 2A. Slow-cycle (CFS = 0) Port Interfaces Supported by the 82C08



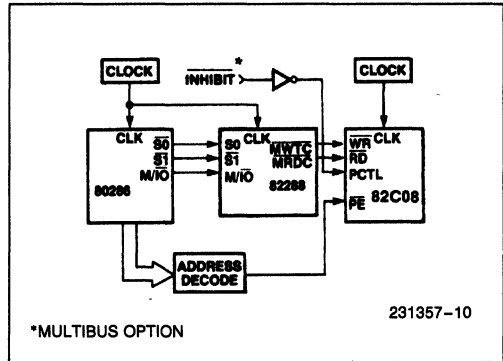
Fast-Cycle Synchronous-Status Interface



Fast-Cycle Asynchronous-Status Interface



Fast-Cycle Synchronous-Command Interface



*MULTIBUS OPTION

Fast-Cycle Asynchronous-Command Interface

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

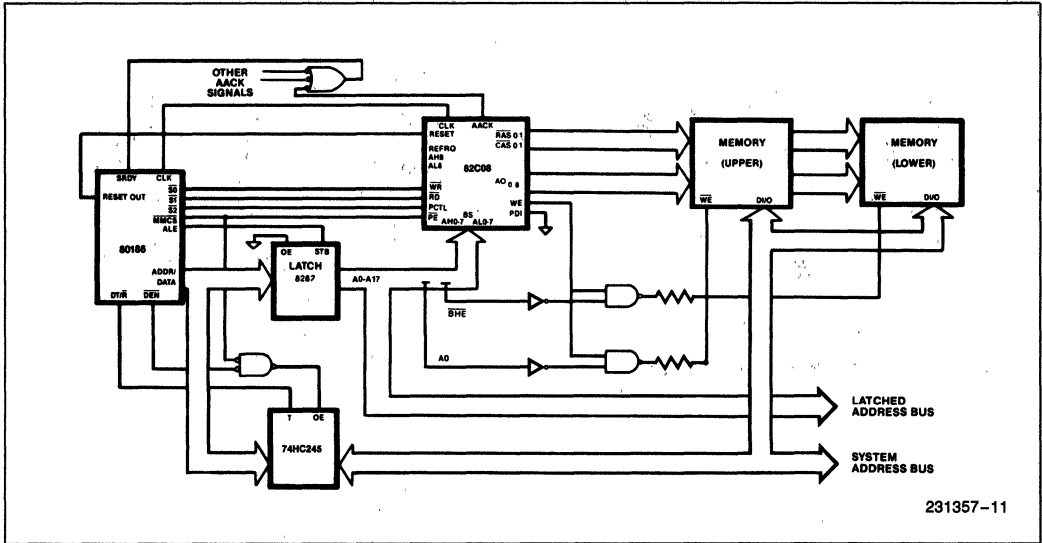


Figure 3A. 82C08 Interface to an 80186

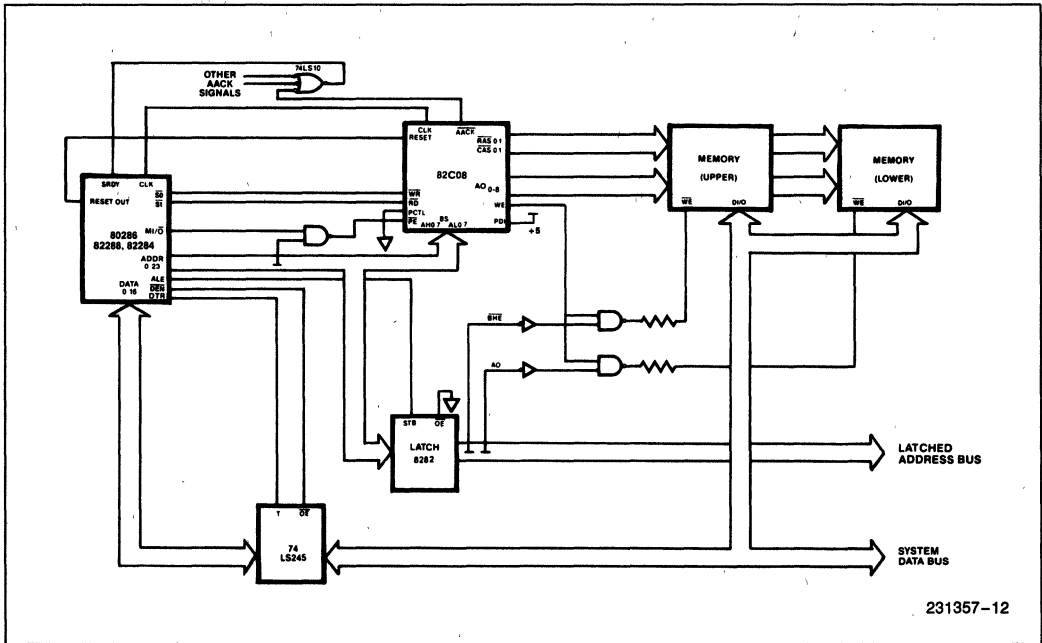


Figure 3B. 82C08 Interface to an 80286

Dynamic RAM Interface

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

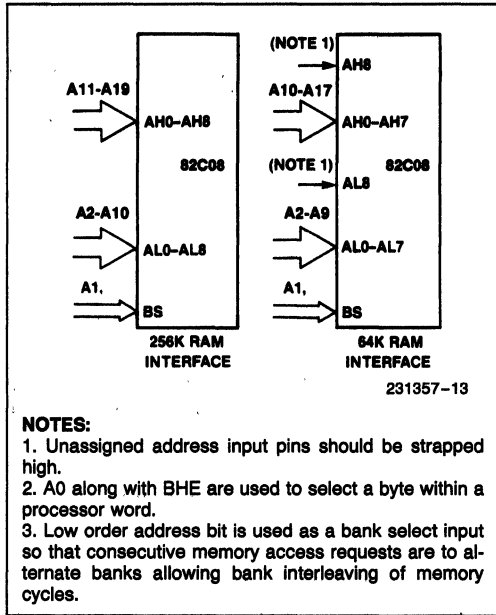


Figure 3. Processor Address Interface to the 82C08 Using 64K, and 256K RAMS

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

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

If not all RAM banks are occupied, the 82C08 can be programmed to reassign the \overline{RAS} and \overline{CAS} strobes to allow using wider data words without increasing the loading on the \overline{RAS} and \overline{CAS} drivers.

Table 2 shows the bank selection decoding and the corresponding \overline{RAS} and \overline{CAS} assignments. For example, if only one RAM bank is occupied, then the two \overline{RAS} and \overline{CAS} strobes are activated with the same timing.

Table 2. Bank Selection Decoding and Word Expansion

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

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

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

Memory Initialization

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

Refresh

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

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

It is possible to decrease the refresh time interval by 10%, 20% or 30%. This option allows the 82C08 to compensate for reduced clock frequencies. Note

that an additional 5% interval shortening is built-in in all refresh interval options to compensate for clock variations and non-immediate response to the internally generated refresh request. (See Refresh Period Options.)

External Refresh Requests after RESET

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

eg. 82C08 System Response:

$$TRESP = TPROG + TPREP$$

where: $TPROG = (40) (TCLCL)$ programming time

$$TPREP = (8) (32) (TCLCL) \text{ RAM warm-up time}$$

$$\text{if } TCLCL = 125 \text{ ns then } TRESP = 37 \mu\text{s}$$

Reset

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

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

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

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

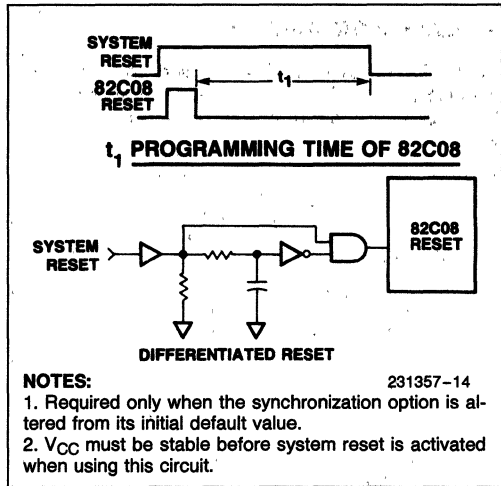


Figure 4. 82C08 Differentiated Reset Circuit

Within four clocks after RESET goes active, all the 82C08 outputs will go high, except for A00-2, which will go low.

OPERATIONAL DESCRIPTION

Programming the 82C08

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

Status/Command Mode

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

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

8088, 80186 or 80188 status, called the 8086 Status interface. The Command interface can also directly interface to the command lines of the bus controllers for the 8086, 8088, 80186 and the 80286.

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

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

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

* Refer to 80286 pin description table

Table 3B. 82C08 Response

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

*Illegal with CFS = 0

Refresh Options

Immediately after system reset, the state of the RFRQ input pin is examined. If RFRQ is high, the 82C08 provides the user with the choice between self-refresh and user-generated refresh with failsafe protection. Failsafe protection guarantees that if the user does not come back with another refresh request before the internal refresh interval counter times out, a refresh request will be automatically

generated. If the RFRQ pin is low immediately after a reset, then the user has the choice of a single external refresh cycle without failsafe, burst refresh or no refresh.

Internal Refresh Only

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

External Refresh with Failsafe

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

External Refresh without Failsafe

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

Burst Refresh

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

No Refresh

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

Option Program Data Word

PROGRAMMING FOR SLOW CYCLE

The program data word consists of 9 program data bits, PD0-PD8. If the first program data bit, PD0 is

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

PROGRAMMING FOR FAST CYCLE

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

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

Table 4A. Program Data Word

Program Data Bit	Name		Polarity/Function
	PD0 = 0	PD0 = 1	
PD0	CFS	CFS	CFS = 0 SLOW CYCLE CFS = 1 FAST CYCLE
PD1	\bar{S}	S	\bar{S} = 0 SYNCHRONOUS* \bar{S} = 1 ASYNCHRONOUS
PD2	\bar{RFS}	RFS	\bar{RFS} = 0 FAST RAM* RFS = 1 SLOW RAM
PD3	\bar{RB}	RB	RAM BANK OCCUPANCY SEE TABLE 2
PD4	CI1	$\bar{CI1}$	COUNT INTERVAL BIT 1; SEE TABLE 6
PD5	CI0	$\bar{CI0}$	COUNT INTERVAL BIT 0; SEE TABLE 6
PD6	PLS	PLS	PLS = 0 LONG REFRESH PERIOD* PLS = 1 SHORT REFRESH PERIOD
PD7	FFS	FFS	FFS = 0 FAST CPU FREQUENCY* FFS = 1 SLOW CPU FREQUENCY
PD8	X	\bar{X}	X = 0 $\bar{A}ACK^*$ X = 1 $XACK$

* Default in both modes

Using an External Shift Register

The 82C08 may be programmed by using an external shift register with asynchronous load capability

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

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

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

Default Programming Options

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

Table 4B. Default Programming

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

NOTE:

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

Synchronous/Asynchronous Mode (S program bit)

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

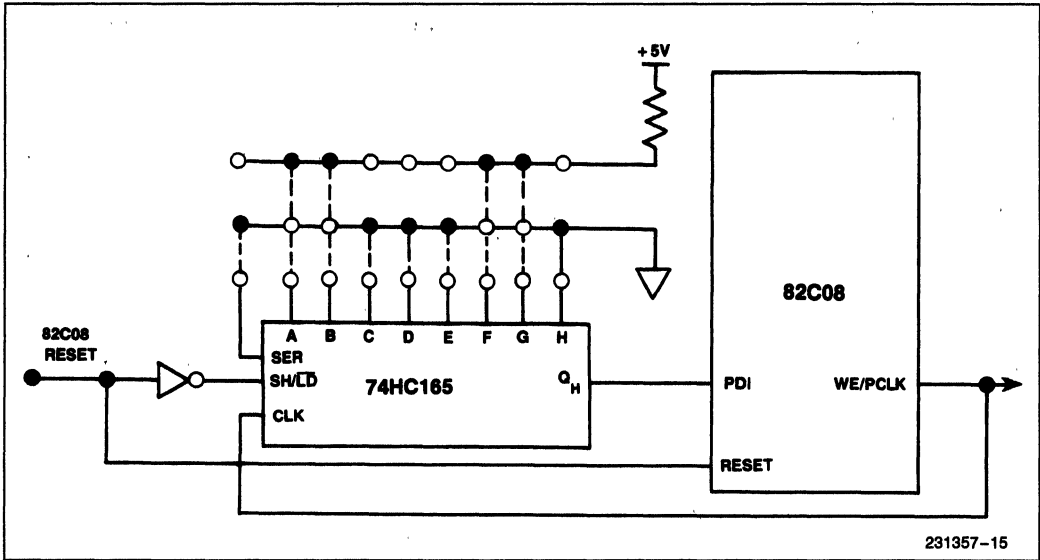
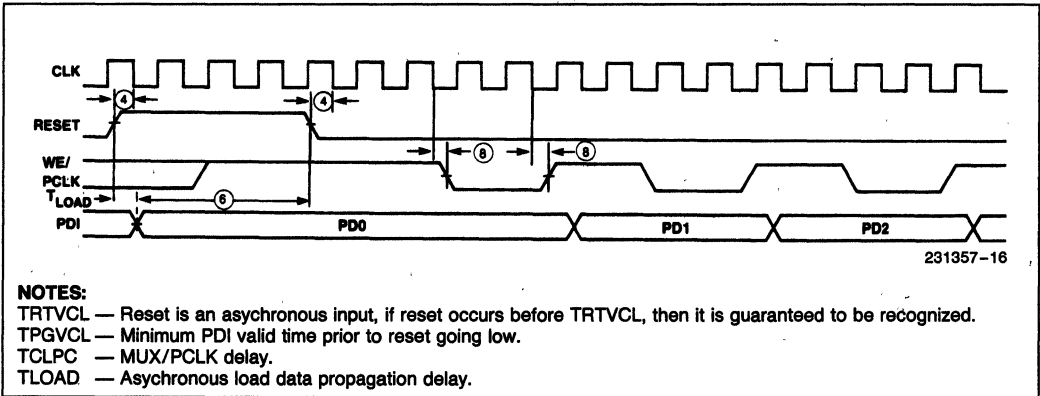


Figure 6. External Shift Register Interface



NOTES:

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

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

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

Microprocessor Clock Cycle Option (CFS and FFS program bits)

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

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

programmed so that the failsafe refresh repetition circuitry can adjust its internal timing accordingly to produce a refresh request as programmed.

Table 5. Microprocessor Clock Frequency Options

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

RAM Speed Option (RFS program bit)

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

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

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

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

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

Table 6. Refresh Count Interval Table

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

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

- 5 MHz slow cycle
- 8 MHz slow cycle
- 10 MHz fast cycle
- 16 MHz fast cycle

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

$$\text{clock period} \times \text{refresh count interval} = \text{refresh period}$$

$$\text{i.e. } 83.3 \text{ ns} \times 164 = 13.6 \mu\text{s}$$

Example: 10 MHz slow cycle

$$\text{CFS} = 0, \text{FFS} = 1, \text{CI1} = 0, \text{CI0} = 0$$

$$\text{i.e. } 100 \text{ ns} \times 118 = 11.8 \mu\text{s}$$

Processor Timing

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

Port Enable ($\overline{\text{PE}}$) set-up time requirements depend on whether the 82C08 is configured for synchronous

or asynchronous, fast or slow cycle operation. In a synchronous fast cycle configuration, \overline{PE} is required to be set-up to the same clock edge as the commands. If \overline{PE} is true (low), a RAM cycle is started; if not, the cycle is not started until the \overline{RD} or \overline{WR} line goes inactive and active again.

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

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

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

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

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

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

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

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

General System Considerations

1. The $\overline{RAS0}$, 1, $\overline{CAS0}$, 1, and AO0-8 output buffers are designed to directly drive the heavy capacitive loads of the dynamic RAM arrays. To keep the RAM driver outputs from ringing excessively in the system environment it is necessary to match the output impedance with the RAM array by using series resistors. Each application may have different impedance characteristics and may require different series resistance values. The series resistance values should be determined for each application.
2. Although the 82C08 has programmable options, in practice there are only a few choices the designer must make. For iAPX 86/186 systems (CFS = 0) the C2 default mode (pin 33 tied low) is the best choice. This permits zero wait states at 8 and 10 MHz with 150 ns DRAMs. The only consideration is the refresh rate, which must be programmed if the CPU is run at less than 8 MHz.
For iAPX 286 systems (CFS = 1) the designer must choose between configuration C0 (RFS = 0) and C1 (RFS = 1, FFS = 0). C0 permits zero wait state, 8 MHz iAPX 286 operation with 120 ns DRAMs. However, for consecutive reads, this performance depends on interleaving between two banks. The C1 configuration trades off 1 wait state performance for the ability to use 150 ns DRAMs. 150 ns DRAMs can be supported by the C0 configuration using 7 MHz iAPX 286.
3. For non-Intel microprocessors, the asynchronous command mode would be the best choice, since Intel status lines are not available. To minimize the synchronization delay, the 82C08 should use a 16 MHz clock. The preferred timing configuration is C0.

Table 7. Memory Acknowledge Summary

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

POWER DOWN

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

Power Supplies

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

Extended Refresh at Power Down (PD)

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

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

The option of refreshing all the memory is enabled in failsafe mode configuration (RFRQ input high at reset). When 82C08 detects power down, (high level at PDD) it examines the RFRQ input. High level at the RFRQ input will cause 3 PD burst refresh cycles to be performed. The user should supply the power and the system clock during the time interval of the 3 PD burst cycles, e.g. 4700 (fast cycle) or 3100 (slow cycle) clock cycles after activating PDD. Low level at RFRQ input enables the 82C08 to enter

power down immediately without executing any bursts.

Power Down Procedure

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

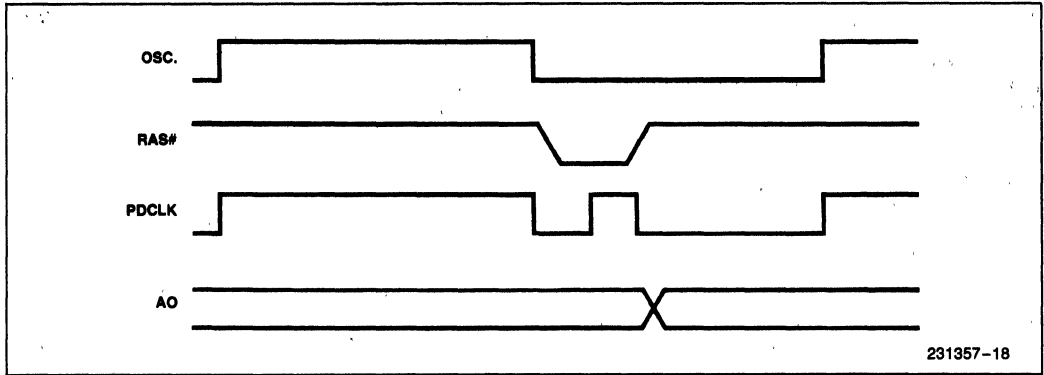
When power is up again (PDD input deactivated), the 82C08 will issue internal reset which will not re-program the device and will not clear the refresh address counter, and therefore, refresh will continue to the next address location. After the internal reset, 82C08 performs 3 PD burst refresh cycles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

82C08 Outputs on Power Down

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

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

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



231357-18

Figure 8

Power Down Detect

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

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

NOTE:

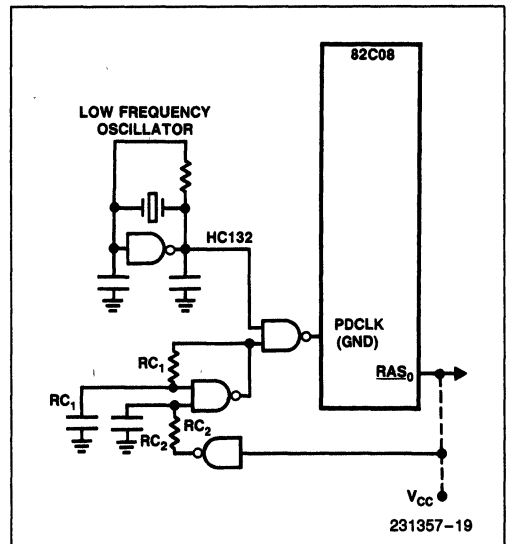
The power supplies and the CLK should go up before the PDD is deactivated. All CPU requests will be ignored when PDD is active.

Refresh during Power Down

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

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

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



231357-19

Figure 9. Low Frequency Oscillator

Power Down Synchronization

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

When entering power down mode, and the whole memory refresh is required, the CLK minimum active

time after PDD is activated is 4700 (fast cycle) or 3100 (slow cycle) clocks.

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

$$20 + \lceil \text{CLK}(\text{MHz}) / \text{PDCLK}(\text{MHz}) + 1 \rceil \text{ clock cycles}$$

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

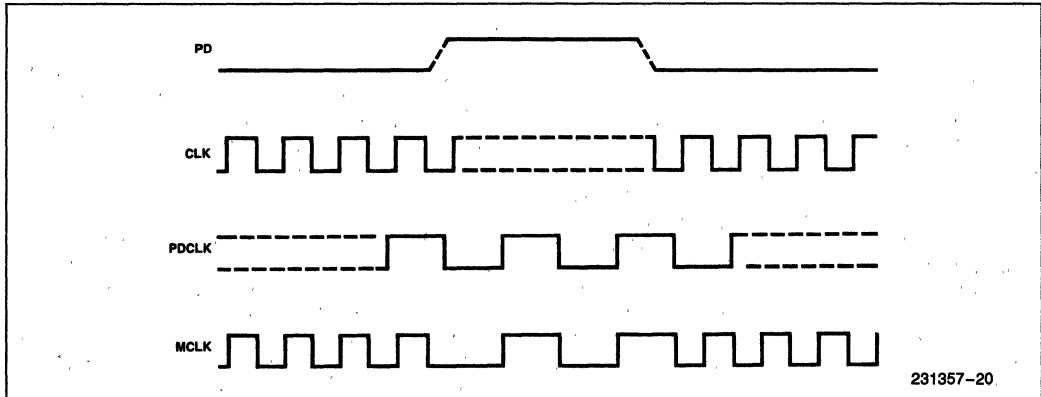


Figure 10

POWER DOWN FLOW

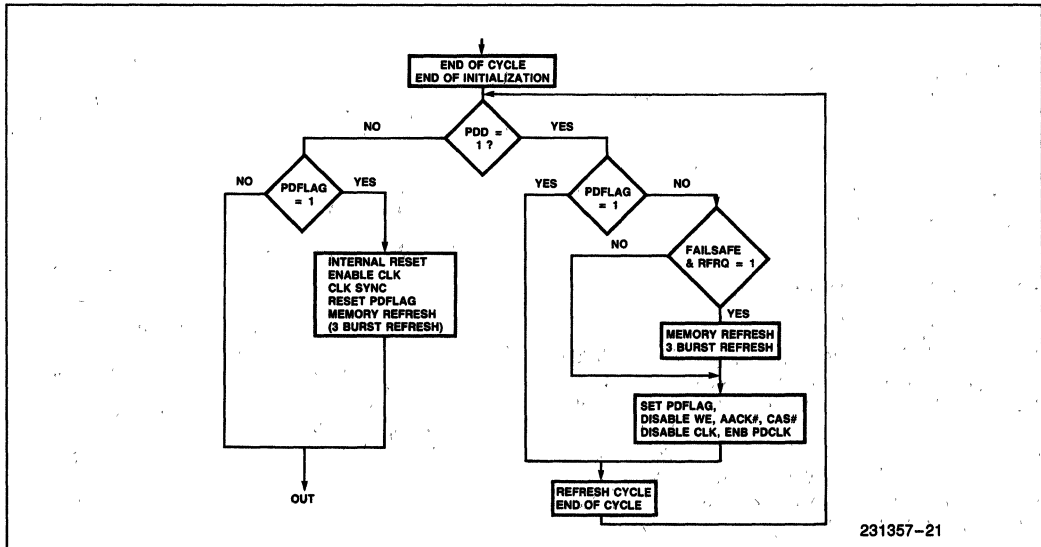


Figure 11

Differences Between 8208 and 82C08

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

LOGICAL DIFFERENCES

- 82C08 has one new feature:
Power Down (PD)
- 82C08 supports CMOS DRAMs with T_{RAC} 100, 150
- Address Mapping:

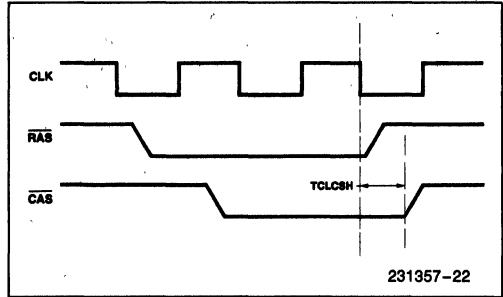
Outputs	9 Most Significant Bits	9 Least Significant Bits
8208	column address	row address
82C08	row address	column address

- Slow cycle shortening:
 - The write cycle is two clocks shorter so consecutive writes will be executed without wait states.
 - The WE output is two clocks shorter. Therefore, an external latch on the WE output is not necessary.
 - CAS output is shorter by one clock on the read cycle. This reduces one level of buffers for address/data bus needed in 8208 designs. Read access margins are improved to support non-Intel spec. RAMs.
 - The address outputs switch from row to column address one clock cycle later in the 82C08 as compared to 8208.
- Fast cycle shortening:
 - The write cycle in C0 configuration is shortened by one clock.
 - For both C0 and C1 synchronous configuration, the CAS signal is shorter by one clock and the activation of RAS is tied to the Φ_2 cycle of the 80286. This prevents contention on the data bus.
- Supports Static Column or Ripplemode DRAMs.

ELECTRICAL DIFFERENCES

- AC parameters:
 - CAS delay: In C2 synchronous read cycle, the CAS is deactivated by some delay from clock falling edge (TCLCSH timing) as in the following diagram:

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



- 82C08 has an additional timing parameter TARH column address to RAS \uparrow hold time.
- DC parameters: The difference is in the current consumption.

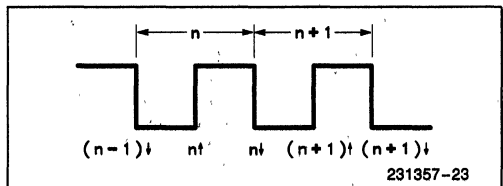
	8208	82C08
I_{CC}	300 mA	30 mA (typical) [10 + 2f] mA (max)
IPD	—	1 mA (max)
I_{SB}	—	2 mA (max)

Configuration Charts

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

Using the Timing Charts

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



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

Clock 0 is defined as the clock in which the 82C08 begins a memory cycle, either as a result of a port request which has just arrived, or of a port request which was stored previously but could not be serv-

iced at the time of its arrival because the 82C08 was performing another memory cycle. Clock 0 is identified externally by the leading edge of RAS, which is always triggered on 0 ↓.

Table 8. 82C08 Configurations

Timing Conf.	CFS(PD0)	RFS(PD2)	FFS(PD7)	Wait States*
C ₀	iAPX286(1)	FAST RAM(1)	20 MHz(1)	0
C ₀	iAPX286(1)	FAST RAM(1)	16 MHz(1)	0
C ₁	iAPX286(1)	SLOW RAM(0)	16 MHz(1)	1
C ₀	iAPX286(1)	FAST RAM(1)	10 MHz (0)	0
C ₀	iAPX286(1)	SLOW RAM(0)	10 MHz (0)	0
C ₂	iAPX186(0)	DON'T CARE	DON'T CARE	0

* Using EAACK (synchronous mode)

Table 9a. Timing Chart — Synchronous Mode

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

Table 9b. Timing Chart — Asynchronous Mode

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

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

NOTES FOR INTERPRETING THE TIMING CHART:

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

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

82C08—DRAM Interface Parameter Equations

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

WRITE CYCLE

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

READ, WRITE REFRESH CYCLES

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

Table 10. Programmable Timings

Read and Refresh Cycles

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

Write Cycles

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

NOTES:

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

ABSOLUTE MAXIMUM RATINGS*

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

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

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}; V_{CC} = 5.0\text{V} \pm 10\%; V_{SS} = \text{GND}$

Symbol	Parameter	Min	Max	Units	Comments
V_{IL}	Input Low Voltage	-0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage		0.45	V	(Note 1)
V_{OH}	Output High Voltage	2.6		V	(Note 1)
I_{CC}	Supply Current		$10 + 2f$	mA	(Note 3)
I_{LI}	Input Leakage Current		± 10	μA	$0\text{V} \leq V_{IN} \leq V_{CC}$
V_{CL}	Clock Input Low Voltage	-0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.8	$V_{CC} + 0.5$	V	
C_{IN}	Input Capacitance		20	pF	$f_c = 1 \text{ MHz}^{(6)}$
V_{OHPD}	RAS Output High Power Down	$V_{CC} - 0.5$		V	(Note 2)
I_{PD}	Power Down Supply Current	—	1.0	mA	(Note 5)
I_{SB}	Standby Current	—	2.0	mA	Estimated Value ⁽⁴⁾

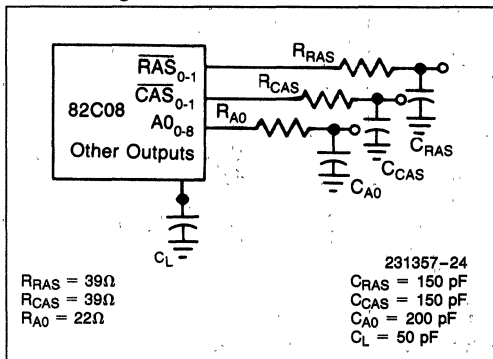
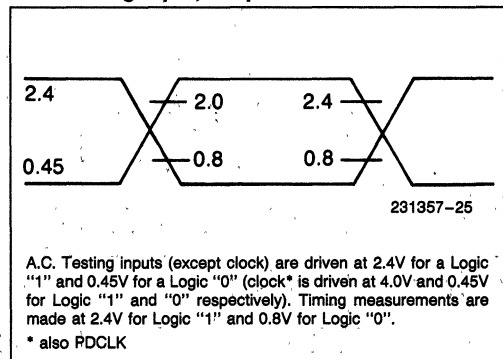
NOTE:

- $I_{OL} = 5 \text{ mA}$ and $I_{OH} = -0.32 \text{ mA}$ WE: $I_{OL} = 8 \text{ mA}$
- RAS Output voltage during power down.
- Typical value. Where f is freq. in MHz. $T_A = 0^\circ\text{C}$ for CMOS: $V_{IL} \text{ max} = 0.5\text{V}$; $V_{IH} \text{ min} = (V_{CC} - 0.5\text{V})$ for TTL: I_{CC} will be higher by 30 mA

4. Measured at $V_{IL} = 0\text{V}$ or $V_{IH} = V_{CC}$ with no loads connected.

5. $I_{PD} = 1 \text{ mA}$ at 32 KHz with no loads connected.

6. Sampled, not 100% tested.

A.C. Testing Load Circuit

A.C. Testing Input, Output Waveform


A.C. CHARACTERISTICS ($T_A = 0^\circ\text{C}$ to 70°C ; $V_{CC} = +5\text{V} \pm 10\%$, $V_{SS} = 0\text{V}$)

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

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

A.C. CHARACTERISTICS (Continued)

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

A.C. CHARACTERISTICS (Continued)

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

A.C. CHARACTERISTICS (Continued)

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

The following RC loading is assumed:

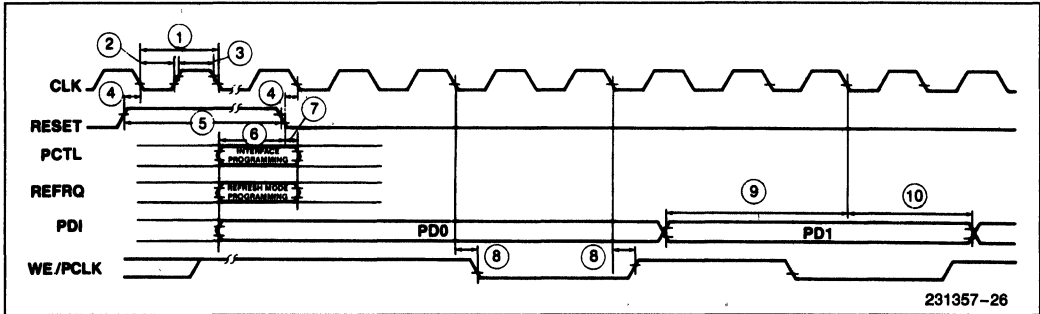
$A0_{0-8}$ $R = 22\Omega$ $C = 200$ pF
 RAS_{0-1}, CAS_{0-1} $R = 39\Omega$ $C = 150$ pF
 $AACK, WE/PCLK$ $C = 50$ pF

NOTES:

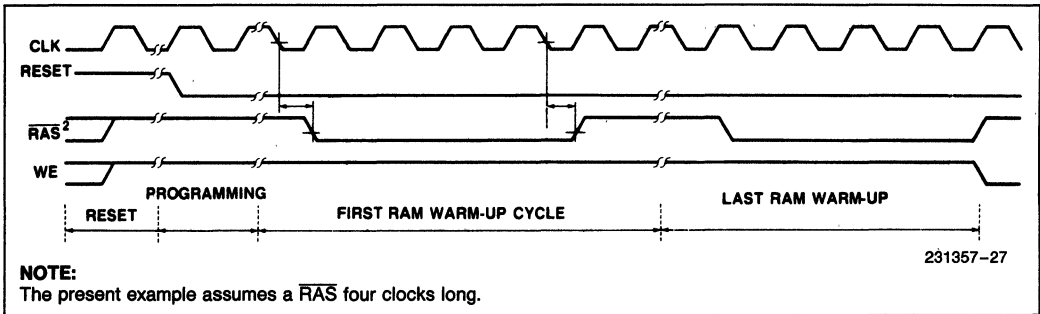
1. Specification when programmed in the Fast Cycle processor mode (iAPX 286 mode). 82C08-20, -16.
2. Specification when programmed in the Slow Cycle processor mode (iAPX 186 mode). 82C08-10, 82C08-8.
3. tR and tF are referenced from the 3.5V and 1.0V levels.
4. RESET is internally synchronized to CLK.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

WAVEFORMS

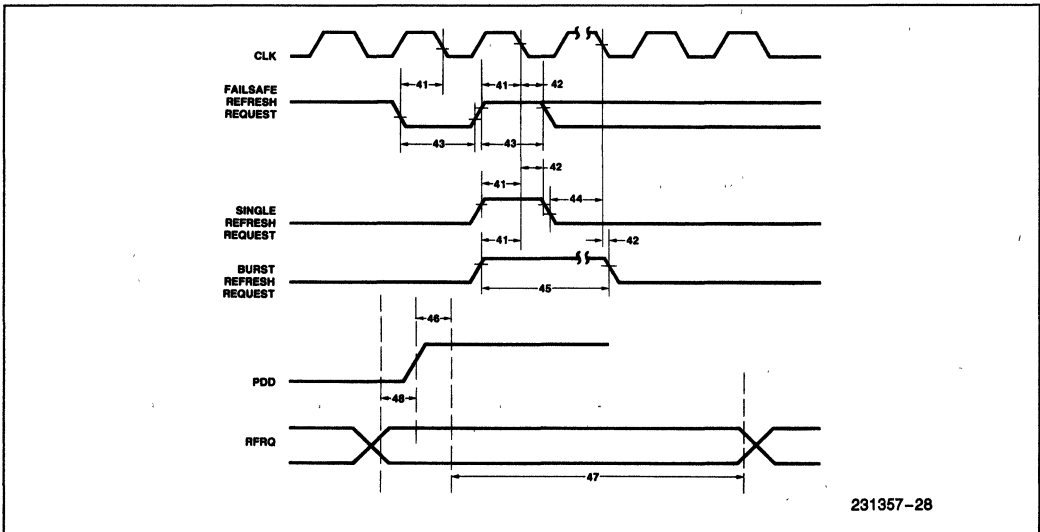
CLOCK AND PROGRAMMING TIMINGS



RAM WARM-UP CYCLES

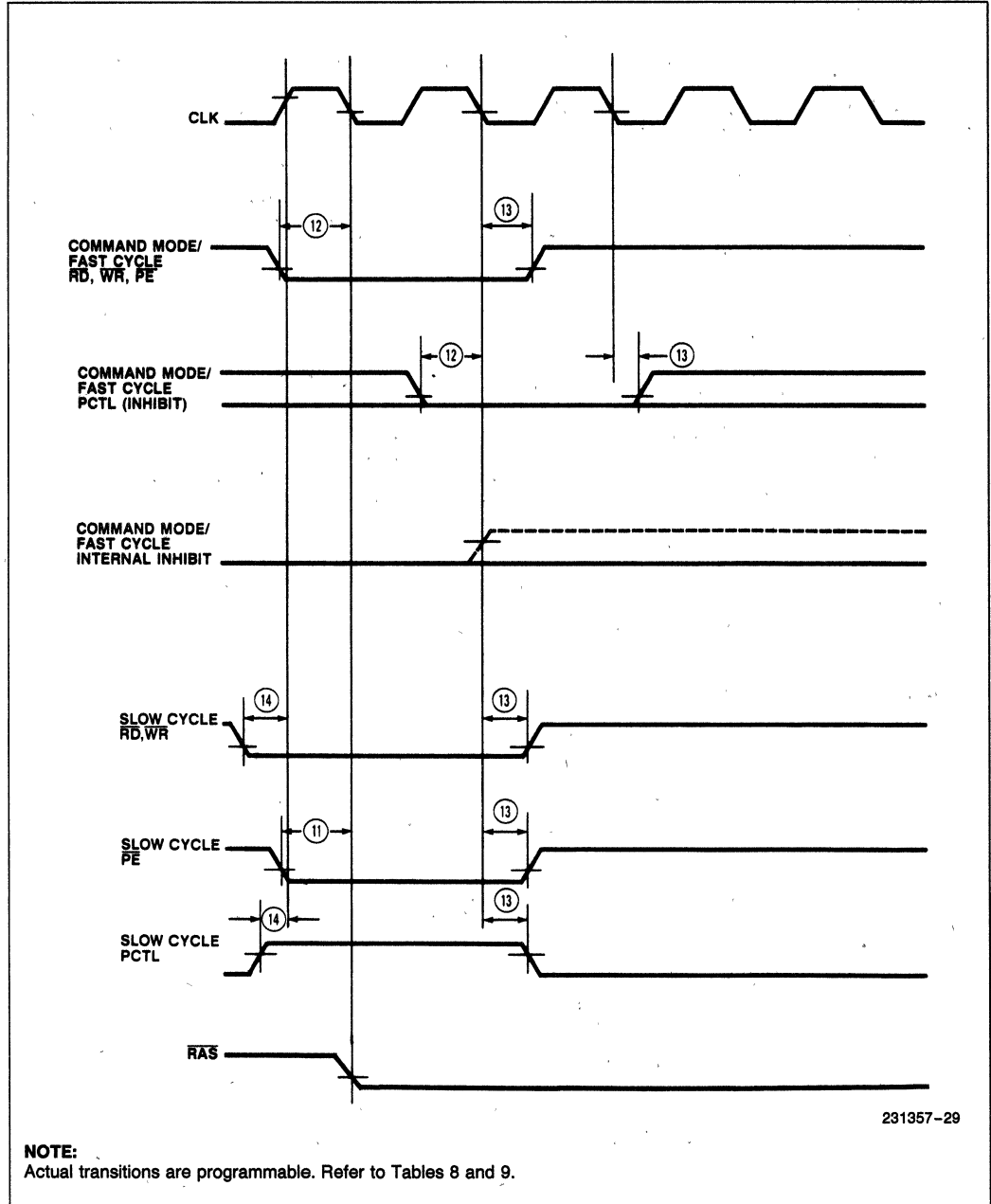


REFRESH REQUEST TIMING



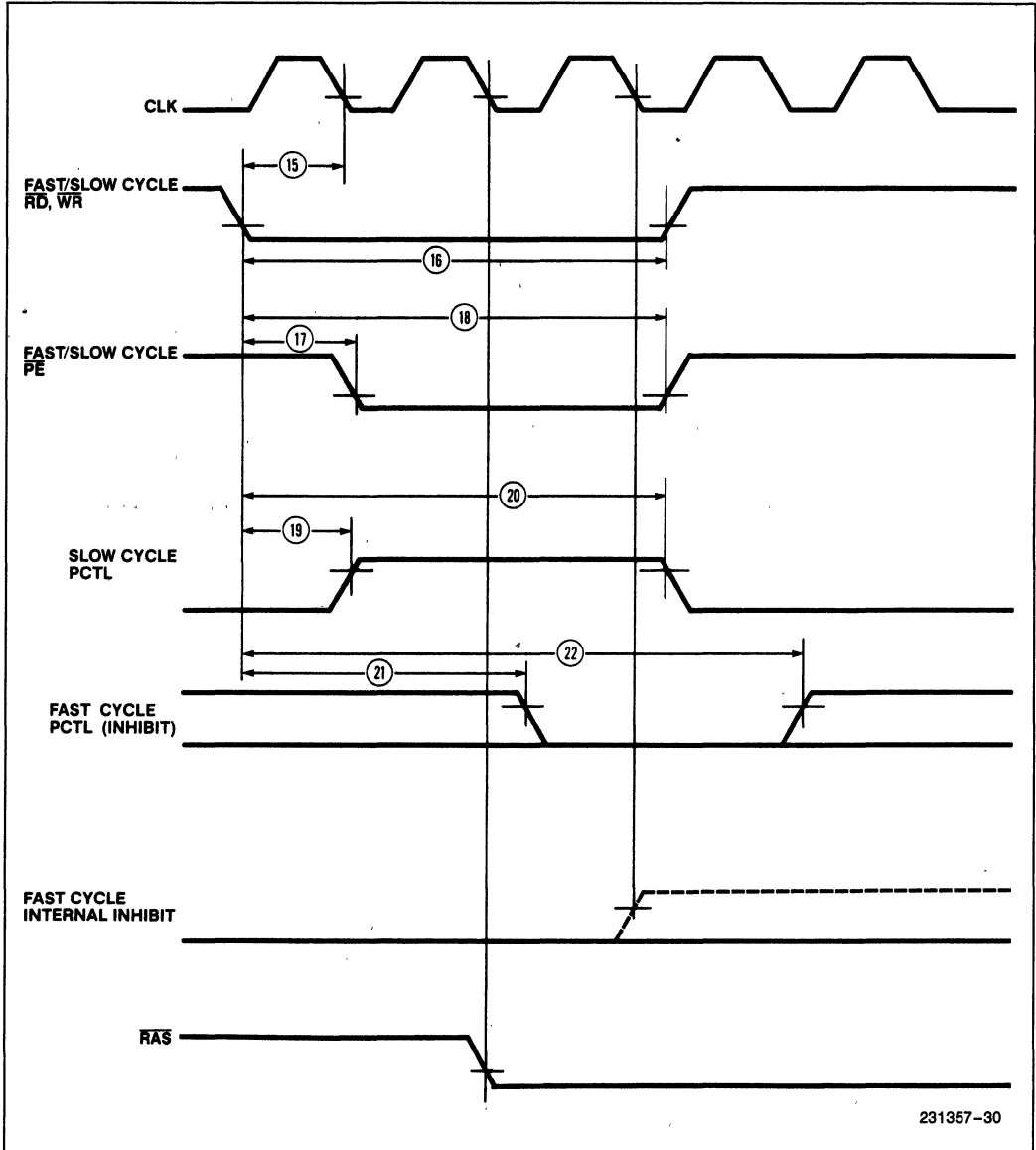
WAVEFORMS (Continued)

SYNCHRONOUS PORT INTERFACE



WAVEFORMS (Continued)

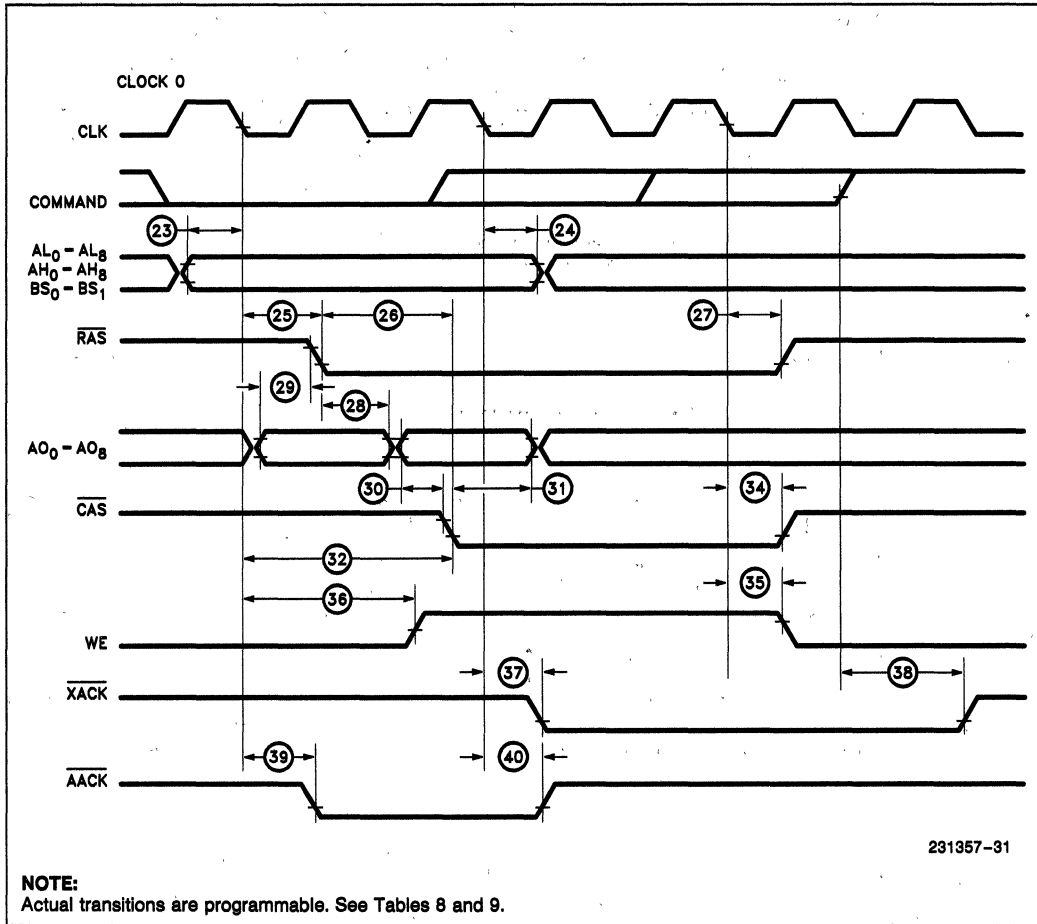
ASYNCHRONOUS PORT INTERFACE



231357-30

WAVEFORMS (Continued)

RAM INTERFACE TIMING





November 1986

Interfacing the 8207 Dynamic RAM Controller to the iAPX 186

JIM SLEEZER
APPLICATION ENGINEER

Order Number: 230809-001

INTRODUCTION

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

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

Integrating all of the above functions (plus a dual port and error correcting interfaces) allows the user to realize significant cost savings over discrete logic. For example, comparing the 8207 to the iSBC012B 512K byte RAM board (where the DRAM control is done entirely with TTL), an 8207 design saved board space 3 in² vs 10 in²); required less power (420 mA vs 1220 mA); and generated less heat. Moreover, design time was reduced, and increased margins were achieved due to less skewing of critical timings. This comparison is based on a single port design and did not include the 8207's RAM warm-up, dual-port and error correcting features. If these features were fully implemented, there would be no change to the 8207 figures, listed above, while the TTL figures would easily double.

This Application Note will illustrate an iAPX design with the 8207 controlling the dynamic RAM array.

The reader should be familiar with the 82097 data sheet, the 80186 data sheet, and a RAM data sheet*.

DESIGN GOALS

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

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

USING THE 8207

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

- 8207 programming logic
- Microprocessor interface
- RAM array

8207 Programming

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

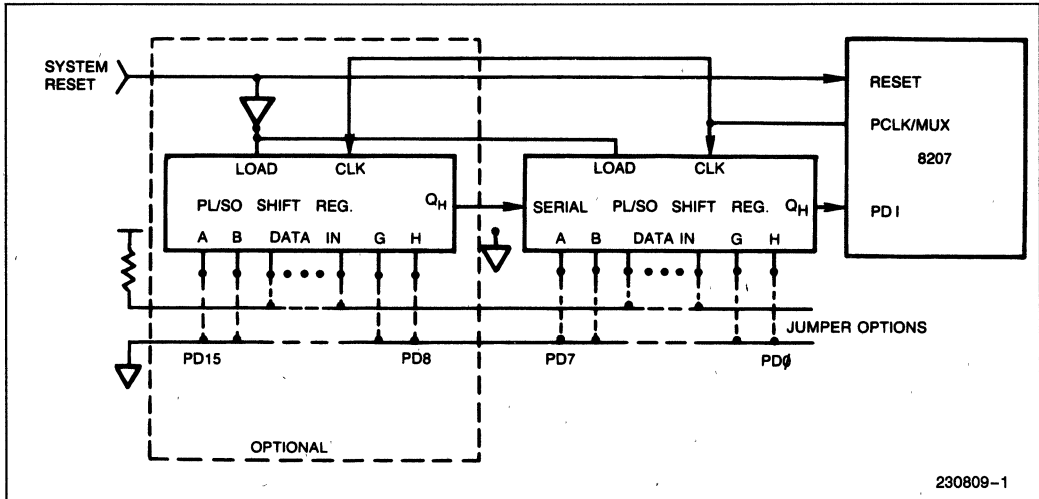


Figure 1. 8207 Programming Shift Registers

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

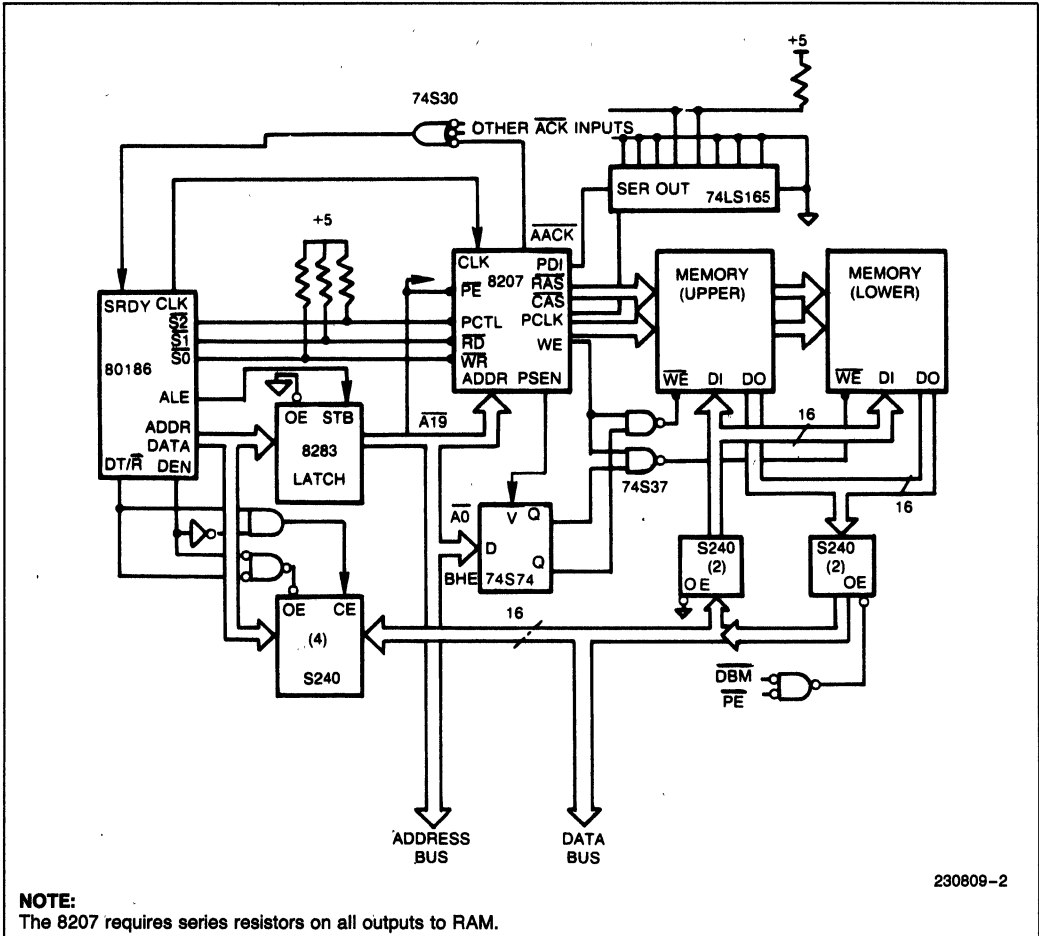


Figure 3. 80186 to 8207, non-ECC, synchronous system single port

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

Acknowledge Setup Time

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

$$1 \text{ clock} - 8207 \text{ TCLAKL max} - 74S30 \text{ tPLH @ } 15 \text{ pf} - 80186 \text{ TSTRYCL} \geq 0$$

$$125 \text{ ns} - 35 - 22 - 35 = 33 \text{ ns}$$

Read Access Margin

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

$$2 \text{ TCLCL} - 8207 \text{ TCLCSL @ } 150 \text{ pF (t34)} - \text{DRAM tCAC} - 74S240 \text{ propagation delay @ } 50 \text{ pF} - \text{additional bus loading delay (250 pF)}^{(1)} - 74S240 \text{ delay @ } 50 \text{ pF} - 80186 \text{ TDVCL} \geq 0$$

$$250 \text{ ns} - 122 - 85 - 7 - 7 - 7 - 20 = 2 \text{ ns}$$

NOTE:

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

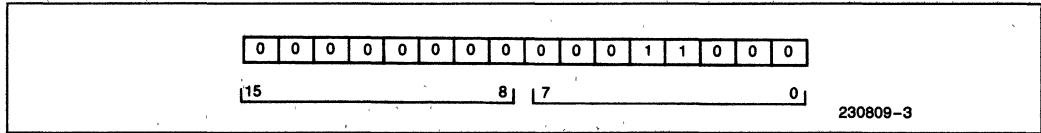


Figure 2. Program Data Word

Write Data Setup and Hold Margin

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

The PCTLA input must be high when RESET goes inactive.

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

Microprocessor Interface

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

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

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

RAM Array

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

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

HARDWARE DESIGN

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

8207 Command Setup

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

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

$$80186 \text{ status valid to } 8207 \text{ rising clock} - \text{status from clock delay} - 8207 \text{ setup to clock} \geq 0$$

$$1 \text{ TCLCL} - 80186 \text{ TCHSV max} - 8207 \text{ TKVCH min} \geq 0$$

$$125 \text{ ns} - 55 - 20 = 50 \text{ ns}$$

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

$$80186 \text{ address valid to } 8207 \text{ falling clock edge} - 80186 \text{ address from clock delay} - 8283 \text{ latch delays} - 8207 \text{ PE setup} \geq 0$$

$$1 \text{ TCLCL} - 80186 \text{ TCLAV max} - 8283 \text{ IVOV @ } 300 \text{ pF} - 8207 \text{ TPEVCL} \geq 0$$

$$125 \text{ ns} - 44 - 22 - 30 - 29 \text{ ns}$$

The hold times are 0 ns and are met.

Address Setup

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

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

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

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

$$125 \text{ ns} + 0 - 44 - 22 - (35 + 0) = 24 \text{ ns}$$

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

NOTE:

(1) Not specified—use 0 ns.

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

$$125 + 62.5 + 0 + 6.5 + 3.5 - 70 - 15 - 7 - 7 - 0 = 98.5 \text{ ns.}$$

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

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

$$62.5 \text{ ns} + 10 + 2 + 3 + 7 + 3.5 - 35 - 3.5 - 30 = 19.5 \text{ ns}$$

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

SUMMARY

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

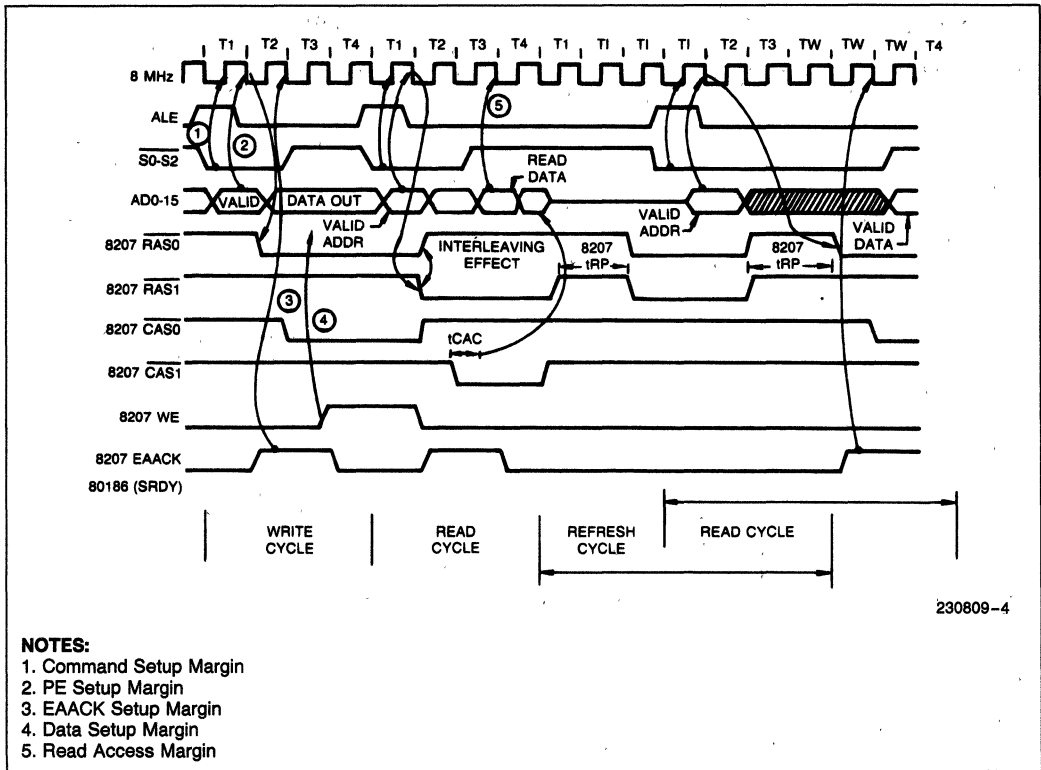


Figure 4. 8207/80186 Timing Relationship

NOTE:

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



November 1986

Interfacing the 8207 Advanced Dynamic RAM Controller to the iAPX 286

JIM SLEEZER
APPLICATION ENGINEER

Order Number: 230862-001

INTRODUCTION

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

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

Integrating these functions (plus dual port and error correcting interfaces) allows the user to realize significant savings in both engineering design time, PC board space and product cost. For example, in comparing the 8207 to the ISBC012B 512k byte RAM board (where the DRAM timing and control is done entirely with TTL), the 8207 design saved board space (3 in² vs 10 in²); used less power (420 mA vs 1220 mA); reduced the design time; and increased margins due to less skewing of timings. The comparison is based upon a single port 8207 design and does not include its RAM warm-up, dual port, error correcting, and error scrubbing or RAM interleaving features.

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

DESIGN GOALS

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

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

8207 INTERFACE

The 8207 Memory design can be subdivided into three sections:

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

Programming the 8207

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

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

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

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

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

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

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

RAM Array

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

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

Microprocessor Array

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

LEN will fall from high to low, which latches the bus address internally, when a valid command is received. LEN can go high in two clock cycles if the RAM cycle started (RAS going low) at the same time LEN went low. If the 8207 is doing a refresh cycle, the 80286 will be put into wait states until the memory cycle can start.

LEN will then go high two clocks after RAS starts, since addresses are no longer needed for the current RAM cycle. Thus the low period of LEN could be much longer than listed in the Data Sheet.

DESIGNING THE HARDWARE

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

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

8207 Command Setup Margin

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

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

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

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

62.5 – 40 ns – 20 ns = 2.5 ns

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

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

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

125 – 60 – 30 = 35 ns

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

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

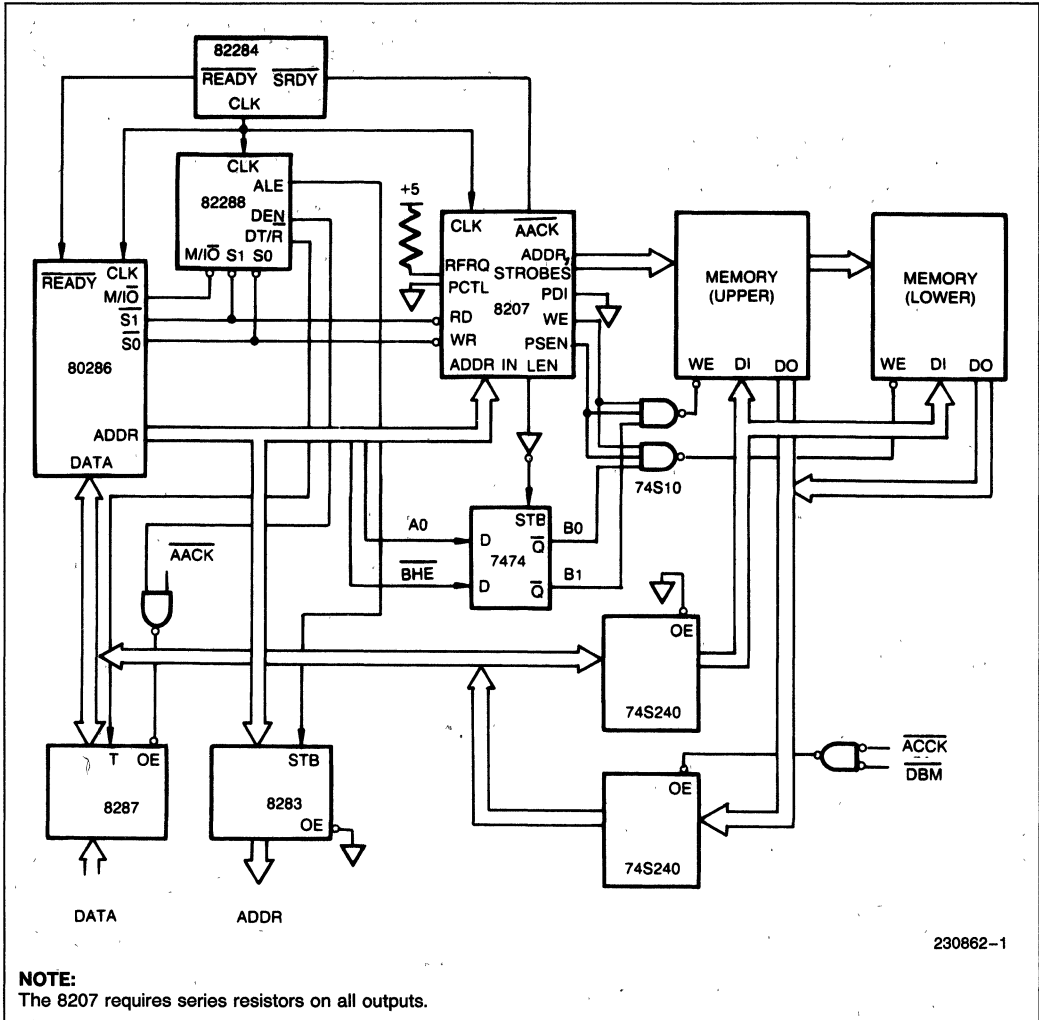


Figure 1. 80286 to 8207, non-ECC, Synchronous System Single Port

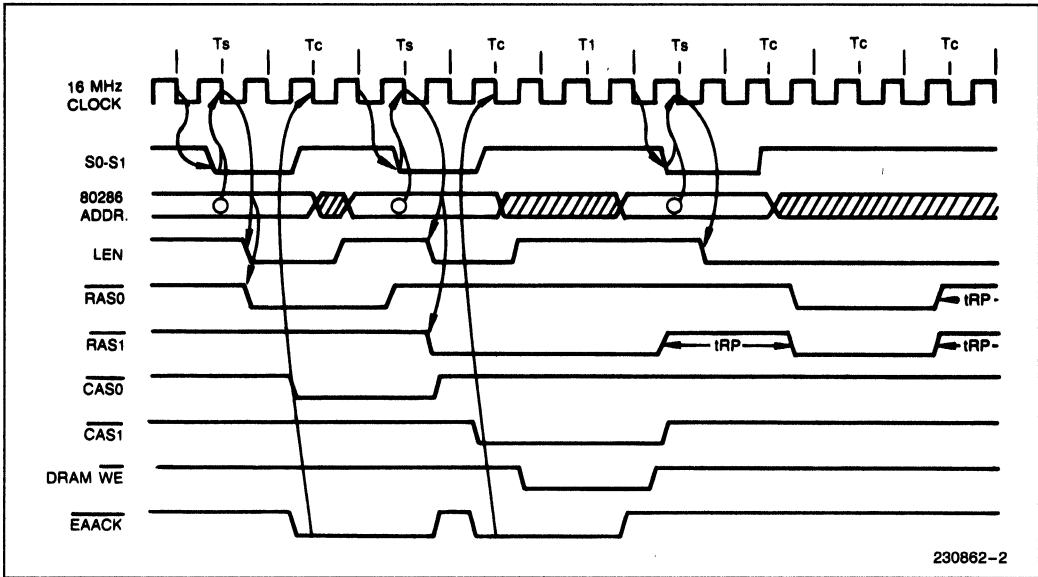


Figure 2. 80286/8207 Timing—"CO"

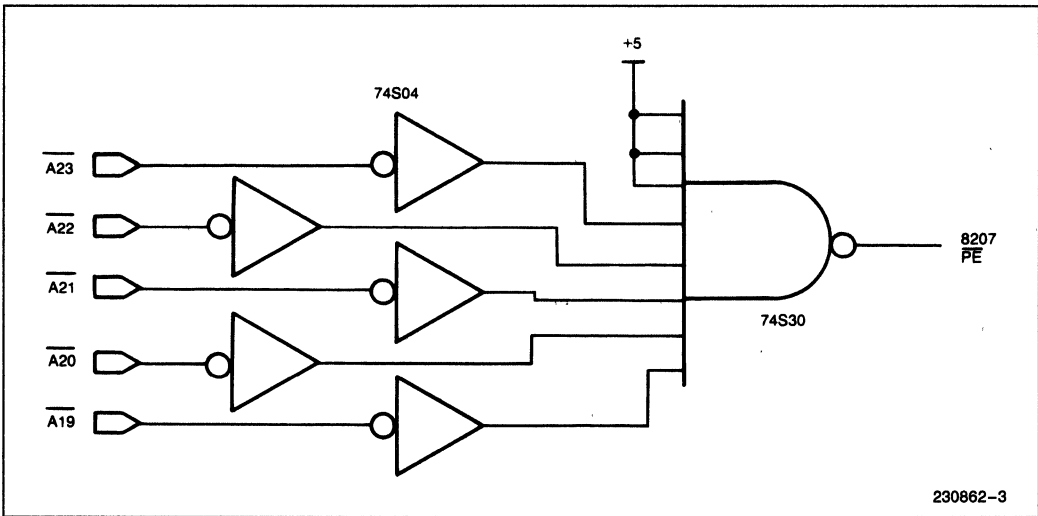


Figure 3. Address Decode Logic

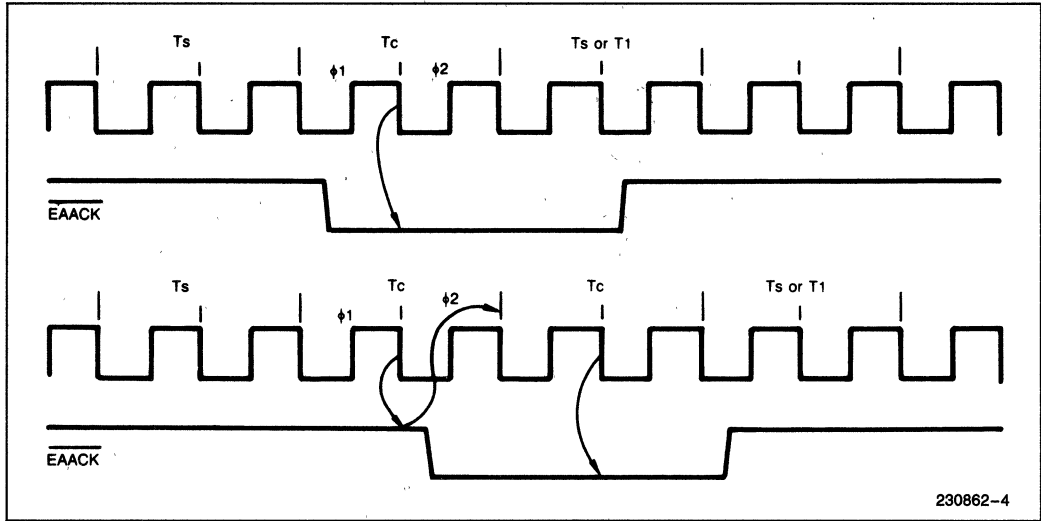


Figure 4. Acknowledge to the 82284

Address Setup Margin

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

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

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

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

$$2TCLCL - 80286 t13 (\text{max}) - 8207 TAVCL (\text{min}) \leq 0$$

$$125 \text{ ns} - 60 \text{ ns} - 35 \text{ ns} = 30 \text{ ns}$$

Acknowledge Setup Margin

The 8207 acknowledge (EAACK) can be issued at any point in the 80286 bus cycle (end of phi1 or phi2 of Ts or Tc). If EAACK is issued at the end of phi2 (Ts or Tc), the 80286 will complete the current bus cycle. If

EAACK is issued at the end of phi1 of Tc, the 82284 will not generate READY to the 80286 in time to end the current bus cycle. A new Tc would then be generated and EAACK would now be sampled in time to terminate the bus cycle. EAACK is 3 clocks long in order to meet setup and hold times for either condition.

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

$$TCLCL - 8207 TCLAKL \text{ max} - 82284 t11 \leq 0$$

$$62.5 \text{ ns} - 35 \text{ ns} - 15 \text{ ns} = 12.5 \text{ ns}$$

Read Access Margin

The 8207 will typically start a memory cycle (i.e. RAS goes low) at the end of phi1 of Ts. But if the start of a memory cycle is delayed (by a refresh cycle for instance), then RAS will be delayed. In the first case, this represents 3 clocks and the second case could require 4 clocks to meet the data setup requirements of the 80286. In either case, data must be valid at the end of Tc. The 8207 holds CAS active long enough to ensure valid data is received by the 80286 in either case.

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

RAS Access Margin

$3TCLCL - 8207\ TCLRSL\ \max\ @\ 150\ pF - DRAM\ tRAC$
 $- 74S240\ \text{propagation delay max @ } 50\ pF - 80286\ t8 \leq 0$

$$187.5\ ns - 35\ ns - 120\ ns - 7\ ns - 10\ ns = 15.5\ ns$$

CAS Access Margin

$2TCLCL - 8207\ TCLCSL\ \max\ @\ 150\ pF - DRAM\ tCAA$
 (or $tCAC - 74S240\ \text{tph max @ } 50\ pF - 80286\ t8) \leq 0$

$$125\ ns - 35\ ns - 60\ ns - 7\ ns - 10\ ns = 13\ ns$$

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

$DRAM\ tRAC = 3\ TCLCL - 8207\ TCLRSL - 74S240\ \text{tph}$
 $- 80286\ t8 = 135.5\ ns$

$DRAM\ tCAC = 2\ TCLCL - 8207\ TCLCSL - 74S240\ \text{tph}$
 $- 80286\ t8 = 73\ ns$

NOTES:

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

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

Write Data Setup and Hold Margin

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

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

$2\ TCLCL - 80286\ t14\ (\max)\ @\ 100\ pF - 74S240\ \text{tph} +$
 $8207\ TCLW\ (\min)^1 + 74S10\ \text{tph}\ @\ 192\ pF^2 - DRAM$
 $tDS = 0$

$$125\ ns - 50\ ns - 7\ ns + 0\ ns + 14\ ns - 0\ ns = 82\ ns$$

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

SUMMARY

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



**APPLICATION
NOTE**

AP-285

October 1986

8207 User's Manual

Order Number: 230822-003

CHAPTER 1 INTRODUCTION

This guide is a supplement to the 8207 Data Sheet and is intended as a design aid and not a stand-alone description of the 8207. The reader should already have read and have a copy of the 8207 Data Sheet, 8206 Error Detection and Correction Unit (EDCU) Data Sheet, a microprocessor Data Sheet, or a Multibus bus specification for interfacing to the 8207, and a dynamic RAM Data Sheet⁽¹⁾.

The Intel 8207 Advanced Dynamic RAM Controller is a high performance, highly integrated device designed

to interface 16K, 64K, and 256K dynamic RAMs to Intel microprocessors. The 8207, with the 8206, provides complete control for memory initialization, error correction, and automatic error scrubbing.

The 8207 has three speed selected versions. The -16 version is for clock speeds up to 16 MHz in "fast cycle" configurations. The -8 and -10 parts can only be used in "slow cycle" configurations and are for clock speeds up to 8 MHz and 10 MHz respectively.

NOTE:

1. All RAM cycle timings and references are based upon DRAMs from one particular vendor and will vary depending upon manufacturer.

CHAPTER 2

PROGRAMMING THE 8207

The many configurations of bus structures, RAM speeds, and system requirements that the 8207 supports require the 8207 to be programmable. The 8207 will modify its outputs to provide the best performance possible. The 8207 must be told what type of interface the memory commands will arrive on, what type of RAM (speed, refresh rate) is being used, the clock rate, and others.

The 8207 uses two means to be informed of the user's requirements. It reads in a 16-bit serial program word and examines the logic states on several input pins. The pins that are sampled for a logic level give the user options on the types of refresh and memory command input timing.

Input Pin Options

The three input pins that configure part of the 8207 are: PCTLA, PCTLB, and REFRQ. Let's examine the options in refresh types the REFRQ pin provides.

REFRESH TYPES

The 8207 gives the user a choice of the following refresh types.

- 1) Internal Refresh: All refresh cycles are generated internally—based on an internal programmable time.
- 2) External Refresh with Failsafe: If the external logic does not generate a refresh cycle within the programmed period, the 8207 will.
- 3) External Refresh—No Failsafe or No Refresh; All refresh cycles are generated at times by the user. This is for systems that cannot tolerate the random delay imposed by refresh (i.e. graphics memory).
- 4) Burst Refresh: The 8207 generates up to 128 consecutive refresh cycles and must be requested by external logic. Memory requests will be performed when the burst is completed.

The 8207 examines the state of the REFRQ pin when RESET goes inactive. This timing is shown in the "Clock and Programming Timings" waveforms in the Data Sheet.

If REFRQ is sampled active by the falling edge of RESET, the 8207's internal timer is enabled. The timer's

period is determined by the CIO, CII, and PLS bits in the program word. External refresh cycles are generated by a low to high transition on the REFRQ input. This transition, besides generating a refresh cycle, also resets the internal timer to zero. Simply tie REFRQ to Vcc if internal refresh is required.

If REFRQ is seen low at the falling edge of RESET, the internal timer is deactivated. All refresh cycles must either be done by external logic or by accessing all RAM (internal) rows within a 2 ms period.

Once the no failsafe option is programmed, the 8207 will generate a burst of up to 128 refresh cycles when the REFRQ input goes from low to high and sampled high for two consecutive clock edges. These cycles are internally counted and the 8207 stops when the refresh address counter reaches the value $XX1111111_2$ (X = don't care; see *Refresh Counter* section). If prior to the burst request the counter is at $XX1111110_2$ then only 2 refresh cycles would be generated.

For a single refresh cycle to be generated via external logic, the REFRQ input will have to go from low to high and then sample high by a falling 8207 clock edge. Since external refresh requests typically arrive asynchronously with respect to the 8207's clock, this requires the REFRQ to be synchronized to the 8207 clock when programmed in the failsafe mode. This is to ensure that the request is seen for one clock—no more, no less. If no external synchronization is performed, then the 8207 could do random burst cycles.

PROCESSOR INTERFACE OPTIONS

The PCTLA, PCTLB input pins will program the 8207 to accept either the standard demultiplexed \overline{RD} and \overline{WR} inputs, or to directly decode the status outputs of Intel's iAPX86, 88 family of microprocessors. The state definitions of the status lines and their timings, relative to the processor clock, differ for the 8086 family and the iAPX286 processor. Table 1 illustrates how the 8207 interprets these inputs after the PCTL pins are programmed.

If PCTL is seen high, as RESET goes inactive, an 8086 status interface is enabled. The commands arriving at the 8207 are sampled by a rising clock edge. When PCTL is low, the 80286 status and Multibus command interface is selected. These commands are sampled by the 8207 by a falling clock edge.

Table 1. Status Coding of 8086, 80186 and 80286

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

Programming Word

The 8207 requires more information to operate in a wide variety of systems. The 8207 alters its timings and pin functions to operate with the 8206 ECC chip. The programming options allow the designer to use asynchronous or synchronous buses, various clock rates, various speeds and types of RAM, and others. This is detailed in Table 2.

This data is supplied to the 8207 over the PDI input pin. There are two methods of supplying this data. One is to strap the PDI pin high or low with the subsequent restrictions on your system. Table 3 shows the required system configuration. Note that your only option when strapping this pin high or low is error correction or not.

If any other configurations are required, then the programming data will have to be supplied by one or two 74LS165 type shift registers. Note that the sense of the bits in the program word change between ECC and non-ECC configurations.

8207 Response

8207 Command			Function	
PCTL	RD	WR	8086 Status Interface	Command Interface
0	0	0	Ignore	Ignore
0	0	1	Ignore	Read
0	1	0	Ignore	Write
0	1	1	Ignore	Ignore
1	0	0	Read	Ignore
1	0	1	Read	Inhibit
1	1	0	Write	Inhibit
1	1	1	Ignore	Ignore

Table 2a.

Non-ECC Mode Program Data Word

PD15			PD8 PD7										PD0		
0	0	TM1	PPR	FFS	EXT	PLS	CI0	CI1	RB1	RB0	RFS	CFS	SB	SA	0

Program Data Bit	Name	Polarity/Function	
PD0	ECC	ECC = 0	For non-ECC mode
PD1	SA	SA = 0 SA = 1	Port A is synchronous Port A is asynchronous
PD2	SB	SB = 0 SB = 1	Port B is asynchronous Port B is synchronous
PD3	CFS	CFS = 0 CFS = 1	Fast-cycle iAPX 286 mode Slow-cycle iAPX 86 mode
PD4	RFS	RFS = 0 RFS = 1	Fast RAM Slow RAM

Table 2a.
Non-ECC Mode Program Data Word (Continued)

PD15	0	0	TM1	PPR	FFS	EXT	PLS	CI0	CI1	RB1	RB0	RFS	CFS	SB	SA	PD0
																0

Program Data Bit	Name	Polarity/Function	
PD5 PD6	$\overline{RB0}$ $\overline{RB1}$	RAM bank occupancy See Table 4	
PD7 PD8	CI1 CI0	Count interval bit 1: see Table 6 in 8207 data sheet Count interval bit 0: see Table 6 in 8207 data sheet	
PD9	PLS	$\overline{PLS} = 0$ $\overline{PLS} = 1$	Long refresh period Short refresh period
PD10	EXT	EXT = 0 EXT = 1	Not extended Extended
PD11	FFS	FFS = 0 FFS = 1	Fast CPU frequency Slow CPU frequency
PD12	PPR	PPR = 0 PPR = 1	Most recently used port priority Port A preferred priority
PD13	TM1	TM1 = 0 TM1 = 1	Test mode 1 off Test mode 1 enabled
PD14	0	Reserved must be zero	
PD15	0	Reserved must be zero	

Table 2b.
ECC Mode Program Data Word

PD15	TM2	RB1	RB0	PPR	FFS	EXT	PLS	CI0	CI1	XB	XA	RFS	CFS	SB	SA	PD0
																1

Program Data Bit	Name	Polarity/Function	
PD0	ECC	ECC = 1	ECC mode
PD1	SA	SA = 0 SA = 1	Port A is asynchronous (late AACK) Port A is synchronous (early AACK)
PD2	\overline{SB}	$\overline{SB} = 0$ $\overline{SB} = 1$	Port B is synchronous (early AACK) Port B is asynchronous (late AACK)
PD3	CFS	CFS = 0 CFS = 1	Slow-cycle iAPX 86 mode Fast-cycle iAPX 286 mode
PD4	RFS	RFS = 0 RFS = 1	Slow RAM Fast RAM
PD5	\overline{XA}	$\overline{XA} = 0$ $\overline{XA} = 1$	Multibus-compatible XACKA AACKA not multibus-compatible
PD6	XB	XB = 0 XB = 1	AACKB not multibus-compatible Multibus-compatible XACKB
PD7 PD8	CI1 CI0	Count interval bit 1: see Table 6 in 8207 data sheet Count interval bit 0: see Table 6 in 8207 data sheet	

Table 2b.
ECC Mode Program Data Word (Continued)

PD15	RB1	RB0	PPR	FFS	EXT	PLS	C10	C11	XB	XA	RFS	CFS	SB	SA	1
------	-----	-----	-----	-----	-----	-----	-----	-----	----	----	-----	-----	----	----	---

Program Data Bit	Name	Polarity/Function	
PD9	PLS	PLS = 0	Short refresh period
		PLS = 1	Long refresh period
PD10	EXT	EXT = 0	Master and slave EDCU
		EXT = 1	Master EDCU only
PD11	FFS	FFS = 0	Slow CPU frequency
		FFS = 1	Fast CPU frequency
PD12	PPR	PPR = 0	Port A preferred priority
		PPR = 1	Most recently used port priority
PD13	RB0	RAM bank occupancy	
PD14	RB1	See Table 4	
PD15	TM2	TM2 = 0	Test mode 2 enabled
		TM2 = 1	Test mode 2 off

Table 3. 8207 Default Programming

Port A is Synchronous—has early AACK
Port B is Asynchronous—has late AACK
Fast RAM
Refresh Interval uses 236 clocks
128 Row refresh in 2 ms; 256 Row refresh in 4 ms
Fast Processor Clock Frequency (16 MHz)
"Most Recently Used" Priority Scheme
4 RAM banks occupied

Reset

If Port A is changed to an asynchronous interface (via the SA bit), then one of two precautions must be taken. Either a differentiated reset must be provided, or else software must not access the 8207 controller RAM for a short period. The 8207 is either adding or deleting internal synchronizing circuits. If a command is re-

ceived during this changing, the 8207 may not perform properly. This is required only if Port A is changed to asynchronous, or if Port B is changed to synchronous.

Several of the bits in the program word determine a particular configuration of the 8207 (reference Tables 10, 11 and the 8207 Data Sheet). The bits are: CFS, CLOCK fast or slow; RFS, RAM access time fast or slow (fast refers to 100 ns—slow is everything greater); and EXT, for memory data word widths greater than 16 (22) bits. Generally speaking, C0 is the fastest configuration at clock frequencies up to 16 MHz, both in the ECC or non-ECC charts. 'C3' is the fastest for 8 MHz clocks in non-ECC mode, and 'C4' is the fastest configuration when using ECC.

Take, for example, a 16 MHz 8207 clock with no error correction, a 16-bit word, and 150 ns (slow) dynamic RAMs. Table 10, in the 8207 data sheet, is used to arrive at the configuration "C1." The Timing chart Table 12 in the 8207 Data Sheet is then used to determine which clock edge to reference all timings from. The Waveforms diagrams then are used to determine the delay from the clock edge.

CHAPTER 3

RAM INTERFACE

The 8207 takes the memory addresses from the microprocessor bus and multiplexes them into row and column addresses as required by dynamic RAMs. The only hardware requirement when interfacing the 8207 to dynamic RAM are series resistors on all the RAM outputs of the 8207, and proper layout of the traces (see Intel's RAM Data Sheets or the Memory Design Handbook). This section mainly details the effects and requirements of input signals to the 8207 on the RAM array.

The 8207 contains an internal address counter used for refresh and error scrubbing (when using the 8206 EDCU) cycles. The 8207 has 18 address inputs (AII0-AII8 and AIH0-AIH8) which are multiplexed to form 9 address outputs (A00-A08). There are also 2 bank select (BS0, BS1) inputs for up to 4 banks of RAM. The Bank Select inputs are decoded internally to generate RAS and CAS outputs.

Refresh Interval

The 8207 supports four different refresh techniques as described in the *Refresh Options* section. In addition, the rate at which refresh cycles are performed is programmable. This is necessary because the refresh period is generated from the CLK input, which may vary over a wide range of frequencies. Programming the Cycle Fast/Slow (CFS) and Frequency Fast/Slow (FFS) bits automatically reprograms the refresh timer to generate the correct refresh interval for a clock frequency of 16, 10, 8, or 5 MHz (CFS, FFS = 11, 10, 01, and 00, respectively). For clock frequencies between those, Count Interval (CII, CIO) programming bits allow "fine tuning" of the refresh interval. Refresh will always be done often enough to satisfy the RAM's requirements without doing refresh more often than needed and wasting memory bandwidth for all clock frequencies.

Refresh Counter

The internal refresh address counter of the 8207 contains 20 bits as organized in Figure 1.

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bank		Col addr										Row addr							

Figure 1. 8207 Refresh Address Counter

In non-ECC mode, the refresh address counter does not count beyond bit 8. For standard RAMs, this will refresh 128 rows every 2 ms or 256 rows every 4 ms.

In ECC mode, the 8207 automatically checks the RAM for errors during refresh. This requires it to access each of the possible 2^{20} words of memory. The 8207 does not delete any of these bits when used with 16K and 64K dynamic RAMs. Each column would be scrubbed 4 times with 16K RAMs, and twice with 64K RAMs. This will have no detrimental effect on reliability. Banks of RAM that are not occupied, as indicated to the 8207 by the RB0, RB1 programming bits, will not be scrubbed.

Bank Selects BS0, BS1; RB0, RB1

The 8207 is designed to drive up to 88 RAMs in various configurations. The 8207 takes 2 inputs, BS0, BS1, and decodes them based on 2 programming bits, RB0, RB1, to generate the required RAS/CAS strobes. Additionally, the 8207 will always recognize (not programmable) whether an access is made to the same RAM bank or to a different bank. The 8207 will interleave the accesses resulting in improved performance.

RAS and CAS Reallocation

The 8207's address lines are designed to drive up to 88 RAMs directly (through impedance matching resistors). The 4 RAS and CAS outputs drive up to 22 RAMs per bank (16 data plus 6 check bits with the 8206). Under these conditions, the 8207 will meet all RAM timing requirements. See Figure 2 for an example.

The 8207 can accommodate other configurations like a 32-bit error corrected memory system. Each bank would have 39 RAMs (32 + 7 check bits) with the total number of RAMs equal to 78. This is within the address driver's capability, but the 39 RAMs per bank exceeds the RAS and CAS driver's limits. The loading of the RAS/CAS drivers should not exceed 22 RAMs per bank, otherwise critical row, column address setup, and hold times would be violated.

In order to prevent these critical timings being violated, the 8207 will re-allocate the RAS and CAS drivers based on the RB0, RB1 programming bits (see Table 4). If the RB0, RB1 bits are programmed for 2 banks, the 8207 will operate RAS0 and RAS1 as a pair along with RAS2 and RAS3, CAS0 and CAS1, and CAS2

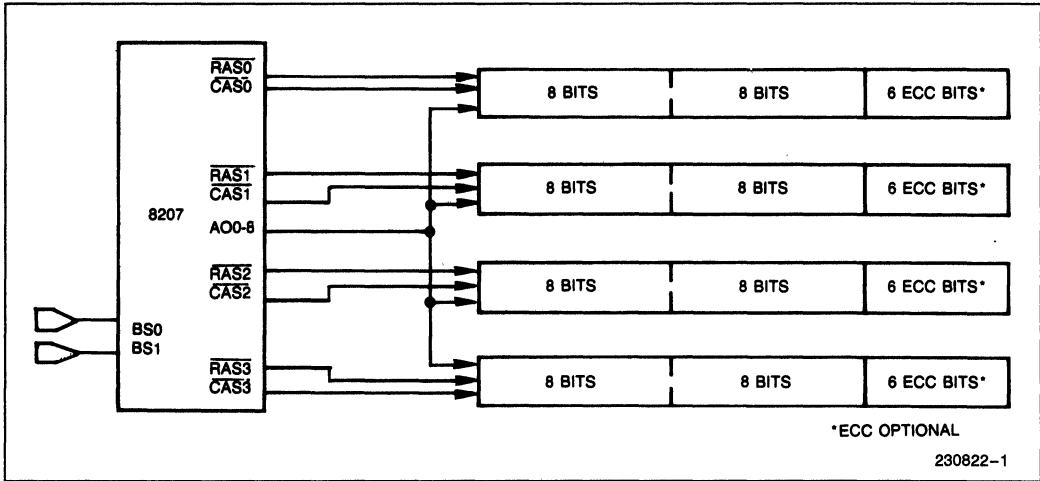


Figure 2. 8207 4 RAM Bank Configuration

and $\overline{CAS3}$. Now the address drivers would be loaded by 78 RAMs and the RAS/CAS drivers by 20 RAMs. This relative loading is almost identical to the first case of four banks of 22 RAMs each. Drive reallocation allows a wide range of memory configurations to be used and still maintain optimal memory timings. Figure 3 shows a 32-bit non-error corrected configuration.

These programming bits do not help to qualify RAM cycles. Their purpose is to reallocate RAS/CAS drivers. For example, if there is one bank of RAM and the bank select inputs (BS0, BS1) select any other bank and no provision is made to deselect the 8207 (via PE), the 8207 will do a RAM cycle and issue an acknowledge. This happens irregardless of the RB0, RB1 programmed value. See the *Optional RAM Bank's* section to provide for this.

Table 4. RAM Bank Selection Decoding and Word Expansion

Program Bits		Bank Input		RAS/CAS Pair Allocation
RB1	RB0	B1	B0	
0	0	0	0	$\overline{RAS}_{0-3}, \overline{CAS}_{0-3}$ to Bank 0
0	0	0	1	Illegal Bank Input
0	0	1	0	Illegal Bank Input
0	0	1	1	Illegal Bank Input
0	1	0	0	$\overline{RAS}_{0,1}, \overline{CAS}_{0,1}$ to Bank 0
0	1	0	1	$\overline{RAS}_{2,3}, \overline{CAS}_{2,3}$ to Bank 1
0	1	1	0	Illegal Bank Input
0	1	1	1	Illegal Bank Input
1	0	0	0	$\overline{RAS}_0, \overline{CAS}_0$ to Bank 0
1	0	0	1	$\overline{RAS}_1, \overline{CAS}_1$ to Bank 1
1	0	1	0	$\overline{RAS}_2, \overline{CAS}_2$ to Bank 2
1	0	1	1	Illegal Bank Input
1	1	0	0	$\overline{RAS}_0, \overline{CAS}_0$ to Bank 0
1	1	0	1	$\overline{RAS}_1, \overline{CAS}_1$ to Bank 1
1	1	1	0	$\overline{RAS}_2, \overline{CAS}_2$ to Bank 2
1	1	1	1	$\overline{RAS}_3, \overline{CAS}_3$ to Bank 3

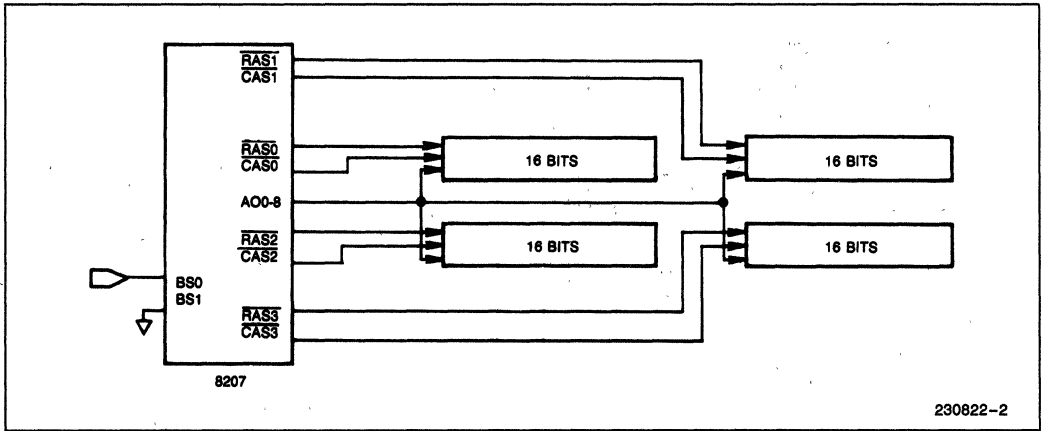


Figure 3. 8207 2 RAM Bank Configuration

Scrubbing

An additional function of the RB0, RB1 bits, besides $\overline{RAS}/\overline{CAS}$ allocation, is to inform the 8207 of how many banks are physically present. The 8207 will, during the refresh cycle, read data from a location and check to see that data and check bits are correct. If there is an error, the 8207 lengthens the refresh cycle and writes the corrected data back into RAM. Scrubbing the entire memory greatly reduces the chance of an uncorrectable error occurring. See the *Refresh* section for more detail on scrubbing.

for \overline{CAS} and acknowledges. The real delay in a system due to refresh would be a fraction of that value⁽¹⁾. The length of the refresh cycle is always $2t_{RP} + t_{RAS}$, and varies based upon the programmed 8207 configuration.

In error-corrected systems, the refresh cycle is actually a read cycle. The 8207 outputs a row address, then all \overline{RAS} outputs go active. Next, a column address is output and then \overline{CAS} . The \overline{CAS} output is based upon the RB0, RB1 allocation bits. Figure 4a shows the general timing for a four bank system, and Figure 4b shows a two bank system.

Refresh Cycles

The 8207 performs \overline{RAS} only refresh cycles in non-ECC systems. It outputs all 8207 control signals except

NOTE:

1. Measurements have shown a delay of 2-4% on program execution time compared to programs running without refresh.

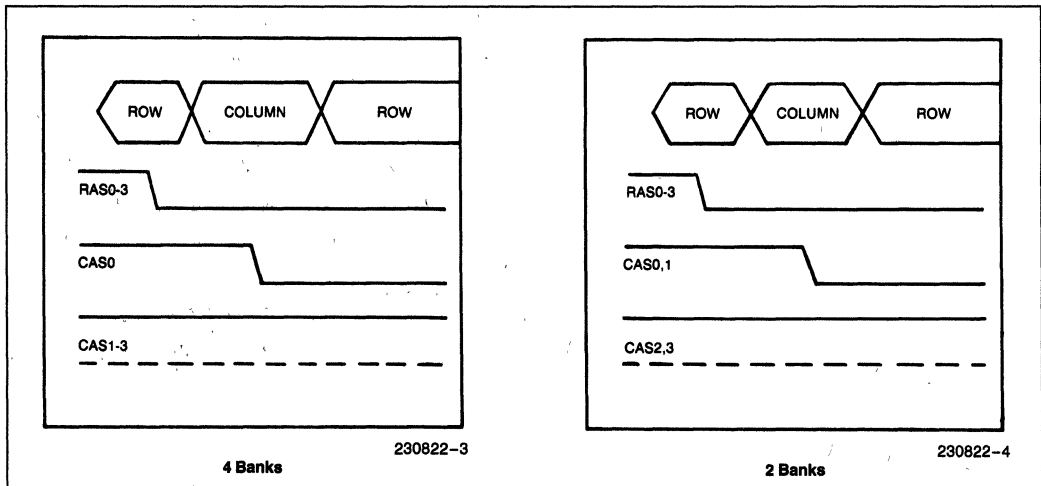


Figure 4. Refresh Cycles for Error Corrected Systems

The 8207 sends the read out word through the 8206 EDCU to check for any errors. If no errors, the refresh cycle ends. If an error is discovered, the 8207 lengthens the cycle. An error is determined if the ERROR output of the 8206 is seen active at the same edge that the 8207 issues the R/W output. The cycle is then lengthened to a RMW cycle. If the error was correctable, the corrected data is written back to the location it was read from. But, if the data is uncorrectable, the cycle is still lengthened to a RMW, but no write pulse is issued. To aid in stabilizing the RAM output data and the Error flag, pullup resistors of 10 K Ω on the data out lines are recommended.

Scrubbing removes soft errors that may accumulate until a double-bit error occurs, which would halt the system. Hard single-bit failures will not stop the system, but could slow it down. This is because read and refresh cycles lengthen to correct the data.

For large RAM arrays some form of error logging or diagnostics should be considered.

Interleaving

The term "interleaving" is often used to refer to overlapping the cycle times of multiple banks (or boards or systems) of RAMs. This has the advantage of using relatively slow cycle time banks to achieve a faster perceived cycle time at the processing unit. The drawbacks of interleaving are more logic to handle the necessary control and, for maximum performance, the program should execute sequentially through the addresses.

Dynamic RAM cycles consist of 2 parts—the RAS active time (tRAS in Dynamic RAM Data Sheets) and precharge time (tRP). The sum of these two times is roughly equal to the cycle time of the RAM. The 8207 determines how long these two periods are, based on the configuration the user picked (via the programming bits). Bank interleaving, as used by the 8207, is slightly different than the previous definition. The 8207 will overlap the precharge time of one bank with the access time of another bank. In either case, the advantage is the effective cycle time is reduced without having to use faster RAMs.

For interleaving to take place there must be more than 1 bank of RAM connected to the 8207. Interleaving is not practical with 3 banks of RAM because 3 is not a power of 2 (the 2 bank inputs BS0, BS1). So, interleaving works only for 2 or 4 banks of RAM. Note that it is easy enough to use three banks of RAM where the bank select inputs are connected to the highest-order address line. For instance, if three banks of 64K

DRAMs are used in an 8086 system, and located at address OH, bank selects BS0 and BS1 would be connected to microprocessor addresses A17 and A18, respectively. Banks 0–2 would be accessed in the address ranges OH–FFFFH, 10000H–1FFFFH, and 20000H–2FFFFH, respectively. In this case, consecutive addresses are almost always in the same bank and very little interleaving can take place.

Figure 5 shows the effects on the performance of the processor with and without interleaving. In both examples, consecutive accesses to the same bank will add 1 wait state to the second access, but no wait states to consecutive accesses to different banks. Irregardless of the 8207 configuration, there will always be a minimum 1 wait state added without interleaving. Therefore, interleaving is very highly recommended!

Interleaving is accomplished by connecting the 8207's BS0, BS1 inputs to the microprocessor's low order word address lines. Each consecutive address is then located in a different bank of RAM. About 90% of memory accesses are sequential, so interleaving will occur about 90% of the time in a single port system.

In a dual port system, the advantages of interleaving are a function of the number of banks of memory. Since the memory accesses of the two ports are presumably independent, and both ports are continuously accessing memory, the 8207 arbiter will tend to interleave accesses from each port (i.e., Port A, Port B, Port A, Port B, . . .). If there are two banks of RAM interleaving will occur 50% of the time and, if there are four banks of RAM, interleaving will take place 75% of the time⁽¹⁾. To the extent that a single port generates a majority of memory cycles, interleaving efficiency will approach 90% as described in the previous paragraph.

NOTE:

1. Don't get confused here. The paragraph is talking about interleaving memory requests from both ports, and their probability of accessing one of the other banks of RAM where tRP has been satisfied. The 8207 will leave the RAM precharge time out if consecutive accesses go to different banks. The 8207 RAM timing logic does not care which port requests a RAM cycle.

Optional RAM Banks

Many users allow various RAM array sizes for customer options and future growth. Some care must be taken during the design to allow for this. Three items should be considered to permit optional RAM banks.

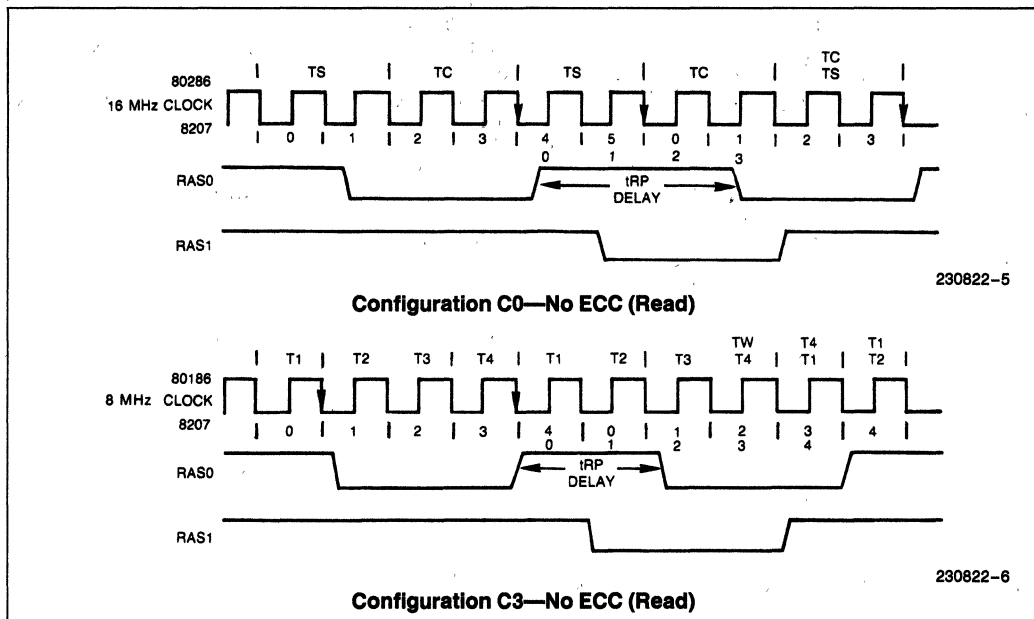


Figure 5. Processor Performance with and without Interleaving

The first item is the total RAM size. The 8207 starts a memory cycle based only upon a valid status or command and \overline{PE} active. So some logic will be required to deselect the 8207 (via \overline{PE}) when the addressed location does not exist within the current memory size. A 7485 type magnitude comparator works well.

The second item to consider is the BS0, BS1 inputs. With one bank of RAM these inputs are tied to ground. Four banks of RAM require two address inputs. So, if the design ever needs four banks of RAM, then the BS0, BS1 inputs must be connected to address lines. Selecting a non-existent RAM bank is illegal. Figure 6 shows a non-interleaved method.

With designs using interleaving, the least significant word address lines are connected to the BS0, BS1 inputs. With two banks of RAM, A1 from the Intel processor is connected to BS0. A2 is connected to BS1, but not allowed to function until four banks are present. However, A2 must still be used since addresses increase sequentially. Two possible ways of implementing this are shown in Figure 7.

The final consideration is for the $\overline{RAS}/\overline{CAS}$ outputs. Remember that when the RB0, RB1 bits are programmed for two banks, the $\overline{RAS}_0, 1$ operates in tandem (non-ECC mode/ECC mode—the \overline{CAS} outputs also work in tandem). Figure 8 shows the proper layout.

Write Enables—Byte Marks

The write enable supplied by the 8207 cannot drive the RAM array directly. It is intended to be NAND with the processor supplied byte marks in a non-ECC system. In error-corrected systems, the write enable output should be inverted before being used by RAMs. Only full word read/writes are allowed in ECC systems. The changing of byte data occurs in the 8206 EDCU.

For single and dual port systems, the byte mark data (A0, \overline{BHE}) must be latched. The 8207 can (and will) change the input addresses midway through a RAM cycle.

Memory Warm-Up and Initialization

After programming, the 8207 performs 8 RAM warm-up cycles. The warm-up cycles are to prepare the RAMs for proper operation. If the 8207 is configured for ECC, it will then prewrite zeros into the entire array.

All \overline{RAS} outputs are driven active for these cycles, once every 32 clock periods. The prewrite cycles are equivalent to write cycles, except all \overline{RAS} and \overline{CAS} will go active, data is generated by the 8206, and the address is generated by the 8207.

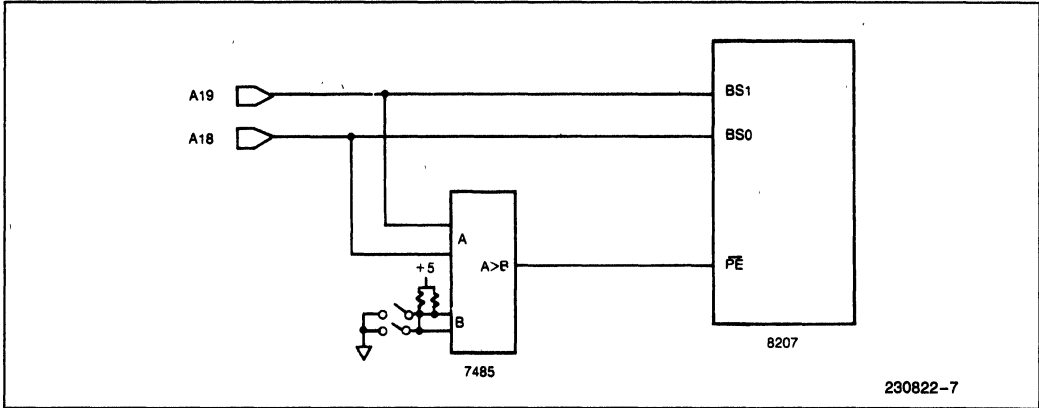


Figure 6. Non-Interleaved 8207 Selection Circuit

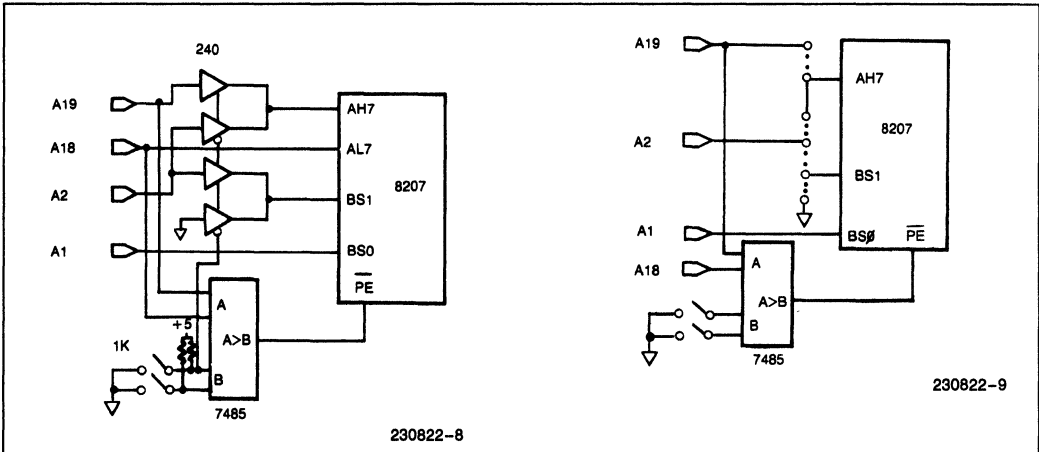


Figure 7. Interleaved 8207 Selection Circuits

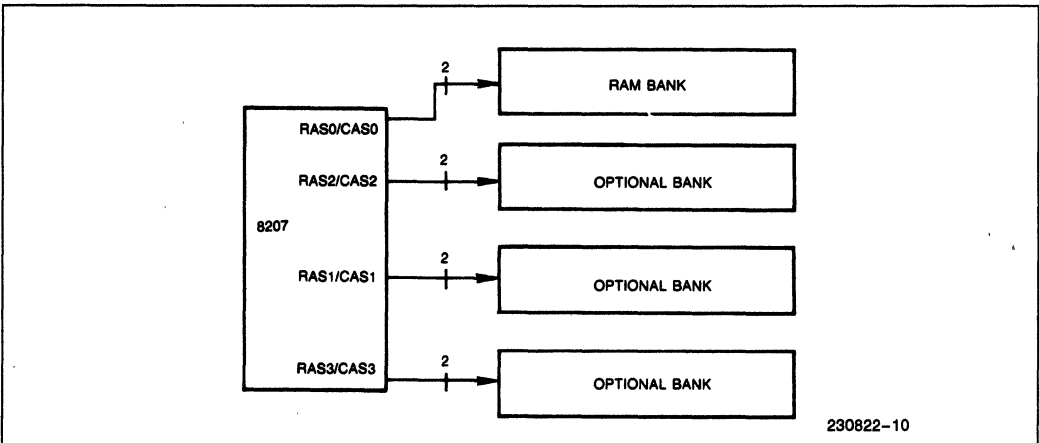


Figure 8. RAM Bank Layout

RAM Cycles/Timings

Tables 12 and 13 of the 8207 Data Sheet show on what clock edge each of the 8207 outputs are generated. This, together with the timing waveforms and A.C. parameters, allows the user to calculate the timings of the 8207 for each of its configurations. To make the job easier, Tables 14–18 of the 8207 Data Sheet precalculate dynamic RAM timings for each 8207 configuration and type of cycle. All that is required is to plug in numerical values for the 8207 parameters.

Write Cycles

The 8207 always issues \overline{WE} after \overline{CAS} has gone valid. These types of cycles are known as “late writes.” The

8207 does this primarily to interface to the iAPX286 processor bus timings. Late writes require separate data in and data out traces to the RAM array, plus the additional drivers.

Data Latches

The 8207 is designed to meet data setup and hold times for the iAPX86 family processors when using a synchronous status interface (see Microprocessor Interface section). Other types of interfaces will require external data latches. This is because the \overline{CAS} pulse is a fixed length—the user has no control (besides programming options) over lengthening \overline{CAS} . When \overline{CAS} goes inactive, data out of the RAMs will disappear. Asynchronous interfaces should use \overline{XACK} or \overline{LAACK} to latch the data.

CHAPTER 4

MICROPROCESSOR INTERFACES

The 8207 is designed to be directly compatible with all Intel iAPX86, 186, 188, and 286 processors. For maximum performance, the 8207 will directly decode the status lines and operate off of the processor's clock. Additionally, the 8207 interfaces easily to other bus types that support demultiplexed address and data with separate read and write strobes.

Bus Interfaces

The 8207 easily supports either an asynchronous or synchronous command timing. The command timing can also be adjusted for various processors via the PCTL pin.

MEMORY COMMANDS

There are four inputs for each port of the 8207 that initiate a memory cycle. The input pins are \overline{WR} , \overline{RD} , PCTL, and PE. The first three inputs connect directly to the iAPX 86, 88, 186, 188 $\overline{S0}$ – $\overline{S2}$ outputs, respectively. For the 80286, the same connections are used except that PCTL is tied to ground. In all configurations \overline{PE} is decoded from the address bus. Multibus type commands use the same input setup as the 80286.

COMMAND/STATUS INTERFACE

The status interface for the 80186 and the 80286 differ both in timing and meaning. The 8207 can be optimized for either processor by programming the PCTL input pin at RESET time. $\overline{S2}$ in 80186 systems, connects directly to PCTL. When the processor is reset it drives $\overline{S2}$ high for one clock, then tristates it. A pullup resistor to +5 will program the PCTL input for the 80186 status interface when RESET goes inactive. A pullup is required only if no component has this pullup internally.

To optimize the 8207 for the 80286 interface, PCTL is tied to ground and not used in 80286 systems. Multibus commands are similar in meaning to the 80286 status interface, and are programmed the same way. In Multibus type systems, PCTL can be used as an inhibit to allow shadow memory. PCTL would be driven high, when required, to prevent the 8207 from performing a memory cycle. It would be connected to the Multibus INH pin through an inverter.

SYNCHRONOUS/ASYNCHRONOUS COMMANDS

Each port of the 8207 can be configured to accept either a synchronous or asynchronous (via programming bits) memory request. Minimum memory request decode time (and maximum performance) is achieved using a synchronous status interface. This type of interface to the processor requires no logic for the user to implement.

An asynchronous interface is used with Multibus bus interfaces when the setup and hold times of the memory commands cannot be guaranteed. Synchronizers are added to the inputs and will require up to two clocks for the 8207 to recognize the command. It should be obvious that better performance will result if the 8207's clock is run as fast as possible.

Figure 2 of the 8207 Data Sheet shows various combinations of interfaces. The additional logic for the asynchronous interfaces is used to either lengthen the command width, to meet the minimum 8207 spec, or to make sure the command does not arrive too soon before the address has stabilized.

PORT ENABLE

The \overline{PE} inputs serve to qualify a memory request. A RAM cycle, once started, cannot be stopped. A RAM cycle starts if \overline{PE} is seen active at the proper clock edge and a valid command is recognized. If \overline{PE} is activated after a command has gone active and inactive, no cycle will start.

Types of logic that work well are 74138 and 7485. \overline{PE} should be valid as much as possible before the command arrives because, as the address bus switches and settles, glitches on \overline{PE} could either: disqualify a memory cycle; delay a memory cycle; or start a memory cycle when none should have. Refer to the Port Interface Waveforms in the Data Sheet. If Port Enable is not seen active by the next or same clock edge, no memory cycle will occur unless the command is removed and brought active again.

Back to Back Commands

Holding the \overline{RD} , \overline{WR} inputs active will not generate continuous memory cycles. Memory commands must go inactive for at least one clock period before another memory request at that port will be considered valid. Holding the inputs active will not keep the other port from gaining access to the RAM. The only signal that can prevent the other port's gaining access to the RAM is LOCK.

Address Inputs (and LOCK)

Two pins control the address inputs on the 8207, MUX and LEN. Neither are used for single port 8086 based systems. MUX is used for dual port configurations, and LEN is used for single and dual port 80286 based systems. MUX is used to gate the proper ports addresses to the 8207. If the output is high, Port A is selected. If it is low, Port B is selected.

The cross coupled NAND gates, shown in the 8207 Data Sheet (Figure 3), are used to minimize contention when switching address buses. Use of a single inverter would have both outputs enabled simultaneously for a short period. The cross coupled hand gates allow only one output enabled.

MUX also allows the single LOCK input to be multiplexed between ports. Figure 9 shows how to multiplex the LOCK input for dual port systems. See the LOCK section for more information.

MUX TIMING

The MUX output is optimized by the Port Arbitration scheme, which is selected in the program word. Figure 10 shows the effects on memory bandwidth with the different schemes. Port A Preferred optimizes consecutive cycles for Port A. Consecutive Port B cycles have at least 1 clock added to their cycle time. There would be no MUX delays for any Port A request.

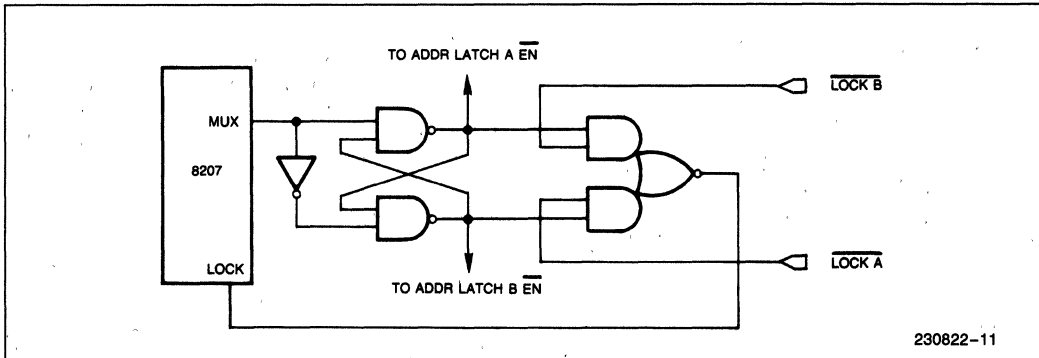


Figure 9. Dual Port LOCK Input Circuit

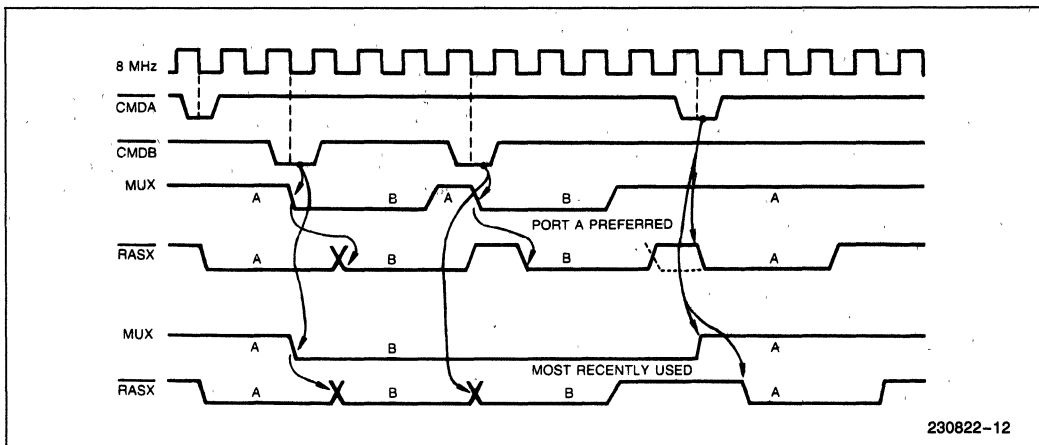


Figure 10. Port Arbitration Effects

The Most Recently Used scheme allows either port to generate consecutive cycles without any MUX delays. The first memory cycle for each port would have the 1 clock delay. But all others would not.

With either scheme, if both ports request the memory at their top speed, the 8207 will interleave the requests; Port A, Port B, Port A, Refresh, Port B.

LEN

LEN is used to hold the 80286 addresses when the 8207 cannot respond immediately. The 8207 will require a separate address latch, with the ALE input replaced with LEN. LEN optimizes the address setup and hold times for the 8207.

LEN goes from high to low when a valid 8207 command is recognized, which latches the 80286 address. This transition of LEN is independent of a memory cycle starting. The low to high transition will occur in the middle of a memory cycle so that the next address will be admitted and subsequently latched.

If Port B is to interface to an 80286 with the synchronous status interface, then LEN must be created using external logic. Figure 11 shows the equivalent 8207 circuit for Port B.

LOCK

The LOCK input allows each port uninterrupted access to memory. It does this by not permitting MUX to switch. It is not intended as a means to improve throughput of one of the ports. To do so is at the designer's risk⁽¹⁾. Obviously, LOCK is only used in dual port systems. The 8207 interprets LOCK as originating from the port that MUX is indicating.

NOTE:

1. The 8207 will not malfunction if this is done. This is a system level concern. For example, a time dependent process may fail if the other port holds LOCK active, preventing its access of memory and relinquishing the bus.

LOCK from the 8086 may be connected directly to the 8207 or to the multiplexing logic. The 8207 requires additional logic when interfaced to an 80286. Figure 12 shows both the synchronous and asynchronous circuitry.

For 16 MHz operation, the 8207 ignores the LOCK input during the clock period that MUX switched. During 8 MHz operation, the 8207 will see LOCK as being active during the clock period when MUX switches.

The LOCK issued in Multibus bus systems may not be compatible with the 8207. The 8207 references LOCK from the beginning of a cycle, while Multibus references LOCK from the end of a cycle. The Multibus LOCK can be used if it meets the 8207 requirements. If the LOCK timing cannot be guaranteed, then additional logic is necessary. The logic would issue LOCK whenever a Multibus command is recognized. The drawback to this is that MUX cannot switch during the RAM cycle. This would delay the other port's memory access by one or two clocks.

DEADLOCK

The designer should ensure that a deadlock hazard has not been created in the design. The simple interfaces shown previously will not create a deadlock condition when the 8207 controls all system memory. If LOCK is issued by both ports, then the above logic would need to be modified to remove LOCK.

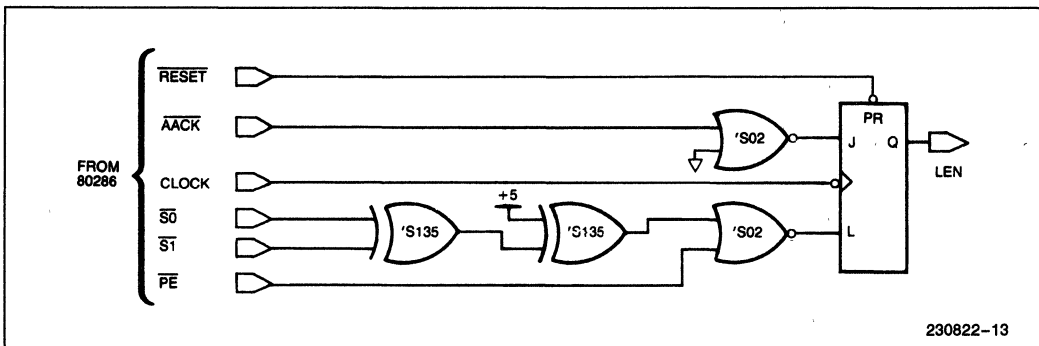


Figure 11. Port B LEN Circuit

Figure 13 shows an illustration of the problem with a single LOCK input.

Suppose the 8207 starts a locked string transfer for the processor. The Multibus bus port requests a memory cycle but must wait for the processor to remove LOCK. But the processor must access Multibus as part of the

locked string transfer. We now have a deadlock. The solution is to force LOCK inactive whenever an access is made to non-8207 memory by the processor. By doing this we have now violated the purpose of LOCK, since the Multibus port could change data. Another solution is to ensure that locked data does not exist in physically separate memory.

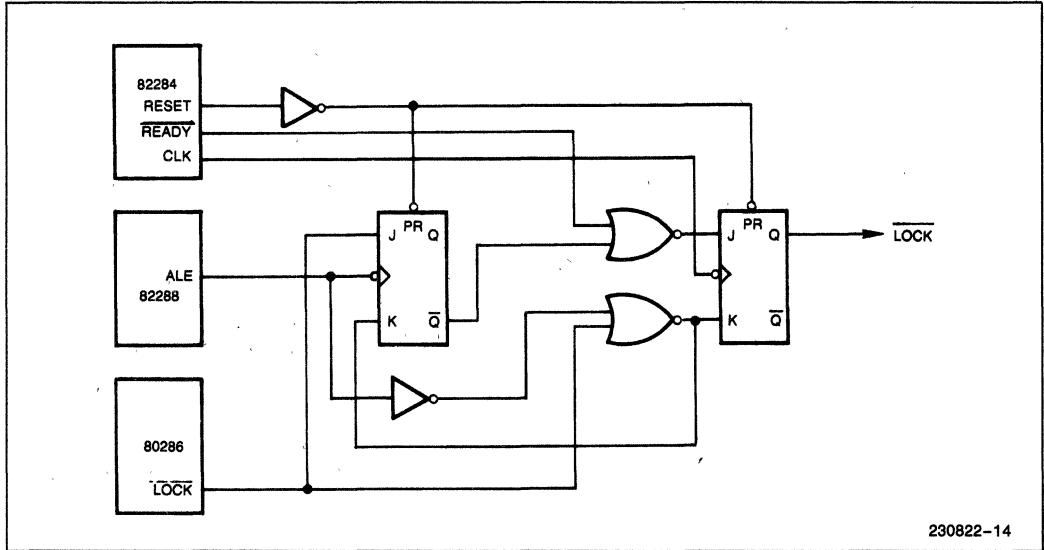


Figure 12a. Synchronous Interface

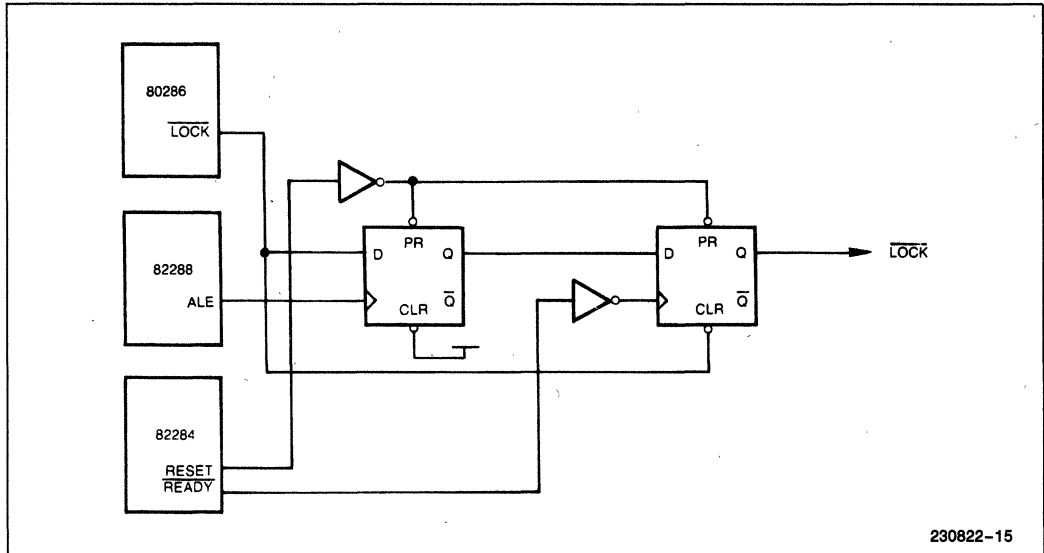


Figure 12b. Asynchronous Interface

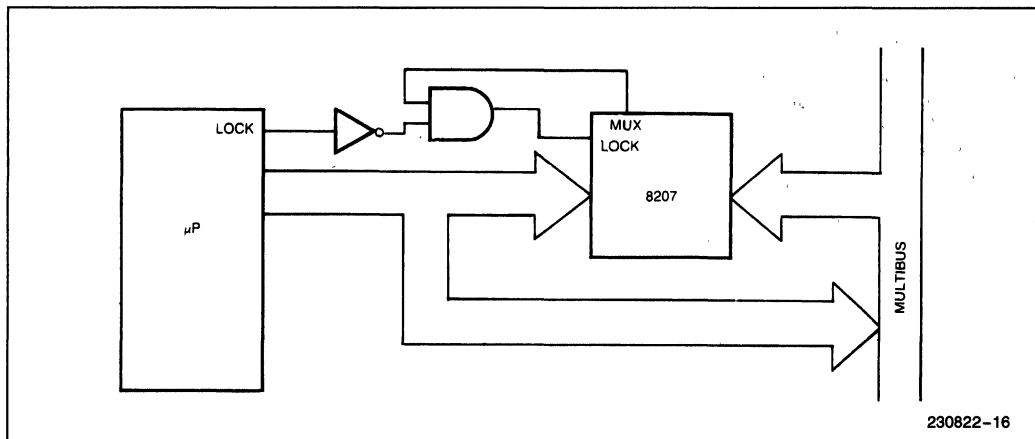


Figure 13. Single LOCK Input Circuit

8207 Acknowledges

The 8207 in non-ECC mode has two active acknowledges per port, $\overline{\text{AACK}}$ and $\overline{\text{XACK}}$. The $\overline{\text{AACK}}$ output is configured into either an "early" or "late" $\overline{\text{AACK}}$ based on the SA, SB bits in the program data word. In ECC systems there is one Acknowledge per port, and it is configured to any one of the three ($\overline{\text{EAACK}}$, $\overline{\text{LAACK}}$ or $\overline{\text{XACK}}$) by the programming bits.

The $\overline{\text{AACK}}$ pin is optimized for either the 80286 or the 8086, based upon the CFS programming bit (fast = 80286; slow = 8086). $\overline{\text{XACK}}$ conforms to the Multibus bus specification. $\overline{\text{XACK}}$ requires a tri-state buffer and must not drive the bus directly.

In synchronous systems, $\overline{\text{XACK}}$ will not go active if the memory command is removed prior to the clock period that issues $\overline{\text{XACK}}$. In asynchronous systems, the $\overline{\text{AACK}}$ pin can also serve as an advanced RAM cycle timing indicator.

Data out, in synchronous systems, should not have to be latched. The 8207 was designed to meet the data setup and hold times of Intel processors, the 8086 family, and the 80286. In asynchronous systems, the 8207 will remove data before the processor recognizes the Acknowledge ($\overline{\text{LAACK}}$ or $\overline{\text{XACK}}$). In these systems, the data should be latched with transparent type latches (Intel 8282/8283).

Output Data Control

NON-ECC

In single port systems, Intel processors supply the necessary timing signals to control the input or output of data to the RAMs. These control signals are $\overline{\text{DEN}}$ and $\overline{\text{DT/R}}$. Refer to the microprocessor handbook for their explanation. If these signals are not available, then PSEN and DBM provide the same function. They can be used directly to control the 8286/8287 bus drivers of the 8207.

Because of the single set of data in/out pins of the RAMs, data must be multiplexed between the two ports in dual port systems. The 8207 provides two outputs for contention-free switching. PSEL operates the same as the MUX output, in that a high selects Port A and a low selects Port B. PSEN acts to enable the selected port. The timing is shown in the 8207 Data Sheet, Port Switching Timing section.

The easiest means of using PSEL and PSEN is shown in Figure 14. At no time will both ports be enabled simultaneously.

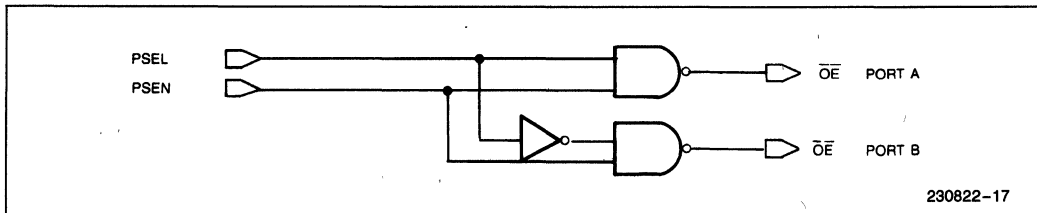


Figure 14. PSEL and PSEN Interface Circuit

Data Bus—Single Port

Recall that the 8207 always performs a late write cycle and that this requires separate data in and out buses. One option for the data bus is shown in Figure 3 of the 8207 Data Sheet. It requires separate data in and out traces on the processor board.

The second option is to keep the processor's combined data, but separate the data at the 8207 RAM. This is shown in Figure 15.

Data Bus—Dual Port

NON-ECC

The multiplexed data of the 8207 RAM must be kept isolated so that an access by one port does not affect another port. Figure 16 illustrates the control logic.

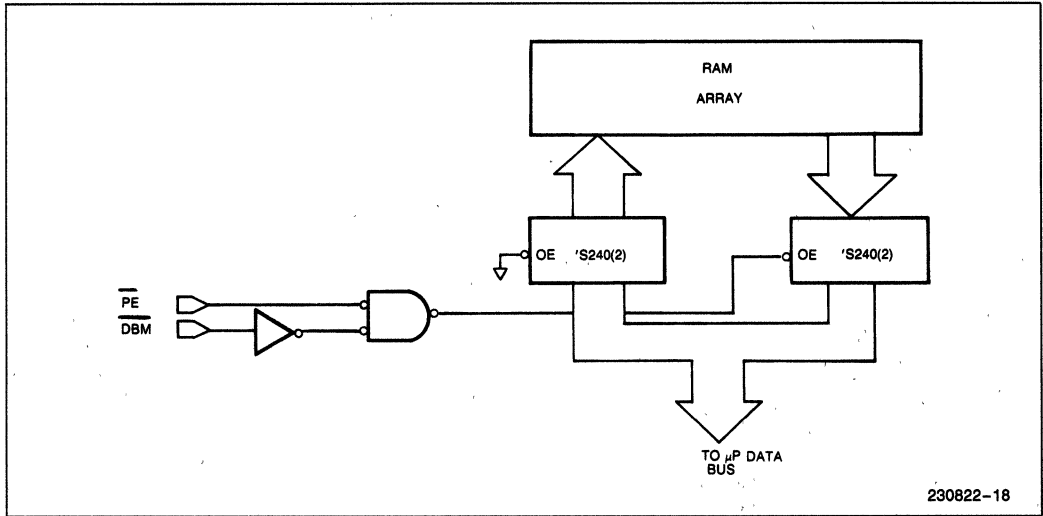


Figure 15. Data Bus Circuit

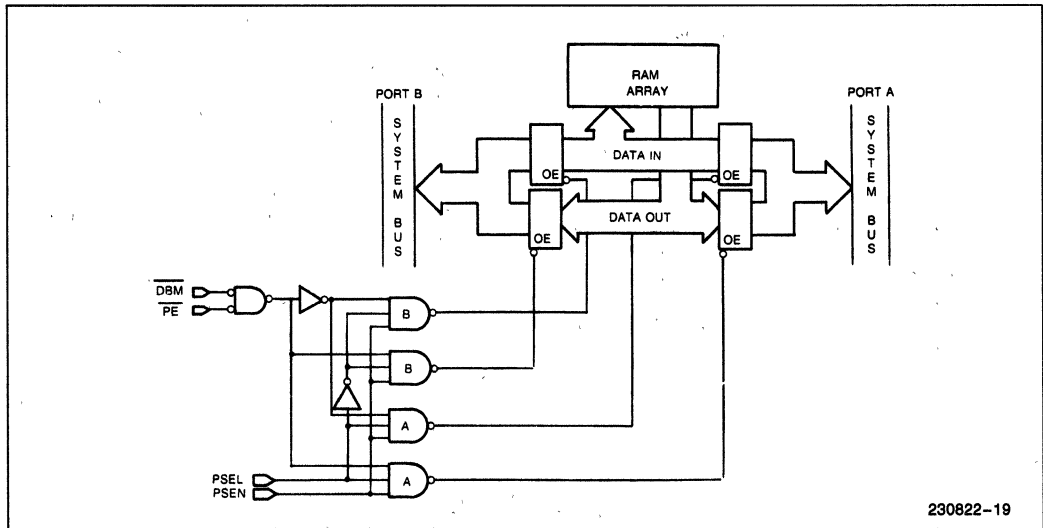


Figure 16. Dual Port Data Bus Control Circuitry

CHAPTER 5

8207 WITH ECC (8206)

This section points out the proper control of the 8206 EDCU by the 8207.

The 8207 performs error correction during read and refresh cycles (scrubbing), and initializes memory after power up to prevent false errors from causing interrupts to the processor. Since the 8207 must refresh RAM, performing scrubbing during refresh allows it to be accomplished without any additional performance penalty. Upon detection of a correctable error during scrubbing, the RAM refresh cycle is lengthened slightly to permit the 8206 to correct the error and for the corrected word to be rewritten into memory. Uncorrectable errors detected during scrubbing are ignored, since the processor may never access that memory location.

Correctable errors detected during a memory read cycle are corrected immediately and written back into memory.

Synchronous/Asynchronous Buses

The many types of configurations that are supported by the 8207/8206 combination can be broken down into two classes: ECC for synchronous or for asynchronous buses.

In synchronous bus systems, performance is optimized for processor throughput. In asynchronous buses, performance is optimized to get valid data onto the bus as quickly as possible (Multibus). While possible to optimize the 8207/8206 for processor throughput in Multibus systems, it is not Multibus compatible. The performance optimization is selected via the XA/XB and SA/SB programming bits.

When optimized for processor throughput, an advanced acknowledge (\overline{AACK} —early or late) is issued at some point (based on the type of processor) so that data will be valid when the processor needs it.

When optimized for quick data access, an \overline{XACK} is issued as soon as valid data is known to exist. If the data was invalid (based on the ERROR flag), then the \overline{XACK} is delayed until the 8206 corrects the data and the data is on the bus.

The first example is known as “correct always” mode. The 8206 CRCT pin is tied to ground and the 8206 requires time to do the correction. Figure 17 shows this implementation. The quick data access method is known as “correct on error.” The CRCT pin is tied to the R/W output of the 8207. When CRCT is high, the 8206 does not do correction, but still checks the data.

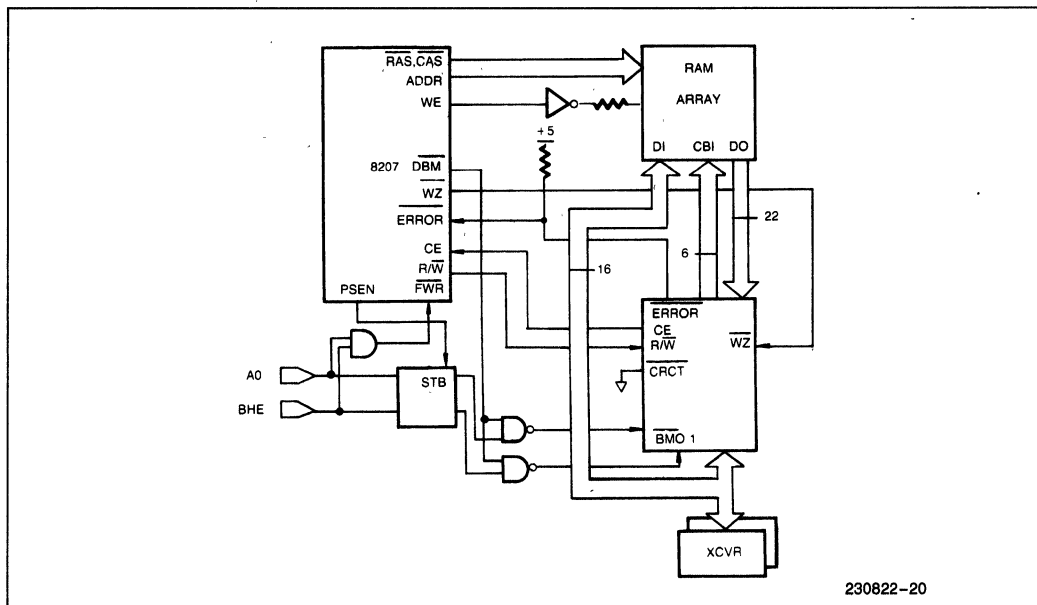


Figure 17. 8206 Interface to the 8207

This delay is typically half of the first. If an error happens, the cycle becomes a RMW and \overline{XACK} is delayed slightly so that data can be corrected.

The correct on error mode is of no real benefit to non-Multibus users. The earliest acknowledge (EAACK) is delayed by one clock to allow for the delays through the 8206. This imposes a 1 wait state delay.

Byte Marks

The only real difference to the 8207 system when adding the 8206 is the treatment of byte writes. Because the encoded check bits apply only to a whole word (including check bits), byte writes must not be permitted at the RAM. Instead, the altering of byte data is done at the 8206. The byte marks previously sent to RAM are now sent to the 8206. These byte marks must also qualify the output enables of the data drivers.

The \overline{DBM} output of the 8207 is meant to be nanded with the processor's byte marks. This output is activated only on reads or refreshes. On write cycles, this output stays high which would force the 8206 byte mark input low. When low, the internal 8206 data out buffers are tristated so that new data may be gated into the device.

Read Modify Writes—ECC

A RMW cycle occurs whenever a processor wants to do byte writes or when the 8207 has detected an error during read or refresh (scrubbing) cycles. A byte write is detected by the \overline{FWR} input to the 8207 and is based on the processor supplied byte marks.

At the start of a RMW cycle, \overline{DBM} stays high, which, when qualified with the byte marks, will enable the data out buffer of the 8206 for the unmodified byte, and tristates the buffer for the new byte; R/\overline{W} is high, which tells the 8206 to do error detection and correcting (if \overline{CRCT} is low). The 8206 can latch data and check bits from the RAM via the STB input, but the

8207 does not use this feature. Instead, the 8207 keeps \overline{CAS} active the entire length of the RMW cycle to hold data at the 8206. The new byte data from the processor goes to the 8206 and to the RAM. The 8207 would have corrected any errors just read, so the old and new bytes of data, plus their check bits, are available at the RAM, and the 8207 generates a write pulse. The data driver for the unmodified byte must not have been enabled, otherwise erroneous data would be written to RAM and possibly made valid (if it was stable) by the 8206.

Data Buffer Control—ECC

The control of the data buffers is essentially the same as in non-ECC systems, with a few exceptions. The processor's byte marks must now qualify the output enable logic. The reason was described earlier in the RMW section. This applies to both single and dual port configurations. A refresh cycle outputs all the control signals that a read cycle will, except for an acknowledge. If complete buffer control is left to the 8207, then it would occasionally (during refreshes) put data on the processor bus. The \overline{DEN} and $\overline{DT/R}$ signals must be qualified by the \overline{PE} input. \overline{PE} would have to be latched for the entire cycle by PSEN.

Test Modes

Neither of the two test modes of the 8207 are to be used in a design. Both test modes reset the refresh address counter to a specific value, which interrupts the refresh sequence and causes loss of data.

In error corrected systems, a reset pulse causes the 8207/8206 to write over the entire RAM array. Test Mode 2 appears to bypass the prewrite sequence. But, the refresh counter is reset to a value of 1F7 (H). So, besides interrupting the refresh sequence, the 8207 still prewrites the 8 locations specified by the counter.

To not overwrite the RAM data, the 8207 RESET will have to be isolated from the system reset logic in ECC systems.

APPENDIX A 8207 PERFORMANCE

The following performance charts were based upon Figure 3 in the 8207 Data Sheet. The charts show the performance of a single cycle with no precharge, refresh, port switching, or arbitration delays.

The read access calculations are: the margin between the 8207 starting a memory cycle to data valid at the processor - 8207 RAS or CAS from clock delay - DRAM RAS or CAS access - 8286 propagation delay - processor set-up.

Assume the RAS/CAS drivers are loaded with 150 pF, and the 8286 is driving a 300 pF data bus.

80286 (example)

$$\begin{aligned} \text{RAS Access: } & 3\text{TCLCL} - 8207 \text{ TCLRSL} - 2118 \\ & \text{iRAC} - 8286 \text{ TIVOV} - 80286 \text{ t8} \\ & = (3)62.5 - 35 \text{ max} - 100 \text{ max} - \\ & \quad 22 - 10 \\ & = 20 \text{ ns} \end{aligned}$$

80186 (example)

$$\begin{aligned} \text{CAS Access: } & 2 \text{ TCLCL} - 8207 \text{ TCLCSL} - \\ & \text{DRAM tCAC} - 8286 \text{ TIVOV} - \\ & 80186 \text{ TDVCL} \\ & = (2)125 - 115 \text{ max} - 85 \text{ max} - \\ & \quad 22 - 20 \\ & = 8 \text{ ns} \end{aligned}$$

8207 Performance (EDC synchronous status interface)

Table 5a. Wait States for Different μP and RAM Combinations

Wait States at Full CPU Speed		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	1-RD, WR 3-Byte WR C0(3)	1-RD, WR 3-Byte WR C0	2-Read 1-Write 3-Byte WR C2	Not Compatible with RAM Parameters(1)
80186, 8086/88-2	8 MHz	1-RD, WR 3-Byte WR C4	1-RD, WR 3-Byte WR C4	1-RD, WR 3-Byte WR C4	
8086/88	5 MHz	1 C6	1 C6	1 C6	1-RD, WR 3-Byte WR C4

Table 5b. μP Clock Frequency for Different μP and RAM Combinations

Maximum Frequency for One Wait-State(4)		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	Full Speed		7.3 MHz C0	6 MHz C0
80286, 8086/88-2	8 MHz			7 MHz C4	
8086/88	5 MHz				

8207 Performance (Non-EDC synchronous status interface)

Table 6a. Wait States for Different μ P and RAM Combinations

Wait States at Full CPU Speed		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	0 C0(3)	1-Read 0-Write C1	1-Read 0-Write C1	Not Compatible with RAM Parameters(1)
80186, 8086/88-2	8 MHz	0 C3	0 C3	0(2) C3	
8086/88	5 MHz	0 C3	0 C3	0 C3	0 C3

Table 6b. μ P Clock Frequency for Different μ P and RAM Combinations

Maximum Frequency for No Wait-State(4)		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	Full Speed	7 MHz	6 MHz	5.3 MHz
80186, 8086/88-2	8 MHz		7 MHz		
8086/88	5 MHz				

NOTES:

1. The DRAM tRAH parameter is not satisfied.
2. 150 ns 64K DRAMs with tCAC = 100 ns won't run with 0 wait-states.
3. Numbers in lower right corners are the programmed configurations of the 8207.
4. To meet read access time.

8207 Performance (MULTIBUS Interface)

This is an *asynchronous, command interface*. Worst case data and transfer acknowledge (XACK #) delays. Including synchronization and data buffer delays, are:

Table 7a. Non-EDC System

	RAM Speed			
	100 ns	120 ns	150 ns	200 ns
Data Access Time	289 ns	299 ns	322 ns	380 ns
XACK # Access Time	333 ns			450 ns

Table 7b. EDC System

RAM Speed				
	100 ns	120 ns	150 ns	200 ns
Data Access Time (Read)	359 ns (324 ns) ⁽¹⁾	369 ns (334 ns)	392 ns (357 ns)	450 ns (415 ns)
XACK# Access Time	400 ns-RD, WR 588 ns-Byte Write			520 ns-RD, WR 806 ns-Byte WR

NOTE:

1. Numbers in parentheses are for when 8206 is in check-only mode (8206 doesn't do error correction until after an error is detected).



October 1986

82C08 User's Manual

SRIDHAR BEGUR

Order Number: 296039-002

CHAPTER 1

1.1 INTRODUCTION

A designer of microprocessor based memory systems has two basic types of devices available to implement random access memory - Static RAMs (SRAMs) or Dynamic RAMs (DRAMs). SRAMs are easy to use, but are comparatively expensive. DRAMs have four times the density of static RAMs, come in smaller packages, consume less power, and cost less per bit. On the other hand, DRAMs require complex support functions, which SRAMs do not, including

- address multiplexing
- timing of address and control strobes
- refreshing, to preserve memory contents
- arbitration, to decide when refresh cycles will be performed vs read/write cycles.

The circuitry required to perform these functions is complex, takes up board space, and initially appears to offset the advantages of DRAMs.

The 82C08 CHMOS DRAM controller provides a highly integrated solution. In addition to performing refreshes, address multiplexing, and control timings, it supports memory bank interleaving for allowing pipelined accesses. The functions are programmable which allow designers to customize their systems. The 82C08 DRAM controller interfaces 100 ns DRAMs to an 10 MHz 80286 processor without introducing wait states. It can also interface a 10 MHz 80186 to 120 ns DRAMS. In addition, the 82C08 supports both power down and battery back up modes, making it useful for low power or portable systems.

1.2 INTRODUCTION TO DYNAMIC RAMS

This section gives a brief introduction to the interfacing requirements of DRAMs. Those who are familiar with DRAM concepts may wish to skip it.

1.2.1 Addressing

The 16 K DRAM can be pictured as a two dimensional array of single bit storage cells, with 256 rows and 256 columns. Each bit of the DRAM is individually addressable. Therefore, a 64K DRAM requires 16 address lines. In order to reduce the number of address pins required, DRAMs time multiplex the addresses into

two halves, over the same pins. Thus a 64K DRAM requires only 8 address input pins. The first address is called row address and the second called the column address. The row address is latched internally by the DRAM on the falling edge of \overline{RAS} (Row Address Strobe) and the column address is latched by the falling edge of \overline{CAS} (Column Address Strobe).

1.2.2 Memory Cycles

The Dynamic RAMs support four different memory cycle types - read, write, read-modify write and \overline{RAS} only refresh. Whether data is read or written during a memory cycle is determined by the RAM's Write Enable control input (\overline{WE}). Data is written only when Write Enable is active.

During read cycles, the \overline{CAS} , in addition to latching the column address, enables the RAM data output when active, assuming \overline{RAS} is also active. When only the \overline{RAS} is active, the data outputs are tristated. This feature allows multiple Dynamic RAM outputs to be tied together.

During write cycles, the write data is latched internally by the falling edge of \overline{CAS} or \overline{WE} , whichever occurs last. If write enable goes active before \overline{CAS} (Early Write), write data is latched by the falling edge of \overline{CAS} . If \overline{WE} goes active after \overline{CAS} (Late Writes), the write data is latched by the falling edge of \overline{WE} .

Late writes are useful in some systems where it is necessary to start the cycle as quickly as possible, by activating \overline{CAS} , to maximize performance, but the CPU cannot get the write data to the DRAMs early enough to be latched by \overline{CAS} . By delaying \overline{WE} , more time is allowed for write data to arrive at the DRAMs.

When late writes are performed by activating \overline{CAS} earlier than \overline{WE} , there exists a period during which \overline{CAS} is active but \overline{WE} is inactive. This is decoded by the DRAMs as a read cycle and it enables its data output. So, if "late write" memory cycles are performed, the data inputs and the data outputs must be electrically isolated from each other to prevent bus contention. If no late writes are performed, the data inputs and the data outputs may be tied together at the RAM to reduce the number of board traces.

1.2.3 Refresh

One unique requirement of Dynamic RAMs is that they have to be refreshed in order to retain data. This becomes evident by examining how the DRAMs are implemented. DRAMs achieve their high density and low cost by employing a single transistor and capacitor pair for storing one bit of information. The presence (or absence) of a charge on the capacitor indicates that the bit is set to a one. This capacitor is selectively accessed for reading or writing by enabling its associated transistor. Unfortunately, if left for very long, the charge will leak out of the capacitor and the data will be lost. To prevent the loss of data, each bit cell must be periodically read, the charge on the capacitor amplified, and the capacitor recharged to its initial state. The circuitry which does this amplification is called a 'sense amplifier'. Typically, most DRAMs have to be refreshed every 2 ms, to prevent loss of data.

Each column in the DRAM has its own sense amplifier, so refresh can be performed on an entire row at a time. Thus, for a 16K DRAM configured as 128 Rows by 128 columns, it is only necessary to refresh 128 rows every 2 ms. The 256K and the 64K DRAMs are implemented such that their refresh requirements are identical to the 16K DRAMs (i.e. 256 rows every 4 ms).

Refresh can be performed by a special cycle called a RAS only refresh. Only the row address for the row to be refreshed is sent. During RAS only refresh cycle, the CAS is never activated, and no data is read or written. Any memory operation will also refresh the row accessed. So if the application can ensure that the entire memory array will be accessed within 2 ms, then no special refresh cycle is necessary.

Three commonly used refresh techniques are

1. Distributed Refresh
2. Burst Refresh
3. Hidden Refresh

1. Distributed Refresh

The Distributed Refresh technique takes advantage of the fact that as long as every row is refreshed every 2 ms the distribution of the individual refresh cycles is unimportant. In the distributed refresh mode, the refresh cycles are performed approximately once every 15 microseconds. This satisfies the refresh requirement of DRAMs while reducing the time the read and write cycles are delayed due to refresh (maximum delay is one memory cycle plus the RAS precharge delay).

2. Burst Refresh

Burst Refresh means waiting almost 2 ms from the last time the refresh was performed, then refreshing the entire memory with a burst of 128 refresh cycles. The advantage of this method is that there is a known period of time when no refreshes are being performed that could delay the memory read or the memory write cycles. On the other hand, the disadvantage is that during the time the refreshes are being performed no read or write cycles can be performed.

3. Hidden Refresh

Hidden Refresh is also called "transparent refresh". This takes advantage of the fact that many microprocessors wait a fixed period of time to decode the first opcode of an instruction after fetching it. This time is necessary to determine what to do next and may be sufficient to complete a refresh cycle. If the status lines of the processor can be examined to determine which cycles are opcode fetches, a refresh cycle can be immediately performed. In this way, the refresh cycles never interfere with read or write cycles, and so appear transparent to the microprocessor. This method has the disadvantage that, if ever the microprocessor stops fetching opcodes for long periods of time, as would be the case during DMA transfers, no refresh cycles will occur, and the data will be lost.

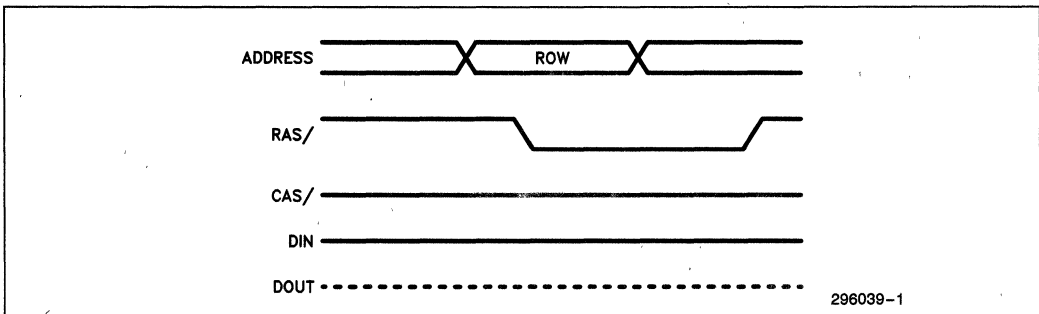


Figure 1.1 RAS-Only Refresh

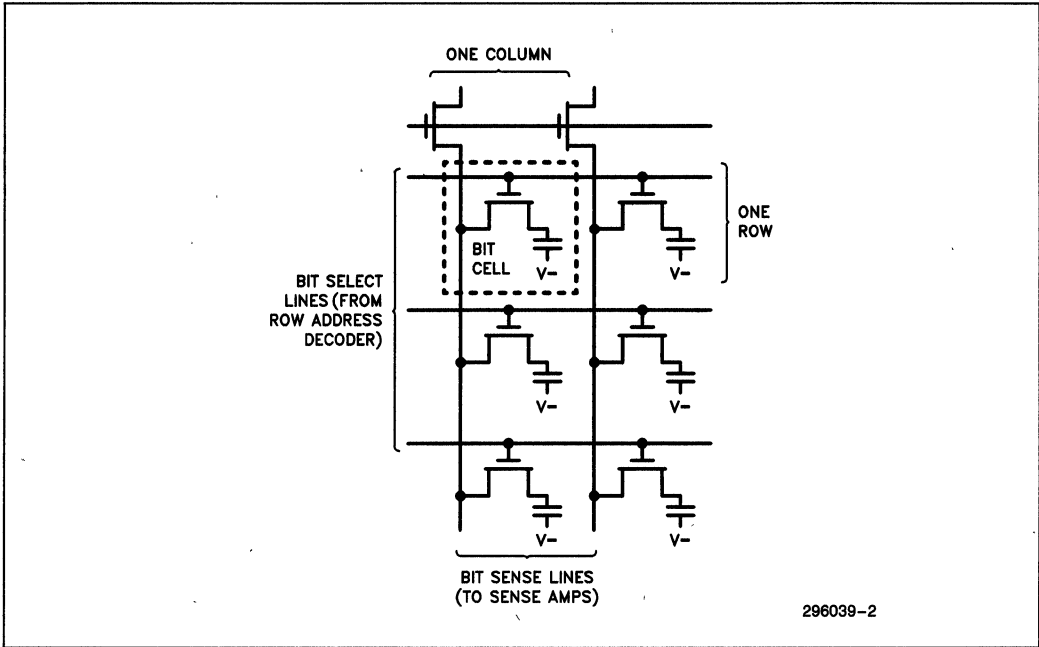


Figure 1.2 Dynamic RAM Cell

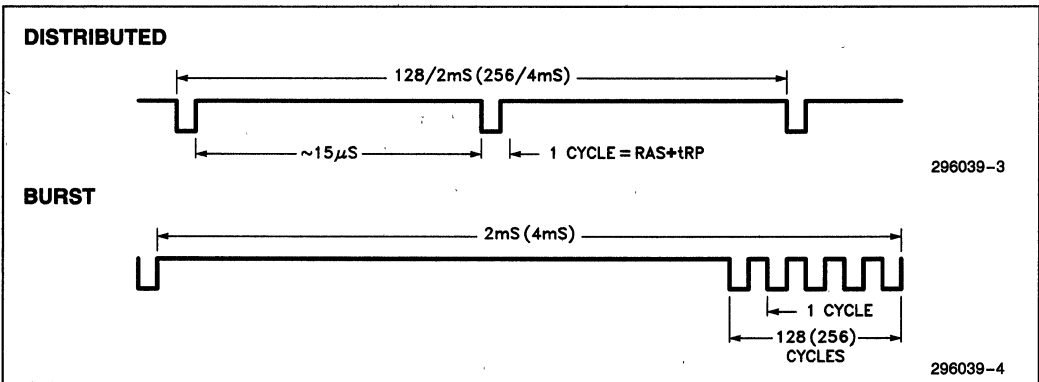


Figure 1.3 DRAM Memory Cycles

CHAPTER 2

PROGRAMMING INFORMATION OVERVIEW

The 82C08 Dynamic RAM controller is intended to support a wide variety of processor and memory configurations. Many of the 82C08's features can be tailored to a given system by means of a serial programming pin. This pin can be strapped either high or low to select one of two default modes or be programmed by means of an external shift register. The external shift register is completely controlled by the 82C08, thereby eliminating the need for local processor support. Nine bits are shifted into the 82C08 to configure up to nine different features. The bits are arranged in the order of increasing importance, using a shift register with less than nine bits permits optimization by programming just those features needed by the application.

Programmable features of the processor interface includes the choice of specifying either synchronous or asynchronous interface and clock compensation. The terms synchronous or asynchronous are conventionally applied to DRAMs depending on whether it exists in a local or a remote environment. In the case of the 82C08, the synchronous operation refers to the case when the incoming commands arrive with a fixed reference to the controller's clock while asynchronous operation refers to the case when the commands arrive with no particular relationship to the controller's clock. The major difference between synchronous and asynchronous operation is that, in the case of asynchronous commands, the controller must first synchronize the incoming commands to its own internal clock. From that point on, the asynchronous operation and the synchronous operation are handled identically. The fastest synchronization time of the 82C08 is one clock cycle, while the slowest synchronization can take up to two clock cycles. As the processor typically requires four or fewer clock periods to complete a cycle, adding a clock for synchronizing reduces the access time by approximately 25% (provided both the processor and the 82C08 operate at the same clock frequency). Synchronous controllers are therefore always preferred when the environment permits them. At the RAM interface, the user can specify fast or slow memory chips, indicate bank configuration and select the optimal refreshing schemes.

2.1 PROCESSOR INTERFACE OPTION

The port control input pin (PCTL) programs the 82C08 to either accept the standard demultiplexed \overline{RD} and \overline{WR} inputs, or directly decode the status outputs of

Intel's iAPX 86, 88, 186 and 188 type processors. The state definitions of the status lines and their timings, relative to the processor clock, differ for the 8086 family and the 80286 processor. Tables 2a, 2b and 2c. illustrates how the 82C08 interprets these inputs, following the programming of the PCTL pin. The 82C08 broadly classifies all processor interfaces into two categories.

- status interface (8086/186 status lines)
- command interface (80286 status or Multibus Bus command)

Table 2a. 80186 Bus Cycle Status Information

S2	S1	S0	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1	0	0	Instruction Fetch
1	0	1	Read Data From Memory
1	1	0	Write Data From Memory
1	1	1	Passive

Status Interface: If PCTL is sampled high following the inactive edge of reset, the 8086 status interface is selected, which allows direct decoding of the 8086/186 microprocessor status lines. As the status lines are activated one clock before the command lines, memory cycles can be initiated earlier. This is the only mode that enables zero waitstate memory accesses.

TABLE 2b. 80286 Bus Cycle Status Information

MIO	S1	S0	Bus Cycle Initiated
1	0	0	Halt Shut Down
1	0	1	Read Data From Memory
1	1	0	Write Data To Memory
1	1	1	None. Not a Status Cycle

TABLE 2c. 82C08 Response

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

Command Interface: If the PCTL pin is low at the end of reset, the command interface is selected. The status lines of the 80286 are similar in code and timing to the Multibus Bus command lines. Command interface should be selected when interfacing to 8086/186 command lines or the command lines of the Multibus Bus, or the status lines of the 80286 processor. When interfacing to the 80286 in the status synchronous mode, the 82C08 directly decodes the status lines to respond to memory requests faster. This allows memory accesses to be performed without incurring wait states.

2.2 PROGRAMMING WORD

The 82C08 requires additional information regarding the type of interface (synchronous/asynchronous), clock rate, speed of dynamic RAMs, in order to optimize its performance. The 82C08 obtains this information via the PDI pin. The 82C08 may be programmed by using an external shift register with asynchronous load capability such as the 74HC165. The reset pulse serves to parallel load this shift register. The 82C08 provides the clocking signal via the multiplexed WEPCLK pin, to shift the data into the PDI programming pin. Nine program data words are read in. The first serial data input corresponds to PD0 bit and the ninth bit shifted in corresponds to the PD8 bit of the program data word.

The PDI input pin can be strapped either high or low to select one of two default modes. By strapping the PDI low, a default of all zeroes is selected. This default mode configures the 82C08 to interface a 8 MHz 80186 (8088, 8086, 80188) to 150 ns DRAMs or a 10 MHz 80186 to 120 ns DRAMs without introducing wait states. By strapping the PDI input high, a default of all ones is selected. When configured in this mode, the 82C08 interfaces a 8 MHz 80286 to 120 ns DRAMs or a 10 MHz 80286 to 100 ns DRAMs without introducing wait states. An external shift register is not required if the system timing requirements are satisfied by the default mode.

The PD0 bit determines if the 82C08 is configured to support the 8086, 80186, 80188, 8088 systems or a 80286 system. When interfacing to the 8086, 8088, 80186, 80188 microprocessors, the 82C08 is said to be operating in the slow cycle mode, and when interfacing to the 80286, it is said to be operating in the fast cycle mode. In the slow cycle mode the 82C08 operates at the same clock frequency as the processor (8 MHz 82C08 for an 8 MHz 80186). When configured in the fast cycle mode, the 82C08 operates at twice the clock frequency of the processor (16 MHz 82C08 for an 8 MHz 80286). The remaining bits PD1-PD8 are then programmed to optimize the 82C08's outputs for the selected configuration.

2.2.1 Programming for 80186/8086 Mode (Slow Cycle)

If the first program data bit PD0 is set to a logic zero, the 82C08 is configured in the slow cycle mode (8088,8086), wherein, it is programmed to operate at a maximum clock frequency of up to 10 MHz. When interfacing to 10 MHz 80186 processors, 120 ns DRAMs must be used to satisfy the timing requirements and to ensure proper memory operation. When the 10 MHz 82C08 is programmed in the default mode, the resulting refresh rate is 11.8 micro-seconds.

If the system design requirements are different from the default configuration, then the 82C08 has to be programmed to support the desired mode, via an external shift register such as the 74HC165. The program data bits PD1 - PD8 have different polarities based on the logic level of PD0 (see Table 2c.1.). For example, if PD0 is set to a logical zero and PD1 is set to a logical one, then the asynchronous processor interface is selected. On the other hand, if PD0 is set to a logical one and PD1 is set to a logical one, then the synchronous processor interface is selected (Refer to Table 2c.1:1f PD0 = 1 and PD1 = 1, then $S = 1$ and $\bar{S} = 0$ this implies synchronous mode). This ensures that the same options are available for both default configurations.

Table 2c.1. Program Data Word

Program Data Bit	PD0 = 0	PD0 = 1	Polarity/Function
PD0	\bar{CFS}	CFS	CFS = 0 Slow Cycle CFS = 1 Fast Cycle
PD1	\bar{S}	S	\bar{S} = 0 Synchronous S = 1 Asynchronous
PD2	\bar{RFS}	RFS	\bar{RFS} = 0 Fast RAM RFS = 1 Slow RAM

CASE 1

The example considered below deliberately assumes a 7 MHz 80186 processor to illustrate the manipulation of the CI0 and CI1 bits to arrive at the optimal refresh rate. Additionally, a single bank memory system is being considered to emphasize the fact that, in the slow cycle mode of operation, consecutive accesses to the same bank will not result in additional wait states.

Processor	80186
Processor Clock	7 MHz
82C08 Clock	7 MHz
DRAM Speed	150 ns
Processor Interface	synchronous
Number of Banks	1

(DRAMs require 256 rows refreshed every 4 milliseconds)

Given the processor, clock frequency and the DRAM speed, the only remaining requirement is to arrive at the corresponding program data word. Referring to Table 2.e for the case described above (7 MHz 80186), CFS' should be set to a zero to indicate that the 82C08 is interfacing to a 8086 type processor and 'FFS' must be set to a one as the processor frequency is 7 MHz (Referring to Table 2.e. in the 82C08 data sheet, clock frequencies greater than or equal to 6 MHz are considered fast). 'CFS' and 'FFS' correspond to PD0 and PD7 bits in the program data word. Therefore, PD0 = 0 and PD7 = 0. (Referring to Table 4a in the 82C08 data sheet, for PD0 = 0, PD7 = FFS/ = 1/ = 0.)

The remaining bits are arrived at, as shown below.

PD1 = 0; synchronous

PD2 = 1; This bit reflects the speed of the DRAMs. 120 ns DRAMs are considered fast when interfacing to 8 MHz 80286 while 150 ns DRAMs are considered slow. 100 ns DRAMs are considered fast for 10 MHz 80286. When programmed in the slow cycle mode, this bit can be set to any value as the same no waitstate performance is achieved by using either the 150 or the 120 ns DRAMs. The processor clock frequency determines the speed of the DRAMs.

PD3 = 1; This case assumes single bank. (Referring to Table 2.d, RB = 0. When PD0 = 0, PD3 = RB = 0 = 1.)

To calculate CI1 and CI0 bits refresh interval should be

Refresh Interval: $\leq 4/256000 \leq 15.6$ microseconds

Clock period = 143 ns

Clock period \times count interval ≤ 15.6 microseconds.

The refresh rate should be selected such that the refreshes are performed often enough to satisfy DRAM requirement, without performing refreshes more often than necessary as there is no benefit. The optimal count interval, for the case when CFS = 0 and FFS = 1 is achieved by setting CI0 and CI1 as zero and one respectively and PLS' to a one. The resulting count interval is 106, and resulting refresh rate is 15.16 us. The program data word for the case above is summarized in Table 2.g.

Table 2d. Bank Selection and Decoding

Program Bit RB	Bank Input	82C08
		RAS, CAS Allocation
0	0	RAS0,1, CAS0,1 to Bank 0
0	1	Illegal
1	0	RAS0, CAS0 to Bank 0
1	1	RAS1, CAS1 to Bank 1

Table 2e. Microprocessor Clock and Frequency Option

Program Bits		Processor	Clock Freq
CFS	FFS		
0	0	iAPX 86, 88, 186, 188	≤ 5 MHz
0	1	iAPX 86, 88, 186, 188	≥ 6 MHz
1	0	iAPX 286	≤ 10 MHz
1	1	iAPX 286	≥ 10 MHz

Table 2f. Refresh Count Interval

Ref Period	CFS	PLS	FFS	Count Interval CI0, CI1			
				00	01	10	11
15.6	0	1	1	118	106	94	8.2

CI0	CI1	Count Interval	Refresh Rate
0	0	118	$118 \times 143 \text{ ns} = 16.874 > 15.6 \mu\text{s}$
0	1	106	$106 \times 143 \text{ ns} = 15.158 \leq 15.6 \mu\text{s}$
1	0	94	$94 \times 143 \text{ ns} = 13.442 < 15.6 \mu\text{s}$
1	1	82	$82 \times 143 \text{ ns} = 11.726 < 15.6 \mu\text{s}$

Table 2g. Program Data Word

Program Data Bit	Level
PD0	0; Synchronous Interface
PD1	0; DRAM speed
PD2	1; or 0 (Don't care term for status mode.)
PD3	1; Single Bank
PD4	0; CI1
PD5	0; CI0
PD6	0; PLS = 1, PLS/ = 1/ = 0
PD7	0; Fast CPU Frequency
PD8	0; Advanced Early Acknowledge

2.2.2 Programming for 80286 Mode (Fast Cycle Mode)

If the first program bit (PD0) is set to a one, the 82C08 is configured to support iAPX 286 systems. In this configuration the 82C08 can operate at frequencies up to 20 MHz. When interfacing to 10 MHz 80286 processors, 100 ns DRAMs must be used to satisfy the timing requirements and to ensure proper memory operation. When the 20 MHz 82C08 is programmed in the default mode, the resulting refresh rate is 11.8 microseconds.

If the system requirements are different from that obtained through the default configuration, the 82C08 has to be programmed via an external shift register to tailor the 82C08's outputs to meet system timing requirements.

Case 2

The example considered below assumes a 5 MHz 286 processor interfacing to the 82C08 in the status synchronous mode. 150 ns DRAMs were selected to highlight that when slower DRAMs are used, the processor frequency has to be reduced to permit zero waitstate memory accesses.

Processor	80286
Processor Clock	5 MHz
82C08 Clock	10 MHz
DRAM Speed	150 ns
Processor Interface	Synchronous
Number of Banks	2

For this example, the default configuration will not satisfy the DRAM's refresh timing requirement. In the default mode the 'CFS', 'FFS' and 'PLS' are all set to ones and the corresponding count interval would be 236 clock periods (Refer to Table 6, 82C08 Data Sheet). At 10 MHz, the resulting refresh rate would be 23.6 (236 × 100) microseconds which would violate the refresh requirement of the DRAMs and result in loss of data.

Referring to Table 2e. for the 5 MHz 80286 synchronous microprocessor interface, CFS must be a 1' and FFS' must be a zero. This translates to PD0 = 1 and PD7 = 0.

The remaining programming bits are arrived at, as follows:

PD1 = 0; Synchronous Interface
 PD2 = 1; Slow RAM, only 120 ns access time DRAMs are considered fast
 PD3 = 1; Two Banks, RAS0/, CAS0/ to Bank0, RAS1 and CAS1 to Bank1

The dynamic RAMs considered in this example require refreshing 256 every 4 ms. The refresh rate chosen by appropriately programming CIO, CI1 and PLS, should be as close to the refresh rate of 15.6 microseconds as possible.

For the case under consideration

Clock Period = 100ns (As the clock input into the Controller is 10 MHz).

CFS = 1; Fast Cycle (80286 processor)

FFS = 0; Slow CPU Frequency as it is less than 6 MHz

PLS = 1; to support refreshes every 15.6 μs

Ref Period	CFS	PLS	FFS	Count Interval			
				00	01	10	11
15.6	1	1	0	148	132	116	100

We have four choices for the count interval.

CIO	CI0	COUNT INTERVAL	REFRESH RATE
0	0	100×146 ns	14.6 μs
0	1	100×132 ns	13.2 μs
1	0	100×116 ns	11.6 μs
1	1	100×100 ns	10.0 μs

The optimal refresh rate is achieved by setting both CIO and CI1 to zeroes.

Two types of memory acknowledge signals are provided by the 82C08. They are the advanced acknowledge (AACK) or the transfer acknowledge (XACK). The PD1 (Synchronous/asynchronous) bit optimizes the AACK for synchronous (EAACK) or asynchronous (LAACK) operation. The transfer acknowledge is a Multibus Bus compatible signal. For the example considered, the program data bit PD8 should be set to a logic one, as the processor interfaces synchronously to the DRAM controller. The resulting program data word for this example is summarized below.

Table 2h. Data Word

Program Level Data Bit	
PD0	1; Fast Cycle (80286)
PD1	1; Synchronous Interface
PD2	0; Slow Ram (120 ns DRAMs are considered Fast)
PD3	1; Two Memory Banks
PD4	1; Count Interval Bits
PD5	1; To achieve a refresh rate of 14.8 microseconds
PD6	1; Long refresh period, DRAMs require 256 rows/4ms
PD7	0; Slow Frequency, < 6 MHz
PD8	1; EAACK

When programmed in this mode, the 82C08 is configured in the C0 configuration, wherein memory accesses do not incur wait states.

2.3 PROGRAMMING OPTIONS

In the slow cycle mode of operation (8086 type processors) only one configuration is supported (C2). The options available to the user are the following:

1. Synchronous or Asynchronous processor interface.
2. In the Asynchronous mode there exists a choice of selecting either LAACK (late advanced acknowledge for command interface) or XACK (transfer acknowledge for Multi Bus type interface). In the synchronous mode of operation only EAACK is generated.
3. Programming the rate at which refreshes are performed.

In the fast cycle mode of operation two configurations are supported - C0, C1. The configuration chosen is dependent on the speed of the processor and the DRAMs. C0 configuration gives the best possible performance (no waitstate memory access) while requiring fast DRAMs. C1 configuration gives relatively slow performance (one waitstate memory access) but requires slow DRAMs. One of two approaches are recommended for arriving at the correct configuration for a given application and to determine the resulting performance.

- METHOD 1
- METHOD 2

2.3.1 Method 1

Given that a decision has already been made regarding the speed of the microprocessor and the DRAMs, then the only remaining requirement is to program the 82C08 and to evaluate the resulting performance. The programming was covered in the 'PROGRAMMING-WORD' section. As an illustration consider the following example.

Case 3:

Processor	80286
Processor Clock	8 MHz
82C08 Clock	16 MHz
DRAMs	150 ns
Process Interface	Status synchronous (Requires EAACK).

Referring to Table 5 (82C08 Data Sheet), 'CFS' and 'FFS' must be set to a logic 1 for an 8 MHz 80286. As 150 ns DRAMs are being used (slow RAMs) RFS must be set to a zero. For this case (CFS = 1, FFS = 1 and RFS = 0), referring to Table 2j., it can be inferred that the resulting configuration is 'C1'.

CONFIGURATION

Table 2i. shows the clock edges which trigger the transition for each of the 82C08s output. Clock 0 is defined as the clock in which the 82C08 begins a memory cycle, either as a result of a port request which just arrived, or of a port request that was stored previously but could not be serviced at the time of its arrival. This can occur if the 82C08 was performing refresh cycle. Figure 2.1 shows the wave form for a memory read cycle.

Table 2i. Timing Chart - Case 1

Cn	Cycle	RAS		ADDRESS		CAS		WE		EAACK	
		L	H	Col	Row	L	H	L	H	L	H
1	RD,RF	0↓	4↓	0↓	4↓	1↓	5↓			2↓	5↓
	WR	0↓	5↓	0↓	4↓	2↓	5↓	1↓	5↓	2↓	5↓

Table 2j. 82C08 Configuration

Timing Conf	CFS (PD0)	RFS (PD2)	FFS (PD7)	Waitstates	Processor Clock
C0	iAPX286(1)	FASTRAM(1)	20 MHz (1)	0*	20 MHz
C0	iAPX286(1)	FASTRAM(1)	16 MHz (1)	0**	16 MHz
C1	iAPX286(1)	SLOWRAM(0)	16 MHz (1)	1	16 MHz
C0	iAPX286(1)	SLOWRAM(0)	10 MHz (0)	0	10 MHz
C2	iAPX186(0)	DON'T CARE	DON'T CARE	0	DON'T CARE

* 100 ns access time DRAMs are considered fast.

** 120 ns access time DRAMs are considered fast.

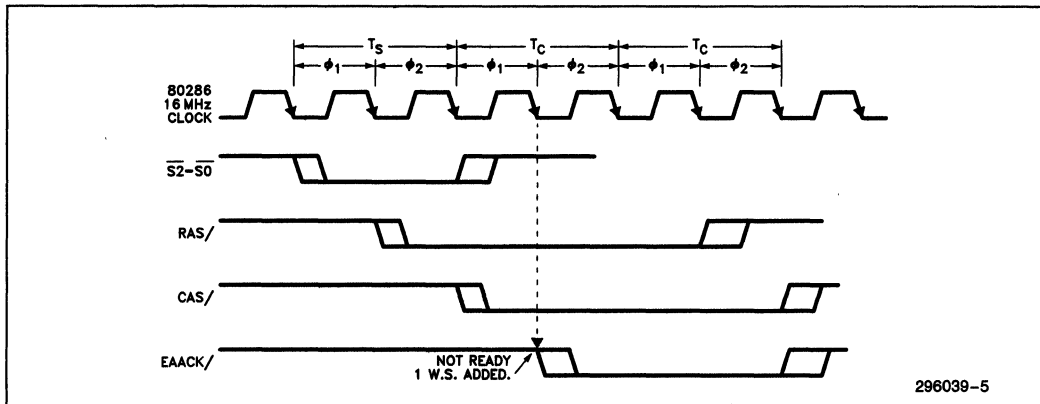


Figure 2.1 Read Cycle Waveform for C1 Configuration (1 W.S.)

PERFORMANCE

As mentioned earlier, the $\overline{\text{AACK}}$ is the “hand-shaking” signal used to tell the microprocessor when it should terminate the current bus cycle. Thus $\overline{\text{AACK}}$ determines how many wait states are introduced. When configured in the C1 configuration, the 82C08 activates the $\overline{\text{AACK}}$ on clock 2 (Clock 0 being the clock edge that activates RAS). In the 80186 synchronous mode of operation Clock 0 is the falling edge of Clock T1 and in the 80286 mode Clock 0 corresponds to the PHI1 falling edge of the TS cycle) for both the read and write memory cycles. If no wait states are to be introduced, the 82284 clock generator (it generates the clock for both the 82C08 and the 80286) expects $\overline{\text{AACK}}$ to be activated during Clock 1, so as to ensure a no waitstate operation. As the $\overline{\text{AACK}}$ is activated on Clock 2, in the C1 configuration, all read or write memory accesses incur one waitstate (except when memory requests arrive when a refresh cycle is in progress).

2.3.2 Method 2

This approach is recommended if the performance level is known (e.g. zero waitstate memory access) and it is required to determine the appropriate processor interface and speed of DRAMs required to support this desired level of performance. This can be illustrated by considering an example.

Case 4

Processor	8 MHz 80286
Processor Clock	8 MHz
82C08 Clock	16 MHz
Requirement	Zero waitstate memory access.

To achieve zero waitstate memory access, the following factors have to be considered.

1. Processor Interface.
2. Speed of DRAMs.
3. Number of banks of memory.

1. Processor Interface

The 80286 microprocessor must interface in the status synchronous mode to the 82C08 to get no waitstate operation. By status synchronous mode we mean that the 82C08 and the 80286 must run off the same clock (16 MHz for 8 MHz 80286) and the status lines of the 80286 processor should be directly connected to the $\overline{\text{RD}}$, $\overline{\text{WR}}$ inputs of the 82C08. When programmed in the status synchronous interface mode, the 82C08 will internally decode the status lines. As the status lines are generated two clock cycles before the $\overline{\text{RD}}$, $\overline{\text{WR}}$ commands (82288 bus controller generates the read or write commands), the memory cycle can be started earlier.

If the asynchronous interface is chosen instead, there is a performance penalty since the 82C08 has to internally synchronize the command. The internal synchronization can take up to two clock cycles which will delay the start of the memory cycle and results in wait states. The synchronous status interface should be selected for no waitstate performance when permitted by the system environment.

2. Speed Of DRAMs

The DRAM speed and the processor clock frequency determine the configuration. The 82C08 supports two configuration - C0, C1. The C0 configuration gives the best possible performance (based on $\overline{\text{RDY}}$) but as mentioned earlier, require fast DRAMs (at 8 MHz, 120 ns). So with the 80286 processor operating at 8 MHz, 120 ns DRAMs are required to achieve a zero waitstate performance.

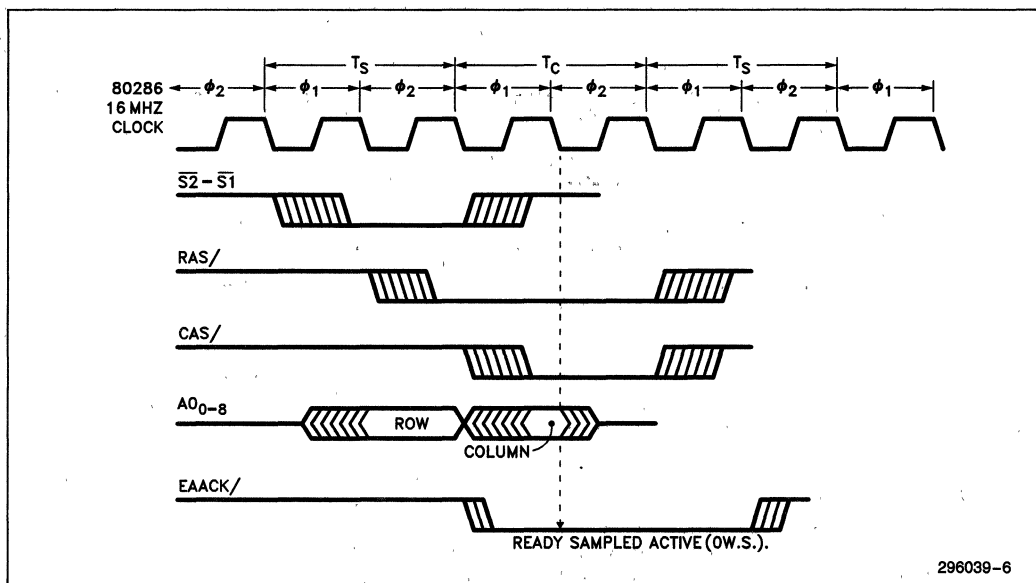


Figure 2.2 80286/82C08 Read Cycle for C0 Configuration (0 W.S.)

Table 2k. Timing Chart - Case 4.

Cn	Cycle	RAS		ADDRESS		CAS		WE		EAACK	
		L	H	Col	Row	L	H	L	H	L	H
0	RD,RF	0↓	3↓	0↓	3↓	1↓	3↓			1↓	4↓
	WR	0↓	4↓	0↓	3↓	2↓	4↓	1↓	4↓	1↓	4↓

3. Number Of Banks

In the fast cycle mode of operation, there will be a performance penalty when using a single bank of memory, as consecutive memory accesses have to wait for the RAS pre-charge delay. By organizing memory into banks so that sequential addresses are in different banks, the RAS precharge time of one bank can be hidden behind the access to the other bank. This is known as "Interleaving" and is explained in detail in Chapter 3.

CONFIGURATION

In conclusion, for the example considered above, the 82C08 has to be configured in the status synchronous mode and use two banks of 120 ns DRAMs, to get the desired performance level. Referring to Table 5 (82C08 Data Sheet), the 'CFS' and 'FFS' bits must be set to a logic one, for an 8 MHz 286. If 120 ns DRAMs are used, the 'RFS' bit should be set to a logic one as well. For this case, when 'CFS', 'FFS' and 'RFS' are set to a

one, the 82C08 is programmed to operate in the C0 configuration. Table 2k. shows the clock edges which trigger the transition, for each of the outputs. The waveform for a read cycle is shown in Figure 2.2.

PERFORMANCE

Table 2k. illustrates the clock edges which trigger transitions on each 82C08 outputs. The synchronous port interface waveforms (82C08 data sheet, Page 29) shows the requirements that have to be met, in order for the command to be recognized by the 82C08. If the setup and hold times are met, then the command is initiated.

The performance can be determined from the waveforms. The activation of the advanced acknowledge (Early for Synchronous interface, late for Asynchronous interface) determines the total number of wait states introduced. As mentioned in the previous example, if no waitstate memory accesses are required, the EAACK should be activated during Clock 1, which is the case when the 82C08 is configured in the C0 configuration.

CHAPTER 3

RAM INTERFACE

Intel's 82C08 Dynamic Ram Controller reliably performs all the support functions required by DRAMs. The 82C08 accepts the processor address and internally multiplexes it into the row and column address. The only other requirement is to have series resistors on all the 82C08 outputs and to ensure proper layout of the traces. The 82C08 is capable of addressing 16K, 64K and 256K DRAMs. It can support two banks of memory with each memory bank having its own individual $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ strobe signals. If both banks of memory are not occupied, the 82C08 can reassign the timing of $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ strobes to allow accessing wider data words without increasing the loading on the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ drivers.

The 82C08 contains an internal address counter used for generating refresh cycles. The 82C08 has 18 address lines (AL0–AL8, AH0–AH8). These address lines are internally multiplexed to generate 9 address outputs. The 9 most significant bits (AH0–AH8) form the ROW address and the 9 least significant bits (AL0–AL8) form the COLUMN address. There is a bank select input for selecting one of the two banks of memory. This input is decoded internally to generate the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ outputs for the appropriate bank.

3.1 REFRESH INTERVAL

The 82C08 supports four different types of refresh options as described in the REFRESH OPTIONS section (CHAPTER 2). The rate at which refreshes are performed is programmable. This programmability is necessary because the refresh period is a function of the clock input into the 82C08. As the clock frequency may vary over a wide range, the rate at which refreshes are performed will vary as well. Programming the cycle Fast/Slow (CFS) bit and frequency Fast/Slow bit (FFS) automatically reprogram the refresh counter to generate the correct refresh interval for clock frequencies of 16, 10, 8 and 5 MHz (CFS, FFS = 11, 10, 01, 00 respectively). For clock frequencies between those specified above, the Count Interval programming bits (C10, C11) allow "fine tuning" of the refresh interval. Refreshes should be performed just often enough to satisfy DRAM's requirements. Performing refreshes more often than necessary would result in wasting memory bandwidth.

3.2 REFRESH DURING POWER DOWN

The 82C08 supports a power down feature which will be discussed in detail in Chapter 8.

3.3 REFRESH COUNTER

The 82C08 has a 9 bit internal address counter. For standard DRAMs, the 82C08 can support DRAMs that require refreshing either 128 rows every 2ms or 256 rows every 4ms or 512 rows every 8ms. In addition, there exists the option of refreshing 256 rows every 2ms via the period Long/Short (PLS) programming option.

3.4 $\overline{\text{RAS}}$ AND $\overline{\text{CAS}}$ REALLOCATION

The 82C08 address output lines are designed to drive up to 32 DRAMs (assuming the capacitance presented by the DRAMs on the address lines are no greater than 5 pF). The $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ outputs can drive up to 16 DRAMs. Under these conditions, the 82C08 will meet all of the RAM timing requirements.

The 82C08 can accommodate other configurations, like a single 32 bit memory bank. This would require the 82C08 address output lines to drive a total of 32 DRAMs, which is well within its drive capability, but the 32 DRAMs per bank exceeds the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ driver's limit. When the capacitive loading requirements are exceeded, the 82C08 will fail to meet critical DRAM parameters like Row Address Hold time (t_{RAH}) and Column Address Setup time (t_{ASC}).

In order to prevent violating these DRAM timing parameters, the 82C08 can be programmed to reallocate the timings of the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines to accommodate the increased load. When the program data bit RB (PD3), is set to logic zero, the 82C08 is programmed to support single bank memory systems. It activates $\overline{\text{RAS}}_0$, $\overline{\text{RAS}}_1$ as a pair and $\overline{\text{CAS}}_0$, $\overline{\text{CAS}}_1$ as a pair. With this scheme the address drivers would be loaded by 32 DRAMs and individual $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines by 16 DRAMs. This relative loading is identical to the first case of two banks, with each bank supporting 16 DRAMs. Drive reallocation allows addressing wider data words while maintaining optimal memory timings.

3.5 INTERLEAVING

All DRAMs require a recovery period for precharging internal lines after each access. If the processor tries to immediately reaccess the RAM, it is the responsibility of the controller to delay it until the precharge time is over. By automatically organizing memory into banks such that sequential addresses go to different banks, the 82C08 is able to hide the $\overline{\text{RAS}}$ precharge time of one bank behind the access time to the next bank. This organization is achieved by using the least significant bit of the processor's address to select the bank of RAM. If

a break in the program flow occurs, such would be caused by encountering a jump or a call instruction, it raises the probability that the same bank may be immediately reaccessed. When accesses are made to the same bank, the start of the new cycle is delayed by the $\overline{\text{RAS}}$ precharge time. To compensate for the $\overline{\text{RAS}}$ precharge delay and to ensure proper memory operation, wait states have to be added. Once the row and column addresses to one bank have been latched by the DRAMs, the controller sends the row address for the next cycle.

The 82C08 internally monitors memory accesses to determine if they are to the same bank. If so, it waits a certain number of clock periods before activating the $\overline{\text{RAS}}$ for the memory cycle (Table 10 in the 82C08 Data Sheet lists the tRP time for the different configurations).

Figures 3.3 and 3.4 illustrate the effect of the performance of the processor with and without interleaving (Figure 3.3 illustrates the impact on performance with and without interleaving for the 80286 microprocessor and Figure 3.4 illustrates the impact on performance on

the 80186 microprocessor). For 80286 microprocessor interface, consecutive accesses to the same bank will result in one wait state for the second access, but no wait states are added if the accesses are to alternate banks.

If the 82C08 is configured in the slow cycle synchronous mode of operation (default mode by strapping PDI to GND or by setting the program bits CFS and S to zeroes) it performs consecutive memory accesses to the same bank without incurring wait states.

3.6 WRITE ENABLES -BYTE MARKS

The write enable supplied by the 82C08 cannot drive the RAM array directly. For eight bit systems, this output should be inverted before interfacing to the RAMs. For sixteen bit systems, this output should be gated by the byte marks supplied by the processor. The byte marks are generated by decoding the latched A0 and $\overline{\text{BHE}}$. This buffer should be capable of driving very large capacitive loads (eg. 74S32).

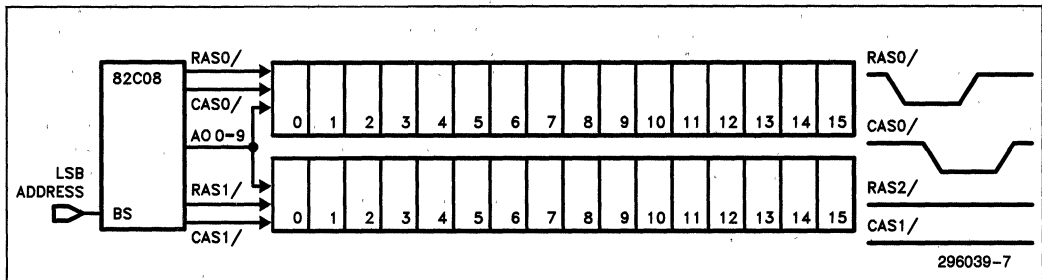


Figure 3.1 82C08 2 Bank Configuration

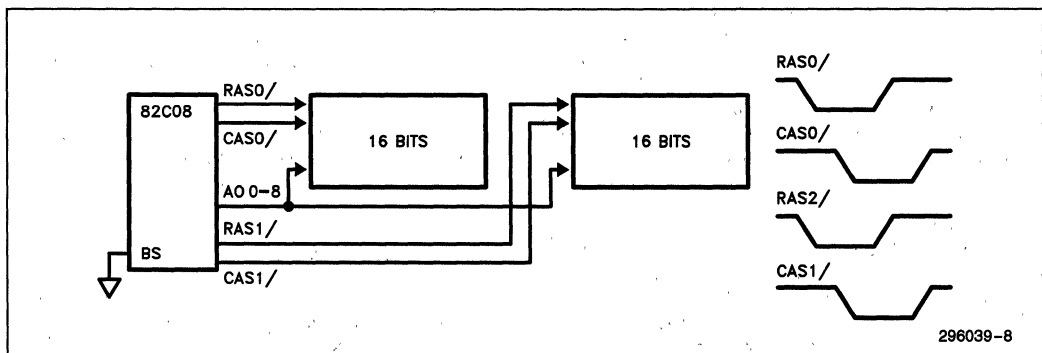


Figure 3.2 82C08 Single 32-BIT Bank Memory

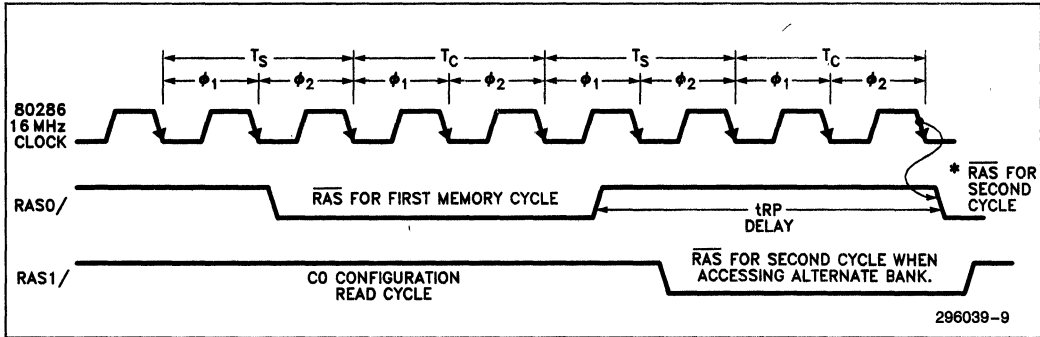


Figure 3.3 Processor Performance With and Without Interleaving (80286)

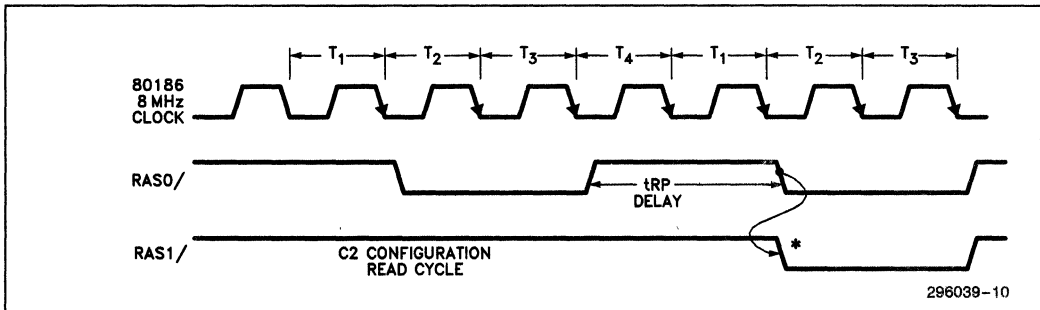


Figure 3.4 Processor Interface With and Without Interleaving (80186)

3.7 RAM CYCLES AND TIMINGS

The 82C08 operates in three configurations - C0, C1 and C2 - depending upon the programming of the CFS (PD0), RFS (PD2) and FFS/ (PD7) bits in the program data word (See section on Programming Information, Chapter 2). Table 3a. illustrates the relationship between the program bits and the resulting configuration. The configuration determines on which clock edge each of the 82C08 outputs are activated. The C0 and the C1 configuration is for the fast cycle mode of operation (CFS and FFS = 1,1.. implies command interface or MB type interface). In the slow cycle mode of operation (CFS = 0.. 186 type interface) only C2 configuration is supported. The configuration, along with the timing wave forms and the A.C. parameters, allows the user to calculate the DRAM and processor interface timings. Table 10 of the 82C08 Data Sheet precalculates the dynamic RAM timings for each of the 82C08's configuration. The numerical values can be arrived at by substituting values for the parameters.

Table 3a.

Timing Conf	CFS (PD0)	RFS (PD2)	FFS (PD7)	Waitstate
C0	iAPX286(1)	FASTRAM	20 MHz (1)	0
C0	iAPX286(1)	FASTRAM	16 MHz (1)	0
C1	iAPX286(1)	SLOWRAM	16 MHz (1)	1
C0	iAPX286(1)	SLOWRAM	10 MHz (0)	0
C2	iAPX186(0)	Don't Care	Don't Care	0

3.8 REFRESH OPTIONS

In Chapter 1 we briefly discussed Distributed, Burst and Hidden refresh. We will now consider the types of refresh schemes the 82C08 supports. The 82C08 supports 5 types of refresh schemes and the appropriate one can be chosen for a given system application.

Internal Refresh. This refresh option takes advantage of the fact that the distribution of the individual refresh cycles is unimportant. The 82C08, when programmed for this mode, performs a refresh cycle every 2 ms/128 or approximately once every 15 us. In this way, the refresh requirements of the DRAMs are satisfied, but the longest time that read or write cycles are delayed (because of refresh) is minimized. This is the most popular refresh method as the refreshes are performed by the 82C08 and is transparent to the CPU. This mode is invoked by strapping the 'RFRQ' pin high. The 82C08, when configured in this mode will perform refreshes automatically, without the need for system intervention. The rate at which refreshes are performed is programmable and is determined by the CFS (PD0), PLS (PD6), FFS (PD7) CI0 and CI1 (PD4, PD5) bits in the program word.

External Refresh With Fail Safe. This option takes advantage of the fact that processors wait a fixed length of time after fetching the opcode of the first instruction, to decode it. This latency time may be sufficient to complete a refresh cycle. If the status outputs of the CPU can be examined to determine which memory cycles are opcode fetches, then a refresh cycle can be immediately performed. This will ensure that the refresh cycle never interferes with read or write cycles. On the other hand, the disadvantage is that if the processor ever stops fetching opcodes for a long time, which would be the case during DMA transfers, then data would be lost. But due to the fail safe option, the Controller would automatically perform the refresh cycle, when the internal timer expires. This option is ideal for low performance microprocessors, operating at relatively slow clock frequencies. Today's high performance processors do not allow sufficient time between opcode fetches, to complete a refresh cycle.

To invoke this mode, it is necessary to hold the 'RFRQ' input high until after reset. An external refresh request is recognized, if the RFRQ input makes a low to high transition. A high to low transition has no effect on the 82C08's refresh logic.

External Refresh Without Fail Safe. This allows the user to generate refresh request at their convenience. To invoke this mode, the RFRQ pin must be held low until after reset. If the refresh request is sampled low at the falling edge of reset, the internal timer is deactivated. To generate a single refresh cycle, the RFRQ must make a low to high transition and be sampled high by the falling edge of the 82C08's clock. All refresh requests are internally synchronized. As refresh requests

arrive asynchronously, it can take the 82C08 a maximum of two clock periods (minimum of one clock period), to internally synchronize the refresh request. An external refresh request has to be held high for a period greater than the 82C08's clock period. On the other hand, if the refresh request is seen high for more than one clock cycle, a burst refresh cycle may be initiated. To avoid the occurrence of a burst cycle, it is recommended that the refresh request be externally synchronized to the 82C08's clock. If the external refresh logic fails to generate a refresh request, no refresh cycles will be performed. This puts restrictions on the system design, because a failure to generate a refresh request might result in the loss of DRAM data.

Burst Refresh. Burst refresh means waiting almost 2 ms from the last time refresh was performed, then refreshing the entire memory with a burst of 128/256 refresh cycles. The advantage of this type of refresh is that there are known periods of time, when no refresh cycles are performed, and so cannot interfere with normal read or write cycles. The inherent disadvantage, however, is that during the burst refresh time, no read or write cycles are performed. This severely limits the worst case response time to interrupts, and makes this approach unsuitable for many applications.

Burst Refresh is implemented in a fashion similar to that of 'external refresh without fail safe'. The RFRQ' must be sampled low until after reset. To perform a burst refresh cycle, the refresh request pin 'RFRQ' needs to be sampled high for at least two clock cycles. The 82C08 will cause a burst of up to 128 consecutive refresh cycles. These cycles are internally counted. The 82C08 has a 9 bit internal refresh address counter. The address counter is incremented following every refresh cycle. The 82C08 terminates the burst refresh when the refresh address counter reaches the value XX1111111 (X = don't care). If prior to the burst cycle, the refresh address counter is at XX1111110, then only two refresh cycles are performed.

No Refresh. This option takes advantage of the fact that any memory read or write cycles also refreshes the row addressed. So, if an application can guarantee that the entire memory array will be accessed every 2 ms, like in Bit-Mapped graphics applications, then no special refresh cycles need be performed.

This mode can be invoked, by strapping the 'RFRQ' input low. This is similar to 'External Refresh without Failsafe'. No refresh is accomplished by keeping refresh request low.

IMPACT OF REFRESH ON MEMORY BANDWIDTH

If a memory request coincides with a refresh cycle that is already in progress, the memory cycle will not be granted until the completion of the refresh cycle and the $\overline{\text{RAS}}$ precharge delay.

Assuming distributed refresh:

Delay due to refresh = 1 memory cycle ($\overline{\text{RAS}}$ cycle time) + $\overline{\text{RAS}}$ precharge time.

In the 'C2' configuration

$$= 2 \text{ TCLCL} + 2\text{TCLCL} - \text{T25}$$

$$= 4 \text{ TCLCL} - \text{T25 (where T25 = } \overline{\text{RAS}} \text{ active delay)}$$

$$= 4 \text{ TCLCL} - 35 \text{ ns.}$$

at 8 MHz

$$= 500 - 35 = 465 \text{ ns.}$$

The resulting impact on the memory bandwidth due to refresh cycle will be: $(465 \text{ ns}/15.6 \text{ } \mu\text{s}) \times 100 =$ approximately 3%.

CHAPTER 4

PRINCIPLES OF OPERATION

In this chapter the internal operation of the 82C08 is briefly explained. A review of the 82C08 data sheet, together with a basic understanding of the various processors and supports, is recommended (Microsystems Components Handbook, Volume 1.).

The 82C08 architecture was designed for the highest possible performance. The 82C08 achieves this high performance by responding to commands as soon as possible, so that data will be valid on the bus in time for the CPU to sample it, without adding wait states. The 82C08, when interfacing to Intel's family of microprocessors, directly decodes the processor's status lines which are valid prior to the command lines of the bus controller, and this enables the 82C08 to start the memory cycle earlier. The 82C08 activates RAM signals on the same clock edge that samples the active edge of \overline{RD} , \overline{WR} and PE (or status lines). This requires that all of the 82C08 blocks that have to issue an output immediately off the command, individually decode the status lines in parallel with other blocks.

4.1 DESCRIPTION OF OPERATION

The operation of the 82C08 can be broadly classified into three stages.

1. Immediately after reset, the 82C08 programs itself to support a particular mode of operation to efficiently support the system environment.
2. After completing the programming, the 82C08 performs 8 warm-up (refresh) cycles, to initialize the memory.

3. After completing the initialization, the 82C08 is ready to accept memory commands and refresh cycles upon receiving a refresh request, which can be generated either internally or externally.

4.2 INITIALIZATION AND PROGRAMMING

The Initialization of the 82C08 consists of three major steps.

4.2.1 Latching the States of Certain Inputs at Reset

This is accomplished by two internal latches. Latching the state of port control programs the 82C08 into the status or command interface. Latching the state of the \overline{RFRQ} input programs the 82C08 to operate in the failsafe or non-failsafe refresh mode of operation.

4.2.2 Loading The Program Data Word Into the 82C08

This step is controlled by the INITIALIZATION machine, which accomplishes this task via a handshake with the PDI block. The nine programming bits are loaded as follows:

- 1) The INIT machine generates two signals
 - a) IPCLK (program clk), which clocks the external parallel load shift register.
 - b) SHPDI (shift PDI) which clocks the internal parallel load shift register.

The relationship between these signals is shown in Figure 4.1.

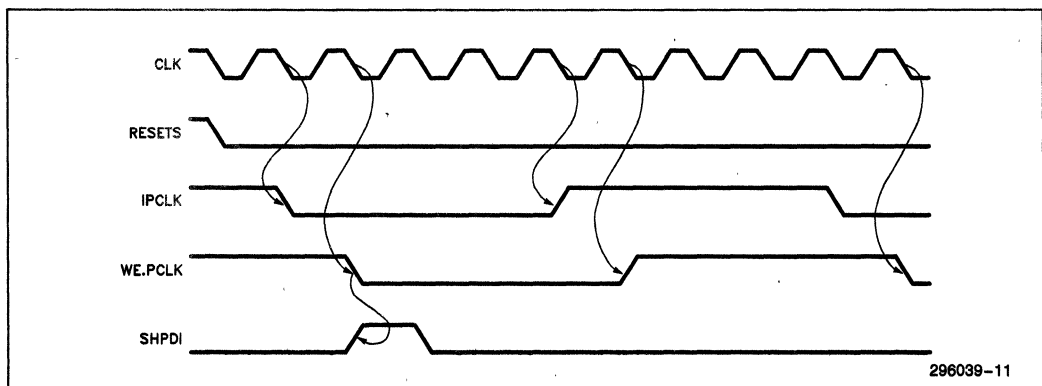


Figure 4.1 PCLK, SHPDI Timing

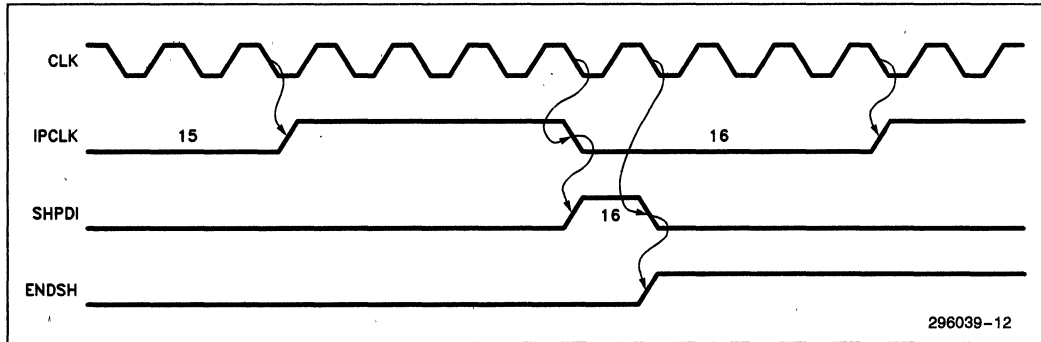


Figure 4.2 ENDSH Timing

- After the PDI block completes shifting in the 9 bits, it generates the ENDSH (end shift) signal. This signal causes the INIT machine to stop the generation of IPCLK and the SHPDI signals, and move to the third step in initialization. The relationship between these signals is indicated in Figure 4.2.

4.2.3 Memory Initialization

After the 82C08 completes loading the program data bits, it starts memory initialization by performing 8 memory warm up cycles. The memory initialization is performed as follows:

- Immediately following RESET, the INIT' machine signal notifies the remaining circuitry that the 82C08 is in the memory initialization period.
- This signal starts the refresh time out' counter. As long as the INIT' machine signal is active, the refresh time out counter is programmed to count intervals of 32 clock periods. The interval was chosen to be 32 clock periods to minimize the power consumption during initialization. At the end of each interval a refresh request signal is generated.
- Each time the refresh request signal is issued, a refresh cycle is started.
- At reset, the refresh address counter, which generates ROW addresses in refresh cycle, is set to the value of (FFF8H) and is incremented at the third clock period of every refresh cycle. When the refresh address counter reaches the state of (FFFFH), it issues a pulse to the INIT machine signifying the completion of the initialization period. Thus, 8 memory warm up cycles are performed.

External request will not be recognized by the 82C08 until the completion of the programming and the memory warm-up cycle. Based on the above information, we can compute the time the processor must wait, after reset, before accessing memory. Additionally, the user must ensure that the power down detect signal is not activated during initialization as the 82C08 ignores the PDD input during initialization.

82C08 System Response: $T_{RESP} = T_{PROG} + T_{PREP}$

Where $T_{PROG} = 9$ Program bits (4) + Delay in activation of the shift clock following the inactive edge of reset, which could take a maximum of 4 clock periods.

$= (40) TCLCL$; $TCLCL =$ Clock period.

and $T_{PREP} = (32)(8)(TCLCL)$; a refresh request is generated every 32 clock periods, to minimize power consumption, during initialization.

$T_{RESP} = (296)(TCLCL)$; if $TCLCL = 125ns$,
 $T_{RESP} = 37$ Microseconds.

4.2.4 Special Notes on Initialization

- During initialization, the 82C08 only registers, but does not service commands. No power down requests will be served during initialization.

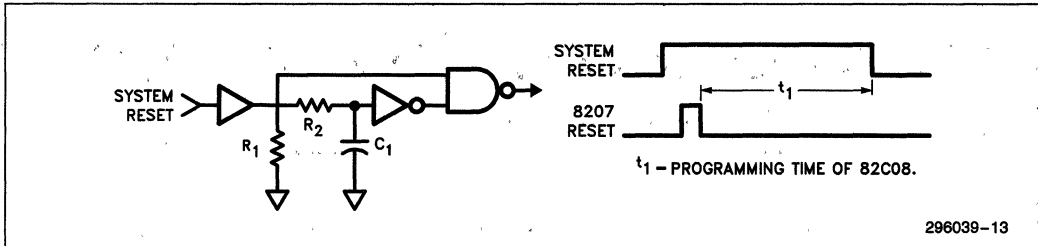


Figure 4.3 Differentiated Reset Circuitry

2. During reset, the 82C08 defaults to synchronous port interface. If the desired interface is different from the default, then commands should be delayed until programming is complete. Otherwise, the 82C08 might fail to recognize the command.
3. During initialization the ARBITER circuitry is forced into the refresh mode.

The system design should inhibit the processor from accessing memory during the initialization period. A simple means of preventing commands or status from occurring (if you cannot guarantee the TRESP period) during the initialization period is to differentiate the system reset pulse, to obtain the reset for the 82C08. The circuitry required to accomplish this, is shown in Figure 4.3. The total time of the 82C08 reset pulse and the 8208 programming must be less than the time it takes for the CPU to issue a memory command. The differentiated reset pulse resets the 82C08 and completes the initialization while the reset of the system is still in reset.

4.3 BLOCK DIAGRAM

Let us look at the detailed block diagram in Figure 4.4 to see how the 82C08 satisfies the timing requirements of the DRAMs.

4.3.1 Address Multiplexing

Address multiplexing is performed by a three to one multiplexer in the 82C08. The three inputs are the ROW address (AH0-AH7), COLUMN address (AL0-AL7), and the internally generated refresh row addresses. In the idle state, the multiplexer selects the Row address, in order to be ready to start a memory cycle. When a refresh request is generated, either internally or externally, the multiplexer switches to select the refresh row address. In the middle of the refresh cycle, the multiplexer switches back to select the row address, but ensures that the refresh addresses are held valid long enough to meet the RAM timing requirements. This feature allows the 82C08 DRAM controller to be ready to start the next memory cycle.

4.3.2 Refresh Circuitry

The refresh circuitry is responsible for generating internal refresh request and to register external refresh request. The operation of the refresh circuitry is summarized below.

The refresh operation is controlled by three blocks:

1. Refresh Control Block
2. Refresh Time Out Counter
3. REFRESH ADDRESS Counter.

1. At reset, the logic level of the 'RFRQ' pin is internally latched by the Refresh Control Block to determine the refreshing mode.

2. During initialization sequence the refresh circuitry generates a refresh request once for every 32 CLK periods.

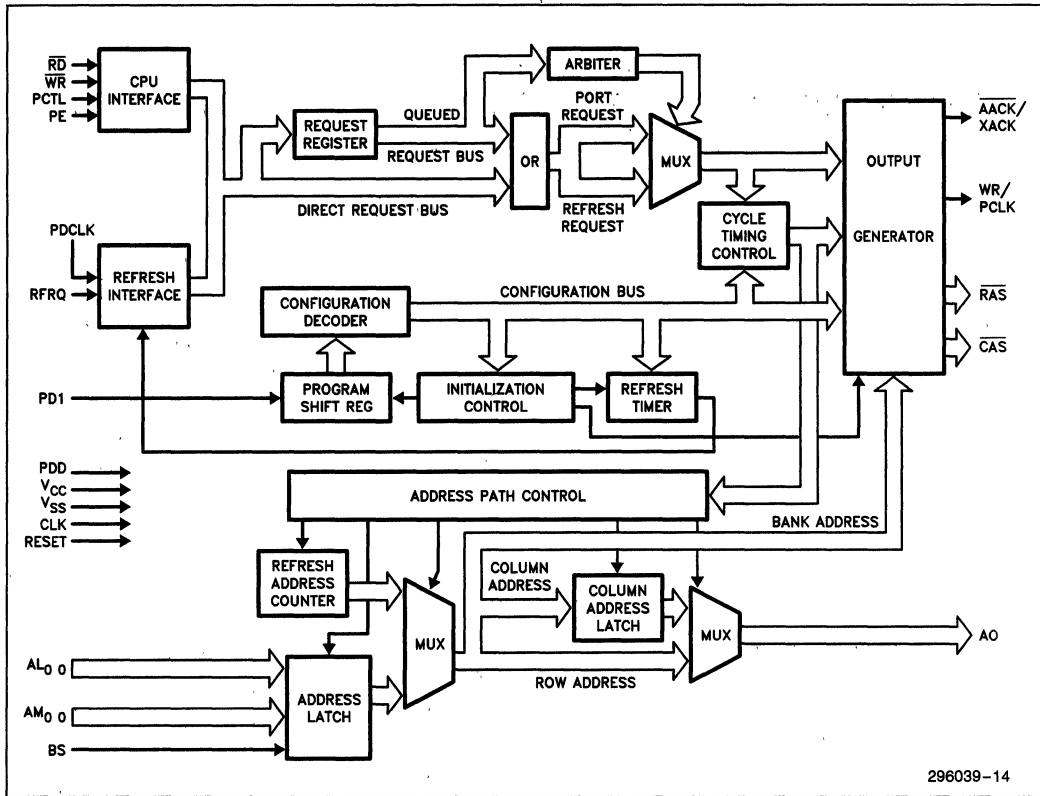
3. After initialization, the Refresh Time Out counter is enabled. The counter interval is determined by the programmed mode values. As the initialization cycle is complete, the count interval is no longer 32 clock periods. If the 82C08 is programmed in the non-failsafe mode, the Refresh Control Block ignores the REFTO counter output, thereby logically disabling the Refresh Time Out Counter.

4. Upon detection of a refresh request, the 82C08 registers the request into an internal register and requests the arbiter for a refresh cycle. The arbiter receives the request one clock period later.

5. If the 82C08 is programmed in the failsafe mode, the refresh counter is reset upon detection of a refresh request, and starts counting the refresh interval from the beginning. If the request was generated internally, then REFTO counter resets itself automatically.

6. The arbiter then activates signals that enable the start of the memory cycle.

7. In the case of burst refresh, the arbiter is locked on to selecting the refresh port. At the termination of the burst cycle, the arbiter switches to select the Row Address Buffer.



296039-14

Figure 4.4 Block Diagram

4.3.3 Arbitration

The 82C08 has to arbitrate between two ports - the microprocessor port and the refresh port. The refresh port is an internal port, dedicated to servicing refresh requests. The selected port has direct access to the timing generators, which are responsible for generating RAM signals. A port has to be selected by the arbitration logic before its RAM cycle can be initiated. Typically, a request on the unselected port incurs a two to three clock period delay before it's memory cycle is initiated, whereas, the selected ports memory request is started immediately. Under normal condition the arbitration time of a refresh cycle is hidden behind the RAM cycle of the selected port, so that the next cycle can be started immediately following the completion of the present memory cycle.

The arbitration rules are summarized below:

- If only one port is requesting service, then that port, if not already selected, becomes selected.
- When no service requests are pending, the processor port is selected.
- During initialization and power down mode, only the refresh port is selected.
- If both the refresh and the processor port request memory access at the same time, then the port that is currently selected is granted memory access.
- During power down, the arbiter is locked on the refresh port.

This arbitration arrangement gives memory cycles a higher priority, but ensures that a refresh cycle will be delayed at most by one RAM cycle.

Table 5a. BHE and A0 Encodings

A0	BHE	Function
0	0	Word Transfer
0	1	Byte Transfer on Upper Half of Data Bus (D8-D15)
1	0	Byte Transfer on Lower Half of Data Bus (D0-D7)
1	1	Not Allowed

The advanced acknowledge out of the 82C08 ties into the synchronous ready (SRDY) input of the 80186 via an inverter (a note on synchronous operation .. **NO BUFFERS ON THE STATUS LINES OR THE CLOCK INPUT**). The 80186 clock out signal is directly connected to the 82C08 clock input. The RFRQ' pin is strapped high, which programs the 82C08 to generate internal refresh requests.

5.3 PROGRAMMING

The design shown in Figure 5.1 does not require an external shift register. The PDI pin is strapped to ground. This results in a default of all zeroes, which configures the 82C08 to support a 8MHz 80186 type interface. When programmed in this configuration, the 82C08 interfaces DRAMs having an access time of 150 ns to the iAPX 186 processor operating at 8 MHz, without introducing wait states. This design can be upgraded to support 10 MHz 80186 microprocessor by replacing the 150 ns DRAMs with 120 ns access DRAMs (assuming they meet the requirement of Table 5.b).

5.4 TIMING ANALYSIS

This section will look at a specific system configuration to show how the 82C08 timing requirements are satis-

Table 5b. Programmable Timings
Read and Refresh Cycles

Parameter	C2-Slow Cycle	82C08	Value	DRAM	Value
t _{RP}	2TCLCL - T27	200	ns	100	ns
t _{CPN}	2.5TCLCL - T34	223.4	ns	60	ns
t _{RSH}	2TCLCL - T32	127.6	ns	75	ns
t _{CSH}	3TCLCL - T25	340	ns	150	ns
t _{CAH}	3TCLCL - T32	252.5	ns	25	ns
t _{AR}	3TCLCL - T25	340	ns	100	ns
t _t	3/30	3/30	ns	3/50	ns
t _{RC}	4TCLCL	500	ns	260	ns
t _{RAS}	2TCLCL - T25	215	ns	75	ns
t _{CAS}	2TCLCL + T34(min) - T32	124.5	ns	45	ns
t _{RCH}	TCLCL + T36(min) - T34 + TBUF	97.18 ns - T _{BUF}		0	ns

Write Cycles

Parameter	C2 - Slow Cycle	82C08	Value	DRAM	Value
t _{RP}	2TCLCL - T27	200	ns	100	ns
t _{CPN}	2TCLCL - T34	160.9	ns	60	ns
t _{RSH}	TCLCL - T32	90	ns	75	ns
t _{CAH}	2TCLCL - T32	200	ns	25	ns
t _{AR}	3TCLCL - T25	315	ns	100	ns
t _t	3/30	3/30	ns	3/50	ns
t _{RC}	4TCLCL	500	ns	260	ns
t _{RAS}	2TCLCL - T25	225	ns	75	ns
t _{CAS}	2TCLCL - T32	215	ns	45	ns
t _{WCH}	TCLCL - T32 + T _{BUF}	90 + T _{BUF}	ns	75	ns
t _{WCR}	2TCLCL - T25 + T _{BUF}	225 + T _{BUF}	ns	120	ns
t _{WP}	2TCLCL - T36 - T _{BUF}	127.6 - T _{BUF}		45	ns
t _{RWL}	2TCLCL - T36 - T _{BUF}	127.6 - T _{BUF}		45	ns
t _{CWL}	3TCLCL - T36 - T _{BUF}	252.6 - T _{BUF}		45	ns

fied by the 80186, and how the 82C08 satisfies the timing requirements of the DRAMs.

DYNAMIC RAM INTERFACE

First, look at the timing requirements of the DRAMs to ensure they are satisfied by the 82C08. Memory compatibility timings are shown in the 82C08 data sheet (page 22, Table 10, June 1985) and are summarized in Table 5.b. The DRAM timings for the read, refresh and write cycles are all a function of the 82C08's clock period (TCLCL). The clock frequency of the 82C08 may be varied to meet the timing requirements of slower DRAMs.

tASR (Row A0 to $\overline{\text{RAS}}$ setup time) requirement is related to the address setup time by the equation: $tAVCL = 45 + tASR$. During read or write cycles, the row address propagates directly from the address bus through to the 82C08 address output A00-A08 pins, which are connected to the RAM address pins. So in read and write cycles, the row address setup time is determined by the microprocessor's address valid timing. The row address set up time calculations are shown in the next section.

For the purpose of this example, the timing calculations carried out are based on DRAMs of a type similar to TMS4256-15 (Texas Instruments 256K DRAM, 150ns access time). Table 5.b compares the RAM timings supported by the 82C08 against the RAM timing requirements of the TMS256-15.

ADDRESS SETUP AND HOLD TIME MARGINS

The microprocessor must put the address on the address bus early enough in the memory cycle to pass through the 82C08 and meet the row address setup time to $\overline{\text{RAS}}$ (tASR) timing requirement of the dynamic RAM. Since the address directly propagates through the 82C08, this setup time is a function of how long the microprocessor holds the address on the data bus before activating the status or the command lines, the address delay through the 82C08, and how long the 82C08 waits before activating $\overline{\text{RAS}}$. This is shown in Figure 5.2 and the calculations are shown in Equation 1. This and all equations show time margins, a positive result indicates extra margin, a zero result indicates that the margins are just met which would warrant further analysis, and a negative result indicates that the margin was not met, under worst case conditions (See Note 1).

EQUATION 1:

Address to clk setup time (TAVCL)
 = valid CPU address from clock edge - propagation delay (address latch).

Note 1. The purpose of conducting worst case analysis is to identify design problems early in the development cycle, and to prevent problems that may occur only under worst case component and environmental condi-

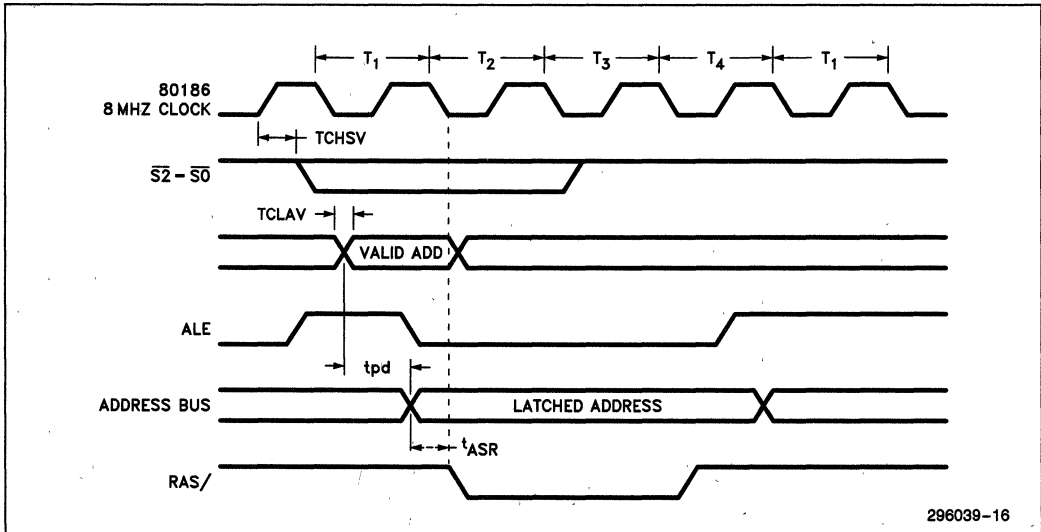


Figure 5.2 Address Setup and Hold Timings

tion. Violating worst case methodology directly affects the yield of a product in a manufacturing environment.

$$= \text{TCLCL} - \text{TCLAV} (80186) \text{ max} - \text{tpd} (74S373) \text{ max.}$$

at 8 MHz

$$\text{TAVCL} = 125 - 55 - 20.8$$

$$\text{tASR} = 49.2 - 45 = 4.2 \text{ ns}$$

Similarly, the microprocessor must maintain the memory address long enough to satisfy the column address address hold time (tCAH). In the slow cycle mode of operation the addresses are latched externally via a 74S373 latch. The addresses are valid until the activation of 'ALE' for the next cycle. The address hold timing requirement is easily met.

Additionally, the A0 and BHE signals, which are decoded to gate the write enable signal to the appropriate upper or lower byte of the memory bank (for 16 bit processors), must be held valid for the duration that $\overline{\text{RAS}}$ is active. This condition is met, as shown in Equation 2.

Equation 2. Address hold time margin referenced to $\overline{\text{RAS}}$.

= Delay from the deactivation of $\overline{\text{RAS}}$ to the activation of ALE for the next processor bus cycle.

$$= \text{TCLCH} (80186) + \text{ALE active delay} (80186) + \text{output enable delay of latch} - \overline{\text{RAS inactive delay}}$$

$$= \text{TCLCH} (80186) \text{ min} + \text{TCHLH} (80186) \text{ min} + \text{t}_{\text{pzl}} (S373) \text{ min} - \text{TCLRSH} (82C08) \text{ max}$$

$$= (\frac{1}{2} \text{TCLCL} - 7.5) + 0 + 5.5 - 50 \text{ ns}$$

$$= 55 + 5.5 - 50$$

$$= 10.5 \text{ ns}$$

Read Data Access Time

The dynamic RAMs have two access time parameters - access time from $\overline{\text{RAS}}$ and access time from $\overline{\text{CAS}}$. Either of these parameters may be the critical parameter in determining the Read Data access time. If the RAM does not meet the access time requirement then either a faster DRAM must be chosen or the clock input to the 80186 and 82C08 must be slowed down.

Access Time From $\overline{\text{RAS}}$

= Delay from the activation of $\overline{\text{RAS}}$ to the time the processor samples data on the data bus.

$$2\text{TCLCL} - \text{TCLRSL} (82C08) \text{ max} - \text{tRAC} (\text{DRAM}) \text{ max} - \text{tpd} (74LS245) \text{ max} - \text{TDVCL} (80186) \text{ max.}$$

$$= 250 - 35 - 150 - 20.8 - 20$$

$$= 24.2 \text{ ns}$$

Access Time From $\overline{\text{CAS}}$

= delay from the activation of $\overline{\text{CAS}}$ to the time the processor samples data from the data bus.

$2\text{TCLCL} - \overline{\text{CAS}}$ active delay - Access time from $\overline{\text{CAS}}$ - transceiver propagation delay - data setup time required by processor.

$$2\text{TCLCL} - \text{TCLCSL} (82C08) \text{ max} - \text{tCAC} (\text{DRAM}) \text{ max} - \text{tpd} (74LS245) \text{ max} - \text{tDVCL} (80186) \text{ max.}$$

$$= 250 - 122.4 - 75 - 20.8 - 20$$

$$= 11.8 \text{ ns}$$

*Example computation of propagation delay taking into consideration the capacitive loading derating and compensation for voltage and temperature variations.

For 74LS245 Transceiver

$\text{tpd} (74LS245) = 18\text{ns}$ maximum (TTL Data Book) speed at 45 pf load. Capacitive loading on the data bus = 5pf (2 DRAMs/data line) = 10pf. As this is well within the capacitive loading requirement of the transceiver, no derating is required.

Derating for the temperature and Vcc range (70 c, 5 V \pm 5%) for the 74LS gates = 2.8 ns.

$$\text{Total prop delay} = 18 + 2.8 = 20.8\text{ns.}$$

For 74S373 Octal Address Latch

$$\text{t}_{\text{phl}} = 13\text{ns max} (\text{TI Data Sheet}) \text{ speed at CL} = 15\text{pf.}$$

Capacitive loading for this case = loading due to 82C08 + loading due to the address decoding logic for the generation of chip-select (assuming the chip-select lines of the 80186 are not being used).

$$= 20\text{pf} (82C08) + 5\text{pf} (\text{TTI } 74S373) = 25 \text{ pf.}$$

$$\text{t}_{\text{phl}} = 13\text{ns} + 0.005 \text{ ns/pf} (10) + 3.1 \text{ pf} (\text{temp and Vcc}) = 13 + 0.05 + 3.1 = 16.15 \text{ pf.}$$

The typical loading on the address lines is 300 pf, for medium size system. For this case the propagation delay is

$$\text{t}_{\text{phl}} = 13 + 0.05\text{ns/pf} (300) + 3.1.$$

$$= 13 + 15 + 3.1 = 31 \text{ ns.}$$

For the TMS4257, the access time from $\overline{\text{CAS}}$ is the limiting factor. This timing wave form is shown in Figure 5.3. In this system, the read data access time requirements are met without the need for wait states.

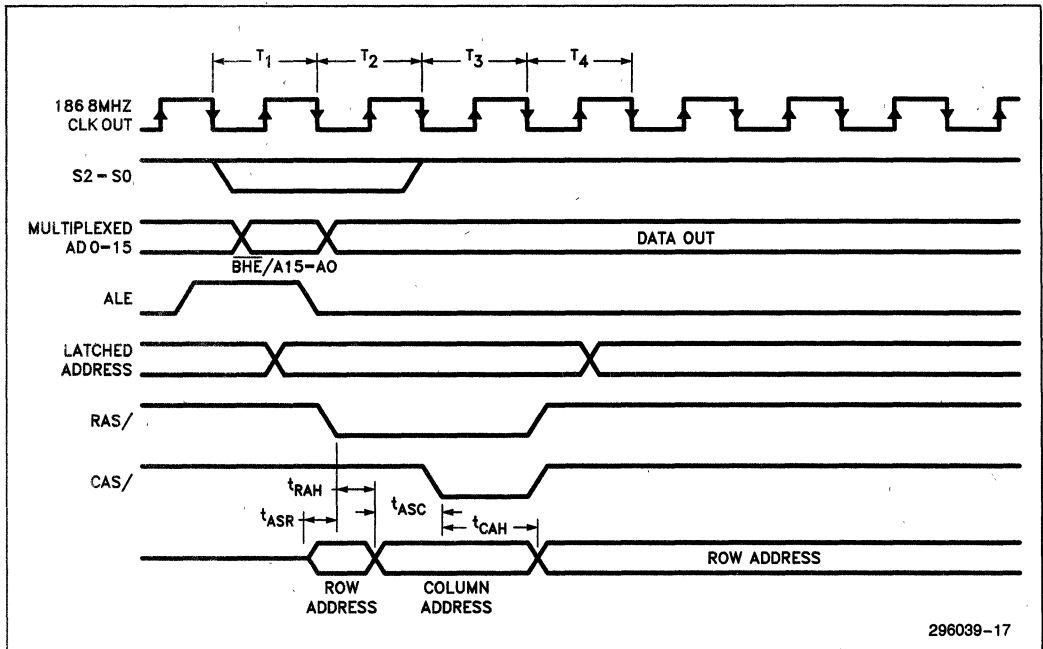


Figure 5.3 80186 READ Cycle Data Setup and Hold Time Waveforms

Write Data Setup and Hold Time Margins

In a write cycle, the write data must:

1. reach the dynamic RAMs before $\overline{\text{CAS}}$, to meet the RAMS data setup time.
2. be valid following $\overline{\text{CAS}}$ to meet the RAMs data hold time.

The write cycle waveforms are shown in Figure 5.4.

Write Data Setup Time

From the waveform

$t_{DS} = \text{TCLCL} + \overline{\text{CAS}}$ active delay - data valid delay - transceiver propagation delay - data setup required by RAMs.

$$\text{TCLCL} + \text{TCLCSL} (82\text{C08}) - \text{TCLDV} - \text{tpd} (74\text{LS245}) - t_{DS}$$

At 8 MHz

$$\begin{aligned} &= 125 + 0 - 44 - 20.8 - 0 \\ &= 60.2\text{ns} \end{aligned}$$

Write Data Hold Time

$t_{DH} = 2\text{TCLCL} + \text{Data invalid delay} + \text{transceiver propagation delay} - \overline{\text{CAS}}$ active delay - data hold time required by RAMs.

$$2\text{TCLCL} + \text{TCLAZ} (80186) + \text{tpd} (74\text{LS245}) - \text{TCLCSL} (82\text{C08}) - t_{DH} (\text{DRAM})$$

$$= 250 + 10 + 6 - 35 - 30$$

$$= 201 \text{ ns}$$

Advanced Acknowledge Setup Time Margin

Advanced Acknowledge ($\overline{\text{AACK}}$) are hand shaking signals to tell the processor when it may terminate the bus cycle currently in progress. Thus, $\overline{\text{AACK}}$ timings determine how many wait states are incurred during a memory cycle.

$\overline{\text{AACK}}$ and $\overline{\text{XACK}}$ serve the same function, but differ only in timing. $\overline{\text{XACK}}$ is a Multibus bus compatible signal and is activated only when data is actually on the bus during a read cycle or when data has been latched

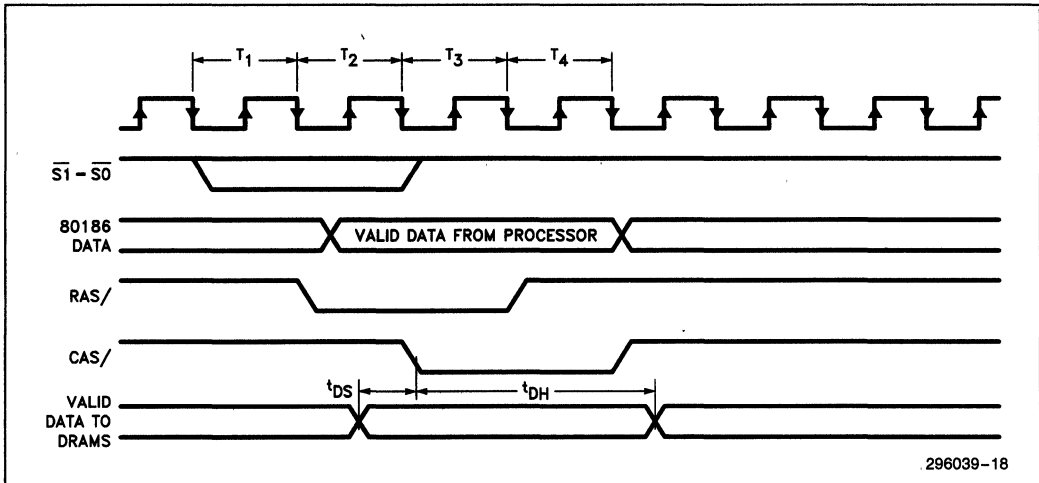


Figure 5.4 WRITE Data Setup and Hold Timings for C2 Configuration

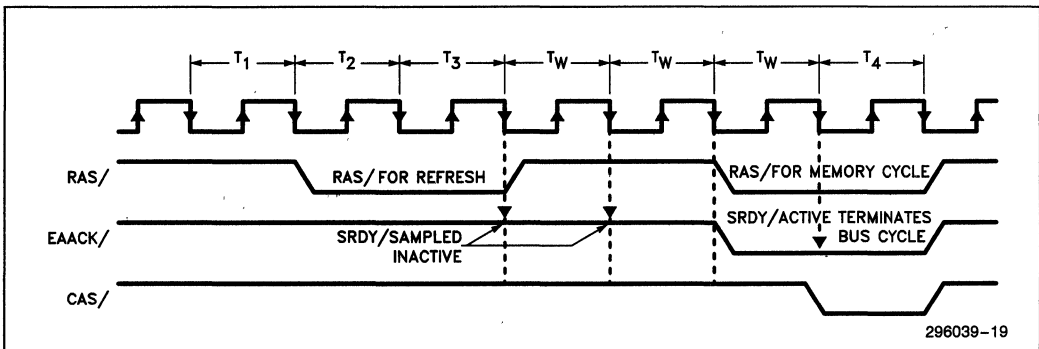


Figure 5.5 READ Cycle Request During REFRESH Cycle (Worst Case 3.W.S.)

by the DRAMs during a write cycle. \overline{AACK} (advanced acknowledge) is activated earlier in the memory cycle. This signal informs the processor that data will be valid, when the processor samples data. This signal is generated earlier in the cycle to compensate for delays in the microprocessor in responding to this signal and terminating its current memory cycle.

In the slow cycle synchronous operation, \overline{AACK} has been optimized to allow zero wait state memory accesses. Both memory read and memory write cycles will be performed without wait states, except when the memory access request coincides with an internal refresh cycle already in progress. In this case, the refresh cycle has to be completed before the requesting processor's memory cycle is started. The \overline{AACK} hand shake signal

keeps the processor in wait states, and releases it when the requested cycle can start. This is shown in Figure 5.5. The \overline{AACK} from the 82C08 connects to the \overline{SRDY} input of the 80186 via an inverter. The 80186 processor samples the \overline{SRDY} line on the falling edge of Clock T3. If \overline{SRDY} is inactive when sampled, the processor introduces a wait state. At the end of this clock cycle, the \overline{SRDY} line is sampled again. The number of wait states incurred, depends upon whether a refresh cycle is in progress when the memory cycle was requested, and if so, how near it is to completion. The worst case timing is when a refresh cycle is started at the same time that the processor requests a memory access. This is shown in Figure 5.5. For this case, the processor will experience three wait states. \overline{AACK} setup time margins is shown in Figure 5.6 and is calculated in Equation 11.

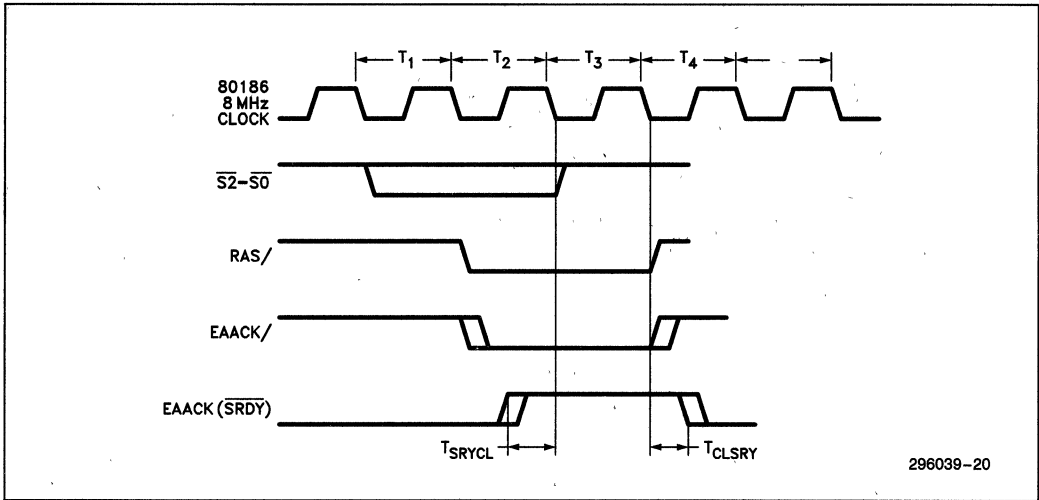


Figure 5.6 AACK Setup and Hold Timing Waveform

AACK Setup Time (Equation 11)

- = Delay from the time the 82C08 activates the $\overline{\text{AACK}}$ to the time the processor samples the ready line.
- = $\text{TCLCL} - \overline{\text{AACK}}$ active delay - propagation delay through inverter - setup time required by 80186
- = $\text{TCLCL} - \text{TCLAKL}$ (82C08) max - tpd (74LS04) - TSRYCL (80186)
- = $125 - 35 - 17.5 - 20$
- = 53.5 ns

AACK Hold Time (Equation 12) (Worst Case)

- = $\overline{\text{AACK}}$ inactive delay + propagation delay through inverter - hold time required by processor
- = TCLAKH (82C08) min + tpd (74LS04) min - TCLSRY (80186) max
- = $0 + 4.5 - 15$
- = -10.5 ns

This timing requirement of the processor is not met under the worst case conditions. One solution would be

to slow down the advanced acknowledge output of the 82C08 by adding additional delay. A crude way of incorporating this delay is to add two additional inverters. This is shown in Figure 5.7.

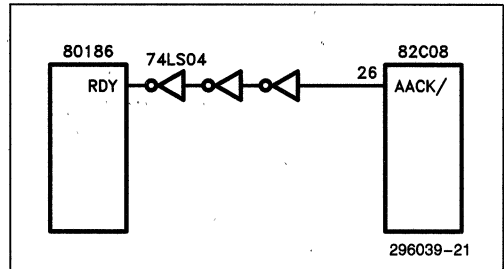


Figure 5.7 Glue TTL to Meet AACK Hold Time

Chip Select Setup Time

The 82C08 qualifies the $\overline{\text{RD}}$, $\overline{\text{WR}}$ inputs with the port enable input, prior to performing a memory cycle. If the port enable setup timing requirement is violated, the memory cycle will be delayed, worse-cause the 82C08 to fail.

If the programmable chip selects of the 80186 microprocessors are used, then the setup time requirements are easily satisfied, as the chip-selects are activated very early in the memory cycle. If chip-selects are generated externally by decoding the address lines, then chip-select generation logic must meet the setup timing requirement of the 82C08.

RAM Data Out Hold Margins

The 82C08 $\overline{\text{CAS}}$ output is held valid for a fixed length of time during a read cycle, after which the RAM data outputs are tristated. This time is not long enough to allow the processor residing on the Multibus Bus to read the data, so data must be latched externally. This latch should be a transparent latch, and should be strobed by $\overline{\text{XACK}}$ from the 82C08. Data hold requirements of the latch:

- = minimum time from $\overline{\text{XACK}}$ active to $\overline{\text{CAS}}$ inactive
- = $\text{TCLCL} + \overline{\text{CAS}}$ inactive delay - $\overline{\text{XACK}}$ active delay - t_{pd} (inverter)
- = $\text{TCLCL} + \text{TCLCSH}$ (82C08) min - TCLTKL (82C08) max - t_{pd} (74LS04)
- = $125 + 0 - 35 - 18.5$
- = 71.5 ns

So any latch with data hold requirements of less than 70 ns can be used for latching data out of DRAMs.

OTHER DESIGN CONSIDERATIONS

1. Pullup resistors on the status lines:

The status lines of the 80186 microprocessor connect directly to the $\overline{\text{RD}}$, $\overline{\text{WR}}$ and PCTL inputs of the 82C08, when interfacing in the status synchronous mode. The status lines of the 80186 float for approximately six clock cycles following system reset. The 82C08 may erroneously decode the status lines, and try to initiate a memory cycle or worse, lock up the processor. In order to prevent a spurious memory cycle, the status lines of the 80186 must be pulled up. The value of the pull-up resistor can be calculated as follows.

Referring to Figure 5.8.

$$I_{OL \text{ max}} = (5 - 0.4)/R = 4.6/R$$

This can be rewritten as

$$I_{OL} (80186) \geq I_{IL} (82C08 \text{ Status lines}) + 4.6/R$$

$$2.5 \text{ ma} \geq \sim 0 + 4.6/R; \text{ For CMOS parts the } I_{il} \sim 0.$$

$$R \text{ min} = 4.6 / 2.5 \text{ ma} = 1870 \text{ Ohms}$$

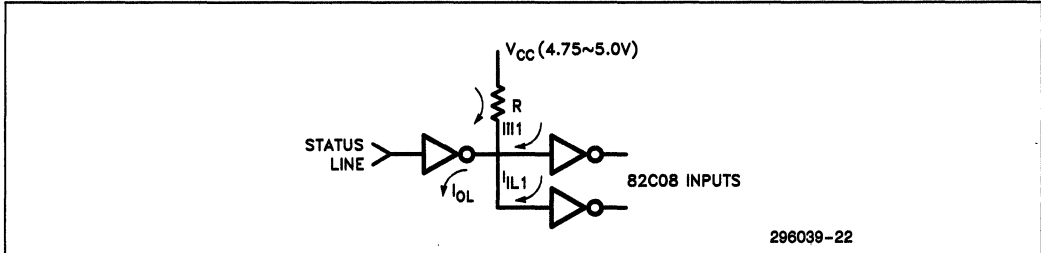


Figure 5.8 Equivalent Circuit for Calculating Pullup Resistor

296039-22

2. Damping Resistors on the 82C08's output lines:

The two common problem areas in a DRAM memory design are:

- Undershoot and overshoot caused by inductive traces and high capacitive loads.
- Switching spikes caused by switching capacitive loads.

The cause and effect of the problems are outlined in Chapter 9. To reduce the current spikes, it is necessary to provide series damping resistors on all the 82C08 output lines (RAS, CAS, Address outputs and WE lines). The values of the series resistor are application dependent, and should be selected to cause critical damping. The nominal value for the series resistors are 39 Ohms on the RAS and CAS outputs, and 22 Ohms on the Address outputs. Generally this value is application dependent and must be chosen empirically.

INTERLEAVING

When the 82C08 is programmed to operate in the status synchronous mode, consecutive memory read or write accesses to the same bank, do not incur wait states. There is no performance enhancement by using multiple banks.

Figure 5.9 Wait States for 80186 for Different RAM Speeds

Dram Speed		100 ns	120 ns	150 ns
80186	6 MHz	0 w.s.	0 w.s.	0 w.s.
	8 MHz	0 w.s.	0 w.s.	0 w.s.
	10 MHz	0 w.s.	0 w.s.	—

5.6 CONCLUSION

This design will operate without wait states, at worst case conditions, at microprocessor frequencies of upto 8 MHz. This design will operate at 10 MHz, provided 120 ns DRAMs are used, without the need for any additional gates. This can be verified by substituting 'TCLCL' with 100 ns and recomputing all of the above parameters.

CHAPTER 6

INTERFACING THE 82C08 CHMOS DYNAMIC RAM CONTROLLER TO THE IAPX 80286 MICROPROCESSOR

6.1 ABSTRACT

This chapter illustrates a Dynamic RAM design, for a 80286 based system using the 82C08 CHMOS Dynamic RAM controller. Two design examples are considered, one using two banks of 120 ns 256K DRAMs while the second example uses two banks of 150ns 256K DRAMs. The design goals for these examples are:

1. Control 1 megabyte of memory, configured as two banks of 256K \times 16 bits.
2. Support 150ns and 120ns DRAMs by just changing the program data word.
3. Use minimal additional logic.
4. Achieve zero waitstate memory access.

6.2 DESIGNING THE HARDWARE

Figure 6.1 shows a memory design for a 80286 based system, using the 82C08 CHMOS DRAM controller. This design can support two speeds of DRAMs, 120ns and 150ns, by changing the program data word. To incorporate this feature, an external shift register is used. The external shift register is not required when supporting only 120 ns DRAMs. As few as two additional ICs complete the memory control function.

In this example, the 82C08 interfaces synchronously to the processor for best performance. The 82284 clock generator provides the clocking for both the 82C08 and the 80286. The status lines of the 80286 are connected to the RD and WR inputs of the 82C08. The 82284 clock generator has internal pull-up resistors, which eliminates the need for external pull ups on the status lines. The address lines of the 80286 connect directly to the ALO-AH8 address input pins of the 82C08 (The 80286 address lines can drive a capacitive load of 100 pf. This capacitive load requirement should not be exceeded). In the fast cycle (80286) mode, the 82C08 generates an internal strobe to latch the addresses. The

latched A0 and BHE signals are decoded to derive the gating signals, for upper and lower byte of the memory array.

The least significant address line, A1, is tied directly to the bank select input. This allows memory interleaving as consecutive addresses are in alternate banks. This allows overlapping the RAS precharge of the first access behind the memory access of the next cycle.

The 82C08 performs early write memory cycles, which allows data in and data out pins to be tied together at the RAM. This reduces the number of board traces. Data buffers are required to prevent bus contention. During the read cycle, the 82C08 CAS line drivers drive data onto the bus upto 50 ns past the end of the bus cycle. If another bus cycle were to start immediately, there could be bus contention between processor and DRAM data.

6.3 TIMING ANALYSIS

This section will look at two specific examples to show how the 82C08's timing requirements are satisfied by the 80286 microprocessor and how the 82C08 meets the timing requirement of the DRAMs. Methods of determining worst case values are also given.

The first example requires no wait states, as it interfaces to fast DRAMs (120 ns). The second example, on the other hand, requires one wait state, as it interfaces to slow DRAMs (150 ns). Both the examples assume an 8 MHz 80286. This design can be upgraded to support a 10 MHz 80286 processor by substituting the 120 ns access time DRAMs with 100 ns access time DRAMs.

6.3.1 Example 1.

This example assumes an 80286 microprocessor, operating at 8 MHz, interfacing to two banks of 120 ns TMS 4257-12 (Texas Instruments) DRAMs.

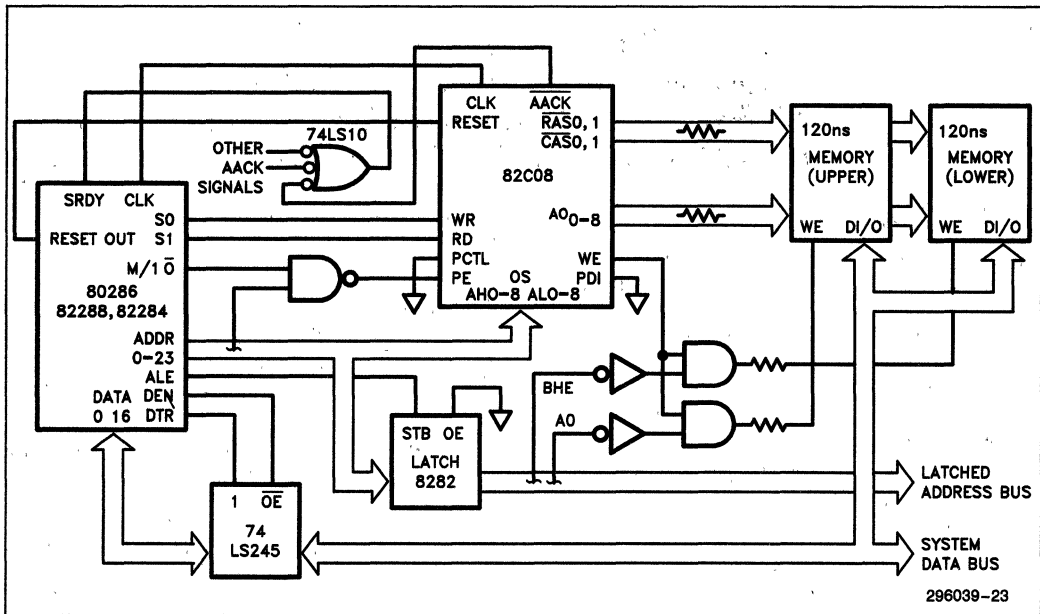


Figure 6.1 82C08 Interface to an 80286

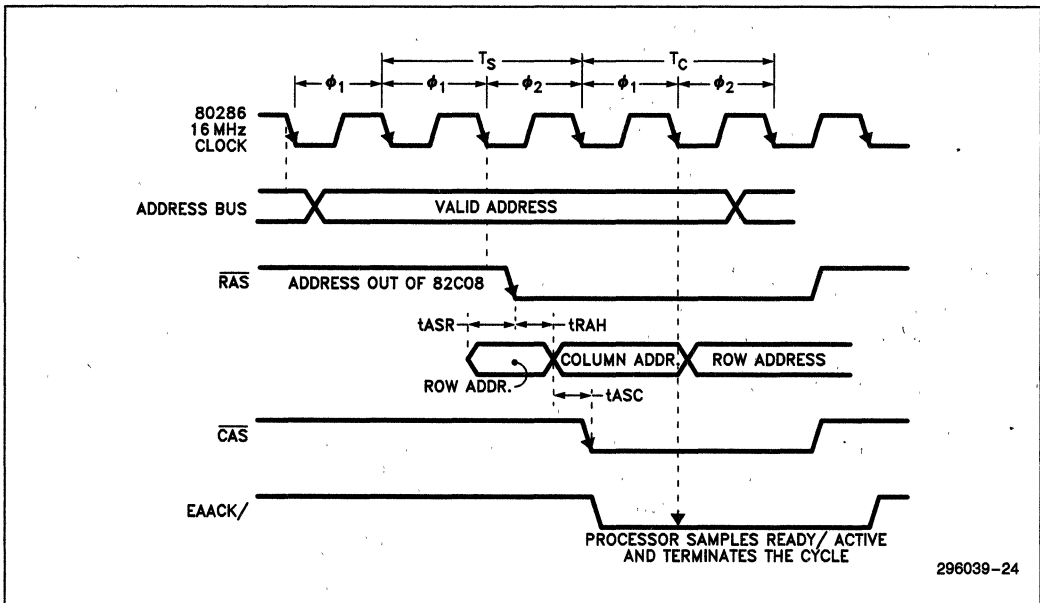
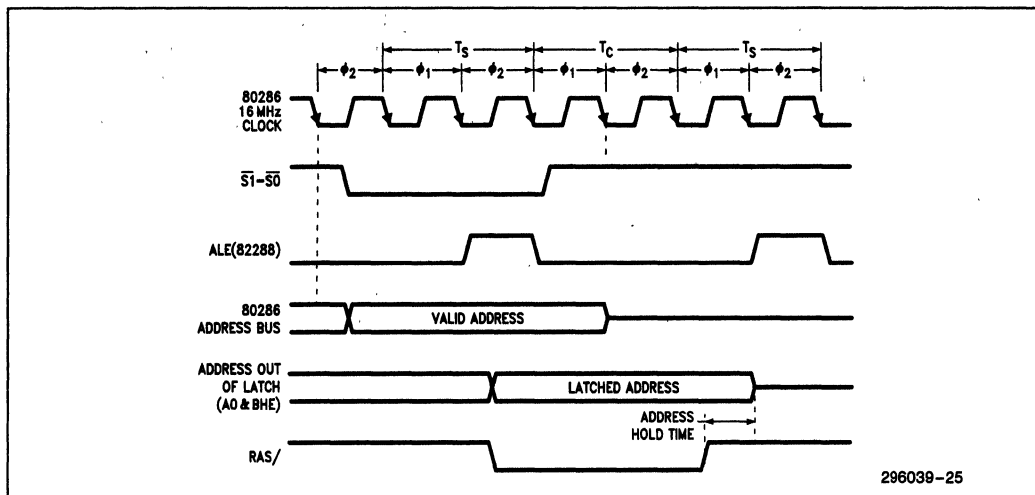


Figure 6.2 Address Setup and Hold Margins


Figure 6.3 Address Hold Time
Table 6a. Programmable Timings
Read and Refresh Cycles

Parameter	C0-Fast Cycle	82C08	Value	DRAM	Value
t_{RP}	3TCLCL - T27	137.5	ns	100	ns
t_{CPN}	3TCLCL - T34	137.5	ns	40	ns
t_{RSH}	2TCLCL - T32	90	ns	60	ns
t_{CSH}	3TCLCL - T25	162.5	ns	120	ns
t_{CAH}	2TCLCL - T32	90	ns	20	ns
t_{AR}	3TCLCL - T25	162.5	ns	70	ns
t_T	3/30	3/30	ns	3/50	ns
t_{RC}	6TCLCL	375	ns	200	ns
t_{RAS}	3TCLCL - T25	162.5	ns	150	ns
t_{CAS}	3TCLCL - T32	152.5	ns	70	ns
t_{RCH}	TCLCL - T34 + T _{BUF}	12.5 + T _{BUF}		0	ns

Write Cycles

Parameter	C0-Fast Cycle	82C08	Value	DRAM	Value
t_{RP}	3TCLCL - T27	137.5	ns	100	ns
t_{CPN}	4TCLCL - T34	200	ns	40	ns
t_{RSH}	2TCLCL - T32	100	ns	60	ns
t_{CSH}	4TCLCL - T25	225	ns	120	ns
t_{CAH}	TCLCL - T32	27.5	ns	20	ns
t_{AR}	3TCLCL - T25	162.5	ns	70	ns
t_T	3/30	3/30	ns	3/50	ns
t_{RC}	7TCLCL	437.5	ns	200	ns
t_{RAS}	4TCLCL - T25	225	ns	150	ns
t_{CAS}	2TCLCL - T32	90	ns	70	ns
t_{WCS}	TCLCL - T36 - T _{BUF}	27.5 - T _{BUF}		0	ns
t_{WCH}	TCLCL - T36 - T31 - T _{BUF}	90 + T _{BUF}		35	ns
t_{WCR}	4TCLCL - T25	225 + T _{BUF}	ns	85	ns
t_{WP}	3TCLCL - T36 - T _{BUF}	152.5 - T _{BUF}		40	ns
t_{RWL}	3TCLCL - T36 - T _{BUF}	152.5 - T _{BUF}		40	ns
t_{CWL}	3TCLCL - T36 - T _{BUF}	152.5 - T _{BUF}		40	ns

DYNAMIC RAM INTERFACE

The first requirement is to ensure that the 82C08 satisfies the timing requirements of the DRAMs. Memory compatibility timings are shown in the 82C08 data sheet (page 22, Table 10, June 1985) and are summarized in Table 6.a. The DRAM timings for the read, write and refresh cycles are a function of the 82C08 clock input which may be slowed down to accommodate slower DRAMs.

During the read or write memory cycles the row address propagates directly from the address bus through to the 82C08 address output AO0 - AO8 pins, which are connected to the RAM address input pins. The row address setup time is determined by the microprocessor's address valid delay timing. The row address setup time calculations are shown in the next section.

Table 6.a shows the 82C08 RAM timings calculated for a clock input of 16MHz. The table compares the RAM timings supported by the 82C08 against the RAM timing requirements of the TMS4256-12.

ADDRESS SETUP AND HOLD MARGIN

The microprocessor must put the address on the address bus early enough in the memory cycle for it to pass through the 82C08 and meet the row address setup time to $\overline{\text{RAS}}$ (tASR) requirement of the DRAM. The internal address multiplexer of the 82C08 switches during the current memory cycle (after the column address hold timing requirements are met) to allow the row addresses for the next cycle to propagate through to the 82C08 address output pins. The clock edge that causes the internal address multiplexer to switch from column address to row address is dependent upon the programmed configuration of the 82C08 (in the C0 configuration, the internal multiplexer switches during the clock 2 cycle to select the row address for the next cycle). The actual clock edges on which the transition takes place are shown in Table 9a. and Table 9b. in the 82C08 data sheet.

Since the address propagates directly through the 82C08, the address setup time is a function of how long the microprocessor holds the address on the bus prior to activating the status lines (synchronous status interface) or the read and write command lines (command interface), the propagation delay through the 82C08, and the delay in activating the $\overline{\text{RAS}}$. This and all the equations to follow, show time margins; a positive result indicates extra margin, a zero result indicates that the parameter is just met, and a negative result indicates that the timing requirement is not met, under worst case condition.

Before calculating tASR (address setup to RAS) timing margin, it is necessary to calculate the address set up to clk (tAVCL) timing, as they are related by the equation.

$$tAVCL = TCLCL + tASR; \text{ Spec 23, 82C08 Data Sheet.}$$

$$tAVCL \text{ (from Figure 6.1.)} = \text{Address to Clk setup time} \\ = 2TCLCL - \text{Address valid delay}$$

$$tAVCL \text{ (for an 8MHz 80286)} \\ = \text{Address valid delay (80286) to the falling edge of clock.} \\ = 2(TCLCL) - \text{Address valid delay (80286)} \\ = 2 \times 62.5 - 60 = 125 - 60 = 65 \text{ ns.}$$

$$tASR = tAVCL - TCLCL \\ = 65 - 62.5 \\ = 2 \text{ ns.}$$

NOTE:

1. For the case when the 80286 interfaces in the status synchronous mode to the 82C08, the address from the 80286 can be directly connected to the address inputs of the 82C08. The 82C08 in this mode of operation, internally generates a signal to latch the addresses.
2. The propagation delay of the 'F373' is specified in the data sheet to be 9 ns at 50 pf. The capacitive loading on the address bus is typically on the order of 300 pf. The propagation delay specified in the data sheet has to be derated for this capacitive load.

$$\text{Row address setup time margin for TMS4256-12} \\ = 2.5 \text{ ns} - (tASR \text{ of TMS4256-12}) \\ = 2.5 \text{ ns} - 0 = 2.5 \text{ ns}$$

There is no restriction on how long the processor's address should be held valid, as they are internally latched by the 82C08. More importantly, the latched A0 and BHE signals, which are decoded to derive the gating signals for the write enable output (if byte write, then to determine if write enable should be activated to upper or lower byte of the memory word), must be held valid until $\overline{\text{RAS}}$ goes inactive. This timing requirement is met, as shown below.

Address hold time margin

$$= \text{Delay from the deactivation of } \overline{\text{RAS}} \text{ to the activation of ALE for the next bus cycle.} \\ = TCLCL + \text{ALE active delay (80288) min} + \text{propagation delay of the latch} - \text{RAS inactive delay} - \text{Address hold time of 82C08.}$$

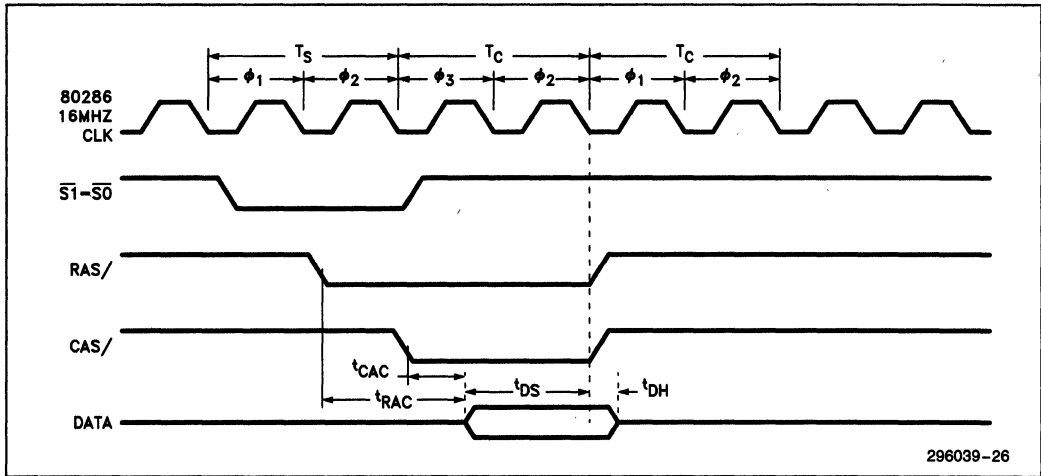


Figure 6.4 Read Data Setup and Hold Time Waveform (C0 Configuration - 120 ns DRAMs)

$$= 62.5 + 3 \text{ ns} + t_{poe} (74F373) \text{ min} - T_{CLRSH} (82C08) \text{ max} - 0 \text{ ns}$$

$$= 62.5 + 3 \text{ ns} + t_{poe} (74F373) \text{ min} - 50 - 0 = 62.5 + 3 \text{ ns} + 4.5 \text{ ns} - 50 \text{ ns} - 0 \text{ ns} = 19.5 \text{ ns.}$$

Read Data Access Time Margin

Read data access times determines how many wait states are required for a read cycle. There are two access times that have to be evaluated to determine the limiting factor for the DRAM access time.

1. access time from \overline{RAS}
2. access time from \overline{CAS}

The wave form for a read cycle is shown in Figure 6.4. The calculations carried out below assume 120ns access time DRAMs.

1. Access time from \overline{RAS} .
 - = the delay from the activation of \overline{RAS} to the time valid data is sampled by the CPU.
 - = $3T_{CLCL} - \overline{RAS}$ active delay - Access time from \overline{RAS} - transceiver propagation delay - CPU data setup requirement.
 - = $3T_{CLCL} - T_{CLSL} (82C08) \text{ max} - t_{RAC} (DRAM) \text{ max} - t_{pd} (74F245) \text{ max} - t_{DS} (80286) \text{ max.}$

at 16MHz (8MHz 80286)

$$= 3 \times 62.5 - 25 - 120 - 10 - 8 (@ 50 \text{ pF})$$

$$= 24.5 \text{ ns}$$

The F245 transceiver is rated at 50pF. The capacitance loading on the data bus for a medium size application would well exceed this value. Timing charts for the capacitive loading vs propagation delay should be used, to determine the actual worst case propagation delay for the transceiver. A general rule of thumb for determining the derating factor is 0.02 ns/pF for excess capacitive load.

So, if the capacitive loading is 300 pF, then the actual propagation delay:

$$t_{pd} (74F245) \text{ actual} = t_{pd} (74F245) \text{ at } 50 \text{ pF} + (300 - 50) 0.02 \text{ ns/pF.}$$

$$= 8 + 5 = 13 \text{ ns.}$$

2. Access Time From CAS

- = the delay from activation of \overline{CAS} to the time when processor samples valid data.
- = $2T_{CLCL} - \overline{CAS}$ active delay - Access time from \overline{CAS} - propagation delay through the transceiver - Data setup required by the CPU.
- = $2T_{CLCL} - T_{CLCSL} (82C08) \text{ max} - t_{CAC} (DRAM) \text{ max} - t_{pd} (74F245) \text{ max} - t_{DS} (80286) \text{ max}$
- = $2 \times 62.5 - 35 - 60 - 8 \text{ (at } 50 \text{ pF)} - 10$
- = $125 - 113 = 12 \text{ ns.}$

For the case being considered, the \overline{CAS} access time is the limiting factor. Summarizing for this system, the read data access time requirement of the DRAM is satisfied without the need for wait states.

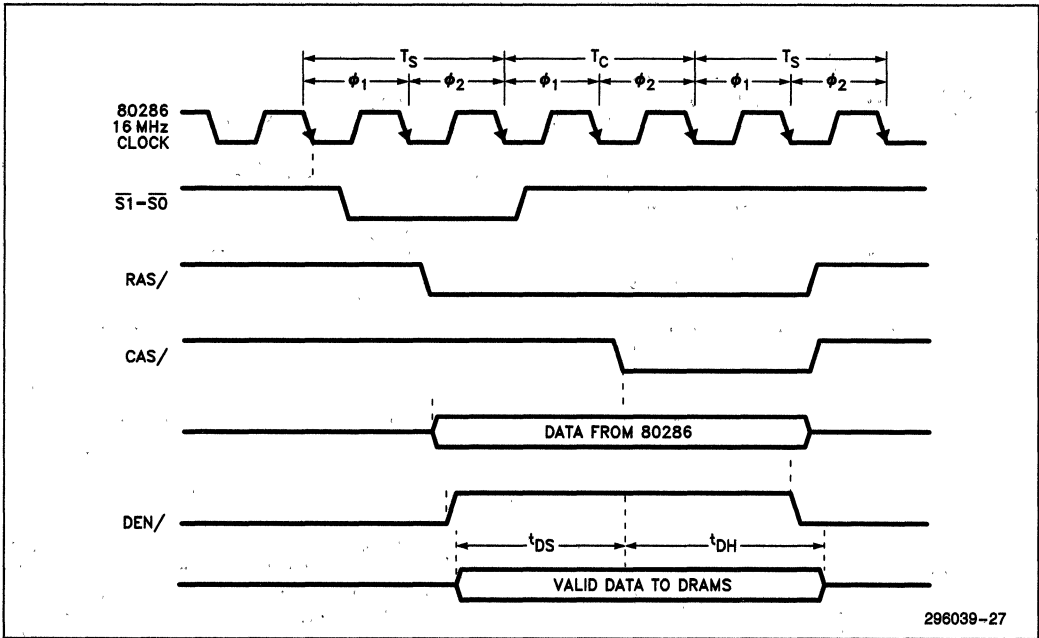


Figure 6.5 Data Set Up and Hold Timing Waveforms for Write Cycle C0 Configuration

Write Data Setup And Hold Time Margins

In the write cycles, the write data must:

1. reach the DRAMs before \overline{CAS} is activated to meet the RAM data setup time.
2. the data must be held after the activation of \overline{CAS} , to meet the data hold timing requirement of DRAM.

Write Data Setup Time

The write data set up time is a function of how long the processor holds valid write data on the bus, prior to the activation of \overline{CAS} . The wave form for a write cycle is shown in Figure 6.5 and calculated below.

Write data setup time

$$= 2TCLCL + \overline{CAS} \text{ active delay} - \text{valid write data delay} - \text{propagation delay through the transceiver} - \text{data setup requirement of DRAM.}$$

$$= 2TCLCL + TCLCSL \text{ (82C08) min} - \text{valid write data delay (80286) at 16MHz} - t_{pd} \text{ (F245) max} - t_{DS} \text{ (DRAM) max.}$$

$$= 2 \times 62.5 + 0 - 50 - 8 \text{ ns} - 0 \text{ ns}$$

$$= 125 - 50 - 8 = 67 \text{ ns}$$

Write Data Hold Time

Data on the bus becomes invalid following the inactive edge of DEN , as it tri-states the output buffers of the transceiver.

Write data hold time

$$= \text{Delay from the activation of } \overline{CAS} \text{ to } DEN \text{ inactive delay}$$

$$= 2TCLCL + DEN \text{ inactive delay} + \text{delay through the inverter} + \text{output disable time from } DEN/ \text{ going inactive} - \overline{CAS} \text{ active delay.}$$

$$= 2TCLCL + DEN \text{ inactive delay (80286) min} + t_{pd} \text{ (74LS04) min} + t_{poe} \text{ (74F245) min} - t_{CLRSL} \text{ (82C08) max}$$

$$= 125 + 3 + 4.5 \text{ ns} + 3 \text{ ns} - 35$$

$$= 100.5 \text{ ns}$$

AACK Setup Time Margin

Advanced acknowledge ($\overline{\text{AACK}}$) and ($\overline{\text{XACK}}$) are "hand shaking" signals used to inform the processor to terminate the present bus cycle. The acknowledge timings directly determine the number of wait states incurred by the processor. The timings of the 82C08's $\overline{\text{AACK}}$ has been optimized to allow memory accesses without incurring wait states.

$\overline{\text{AACK}}$ and $\overline{\text{XACK}}$ serve the same function but differ in timing. $\overline{\text{XACK}}$ is a Multibus Bus compatible signal, and is activated when data is valid on the data bus. $\overline{\text{AACK}}$ is activated earlier in the memory cycle and informs the processor that data will be valid when it is ready to sample data. The $\overline{\text{AACK}}$ is generated earlier in the memory cycle to compensate for the inherent delay of the microprocessor in responding to this signal to terminate the current bus cycle. The timings of $\overline{\text{AACK}}$ have been optimized for systems that interface to the DRAM controller in the status synchronous mode. Using $\overline{\text{AACK}}$ is recommended for this type of interface as no wait states are introduced during normal memory cycles.

The number of wait states incurred depend upon:

1. if a refresh cycle is in progress when the memory cycle is requested.
2. if memory accesses are made to the same bank.

1. Memory Request Occurs When a Refresh Cycle is in Progress

If a refresh cycle is in progress when the processor requests a memory cycle, then the number of wait states incurred would depend on:

1. how near the refresh cycle is to completion.

2. The $\overline{\text{RAS}}$ precharge time (as the 82C08 activates both the $\overline{\text{RAS}}$ lines, the memory cycle can only be started following the $\overline{\text{RAS}}$ precharge time).

Figure 6.6 shows a case when memory request is received just as the 82C08 starts a refresh cycle. In this case, the memory request is internally latched by the 82C08 if the command meets the setup and hold time required by the 82C08. The advanced acknowledge output of the 82C08 is held inactive, until the requested memory cycle can start. This ensures that the correct number of wait states are introduced. The advanced acknowledge output is activated only after the completion of the refresh cycle and the $\overline{\text{RAS}}$ precharge time. This is shown in Figure 6.5. It can be seen from the figure that the activation of $\overline{\text{RAS}}$ is delayed by one clock cycle following the $\overline{\text{RAS}}$ precharge time. This is because the 82C08 allows activation of $\overline{\text{RAS}}$ only in the middle of the 80286 clock cycle. This ensures that data is valid, when the 80286 samples data. (This is stated in the 82C08 data sheet, page 19). The worst case delay for the read or the write cycle during a refresh cycle is 3 wait states.

2. Memory Accesses To The Same Bank

The example considered here assumes two banks of memory. With the Bank Select (BS) input tied directly to the least significant address line (A1 address line from the 80286), sequential memory accesses will be to alternate banks. But if the accesses were to the same bank, then only the second immediate access will be delayed by the $\overline{\text{RAS}}$ precharge time. Figure 6.7. shows the timing wave form for a write cycle followed by a read cycle to the same bank. It is seen from the wave forms that when sequential memory accesses are performed to the same bank, the memory cycles that follow will experience two wait states.

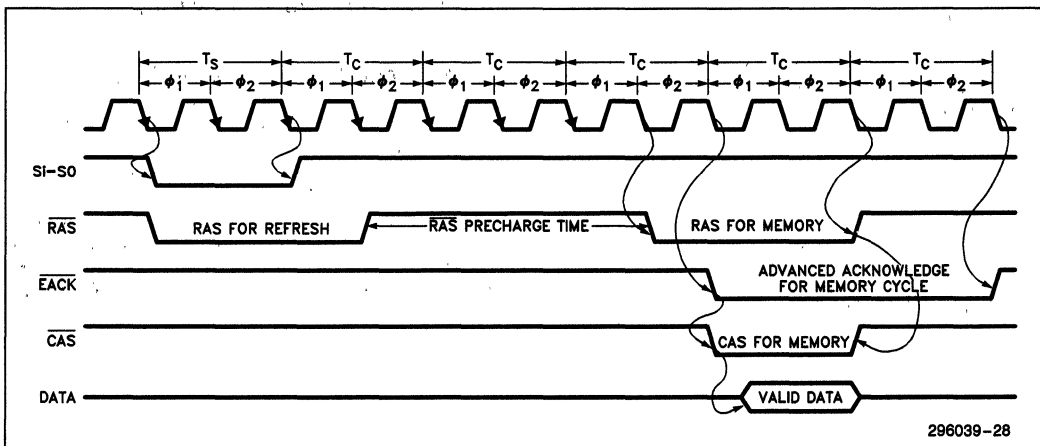


Figure 6.6 Memory Request During a Refresh Cycle

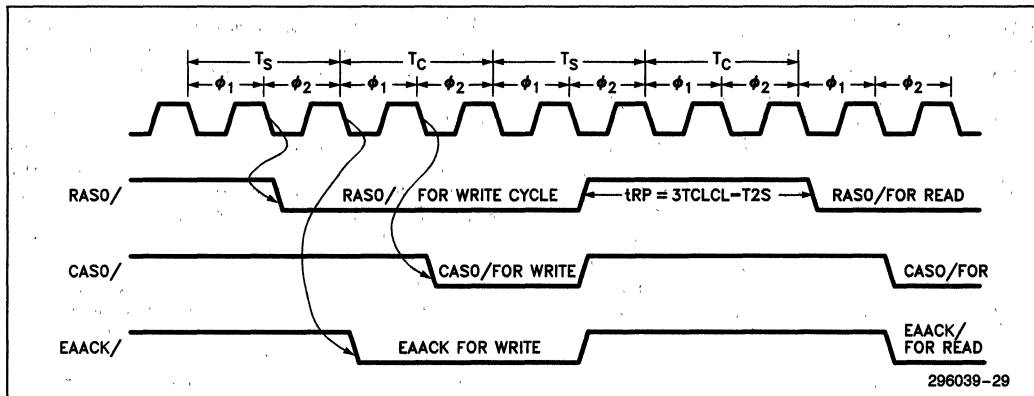


Figure 6.7 Write Cycle Followed by Read Cycle (C0 Configuration)

The advanced acknowledge ($\overline{\text{AACK}}$) is connected to the $\overline{\text{SRDY}}$ (synchronous ready) input of the 82284 clock generator. The advanced acknowledge timings must satisfy the setup and hold requirements of the 82284.

The Advanced Acknowledge setup time is shown in Figure 6.7 and calculated below. The calculations show that the $\overline{\text{AACK}}$ meets the setup requirement of the 82284 clock generator.

SRDY setup time

$$\begin{aligned}
 &= \text{TCLCL} - \overline{\text{AACK}} \text{ active delay} - \text{setup time} \\
 &\quad \text{required by 82284} \\
 &= 62.5 - 35 - 15 \\
 &= 12.5 \text{ ns.}
 \end{aligned}$$

PCS Setup Time Margin

The 82C08's $\overline{\text{RD}}$, $\overline{\text{WR}}$ inputs are qualified by the PCS/, in order to grant the requested memory cycle. If the PCS/ active setup time is violated, then the memory cycle will be delayed - worse cause the 82C08 to not function correctly. The chip select generation logic is shown in Figure 6.8.

PCS Setup time margin

$$\begin{aligned}
 &= \text{CPU address valid delay} - \text{PCS decode time} - \\
 &\quad \text{setup time required by 82C08} \\
 &= 2\text{TCLCL} - \text{address valid delay} - \text{tpd (74LS04)} \\
 &\quad - \text{tpd (74LS00)} \\
 &= 125 - 60 - 18.5 - 18.5 - 20 = 8 \text{ ns}
 \end{aligned}$$

RAM Data Out Hold Time Margins

The 82C08 CAS output is held valid for a fixed length of time, during a read or write cycle. Following the deactivation of $\overline{\text{CAS}}$, the RAM data outputs are tri-stated. The data is not valid long enough to allow the processor on the Multibus Bus to read the data from the bus, so data must be latched externally. This latch must be a transparent latch and should be strobed by $\overline{\text{XACK}}$ from the 82C08.

RAM data out hold time margin from $\overline{\text{XACK}}$,

$$\begin{aligned}
 &= \text{TCLCL} - \overline{\text{XACK}} \text{ active delay} - \text{setup time} \\
 &\quad \text{required by latch} \\
 &= \text{TCLCL} - \text{TCLTKL (82C08) max} - \text{tH} \\
 &\quad \text{(74LS373) min.} \\
 &= 62.5 - 35 - 10 \\
 &= 17.5 \text{ ns}
 \end{aligned}$$

Other Design Considerations:

Series Resistors on the 82C08 outputs

Series resistors are required on all of the 82C08s outputs to reduce undershoot and overshoot caused due to the complex impedance presented by the high capacitive loads and trace inductance. The resistor value chosen should cause critical damping. The resistor value is system dependent, and varies from 33 Ohms to 60 Ohms, depending upon the number of DRAMs and trace lengths.

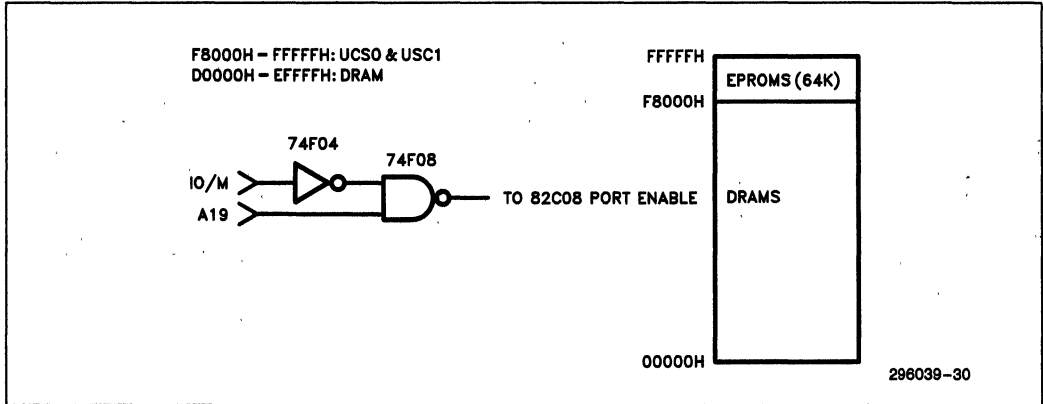


Figure 6.8 Chip Select Generation Logic

CONCLUSION

This design will operate with zero wait states at worst case (except when memory requests coincide with a refresh cycle), at microprocessor frequencies up to 8MHz, using two banks of 120ns DRAMs. This design can be upgraded to support 10 MHz 80286 by using 100 ns DRAMs. When single bank of memory is used, then consecutive memory accesses will incur two wait states.

6.3.2 Example 2

Alternate Configuration

Figure 6.9 shows another 80286 based memory system design, but this time using 150ns access time DRAMs. When 150ns DRAMs are used, memory requests require 1 wait state (Advanced Acknowledge is activated one clock cycle later). Additional wait states are incurred, when a memory access coincides with a refresh cycle already in progress, or if consecutive memory accesses are to the same bank.

The 82C08 has to be programmed via an external shift register into C1 configuration, to support the timing requirements of the 150 ns DRAMs. Improper operation results when the default programming is used (default mode configures the 82C08 in the C0 configuration), as the DRAMs cannot put out valid data fast enough to meet the processor's data setup and hold timing requirements. When the 82C08 is programmed in the C1 configuration, the length of \overline{RAS} , CAS, WE and \overline{EACK} timings are increased by one clock cycle. Delaying the activation of \overline{EACK} introduces the additional wait state required, to allow correct system operation.

The only additional logic required to support 150ns access DRAMs is an external shift register. The program data bit (PD2) now needs to be programmed to a zero, to configure the 82C08 in the slower C1 configuration.

Dynamic RAM Interface

The first step is to ensure that the DRAM timing requirements are satisfied by the 82C08. Table 6.b compares the DRAM timing requirements to those supported by the 82C08.

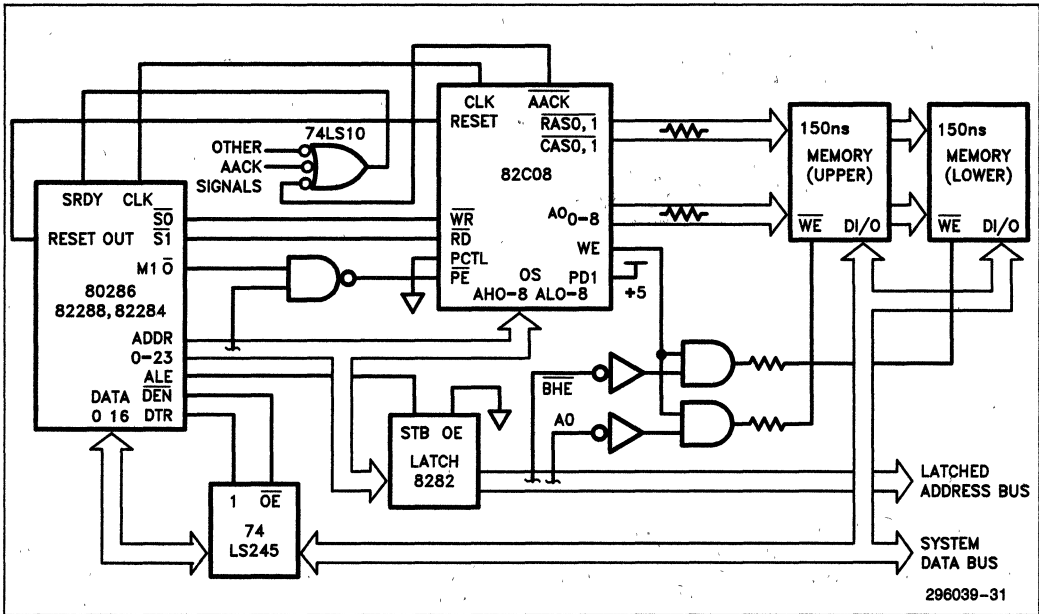


Figure 6.9 82C08 Interface to an 80286

Table 6b. Programmable Timings
Read and Refresh Cycles

Parameter	C1 - Slow Cycle	82C08	Value	DRAM	Value
t _{RP}	3TCLCL - T27	137.5	ns	100	ns
t _{CPN}	3TCLCL - T34	137.5	ns	60	ns
t _{RSH}	3TCLCL - T32	152.5	ns	75	ns
t _{CSH}	5TCLCL - T25	287.5	ns	150	ns
t _{CAH}	3TCLCL - T32	152.5	ns	25	ns
t _{AR}	4TCLCL - T25	225	ns	100	ns
t _T	3/30	3/30	ns	3/50	ns
t _{RC}	7TCLCL	437.5	ns	260	ns
t _{RAS}	4TCLCL - T25	225	ns	75	ns
t _{CAS}	4TCLCL - T32	215	ns	45	ns
t _{RCS}	2TCLCL - T35 + T _{BUF}	90 - T _{BUF}	ns	0	ns
t _{RCH}	2TCLCL - T34 + T _{BUF}	75 - T _{BUF}	ns	0	ns

Write Cycles

Parameter	C1 - Fast Cycle	82C08	Value	DRAM	Value
t _{RP}	3TCLCL - T27	137.5	ns	100	ns
t _{CPN}	4TCLCL - T34	200	ns	60	ns
t _{RSH}	3TCLCL - T32	162.5	ns	75	ns
t _{CSH}	5TCLCL - T25	287.5	ns	150	ns
t _{CAH}	2TCLCL - T32	90	ns	25	ns
t _{AR}	4TCLCL - T25	225	ns	100	ns
t _T	3/30	3/30	ns	3/50	ns
t _{RC}	8TCLCL	500	ns	260	ns
t _{RAS}	5TCLCL - T25	287.5	ns	75	ns
t _{CAS}	3TCLCL - T32	152.5	ns	45	ns
t _{WCH}	3TCLCL - T32 + T _{BUF}	152.5 + T _{BUF}	ns	75	ns
t _{WP}	4TCLCL - T36 - T _{BUF}	215 - T _{BUF}	ns	45	ns
t _{RWL}	4TCLCL - T36 - T _{BUF}	215 - T _{BUF}	ns	45	ns
t _{CWL}	4TCLCL - T36 - T _{BUF}	215 - T _{BUF}	ns	45	ns
t _{WCS}	TCLCL - T36 - T _{BUF}	27.5	ns	0	ns

As the processor interface remains unchanged, the address setup and hold time margins, E_AACK setup time margin and PCS setup time margin will remain the same as the previous example.

Read Data Access Time Margin

There are two access time parameters that have to be considered - RAS access time and CAS access time, to determine which is the limiting factor. The read cycle timing waveform is shown in Figure 6.10, and the calculations are carried out below. As AACK is activated later in the memory cycle, one wait state is introduced.

Access Time From RAS

5TCLCL - RAS active delay - Access time from RAS - propagation delay through the transceiver - data setup time required by the processor.

$$= 5TCLCL - TCLRSL (82C08) \max - tRAC (DRAM) \max - tpd (74F245) \max - tDS (80286) \max$$

at 8MHz

$$= 5 \times 62.5 - 25 - 150 - 8 - 10$$

$$= 312.5 - 25 - 150 - 8 - 10$$

$$= 312.5 - 193 = 119.5 \text{ ns}$$

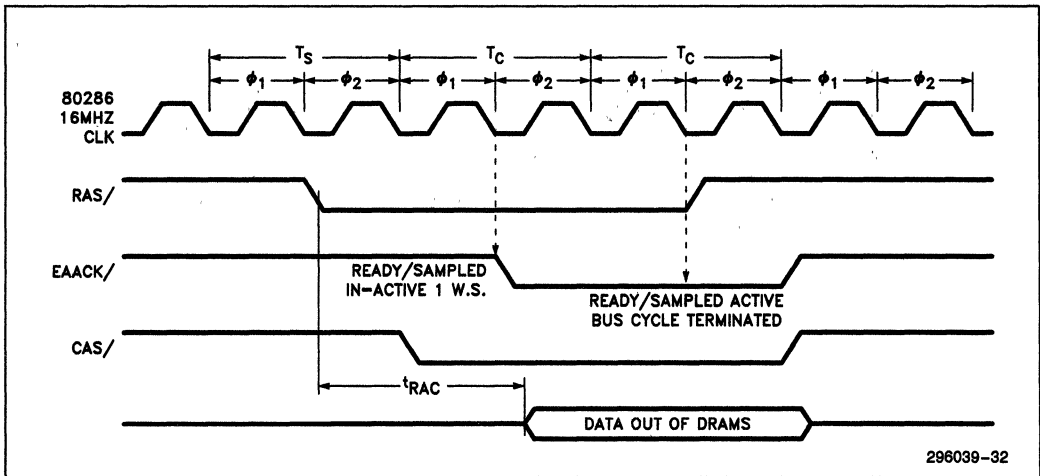


Figure 6.10 Read Cycle for C1 Configuration (1.W.S)

If the wait state was not introduced, the access time from \overline{RAS} =

$$= 3 \times 62.5 - 25 - 150 - 8 - 10$$

$$= -5.5 \text{ ns... Data setup time is not met.}$$

In conclusion, 1 W.S. is essential for proper memory operation.

Access Time From \overline{CAS}

$4TCLCL - \overline{CAS}$ active delay - Access time from \overline{CAS} - propagation delay through transceiver - data setup time required by the processor

$$= 4TCLCL - TCLCSL - tCAC - tpd (74F245) - tDS (80286)$$

$$= 250 - 35 - 65 - 8 - 10$$

$$= 132 \text{ ns}$$

In this case the access time from \overline{RAS} is the limiting factor.

Write Data Setup and Hold Time Margins

In write cycle, the write data must:

1. Reach the DRAMs long enough before the activation of \overline{CAS} to meet the data setup requirement of DRAM.
2. Be held long enough after \overline{CAS} to meet the data hold requirement of DRAMs.

Data setup and hold time calculations are shown below. The timing wave forms are shown in Figure 6.11.

Write Data Setup Time

$2TCLCL + \overline{CAS}$ active delay - Write data valid delay - propagation delay through the transceiver - data setup requirement of DRAM.

$2TCLCL + TCLCSL (82C08) \text{ min} - \text{Write data valid delay } (80286) \text{ max} - tpd (74F245) \text{ max} - tDS (DRAM) \text{ max.}$

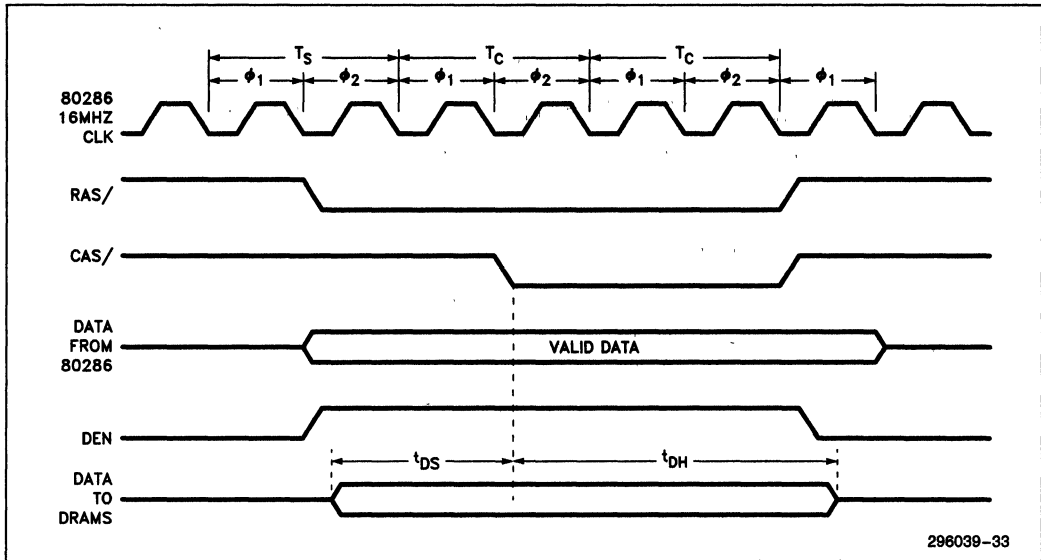


Figure 6.11 Write Data Timing Waveforms for C1 Configuration (1.W.S)

$$125 + 0 - 50 - 8 - 0$$

$$= 125 - 58$$

$$= 67 \text{ ns}$$

Figure 6.12 Wait States for 8026 for Different DRAM Speeds

	DRAM Speed	100 ns	120 ns	150 ns	200 ns
80286	6 MHz	0 w.s.	0 w.s.	0 w.s.	NA
	8 MHz	0 w.s.	0 w.s.	(C1) 1 w.s.	NA
	10 MHz	0 w.s.	NA		
	Config	C0	C0		

The 74F245 transceiver is rated at 50 Pf. The capacitive loading on the data bus is usually much greater, say 300 Pf for a medium sized application. The propagation delay of the transceiver is much greater at 300 Pf, so appropriate derating has to be applied. A typical rule of thumb is to use a derating factor of 0.02 ns/Pf for excess capacitive loading.

$$t_{pd} (74F245)$$

$$= 8 + 0.05 (300 - 50)$$

$$= 8 + 0.02 (250)$$

$$= 8 + 5 = 13 \text{ ns}$$

Write data hold time

- = Delay from activation of $\overline{\text{CAS}}$ to data becoming invalid
- = $5TCLCL + \text{data invalid delay} + \text{propagation delay through the transceiver} - \overline{\text{CAS}} \text{ active delay} - \text{data hold time requirement of DRAM.}$

$$5TCLCL + 0 + t_{pd} (74F245) - TCLCSH (82C08)$$

$$\text{max} - t_{DH} (\text{DRAM})$$

$$= 312.5 + 8 - 35 - 30$$

$$= 255.5 \text{ ns}$$

CONCLUSION

This design will operate, at worst case, with one wait state, at microprocessor frequencies of up to 8 MHz using slow 150 ns access time dynamic RAMs.

CHAPTER 7

INTERFACING 82C08 CHMOS DRAM CONTROLLER TO STATIC COLUMN 64K × 4 DYNAMIC RAM

The 82C08 can support the normal mode of static column and ripple mode DRAMs. It does not however support the static column or the ripple mode fast transfer cycles.

This chapter describes a method of interfacing static column 64K × 4 dynamic RAM (SCRAM) to an 80186 microprocessor under control of the 82C08 dynamic Ram controller. There is a considerable reduction in board space and chip count, when using the SCRAM as opposed to the conventional 64k × 1 dynamic RAMs at the expense of memory density.

Figure 7.1 is a block diagram of a design using 82C08 dynamic RAM controller and the SCRAMs, showing all of the relevant logic. The read signal from the processor is 'anded' with the port enable signal of the 82C08 and connected to the SCRAM output enable pin. The read signal is chosen, as the output enable is active only when performing a read cycle. The read cycle is 'anded' with the port enable signal, to ensure that the output enable is not activated by any other bus cycle and therefore cause bus contention.

The example being considered assumes a 8 MHz 80186 interfacing to 150 ns access SCRAMs via the 82C08. A detailed analysis of the interface between the 80186 microprocessor and the 82C08 dynamic RAM controller is illustrated in Chapter 5 "Interfacing the Dynamic RAM Controller to the iAPX 80186".

TIMING ANALYSIS

As shown in the schematic, the output enable pin on the SCRAM is tied to the read signal, qualified by the dynamic RAM chip select signal MCS0. The timing analysis for this interface is covered in the paragraphs to follow.

Timing Analysis

This analysis is carried out to verify if the data set-up and hold timings are met for an 8MHz 80186 microprocessor, when performing a read operation.

The timing analysis has been carried out based on the following assumptions:

SCRAM CHARACTERISTICS

- tCAC = 30 ns.
- tRAC = 150 ns.
- tCAA = 70 ns.
- tOAC = 25 ns.

Data Set-Up Time

The data out of the SCRAM is dependent upon the four specifications.

- tCAC - Access time from CAS
- tRAC - Access time from RAS
- tCAA - Access time from column address
- tOAC - Access time from output enable

Either of these four parameters may be the limiting factor in determining the RAM access time.

I. Data out with reference to $\overline{\text{CAS}}$: (Equation 1)

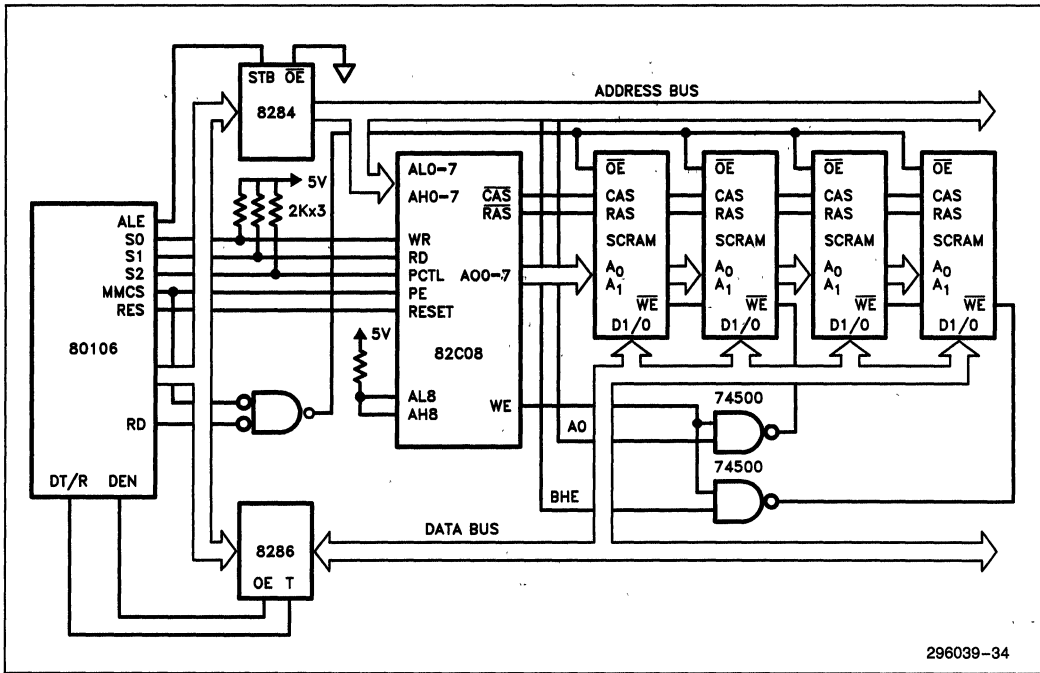
$$\begin{aligned} &\overline{\text{CAS}} \text{ active from clock } \downarrow \text{ delay (82C08) + Access} \\ &\text{time from } \overline{\text{CAS}}(\text{SCRAM}) \\ &= \text{TCLCSL (82C08) max + tCAC (SCRAM)} \\ &\text{max.} \\ &= 122.4 + 30 = 152.4 \text{ ns} \end{aligned}$$

II. Data out with reference to $\overline{\text{RAS}}$: (Equation 2)

$$\begin{aligned} &\overline{\text{RAS}} \text{ active from clock } \downarrow \text{ delay (82C08) max +} \\ &\text{Access time from } \overline{\text{RAS}}(\text{SCRAM}) \text{ max} \\ &= \text{TCLRSL (82C08) max + tRAC (SCRAM)} \\ &\text{max} \\ &= 35 + 150 = 185 \text{ ns} \end{aligned}$$

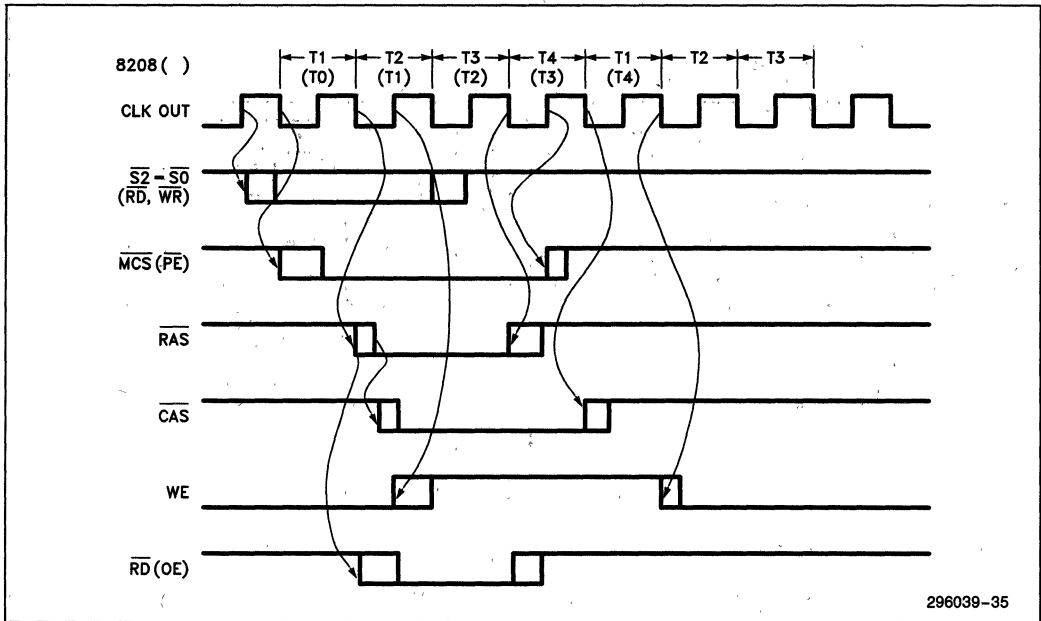
III. Data out with reference to column address: (Equation 3)

$$\begin{aligned} &\overline{\text{CAS}} \text{ active from clock } \downarrow \text{ delay (82C08) max +} \\ &\text{Access time from column address (SCRAM) max} \\ &- \text{column AO to } \overline{\text{CAS}} \text{ set-up (82C08) min} \\ &= \text{TCLCSL (82C08) max + tCAA (SCRAM)} \\ &\text{max - tASC (82C08) min} \\ &= 122.4 + 70 - 5 = 187.4 \text{ ns} \end{aligned}$$



296039-34

Figure 7.1 80186/82C08 Interface Block Diagram



296039-35

Figure 7.2 80186/82C08 Interface Block Diagram

IV. Data out with reference to output enable:
(Equation 4)

$$\begin{aligned}
 & \text{Rd active delay (80186) max} + \text{buff delay (74LS32) max} + \text{Access time from OE/ (SCRAM) max} \\
 & = \text{TCLRL max (80186)} + \text{tpd (74LS32) max} + \text{tOAC max (SCRAM)} \\
 & = 70 + 25 + 25 = 130 \text{ ns}
 \end{aligned}$$

The calculations indicate that the delay from the column address is the limiting factor as it introduces the most delay. Using this value to see if the 80186's data set-up time is met:

$$\begin{aligned}
 & 2 \text{ clks} - \text{Access time from column address (Equation 3)} - \text{buff delay} \geq \text{Data set-up (80186)} \\
 & 2 \text{ clks} - (\text{Equation 3}) - \text{tpd (8286) max} \geq \text{TDVCL (80186)} \\
 & 250 - 187.4 \text{ ns} - 30 \\
 & = 32.6 \text{ ns} > 20 \text{ ns}
 \end{aligned}$$

So, the timing requirement is met.

Data Hold Time

In this case, read signal from the processor is the most limiting factor, as this signal becomes inactive first. Using this parameter, and computing data hold time:

$$\begin{aligned}
 & \text{Rd inactive delay (80186) min.} + \text{tpd (8286) min.} \\
 & + \text{buffer delay (LS32) min.} > \text{Data in hold}
 \end{aligned}$$

$$\begin{aligned}
 & \text{TCLRH (80186) min} + \text{tpd (74LS32) min.} + \text{tpd (8286) min} \geq \text{TCLDX (80186)} \\
 & 10 + 5 + 5 = 20 \text{ ns} > 10 \text{ ns}
 \end{aligned}$$

Therefore, the data set-up and hold timings are satisfied.

"tARH" Column Address Hold Time To RAS

The additional timing parameter unique to static column mode DRAMs is tARH (column address hold time to RAS). The static column mode DRAMs do not latch the row addresses during a read cycle. So the row addresses have to be held valid for the duration RAS is valid. This timing waveform is shown in Figure 7.2 and calculated in Equation 9. It is seen from the calculation that this timing requirement is satisfied.

Column Address Hold time to RAS: Equation 9.

$$\begin{aligned}
 & \text{TCLCL} - \text{RAS Inactive delay (82C08) max} - \text{delay in switching from column to row address.} \\
 & @ 8\text{MHz} \\
 & 125 - 50 - 0 = 75 \text{ ns.}
 \end{aligned}$$

Conclusion

The 82C08 can support the normal access feature of the static column and the ripple mode DRAMs.

CHAPTER 8

POWER DOWN

8.1 INTRODUCTION

The 82C08 supports two modes of operation

- Normal mode
- Power Down mode

Following initialization, the 82C08 operates in the normal mode. In this mode, the 82C08 uses system power supply and clock to perform memory and refresh cycles. The 82C08 monitors the Power Down Detect input (Pin 17) at each clock cycle. If the PDD input is sampled high, the power down mode is invoked. When operating in the power down mode, the 82C08 will only perform refresh cycles to preserve memory contents. The 82C08 operates in the power down mode until the PDD input goes inactive, whereupon, it returns to the normal mode.

8.2 POWER DOWN MODE

8.2.1 Power Supplies and Clocks:

The 82C08 has two power pins (Pins 24, 48), one supplies power to the 82C08 logic (Pin 48) and the other supplies power to the output buffers (Pin 24). During power down, all the logic circuitry and the output buffers are connected to the logic Vcc. So during power down mode, only logic Vcc need be tied to the back up power supply. But in order to shorten the power up time, it is recommended that both pins be tied to the same power supply.

During power down, clock is supplied only to the refresh generation logic circuitry. This clock is derived from the Power Down Clock input. To reduce power dissipation during power down, the PDCLK can be slowed down to support the extended refresh feature of certain CMOS DRAMs, wherein the refresh period can be extended up to 64 ms versus the 4 ms during non-extended cycles.

When supporting the extended refresh at power down, the DRAMs must be refreshed completely, both before going into and after coming out of the power down mode, without allowing active memory cycles. If the user cannot guarantee this idle time requirement, the

82C08 can be programmed to perform three consecutive burst refresh cycles which are activated internally, by strapping the 'RFRQ' input high at reset. The difference between power down burst refresh and the standard burst refresh during normal mode of operation is that during power down the 256 refresh cycles are generated against 128 burst cycles during normal burst cycles. This allows the 82C08 to support the extended refresh DRAMs that have 512 rows.

8.2.2 POWER DOWN PROCEDURE

The 82C08 will not service the power down request during initialization. Upon detection of power down, and after initialization, the 82C08 saves its configuration status, refresh counter content, and performs three power down burst cycles if programmed in the failsafe refresh mode (strapping RFRQ high). A low level at the RFRQ enables the 82C08 to enter power down immediately without executing any burst cycles. After the burst cycles, the 82C08 will continue refreshing the next address location. When the PDD input is deactivated, the 82C08 issues an internal reset. This internal reset does not reprogram the device or change the contents of the refresh address counter. Following the internal reset, the 82C08 performs 3 power down burst cycles, which refreshes the entire memory array. The 82C08 will not service memory requests during the three power down burst cycles. Reset should not be activated, when exiting from the power down mode, as this will cause the 82C08 to reprogram itself, and corrupt memory contents.

8.2.3 POWER DOWN REQUIREMENTS

The Power Down Detect is a user generated signal, used to inform the 82C08 of a power failure. The 82C08 samples the PDD input after the completion of the programming and the RAM warm up period. In the slow cycle mode, the system power and clock should be stable at least 3100 system clocks, while in the fast cycle mode the clock and power must be stable at least 4700 system clocks, following the activation of the PDD signal for the three burst cycles. If the three PD burst cycles are not required by the DRAMs, then the 'RFRQ' must be strapped low, and the clock and the power must remain stable for at least 20 system clocks, following the activation of the PDD signal. Additionally the system power and clock should be stable prior to deactivating the PDD signal.

8.2.4 82C08 Outputs on Power Down

Four of the 82C08's outputs are not activated during power down, AACK, CAS 0 - 1, and WE. These outputs are forced to an inactive state. The Address Outputs and the RAS outputs will switch to perform RAS only refresh cycles. The RAS lines have internal pull-ups to reduce DRAM power consumption. The size of the output buffers, in the power down mode, is much smaller than the buffers used during normal operations. This increases the propagation delay through the buffers during power down. This is done to reduce the speed of charging and discharging the outputs, which results in a reduction in the spikes on the power lines. This is especially required, as there is a single power supply that drives the logic circuits as well as the buffers. All the device inputs are ignored during power down.

8.2.5 Refresh During Power Down

The 82C08 has two clock inputs, the system clock and power down clock. The power down clock may be generated externally from a crystal oscillator circuit. When entering the power down mode, the 82C08 disables the system clock and uses the PDCLK to generate refresh cycles. During power down, RAS only refresh cycles are performed. In the power down mode the time interval between refresh cycles is 5 PDCLKs and this is fixed for all applications.

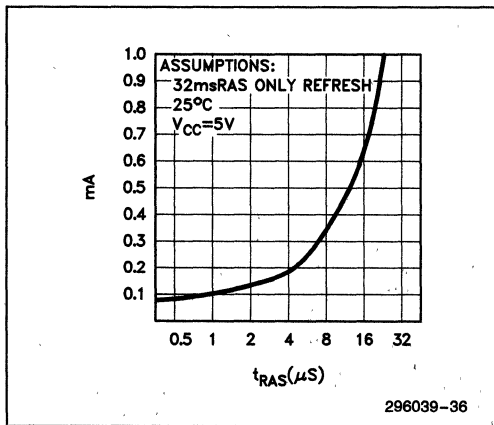


Figure 8.1 Data Retention Current (Typical)

The 82C08 supports the extended refresh of certain DRAMs (upto 64 ms) by slowing down the PDCLK frequency. During the power down refresh cycle, the RAS is activated for one PDCLK cycle only. At extended refresh, the PDCLK frequency will be below 50 KHz and this will cause a long duration of the RAS signal. But reducing the PDCLK creates another problem. CMOS DRAMs have the data retention characteristics as shown in Figure 8.1. The current rapidly increases as the duration of the RAS low pulse increases beyond one or two microseconds. To minimize the RAS low pulse, the two RC networks, shown in Figure 8.2, are designed to insert one very fast cycle whenever RAS is low (1 microsecond). The time constant of the RC1 and RC2 should be centered around 300 ns and 100 ns respectively.

PDCLK FREQUENCY

At power down the count interval will have four long clocks and one short clock (1 microsecond duration). The equation for calculating the PDCLK frequency will be:

$$4/f + 1 + 8 tp \geq (a \cdot 1000)/n \text{ Equation 10}$$

where

- f — The PDCLK frequency in MHz
- tp — The duration of the system clock in microseconds.
- a — The extended refresh period of DRAMs in ms
- n — The number of rows in the DRAMs.

For example:

Let us assume that the system clock is 8 MHz ($tp = 0.125$). We are supporting a DRAM which has an extended refresh of 32 ms ($a = 32$) and the number of rows is 256 ($n = 256$). Then by substituting these values in the Equation 10 we get $f = 32.52$ KHz.

Note: Even when supporting the standard refresh period of 4 ms, the PDCLK frequency will not exceed 300 KHz.

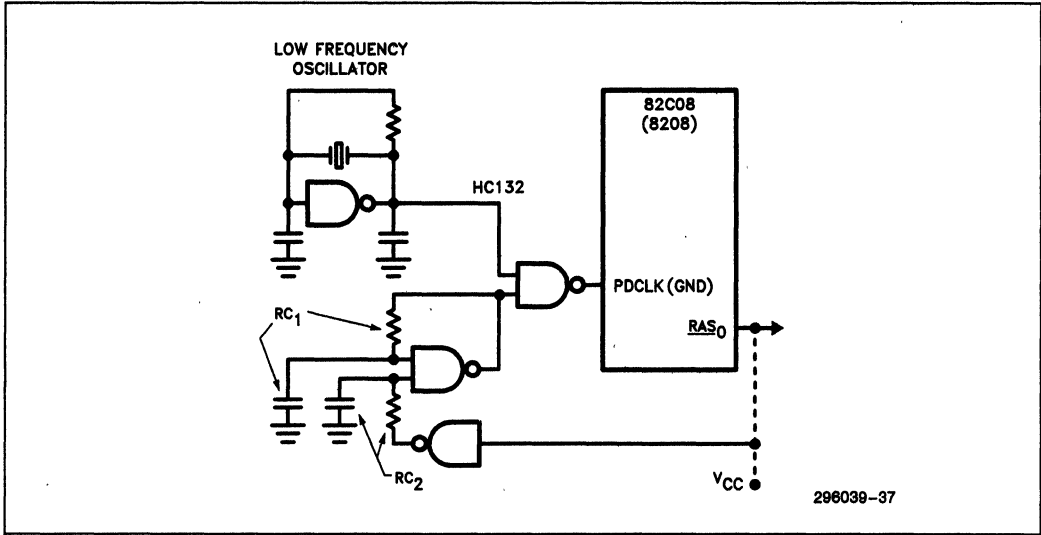


Figure 8.2 Low Frequency Oscillator

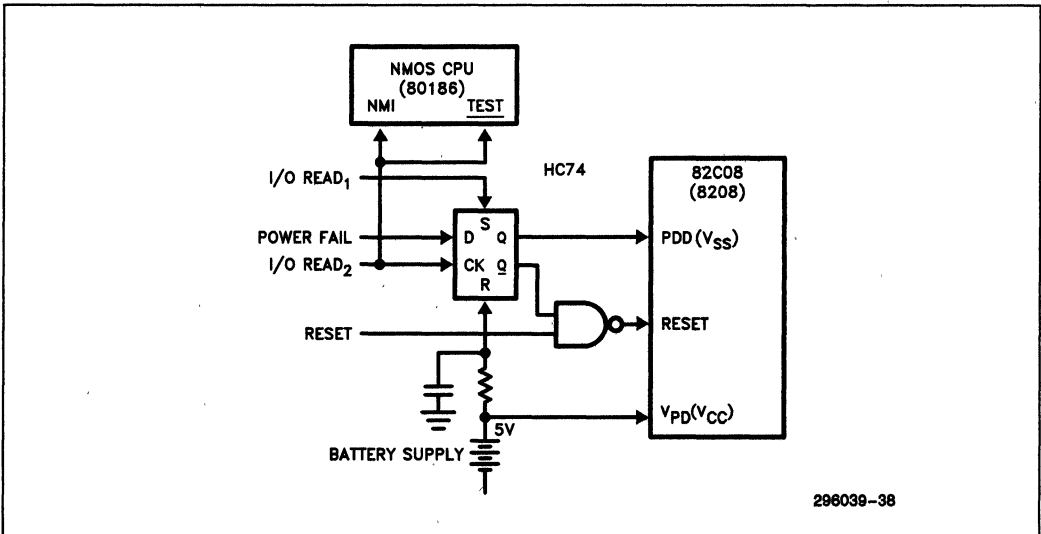


Figure 8.3 Power Down Control

8.2.6 Power Down Control

Figure 8.3 shows one method of invoking the power down mode of the 82C08. The logical flow is shown in Figure 8.4. The example considered assumes that battery backup on power fail is a required mode of operation. The power fail signal is generated by the power supply. Upon power fail, the NMI causes the CPU to store all the registers in predetermined locations in memory. The CPU then activates the Power Down Detect signal by accessing a predefined I/O location. The 82C08 internally synchronizes the PDD input and performs three burst refresh cycles before synchronizing to the PDCLK. Once set, the PDD signal cannot be reset by the CPU unless the power is back on. This prevents spurious signals from deactivating the power down mode erroneously. The internal reset circuitry has been designed to prevent resetting the 82C08 when the PDD signal is active since this would result in loss of data. The CPU can gain access to memory by first resetting the PDD signal by reading the predefined I/O address. When the PDD is deactivated, the 82C08 automatically synchronizes to the system clock and then performs three burst refresh cycles before allowing the CPU to access memory.

8.3 SUMMARY

The expected data retention current for the 82C08, in the power down mode when operating with a PDCLK of 32.768 KHz is

16 CMOS DRAMs	= 1.60 mA
(Estimated values for CMOS DRAMs)	
82C08	= 1.00 mA
@ 32.768 KHz (estimated)	
32.768 KHz Oscillator	= 0.30 mA
HC 132, HC 74, HC 04	= 0.03 mA
Total	2.93 mA

A 1 A hr battery will retain data for ~ 340 hrs or 14 days.

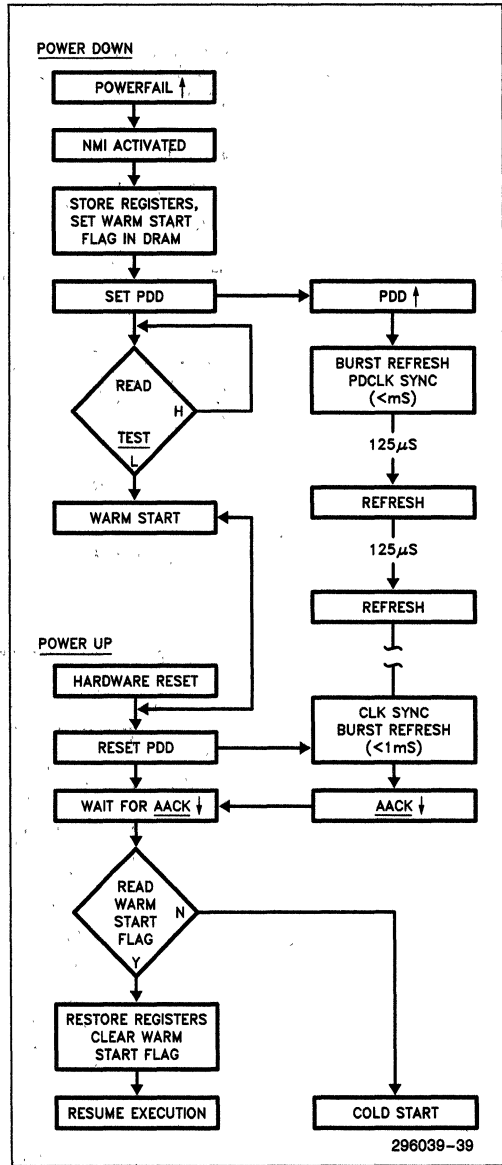


Figure 8.4 Power Up/Power Down Sequence

CHAPTER 9

BOARD LAYOUT CONSIDERATIONS AND COMMON PROBLEM AREAS

The purpose of this chapter is to forewarn new designers of the problems they might encounter. This chapter also identifies some of the common problem areas, and the steps to be taken.

9.1 The need for damping resistors:

Most memory system designs, for 16 bit processors, will comprise from 16 to 32 DRAMs, providing 1 Mbyte of RAM. The 82C08 address output goes to every DRAM. The $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ outputs go only to one DRAM bank (16 DRAMs).

Implementing this typical memory system is simple, provided a little care and attention is paid in avoiding certain areas which can be easily overlooked. The two areas in particular, that could cause memory errors or damage, are:

1. Undershoot and overshoot caused by inductive traces and high capacitive loads.
2. Switching spikes caused by switching capacitive loads.

9.1.1 Undershoot and Overshoot

Using a large number of DRAMs result in high capacitive loads which are caused by a combination of RAM input capacitance and trace capacitance. The input capacitance is further compounded by wire wrapping. The trace capacitance and the lead inductance combine to present a complex impedance to the drivers, which results in overshoot and undershoot, each time a memory driver changes state. In order to change the state, the load capacitance have to be charged or discharged, which cannot be done instantaneously due to current limitations. This results in a spike, that lasts as long as it takes to change the voltages of all the capacitances. This current, if large (depends upon the trace inductance. Even a small inductance can cause an excessive voltage across the inductance) can cause

1. Damage to the DRAMs at the far end of the trace.
2. Glitches, which if they occur on the $\overline{\text{RAS}}$ lines, can destroy the contents of memory.

To reduce current spikes, it is necessary to provide series damping resistors in the path between the DRAMs and the controller. The value for the resistor, which is application dependent, should be selected to cause critical damping. Too high a value can cause overdamping,

which will cause slow transistions. This increased switching delay can cause excessive skew problems, and will violate certain DRAM timing requirements, such as column address set up time (tASC) or row address hold time (tARH). Using too small a value for this resistor, will not correct the problem of undershoot and overshoot will still exist. Typically, the resistor value will vary between 33 Ohms - 60 Ohms, depending on the loading (lighter capacitive loads require higher value of damping resistor). Generally the correct resistor value must be chosen empirically.

9.1.2 SWITCHING CURRENT SPIKES

Another undesirable effect of the current spikes, is its effect on the Vcc and GND pins. When performing refreshes, the 82C08 simultaneously activates both the $\overline{\text{RAS}}$ lines. If all the address outputs switch in the same direction during the refresh cycle, it will result in a large current surge in a very short time period. This can cause excessive voltage drops, during this switching time, which may upset the latches internal to the 82C08 and result in incorrect operation. To reduce this spike, a 22 uF ceramic capacitor in parallel with a 0.1 uF capacitor, should be placed between Pins 24 (Power pin to the drivers) and GND, and a 1 uF ceramic capacitor between Pins 48 (Power pin to the logic circuits) and GND. Additionally each DRAM must have its own decade capacitor (typically 0.1 uF). The capacitors should be placed as close to the leads as possible, to minimize lead inductance.

9.2 GENERAL PROBLEM AREAS

Listed below, are some of the common problems that have been encountered:

1. $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ disappears or remains active indefinitely.
2. No $\overline{\text{AACK}}$.
3. Incorrect refresh rate.

When experiencing the above mentioned problems, it is necessary to ensure

1. The clock input into the 82C08 is a MOS level clock.
2. No glitches on the $\overline{\text{RD}}$, $\overline{\text{WR}}$ or the RESET inputs. This should be checked using an Oscilloscope, as it can be easily overlooked when using a logic analyzer.

3. Check to see that the correct program data bits are being read in by verifying that the refreshes are being performed at the correct intervals (14 - 15 ms).
4. Unnecessary buffering of signals, which may violate certain RAM parameters.
5. Chose the correct damping resistors on the $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and address outputs.
6. In a synchronous application, never buffer the Clock or the $\overline{\text{RD}}$, $\overline{\text{WR}}$ lines (status outputs).



8231A ARITHMETIC PROCESSING UNIT

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

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

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

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

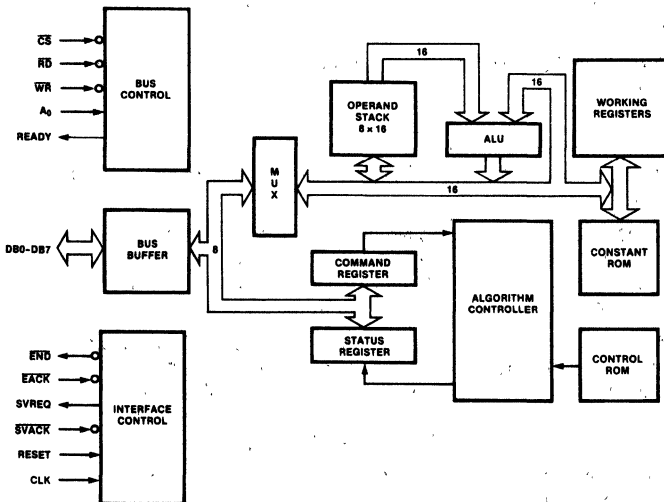


Figure 1. Block Diagram

231305-1

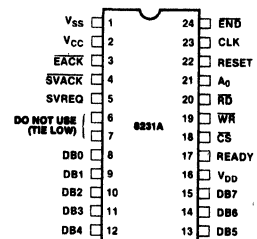


Figure 2. Pin Configuration

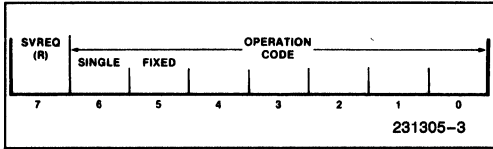
231305-2

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function																				
V _{CC}	2		POWER: +5V power supply.																				
V _{DD}	16		POWER: +12V power supply.																				
V _{SS}	1		GROUND.																				
CLK	23	I	CLOCK: An external, TTL compatible, timing source is applied to the CLK pin.																				
RESET	22	I	RESET: The active high reset signal provides initialization for the chip. RESET also terminates any operation in progress. RESET clears the status register and places the 8231A into the idle state. Stack contents and command registers are not affected (5 clock cycles).																				
\overline{CS}	18	I	CHIP SELECT: \overline{CS} is an active low input signal which selects the 8231A and enables communication with the data bus.																				
A ₀	21	I	ADDRESS: In conjunction with the \overline{RD} and \overline{WR} signals, the A ₀ control line establishes the type of communication that is to be performed with the 8231A as shown below:																				
			<table border="1"> <thead> <tr> <th>A₀</th> <th>\overline{RD}</th> <th>\overline{WR}</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>0</td> <td>Enter data byte into stack</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>Read data byte from stack</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>Enter command</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>Read status</td> </tr> </tbody> </table>	A ₀	\overline{RD}	\overline{WR}	Function	0	1	0	Enter data byte into stack	0	0	1	Read data byte from stack	1	1	0	Enter command	1	0	1	Read status
A ₀	\overline{RD}	\overline{WR}	Function																				
0	1	0	Enter data byte into stack																				
0	0	1	Read data byte from stack																				
1	1	0	Enter command																				
1	0	1	Read status																				
\overline{RD}	20	I	READ: This active low input indicates that data or status is to be read from the 8231A if \overline{CS} is low.																				
\overline{WR}	19	I	WRITE: This active low input indicates that data or a command is to be written into the 8231A if \overline{CS} is low.																				
\overline{EACK}	3	I	END OF EXECUTION: This active low input clears the end of execution output signal (\overline{END}). If \overline{EACK} is tied low, the \overline{END} output will be a pulse that is one clock period wide.																				
\overline{SVACK}	4	I	SERVICE REQUEST: This active low input clears the service request output (\overline{SVREQ}).																				
\overline{END}	24	O	END: This active low, open-drain output indicates that execution of the previously entered command is complete. It can be used as an interrupt request and is cleared by \overline{EACK} , RESET or any read or write access to the 8231.																				
SVREQ	5	O	SERVICE REQUEST: This active high output signal indicates that command execution is complete and that post execution service was requested in the previous command byte. It is cleared by \overline{SVACK} , the next command output to the device, or by RESET.																				
READY	17	O	READY: This active high output indicates that the 8231A is able to accept communication with the data bus. When an attempt is made to read data, write data or to enter a new command while the 8231A is executing a command, READY goes low until execution of the current command is complete (See READY Operation, p. 6).																				
DB0-DB7	8-15	I/O	DATA BUS: These eight bidirectional lines provide for transfer of commands, status and data between the 8231A and the CPU. The 8231A can drive the data bus only when \overline{CS} and \overline{RD} are low.																				

COMMAND STRUCTURE

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



Bits 0–4 select the operation to be performed as shown in the table. Bits 5–6 select the data format appropriate to the selected operation. If bit 5 is a 1, a fixed point data format is specified. If bit 5 is 0, floating point format is specified. Bit 6 selects the preci-

sion of the data to be operated upon by fixed point commands only (if bit 5 = 0, bit 6 must be 0). If bit 6 is a 1, single-precision (16-bit) operands are assumed. If bit 6 is a 0, double-precision (32-bit) operands are indicated. Results are undefined for all illegal combinations of bits in the command byte. Bit 7 indicates whether a service request is to be issued after the command is executed. If bit 7 is a 1, the service request output (SVREQ) will go high at the conclusion of the command and will remain high until reset by a low level on the service acknowledge pin (SVACK) or until completion of execution of the succeeding command where service request (bit 7) is 0. Each command issued to the 8231A requests post execution service based upon the state of bit 7 in the command byte. When bit 7 is a 0, SVREQ remains low.

Table 2. 32-Bit Floating Point Instructions

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

Table 3. 32-Bit Integer Instructions

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

Table 4. 16-Bit Integer Instructions

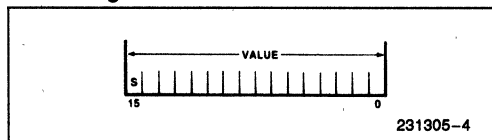
Instruction	Description	Hex ⁽¹⁾ Code	Stack Contents ⁽³⁾ After Execution								Status Flags ⁽⁴⁾ Affected
			A _U	A _L	B _U	B _L	C _U	C _L	D _U	D _L	
CHSS	Change Sign of A _U	7 4	R	A _L	B _U	B _L	C _U	C _L	D _U	D _L	S, Z, O
FIXS	Floating Point to Integer Conversion	1 F	R	B _U	B _L	C _U	C _L	U	U	U	S, Z,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

NOTES:

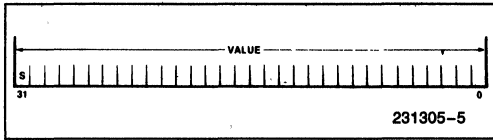
- In the hex code column, SVREQ is a 0.
- The stack initially is composed of four 32-bit numbers (A, B, C, D). A is equivalent to Top Of Stack (TOS) and B is Next On Stack (NOS). Upon completion of a command the stack is composed of: the result (R); undefined (U); or the initial contents (A, B, C, or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

DATA FORMATS

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

Single Precision Fixed Point Format


Double Precision Fixed Point Format



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

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

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

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

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

The 8231A is a binary arithmetic processor and requires that floating point data be represented by a

fractional mantissa value between 0.5 and 1 multiplied by 2 raised to an appropriate power. This is expressed as follows:

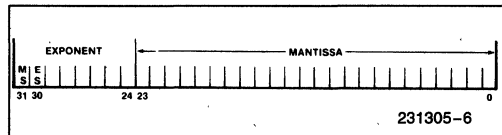
$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

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

$$\begin{aligned} \text{value} &= (2^{-1} + 2^{-2} + 2^{-5} + 2^{-8}) \times 2^7 \\ &= (0.5 + 0.25 + 0.03125 + 0.00290625) \times 128 \\ &= 0.78515625 \times 128 \\ &= 100.5 \end{aligned}$$

FLOATING POINT FORMAT

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

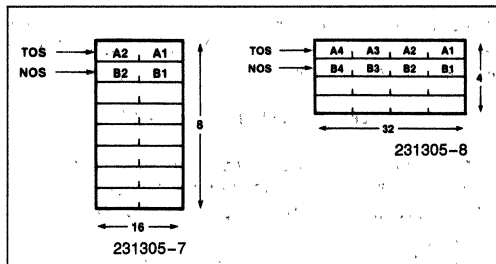


The range of values that can be represented in this format is $\pm(2.7 \times 10^{-20}$ to $9.2 \times 10^{18})$ and zero.

FUNCTIONAL DESCRIPTION

STACK CONTROL

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



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

DATA ENTRY

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

DATA REMOVAL

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

COMMAND ENTRY

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

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

COMMAND COMPLETION

The 8231A signals the completion of each command execution by lowering the End Execution line (\overline{END}). Simultaneously, the busy bit in the status reg-

ister is cleared and the Service Request bit of the command register is checked. If it is a "1" the service request output level (SVREQ) is raised. \overline{END} is cleared on receipt of an active low End Acknowledge (\overline{EACK}) pulse. Similarly, the service request line is cleared by recognition of an active low Service Acknowledge (SVACK) pulse.

READY OPERATION

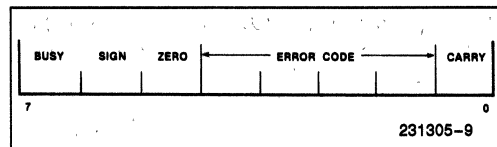
An active high ready (READY) is provided. This line is high in its quiescent state and is pulled low by the 8231A under the following conditions:

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

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

DEVICE STATUS

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



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

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

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

Error Code: This field contains an indication of the validity of the result of the last operation. The error codes are:
 0000—No error
 1000—Divide by zero
 0100—Square root or log of negative number
 1100—Argument of inverse sine, cosine, or e^x too large
 XX10—Underflow
 XX01—Overflow

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

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

READ STATUS

The 8231A status register can be read by the CPU at any time (whether an operation is in progress or

not) by bringing the chip select (\overline{CS}) low, the command/data line (A_0) high, and lowering \overline{RD} . The status register is then gated onto the data bus and may be input by the CPU.

EXECUTION TIMES

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

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

DERIVED FUNCTION DISCUSSION

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

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

Table 5. Command Execution Times

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

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

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

$$T_n(X) = \cos n\theta; \text{ where } n = 0, 1, 2, \dots \quad (1-2)$$

$$\theta = \cos^{-1} X$$

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

$$T_0(X) = \cos(0 \times \theta) = \cos(0) = 1 \quad (1-4)$$

$$T_1(X) = \cos(\cos^{-1} X) = X \quad (1-5)$$

$$T_2(X) = \cos 2\theta = 2\cos^2\theta - 1 = 2\cos^2(\cos^{-1} X) - 1 = 2X^2 - 1 \quad (1-6)$$

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

$$T_n(X) = 2X[T_{n-1}(X)] - T_{n-2}(X); n \geq 2 \quad (1-7)$$

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

$$X^Y = e^{Y \ln X}$$

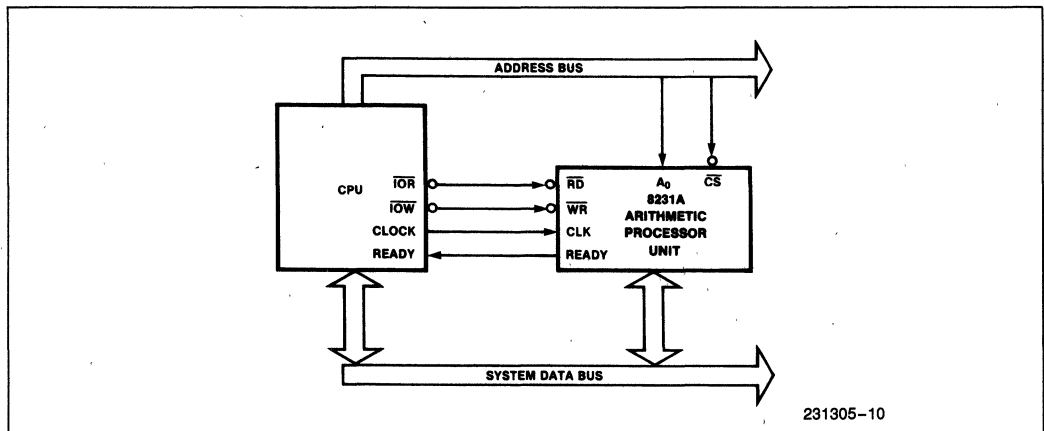
The error for the power function is a combination of that for the logarithm and exponential functions.

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

APPLICATION INFORMATION

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

In many systems it will be convenient to use the microcomputer system clock to drive the APU clock input. In the case of 8080A systems it would be the $\phi 2TTL$ signal. Its cycle time will usually fall in the range of 250 ns to 1000 ns, depending on the system speed.



231305-10

Figure 3. Minimum Configuration Example

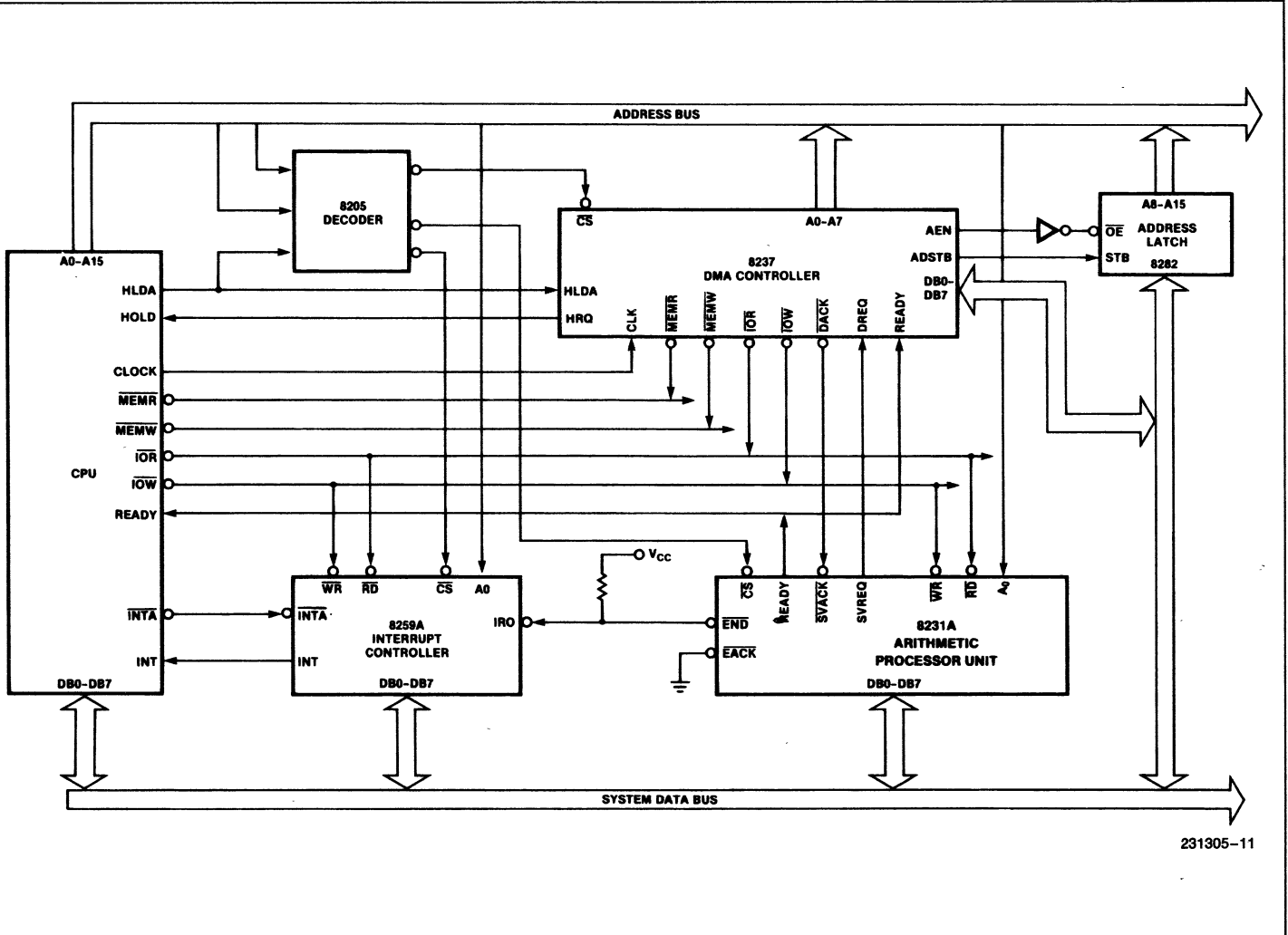


Figure 4. High Performance Configuration Example

ABSOLUTE MAXIMUM RATINGS*

Storage Temperature -65°C to +150°C
 Ambient Temperature Under Bias 0°C to 70°C
 V_{DD} with Respect to V_{SS} -0.5V to +15.0V
 V_{CC} with Respect to V_{SS} -0.5V to +7.0V
 All Signal Voltages with Respect to V_{SS} -0.5V to +7.0V
 Power Dissipation..... 2.0W

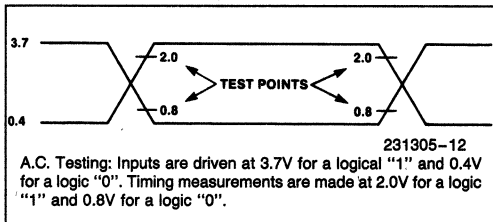
**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.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

D.C. AND OPERATING CHARACTERISTICS

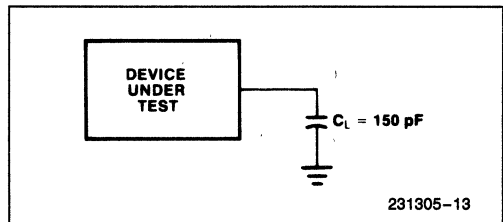
$T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{SS} = 0\text{V}$, $V_{CC} = +5\text{V} \pm 10\%$, $V_{DD} = +12\text{V} \pm 10\%$

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

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{SS} = 0\text{V}$, $V_{CC} = +5\text{V} \pm 10\%$, $V_{DD} = +12\text{V} \pm 10\%$
READ OPERATION

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

WRITE OPERATION

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

OTHER TIMINGS

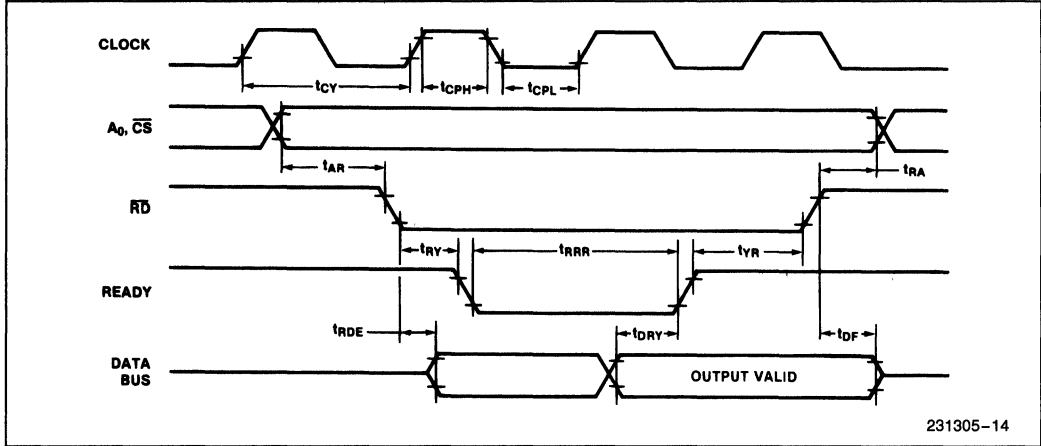
Symbol	Parameter	8231A-8		8231A		Units
		Min	Max	Min	Max	
t _{CY}	Clock Period	480	5000	250	2500	ns
t _{CPH}	Clock Pulse High Width	200		100		ns
t _{CPL}	Clock Pulse Low Width	240		120		ns
t _{EE}	$\overline{\text{END}}$ Pulse Width (Note 5)	400		200		ns
t _{EAE}	$\overline{\text{EACK}} \downarrow$ to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

NOTES:

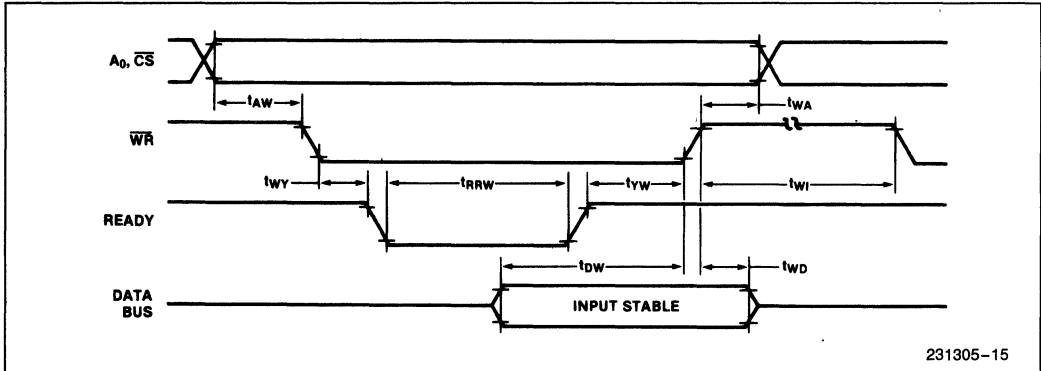
1. Typical values are for $T_A = 25^\circ\text{C}$, nominal supply voltages processing parameters.
2. $\overline{\text{READY}}$ is pulled low for both command and data operations.
3. Minimum values shown assume no previously entered command is being executed for the data access. If a previously entered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

WAVEFORMS

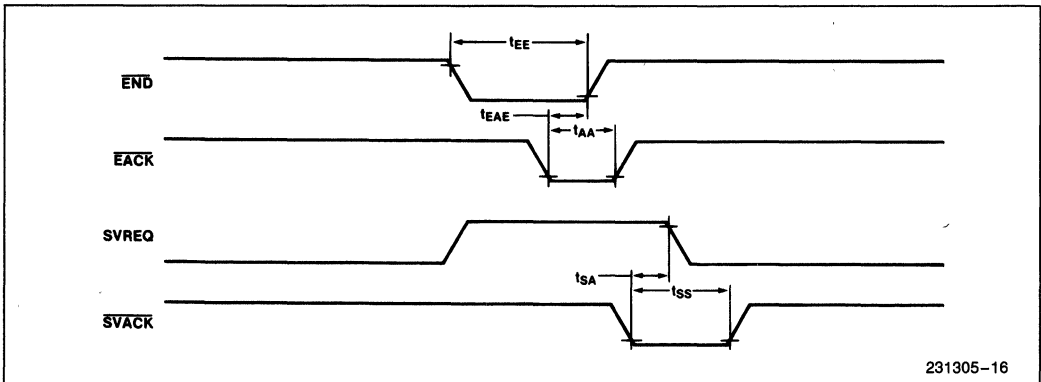
READ OPERATION



WRITE OPERATION



INTERRUPT OPERATION





8253/8253-5 PROGRAMMABLE INTERVAL TIMER

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

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

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

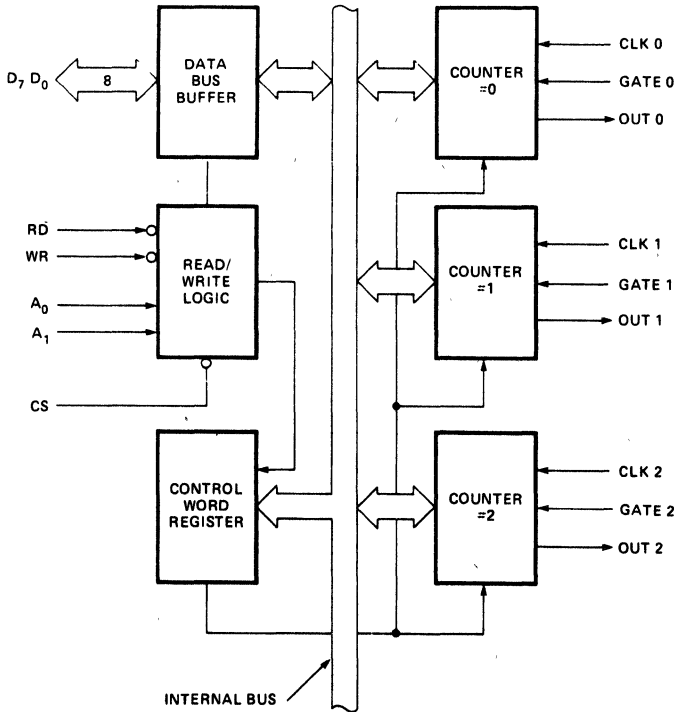


Figure 1. Block Diagram

231306-1

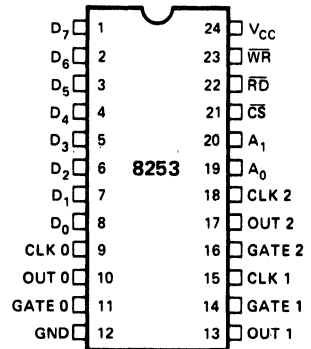


Figure 2. Pin Configuration

231306-2

FUNCTIONAL DESCRIPTION

General

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

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

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

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

Data Bus Buffer

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

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

Read/Write Logic

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

\overline{RD} (Read)

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

\overline{WR} (Write)

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

A0, A1

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

\overline{CS} (Chip Select)

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

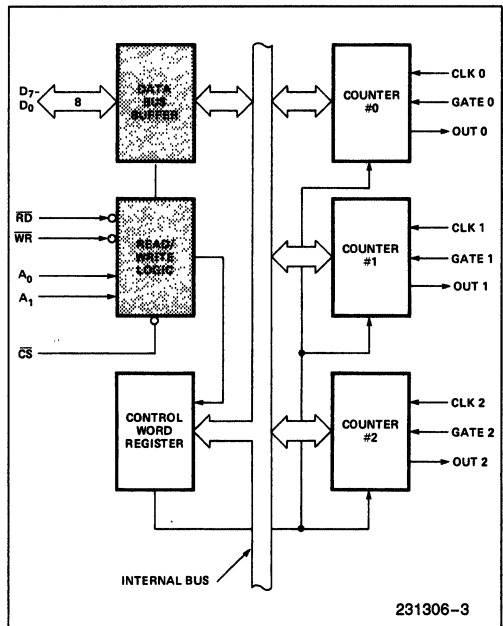


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

CS	RD	WR	A ₁	A ₀	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

Control Word Register

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

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

Counter # 0, Counter # 1, Counter # 2

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

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

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

8253 SYSTEM INTERFACE

The 8253 is a component of the Intel™ Microcomputer systems and interfaces in the same manner as all other peripherals of the family. It is treated by the

systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

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

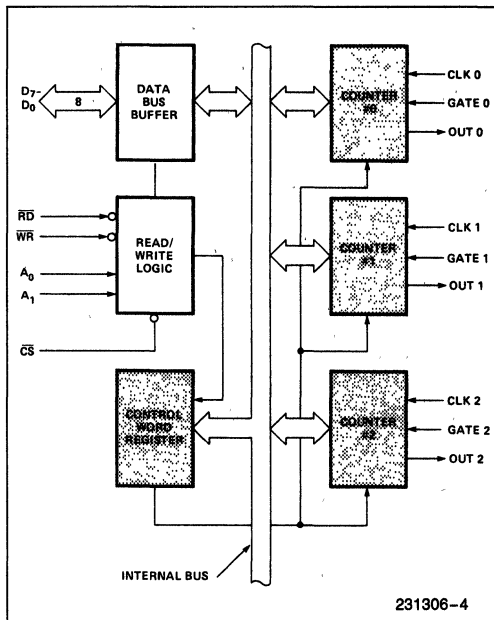


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

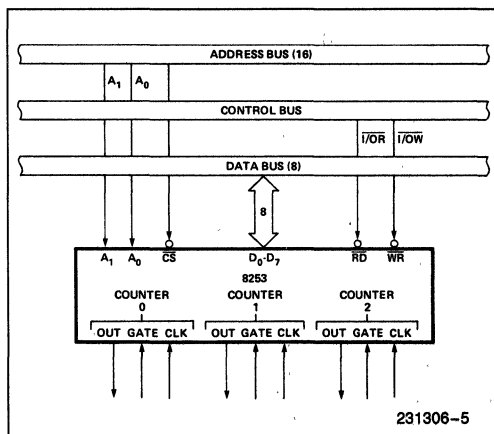


Figure 5. 8253 System Interface

OPERATIONAL DESCRIPTION

General

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

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

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

Programming the 8253

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

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

Control Word Format

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definition Of Control

SC—SELECT COUNTER:

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

RL—READ/LOAD:

RL1 RL0

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

M—MODE:

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

BCD:

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

Counter Loading

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

MODE DEFINITION

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

Rewriting a counter register during counting results in the following:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 6. Gate Pin Operations Summary

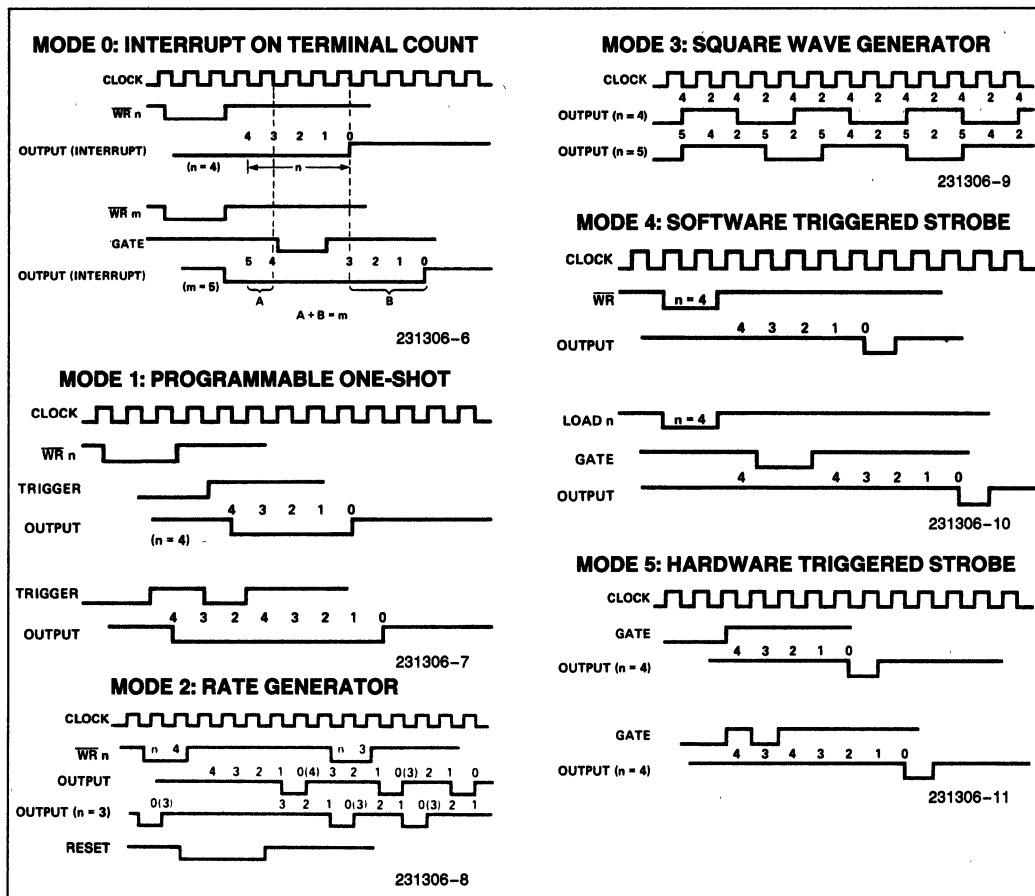


Figure 7. 8253 Timing Diagrams

8253 READ/WRITE PROCEDURE

Write Operations

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

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

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

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

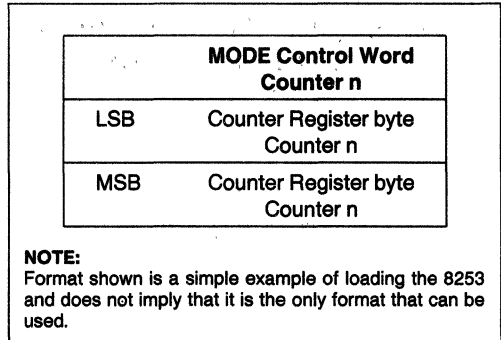


Figure 8. Programming Format

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

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

Figure 9. Alternate Programming Formats

Read Operations

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

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

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

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

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

Read Operation Chart

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

Reading While Counting

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

MODE Register for Latching Count

A0, A1 = 11

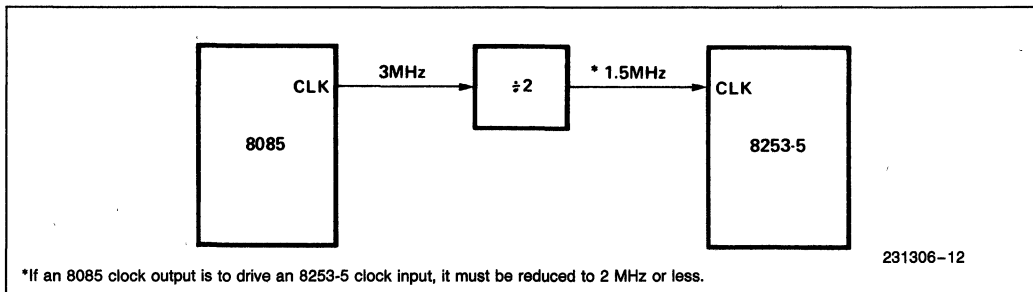
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1, SC0— specify counter to be latched.

D5, D4 — 00 designates counter latching operation.

X — don't care.

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



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

Figure 10. MCS-85™ Clock Interface*

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage On Any Pin
 with Respect to Ground -0.5V to 7V
 Power Dissipation 1 Watt

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

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}, V_{CC} = 5\text{V} \pm 10\%*$

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

CAPACITANCE $T_A = 25^\circ\text{C}, V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	$f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}, V_{CC} = 5.0\text{V} \pm 10\%, \text{GND} = 0\text{V}*$
Bus Parameters⁽³⁾
READ CYCLE

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

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

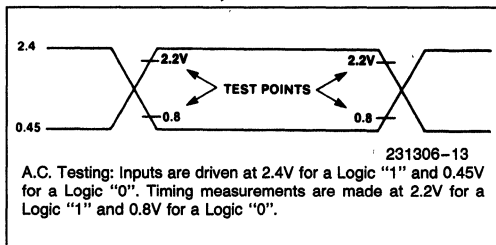
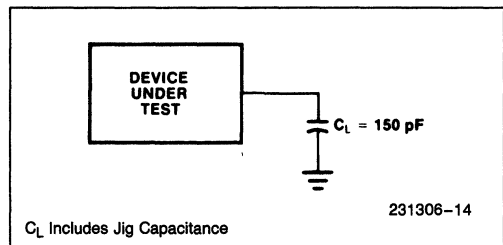
Symbol	Parameter	8253		8253-5		Unit
		Min	Max	Min	Max	
t_{AW}	Address Stable before \overline{WRITE}	50		30		ns
t_{WA}	Address Hold Time for \overline{WRITE}	30		30		ns
t_{WW}	\overline{WRITE} Pulse Width	400		300		ns
t_{DW}	Data Set Up Time for \overline{WRITE}	300		250		ns
t_{WD}	Data Hold Time for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

CLOCK AND GATE TIMING

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

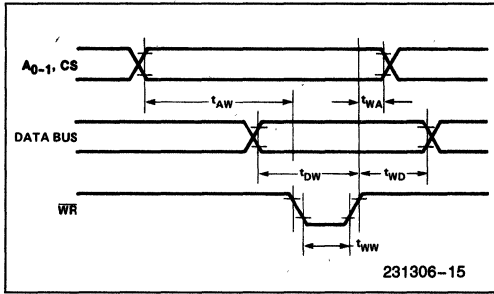
NOTES:

- $I_{OL} = 2.2$ mA.
 - $I_{OH} = -400$ μ A.
 - AC timings measured at $V_{OH} 2.2$, $V_{OL} = 0.8$.
 - $C_L = 150$ pF.
- *For Extended Temperature EXPRESS, use M8253 electrical parameters.

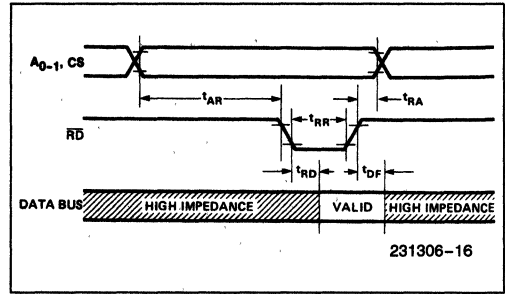
A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. TESTING LOAD CIRCUIT


WAVEFORMS

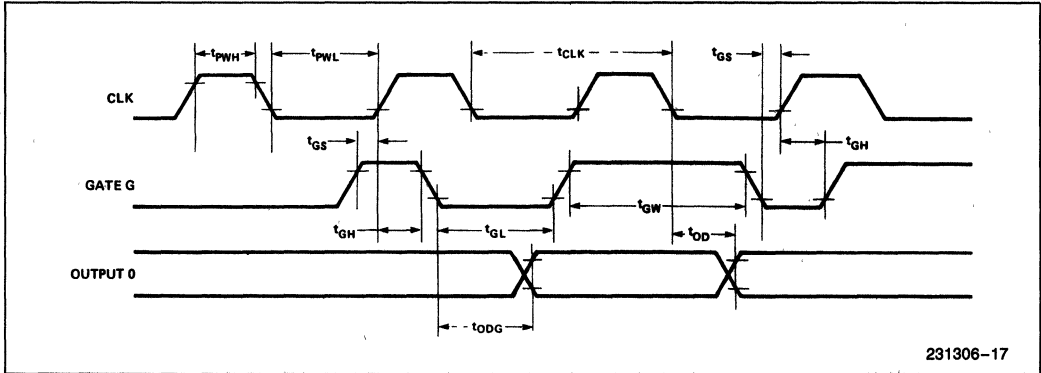
WRITE TIMING



READ TIMING



CLOCK AND GATE TIMING



8254 PROGRAMMABLE INTERVAL TIMER

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

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

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

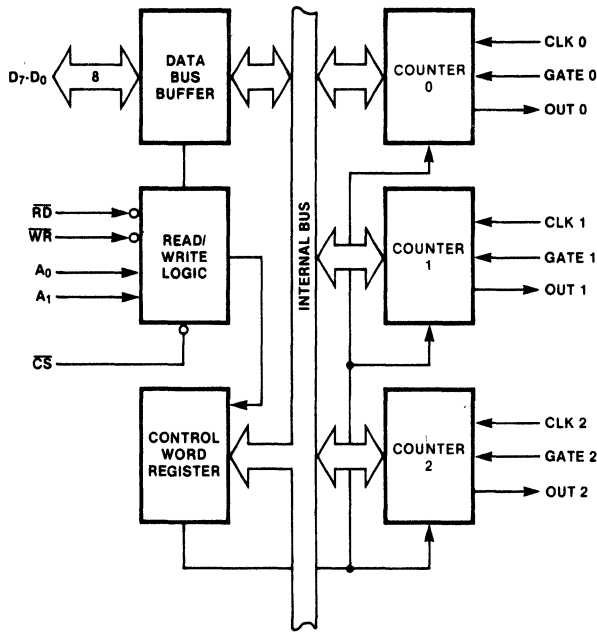


Figure 1. 8254 Block Diagram

231164-1

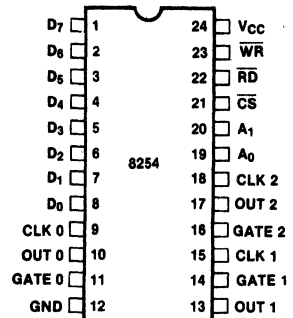


Figure 2. Pin Configuration

231164-2

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function		
D ₇ -D ₀	1-8	I/O	DATA: Bi-directional three state data bus lines, connected to system data bus.		
CLK 0	9	I	CLOCK 0: Clock input of Counter 0.		
OUT 0	10	O	OUTPUT 0: Output of Counter 0.		
GATE 0	11	I	GATE 0: Gate input of Counter 0.		
GND	12		GROUND: Power supply connection.		
V _{CC}	24		POWER: +5V power supply connection.		
WR	23	I	WRITE CONTROL: This input is low during CPU write operations.		
RD	22	I	READ CONTROL: This input is low during CPU read operations.		
CS	21	I	CHIP SELECT: A low on this input enables the 8254 to respond to RD and WR signals. RD and WR are ignored otherwise.		
A ₁ , A ₀	20-19	I	ADDRESS: Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus.		
			A ₁	A ₀	Selects
			0	0	Counter 0
			0	1	Counter 1
			1	0	Counter 2
1	1	Control Word Register			
CLK 2	18	I	CLOCK 2: Clock input of Counter 2.		
OUT 2	17	O	OUT 2: Output of Counter 2.		
GATE 2	16	I	GATE 2: Gate input of Counter 2.		
CLK 1	15	I	CLOCK 1: Clock input of Counter 1.		
GATE 1	14	I	GATE 1: Gate input of Counter 1.		
OUT 1	13	O	OUT 1: Output of Counter 1.		

FUNCTIONAL DESCRIPTION

General

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

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

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

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

Block Diagram

DATA BUS BUFFER

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

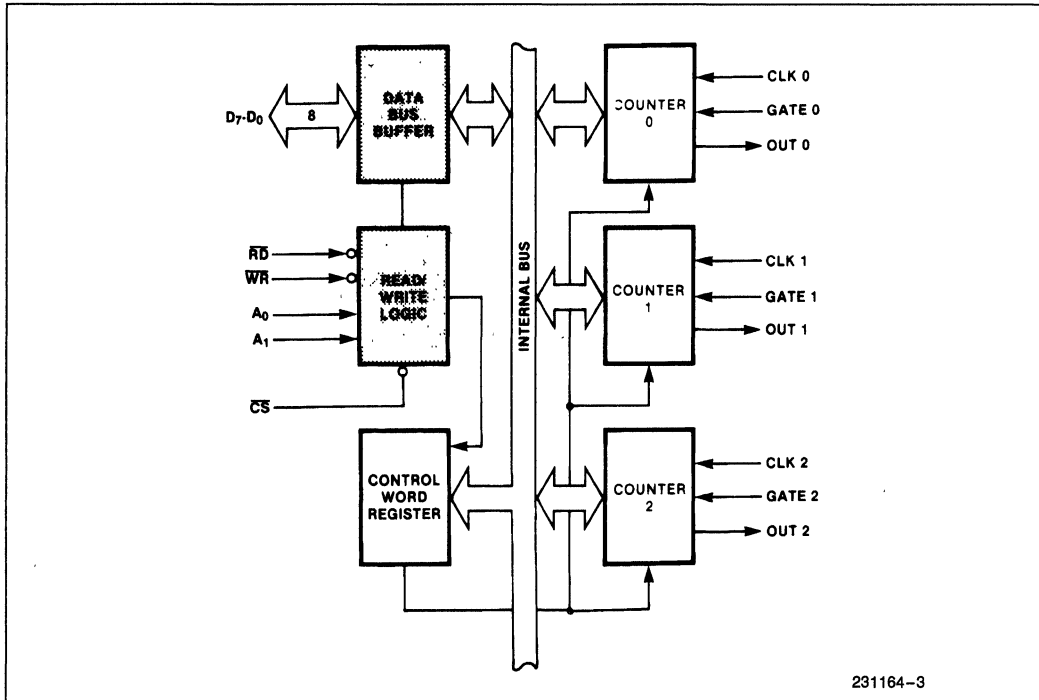


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

READ/WRITE LOGIC

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

CONTROL WORD REGISTER

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

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

COUNTER 0, COUNTER 1, COUNTER 2

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

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

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

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

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

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

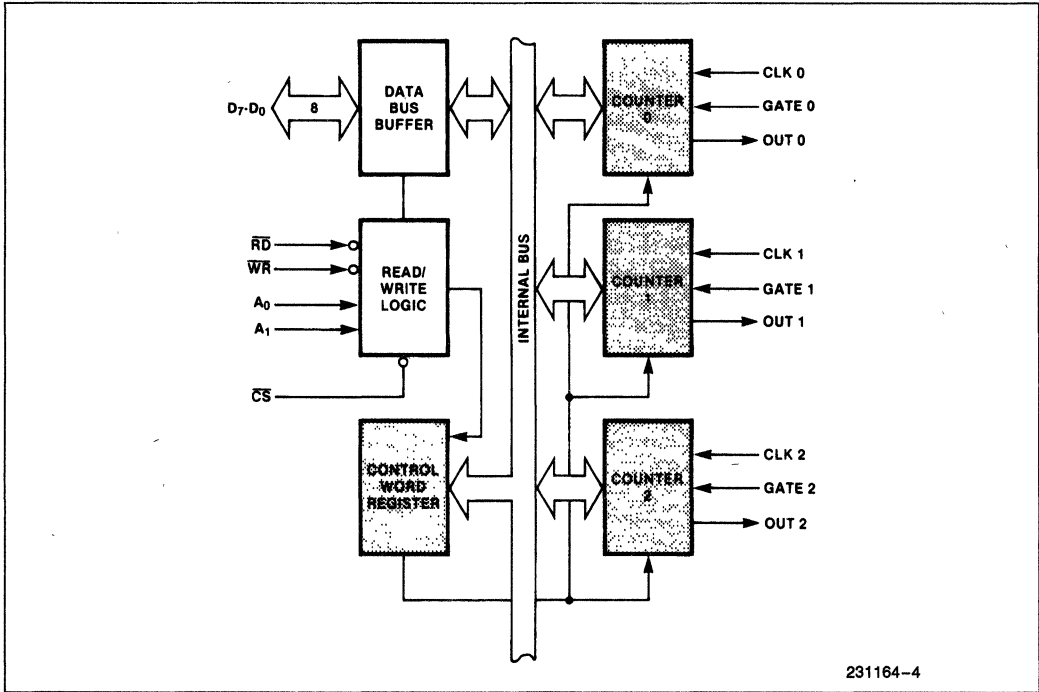


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

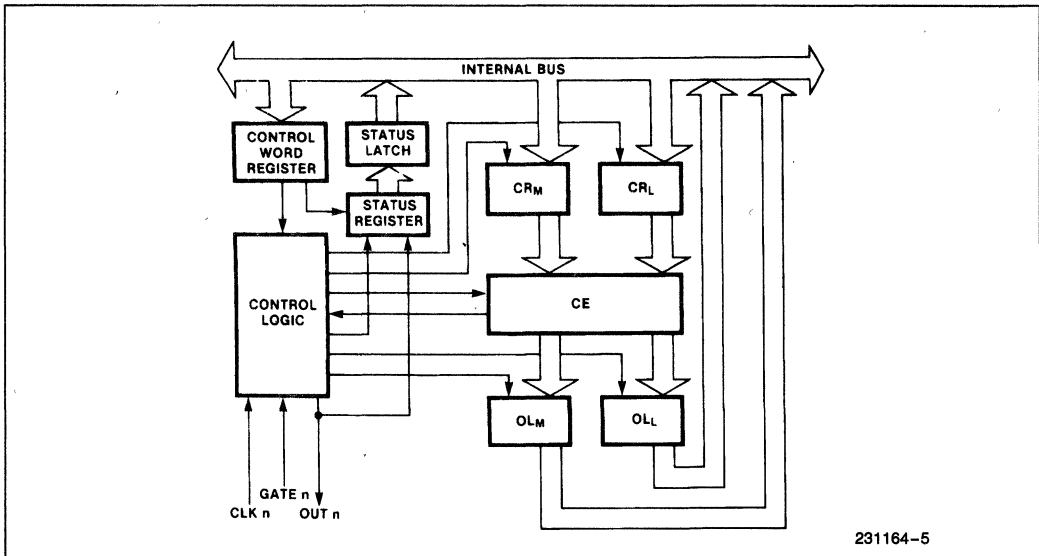


Figure 5. Internal Block Diagram of a Counter

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

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

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

8254 SYSTEM INTERFACE

The 8254 is a component of the Intel Microcomputer Systems and interfaces in the same manner as all

other peripherals of the family. It is treated by the system's software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A_0, A_1 connect to the A_0, A_1 address bus signals of the CPU. The CS can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

OPERATIONAL DESCRIPTION

General

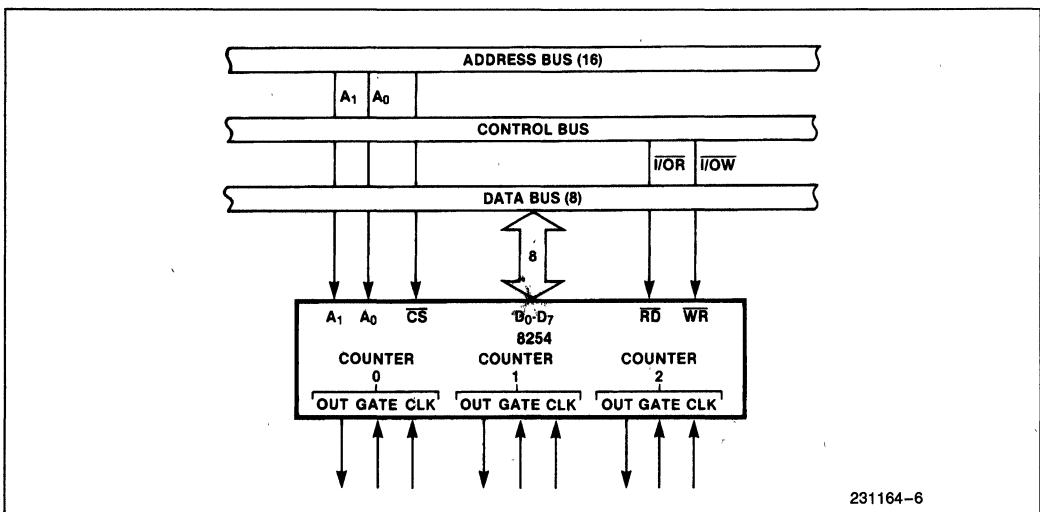
After power-up, the state of the 8254 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Programming the 8254

Counters are programmed by writing a Control Word and then an initial count.

The Control Words are written into the Control Word Register, which is selected when $A_1, A_0 = 11$. The Control Word itself specifies which Counter is being programmed.



231164-6

Figure 6. 8254 System Interface

Control Word Format

$A_1, A_0 = 11$ $\overline{CS} = 0$ $\overline{RD} = 1$ $\overline{WR} = 0$

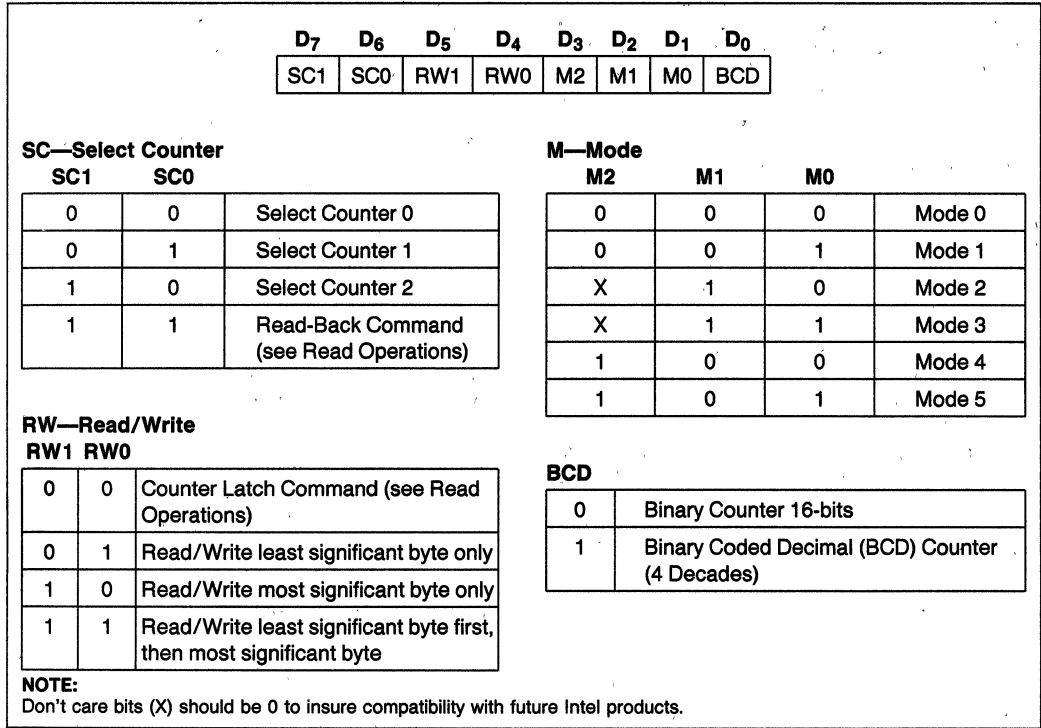


Figure 7. Control Word Format

By contrast, initial counts are written into the Counters, not the Control Word Register. The A_1, A_0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

Write Operations

The programming procedure for the 8254 is very flexible. Only two conventions need to be remembered:

- 1) For each Counter, the Control Word must be written before the initial count is written.
- 2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A_1, A_0 inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions in Figure 7 is acceptable.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.



If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

	A₁	A₀		A₁	A₀
Control Word—Counter 0	1	1	Control Word—Counter 2	1	1
LSB of count—Counter 0	0	0	Control Word—Counter 1	1	1
MSB of count—Counter 0	0	0	Control Word—Counter 0	1	1
Control Word—Counter 1	1	1	LSB of count—Counter 2	1	0
LSB of count—Counter 1	0	1	MSB of count—Counter 2	1	0
MSB of count—Counter 1	0	1	LSB of count—Counter 1	0	1
Control Word—Counter 2	1	1	MSB of count—Counter 1	0	1
LSB of count—Counter 2	1	0	LSB of count—Counter 0	0	0
MSB of count—Counter 2	1	0	MSB of count—Counter 0	0	0
	A₁	A₀		A₁	A₀
Control Word—Counter 0	1	1	Control Word—Counter 1	1	1
Control Word—Counter 1	1	1	Control Word—Counter 0	1	1
Control Word—Counter 2	1	1	LSB of count—Counter 1	0	1
LSB of count—Counter 2	1	0	Control Word—Counter 2	1	1
LSB of count—Counter 1	0	1	LSB of count—Counter 0	0	0
LSB of count—Counter 0	0	0	MSB of count—Counter 1	0	1
MSB of count—Counter 0	0	0	LSB of count—Counter 2	1	0
MSB of count—Counter 1	0	1	MSB of count—Counter 0	0	0
MSB of count—Counter 2	1	0	MSB of count—Counter 2	1	0

NOTE:
In all four examples, all Counters are programmed to read/write two-byte counts. These are only four of many possible programming sequences.

Figure 8. A Few Possible Programming Sequences

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 8254.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command. Each is explained below. The first method is to perform a simple read operation. To read the Counter, which is selected with the A₁, A₀ inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result.

COUNTER LATCH COMMAND

The second method uses the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when A₁, A₀ = 11. Also like a Control Word, the SC₀, SC₁ bits select one of the three Counters, but two other bits, D₅ and D₄, distinguish this command from a Control Word.

A₁, A₀ = 11; CS = 0; RD = 1; WR = 0

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
SC ₁	SC ₀	0	0	X	X	X	X

SC₁, SC₀—specify counter to be latched

SC ₁	SC ₀	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D₅, D₄—00 designates Counter Latch Command

X—don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 9. Counter Latching Command Format

The selected Counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 8254 is that reads and writes of the same Counter may be interleaved; for example, if the Counter is programmed for two byte counts, the following sequence is valid.

- 1) Read least significant byte.
- 2) Write new least significant byte.
- 3) Read most significant byte.
- 4) Write new most significant byte.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

READ-BACK COMMAND

The third method uses the Read-Back Command. This command allows the user to check the count value, programmed Mode, and current states of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 10. The command applies to the counters selected by setting their corresponding bits D3, D2, D1 = 1.

A0, A1 = 11		CS = 0		RD = 1		WR = 0	
D7	D6	D5	D4	D3	D2	D1	D0
1	1	COUNT	STATUS	CNT 2	CNT 1	CNT 0	0

D5: 0 = Latch count of selected counter(s)
 D4: 0 = Latch status of selected counters(s)
 D3: 1 = Select Counter 2
 D2: 1 = Select Counter 1
 D1: 1 = Select Counter 0
 D0: Reserved for future expansion; Must be 0

Figure 10. Read-Back Command Format

The read-back command may be used to latch multiple counter output latches (OL) by setting the COUNT bit D5 = 0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). The counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting STATUS bit D4 = 0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Figure 11. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

D7	D6	D5	D4	D3	D2	D1	D0
Output	Null Count	RW1	RW0	M2	M1	M0	BCD

D7 1 = OUT Pin is 1
 0 = OUT Pin is 0
 D6 1 = Null Count
 0 = Count available for reading
 D5-D0 Counter programmed mode (see Figure 7)

Figure 11. Status Byte

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Figure 12.

This Action	Causes
A. Write to the control word register;(1)	Null Count = 1
B. Write to the count register (CR);(2)	Null Count = 1
C. New Count is loaded into CE (CR → CE);	Null Count = 0

NOTE:

- Only the counter specified by the control word will have its Null Count set to 1. Null count bits of other counters are unaffected.
- If the counter is programmed for two-byte counts (least significant byte then most significant byte) Null Count goes to 1 when the second byte is written.

Figure 12. Null Count Operation

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both

COUNT and STATUS bits D5,D4 = 0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 13.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

CS	RD	WR	A ₁	A ₀	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

Figure 14. Read/Write Operations Summary

Command								Description	Result
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of Counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2, 1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

Figure 13. Read-Back Command Example

Mode Definitions

The following are defined for use in describing the operation of the 8254.

- CLK Pulse: a rising edge, then a falling edge, in that order, of a Counter's CLK input.
- Trigger: a rising edge of a Counter's GATE input.
- Counter loading: the transfer of a count from the CR to the CE (refer to the "Functional Description")

MODE 0: INTERRUPT ON TERMINAL COUNT

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until $N + 1$ CLK pulses after the initial count is written.

If a new count is written to the Counter, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- 1) Writing the first byte disables counting. OUT is set low immediately (no clock pulse required)
- 2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go high until $N + 1$ CLK pulses after the new count of N is written.

If an initial count is written while GATE = 0, it will still be loaded on the next CLK pulse. When GATE goes high, OUT will go high N CLK pulses later; no CLK pulse is needed to load the Counter as this has already been done.

MODE 1: HARDWARE RETRIGGERABLE ONE-SHOT

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero.

OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

MODE 2: RATE GENERATOR

This Mode functions like a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

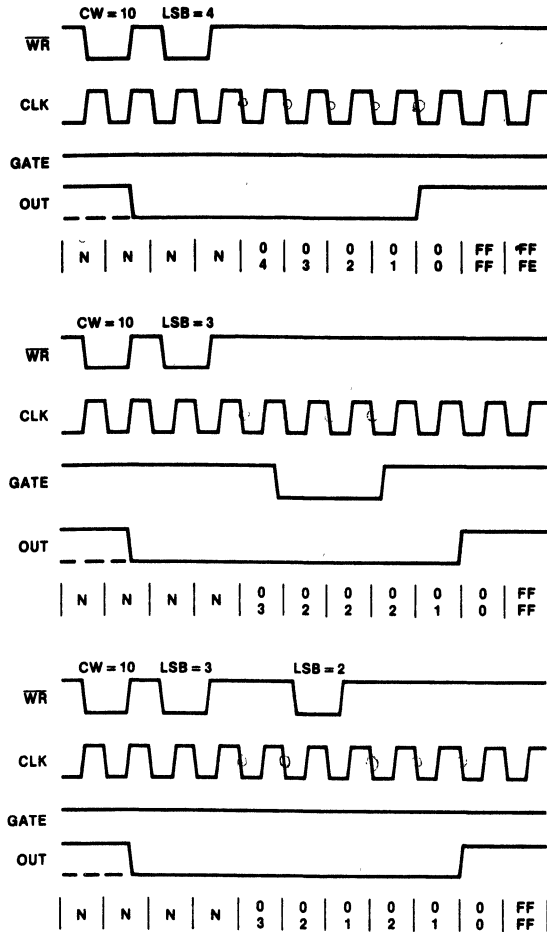
GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK Pulses after the initial count is written. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current period, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current counting cycle. In mode 2, a COUNT of 1 is illegal.

MODE 3: SQUARE WAVE MODE

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the



231164-7

NOTE:

The following conventions apply to all mode timing diagrams:

1. Counters are programmed for binary (not BCD) counting and for reading/writing least significant byte (LSB) only.
 2. The counter is always selected (CS always low).
 3. CW stands for "Control Word"; CW = 10 means a control word of 10 HEX is written to the counter.
 4. LSB stands for "Least Significant Byte" of count.
 5. Numbers below diagrams are count values. The lower number is the least significant byte. The upper number is the most significant byte. Since the counter is programmed to read/write LSB only, the most significant byte cannot be read. N stands for an undefined count.
- Vertical lines show transitions between count values.

Figure 15. Mode 0

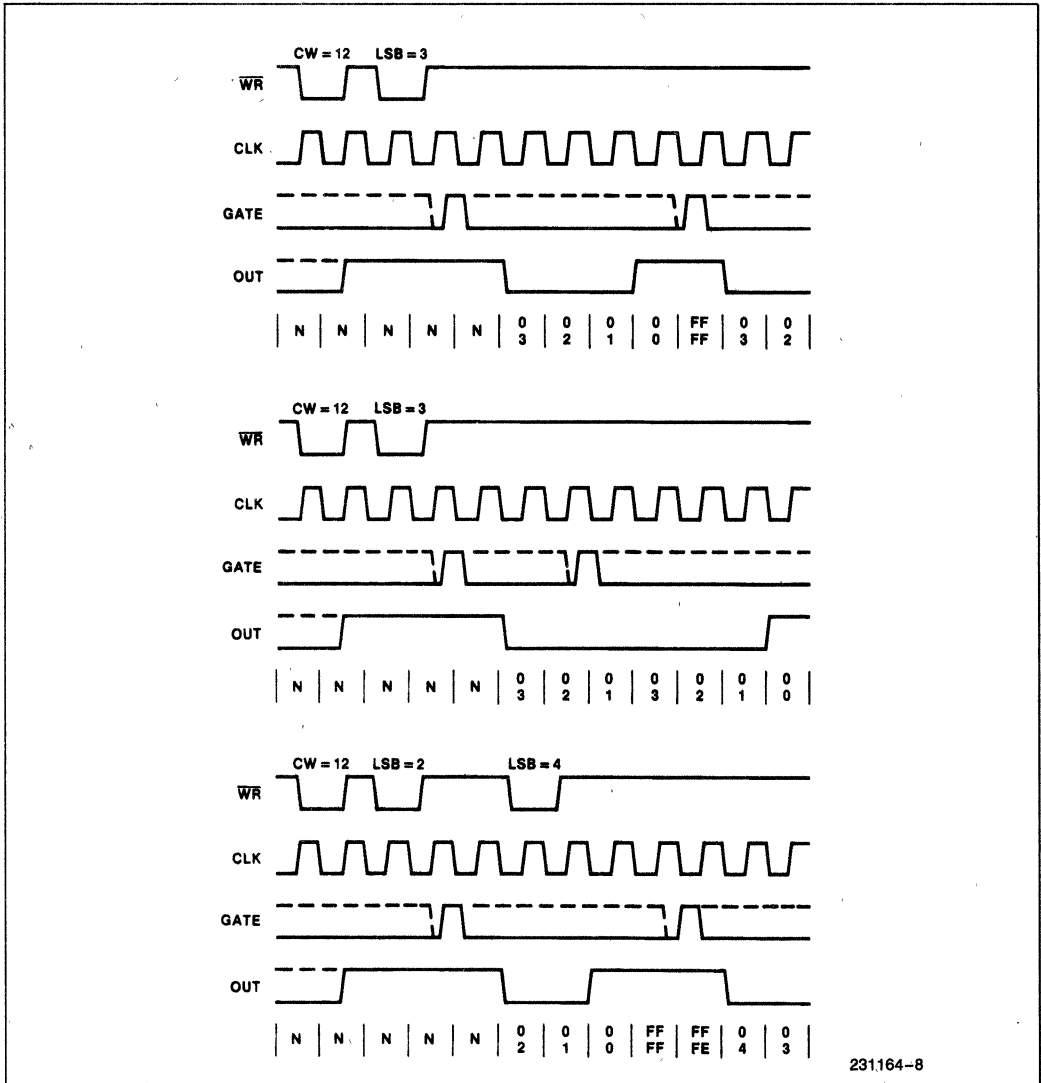


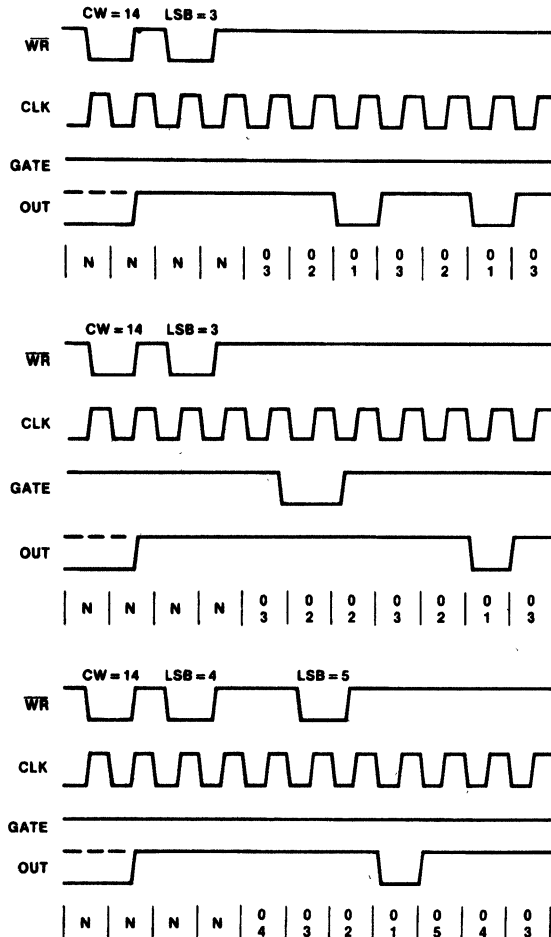
Figure 16. Mode 1

initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the



231164-9

NOTE:

A GATE transition should not occur one clock prior to terminal count.

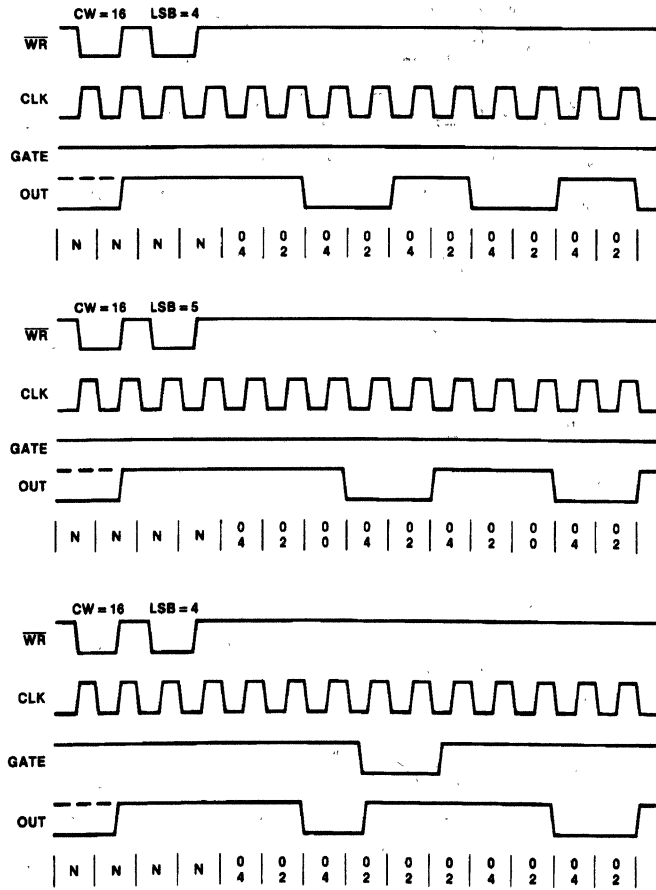
Figure 17. Mode 2

new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse *after* the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter is reloaded with the initial count minus one. The above process is repeated indefinitely. So for odd counts, OUT will be high for $(N + 1)/2$ counts and low for $(N - 1)/2$ counts.



231164-10

NOTE:
A GATE transition should not occur one clock prior to terminal count.

Figure 18. Mode 3

MODE 4: SOFTWARE TRIGGERED STROBE

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

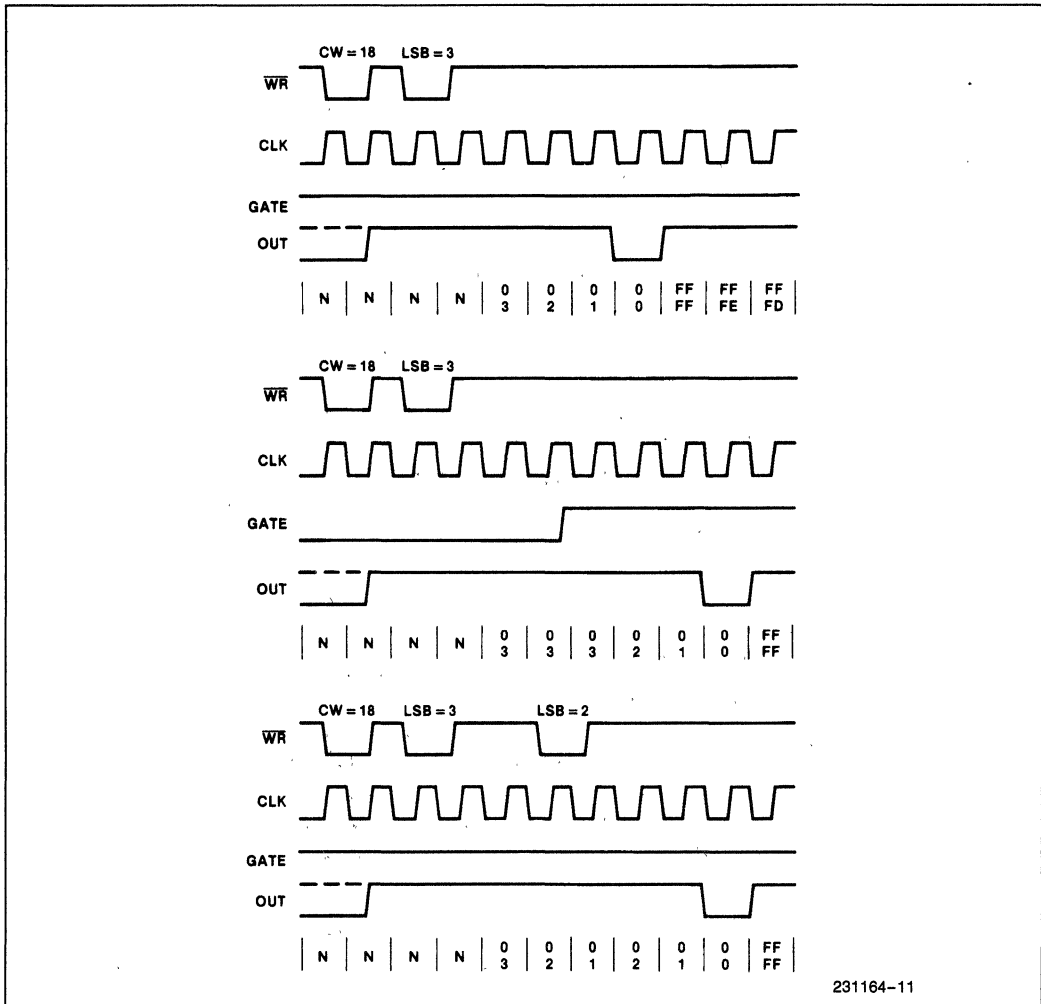
After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an

initial count of N, OUT does not strobe low until N + 1 CLK pulses after the initial count is written.

If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- 1) Writing the first byte has no effect on counting.
- 2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be "retriggered" by software. OUT strobes low N + 1 CLK pulses after the new count of N is written.



231164-11

Figure 19. Mode 4

MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low for N + 1 CLK pulses after any trigger. GATE has no effect on OUT.

If a new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.

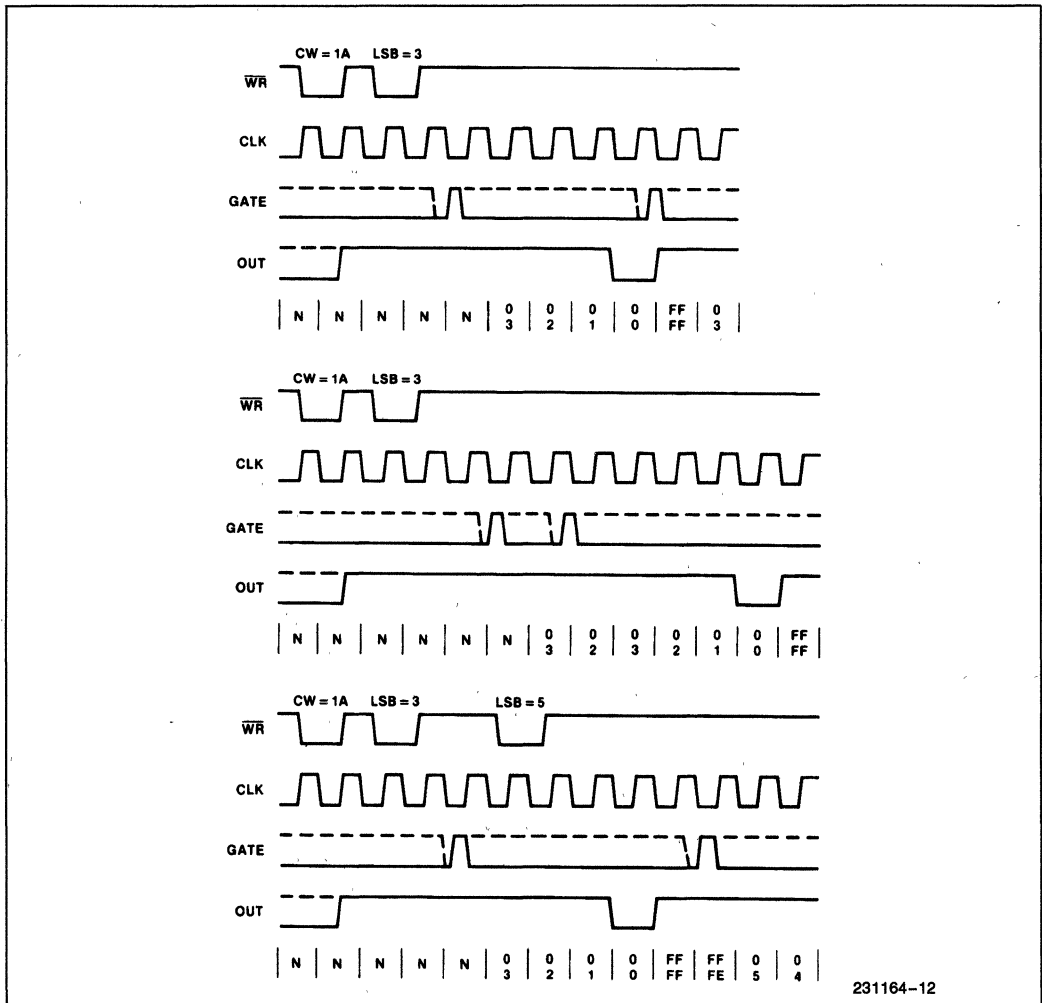


Figure 20. Mode 5

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables Counting	— —	Enables Counting
1	— —	1) Initiates Counting 2) Resets Output after Next Clock	— —
2	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
3	1) Disables Counting 2) Sets Output Immediately High	Initiates Counting	Enables Counting
4	Disables Counting	— —	Enables Counting
5	— —	Initiates Counting	— —

Figure 21. Gate Pin Operations Summary

Mode	Min Count	Max Count
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

NOTE:
0 is equivalent to 2¹⁶ for binary counting and 10⁴ for BCD counting.

Figure 22. Minimum and Maximum Initial Counts

Operation Common to All Modes

PROGRAMMING

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. In Modes 1, 2, 3, and 5 the GATE input is rising-edge sensitive. In these Modes, a rising edge of GATE (trigger) sets an edge-sensitive flip-flop in the Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs—a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive. In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following \overline{WR} of a new count value.

COUNTER

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to 2¹⁶ for binary counting and 10⁴ for BCD counting.

The Counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5 the Counter "wraps around" to the highest count, either FFFF hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic; the Counter reloads itself with the initial count and continues counting from there.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage on Any Pin with
 Respect to Ground -0.5V to +7V
 Power Dissipation 1W

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

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.5V$	V	
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.0\text{ mA}$
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -400\ \mu\text{A}$
I_{IL}	Input Load Current		± 10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		± 10	μA	$V_{OUT} = V_{CC}$ to 0.45V
I_{CC}	V_{CC} Supply Current		170	mA	

CAPACITANCE $T_A = 25^\circ\text{C}$, $V_{CC} = \text{GND} = 0V$

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	$f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to V_{SS}

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = 5V \pm 10\%$, $\text{GND} = 0V$
Bus Parameters(1)
READ CYCLE

Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{AR}	Address Stable Before $\overline{RD} \downarrow$	45		45		30		ns
t_{SR}	\overline{CS} Stable Before $\overline{RD} \downarrow$	0		0		0		ns
t_{RA}	Address Hold Time After $\overline{RD} \uparrow$	0		0		0		ns
t_{RR}	\overline{RD} Pulse Width	150		150		95		ns
t_{RD}	Data Delay from $\overline{RD} \downarrow$		120		120		85	ns
t_{AD}	Data Delay from Address		220		220		185	ns
t_{DF}	$\overline{RD} \uparrow$ to Data Floating	5	90	5	90	5	65	ns
t_{RV}	Command Recovery Time	200		200		165		ns

NOTE:

1. AC timings measured at $V_{OH} = 2.0V$, $V_{OL} = 0.8V$.

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 10\%$, $\text{GND} = 0\text{V}$ (Continued)

WRITE CYCLE

Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{AW}	Address Stable Before $\overline{\text{WR}} \downarrow$	0		0		0		ns
t_{SW}	$\overline{\text{CS}}$ Stable Before $\overline{\text{WR}} \downarrow$	0		0		0		ns
t_{WA}	Address Hold Time After $\overline{\text{WR}} \downarrow$	0		0		0		ns
t_{WW}	$\overline{\text{WR}}$ Pulse Width	150		150		95		ns
t_{DW}	Data Setup Time Before $\overline{\text{WR}} \uparrow$	120		120		95		ns
t_{WD}	Data Hold Time After $\overline{\text{WR}} \uparrow$	0		0		0		ns
t_{RV}	Command Recovery Time	200		200		165		ns

CLOCK AND GATE

Symbol	Parameter	8254-5		8254		8254-2		Unit
		Min	Max	Min	Max	Min	Max	
t_{CLK}	Clock Period	200	DC	125	DC	100	DC	ns
t_{PWH}	High Pulse Width	60 ⁽³⁾		60 ⁽³⁾		30 ⁽³⁾		ns
t_{PWL}	Low Pulse Width	60 ⁽³⁾		60 ⁽³⁾		50 ⁽³⁾		ns
t_R	Clock Rise Time		25		25		25	ns
t_F	Clock Fall Time		25		25		25	ns
t_{GW}	Gate Width High	50		50		50		ns
t_{GL}	Gate Width Low	50		50		50		ns
t_{GS}	Gate Setup Time to CLK \uparrow	50		50		40		ns
t_{GH}	Gate Setup Time After CLK \uparrow	50 ⁽²⁾		50 ⁽²⁾		50 ⁽²⁾		ns
t_{OD}	Output Delay from CLK \downarrow		150		150		100	ns
t_{ODG}	Output Delay from Gate \downarrow		120		120		100	ns
t_{WC}	CLK Delay for Loading \downarrow	0	55	0	55	0	55	ns
t_{WG}	Gate Delay for Sampling	-5	50	-5	50	-5	40	ns
t_{WO}	OUT Delay from Mode Write		260		260		240	ns
t_{CL}	CLK Set Up for Count Latch	-40	45	-40	45	-40	40	ns

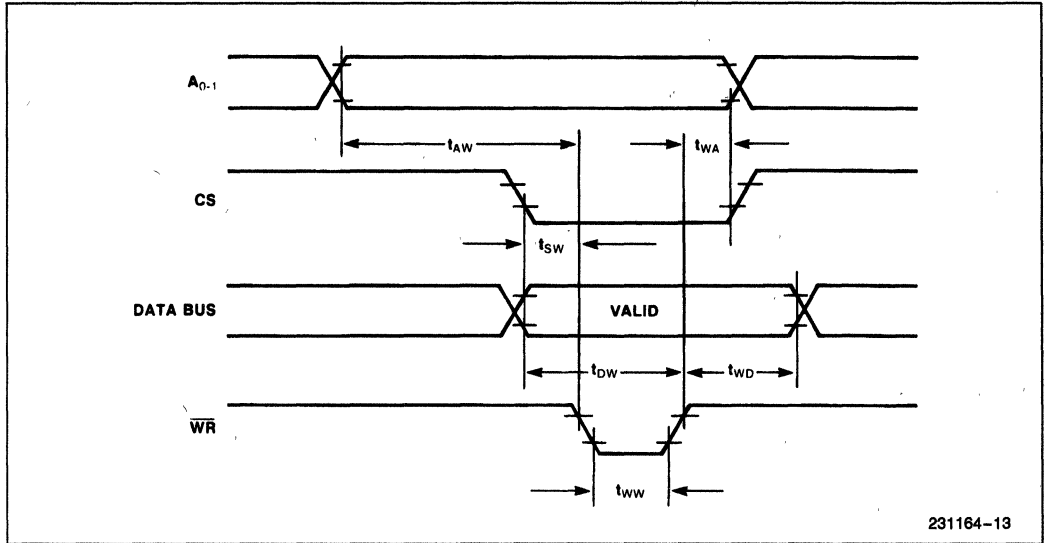
NOTES:

2. In Modes 1 and 5 triggers are sampled on each rising clock edge. A second trigger within 120 ns (70 ns for the 8254-2) of the rising clock edge may not be detected.

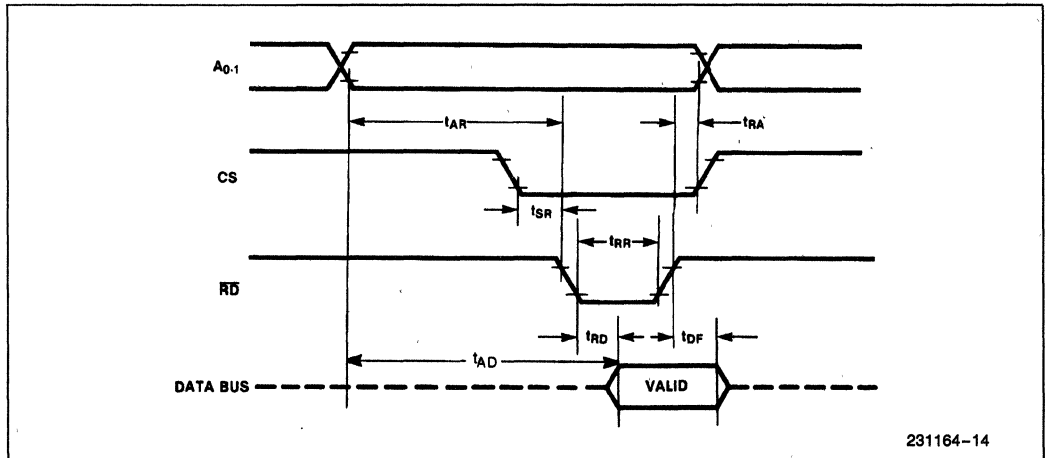
3. Low-going glitches that violate t_{PWH} , t_{PWL} may cause errors requiring counter reprogramming.

WAVEFORMS

WRITE

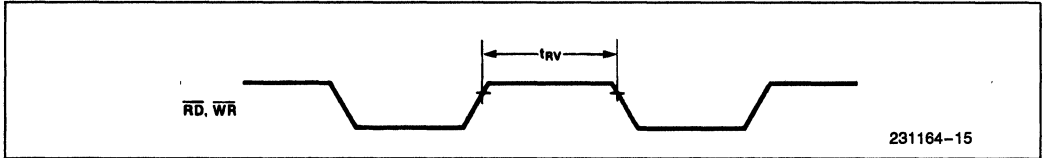


READ

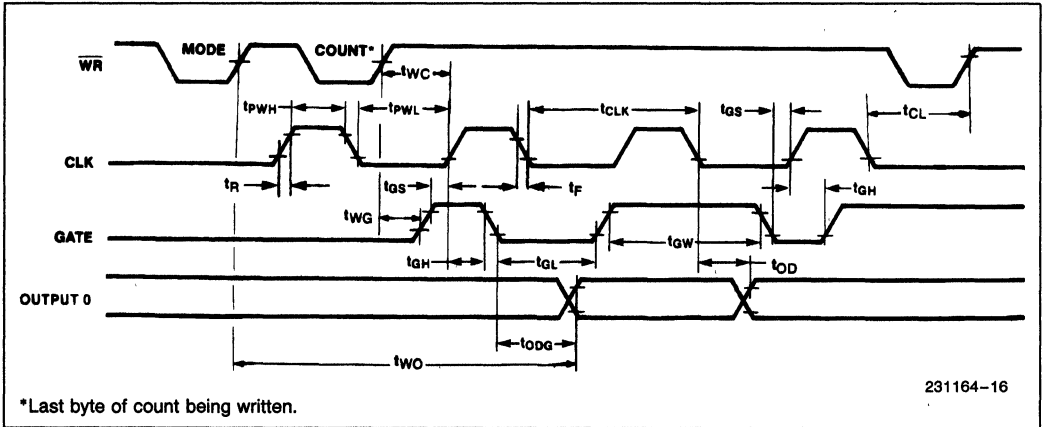


WAVEFORMS (Continued)

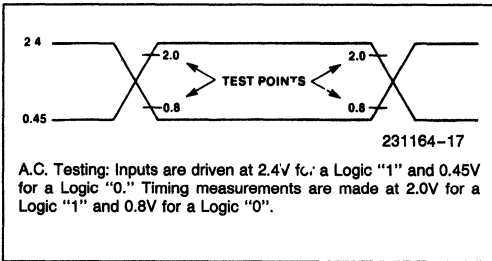
RECOVERY



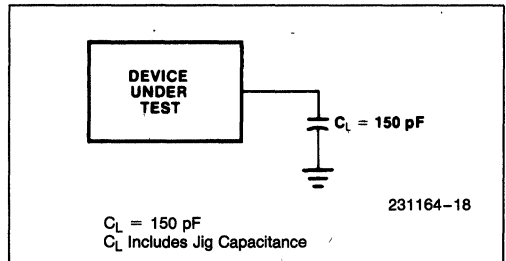
CLOCK AND GATE



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT





82C54 CHMOS PROGRAMMABLE INTERVAL TIMER

- Compatible with all Intel and most other microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- Handles Inputs from DC to 8 MHz — 10 MHz for 82C54-2
- Available in EXPRESS — Standard Temperature Range — Extended Temperature Range
- Three independent 16-bit counters
- Low Power CHMOS — $I_{CC} = 10 \text{ mA @ 8 MHz Count frequency}$
- Completely TTL Compatible
- Six Programmable Counter Modes
- Binary or BCD counting
- Status Read Back Command
- Available in 24-Pin DIP and 28-Pin PLCC

The Intel 82C54 is a high-performance, CHMOS version of the industry standard 8254 counter/timer which is designed to solve the timing control problems common in microcomputer system design. It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz. All modes are software programmable. The 82C54 is pin compatible with the HMOS 8254, and is a superset of the 8253.

Six programmable timer modes allow the 82C54 to be used as an event counter, elapsed time indicator, programmable one-shot, and in many other applications.

The 82C54 is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent HMOS product. The 82C54 is available in 24-pin DIP and 28-pin plastic leaded chip carrier (PLCC) packages.

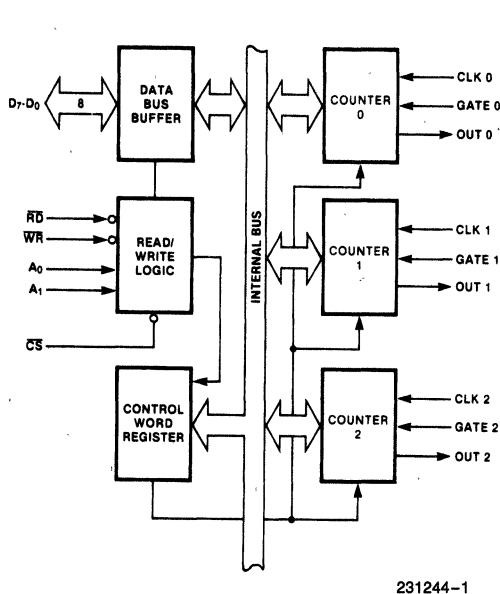
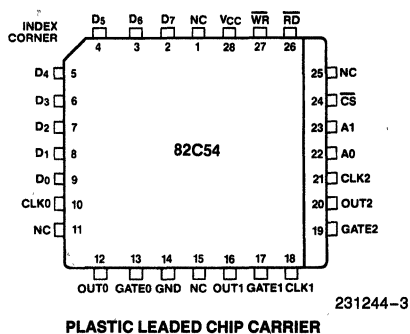
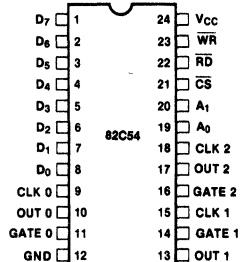


Figure 1. 82C54 Block Diagram



PLASTIC LEADED CHIP CARRIER



Diagrams are for pin reference only. Package sizes are not to scale.

Figure 2. 82C54 Pinout

Table 1. Pin Description

Symbol	Pin Number		Type	Function		
	DIP	PLCC				
D ₇ -D ₀	1-8	2-9	I/O	Data: Bidirectional tri-state data bus lines, connected to system data bus.		
CLK 0	9	10	I	Clock 0: Clock input of Counter 0.		
OUT 0	10	12	O	Output 0: Output of Counter 0.		
GATE 0	11	13	I	Gate 0: Gate input of Counter 0.		
GND	12	14		Ground: Power supply connection.		
OUT 1	13	16	O	Out 1: Output of Counter 1.		
GATE 1	14	17	I	Gate 1: Gate input of Counter 1.		
CLK 1	15	18	I	Clock 1: Clock input of Counter 1.		
GATE 2	16	19	I	Gate 2: Gate input of Counter 2.		
OUT 2	17	20	O	Out 2: Output of Counter 2.		
CLK 2	18	21	I	Clock 2: Clock input of Counter 2.		
A ₁ , A ₀	20-19	23-22	I	Address: Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus.		
					A₁	A₀
				0	0	Counter 0
				0	1	Counter 1
1	0	Counter 2				
1	1	Control Word Register				
\overline{CS}	21	24	I	Chip Select: A low on this input enables the 82C54 to respond to \overline{RD} and \overline{WR} signals. \overline{RD} and \overline{WR} are ignored otherwise.		
\overline{RD}	22	26	I	Read Control: This input is low during CPU read operations.		
\overline{WR}	23	27	I	Write Control: This input is low during CPU write operations.		
V _{CC}	24	28		Power: +5V power supply connection.		
NC		1, 11, 15, 25		No Connect		

FUNCTIONAL DESCRIPTION

General

The 82C54 is a programmable interval timer/counter designed for use with Intel microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 82C54 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 82C54 to match his requirements and programs one of the counters for the de-

sired delay. After the desired delay, the 82C54 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

Some of the other counter/timer functions common to microcomputers which can be implemented with the 82C54 are:

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

Block Diagram

DATA BUS BUFFER

This 3-state, bi-directional, 8-bit buffer is used to interface the 82C54 to the system bus (see Figure 3).

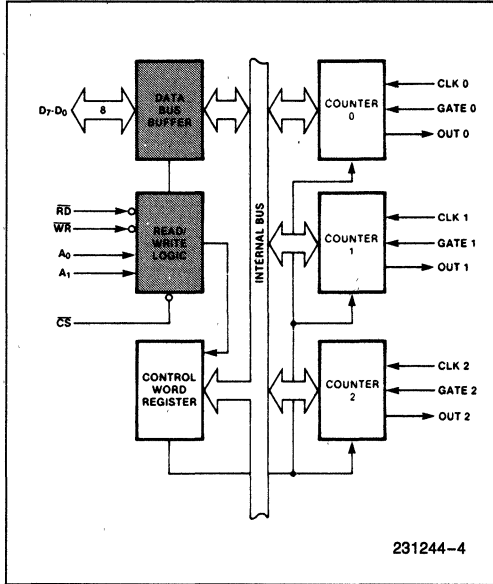


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

READ/WRITE LOGIC

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 82C54. A₁ and A₀ select one of the three counters or the Control Word Register to be read from/written into. A "low" on the RD input tells the 82C54 that the CPU is reading one of the counters. A "low" on the WR input tells the 82C54 that the CPU is writing either a Control Word or an initial count. Both RD and WR are qualified by CS; RD and WR are ignored unless the 82C54 has been selected by holding CS low.

CONTROL WORD REGISTER

The Control Word Register (see Figure 4) is selected by the Read/Write Logic when A₁, A₀ = 11. If the CPU then does a write operation to the 82C54, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

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

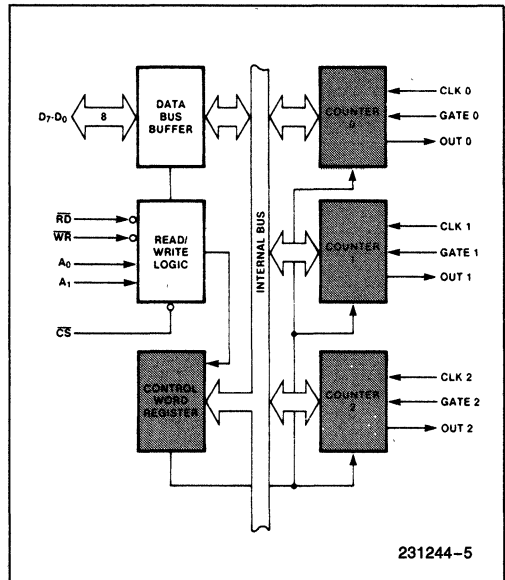


Figure 4. Block Diagram Showing Control Word Register and Counter Functions

COUNTER 0, COUNTER 1, COUNTER 2

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

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

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

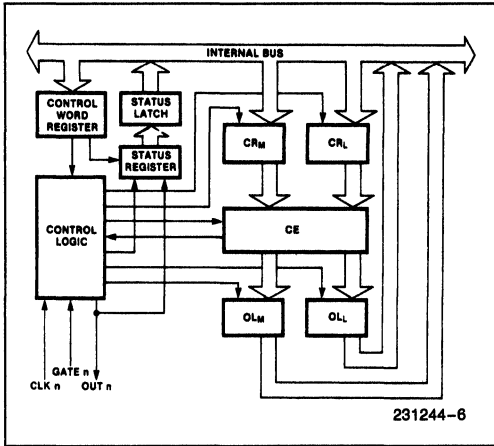


Figure 5. Internal Block Diagram of a Counter

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

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

OL_M and OL_L are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L stand for "Most significant byte" and "Least significant byte" respectively. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 82C54, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CR_M and CR_L (for "Count Register"). Both are normally referred to as one unit and called just CR. When a new count is written to the Counter, the count is

stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CR_M and CR_L are cleared when the Counter is programmed. In this way, if the Counter has been programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

The Control Logic is also shown in the diagram. CLK_n, GATE_n, and OUT_n are all connected to the outside world through the Control Logic.

82C54 SYSTEM INTERFACE

The 82C54 is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

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

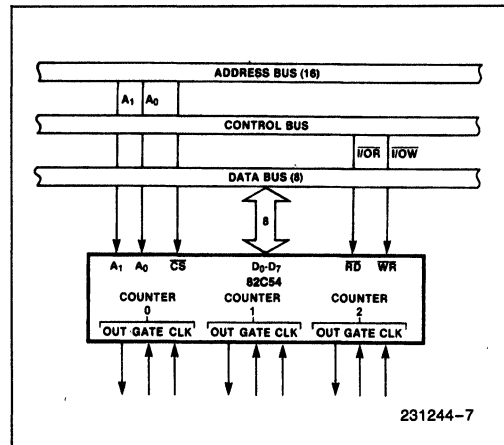


Figure 6. 82C54 System Interface

OPERATIONAL DESCRIPTION

General

After power-up, the state of the 82C54 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

Programming the 82C54

Counters are programmed by writing a Control Word and then an initial count. The control word format is shown in Figure 7.

All Control Words are written into the Control Word Register, which is selected when $A_1, A_0 = 11$. The Control Word itself specifies which Counter is being programmed.

By contrast, initial counts are written into the Counters, not the Control Word Register. The A_1, A_0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

Control Word Format

$A_1, A_0 = 11 \quad \overline{CS} = 0 \quad \overline{RD} = 1 \quad \overline{WR} = 0$

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

M — MODE:

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

RW — Read/Write:

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

BCD:

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

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 7. Control Word Format

Write Operations

The programming procedure for the 82C54 is very flexible. Only two conventions need to be remembered:

- 1) For each Counter, the Control Word must be written before the initial count is written.
- 2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A₁, A₀ inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), no special in-

struction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

Control Word — Counter 0	A ₁	A ₀	Control Word — Counter 2	A ₁	A ₀
LSB of count — Counter 0	1	1	Control Word — Counter 1	1	1
MSB of count — Counter 0	0	0	Control Word — Counter 0	1	1
Control Word — Counter 1	1	1	LSB of count — Counter 2	1	0
LSB of count — Counter 1	0	1	MSB of count — Counter 2	1	0
MSB of count — Counter 1	0	1	LSB of count — Counter 1	0	1
Control Word — Counter 2	1	1	MSB of count — Counter 1	0	1
LSB of count — Counter 2	1	0	LSB of count — Counter 0	0	0
MSB of count — Counter 2	1	0	MSB of count — Counter 0	0	0
	A ₁	A ₀		A ₁	A ₀
Control Word — Counter 0	1	1	Control Word — Counter 1	1	1
Control Word — Counter 1	1	1	Control Word — Counter 0	1	1
Control Word — Counter 2	1	1	LSB of count — Counter 1	0	1
LSB of count — Counter 2	1	0	Control Word — Counter 2	1	1
LSB of count — Counter 1	0	1	LSB of count — Counter 0	0	0
LSB of count — Counter 0	0	0	MSB of count — Counter 1	0	1
MSB of count — Counter 0	0	0	LSB of count — Counter 2	1	0
MSB of count — Counter 1	0	1	MSB of count — Counter 0	0	0
MSB of count — Counter 2	1	0	MSB of count — Counter 2	1	0

NOTE:
In all four examples, all counters are programmed to read/write two-byte counts. These are only four of many possible programming sequences.

Figure 8. A Few Possible Programming Sequences

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 82C54.

There are three possible methods for reading the counters: a simple read operation, the Counter

Latch Command, and the Read-Back Command. Each is explained below. The first method is to perform a simple read operation. To read the Counter, which is selected with the A₁, A₀ inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result.

COUNTER LATCH COMMAND

The second method uses the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when $A_1, A_0 = 11$. Also like a Control Word, the SC_0, SC_1 bits select one of the three Counters, but two other bits, D_5 and D_4 , distinguish this command from a Control Word.

$A_1, A_0 = 11; \overline{CS} = 0; \overline{RD} = 1; \overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 - specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

$D_5, D_4 = 00$ designates Counter Latch Command

X - don't care

NOTE:
Don't care bits (X) should be 0 to insure compatibility with future Intel products.

Figure 9. Counter Latching Command Format

The selected Counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or pro-

gramming operations of other Counters may be inserted between them.

Another feature of the 82C54 is that reads and writes of the same Counter may be interleaved; for example, if the Counter is programmed for two byte counts, the following sequence is valid.

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a Counter is programmed to read/write two-byte counts, the following precaution applies; A program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

READ-BACK COMMAND

The third method uses the Read-Back command. This command allows the user to check the count value, programmed Mode, and current state of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 10. The command applies to the counters selected by setting their corresponding bits $D_3, D_2, D_1 = 1$.

$A_0, A_1 = 11 \quad \overline{CS} = 0 \quad \overline{RD} = 1 \quad \overline{WR} = 0$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	COUNT	STATUS	CNT 2	CNT 1	CNT 0	0

$D_5: 0 =$ Latch count of selected counter(s)
 $D_4: 0 =$ Latch status of selected counter(s)
 $D_3: 1 =$ Select counter 2
 $D_2: 1 =$ Select counter 1
 $D_1: 1 =$ Select counter 0
 $D_0:$ Reserved for future expansion; must be 0

Figure 10. Read-Back Command Format

The read-back command may be used to latch multiple counter output latches (OL) by setting the COUNT bit $D_5 = 0$ and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). That counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the

count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting STATUS bit D4=0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Figure 11. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

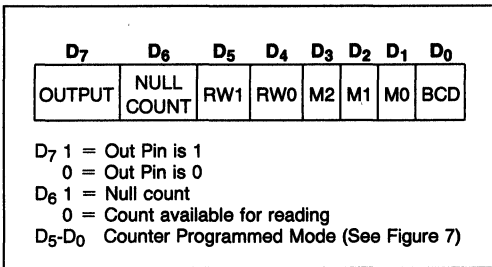


Figure 11. Status Byte

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Figure 12.

THIS ACTION:	CAUSES:
A. Write to the control word register;[1]	Null count = 1
B. Write to the count register (CR);[2]	Null count = 1
C. New count is loaded into CE (CR → CE);	Null count = 0

[1] Only the counter specified by the control word will have its null count set to 1. Null count bits of other counters are unaffected.
 [2] If the counter is programmed for two-byte counts (least significant byte then most significant byte) null count goes to 1 when the second byte is written.

Figure 12. Null Count Operation

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both COUNT and STATUS bits D5,D4=0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 13.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

Command								Description	Results
D7	D6	D5	D4	D3	D2	D1	D0		
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of Counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2, 1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

Figure 13. Read-Back Command Example

CS	RD	WR	A ₁	A ₀	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

Figure 14. Read/Write Operations Summary

Mode Definitions

The following are defined for use in describing the operation of the 82C54.

CLK PULSE: a rising edge, then a falling edge, in that order, of a Counter's CLK input.

TRIGGER: a rising edge of a Counter's GATE input.

COUNTER LOADING: the transfer of a count from the CR to the CE (refer to the "Functional Description")

MODE 0: INTERRUPT ON TERMINAL COUNT

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

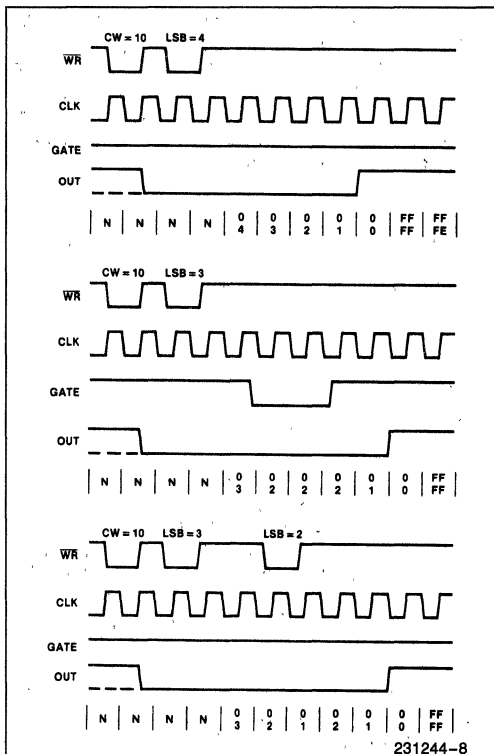
After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N + 1 CLK pulses after the initial count is written.

If a new count is written to the Counter, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- 1) Writing the first byte disables counting. OUT is set low immediately (no clock pulse required).
- 2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go high until N + 1 CLK pulses after the new count of N is written.

If an initial count is written while GATE = 0, it will still be loaded on the next CLK pulse. When GATE goes high, OUT will go high N CLK pulses later; no CLK pulse is needed to load the Counter as this has already been done.



NOTE:

The Following Conventions Apply To All Mode Timing Diagrams:

1. Counters are programmed for binary (not BCD) counting and for Reading/Writing least significant byte (LSB) only.
2. The counter is always selected (\overline{CS} always low).
3. CW stands for "Control Word"; CW = 10 means a control word of 10, hex is written to the counter.
4. LSB stands for "Least Significant Byte" of count.
5. Numbers below diagrams are count values. The lower number is the least significant byte. The upper number is the most significant byte. Since the counter is programmed to Read/Write LSB only, the most significant byte cannot be read. N stands for an undefined count. Vertical lines show transitions between count values.

Figure 15. Mode 0

MODE 1: HARDWARE RETRIGGERABLE ONE-SHOT

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the Counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

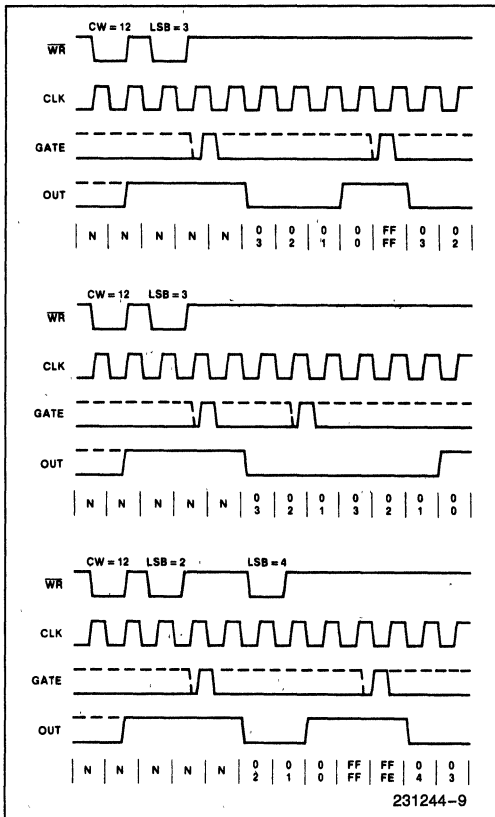


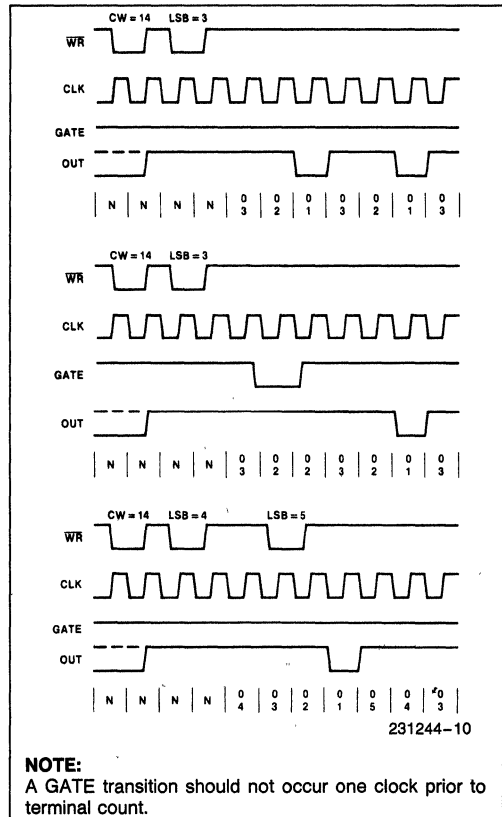
Figure 16. Mode 1

MODE 2: RATE GENERATOR

This Mode functions like a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK Pulses after the initial count is written. This allows the Counter to be synchronized by software also.



NOTE:

A GATE transition should not occur one clock prior to terminal count.

Figure 17. Mode 2

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current period, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current counting cycle. In mode 2, a COUNT of 1 is illegal.

MODE 3: SQUARE WAVE MODE

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse *after* the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter is reloaded with the initial count minus one. The above process is repeated indefinitely. So for odd counts,

OUT will be high for $(N + 1)/2$ counts and low for $(N - 1)/2$ counts.

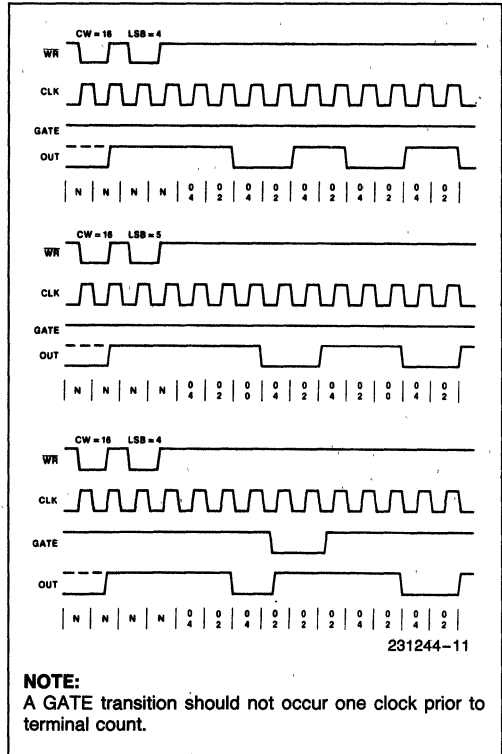


Figure 18. Mode 3

MODE 4: SOFTWARE TRIGGERED STROBE

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after the initial count is written.

If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- 1) Writing the first byte has no effect on counting.
- 2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be "retriggered" by software. OUT strobes low $N + 1$ CLK pulses after the new count of N is written.

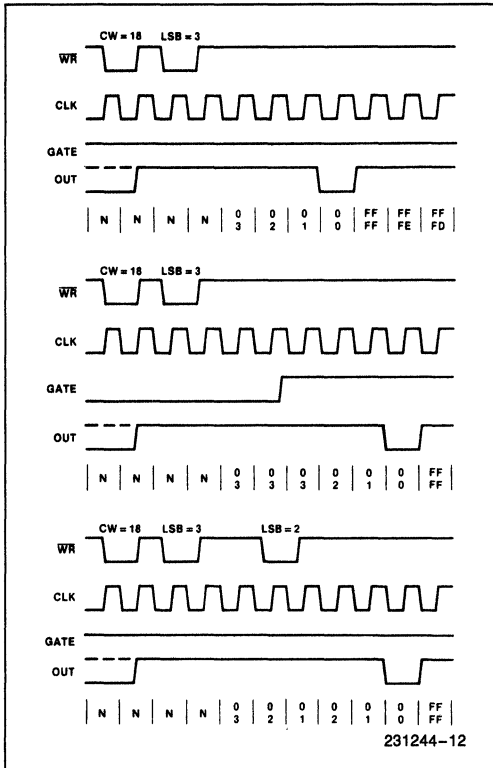


Figure 19. Mode 4

MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count, so for an initial count of N , OUT does not strobe low until $N + 1$ CLK pulses after a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low for $N + 1$ CLK pulses after any trigger. GATE has no effect on OUT.

If a new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.

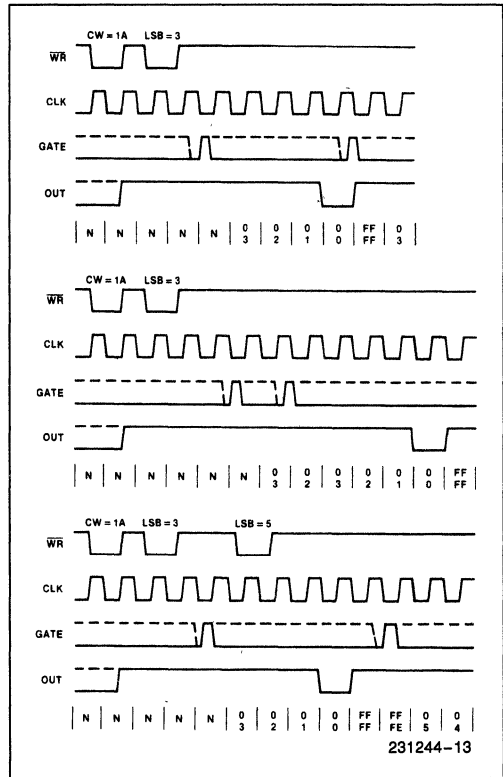


Figure 20. Mode 5

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

Figure 21. Gate Pin Operations Summary

MODE	MIN COUNT	MAX COUNT
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0

NOTE:
0 is equivalent to 2^{16} for binary counting and 10^4 for BCD counting

Figure 22. Minimum and Maximum Initial Counts

Operation Common to All Modes

Programming

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. In Modes 1, 2, 3, and 5 the GATE input is rising-edge sensitive. In these Modes, a rising edge of GATE (trigger) sets an edge-sensitive flip-flop in the Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs—a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive. In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following WR of a new count value.

COUNTER

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to 2^{16} for binary counting and 10^4 for BCD counting.

The Counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5 the Counter “wraps around” to the highest count, either FFFF hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic; the Counter reloads itself with the initial count and continues counting from there.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias.....0°C to 70°C
 Storage Temperature -65° to +150°C
 Supply Voltage -0.5 to +8.0V
 Operating Voltage +4V to +7V
 Voltage on any Input.....GND -2V to +6.5V
 Voltage on any Output ..GND-0.5V to V_{CC} + 0.5V
 Power Dissipation1 Watt

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

D.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, GND = 0V) (T_A = -40°C to +85°C for Extended Temperature)

Symbol	Parameter	Min	Max	Units	Test Conditions
V _{IL}	Input Low Voltage	-0.5	0.8	V	
V _{IH}	Input High Voltage	2.0	V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage		0.4	V	I _{OL} = 2.5 mA
V _{OH}	Output High Voltage	3.0 V _{CC} - 0.4		V V	I _{OH} = -2.5 mA I _{OH} = -100 μA
I _{IL}	Input Load Current		±20	μA	V _{IN} = V _{CC} to 0V
I _{OFL}	Output Float Leakage Current		±10	μA	V _{OUT} = V _{CC} to 0.45V
I _{CC}	V _{CC} Supply Current		20	mA	Clk Freq = 8MHz 82C54 10MHz 82C54-2
I _{CCSB}	V _{CC} Supply Current-Standby		10	μA	CLK Freq = DC CS = HIGH. All Inputs/Data Bus HIGH All Outputs Floating
I _{CCSB1}	V _{CC} Supply Current-Standby		150	μA	CLK Freq = DC CS = HIGH. All Other Inputs, Outputs, I/O Plus Floating

CAPACITANCE (T_A = 25°C, V_{CC} = GND = 0V)

Symbol	Parameter	Min	Max	Units	Test Conditions
C _{IN}	Input Capacitance		10	pF	f _c = 1 MHz Unmeasured pins returned to GND
C _{I/O}	I/O Capacitance		20	pF	
C _{OUT}	Output Capacitance		20	pF	

A.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, GND = 0V) (T_A = -40°C to +85°C for Extended Temperature)

BUS PARAMETERS (Note 1)

READ CYCLE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t _{AR}	Address Stable Before $\overline{RD} \downarrow$	45		30		ns
t _{SR}	\overline{CS} Stable Before $\overline{RD} \downarrow$	0		0		ns
t _{RA}	Address Hold Time After $\overline{RD} \downarrow$	0		0		ns
t _{RR}	\overline{RD} Pulse Width	150		95		ns
t _{RD}	Data Delay from $\overline{RD} \downarrow$		120		85	ns
t _{AD}	Data Delay from Address		220		185	ns
t _{DF}	$\overline{RD} \uparrow$ to Data Floating	5	90	5	65	ns
t _{RV}	Command Recovery Time	200		165		ns

NOTE:

1. $t_{\overline{RD}}$ timings measured at V_{OH} = 2.0V, V_{OL} = 0.8V.

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{AW}	Address Stable Before $\overline{WR} \downarrow$	0		0		ns
t_{SW}	\overline{CS} Stable Before $\overline{WR} \downarrow$	0		0		ns
t_{WA}	Address Hold Time After $\overline{WR} \uparrow$	0		0		ns
t_{WW}	\overline{WR} Pulse Width	150		95		ns
t_{DW}	Data Setup Time Before $\overline{WR} \uparrow$	120		95		ns
t_{WD}	Data Hold Time After $\overline{WR} \uparrow$	0		0		ns
t_{RV}	Command Recovery Time	200		165		ns

CLOCK AND GATE

Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{CLK}	Clock Period	125	DC	100	DC	ns
t_{PWH}	High Pulse Width	60 ⁽³⁾		30 ⁽³⁾		ns
t_{PWL}	Low Pulse Width	60 ⁽³⁾		50 ⁽³⁾		ns
T_R	Clock Rise Time		25		25	ns
t_F	Clock Fall Time		25		25	ns
t_{GW}	Gate Width High	50		50		ns
t_{GL}	Gate Width Low	50		50		ns
t_{GS}	Gate Setup Time to CLK \uparrow	50		40		ns
t_{GH}	Gate Hold Time After CLK \uparrow	50 ⁽²⁾		50 ⁽²⁾		ns
T_{OD}	Output Delay from CLK \downarrow		150		100	ns
t_{ODG}	Output Delay from Gate \downarrow		120		100	ns
t_{WC}	CLK Delay for Loading ⁽⁴⁾	0	55	0	55	ns
t_{WG}	Gate Delay for Sampling ⁽⁴⁾	-5	50	-5	40	ns
t_{WO}	OUT Delay from Mode Write		260		240	ns
t_{CL}	CLK Set Up for Count Latch	-40	45	-40	40	ns

NOTES:

2. In Modes 1 and 5 triggers are sampled on each rising clock edge. A second trigger within 120 ns (70 ns for the 82C54-2) of the rising clock edge may not be detected.

3. Low-going glitches that violate t_{PWH} , t_{PWL} may cause errors requiring counter reprogramming.

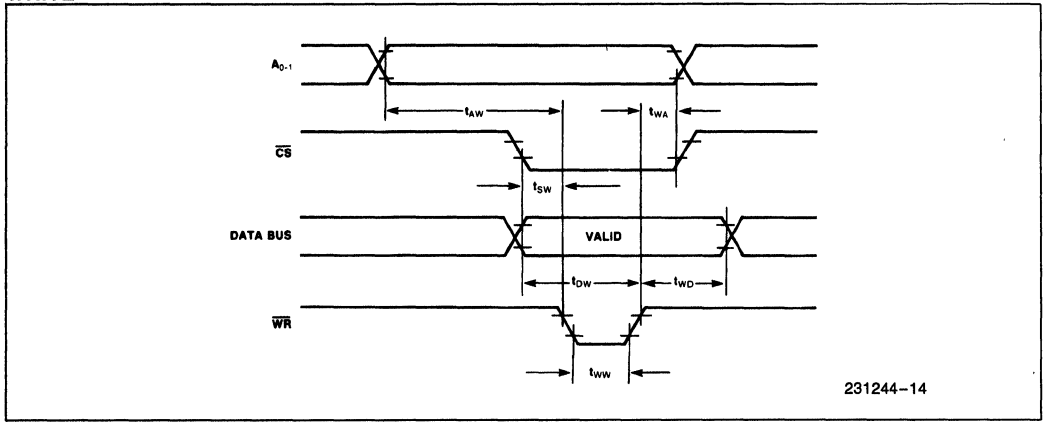
4. Except for Extended Temp., See Extended Temp. A.C. Characteristics below.

EXTENDED TEMPERATURE ($T_A = -40^\circ\text{C}$ to $+85^\circ\text{C}$ for Extended Temperature)

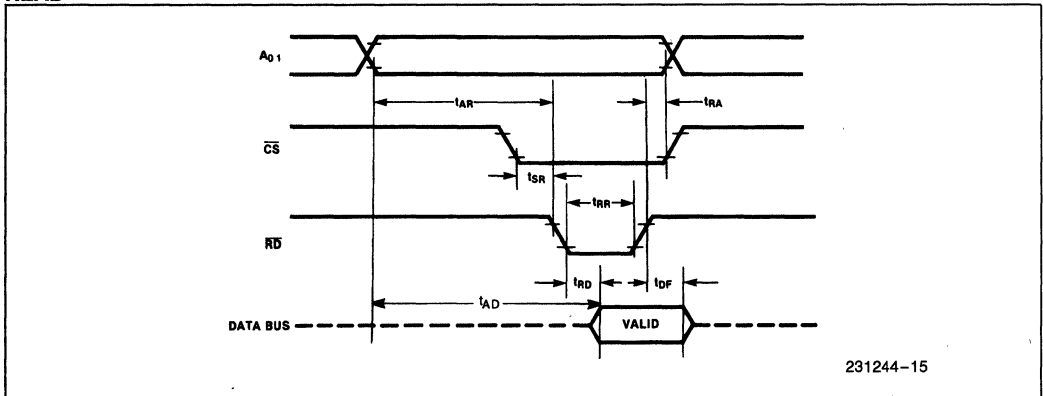
Symbol	Parameter	82C54		82C54-2		Units
		Min	Max	Min	Max	
t_{WC}	CLK Delay for Loading	-25	25	-25	25	ns
t_{WG}	Gate Delay for Sampling	-25	25	-25	25	ns

WAVEFORMS

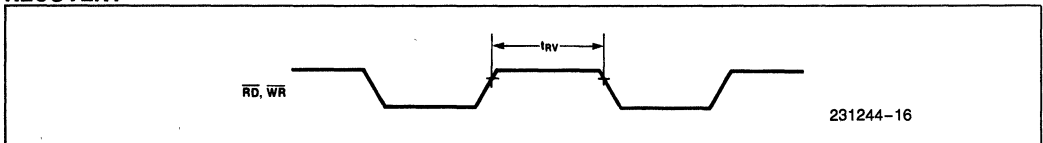
WRITE



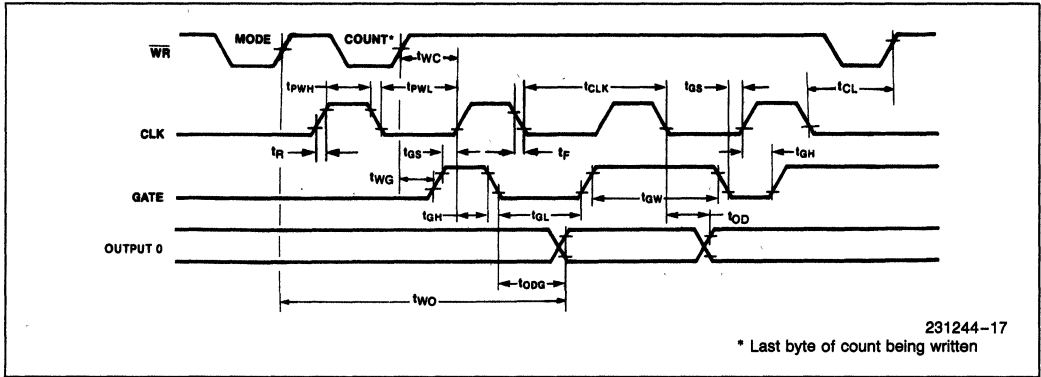
READ



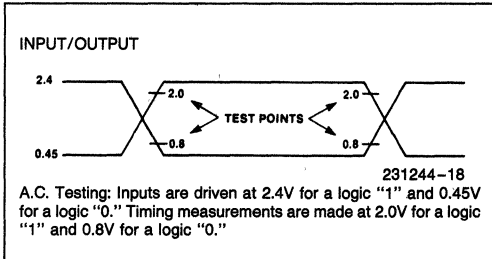
RECOVERY



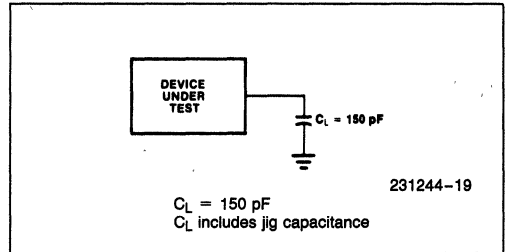
CLOCK AND GATE



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT





8255A/8255A-5 PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
 - 24 Programmable I/O Pins
 - Completely TTL Compatible
 - Fully Compatible with Intel Microprocessor Families
 - Improved Timing Characteristics
 - Direct Bit Set/Reset Capability Easing Control Application Interface
 - Reduces System Package Count
 - Improved DC Driving Capability
 - Available in EXPRESS
 - Standard Temperature Range
 - Extended Temperature Range
 - 40 Pin DIP Package or 44 Lead PLCC
- (See Intel Packaging: Order Number: 231369)

The Intel 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

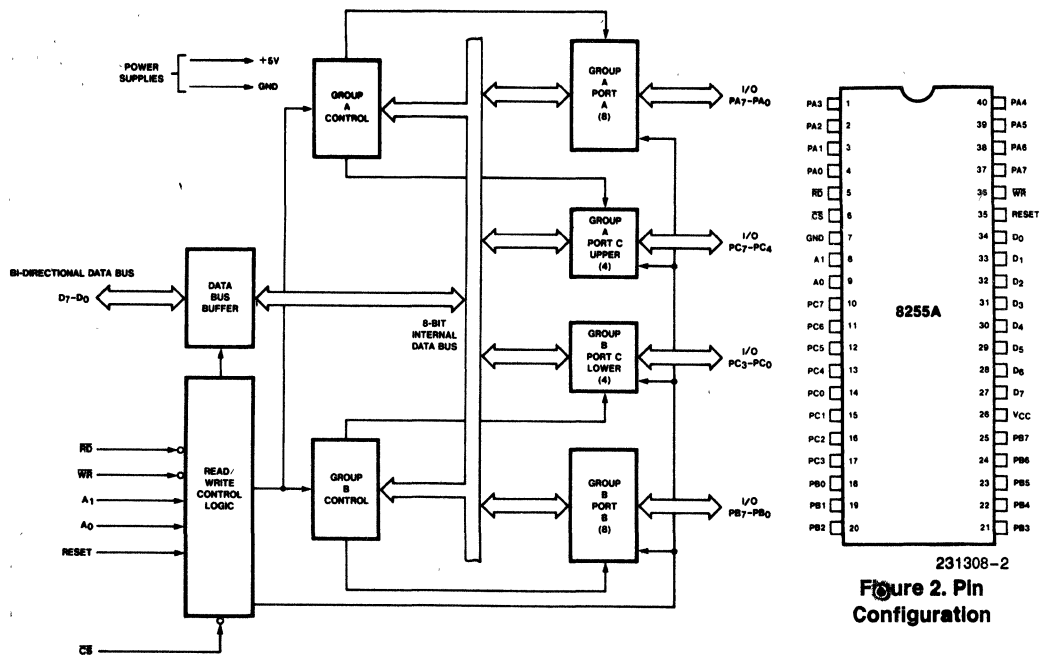


Figure 1. 8255A Block Diagram

Figure 2. Pin Configuration

8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the

CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

(CS)

Chip Select. A "low" on this input pin enables the communication between the 8255A and the CPU.

(RD)

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

(WR)

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A₀ and A₁)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A₀ and A₁).

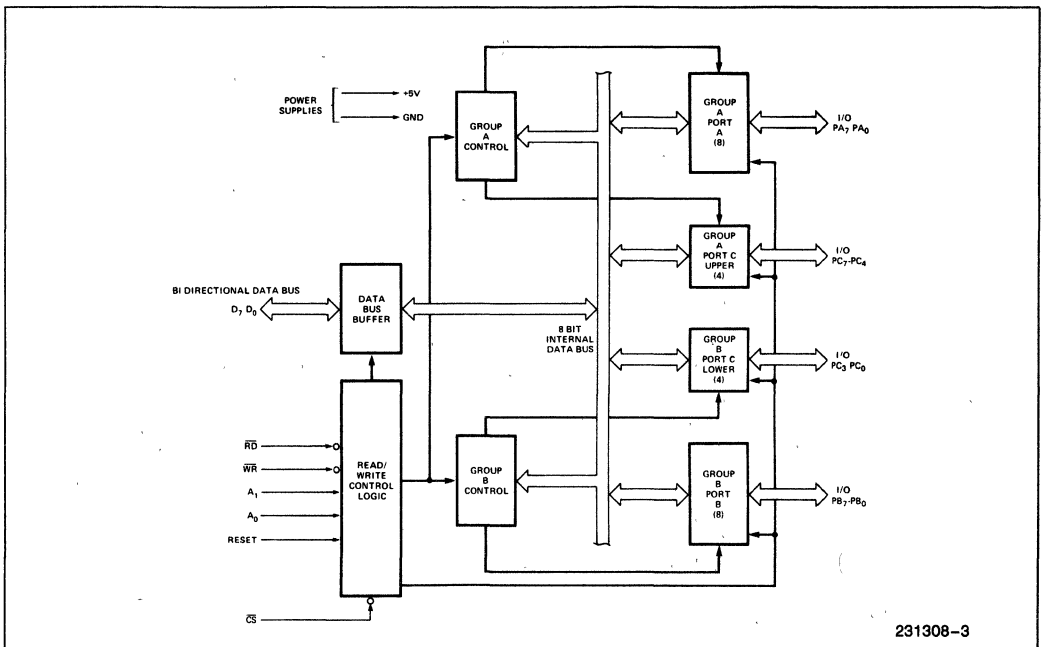


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

8255A BASIC OPERATION

A ₁	A ₀	\overline{RD}	\overline{WR}	\overline{CS}	Input Operation (READ)
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
					Output Operation (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					Disable Function
X	X	X	X	1	Data Bus → 3-State
1	1	0	1	0	Illegal Condition
X	X	1	1	0	Data Bus → 3-State

(RESET)

Reset. A "high" on this input clears the control register and all ports (A, B, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A—Port A and Port C upper (C7–C4)

Control Group B—Port B and Port C lower (C3–C0)

The Control Word Register can only be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

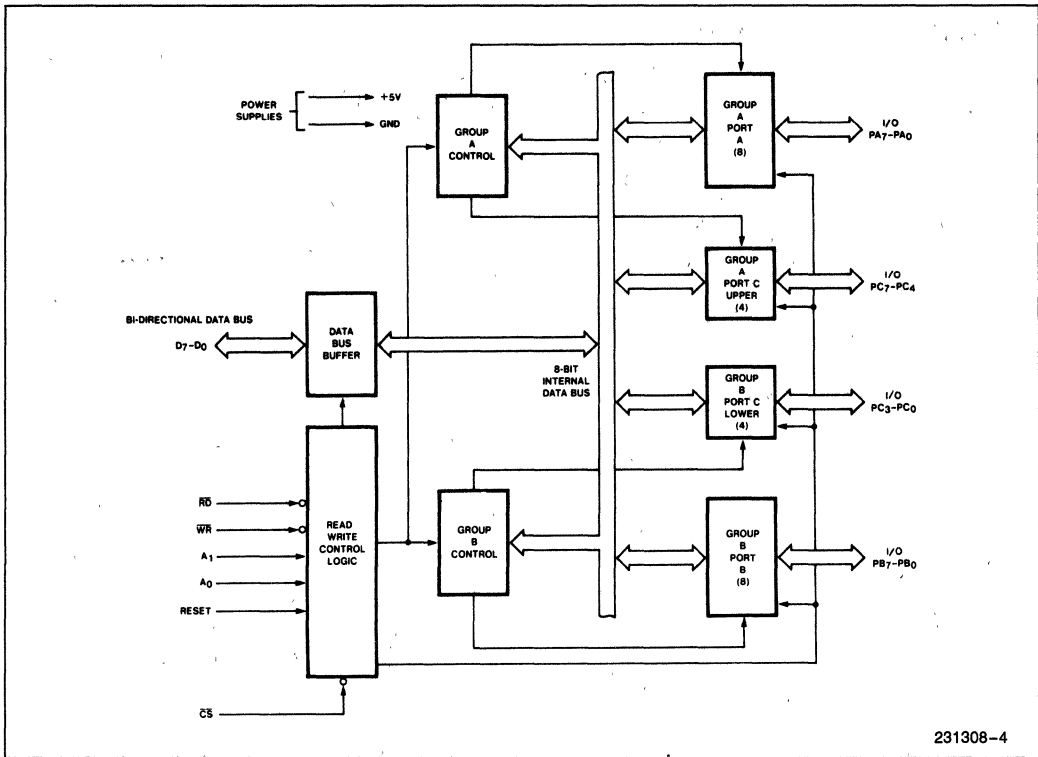
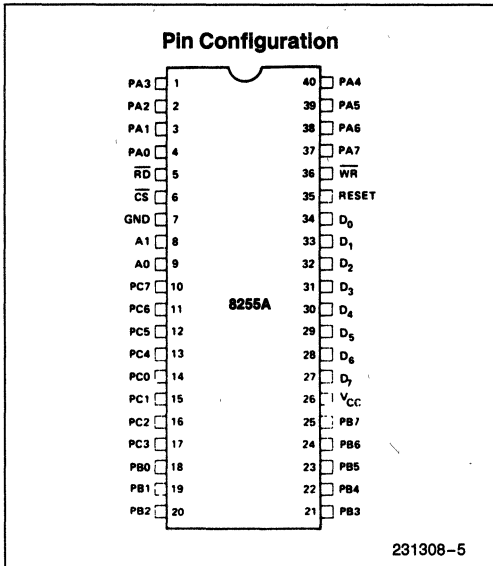


Figure 4. 8225A Block Diagram Showing Group A and Group B Control Functions



Pin Names

D ₇ -D ₀	Data Bus (Bi-Directional)
RESET	Reset Input
\overline{CS}	Chip Select
\overline{RD}	Read Input
\overline{WR}	Write Input
A ₀ , A ₁	Port Address
PA ₇ -PA ₀	Port A (BIT)
PB ₇ -PB ₀	Port B (BIT)
PC ₇ -PC ₀	Port C (BIT)
V _{CC}	+ 5 Volts
GND	0 Volts

8255A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0—Basic Input/Output

Mode 1—Strobed Input/Output

Mode 2—Bi-Directional Bus

When the reset input goes “high” all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be “tailored” to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

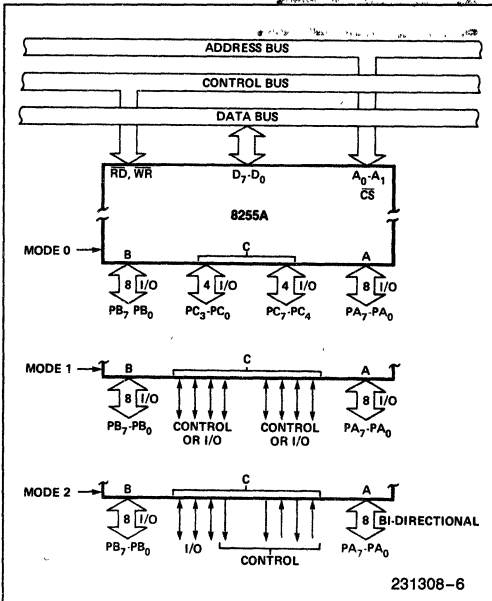


Figure 5. Basic Mode Definitions and Bus Interface

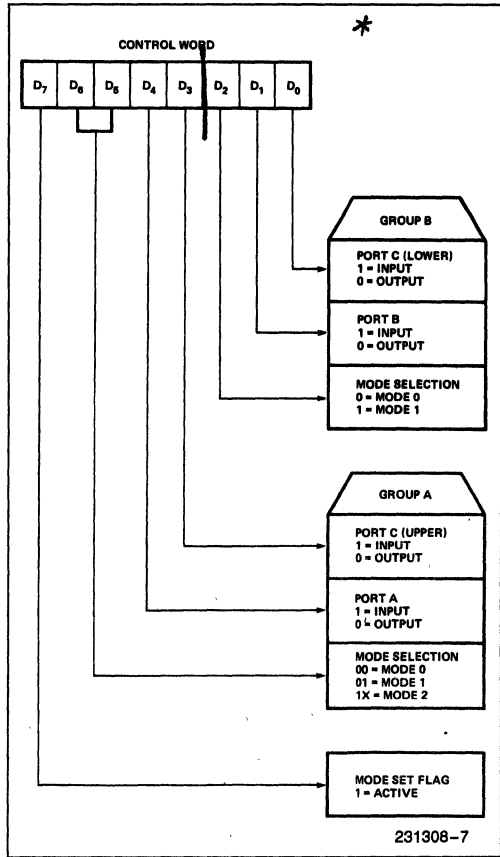


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTPUT instruction. This feature reduces software requirements in Control-based applications.

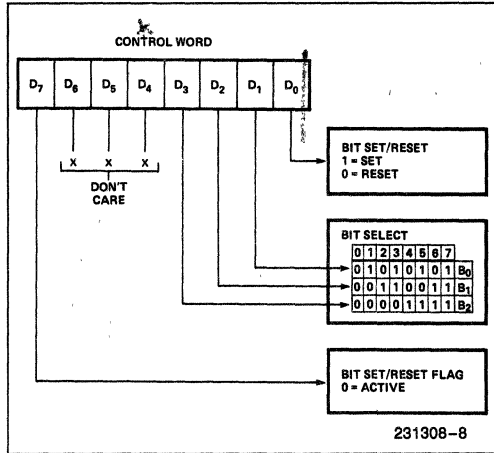


Figure 7. Bit Set/Reset Format

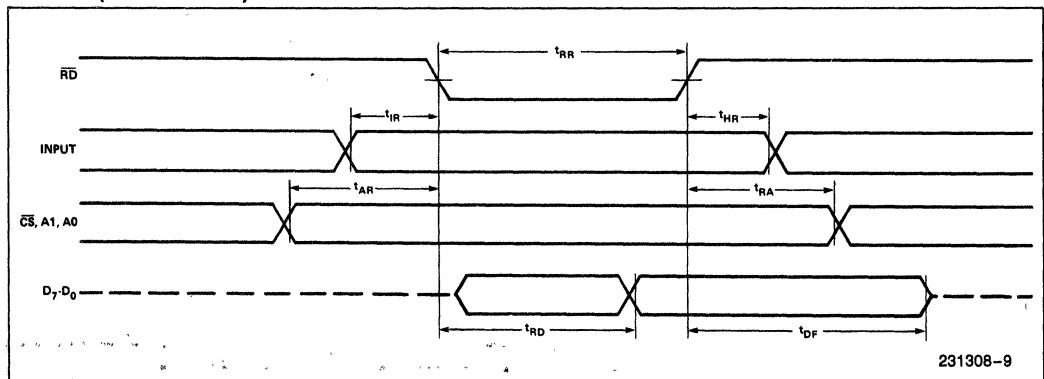
When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

INTEA
& INTEB

MODE 0 (BASIC INPUT)



231308-9

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is set—Interrupt enable

(BIT-RESET)—INTE is RESET—Interrupt disable

NOTE:

All Mask flip-flops are automatically reset during mode selection and device Reset.

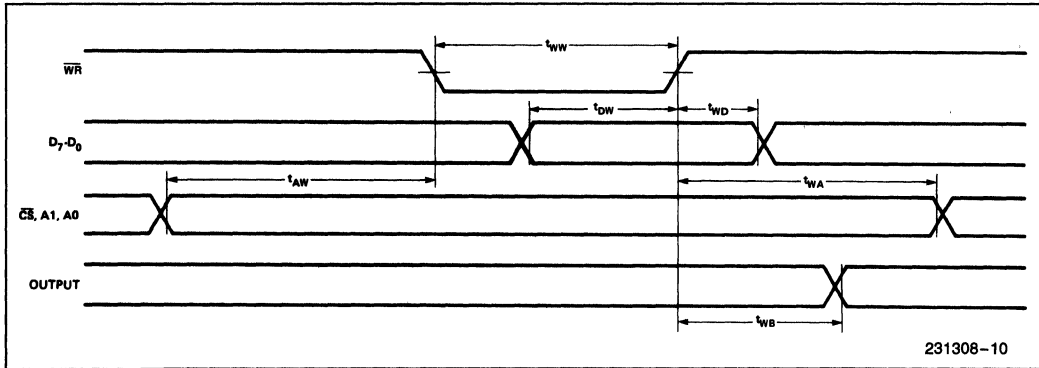
Operating Modes

MODE 0 (Basic Input/Output) This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC OUTPUT)

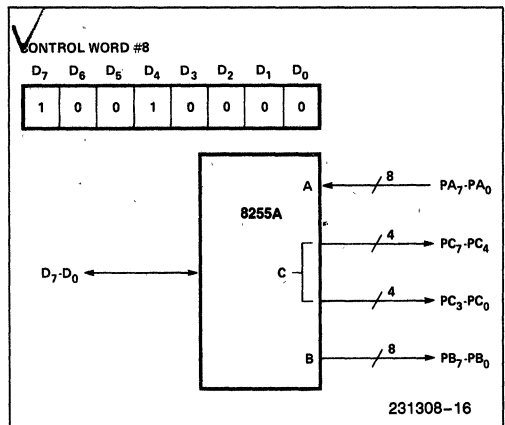
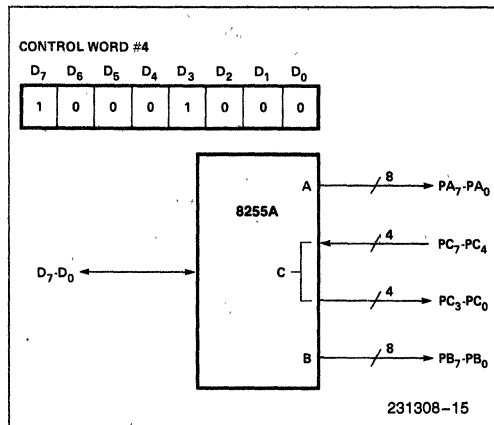
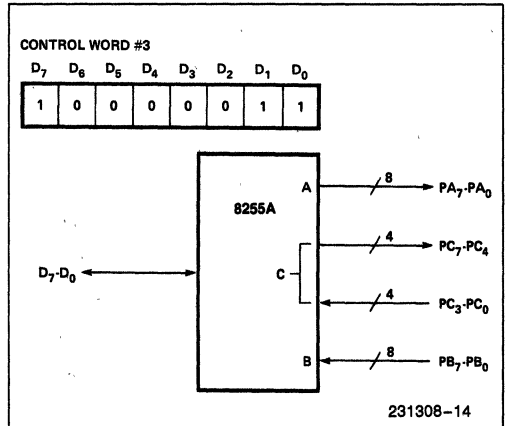
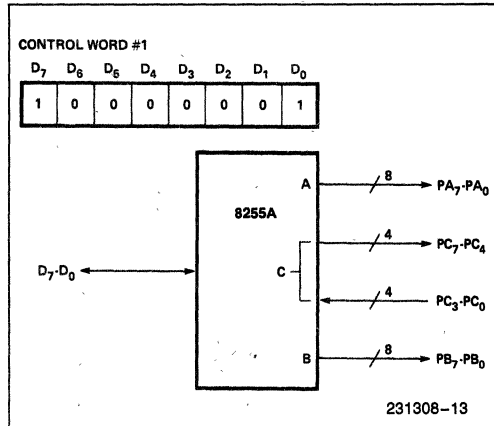
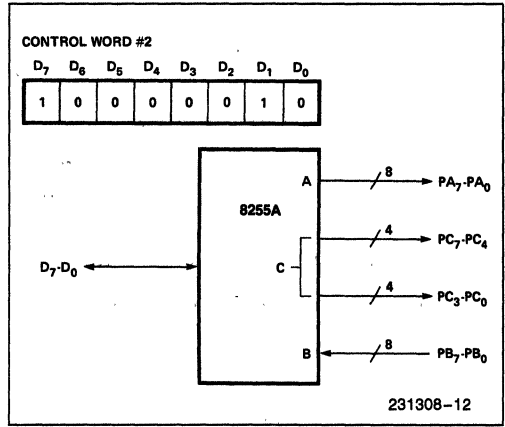
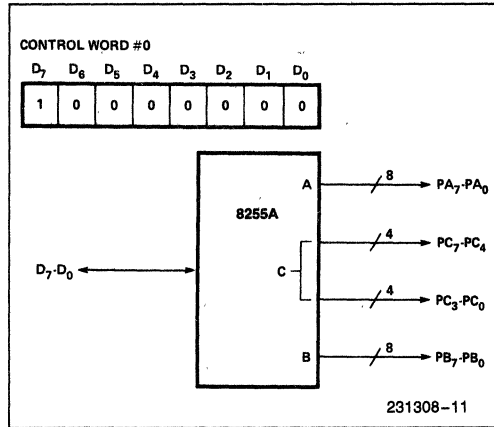


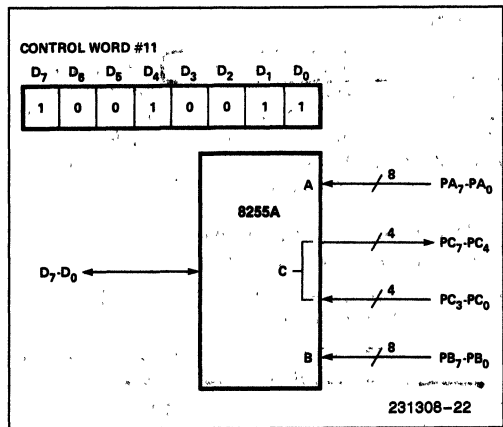
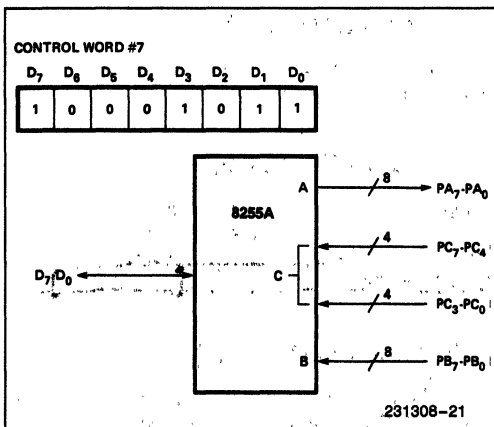
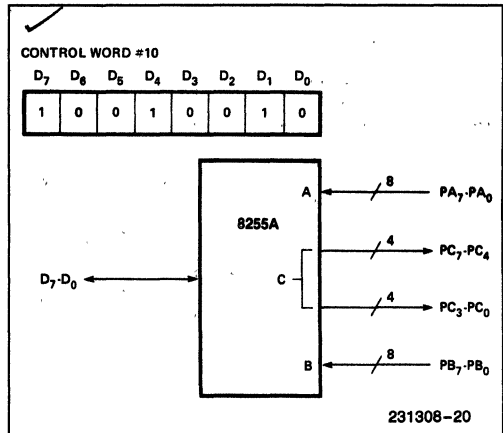
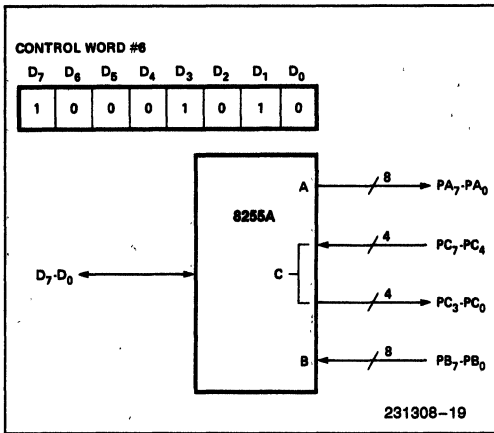
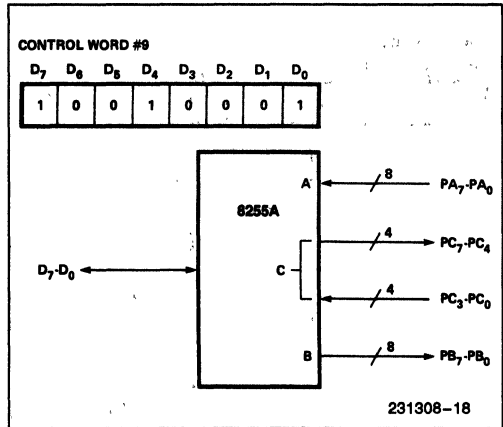
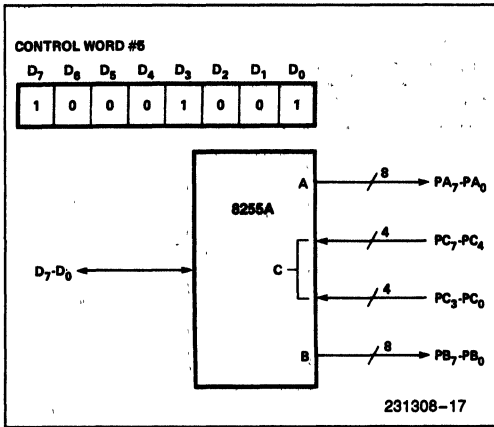
MODE 0 PORT DEFINITION

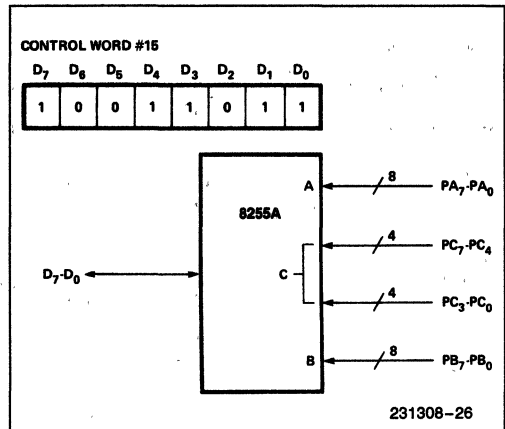
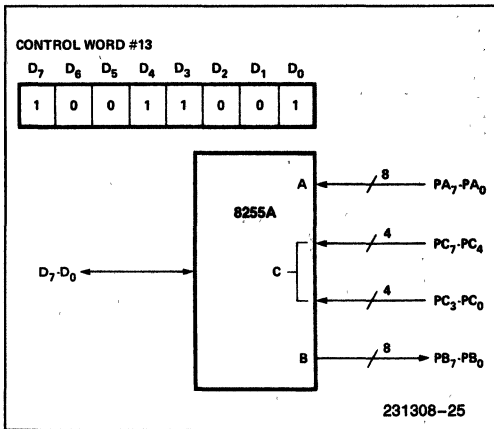
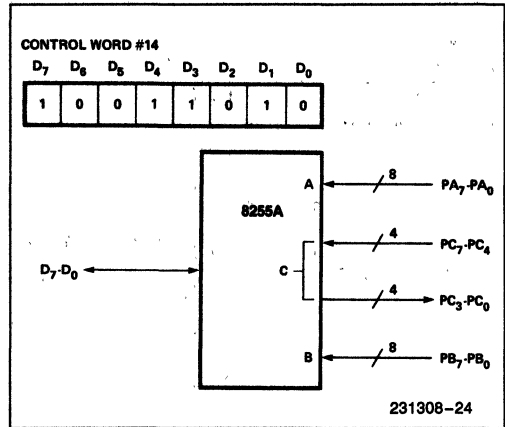
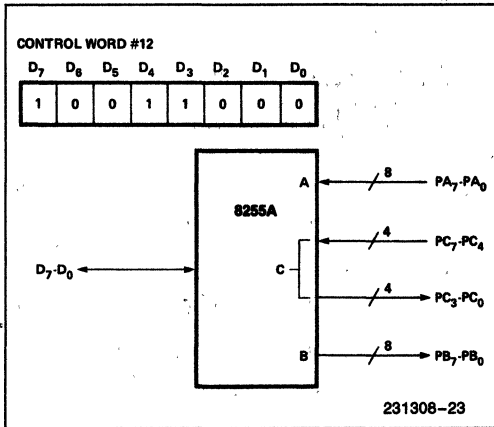
1000000000

A		B		Group A			Group B	
D ₄	D ₃	D ₁	D ₀	Port A	Port C (Upper)	#	Port B	Port C (Lower)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE CONFIGURATIONS







Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and port B use the lines on port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the STB is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of RD. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC₄.

INTE B

Controlled by bit set/reset of PC₂.

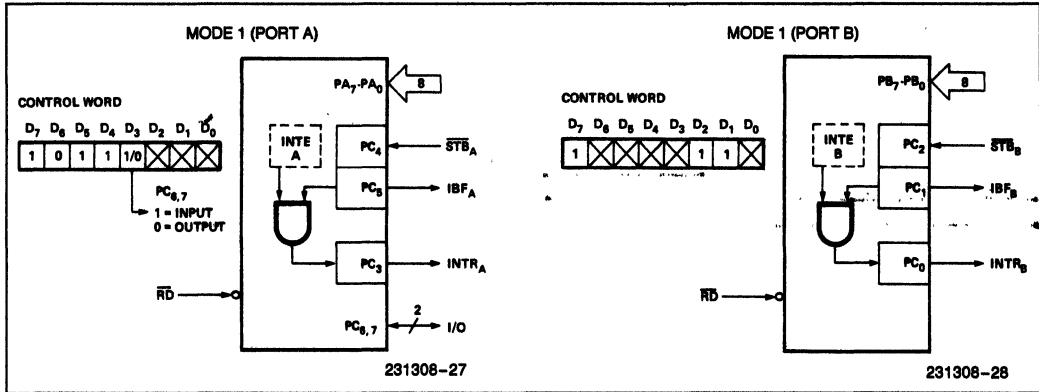


Figure 8. MODE 1 Input

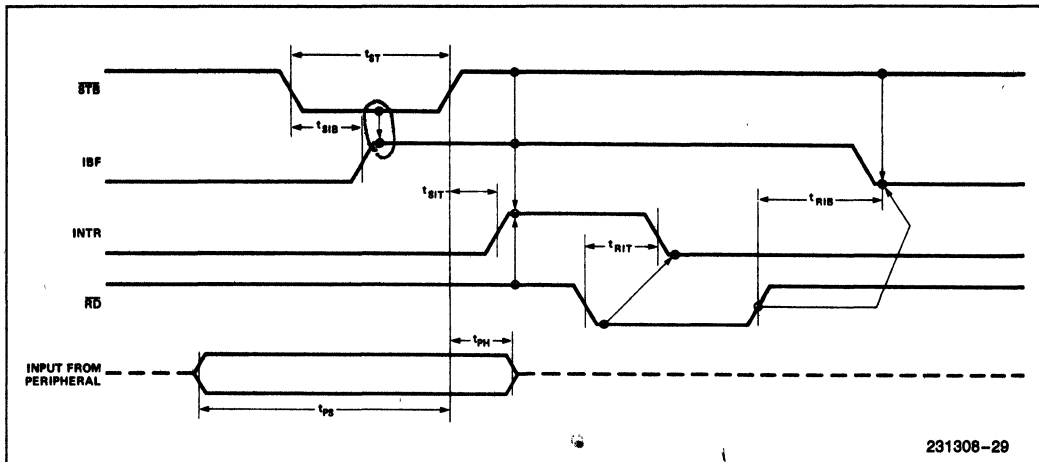


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

OB \bar{F} (Output Buffer Full F/F). The OB \bar{F} output will go "low" to indicate that the CPU has written data out to the specified port. The OB \bar{F} F/F will be set by the rising edge of the WR input and reset by ACK input being low.

ACK (Acknowledge Input). A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output

device has accepted data transmitted by the CPU. INTR is set when ACK is a "one", OB \bar{F} is a "one", and INTE is a "one". It is reset by the falling edge of WR.

INTE A

Controlled by bit set/reset of PC₆.

INTE B

Controlled by bit set/reset of PC₂.

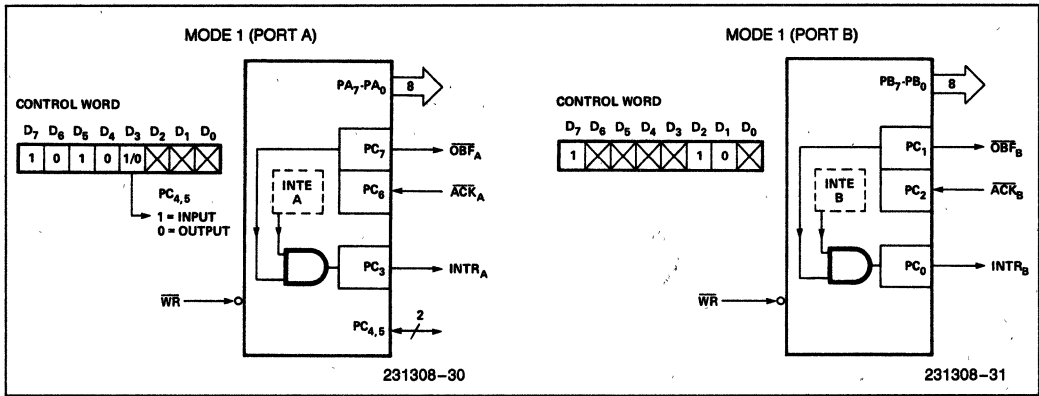


Figure 10. MODE 1 Output

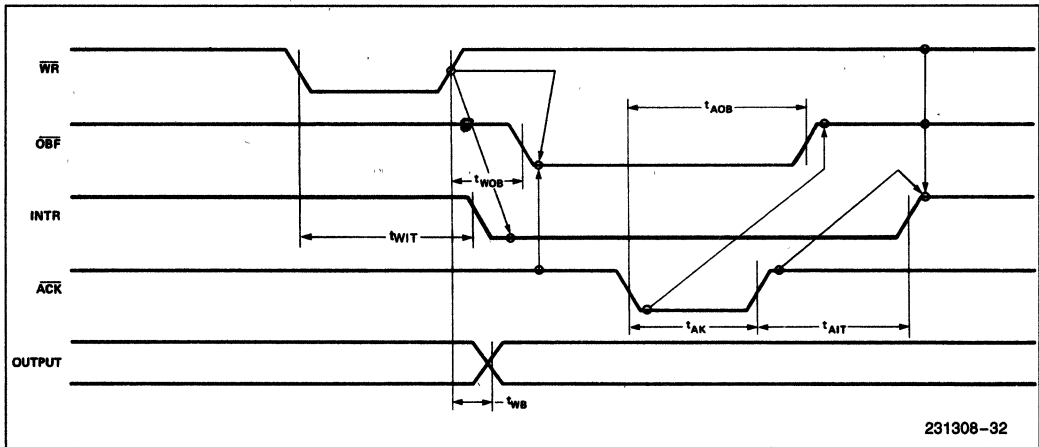


Figure 11. MODE 1 (Strobed Output)

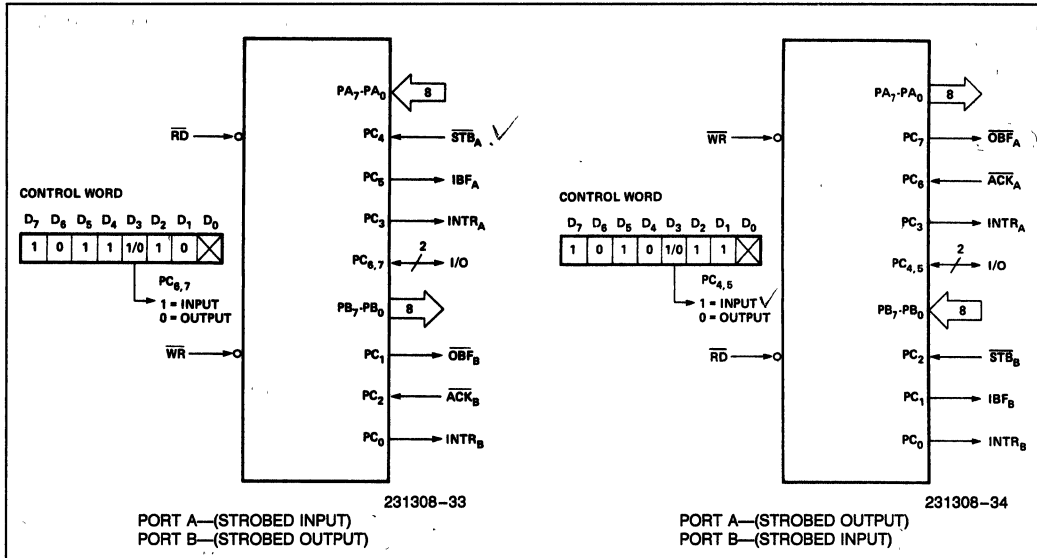


Figure 12. Combinations of MODE 1

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in MODE 1 to support a wide variety of strobed I/O applications.

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O) This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

OBF (Output Buffer Full). The OBF output will go "low" to indicate that the CPU has written data out to port A.

ACK (Acknowledge). A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC₆.

Input Operations

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC₄.

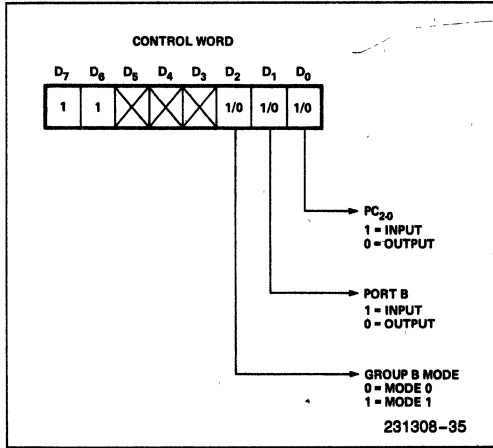


Figure 13. MODE Control Word

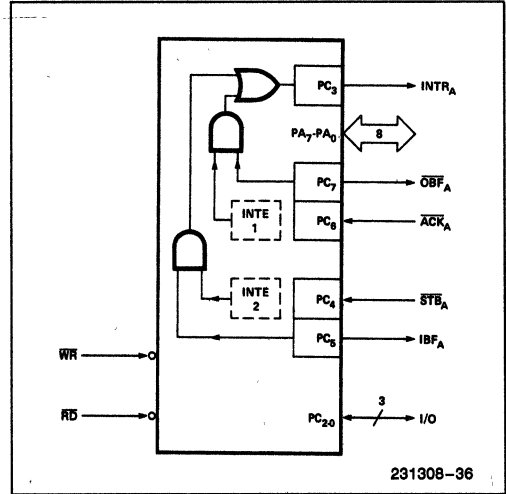
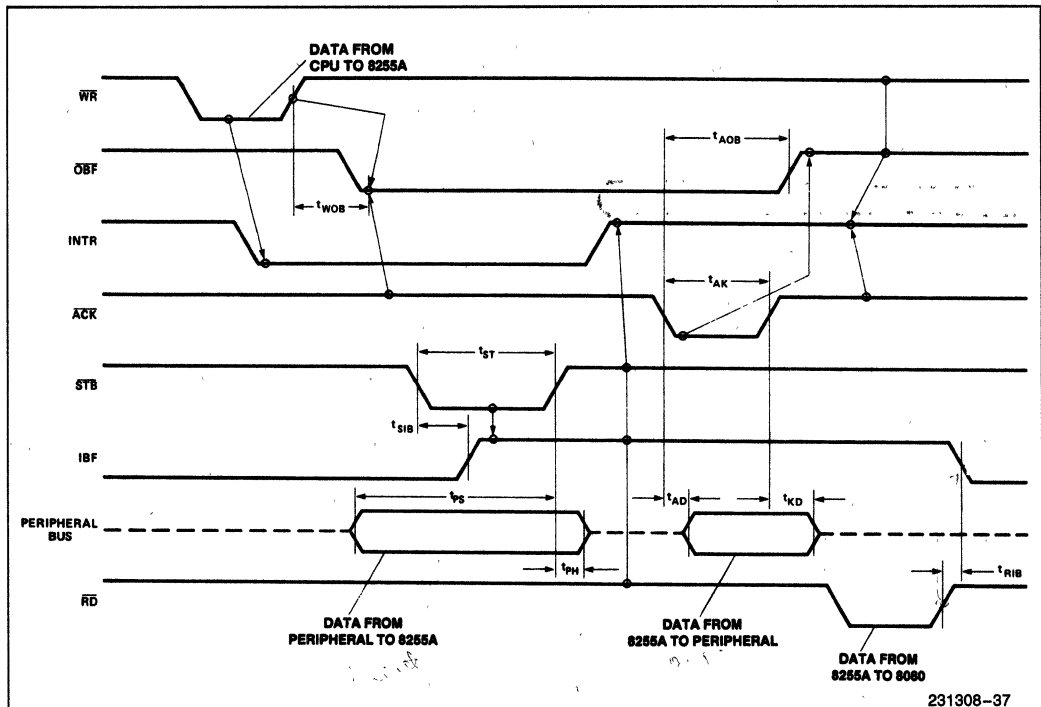


Figure 14. MODE 2



NOTE:
 Any sequence where \overline{WR} occurs before \overline{ACK} and \overline{STB} occurs before \overline{RD} is permissible.
 (INTR = IBF • MASK • STB • RD + OBF • MASK • ACK • WR)

Figure 15. MODE 2 (Bidirectional)

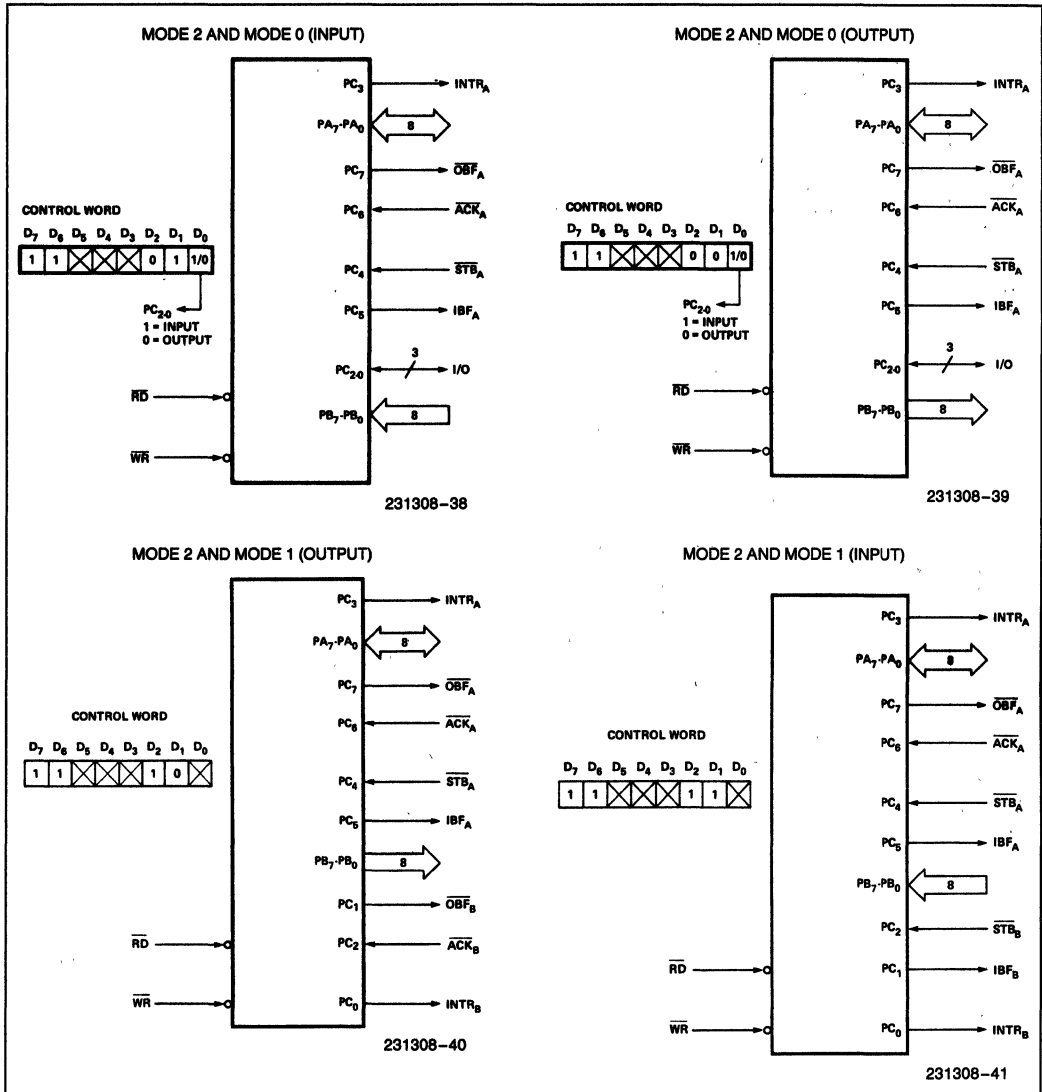


Figure 16. MODE 1/4 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2
	IN	OUT	IN	OUT	GROUP A ONLY
PA ₀	IN	OUT	IN	OUT	↔
PA ₁	IN	OUT	IN	OUT	↔
PA ₂	IN	OUT	IN	OUT	↔
PA ₃	IN	OUT	IN	OUT	↔
PA ₄	IN	OUT	IN	OUT	↔
PA ₅	IN	OUT	IN	OUT	↔
PA ₆	IN	OUT	IN	OUT	↔
PA ₇	IN	OUT	IN	OUT	↔
PB ₀	IN	OUT	IN	OUT	—
PB ₁	IN	OUT	IN	OUT	—
PB ₂	IN	OUT	IN	OUT	—
PB ₃	IN	OUT	IN	OUT	—
PB ₄	IN	OUT	IN	OUT	—
PB ₅	IN	OUT	IN	OUT	—
PB ₆	IN	OUT	IN	OUT	—
PB ₇	IN	OUT	IN	OUT	—
PC ₀	IN	OUT	INTR _B	INTR _B	I/O
PC ₁	IN	OUT	IBF _B	OBFB	I/O
PC ₂	IN	OUT	STB _B	ACK _B	I/O
PC ₃	IN	OUT	INTR _A	INTR _A	INTR _A
PC ₄	IN	OUT	STB _A	I/O	STB _A
PC ₅	IN	OUT	IBF _A	I/O	IBF _A
PC ₆	IN	OUT	I/O	ACK _A	ACK _A
PC ₇	IN	OUT	I/O	OBFA	OBFA

} MODE 0
OR MODE 1
ONLY

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs—

All input lines can be accessed during a normal Port C read.

If Programmed as Outputs—

Bits in C upper (PC₇–PC₄) must be individually accessed using the bit set/reset function.

Bits in C lower (PC₃–PC₀) can be accessed using the bit set/reset function or accessed as a three-some by writing into Port C.

Source Current Capability on Port B and Port C

Any set of **eight** output buffers, selected randomly from Ports B and C can source 1 mA at 1.5 volts.

This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

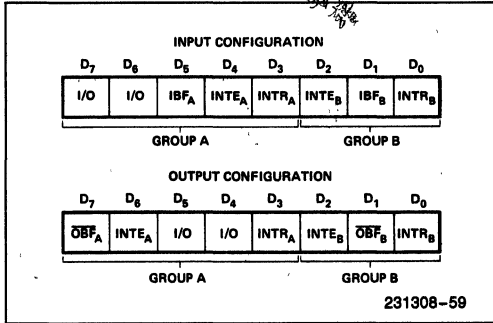


Figure 17. MODE 1 Status Word Format

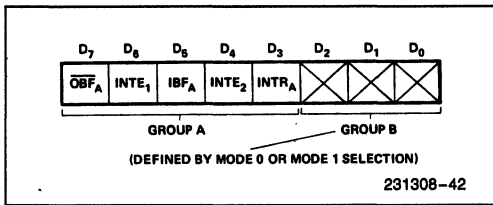


Figure 18. MODE 2 Status Word Format

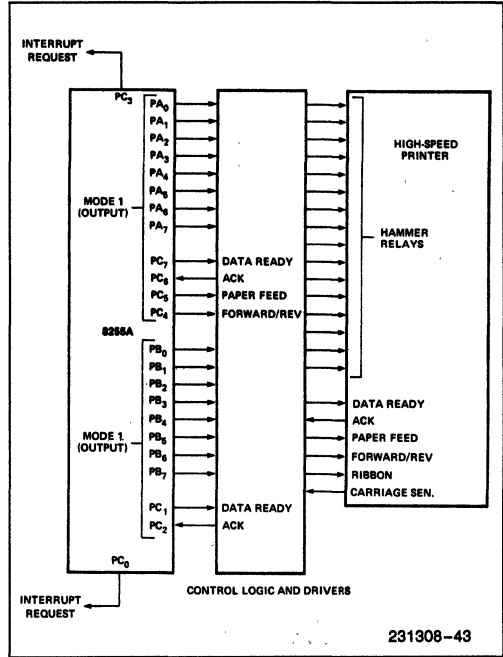


Figure 19. Printer Interface

APPLICATIONS OF THE 8255A

The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 19 through 25 represent a few examples of typical applications of the 8255A.

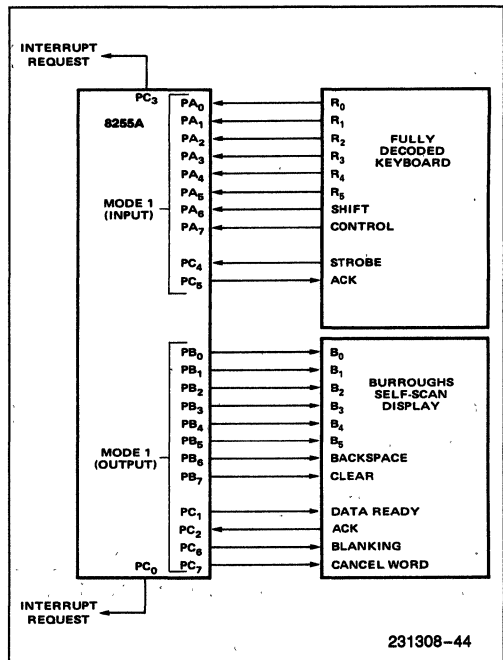


Figure 20. Keyboard and Display Interface

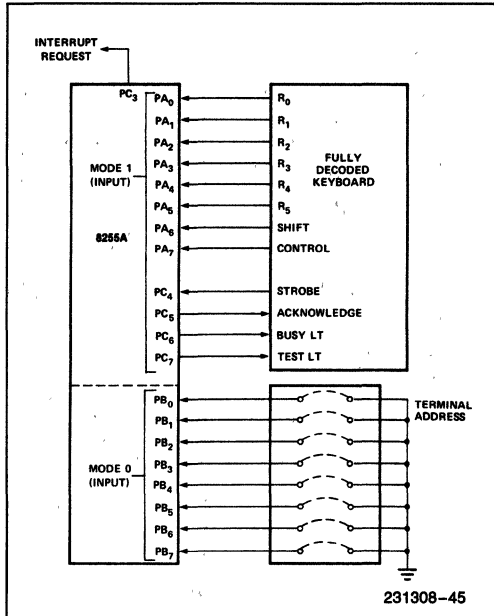


Figure 21. Keyboard and Terminal Address Interface

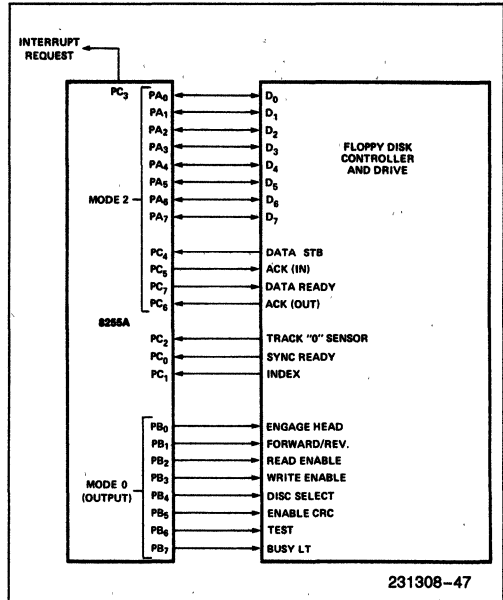


Figure 23. Basic Floppy Disk Interface

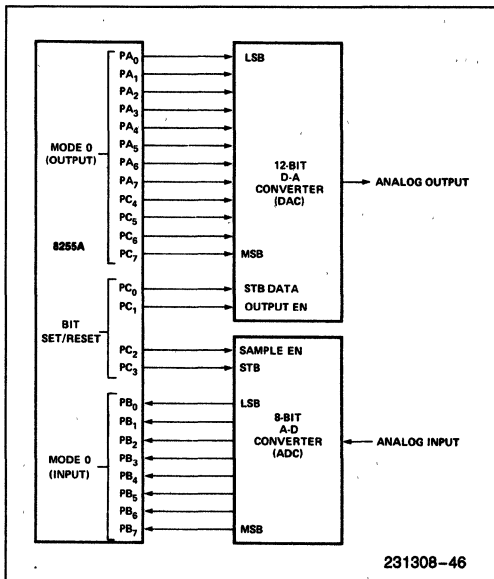


Figure 22. Digital to Analog, Analog to Digital

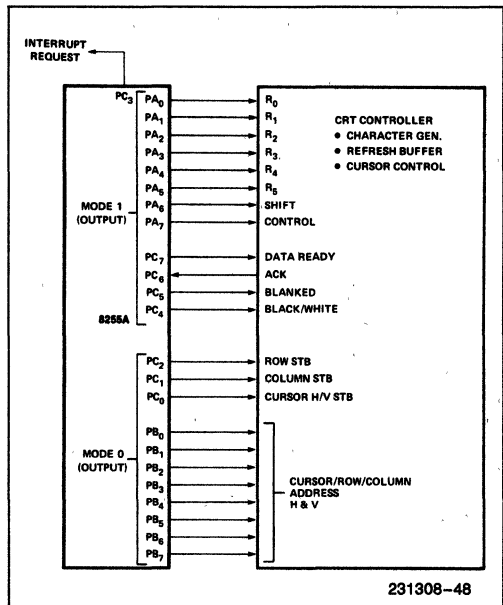


Figure 24. Basic CRT Controller Interface

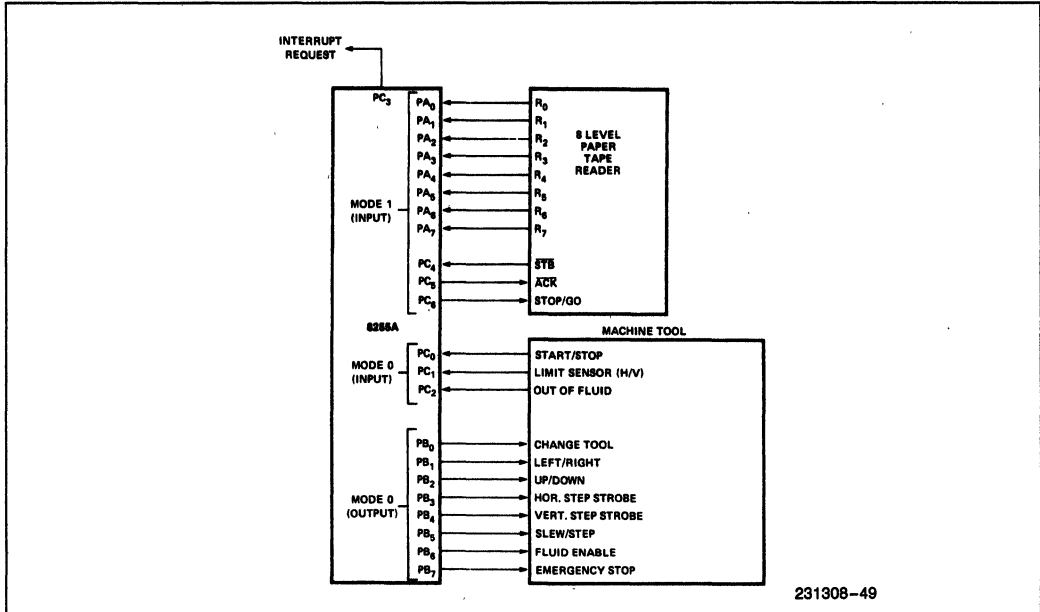


Figure 25. Machine Tool Controller Interface

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage on Any Pin
 with Respect to Ground..... -0.5V to +7V
 Power Dissipation 1 Watt

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

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5V \pm 10\%$, $GND = 0V^*$

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
$V_{OL} (DB)$	Output Low Voltage (Data Bus)		0.45*	V	$I_{OL} = 2.5 \text{ mA}$
$V_{OL} (PER)$	Output Low Voltage (Peripheral Port)		0.45*	V	$I_{OL} = 1.7 \text{ mA}$
$V_{OH} (DB)$	Output High Voltage (Data Bus)	2.4		V	$I_{OH} = -400 \mu\text{A}$
$V_{OH} (PER)$	Output High Voltage (Peripheral Port)	2.4		V	$I_{OH} = -200 \mu\text{A}$
$I_{DAR}^{(1)}$	Darlington Drive Current	-1.0	-4.0	mA	$R_{EXT} = 750\Omega$; $V_{EXT} = 1.5V$
I_{CC}	Power Supply Current		120	mA	
I_{IL}	Input Load Current		± 10	μA	$V_{IN} = V_{CC} \text{ to } 0V$
I_{OFL}	Output Float Leakage		± 10	μA	$V_{OUT} = V_{CC} \text{ to } 0.45V$

NOTE:

1. Available on any 8 pins from Port B and C.

CAPACITANCE $T_A = 25^\circ\text{C}$, $V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	$f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = +5\text{V} \pm 10\%$, $\text{GND} = 0\text{V}^*$
Bus Parameters
READ

Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{AR}	Address Stable before READ	0		0		ns
t_{RA}	Address Stable after READ	0		0		ns
t_{RR}	READ Pulse Width	300		300		ns
t_{RD}	Data Valid from READ ⁽¹⁾		250		200	ns
t_{DF}	Data Float after READ	10	150	10	100	ns
t_{RV}	Time between READs and/or WRITEs	850		850		ns

WRITE

Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{AW}	Address Stable before WRITE	0		0		ns
t_{WA}	Address Stable after WRITE	20		20		ns
t_{WW}	WRITE Pulse Width	400		300		ns
t_{DW}	Data Valid to WRITE (T.E.)	100		100		ns
t_{WD}	Data Valid after WRITE	30		30		ns

OTHER TIMINGS

Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{WB}	WR = 1 to Output ⁽¹⁾		350		350	ns
t_{IR}	Peripheral Data before RD	0		0		ns
t_{HR}	Peripheral Data after RD	0		0		ns
t_{AK}	ACK Pulse Width	300		300		ns
t_{ST}	STB Pulse Width	500		500		ns
t_{PS}	Per. Data before T.E. of STB	0		0		ns
t_{PH}	Per. Data after T.E. of STB	180		180		ns
t_{AD}	ACK = 0 to Output ⁽¹⁾		300		300	ns
t_{KD}	ACK = 1 to Output Float	20	250	20	250	ns

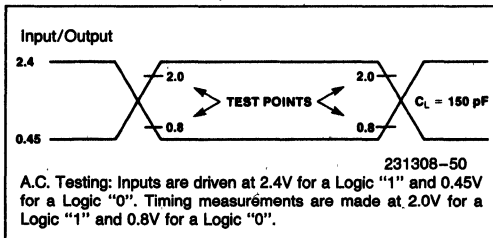
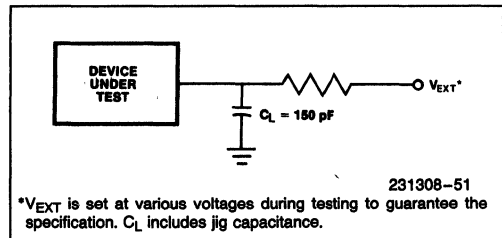
A.C. CHARACTERISTICS (Continued)
OTHER TIMINGS (Continued)

Symbol	Parameter	8255A		8255A-5		Unit
		Min	Max	Min	Max	
t_{WOB}	WR = 1 to OBF = 0(1)		650		650	ns
t_{AOB}	ACK = 0 to OBF = 1(1)		350		350	ns
t_{SIB}	STB = 0 to IBF = 1(1)		300		300	ns
t_{RIB}	RD = 1 to IBF = 0(1)		300		300	ns
t_{RIT}	RD = 0 to INTR = 0(1)		400		400	ns
t_{SIT}	STB = 1 to INTR = 1(1)		300		300	ns
t_{AIT}	ACK = 1 to INTR = 1(1)		350		350	ns
t_{WIT}	WR = 0 to INTR = 0(1,3)		450		450	ns

NOTES:

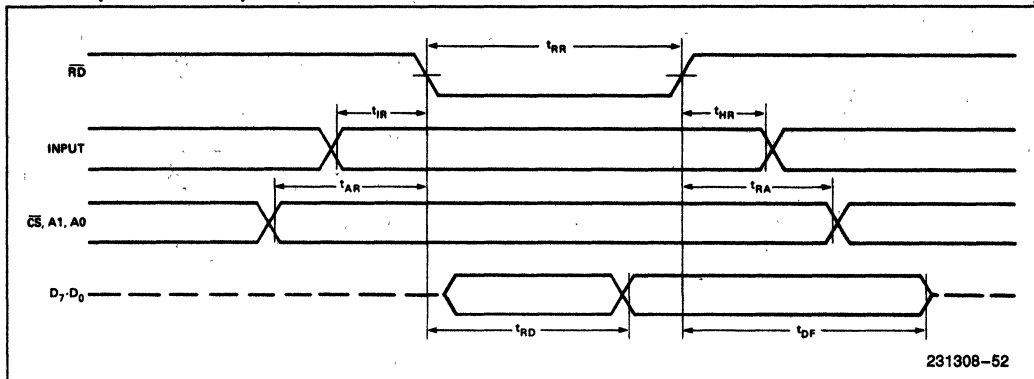
1. Test Conditions: $C_L = 150$ pF.
2. Period of Reset pulse must be at least 50 μ s during or after power on. Subsequent Reset pulse can be 500 ns min.
3. INTR \uparrow may occur as early as WR \downarrow .

*For Extended Temperature EXPRESS, use M8255A electrical parameters.

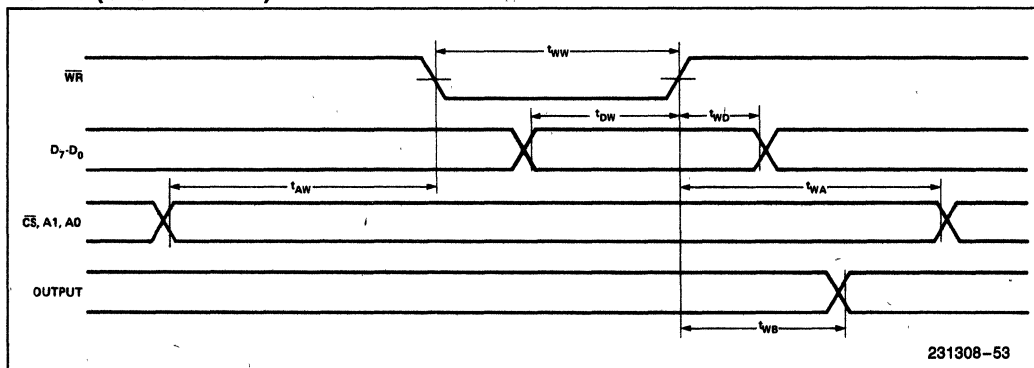
A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. TESTING LOAD CIRCUIT


WAVEFORMS

MODE 0 (BASIC INPUT)

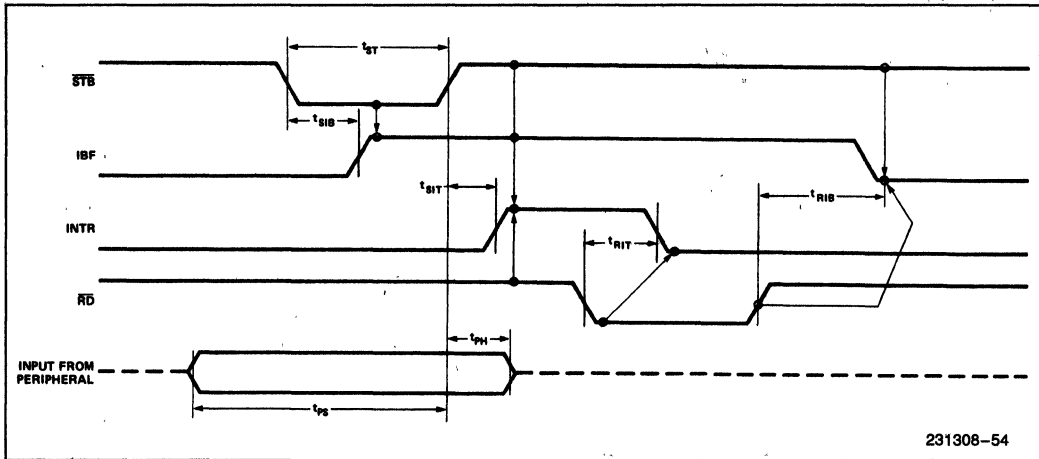


MODE 0 (BASIC OUTPUT)

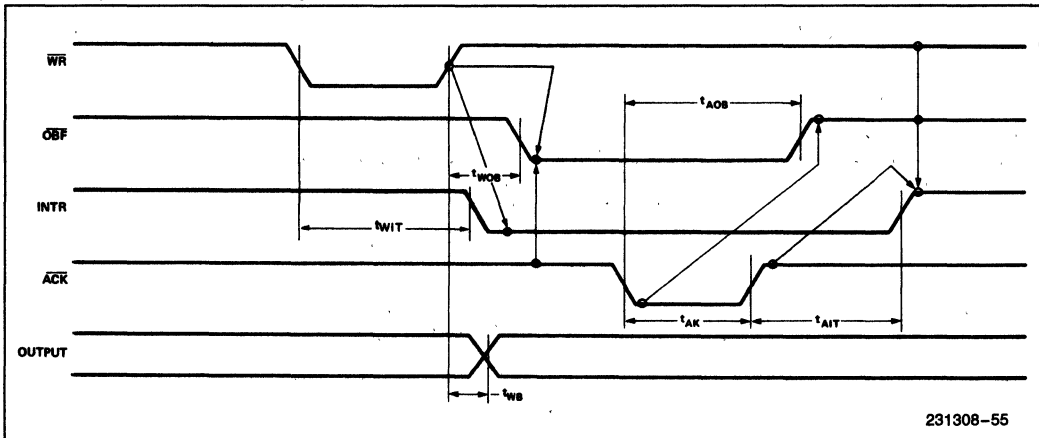


WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)

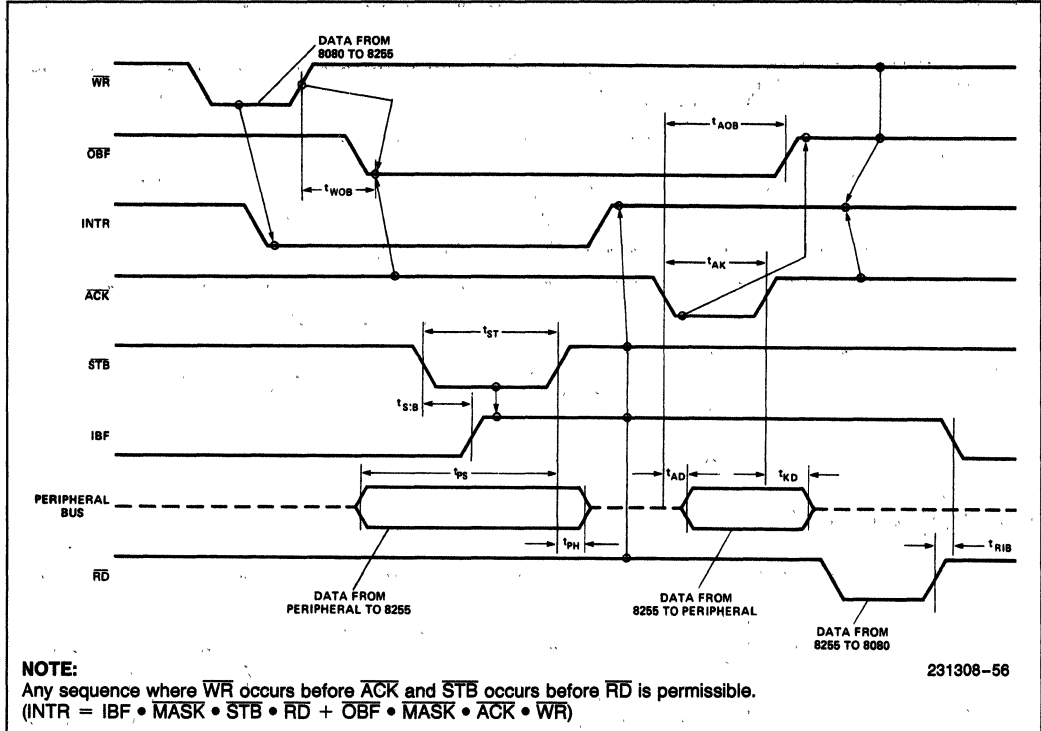


MODE 1 (STROBED OUTPUT)

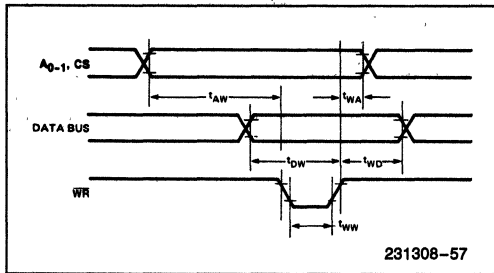


WAVEFORMS (Continued)

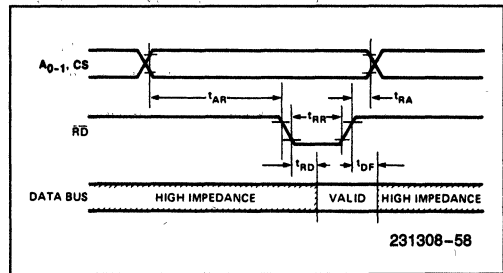
MODE 2 (BIDIRECTIONAL)



WRITE TIMING



READ TIMING





82C55A CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- Compatible with all Intel and Most Other Microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- 24 Programmable I/O Pins
- Low Power CHMOS
- Completely TTL Compatible
- Control Word Read-Back Capability
- Direct Bit Set/Reset Capability
- 2.5 mA DC Drive Capability on all I/O Port Outputs
- Available in 40-Pin DIP and 44-Pin PLCC
- Available in EXPRESS
 - Standard Temperature Range
 - Extended Temperature Range

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.

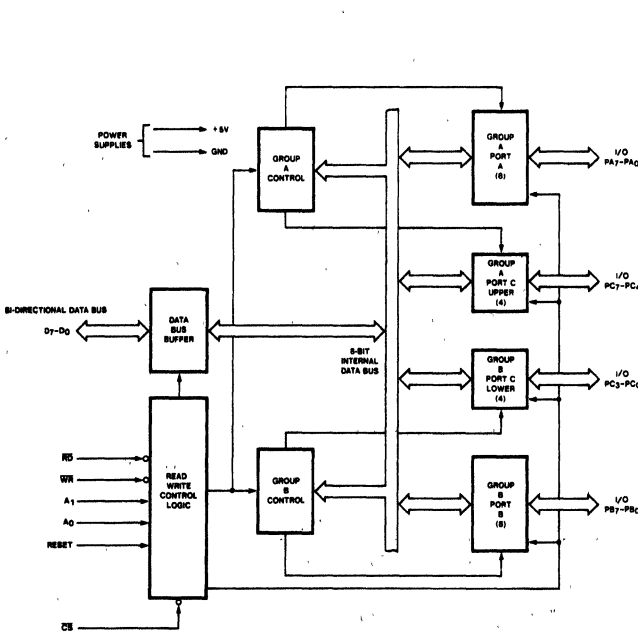
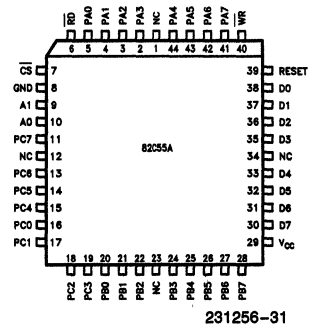
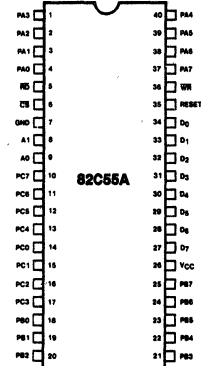


Figure 1. 82C55A Block Diagram

231256-1



231256-31



231256-8

Figure 2. 82C55A Pinout
Diagrams are for pin reference only. Package sizes are not to scale.

Table 1. Pin Description

Symbol	Pin Number Dip	PLCC	Type	Name and Function																																																																														
PA ₃₋₀	1-4	2-5	I/O	PORT A, PINS 0-3: Lower nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.																																																																														
\overline{RD}	5	6	I	READ CONTROL: This input is low during CPU read operations.																																																																														
\overline{CS}	6	7	I	CHIP SELECT: A low on this input enables the 82C55A to respond to \overline{RD} and \overline{WR} signals. \overline{RD} and \overline{WR} are ignored otherwise.																																																																														
GND	7	8		System Ground																																																																														
A ₁₋₀	8-9	9-10	I	<p>ADDRESS: These input signals, in conjunction \overline{RD} and \overline{WR}, control the selection of one of the three ports or the control word registers.</p> <table border="1"> <thead> <tr> <th>A₁</th> <th>A₀</th> <th>\overline{RD}</th> <th>\overline{WR}</th> <th>\overline{CS}</th> <th>Input Operation (Read)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>Port A - Data Bus</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>Port B - Data Bus</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>Port C - Data Bus</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>Control Word - Data Bus</td> </tr> <tr> <th colspan="6">Output Operation (Write)</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Port A</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Port B</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Port C</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>Data Bus - Control</td> </tr> <tr> <th colspan="6">Disable Function</th> </tr> <tr> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>1</td> <td>Data Bus - 3 - State</td> </tr> <tr> <td>X</td> <td>X</td> <td>1</td> <td>1</td> <td>0</td> <td>Data Bus - 3 - State</td> </tr> </tbody> </table>	A ₁	A ₀	\overline{RD}	\overline{WR}	\overline{CS}	Input Operation (Read)	0	0	0	1	0	Port A - Data Bus	0	1	0	1	0	Port B - Data Bus	1	0	0	1	0	Port C - Data Bus	1	1	0	1	0	Control Word - Data Bus	Output Operation (Write)						0	0	1	0	0	Data Bus - Port A	0	1	1	0	0	Data Bus - Port B	1	0	1	0	0	Data Bus - Port C	1	1	1	0	0	Data Bus - Control	Disable Function						X	X	X	X	1	Data Bus - 3 - State	X	X	1	1	0	Data Bus - 3 - State
A ₁	A ₀	\overline{RD}	\overline{WR}	\overline{CS}	Input Operation (Read)																																																																													
0	0	0	1	0	Port A - Data Bus																																																																													
0	1	0	1	0	Port B - Data Bus																																																																													
1	0	0	1	0	Port C - Data Bus																																																																													
1	1	0	1	0	Control Word - Data Bus																																																																													
Output Operation (Write)																																																																																		
0	0	1	0	0	Data Bus - Port A																																																																													
0	1	1	0	0	Data Bus - Port B																																																																													
1	0	1	0	0	Data Bus - Port C																																																																													
1	1	1	0	0	Data Bus - Control																																																																													
Disable Function																																																																																		
X	X	X	X	1	Data Bus - 3 - State																																																																													
X	X	1	1	0	Data Bus - 3 - State																																																																													
PC ₇₋₄	10-13	11,13-15	I/O	PORT C, PINS 4-7: Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.																																																																														
PC ₀₋₃	14-17	16-19	I/O	PORT C, PINS 0-3: Lower nibble of Port C.																																																																														
PB ₀₋₇	18-25	20-22, 24-28	I/O	PORT B, PINS 0-7: An 8-bit data output latch/buffer and an 8-bit data input buffer.																																																																														
V _{CC}	26	29		SYSTEM POWER: + 5V Power Supply.																																																																														
D ₇₋₀	27-34	30-33, 35-38	I/O	DATA BUS: Bi-directional, tri-state data bus lines, connected to system data bus.																																																																														
RESET	35	39	I	RESET: A high on this input clears the control register and all ports are set to the input mode.																																																																														
\overline{WR}	36	40	I	WRITE CONTROL: This input is low during CPU write operations.																																																																														
PA ₇₋₄	37-40	41-44	I/O	PORT A, PINS 4-7: Upper nibble of an 8-bit data output latch/buffer and an 8-bit data input latch.																																																																														
NC		1, 12, 23, 34		No Connect																																																																														

82C55A FUNCTIONAL DESCRIPTION

General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7-C4)
Control Group B - Port B and Port C lower (C3-C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A. One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

Port B. One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

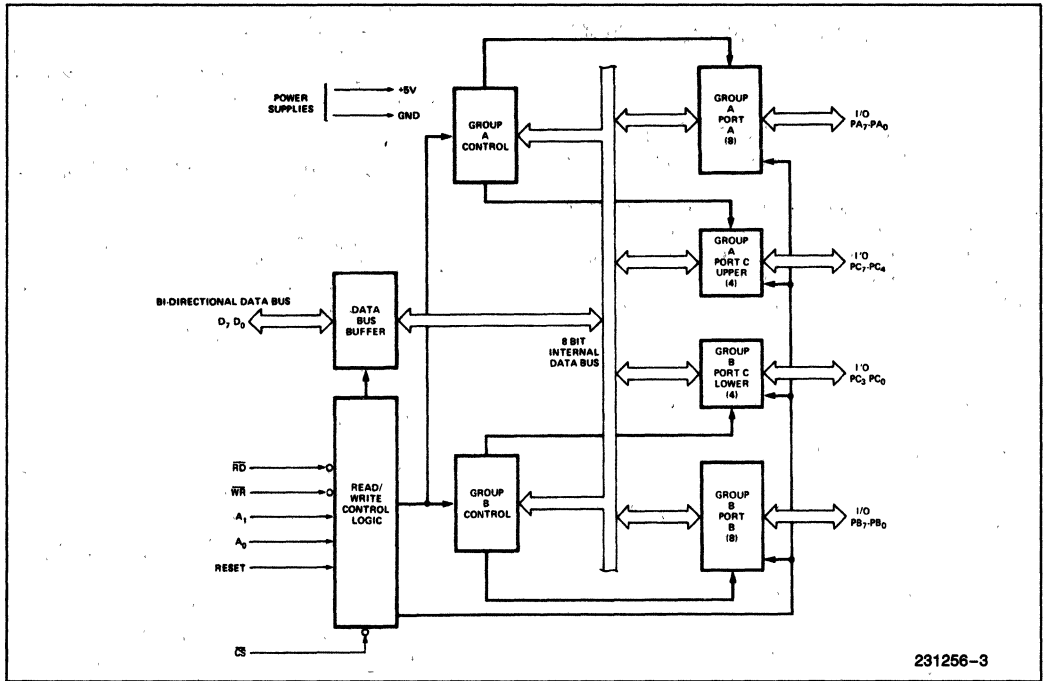
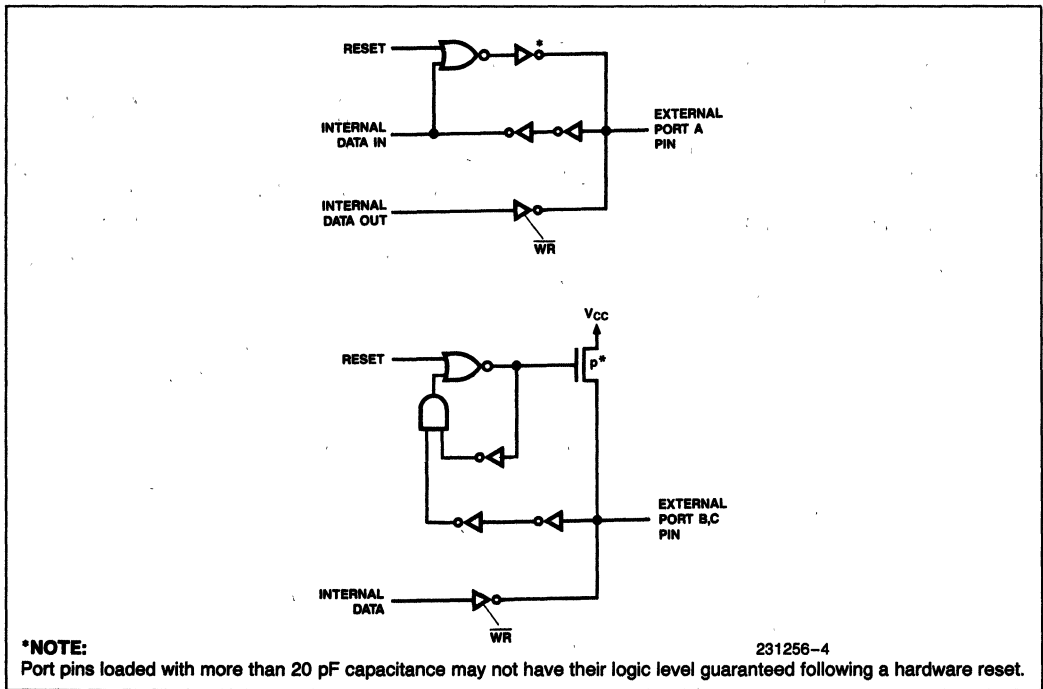


Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



*NOTE:

Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

231256-4

Figure 4. Port A, B, C, Bus-hold Configuration

82C55A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

- Mode 0 — Basic input/output
- Mode 1 — Strobed Input/output
- Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

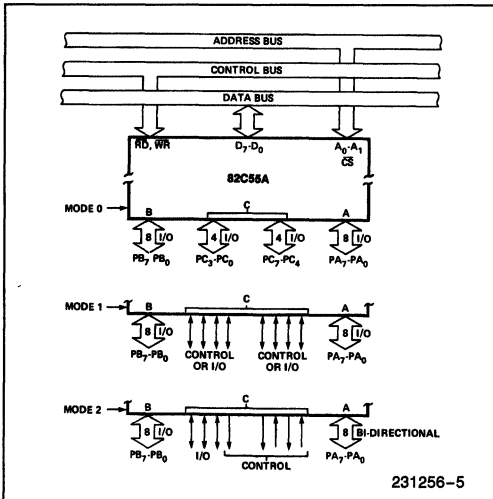


Figure 5. Basic Mode Definitions and Bus Interface

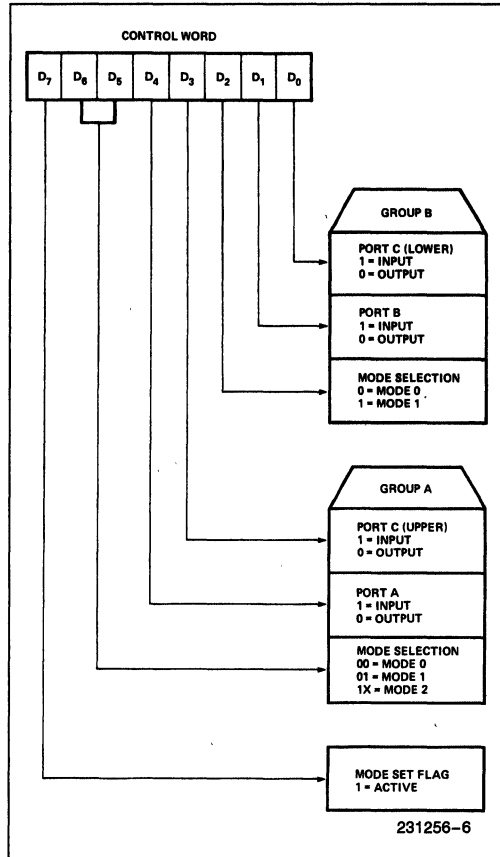


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

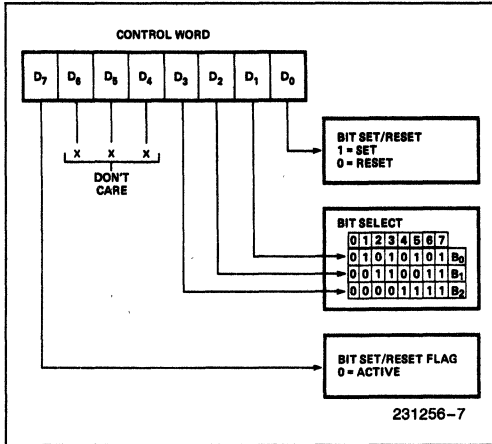


Figure 7. Bit Set/Reset Format

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

- (BIT-SET)—INTE is SET—Interrupt enable
- (BIT-RESET)—INTE is RESET—Interrupt disable

Note:

All Mask flip-flops are automatically reset during mode selection and device Reset.

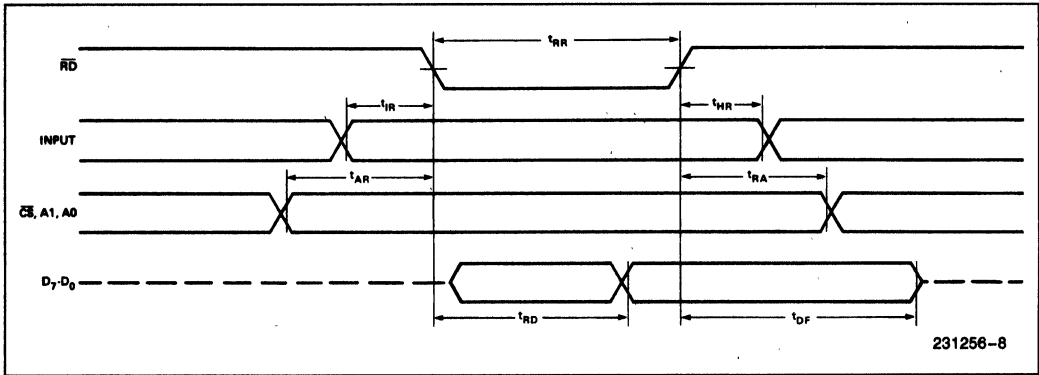
Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

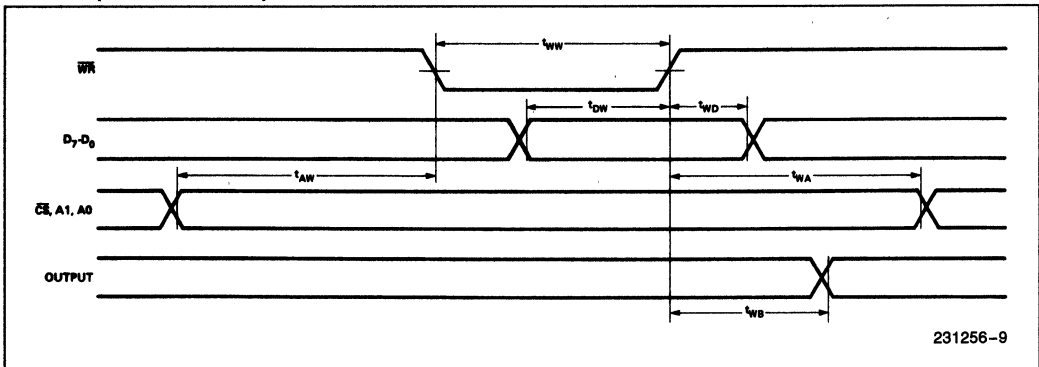
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

MODE 0 (BASIC INPUT)



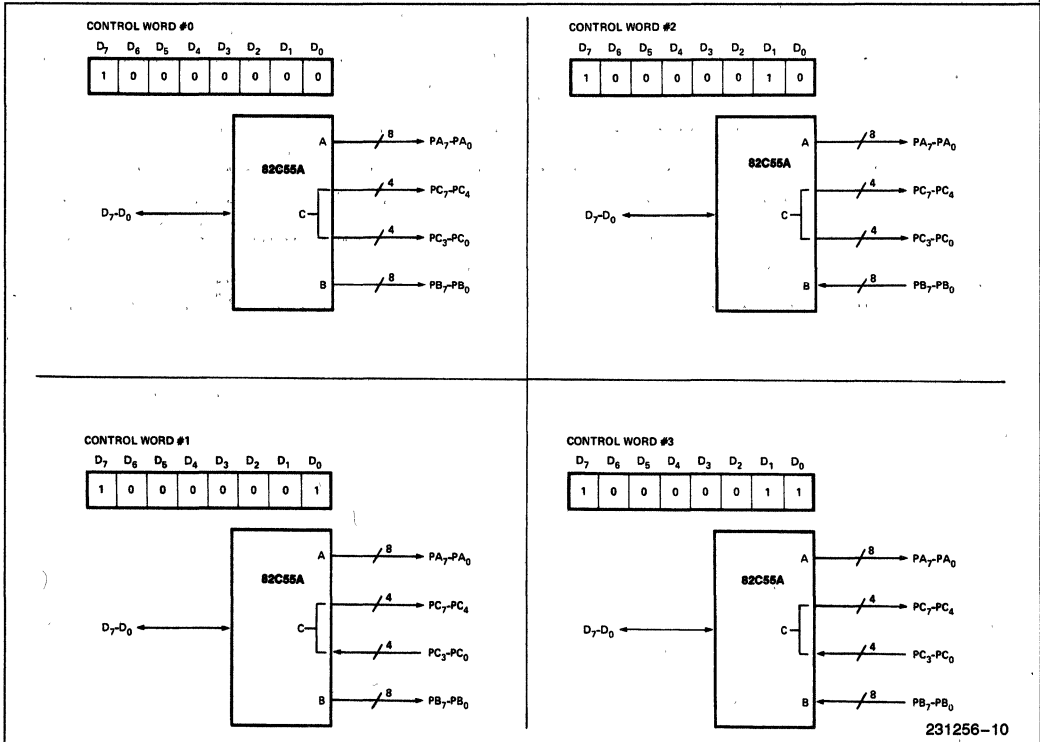
MODE 0 (BASIC OUTPUT)



MODE 0 Port Definition

A		B		GROUP A			GROUP B	
D ₄	D ₃	D ₁	D ₀	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 Configurations

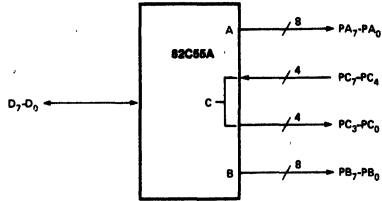


231256-10

MODE 0 Configurations (Continued)

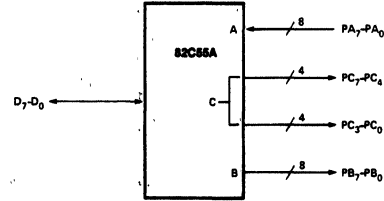
CONTROL WORD #4

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	0	0



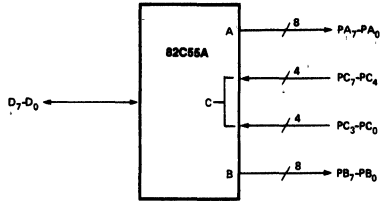
CONTROL WORD #8

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	0	0	0	0



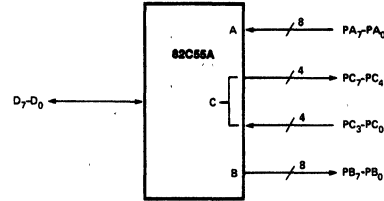
CONTROL WORD #5

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	0	1



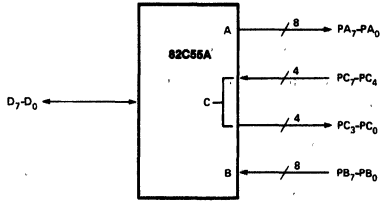
CONTROL WORD #9

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	0	0	0	1



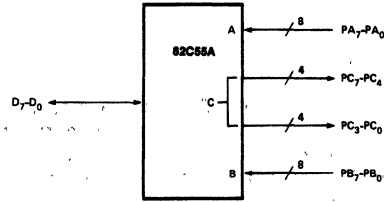
CONTROL WORD #6

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	1	0



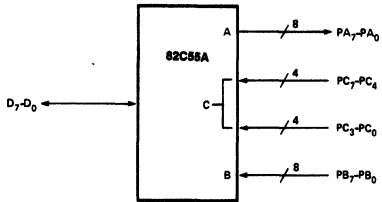
CONTROL WORD #10

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	0	0	1	0



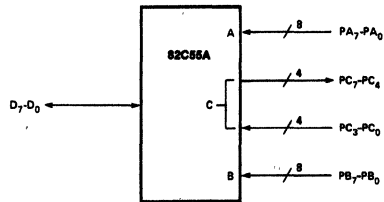
CONTROL WORD #7

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	0	1	0	1	1

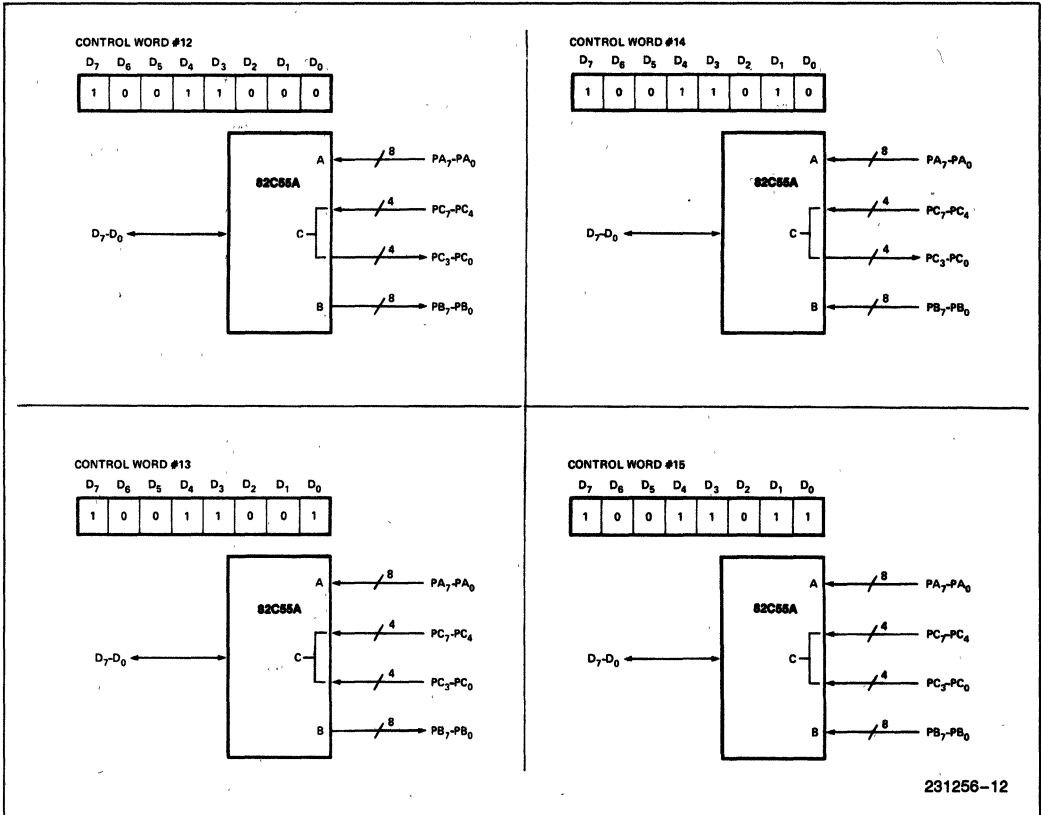


CONTROL WORD #11

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	0	0	1	0	0	1	1



MODE 0 Configurations (Continued)



Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Input Control Signal Definition

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the STB is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of RD. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC₄.

INTE B

Controlled by bit set/reset of PC₂.

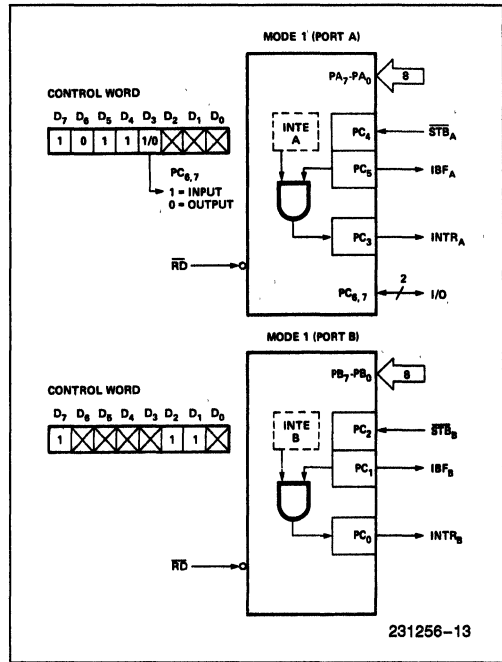


Figure 8. MODE 1 Input

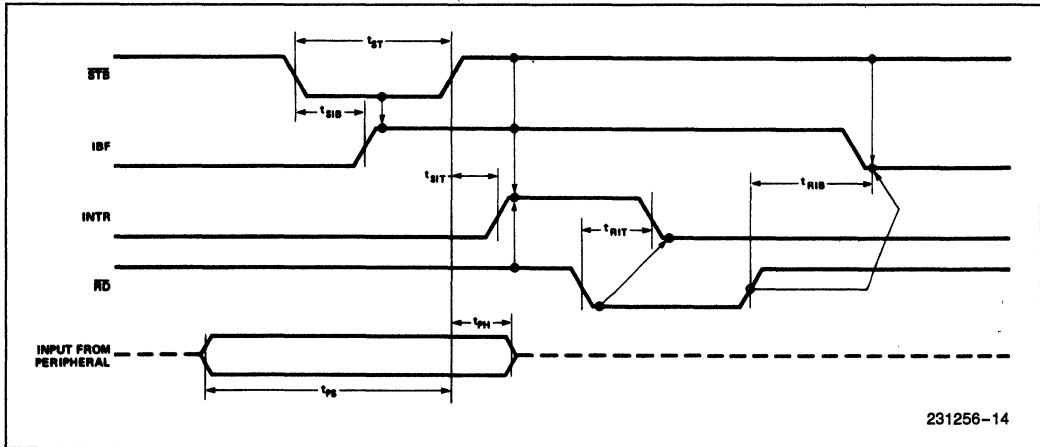


Figure 9. MODE 1 (Strobed Input)

Output Control Signal Definition

\overline{OBF} (Output Buffer Full F/F). The \overline{OBF} output will go "low" to indicate that the CPU has written data out to the specified port. The \overline{OBF} F/F will be set by the rising edge of the WR input and reset by ACK Input being low.

\overline{ACK} (Acknowledge Input). A "low" on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when \overline{ACK} is a "one", \overline{OBF} is a "one" and INTE is a "one". It is reset by the falling edge of WR.

INTE A

Controlled by bit set/reset of PC₆.

INTE B

Controlled by bit set/reset of PC₂.

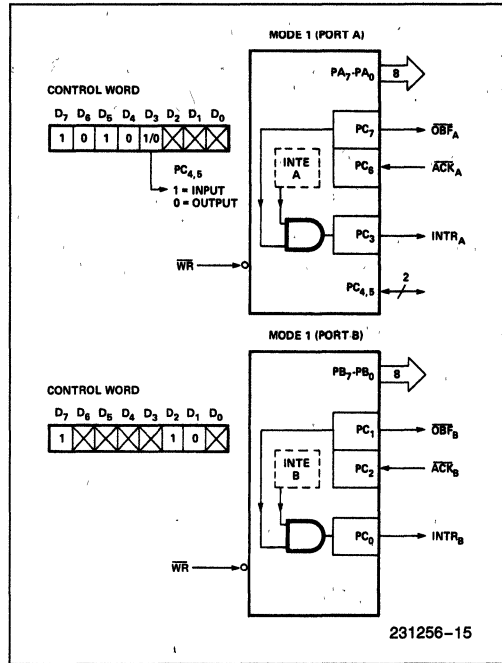


Figure 10. MODE 1 Output

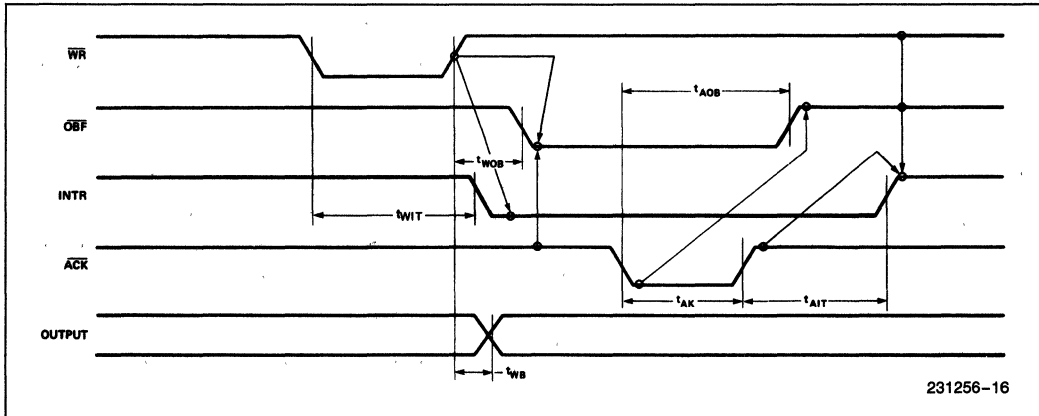


Figure 11. MODE 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

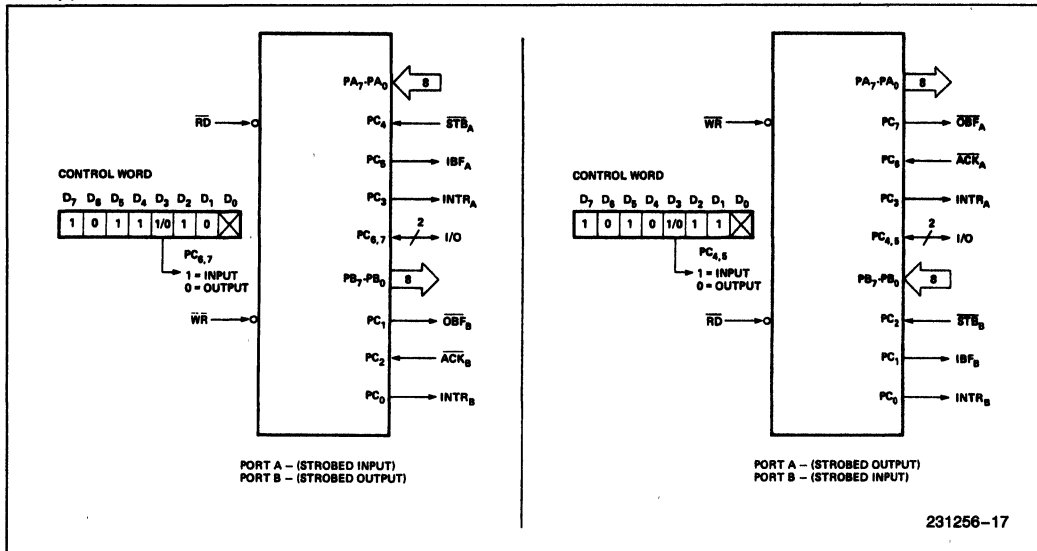


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only**.
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for input or output operations.

Output Operations

OBF (Output Buffer Full). The OBF output will go "low" to indicate that the CPU has written data out to port A.

ACK (Acknowledge). A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC₆.

Input Operations

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC₄.

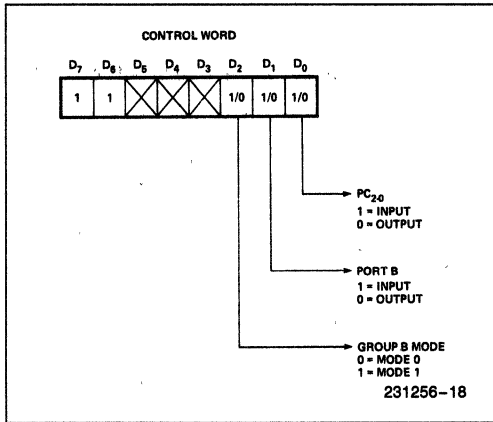


Figure 13. MODE Control Word

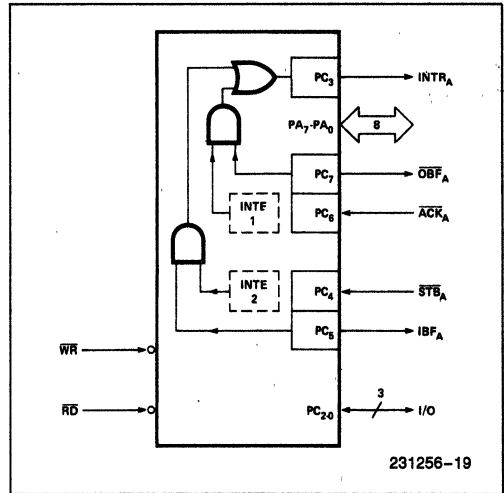


Figure 14. MODE 2

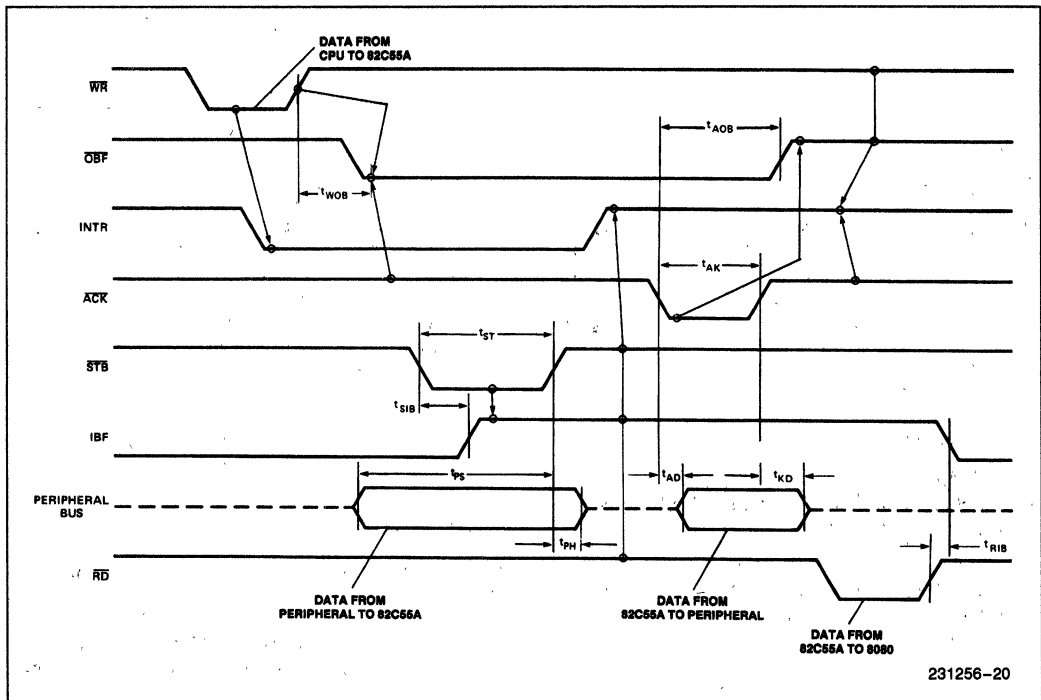
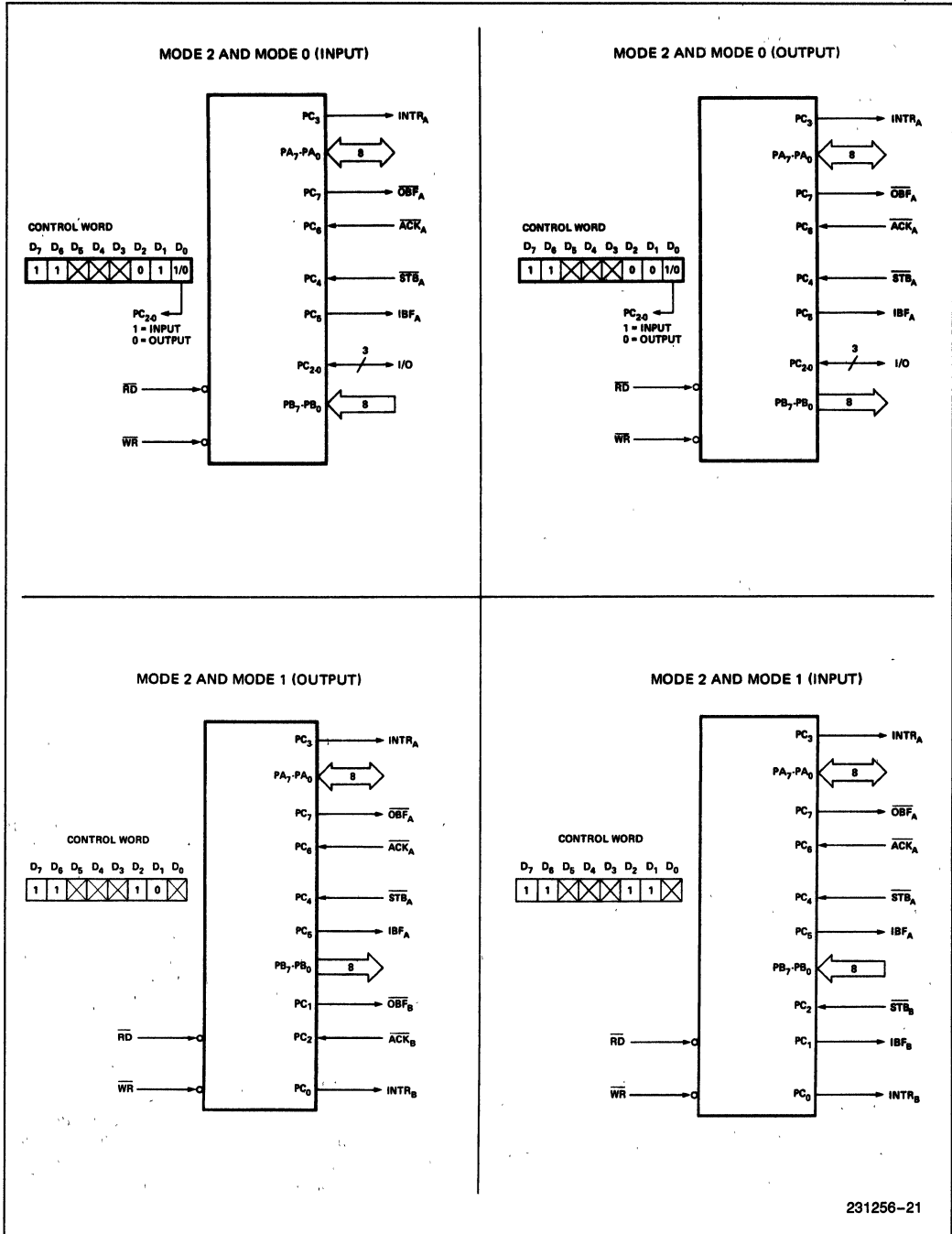


Figure 15. MODE 2 (Bidirectional)

NOTE:

Any sequence where \overline{WR} occurs before \overline{ACK} , and \overline{STB} occurs before \overline{RD} is permissible.
 $(INTR = IBF \cdot MASK \cdot \overline{STB} \cdot \overline{RD} + OBF \cdot MASK \cdot \overline{ACK} \cdot \overline{WR})$



231256-21

Figure 16. MODE 1/4 Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA ₀	IN	OUT	IN	OUT	↔	MODE 0 OR MODE 1 ONLY
PA ₁	IN	OUT	IN	OUT	↔	
PA ₂	IN	OUT	IN	OUT	↔	
PA ₃	IN	OUT	IN	OUT	↔	
PA ₄	IN	OUT	IN	OUT	↔	
PA ₅	IN	OUT	IN	OUT	↔	
PA ₆	IN	OUT	IN	OUT	↔	
PA ₇	IN	OUT	IN	OUT	↔	
PB ₀	IN	OUT	IN	OUT	—	
PB ₁	IN	OUT	IN	OUT	—	
PB ₂	IN	OUT	IN	OUT	—	
PB ₃	IN	OUT	IN	OUT	—	
PB ₄	IN	OUT	IN	OUT	—	
PB ₅	IN	OUT	IN	OUT	—	
PB ₆	IN	OUT	IN	OUT	—	
PB ₇	IN	OUT	IN	OUT	—	
PC ₀	IN	OUT	INTR _B	INTR _B	I/O	
PC ₁	IN	OUT	IBF _B	OBF _B	I/O	
PC ₂	IN	OUT	STB _B	ACK _B	I/O	
PC ₃	IN	OUT	INTR _A	INTR _A	INTR _A	
PC ₄	IN	OUT	STB _A	I/O	STB _A	
PC ₅	IN	OUT	IBF _A	I/O	IBF _A	
PC ₆	IN	OUT	I/O	ACK _A	ACK _A	
PC ₇	IN	OUT	I/O	OBF _A	OBF _A	

Special Mode Combination Considerations

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ lines, will be placed on the data bus. In place of the $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to

change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and OBF) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{\text{ACK}}$ and $\overline{\text{STB}}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

Current Drive Capability

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

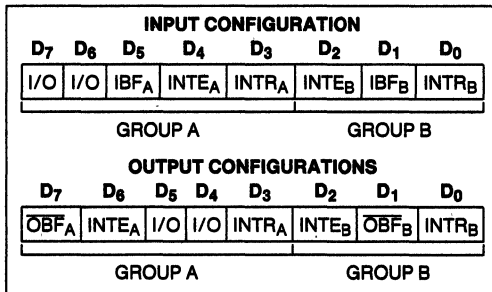


Figure 17a. MODE 1 Status Word Format

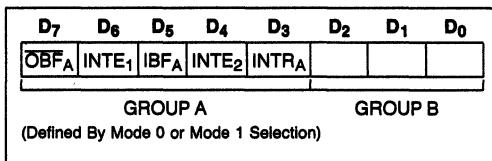


Figure 17b. MODE 2 Status Word Format

Interrupt Enable Flag	Position	Alternate Port C Pin Signal (Mode)
INTE B	PC2	\overline{ACK}_B (Output Mode 1) or \overline{STB}_B (Input Mode 1)
INTE A2	PC4	\overline{STB}_A (Input Mode 1 or Mode 2)
INTE A1	PC6	\overline{ACK}_A (Output Mode 1 or Mode 2)

Figure 18. Interrupt Enable Flags In Modes 1 and 2

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to + 70°C
 Storage Temperature - 65°C to + 150°C
 Supply Voltage - 0.5 to + 8.0V
 Operating Voltage + 4V to + 7V
 Voltage on any Input GND - 2V to + 6.5V
 Voltage on any Output . . GND - 0.5V to V_{CC} + 0.5V
 Power Dissipation 1 Watt

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

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 10%, GND = 0V (T_A = -40°C to +85°C for Extended Temperature)

Symbol	Parameter	Min	Max	Units	Test Conditions
V _{IL}	Input Low Voltage	-0.5	0.8	V	
V _{IH}	Input High Voltage	2.0	V _{CC}	V	
V _{OL}	Output Low Voltage		0.4	V	I _{OL} = 2.5 mA
V _{OH}	Output High Voltage	3.0 V _{CC} - 0.4		V V	I _{OH} = -2.5 mA I _{OH} = -100 μA
I _{IL}	Input Leakage Current		± 1	μA	V _{IN} = V _{CC} to 0V (Note 1)
I _{OFL}	Output Float Leakage Current		± 10	μA	V _{IN} = V _{CC} to 0V (Note 2)
I _{DAR}	Darlington Drive Current	± 2.5		mA	Ports A, B, C R _{ext} = 750Ω V _{ext} = 1.5V
I _{PHL}	Port Hold Low Leakage Current	+ 50	+ 300	μA	V _{OUT} = 1.0V Port A only
I _{PHH}	Port Hold High Leakage Current	- 50	- 300	μA	V _{OUT} = 3.0V Ports A, B, C
I _{PHLO}	Port Hold Low Overdrive Current	- 350		μA	V _{OUT} = 0.8V
I _{PHHO}	Port Hold High Overdrive Current	+ 350		μA	V _{OUT} = 3.0V
I _{CC}	V _{CC} Supply Current		10	mA	(Note 3)
I _{CCSB}	V _{CC} Supply Current-Standby		10	μA	V _{CC} = 5.5V V _{IN} = V _{CC} or GND Port Conditions If I/P = Open/High O/P = Open Only With Data Bus = High/Low CS = High Reset = Low Pure Inputs = Low/High

NOTES:

1. Pins A₁, A₀, CS, WR, RD, Reset.
2. Data Bus; Ports B, C.
3. Outputs open.

CAPACITANCE
 $T_A = 25^\circ\text{C}, V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	Unmeasured pins returned to GND $f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance		20	pF	

A.C. CHARACTERISTICS
 $T_A = 0^\circ\text{ to }70^\circ\text{C}, V_{CC} = +5\text{V} \pm 10\%, \text{GND} = 0\text{V}$
 $T_A = -40^\circ\text{C to }+85^\circ\text{C for Extended Temperature}$
BUS PARAMETERS
READ CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AR}	Address Stable Before $\overline{RD} \downarrow$	0		ns	
t_{RA}	Address Hold Time After $\overline{RD} \uparrow$	0		ns	
t_{RR}	\overline{RD} Pulse Width	150		ns	
t_{RD}	Data Delay from $\overline{RD} \downarrow$		120	ns	
t_{DF}	$\overline{RD} \uparrow$ to Data Floating	10	75	ns	
t_{RV}	Recovery Time between $\overline{RD}/\overline{WR}$	200		ns	

WRITE CYCLE

Symbol	Parameter	82C55A-2		Units	Test Conditions
		Min	Max		
t_{AW}	Address Stable Before $\overline{WR} \downarrow$	0		ns	
t_{WA}	Address Hold Time After $\overline{WR} \uparrow$	20		ns	Ports A & B
		20		ns	Port C
t_{WW}	\overline{WR} Pulse Width	100		ns	
t_{DW}	Data Setup Time Before $\overline{WR} \uparrow$	100		ns	
t_{WD}	Data Hold Time After $\overline{WR} \uparrow$	30		ns	Ports A & B
		30		ns	Port C

OTHER TIMINGS

Symbol	Parameter	82C55A-2		Units Conditions	Test
		Min	Max		
t _{WB}	$\overline{WR} = 1$ to Output		350	ns	
t _{IR}	Peripheral Data Before \overline{RD}	0		ns	
t _{HR}	Peripheral Data After \overline{RD}	0		ns	
t _{AK}	\overline{ACK} Pulse Width	200		ns	
t _{ST}	\overline{STB} Pulse Width	100		ns	
t _{PS}	Per. Data Before \overline{STB} High	20		ns	
t _{PH}	Per. Data After \overline{STB} High	50		ns	
t _{AD}	$\overline{ACK} = 0$ to Output		175	ns	
t _{KD}	$\overline{ACK} = 1$ to Output Float	20	250	ns	
t _{WOB}	$\overline{WR} = 1$ to $\overline{OBF} = 0$		150	ns	
t _{AOB}	$\overline{ACK} = 0$ to $\overline{OBF} = 1$		150	ns	
t _{SIB}	$\overline{STB} = 0$ to $IBF = 1$		150	ns	
t _{RIB}	$\overline{RD} = 1$ to $IBF = 0$		150	ns	
t _{RIT}	$\overline{RD} = 0$ to $INTR = 0$		200	ns	
t _{SIT}	$\overline{STB} = 1$ to $INTR = 1$		150	ns	
t _{AIT}	$\overline{ACK} = 1$ to $INTR = 1$		150	ns	
t _{WIT}	$\overline{WR} = 0$ to $INTR = 0$		200	ns	see note 1
t _{RES}	Reset Pulse Width	500		ns	see note 2

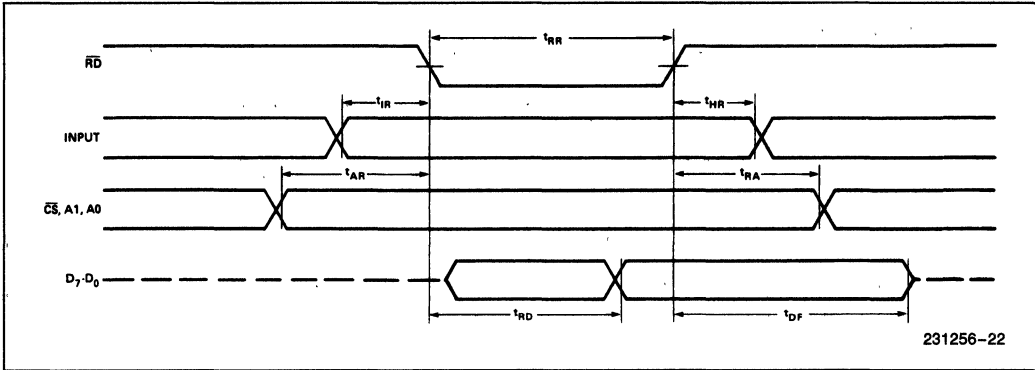
NOTE:

1. $INTR \uparrow$ may occur as early as $\overline{WR} \downarrow$.

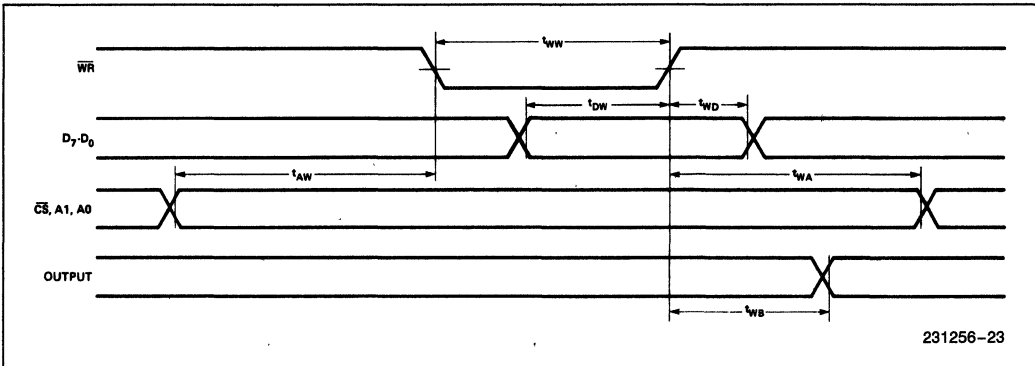
2. Pulse width of initial Reset pulse after power on must be at least 50 μ Sec. Subsequent Reset pulses may be 500 ns minimum.

WAVEFORMS

MODE 0 (BASIC INPUT)

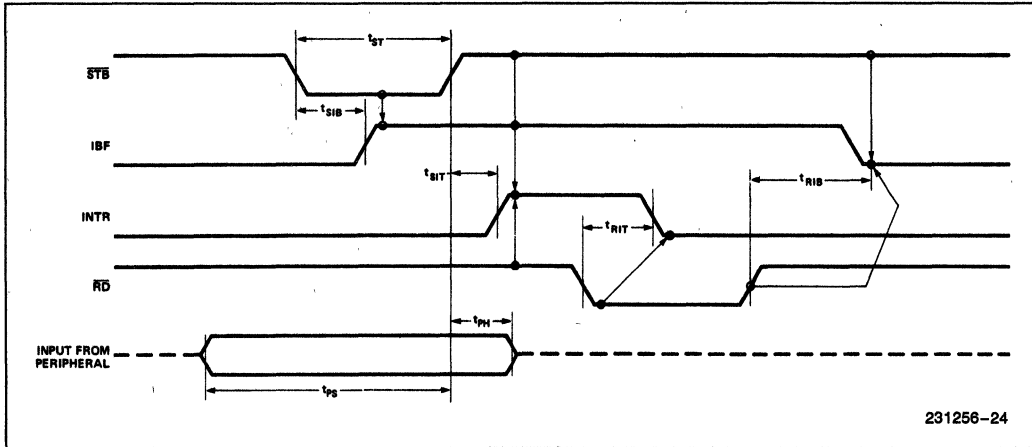


MODE 0 (BASIC OUTPUT)

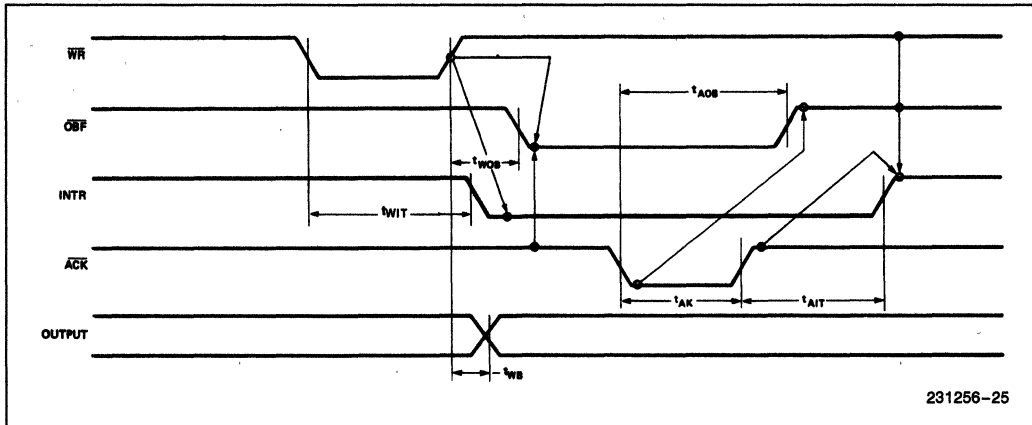


WAVEFORMS (Continued)

MODE 1 (STROBED INPUT)

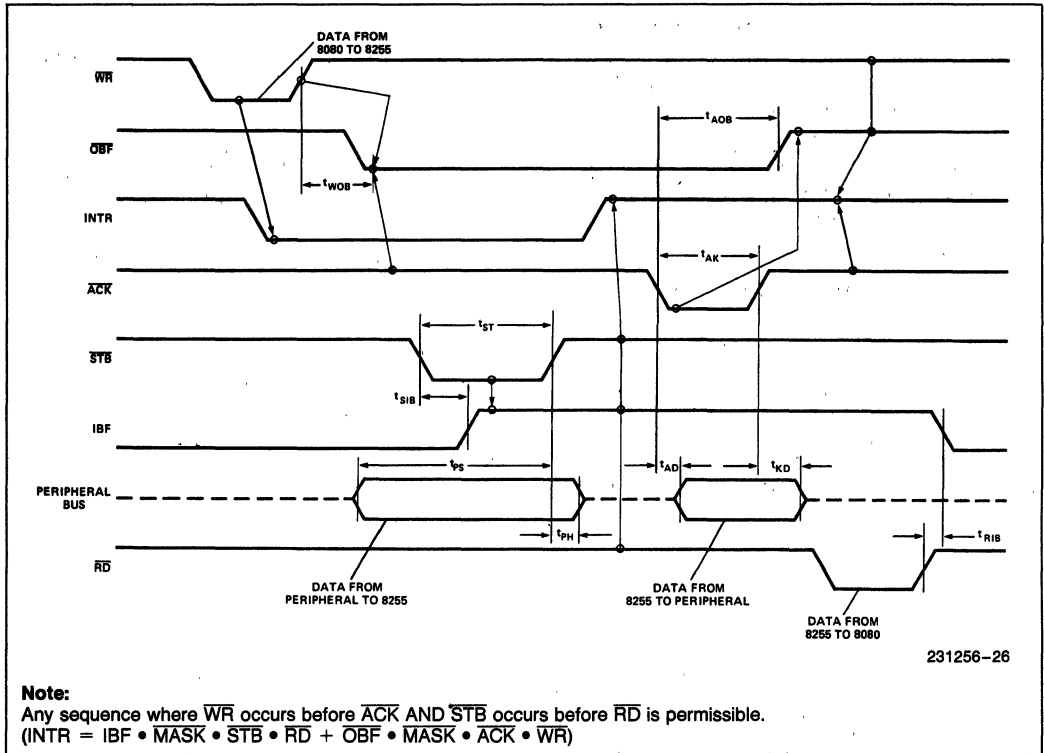


MODE 1 (STROBED OUTPUT)

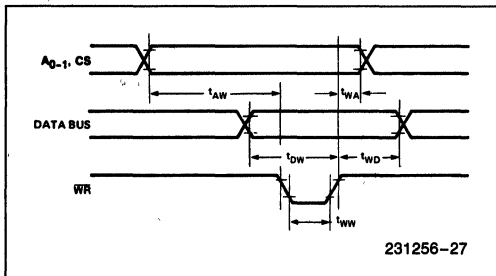


WAVEFORMS (Continued)

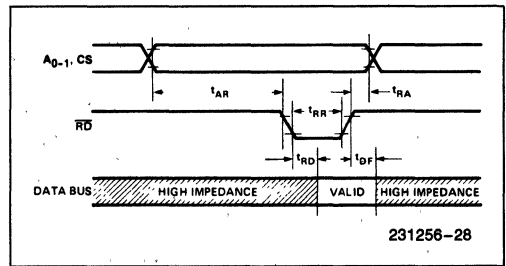
MODE 2 (BIDIRECTIONAL)



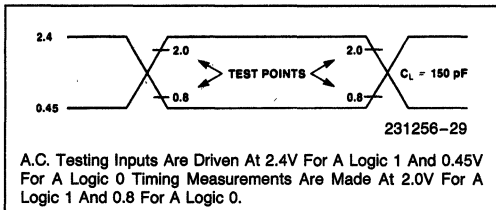
WRITE TIMING



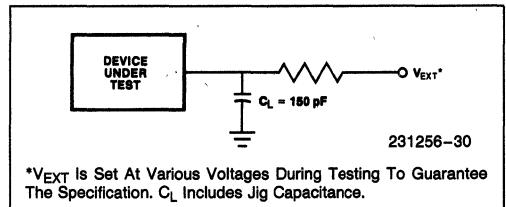
READ TIMING



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



8256AH MULTIFUNCTION MICROPROCESSOR SUPPORT CONTROLLER

- Programmable Serial Asynchronous Communications Interface for 5-, 6-, 7-, or 8-Bit Characters, 1, 1½, or 2 Stop Bits, and Parity Generation
 - On-Board Baud Rate Generator Programmable for 13 Common Baud Rates up to 19.2 KBits/Second, or an External Baud Clock Maximum of 1M Bit/Second
 - Five 8-Bit Programmable Timer/Counters; Four Can Be Cascaded to Two 16-Bit Timer/Counters
- Two 8-Bit Programmable Parallel I/O Ports; Port 1 Can Be Programmed for Port 2 Handshake Controls and Event Counter Inputs
 - Eight-Level Priority Interrupt Controller Programmable for 8085 or IAPX 86, IAPX 88 Systems and for Fully Nested Interrupt Capability
 - Programmable System Clock to 1 ×, 2 ×, 3 ×, or 5 × 1.024 MHz

The Intel® 8256AH Multifunction Universal Asynchronous Receiver-Transmitter (MUART) combines five commonly used functions into a single 40-pin device. It is designed to interface to the 8086/88, iAPX 186/188, and 8051 to perform serial communications, parallel I/O, timing, event counting, and priority interrupt functions. All of these functions are fully programmable through nine internal registers. In addition, the five timer/counters and two parallel I/O ports can be accessed directly by the microprocessor.

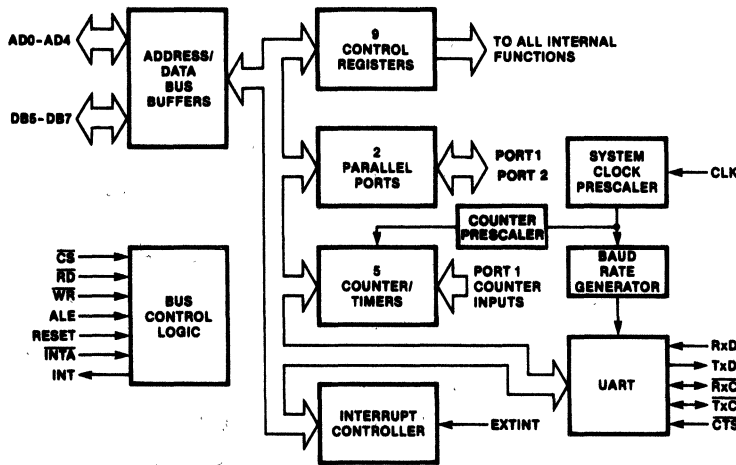


Figure 1. MUART Block Diagram

230759-1

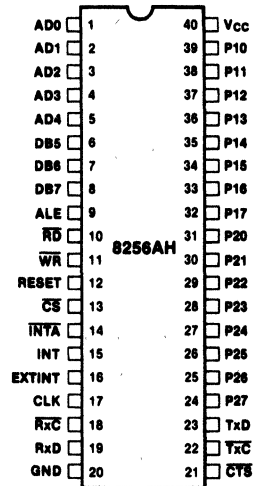


Figure 2. MUART Pin Configuration

230759-2

Table 1. Pin Description

Symbol	Pin	Type	Name and Function
AD0-AD4 DB5-DB7	1-5 6-8	I/O	ADDRESS/DATA: Three-state address/data lines which interface to the lower 8 bits of the microprocessor's multiplexed address/data bus. The 5-bit address is latched on the falling edge of ALE. In the 8-bit mode, AD0-AD3 are used to select the proper register, while AD1-AD4 are used in the 16-bit mode. AD4 in the 8-bit mode is ignored as an address, while AD0 in the 16-bit mode is used as a second chip select, active low.
ALE	9	I	ADDRESS LATCH ENABLE: Latches the 5 address lines on AD0-AD4 and \overline{CS} on the falling edge.
\overline{RD}	10	I	READ CONTROL: When this signal is low, the selected register is gated onto the data bus.
\overline{WR}	11	I	WRITE CONTROL: When this signal is low, the value on the data bus is written into the selected register.
RESET	12	I	RESET: An active high pulse on this pin forces the chip into its initial state. The chip remains in this state until control information is written.
\overline{CS}	13	I	CHIP SELECT: A low on this signal enables the MUART. It is latched with the address on the falling edge of ALE, and \overline{RD} and \overline{WR} have no effect unless \overline{CS} was latched low during the ALE cycle.
\overline{INTA}	14	I	INTERRUPT ACKNOWLEDGE: If the MUART has been enabled to respond to interrupts, this signal informs the MUART that its interrupt request is being acknowledged by the microprocessor. During this acknowledgement the MUART puts an $RSTn$ instruction on the data bus for the 8-bit mode or a vector for the 16-bit mode.
INT	15	O	INTERRUPT REQUEST: A high signals the microprocessor that the MUART needs service.
EXTINT	16	I	EXTERNAL INTERRUPT: An external device can request interrupt service through this input. The input is level sensitive (high), therefore it must be held high until an \overline{INTA} occurs or the interrupt address register is read.
CLK	17	I	SYSTEM CLOCK: The reference clock for the baud rate generator and the timers.
RxC	18	I/O	RECEIVE CLOCK: If the baud rate bits in the Command Register 2 are all 0, this pin is an input which clocks serial data into the RxD pin on the rising edge of RxC. If baud rate bits in Command Register 2 are programmed from 1-0FH, this pin outputs a square wave whose rising edge indicates when the data on RxD is being sampled. This output remains high during start, stop, and parity bits.
RxD	19	I	RECEIVE DATA: Serial data input.
GND	20	PS	GROUND: Power supply and logic ground reference.

Table 1. Pin Description (Continued)

Symbol	Pin	Type	Name and Function
CTS	21	I	CLEAR TO SEND: This input enables the serial transmitter. If 1, 1.5, or 2 stop bits are selected CTS is level sensitive. As long as CTS is low, any character loaded into the transmitter buffer register will be transmitted serially. A single negative going pulse causes the transmission of a single character previously loaded into the transmitter buffer register. If a baud rate from 1-0FH is selected, CTS must be low for at least $\frac{1}{32}$ of a bit, or it will be ignored. If the transmitter buffer is empty, this pulse will be ignored. If this pulse occurs during the transmission of a character up to the time where $\frac{1}{2}$ the first (or only) stop bit is sent out, it will be ignored. If it occurs afterwards, but before the end of the stop bits, the next character will be transmitted immediately following the current one. If CTS is still high when the transmitter register is sending the last stop bit, the transmitter will enter its idle state until the next high-to-low transition on CTS occurs. If 0.75 stop bits is chosen, the CTS input is edge sensitive. A negative edge on CTS results in the immediate transmission of the next character. The length of the stop bits is determined by the time interval between the beginning of the first stop bit and the next negative edge on CTS. A high-to-low transition has no effect if the transmitter buffer is empty or if the time interval between the beginning of the stop bit and next negative edge is less than 0.75 bits. A high or a low level or a low-to-high transition has no effect on the transmitter for the 0.75 stop bit mode.
TxC	22	I/O	TRANSMIT CLOCK: If the baud rate bits in command register 2 are all set to 0, this input clocks data out of the transmitter on the falling edge. If baud rate bits are programmed for 1 or 2, this input permits the user to provide a $32\times$ or $64\times$ clock which is used for the receiver and transmitter. If the baud rate bits are programmed for 3-0FH, the internal transmitter clock is output. As an output it delivers the transmitter clock at the selected bit rate. If $\frac{1}{2}$ or 0.75 stop bits are selected, the transmitter divider will be asynchronously reset at the beginning of each start bit, immediately causing a high-to-low transition on TxC. TxC makes a high-to-low transition at the beginning of each serial bit, and a low-to-high transition at the center of each bit.
TxD	23	O	TRANSMIT DATA: Serial data output.
P27-P20	24-31	I/O	PARALLEL I/O PORT 2: Eight bit general purpose I/O port. Each nibble (4 bits) of this port can be either an input or an output. The outputs are latched whereas the input signals are not. Also, this port can be used as an 8-bit input or output port when using the two-wire handshake. In the handshake mode both inputs and outputs are latched.
P17-P10	32-39	I/O	PARALLEL I/O PORT 1: Each pin can be programmed as an input or an output to perform general purpose I/O. All outputs are latched whereas inputs are not. Alternatively these pins can serve as control pins which extend the functional spectrum of the chip.
V _{CC}	40	PS	POWER: +5V power supply.

FUNCTIONAL DESCRIPTION

The 8256AH Multi-Function Universal Asynchronous Receiver-Transmitter (MUART) combines five commonly used functions into a single 40-pin device. The MUART performs asynchronous serial communications, parallel I/O, timing, event counting, and interrupt control. For detailed application information, see Intel AP Note #153, Designing with the 8256.

Serial Communications

The serial communications portion of the MUART contains a full-duplex asynchronous receiver-transmitter (UART). A programmable baud rate generator is included on the MUART to permit a variety of operating speeds without external components. The UART can be programmed by the CPU for a variety of character sizes, parity generation and detection, error detection, and start/stop bit handling. The receiver checks the start and stop bits in the center of the bit, and a break halts the reception of data. The transmitter can send breaks and can be controlled by an external enable pin.

Parallel I/O

The MUART includes 16 bits of general purpose parallel I/O. Eight bits (Port 1) can be individually changed from input to output or used for special I/O functions. The other eight bits (Port 2) can be used as nibbles (4 bits) or as bytes. These eight bits also include a handshaking capability using two pins on Port 1.

Counter/Timers

There are five 8-bit counter/timers on the MUART. The timers can be programmed to use either a 1 kHz or 16 kHz clock generated from the system clock. Four of the 8-bit counter/timers can be cascaded to two 16-bit counter/timers, and one of the 8-bit counter/timers can be reset to its initial value by an external signal.

Interrupts

An eight-level priority interrupt controller can be configured for fully nested or normal interrupt priority. Seven of the eight interrupts service functions on the MUART (counter/timers, UART), and one external interrupt is provided which can be used for a particular function or for chaining interrupt controllers or more MUARTs. The MUART will support

8085 and 8086/88 systems with direct interrupt vectoring, or the MUART can be polled to determine the cause of the interrupt. If additional interrupt control capability is needed, the MUART's interrupt controller can be cascaded into another MUART, into an Intel 8259A Programmable Interrupt Controller, or into the interrupt controller of the iAPX 186/188 High-Integration Microprocessor.

INITIALIZATION

In general the MUART's functions are independent of each other and only the registers and bits associated with a particular function need to be initialized, not the entire chip. The command sequence is arbitrary since every register is directly addressable; however, **Command Byte 1** must be loaded first. To put the device into a fully operational condition, it is necessary to write the following commands:

- Command byte 1
- Command byte 2
- Command byte 3
- Mode byte
- Port 1 control
- Set Interrupts

The modification register may be loaded if required for special applications; normally this operation is not necessary. The MUART should be reset before initialization. (Either a hardware or a software reset will do.)

INTERFACING

This section describes the hardware interface between the 8256 MUART and the 80186 microprocessor. Figure 3 displays the block diagram for this interface. The MUART can be interfaced to many other microprocessors using these basic principles.

In all cases the 8256 will be connected directly to the CPU's multiplexed address/data bus. If latches or data bus buffers are used in a system, the MUART should be on the microprocessor side of the address/data bus. The MUART latches the address internally on the falling edge of ALE. The address consists of Chip Select (CS) and four address lines. For 8-bit microprocessors, AD0-AD3 are the address lines. For 16-bit microprocessors, AD1-AD4 are the address lines; AD0 is used as a second chip select which is active low. Since chip select is internally latched along with the address, it does not have to remain active during the entire instruction cycle. As long as the chip select setup and hold times are met, it can be derived from multiplexed ad-

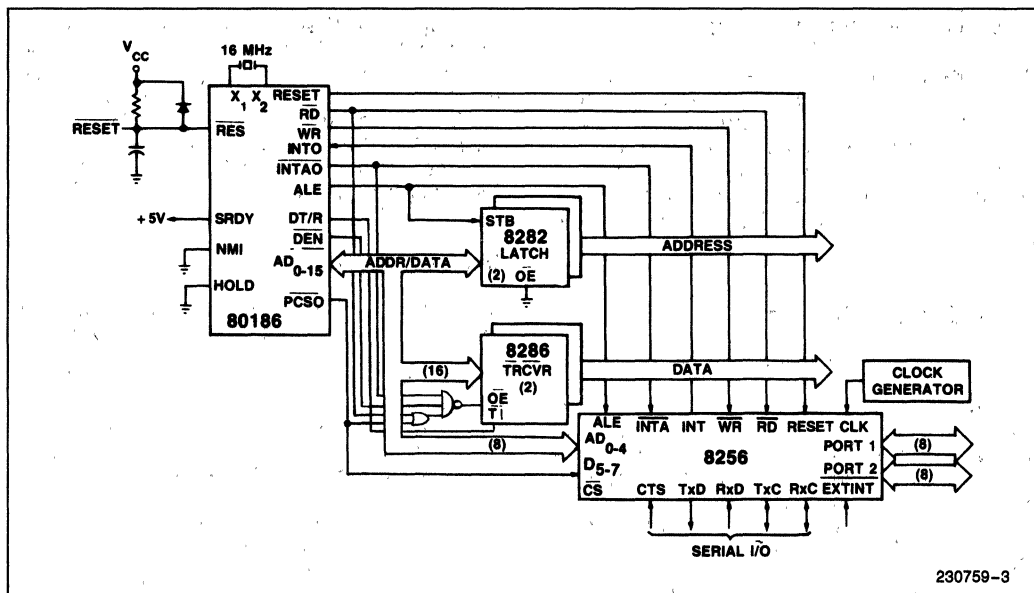


Figure 3. 80186/8256 Interface

dress/data lines or multiplexed address/status lines. When the 8256 is in the 16-bit mode, A0 serves as a second chip select. As a result the MUART's internal registers will all have even addresses since A0 must be zero to select the device. Normally the MUART will be placed on the lower data byte. If the MUART is placed on the upper data byte, the internal registers will be 512 address locations apart and the chip would occupy an 8k word address space.

DESCRIPTION OF THE REGISTERS

The following section will provide a description of the registers and define the bits within the registers where appropriate. Table 2 lists the registers and their addresses.

Command Register 1

L1	L0	S1	S0	BRKI	BITI	8086	FRQ
(OR)				(OW)			

FRQ—TIMER FREQUENCY SELECT

This bit selects between two frequencies for the five timers. If FRQ = 0, the timer input frequency is 16 kHz (62.5 μs). If FRQ = 1, the timer input frequency is 1 kHz (1 ms). The selected clock frequen-

cy is shared by all the counter/timers enabled for timing; thus, all timers must run with the same time base.

8086—8086 MODE ENABLE

This bit selects between 8085 mode and 8086/8088 mode. In 8085 mode (8086 = 0), A0 to A3 are used to address the internal registers, and an RSTn instruction is generated in response to the first INTA. In 8086 mode (8086 = 1), A1 to A4 are used to address the internal registers, and A0 is used as an extra chip select (A0 must equal zero to be enabled). The response to INTA is for 8086 interrupts where the first INTA is ignored, and an interrupt vector (40H to 47H) is placed on the bus in response to the second INTA.

BITI—INTERRUPT ON BIT CHANGE

This bit selects between one of two interrupt sources on Priority Level 1, either Counter/Timer 2 or Port 1 P17 interrupt. When this bit equals 0, Counter/Timer 2 will be mapped into Priority Level 1. If BITI equals 0 and Level 1 interrupt is enabled, a transition from 1 to 0 in Counter/Timer 2 will generate an interrupt request on Level 1. When BITI equals 1, Port 1 P17 external edge triggered interrupt source is mapped into Priority Level 1. In this case if Level 1 is enabled, a low-to-high transition on P17 generates an interrupt request on Level 1.

Table 2. MUART Registers

Read Registers								Write Registers											
8085 Mode: AD3 AD2 AD1 AD0								8086 Mode: AD4 AD3 AD2 AD1											
L1	L0	S1	S0	BRKI	BITI	8086	FRQ	0	0	0	0	L1	L0	S1	S0	BRKI	BITI	8086	FRQ
Command 1								Command 1											
PEN	EP	C1	C0	B3	B2	B1	B0	0	0	0	1	PEN	EP	C1	C0	B3	B2	B1	B0
Command 2								Command 2											
0	RxE	IAE	NIE	0	SBRK	TBRK	0	0	0	1	0	SET	RxE	IAE	NIE	END	SBRK	TBRK	RST
Command 3								Command 3											
T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0	0	0	1	1	T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0
Mode								Mode											
P17	P16	P15	P14	P13	P12	P11	P10	0	1	0	0	P17	P16	P15	P14	P13	P12	P11	P10
Port 1 Control								Port 1 Control											
L7	L6	L5	L4	L3	L2	L1	L0	0	1	0	1	L7	L6	L5	L4	L3	L2	L1	L0
Interrupt Enable								Set Interrupts											
D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	0	L7	L6	L5	L4	L3	L2	L1	L0
Interrupt Address								Reset Interrupts											
D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	1	D7	D6	D5	D4	D3	D2	D1	D0
Receiver Buffer								Transmitter Buffer											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
Port 1								Port 1											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	0	1	D7	D6	D5	D4	D3	D2	D1	D0
Port 2								Port 2											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Timer 1								Timer 1											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	1	1	D7	D6	D5	D4	D3	D2	D1	D0
Timer 2								Timer 2											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	0	0	D7	D6	D5	D4	D3	D2	D1	D0
Timer 3								Timer 3											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	0	1	D7	D6	D5	D4	D3	D2	D1	D0
Timer 4								Timer 4											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Timer 5								Timer 5											
INT	RBF	TBE	TRE	BD	PE	OE	FE	1	1	1	1	0	RS4	RS3	RS2	RS1	RS0	TME	DSC
Status								Modification											

BRKI—BREAK-IN DETECT ENABLE

If this bit equals 0, Port 1 P16 is a general purpose I/O port. When BRKI equals 1, the Break-In Detect feature is enabled on Port 1 P16. A Break-In condition is present on the transmission line when it is forced to the start bit voltage level by the receiving station. Port 1 P16 must be connected externally to the transmission line in order to detect a Break-In. A Break-In is polled by the MUART during the transmission of the last or only stop bit of a character.

A Break-In Detect is OR-ed with Break Detect in Bit 3 of the Status Register. The distinction can be made through the interrupt controller. If the transmit and receive interrupts are enabled, a Break-In will generate an interrupt on Level 5, the transmit interrupt, while Break will generate an interrupt on Level 4, the receive interrupt.

S0, S1—STOP BIT LENGTH

S1	S0	Stop Bit Length
0	0	1
0	1	1.5
1	0	2
1	1	0.75

The relationship of the number of stop bits and the function of input CTS is discussed in the Pin Description section under "CTS".

L0, L1—CHARACTER LENGTH

L1	L0	Character Length
0	0	8
0	1	7
1	0	6
1	1	5

Command Register 2

PEN	EP	C1	C0	B3	B2	B1	B0
(1R)				(1W)			

Programming bits 0 . . . 3 with values from 3H to FH enables the internal baud rate generator as a common clock source for the transmitter and receiver and determines its divider ratio.

Programming bits 0 . . . 3 with values of 1H or 2H enables input TxC as a common clock source for the transmitter and receiver. The external clock must

provide a frequency of either 32× or 64× the baud rate. The data transmission rates range from 0 . . . 32 Kbaud.

If bits 0 . . . 3 are set to 0, separate clocks must be input to pin RxC for the receiver and pin TxC for the transmitter. Thus, different baud rates can be used for transmission and reception. In this case, pre-scalers are disabled and the input serial clock frequency must match the baud rate. The input serial clock frequency can range from 0 MHz to 1.024 MHz.

B0, B1, B2, B3—BAUD RATE SELECT

These four bits select the bit clock's source, sampling rate, and serial rate for the internal baud rate generator.

B3	B2	B1	B0	Baud Rate	Sampling Rate
0	0	0	0	TxC, RxC	1
0	0	0	1	TxC/64	64
0	0	1	0	TxC/32	32
0	0	1	1	19200	32
0	1	0	0	9600	64
0	1	0	1	4800	64
0	1	1	0	2400	64
0	1	1	1	1200	64
1	0	0	0	600	64
1	0	0	1	300	64
1	0	1	0	200	64
1	0	1	1	150	64
1	1	0	0	110	64
1	1	0	1	100	64
1	1	1	0	75	64
1	1	1	1	50	64

The following table gives an overview of the function of pins TxC and RxC:

Bits 3 to 0 (Hex.)	TxC	RxC
0	Input: 1 × baud rate clock for the transmitter	Input: 1 × baud rate clock for the receiver
1, 2	Input: 32 × or 64 × baud rate for transmitter and receiver	Output: receiver bit clock with a low-to-high transition at data bit sampling time. Otherwise: high level
3 to F	Output: baud rate clock of the transmitter	Output: as above

As an output, RxC outputs a low-to-high transition at sampling time of every data bit of a character. Thus, data can be loaded, e.g., into a shift register externally. The transition occurs only if data bits of a character are present. It does not occur for start, parity, and stop bits (RxC = high).

As an output, TxC outputs the internal baud rate clock of the transmitter. There will be a high-to-low transition at every beginning of a bit.

C0, C1—SYSTEM CLOCK PRESCALER (BITS 4, 5)

Bits 4 and 5 define the system clock prescaler divider ratio. The internal operating frequency of 1.024 MHz is derived from the system clock.

C1	C0	Divider Ratio	Clock at Pin CLK
0	0	5	5.12 MHz
0	1	3	3.072 MHz
1	0	2	2.048 MHz
1	1	1	1.024 MHz

EP—EVEN PARITY (BIT 6)

EP = 0: Odd parity
 EP = 1: Even parity

PEN—PARITY ENABLE (BIT 7)

Bit 7 enables parity generation and checking.

PEN = 0: No parity bit
 PEN = 1: Even parity bit

The parity bit according to Command Register 2 bit 6 (see above) is inserted between the last data bit of a character and the first or only stop bit. The parity bit is checked during reception. A false parity bit generates an error indication in the Status Register and an Interrupt Request on Level 4.

Command Register 3

SET	RxE	IAE	NIW	END	SBRK	TBRK	RST
(2R)				(2W)			

Command Register 3 is different from the first two registers because it has a bit set/reset capability. Writing a byte with Bit 7 high sets any bits which were also high. Writing a byte with Bit 7 low resets

any bits which were high. If any bit 0–6 is low, no change occurs to that bit. When command Register 3 is read, bits 0, 3, and 7 will always be zero.

RST—RESET

If RST is set, the following events occur:

1. All bits in the Status Register except bits 4 and 5 are cleared, and bits 4 and 5 are set.
2. The Interrupt Enable, Interrupt Request, and Interrupt Service Registers are cleared. Pending requests and indications for interrupts in service will be cancelled. Interrupt signal INT will go low.
3. The receiver and transmitter are reset. The transmitter goes idle (TxD is high), and the receiver enters start bit search mode.
4. If Port 2 is programmed for handshake mode, IBF and OBF are reset high.

RST does *not* alter ports, data registers or command registers, but it halts any operation in progress. RST is automatically cleared.

RST = 0 has no effect. The reset operation triggered by Command Register 3 is a subset of the hardware reset.

TBRK—TRANSMIT BREAK

The transmission data output TxD will be set low as soon as the transmission of the previous character has been finished. It stays low until TBRK is cleared. The state of CTS is of no significance for this operation. As long as break is active, data transfer from the Transmitter Buffer to the Transmitter Register will be inhibited. As soon as TBRK is reset, the break condition will be deactivated and the transmitter will be re-enabled.

SBRK—SINGLE CHARACTER BREAK

This causes the transmitter data to be set low for one character including start bit, data bits, parity bit, and stop bits. SBRK is automatically cleared when time for the last data bit has passed. It will start after the character in progress completes, and will delay the next data transfer from the Transmitter Buffer to the Transmitter Register until TxD returns to an idle (marking) state. If both TBRK and SBRK are set, break will be set as long as TBRK is set, but SBRK will be cleared after one character time of break. If SBRK is set again, it remains set for another character. The user can send a definite number of break characters in this manner by clearing TBRK after setting SBRK for the last character time.

END—END OF INTERRUPT

If fully nested interrupt mode is selected, this bit resets the currently served interrupt level in the Interrupt Service Register. *This command must occur at the end of each interrupt service routine during fully nested interrupt mode.* END is automatically cleared when the Interrupt Service Register (internal) is cleared. END is ignored if nested interrupts are not enabled.

NIE—NESTED INTERRUPT ENABLE

When NIE equals 1, the interrupt controller will operate in the nested interrupt mode. When NIE equals 0, the interrupt controller will operate in the normal interrupt mode. Refer to the "Interrupt controller" section of AP-153 under "Normal Mode" and "Nested Mode" for a detailed description of these operations.

IAE—INTERRUPT ACKNOWLEDGE ENABLE

This bit enables an automatic response to \overline{INTA} . The particular response is determined by the 8086 bit in Command Register 1.

RxE—RECEIVE ENABLE

This bit enables the serial receiver and its associated status bits in the status register. If this bit is reset, the serial receiver will be disabled and the receive status bits will not be updated.

Note that the detection of break characters remains enabled while the receiver is disabled; i.e., Status Register Bit 3 (BD) will be set while the receiver is disabled whenever a break character has been recognized at the receive data input RxD.

SET—BIT SET/RESET

If this bit is high during a write to Command Register 3, then any bit marked by a high will set. If this bit is low, then any bit marked by a high will be cleared.

MODE REGISTER

T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0
(3R)			(3W)				

If test mode is selected, the output from the internal baud rate generator is placed on bit 4 of Port 1 (pin 35).

To achieve this, it is necessary to program bit 4 of Port 1 as an output (Port 1 Control Register Bit P14 = 1), and to program Command Register 2 bits B3-B0 with a value $\geq 3H$.

P2C2, P2C1, P2C0—PORT 2 CONTROL

P2C2	P2C1	P2C0	Mode	Direction	
				Upper	Lower
0	0	0	Nibble	Input	Input
0	0	1	Nibble	Input	Output
0	1	0	Nibble	Output	Input
0	1	1	Nibble	Output	Output
1	0	0	Byte Handshake	Input	
1	0	1	Byte Handshake	Output	
1	1	0	DO NOT USE		
1	1	1	Test		

NOTE:

If Port 2 is operating in handshake mode, Interrupt Level 7 is not available for Timer 5. Instead it is assigned to Port 2 handshaking.

CT2, CT3—COUNTER/TIMER MODE

Bit 3 and 4 defines the mode of operation of event counter/timers 2 and 3 regardless of its use as a single unit or as a cascaded one.

If CT2 or CT3 are high, then counter/timer 2 or 3 respectively is configured as an event counter on bit 2 or 3 respectively of Port 1 (pins 37 or 36). The event counter decrements the count by one on each low-to-high transition of the external input. If CT2 or CT3 is low, then the respective counter/timer is configured as a timer and the Port 1 pins are used for parallel I/O.

T5C—TIMER 5 CONTROL

If T5C is set, then Timer 5 can be preset and started by an external signal. Writing to the Timer 5 register loads the Timer 5 save register and stops the timer. A high-to-low transition on bit 5 of Port 1 (pin 34) loads the timer with the saved value and starts the timer. The next high-to-low transition on pin 34 re-triggers the timer by reloading it with the initial value and continues timing.

Following a hardware reset, the save register is reset to 00H and both clock and trigger inputs are disabled. Transferring an instruction with T5C = 1 enables the trigger input; the save register can now be loaded with an initial value. The first trigger pulse causes the initial value to be loaded from the save register and enables the counter to count down to zero.

When the timer reaches zero it issues an interrupt request, disables its interrupt level and continues

counting. A subsequent high-to-low transition on pin 5 resets Timer 5 to its initial value. For another timer interrupt, the Timer 5 interrupt enable bit must be set again.

T35, T24—CASCADE TIMERS

These two bits cascade Timers 3 and 5 or 2 and 4.

Timers 2 and 3 are the lower bytes, while Timers 4 and 5 are the upper bytes. If T5C is set, then both Timers 3 and 5 can be preset and started by an external pulse.

When a high-to-low transition occurs, Timer 5 is preset to its saved value, but Timer 3 is always preset to all ones. If either CT2 or CT3 is set, then the corresponding timer pair is a 16-bit event counter.

A summary of the counter/timer control bits is given in Table 3.

NOTE:

Interrupt levels assigned to single counters are partly not occupied if event counters/timers are cascaded. Level 2 will be vacated if event counters/timers 2 and 4 are cascaded. Likewise, Level 7 will be vacated if event counters/timers 3 and 5 are cascaded.

Single event counters/timers generate an interrupt request on the transition from 01H to 00H, while cascaded ones generate it on the transition from 0001H to 0000H.

Port 1 Control Register

P17	P16	P15	P14	P13	P12	P11	P10
(4R)				(4W)			

Each bit in the Port 1 Control Register configures the direction of the corresponding pin. If the bit is high, the pin is an output, and if it is low the pin is an input. Every Port 1 pin has another function which is controlled by other registers. If that special function is disabled, the pin functions as a general I/O pin as specified by this register. The special functions for each pin are described below.

Port 10, 11—HANDSHAKE CONTROL

If byte handshake control is enabled for Port 2 by the Mode Register, then Port 10 is programmed as STB/ACK handshake control input, and Port 11 is programmed as IBF/OBF handshake control output.

If byte handshake mode is enabled for output on Port 2 OBF indicates that a character has been loaded into the Port 2 output buffer. When an external

Table 3. Event Counters/Timers Mode of Operation

Event Counter/Timer	Function	Programming (Mode Word)	Clock Source
1	8-bit timer	—	Internal clock
2	8-bit timer	T24 = 0, CT2 = 0	Internal clock
	8-bit event counter	T24 = 0, CT2 = 1	P12 pin 37
2	8-bit timer	T35 = 0, CT3 = 0	Internal clock
	8-bit event counter	T35 = 0, CT3 = 1	P13 pin 36
4	8-bit timer	T24 = 0	Internal clock
5	8-bit timer, normal mode	T35 = 0, T5C = 0	Internal clock
	8-bit timer, retriggerable mode	T35 = 0, T5C = 1	Internal clock
2 and 4 cascaded	16-bit timer	T24 = 1, CT2 = 0	Internal clock
	16-bit event counter	T24 = 1, CT2 = 1	P12 pin 37
3 and 5 cascaded	16-bit timer, normal mode	T35 = 1, T5C = 0, CT3 = 0	Internal clock
	16-bit event counter, normal mode	T35 = 1, T5C = 0, CT3 = 1	P13 pin 36
	16-bit timer, retriggerable mode	T35 = 1, T5C = 1, CT3 = 0	Internal clock
	16-bit event counter, retriggerable mode	T35 = 1, T5C = 1, CT3 = 1	P13 pin 36

device reads the data, it acknowledges this operation by driving \overline{ACK} low. \overline{OBF} is set low by writing to Port 2 and is reset by \overline{ACK} .

If byte handshake mode is enabled for input on Port 2, \overline{STB} is an input. \overline{IBF} is driven low after \overline{STB} goes low. On the rising edge of \overline{STB} the data from Port 2 is latched.

\overline{IBF} is reset high when Port 2 is read.

PORT 12, 13—COUNTER 2, 3 INPUT

If Timer 2 or Timer 3 is programmed as an event counter by the Mode Register, then Port 12 or Port 13 is the counter input for Event Counter 2 or 3, respectively.

PORT 14—BAUD RATE GENERATOR OUTPUT CLOCK

If test mode is enabled by the Mode Register and Command Register 2 baud rate select is greater than 2, then Port 14 is an output from the internal baud rate generator.

P14 in Port 1 control register must be set to 1 for the baud rate generator clock to be output. The baud rate generator clock is $64 \times$ the serial bit rate except at 19.2 Kbps when it is $32 \times$ the bit rate.

PORT 15—TIMER 5 TRIGGER

If T5C is set in the Mode Register enabling a retriggerable timer, then Port 15 is the input which starts and reloads Timer 5.

A high-to-low transition on P15 (Pin 34) loads the timer with the slave register and starts the timer.

PORT 16—BREAK-IN DETECT

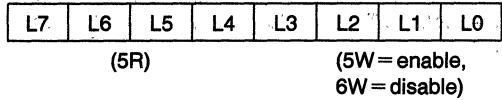
If Break-In Detect is enabled by BRKI in Command Register 1, then this input is used to sense a Break-In. If Port 16 is low while the serial transmitter is sending the last stop bit, then a Break-In condition is signaled.

PORT 17—PORT INTERRUPT SOURCE

If BITI in Command Register 1 is set, then a low-to-high transition on Port 17 generates an interrupt request on Priority Level 1.

Port 17 is edge triggered.

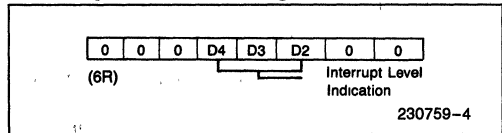
Interrupt Enable Register



Interrupts are enabled by writing to the Set Interrupts Register (5W). Interrupts are disabled by writing to the Reset Interrupts Register (6W). Each bit set by the Set Interrupts Register (5W) will enable that level interrupt, and each bit set in the Reset Interrupts Register (6W) will disable that level interrupt. The user can determine which interrupts are enabled by reading the Interrupt Enable Register (5R).

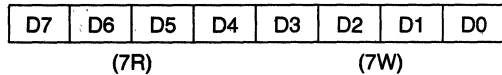
Priority	Source
Highest	L0 Timer 1
	L1 Timer 2 or Port Interrupt
	L2 External Interrupt (EXTINT)
	L3 Timer 3 or Timers 3 & 5
	L4 Receive Interrupt
	L5 Transmitter Interrupt
	L6 Timer 4 or Timers 2 & 4
Lowest	L7 Timer 5 or Port 2 Handshaking

Interrupt Address Register



Reading the interrupt address register transfers an identifier for the currently requested interrupt level on the system data bus. This identifier is the number of the interrupt level multiplied by 4. It can be used by the CPU as an offset address for interrupt handling. Reading the interrupt address register has the same effect as a hardware interrupt acknowledge \overline{INTA} ; it clears the interrupt request pin (INT) and indicates an interrupt acknowledgement to the interrupt controller.

Receiver and Transmitter Buffer



Both the receiver and transmitter in the MUART are double buffered. This means that the transmitter and receiver have a shift register and a buffer register. The buffer registers are directly addressable by reading or writing to register seven. After the receiver buffer is full, the RBF bit in the status register is set.

Reading the receive buffer clears the RBF status bit. The transmit buffer should be written to only if the TBE bit in the status register is set. Bytes written to the transmit buffer are held there until the transmit shift register is empty, assuming CTS is low. If the transmit buffer and shift register are empty, writing to the transmit buffer immediately transfers the byte to the transmit shift register. If a serial character length is less than 8 bits, the unused most significant bits are set to zero when reading the receive buffer, and are ignored when writing to the transmit buffer.

Port 1

D7	D6	D5	D4	D3	D2	D1	D0
(8R)				(8W)			

Writing to Port 1 sets the data in the Port 1 output latch. Writing to an input pin does not affect the pin, but the data is stored and will be output if the direction of the pin is changed later. If the pin is used as a control signal, the pin will not be affected, but the data is stored. Reading Port 1 transfers the data in Port 1 onto the data bus.

Port 2

D7	D6	D5	D4	D3	D2	D1	D0
(9R)				(9W)			

Writing to Port 2 sets the data in the Port 2 output latch. Writing to an input pin does not affect the pin, but it does store the data in the latch. Reading Port 2 puts the input pins onto the bus or the contents of the output latch for output pins.

Timer 1-5

D7	D6	D5	D4	D3	D2	D1	D0
(0A ₁₆ -OE ₁₆ R)				(0A ₁₆ -OE ₁₆ W)			

Reading Timer N puts the contents of the timer onto the data bus. If the counter changes while RD is low, the value on the data bus will not change. If two timers are cascaded, reading the high-order byte will cause the low-order byte to be latched. Reading the low-order byte will unlatch them both. Writing to either timer or decascading them also clears the latch condition. Writing to a timer sets the starting value of that timer. If two timers are cascaded, writing to the high-order byte presets the low-order byte to all ones. Loading only the high-order byte with a value of X leads to a count of X * 256 + 255. Timers count down continuously. If the interrupt is enabled, it occurs when the counter changes from 1 to 0.

The timer/counter interrupts are automatically disabled when the interrupt request is generated.

Status Register

INT	RBF	TBE	TRE	BD	PE	OE	FE
(OF ₁₆ R)							

Reading the status register gates its contents onto the data bus. It holds the operational status of the serial interface as well as the status of the interrupt in INT. The status register can be read at any time. The flags are stable and well defined at all instants.

FE—FRAMING ERROR, TRANSMISSION MODE

Bit 0 can be used in two modes. Normally, FE indicates framing error which can be changed to transmission mode indication by setting the TME bit in the modification register.

If transmission mode is disabled (in Modification Register), then FE indicates a framing error. A framing error is detected during the first stop bit. The error is reset by reading the Status Register or by a chip reset. A framing error does not inhibit the loading of the Receiver Buffer. If RxD remains low, the receiver will assemble the next character. The false stop bit is treated as the next start bit, and no high-to-low transition on RxD is required to synchronize the receiver.

When the TME bit in the Modification Register is set, FE is used to indicate that the transmitter was active during the reception of a character, thus indicating that the character received was transmitted by its own transmitter. FE is reset when the transmitter is not active during the reception of character. Reading the status register will not reset the FE bit in the transmission mode.

OE—OVERRUN ERROR

If the user does not read the character in the Receiver Buffer before the next character is received and transferred to this register, then the OE bit is set. The OE flag is set during the reception of the first stop bit and is cleared when the Status Register is read or when a hardware or software reset occurs. The first character received in this case will be lost.

PE—PARITY ERROR

This bit indicates a parity error has occurred during the reception of a character. A parity error is present

if value of the parity bit in the received character is different from the one expected according to command word 2 bits 6 EP. The parity bit is expected and checked only if it is enabled by command word 2 bit 7 PEN.

A parity error is set during the first stop bit and is reset by reading the Status Register or by a chip reset.

BD—BREAK/BREAK-IN

The BD bit flags whether a break character has been received, or a Break-In condition exists on the transmission line. Command Register 1 Bit 3 (BRKI) enables the Break-In Detect function.

Whenever a break character has been received, Status Register Bit 3 will be set and in addition an interrupt request on Level 4 is generated. The receiver will be idled. It will be started again with the next high-to-low transition at pin RxD.

The break character received will not be loaded into the receiver buffer register.

If Break-In Detection is enabled and a Break-In condition occurs, Status Register Bit 3 will be set and in addition an interrupt request on Level 5 is generated.

The BD status bit will be reset on reading the status register or on a hardware or software reset. For more information on Break/Break-In, refer to the "Serial Asynchronous Communication" section of AP-153 under "Receive Break Detect" and "Break-In Detect."

TRE—TRANSMIT REGISTER EMPTY

When TRE is set the transmit register is empty and an interrupt request is generated on Level 5 if enabled. When TRE equals 0 the transmit register is in the process of sending data. TRE is set by a chip reset and when the last stop bit has left the transmitter. It is reset when a character is loaded into the Transmitter Register. If CTS is low, the Transmitter Register will be loaded during the transmission of the start bit. If CTS is high at the end of a character, TRE will remain high and no character will be loaded into the Transmitter Register until CTS goes low. If the transmitter was inactive before a character is loaded into the Transmitter Buffer, the Transmitter Register will be empty temporarily while the buffer is full. However, the data in the buffer will be transferred to the transmitter register immediately and TRE will be cleared while TBE is set.

TBE—TRANSMITTER BUFFER EMPTY

TBE indicates the Transmitter Buffer is empty and is ready to accept a character. TBE is set by a chip reset or the transfer of data to the Transmitter Register, and is cleared when a character is written to the transmitter buffer. When TBE is set, an interrupt request is generated on Level 5 if enabled.

RBF—RECEIVER BUFFER FULL

RBF is set when the Receiver Buffer has been loaded with a new character during the sampling of the first stop bit. RBF is cleared by reading the receiver buffer or by a chip reset.

INT—INTERRUPT PENDING

The INT bit reflects the state of the INT Pin (Pin 15) and indicates an interrupt is pending. It is reset by INTA or by reading the Interrupt Address Register if only one interrupt is pending and by a chip reset.

FE, OE, PE, RBF, and Break Detect all generate a Level 4 interrupt when the receiver samples the first stop bit. TRE, TBE, and Break-In Detect generate a Level 5 interrupt. TRE generates an interrupt when TBE is set and the Transmitter Register finished transmitting. The Break-In Detect interrupt is issued at the same time as TBE or TRE.

MODIFICATION REGISTER

0	RS4	RS3	RS2	RS1	RS0	TME	DSC
---	-----	-----	-----	-----	-----	-----	-----

(OF₁₆W)

DSC—DISABLE START BIT CHECK

DSC disables the receivers start bit check. In this state the receiver will not be reset if RxD is not low at the center of the start bit.

TME—TRANSMISSION MODE ENABLE

TME enables transmission mode and disables framing error detection. For information on transmission mode see the description of the framing error bit in the status register.

RS0, RS1, RS2, RS3, RS4—RECEIVER SAMPLE TIME

The number in RS_n alters when the receiver samples RxD. The receiver sample time can be modified only if the receiver is *not* clocked by RxC.

NOTE:

The modification register cannot be read. Reading from address 0FH, 8086: 1EH gates the contents of the status register onto the data bus.

A hardware reset (reset, Pin 12) resets all modification register bits to 0, i.e.:

- The start bit check is enabled.
- Status Register Bit 0 (FE) indicates framing error.
- The sampling time of the serial receiver is the bit center.

A software reset (Command Word 3, RST) does not affect the modification register.

Hardware Reset

A reset signal on pin RESET (HIGH level) forces the device 8256 into a well-defined initial state. This state is characterized as follows:

- 1) Command registers 1, 2 and 3, mode register, Port 1 control register, and modification register are reset. Thus, all bits of the parallel interface are set to be inputs and event counters/timers are configured as independent 8-bit timers.
- 2) Status register bits are reset with the exception of bits 4 and 5. Bits 4 and 5 are set indicating that both transmitter register and transmitter buffer register are empty.
- 3) The interrupt mask, interrupt request, and interrupt service register bits are reset and disable all requests. As a consequence, interrupt signal INT IS INACTIVE (LOW).
- 4) The transmit data output is set to the marking state (HIGH) and the receiver section is disabled until it is enabled by Command Register 3 Bit 6.
- 5) The start bit will be checked at sampling time. The receiver will return to start bit search mode if input RxD is not LOW at this time.
- 6) Status Register Bit 0 implies framing error.
- 7) The receiver samples input RxD at bit center.

Reset has no effect on the contents of receiver buffer register, transmitter buffer register, the intermediate latches of parallel ports, and event counters/timers, respectively.

RS4	RS3	RS2	RS1	RS0	Point of Time Between Start of Bit and End of Bit Measured in Steps of 1/32 Bit Length
0	1	1	1	1	1 (Start of Bit)
0	1	1	1	0	2
0	1	1	0	1	3
0	1	1	0	0	4
0	1	0	1	1	5
0	1	0	1	0	6
0	1	0	0	1	7
0	1	0	0	0	8
0	0	1	1	1	9
0	0	1	1	0	10
0	0	1	0	1	11
0	0	1	0	0	12
0	0	0	1	1	13
0	0	0	1	0	14
0	0	0	0	1	15
0	0	0	0	0	16 (Bit center)
1	1	1	1	1	17
1	1	1	1	0	18
1	1	1	0	1	19
1	1	1	0	0	20
1	1	0	1	1	21
1	1	0	1	0	22
1	1	0	0	1	23
1	1	0	0	0	24
1	0	1	1	1	25
1	0	1	1	0	26
1	0	1	0	1	27
1	0	1	0	0	28
1	0	0	1	1	29
1	0	0	1	0	30
1	0	0	0	1	31
1	0	0	0	0	32 (End of Bit)

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage On Any Pin
 with Respect to Ground -0.5V to +7V
 Power Dissipation 1 Watt

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

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5.0\text{V} \pm 10\%$

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_{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_{IL}	Input Leakage		10 -10	μA μA	$V_{IN} = V_{CC}$ $V_{IN} = 0\text{V}$
I_{LO}	Output Leakage		10 -10	μA μA	$V_{OUT} = V_{CC}$ $V_{OUT} = 0.45\text{V}$
I_{CC}	V_{CC} Supply Current		160	mA	

CAPACITANCE $T_A = 25^\circ\text{C}$, $V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	$f_c = 1 \text{ MHz}$
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured Pins Returned to V_{SS}

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5.0\text{V} \pm 10\%$, $\text{GND} = 0\text{V}$

Symbol	Parameter	8256AH		Units
		Min	Max	
BUS PARAMETERS				
t_{LL}	ALE Pulse Width	50		ns
t_{CSL}	$\overline{\text{CS}}$ to ALE Setup Time	0		ns
t_{AL}	Address to ALE Setup Time	20		ns
t_{LA}	Address Hold Time after ALE	25		ns
t_{LC}	ALE to $\overline{\text{RD}}/\overline{\text{WR}}$	20		ns
t_{CC}	$\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{INTA}}$ Pulse Width	200		ns
t_{RD}	Data Valid from $\overline{\text{RD}}$ (1)		120	ns

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5.0\text{V} \pm 10\%$, $\text{GND} = 0\text{V}$ (Continued)

Symbol	Parameter	8256AH		Units
		Min	Max	
BUS PARAMETERS (Continued)				
t_{DF}	Data Float after \overline{RD} (2)		50	ns
t_{DW}	Data Valid to \overline{WR}	150		ns
t_{WD}	Data Valid after \overline{WR}	50		ns
t_{CL}	$\overline{RD}/\overline{WR}$ Control to Latch Enable	25		ns
t_{LDR}	ALE to Data Valid		150	ns
t_{RST}	Reset Pulse Width	300		ns
t_{RV}	Recovery Time between $\overline{RD}/\overline{WR}$	500		ns
TIMER/COUNTER PARAMETERS				
t_{CPI}	Counter Input Cycle Time (P12, P13)	2.2		μs
t_{CPWH}	Counter Input Pulse Width High	1.1		μs
t_{CPWL}	Counter Input Pulse Width Low	1.1		μs
t_{TPI}	Counter Input \uparrow to INT \uparrow at Terminal Count		2.75	μs
t_{TIH}	LOAD Pulse High Time Counter 5	1.1		μs
t_{TIL}	LOAD Pulse Low Time Counter 5	1.1		μs
t_{PP}	Counter 5 Load before Next Clock Pulse on P13	1.1		μs
t_{CR}	External Count Clock \uparrow to \overline{RD} \downarrow to Ensure Clock is Reflected in Count	2.2		μs
t_{RC}	\overline{RD} \uparrow to External Count Clock \uparrow to Ensure Clock is not Reflected in Count	0		ns
t_{CW}	External Count Clock \uparrow to \overline{WR} \uparrow to Ensure Count Written is Not Decrementd	2.2		μs
t_{WC}	\overline{WR} \uparrow to External Count Clock to Ensure Count Written is Decrementd	0		ns
INTERRUPT PARAMETERS				
t_{DEX}	EXTINT \uparrow to INT \uparrow		200	ns
t_{DPI}	Interrupt Request on P17 \uparrow to INT \uparrow		$2t_{CY} + 500$	ns
t_{PI}	Pulse Width of Interrupt Request on P17	$t_{CY} + 100$		ns
t_{HEA}	\overline{INTA} \uparrow or \overline{RD} \uparrow to EXTINT \downarrow	30		ns
t_{HIA}	\overline{INTA} \uparrow or \overline{RD} \uparrow to INT \downarrow		300	μs
SERIAL INTERFACE AND CLOCK PARAMETERS				
t_{CY}	Clock Period	195	1000	ns
t_{CLKH}	Clock High Pulse Width	65		ns
t_{CLKL}	Clock Low Pulse Width	65		ns
t_R	Clock Rise Time		20	ns
t_F	Clock Fall Time		20	ns

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5.0\text{V} \pm 10\%$, $GND = 0\text{V}$ (Continued)

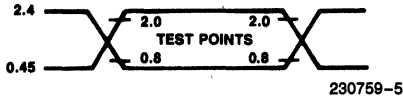
Symbol	Parameter	8256AH		Units
		Min	Max	
SERIAL INTERFACE AND CLOCK PARAMETERS (Continued)				
t_{SCY}	Serial Clock Period (4)	975		ns
t_{SPD}	Serial Clock High (4)	350		ns
t_{SPW}	Serial Clock Low (4)	350		ns
t_{STD}	Internal Status Update Delay from Center of Stop Bit (5)		300	ns
t_{DTX}	$\overline{\text{TxC}}$ to TxD Data Valid		300	ns
t_{IRBF}	INT Delay from Center of First Stop Bit		$2t_{CY} + 500$	ns
t_{ITBE}	INT Delay from Falling Edge of Transmit Clock at End of Start Bit		$2t_{CY} + 500$	ns
t_{CTS}	Pulse Width for Single Character Transmission	(6)		
PARALLEL I/O PORT PARAMETERS				
t_{WP}	$\overline{\text{WR}} \uparrow$ to P1/P2 Data Valid		0	ns
t_{PR}	P1/P2 Data Stable before $\overline{\text{RD}} \downarrow$ (7)	300		ns
t_{RP}	P1/P2 Data Hold Time	50		ns
t_{AK}	$\overline{\text{ACK}}$ Pulse Width	150		ns
t_{ST}	Strobe Pulse Width	t_{SIB}		ns
t_{PS}	Data Setup to $\overline{\text{STB}} \uparrow$	50		ns
t_{PH}	Data Hold after $\overline{\text{STB}} \uparrow$	50		ns
t_{WOB}	$\overline{\text{WR}} \uparrow$ to $\overline{\text{OBF}} \uparrow$		250	ns
t_{AOB}	$\overline{\text{ACK}} \downarrow$ to $\overline{\text{OBF}} \downarrow$		250	ns
t_{SIB}	$\overline{\text{STB}} \downarrow$ to $\overline{\text{IBF}} \downarrow$		250	ns
t_{RI}	$\overline{\text{RD}} \uparrow$ to $\overline{\text{IBF}} \uparrow$		250	ns
t_{SIT}	$\overline{\text{STB}} \uparrow$ to $\overline{\text{INT}} \uparrow$		$2t_{CY} + 500$	ns
t_{AIT}	$\overline{\text{ACK}} \uparrow$ to $\overline{\text{INT}} \uparrow$		$2t_{CY} + 500$	ns
t_{AED}	$\overline{\text{OBF}} \downarrow$ to $\overline{\text{ACK}} \downarrow$ Delay	0		ns

NOTES:

- $C_L = \text{pF}$ all outputs.
- Measured from logic "one" or "zero" to 1.5V at $C_L = 150 \text{ pF}$.
- P12, P13 are external clock inputs.
- Note that RxC may be used as an input only in 1X mode, otherwise it will be an output.
- The center of the Stop Bit will be the receiver sample time, as programmed by the modification register.
- $\frac{1}{4}$ th bit length for 32X, 64X; 100 ns for 1X.
- To ensure t_{RD} spec is met.

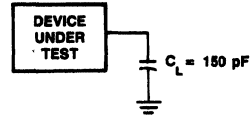
WAVEFORMS

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

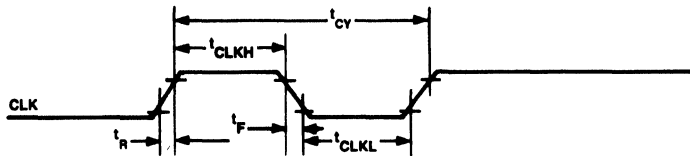
A.C. TESTING LOAD CIRCUIT



$C_L = 150 \text{ pF}$
 C_L Includes Jig Capacitance

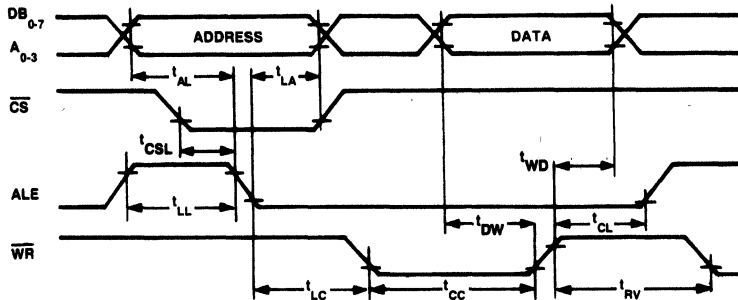
230759-6

SYSTEM CLOCK



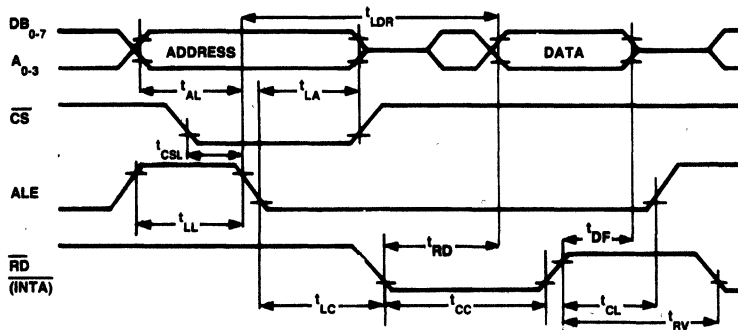
230759-7

WRITE CYCLE



230759-8

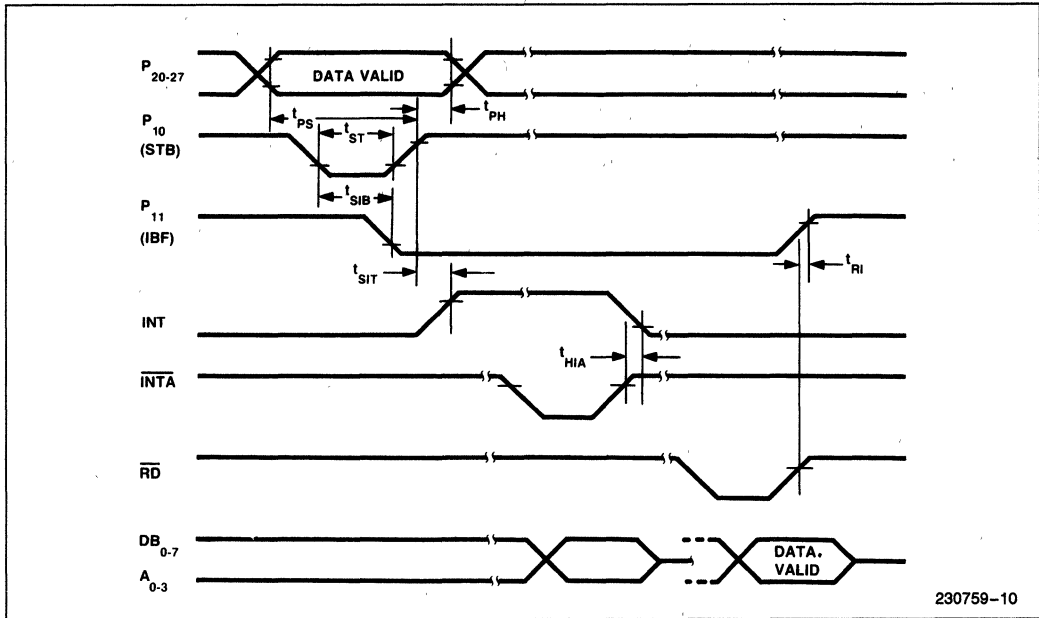
READ CYCLE



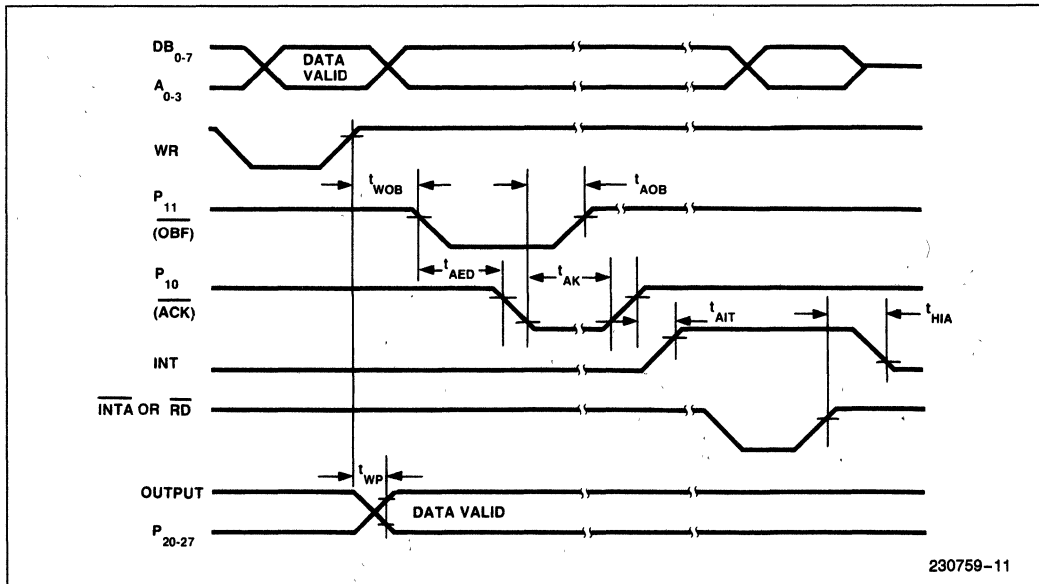
230759-9

WAVEFORMS (Continued)

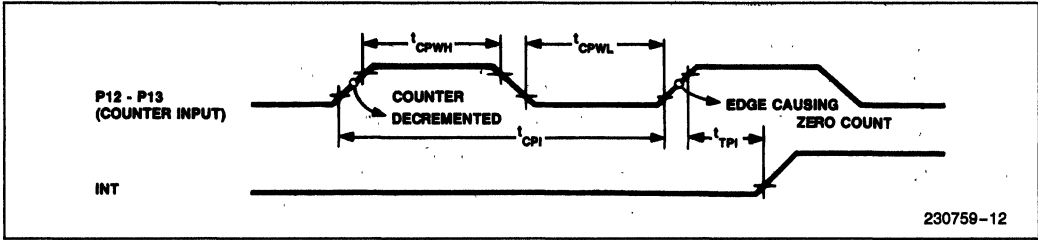
PARALLEL PORT HANDSHAKING—INPUT MODE



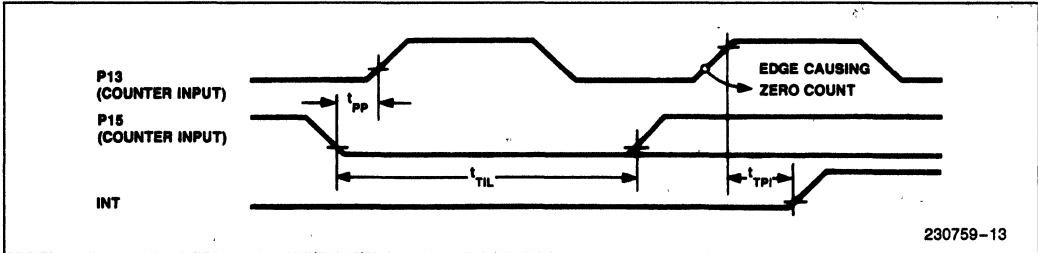
PARALLEL PORT HANDSHAKING—OUTPUT MODE



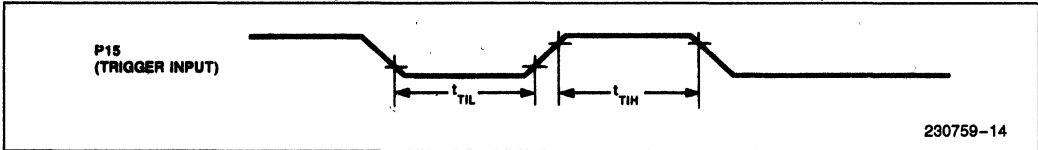
COUNT PULSE TIMINGS



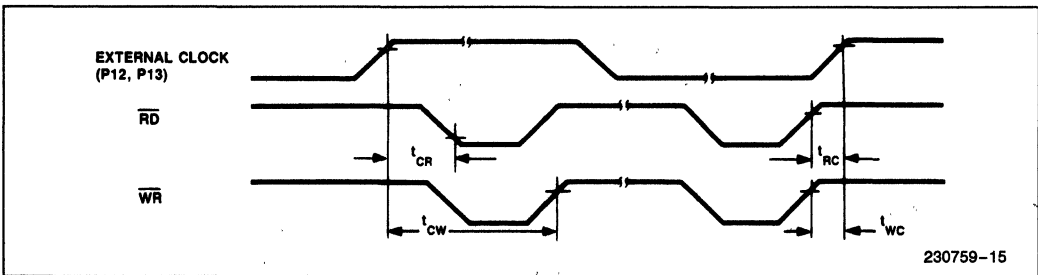
LOADING TIMER (OR CASCADED COUNTER/TIMER 3 AND 5)



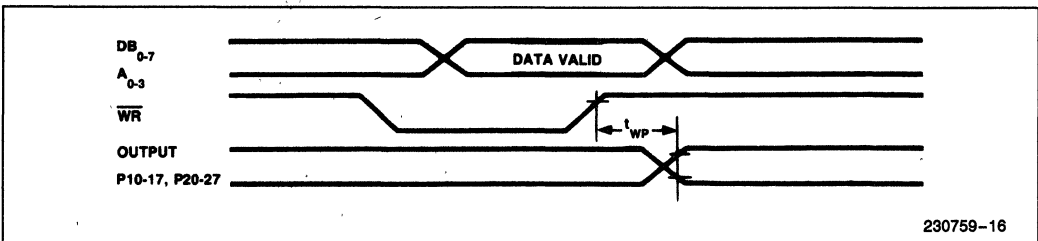
TRIGGER PULSE FOR TIMER 5 (CASCADED EVENT COUNTER/TIMER 3 AND 5)



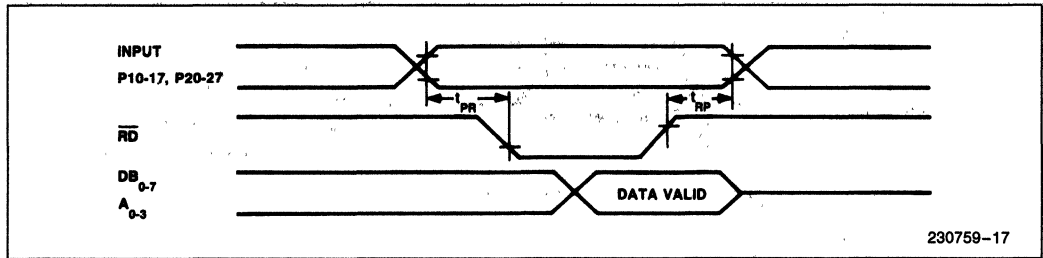
COUNTER TIMER TIMING



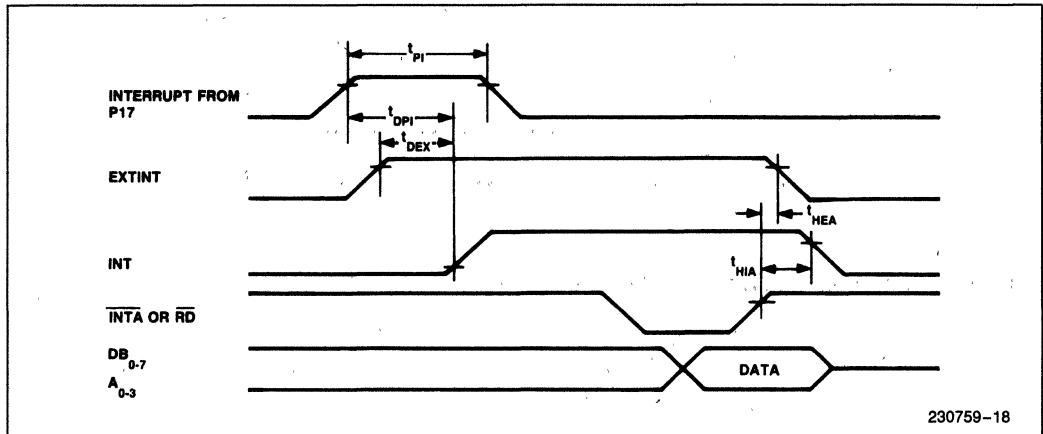
OUTPUT FROM PORT 1 AND PORT 2



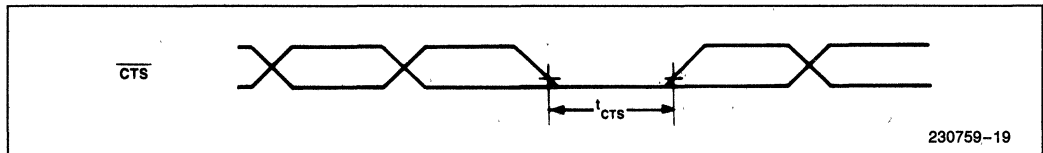
INPUT FROM PORT 1 AND PORT 2



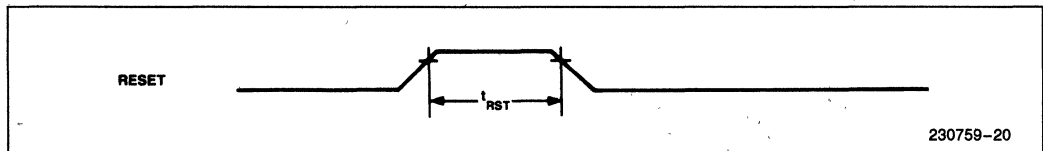
INTERRUPT TIMING



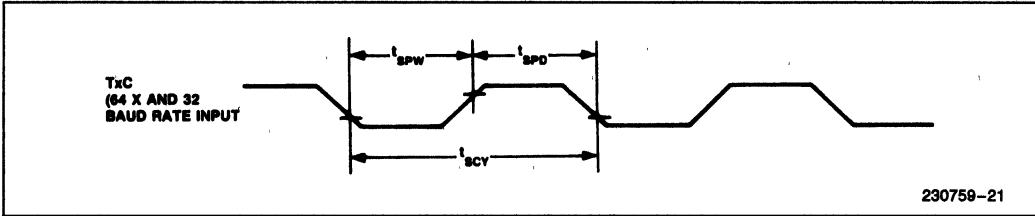
CTS FOR SINGLE CHARACTER TRANSMISSION



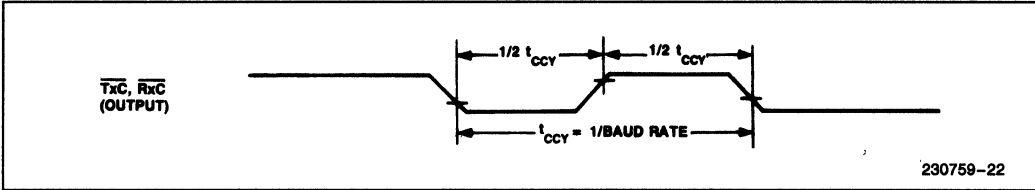
RESET TIMING



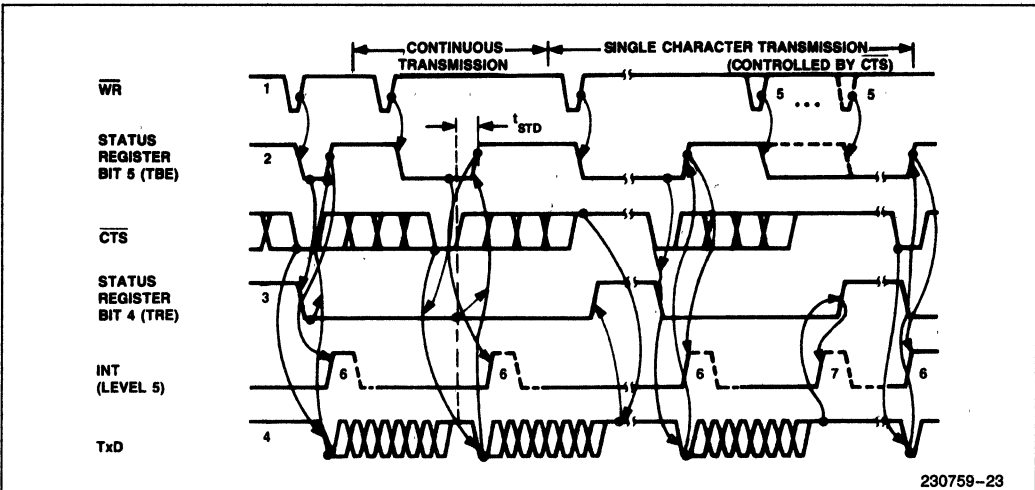
EXTERNAL BAUD RATE CLOCK FOR SERIAL INTERFACE



TRANSMITTER AND RECEIVER CLOCK FROM INTERNAL CLOCK SOURCE



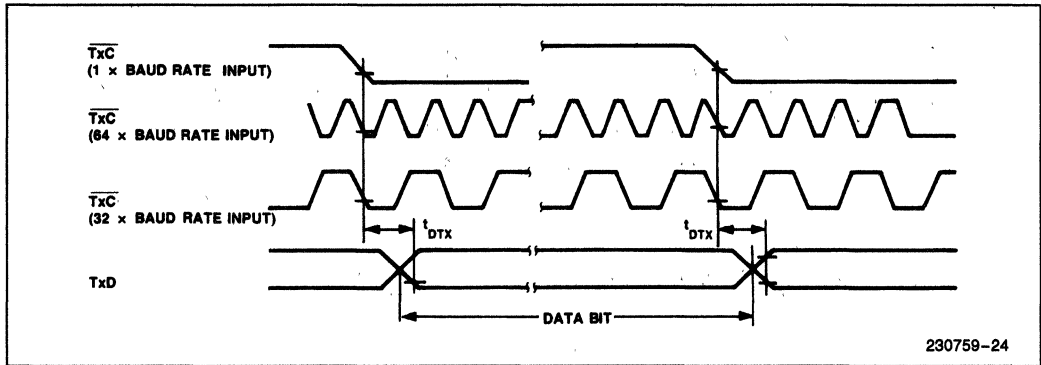
TRANSMISSION OF CHARACTERS ON SERIAL INTERFACE



NOTES:

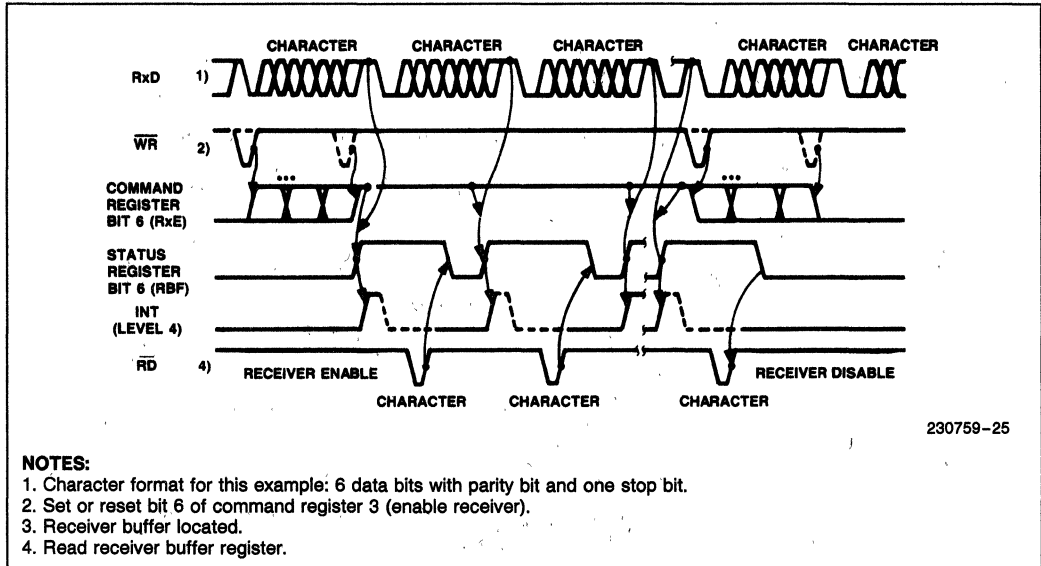
1. Load transmitter buffer register.
 2. Transmitter buffer register is empty.
 3. Transmitter register is empty.
 4. Character format for this example: 7 Data Bits with Parity Bit and 2 Stop Bits.
 5. Loading of transmitter buffer register must be complete before \overline{CTS} goes low.
 6. Interrupt due to transmitter buffer register empty.
 7. Interrupt due to transmitter register empty.
- No status bits are altered when RD is active.

DATA BIT OUTPUT ON SERIAL INTERFACE



230759-24

CONTINUOUS RECEPTION OF CHARACTERS ON SERIAL INTERFACE WITHOUT ERROR CONDITION

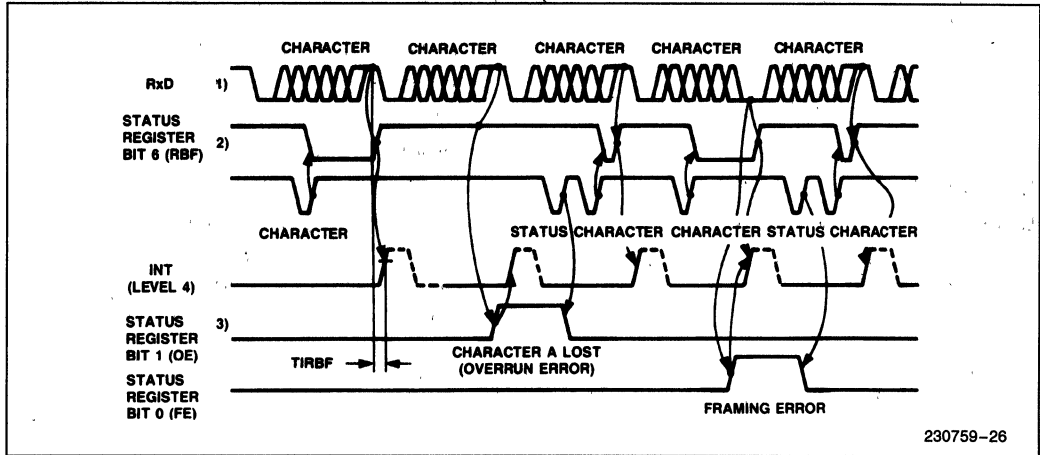


230759-25

NOTES:

1. Character format for this example: 6 data bits with parity bit and one stop bit.
2. Set or reset bit 6 of command register 3 (enable receiver).
3. Receiver buffer located.
4. Read receiver buffer register.

ERROR CONDITIONS DURING RECEPTION OF CHARACTERS ON THE SERIAL INTERFACE



NOTES:

1. Character format for this example: 6 data bits without parity and one stop bit.
 2. Receiver buffer register loaded.
 3. Overrun error.
 4. Framing error.
 5. Interrupt from receiver buffer register loading.
 6. Interrupt from overrun error.
 7. Interrupt from framing error and loading receiver buffer register.
- No status bits are altered when RD is active.



8279/8279-5 PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- Simultaneous Keyboard Display Operations
- Scanned Keyboard Mode
- Scanned Sensor Mode
- Strobed Input Entry Mode
- 8-Character Keyboard FIFO
- 2-Key Lockout or N-Key Rollover with Contact Debounce
- Dual 8- or 16-Numerical Display
- Single 16-Character Display
- Right or Left Entry 16-Byte Display RAM
- Mode Programmable from CPU
- Programmable Scan Timing
- Interrupt Output on Key Entry
- Available in EXPRESS
 - Standard Temperature Range
 - Extended Temperature Range

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16x8 display RAM which can be organized into dual 16x4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

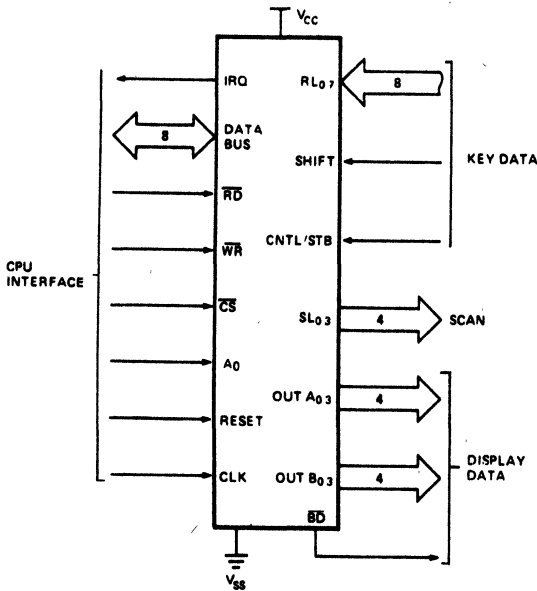
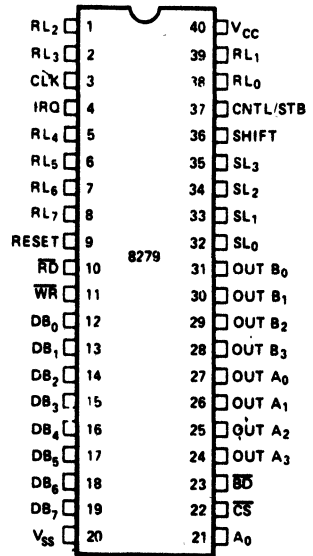


Figure 1. Logic Symbol

290123-1



290123-2

Figure 2. Pin Configuration

HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

Table 1. Pin Description

Symbol	Pin No.	Name and Function
DB ₀ -DB ₇	19-12	BI-DIRECTIONAL DATA BUS: All data and commands between the CPU and the 8279 are transmitted on these lines.
CLK	3	CLOCK: Clock from system used to generate internal timing.
RESET	9	RESET: A high signal on this pin resets the 8279. After being reset the 8279 is placed in the following mode: 1) 16 8-bit character display—left entry. 2) Encoded scan keyboard—2 key lockout. Along with this the program clock prescaler is set to 31.
CS	22	CHIP SELECT: A low on this pin enables the interface functions to receive or transmit.
A ₀	21	BUFFER ADDRESS: A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
RD, WR	10-11	INPUT/OUTPUT READ AND WRITE: These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
IRQ	4	INTERRUPT REQUEST: In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
V _{SS} , V _{CC}	20, 40	GROUND AND POWER SUPPLY PINS.
SL ₀ -SL ₃	32-35	SCAN LINES: Scan lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
RL ₀ -RL ₇	38, 39, 1, 2, 5-8	RETURN LINE: Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.
SHIFT	36	SHIFT: The shift input status is stored along with the key position on key closure in the Scanned Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.
CNTL/STB	37	CONTROL/STROBED INPUT MODE: For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode. (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
OUT A ₀ -OUT A ₃ OUT B ₀ -OUT B ₃	27-24, 31-28	OUTPUTS: These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL ₀ -SL ₃) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8-bit port.
BD	23	BLANK DISPLAY: This output is used to blank the display during digit switching or by a display blanking command.

FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

Input Modes

- Scanned Keyboard—with encoded (8 x 8 key keyboard) or decoded (4 x 8 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key roll-over.
- Scanned Sensor Matrix—with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input—Data on return lines during control line strobe is transferred to FIFO.

Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit ($B_0 = D_0, A_3 = D_7$).
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Clock Prescaler
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.

PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 3.

I/O Control and Data Buffers

The I/O control section uses the \overline{CS} , A_0 , \overline{RD} and \overline{WR} lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by \overline{CS} . The character of the information, given or desired by the CPU, is identified by A_0 . A logic one means the information is a command or status. A logic zero means the information is data. \overline{RD} and \overline{WR} determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ($\overline{CS} = 1$), the devices are in a high impedance state. The drivers input during $\overline{WR} \cdot \overline{CS}$ and output during $\overline{RD} \cdot \overline{CS}$.

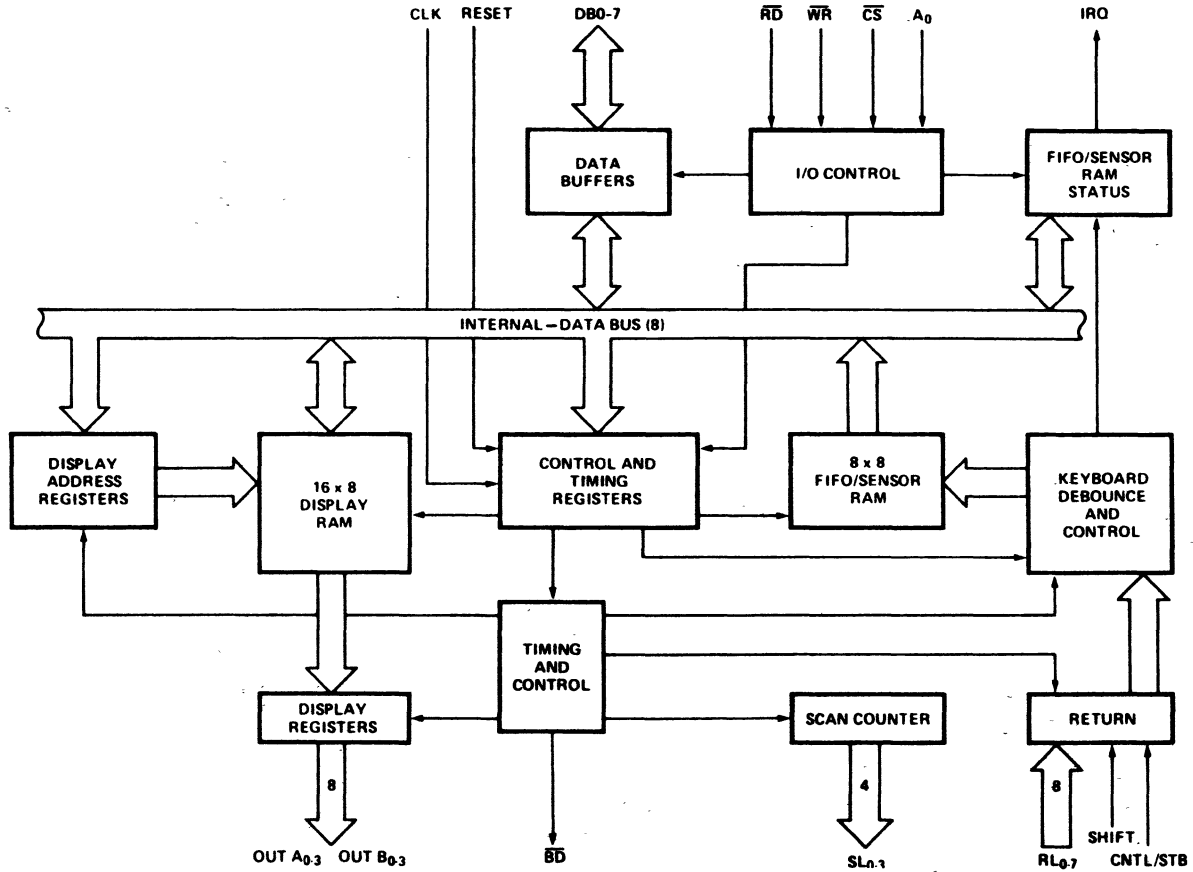
Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with $A_0 = 1$ and then sending a \overline{WR} . The command is latched on the rising edge of \overline{WR} . The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a $\div N$ prescaler that can be programmed to yield an internal frequency of 100 kHz which gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

Scan Counter

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note that when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.



290123-3

Figure 3. Internal Block Diagram

Return Buffers and Keyboard Debounce and Control

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 ms to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

FIFO/Sensor RAM and Status

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an RD with CS low and A₀ high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

Display Address Registers and Display RAM

The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

SOFTWARE OPERATION

8279 Commands

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with CS low and A₀ high and are loaded to the 8279 on the rising edge of WR.

Keyboard/Display Mode Set

	MSB						LSB	
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the Keyboard Mode.

DD

- 0 0 8 8-bit character display—Left entry
- 0 1 16 8-bit character display—Left entry*
- 1 0 8 8-bit character display—Right entry
- 1 1 16 8-bit character display—Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

- 0 0 0 Encoded Scan Keyboard—2 Key Lock-out*
- 0 0 1 Decoded Scan Keyboard—2-Key Lock-out
- 0 1 0 Encoded Scan Keyboard—N-Key Roll-over
- 0 1 1 Decoded Scan Keyboard—N-Key Roll-over
- 1 0 0 Encoded Scan Sensor Matrix
- 1 0 1 Decoded Scan Sensor Matrix
- 1 1 0 Strobed Input, Encoded Display Scan
- 1 1 1 Strobed Input, Decoded Display Scan

*Default after reset.

Program Clock

Code:	0	0	1	P	P	P	P	P
-------	---	---	---	---	---	---	---	---

All timing and multiplexing signals for the 8279 are generated by an internal prescaler. This prescaler divides the external clock (pin 3) by a programmable integer. Bits P P P P P determine the value of this integer which ranges from 2 to 31. Choosing a divisor that yields 100 kHz will give the specified scan and

debounce times. For instance, if Pin 3 of the 8279 is being clocked by a 2 MHz signal, PPPPP should be set to 10100 to divide the clock by 20 to yield the proper 100 kHz operating frequency.

Read FIFO/Sensor RAM

Code:

0	1	0	AI	X	A	A	A
---	---	---	----	---	---	---	---

 X = Don't Care

The CPU sets the 8279 for a read of the FIFO/Sensor RAM by first writing this command. In the Scan Keyboard Mode, the Auto-Increment flag (AI) and the RAM address bits (AAA) are irrelevant. The 8279 will automatically drive the data bus for each subsequent read ($A_0 = 0$) in the same sequence in which the data first entered the FIFO. All subsequent reads will be from the FIFO until another command is issued.

In the Sensor Matrix Mode, the RAM address bits AAA select one of the 8 rows of the Sensor RAM. If the AI flag is set ($AI = 1$), each successive read will be from the subsequent row of the sensor RAM.

Read Display RAM

Code:

0	1	1	AI	A	A	A	A
---	---	---	----	---	---	---	---

The CPU sets up the 8279 for a read of the Display RAM by first writing this command. The address bits AAAA select one of the 16 rows of the Display RAM. If the AI flag is set ($AI = 1$), this row address will be incremented after each following read or write to the Display RAM. Since the same counter is used for both reading and writing, this command sets the next read or write address and the sense of the Auto-Increment mode for both operations.

Write Display RAM

Code:

1	0	0	AI	A	A	A	A
---	---	---	----	---	---	---	---

The CPU sets up the 8279 for a write to the Display RAM by first writing this command. After writing the command with $A_0 = 1$, all subsequent writes with $A_0 = 0$ will be to the Display RAM. The addressing and Auto-Increment functions are identical to those for the Read Display RAM. However, this command does not affect the source of subsequent Data Reads; the CPU will read from whichever RAM (Display of FIFO/Sensor) which was last specified. If, indeed, the Display RAM was last specified, the Write-Display RAM will, nevertheless, change the next Read location.

Display Write Inhibit/Blanking

Code:

				A	B	A	B
1	0	1	X	IW	IW	BL	BL

The IW Bits can be used to mask nibble A and nibble B in applications requiring separate 4-bit display ports. By setting the IW flag ($IW = 1$), for one of the ports, the port becomes masked so that entries to the Display RAM from the CPU do not affect that port. Thus, if each nibble is input to a BCD decoder, the CPU may write a digit to the Display RAM without affecting the other digit being displayed. It is important to note that bit B_0 corresponds to bit D_0 on the CPU bus, and that bit A_3 corresponds to bit D_7 .

If the user wishes to blank the display, the BL flags are available for each nibble. The last Clear command issued determines the code to be used as a "blank." This code defaults to all zeros after a reset. Note that both BL flags must be set to blank a display formatted with a single 8-bit port.

Clear

Code:

1	1	0	C_D	C_D	C_D	C_F	C_A
---	---	---	-------	-------	-------	-------	-------

The C_D bits are available in this command to clear all rows of the Display RAM to a selectable blanking code as follows:

	C_D	C_D	
↑	0	X	All Zeros (X = Don't Care)
	1	0	AB = Hex 20 (0010 0000)
	1	1	All Ones
	— Enable clear display when = 1 (or by $C_A = 1$)		

290123-13

During the time the Display RAM is being cleared ($\sim 160 \mu s$), it may not be written to. The most significant bit of the FIFO status word is set during this time. When the Display RAM becomes available again, it automatically resets.

If the C_F bit is asserted ($C_F = 1$), the FIFO status is cleared and the interrupt output line is reset. Also, the Sensor RAM pointer is set to row 0.

C_A , the Clear All bit, has the combined effect of C_D and C_F ; it uses the C_D clearing code on the Display RAM and also clears FIFO status. Furthermore, it resynchronizes the internal timing chain.

End Interrupt/Error Mode Set

Code:

1	1	1	E	X	X	X	X
---	---	---	---	---	---	---	---

 X = Don't care

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode—if the E bit is programmed to “1” the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

Status Word

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when A_0 is high and \overline{CS} and \overline{RD} are low. See Interface Considerations for more detail on status word.

Data Read

Data is read when A_0 , \overline{CS} and \overline{RD} are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of \overline{RD} will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

Data Write

Data that is written with A_0 , \overline{CS} and \overline{WR} low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of \overline{WR} occurs if AI is set by the latest display command.

INTERFACE CONSIDERATIONS

Scanned Keyboard Mode, 2-Key Lockout

There are three possible combinations of conditions that can occur during debounce scanning. When a key is depressed, the debounce logic is set. Other depressed keys are looked for during the next two scans. If none are encountered, it is a single key depression and the key position is entered into the

FIFO along with the status of CNTL and SHIFT lines. If the FIFO was empty, IRQ will be set to signal the CPU that there is an entry in the FIFO. If the FIFO was full, the key will not be entered and the error flag will be set. If another closed switch is encountered, no entry to the FIFO can occur. If all other keys are released before this one, then it will be entered to the FIFO. If this key is released before any other, it will be entirely ignored. A key is entered to the FIFO only once per depression, no matter how many keys were pressed along with it or in what order they were released. If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one key remains depressed alone. The last key will be treated as a single key depression.

Scanned Keyboard Mode, N-Key Rollover

With N-key Rollover each key depression is treated independently from all others. When a key is depressed, the debounce circuit waits 2 keyboard scans and then checks to see if the key is still down. If it is, the key is entered into the FIFO. Any number of keys can be depressed and another can be recognized and entered into the FIFO. If a simultaneous depression occurs, the keys are recognized and entered according to the order the keyboard scan found them.

Scanned Keyboard—Special Error Modes

For N-key rollover mode the user can program a special error mode. This is done by the “End Interrupt/Error Mode Set” command. The debounce cycle and key-validity check are as in normal N-key mode. If during a *single debounce cycle*, two keys are found depressed, this is considered a simultaneous multiple depression, and sets an error flag. This flag will prevent any further writing into the FIFO and will set interrupt (if not yet set). The error flag could be read in this mode by reading the FIFO STATUS word. (See “FIFO STATUS” for further details.) The error flag is reset by sending the normal CLEAR command with $CF = 1$.

Sensor Matrix Mode

In Sensor Matrix mode, the debounce logic is inhibited. The status of the sensor switch is inputted directly to the Sensor RAM. In this way the Sensor RAM keeps an image of the state of the switches in the sensor matrix. Although debouncing is not provided, this mode has the advantage that the CPU knows how long the sensor was closed and when it

was released. A keyboard mode can only indicate a validated closure. To make the software easier, the designer should functionally group the sensors by row since this is the format in which the CPU will read them.

The IRQ line goes high if any sensor value change is detected at the end of a sensor matrix scan. The IRQ line is cleared by the first data read operation if the Auto-Increment flag is set to zero, or by the End Interrupt command if the Auto-Increment flag is set to one.

NOTE:

Multiple changes in the matrix Addressed by (SL₀₋₃ = 0) may cause multiple interrupts. (SL₀ = 0 in the Decoded Mode.) Reset may cause the 8279 to see multiple changes.

Data Format

In the Scanned Keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard plus the status of the CNTL and SHIFT lines (non-inverted). CNTL is the MSB of the character and SHIFT is the next most significant bit. The next three bits are from the scan counter and indicate the row the key was found in. The last three bits are from the column counter and indicate to which return line the key was connected.

MSB			LSB			
CNTL	SHIFT	SCAN	SCAN	SCAN	SCAN	RETURN

SCANNED KEYBOARD DATA FORMAT

In Sensor Matrix mode, the data on the return lines is entered directly in the row of the Sensor RAM that corresponds to the row in the matrix being scanned. Therefore, each switch position maps directly to a Sensor RAM position. The SHIFT and CNTL inputs are ignored in this mode. Note that switches are not necessarily the only thing that can be connected to the return lines in this mode. Any logic that can be triggered by the scan lines can enter data to the return line inputs. Eight multiplexed input ports could be tied to the return lines and scanned by the 8279.

MSB			LSB				
RL ₇	RL ₆	RL ₅	RL ₄	RL ₃	RL ₂	RL ₁	RL ₀

In Strobed Input mode, the data is also entered to the FIFO from the return lines. The data is entered

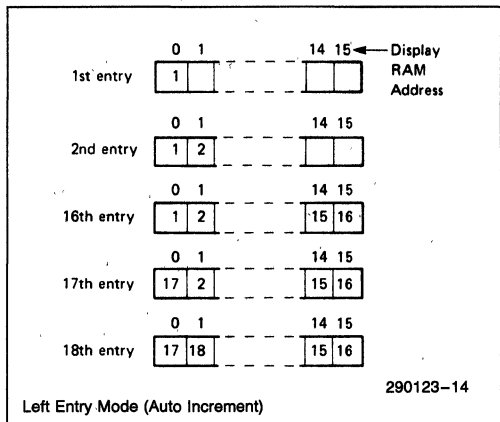
by the rising edge of a CNTL/STB line pulse. Data can come from another encoded keyboard or simple switch matrix. The return lines can also be used as a general purpose strobed input.

MSB			LSB				
RL ₇	RL ₆	RL ₅	RL ₄	RL ₃	RL ₂	RL ₁	RL ₀

Display

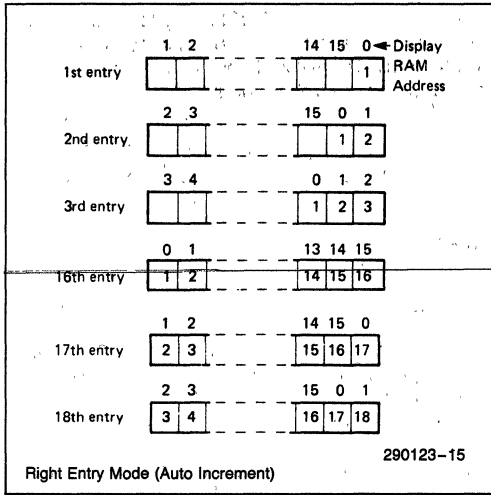
Left Entry

Left Entry mode is the simplest display format in that each display position directly corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 (or address 7 in 8 character display) is the right most display character. Entering characters from position zero causes the display to fill from the left. The 17th (9th) character is entered back in the left most position and filling again proceeds from there.



Right Entry

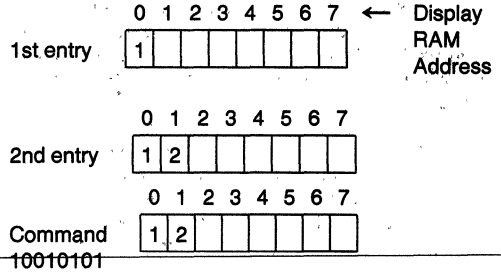
Right entry is the method used by most electronic calculators. The first entry is placed in the right most display character. The next entry is also placed in the right most character after the display is shifted left one character. The left most character is shifted off the end and is lost.



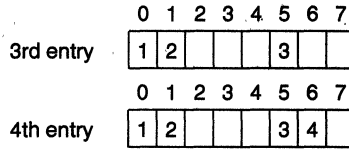
Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended.

Auto Increment

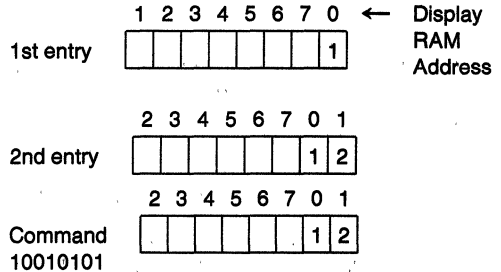
In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Auto Increment mode has no undesirable side effects and the result is predictable:



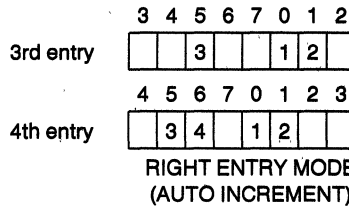
Enter next at Location 5 Auto Increment



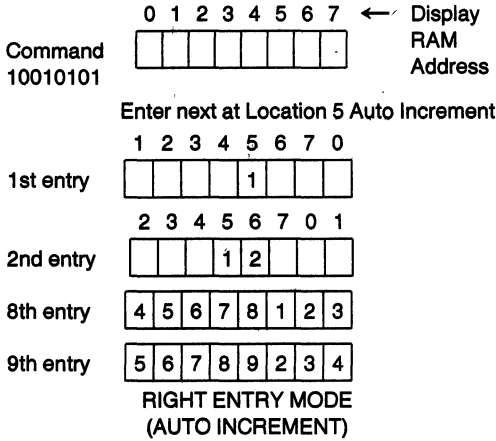
In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except if the address sequence is interrupted.



Enter next at Location 5 Auto Increment



Starting at an arbitrary location operates as shown below:



Entry appears to be from the initial entry point.

8/16 Character Display Formats

If the display mode is set to an 8 character display, the on duty-cycle is double what it would be for a 16 character display (e.g., 5.1 ms scan time for 8 characters vs. 10.3 ms for 16 characters with 100 kHz internal frequency).

G. FIFO Status

FIFO status is used in the Keyboard and Strobed Input modes to indicate the number of characters in

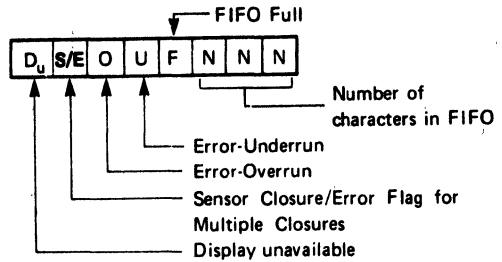
the FIFO and to indicate whether an error has occurred. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation.

In a Sensor Matrix mode, a bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occurred.

FIFO STATUS WORD



290123-4

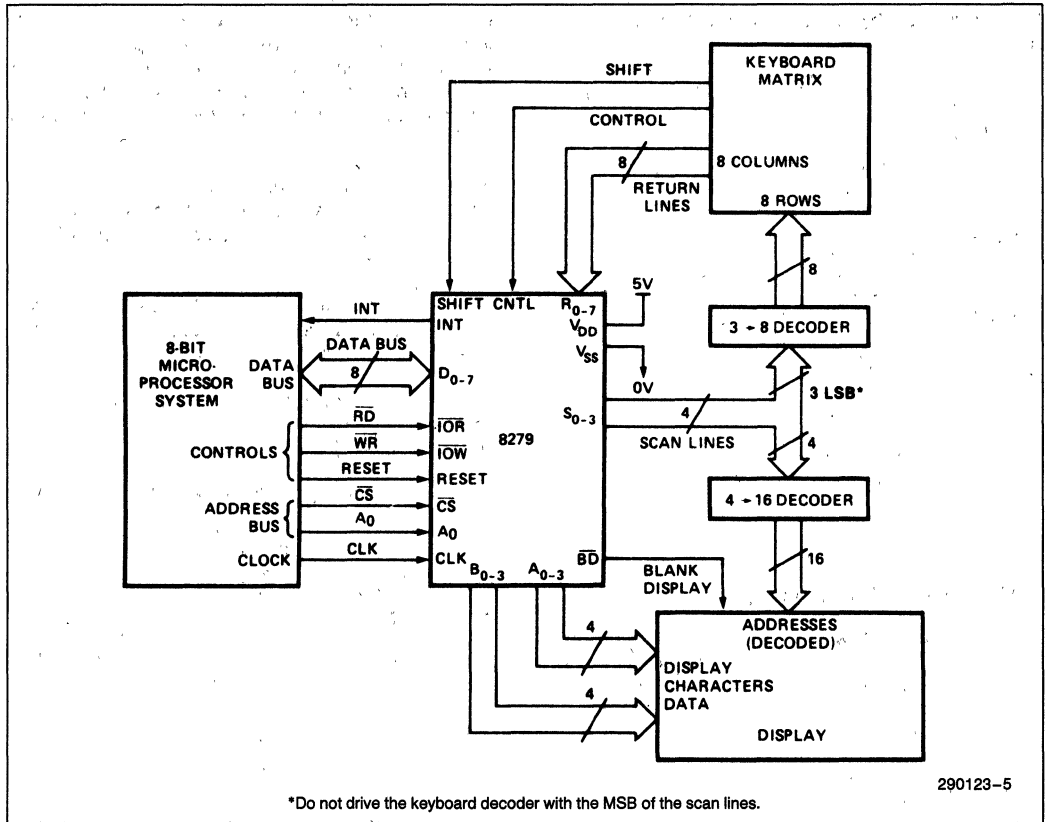


Figure 4. System Block Diagram

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature 0°C to 70°C
 Storage Temperature -65°C to 125°C
 Voltage on any Pin with
 Respect to Ground -0.5V to +7V
 Power Dissipation 1 Watt

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

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ $V_{SS} = 0\text{V}$ (Note 3)*

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL1}	Input Low Voltage for Return Lines	-0.5	1.4	V	
V_{IL2}	Input Low Voltage for All Others	-0.5	0.8	V	
V_{IH1}	Input High Voltage for Return Lines	2.2		V	
V_{IH2}	Input High Voltage for All Others	2.0		V	
V_{OL}	Output Low Voltage		0.45	V	(Note 1)
V_{OH1}	Output High Voltage on Interrupt Line	3.5		V	(Note 2)
V_{OH2}	Other Outputs	2.4			$I_{OH} = -400 \mu\text{A}$ 8279-5 $-100 \mu\text{A}$ 8279
I_{IL1}	Input Current on Shift, Control and Return Lines		+10 -100	μA	$V_{IN} = V_{CC}$ $V_{IN} = 0\text{V}$
I_{IL2}	Input Leakage Current on All Others		± 10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		± 10	μA	$V_{OUT} = V_{CC}$ to 0.45V
I_{CC}	Power Supply Current		120	mA	

CAPACITANCE

Symbol	Parameter	Typ	Max	Unit	Test Conditions
C_{IN}	Input Capacitance	5	10	pF	$f_C = 1 \text{ MHz}$ Unmeasured Pins Returned to V_{SS}
C_{OUT}	Output Capacitance	10	20	pF	

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{SS} = 0\text{V}$ (Note 3)*

Bus Parameters
READ CYCLE

Symbol	Parameter	8279		8279-5		Unit
		Min	Max	Min	Max	
t_{AR}	Address Stable Before $\overline{\text{READ}}$	50		0		ns
t_{RA}	Address Hold Time for $\overline{\text{READ}}$	5		0		ns
t_{RR}	$\overline{\text{READ}}$ Pulse Width	420		250		ns
$t_{RD}^{(4)}$	Data Delay from $\overline{\text{READ}}$		300		150	ns
$t_{AD}^{(4)}$	Address to Data Valid		450		250	ns
t_{DF}	$\overline{\text{READ}}$ to Data Floating	10	100	10	100	ns
t_{RCY}	Read Cycle Time	1		1		μs

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

Symbol	Parameter	8279		8279-5		Unit
		Min	Max	Min	Max	
t_{AW}	Address Stable Before \overline{WRITE}	50		0		ns
t_{WA}	Address Hold Time for \overline{WRITE}	20		0		ns
t_{WW}	\overline{WRITE} Pulse Width	400		250		ns
t_{DW}	Data Set Up Time for \overline{WRITE}	300		150		ns
t_{WD}	Data Hold Time for \overline{WRITE}	40		0		ns
t_{WCY}	Write Cycle Time	1		1		μ s

OTHER TIMINGS

Symbol	Parameter	8279		8279-5		Unit
		Min	Max	Min	Max	
$t_{\phi W}$	Clock Pulse Width	230		120		ns
t_{CY}	Clock Period	500		320		ns

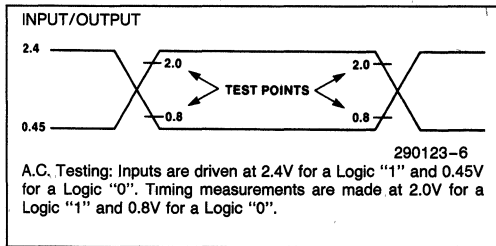
Keyboard Scan Time 5.1 ms
 Keyboard Debounce Time 10.3 ms
 Key Scan Time 80 μ s
 Display Scan Time 10.3 ms

Digit-on Time 480 μ s
 Blanking Time 160 μ s
 Internal Clock Cycle⁽⁵⁾ 10 μ s

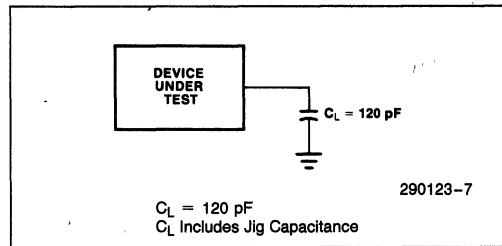
NOTES:

1. 8279, $I_{OL} = 1.6$ mA; 8279-5, $I_{OL} = 2.2$ mA.
 2. $I_{OH} = -100$ μ A
 3. 8279, $V_{CC} = +5V \pm 5\%$; 8279-5, $V_{CC} = +5V \pm 10\%$
 4. 8279, $C_L = 100$ pF; 8279-5, $C_L = 150$ pF.
 5. The Prescaler should be programmed to provide a 10 μ s internal clock cycle.
- * For Extended Temperature EXPRESS, use M8279A electrical parameters.

A.C. TESTING INPUT, OUTPUT WAVEFORM

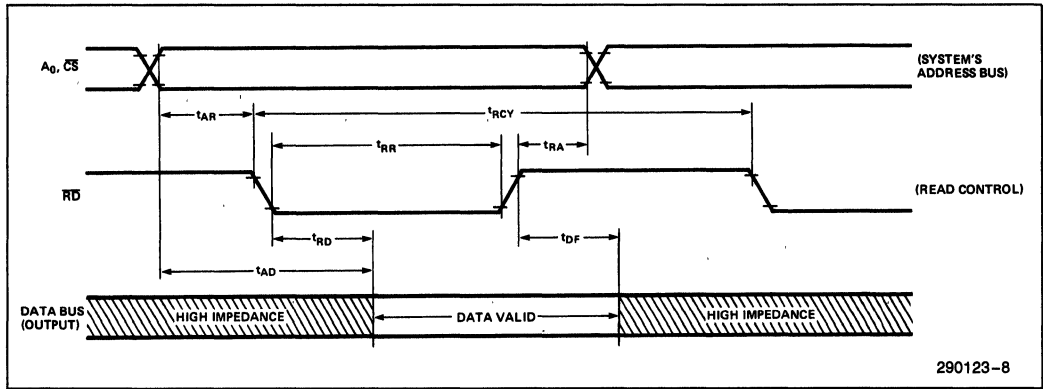


A.C. TESTING LOAD CIRCUIT

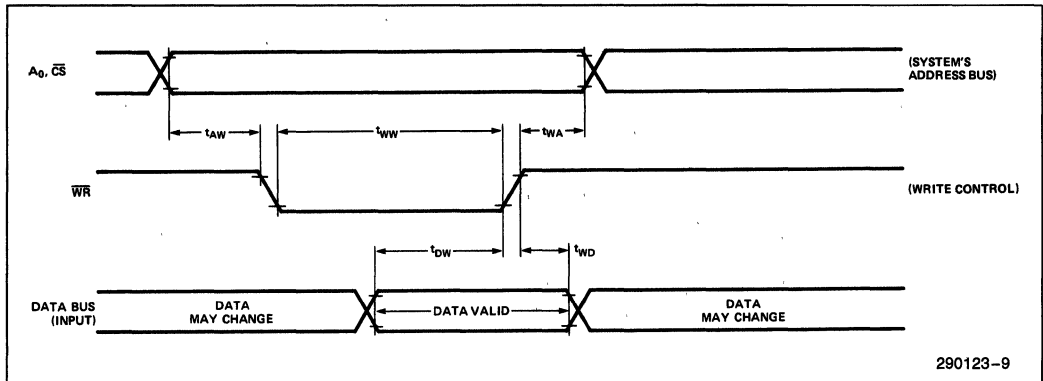


WAVEFORMS

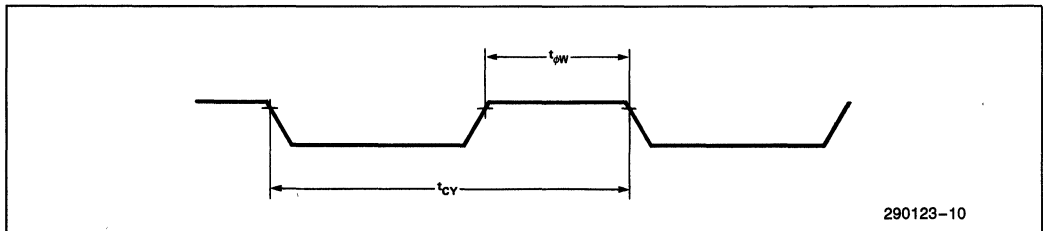
READ OPERATION



WRITE OPERATION

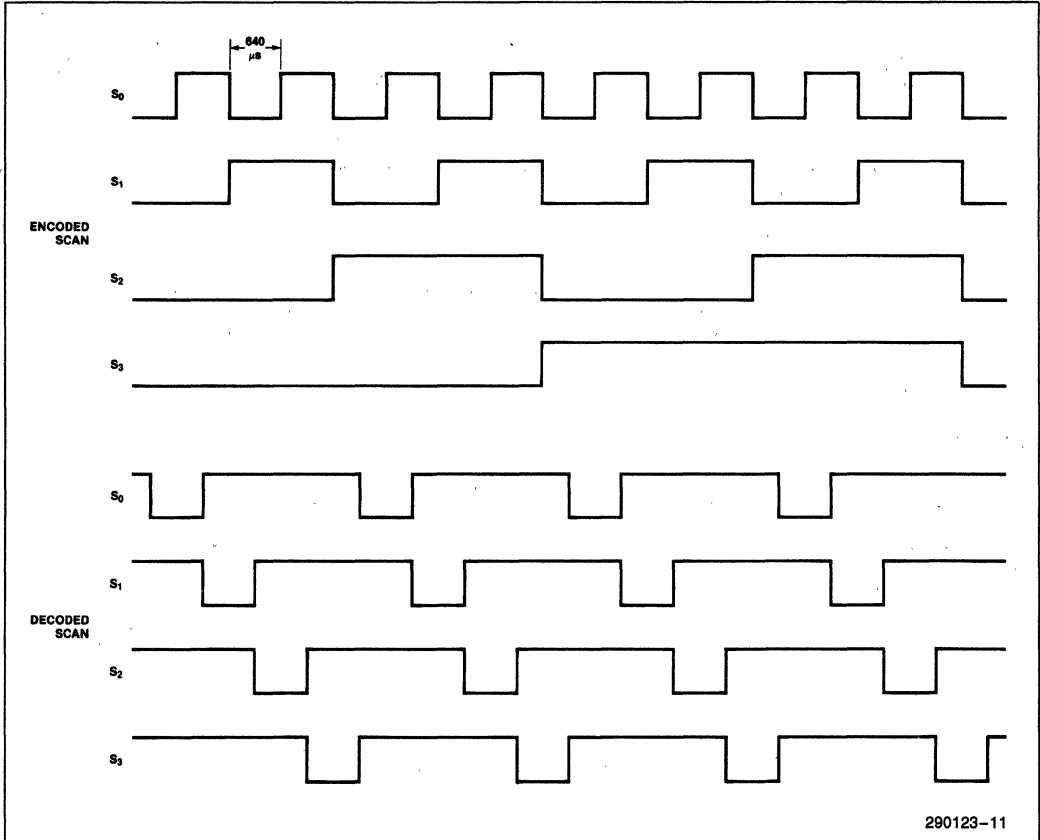


CLOCK INPUT



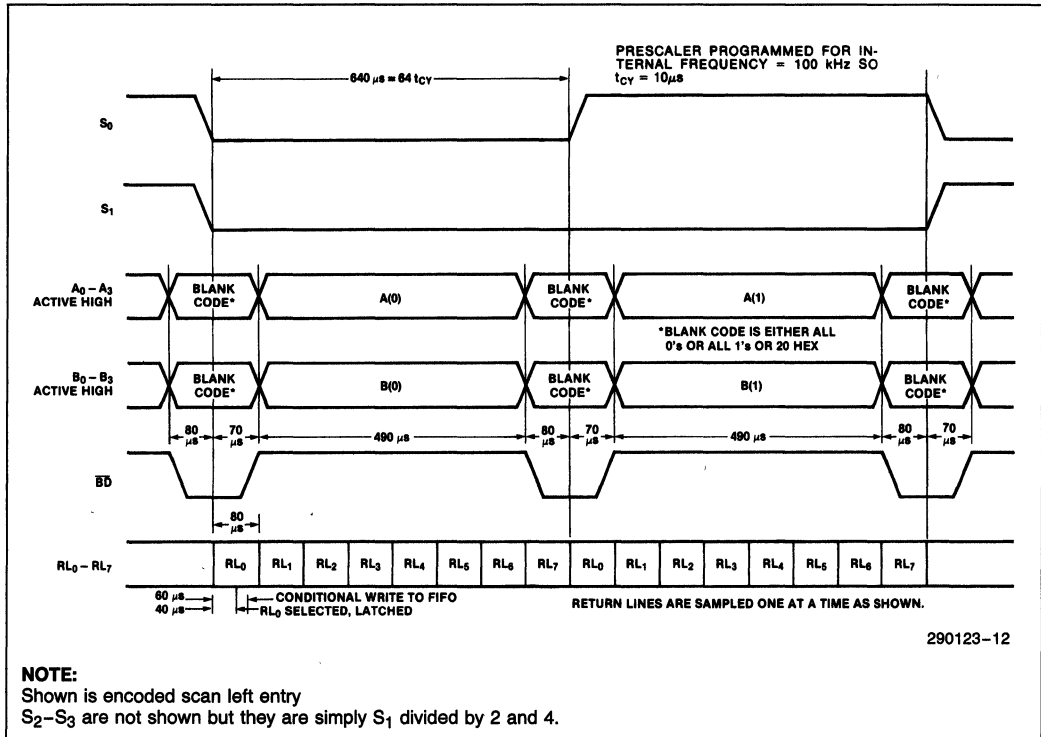
WAVEFORMS (Continued)

SCAN



WAVEFORMS (Continued)

DISPLAY





**APPLICATION
NOTE**

AP-153

November 1986

Designing with the 8256

CHARLES T. YAGER
APPLICATIONS ENGINEER
INTEL CORPORATION

Order Number: 210907-001

INTRODUCTION

The INTEL 8256 MUART is a Multifunction Universal Asynchronous Receiver Transmitter designed to be used for serial asynchronous communication while also providing hardware support for parallel I/O, timing, counting and interrupt control. Its versatile design allows it to be directly connected to the MCS[®]-85, iAPX-86, iAPX-88, iAPX-186, and iAPX-188 microcomputer systems plus the MCS-48 and MCS-51 family of single-chip microcomputers.

The four commonly used peripheral functions contained in the MUART are:

- 1) Full-duplex, double-buffered serial asynchronous Receiver/Transmitter with an on-chip Baud Rate Generator
- 2) Two—8-bit parallel I/O ports
- 3) Five—8-bit counters/timers
- 4) 8-level priority interrupt controller

This manual can be divided into two parts. The first part describes the MUART in detail, including its functions, registers and pins. This section also describes the interface between the MUART and Intel CPUs plus a discussion on programming considerations. The second section provides an application example: a MUART-based line printer multiplexer. The Appendix contains software listings for the line printer multiplexer and some useful reference information.

DESCRIPTION OF THE MUART

The MUART can be logically partitioned into seven sections: the microprocessor bus interface, the command and status registers, clocking circuitry, asynchronous serial communication, parallel I/O, timer/event counters, and the interrupt controller. This can be seen from the block diagram of the 8256 MUART as shown in Figure 1. The MUART's pin configuration can be seen in Figure 2.

Microprocessor Bus Interface

The microprocessor bus interface is the hardware section of the MUART which allows a μP to communicate with the MUART. It consists of tristate bi-directional data-bus buffers, an address latch, a chip select $\overline{\text{CS}}$ latch and bus control logic. In order to provide all of the MUART's functions in a 40-pin DIP while retaining direct register addressing, a multiplexed address/data bus is used.

ADDRESS/DATA BUS

The MUART contains 16 internal directly addressable read/write registers. Four of the eight address/data lines are used to generate the address. When using 8-bit microprocessors such as MCS-85, MCS-48 and MCS-51, AD0-AD3 are used to address the 16 internal registers while Address/Data line 4 (AD4) is not used for addressing. For 16-bit systems, AD1-AD4 are used to generate the address for the internal data registers and AD0 is used as a second active low chip select.

$\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{CS}}$

The 8256 bus interface uses the standard bus control signals which are compatible with all Intel peripherals and microprocessors. The chip select signal ($\overline{\text{CS}}$), typically derived from an address decoder, is latched along with the address on the falling edge of ALE. As a result, chip select does not have to remain low for the entire bus cycle. However, the data bus buffers will remain tristated unless an $\overline{\text{RD}}$ or a $\overline{\text{WR}}$ signal becomes active while chip select has been latched in low.

INT, $\overline{\text{INTA}}$

The INT and $\overline{\text{INTA}}$ signals are used to interrupt the CPU and receive the CPU's acknowledgment to the interrupt request. The MUART can vector the CPU to the appropriate service routine depending on the source of the interrupt.

RESET

When a high level occurs on the RESET pin, the MUART is placed in a known initial state. This initial state is described under "Hardware Reset".

Command and Status Register

There are three command registers and one status register as shown in Figure 1. The three command registers are read/write registers while the status register is a read only. The command registers configure the MUART for its operating environment (i.e., 8 or 16 bits CPU, system clock frequency). In addition, they direct its higher level functions such as controlling the UART, selecting modes of operation for the interrupt controller, and choosing the fundamental frequency for the timers. Command Register 3 is the only register in the MUART which is a bit set/reset register, allowing the programmer to simply perform one write to set or reset any of the bits.

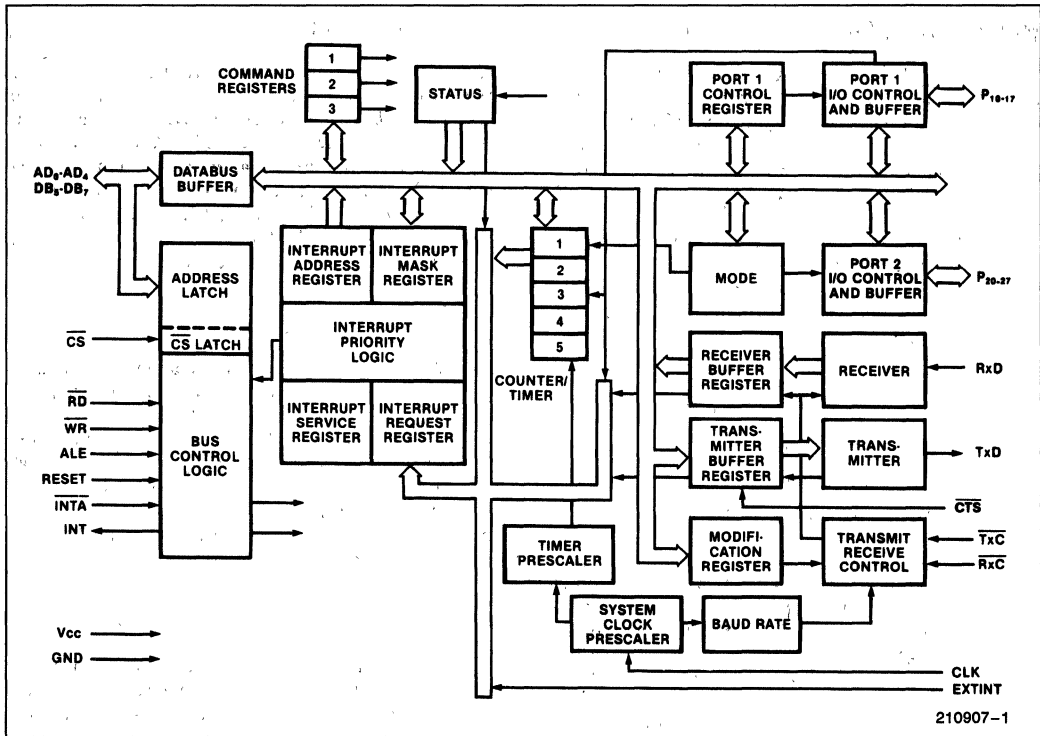


Figure 1. Block Diagram of the 8256 MUART

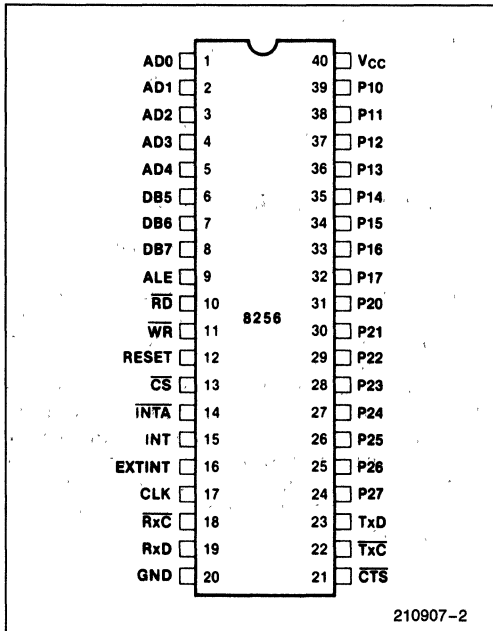


Figure 2. MUART Pin Configuration

The status register provides all of the information about the status of the UART's transmitter and receiver as well as the status of the interrupt pin. The status register is the only read only register in the MUART.

CLOCK CIRCUITRY

The clock for the five timers and baud rate generator is derived from the system clock. The system clock, pin 17 (CLK), is fed into a system clock prescaler which in turn feeds the five timers and the baud rate generator. The MUART's system clock can be asynchronous to the microprocessor's clock.

SYSTEM CLOCK PRESCALER

The system clock prescaler is a programmable divider which normalizes the internal clocking frequency for the timers and baud rate generator to 1.024 MHz. It divides the system clock (CLK) by 1, 2, 3, or 5, allowing clock frequencies of 1.024 MHz, 2.048 MHz, 3.072 MHz or 5.12 MHz. (The commonly used 6.144 MHz crystal frequency for the 8085 results in a 3.072 MHz frequency from the 8085's CLK pin.) If the system clock is not one of the four frequencies mentioned above, then the frequency of the baud rate generator

and the timers will be nonstandard; however, the MUART will still run as long as the system clock meets the data sheet tcy spec.

Timer Prescaler

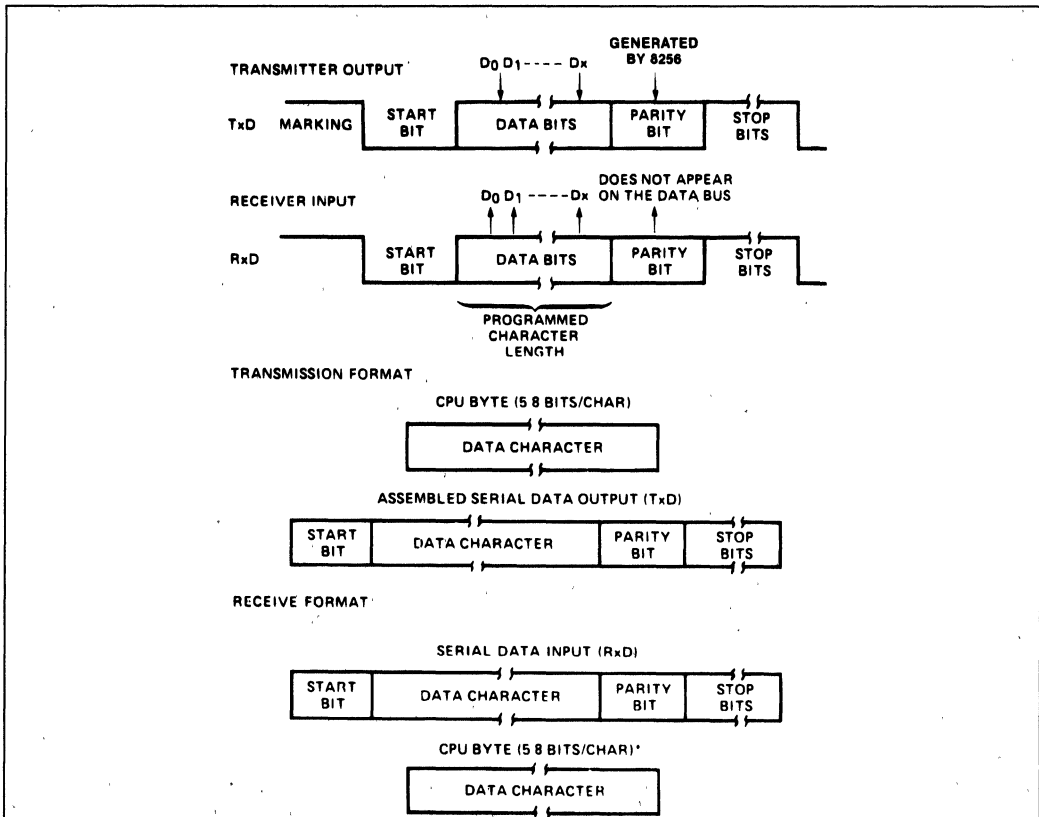
The timer prescaler permits the user to select one of two fundamental timing frequencies for all of the MUART's timers, either 1 kHz or 16 kHz. The frequency selection is made via Command Register 0.

Asynchronous Serial Interface

The asynchronous serial interface of the MUART is a full-duplex double-buffered transmitter and receiver with separate control registers. The standard asynchronous format is used as shown in Figure 3. The operation of the UART section of the MUART is very similar to the operation of the 8251A USART.

Receiver Section of the UART

The serial asynchronous receiver section contains a serial shift register, a receiver buffer register and receiver control logic. The serial input data is clocked into the receive shift register from the RxD pin at the specified baud rate. The sampling actually takes place at the rising edge of RxC, assuming an external clock, or at the rising edge of the internal baud clock. When the receiver is enabled but inactive, the receive logic is sampling RxD at either 32 or 64 times the bit rate, looking for a change from the Mark (high) to the Space (low) state. This is commonly referred to as the start bit search mode. When this state change occurs, the receive logic waits one half of a bit time and then samples RxD again. If RxD is still in the Space state, the receive logic begins to clock in the receive data beginning one bit period later. If RxD has returned to the Mark state (i.e., false start bit), the receive logic will return to the start bit search mode.



210907-3

***NOTE:**

If character length is defined as 5, 6, or 7 bits the unused bits are set to "zero".

Figure 3. Asynchronous Format

Normally the received data is sampled in the center of each bit, however it is possible to adjust the location where the bit is sampled. This feature is controlled by the modification register.

The bit rate of the serial receive data is derived from either the internal baud rate generator or an external clock. When using an external clock, the programmer has a choice of three sampling rates: 1x, 32x, or 64x. Using the internal baud rate generator, the sampling rates are all 64x except for 19.2 Kbps which is 32x.

When the serial shift register clocks in the stop bit, an internal load pulse is generated which transfers the contents of the shift register into the receive buffer. This transfer takes place during the first half of the first stop bit. The load pulse also triggers several other signals relevant to the receive section including Receive Buffer Full (RBF), Parity Error (PE), Overrun Error (OE), and Framing Error (FE). These four status bits are updated after the middle of the first stop bit when the receive buffer has already been latched. Each one of these four status bits are latched. They are reset on the rising edge of the first read pulse (RD) addressed to the status register. A complete description of the status register is given in the section "Description of the Registers".

When the serial receiver is disabled (via bit 6 of Command Register 3) the load pulse is suppressed. The result is that the receive buffer is not loaded with the contents of the shift register, and the RBF, PE, OE, and FE bits in the status register are not updated. Even though the receiver is disabled, the serial shift register will still be clocking in the data from RxD, if any. This means that the receiver will still be synchronized with the start and stop bits. For example, if the receiver is enabled via Command Register 3 in the middle of receiving a serial character, the character will still be assembled correctly. When the receiver is disabled the last character received will remain in the receive buffer. On power-up the value in the receive buffer is undefined.

Whenever a character length of fewer than 8 bits is programmed, the most significant bits of a received character will read as zero. Also, the receiver will only check the first stop bit of any character, regardless of how many stop bits are programmed into the device.

Receive Break Detect

A Receive Break occurs when RxD remains in the space state for one character time, including the parity bit (if any) and the first stop bit. The MUART will set the Break Detect status bit (BD) when it receives a break. The Break Detect status bit is set after the middle of the first stop bit. If the MUART detects a break

it will inhibit the receive buffer load pulse, thus the receive buffer will not be loaded with the null character, and none of the four status bits (PE, OE, FE, and RBF) will be updated. The last character received will remain in the receive buffer. A break detect state has the same effect as disabling the receiver—they both inhibit the load pulse—therefore one can think of the break status as disabling the receiver.

The Break Detect status bit is latched. It is cleared by the rising edge of the read pulse addressed to the status register. If a break occurs, and then the RxD data line returns to the Mark state before the status register is read, the BD status bit will remain set until it is read. If RxD returns to the Mark state after the BD status bit has been read true, the BD status bit will be reset automatically without reading the status register.

The receive break detect logic of the MUART is independent of whether the receiver is enabled or disabled; therefore even if the receiver is disabled the MUART will recognize a break. When the RxD line returns to the Mark state after a break, the 8256 will be in the start bit search mode.

If the receiver interrupt level is enabled, break will generate an interrupt request regardless of whether the receiver is enabled. Another receive interrupt will not be generated until the RxD pin returns to the Mark state.

Transmitter Section of the UART

The serial asynchronous transmitter section of the MUART consists of a transmit buffer, a transmit (shift) register, and the associated control logic. There are two bits in the status register which indicate the status of the transmit buffer and transmit register: TBE (transmit buffer empty) and TRE (transmit register empty).

To transmit a character, a byte is written to the transmit buffer. The transmit buffer should only be written to when $TBE = 1$. When the transmit register is empty and $CTS = 0$, the character will be automatically transferred from the transmit buffer into the transmit register. The data transfer from the transmit buffer to the transmit register takes place during the transmission of the start bit. After this transfer takes place, sometime at the beginning of the transmission of the first data bit, TBE is set to 1.

When the transmitter is idle, both TBE and TRE will be set to 1. After a character is written to the transmit buffer, $TBE = 0$ and $TRE = 1$. This state will remain for a short period of time, then the character will be transferred into the transmit register and the status bits will read $TBE = 1$ and $TRE = 0$. At this point a second character may be written to the transmit buffer

after which $TBE = 0$ and $TRE = 0$. TBE will not be set to 1 again until the transmit register becomes empty and is reloaded with the byte in the transmit buffer.

The transmitter can be disabled only one way—using the \overline{CTS} pin. When $\overline{CTS} = 0$ the transmitter is enabled, and when $\overline{CTS} = 1$ the transmitter is disabled. If the transmitter is idle and \overline{CTS} goes from 0 to 1, disabling the transmitter, TBE and TRE will remain set to 1. Since $TBE = 1$, a character can be written into the transmit buffer. The character will be stored in the transmit buffer but it will not be transferred to the transmit register until \overline{CTS} goes low.

If \overline{CTS} goes from low to high during transmission of a character, the character in transmission will be completed and TxD will return to the Mark state. If the transmitter is full (i.e., TBE and $TRE = 0$), the transmit shift register will be emptied but the transmit buffer will not; therefore $TBE = 0$ and $TRE = 1$.

Transmitter Break Features

The MUART has three transmit break features: Break-In Detect, Transmit Break (TBRK), and Single Character Break (SBRK).

Break-In Detect—A Break-In condition occurs when the MUART is sending a serial message and the transmission line is forced to the space state by the receiving station. Break-In is usually used with half-duplex transmission so that the receiver can signal a break to the transmitter. Port 16 must be connected externally to the transmission line in order to detect a Break-In. If transmission voltage levels other than TTL are used, then proper buffering must be provided so that Port 16 on the MUART will receive the correct polarity and voltage levels.

When Break-In Detect is enabled, Port 16 is polled internally during the transmission of the last or only stop bit of a character. If this pin is low during transmission of the stop bit, the Break Detect status bit (BD) will be set. Break-In Detect and receive Break Detect are OR-ed to set the BD status bit. (Either one can set this bit.) The distinction can be made through the interrupt controller. If the transmit and receive interrupts are enabled, a Break-In will generate an interrupt on level 5, the transmit interrupt, while Break will generate an interrupt on level 4, the receive interrupt. If RxC and TxC are used for the serial bit rates, Break-In cannot be detected.

Transmit Break—This causes the TxD pin to be forced low for as long as the TBRK bit in Command Register 3 is set. While Transmit Buffer is active, data transfers from the Transmit Buffer to the Transmit register will be inhibited.

If both the Transmit Buffer and the Transmit Register are full, and a Transmit Break command is issued

(command register 3, TBRK = 1), the entire character in the Transmit register is sent including the stop bits. TxD is then driven low and the character in the Transmit Buffer remains there until Transmit Break is disabled (command register 3, TBRK = 0). At this time TxD will go high for one bit time and then send the character in the Transmit Buffer.

Single Character Break—This causes TxD to be set low for one character including start bit, data bits, parity bit, and stop bits. The user can send a specific number of Break characters using this feature.

If both the Transmit Buffer and the Transmit Register are full and a Send Break command is issued (command register 3, SBRK = 1) the entire character in the Transmit Register is sent including the stop bits. TxD is driven low for one complete character time followed by a high for two bit times after which the character in the Transmit Buffer is sent.

Modification Register

The modification register is used to alter two standard functions of the receiver (start bit check, and sampling time) and to enable a special indicator flag for half-duplex operation (transmitter status). Disabling start bit check means that the receiver will not return to the start bit search mode if RxD has returned to the Mark state in the center of the start bit. It will simply proceed to assemble a character from the RxD pin regardless of whether it received a false start bit or not. The modification register also allows the user to define where within the receive data bits the MUART will sample.

Parallel I/O

The MUART contains 16 parallel I/O pins which are divided into two 8-bit ports. These two parallel I/O ports (Port 1 and Port 2) can be used for basic digital I/O such as setting a bit high or low, or for byte transfers using a two-wire handshake. Port 1 is bit programmable for input or output, so any combination of the eight bits in Port 1 can be selected as either an input or an output. Port 2 is nibble programmable, which means that all four bits in the upper or lower nibble have to be selected as either inputs or outputs. For byte transfers using the two-wire handshake, Port 2 can either input or output the byte while two bits in Port 1 are used for the handshaking signals.

All of the bits in Port 1 have alternate functions other than I/O ports. As mentioned above, when using the byte handshake mode, two bits on Port 1 are used for the handshaking signals. As a result, these two bits cannot be used for general purpose I/O. The other six bits in Port 1 also have alternate functions if they are not used as I/O ports. Table 1 lists each bit from Port 1 and its corresponding alternate function.

Table 1. Port 1 Control Signals

Pin Symbol	Pin Number	Control Function	Condition
P10 P11	39 38	$\overline{\text{ACK}}$ Control Signals for Port 2 $\overline{\text{OBF}}$ 8-bit Handshake Output	Mode Register $\text{P2C2} - \text{P2C0} = 101$
P10 P11	39 38	$\overline{\text{STB}}$ Control Signals for Port 2 $\overline{\text{IBF}}$ 8-bit Handshake Input	Mode register $\text{P2C2} - \text{P2C0} = 100$
P12	37	Event Counter 2 Clock Input	Mode Register $\text{CT2} = 1$
P13	36	Event Counter 3 Clock Input	Mode Register $\text{CT3} = 1$
P14	35	Internal Baud Rate Generator Clock Output	Mode Word $\text{P2C0} - \text{P2C2} = 111$ Port 1 Control Word P14 = 1 Command Register 2 $\text{B3} - \text{B0} \geq 3\text{H}$
P15	34	Timer 5 Trigger Input	Mode Register $\text{T5C} = 1$
P16	33	Break-In Detection Input	Command Register 1 $\text{BRKI} = 1$
P17	32	External Edge Sensitive Interrupt Input	Command Register 1 $\text{BITI} = 1$

The bits in the Port 1 Control Register select whether the pins on Port 1 are inputs or outputs. The pins on Port 1 are selected as control pins through the other programming registers which are relevant to the control signal. Configuring a bit in Port 1 as a control function overrides its definition in the Port 1 Control Register. If the pins on Port 1 are redefined as control signals, the definition of whether the pin is an input or an output in the Port 1 Control Register remains unchanged. If the pins on Port 1 are converted back to I/O pins, they assume the state which was defined in the Port 1 Control Register.

Each parallel I/O port has a latch and drivers. When the port is in the output mode, the data written to the port is latched and driven on the pins. The data which is latched in the I/O ports remains unchanged unless the port is written to again. Reading the ports, whether the port is an input or output, gates the state at the pins onto the data bus. Writing to an input port has no effect on the pin, but the data is stored in the latch and will be output if the direction on the pin is changed later. Writing to a control pin on Port 1 has the same effect as writing to an input pin. If pins 2, 3, 5, and 6 in Port 1 are used for control signals, the contents of the respective output latches will be read, not the state of the control signals. If pins 0, 1, and 7 on Port 1 are used for control signals, the state of the control signals will be

read. If pin 4 on Port 1 is used as a test output for the internal baud rate, this clock signal will be output through the output latch, thus the information in the output latch will be lost.

The Two-Wire Byte Handshake

The 8256 can be programmed, via the Mode Register, to implement an input or output two-wire byte handshake. When the Mode Register is programmed for the byte handshake, Port 2 is used to transmit or receive the byte, and pins P10 and P11 are used for the two handshake control signals. Figures 4 and 5 show a block diagram and timing signals for the two-wire handshake input and output.

To set up the two-wire handshake output using interrupts one must first program the Mode Register, and then enable the interrupt via the interrupt mask register. An interrupt will not occur immediately after the two-wire handshake interrupt is enabled. The interrupt is triggered by the rising edge of $\overline{\text{ACK}}$. There are two ways to generate the first interrupt. Either the first data byte must be written to Port 2 and completely transferred before an interrupt will occur, or the two-wire handshake interrupt is enabled while $\overline{\text{ACK}}$ is low, and then $\overline{\text{ACK}}$ goes high.

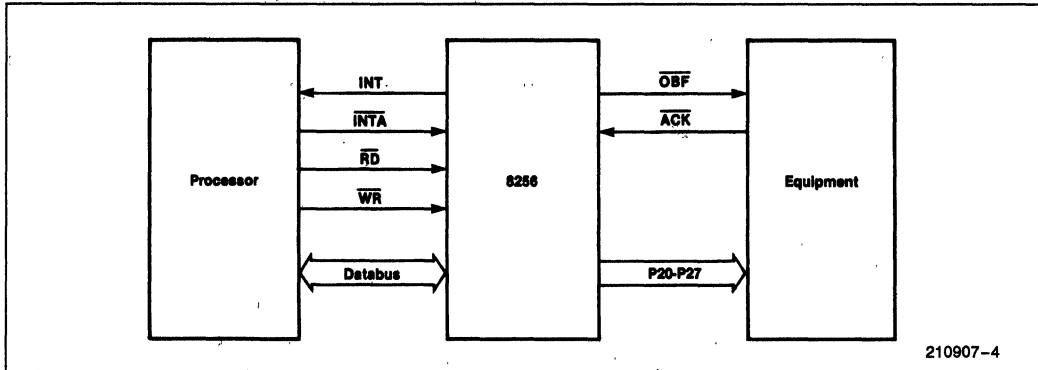


Figure 4. Block Diagram of Handshake Output

Event Counters/Timers

The MUART's five 8-bit programmable counters/timers are binary presettable down counters. The distinction between timer and counter is determined by the clock source. A timer measures an absolute time interval, and its input clock frequency is derived from the MUART's system clock. A counter's input clock frequency is derived from a pulse applied to an external pin. The counter is decremented on the rising edge of this pulse.

When the counters/timers are configured as timers their clock source passes through two dividers: the system clock prescaler, and the timer prescaler. As mentioned before, the system clock prescaler normalizes the internal system clock to 1.024 MHz. The timer prescaler receives this normalized system clock and divides it down to either 1 kHz or 16 kHz, depending on how Command Register 1 is programmed. If more timing resolution is needed the clock frequency can be input externally through the I/O ports.

By programming the Mode Register, four of the 8-bit counters/timers can be cascaded to form two 16-bit counters. Counters/timers 3 and 5 can be cascaded together, and counters/timers 2 and 4 can be cascaded together. Counters/timers 2 and 3 are the lower bytes, while counters/timers 4 and 5 are the upper bytes in the cascaded mode.

Each counter can be loaded with an arbitrary initial value. Timer 5 is the only timer which has a special save register which holds its initial value. Whenever Timer 5 is loaded with an initial value the special save register is also loaded with this value. Timer 5 can be reloaded to its initial value from the detection of a high-to-low transition on Port P15.

The counters are decremented on the first rising edge of the clock after the initial value has been loaded. The setup time for loading the counter when using an external clock is specified in the data sheet. When using internal clocks, the user has no way of knowing the phase relationship of the clock to the write pulse; therefore the timing accuracy is one clock period.

The timers are counting continuously, and an interrupt request is issued any time a single counter or pair of cascaded counters reaches zero. If the timers are going to be used with interrupts, then the programmer should first load the timer with the initial value, then enable the interrupt. If the programmer enables the interrupt first, it is possible that the interrupt will occur before the initial value is loaded. When an interrupt from any one of the timers occurs, the corresponding bit in the interrupt mask register is automatically reset, preventing further interrupt requests from occurring.

The event counters/timers can be used in the following modes of operation:

Timer 1

- Serves as an 8-bit timer.

Event Counter/Timer 2

- Serves as an 8-bit timer or event counter, or cascaded with Timer 4 as a 16-bit timer or event counter.

Event Counter/Timer 3

- Serves as an 8-bit timer or event counter, or cascaded with Timer 5 as a 16-bit timer or event counter, with the additional modes of operation selectable for Timer 5.

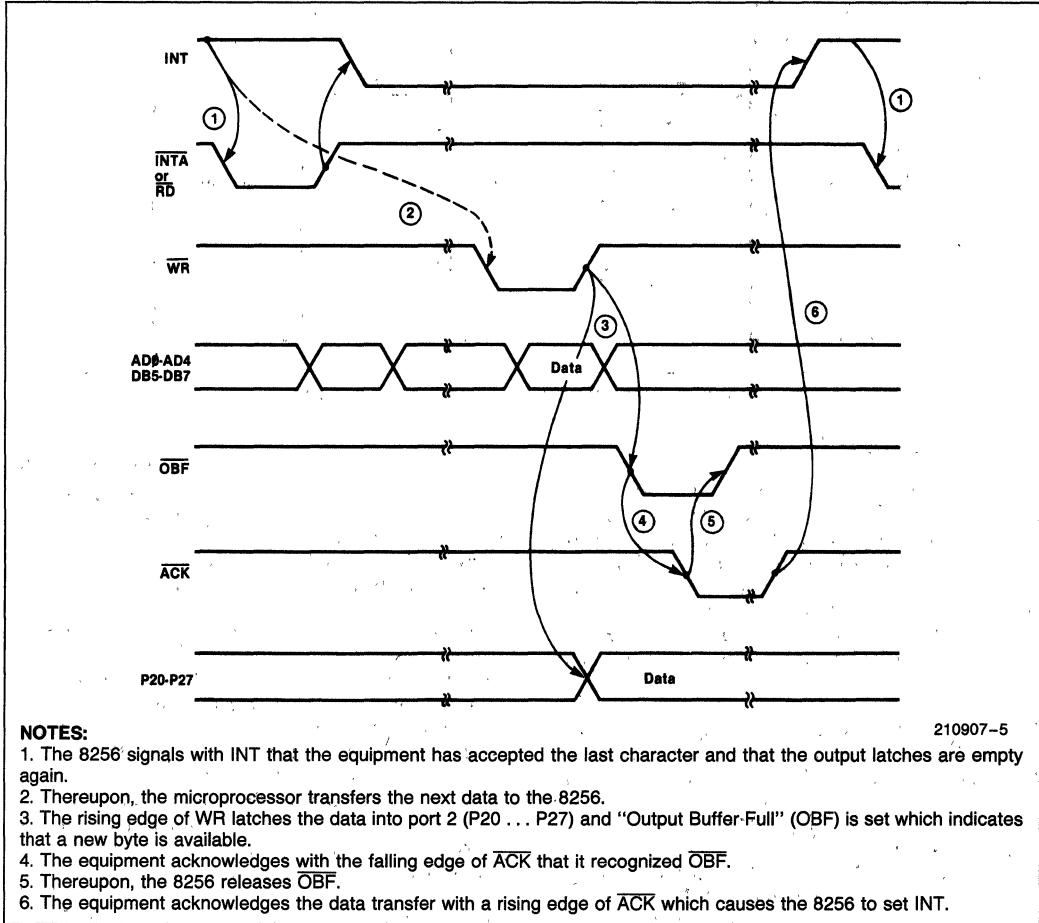


Figure 4a. Timing of Handshake Output

Timer 4

— Serves as an 8-bit timer, or cascaded with Event Counter/Timer 2 as a 16-bit timer or event counter.

Timer 5

1. Non-retriggerable 8-bit timer.
2. Retriggerable 8-bit timer whose initial value is loaded from a save register which starts following the negative transition of an external signal. Subsequent transitions of this signal after the counting has started, reloads the initial value and restarts the counting.
3. Cascaded with Event Counter/Timer 3, non-retriggerable 16-bit timer, which can be loaded with an initial value by two write operations.
4. Cascaded with event counter/timer 3, non-retriggerable 16-bit event counter, which can be loaded with an initial value by two write operations.

5. Cascaded with Event Counter/Timer 3, retriggerable 16-bit timer. The most significant byte (Timer 5) will be loaded with its initial value from the save register, while the last significant byte (Event Counter/Timer 3) will be set to 0FFH automatically. Loading, starting, and retriggering operations follow the same pattern as in 2.
6. Cascaded with Event Counter/Timer 3, retriggerable 16-bit event counter. The most significant byte (Timer 5) will be loaded with its initial value from the save register, while the least significant byte (Event Counter/Timer 3) will be set to 0FFH automatically. Loading, starting, and retriggering operations follow the same pattern as in 2.

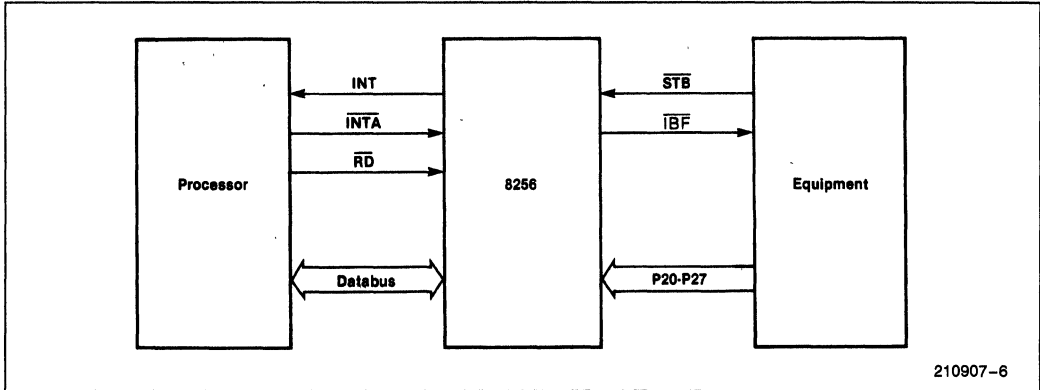


Figure 5. Block Diagram of Handshake Input

Interrupt Controller

In a microcomputer system there are several ways for the CPU to recognize that a peripheral device needs service. Two of the most common ways are the polling method and the interrupt service method.

In the polling method the CPU reads the status of each peripheral to determine whether it needs service. If the peripheral does not need service, the time the CPU spends polling is wasted; therefore this overhead results in increasing the execution time. Some systems must meet a specific request to response time such as a real time signal. In this case the programmer must guarantee that the peripheral is polled at a certain frequency. This polling frequency cannot always easily be met when the CPU must execute a main program as well as subroutines. Usually each peripheral has its own request to response time requirements; therefore the user must establish a priority scheme.

The interrupt method provides certain advantages over the polling method. When a peripheral device needs service it signals the CPU through hardware asynchronously, thus reducing the overhead of polling a device which does not need service. The CPU would typically

finish the instruction it is executing, save the important registers, and acknowledge the peripheral's interrupt request. During the acknowledgment, the CPU reads a vector which directs the CPU to the starting location of the appropriate interrupt service routine. If several interrupt requests occur at the same time, special logic can prioritize the requests so that when the CPU acknowledges the interrupt, the highest priority request is vectored to the CPU.

An interrupt driven system requires additional hardware to control the interrupt request signal, priority, and vectoring. The 8256 integrates this additional hardware onto the chip. The interrupt controller on the MUART is directly compatible with the MCS-85, iAPX-86, iAPX-88, iAPX-186, iAPX-188 family of microcomputer systems, and it can also be used with other microprocessors as well. It contains eight priority levels, however, there are a total of 12 interruptable sources: 10 internal and 2 external. Since there are eight priority levels, only eight interrupts can be used at one time. The assignment of the interrupts used is selected by Command Register 1 and by the mode register. The MUART's interrupt sources have a fixed priority. Table 2 displays how the 12 interrupt sources are mapped into the 8 priority levels.

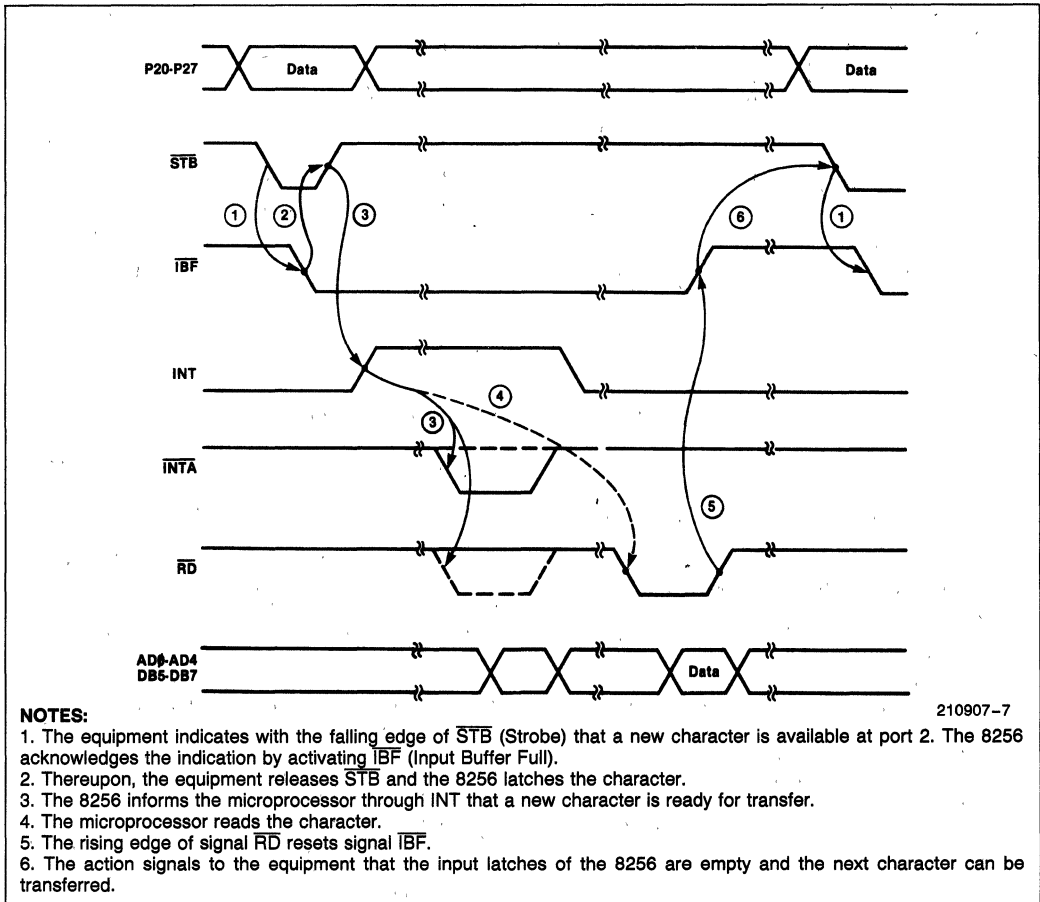


Figure 5a. Timing for Handshake Input

Table 2. Mapping of Interrupt Sources to Priority Levels

Priority	Source
Highest	L0 Timer 1
	L1 Timer 2 or Port Interrupt
	L2 External Interrupt (EXTINT)
	L3 Timer 3 or Timers 3 & 5
	L4 Receiver Interrupt
	L5 Transmitter Interrupt
Lowest	L6 Timer 4 or Timers 2 & 4
	L7 Timer 5 or Port 2 Handshaking

MCS[®]-85/8256 Interrupt Operation

The 8256 is compatible with the 8085 interrupt vectoring method when the 8086 bit in Command Register 1 of the MUART is set to 0. This is the default condition after a hardware reset. The 8085 has five hardware interrupt pins: INTR, RST 7.5, RST 6.5, RST 5.5, and TRAP. When the MUART's interrupt acknowledge feature is enabled (IAE bit 5 Command Register 3 = 1) the MUART's INT Pin 15 should be tied to the 8085's INTR, and both the 8085 and the MUART's INTA pins should be tied together. All of the interrupt pins on the 8085 except INTR automatically vector the program counter to a specified location in memory.

When the INTR pin becomes active (HIGH), assuming the 8085 has interrupts enabled, the 8085 fetches the next instruction from the data bus where it has been placed by the 8256 or some other interrupt controller. This instruction is usually a Call or an RST0 through RST7. Figure 6 shows the memory locations where the 8085 will vector to based on which type of interrupt occurred.

The 8085 can receive an interrupt request any time, since its INTR input is asynchronous. The 8085, however, doesn't always acknowledge an interrupt request immediately. It can accept or disregard requests under software control using the EI (Enable Interrupt) or DI (Disable Interrupt) instructions.

At the end of each instruction cycle, the 8085 examines the state of its INTR pin. If an interrupt request is present and interrupts are enabled, the 8085 enters an interrupt machine cycle. During the interrupt machine cycle the 8085 automatically disables further interrupts until the EI instruction is executed. Unlike normal machine cycles, the interrupt machine cycle doesn't increment the program counter. This ensures that the 8085 can return to the pre-interrupt program location after the interrupt service is completed. The 8085 issues an INTA pulse indicating that it is honoring the request and is ready to process the interrupt.

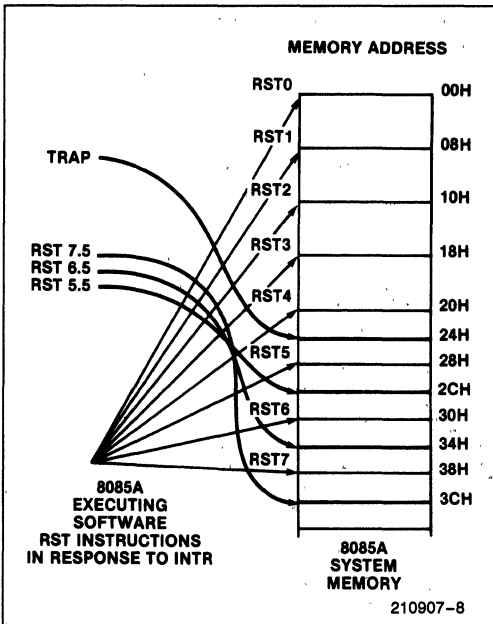


Figure 6. 8085A Hardware and Software RST Branch Locations

The 8256 can now vector program execution to the corresponding service routine. This is done during the first and only INTA pulse. Upon receiving the INTA pulse, the 8256 places the opcode RSTn on the data bus; where n equals 0 through 7 based on the level of the interrupt requested. The RSTn instruction causes the contents of the program counter to be pushed onto the stack, then transfers control to the instruction whose address is eight times n, as shown in Figure 6.

Note that because interrupts are disabled during the interrupt acknowledge sequence, the EI instruction must be executed in either the service routine or the main program before further interrupts can be processed.

For additional information on the 8085 interrupt operation and the RSTn instruction, refer to the *MCS-85 User's Manual*.

IAPX-86/88—8256 Interrupt Operation

The MUART is compatible with the 8086/8088 method of interrupt vectoring when the 8086 bit in Command Register 1 is set to 1. The MUART's INT pin is tied to the 8086/8088 INTR pin, and its INTA pin connected to the 8086/88's INTA pin. Like the 8085, the 8086/8088's INTR pin is also asynchronous so that an interrupt request can occur at any time. The 8086/8088 can accept or disregard requests on the INTR pin under software control instructions. These instructions set or clear the interrupt-enabled flag IF. When the 8086/8088 is powered-on or reset, the IF flag is cleared, disabling external interrupts on INTR.

Although there are some basic similarities, the actual processing of interrupts with an 8086/8088 is different from the 8085. When an interrupt request is present and interrupts are enabled, the 8086/8088 enters its interrupt acknowledge machine cycle. The interrupt acknowledge machine cycle pushes the flag registers onto the stack (as in PUSHF instruction). It then clears the IF flag, which disables interrupts. Finally, the contents of both the code segment register and the instruction pointer are pushed onto the stack. Thus, the stack retains the pre-interrupt flag status and program location which are used to return from the service routine. The 8086/8088 then issues the first of two INTA pulses which signals the 8256 that the 8086/8088 has honored its interrupt request.

The 8256 is now ready to vector program execution to the appropriate service routine. Unlike the 8085 where the first INTA pulse is used to place an instruction on the data bus, the first INTA pulse from the 8086/8088 is used only to signal the 8256 of the honored request. The second INTA pulse causes the 8256 to place a single interrupt vector byte onto the data bus. The 8256 places the interrupt vector bytes 40H through 47H cor-

responding to the level of the interrupt to be serviced. Not used as a direct address, this interrupt vector byte pertains to one of 256 interrupt "types" supported by the 8086/8088 memory. Program execution is vectored to the corresponding service routine by the contents of a specified interrupt type.

All 256 interrupt types are located in absolute memory locations 0 through 3FFH which make up the 8086/8088's interrupt vector table. Each type in the interrupt vector table requires 4 bytes of memory and stores a code segment address and an instruction pointer address. Figure 7 shows a block diagram of the interrupt vector table. When the 8086/8088 receives an interrupt vector byte, it multiplies its value by four to acquire the address of the interrupt type.

Once the service routine is completed the main program may be re-entered by using an IRET (Interrupt Return) instruction. The IRET instruction will pop the pre-interrupt instruction pointer, code segment and flags off the stack. Thus the main program will resume where it was interrupted with the same flag status regardless of changes in the service routine. Note especially that this includes the state of the IF flag; thus interrupts are re-enabled automatically when returning from the service routine. For further information refer to the *iAPX 86,88 User's Manual*.

Using the 8256's Interrupt Controller without INTA

There are several configurations where the 8256 will not have an INTA signal connected to it. Some examples are when using the 8256 with an 8051 or 8048, or when connecting the INT pin on the 8256 to the 8085's RST 7.5, RST 6.5, or RST 5.5 inputs. In these configurations the IAE bit in Command Register 3 is set to 0, and the INTA pin on the 8256 is tied high. When the interrupt occurs the CPU should branch to a service routine which reads the interrupt address register to determine which interrupt request level occurred. The interrupt address register contains the level of the interrupt multiplied by four. Reading the interrupt address register is equivalent in effect to the INTA signal; it clears the INT pin and indicates to the MUART that the interrupt request has been acknowledged. After the CPU reads the value in the interrupt address register, it can add an offset to this value and branch to an interrupt vector table which contains jump instructions to the appropriate interrupt service routines. An 8085 program which demonstrates this routine is given in Figure 8.

Table 3 summarizes the priority levels and the interrupt vectors which the 8256 sends back to the CPU. Note that when using Timer 1 there is a conflict present between RST0 in the 8085 mode and a hardware reset, because both expect instructions starting at address 0H. However, there is a way to distinguish between the two.

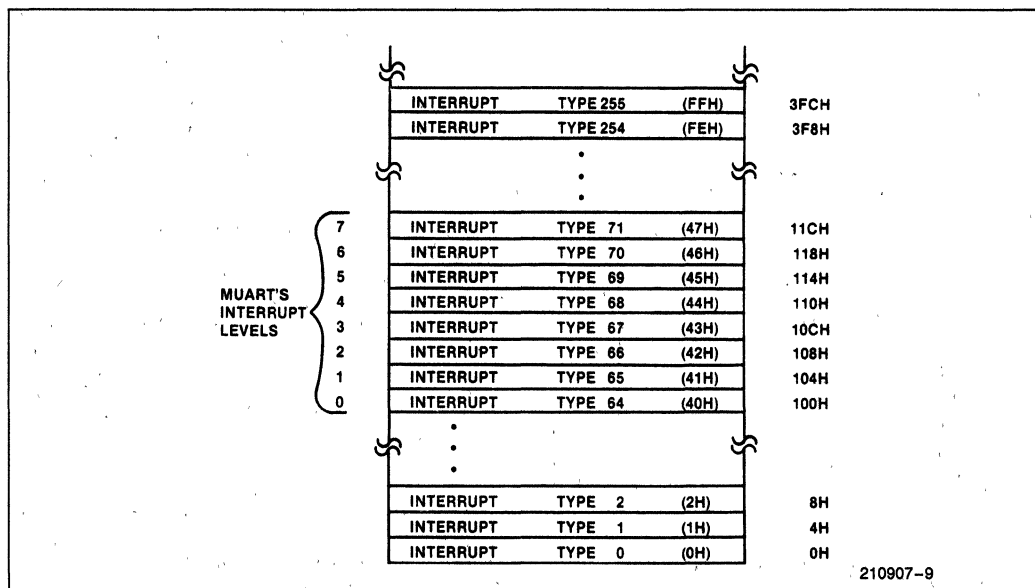


Figure 7. 8086/8088 Interrupt Vector Table

```

INTA: IN  INTADD      ;Read the Interrupt Address Register
      MOV  L, A       ;Put the interrupt address in HL
      XRA  A
      MOV  H, A

      LXI  B, TABLE  ;Load BE with the interrupt table offset
      DAD  B          ;Add the offset to the interrupt address
      PCHL           ;Jump to the interrupt vector table
    
```

Figure 8. Software Interrupt Acknowledge Routine

Table 3. Assignment of Interrupt Levels to Interrupt Sources

Interrupt Level	Restart Command 8085 Mode	Interrupt Vector 8086 Mode	Interrupt Address	Trigger Mode	Sources (Only One Source Can Be Assigned at Any Time)	Selection By
Highest Priority 0	RST0	40H	0H	Edge	Timer 1	—
1	RST1	41H	4H	Edge	Event Counter/Timer 2 or External Interrupt Request on Port 1 P17	Command Word 1 BITI (Bit 2)
2	RST2	42H	8H	Level	Input EXTINT	—
3	RST3	43H	CH	Edge	Event Counter/Timer 3 or Cascaded Event Counters/Timers 3 and 5	Mode Word T35 (Bit 7)
4	RST4	44H	10H	Edge	Serial Receiver	—
5	RST5	45H	14H	Edge	Serial Transmitter	—
6	RST6	46H	28H	Edge	Timer 4 or Cascaded Event Counters/Timers 2 and 4	Mode Word T24 (Bit 6)
7 Lowest Priority	RST7	47H	1CH	Edge	Timer 5 or Port 2 with Handshaking Interrupt Request	Mode Word P2C2-P2C0 (Bits 2 . . . 0)

NOTE:

1. If no interrupt requests are pending and $\overline{\text{INTA}}$ cycle occurs, interrupt level 2 will be the default value vectored to the CPU.

After a hardware reset, all control registers are reset to a value of 0H; therefore when using Timer 1, Reset and RST0 can be distinguished by reading one of the control registers of the 8256 which has not been programmed with a value of 0H. The control registers will contain the previously programmed values if RST0 occurs.

Interrupt Registers

The 8256's interrupt controller has several registers associated with it: an Interrupt Mask Register, an Interrupt Address Register, an Interrupt Request Register, an Interrupt Service Register, and a Priority Controller. Only the Interrupt Mask Registers and the Interrupt Address Register can be accessed by the user.

Interrupt Mask Registers

The Interrupt Mask Registers consist of two write registers—the Set Interrupts Register and Reset Interrupts Register, and one read register—the Interrupt Enable Register. Each one of the eight levels of interrupts may be individually enabled or disabled through these registers. Writing a one to any of the bits in the Set Interrupts Register enables the corresponding interrupt level, while writing a one to a bit in the Reset Interrupts Register disables the corresponding interrupt level. Reading the Interrupt Enable Register allows the user to determine which interrupt levels are enabled. The bits which are set to one in the Interrupt Enable Register correspond to the levels which are enabled. All of the interrupt levels will remain enabled until disabled by the Reset Interrupts Register except the counter/timer interrupts which automatically disable themselves when they reach zero.

Interrupt requests occurring when the corresponding interrupt level is disabled are lost. An interrupt will only occur if the interrupt is enabled before the interrupt request occurs.

Interrupt Address Register

The Interrupt Address Register contains an identifier for the currently requested interrupt level. The numerical value in this register is equal to the interrupt level multiplied by four. It can be used in lieu of an \overline{INTA} signal to vector the CPU to the appropriate interrupt service routine. Reading this register has the same effect as the \overline{INTA} pulse: it clears the INT pin and indicates an interrupt acknowledgement to the MUART. If the Interrupt Address Register is read while no interrupts are pending, the external interrupt EXTINT will be the default value, 08H.

Interrupt Request Register

The Interrupt Request Register latches all pending interrupt requests unless they are masked off. The request is set whenever the associated event occurs.

Interrupt Service Register

In the fully nested mode of operation, every interrupt request which is granted service is entered into this register. The appropriate bit will be set whenever the interrupt is acknowledged by \overline{INTA} or by reading the Interrupt Address Register. At the same time, the corresponding bit in the Interrupt Request Register is reset. The Interrupt Service Register bit remains set until the microcomputer transfers the End Of Interrupt command (EOI) to the device by writing it into Command Register 3. In the normal mode the bits in the Interrupt Service Register are never set.

Priority Controller

The priority controller selects the highest priority request in the Interrupt Request Register from up to eight requests pending. If the \overline{INTA} signal is enabled and becomes active, the priority controller will cause the highest priority level in the Interrupt Request Register to be vectored back to the CPU, regardless of whether the 8256 is in the normal mode or the nested mode. In the normal mode, if any bits are set in the Interrupt Request Register, the INT pin is activated. The highest priority level in the Interrupt Request Register will be transferred to the Interrupt Address Register at the same time the interrupt request occurs. In the Fully Nested mode, the priorities of all pending requests are compared to the priorities in the Interrupt Service Register. If there is a higher priority in the Interrupt Request Register than in the Interrupt Service Register, the INT signal will be activated and the new interrupt level will be loaded into the Interrupt Address Register.

Interrupt Modes

There are two modes of operation for the interrupt controller: a normal mode and a fully nested mode. In the normal mode the CPU should only be a maximum of one interrupt level deep; therefore, the CPU can be interrupted only while in the main program and not while in an interrupt service routine. In the fully nested mode it is possible for the CPU to be nested up to eight interrupt levels deep. Using the fully nested mode, the MUART will activate the INT pin only when a higher priority than the one in service is requested. The fully nested mode is used to protect high priority interrupt service routines from being interrupted by equal or lower priority requests.

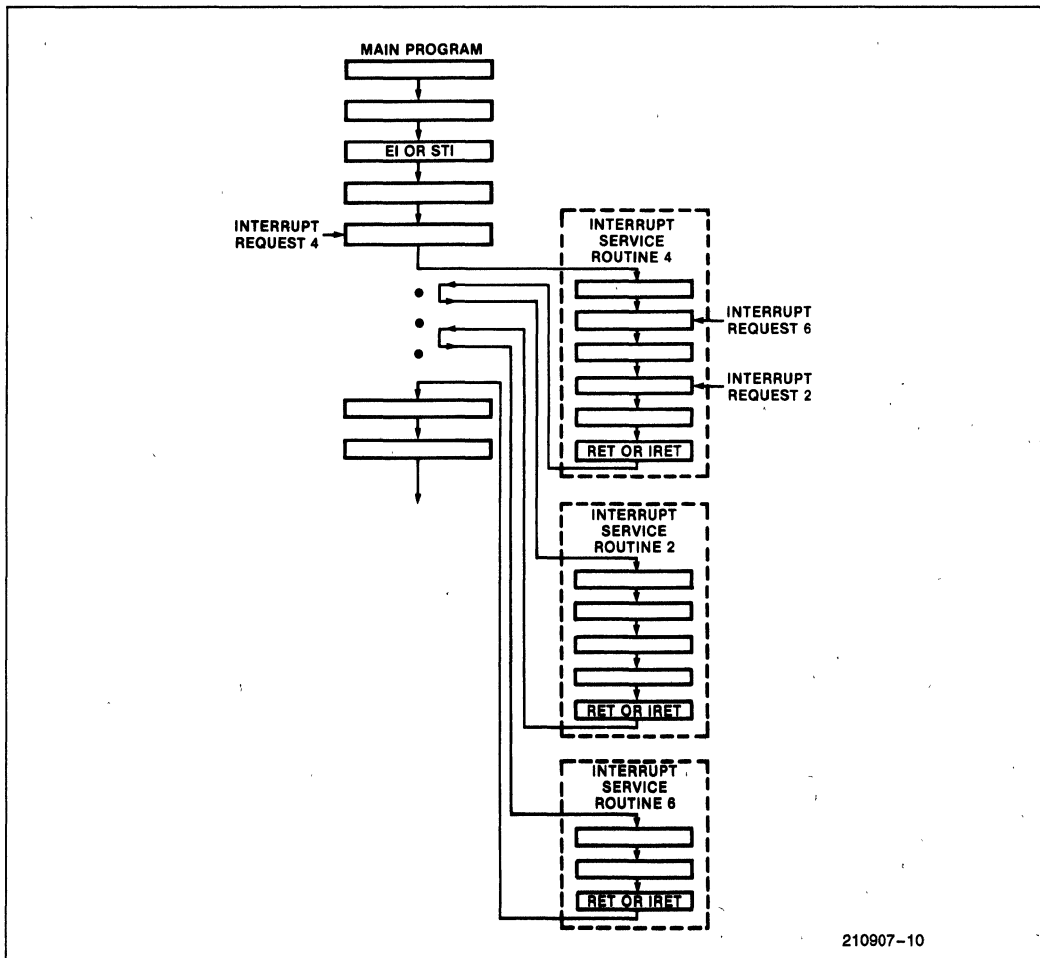
Normal Mode

In the normal mode of operation the 8256 will activate the INT pin whenever any of the bits in the Interrupt Request Register are set. The bits in the Interrupt Request Register can be set only if the corresponding interrupts are enabled. If more than one interrupt request bit is set, the MUART will always place the highest priority level in the Interrupt Address Register and vector this level to the CPU during an $\overline{\text{INTA}}$ cycle. When the CPU acknowledges the interrupt request, using either the $\overline{\text{INTA}}$ signal or by reading the Interrupt Address Register, the corresponding Interrupt Request Register bit is reset. Since the Interrupt Service Register bits are never set, there is no indication in the MUART that an interrupt service routine is in progress. Therefore, the priority controller will interrupt the

CPU again if any of the interrupt request bits are set, regardless of whether the next request is a higher, lower, or equal priority.

The implied way to design a program using the normal mode is to have the CPU's interrupt flag enabled during portions of the main program, but to leave the interrupt flag disabled while the CPU is executing code in an interrupt service routine. This way, the CPU can never be interrupted in an interrupt service routine. Upon completion of an interrupt service routine the program can enable the CPU's interrupt flag, then return to the main program.

Figure 9 shows an example of how the normal mode of interrupts may operate. As the CPU begins executing code in the main program, certain I/O ports, variables,



210907-10

Figure 9. Normal Interrupt Mode Example

and arrays need to be initialized. During this time the CPU's interrupt flag is disabled. Once the program has completed the initialization routine and can accept an interrupt, the interrupt flag is enabled. In the 8085 this is done with the assembly language instruction EI, and on the 8086 with STI.

A short time later, an interrupt request comes in on Level 4. Since the CPU's interrupt flag is enabled, the interrupt acknowledge signal is activated and the CPU branches off to Interrupt Service Routine 4. While the CPU is executing code in Interrupt Service Routine 4, an interrupt request comes in on Level 6 and then a short time later on Level 2. The 8256 activates the INT signal; however, the CPU ignores this because its interrupt flag is disabled. Upon returning to the main program the interrupt flag is enabled. When the interrupt acknowledge signal is activated, the MUART places the highest priority interrupt request on the data bus regardless of the order in which the requests came in. Therefore, during the interrupt acknowledge the MUART vectors the indirect address for Interrupt Level 2. The INT signal is not cleared after the acknowledge because there is still a pending interrupt.

The normal mode of operation is advantageous in that it simplifies programming and lowers code requirements within interrupt routines; however, there are also several disadvantages. One disadvantage is that the interrupt response time for higher priority interrupts may be excessive. For example, if the CPU is executing code in an interrupt service routine during a higher priority request, the CPU will not branch off to the higher priority service routine until the current interrupt service routine is completed. This delay time may not be acceptable for interrupts such as the serial receiver or a real time signal. For these cases the MUART provides the nested mode.

Nested Mode

In the nested mode of operation, whenever a bit in the Interrupt Request Register is set, the Priority Controller compares the Interrupt Request Register to the Interrupt Service Register. If the bit set in the Request Register is of a higher priority than the highest priority bit set in the Service Register, the MUART will activate the INT signal and update the Interrupt Address Register. If the bit in the Request Register is of equal or lower priority than the highest priority bit set in the Service Register, the INT signal will not be activated. When an INTA signal is activated or the Interrupt Address Register is read, the corresponding bit in the Request Register which caused the INT signal to be asserted is reset and set in the Service Register. When an EOI (End Of Interrupt) command is issued, the highest priority bit in the Service Register is reset.

Figure 10 shows an example of the program flow using the nested mode of interrupts. During the main program an interrupt request is generated from Level 4. Since the interrupt flag is enabled, the interrupt acknowledge signal is activated, and the microprocessor is vectored to Service Routine 4. During Service Routine 4, Level 2 requests an interrupt. Since Level 2 is a higher priority than Level 4, the 8256 activates its INT signal. An interrupt acknowledge is not generated because the interrupt flag is disabled. This section of code in Service Routine 4 is protected and cannot be interrupted. A protected section of code may reinitialize a timer, take a sample, or update a global variable. When the interrupt flag is enabled the microprocessor acknowledges the interrupt and vectors into Service Routine 2. Service Routine 2 immediately enables the interrupt flag because it does not have a protected section of code. During Service Routine 2, Interrupt Request 6 is generated. However, the MUART will not interrupt the microprocessor until service routines 2 and 4 have issued the EOI command.

Edge Triggering

The MUART has a maximum of two external interrupts—EXTINT and P17. EXTINT is a dedicated interrupt pin which is level triggered, where P17 is either an I/O port or an edge triggered interrupt. If P17 is selected as an interrupt through Command Register 1 and its interrupt level is enabled, it will generate an interrupt when the level on this pin changes from low to high. The edge triggered mode incorporates an edge lockout feature. This means that after the rising edge of an interrupt request and the acknowledgment of the request, the positive level on P17 won't generate further interrupts. Before another interrupt can be generated, P17 must return low.

External devices which generate a pulse for an interrupt request can use the edge triggered mode as long as the minimum high time specified in the data sheet is met.

Level Triggering

The external interrupt (EXTINT pin 16) is the only level triggered interrupt on the MUART. The 8256 will recognize any active (high) level on the EXTINT as an interrupt request. The EXTINT pin must stay high until a short time after the rising edge of the first INTA pulse. If the voltage level on the EXTINT pin is high, then goes low, the bit in the interrupt request register corresponding to EXTINT will be reset.

In the normal mode of operation if EXTINT is still high after the INTA pulse has been activated, the INT

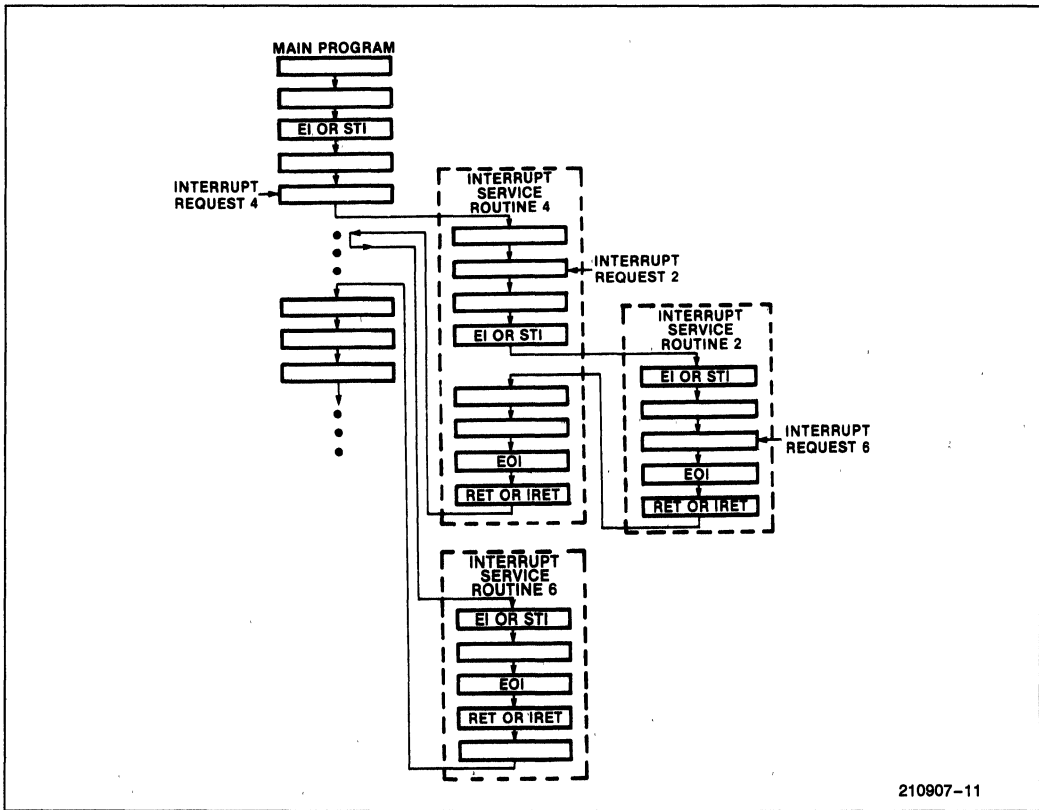


Figure 10. Fully Nested Interrupt Mode Example

signal will remain active. If the microprocessor's interrupt flag is immediately re-enabled, another interrupt will occur. Unless repeated interrupt generation is desired, the programmer should not re-enable the CPU's interrupt flag until EXTINT has gone low.

In the nested mode of operation, if EXTINT is still high after the INTA pulse has been activated, the INT signal will not be reactivated. This is because in the nested mode only a higher priority interrupt than the one being serviced can activate the INT signal. The EXTINT pin should go inactive (low) before the EOI command is issued if an immediate interrupt is not desired.

Depending upon the particular design and application, the EXTINT pin has a number of uses. For example, it can provide repeated interrupt generation in the normal mode. This is useful in cases when a service routine needs to be continually executed until the interrupt request goes inactive. Another use of the EXTINT pin is

that a number of external interrupt requests can be wire-ORed. This can't be done using P17, for if a device makes an interrupt request while P17 is high (from another request), its transition will be shadowed. Note that when a wire-OR'ed scheme is used, the actual requesting device has to be determined by the software in the service routine.

Cascading the MUART's Interrupt Controller

Cascading the MUART's interrupt controller is necessary in an interrupt driven system which contains more than one interrupt controller, such as a system using more than one MUART, or using a MUART with another interrupt controller like the 8259A. For a system which uses several MUART's, one of them is tied directly to the microprocessor's INT and INTA pins, while the remaining MUARTs are daisy-chained using the EXTINT and INT pins. This is shown in Figure 11.

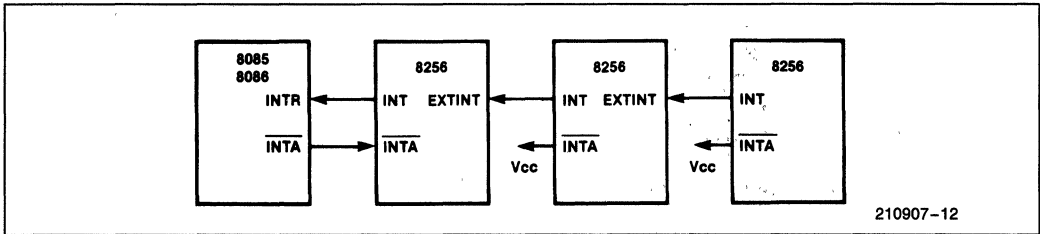


Figure 11. Cascading the MUART's Interrupt Controller

Using the configuration in Figure 11, when the microprocessor receives an interrupt, it generates an interrupt acknowledge and branches into an interrupt service routine. For the interrupt service routine of the external interrupt, EXTINT Level 2, the microprocessor will read the next MUART's interrupt address register and branch to the appropriate service routine. In effect, this would be a software interrupt acknowledge. An

example of this type of interrupt acknowledge is given in Figure 8. If the last MUART in the chain indicated an external interrupt, the microprocessor would simply return to the main program; however, this would be an error condition caused by a spurious interrupt. A flow chart of the software to handle cascaded interrupts is given in Figure 12.

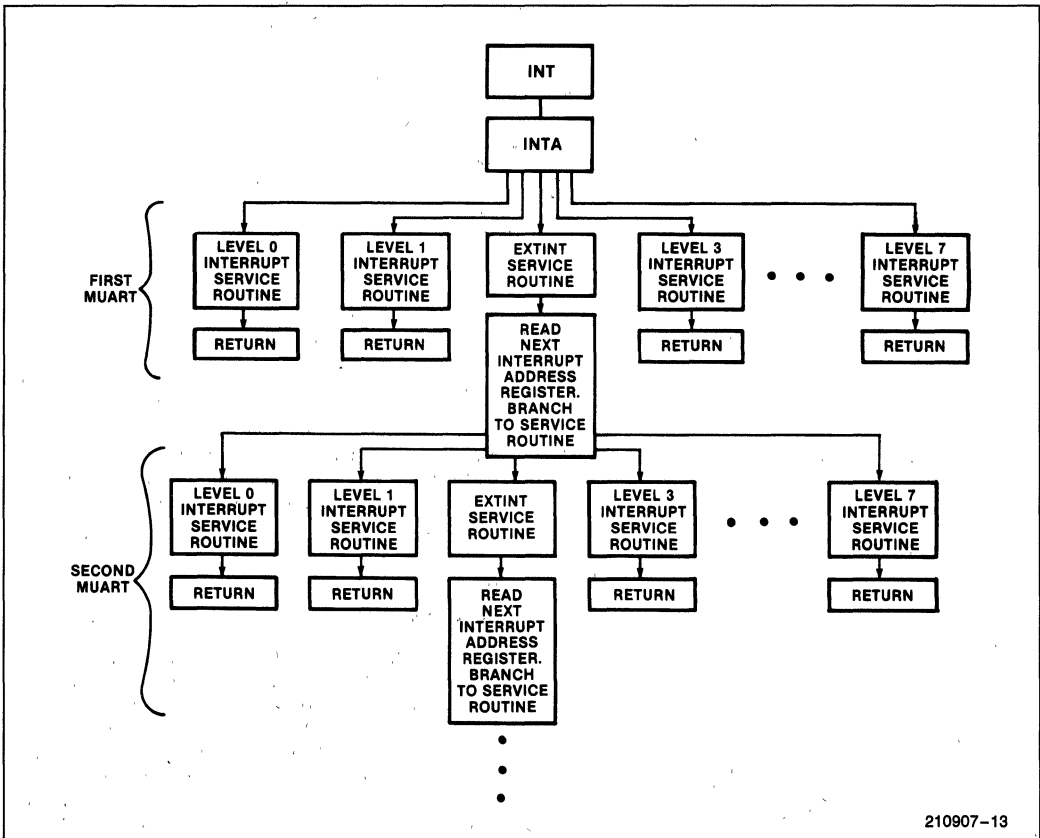


Figure 12. Flow Chart to Resolve Interrupt Request When Cascading MUART Interrupt Controllers

Some consideration should be given to the priority of the interrupts when cascading MUARTs. If all of the MUART's Level 0 and Level 1 interrupts are disabled, the highest priority interrupt is the EXTINT. In this case the last MUART in the chain would have the longest time to propagate back to the CPU. If, however, Level 0 or Level 1 interrupts were enabled, the closer to the microprocessor the MUART is, the higher the priority these two levels would have.

When using the 8256 interrupt controller along with some other interrupt controller, such as the 8259A, the MUART's INT signal would simply be tied to one of the interrupt controller's request inputs. The service routine for the MUART's interrupt request would initially perform the software interrupt acknowledge before servicing the MUART's interrupt request. A block diagram of this configuration is given in Figure 13.

Polling the MUART

If interrupts are not used, the only other way to control the MUART is to poll it. It is still possible to use the priority structure of the MUART with polling. In this mode of operation the MUART's INT signal (Pin 15) is not used, and the \overline{INTA} pin is tied high. Since the INT pin's level is duplicated in the MSB of the Status Register, a program can poll this bit. When it becomes set, the program could read the Interrupt Address Register to determine the cause. Either the normal or nested mode of operation can be used. Note that the functions used with this polled method must have their interrupts enabled.

It is also possible to poll the counters/timers, parallel I/O, and UART separately. To control the UART, one could poll the Status Register. Byte handshakes with the parallel I/O can be controlled by polling Port 1. Finally, each counter/timer has its own register which can be polled.

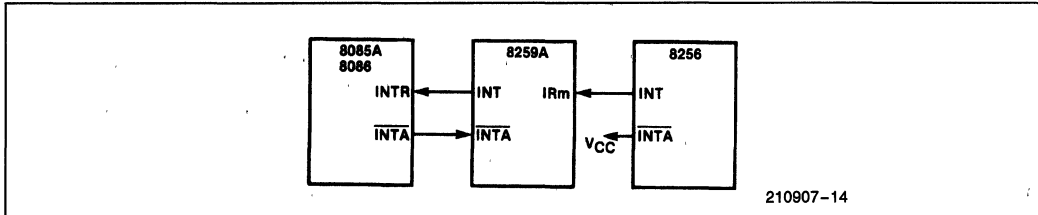


Figure 13. Connecting the 8256 to the 8259A Interrupt Controller

PIN DESCRIPTIONS

Symbol	Pin No.	Type	Name and Function
AD0-AD4 DB5-DB7	1-5 6-8	I/O	ADDRESS/DATA: Three-state address/data lines which interface to the lower 8 bits of the microprocessor's multiplexed address/data bus. The 5-bit address is latched on the falling edge of ALE. In the 8-bit mode, AD0-AD3 are used to select the proper register, while AD1-AD4 are used in the 16-bit mode. AD4 in the 8-bit mode is ignored as an address, while AD0 in the 16-bit mode is used as a second chip select, active low.
ALE	9	I	ADDRESS LATCH ENABLE: Latches the 5 address lines on AD0-AD4 and CS on the falling edge.
RD	10	I	READ CONTROL: When this signal is low, the selected register is gated onto the data bus.
WR	11	I	WRITE CONTROL: When this signal is low, the value on the data bus is written into the selected register.
RESET	12	I	RESET: An active high pulse on this pin forces the chip into its initial state. The chip remains in this state until control information is written.
CS	13	I	CHIP SELECT: A low on this signal enables the MUART. It is latched with the address on the falling edge of ALE, and RD and WR have no effect unless CS was latched low during the ALE cycle.
INTA	14	I	INTERRUPT ACKNOWLEDGE: If the MUART has been enabled to respond to interrupts, this signal informs the MUART that its interrupt request is being acknowledged by the microprocessor. During this acknowledgement the MUART puts an RSTn instruction on the data bus for the 8-bit mode or a vector for the 16-bit mode.
INT	15	O	INTERRUPT REQUEST: A high signals the microprocessor that the MUART needs service.
EXTINT	16	I	EXTERNAL INTERRUPT: An external device can request interrupt service through this input. The input is level sensitive (high), therefore it must be held high until an INTA occurs or the interrupt address register is read.
CLK	17	I	SYSTEM CLOCK: The reference clock for the baud rate generator and the timers.
RxC	18	I/O	RECEIVE CLOCK: If the baud rate bits in Command Register 2 are all 0, this pin is an input which clocks serial data into the RxD pin on the rising edge of RxC. If baud rate bits in Command Register 2 are programmed from 1-0FH, this pin outputs a square wave whose rising edge indicates when the data on RxD is being sampled. This output remains high during start, stop, and parity bits.
RxD	19	I	RECEIVE DATA: Serial data input.

PIN DESCRIPTIONS (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{\text{CTS}}$	21	I	CLEAR TO SEND: This input enables the serial transmitter. If 1, 1.5, or 2 stop bits are selected, $\overline{\text{CTS}}$ is level sensitive. As long as $\overline{\text{CTS}}$ is low, any character loaded into the transmitter buffer register will be transmitted serially. A single negative going pulse causes the transmission of a single character previously loaded into the transmitter buffer register. If a baud rate from 1-0FH is selected, $\overline{\text{CTS}}$ must be low for at least $\frac{1}{32}$ of a bit, or it will be ignored. If the transmitter buffer is empty, this pulse will be ignored. If this pulse occurs during the transmission of a character up to the time where $\frac{1}{2}$ of the first (or only) stop bit is sent out, it will be ignored. If it occurs afterwards, but before the end of the stop bits, the next character will be transmitted immediately following the current one. If $\overline{\text{CTS}}$ is still high when the transmitter register is sending the last stop bit, the transmitter will enter its idle state until the next high-to-low transition on $\overline{\text{CTS}}$ occurs. If 0.75 stop bits is chosen, the $\overline{\text{CTS}}$ input is edge sensitive. A negative edge on $\overline{\text{CTS}}$ results in the immediate transmission of the next character. The length of the stop bits is determined by the time interval between the beginning of the first stop bit and the next negative edge on $\overline{\text{CTS}}$. A high-to-low transition has no effect if the transmitter buffer is empty or if the time interval between the beginning of the stop bit and next negative edge is less than 0.75 bits. A high or a low level or a low-to-high transition has no effect on the transmitter for the 0.75 stop bit mode.
TxC	22	I/O	TRANSMIT CLOCK: If the baud rate bits in command register 2 are all set to 0, this input clocks data out of the transmitter on the falling edge. If baud rate bits are programmed for 1 or 2, this input permits the user to provide a 32x or 64x clock which is used for the receiver and transmitter. If the baud rate bits are programmed for 3-0FH, the internal transmitter clock is output. As an output it delivers the transmitter clock at the selected bit rate. If $1\frac{1}{2}$ or 0.75 stop bits are selected, the transmitter divider will be asynchronously reset at the beginning of each start bit, immediately causing a high-to-low transition on TxC. TxC makes a high-to-low transition at the beginning of each serial bit, and a low-to-high transition at the center of each bit.
TxD	23	O	TRANSMIT DATA: Serial data output.
P27-P20	24-31	I/O	PARALLEL I/O PORT 2: Eight bit general purpose I/O port. Each nibble (4 bits) of this port can be either an input or an output. The outputs are latched whereas the input signals are not. Also, this port can be used as an 8-bit input or output port when using the two-wire handshake. In the handshake mode both inputs and outputs are latched.
P17-P10	32-39	I/O	PARALLEL I/O PORT 1: Each pin can be programmed as an input or an output to perform general purpose I/O. All outputs are latched whereas inputs are not. Alternatively these pins can serve as control pins which extend the functional spectrum of the chip.
GND	20	PS	GROUND: Power supply and logic ground reference.
V _{CC}	40	PS	POWER: +5V power supply.

DESCRIPTION OF THE REGISTERS

The following section will provide a description of the registers and define the bits within the registers where appropriate. Table 4 lists the registers and their addresses.

Command Register 1

L1	L0	S1	S0	BRKI	BITI	8086	FRQ
(OR)				(OW)			

FRQ—Timer Frequency Select

This bit selects between two frequencies for the five timers. If FRQ = 0, the timer input frequency is 16 kHz (62.5 μs). If FRQ = 1, the timer input frequency is 1 kHz (1 ms). The selected clock frequency is shared by all the counter/timers enabled for timing; thus, all timers must run with the same time base.

8086—8086 Mode Enable

This bit selects between 8085 mode and 8086/8088 mode. In 8085 mode (8086 = 0), A0 to A3 are used to

Table 4. MUART Registers

Read Registers								Write Registers											
8085 Mode: AD3 AD2 AD1 AD0								8086 Mode: AD4 AD3 AD2 AD1											
L1	L0	S1	S0	BRKI	BITI	8086	FRQ	0	0	0	0	L1	L0	S1	S0	BRKI	BITI	8086	FRQ
Command 1								Command 1											
PEN	EP	C1	C0	B3	B2	B1	B0	0	0	0	1	PEN	EP	C1	C0	B3	B2	B1	B0
Command 2								Command 2											
0	RxE	IAE	NIE	0	SBRK	TBRK	0	0	0	1	0	SET	RxE	IAE	NIE	END	SBRK	TBRK	RST
Command 3								Command 3											
T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0	0	0	1	1	T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0
Mode								Mode											
P17	P16	P15	P14	P13	P12	P11	P10	0	1	0	0	P17	P16	P15	P14	P13	P12	P11	P10
Port 1 Control								Port 1 Control											
L7	L6	L5	L4	L3	L2	L1	L0	0	1	0	1	L7	L6	L5	L4	L3	L2	L1	L0
Interrupt Enable								Set Interrupts											
D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	0	L7	L6	L5	L4	L3	L2	L1	L0
Interrupt Address								Reset Interrupts											
D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	1	D7	D6	D5	D4	D3	D2	D1	D0
Receiver Buffer								Transmitter Buffer											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
Port 1								Port 1											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	0	1	D7	D6	D5	D4	D3	D2	D1	D0
Port 2								Port 2											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Timer 1								Timer 1											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	1	1	D7	D6	D5	D4	D3	D2	D1	D0
Timer 2								Timer 2											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	0	0	D7	D6	D5	D4	D3	D2	D1	D0
Timer 3								Timer 3											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	0	1	D7	D6	D5	D4	D3	D2	D1	D0
Timer 4								Timer 4											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	1	1	D7	D6	D5	D4	D3	D2	D1	D0
Timer 5								Timer 5											
INT	RBF	TBE	TRE	BD	PE	OE	FE	1	1	1	1	0	RS4	RS3	RS2	RS1	RS0	TME	DSC
Status								Modification											

address the internal registers, and an $RSTn$ instruction is generated in response to the first \overline{INTA} . In 8086 mode ($8086 = 1$), A1 to A4 are used to address the internal registers, and A0 is used as an extra chip select (A0 must equal zero to be enabled). The response to \overline{INTA} is for 8086 interrupts where the first \overline{INTA} is ignored, and an interrupt vector (40H to 47H) is placed on the bus in response to the second \overline{INTA} .

BITI—Interrupt on Bit Change

This bit selects between one of two interrupt sources on Priority Level 1, either Counter/Timer 2 or Port 1 P17 interrupt. When this bit equals 0, Counter/Timer 2 will be mapped into Priority Level 1. If BITI equals 0 and Level 1 interrupt is enabled, a transition from 1 to 0 in Counter/Timer 2 will generate an interrupt request on Level 1. When BITI equals 1, Port 1 P17 external edge triggered interrupt source is mapped into Priority Level 1. In this case if Level 1 is enabled, a low-to-high transition on P17 generates an interrupt request on Level 1.

BRKI—Break-In Detect Enable

If this bit equals 0, Port 1 P16 is a general purpose I/O port. When BRKI equals 1, the Break-In Detect feature is enabled on Port 1 P16. A Break-In condition is present on the transmission line when it is forced to the start bit voltage level by the receiving station. Port 1 P16 must be connected externally to the transmission line in order to detect a Break-In. A Break-In is polled by the MUART during the transmission of the last or only stop bit of a character.

A Break-In Detect is OR-ed with Break Detect in Bit 3 of the Status Register. The distinction can be made through the interrupt controller. If the transmit and receive interrupts are enabled, a Break-In will generate an interrupt on Level 5, the transmit interrupt, while Break will generate an interrupt on Level 4, the receive interrupt.

S0, S1—Stop Bit Length

S1	S0	Stop Bit Length
0	0	1
0	1	1.5
1	0	2
1	1	0.75

The relationship of the number of stop bits and the function of input CTS is discussed in the Pin Description section under "CTS".

L0, L1—Character Length

L1	L0	Character Length
0	0	8
0	1	7
1	0	6
1	1	5

Command Register 2

PEN	EP	C1	C0	B3	B2	B1	B0
(1R)				(1W)			

Programming bits 0 . . . 3 with values from 3H to FH enables the internal baud rate generator as a common clock source for the transmitter and receiver and determines its divider ratio.

Programming bits 0 . . . 3 with values of 1H or 2H enables input TxC as a common clock source for the transmitter and receiver. The external clock must provide a frequency of either 32x or 64x the baud rate. The data transmission rates range from 0 . . . 32 Kbaud.

If bits 0 . . . 3 are set to 0, separate clocks must be input to pin RxC for the receiver and pin TxC for the transmitter. Thus, different baud rates can be used for transmission and reception. In this case, prescalers are disabled and the input serial clock frequency must match the baud rate. The input serial clock frequency can range from 0 to 1.024 MHz.

B0, B1, B2, B3—Baud Rate Select

These four bits select the bit clock's source, sampling rate, and serial bit rate for the internal baud rate generator.

B3	B2	B1	B0	Baud Rate	Sampling Rate
0	0	0	0	$\overline{TxC}, \overline{RxC}$	1
0	0	0	1	$\overline{TxC}/64$	64
0	0	1	0	$\overline{TxC}/32$	32
0	0	1	1	19200	32
0	1	0	0	9600	64
0	1	0	1	4800	64
0	1	1	0	2400	64
0	1	1	1	1200	64
1	0	0	0	600	64
1	0	0	1	300	64
1	0	1	0	200	64
1	0	1	1	150	64
1	1	0	0	110	64
1	1	0	1	100	64
1	1	1	0	75	64
1	1	1	1	50	64

The following table gives an overview of the function of pins TxC and RxC:

Bits 3 to 0 (Hex.)	TxC	RxC
0	Input: 1 × baud rate clock for the transmitter	Input: 1 × baud rate clock for the receiver
1, 2	Input 32 × or 64 × baud rate for transmitter and receiver	Output: receiver bit clock with a low-to-high transition at data bit sampling time. Otherwise: high level
3 to F	Output: baud rate clock of the transmitter	Output: as above

As an output, RxC outputs a low-to-high transition at sampling time of every data bit of a character. Thus, data can be loaded, e.g., into a shift register externally. The transition occurs only if data bits of a character are present. It does not occur for start, parity, and stop bits (RxC = high).

As an output, TxC outputs the internal baud rate clock of the transmitter. There will be a high-to-low transition at every beginning of a bit.

C0, C1—System Clock Prescaler (Bits 4, 5)

Bits 4 and 5 define the system clock prescaler divider ratio. The internal operating frequency of 1.024 MHz is derived from the system clock.

C1	C0	Divider Ratio	Clock at Pin CLK
0	0	5	5.12 MHz
0	1	3	3.072 MHz
1	0	2	2.048 MHz
1	1	1	1.024 MHz

EP—Even Parity (Bit 6)

EP = 0: Odd parity

EP = 1: Even parity

PEN—Parity Enable (Bit 7)

Bit 7 enables parity generation and checking.

PEN = 0: No parity bit

PEN = 1: Enable parity bit

The parity bit according to Command Register 2 bit 6 (see above) is inserted between the last data bit of a character and the first or only stop bit. The parity bit is checked during reception. A false parity bit generates an error indication in the Status Register and an Interrupt Request on Level 4.

Command Register 3

SET	RxE	IAE	NIE	END	SBRK	TBRK	RST
(2R)				(2W)			

Command Register 3 is different from the first two registers because it has a bit set/reset capability. Writing a byte with Bit 7 high sets any bits which were also high. Writing a byte with Bit 7 low resets any bits which were high. If any bit 0–6 is low, no change occurs to that bit. When Command Register 3 is read, bits 0, 3, and 7 will always be zero.

RST—Reset

If RST is set, the following events occur:

- 1) All bits in the Status Register except bits 4 and 5 are cleared, and bits 4 and 5 are set.
- 2) The Interrupt Enable, Interrupt Request, and Interrupt Service Registers are cleared. Pending requests and indications for interrupts in service will be canceled. Interrupt signal INT will go low.
- 3) The receiver and transmitter are reset. The transmitter goes idle (TxD is high), and the receiver enters start bit search mode.
- 4) If Port 2 is programmed for handshake mode, IBF and OBF are reset high.

RST does *not* alter ports, data registers or command registers, but it halts any operation in progress. RST is automatically cleared.

RST = 0 has no effect. The reset operation triggered by Command Register 3 is a subset of the hardware reset.

TBRK—Transmit Break

The transmission data output TxD will be set low as soon as the transmission of the previous character has

been finished. It stays low until TBRK is cleared. The state of CTS is of no significance for this operation. As long as break is active, data transfer from the Transmitter Buffer to the Transmitter Register will be inhibited. As soon as TBRK is reset, the break condition will be deactivated and the transmitter will be re-enabled.

SBRK—Single Character Break

This causes the transmitter data to be set low for one character including start bit, data bits, parity bit, and stop bits. SBRK is automatically cleared when time for the last data bit has passed. It will start after the character in progress completes, and will delay the next data transfer from the Transmitter Buffer to the Transmitter Register until TxD returns to an idle (marking) state. If both TBRK and SBRK are set, break will be set as long as TBRK is set, but SBRK will be cleared after one character time of break. If SBRK is set again, it remains set for another character. The user can send a definite number of break characters in this manner by clearing TBRK after setting SBRK for the last character time.

END—End of Interrupt

If fully nested interrupt mode is selected, this bit resets the currently served interrupt level in the Interrupt Service Register. *This command must occur at the end of each interrupt service routine during fully nested interrupt mode.* END is automatically cleared when the Interrupt Service Register (internal) is cleared. END is ignored if nested interrupts are not enabled.

NIE—Nested Interrupt Enable

When NIE equals 1, the interrupt controller will operate in the nested interrupt mode. When NIE equals 0, the interrupt controller will operate in the normal interrupt mode. Refer to the "Interrupt controller" section under "Normal Mode" and "Nested Mode" for a detailed description of these operations.

IAE—Interrupt Acknowledge Enable

This bit enables an automatic response to $\overline{\text{INTA}}$. The particular response is determined by the 8086 bit in Command Register 1.

RxE—Receive Enable

This bit enables the serial receiver and its associated status bits in the status register. If this bit is reset, the serial receiver will be disabled and the receive status bits will not be updated.

Note that the detection of break characters remains enabled while the receiver is disabled; i.e., Status Register Bit 3 (BD) will be set while the receiver is disabled whenever a break character has been recognized at the receive data input RxD.

SET—Bit Set/Reset

If this bit is high during a write to Command Register 3, then any bit marked by a high will set. If this bit is low, then any bit marked by a high will be cleared.

Mode Register

T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0
(3R)				(3W)			

P2C2, P2C1, P2C0—Port 2 Control

P2C2	P2C1	P2C0	Mode	Direction	
				Upper	Lower
0	0	0	nibble	input	input
0	0	1	nibble	input	output
0	1	0	nibble	output	input
0	1	1	nibble	output	output
1	0	0	byte handshake	input	
1	0	1	byte handshake	output	
1	1	0	<i>DO NOT USE</i>		
1	1	1	test		

If test mode is selected, the output from the internal baud rate generator is placed on bit 4 of Port 1 (pin 35).

To achieve this, it is necessary to program bit 4 of Port 1 as an output (Port 1 Control Register Bit P14 = 1), and to program Command Register 2 bits B3–B0 with a value $\geq 3H$.

NOTE:

If Port 2 is operating in handshake mode, Interrupt Level 7 is not available for Timer 5. Instead it is assigned to Port 2 handshaking.

CT2, CT3—Counter/Timer Mode

Bit 3 and 4 defines the mode of operation of event counter/timers 2 and 3 regardless of its use as a single unit or as a cascaded one.

If CT2 or CT3 are high, then counter/timer 2 or 3 respectively is configured as an event counter on bit 2 or 3 respectively of Port 1 (pins 37 or 36). The event counter decrements the count by one on each low-to-high transition of the external input. If CT2 or CT3 is low, then the respective counter/timer is configured as a timer and the Port 1 pins are used for parallel I/O.

T5C—Timer 5 Control

If T5C is set, then Timer 5 can be preset and started by an external signal. Writing to the Timer 5 register loads the Timer 5 save register and stops the timer. A high-to-low transition on bit 5 of Port 1 (pin 34) loads the timer with the saved value and starts the timer. The next high-to-low transition on pin 34 retriggers the timer by reloading it with the initial value and continues timing.

Following a hardware reset, the save register is reset to 00H and both clock and trigger inputs are disabled. Transferring an instruction with T5C = 1 enables the trigger input; the save register can now be loaded with an initial value. The first trigger pulse causes the initial value to be loaded from the save register and enables the counter to count down to zero.

When the timer reaches zero it issues an interrupt request, disables its interrupt level and continues counting. A subsequent high-to-low transition on pin 5 resets Timer 5 to its initial value. For another timer interrupt, the Timer 5 interrupt enable bit must be set again.

T35, T24—Cascade Timers

These two bits cascade Timers 3 and 5 or 2 and 4. Timers 2 and 3 are the lower bytes, while Timers 4 and 5 are the upper bytes. If T5C is set, then both Timers 3 and 5 can be preset and started by an external pulse. When a high-to-low transition occurs, Timer 5 is preset to its saved value, but Timer 3 is always preset to all ones. If either CT2 and CT3 is set, then the corresponding timer pair is a 16-bit event counter.

A summary of the counter/timer control bits is given in Table 5.

NOTE:

Interrupt levels assigned to single counters are partly not occupied if event counters/timers are cascaded. Level 2 will be vacated if event counters/timers 2 and 4 are cascaded. Likewise, Level 7 will be vacated if event counters/timers 3 and 5 are cascaded.

Single event counters/timers generate an interrupt request on the transition from 01H to 00H, while cascaded ones generate it on the transition from 0001H to 0000H.

Table 5. Event Counters/Timers Mode of Operation

Event Counter/ Timer	Function	Programming (Mode Word)	Clock Source
1	8-Bit Timer	—	Internal Clock
2	8-Bit Timer	T24 = 0, CT2 = 0	Internal Clock
	8-Bit Event Counter	T24 = 0, CT2 = 1	P12 Pin 37
3	8-Bit Timer	T35 = 0, CT3 = 0	Internal Clock
	8-Bit Event Counter	T35 = 0, CT3 = 1	P13 Pin 36
4	8-Bit Timer	T24 = 0	Internal Clock
5	8-Bit Timer, Normal Mode	T35 = 0, T5C = 0	Internal Clock
	8-Bit Timer, Retriggerable Mode	T35 = 0, T5C = 1	Internal Clock
2 and 4 Cascaded	16-Bit Timer	T24 = 1, CT2 = 0	Internal Clock
	16-Bit Event Counter	T24 = 1, CT2 = 1	P12 Pin 37
3 and 5 Cascaded	16-Bit Timer, Normal Mode	T35 = 1, T5C = 0, CT3 = 1	Internal Clock
	16-Bit Event Counter, Normal Mode	T35 = 1, T5C = 0, CT3 = 1	P13 Pin 36
	16-Bit Timer, Retriggerable Mode	T35 = 1, T5C = 1 CT3 = 0	Internal Clock
	16-Bit Event Counter, Retriggerable Mode	T35 = 1, T5C = 1, CT3 = 1	P13 Pin 36

Port 1 Control Register

P17	P16	P15	P14	P13	P12	P11	P10
(4R)				(4W)			

Each bit in the Port 1 Control Register configures the direction of the corresponding pin. If the bit is high, the pin is an output, and if it is low the pin is an input. Every Port 1 pin has another function which is controlled by other registers. If that special function is disabled, the pin functions as a general I/O pin as specified by this register. The special functions for each pin are described below.

Port 10, 11—Handshake Control

If byte handshake control is enabled for Port 2 by the Mode Register, then Port 10 is programmed as \overline{STB}/ACK handshake control input, and Port 11 is programmed as $\overline{IBF}/\overline{OBF}$ handshake control output.

If byte handshake mode is enabled for output on Port 2, \overline{OBF} indicates that a character has been loaded into the Port 2 output buffer. When an external device reads the data, it acknowledges this operation by driving \overline{ACK} low. \overline{OBF} is set low by writing to Port 2 and is reset high by \overline{ACK} .

If byte handshake mode is enabled for input on Port 2, \overline{STB} is an input. \overline{IBF} is driven low after \overline{STB} goes low. On the rising edge of \overline{STB} the data from Port 2 is latched.

\overline{IBF} is reset high when Port 2 is read.

Port 12, 13—Counter 2, 3 Input

If Timer 2 or Timer 3 is programmed as an event counter by the Mode Register, then Port 12 or Port 13 is the counter input for Event Counter 2 or 3, respectively.

Port 14—Baud Rate Generator Output Clock

If test mode is enabled by the Mode Register and Command Register 2 baud rate select is greater than 2, then Port 14 is an output from the internal baud rate generator.

P14 in Port 1 control register must be set to 1 for the baud rate generator clock to be output. The baud rate generator clock is 64 x the serial bit rate except at 19.2 Kbps when it is 32 x the bit rate.

Port 15—Timer 5 Trigger

If T5C is set in the Mode Register enabling a retriggerable timer, then Port 15 is the input which starts and reloads Timer 5.

A high-to-low transition on P15 (Pin 34) loads the timer with the save register and starts the timer.

Port 16—Break-In Detect

If Break-In Detect is enabled by BRK1 in Command Register 1, then this input is used to sense a Break-In. If Port 16 is low while the serial transmitter is sending the last stop bit, then a Break-In condition is signaled.

Port 17—Port Interrupt Source

If BITI in Command Register 1 is set, then a low-to-high transition on Port 17 generates an interrupt request on Priority Level 1.

Port 17 is edge triggered.

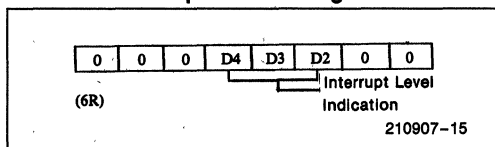
Interrupt Enable Register

L7	L6	L5	L4	L3	L2	L1	L0
(5R)				(5W = enable, 6W = disable)			

Interrupts are enabled by writing to the Set Interrupts Register (5W). Interrupts are disabled by writing to the Reset Interrupts Register (6W). Each bit set by the Set Interrupts Register (5W) will enable that level interrupt, and each bit set in the Reset Interrupts Register (6W) will disable that level interrupt. The user can determine which interrupts are enabled by reading the Interrupt Enable Register (5R).

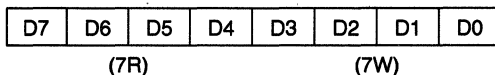
Priority	Source
Highest	L0 Timer 1
	L1 Timer 2 or Port Interrupt
	L2 External Interrupt (EXTINT)
	L3 Timer 3 or Timers 3 & 5
	L4 Receiver Interrupt
	L5 Transmitter Interrupt
	L6 Timer 4 or Timers 2 & 4
Lowest	L7 Timer 5 or Port 2 Handshaking

Interrupt Address Register



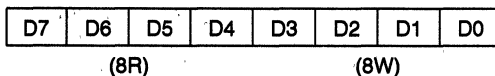
Reading the interrupt address register transfers an identifier for the currently requested interrupt level on the system data bus. This identifier is the number of the interrupt level multiplied by 4. It can be used by the CPU as an offset address for interrupt handling. Reading the interrupt address register has the same effect as a hardware interrupt acknowledge INTA; it clears the interrupt request pin (INT) and indicates an interrupt acknowledgement to the interrupt controller.

Receiver and Transmitter Buffer



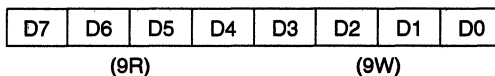
Both the receiver and transmitter in the MUART are double buffered. This means that the transmitter and receiver have a shift register and a buffer register. The buffer registers are directly addressable by reading or writing to register seven. After the receiver buffer is full, the RBF bit in the status register is set. Reading the receive buffer clears the RBF status bit. The transmit buffer should be written to only if the TBE bit in the status register is set. Bytes written to the transmit buffer are held there until the transmit shift register is empty, assuming CTS is low. If the transmit buffer and shift register are empty, writing to the transmit buffer immediately transfers the byte to the transmit shift register. If a serial character length is less than 8 bits, the unused most significant bits are set to zero when reading the receive buffer, and are ignored when writing to the transmit buffer.

Port 1



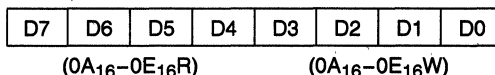
Writing to Port 1 sets the data in the Port 1 output latch. Writing to an input pin does not affect the pin, but the data is stored and will be output if the direction of the pin is changed later. If the pin is used as a control signal, the pin will not be affected, but the data is stored. Reading Port 1 transfers the data in Port 1 onto the data bus.

Port 2



Writing to Port 2 sets the data in the Port 2 output latch. Writing to an input pin does not affect the pin, but it does store the data in the latch. Reading Port 2 puts the input pins onto the bus or the contents of the output latch for output pins.

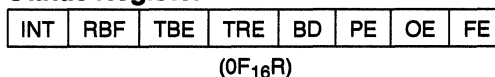
Timer 1-5



Reading Timer N puts the contents of the timer onto the data bus. If the counter changes while RD is low, the value on the data bus will not change. If two timers are cascaded, reading the high-order byte will cause the low-order byte to be latched. Reading the low-order byte will unlatch them both. Writing to either timer or decascading them also clears the latch condition. Writing to a timer sets the starting value of that timer. If two timers are cascaded, writing to the high-order byte presets the low-order byte to all ones. Loading only the high-order byte with a value of X leads to a count of X 256 + 255. Timers count down continuously. If the interrupt is enabled, it occurs when the counter changes from 1 to 0.

The timer/counter interrupts are automatically disabled when the interrupt request is generated.

Status Register



Reading the status register gates its contents onto the data bus. It holds the operational status of the serial interface as well as the status of the interrupt pin INT. The status register can be read at any time. The flags are stable and well defined at all instants.

FE—Framing Error, Transmission Mode

Bit 0 can be used in two modes. Normally, FE indicates framing error which can be changed to transmission mode indication by setting the TME bit in the modification register.

If transmission mode is disabled (in Modification Register), then FE indicates a framing error. A framing error is detected during the first stop bit. The error is reset by reading the Status Register or by a chip reset. A fram-

ing error does not inhibit the loading of the Receiver Buffer. If RxD remains low, the receiver will assemble the next character. The false stop bit is treated as the next start bit, and no high-to-low transition on RxD is required to synchronize the receiver.

When the TME bit in the Modification Register is set, FE is used to indicate that the transmitter was active during the reception of a character, thus indicating that the character received was transmitted by its own transmitter. FE is reset when the transmitter is not active during the reception of character. Reading the status register will not reset the FE bit in the transmission mode.

OE—Overrun Error

If the user does not read the character in the Receiver Buffer before the next character is received and transferred to this register, then the OE bit is set. The OE flag is set during the reception of the first stop bit and is cleared when the Status Register is read or when a hardware or software reset occurs. The first character received in this case will be lost.

PE—Parity Error

This bit indicates that a parity error has occurred during the reception of a character. A parity error is present if value of the parity bit in the received character is different from the one expected according to command word 2 bits 6 EP. The parity bit is expected and checked only if it is enabled by command word 2 bit 7 PEN.

A parity error is set during the first stop bit and is reset by reading the Status Register or by a chip reset.

BD—Break/Break-In

The BD bit flags whether a break character has been received, or a Break-In condition exists on the transmission line. Command Register 1 Bit 3 (BRKI) enables the Break-In Detect function.

Whenever a break character has been received, Status Register Bit 3 will be set and in addition an interrupt request on Level 4 is generated. The receiver will be idled. It will be started again with the next high-to-low transition at pin RxD.

The break character received will not be loaded into the receiver buffer register.

If Break-In Detection is enabled and a Break-In condition occurs, Status Register Bit 3 will be set and in addition an interrupt request on Level 5 is generated.

The BD status will be reset upon reading the status register or on a hardware or software reset. For more information on Break/Break-In, refer to the “Serial Asynchronous Communication” section under “Receive Break Detect and “Break-In Detect”.

TRE—Transmit Register Empty

When TRE is set the transmit register is empty and an interrupt request is generated on Level 5 if enabled. When TRE equals 0 the transmit register is in the process of sending data. TRE is set by a chip reset and when the last stop bit has left the transmitter. It is reset when a character is loaded into the Transmitter Register. If CTS is low, the Transmitter Register will be loaded during the transmission of the start bit. If CTS is high at the end of a character, TRE will remain high and no character will be loaded into the Transmitter Register until CTS goes low. If the transmitter was inactive before a character is loaded into the Transmitter Buffer, the Transmitter Register will be empty temporarily while the buffer is full. However, the data in the buffer will be transferred to the transmitter register immediately and TRE will be cleared while TBE is set.

TBE—Transmitter Buffer Empty

TBE indicates the Transmitter Buffer is empty and is ready to accept a character. TBE is set by a chip reset or the transfer of data to the Transmitter Register, and is cleared when a character is written to the transmitter buffer. When TBE is set, an interrupt request is generated on Level 5 if enabled.

RBF—Receiver Buffer Full

RBF is set when the Receiver Buffer has been loaded with a new character during the sampling of the first stop bit. RBF is cleared by reading the receiver buffer or by a chip reset.

INT—Interrupt Pending

The INT bit reflects the state of the INT Pin (Pin 15) and indicates an interrupt is pending. It is reset by INTA or by reading the Interrupt Address Register if only one interrupt is pending and by a chip reset.

FE, OE, PE, RBF, and Break Detect all generate a Level 4 interrupt when the receiver samples the first stop bit. TRE, TBE, and Break-In Detect generate a Level 5 interrupt. TRE generates an interrupt when TBE is set and the Transmitter Register finished transmitting. The Break-In Detect interrupt is issued at the same time as TBE or TRE.

Modification Register

0	RS4	RS3	RS2	RS1	RS0	TME	DSC
---	-----	-----	-----	-----	-----	-----	-----

(OF₁₆W)

DSC—Disable Start Bit Check

DSC disables the receiver's start bit check. In this state the receiver will not be reset if RxD is not low at the center of the start bit.

TME—Transmission Mode Enable

TME enables transmission mode and disables framing error detection. For information on transmission mode see the description of the framing error bit in the Status Register.

RS0, RS1, RS2, RS3, RS4—Receiver Sample Time

The number in RS_n alters when the receiver samples RxD. The receiver sample time can be modified only if the receiver is *not* clocked by RxC.

NOTE:

The modification register cannot be read. Reading from address OFH, 8086: 1EH gates the contents of the status register onto the data bus.

— A hardware reset (reset, Pin 12) resets all modification register bits to 0, i.e.:

- The start bit check is enabled.
- Status Register Bit 0 (FE) indicates framing error.
- The sampling time of the serial receiver is the bit center.

A software reset (Command Word 3, RST) does not affect the modification register.

Hardware Reset

A reset signal on pin RESET (HIGH level) forces the device 8256 into a well-defined initial state. This state is characterized as follows:

RS4	RS3	RS2	RS1	RS0	Point of Time between Start of Bit and End of Bit Measured in Steps of 1/32 Bit Length
0	1	1	1	1	1 (Start of Bit)
0	1	1	1	0	2
0	1	1	0	1	3
0	1	1	0	0	4
0	1	0	1	1	5
0	1	0	1	0	6
0	1	0	0	1	7
0	1	0	0	0	8
0	0	1	1	1	9
0	0	1	1	0	10
0	0	1	0	1	11
0	0	1	0	0	12
0	0	0	1	1	13
0	0	0	1	0	14
0	0	0	0	1	15
0	0	0	0	0	16 (Bit Center)
1	1	1	1	1	17
1	1	1	1	0	18
1	1	1	0	1	19
1	1	1	0	0	20
1	1	0	1	1	21
1	1	0	1	0	22
1	1	0	0	1	23
1	1	0	0	0	24
1	0	1	1	1	25
1	0	1	1	0	26
1	0	1	0	1	27
1	0	1	0	0	28
1	0	0	1	1	29
1	0	0	1	0	30
1	0	0	0	1	31
1	0	0	0	0	32 (End of Bit)

1) Command registers 1, 2 and 3, mode register, Port 1 control register, and modification register are reset. Thus, all bits of the parallel interface are set to be inputs and event counters/timers are configured as independent 8-bit timers.

- 2) Status register bits are reset with the exception of bits 4 and 5. Bits 4 and 5 are set indicating that both transmitter register and transmitter buffer register are empty.
- 3) The interrupt mask, interrupt request, and interrupt service register bits are reset and disable all requests. As a consequence, interrupt signal INT is inactive (LOW).
- 4) The transmit data output is set to the marking state (HIGH) and the receiver section is disabled until it is enabled by Command Register 3 Big 6.
- 5) The start bit will be checked at sampling time. The receiver will return to start bit search mode if input RxD is not LOW at this time.
- 6) Status Register Bit 0 implies framing error.
- 7) The receiver samples input RxD at bit center.

Reset has no effect on the contents of receiver buffer register, transmitter buffer register, the intermediate latches of parallel ports, and event counters/timers, respectively.

INTERFACING

This section describes the hardware interface between the 8256 MUART and the 8085, 8086, 8088, and 80186 microprocessors. Figures 14 through 19 display the block diagrams for these interfaces. The MUART can be interfaced to many other microprocessors using these basic principles.

In all cases the 8256 will be connected directly to the CPU's multiplexed address/data bus. If latches or data bus buffers are used in a system, the MUART should be on the microprocessor side of the address/data bus. The MUART latches the address internally on the falling edge of ALE. The address consists of Chip Select (CS) and four address lines. For 8-bit microprocessors, AD0-AD3 are the address lines. For 16-bit microprocessors, AD1-AD4 are the address lines; AD0 is used as a second chip select which is active low. Since chip select is internally latched along with the address, it does not have to remain active during the entire instruction cycle. As long as the chip select setup and hold times are met, it can be derived from multiplexed address/data lines or multiplexed address/status lines.

In Figure 15, the 8088 min mode, the 8205 chip select decoder is connected to the 8088's address bus lines A8-A15. These address lines are stable throughout the entire instruction cycle. However, the MUART's chip select signal could have been derived from A16/S3-A19/S6.

Figure 16 shows the 8256 interfaced with an 8086 in the min mode. When the 8256 is in the 16-bit mode, A0 serves as a second chip select. As a result the

MUART's internal registers will all have even addresses since A0 must be zero to select the device. Normally the MUART will be placed on the lower data byte. If the MUART is placed on the upper data byte the internal registers will be 512 address locations apart and the chip would occupy an 8K word address space. Figure 16A shows a table and a diagram of how the 8256 may be selected in an 8086 system where the MUART is I/O mapped and used on the lower byte of the address/data bus.

PROGRAMMING

Initialization

In general the MUART's functions are independent of each other and only the registers and bits associated with a particular function need to be initialized, not the entire chip. The command sequence is arbitrary since every register is directly addressable; however, Command Word 1 must be loaded first. To put the device into a fully operational condition, it is necessary to write the following commands:

```

Command byte 1
Command byte 2
Command byte 3
  Mode byte
    Port 1 control
    Set Interrupts

```

The modification register may be loaded if required for special applications; normally this operation is not necessary. It is a good idea to reset the part before initialization. (Either a hardware or a software reset will do.)

Operating the Serial Interface

The microprocessor transfers data to the serial interface by writing bytes to the Transmit Buffer Register. Receive characters are transferred by reading the Receiver Buffer Register. The Status Register provides all of the necessary information to operate the serial I/O, including when to write to the Transmit Buffer, and when to read the Receive Buffer and error information.

Transmitting

The transmitter and the receiver may be operated by using either polling or interrupts. If polling is used then the software may poll the Status Register and write a byte to the Transmit Buffer whenever TBE = 1. Writing a byte to the Transmit Buffer clears the TBE status bit. If the CTS pin is low, then the Transmit Buffer will transfer the data to the Transmit Register when it becomes empty. When this transfer takes place the TRE bit is reset, and the TBE bit is set indicating the next

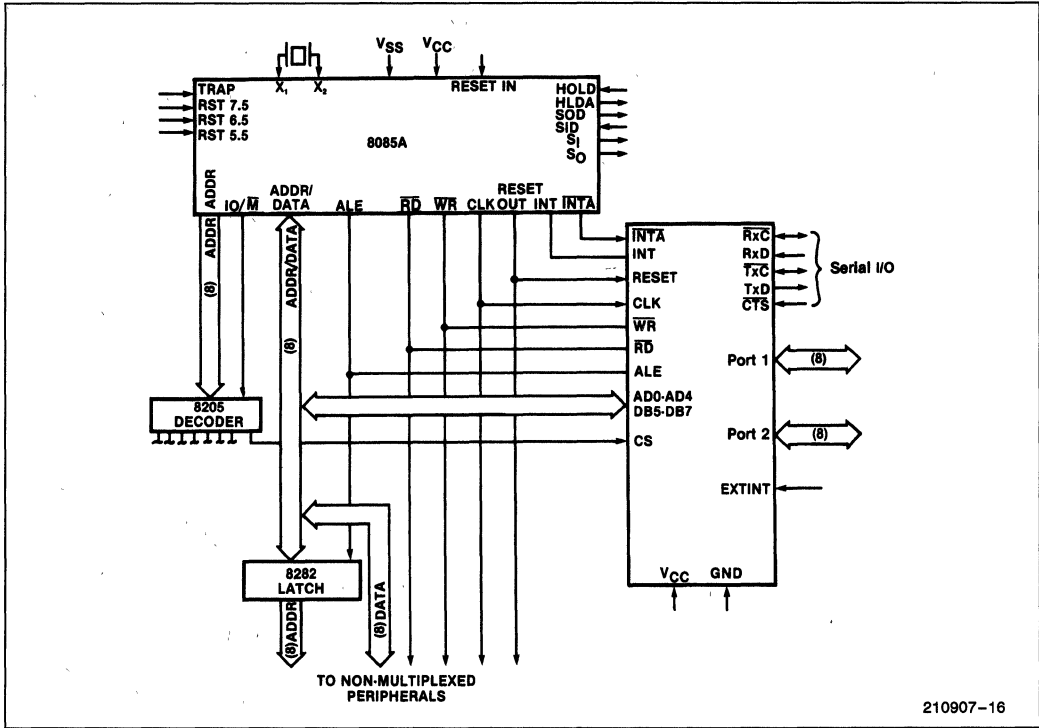


Figure 14. 8085/8256 Interface

byte may be written to the Transmit Buffer. If \overline{CTS} is high, disabling the transmitter, the data byte will remain in the Transmit Buffer and TBE will remain low until \overline{CTS} goes low. The transmitter can only buffer one byte if it is disabled.

There is no way of knowing that the transmitter is disabled unless the \overline{CTS} signal is fed into one of the I/O ports. Using the transmitter interrupt will free up the CPU to perform other functions while the transmitter is disabled or while the Transmit Buffer is full.

To enable the transmit interrupt feature Bit L5 in the Set Interrupt Register must be set. An interrupt request will not occur immediately after this bit has been set. Before any transmit interrupt request will occur a byte must be written to the Transmit Buffer. After the first byte has been written to the Transmit Buffer, a transmit interrupt request will occur, providing the transmitter is enabled.

There are three sources of transmitter interrupt requests: $TBE = 1$, $TRE = 1$, and Break-In Detect. Assuming the Break-In Detect feature is disabled, after

the transmit interrupt is enabled and the first byte is written, a transmit interrupt request will be generated by TBE going active. The microprocessor can immediately write a byte to the Transmit Buffer without reading any status. However if Break-In Detect is enabled, the Status Register must be read to determine whether the transmit interrupt request was generated by Break-In Detect or TBE.

The TRE interrupt request can be used to indicate when the transmitter has completely sent all of the data. For example, using half-duplex communications, all of the data written to the MUART must be transmitted before the line can be turned around. After the last byte is written, an interrupt request will be generated by TBE. If this interrupt is acknowledged without writing another byte, then the next transmitter interrupt request, $TRE = 1$, will indicate that the transmitter is empty and the line may be turned around.

RECEIVING

Valid data may be read from the Receive Buffer whenever the RBF bit in the Status Register is set. Reading

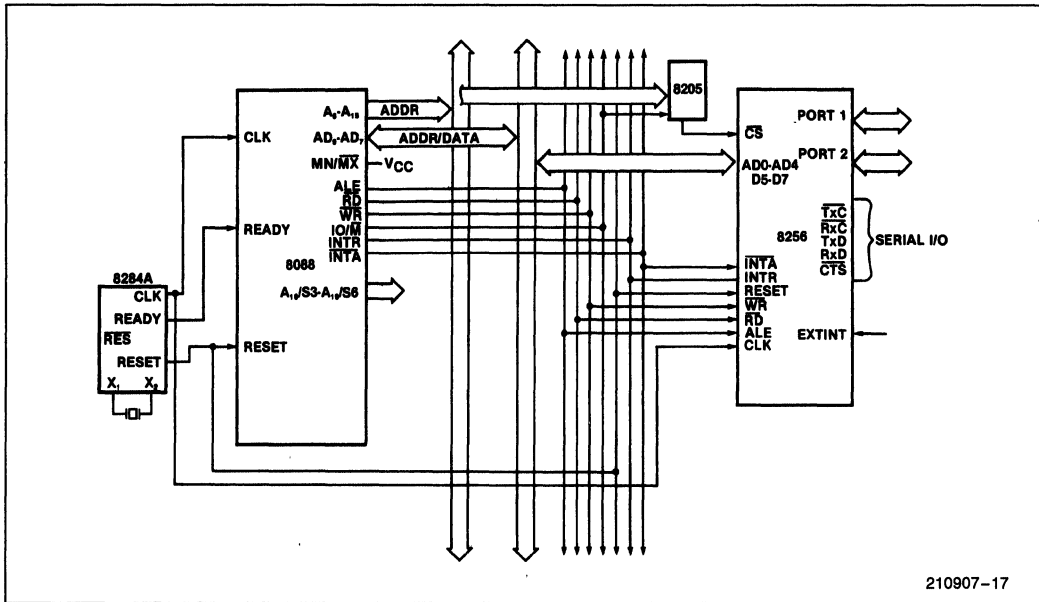


Figure 15. 8088 Min Mode/8256 Interface Multiplexed Bus

the Receive Buffer resets the RBF status bit. The RBF bit in the Status Register can be used for polling. When the RBF bit is set, the three receive status bits, PE, OE, and FE are updated. These three status bits are reset when they are read. Therefore when the status register is read with RBF set, the three error status bit should be tested too.

If interrupts are used for serial receive data, the receiver must be enabled by setting the RxE bit in Command Register 3, and Bit L4 must be set in the Set Interrupt Register. When the receive interrupt request occurs the Receive Buffer may be read, but the status register should also be read since the receive interrupt could have been generated by the Break Detect. Also, reading the status register will indicate whether there were any errors in the received character.

Operating the Parallel Interface

Data can be transferred to or read from Port 1 and Port 2 by using the appropriate write and read operations.

LOADING PORT 1 AND PORT 2

Writing to the ports transfers the data present on the data bus into the output latches. This operation is independent of the programmed I/O characteristics of the individual port pins. Writing to control or input ports has no effect on the state of the pins. Pins defined as outputs immediately assume the state which is associated with the transferred data. If inputs or control pins are reprogrammed into outputs, they assume the states stored in their output latches which were transferred by the most recent port write operation.

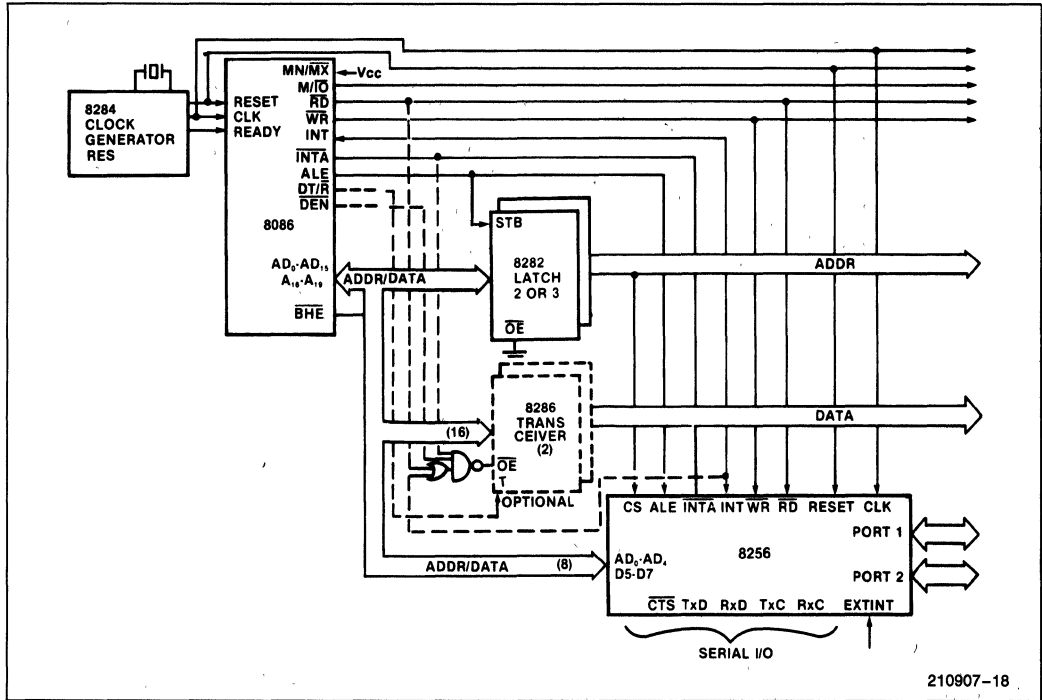


Figure 16. 8086 Min. Mode/8256 Interface

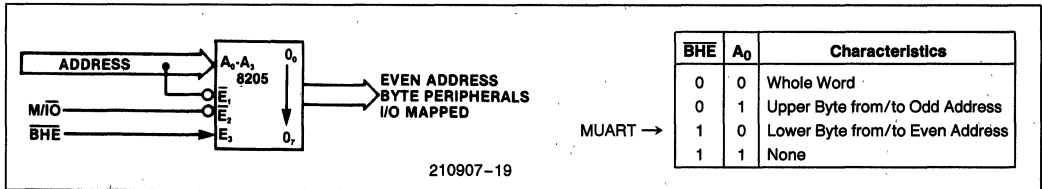


Figure 16a. Technique for Generating the MUART's Chip Select

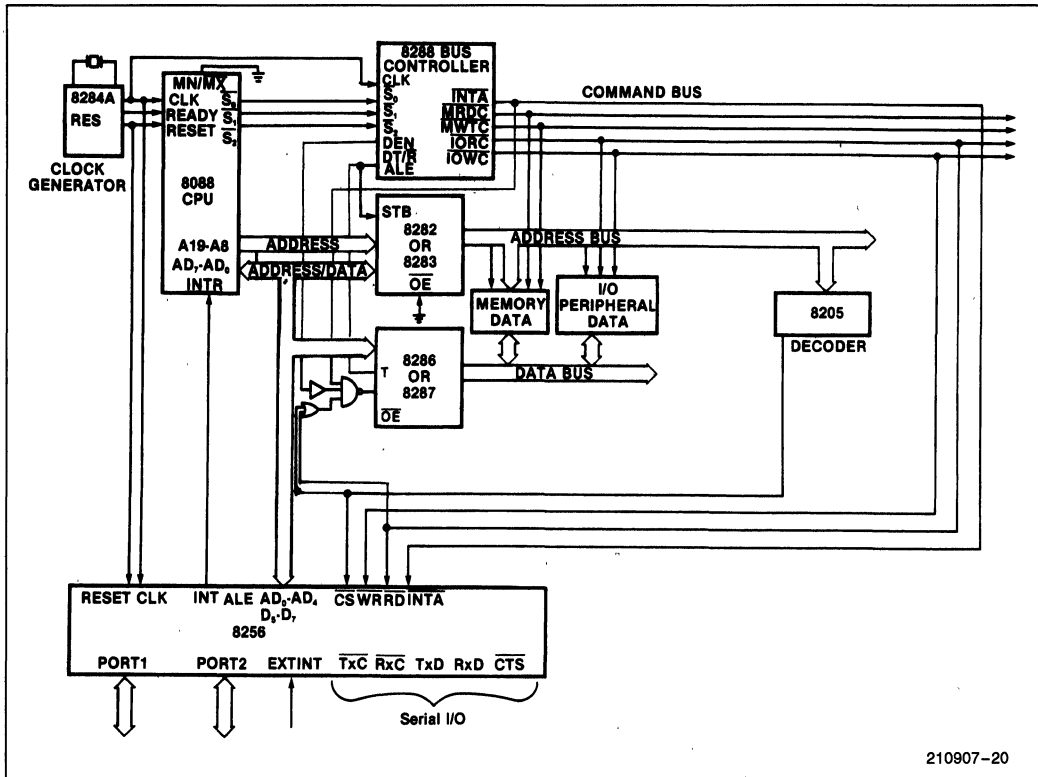


Figure 17. 8088 Max Mode/8256 Interface

READING PORT 1 AND PORT 2

Reading the ports gates the state at the pins onto the data bus if they are defined as I/O pins. A read operation transfers the contents of the associated output latches of pins P12, P13, P15, and P16, which are defined as control function pins. Reading control pins P10, P11, and P17 delivers the state of these pins.

Operating the Event Counters/Timers

The event counters/timers can be loaded with an initial value at any time. Reading event counters/timers is possible without interfering with the counting process.

LOADING EVENT COUNTERS/TIMERS

Loading event counters/timers 1-5 under their respective addresses transfers the data present on the data bus as an initial value into the addressed event counter/timer. The event counter/timer counts from the new initial value immediately following the data transfer (exception: retriggerable mode of Timer 5, or 3 and 5).

Cascaded counters/timers can be loaded with an initial value using one of two procedures:

- 1) Only the event counter/timer representing the most significant byte will be loaded. The event counter/timer representing the least significant byte is set to 0FFH automatically. Counting is started immediately after the data transfer.
- 2) The event counter/timer representing the most significant byte will be loaded, causing the least significant byte to be set to 0FFH automatically. Counting is started immediately following the data transfer. Next, the counter representing the least significant byte will be loaded and counting is started again, but this time with a complete 16-bit initial value. The least significant byte of the initial value must be transferred before the counter representing the least significant byte exhibits its zero transition to prevent the most significant byte of the initial value from being decremented improperly.

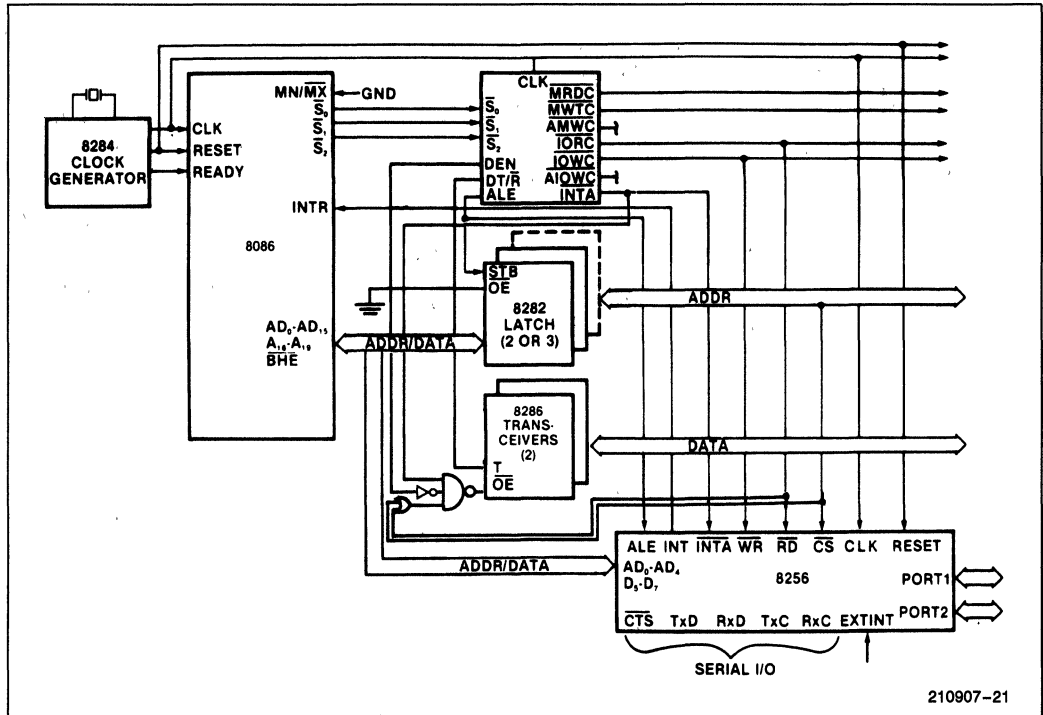


Figure 18. 8086 Max Mode/8256 Interface

In the case of an 8-bit initial value for Timer 5 or for cascaded Event Counter/Timer 3 and 5, the initial value for Timer 5 is loaded from a save register, if it is operated in retriggerable counting mode. Counting is started after an initial value has been transferred whenever a high-to-low transition occurs on Port P15.

Cascaded Event Counter/Timer 3 and 5 operating in retriggerable counting mode can be loaded directly with an initial value for Timer 5 representing the most significant byte; Event Counter/Timer 3 will be set to 0FFH automatically.

READING EVENT COUNTERS/TIMERS

Reading event counters/timers 1-5 from their respective addresses gates the counter contents onto the data bus. The counter contents gated onto the data bus remain stable during the read operation while the counter just being read continues to count. The minimum time between the two read operations from the same counter is 1 μ s.

The procedure to be followed when reading cascaded event counters/timers is:

- 1) The event counter/timer representing the most significant byte will be read first. At this time, the least significant byte is latched onto read latches.
- 2) When the event counter/timer representing the least significant byte is addressed, the byte stored in the read latches will be gated onto the data bus. The value stored in the read latches remains valid until it is read, the cascading condition is removed, or a write operation affecting one of the two event counters/timers is executed.

The time between reading the most significant byte and the least significant byte must be a least 1 μ s.

NOTE:

For cascaded event counters/timers the least significant counter/timer is latched after reading the most significant counter/timer. If the lower byte changes from 00H to 0FFH between the reading of the MSB and the latching of the LSB, the carry from the most significant event counter/timer to the least significant event counter/timer is lost. Therefore, it is necessary to repeat the whole reading once if the value of the least significant event counter/timer is 0FFH. Doing this will avoid working with a wrong value (correct value + 255).

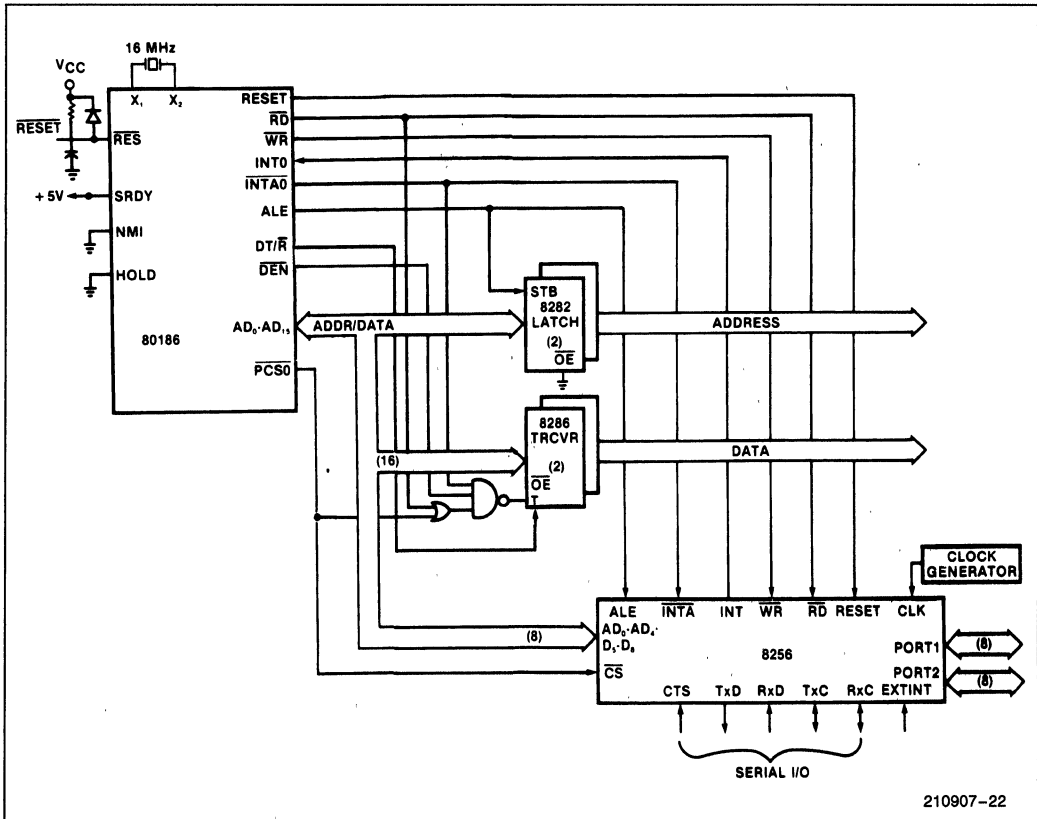


Figure 19. 80186/8256 Interface

APPLICATION EXAMPLE

This section describes how the 8256 was designed into a Line Printer Multiplexer (LPM). This application example was chosen because it employs a majority of the MUART's features. The information in this section will be applicable to many other designs since it describes some common software and hardware aspects of using the MUART.

Description of the Line Printer Multiplexer (LPM)

The Line Printer Multiplexer allows up to eight workstations to share one printer. The workstations transmit serial asynchronous data to the LPM. The LPM receives the serial data, buffers it, then transmits it to the line printer using a two-wire byte handshake Dataproducts interface. A conceptual diagram of this system is shown in Figure 20. Note that only one workstation can transmit at a time. This workstation will transmit its entire file before another workstation will be allowed to transmit.

The LPM sequentially polls each of the eight RS-232 ports for a Request To Send (RTS). When it finds a serial port which has asserted RTS, it configures itself for the appropriate data format and bit rate, establishes the connection and sends back to the serial port a Clear To Send (CTS) which enables transmission. The LPM receives the serial asynchronous data, buffers it in a software FIFO, and transmits the data to the line printer. If the LPM detects an error in any of the serial characters it receives, it transmits an error message to the serial port and ignores the bad character. If the LPM does not receive a serial character after 18 seconds, it assumes that the transmission is complete. It transmits the final status to the serial port, and returns to scanning.

This LPM was designed to be used with single-user workstations and a 300 lines per minute line printer. These workstations are not multitasking; therefore in the middle of a file transfer when the CPU needs to reload its buffer from the disk, no serial data is transmitted. During this time the LPM is emptying its FIFO; thus, the line printer never stops printing.

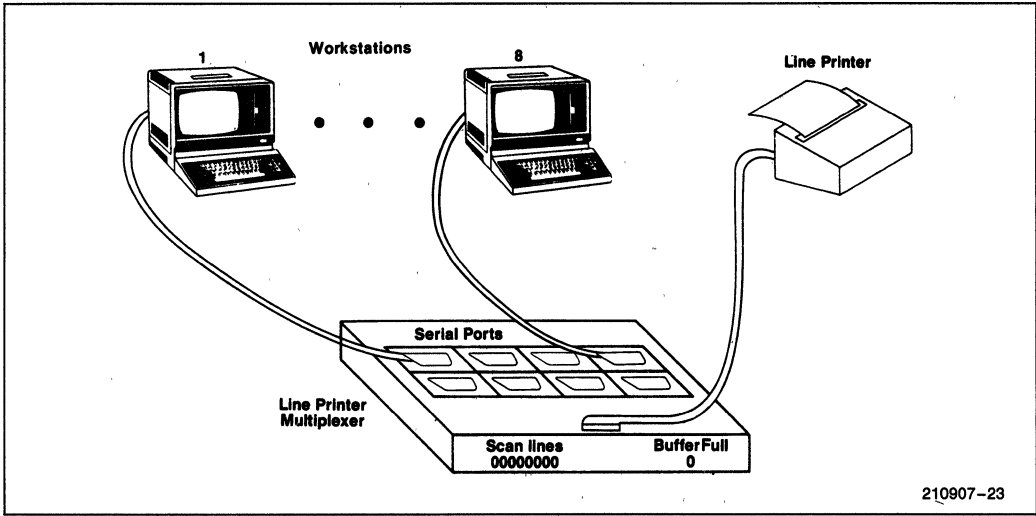


Figure 20. Using the Line Printer Multiplexer to Share a Line Printer

The buffer size on the LPM was chosen to complement the disk access time on the workstations. Figure 21 illustrates the buffer size calculation. The line printer can print up to 300 lines per minute, or approximately 660 characters per second. This corresponds to a serial transmission rate of 6,600 bps (assuming ASCII character codes and a parity bit) as shown in equation 1.

$$\text{Serial bit rate for the line printer} = \frac{(300 \text{ lines/min}) * (132 \text{ char/line}) * (10 \text{ bits/char})}{(60 \text{ sec/min})} \quad (1)$$

The bottleneck in this data transfer is the line printer since the MUART and the workstations can both

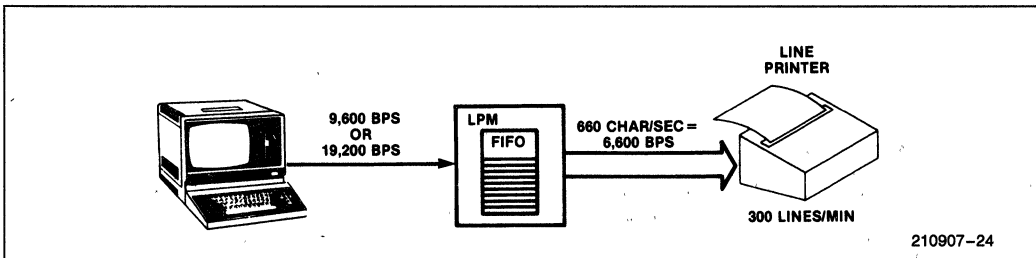


Figure 21. LPM Buffer Size Calculation

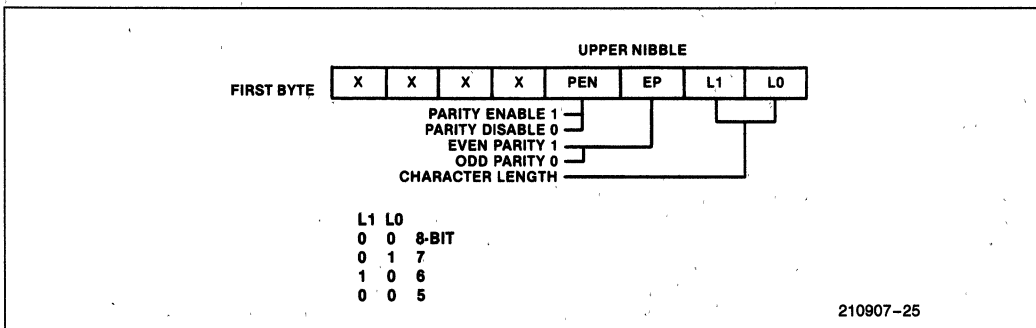


Figure 22. Programming Words Format for LPM

transmit and receive at 19.2 Kbps. To realize the maximum data transfer rate of this system the LPM must guarantee that the average transfer rate to the line printer is 660 characters per second. The maximum amount of dead time that the serial port on the workstation is not transmitting, multiplied by 660 is the number of bytes which the LPM should buffer. It was determined through experimentation that it takes about 3 seconds to load 40K bytes of data from the disk into the workstation's RAM. During these 3 seconds no serial data is being sent; therefore the buffer size on the LPM should be 2K bytes. (Note: even though only a 2K byte FIFO is required, this design used an 8K byte FIFO.)

To keep the LPM's buffer full the serial data rate must be greater than 6.6 Kbps. The two bit rates which the workstations use are 9.6 Kbps and 19.2 Kbps. The CTS signal is used to control the flow of the serial data so that the LPM buffer will not overflow.

Each serial port on the LPM can have a different bit rate, character length, and parity format. These param-

eters are programmable through the serial port. When the LPM powers up, or is reset, it expects a bit rate of 9600 bps, 7 bit characters, and odd parity. When a serial port receives an ASCII ESC character (1BH), it puts that port in the program mode. The next two bytes will program these three parameters. Only the lower nibbles of these two bytes are used, and the upper nibbles are discarded. The format of these programming words is given in Figure 22. If the word following the ESC is an ASCII NUL (0), the LPM will exit from the programming mode and not change any of its parameters.

Description of the Hardware

Figure 23 shows a block diagram of the LPM. In addition to the standard components of most microprocessor systems such as CPU, RAM, and ROM this particular design requires a UART, timers, parallel I/O and an interrupt controller. The MUART is the ideal choice for this design since it integrates these four functions onto one device.

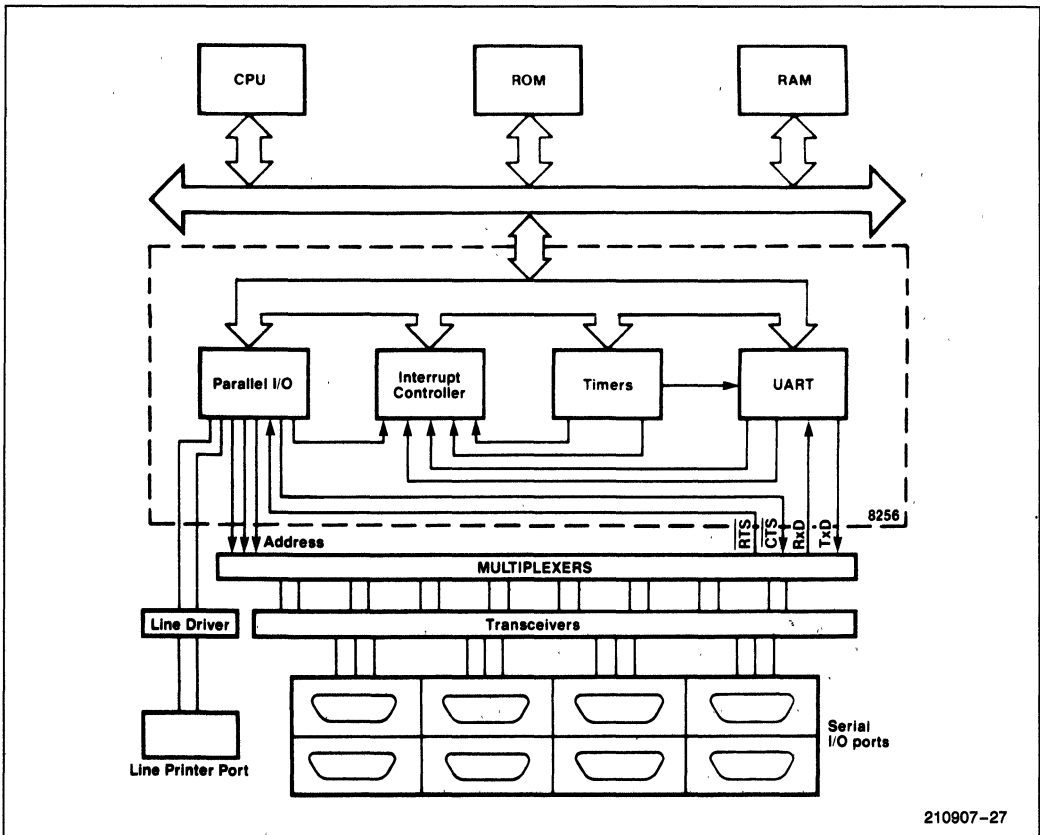


Figure 23. Functional Block Diagram of the Line Printer Multiplexer

The eight serial I/O ports use four signals: Transmit Data (TxD), Receive Data (RxD), Request To Send (RTS), and Clear To Send (CTS). These four signals, controlled by the MUART, are connected to one port at a time using TTL multiplexers. The TTL multiplexers are interfaced to RS-232 transceivers to be electrically compatible with the RS-232 spec. The serial port select address is derived from three bits of the MUART's parallel I/O port (Port 1). Two more bits from Port 1 control CTS and RTS, and another bit lights up an LED to indicate when the LPM's buffer is full. Parallel Port 2 and two bits from Port 1 are connected to the line printer implementing a two-wire byte handshake transfer. These signals are passed through a line driver so that they can reliably drive a long cable.

There are three timing functions needed for the LPM: a scan timer, a debounce timer, and a receive timeout. The Scan timer determines the amount of time spent sampling RTS on each port before the next port is addressed. By using one of the MUART's timers to do this function, the CPU is free to perform other func-

tions instead of implementing the timer in software. If RTS is recognized as true, the CPU branches into a debounce procedure. This procedure uses another one of the MUART's timers to wait 10 ms then sample RTS again, thus preventing any glitches from registering as a false RTS. The receive timeout timer uses two 8-bit timers in the cascaded mode to measure an 18-second interval. After a valid RTS is recognized, the LPM sends back a CTS and initializes the receive timeout timer for 18 seconds. Each time a character is received by the LPM, this timer is reinitialized. If this timer times out, the LPM considers the transmission complete and returns to scanning.

The schematic diagram of the LPM is shown in Figure 24. The CPU is an 8088 used in the min mode. It is interfaced directly to the 8256. An 8282 latch is employed in the system so that nonmultiplexed bus memory can be used. A 2716 holds the entire program, and six 2016s (2K x 8 static RAMs) are used to store the buffer, temporary data, stack area, and interrupt vector table. The 2716 is located in the upper 2K of the 8088

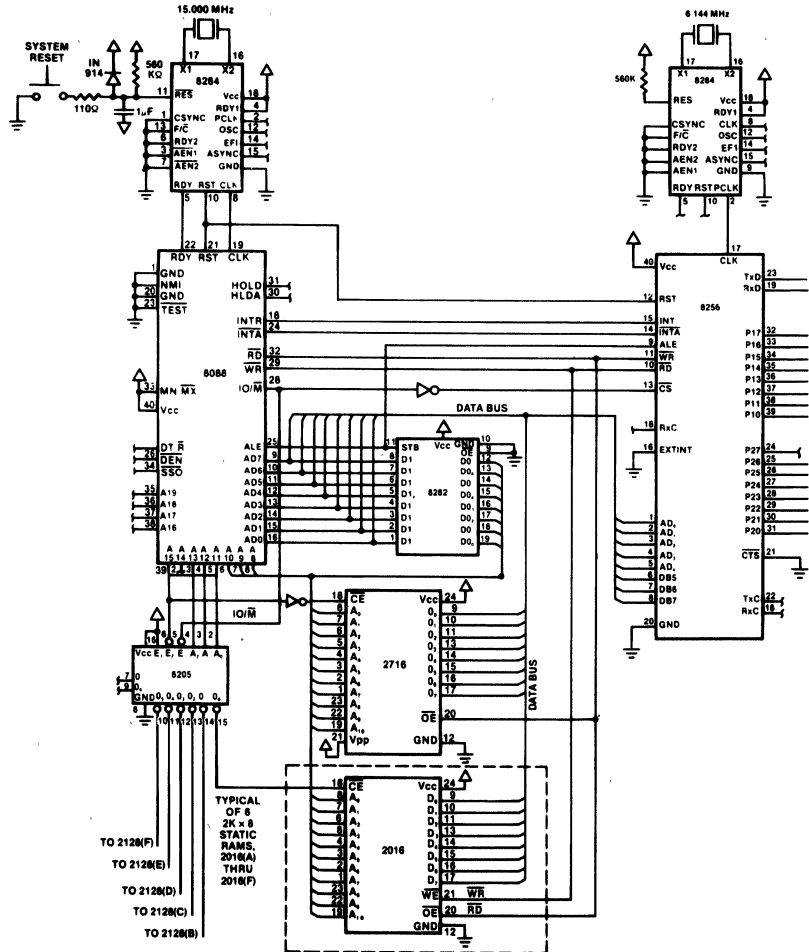
Table 6. Dataproducts Interface Line Functions

Signal	Description	Connector Pin
Data Request	Sent by printer to synchronize data transmission. When true, requests a character. Remains true until Data Strobe is received, then goes false within 100 ns.	E(return C)
Data Strobe	Sent by user system to cause printer to accept information on data lines. Should remain true until printer drops Data Request line. Data lines must stabilize for at least 50 ns before Data Strobe is sent.	j(return m)
Data Bit 1 Data Bit 2 Data Bit 3 Data Bit 4 Data Bit 5 Data Bit 6 Data Bit 7 Data Bit 8	Bit 8 controls optional character set Refer to <i>Commands and Formats</i> .	B(return D) F(return J) L(return N) R(return T) V(return X) Z(return b) n(return k) h(return e)
VFU Control (PI)	Optional control from user system. Used for VFU control. Data Request/Strobe timing is same as for data lines.	p(return s)
Ready	Sent to user system by printer. True when no Check condition exists.	CC(return EE)
On Line	Sent to user system by printer. True when Ready line is true and operator has activated ON LINE Pushbutton. Enables interface activity.	y (return AA)
Interface Verify	Jumper in printer connector. Continuity informs user system that connector is properly seated.	x to v
+5V	Supply voltage for Exerciser only.	HH

address space (FF800-FFFFFFH) so that the reset vectors can be stored starting at location FFFF0H. The RAM address space spans 0-2FFFFH so that the interrupt vector table can be stored starting at location 0. The MUART is I/O mapped and its registers occupy even addresses from 0 to 1EH. Using an 8088 CPU the MUART must be placed in the 8086 mode since the INTA signal is used; hence the register addresses are all even numbers.

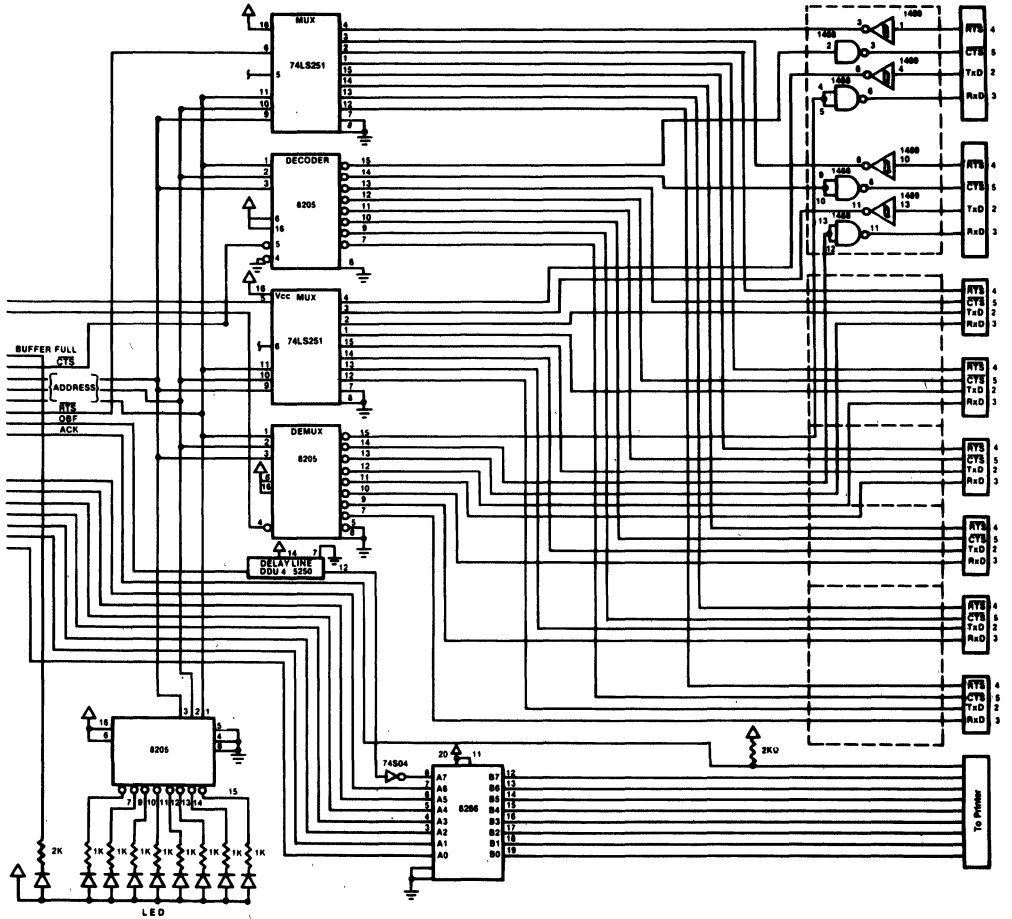
The line printer used provides a choice of two standard parallel interfaces: Centronics or Dataproducts. The

Centronics interface uses a two-wire handshake pulsed strobe where the transmitter asserts a complete strobe pulse before an acknowledge is received. The Dataproducts interface is an interlocking two-wire handshake. The Dataproducts interface was chosen since it is directly compatible with the MUART's two-wire byte handshake. The MUART could also be connected to the Centronics interface; however, additional hardware would be necessary to generate the pulsed strobe for correct interrupt operation. Figure 25 shows the timing of the Dataproducts interface and Table 6 lists the connector pin configuration.



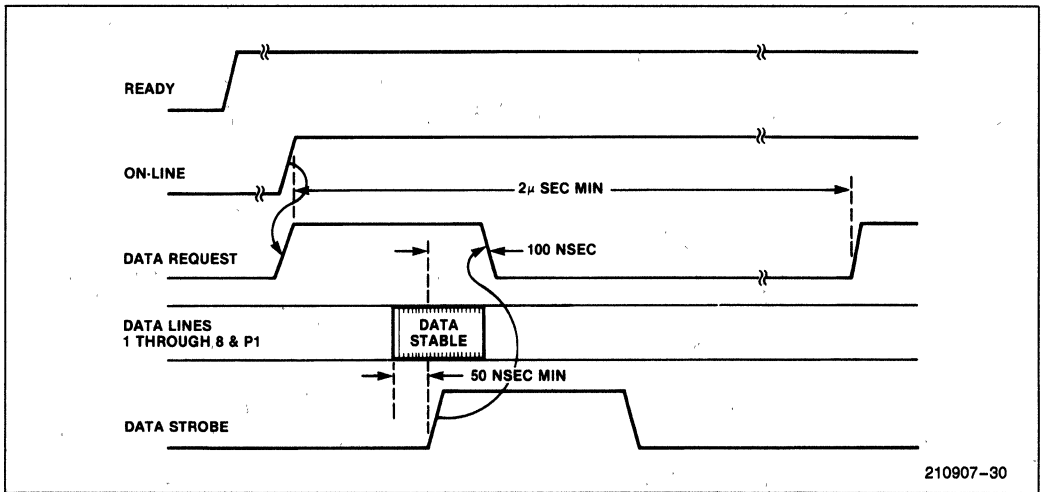
210907-28

Figure 24. Schematic of LPM



210907-29

Figure 24. Schematic of LPM (Continued)



210907-30

Figure 25. Timing of Dataproducts Interface

Only ten signals are used to interface the LPM to the line printer: Data Request, Data Strobe, and the eight data lines. The most significant data line is not used since the character code is 7-bit ASCII. Data Strobe connects to OBF on the MUART; however, for the Dataproducts interface this signal must be inverted. Data Request is connected to ACK on the MUART. When the line printer is ready to accept data, the Data Request signal goes high. The 8256 will not interrupt the CPU to transmit parallel data unless this signal is high.

The Dataproducts interface is slightly different from the MUART's two-wire handshake in that it latches the data on the leading edge of the strobe signal. When the MUART receives bytes it latches the data on the trailing edge. As a result the Dataproducts interface has a 50 ns setup time for data stable to the leading edge of Data Strobe. In the LPM hardware a delay line was used to realize this setup time.

Description of the Software

The software is written in PL/M and is broken up into four separate modules, each containing several procedures. A block diagram of the software structure is given in Figure 26. The modules are identified by the dotted boxes, and the procedures are identified by the solid boxes. Two or more procedures connected by a solid line means the procedure above calls the procedure below. The procedures without any solid lines connected above are interrupt procedures. They are entered when the MUART interrupts the CPU and vectors an indirect address to it.

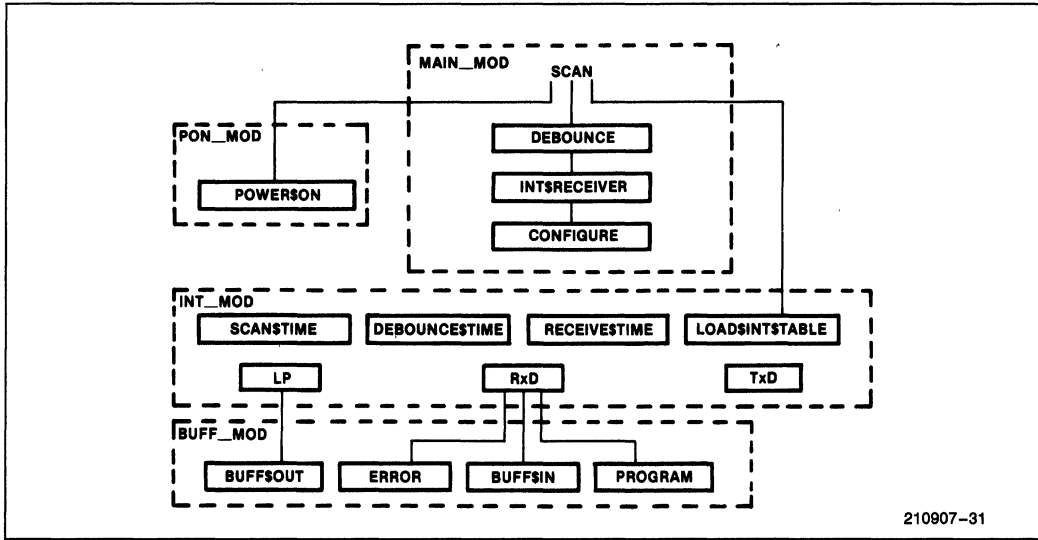
The LPM program uses nested interrupts; the priority of the interrupt procedures is given in Table 7.

Table 7. Line Printer Multiplexers' Interrupt Priority

Priority	Source
Highest	0
	1
	2
	3
	4
	5
	6
	7

The priority of the interrupts is not programmable but they are logically oriented so that for this application the priority is correct. In the steady state of the LPM's operation the UART will be receiving data, and the parallel port will be transmitting data. The serial receiver should be the highest priority since it can have overrun errors. This is the case because the debounce timer will be disabled, and the receive timeout interrupt will only occur when serial reception has ended. Therefore the RxD request can interrupt any other service routine, thus preventing any possibility of an overrun error.

On power-up the CPU branches from 0FFFF0H to the INITCODE routine which is included in the machine code by the MDS locator utility. INITCODE initializes the 8088's segment registers, stack pointer, and instruction pointer, then it disables interrupts and jumps into MAIN_MOD. The first executable instruction in MAIN_MOD calls POWERSON, which initializes



210907-31

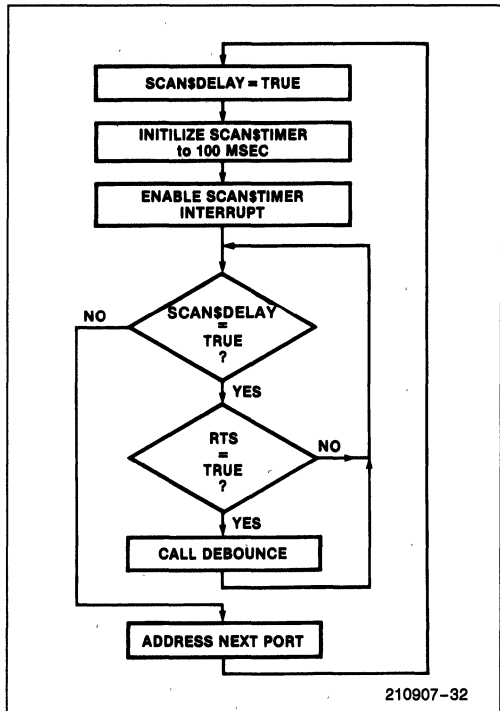
Figure 26. Block Diagram of LPM Software Structure

the MUART, flags, variables, and arrays. The MAIN_MOD calls LOAD\$INT\$TABLE, which initializes the interrupt vector table. The CPU's interrupt is then enabled and the program enters into a DO FOREVER loop which scans the eight serial ports for an RTS.

There are three software functions which employ the MUART's timers and interrupt controller to measure time intervals: SCAN, debounce, and INT\$RECEIVER. DEBOUNCE and INT\$RECEIVER procedures, employ the MUART's timers and interrupt controller to measure time intervals. The CPU remains in a loop for a specific amount of time before it proceeds with the next section of code. In this loop the CPU is waiting for a global status flag to change while servicing any interrupts which may occur. When the appropriate timer interrupt occurs, the interrupt service routine will set the global flag which causes the CPU to exit the loop and proceed to the next section of code. An example can be seen from the scan flow chart in Figure 27.

The first thing the program does before entering the loop is set the flag (in this case SCAN\$DELAY) TRUE. The timer is initialized and the loop is entered. As long as SCAN\$DELAY is TRUE the CPU will continue to sample RTS. If RTS remains false for more than 100 ms, the timer interrupts the CPU and the interrupt service routine sets SCAN\$DELAY FALSE. This causes the CPU to exit the loop and address the next port. The process is then repeated. If RTS becomes true while it is being sampled, the DEBOUNCE procedure is called.

DEBOUNCE does nothing more than wait 10 ms and sample RTS again using the same technique discussed above. If RTS is still valid INT\$RECEIVER is called, otherwise the CPU returns to scan.



210907-32

Figure 27. Scan Flow Chart

INIT\$RECEIVER calls CONFIGURE which programs the MUART for the bit rate, number of bits in a character, and parity format. This information is stored in an array called SERIAL\$FORMAT, which contains a byte for each port. The bytes in the SERIAL\$FORMAT array have the same bit definition as the two nibbles in the programming words in Figure 22. Upon returning to INIT\$RECEIVER the receiver is enabled, the receive timeout timer is initialized, and the timer and receiver interrupts are enabled. CTS on the serial port is then set true, and the CPU enters a loop which does nothing except wait for 18 seconds. If no characters are received within 18 seconds, the receive timeout interrupt occurs and the loop flag is set false, which causes the CPU to exit the loop. If a character is received, a receive interrupt occurs, and the CPU vectors into the RxD interrupt service routine.

Figure 28 shows a flow chart of the RxD interrupt service routine. This routine begins by reading the receive buffer and reinitializing the receive timeout timer. There are two conditions to check for before the character can be inserted into the FIFO. First, if there are any errors in the received character, an ERROR procedure is called which reports back to the serial port what

the error condition was. The character in error is discarded and the routine returns. The other condition is that if the received character is an ASCII ESC, the PROGRAM procedure is called. If neither one of these conditions occurs, the character is placed in the FIFO by the BUFF\$IN procedure.

The LP interrupt routine is entered when the byte handshake interrupt request is acknowledged. This routine simply calls the BUFF\$OUT procedure, which extracts a byte out of the FIFO. BUFF\$OUT returns the byte to the LP interrupt procedure, which then writes it to Port 2. One small problem with getting the handshake interrupt going is that the first byte has to be written to Port 2 before the first handshake interrupt will occur. The problem is that the line printer may not be ready for the first byte. This would be indicated by DATA REQUEST being low. If the byte was written to the LP while DATA REQUEST is low, it would be lost. Note that if the handshake interrupt is enabled while DATA REQUEST is low, then DATA REQUEST goes high, the interrupt will occur without writing the first byte. There are several ways to solve this problem. Port 1 can be read to find out what the state of the DATA REQUEST line is. If DATA REQUEST is low, the CPU can simply wait for the interrupt without writing the first byte. If DATA REQUEST is high, then the first data byte may be written. Another solution would be to write a NUL character as the first byte to Port 2. If DATA REQUEST is low, then a worthless character is lost. If DATA REQUEST is high, the NUL character would be sent to the line printer; however, it is not printed since NUL is a non-printable character. The LPM program uses the NUL character solution.

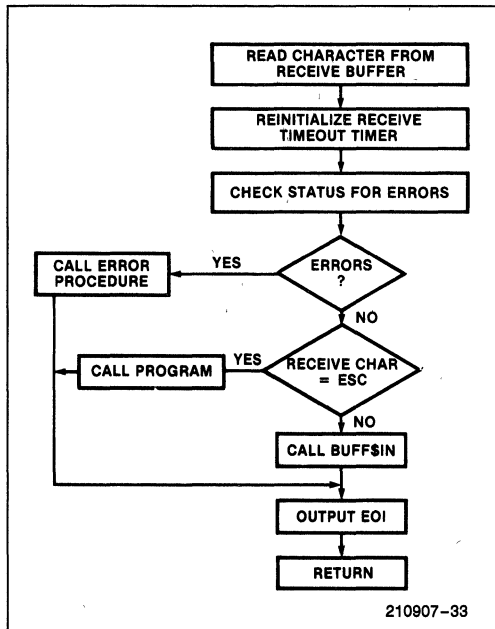


Figure 28. RxD Interrupt Procedure Flow Chart

BUFFER MANAGEMENT

The FIFO implementation uses an 8K byte array to store the characters. There are two pointers used as indexes in the array to address the characters: IN\$POINTER and OUT\$POINTER. IN\$POINTER points to the location in the array which will store the next byte of data inserted. OUT\$POINTER points to the next byte of data which will be removed from the array. Both IN\$POINTER and OUT\$POINTER are declared as words. Figure 29 illustrates the FIFO in a block diagram.

The BUFF\$IN procedure receives a byte from the RxD interrupt routine and stores it in the array location pointed to by IN\$POINTER, then IN\$POINTER is incremented. Similarly, when BUFF\$OUT is called by the LP interrupt routine, the byte in the array pointed

to by OUT\$POINTER is read. OUT\$POINTER is incremented, and the byte which was read is passed back to the LP interrupt routine. Since IN\$POINTER and OUT\$POINTER are always incremented, they must be able to roll over when they hit the top of the 8K byte address space. This is done by clearing the upper three bits of each pointer after it is incremented.

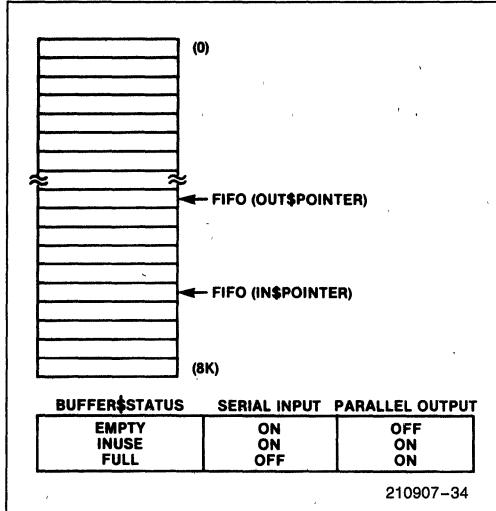


Figure 29. FIFO Structure and Status

IN\$POINTER and OUT\$POINTER not only point to the locations in the FIFO, they also indicate how many bytes are in the FIFO and whether the FIFO is full or empty. When a character is placed into the FIFO and IN\$POINTER is incremented, the FIFO is full if IN\$POINTER equals OUT\$POINTER. When a character is read from the FIFO and OUT\$POINTER is incremented, the FIFO is empty if OUT\$POINTER equals IN\$POINTER. If the buffer is neither full nor empty, then it is in use. A byte called BUFFER\$STATUS is used to indicate one of these three conditions.

The software uses the buffer status information to control the flow into and out of the FIFO. When the FIFO is empty the handshake interrupt must be turned off. When the FIFO is full, CTS must be sent false so that no more data will be received. If the buffer status is in use, CTS is true and the handshake interrupt is enabled.

Figure 30 shows the flow chart of the BUFF\$IN procedure. The BUFF\$IN procedure begins by checking the BUFFER\$STATUS. If it is empty and the character to be inserted into the FIFO is a CR or LF, the handshake interrupt is enabled, a NUL character is output, and the BUFFER\$STATUS is set to INUSE. The character

passed to BUFF\$IN from RxD is put into the FIFO. If the FIFO is now full, the BUFFER\$STATUS is set to FULL, CTS is set false, and the buffer full LED is turned on.

Figure 31 shows the flow chart of the BUFF\$OUT procedure. After the character is read from the FIFO, the FIFO is tested to determine if it is empty. If it is not empty, the BUFFER\$STATUS is FULL and there are 200 bytes available in the FIFO, serial data reception is reenabled, and the FIFO fills again. White data is being received from the workstation, CTS toggles high and low, filling up and emptying the last 200 bytes in the FIFO. Referring to the top of the flow chart (FIFO empty test) if it's empty, the BUFFER\$STATUS is set to EMPTY, and the handshake interrupt is disabled. During this time all interrupts are disabled at the CPU. (Remember that the RxD interrupt routine can interrupt the LP and BUFF\$OUT procedures since it has a higher priority, and the MUART is in the nested mode.)

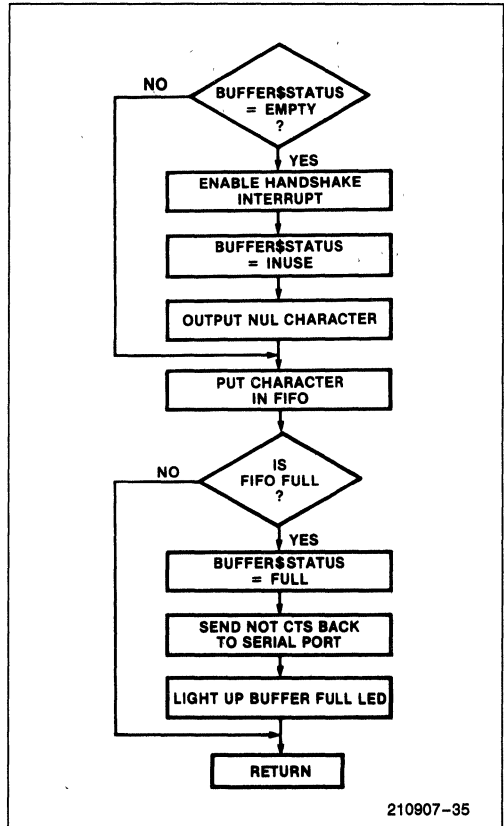


Figure 30. Flow Chart of the BUFF\$IN Procedure

If the CPU interrupt was not disabled during this time, the following events could occur which would cause the LPM to crash. Assume that the RxD interrupt occurred where the asterisk is in the flow chart, after BUFFER\$STATUS is set to EMPTY. The BUFF\$IN procedure would set BUFFER\$STATUS to INUSE and enable the handshake interrupt. When the RxD interrupt routine returned to BUFF\$OUT, the handshake interrupt is disabled, but the BUFFER\$STATUS is INUSE. The handshake interrupt could never be reenabled, and the FIFO would fill up. This is known as a critical section of code. Suspicion should arise for a critical section of code when two or more nested interrupt routines can affect the same status. One solution is to disable the interrupt flag at the CPU while the status and conditional operations are being modified.

The flow chart for the TxD interrupt procedure is given in Figure 32. For this program five different messages can be transmitted, and they are stored in ROM. It is possible to download the messages into a dedicated RAM buffer; however, the RAM buffer would have to be as large as the largest message. A more efficient way to transmit the messages is to read them from ROM. In this case the address of the first byte of the message would have to be accessible by the transmit interrupt procedure. Since parameters cannot be passed to interrupt procedures, this message pointer is declared PUBLIC in one module and EXTERNAL in the other modules.

To get the transmit interrupt started, the first byte of the message must be written to the transmit buffer.

When a section of code decides to transmit a message serially, it loads the global message pointer with the address of the first byte of the message, enables the transmit interrupt, and calls the TxD interrupt procedure. Calling the TxD interrupt procedure writes the first byte to the transmit buffer to initiate transmit interrupts. This can be done by calling PL/M's built-in procedure CAUSE\$INTERRUPT.

The transmit interrupt routine checks each byte before it writes it to the transmit buffer. The last character in each message is a 0, so if the character fetched is 0, the transmit interrupt is disabled and the character is ignored.

USING THE LPM WITH THE INTELLEC® MICROCOMPUTER DEVELOPMENT SYSTEM, SERIES II OR SERIES III

A special driver program was written for the MDS to communicate to the LPM. This program, called WRITE, reads a specified file from the disk, expands any TAB characters, and transmits the data through Serial Channel 2 to the LPM. Serial Channel 2 was chosen because CTS and RTS are brought out to the RS-232 connector. The WRITE program is listed in Appendix B. It was also necessary to modify the boot ROM of the development system so that Serial Channel 2 initializes with RTS false and a bit rate of 9600 bps.

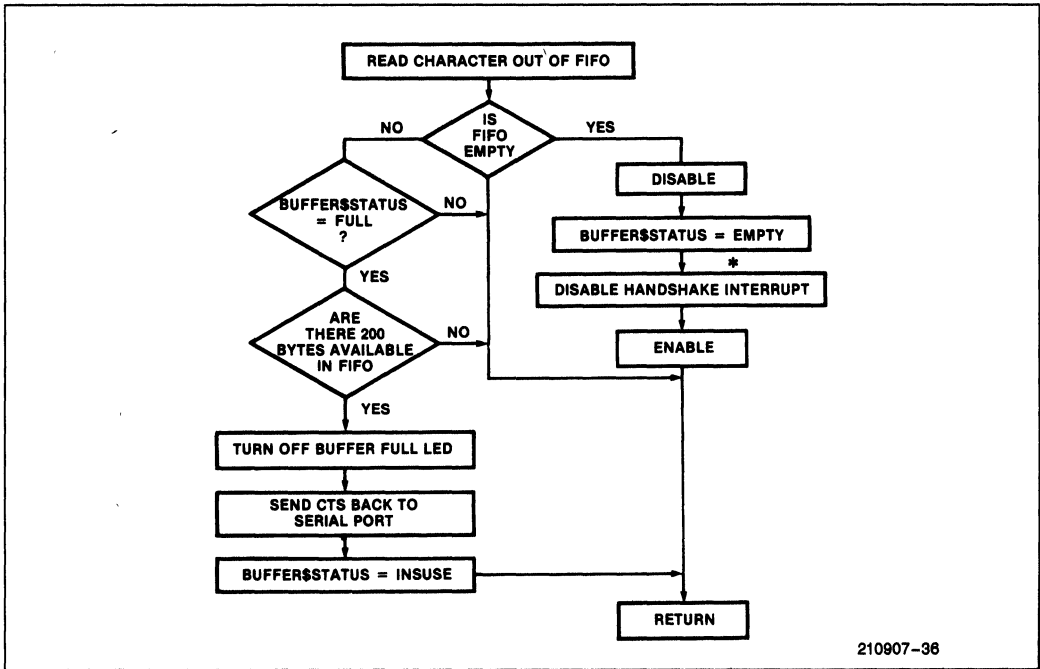


Figure 31. Flow Chart of the BUFF\$OUT Procedure

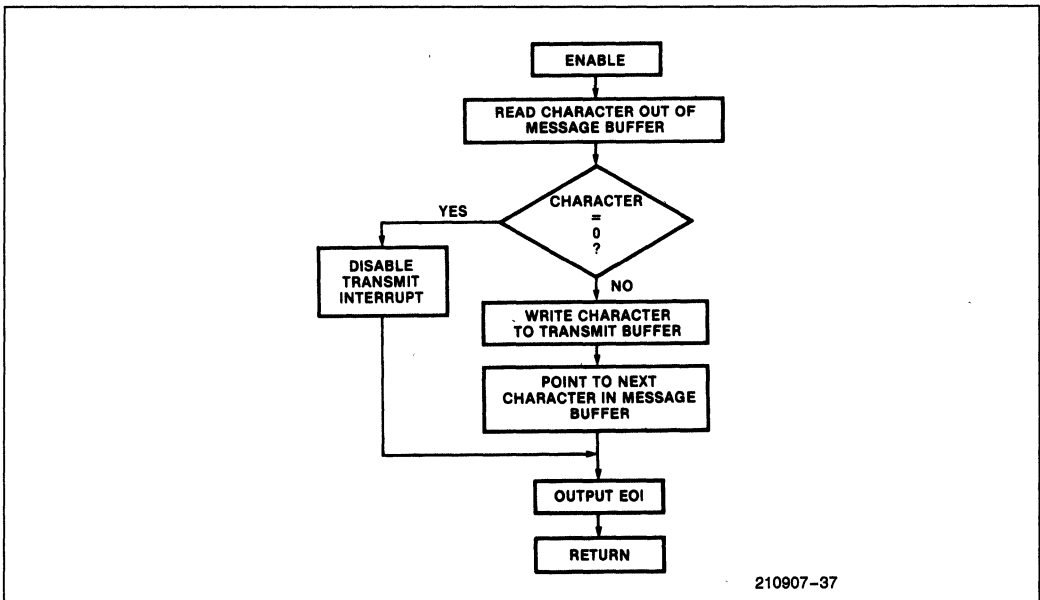


Figure 32. Flow Chart for TxD Interrupt Procedure

APPENDIX A LISTING OF THE LINE PRINTER MULTIPLEXER SOFTWARE

PL/M-86 COMPILER MAINMOD

SERIES-III PL/M-86 V1 0 COMPILATION OF MODULE MAINMOD
OBJECT MODULE PLACED IN '1 MAIN OBJ'
COMPILER INVOKED BY PLM86 86 F1 MAIN SRC

```

/*****
*
*           MAIN MODULE FOR THE LINE PRINTER MULTIPLEXER
*
*****/

$DEBUG
MAIN$MOD DD.

/*****
* PORT 1 BIT CONFIGURATION
*
* BUFFER FULL    CTS    ADDRESS    RTS    TWO WIRE HANDSHAKE
*            B7    B6            B5 B4 B3    B2            B1 B0
*****/

2 1    DECLARE LIT                    LITERALLY            'LITERALLY',
         TRUE                        LIT                    'OFF',
         FALSE                       LIT                    '0',
         FOREVER                     LIT                    'WHILE 1',

         CMD$1                        LIT                    '0',            /*8256 REGISTERS*/
         CMD$2                        LIT                    '2',
         CMD$3                        LIT                    '4',
         MODE                         LIT                    '6',
         PORT$1$CTRL                  LIT                    '8',
         SET$INT                      LIT                    '0AH',
         INT$EN                        LIT                    '0AH',
         RST$INT                      LIT                    '0CH',
         INT$ADDR                     LIT                    '0CH',
         TX$BUFF                      LIT                    '0EH',
         RX$BUFF                      LIT                    '0EH',
         PORT$1                        LIT                    '10H',
         PORT$2                        LIT                    '12H',
         DEBOUNCE$TIMER               LIT                    '14H',
         SCAN$TIMER                   LIT                    '1AH',
         RECEIVE$TIMER                LIT                    '1CH',
         STATUS$REG                   LIT                    '1EH',

         SCAN$INT                     LIT                    '40H',
         DEBOUNCE$INT                 LIT                    '01H',
         RECEIVER$INT                 LIT                    '10H',
         TIME$OUT$INT                 LIT                    '0BH',
         TRANSMIT$INT                 LIT                    '20H',

         EMPTY                        LIT                    '0',
         INUSE                         LIT                    '1',
         FULL                         LIT                    '2',

         RTS                          LIT                    '(INPUT(PORT$1) AND 04H)',

```

210907-38

PL/M-B6 COMPILER MAINMOD

```

                BEGIN          LABEL          PUBLIC
                TEMP          BYTE           PUBLIC
                SCAN$DELAY    BYTE           PUBLIC
                DEBOUNCE$DELAY BYTE           PUBLIC
                RECEIVE$DELAY  BYTE           PUBLIC
                PORT$PTR       BYTE           PUBLIC
                SERIAL$FORMAT(8) BYTE         PUBLIC, /* PEN EP L1 LO B3 B2 B1 B0 */

                MESSAGE$PTR   POINTER        EXTERNAL
                J              BYTE          EXTERNAL
                OK(1)         BYTE          EXTERNAL
                BUFFER$STATUS  BYTE          EXTERNAL

/*****
 *          EXTERNAL PROCEDURE DECLARATIONS
 *****/

3 1  POWER$ON PROCEDURE EXTERNAL,
4 2  END POWER$ON.

5 1  LOAD$INT$TABLE PROCEDURE EXTERNAL,
6 2  END LOAD$INT$TABLE.

/*****
 *          SET THE BIT RATE AND DATA FORMAT FOR THE SERIAL PORT
 *****/

7 1  CONFIGURE PROCEDURE, /*Initialize bit rate and data format*/
8 2  TEMP=SERIAL$FORMAT(SHR(PORT$PTR,3)),
9 2  OUTPUT(CMD$1)=(SHL(TEMP,2) AND 0C0H) OR 03H,
10 2 OUTPUT(CMD$2)=(TEMP OR 30H),
11 2  END CONFIGURE.

/*****
 *          INITIALIZE SERIAL RECEIVER
 *****/

12 1  INIT$RECEIVER PROCEDURE,
13 2  CALL CONFIGURE, /*Initialize 8256 serial port*/
14 2  RECEIVE$DELAY=TRUE,
15 2  OUTPUT(CMD$3)=0C0H, /*Enable serial receiver*/
16 2  OUTPUT(RECEIVE$TIMER)=70, /*18 second TIME$OUT*/
17 2  OUTPUT(SET$INT)=1BH, /*Enable RECEIVER and TIME$OUT interrupts*/
18 2  IF (BUFFER$STATUS<<FULL)
      THEN
19 2      OUTPUT(PORT$1)=(INPUT(PORT$1) AND 0BFH), /*Send CTS TRUE*/
20 2      DO WHILE RECEIVE$DELAY=TRUE, /* Wait here while receiving serial data */
21 3      END.

      /* After 18 seconds of not receiving a character, proceed */

22 2  OUTPUT(SET$INT)=TRANSMIT$INT, /* Send the terminating message */
23 2  J=0,
24 2  MESSAGE$PTR=@OK(0),
25 2  CAUSE$INTERRUPT (45H),

```

210907-39

```
PL/M-86 COMPILER    MAINMOD
```

```

26 2      OUTPUT(PORT%1)=(INPUT(PORT%1) OR 40H). /*Send CTS FALSE*/
27 2      OUTPUT(RST%INT)=1BH. /*Clear RECEIVER and TIMER Interrupts*/
28 2      OUTPUT(CMD%3)=40H. /*Disable serial receiver*/
29 2      END INIT%RECEIVER.

/*****
*                      DEBOUNCE RTS                      *
*****/

30 1      DEBOUNCE PROCEDURE,
31 2      DEBOUNCE%DELAY=TRUE,
32 2      OUTPUT(DEBOUNCE%TIMER)=10. /* 10 msec debounce time delay */
33 2      OUTPUT(SET%INT)=DEBOUNCE%INT,
34 2          DO WHILE DEBOUNCE%DELAY=TRUE.
35 3          END,
36 2      IF RTS=0 THEN CALL INIT%RECEIVER,
38 2      END DEBOUNCE.

/*****
*                      BEGIN MAIN PROGRAM                *
*****/

39 1      BEGIN CALL POWER%ON.
40 1      CALL LOAD%INT%TABLE.
41 1      ENABLE,
42 1      DO FOREVER,
43 2          SCAN%DELAY=TRUE,
44 2          OUTPUT(SCAN%TIMER)=100. /*Spend 100 msec on each serial port sampling RTS*/
45 2          OUTPUT(SET%INT)=SCAN%INT,
46 2              DO WHILE SCAN%DELAY=TRUE. /*Sample RTS*/
47 3              IF RTS=0
48 3                  THEN
49 3                      CALL DEBOUNCE,
50 2          TEMP=INPUT(PORT%1). /*Increment PORT%PTR*/
51 2          PORT%PTR=TEMP AND 3BH,
52 2          TEMP=TEMP AND (NOT 3BH),
53 2          PORT%PTR=(PORT%PTR+B) AND 3BH,
54 2          OUTPUT(PORT%1)=TEMP OR PORT%PTR. /*Look at next serial port*/
55 2      END. /*DO FOREVER*/
56 1      END MAIN%MOD.

```

```
MODULE INFORMATION
```

```
CODE AREA SIZE    = 011CH    284D
```

```
PL/M-86 COMPILER    MAINMOD
```

```

CONSTANT AREA SIZE = 0000H    0D
VARIABLE AREA SIZE = 000DH    13D
MAXIMUM STACK SIZE = 000CH    12D
159 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS

```

```
END OF PL/M-86 COMPILATION
```

```
210907-40
```

PL/M-86 COMPILER INTMOD

SERIES 111 PL/M-86 V1.11 COMPILATION OF MODULE INTMOD
 OBJECT MODULE PLACED IN FILE INT.OBJ
 COMPILER INVOKED BY PLM86 86 11 INT.CRM

```

/*****
 *
 *      INTERRUPT MODULE  CONTAINS ALL INTERRUPT ROUTINES
 *      PLUS LOCAL INTERRUPT TABLE PROCEDURE
 *
 *****/

$DEBUG
1  INT$MOD DO,
  $NOLIST

3  1  DECLARE
      ESC          LIT          'IBH',
      SCAN$DELAY  BYTE         EXTERNAL,
      DEBOUNCE$DELAY  BYTE     EXTERNAL,
      RECEIVE$DELAY  BYTE     EXTERNAL,
      MESSAGE$PTR  POINTER    EXTERNAL,
      J            BYTE         EXTERNAL.

/*****
 *
 *      MESSAGES SENT TO SERIAL PORTS
 *
 *****/

OK (*) BYTE PUBLIC DATA ('TRANSMISSION COMPLETE', 0AH, 0DH, 00),
BREAK (*) BYTE PUBLIC DATA ('BREAK DETECT ERROR', 0AH, 0DH, 00),
PARITY (*) BYTE PUBLIC DATA ('PARITY ERROR DETECTED', 0AH, 0DH, 00),
FRAME (*) BYTE PUBLIC DATA ('FRAMING ERROR DETECTED', 0AH, 0DH, 00),
OVER$RUN(*) BYTE PUBLIC DATA ('OVER RUN ERROR DETECTED', 0AH, 0DH, 00).

/*****
 *
 *      EXTERNAL PROCEDURES CALLED BY THE INTERRUPT ROUTINES
 *
 *****/

4  1  ERROR PROCEDURE (STATUS) EXTERNAL
5  2  DECLARE STATUS BYTE,
6  2  END ERROR,

7  1  PROGRAM PROCEDURE EXTERNAL,
8  2  END PROGRAM,

9  1  BUFF$IN PROCEDURE (CHAR) EXTERNAL,
10 2  DECLARE CHAR BYTE,
11 2  END BUFF$IN,

12 1  BUFF$OUT PROCEDURE BYTE EXTERNAL,
13 2  END BUFF$OUT.

/*****
 *
 *      LOAD THE INTERRUPT TABLE
 *
 *****/

14 1  LOAD$INT$TABLE PROCEDURE PUBLIC,

```

210907-41


```

15 2 CALL SET$INTERRUPT (40H,DEBOUNCE$TIME),
16 2 CALL SET$INTERRUPT (43H,RECEIVE$TIME),
17 2 CALL SET$INTERRUPT (44H,RXD),
18 2 CALL SET$INTERRUPT (45H,TXD),
19 2 CALL SET$INTERRUPT (46H,SCAN$TIME),
20 2 CALL SET$INTERRUPT (47H,LP),

21 2 END LOAD$INT$TABLE,

/*****
 * INTERRUPT ROUTINES
 *****/

/*****
 * SET SCAN DELAY FLAG FALSE
 *****/

22 1 SCAN$TIME PROCEDURE INTERRUPT 46H,

23 2 ENABLE,
24 2 SCAN$DELAY=FALSE,
25 2 OUTPUT(CMD$3)=BBH, /*Output end for nested mode*/
26 2 END SCAN$TIME,

/*****
 * SET DEBOUNCE DELAY FLAG FALSE
 *****/

27 1 DEBOUNCE$TIME PROCEDURE INTERRUPT 40H,
28 2 DEBOUNCE$DELAY=FALSE,
29 2 OUTPUT(CMD$3)=BBH,
30 2 END DEBOUNCE$TIME,

/*****
 * SET RECEIVE DELAY FLAG FALSE
 *****/

31 1 RECEIVE$TIME PROCEDURE INTERRUPT 43H,
32 2 ENABLE,
33 2 RECEIVE$DELAY=FALSE,
34 2 OUTPUT(CMD$3)=BBH,
35 2 END RECEIVE$TIME,

/*****
 * READ SERIAL RECEIVE BUFFER
 *****/

36 1 RXD PROCEDURE INTERRUPT 44H,

37 2 DECLARE
        STATUS      BYTE,
        CHAR        BYTE,

38 2 CHAR=INPUT(RX$BUFF),
39 2 OUTPUT(RECEIVE$TIMER)=70, /* REINITIALIZE RECEIVE TIME OUT */
40 2 STATUS=INPUT(STATUS$REG) AND OFH,

```

PL/M-86 COMPILER INTMOD

```

41 2      IF STATUS.0
42 2          THEN
            CALL ERROR (STATUS),
43 2      ELSE IF CHAR=ESC
44 2          THEN
            CALL PROGRAM,
45 2      ELSE
46 2          CALL BUFF$IN (CHAR),
47 2      OUTPUT (CMD$3)=8BH,
            END RXD,

            /*****
            *          SEND A BYTE TO THE LINE PRINTER
            *****/

48 1      LP PROCEDURE INTERRUPT 47H,
49 2      ENABLE,
50 2      OUTPUT (PORT$2)=BUFF$OUT,
51 2      OUTPUT (CMD$3)=8BH,
52 2      END LP,

            /*****
            *          SEND A BYTE TO THE SERIAL PORTS
            *****/

53 1      TXD PROCEDURE INTERRUPT 45H,
54 2      DECLARE
            MESSAGE BASED MESSAGE$PTR (1) BYTE,
            I
            BYTE,
55 2      ENABLE,
56 2      I=MESSAGE{J},
57 2      IF I<>0
            THEN OUTPUT (TX$BUFF)=I,
59 2      ELSE OUTPUT (RST$INT)=TRANSMIT$INT,
60 2      J=J+1,
61 2      OUTPUT (CMD$3)=8BH,
62 2      END TXD,
63 1      END INT$MOD.

```

MODULE INFORMATION

```

CODE AREA SIZE      = 01BDH    445D
CONSTANT AREA SIZE  = 007BH    120D
VARIABLE AREA SIZE  = 0003H     3D
MAXIMUM STACK SIZE  = 0022H    34D
181 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS

```

END OF PL/M-86 COMPILATION

210907-43

PL/M-86 COMPILER BUFFMOD

SERIES-III PL/M-86 V1 0 COMPILATION OF MODULE BUFFMOD
 OBJECT MODULE PLACED IN F1 BUFF OBJ
 COMPILER INVOKED BY PLM86 86 F1 BUFF SRC

```

/*****
*
*   BUFFER MODULE  INSERTS AND REMOVES CHARACTERS FROM FIFO
*                   REPORTS SERIAL RECEIVE ERRORS AND
*                   RE-PROGRAMS SERIAL PORTS
*
*****/

$DEBUG
BUFF$MOD DO,
$NOLIST

3 1  DECLARE
      MESSAGE$PTR    POINTER    PUBLIC,
      J              BYTE      PUBLIC,
      OK(1)          BYTE      EXTERNAL,
      BREAK(1)       BYTE      EXTERNAL,
      PARITY(1)      BYTE      EXTERNAL,
      FRAME(1)       BYTE      EXTERNAL,
      OVERSRUN(1)    BYTE      EXTERNAL,
      SERIAL$FORMAT(1) BYTE    EXTERNAL,
      PORT$PTR       BYTE      EXTERNAL,

      FIFO(8192)     BYTE,
      IN$POINTER     WORD      PUBLIC,
      OUT$POINTER    WORD      PUBLIC,
      BUFFER$STATUS  BYTE      PUBLIC,

/*****
*
*   INSERT CHARACTER INTO FIFO
*
*****/

4 1  BUFF$IN PROCEDURE (CHAR) PUBLIC,
5 2  DECLARE
      CHAR    BYTE,

6 2  IF ((BUFFER$STATUS=EMPTY) AND ((CHAR=LF) OR (CHAR=CR)))
7 2  THEN
8 3  DO,
9 3  OUTPUT(SET$INT)=HANDSHAKE$INT, /* Enable two-wire handshake interrupt */
10 3  BUFFER$STATUS=INUSE,
11 3  OUTPUT(PORT$2)=0, /* Output NULL character to get
                        the interrupt started */
12 3  END,
12 2  FIFO(IN$POINTER)=CHAR, /* Put CHAR into FIFO and increment pointer */
13 2  IN$POINTER=((IN$POINTER+1) AND 1FFF),

14 2  IF (((IN$POINTER+4) AND 1FFF)=OUT$POINTER) /* If the buffer is full stop reception */
15 2  THEN
      DO, /* Send CTS FALSE, and light up buffer full LED */

```

210907-44

```

PL/M-86 COMPILER   BUFFMOD

16 3      OUTPUT(PORT%1)=((INPUT(PORT%1) OR 40H) AND 7FH),
17 3      BUFFER%STATUS=FULL,
18 3      END,
19 2      END BUFF%IN;

/*****
 *          REMOVE CHARACTER FROM FIFO
 *****/

20 1      BUFF%OUT PROCEDURE BYTE PUBLIC,
21 2      DECLARE CHAR BYTE,
22 2      CHAR=FIFO(OUT%POINTER),
23 2      OUT%POINTER=((OUT%POINTER+1) AND 1FFFH),
24 2      IF OUT%POINTER=IN%POINTER /* If the buffer is EMPTY disable the output to LP */
      THEN
25 2          DO,
26 3              DISABLE,
27 3              BUFFER%STATUS=EMPTY,
28 3              OUTPUT(RST%INT)=HANDSHAKE%INT,
29 3              ENABLE,
30 3      END,

/* If the buffer is ready to fill up again then send CTS TRUE */

31 2      ELSE IF ((BUFFER%STATUS=FULL) AND (((OUT%POINTER-200) AND 1FFFH)=IN%POINTER))
      THEN
32 2          DO: /* Turn off buffer-full LED and turn on CTS */
33 3              OUTPUT(PORT%1)=((INPUT(PORT%1) AND 0BFH) OR 80H),
34 3              BUFFER%STATUS=INUSE,
35 3      END,

      RETURN CHAR;

37 2      END BUFF%OUT;

/*****
 *          SEND ERROR MESSAGE TO SERIAL PORT
 *****/

38 1      ERROR PROCEDURE (STATUS) PUBLIC,
39 2      DECLARE STATUS BYTE,
      MESSAGE BASED MESSAGE%PTR(1) BYTE,

40 2          IF (STATUS AND 02H)>0
      THEN
41 2              STATUS=2;
42 2      ELSE IF (STATUS AND 04H)>0
      THEN
43 2              STATUS=3;
44 2      ELSE IF (STATUS AND 08H)>0
      THEN
45 2              STATUS=4;
46 2      ELSE IF (STATUS AND 01H)>0
      THEN
47 2              STATUS=1;
      DO CASE STATUS:
49 3          MESSAGE%PTR=@FRAME(0),
50 3

```

210907-45

PL/M-86 COMPILER BUFFMOD

```

51 3          MESSAGE$PTR=@OVER$RUN(0),
52 3          MESSAGE$PTR=@PARITY(0),
53 3          MESSAGE$PTR=@BREAK(0),
54 3          END;

55 2          J=1;          /* Point to second character in string */
56 2          OUTPUT(SET$INT)=TRANSMIT$INT,
57 2          OUTPUT(TX$BUFF)=MESSAGE(0),
58 2          END ERROR;

          /*****
          *          RELOAD SERIAL PORT CONFIGURE BYTE
          *****/

59 1          PROGRAM:PROCEDURE PUBLIC,
60 2          DECLARE TEMP    BYTE,
                   CHAR    BYTE,

61 2          DO WHILE (INPUT(STATUS$REG) AND 40H)=0; /* Wait for next byte */
62 3          END;

63 2          CHAR=INPUT(RX$BUFF);

64 2          IF CHAR=0          /* If second byte is 0, exit program mode */
65 3          THEN
66 4          DO,
67 4          OUTPUT(RECEIVE$TIMER)=70,
68 4          CALL BUFF$IN (CHAR),
69 4          RETURN;
70 2          END;

71 2          TEMP=(CHAR AND 0FH);

72 2          DO WHILE (INPUT(STATUS$REG) AND 40H)=0;
73 3          END;

74 2          TEMP=(INPUT(RX$BUFF) AND 0FH) OR SHL(TEMP,4),
75 2          SERIAL$FORMAT (SHR(PORT$PTR,3))=TEMP,
76 1          END PROGRAM;

76 1          END BUFF$MOD;

```

MODULE INFORMATION.

```

CODE AREA SIZE    = 01E4H    484D
CONSTANT AREA SIZE = 0000H    0D
VARIABLE AREA SIZE = 200BH    8203D
MAXIMUM STACK SIZE = 000AH    10D
189 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS

```

END OF PL/M-86 COMPILATION

210907-46

PL/M-86 COMPILER PON_MOD

SERIES-III PL/M-86 V1.0 COMPILATION OF MODULE PON_MOD
 OBJECT MODULE PLACED IN F1 PON.ORG
 COMPILER INVOKED BY PLMB6 86 F1 PON SRC

```

$DEBUG
/*****
*
*   POWER ON INITIALIZATION OF THE LINE PRINTER MULTIPLEXER
*
*   *****/
1  PON_MOD DD.
$NOLIST
3  1  DECLARE BUFFER$STATUS  BYTE  EXTERNAL,
      IN$POINTER  WORD  EXTERNAL,
      OUT$POINTER  WORD  EXTERNAL,
      PORT$PTR  BYTE  EXTERNAL,
      SERIAL$FORMAT(8)  BYTE  EXTERNAL,
4  1  POWER$ON PROCEDURE PUBLIC,
5  2  DECLARE I  BYTE,
6  2  DISABLE,
      /* INITIALIZE THE MUART */
7  2  OUTPUT(CMD$1)=01000011B, /*8086 MODE, FREQ=1KHz, 1 STOP BIT, &
8  2  OUTPUT(CMD$2)=10110100B, /*7 BITS/CHARACTER*/
9  2  OUTPUT(CMD$3)=01111111B, /*ODD PARITY, SYSTEM CLOCK=1 024 MHz, &
10 2  OUTPUT(CMD$3)=10110001B, /*9600 bps*/
11 2  OUTPUT(MODE)=10000101B, /*CLEAR CMD$3 REGISTER*/
      /*RESET, INTERRUPT ACKNOWLEDGE ENABLE, &
      NESTED INTERRUPT MODE*/
      /*CASCADE TIMERS 35 FOR THE
      RECEIVE$TIME$OUT TIMER, BYTE OUTPUT MODE*/
12 2  OUTPUT(PORT$1$CTRL)=11111000B, /*PORT 1 RTS=INPUT, THE REST ARE OUTPUTS*/
13 2  OUTPUT(PORT$1)=11000000B, /*POINT TO THE FIRST PORT, CTS IS F
      AND BUFFER IS NOT FULL*/
      /* INITIALIZE FLAGS, VARIABLES, AND ARRAYS */
14 2  BUFFER$STATUS=EMPTY,
15 2  IN$POINTER=0, OUT$POINTER=0,
17 2  PORT$PTR=0,
18 2  DD I=0 TO 7,

```

210907-47

```
PL/M-86 COMPILER  PON_MOD
```

```
19 3          SERIAL$FORMAT(1)=10010100B.  /* ON POWER-UP ALL EIGHT SERIAL PORTS
20 3          END;                          DEFAULT TO 9600 bps, ODD PARITY, AND
21 2          END POWER$ON.                  7 BITS/CHARACTER*/
22 1          END PON_MOD.
```

```
MODULE INFORMATION.
```

```
CODE AREA SIZE      = 0058H      88D
CONSTANT AREA SIZE  = 0000H      0D
VARIABLE AREA SIZE  = 0001H      1D
MAXIMUM STACK SIZE  = 0002H      2D
98 LINES READ
0 PROGRAM WARNINGS
0 PROGRAM ERRORS
```

```
END OF PL/M-86 COMPILATION
```

```
210907-48
```


PL/M-80 COMPILER

```

          PROCEDURE (AFTN,BUFFER,COUNT,STATUS) EXTERNAL,
11  2      DECLARE (AFTN,BUFFER,COUNT,STATUS) ADDRESS,
12  2      END WRITE,

13  1      CLOSE
          PROCEDURE (AFTN,STATUS) EXTERNAL,
14  2      DECLARE (AFTN,STATUS) ADDRESS,
15  2      END CLOSE,

16  1      ERROR
          PROCEDURE (ERRNUM) EXTERNAL,
17  2      DECLARE (ERRNUM) ADDRESS,
18  2      END ERROR,

19  1      EXIT
          PROCEDURE EXTERNAL,
20  2      END EXIT,

          /*****
          *          WAIT UNTIL USART TRANSMITTER IS READY
          *****/

21  1      TXRDY
          PROCEDURE,
22  2      DO WHILE: ( (INPUT(USART*STATUS) AND 01H) = 0 ),
23  3      END,
24  2      END TXRDY,

          /*****
          *          BEGIN MAIN PROGRAM
          *****/

25  1      BEGIN
          STATUS=0;

26  1          CALL READ(1, FILENAME,15, ACTUAL, STATUS), /* Read in file and path name */
27  1          IF STATUS <> 0
28  1              THEN
29  1                  GO TO DONE,

29  1          CALL OPEN( AFT*IN, FILENAME,1,0, STATUS), /* Open up the file */
30  1          IF STATUS <> 0
31  1              THEN
32  1                  GO TO DONE,

32  1      REPEAT
          CALL READ(AFT*IN, BUFFER,32000, ACTUAL, STATUS),
33  1          IF STATUS <> 0
34  1              THEN
35  1                  GO TO DONE,

35  1          CHAR*COUNT=0, /* CHAR*COUNT keeps track of the tab columns in each line */
36  1          OUTPUT(USART*STATUS)= RTS OR TXEN,

```

210907-50

PL/M-80 COMPILER

```

37 1      IF BUFFER(0)=FORM$FEED /* If the first character is a form feed
                                         remove it. Form feeds are inserted at the
                                         end of a file */
      THEN
38 1          DO,
39 2          BUFFER(0)=00H,
40 2          CHAR$COUNT=-1,
41 2          END,
42 1      DO I = 0 TO (ACTUAL - 1),
43 2          IF (BUFFER(I)=TAB) /* Replace TAB characters with the
                                         appropriate number of spaces */
      THEN
44 2              DO,
45 3              CALL TXRDY,
46 3              OUTPUT(USART$DATA)=SP,
47 3              CHAR$COUNT=CHAR$COUNT+1,
48 3              DO WHILE ((CHAR$COUNT AND 0007H) < 0),
49 4                  CALL TXRDY,
50 4                  OUTPUT(USART$DATA)=SP,
51 4                  CHAR$COUNT=CHAR$COUNT+1,
52 4              END,
53 3          END,
      ELSE
54 2          IF BUFFER(I)=ESC /* If outputting ESC, then output a
                                         0 next so the LPM does not get
                                         re-programmed */
      THEN
55 2              DO J=0 TO 1,
56 3              CALL TXRDY,
57 3              OUTPUT(USART$DATA)=0,
58 3              END,
      ELSE /* If the character is not an ESC or TAB then output it */
59 2          DO,
60 3          CALL TXRDY,
61 3          OUTPUT(USART$DATA)=BUFFER(I),
62 3          IF (BUFFER(I) < 1FH AND BUFFER(I) > 7FH)
      THEN /* Only increment CHAR$COUNT
                                         for printable characters */
63 3              CHAR$COUNT=CHAR$COUNT+1,
64 3          IF ((BUFFER(I)=CR) OR (BUFFER(I)=LF))
      THEN /* Reset CHAR$COUNT for CR or LF */
65 3              CHAR$COUNT=0,
66 3          END,
67 2      END,
68 1      IF ACTUAL = 32000 /*If the file is more than 32K, get some more data */
      THEN
69 1          GO TO REPEAT,
70 1      CALL TXRDY, /* Terminate file with CR, LF, and FF */
71 1      OUTPUT(USART$DATA)=CR,
72 1      CALL TXRDY,

```

210907-51

PL/M-80 COMPILER

```
73 1      OUTPUT(USART*DATA)=LF,
74 1      CALL TXRDY,
75 1      OUTPUT(USART*DATA)=FORM*FEED,

76 1      OUTPUT(USART*STATUS)=RXE OR TXEN, /* Shut off RTS */
77 1      CALL CLOSE (AFT*IN, STATUS),

78 1      DO I=0 TO 14, /* Output sign off message */
79 2          IF FILENAME(I)=CR
80 2              THEN
81 2                  GO TO SKIP,
82 2                  BYE(I+5)=FILENAME(I),
83 1      SKIP  CALL WRITE(0, BYE, 42, STATUS),
84 1      GO TO NEXT,
85 1      DONE  CALL ERROR(STATUS),
86 1      NEXT  CALL EXIT,
87 1      END WRITE*MOD,
```

MODULE INFORMATION

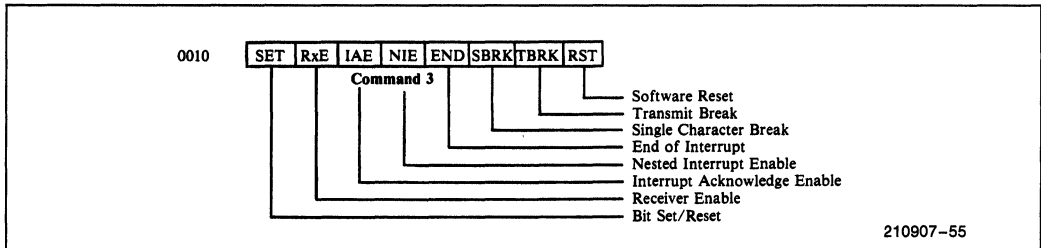
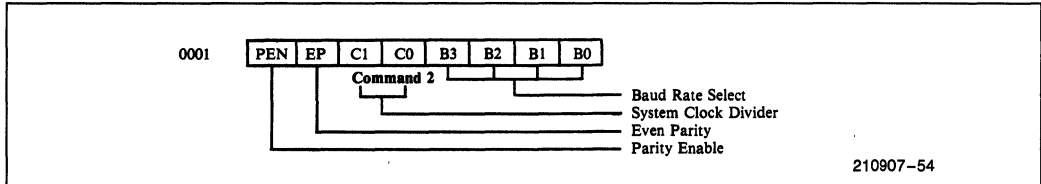
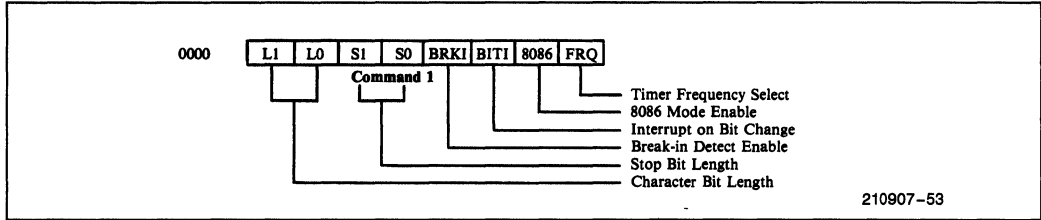
```
CODE AREA SIZE   = 0209H   521D
VARIABLE AREA SIZE = 7D44H 32068D
MAXIMUM STACK SIZE = 000BH   8D
191 LINES READ
0 PROGRAM ERRORS
```

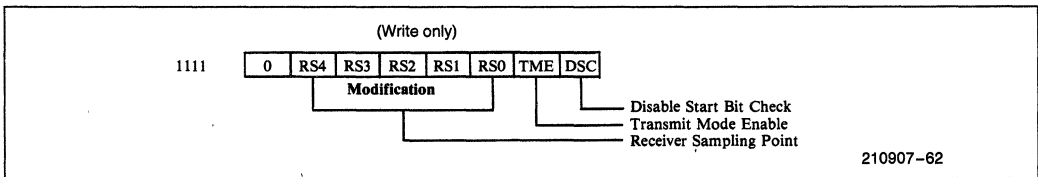
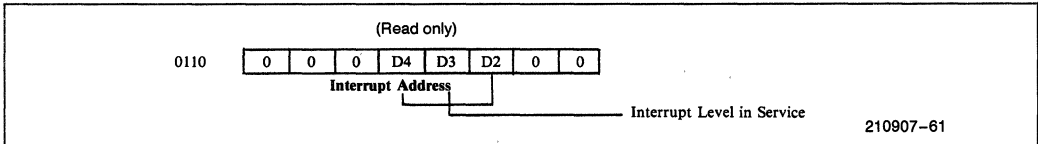
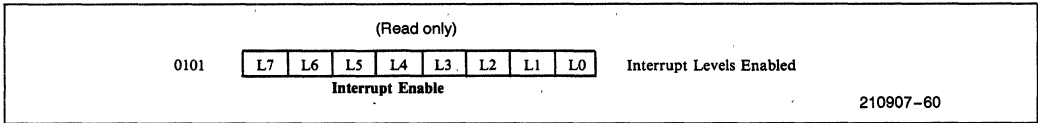
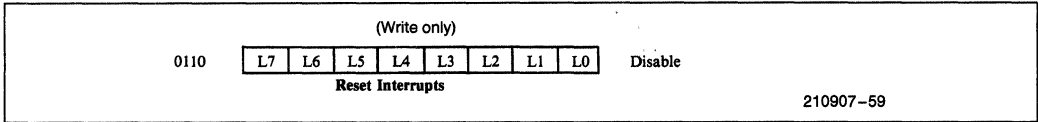
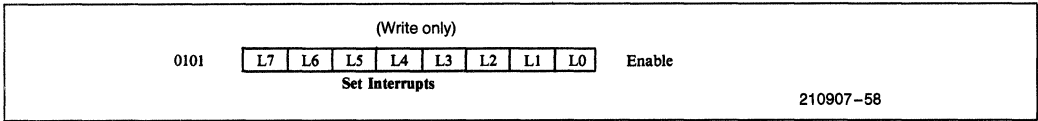
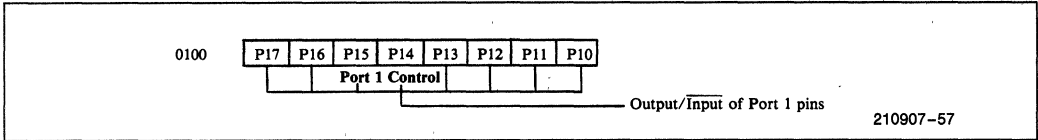
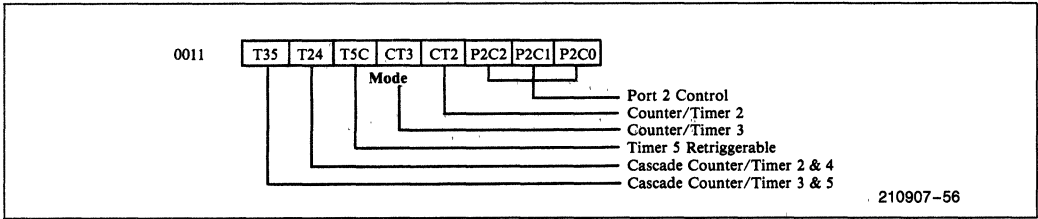
END OF PL/M-80 COMPILATION

210807-52

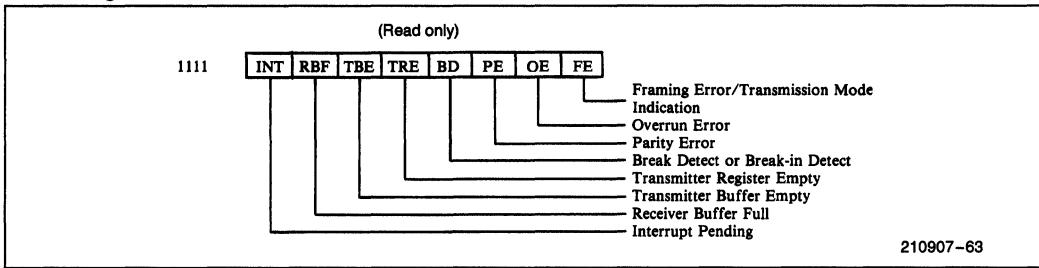
APPENDIX C MUART REGISTERS

8085 Mode: AD3 AD2 AD1 AD0
8085 Mode: AD4 AD3 AD2 AD1

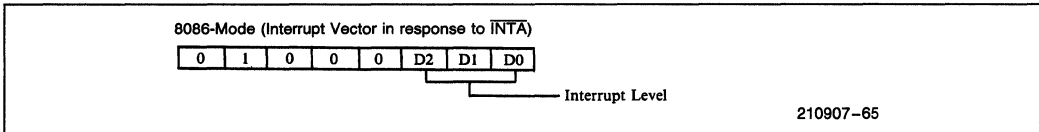
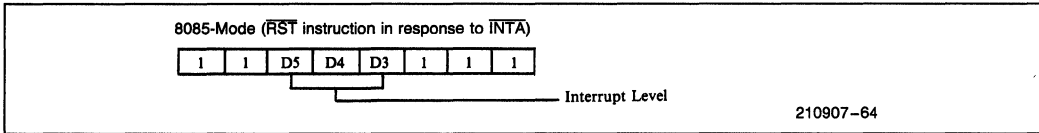




Status Register



Response to INTA



August 1984

**8256AH Multifunction Peripheral
Simplifies Microcomputer
I/O Design**

CHRISTOPHER SCOTT

Order Number: 231125-001

INTRODUCTION

A primary goal of microcomputer system design is to provide the required functionality and flexibility with the fewest number of components. The 8256AH Multi-function Peripheral is designed specifically to meet these conflicting requirements. Four of the most common microcomputer system functions, previously requiring up to four separate MSI or LSI devices, are combined into one LSI device. The 8256AH incorporates a serial asynchronous communication channel, two 8-bit parallel I/O ports, five 8-bit timer/counters and an eight level priority interrupt controller in one 40 pin package. Its flexible design allows it to directly interface to most microprocessors, including Intel's MCS-85, iAPX-86, iAPX-88, iAPX-186 and iAPX-188, and the MCS-48 and MCS-51 family of single-chip microcomputers.

This application note describes using the 8256AH to implement a Data Terminal Equipment (DTE) RS-232C serial asynchronous communication link with the control signals necessary to interface to a Bell 103/212A modem. The interface requires a total of nine interface signals. Three of these signals, TxD, RxD and CTS, are provided by the UART section of the 8256AH. The balance of the RS-232C interface signals are implemented using six of the independently programmable parallel PORT 1 lines. In addition, the application design provides an eight bit parallel I/O port with handshaking signals. The on-chip priority interrupt controller enables the RS-232C serial interface and the parallel interface to operate on an interrupt basis. The 8256AH uniquely addresses the complexities of implementing an RS-232C communications interface. By utilizing the built-in hardware and software features of the 8256AH, the design achieves flexibility with simplicity, qualities often exclusive of one another.

Previous solutions required four components to implement the same interface. Figure 1 illustrates the basic system block diagrams for the two solutions. In Figure 1a the 8251A Programmable Communications Interface provides the UART serial communications interface. The 8254 Programmable Interval Timer provides baud rate generation and other timing functions, such as time-out loops, needed for software support of an RS-232C interface. These are especially needed if the RS-232C channel is to operate in an interrupt system environment. The 8255A Programmable Peripheral Interface provides parallel I/O with one port dedicated to the RS-232C control signals. The 8259A Priority Interrupt Controller provides an eight level priority interrupt structure. This represents a total of 120 device pins compared to the single 40 pin 8256AH, and 465 mA current requirement versus a 160 mA current requirement. Figure 1b represents the 8256AH solution incorporating the four functions in one package.

In some data communication applications only three lines - ground, Transmit Data and Receive Data - are used for serial communication. An example is communication between an ASCII terminal or printer and a personal computer. These devices are usually located close to one another and in general do not require the additional control signals of the EIA RS-232C serial communications standard. In other data communications applications, this same equipment requires that the integrity of the serial communications link be constantly monitored. This enables the host system to control the data transmission at all times, whether it be a host computer or intelligence local to a communications device, such as an ASCII terminal. The need for control and monitoring of the serial line is particularly important when the communications link is over telephone lines using a modem. In a Switched Network, where a number of serial devices share the same communications line, the control signals are crucial to the system's multiplexing the single line.

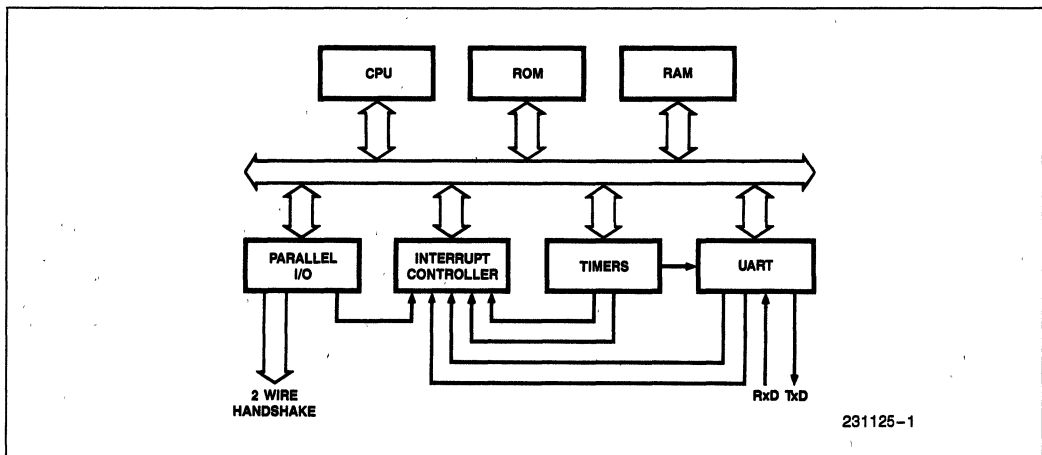


Figure 1a. System Block Diagram Without the 8256AH

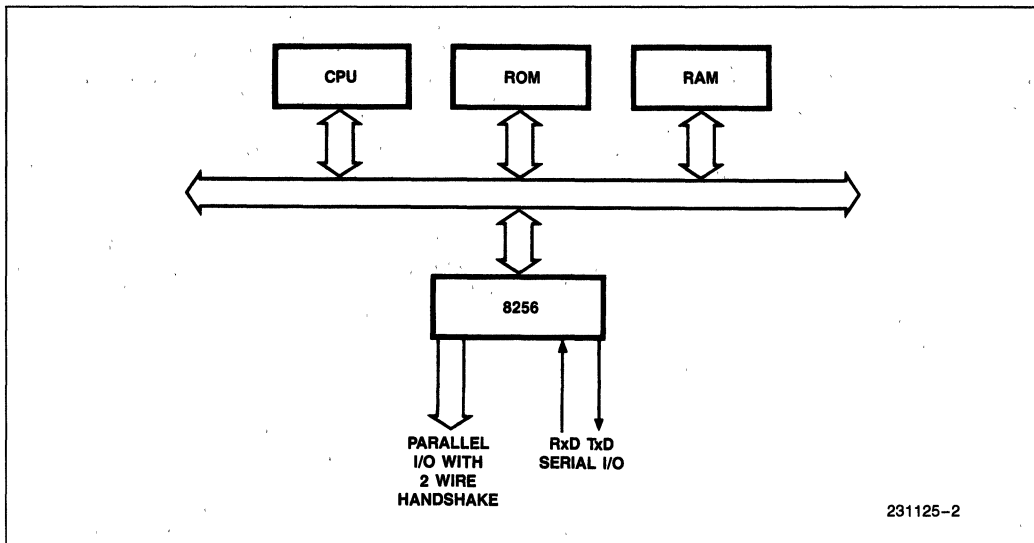


Figure 1b. System Block Diagram With the 8256AH

This Application Note assumes that the reader is familiar with the 8256AH Data Sheet and with the RS-232C communication protocol and terminology. A complete software listing is provided in Appendix A. A complete description and definition of the RS-232C interface standard may be found in the book "Data Communications: A Users Guide" by Kenneth Sherman, Reston Publishing 1981.

DESCRIPTION OF THE 8256AH

The 8256AH combines four commonly used peripheral functions into one device (see Figure 2);

1. A full-duplex, double-buffered serial asynchronous Receiver/Transmitter (UART) with an on-chip Baud Rate Generator.
2. Two 8-bit parallel I/O ports; One bit programmable, One nibble programmable.
3. Five 8-bit timer/counters; 4 can be cascaded to form 2 16-bit timer/counters
4. An 8-level priority interrupt controller.

The 8256AH uses the standard bus control signals compatible with Intel's family of peripherals and microprocessors. The microprocessor interface utilizes a multiplexed address/data bus. Four of the eight address/data lines are used to generate the register address. This enables all of the 8256AH's functionality to be contained in a 40 pin package while retaining direct register addressing.

The sixteen directly addressable internal read/write registers provide control for all of the 8256AH's various functions. Fourteen of the registers are read/write, one, the Status Register, is read only and one, the Modification Register, is write only. Three Command Registers configure the operating environment including the type of CPU, 8 or 16 bit, and system clock frequency. Command Register Three provides bit set-reset capability for control of such functions as End of Interrupt, Nested Interrupts, Interrupt Acknowledge and UART Receive Enable. The Status Register provides all information about the UART's transmitter and receiver, and the state of the interrupt (INT) output pin to the microprocessor. The Mode Register defines the configuration of the two parallel ports and the five timer/counters. The write only Modification Register is used to alter two standard functions of the receiver, start bit sampling and to enable a special indicator flag for half-duplex operation. In addition, six registers control the two parallel ports. Two registers provide for UART Transmit and Receive Buffers. Ten registers are used for timer/counter interface, and four registers provide for Priority Interrupt Controller support.

The UART section of the 8256AH features a full-duplex double-buffered transmitter and receiver with separate control registers. The internal baud rate generator provides the thirteen common sampling rates from 50 bps to 19.2 kbps. An external baud rate clock can also be used, with programmable choice of 1X, 32X or 64X sampling rates.

The two parallel I/O ports can be configured as two independent 8-bit parallel I/O ports, or as one 8-bit

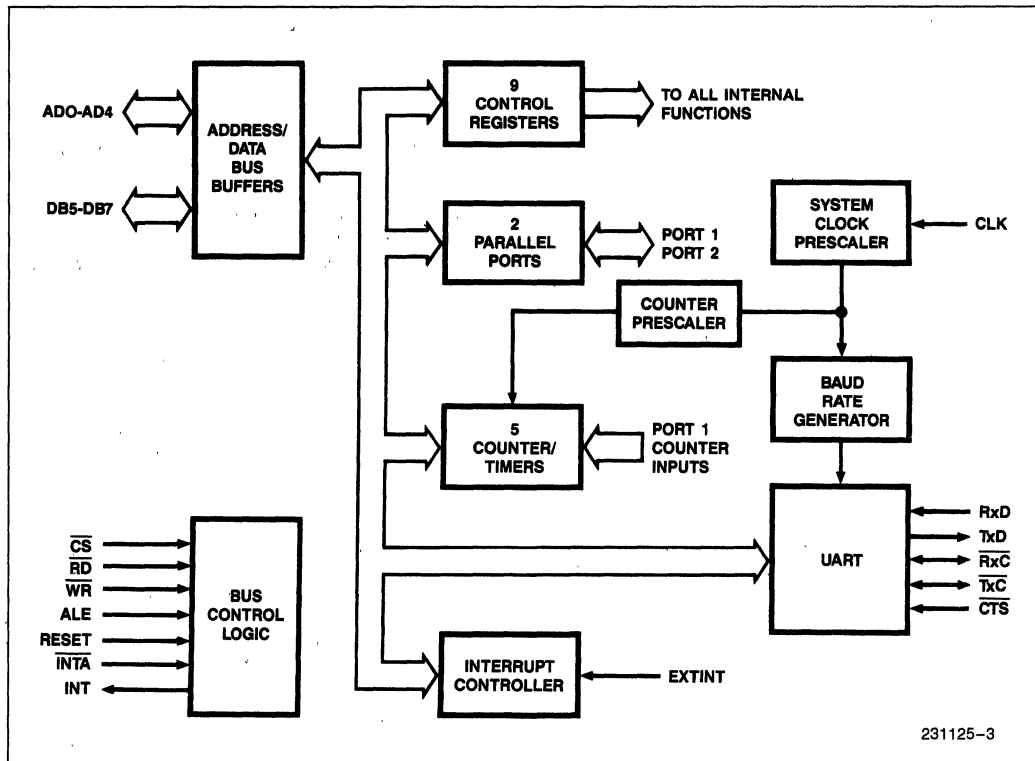


Figure 2. 8256AH Internal Block Diagram

parallel port with ACK/OBF and STB/IBF two wire handshake signals. In the latter configuration, the six remaining I/O lines may be used as either independently programmable I/O lines, or as predefined special function inputs and/or outputs, such as a second external interrupt input or timer/counter inputs.

The five 8-bit programmable timer/counters are binary presettable downcounters. In addition, an independent on-chip Baud Rate Generator is provided for the UART. The clock sources for the timers/counters may be either internal or external - via programmed parallel port pins - depending upon whether they are configured as timers or counters. Four of the timer/counters may be cascaded to form two 16-bit timer/counters. Each of the five timer/counters has its own read/write register.

The eight level priority interrupt controller has twelve possible interrupt sources. Ten of the sources are internal and two are external. One of the external interrupt sources is a fixed pin; EXTINT. The second is one of the parallel Port 1 pins which can be programmed as an external interrupt source. The twelve interrupt sources are internally mapped to the eight interrupt priority levels.

The interrupt controller may be programmed to operate in either a Normal or Nested Interrupt Mode. In Normal Mode any interrupt may interrupt any other interrupt based upon the enable/disable bits in the Interrupt Enable, or Mask, Register. In the Nested Mode only an interrupt of higher priority may interrupt one of lower priority, again based upon the bits in the Enable Register.

The 8256AH interrupt structure supports both 8085 and 8086 interrupt vectoring methods via the INTR and INTA signals. In vectored interrupt operation the 8256AH places the interrupt vector address on the data bus during the INTA sequence. In addition the 8256AH supports non-vectored interrupt interfaces, such as MCS-51 and MCS-48 systems. In non-vectored interrupt applications the host system simply reads the interrupt vector address from the Interrupt Address Register of the 8256AH. Reading the interrupt address register clears the INTA pin and acknowledges that the interrupt has been serviced. This is the functional equivalent to an INTA sequence generated by the host processor.

DESIGN DESCRIPTION

Hardware Description

Figure 3 shows a block diagram of this application's system design. The microprocessor used is an iAPX-186 with two 8256AH's for parallel and serial I/O, as well as for providing a variety of system support functions. One 8256AH is used to implement both the RS-232C modem interface and provide multiplexed parallel I/O. The system uses the Intel 957B System Monitor for control of the system hardware and software development support. The second 8256AH is used for basic serial communication between an ASCII terminal and the Intel 957B System Monitor residing in 16K bytes of EPROM. The two 8256AHs provide a total of six I/O channels - two UARTs and four parallel I/O ports.

When one of the 8256AHs is configured for the serial RS-232C interface, one of its parallel ports, Port 1 pins 2-7, provides control signals for the serial interface. Four of the RS-232C control signals (CTS, DSRs, DSR and CD) are OR'd to the EXTINT pin of the 8256AH. If any of these signals change from their defined state, an interrupt is generated to the 8256AH. The modem driver software then responds to the interrupt by reading the Port 1 register, determines the signal generating the interrupt and responds accordingly (see the software listing; INT—MOD). In addition to the RS-232C control signals, the communications software can support all of the standard UART error conditions such as framing errors, underrun, overrun and parity, if parity is enabled.

Parallel I/O With Handshaking

The remaining two Port 1 lines, not used for the RS-232C control signals, provide ACK/OBF and STB/IBF handshaking signals for parallel Port 2. In an environment which utilized the second parallel port, while implementing the above described RS-232C channel, both would operate on an interrupt basis. The interrupt software algorithm depends upon whether the parallel port is configured as input or output, and whether Nested or Normal interrupt mode is programmed. If Nested Interrupt Mode is used, the software flow would default to parallel input or output (as programmed) with Port 2 handshaking the lowest priority interrupt. The serial channel would then interrupt parallel Port 2 transmission whenever the serial channel transmitted or received a character. The RS-232C control signals, OR'd to the External Interrupt (EXTINT) pin, would have the highest interrupt controller priority. The Software Description below describes this in greater detail.

SOFTWARE DESCRIPTION

Serial RS—232C Interface

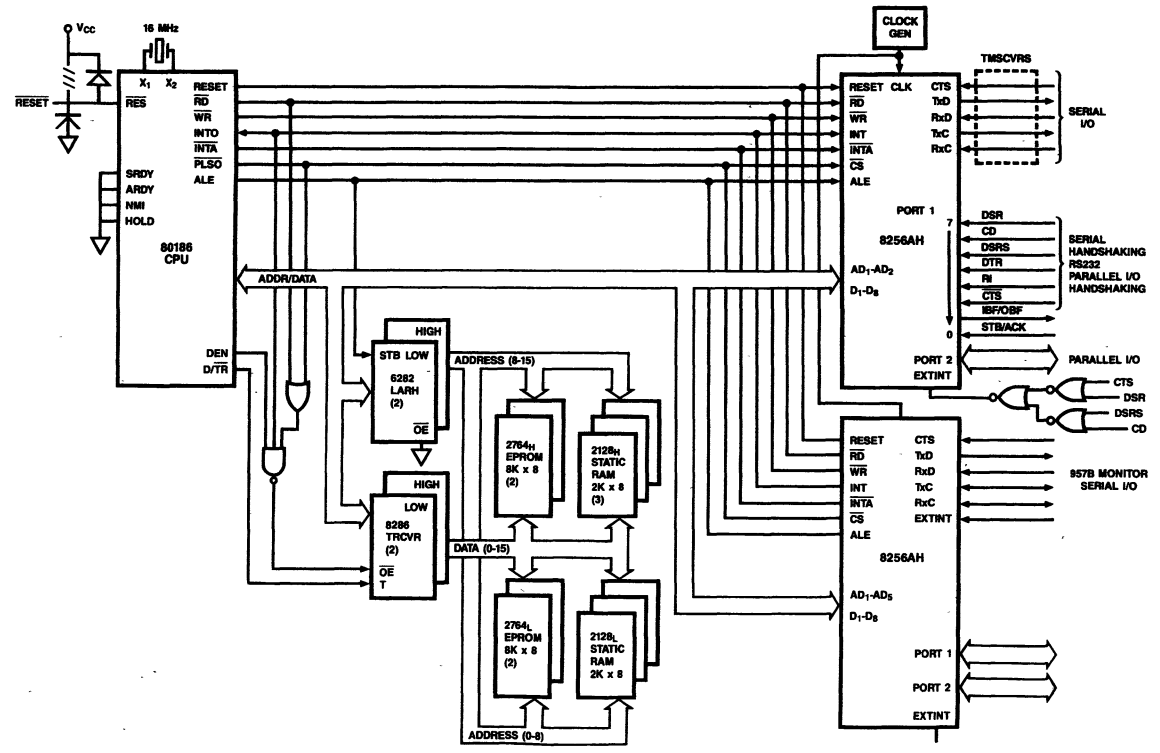
The software is written in PL/M and is broken up into four separate modules, each containing several procedures. A block diagram of the software structure is given in Figure 4. The modules are identified by the dotted boxes, and the procedures are identified by the solid boxes. Two or more procedures connected by a solid line means the procedure above calls the procedure below. The procedures without any solid lines connecting them are interrupt procedures. They are entered when the 8256AH interrupts the 80186 and vectors an indirect address to the 80186.

The Serial RS-232C Interface software uses nested interrupts. The priority of the interrupt procedures is given in Figure 5.

The priority of the interrupts is not programmable but they are logically oriented so that for this application the priority is correct. The serial receiver should have the highest priority since it could have overrun errors. Therefore the RxD request can interrupt any other interrupt service routine thus preventing any possibility of an overrun error.

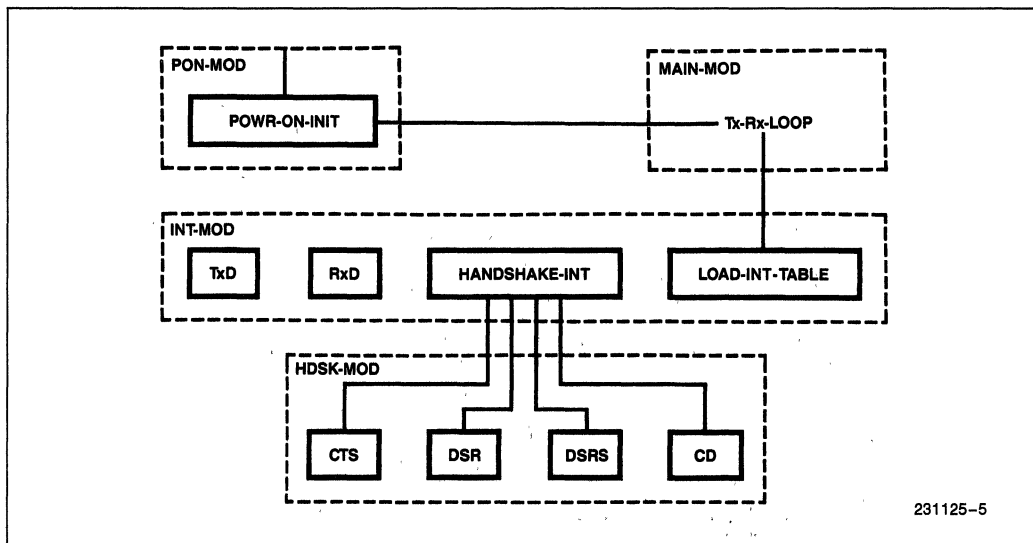
The Serial RS-232C Interface software is entered via a GO instruction from the 957B System Monitor console. The software first calls POWR—ON—INIT which initializes the 8256AH. This sets the 8256AH to 8086 Mode with parallel Port 2 in two wire handshake mode using Port 1 pin 0-1 for Port 2 handshaking. The initialization configures six of the Port 1 lines, pins 2-7, for RS-232C handshaking—input or output depending upon the specific signal tied to the pin. Figure 6 illustrates the definition of each Port 1 RS-232C handshaking line and its direction.

Both the Serial RS-232C Interface and the parallel interface with handshaking operate on an interrupt basis. Following initialization the software enters an endless loop and awaits an interrupt from one of three sources; Receive Data (RxD), Transmit Data (TxD) or the parallel interface. In the serial interface idle state, neither transmitting nor receiving data, the software is constantly responding to TxD interrupts; a result of the Transmit Buffer (TBE) and/or Transmit Register (TRE) being continually empty. When data is received by the RS-232C channel the RxD interrupt, being of higher priority, asserts its interrupt.



231125-4

Figure 3. 8256AH / 80186 Schematic



231125-5

Figure 4. Block Diagram of the 8256AH Serial RS-232C Interface Software Structure

Priority	Source
Highest 0	Not Used
1	Not Used
2	External Interrupt (EXTINT)
3	Not Used
4	RxD Interrupt
5	TxD Interrupt
6	Timer 2 or 2 & 4 (16 bit)
7	Port 2 Handshaking

Figure 5. 8256AH Interrupt Source To Priority Level Map

Port 1 Pin No.	Circuit	I/O	Abrev.	Signal Name
0			STB/ACK	Parallel Port 2
1			IBF/OBF	Handshaking Signals
2	CG	I	CTS	Clear To Send
3	CE	I	RI	Ring Indicator
4	CD	O	DTR	Data Terminal Ready
5	CI	I	DSRS	Data Signal Rate Selector
6	CF	I	RLSD (or CD)	Receive Line Signal Detector (Carrier Detect)
7	CC	I	DSR	Data Set Ready

Figure 6. Port 1 RS-232C Pin Definition

Although the parallel interface software is not implemented in the software listing of Appendix A, the algorithm for implementing multiplexed parallel and serial I/O is to input or output data on the parallel port during the relatively lengthy time required for serial communication overhead. The algorithm differs slightly during the serial channel idle state when the software responds to repetitive TxD interrupts. In this case the endless loop would detect the idle state repetitive TxD interrupts and disable the TxD interrupt for a short time while the parallel inputs or outputs data. This would require using one of the 8256AH timers to time out repetitive TxD interrupts. The timer used has to be lower in priority than the RxD interrupt to guarantee protection against overrun errors. Timer 2, or 2 and 4 cascaded if longer time delays are desired, provides the proper interrupt level as shown in Figure 5.

Figure 7 shows the Receive Data (RxD) interrupt service routine software flowchart. Since two conditions can generate an RxD Interrupt the Software first reads the Status Register and checks for the Break Detect

(DB) bit being set. If the BD bit is clear, no Break condition being present, the data byte is read, stripped to seven bits, for an ASCII character, and sent to the system console via a call to the 957B System Monitor Console Output (CO) routine. Upon return from the 957B monitor call an End Of Interrupt (EOI) is sent to the 8256AH to reset the currently served interrupt level bit in the Interrupt Service Register.

Figure 8 shows the Transmit Data (TxD) interrupt service routine software flowchart. There are three conditions which may cause a TxD Interrupt; TBE, TRE and Break-In Detect. The TxD service routine first reads the Status Register to determine if the interrupt source is the TBE (Transmit Buffer Empty), if not then the interrupt service routine returns to the MAIN—MOD loop. If TBE = 1 (true) then a data byte is read from the 957B System Monitor Console Input (CI) routine. If the data byte is an ASCII character it is written to the 8256AH Transmit Buffer. The software exists via an EOI (End Of Interrupt) command to the 8256AH then returns to the MAIN—MOD Rx—Tx—Loop.

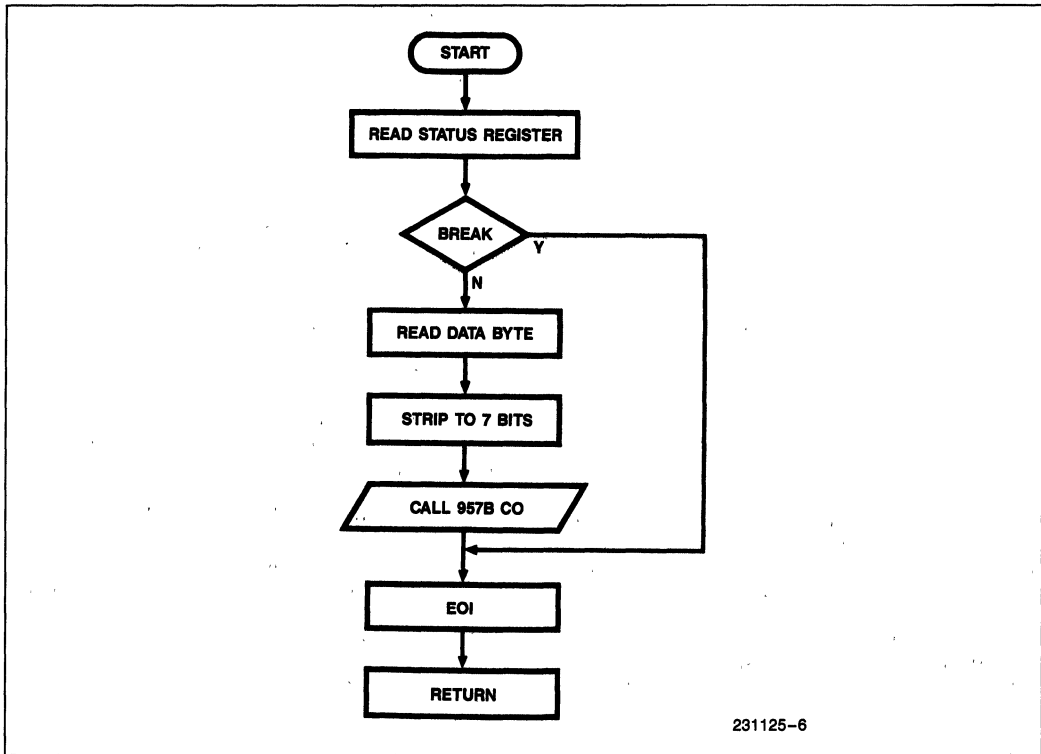


Figure 7. Receive Data Interrupt Service Routine Software Flowchart

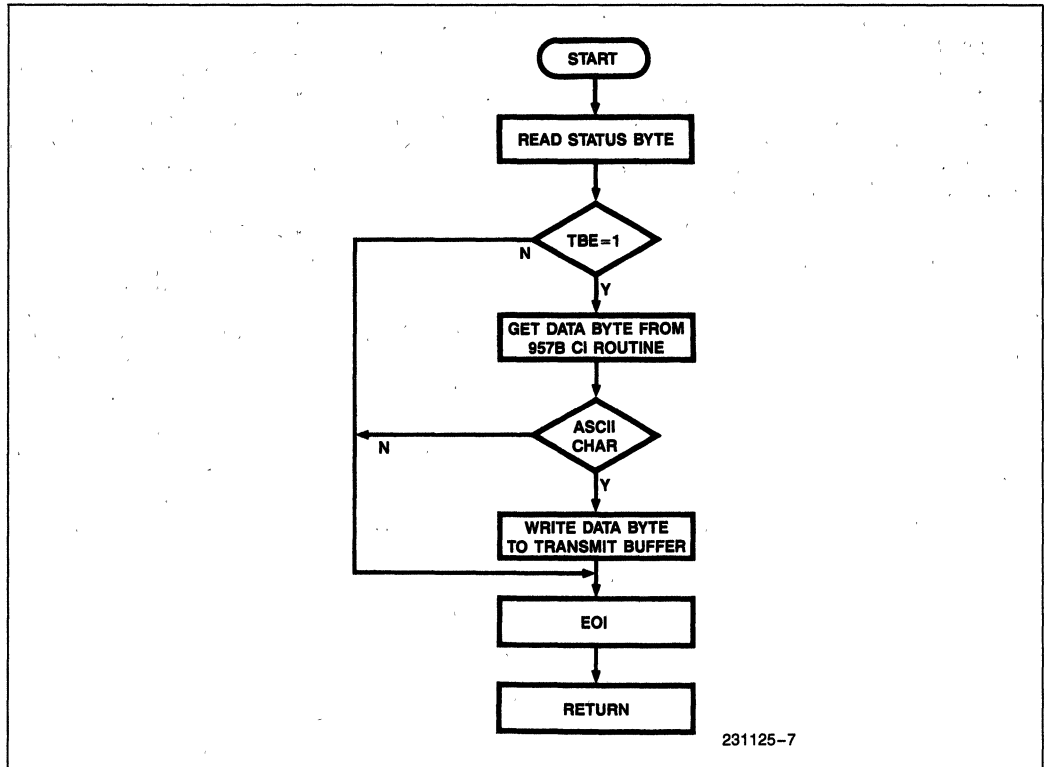


Figure 8. Transmit Data Interrupt Service Routine Software Flowchart

RS-232C Control Signals Interrupt Structure

The overall interrupt scheme is such that a change in a RS-232C handshake line causes an interrupt via the EXTINT pin on the 8256AH (see Figure 3 8256AH/80186 Schematic). The EXTINT interrupt is of higher priority than either the RxD or TxD interrupt. This enables the RS-232C handshake signals to manage the receipt or transmission of data via the nested interrupt mode of the 8256AH. The EXTINT interrupt service routine first reads the Port 1 pins 2-7 data and compares it to default state for the signal requiring service. The EXTINT interrupt service routine then calls the appropriate handshake signal service procedure as shown in the bottom module of Figure 4 Software Structure Block Diagram.

Each of the individual RS-232C control signal service procedures displays a message on the 957B monitor console device indicating the signal requiring a response. The service procedure then either initiates specific predefined actions or prompts the user with options. In a system which utilized file storage, such as a personal computer, the RS-232C software driver could

pass a flag to the communications software, rather than a message. The communications software would in turn perform the same types of action but could also protect disk buffering files which might be open at the time of the interrupt. Two examples of the RS-232C Control Signal interrupt service routines, CTS and DSRS, are described below;

If Clear To Send (CTS) changes state, the UART automatically disables the transmitter. The CTS interrupt service procedure initializes the 8256AH's internal Timer. If the timer times out before CTS goes active again an interrupt is generated, a second message is displayed at the 957B monitor console prompting the user that the CTS line remains inactive. The options available at this point are to wait again, re-initializing Timer 1, or to disconnect the RS-232C channel.

If Data Signal Rate Selector (DSRS) changes state, the software prompts the user with a message that the Data Rates of the two RS-232C channels are not the same and the user is given the option of altering the data rate. This application example was interfaced to a 103A/212 Bell modem and as such prompts the user to select between 300 or 1200 bps data rates. In the case of a

non-modem interface the routine could prompt the user for one of the thirteen standard data rates. The software then returns to the TxD/RxD software loop. The balance of the interrupt service procedures for the RS-232C handshaking signals function in a similar manner.

Depending upon the specific system design and software requirements, a variety of enhancements could be added to the system design. These could include interrupt traps that initiate specific corrective options or cascading multiple 8256AHs each with an RS-232C interfaces as described above. An example of an interrupt trap might be auto redial upon time out for lack of Carrier Detect (CD) upon initiating a communications link, or automatic disk file update when a receive buffer approaches overflow.

The ability of the 8256AH to be reprogrammed to meet the changing requirements of a system simplifies the overall system design and multiplies its capabilities. A simple reinitialization sequence could reconfigure the 8256AH as a UART with two parallel ports or utilize any of the various special functions of the parallel Port 1; e.g., an external timer input or an additional external interrupt input, etc. The reinitialization could also con-

figure the 8256AH Multifunction Peripheral for a variety of custom applications.

CONCLUSION

The functional integration of the 8256AH makes it ideal for designs which require maximum flexibility and simplicity of implementation. The implementation of the RS-232C serial channel modem interface and multiplexed parallel I/O described in this application note represent a level of efficiency in peripheral performance and design previously unavailable. The 8256AH Multifunction Peripheral represents a savings of two-thirds the board space and power required by the previous four chip solution, with the added benefit of increased system reliability. The application note demonstrates the ease of implementing the variety of I/O capabilities and system support functions of the 8256AH. The integration of four common microprocessor system functions into one VLSI device enables the designer to devote valuable resources to adding features to enhance the system design, adding performance and flexibility, and reducing the system's overhead.

APPENDIX A

SOFTWARE LISTING

PL/M-86 COMPILER MAINMOD

SERIES-III PL/M-86 V2.3 COMPILATION OF MODULE MAINMOD
 OBJECT MODULE PLACED IN : F2:56.OBJ
 COMPILER INVOKED BY: PLM86.86 : F2:56

```

/* *****
*
*      8256AH MULTIFUNCTION PERIPHERAL SIMPLIFIES
*      MICROCOMPUTER I/O DESIGN
*
*      Intel Corporation
*      3065 Bowers Avenue
*      Santa Clara, Ca. 95051
*
*      Written By      Christopher Scott
*
* *****
    
```

1 \$MOD186 DEBUG LARGE
 MAINMOD:
 DO:

```

/* ----- */
/*      8256AH Register / Value / Constant Definitions      */
/* ----- */
    
```

2	1	Declare	Lit DCL	Literally lit	'literally', 'Declare',
			True	lit	'0ffh',
			False	lit	'0h',
			Forever	lit	'while 1',
			Pcs1	lit	'B0h',
			Cmd1reg	lit	'pcs1 + 0',
			Cmd2reg	lit	'pcs1 + 2',
			Cmd3reg	lit	'pcs1 + 4',
			Modereg	lit	'pcs1 + 6',
			Port1Ctrl1Reg	lit	'pcs1 + B',
			SetIntReg	lit	'pcs1 + 0ah',
			EnIntReg	lit	'pcs1 + 0ah',
			RstIntReg	lit	'pcs1 + 0ch',
			IntAddrReg	lit	'pcs1 + 0ch',
			TxBuffReg	lit	'pcs1 + 0ah',
			RxBuffReg	lit	'pcs1 + 0ch',
			Port1Reg	lit	'pcs1 + 10h',
			Port2Reg	lit	'pcs1 + 12h',
			Timer1Reg	lit	'pcs1 + 14h',
			Timer2Reg	lit	'pcs1 + 1ah',
			Timer3Reg	lit	'pcs1 + 1ch',
			StatReg	lit	'pcs1 + 1eh',
			Intr1	lit	'pcs1 + 40h',
			Intr2	lit	'cs1 + 01h',
			Intr3	lit	'pcs1 + 10h',

231125-8

PL/M-86 COMPILER MAINMOD

```

Intr4          lit          'pcsl + 08h',

Int_Reset     lit          '88h',
SioTxEn       lit          '10h',
SioTxRdy      lit          '20h',
SioRxRdy      lit          '40h',
Break         lit          '04h',
DisIntr       lit          '00h',
StripTo7fh    lit          '7fh',
Port1_Strip   lit          '0fcH',

Cmd1          lit          '43h',
Cmd2A         lit          '07h',
Cmd2B         lit          '09h',
Cmd3C1r       lit          '7fh',
Cmd3          lit          '0aih',
Mode          lit          '00h',
EnRcvr        lit          '0c0h',

A             lit          '41h',
B             lit          '42h',
DSR           lit          '80h',
DSR_Flag      lit          '80h',
CD            lit          '40h',
CD_Flag       lit          '40h',
DSRS          lit          '20h',
DSRS_Flag     lit          '20h',
DTR           lit          '10h',
RI            lit          '08h',
CTS           lit          '04h',
CTS_Flag      lit          '04h',

(Status,
 Hndshk_Pins,
 J)           Byte,

Char          Byte          External,

Message_Ptr   Pointer;
    
```

```

/* ----- */
/*      Message Declarations      */
3  1  DCL  CTS_MSG (*) Byte Public Data ('CTS Disabled. Receive Data stopped.',
                                         OAH, ODH, 0),
        DSR_MSG (*) Byte Public Data ('DSR Disabled.', OAH, ODH, 00),
        CD_MSG  (*) Byte Public Data ('CD Disabled.', OAH, ODH, 00),
        DSRS_MSG (*) Byte Public Data ('Enter Baud Rate: A. 300 B. 1200
                                         (A/B) : ', 00),
        CTS2_MSG (*) Byte Public Data ('CTS Disabled. Receive Data stopped.',
                                         OAH, ODH, 00),
        Break_MSG (*) Byte Public Data ('Break in Receive Data.', OAH, ODH, 00);
/* ----- */
    
```

```

/* ----- */
/*      External Procedures:      */
/* ----- */
    
```

231125-9

PL/M-86 COMPILER MAINMOD

```

/*
/*      MCO:   957B Monitor Console Output Routine      */
/*
/*      MCI:   957B Monitor Console Input Routine      */
/*
/* ----- */
4  1  MCO: Procedure(Char) External;
5  2  DCL   Char      Byte;
6  2  End MCO;

7  1  MCI: Procedure Byte External;
8  2  End MCI;

/* ----- */

/* ----- */
9  1  /* Initialize 8256AH Procedure */
Init56: Procedure;

10 2  Disable;
/* Output 8256AH Init Data */
11 2  Output(Cmd1Reg)=Cmd1; /* 8086 mode, freq=1khz, 1 stop bit,
                          and 7 bit char */
12 2  Output(Cmd1Reg)=Cmd2A; /* odd parity, system clk=1.024mhz,
                          and 1200 bps */
13 2  Output(Cmd1Reg)=Cmd3C1r; /* clear cmd reg 3 */
14 2  Output(Cmd1Reg)=Cmd3; /* reset, itr ack enabled, nested
                          intr mode */
15 2  Output(Cmd1Reg)=EnRcvr; /* enable receiver */
16 2  Output(Cmd1Reg)=Mode; /* cascade timers 3&5, for the
                          receiver#timer#out timer, byte &
                          output mode */

17 2  Call Load_Int_Table;
18 2  Enable;

19 2  End Init56;
/* ----- */

/* ----- */
20 1  /* Procedure: Load Interrupt Address Vectors */
Load_Int_Table: Procedure Public;

21 2  Call Set$Interrupt(42H,EXTINT);
22 2  Call Set$Interrupt(44H,Receive_Char);
23 2  Call Set$Interrupt(45H,Transmit_Char);
24 2  Call Set$Interrupt(46H,Timer_2_4);

25 2  End Load_Int_Table;
/* ----- */

/* ----- */
/*      EXTINT Interrpt Procedure:      */
/*
/*

```

PL/M-86 COMPILER MAINMOD

```

                /*          Service routine reads the Port 1 RS232          */
                /*          handshake signals and sets the message pointer */
                /*          corresponding to the signal detected.          */
                /*          */
                /* ----- */
26  1  EXTINT: Procedure Interrupt 42H;

27  2      Enable;
28  2      HndShk_Pins=Input(Port1Reg) and Port1_Strip;
29  2      If CTS_Flag = HndShk_Pins and CTS Then
30  2          Do;
31  3          Message_Ptr=@CTS_MSG(0);
32  3          Output(Timer2Reg)=100;
33  3          End;
34  2      Else
35  2          If DSR_Flag = HndShk_Pins and DSR Then
36  2              Message_Ptr=@DSR_MSG(0);
37  2          Else
38  2              If CD_Flag = HndShk_Pins and CD Then
39  2                  Message_Ptr=@CD_MSG(0);
40  2              Else
41  2                  If DSRS_Flag = HndShk_Pins and DSRS Then
42  2                      DO;
43  3                      Message_Ptr=@DSRS_MSG(0);
44  3                      If MCI = A Then
45  3                          Output(Cmd1Reg)=Cmd2A;          /* odd parity, system clk=1.024mhz,
46  3                          and 1200 bps */
47  3                      Else
48  3                          If MCI = B Then
49  3                              Output(Cmd1Reg)=Cmd2B;      /* odd parity, system clk=1.024mhz,
50  3                              and 300 bps */
51  2                      End;
52  2                  Call Send_Msg;
53  2                  OutPut(RstIntReg)=Int_Reset;
54  2      End EXTINT;
55  2      /* ----- */

/* ----- */
/*          Procedure          Receive a character          */
/*          Receive_Char: Procedure Interrupt 44H;          */
49  1

50  2      Enable;
51  2      Status=(Input(StatReg) and SiORxRdy);
52  2      If Status AND Break Then
53  2          DO;
54  3          Message_Ptr=@Break_MSG(0);
55  3          Call Send_Msg;
56  3          End;
57  2      Else
58  2          Do;
59  3          Char=Input(RxBuffReg) and StripTo7fh;
60  3          Call MCO(Char);
61  3          End;
62  2          OutPut(RstIntReg)=Int_Reset;

63  2      End Receive_Char;

```

231125-11

PL/M-86 COMPILER MAINMOD

```

/* ----- */
/* ----- */
/* Procedure:        Write character to 8256AH UART        */
63 1 Transmit_Char: Procedure Interrupt 45H;
64 2     Status=(Input(StatReg) and SioRxRdy);
65 2     If Status and SioRxRdy Then
66 2       Char=(MCI And StripTo7FH);       /* strip to 7 bits        */
67 2       If Char >= 20H And Char <= 7FH Then   /* if char is ASCII output it */
68 2         Output(TxBuffReg)=Char;
69 2         OutPut(RstIntReg)=Int_Reset;
70 2     End Transmit_Char;
/* ----- */

/* ----- */
/* Procedure:        Write character to 856AH UART        */
71 1 Timer_2_4: Procedure Interrupt 46H;
72 2     Message_Ptr=@CTS2_MSG(0);
73 2     Call Send_Msg;
74 2     OutPut(RstIntReg)=Int_Reset;
75 2     End Timer_2_4;
/* ----- */

/* ----- */
/* Message Output Procedure        */
76 1 Send_Msg: Procedure;
77 2     DCL     Message Based Message_Ptr (1) Byte;
78 2     J=0;
79 2     Do While Message(J) <> 0;
80 3       Char=Message(J);
81 3       Call MCD(Char);
82 3       J=J+1;
83 3     End;
84 2     Return;
85 2     End Send_Msg;
/* ----- */

/* ----- */
/* Main Program Body        */
86 1     Call Init56;
87 1     Do Forever;
88 2     End;

/* ----- */

```

231125-12

PL/M-86 COMPILER MAINMOD

89 1 End MainMod;

MODULE INFORMATION:

CODE AREA SIZE	= 0235H	565D
CONSTANT AREA SIZE	= 00BEH	190D
VARIABLE AREA SIZE	= 0007H	7D
MAXIMUM STACK SIZE	= 0034H	52D
280 LINES READ		
0 PROGRAM WARNINGS		
0 PROGRAM ERRORS		

DICTIONARY SUMMARY:

31KB MEMORY AVAILABLE
6KB MEMORY USED (19%)
0KB DISK SPACE USED

END OF PL/M-86 COMPILATION

231125-13

Floppy Disk Controllers

3





8272A

SINGLE/DOUBLE DENSITY FLOPPY DISK CONTROLLER

- IBM Compatible in Both Single and Double Density Recording Formats
- Programmable Data Record Lengths: 128, 256, 512, or 1024 Bytes/Sector
- Multi-Sector and Multi-Track Transfer Capability
- Drives Up to 4 Floppy or Mini-Floppy Disks
- Controls 8", 5 1/4" and 3 1/2" Floppy Disk Drives
- Data Transfers in DMA or Non-DMA Mode
- Parallel Seek Operations on Up to Four Drives
- Compatible with all Intel and Most Other Microprocessors
- Single-Phase 8 MHz Clock
- Single +5V Power Supply (± 10%)
- Plastic 40 Pin DIP or 40 Pin CERDIP Packages

The 8272A is an LSI Floppy Disk Controller (FDC) Chip, which contains the circuitry and control functions for interfacing a processor to 4 Floppy Disk Drives. It is capable of supporting either IBM 3740 single density format (FM), or IBM System 34 Double Density format (MFM) including double sided recording. The 8272A provides control signals which simplify the design of an external phase locked loop and write precompensation circuitry. The FDC simplifies and handles most of the burdens associated with implementing a Floppy Disk Drive Interface. The 8272A is a pin-compatible upgrade to the 8272.

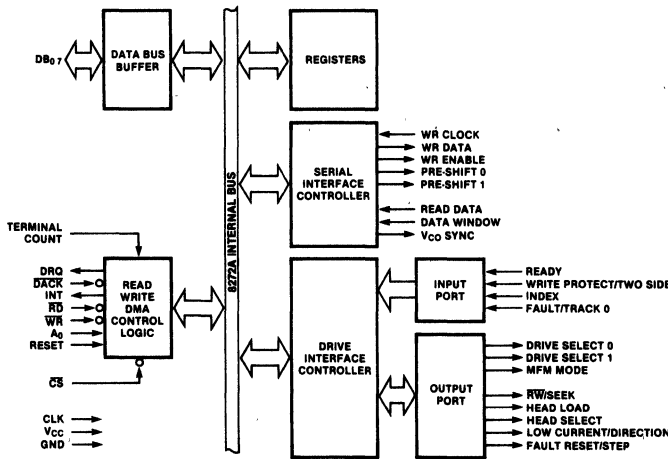


Figure 1. 8272A Internal Block Diagram

210606-1

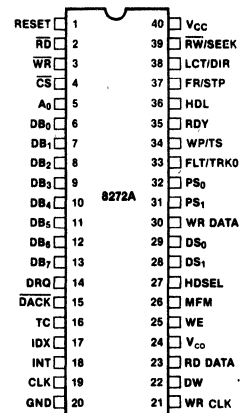


Figure 2. Pin Configuration

210606-2

Table 1. Pin Description

Symbol	Pin No.	Type	Connection To	Name and Function
RESET	1	I	μ P	RESET: Places FDC in idle state. Resets output lines to FDD to "0" (low). Does not clear the last specify command.
\overline{RD}	2	I(1)	μ P	READ: Control signal for transfer of data from FDC to Data Bus, when "0" (low).
\overline{WR}	3	I(1)	μ P	WRITE: Control signal for transfer of data to FDC via Data Bus, when "0" (low).
\overline{CS}	4	I	μ P	CHIP SELECT: IC selected when "0" (low) allowing \overline{RD} and \overline{WR} to be enabled.
A_0	5	I(1)	μ P	DATA/STATUS REGISTER SELECT: Selects Data Reg ($A_0 = 1$) or Status Reg ($A_0 = 0$) contents to be sent to Data Bus.
DB_0 - DB_7	6-13	I/O(1)	μ P	DATA BUS: Bidirectional 8-Bit Data Bus.
DRQ	14	O	DMA	DATA DMA REQUEST: DMA Request is being made by FDC when DRQ "1".(3)
\overline{DACK}	15	I	DMA	DMA ACKNOWLEDGE: DMA cycle is active when "0" (low) and Controller is performing DMA transfer.
TC	16	I	DMA	TERMINAL COUNT: Indicates the termination of a DMA transfer when "1" (high)(2).
IDX	17	I	FDD	INDEX: Indicates the beginning of a disk track.
INT	18	O	μ P	INTERRUPT: Interrupt Request Generated by FDC.
CLK	19	I		CLOCK: Single Phase 8 MHz (4 MHz for mini floppies) Squarewave Clock.
GND	20			GROUND: D.C. Power Return.
V_{CC}	40			D.C. POWER: +5V
$\overline{RW/SEEK}$	39	O	FDD	READ WRITE/SEEK: When "1" (high) Seek mode selected and when "0" (low) Read/Write mode selected.
LCT/DIR	38	O	FDD	LOW CURRENT/DIRECTION: Lowers Write current on inner tracks in Read/Write mode, determines direction head will step in Seek mode.
FR/STP	37	O	FDD	FAULT RESET/STEP: Resets fault FF in FDD in Read/Write mode, provides step pulses to move head to another cylinder in Seek mode.
HDL	36	O	FDD	HEAD LOAD: Command which causes Read/Write head in FDD to contact diskette.
RDY	35	I	FDD	READY: Indicates FDD is ready to send or receive data. Must be tied high (gated by the index pulse) for mini floppies which do not normally have a Ready line.
WP/TS	34	I	FDD	WRITE PROTECT/TWO-SIDE: Senses Write Protect status in Read/Write mode, and Two Side Media in Seek mode.
FLT/TRK0	33	I	FDD	FAULT/TRACK 0: Senses FDD fault condition in Read/Write mode and Track 0 condition in Seek mode.
PS_1, PS_0	31, 32	O	FDD	PRECOMPENSATION (PRE-SHIFT): Write precompensation status during MFM mode. Determines early, late, and normal times.
WR DATA	30	O	FDD	WRITE DATA: Serial clock and data bits to FDD.
DS_1, DS_0	28, 29	O	FDD	DRIVE SELECT: Selects FDD unit.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Connection To	Name and Function
HDSEL	27	O	FDD	HEAD SELECT: Head 1 selected when "1" (high) Head 0 selected when "0" (low).
MFM	26	O	PLL	MFM MODE: MFM mode when "1," FM mode when "0".
WE	25	O	FDD	WRITE ENABLE: Enables write data into FDD.
VCO	24	O	PLL	VCO SYNC: Inhibits VCO in PLL when "0" (low), enables VCO when "1."
RD DATA	23	I	FDD	READ DATA: Read data from FDD, containing clock and data bits.
DW	22	I	PLL	DATA WINDOW: Generated by PLL, and used to sample data from FDD.
WR CLK	21	I		WRITE CLOCK: Write data rate to FDD FM = 500 kHz, MFM = 1 MHz, with a pulse width of 250 ns for both FM and MFM. Must be enabled for all operations, both Read and Write.

NOTES:

1. Disabled when $\overline{CS} = 1$.
2. TC must be activated to terminate the Execution Phase of any command.
3. DRQ is also an input for certain test modes. It should have a 5 kΩ pull-up resistor to prevent activation.

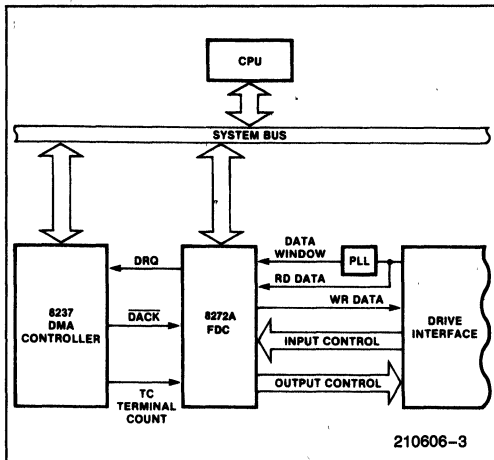


Figure 3. 8272A System Block Diagram

load a command into the FDC and all data transfers occur under control of the 8272A and DMA controller.

There are 15 separate commands which the 8272A will execute. Each of these commands require multiple 8-bit bytes to fully specify the operation which the processor wishes the FDC to perform. The following commands are available.

- | | |
|--------------------|----------------------------------|
| Read Data | Write Data |
| Read ID | Format a Track |
| Read Deleted Data | Write Deleted Data |
| Read a Track | Seek |
| Scan Equal | Recalibrate (Restore to Track 0) |
| Scan High or Equal | Sense Interrupt Status |
| Scan Low or Equal | Sense Drive Status |
| Specify | |

For more information see the Intel Application Notes AP-116 and AP-121.

DESCRIPTION

Hand-shaking signals are provided in the 8272A which make DMA operation easy to incorporate with the aid of an external DMA Controller chip, such as the 8237A. The FDC will operate in either DMA or Non-DMA mode. In the Non-DMA mode, the FDC generates interrupts to the processor for every transfer of a data byte between the CPU and the 8272A. In the DMA mode, the processor need only

FEATURES

Address mark detection circuitry is internal to the FDC which simplifies the phase locked loop and read electronics. The track stepping rate, head load time, and head unload time may be programmed by the user. The 8272A offers many additional features such as multiple sector transfers in both read and write modes with a single command, and full IBM compatibility in both single (FM) and double density (MFM) modes.

8272A ENHANCEMENTS

On the 8272A, after detecting the Index Pulse, the VCO Sync output stays low for a shorter period of time. See Figure 4a.

On the 8272 there can be a problem reading data when Gap 4A is 00 and there is no IAM. This occurs on some older floppy formats. The 8272A cures this problem by adjusting the VCO Sync timing so that it is not low during the data field. See Figure 4b.

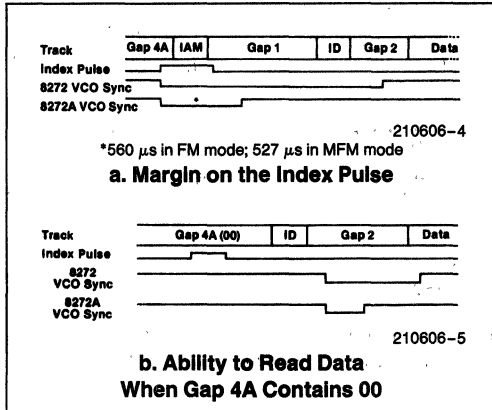


Figure 4. 8272A Enhancements over the 8272

8272A REGISTERS—CPU INTERFACE

The 8272A contains two registers which may be accessed by the main system processor; a Status Register and a Data Register. The 8-bit Main Status Register contains the status information of the FDC, and may be accessed at any time. The 8-bit Data Register (actually consists of several registers in a stack with only one register presented to the data bus at a time), stores data, commands, parameters, and FDD status information. Data bytes are read out of, or written into, the Data Register in order to program or obtain the results after execution of a command. The Status Register may only be read and is used to facilitate the transfer of data between the processor and 8272A.

The relationship between the Status/Down registers and the signals RD, WR, and A₀ is shown in Table 2.

Table 2. A₀, RD, WR Decoding for the Selection of Status/Data Register Functions.

A ₀	RD	WR	Function
0	0	1	Read Main Status Register
0	1	0	Illegal ⁽¹⁾
0	0	0	Illegal ⁽¹⁾
1	0	0	Illegal ⁽¹⁾
1	0	1	Read from Data Register
1	1	0	Write into Data Register

NOTE:

1. Design must guarantee that the 8272A is not subjected to illegal inputs.

The Main Status Register bits are defined in Table 3.

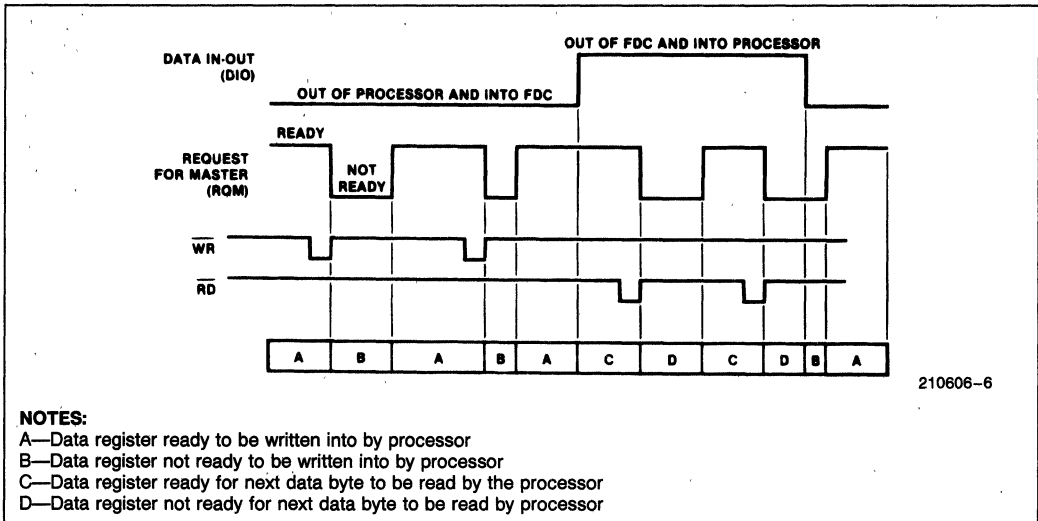
Table 3. Main Status Register Bit Description

Bit Number	Name	Symbol	Description
D ₀	FDD 0 Busy	D ₀ B	FDD number 0 is in the Seek mode.
D ₁	FDD 1 Busy	D ₁ B	FDD number 1 is in the Seek mode.
D ₂	FDD 2 Busy	D ₂ B	FDD number 2 is in the Seek mode.
D ₃	FDD 3 Busy	D ₃ B	FDD number 3 is in the Seek mode.
D ₄	FDC Busy	CB	A read or write command is in process.
D ₅	Non-DMA Mode	NDM	The FDC is in the non-DMA mode. This bit is set only during the execution phase in non-DMA mode. Transition to "0" state indicates execution phase has ended.
D ₆	Data Input/Output	DIO	Indicates direction of data transfer between FDC and Data Register. If DIO = "1" then transfer is from Data Register to the Processor. If DIO = "0", then transfer is from the Processor to Data Register.
D ₇	Request for Master	RQM	Indicates Data Register is ready to send or receive data to or from the Processor. Both bits DIO and RQM should be used to perform the handshaking functions of "ready" and "direction" to the processor.

The DIO and RQM bits in the Status Register indicate when Data is ready and in which direction data will be transferred on the Data Bus.

NOTE:

There is a 12 μs or 24μs RQM flag delay when using an 8 or 4 MHz clock respectively.



NOTES:

- A—Data register ready to be written into by processor
- B—Data register not ready to be written into by processor
- C—Data register ready for next data byte to be read by the processor
- D—Data register not ready for next data byte to be read by processor

Figure 5. Status Register Timing

The 8272A is capable of executing 15 different commands. Each command is initiated by a multi-byte transfer from the processor, and the result after execution of the command may also be a multi-byte transfer back to the processor. Because of this multi-byte interchange of information between the 8272A and the processor, it is convenient to consider each command as consisting of three phases:

Command Phase: The FDC receives all information required to perform a particular operation from the processor.

Execution Phase: The FDC performs the operation it was instructed to do.

Result Phase: After completion of the operation, status and other house-keeping information are made available to the processor.

During Command or Result Phases the Main Status Register (described in Table 3) must be read by the processor before each byte of information is written into or read from the Data Register. Bits D6 and D7 in the Main Status Register must be in a 0 and 1 state, respectively, before each byte of the command word may be written into the 8272A. Many of the commands require multiple bytes, and as a result the Main Status Register must be read prior to each byte transfer to the 8272A. On the other hand, during the Result Phase, D6 and D7 in the Main Status Register must both be 1's (D6 = 1 and D7 = 1) before reading each byte from the Data Register.

NOTE:

This reading of the Main Status Register before each byte transfer to the 8272A is required in only the Command and Result Phases, and NOT during the Execution Phase.

During the Execution Phase, the Main Status Register need not be read. If the 8272A is in the non-DMA Mode, then the receipt of each data byte (if 8272A is reading data from FDD) is indicated by an interrupt signal on pin 18 (INT = 1). The generation of a Read signal (RD = 0) will reset the interrupt as well as output the Data onto the Data Bus. For example, if the processor cannot handle Interrupts fast enough (every 13 μ s for MFM mode) then it may poll the Main Status Register and then bit D7 (RQM) functions just like the Interrupt signal. If a Write Command is in process, then the WR signal performs the reset to the Interrupt signal.

The 8272A always operates in a multi-sector transfer mode. It continues to transfer data until the TC input is active. In Non-DMA Mode, the system must supply the TC input.

If the 8272A is in the DMA Mode, no Interrupts are generated during the Execution Phase. The 8272A generates DRQ's (DMA Requests) when each byte of data is available. The DMA Controller responds to this request with both a DACK = 0 (DMA Acknowledge) and a RD = 0 (Read signal). When the DMA Acknowledge signal goes low (DACK = 0) then the DMA Request is reset (DRQ = 0). If a Write Command has been programmed then a WR signal will appear instead of RD. After the Execution Phase has been completed (Terminal Count has occurred) then an Interrupt will occur (INT = 1). This signifies the beginning of the Result Phase. When the first byte of data is read during the Result Phase, the Interrupt is automatically reset (INT = 0).

It is important to note that during the Result Phase all bytes shown in the Command Table must be read. The Read Data Command, for example has seven bytes of data in the Result Phase. All seven bytes must be read in order to successfully complete the Read Data Command. The 8272A will not accept a new command until all seven bytes have been read. Other commands may require fewer bytes to be read during the Result Phase.

The 8272A contains five Status Registers. The Main Status Register mentioned above may be read by the processor at any time. The other four Status Registers (ST0, ST1, ST2, and ST3) are only available during the Result Phase, and may be read only after successfully completing a command. The particular command which has been executed determines how many of the Status Registers will be read.

The bytes of data which are sent to the 8272A to form the Command Phase, and are read out of the 8272A in the Result Phase, must occur in the order shown in the Table 4. That is, the Command Code must be sent first and the other bytes sent in the prescribed sequence. No foreshortening of the Command or Result Phases are allowed. After the last byte of data in the Command Phase is sent to the 8272A, the Execution Phase automatically starts. In a similar fashion, when the last byte of

Table 4. 8272A Command Set

Phase	R/W	Data Bus								Remarks	
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		
READ DATA											
Command	W	MT	MFM	SK	0		0	1	1	0	Command Codes
	W	0	0	0	0		0	HDS	DS1	DS0	
	W		_____			C		_____			Sector ID Information
	W		_____			H		_____			Prior to Command
	W		_____			R		_____			Execution
	W		_____			N		_____			
	W		_____			EOT		_____			
	W		_____			GPL		_____			
Execution	W		_____			DTL		_____			Data Transfer Between the FDD and Main-System
Result	R		_____			ST 0		_____			Status Information
	R		_____			ST 1		_____			After Command
	R		_____			ST 2		_____			Execution
	R		_____			C		_____			
	R		_____			H		_____			Sector ID Information
	R		_____			R		_____			After Command
	R		_____			N		_____			Execution
READ DELETED DATA											
Command	W	MT	MFM	SK	0		1	1	0	0	Command Codes
	W	0	0	0	0		0	HDS	DS1	DS0	
	W		_____			C		_____			Sector ID Information
	W		_____			H		_____			Prior to Command
	W		_____			R		_____			Execution
	W		_____			N		_____			
	W		_____			EOT		_____			
	W		_____			GPL		_____			
Execution	W		_____			DTL		_____			Data Transfer Between the FDD and Main-System
Result	R		_____			ST 0		_____			Status Information
	R		_____			ST 1		_____			After Command
	R		_____			ST 2		_____			Execution
	R		_____			C		_____			
	R		_____			H		_____			Sector ID Information
	R		_____			R		_____			After Command
	R		_____			N		_____			Execution

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus									Remarks
		D7	D6	D5	D4	D3	D2	D1	D0		
WRITE DATA											
Command	W	MT	MFM	0	0		0	1	0	1	Command Codes
	W	0	0	0	0		0	HDS	DS1	DS0	
Execution	W					C					Sector ID Information Prior to Command Execution
	W					H					
	W					R					
	W					N					
	W					EOT					
	W					GPL					
	W					DTL					
	W										
Result	R					ST 0					Data Transfer Between the Main- System and FDD
	R					ST 1					
	R					ST 2					
	R					C					
	R					H					
	R					R					
	R					N					
	R										
WRITE DELETED DATA											
Command	W	MT	MFM	0	0		1	0	0	1	Command Codes
	W	0	0	0	0		0	HDS	DS1	DS0	
Execution	W					C					Sector ID Information Prior to Command Execution
	W					H					
	W					R					
	W					N					
	W					EOT					
	W					GPL					
	W					DTL					
	W										
Result	R					ST 0					Data Transfer Between the FDD and Main-System
	R					ST 1					
	R					ST 2					
	R					C					
	R					H					
	R					R					
	R					N					
	R										

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D7	D6	D5	D4	D3	D2	D1	D0	
READ A TRACK										
Command	W	0	MFM	SK	0	0	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W					C				Sector ID Information
	W					H				Prior to Command
	W					R				Execution
	W					N				
	W					EOT				
	W					GPL				
	W					DTL				
Execution										Data Transfer Between the FDD and Main-System. FDC Reads all of Cylinders Contents from Index Hole to EOT
Result	R					ST 0				Status Information
	R					ST 1				After Command
	R					ST 2				Execution
	R					C				
	R					H				Sector ID Information
	R					R				After Command
	R					N				Execution
READ ID										
Command	W	0	MFM	0	0	1	0	1	0	Commands
Execution	W	0	0	0	0	0	HDS	DS1	DS0	
										The First Correct ID Information on the Cylinder is Stored in Data Register
Result	R					ST 0				Status Information
	R					ST 1				After Command
	R					ST 2				Execution
	R					C				
	R					H				Sector ID Information
	R					R				During Execution
	R					N				Phase

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks	
		D7	D6	D5	D4	D3	D2	D1	D0		
FORMAT A TRACK											
Command	W	0	MFM	0	0	1	1	0	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		
	W					N					Bytes/Sector Sectors/Cylinder Gap 3 Filler Byte
	W					SC					
	W					GPL					
	W					D					
W											
Execution										FDC Formats an Entire Cylinder	
Result	R					ST 0				Status Information After Command Execution	
	R					ST 1					
	R					ST 2					
	R					C					
	R					H					In This Case, the ID Information has no Meaning
	R					R					
	R					N					

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks		
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀			
SCAN EQUAL												
Command	W	MT	MFM	SK	1		0	0	0	1	Command Codes	
	W	0	0	0	0		0	HDS	DS1	DS0		
	W					C						Sector ID Information Prior to Command Execution
	W					H						
	W					R						
	W					N						
	W					EOT						
	W					GPL						
	W					STP						
	Execution											
Result	R					ST 0					Status Information After Command Execution	
	R					ST 1						
	R					ST 2						
	R					C						
	R					H						Sector ID Information After Command Execution
	R					R						
	R					N						
SCAN LOW OR EQUAL												
Command	W	MT	MFM	SK	1		1	0	0	1	Command Codes	
	W	0	0	0	0		0	HDS	DS1	DS0		
	W					C						Sector ID Information Prior to Command Execution
	W					H						
	W					R						
	W					N						
	W					EOT						
	W					GPL						
	W					STP						
	Execution											
Result	R					ST 0					Status Information After Command Execution	
	R					ST 1						
	R					ST 2						
	R					C						
	R					H						Sector ID Information After Command Execution
	R					R						
	R					N						

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks	
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		
SCAN HIGH OR EQUAL											
Command	W	MT	MFM	SK	1		1	1	0	1	Command Codes
	W	0	0	0	0		0	HDS	DS1	DS0	
	W					C					Sector ID Information Prior to Command Execution
	W					H					
	W					R					
	W					N					
	W					EOT					
	W					GPL					
	W					STP					
Execution											Data Compared Between the FDD and Main-System
Result	R					ST 0					Status Information After Command Execution
	R					ST 1					
	R					ST 2					
	R					C					
	R					H					
	R					R					
	R					N					
RECALIBRATE											
Command	W	0	0	0	0		0	1	1	1	Command Codes
Execution	W	0	0	0	0		0	0	DS1	DS0	Head Retracted to Track 0
SENSE INTERRUPT STATUS											
Command	W	0	0	0	0		1	0	0	0	Command Codes
Result	R					ST 0					Status Information at the End of Each Seek Operation About the FDC
	R					PCN					

Table 4. 8272A Command Set (Continued)

Phase	R/W	Data Bus								Remarks
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
SPECIFY										
Command	W	0	0	0	0	0	0	1	1	Command Codes
	W	—	SRT	→		←		HUT	—	
	W	—	HLT	→		→			ND	
SENSE DRIVE STATUS										
Command	W	0	0	0	0	0	1	0	0	Command Codes
Result	R	0	0	0	0	0	HDS	DS1	DS0	
		_____			ST 3	_____				Status Information about FDD
SEEK										
Command	W	0	0	0	0	1	1	1	1	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	_____			NCN	_____				
Execution										Head is Positioned Over Proper Cylinder on Diskette
INVALID										
Command	W	_____ Invalid Codes _____								Invalid Command Codes (NoOp—FDC Goes Into Standby State) ST 0 = 80 (16)
Result	R	_____			ST 0	_____				

Table 5. Command Mnemonics

Symbol	Name	Description
A ₀	Address Line 0	A ₀ controls selection of Main Status Register (A ₀ = 0) or Data Register (A ₀ = 1).
C	Cylinder Number	C stands for the current selected Cylinder track number 0 through 76 of the medium.
D	Data	D stands for the data pattern which is going to be written into a Sector.
D ₇ -D ₀	Data Bus	8-bit Data bus where D ₇ is the most significant bit, and D ₀ is the least significant bit.
DS0, DS1	Drive Select	DS stands for a selected drive number 0 or 1.
DTL	Data Length	When N is defined as 00, DTL stands for the data length which users are going to read out or write into the Sector.
EOT	End of Track	EOT stands for the final Sector number of a Cylinder.
GPL	Gap Length	GPL stands for the length of Gap 3 (spacing between Sectors excluding VCO Sync Field).
H	Head Address	H stands for head number 0 or 1, as specified in ID field.
HDS	Head Select	HDS stands for a selected head number 0 or 1 (H = HDS in all command words).
HLT	Head Load Time	HLT stands for the head load time in the FDD (2 to 254 ms in 2 ms increments).
HUT	Head Unload Time	HUT stands for the head unload time after a read or write operation has occurred (16 to 240 ms in 16 ms increments).
MFM	FM or MFM Mode	If MF is low, FM mode is selected and if it is high, MFM mode is selected.
MT	Multi-Track	If MT is high, a multi-track operation is to be performed (a cylinder under both HD0 and HD1 will be read or written).
N	Number	N stands for the number of data bytes written in a Sector.
NCN	New Cylinder Number	NCN stands for a new Cylinder number, which is going to be reached as a result of the Seek operation. Desired position of Head.
ND	Non-DMA Mode	ND stands for operation in the Non-DMA Mode.
PCN	Present Cylinder Number	PCN stands for the Cylinder number at the completion of SENSE INTERRUPT STATUS Command. Position of Head at present time.
R	Record	R stands for the Sector number, which will be read or written.
R/W	Read/Write	R/W stands for either Read (R) or Write (W) signal.
SC	Sector	SC indicates the number of Sectors per Cylinder.
SK	Skip	SK stands for Skip Deleted Data Address Mark.
SRT	Step Rate Time	SRT stands for the Stepping Rate for the FDD (1 to 16 ms in 1 ms increments). The same Stepping Rate applies to all drives (F = 1 ms, E = 2 ms, etc.).

Table 5. Command Mnemonics (Continued)

Symbol	Name	Description
ST 0 ST 1 ST 2 ST 3	Status 0 Status 1 Status 2 Status 3	ST 0-3 stand for one of four registers which store the status information after a command has been executed. This information is available during the result phase after command execution. These registers should not be confused with the main status register (selected by $A_0 = 0$). ST 0-3 may be read only after a command has been executed and contain information relevant to that particular command.
STP		During a Scan operation, if STP = 1, the data in contiguous sectors is compared byte by byte with data sent from the processor (or DMA), and if STP = 2, then alternate sectors are read and compared.

data is read out in the Result Phase, the command is automatically ended and the 8272A is ready for a new command. A command may be aborted by simply sending a Terminal Count signal to pin 16 (TC=1). This is a convenient means of ensuring that the processor may always get the 8272A's attention even if the disk system hangs up in an abnormal manner.

POLLING FEATURE OF THE 8272A

After power-up RESET, the Drive Select Lines DS0 and DS1 will automatically go into a polling mode. In between commands (and between step pulses in the SEEK command) the 8272A polls all four FDDs looking for a change in the Ready line from any of the drives. If the Ready line changes state (usually due to a door opening or closing) then the 8272A will generate an interrupt. When Status Register 0 (ST0) is read (after Sense Interrupt Status is issued), Not Ready (NR) will be indicated. The polling of the Ready line by the 8272A occurs continuously between instructions, thus notifying the processor which drives are on or off line. Approximate scan timing is shown in Table 6.

Table 6. Scan Timing

DS1	DS0	Approximate Scan Timing
0	0	220 μ s
0	1	220 μ s
1	0	220 μ s
1	1	440 μ s

COMMAND DESCRIPTIONS

During the Command Phase, the Main Status Register must be polled by the CPU before each byte is

written into the Data Register. The DIO (DB6) and RQM (BD7) bits in the Main Status Register must be in the "0" and "1" states respectively, before each byte of the command may be written into the 8272A. The beginning of the execution phase for any of these commands will cause DIO and RQM to switch to "1" and "0" states respectively.

READ DATA

A set of nine (9) byte words are required to place the FDC into the Read Data Mode. After the Read Data command has been issued the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify Command), and begins reading ID Address Marks and ID fields. When the current sector number ("R") stored in the ID Register (IDR) compares with the sector number read off the diskette, then the FDC outputs data (from the data field) byte-by-byte to the main system via the data bus.

After completion of the read operation from the current sector, the Sector Number is incremented by one, and the data from the next sector is read and output on the data bus. This continuous function is called a "Multi-Sector Read Operation." The Read Data Command must be terminated by the receipt of a Terminal Count signal. Upon receipt of this signal, the FDC stops outputting data to the processor, but will continue to read data from the current sector, check CRC (Cyclic Redundancy Count) bytes, and then at the end of the sector terminate the Read Data Command.

The amount of data which can be handled with a single command to the FDC depends upon MT (multi-track), MFM (MFM/FM), and N (Number of Bytes/Sector). Table 7 on the next page shows the Transfer Capacity.

Table 7. Transfer Capacity

Multi-Track MT	MFM/FM MFM	Bytes/Sector N	Maximum Transfer Capacity (Bytes/Sector)(Number of Sectors)	Final Sector Read from Diskette
0	0	00	(128) (26) = 3,328	26 at Side 0
0	1	01	(256) (26) = 6,656	or 26 at Side 1
1	0	00	(128) (52) = 6,656	26 at Side 1
1	1	01	(256) (52) = 13,312	
0	0	01	(256) (15) = 3,840	15 at Side 0
0	1	02	(512) (15) = 7,680	or 15 at Side 1
1	0	01	(256) (30) = 7,680	15 at Side 1
1	1	02	(512) (30) = 15,360	
0	0	02	(512) (8) = 4,096	8 at Side 0
0	1	03	(1024) (8) = 8,192	or 8 at Side 1
1	0	02	(512) (16) = 8,192	8 at Side 1
1	1	03	(1024) (16) = 16,384	

The "multi-track" function (MT) allows the FDC to read data from both sides of the diskette. For a particular cylinder, data will be transferred starting at Sector 1, Side 0 and completing at Sector L, Side 1 (Sector L = last sector on the side).

NOTE:

This function pertains to only one cylinder (the same track) on each side of the diskette.

When N = 0, then DTL defines the data length which the FDC must treat as a sector. If DTL is smaller than the actual data length in a Sector, the data beyond DTL in the Sector is not sent to the Data Bus. The FDC reads (internally) the complete Sector performing the CRC check, and depending upon the manner of command termination, may perform a Multi-Sector Read Operation. When N is non-zero, then DTL has no meaning and should be set to 0FFH.

At the completion of the Read Data Command, the head is not unloaded until after Head Unload Time Interval (specified in the Specify Command) has elapsed. If the processor issues another command before the head unloads then the head settling time may be saved between subsequent reads. This time out is particularly valuable when a diskette is copied from one drive to another.

If the FDC detects the Index Hole twice without finding the right sector, (indicated in "R"), then the FDC sets the ND (No Data) flag in Status Register 1 to a 1 (high), and terminates the Read Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

After reading the ID and Data Fields in each sector, the FDC checks the CRC bytes. If a read error is detected (incorrect CRC in ID field), the FDC sets the DE (Data Error) flag in Status Register 1 to a 1 (high), and if a CRC error occurs in the Data Field the FDC also sets the DD (Data Error in Data Field) flag in Status Register 2 to a 1 (high), and terminates the Read Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

If the FDC reads a Deleted Data Address Mark off the diskette, and the SK bit (bit D5 in the first Command Word) is not set (SK = 0), then the FDC sets the CM (Control Mark) flag in Status Register 2 to a 1 (high), and terminates the Read Data Command, after reading all the data in the Sector. If SK = 1, the FDC skips the sector with the Deleted Data Address Mark and reads the next sector.

During disk data transfers between the FDC and the processor, via the data bus, the FDC must be serviced by the processor every 27 μ s in the FM Mode, and every 13 μ s in the MFM Mode, or the FDC sets the OR (Over Run) flag in Status Register 1 to a 1 (high), and terminates the Read Data Command.

If the processor terminates a read (or write) operation in the FDC, then the ID Information in the Result Phase is dependent upon the state of the MT bit and EOT byte. Table 5 shows the values for C, H, R, and N, when the processor terminates the Command.

Table 8. ID Information When Processor Terminates Command

MT	EOT	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	C + 1	NC	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	NC	R = 01	NC
1	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	NC	LSB	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	LSB	R = 01	NC

NOTES:

1. NC (No Change): The same value as the one at the beginning of command execution.
2. LSB (Least Significant Bit): The least significant bit of H is complemented.

WRITE DATA

A set of nine (9) bytes are required to set the FDC into the Write Data mode. After the Write Data command has been issued the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify Command), and begins reading ID Fields. When the current sector number ("R"), stored in the ID Register (IDR) compares with the sector number read off the diskette, then the FDC takes data from the processor byte-by-byte via the data bus, and outputs it to the FDD.

After writing data into the current sector, the Sector Number stored in "R" is incremented by one, and the next data field is written into. The FDC continues this "Multi-Sector Write Operation" until the issu-

ance of a Terminal Count signal. If a Terminal Count signal is sent to the FDC it continues writing into the current sector to complete the data field. If the Terminal Count signal is received while a data field is being written then the remainder of the data field is filled with 00 (zeros).

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (incorrect CRC) in one of the ID Fields, it sets the DE (Data Error) flag of Status Register 1 to a 1 (high), and terminates the Write Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

The Write Command operates in much the same manner as the Read Command. The following items

are the same; refer to the Read Data Command for details:

- Transfer Capacity
- EN (End of Cylinder) Flag
- ND (No Data) Flag
- Head Unload Time Interval
- ID Information when the processor terminates command (see Table 2)
- Definition of DTL when $N = 0$ and when $N \neq 0$

In the Write Data mode, data transfers between the processor and FDC must occur every $31 \mu\text{s}$ in the FM mode, and every $15 \mu\text{s}$ in the MFM mode. If the time interval between data transfers is longer than this then the FDC sets the OR (Over Run) flag in Status Register 1 to a 1 (high), and terminates the Write Data Command.

For mini-floppies, multiple track writes are usually not permitted. This is because of the turn-off time of the erase head coils—the head switches tracks before the erase head turns off. Therefore the system should typically wait 1.3 ms before attempting to step or change sides.

WRITE DELETED DATA

This command is the same as the Write Data Command except a Deleted Data Address Mark is written at the beginning of the Data Field instead of the normal Data Address Mark.

READ DELETED DATA

This command is the same as the Read Data Command except that when the FDC detects a Data Address Mark at the beginning of a Data Field (and $SK = 0$ (low)), it will read all the data in the sector and set the CM flag in Status Register 2 to a 1 (high), and then terminate the command. If $SK = 1$, then the FDC skips the sector with the Data Address Mark and reads the next sector.

READ A TRACK

This command is similar to READ DATA Command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering the INDEX HOLE, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID or DATA CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the value stored in the IDR, and sets the ND flag of Status Register 1 to

a 1 (high) if there is no comparison. Multi-track or skip operations are not allowed with this command.

This command terminates when EOT number of sectors have been read. If the FDC does not find an ID Address Mark on the diskette after it encounters the INDEX HOLE for the second time, then it sets the MA (missing address mark) flag in Status Register 1 to a 1 (high), and terminates the command. (Status Register 0 has bits 7 and 6 set to 0 and 1 respectively.)

READ ID

The READ ID Command is used to give the present position of the recording head. The FDC stores the values from the first ID Field it is able to read. If no proper ID Address Mark is found on the diskette, before the INDEX HOLE is encountered for the second time then the MA (Missing Address Mark) flag in Status Register 1 is set to a 1 (high), and if no data is found then the ND (No Data) flag is also set in Status Register 1 to a 1 (high) and the command is terminated.

FORMAT A TRACK

The Format Command allows an entire track to be formatted. After the INDEX HOLE is detected, Data is written on the Diskette: Gaps, Address Marks, ID Fields and Data Fields, all per the IBM System 34 (Double Density) or System 3740 (Single Density) Format are recorded. The particular format which will be written is controlled by the values programmed into N (number of bytes/sector), SC (sectors/cylinder), GPL (Gap Length), and D (Data Pattern) which are supplied by the processor during the Command Phase. The Data Field is filled with the Byte of data stored in D. The ID Field for each sector is supplied by the processor; that is, four data requests per sector are made by the FDC for C (Cylinder Number), H(Head Number), R(Sector Number) and N(Number of Bytes/Sector). This allows the diskette to be formatted with nonsequential sector numbers, if desired.

After formatting each sector, the processor must send new values for C, H, R, and N to the 8272A for each sector on the track. The contents of the R Register is incremented by one after each sector is formatted, thus, the R register contains a value of $R + 1$ when it is read during the Result Phase. This incrementing and formatting continues for the whole track until the FDC encounters the INDEX HOLE for the second time, whereupon it terminates the command.

If a FAULT signal is received from the FDD at the end of a write operation, then the FDC sets the EC flag of Status Register 0 to a 1 (high), and terminates the command after setting bits 7 and 6 of Status Register 0 to 0 and 1 respectively. Also the loss of a READY signal at the beginning of a com-

mand execution phase causes command termination.

Table 9 shows the relationship between N, SC, and GPL for various sector sizes:

Table 9. Sector Size Relationships

Format	Bytes/ Sector	8" Floppy				Bytes/ Sector	5¼" Floppy				Bytes/ Sector	3½" Mini Floppy			
		N	SC	GPL(1)	GPL(2)		N	SC	GPL(1)	GPL(2)		N	SC	GPL(1)	GPL(2)
FM Mode	128	00	1A	07	1B	128	00	12	07	09	128	0	0F	07	1B
	256	01	0F	0E	2A	128	00	10	10	19	—	—	—	—	
	512	02	08	1B	3A	256	01	08	18	30	256	1	09	0F	2A
	1024	03	04	47	8A	512	02	04	46	87	512	2	05	1B	3A
	2048	04	02	C8	FF	1024	03	02	C8	FF	—	—	—	—	—
	4096	05	01	C8	FF	2048	04	01	C8	FF	—	—	—	—	—
MPM Mode	256	01	1A	0E	36	256	01	12	0A	0C	256	1	0F	CE	36
	512	02	0F	1B	54	256	01	10	20	32	—	—	—	—	
	1024	03	08	35	74	512	02	08	2A	50	512	2	09	1B	54
	2048	04	04	99	FF	1024	03	04	80	F0	1024	3	05	35	74
	4096	05	02	C8	FF	2048	04	02	C8	FF	—	—	—	—	—
	8192	06	01	C8	FF	4096	05	01	C8	FF	—	—	—	—	—

NOTES:

1. Suggested values of GPL in Read or Write Commands to avoid splice point between data field and ID field of contiguous sections.
2. Suggested values of GPL in format command.

SCAN COMMANDS

The SCAN Commands allow data which is being read from the diskette to be compared against data which is being supplied from the main system (Processor in NON-DMA mode, and DMA Controller in DMA mode). The FDC compares the data on a byte-by-byte basis, and looks for a sector of data which meets the conditions of $D_{FDD} = D_{Processor}$, $D_{FDD} \leq D_{Processor}$, or $D_{FDD} \geq D_{Processor}$. Ones complement arithmetic is used for comparison (FF = largest number, 00 = smallest number). After a whole

sector of data is compared, if the conditions are not met, the sector number is incremented ($R + STP \rightarrow R$), and the scan operation is continued. The scan operation continues until one of the following conditions occur; the conditions for scan are met (equal, low, or high), the last sector on the track is reached (EOT), or the terminal count signal is received.

If the conditions for scan are met then the FDC sets the SH (Scan Hit) flag of Status Register 2 to a 1 (high), and terminates the Scan Command. If the

Table 10. Scan Status Codes

Command	Status Register 2		Comments
	Bit 2 = SN	Bit 3 = SH	
Scan Equal	0	1	$D_{FDD} = D_{Processor}$
	1	0	$D_{FDD} \neq D_{Processor}$
Scan Low or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} < D_{Processor}$
	1	0	$D_{FDD} > D_{Processor}$
Scan High or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} > D_{Processor}$
	1	0	$D_{FDD} < D_{Processor}$

conditions for scan are not met between the starting sector (as specified by R) and the last sector on the cylinder (EOT), then the FDC sets the SN (Scan Not Satisfied) flag of Status Register 2 to a 1 (high), and terminates the Scan Command. The receipt of a TERMINAL COUNT signal from the Processor or DMA Controller during the scan operation will cause the FDC to complete the comparison of the particular byte which is in process, and then to terminate the command. Table 10 shows the status of bits SH and SN under various conditions of SCAN.

If the FDC encounters a Deleted Data Address Mark on one of the sectors (and SK = 0), then it regards the sector as the last sector on the cylinder, sets CM (Control Mark) flag of Status Register 2 to a 1 (high) and terminates the command. If SK = 1, the FDC skips the sector with the Deleted Address Mark, and reads the next sector. In the second case (SK = 1), the FDC sets the CM (Control Mark) flag of Status Register 2 to a 1 (high) in order to show that a Deleted Sector had been encountered.

When either the STP (contiguous sectors STP = 01, or alternate sectors STP = 02 sectors are read) or the MT (Multi-Track) are programmed, it is necessary to remember that the last sector on the track must be read. For example, if STP = 02, MT = 0, the sectors are numbered sequentially 1 through 26, and we start the Scan Command at sector 21; the following will happen. Sectors 21, 23, and 25 will be read, then the next sector (26) will be skipped and the Index Hole will be encountered before the EOT value of 26 can be read. This will result in an abnormal termination of the command. If the EOT had been set at 25 or the scanning started at sector 20, then the Scan Command would be completed in a normal manner.

During the Scan Command data is supplied by either the processor or DMA Controller for comparison against the data read from the diskette. In order to avoid having the OR (Over Run) flag set in Status Register 1, it is necessary to have the data available in less than 27 μ s (FM Mode) or 13 μ s (MFM Mode). If an Overrun occurs the FDC terminates the command.

SEEK

The read/write within the FDD is moved from cylinder to cylinder under control of the Seek Command. The FDC compares the PCN (Present Cylinder Number) which is the current head position with the NCN

(New Cylinder Number), and performs the following operation if there is a difference:

PCN < NCN: Direction signal to FDD set to a 1 (high), and Step Pulses are issued. (Step In.)

PCN > NCN: Direction signal to FDD set to a 0 (low), and Step Pulses are issued. (Step Out.)

The rate at which Step Pulses are issued is controlled by SRT (Stepping Rate Time) in the SPECIFY Command. After each Step Pulse is issued NCN is compared against PCN, and when NCN = PCN, then the SE (Seek End) flag is set in Status Register 0 to a 1 (high), and the command is terminated.

During the Command Phase of the Seek operation the FDC is in the FDC BUSY state, but during the Execution Phase it is in the NON BUSY state. While the FDC is in the NON BUSY state, another Seek Command may be issued, and in this manner parallel seek operations may be done on up to 4 Drives at once.

If an FDD is in a NOT READY state at the beginning of the command execution phase or during the seek operation, then the NR (NOT READY) flag is set in Status Register 0 to a 1 (high), and the command is terminated.

Note that the 8272A Read and Write Commands do not have implied Seeks. Any R/W command should be preceded by: 1) Seek Command; 2) Sense Interrupt Status; and 3) Read ID.

RECALIBRATE

This command causes the read/write head within the FDD to retract to the Track 0 position. The FDC clears the contents of the PCN counter, and checks the status of the Track 0 signal from the FDD. As long as the Track 0 signal is low, the Direction signal remains 1 (high) and Step Pulses are issued. When the Track 0 signal goes high, the SE (SEEK END) flag in Status Register 0 is set to a 1 (high) and the command is terminated. If the Track 0 signal is still low after 77 Step Pulses have been issued, the FDC sets the SE (SEEK END) and EC (EQUIPMENT CHECK) flags of Status Register 0 to both 1s (highs), and terminates the command.

The ability to overlap RECALIBRATE Commands to multiple FDDs, and the loss of the READY signal, as described in the SEEK Command, also applies to the RECALIBRATE Command.

SENSE INTERRUPT STATUS

An Interrupt signal is generated by the FDC for one of the following reasons:

- 1) Upon entering the Result Phase of:
 - a) Read Data Command
 - b) Read a Track Command
 - c) Read ID Command
 - d) Read Deleted Data Command
 - e) Write Data Command
 - f) Format a Cylinder Command
 - g) Write Deleted Data Command
 - h) Scan Commands
- 2) Ready Line of FDD changes state
- 3) End of Seek or Recalibrate Command
- 4) During Execution Phase in the NON-DMA Mode

Interrupts caused by reasons 1 and 4 above occur during normal command operations and are easily discernible by the processor. However, interrupts caused by reasons 2 and 3 above may be uniquely identified with the aid of the Sense Interrupt Status Command. This command when issued resets the interrupt signal and via bits 5, 6, and 7 of Status Register 0 identifies the cause of the interrupt.

Neither the Seek or Recalibrate Command have a Result Phase. Therefore, it is mandatory to use the Sense Interrupt Status Command after these commands to effectively terminate them and to provide verification of the head position (PCN).

Table 11. Seek, Interrupt Codes

Seek End Bit 5	Interrupt Code		Cause
	Bit 6	Bit 7	
0	1	1	Ready Line Changed State, Either Polarity
1	0	0	Normal Termination of Seek or Recalibrate Command
1	1	0	Abnormal Termination of Seek or Recalibrate Command

SPECIFY

The Specify Command sets the initial values for each of the three internal timers. The HUT (Head Unload Time) defines the time from the end of the Execution Phase of one of the Read/Write Com-

mands to the head unload state. This timer is programmable from 16 to 240 ms in increments of 16 ms (01 = 16 ms, 02 = 32 ms OF = 240 ms). The SRT (Step Rate Time) defines the time interval between adjacent step pulses. This timer is programmable from 1 to 16 ms in increments of 1 ms (F = 1 ms, E = 2 ms, D = 3 ms, etc.). The HLT (Head Load Time) defines the time between when the Head Load signal goes high and when the Read/Write operation starts. This timer is programmable from 2 to 254 ms in increments of 2 ms (01 = 2 ms, 02 = 4 ms, 03 = 6 ms FE = 254 ms).

The step rate should be programmed 1 ms longer than the minimum time required by the drive.

The time intervals mentioned above are a direct function of the clock (CLK on pin 19). Times indicated above are for an 8 MHz clock, if the clock was reduced to 4 MHz (mini-floppy application) then all time intervals are increased by a factor of 2.

The choice of DMA or NON-DMA operation is made by the ND (NON-DMA) bit. When this bit is high (ND = 1) the NON-DMA mode is selected, and when ND = 0 the DMA mode is selected.

SENSE DRIVE STATUS

This command may be used by the processor whenever it wishes to obtain the status of the FDDs. Status Register 3 contains the Drive Status information.

INVALID

If an invalid command is sent to the FDC (a command not defined above), then the FDC will terminate the command. No interrupt is generated by the 8272A during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both high ("1") indicating to the processor that the 8272A is in the Result Phase and the contents of Status Register 0 (ST0) must be read. When the processor reads Status Register 0 it will find an 80H indicating an invalid command was received.

A Sense Interrupt Status Command must be sent after a Seek or Recalibrate interrupt, otherwise the FDC will consider the next command to be an Invalid Command.

In some applications the user may wish to use this command as a No-Op command, to place the FDC in a standby or no operation state.

Table 12. Status Registers

Bit			Description
No.	Name	Symbol	
STATUS REGISTER 0			
D ₇	Interrupt Code	IC	D ₇ = 0 and D ₆ = 0 Normal Termination of Command, (NT). Command was completed and properly executed.
D ₆			D ₇ = 0 and D ₆ = 1 Abnormal Termination of Command, (AT). Execution of Command was started, but was not successfully completed.
			D ₇ = 1 and D ₆ = 0 Invalid Command issue, (IC). Command which was issued was never started.
			D ₇ = 1 and D ₆ = 1 Abnormal Termination because during command execution the ready signal from FDD changed state.
D ₅	Seek End	SE	When the FDC completes the SEEK Command, this flag is set to 1 (high).
D ₄	Equipment Check	EC	If a fault Signal is received from the FDD, or if the Track 0 Signal fails to occur after 77 Step Pulses (Recalibrate Command) then this flag is set.
D ₃	Not Ready	NR	When the FDD is in the not-ready state and a read or write command is issued, this flag is set. If a read or write command is issued to Side 1 of a single sided drive, then this flag is set.
D ₂	Head Address	HD	This flag is used to indicate the state of the head at Interrupt.
D ₁	Unit Select 1	US 1	These flags are used to indicate a Drive Unit Number at Interrupt.
D ₀	Unit Select 0	US 0	
STATUS REGISTER 1			
D ₇	End of Cylinder	EN	When the FDC tries to access a Sector beyond the final Sector of a Cylinder, this flag is set.
D ₆			Not used. This bit is always 0 (low).
D ₅	Data Error	DE	When the FDC detects a CRC error in either the ID field or the data field, this flag is set.
D ₄	Over Run	OR	If the FDC is not serviced by the main-systems during data transfers, within a certain time interval, this flag is set.
D ₃			Not used. This bit always 0 (low).
D ₂	No Data	ND	During execution of READ DATA, WRITE DELETED DATA or SCAN Command, if the FDC cannot find the Sector specified in the IDR Register, this flag is set.
			During executing the READ ID Command, if the FDC cannot read the ID field without an error, then this flag is set.
			During the execution of the READ A Cylinder Command, if the starting sector cannot be found, then this flag is set.

Table 12. Status Register (Continued)

Bit			Description
No.	Name	Symbol	
STATUS REGISTER 1 (Continued)			
D ₁	Not Writable	NW	During execution of WRITE DATA, WRITE DELETED DATA or Format A Cylinder Command, if the FDC detects a write protect signal from the FDD, then this flag is set.
D ₀	Missing Address Mark	MA	If the FDC cannot detect the ID Address Mark after encountering the index hole twice, then this flag is set.
			If the FDC cannot detect the Data Address Mark or Deleted Data Address Mark, this flag is set. Also at the same time, the MD (Missing Address Mark in Data Field) of Status Register 2 is set.
STATUS REGISTER 2			
D ₇			Not used. This bit is always 0 (low).
D ₆	Control Mark	CM	During executing the READ DATA or SCAN Command, if the FDC encounters a Sector which contains a Deleted Data Address Mark, this flag is set.
D ₅	Data Error in Data Field	DD	If the FDC detects a CRC error in the data field then this flag is set.
D ₄	Wrong Cylinder	WC	This bit is related with the ND bit, and when the contents of C on the medium is different from that stored in the IDR, this flag is set.
D ₃	Scan Equal Hit	SH	During execution, the SCAN Command, if the condition of "equal" is satisfied, this flag is set.
D ₂	Scan Not Satisfied	SN	During executing the SCAN Command, if the FDC cannot find a Sector on the cylinder which meets the condition, then this flag is set.
D ₁	Bad Cylinder	BC	This bit is related with the ND bit, and when the content of C on the medium is different from that stored in the IDR and the content of C is FF, then this flag is set.
D ₀	Missing Address Mark in Data Field	MD	When data is read from the medium, if the FDC cannot find a Data Address Mark or Deleted Data Address Mark, then this flag is set.
STATUS REGISTER 3			
D ₇	Fault	FT	This bit is used to indicate the status of the Fault signal from the FDD.
D ₆	Write Protected	WP	This bit is used to indicate the status of the Write Protected signal from the FDD.
D ₅	Ready	RDY	This bit is used to indicate the status of the Ready signal from the FDD.
D ₄	Track 0	T0	This bit is used to indicate the status of the Track 0 signal from the FDD.
D ₃	Two Side	TS	This bit is used to indicate the status of the Two Side signal from the FDD.
D ₂	Head Address	HD	This bit is used to indicate the status of Side Select signal to the FDD.
D ₁	Unit Select 1	US 1	This bit is used to indicate the status of the Unit Select 1 signal to the FDD.
D ₀	Unit Select 0	US 0	This bit is used to indicate the status of the Unit Select 0 signal to the FDD.

ABSOLUTE MAXIMUM RATINGS*

Operating Temperature0°C to +70°C
Storage Temperature-40°C to +125°C
All Output Voltages-0.5 to +7V
All Input Voltages-0.5 to +7V
Supply Voltage V_{CC}-0.5 to +7V
Power Dissipation1 Watt

* $T_A = 25^\circ\text{C}$

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

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } +70^\circ\text{C}$, $V_{CC} = +5V \pm 10\%$

Symbol	Parameter	Limits		Unit	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.0 \text{ mA}$
V_{OH}	Output High Voltage	2.4	V_{CC}	V	$I_{OH} = -400 \mu\text{A}$
I_{CC}	V_{CC} Supply Current		120	mA	
I_{IL}	Input Load Current (All Input Pins)		10 -10	μA μA	$V_{IN} = V_{CC}$ $V_{IN} = 0V$
I_{LOH}	High Level Output Leakage Current		10	μA	$V_{OUT} = V_{CC}$
I_{OFL}	Output Float Leakage Current		± 10	μA	$0.45C \leq V_{OUT} \leq V_{CC}$

CAPACITANCE $T_A = 25^\circ\text{C}$, $f_c = 1 \text{ MHz}$, $V_{CC} = 0V$

Symbol	Parameter	Limits		Unit	Test Conditions
		Min	Max		
$C_{IN(\phi)}$	Clock Input Capacitance		20	pF	All Pins Except Pin Under Test Tied to AC Ground
C_{IN}	Input Capacitance		10	pF	
$C_{I/O}$	Input/Output Capacitance		20	pF	

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } +70^\circ\text{C}$, $V_{CC} = +5.0\text{V} \pm 10\%$

Symbol	Parameter	Typ ⁽¹⁾	Min	Max	Unit	Notes
CLOCK TIMING						
t_{CY}	Clock Period		120	500	ns	(Note 5)
t_{CH}	Clock High Period		40		ns	(Note 4, 5)
t_{RST}	Reset Width		14		t_{CY}	
READ CYCLE						
t_{AR}	Select Setup to $\overline{RD} \downarrow$		0		ns	
t_{RA}	Select Hold from $\overline{RD} \uparrow$		0		ns	
t_{RR}	\overline{RD} Pulse Width		250		ns	
t_{RD}	Data Delay from $\overline{RD} \downarrow$			200	ns	
t_{DF}	Output Float Delay		20	100	ns	
WRITE CYCLE						
t_{AW}	Select Setup to $\overline{WR} \downarrow$		0		ns	
t_{WA}	Select Hold from $\overline{WR} \uparrow$		0		ns	
t_{WW}	\overline{WR} Pulse Width		250		ns	
t_{DW}	Data Setup to $\overline{WR} \uparrow$		150		ns	
t_{WD}	Data Hold from $\overline{WR} \uparrow$		5		ns	
INTERRUPTS						
t_{RI}	INT Delay from $\overline{RD} \uparrow$			500	ns	(Note 6)
t_{WI}	INT Delay from $\overline{WR} \uparrow$			500	ns	(Note 6)
DMA						
t_{RQCY}	DRQ Cycle Period		13		μs	(Note 6)
t_{AKRQ}	$\overline{DACK} \downarrow$ to DRQ \downarrow			200	ns	
t_{RQR}	DRQ \uparrow to $\overline{RD} \downarrow$		800		ns	(Note 6)
t_{RQW}	DRQ \uparrow to $\overline{WR} \downarrow$		250		ns	(Note 6)
t_{RQRW}	DRQ \uparrow to $\overline{RD} \uparrow$ or $\overline{WR} \uparrow$			12	μs	(Note 6)

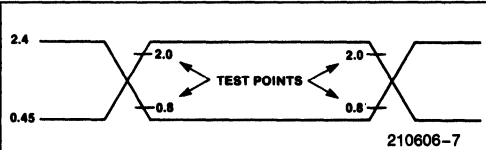
A.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$, $V_{CC} = +5.0\text{V} \pm 10\%$ (Continued)

Symbol	Parameter	Typ ⁽¹⁾	Min	Max	Unit	Notes
FDD INTERFACE						
t_{WCY}	WCK Cycle Time	2 or 4 1 or 2			μs	MFM = 0 MFM = 1 (Note 2)
t_{WCH}	WCK High Time	250	80	350	ns	
t_{CP}	Pre-Shift Delay from WCK \uparrow		20	100	ns	
t_{CD}	WDA Delay from WCK \uparrow		20	100	ns	
t_{WDD}	Write Data Width		$t_{WCH} - 50$		ns	
t_{WE}	WE \uparrow to WCK \uparrow or WE \downarrow to WCK \downarrow Delay		20	100	ns	
t_{WWCY}	Window Cycle Time	2 1			μs	MM = 0 MFM = 1
t_{WRD}	Window Setup to RDD \uparrow		15		ns	
t_{RDW}	Window Hold from RDD \downarrow		15		ns	
t_{RDD}	RDD Active Time (HIGH)		40		ns	
FDD SEEK/DIRECTION/STEP						
t_{US}	US _{0,1} Setup to \overline{RW} /SEEK \uparrow		12		μs	(Note 6)
t_{SU}	US _{0,1} Hold after \overline{RW} /SEEK \downarrow		15		μs	(Note 6)
t_{SD}	\overline{RW} /SEEK Setup to LCT/DIR		7		μs	(Note 6)
t_{DS}	\overline{RW} /SEEK Hold from LCT/DIR		30		μs	(Note 6)
t_{DST}	LCT/DIR Setup to FR/STEP \uparrow		1		μs	(Note 6)
t_{STD}	LCT/DIR Hold from FR/STEP \downarrow		24		μs	(Note 6)
t_{STU}	DS _{2,1} Hold from FR/Step \downarrow		5		μs	(Note 6)
t_{STP}	STEP Active Time (High)	5			μs	(Note 6)
t_{SC}	STEP Cycle Time		33		μs	(Note 3, 6)
t_{FR}	FAULT RESET Active Time (High)		8	10	μs	(Note 6)
t_{IDX}	INDEX Pulse Width	10			t_{CY}	
t_{TC}	Terminal Count Width		1		t_{CY}	

NOTES:

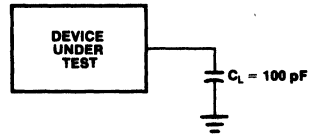
1. Typical values for $T_A = 25^\circ\text{C}$ and nominal supply voltage.
2. The former values are used for standard floppy and the latter values are used for mini-floppies.
3. $t_{SC} = 33 \mu\text{s}$ min. is for different drive units. In the case of same unit, t_{SC} can be ranged from 1 ms to 16 ms with 8 MHz clock period, and 2 ms to 32 ms with 4 MHz clock, under software control.
4. From 2.0V to +2.0V.
5. At 4 MHz, the clock duty cycle may range from 16% to 76%. Using an 8 MHz clock the duty cycle can range from 32% to 52%. Duty cycle is defined as: D.C. = $100 (t_{CH} \div t_{CY})$ with typical rise and fall times of 5 ns.
6. The specified values listed are for an 8 MHz clock period. Multiply timings by 2 when using a 4 MHz clock period.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

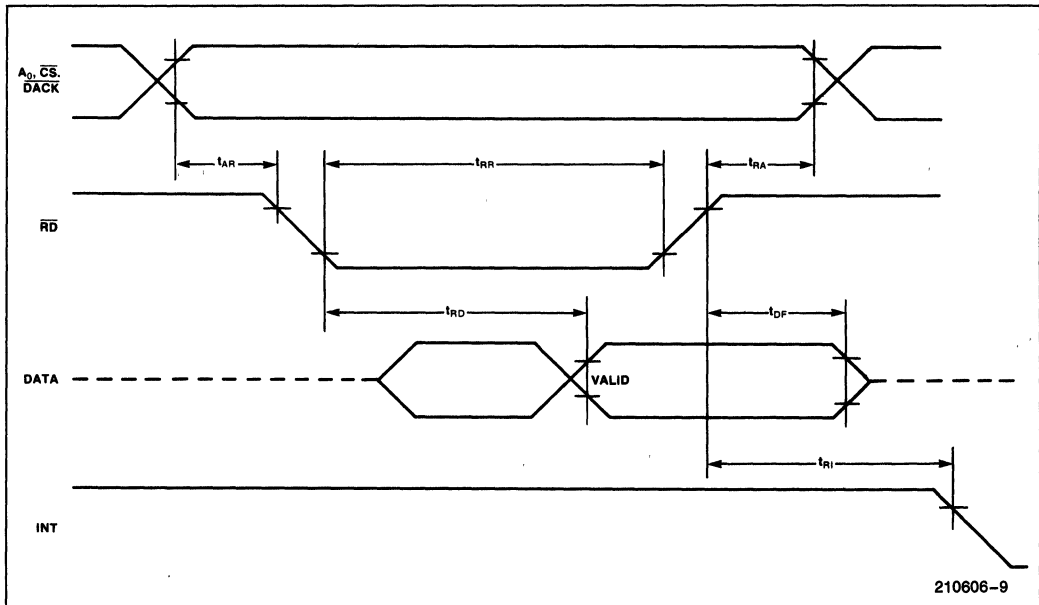


210606-8

$C_L = 100 \text{ pF}$
 C_L Includes Jig Capacitance

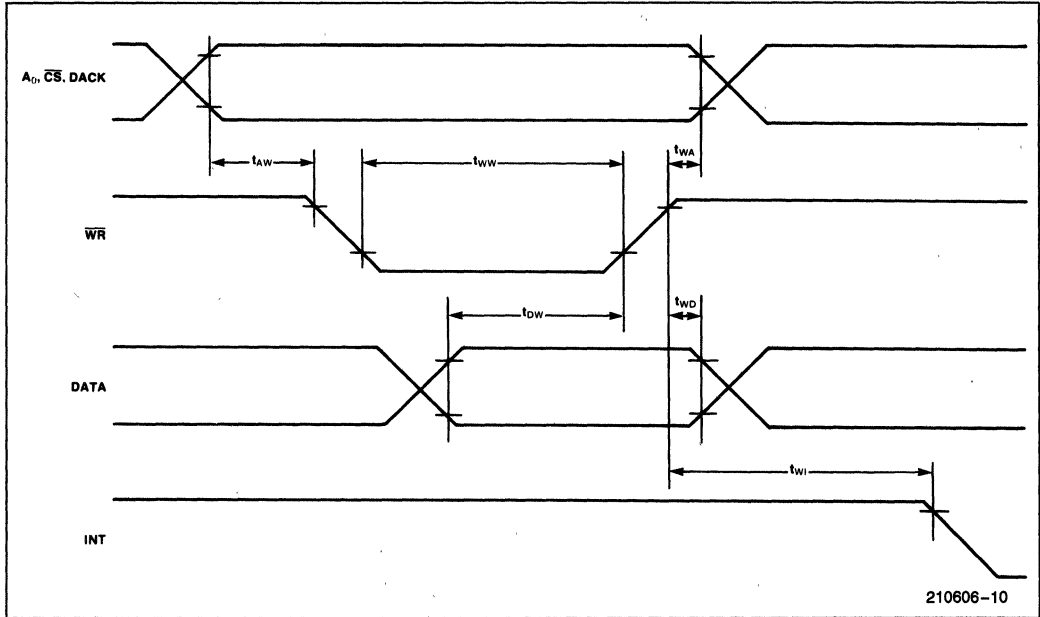
WAVEFORMS

PROCESSOR READ OPERATION

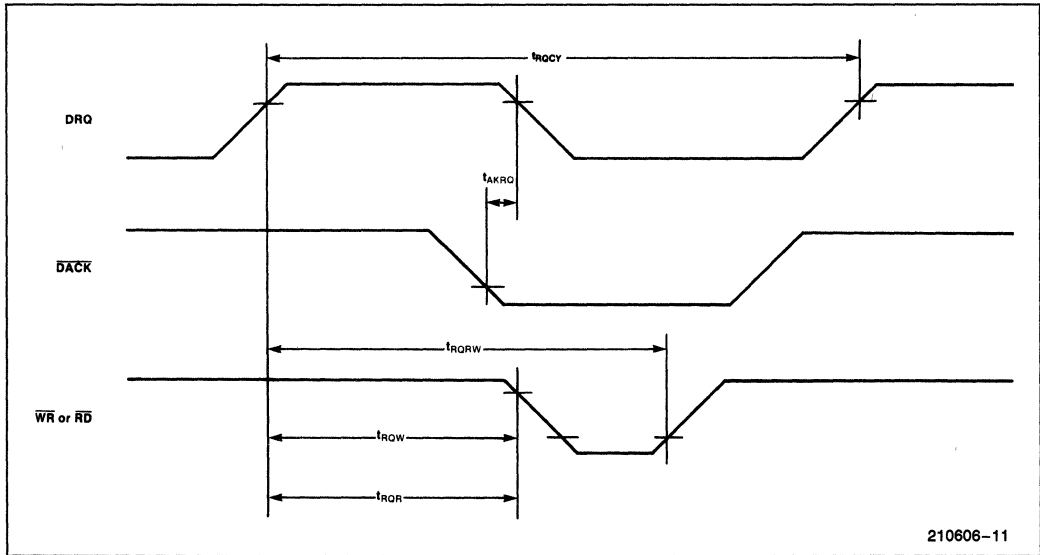


WAVEFORMS (Continued)

PROCESSOR WRITE OPERATION

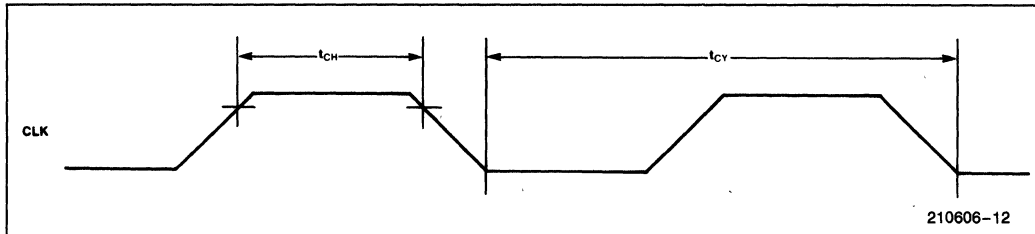


DMA OPERATION



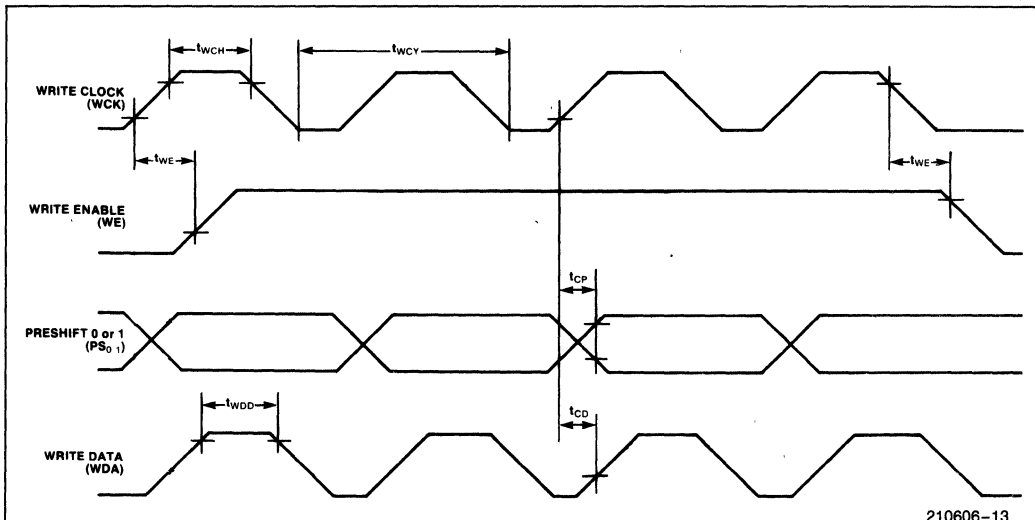
WAVEFORMS (Continued)

CLOCK TIMING



210606-12

FDD WRITE OPERATION

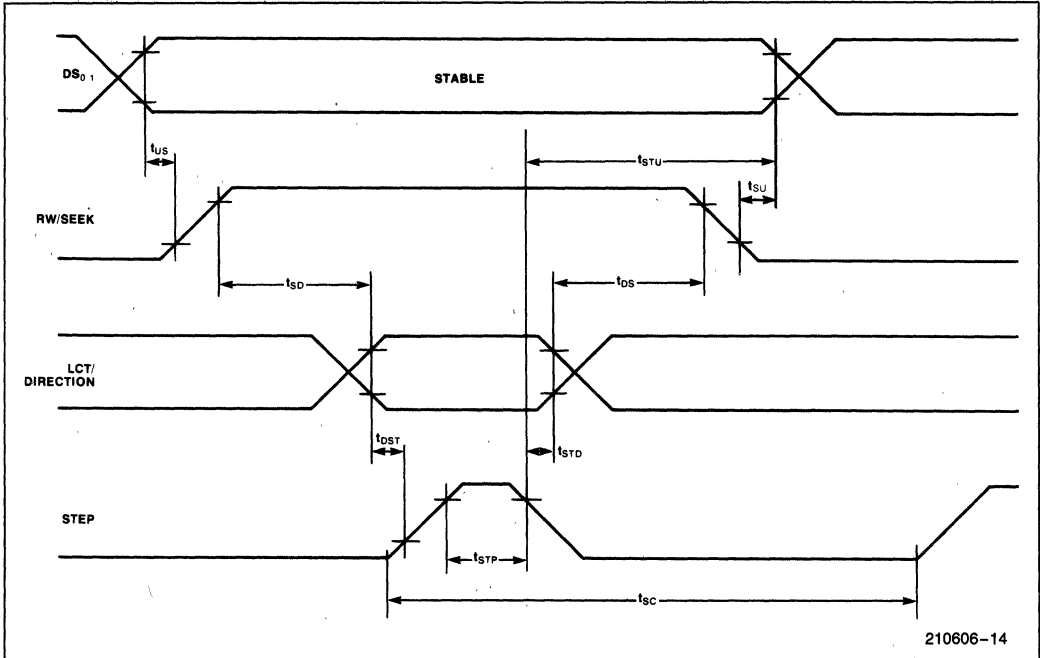


210606-13

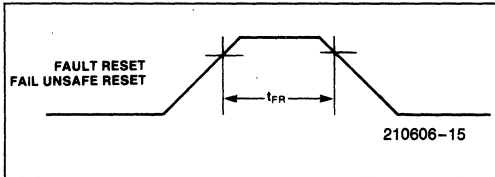
	Preshift 0	Preshift 1
Normal	0	0
Late	0	1
Early	1	0
Invalid	1	1

WAVEFORMS (Continued)

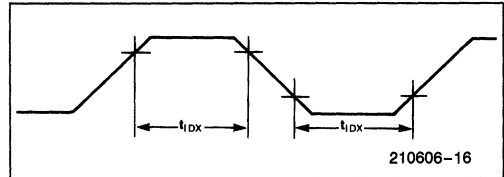
SEEK OPERATION



FLT RESET

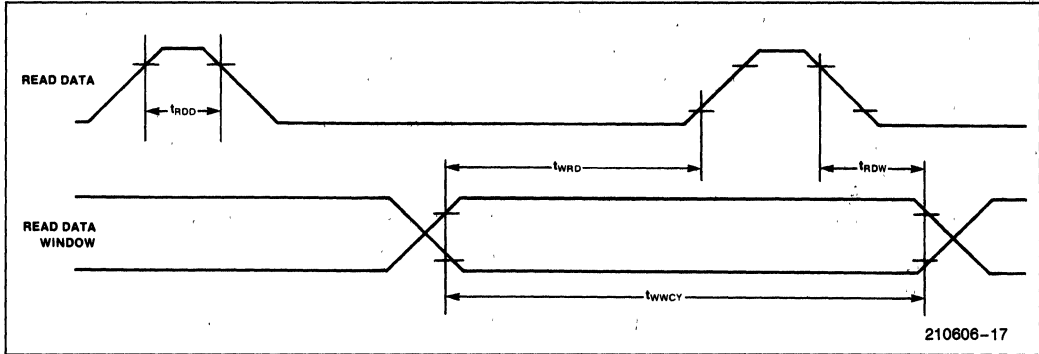


INDEX

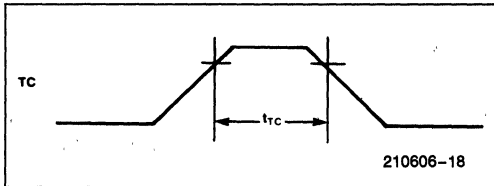


WAVEFORMS (Continued)

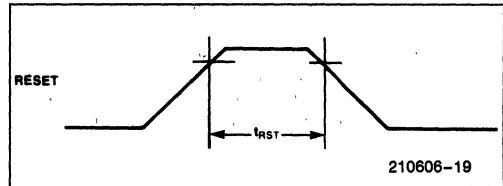
FDD READ OPERATION



TERMINAL COUNT



RESET



82072

CHMOS HIGH INTEGRATION FLOPPY DISK CONTROLLER

- Integrated Analog Data Separator with Software Selectable Data Rates (250K, 300K, 500K, 1M Bit/Sec-MFM Mode)
- 16 Byte FIFO with Programmable Threshold
- High Speed Processor Interface
 - 16 MHz iAPX 386— 1 Wait State
 - 12.5 MHz iAPX 286—1 Wait State
 - 8 MHz iAPX 286— 0 Wait State
- Programmable Internal Write Precompensation Delays
- Programmable Drive Motor On/Off Delays
- Addresses Up to 256 Tracks Directly, Supports Unlimited Tracks
- CHMOS and POWERDOWN MODE for Low Power/Portable Applications
- Implied Seek with Read/Write Disk Commands
- Supports IBM Single and Double Density Recording Formats
- Software Compatible with 8272A
- On-Chip Crystal Oscillator
- Controls 8", 5¼" and 3½" Floppy Disk Drives
- Plastic 40 Pin DIP or 44 Pin PLCC Packages

(See Packaging Spec. Order #231369)

The 82072 CHMOS high integration floppy disk controller solves the many complex disk drive and microprocessor interface issues that exist today, while maintaining software compatibility with the industry standard 8272A. Features include a sophisticated on-chip analog phase lock loop with software selectable data rates, write precompensation delay, and motor on/off delays to simplify the disk drive interface. System interfacing is enhanced with the addition of a FIFO which allows a more flexible system to be designed. Its CHMOS technology and a powerdown mode that drops the current requirement into the hundred microamp range make the 82072 ideal for low power or portable applications.

The standard 82072 supports a maximum data rate of 500 Kbits per second. The 82072-1 supports data rates up to 1 Mbit per second.

The 82072 is fabricated on Intel's advanced CHMOS III technology for minimal power consumption and is available in a plastic 40 pin DIP or a plastic 44-leaded chip carrier (PLCC) package.

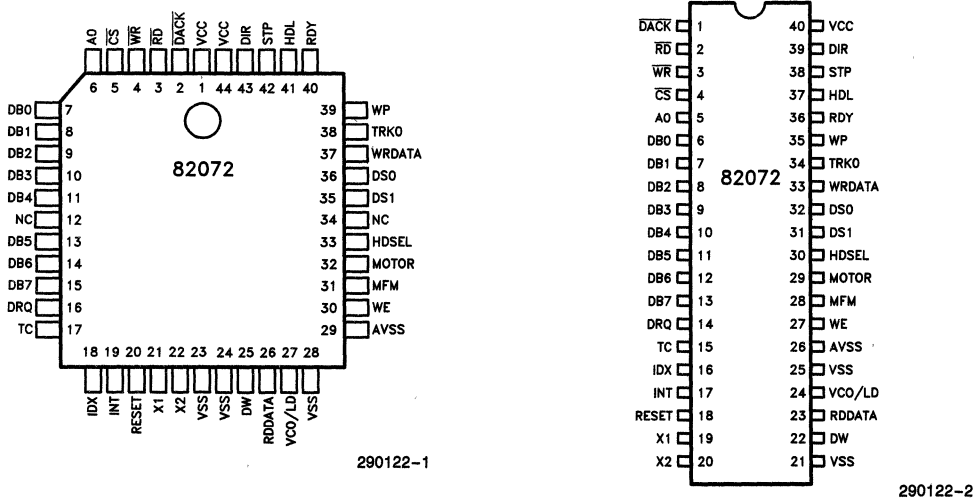


Figure 1. 82072 Pinout

Table 1. 82072 Pin Description

Symbol	DIP	PLCC	I/O	Function																																				
$\overline{\text{DACK}}$	1	2	I	DMA ACKNOWLEDGE: DMA control line that qualifies the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ inputs during DMA cycles.																																				
$\overline{\text{RD}}$	2	3	I	READ: Control signal to transfer data to the data bus from the 82072.																																				
$\overline{\text{WR}}$	3	4	I	WRITE: Control signal to transfer data into the 82072 from the data bus.																																				
$\overline{\text{CS}}$	4	5	I	CHIP SELECT: Control signal that qualifies the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ inputs.																																				
A0	5	6	I	ADDRESS: <table border="1"> <thead> <tr> <th>A0</th> <th>$\overline{\text{RD}}$</th> <th>$\overline{\text{WR}}$</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>Illegal *</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>Read Main Status Register</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>Write to the Data Rate Select Register **</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>No Action</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>Illegal *</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>Read from FIFO</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>Write into FIFO</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>No Action</td> </tr> </tbody> </table> <p>* User must ensure that these inputs do not occur. ** Change from 8272A—was illegal.</p>	A0	$\overline{\text{RD}}$	$\overline{\text{WR}}$	Function	0	0	0	Illegal *	0	0	1	Read Main Status Register	0	1	0	Write to the Data Rate Select Register **	0	1	1	No Action	1	0	0	Illegal *	1	0	1	Read from FIFO	1	1	0	Write into FIFO	1	1	1	No Action
A0	$\overline{\text{RD}}$	$\overline{\text{WR}}$	Function																																					
0	0	0	Illegal *																																					
0	0	1	Read Main Status Register																																					
0	1	0	Write to the Data Rate Select Register **																																					
0	1	1	No Action																																					
1	0	0	Illegal *																																					
1	0	1	Read from FIFO																																					
1	1	0	Write into FIFO																																					
1	1	1	No Action																																					
DB0-7	6-13	7-11, 13-15	I/O	DATA BUS: Bidirectional 8-bit data bus. A0 determines whether transfer is to/from the FIFO or Main Status Register.																																				
DRQ	14	16	O	DMA REQUEST: Used to request service from a DMA controller.																																				
TC	15	17	I	TERMINAL COUNT: Control line from a DMA controller used to terminate requests for data transfers. Disk read and write commands complete the transfer to the current sector with valid CRC checking/generation.																																				
IDX	16	18	I	INDEX: Disk drive signal that indicates the beginning of a track. It is used to count retries and delay periods for internal (i.e. Motor On/Off) timers and is rising edge triggered.																																				
INT	17	19	O	Interrupt: Interrupt to host to indicate command completion or that a data transfer is required (depending upon the data transfer mode). Command completion interrupts are cleared by reading the Main Status Register. Data transfer interrupts are cleared when the amount of data in the FIFO reaches the full or empty level (depending on FIFO direction).																																				

Table 1. 82072 Pin Description (Continued)

Symbol	DIP	PLCC	I/O	Function
RESET	18	20	I	RESET: Places the 82072 in a known idle state. All disk outputs are set to a low level. All registers, except those set by the SPECIFY command, are cleared. From the trailing edge of Reset, there is a maximum delay of 4 microseconds until the Main Status register is valid. Following reset, the 82072 defaults to polling disabled mode. The default values for the new features are: internal data separator enabled, write precompensation value is 250 ns, MOTOR on delay is 1 sec., MOTOR off delay is 5.2 sec., data rate is 250 kbps, FIFO enabled, threshold = 1.
X1	19	21	I	CRYSTAL 1: External connection for a fundamental mode parallel resonant 24 MHz crystal for the internal oscillator. May be driven with a MOS level clock instead of a crystal. Refer to the D.C. Specifications.
X2	20	22	I	CRYSTAL 2: If an external clock is supplied on X1, this input must be left unconnected (floating).
V _{SS}	21, 25	23, 24, 28		LOGIC GROUND.
DW	22	25	I***	DATA WINDOW: Clock from the external PLL logic used to sample the Read Data input. When the internal PLL is used, this input is ignored and should be tied to VCC or VSS.
RDDATA	23	26	I	READ DATA: Serial FM or MFM encoded data from the disk drive.
VCO/LD	24	27	O***	READ DATA GATE: This active high output enables an external PLL to synchronize to Read Data input from the disk drive. LOW DENSITY: This active high output is used by quad density disk drives to modify Read/Write head and data channel characteristics. This signal is activated when internal PLL is enabled and a data transfer rate of 300 Kbps is chosen.
AVSS	26	29		Analog ground for the Data Separator. This must not be connected directly to logic ground.
WE	27	30	O	WRITE ENABLE: Disk drive control signal that enables the head to write onto the disk.
MFM	28	31	O***	MFM MODE: When an external PLL is used, this output selects between single and double density (FM and MFM) modes. 1 = MFM, 0 = FM mode.
MOTOR	29	32	O	MOTOR ENABLE: Output used to activate the drive motor on the selected drive. Delays are programmable. With one output, this pin must be qualified with the drive select logic to provide motor enables for each drive.
HDSEL	30	33	O	HEAD SELECT: Signal used to select one of two sides on the disk. A 0 = side 0, a 1 = side 1.
DS1, 0	31, 32	35, 36	O	DRIVE SELECT: These outputs select one of four disk drives. DS0, DS1 = 0, 0 will select drive 0.
WRDATA	33	37	O	WRITE DATA: FM or MFM encoded serial data to the disk drive. No external precompensation is required.

Table 1. 82072 Pin Description (Continued)

Symbol	DIP	PLCC	I/O	Function
TRK0	34	38	I	TRACK 0: Control line from the disk drive that indicates the head is on track 0 (outermost track).
WP	35	39	I	WRITE PROTECT: Input from the disk drive that indicates if the disk is physically write protected. This input is treated the same as the RDY pin.
RDY	36	40	I	READY: Input from the disk drive that indicates whether the drive is ready for an operation. Command terminates if this is not valid.
HDL	37	41	O	HEAD LOAD: This output loads the head onto the disk drive if required. Typically used by 8" drives.
STP	38	42	O	STEP: Output used to supply step pulses to the disk drive.
DIR	39	43	O	DIRECTION: This output, in conjunction with STP, causes the drive to move the head outward if a "0", and inward if a "1".
VCC	40	1, 44		Logic DC power supply.

NOTES:

1. Pins 12, 34 of the 44 pin PLCC package are not connected.

***: These pins are intended for an external PLL. The outputs are always active and are not a function of the programming.

INTRODUCTION

The 82072 has integrated all of the complex circuitry required to interface microprocessor systems with disk drives that comply with the IBM System 34 Double Density (MFM) format or the IBM 3740 single density format (FM). The 82072 is a superset of the

8272A. Control over the new features was accomplished by adding extra registers and commands to the 82072. The 82072 will function like the 8272A Floppy Disk Controller (FDC) after being reset, with the added features being set to 8272A compatible default values. When accessing the disk drives, the 82072 is programmed the same as the 8272A.

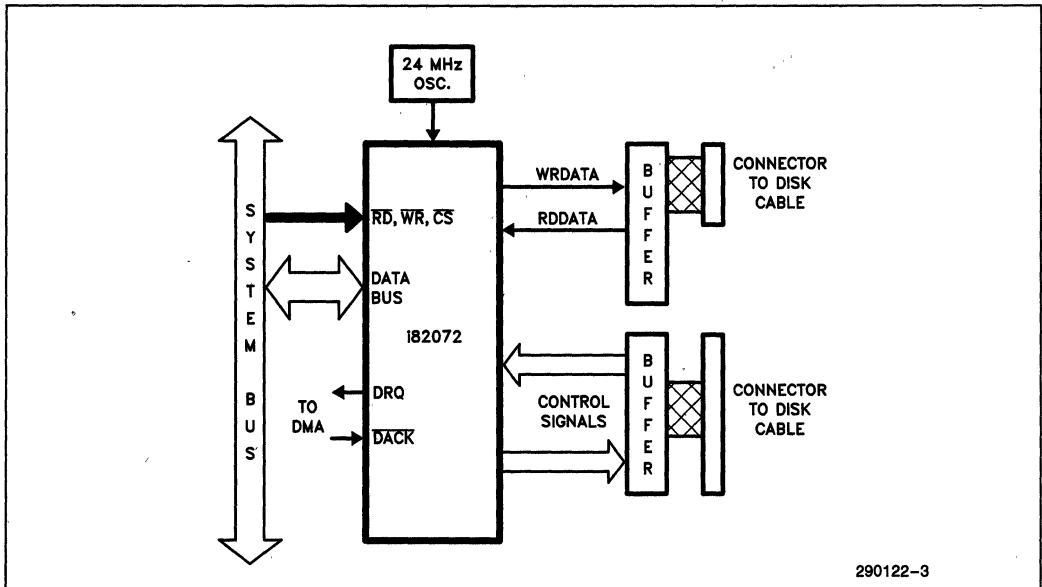


Figure 2. 82072 Typical System Block Diagram

The microprocessor interface was enhanced by adding a 16 byte FIFO to reduce the timing constraints that most floppy disk controllers impose upon a system. The point at which the 82072 generates a request for a data transfer is selectable within the 16 byte range of the FIFO. The interface was further enhanced to support today's faster microprocessors (i.e., 10 MHz iAPX286, 10 MHz iAPX186) without incurring wait states. A powerdown mode has been added for low power or portable applications. With one command, the 82072 resets itself and then disconnects the power from the internal oscillator. Reset will reconnect the clock and once the 82072 is reprogrammed (if necessary), it will be ready to read and write disks again.

All of the control logic of the disk interface has been integrated into the 82072. Flexibility is maintained by allowing the user to select read and write data rates

(without any external hardware); write precompensation delays; motor on/off delay; and the track to start the precompensation on. The typical design will need the 82072, a crystal and high current drivers for the signals that interface to the disk drive. The new features (when used) need only to be chosen once after reset (although they may be modified at any time). From then on, the user programs the 82072 for disk accesses in the same manner as the 8272A.

ARCHITECTURE

Figure 3 is a block diagram of the 82072. The highlighted blocks represent areas that are either completely new or highly enhanced. All new features were adopted with the requirement of being software compatible with the 8272A.

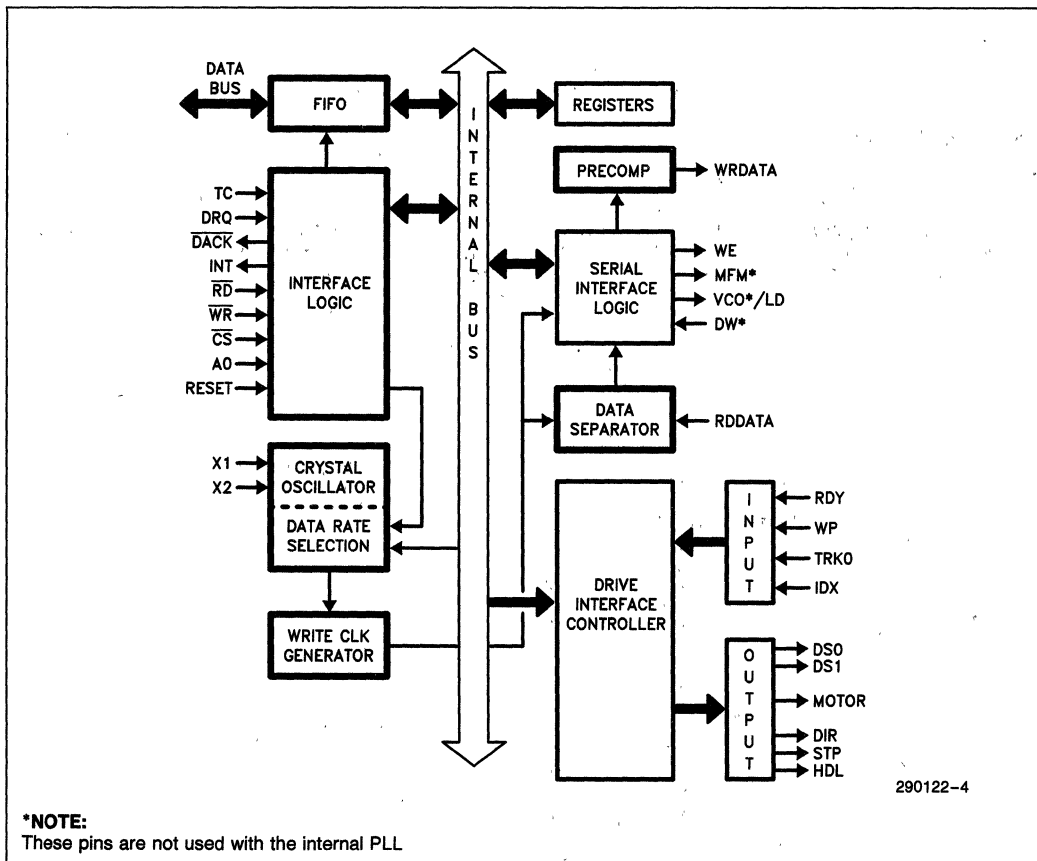


Figure 3. 82072 Block Diagram

Summary of Differences Between the 8272A and the 82072

Table 2 lists the features that are different between the 8272A and the 82072. The Scan commands are not supported by the 82072.

Table 2. Summary of Differences Between 82072 and 8272A

Features	82072	8272A
Process	CHMOS III	HMOS II
Data Separator	Internal	User Supplied
Data Rates	Up to 1 Mbit/sec. Software Selectable Without Hardware	Up to 500 kbit/sec External Hardware Needed
Power Down Mode	Internal	None
Commands	Superset of 8272A	
Host Interface	16 byte FIFO	1 byte Register
Software Reset	Yes	None
Implied Seeks	Yes	None
Relative Seek	Yes	None
Motor Delays	Yes	External Hardware
Write Precompensation Delays	Yes	External Hardware
Recalibrate	Issues 256 Step Pulses	Issues 77 Step Pulses

Crystal Oscillator Specification

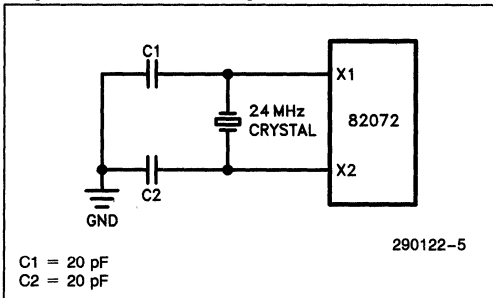


Figure 4. Crystal Oscillator Circuit

Specification of the parallel resonant crystal (typical values):

- Frequency: 24 MHz
- Mode: Parallel resonant
Fundamental mode
- Series Resistance: $R_S = 8\Omega$
- Motional Inductance: $L = 2.2 \text{ mH}$
- Motional Capacitance: $C = 0.02 \text{ pF}$
- Shunt Capacitance: $C_0 = 4.5 \text{ pF}$

The 24 MHz clock can be supplied either by a crystal or a MOS level square wave. All internal timings are referenced to this clock or a scaled count (which is determined by the data rate selection).

The crystal oscillator must be allowed to run for 10 ms after the VCC supply has reached 4.5 volts or 1 ms after the chip has been taken out of the Power-down mode before it is guaranteed to have stabilized.

MICROPROCESSOR INTERFACE

There are three ports accessible from the host's point of view; the FIFO, the Main Status Register (MSR) and the Data rate Select Register (DSR). Communication between the microprocessor and the 82072 is done by reading the Main Status register to determine if the controller is ready. If it is ready, a command, followed by the correct parameters, is sent to the 82072 through the FIFO (data port). The MSR can be read at any time and the DSR can be written at any time (before, during and after a command). The FIFO should be accessed when the RQM bit in the MSR is set, or if a DMA transfer is in progress.

MAIN STATUS REGISTER

RQM	DIO	NDM	CB	D3B	D2B	D1B	D0B
7	6	5	4	3	2	1	0

Bit 7—RQM (Request For Master)

This bit indicates that the host can access the FIFO (data port) if a "1". No accesses are permitted if set to a "0".

Bit 6—DIO (Data In/Out)

Indicates the direction of the data transfer only when RQM is a "1". If DIO is a "1" the host should remove the bytes in the FIFO. A "0" means that the host must write into the FIFO.

Bit 5—NDM (Non-DMA Mode)

If the non-DMA mode is selected in the SPECIFY command, this bit will be set = "1" only during the execution phase of a command. This bit is intended to support polled data transfers along with RQM and DIO. If DMA mode is selected, it remains a "0".

Bit 4—CB (Controller Busy)

CB is set to a "1" when a command is in progress and indicates that the controller is processing a command. This bit will go active in the command phase after the command byte has been accepted. CB goes inactive at the end of the result phase which indicates the start of the next command phase. If the command has no result phase (i.e. SEEK and RECALIBRATE) the CB bit is cleared after receiving the last byte in the command phase.

Bit 0-3; Drive Busy

These bits are set only when a drive is in the seek portion of a command, including implied seeks and overlapped seeks.

FIFO

A 16 byte FIFO was added to increase the flexibility of a system when transferring data between the disk and memory by increasing the amount of time before service is required. This permits the 82072 to have a lower priority in servicing without losing data.

The 82072 FIFO is 16 bytes deep with the threshold set at any point (the threshold is set with the Configure command). A reset will reset the threshold to an 8272A value of 1 byte. During writes into the FIFO (from the host), the 82072 will request service (via DRQ or INT-depending upon the selected transfer mode) when there are the programmed threshold number of bytes left in the FIFO. The request will go inactive once the FIFO is filled. When reading from the FIFO (to the host), service is requested when there are more than (16-threshold number) bytes in the FIFO. The request for service goes inactive when the FIFO is emptied.

During the Command phase (see next section), the FIFO is disabled and data must be sent only after the Main Status register has indicated that it is ready. Improper operation will result if command bytes are sent before the 82072 is ready. The 82072 does not ask for command parameters by generating interrupts or DMA requests. All command parameters are sent by polling the MSR.

As the 82072 enters the execution phase, it clears the FIFO of any data to ensure that invalid data is not written or read from the disk. During writes to the disk, if the host fails to respond in time, "00" will be written to the disk. During reads from the disk, the 82072 will not go into the Result phase until all data has been removed from the FIFO.

An overrun or underrun condition will terminate the transfer of data between the FIFO and the host. Writes to the disk will complete the current sector (using "00"s) and generate CRC. Reads will require the host to read all data bytes out of the FIFO so that the Result phase may begin. The proper error bit will be set in the Status register. A Terminal Count (TC) will always get the 82072 into the Result phase.

During the Result phase, the result bytes are read out one at a time and the FIFO is again disabled.

During disk transfers between the FIFO and the host (or DMA controller), the FIFO must be serviced within a specified period after the appropriate indicator is set (INT, DRQ). Table 3 gives an example of the delays. The 1.5 μ s delay is to convert the parallel data into serial MFM/(FM) data (with a clock speed of 24 MHz). Other data rate service delays are determined by:

$$\text{Threshold \#} \times \left[\frac{1}{\text{data rate}} \times 8 \right] - 1.5 \mu\text{s} = \text{DELAY}$$

Table 3. FIFO Service Delay

FIFO Threshold Examples	Maximum Delay to Servicing at 1 Mbps Data Rate
1 byte	$1 \times 8 \mu\text{s} - 1.5 \mu\text{s} = 6.5 \mu\text{s}$
2 bytes	$2 \times 8 \mu\text{s} - 1.5 \mu\text{s} = 14.5 \mu\text{s}$
8 bytes	$8 \times 8 \mu\text{s} - 1.5 \mu\text{s} = 70.5 \mu\text{s}$
15 bytes	$15 \times 8 \mu\text{s} - 1.5 \mu\text{s} = 118.5 \mu\text{s}$

FIFO Threshold Examples	Maximum Delay to Servicing at 500 Kbps Data Rate
1 byte	$1 \times 16 \mu\text{s} - 1.5 \mu\text{s} = 14.5 \mu\text{s}$
2 bytes	$2 \times 16 \mu\text{s} - 1.5 \mu\text{s} = 30.5 \mu\text{s}$
8 bytes	$8 \times 16 \mu\text{s} - 1.5 \mu\text{s} = 126.5 \mu\text{s}$
15 bytes	$15 \times 16 \mu\text{s} - 1.5 \mu\text{s} = 238.5 \mu\text{s}$

The FIFO defaults to enabled but with an 8272A compatible threshold level of 1 byte. If the FIFO is enabled, DRQ stays active until the FIFO is emptied/filled. If the FIFO is disabled with the EFIFO bit in the CONFIGURE command, individual DRQ's are issued. Refer to the section on DMA transfers for more detail.

COMMAND PHASES

Command Phase

After a RESET, the 82072 enters the Command phase and is ready to accept a command from host. For each of the commands, a defined set of command code bytes and parameter bytes has to be written to the 82072 before the command phase is complete. (Please refer to the Command Description section). These bytes of data must occur in the order prescribed. For example:

SEEK Command

D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	0	1	1	1	1	1st byte to the 82072
0	0	0	0	0	HDS	DS1	DS0	2nd byte to the 82072
<-----				NCN	-----		>	3rd byte to the 82072

Before writing to the 82072, the host must examine the RQM and DIO bits of the Main Status Register. RQM, DIO must be equal to "1" and "0", respectively before command bytes may be written. RQM is set false by the 82072 after each write cycle until the received byte is processed. The 82072 asserts RQM again to request each parameter byte of the command, unless an illegal command condition is detected. After the last parameter byte is received, RQM, DIO switch to "0", "1", and the 82072 automatically enters the next phase as defined by the command definition.

The FIFO is disabled during the command phase to retain compatibility with the 8272A, and to provide for the proper handling of the "invalid command" condition.

Execution Phase

All data transfers to or from the 82072 occur during the execution phase, which can proceed in DMA or non-DMA mode as indicated in the SPECIFY command.

The following paragraphs detail the operation of the FIFO flow control. In these descriptions, <threshold> is defined as the number of bytes available to the 82072 when service is requested from the host, and ranges from 1 to 16. The parameter FIFOTHR which the user programs is one less, and ranges from 0 to 15.

A low threshold value (i.e. 2) results in longer periods of time between service requests, but requires faster servicing of the request, for both read and write cases. The host must be very responsive to the service request. This is the desired case for use with a "fast" system.

A high value of threshold (i.e. 12) is used with a "sluggish" system by affording a long latency period after a service request, but results in more frequent service requests.

NON-DMA MODE, Transfers from the FIFO to the Host:

The INT pin and RQM bits in the Main Status Register are activated when the FIFO contains (16 - <threshold>) bytes, or the last bytes of a full sector transfer has been placed in the FIFO. The INT pin can be used for interrupt driven systems and RQM can be used for polled systems. The host must respond to the request by reading data from the FIFO. The 82072 will deactivate the INT pin and RQM bit when the FIFO becomes empty. This process is repeated until the last byte is transferred out of the FIFO.

NON-DMA MODE, Transfers from the Host to the FIFO:

The INT pin and RQM bit in the Main Status Register are activated upon entering the execution phase of data transfer commands. The host must respond to the request by writing data into the FIFO. The INT pin and RQM bit remain true until the FIFO becomes full. They are set true again when the FIFO has <threshold> bytes remaining in the FIFO. The 82072 will also deactivate the INT pin and RQM bit when TC goes active, indicating that no more data is required. The 82072 enters the results phase after the last byte is taken by the 82072 from the FIFO (i.e., FIFO empty condition).

DMA MODE, Transfers from the FIFO to the Host:

The 82072 activates the DRQ pin when the FIFO contains $(16 - \text{<threshold>})$ bytes, or the last byte of a full sector transfer has been placed in the FIFO. The DMA controller must respond to the request by reading data from the FIFO. The 82072 will deactivate the DRQ pin when the FIFO becomes empty. DRQ goes inactive after $\overline{\text{DACK}}$ goes active for the last byte of a data transfer (or on the active edge of $\overline{\text{RD}}$, on the last byte, if no edge is present on $\overline{\text{DACK}}$). A data underrun may occur if DRQ is not removed in time to prevent an unwanted cycle. Refer to the section on DMA Transfers for more detail.

DMA MODE, Transfers from the Host to the FIFO:

The 82072 activates the DRQ pin when entering the execution phase of the data transfer commands. The DMA controller must respond by activating the $\overline{\text{DACK}}$ and $\overline{\text{WR}}$ pins, and placing data in the FIFO. DRQ remains active until the FIFO becomes full. DRQ is again set true when the FIFO has <threshold> bytes remaining in the FIFO. The 82072 will also deactivate the DRQ pin when TC becomes true, indicating that no more data is required. DRQ goes inactive after $\overline{\text{DACK}}$ goes active for the last byte of a data transfer (or on the active edge of $\overline{\text{WR}}$ on the last byte, if no edge is present on $\overline{\text{DACK}}$). A data overrun may occur if DRQ is not removed in time to prevent an unwanted cycle. Refer to the section on DMA Transfers for more detail.

The 82072 supports terminal count explicitly through the TC pin and implicitly through the underrun/overrun and end-of-track (EOT) functions. The EOT parameter can define the last sector to be transferred in a multisector transfer. If the host stops transferring data mid-sector, the 82072 will complete the transfer in the same way as if a hardware TC was received. The only difference between these implicit functions and TC is that they return "abnormal termination" result status. Such status indications can be ignored if they were expected.

Note that when the host is sending data to the 82072's FIFO, the internal sector count will be complete when the 82072 reads the last byte from its side of the FIFO. There may be a delay in the removal of the transfer request signal of up to the time taken for the 82072 to read the last 16 bytes from the FIFO. The host must tolerate this delay.

If before the execution of a command, the RDY pin is "0", the 82072 will set the corresponding bit in Status Register 0, and terminate the command phase. When an external TC signal is received, the external transfer is suspended, but the sector transfer is completed internally before the command phase is ended. The command phase is also terminated if the last sector on the track has been read or written, (as defined by the EOT parameter).

The 82072 activates INT to indicate the beginning of the result phase. The first reading of status from the 82072 resets the INT pin.

Result Phase

The generation of INT determines the beginning of the result phase. For each of the commands, a defined set of result bytes has to be read from the 82072 before the result phase is complete. (Refer to the Command Description section). These bytes of data must be read out for another command to start.

RQM and DIO must be equal to "1" and "1", before the result bytes may be read from the FIFO. RQM and DIO will remain "1" and "1" until the required number of bytes are read from the 82072, then RQM, DIO will switch to "0", "1" and the 82072 automatically returns to the command phase. At some point within the result phase, the CB bit will be set to a "0".

DATA RATE SELECT REGISTER (DSR)

The Data Rate select register gives the user control over the read and write disk data. The user can select between an internal or external data separator, the data rate of the data separator and the delays for the write precompensation logic. The internal data separator requires no external control logic or analog components with its selectable data rates. SOFTWARE RESET and the POWERDOWN mode are invoked by setting the appropriate bit in this register.

When the processor writes into the DSR, the data is loaded after a delay to synchronize to an internal machine state. The processor must not perform successive writes into the DSR until this synchronization

time (24 clock periods or 1 microsecond at 24 MHz) has elapsed. This register should be programmed before issuing a command that accesses the disk and uses values that this register controls. There is a 2 millisecond delay between writing the data rate select bits and having the PLL stabilize at the new frequency. This can be hidden by overlapping it in the stepping and head load delays. There is no minimum delay between writing this register and any other 82072 register.

A write to this register during data transfers will alter the contents of this register and the logic it controls. Data rates and precompensation values will change which may give undesirable results.

Changing the data rate also changes the timings of the drive control signals that are initialized with the SPECIFY command. To ensure that drive timings are not violated, the user should either choose values of drive timings such that the fastest data rate would not violate a drive timing specification or should follow a write to the data rate select bits with a new SPECIFY command. Refer to the SPECIFY command for more detail on these timings.

The Datarate select register (DSR) is a write only register. The functions of the DSR are indicated below:

SWR	PD	EPLL	PRE-COMP			DRATESEL	
7	6	5	4	3	2	1	0

Bit 7—SWR; (SOFTWARE RESET)

When set to “1”, enables software reset of the 82072. A software reset causes the hardware reset line to go active internally. Refer to Figure 5 for an illustration. This method ensures that software and hardware resets perform the same function. One function not reset is the DSR (since it contains the software reset bit). Since a hardware reset is being performed, the MSR will not be valid for up to 4 microseconds after the Software Reset is issued. The users software should wait that period before attempting to read the Main Status register. Software reset can also be cleared with a hardware reset.

In the case of software reset, the DSR retains the values previously loaded into it. Upon hardware reset, the DSR is configured to the following, to provide compatibility with the 8272A:

SWR	PD	EPLL	PRE-COMP			DRATESEL	
7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0

Default values: 250 Kbit/sec., MFM, Internal data separator, Precompensation = 125 ns

Bit 6—PD; (POWER DOWN)

When set to “1”, the 82072 goes into its POWER DOWN mode. A SOFTWARE RESET is performed and held when entering POWER DOWN to ensure that no disk control pins are in the active state. The powerdown mode deactivates the internal clocks and shuts off the oscillator. Only writes to the DSR are allowed during powerdown and ALL input signals must be held in a valid state (either D.C. 1 or 0). The only exception is the data bus may be floated and there will be no increase in the ICC specification. To exit the Power down mode, either a hardware or software reset must be given. Once reset is given, there is a delay of 1 ms for the oscillator to stabilize at the correct frequency.

The 82072 preserves the contents of the PCN register during POWERDOWN. However, all status registers are cleared. Prior to issuing the POWERDOWN command, it is highly recommended that the user service all pending interrupts through the SENSE INTERRUPT STATUS command.

Bit 5—EPLL; (Enable PLL)

When set to “0”, the internal PLL data separator supplies the data window input and the DW pin is ignored. The data rate of the internal PLL is determined by the DRATESEL bits. When set to “1”, the internal data separator is disconnected from the serial interface controller and an external data separator is required to supply the DW signal. Refer to the section on the internal data separator for more detail.

Bits 2–4—PRECOMP; (Pre-compensation)

The write precompensation circuitry adjusts the write data pulse before it is sent to the drive on the WRDATA pin. A programmed compensation interval is added to or subtracted from the normal write pulse timing as a function of the data pattern. The CONFIGURE command is used to specify the track number that precompensation starts upon. If you don't issue a CONFIGURE command, the 82072 defaults to beginning pre-compensation on Track 0.

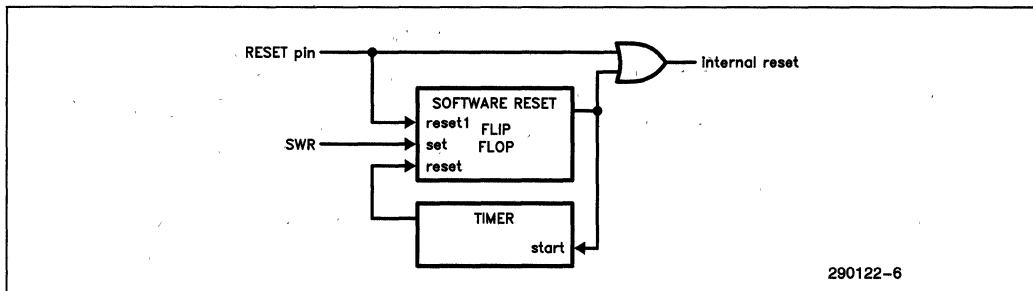


Figure 5. Reset Block Diagram

Table 5. Pre-Compensation Delays

PRECOMP 432	Pre-Compensation Delay
111	0.00 ns-DISABLED
001	41.67 ns
010	83.34 ns
011	125.00 ns
100	166.67 ns
101	208.33 ns
110	250.00 ns
000	DEFAULT

If the Data Rate Select Register (DSR) is programmed with bits 2-4 set to zeroes, a default pre-compensation value matching the PC-AT is automatically chosen depending upon the data rate select bits (bits 0 and 1). The default values corresponding to the data rate select values are illustrated in Table 5A.

Table 5A. Default Pre-Compensation Delays

Data Rate	Pre-Compensation Delays
1 Mbps	41.67 ns
500 Kbps	125 ns
300 Kbps	125 ns
250 Kbps	125 ns

Bits 0, 1—DRATESEL; (Data Rate Select)

Programs both the read and write data rates. For single density (FM mode), data rates are one half the values stated for the double density (MFM) mode. MFM or FM selection is made in the disk read and write commands.

Table 4. Data Rates

DRATESEL		DATA RATE	
1	0	MFM	FM
1	1	1 Mbps	Illegal
0	0	500 Kbps	250 Kbps
0	1	300 Kbps	150 Kbps
1	0	250 Kbps	125 Kbps

WRITE CLOCK

The basic "core" of the 8272A was carried over to the 82072 virtually unchanged. To keep the flexibility of the 8272A, where the user could vary data rates by changing the frequency on the WRCLK pin, and to minimize the external logic, a prescaler for the internal WRCLK was added. The prescaler divides down the 24 MHz clock to rates that are equivalent to those used with the 8272A. The user changes the prescaler value with the data rate selection bits in the DATA RATE SELECT register.

POLLING OF DRIVES

The 82072 defaults to polling disabled mode. Polling can be enabled by issuing a CONFIGURE command. When enabled, the 82072 polls the drives and looks for a change in the state of the RDY pin. Each of the drives is selected for a period of time and its RDY pin sampled. After a delay, the next drive is selected. This sequence occurs whenever the 82072 is in the command phase. The drives are assumed to be not ready initially and a "ready" value for each drive is saved in an internal register. An interrupt will be issued when a drive signals that it changed state from not ready to ready. This interrupt must be followed with a SENSE INTERRUPT STATUS command from the host. The next time that the drive changes its state, an interrupt will again be issued.

The length of time that a drive is selected is dependent upon the selected data rate. At 1 Mbps, drives 0-2 are selected for approximately 115 microseconds, and drive 3 for twice that amount. At 500 Kbps, the selected period is doubled for all drives. This period continues to scale with the data rate selection. The amount of time that the 82072 actually samples the RDY pin is about 15 microseconds at 1 Mbps and also scales with the data rate. This value is equal for all drives. The remaining time the drive is selected, changes on RDY will not be sampled.

NOTE:

Polling is disabled when an Implied Seek is in progress.

TERMINAL COUNT (TC)

TC completes the current disk data transfer and then puts the 82072 into the result phase.

RQM will go inactive immediately after TC is received. There will be no more requests for data transfers. All remaining bytes in the FIFO are read out by the 82072. The remaining space in the disk sector will be filled with "00"s. For read's RQM goes inactive and remaining data bytes are "dumped" internally and CRC checked. An overrun/underrun error will not be reported in the result phase. The FIFO will be reset upon entering the result phase. This ensures that any remaining data bytes sent by the host do not interfere with the result bytes.

An implied terminal count also exists. Letting the FIFO overrun or underrun will terminate all requests for data transfers from the 82072. It terminates the transfer exactly the same as the hardware Terminal Count except that the overrun error bit will be set in the status register.

Terminal count will always get the 82072 out of the execution phase which would be caused by the host failing to perform the correct action.

DMA TRANSFERS

DMA transfers are initiated by the 82072 when it activates its DRQ (DMA Request) pin. When ready, the DMA controller responds by activating $\overline{\text{DACK}}$ (DMA Acknowledge). If the DMA transfer mode is selected in the 82072 (with the SPECIFY command), $\overline{\text{DACK}}$ must be used in the data transfer sequence. $\overline{\text{DACK}}$ is used to clock internal logic.

If the FIFO is disabled by setting the EFIFO bit in the CONFIGURE command, the FIFO does not exist

and each byte to be transferred will request a cycle. Figure 6 is an example of the handshaking that occurs.

If the FIFO is enabled (the default value), DRQ will go active for a transfer and stay active until the FIFO is filled or cleared (write and read respectively).

The 82072 can be given pulsed $\overline{\text{DACK}}$'s (which is compatible with the iAPX186 and most two cycle DMA controllers) or $\overline{\text{DACK}}$'s that stay active until the transfer is complete (compatible with the Intel 8237A controller). A pulsed $\overline{\text{DACK}}$ provides more time for the 82072 to determine if DRQ should be removed (since $\overline{\text{DACK}}$ goes active before RD). If $\overline{\text{DACK}}$ is not pulsed, then the active going edge of RD is used for this decision. The edge of RD is generally too late for DRQ to be removed and prevent an unwanted DMA cycle.

There is a small delay after $\overline{\text{DACK}}$ (or RD) goes active at the 82072 before DRQ is removed, but this may not satisfy the DMA controller specification ($\overline{\text{DACK}}$ active to DRQ inactive) to prevent an unwanted DMA cycle. It is the users responsibility to ensure that an unwanted cycle does not happen due to DRQ not going inactive quickly enough. It may be necessary for the design to generate a pulsed $\overline{\text{DACK}}$ or one that is earlier than the one supplied by the DMA controller.

DRIVE INTERFACE

Figure 8 is a block diagram of the floppy drive interface for the 82072. The external logic needed are high current buffers to drive the cable, termination and MOTOR and drive select generation.

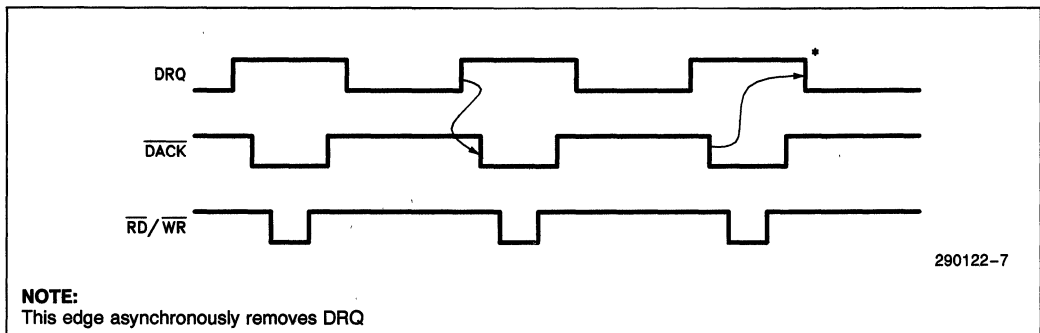


Figure 6. Disabled FIFO DMA Cycles

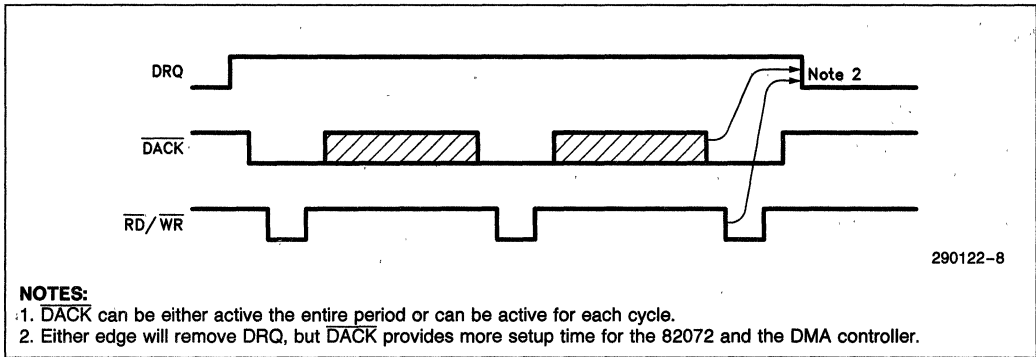


Figure 7. DMA Transfers with the FIFO Enabled

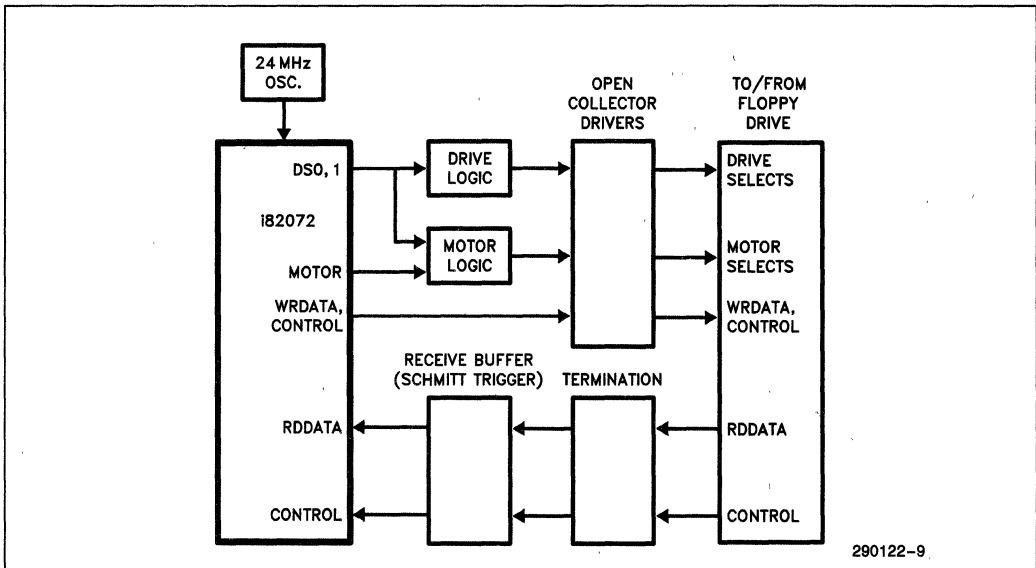


Figure 8. Drive Interface Block Diagram

When a read operation begins, the sync detect logic samples the read data stream until it finds a valid sync field, which is a pattern of eight contiguous zero bit cells. This is found with an internal one-shot. It assumes that this is a synchronization field (Refer to the FORMAT command for detail) and will switch the VCO from the reference clock to the data stream after waiting the delay specified in the GAP parameter of the READ and WRITE commands. The sync logic then switches the READ DATA stream to the input of the data separator. The clock that the PLL is synchronized to is forced to be in phase with the read data as the switch over occurs. The PLL of the data separator starts with almost zero phase error, which greatly reduces the capture time. Once VCO

is activated and the PLL has locked onto the serial data, it is searched for an ID address mark. If the first non-sync data is not an ID address mark, the VCO line is deactivated and the search for a sync. field begins again with the data separator waiting for the one-shot to find another sync. field. If the address mark was good, the ID field is examined for the correct parameters and CRC is checked. When the 82072 is not looking at the disk data, it remains synchronized to the programmed data rate. This method ensures that the PLL does not lock onto harmonics. It also allows a faster lock-up time when disk data is fed into the PLL.

PHASE LOCK LOOP

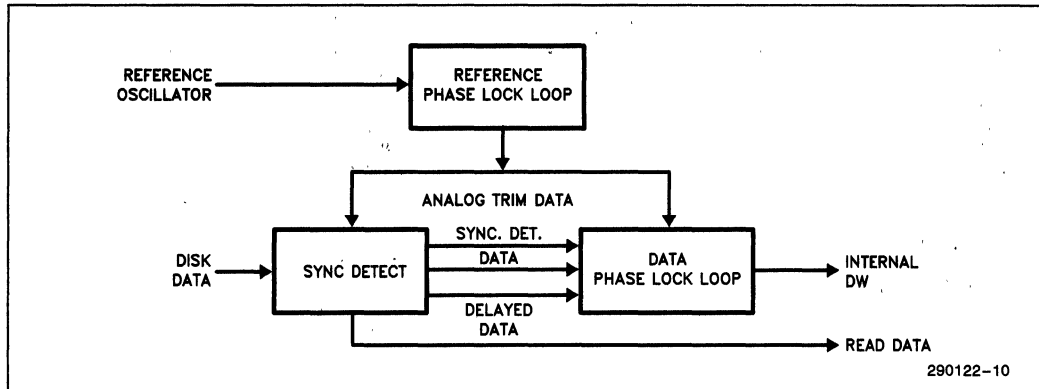


Figure 9. Data Separator Block Diagram

The internal Data Separator consists of two analog phase lock loops as is shown in Figure 9. The data separator PLL's function is to recover the clock from the data stream so that the data may be decoded. The data separator PLL tracks variations in drive speed and pulse jitter. To increase the effectiveness

of the data PLL, a second PLL was added. This REFERENCE PLL takes into account temperature, voltage and data rate variations and adjusts the data PLL proportionately. The combination of the two PLL's eliminates the need for any external trimming components over the range of data rates.

PHASE LOCK LOOP OVERVIEW

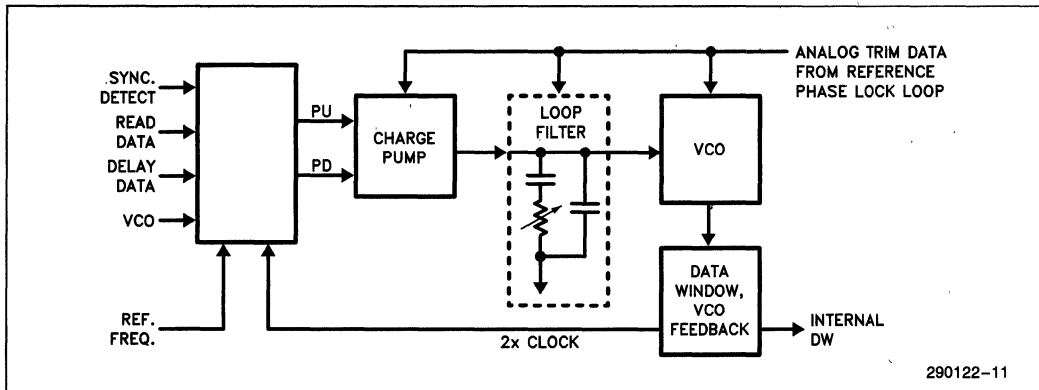


Figure 10. Data PLL

The PLL design is optimized to reduce the chance of noise induced errors. The entire analog section has its own voltage reference which has good VCC independence. A 10% AVCC change translates into a 1% change seen by the analog circuitry.

The Phase/Frequency Comparator section is a standard implementation. It generates two signals (PUMP UP-PU and PUMP DOWN-PD) which indicate how far from the center frequency the data stream is. The Charge Pump essentially amplifies these signals for the Loop Filter.

The Loop Filter is a second order type with all of the capacitors on-chip and compensated for against temperature, frequency and VCC variations. The filter bandwidth is the data rate divided by 88. Data rate selection in the filter is made via the variable resistor as shown in the diagram. The second capacitor is much smaller than the one in series with the resistor. Its purpose is to filter out high frequency noise.

The VCO design is optimized to reduce its sensitivity to noise on AVCC and in the read data stream.

Lock Definition

The data separator phase locked loop is defined to be in lock when the phase error is equal to or less than 5% of the bit cell. As an example, at 1 Mbps the bit cells are 1 μ s long, 5% of that is 50 ns. Thus by definition the loop will be in lock for this data rate if the phase error is equal to or less than ± 50 ns @ 1 Mbps.

Locktime

The 82072 data separator is designed to acquire lock, as defined above, within 48 bit times, independent of data rate.

Jitter Response

The 82072 data separator can tolerate a phase error impulse of 17.5% maximum, and still remain in lock, as defined above. As an example, at 1 Mbs, a phase impulse of 175 ns should result in a phase error of 50 ns or less. At 500 Kbps, a phase impulse (error) of 350 ns or less will not cause the PLL to lose lock.

Capture Frequency Range

The 82072 data separator is designed to lock onto read data with a maximum frequency error of 6.0%. This total is made up of the following components:

- Motor speed change $\pm 1.5\%$
- Instantaneous speed change $\pm 1.5\%$

SYSTEM CONSIDERATIONS

By far, most problems that will be seen will center upon noise even though many measures were implemented to reduce the possibility of external noise related problems. Any digital signals next to the RDDATA input line on the board should also be avoided. Taking above average precautions will pay off with fewer "soft" errors when reading the disk and enhance system reliability.

WRITE PRECOMPENSATION

The write precompensation logic is used to minimize bit shifts in the read data stream from the disk drive. The shifting of bits is a known phenomena of magnetic media and is dependent upon the disk media AND the floppy drive.

The 82072 monitors the bit stream that is being sent to the drive. The data patterns that require precompensation are well known. Depending upon the pattern, the bit is shifted either early or late (or not at all) relative to the surrounding bits. Figure 11 is a block diagram of the internal circuit

The top is a 13 bit shift register with the no delay tap being in the center. This allows 6 levels of early and late shifting with respect to nominal. The shift register is clocked at the main clock rate (24 MHz). The output is fed into 2 multiplexors—one for early and one for late. A final stage of multiplexors combines the early, late and normal data stream back into one which is the WRDATA output.

COMMAND SET

Table 5B. 82072 Command Set

Phase	R/W	DATA BUS								Remarks		
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀			
READ DATA												
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0			
	W					C					Sector ID information prior to Command execution	
	W					H						
	W					R						
	W					N						
	W					EOT						
W					GPL							
W					DTL							
Execution	Data transfer between the FDD and main-system											
	Result	R					ST 0					Status information after Command execution
		R					ST 1					
		R					ST 2					
		R					C					Sector ID information after Command execution
		R					H					
		R					R					
R						N						
READ DELETED DATA												
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0			
	W					C					Sector ID information prior to Command execution	
	W					H						
	W					R						
	W					N						
	W					EOT						
W					GPL							
W					DTL							
Execution	Data transfer between the FDD and main-system											
	Result	R					ST 0					Status information after Command execution
		R					ST 1					
		R					ST 2					
		R					C					Sector ID information after Command execution
		R					H					
		R					R					
R						N						
WRITE DATA												
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0			
	W					C					Sector ID information prior to Command execution	
	W					H						
	W					R						
	W					N						
	W					EOT						
W					GPL							
W					DTL							
Execution	Data transfer between the main-system and FDD											
	Result	R					ST 0					Status information after Command execution
		R					ST 1					
		R					ST 2					
		R					C					Sector ID information after Command execution
		R					H					
		R					R					
R						N						

Table 5B. 82072 Command Set (Continued)

Phase	R/W	DATA BUS								Remarks		
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀			
WRITE DELETED DATA												
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0			
	W					C					Sector ID information prior to Command execution	
	W					H						
	W					R						
	W					N						
	W					EOT						
	W					GPL						
W					DTL							
Execution										Data transfer between the FDD and main-system		
	Result	R					ST 0					Status information after Command execution
		R					ST 1					
		R					ST 2					
		R					C					Sector ID information after Command execution
		R					H					
		R					R					
		R					N					
READ TRACK												
Command	W	0	MFM	SK	0	0	0	1	0	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0			
	W					C					Sector ID information prior to Command execution	
	W					H						
	W					R						
	W					N						
	W					EOT						
	W					GPL						
W					DTL							
Execution										Data transfer between the FDD and main-system. FDC reads all of cylinders contents from index hole to EOT		
	Result	R					ST 0					Status information after Command execution
		R					ST 1					
		R					ST 2					
		R					C					Sector ID information after Command execution
		R					H					
		R					R					
		R					N					
READ ID												
Command	W	0	MFM	0	0	1	0	1	0	Commands		
	W	0	0	0	0	0	HDS	DS1	DS0			
Execution										The first correct ID information on the Cylinder is stored in Data Register		
Result	R					ST 0					Status information after Command execution	
	R					ST 1						
	R					ST 2						
	R					C					Disk status after the Command has completed.	
	R					H						
	R					N						

Table 5B. 82072 Command Set (Continued)

Phase	R/W	DATA BUS								Remarks	
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		
FORMAT TRACK											
Command	W	0	MFM	0	0	1	1	0	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		
	W	_____			N	_____					
	W	_____			SC	_____					
	W	_____			GPL	_____					
Execution	W	_____			D	_____					
	FDC formats an entire cylinder										
	Result	R	_____			ST 0	_____				
		R	_____			ST 1	_____				
		R	_____			ST 2	_____				
R		_____			C	_____					
R		_____			H	_____					
R	_____			R	_____						
R	_____			N	_____						
Status information after Command execution											
Disk status after the Command has completed											
RECALIBRATE											
Command	W	0	0	0	0	0	1	1	1	Command Codes	
	W	0	0	0	0	0	0	DS1	DS0		
Execution	Head retracted to Track 0 Interrupt										
SENSE INTERRUPT STATUS											
Command	W	0	0	0	0	1	0	0	0	Command Codes	
	Result	R	_____			ST 0	_____				
R	_____			PCN	_____						
Status information at the end of each seek operation about the FDC											
SPECIFY											
Command	W	0	0	0	0	0	0	1	1	Command Codes	
	W	_____ SRT _____ > < _____ HUT _____									
	W	_____ HLT _____			> ND						
SENSE DRIVE STATUS											
Command	W	0	0	0	0	0	1	0	0	Command Codes	
	W	MOT	0	0	0	0	HDS	DS1	DS0		
Result	R	_____			ST 3	_____					
Status information about FDD											
SEEK											
Command	W	0	0	0	0	1	1	1	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		
	W	_____			NCN	_____					
Execution	Head is positioned over proper Cylinder on Diskette										
CONFIGURE											
Command	W	0	0	0	1	0	0	1	1	Configure Information	
	W	HSDA	< _____ MOFF _____ >			< _____ MON _____ >					
	W	0	EIS	EFIFO	POLL	< _____ FIFOTHR _____ >					
	W	< _____ PRETRK _____ >									
MOTOR ON/OFF											
Command	W	On/Off	DS1	DS0	0	1	0	1	1	Command Code	
RELATIVE SEEK											
Command	W	1	DIR	0	0	1	1	1	1		
	W	0	0	0	0	0	HDS	DS1	DS0		
	W	_____			RCN	_____					

Table 5B. 82072 Command Set (Continued)

Phase	R/W	DATA BUS								Remarks
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
DUMPREG										
Command Execution	W	0	0	0	0	1	1	1	1	*Note Registers placed in FIFO See Note
Result	R	_____ PCN-Drive 0 _____				_____				
	R	_____ PCN-Drive 1 _____				_____				
	R	_____ PCN-Drive 2 _____				_____				
	R	_____ PCN-Drive 3 _____				_____				
	R	< _____ SRT _____ >				< _____ HUT _____ >				
	R	< _____ HLT _____ >				ND				
	R	< _____ SC/EOT _____ >				_____				
	R	HSDA < _____ MOFF _____ >				< _____ MON _____ >				
	R	< _____ PRETRK _____ >				_____				
	R	0	EIS	EFIFO	POLL	< _____ FIFOTHR _____ >				
INVALID										
Command	W	_____ Invalid Codes _____								Invalid Command Codes (NoOp — FDC goes into Standby State) ST 0 = 80 (16)
Result	R	_____ ST 0 _____								

SC is returned if the last command that was issued was the FORMAT command. EOT is returned if the last command was a READ or WRITE.

NOTES:

1. Symbols used in this table are described below.
2. A₀ = 1 for all operations.
3. X = 'Don't care, usually made to equal binary 0.'

PARAMETER ABBREVIATIONS:

Symbol Description

- C Cylinder address. The currently selected cylinder address, 0 to 255.
- D Data pattern. The pattern to be written in each sector data field during formatting. DS0, DS1 Disk Drive Select.

DS1	DS0	
0	0	drive 0
0	1	drive 1
1	0	drive 2
1	1	drive 3

DTL Special sector size. By setting N to zero (00), DTL may be used to control the number of bytes transferred in disk read/write commands. The sector size (N = 0) is set to 128. If the actual sector (on the diskette) is larger than DTL, the remainder of the actual sector is read but is not passed to the host during read commands; during write commands, the remainder of the actual sector is written with all zero bytes. The CRC check code is calculated with the actual sector. When N is not zero, DTL has no meaning and should be set to FF HEX.

EOT End of track. The final sector number of the current track.

Symbol Description

- GPL Gap length. The gap 3 size. (Gap 3 is the space between sectors excluding the VCO synchronization field).
- H/HDS Head address. Selected head: 0 or 1 (disk side 0 or 1) as encoded in the sector ID field.
- HLT Head load time. The time interval that 82072 waits after loading the head and before initiating a read or write operation. Refer to the SPECIFY command for actual delays.
- HUT Head unload time. The time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Refer to the SPECIFY command for actual delays.
- MFM MFM/FM mode selector. A one selects the double density (MFM) mode. A zero selects single density (FM) mode.
- MT Multi-track selector. When set, this flag selects the multi-track operating mode. In this mode, the 82072 treats a complete cylinder, under head 0 and 1, as a single track. The 82072 operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set, a multitrack read or write operation will automatically continue to the first sector

Symbol Description
 under head 1 when the 82072 finishes operating on the last sector under head 0.
N Sector size code. This specifies the number of bytes in a sector. If this parameter is "00", then the sector size is 128 bytes. The number of bytes transferred is determined by the DTL parameter. Otherwise the sector size is (2 raised to the "N'th" power) times 128. All values up to "07" hex are allowable. "07" would equal a sector size of 16k. It is the users responsibility to not select combinations that are not possible with the drive.

N	Sector Size
00	128 bytes
01	256 bytes
02	512 bytes
03	1024 bytes
..	...
07	16 Kbytes

NCN New cylinder number. The desired cylinder number.
ND Non-DMA mode flag. When set to 1, indicates that the 82072 is to operate in the non-DMA mode. In this mode, the host is interrupted for each data transfer. When set to 0, the 82072 operates in DMA mode, interfacing to a DMA controller by means of the DRQ and DACK# signals.

Symbol Description
PCN Present cylinder number. The current position of the head at the completion of SENSE INTERRUPT STATUS command.
R Sector address. The sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of the first sector to be read or written.
SC Number of sectors per track. The number of sectors per track to be initialized by the FORMAT command.
SK Skip flag. When set to 1, sectors containing a deleted data address mark will automatically be skipped during the execution of READ DATA. If READ DELETED is executed, only sectors with a deleted address mark will be accessed. When set to "0", the sector is read or written the same as the read and write commands.
SRT Step rate interval. The time interval between step pulses issued by the 82072. Programmable from 0.5 to 8 milliseconds, in increments of 0.5 ms at the 1 Mbit data rate. Refer to the SPECIFY command for actual delays.
ST0 Status register 0-3. Registers within the 82072 that store status information after a command has been executed. This status information is available to the host during the result phase after command execution.

Commands can be written whenever the 82072 is in the command phase. The command set falls into two categories as is shown in Table 6. The encoding of the bits used in this description can be found in the Parameter Abbreviation section which follows this section.

Each command has a unique set of parameters that are needed and that are returned as status. The 82072 checks to see that the first byte received while in the command phase is a valid command. If valid, there will be no change in the status bits in the

MSR. If the value was not a valid command, an interrupt is issued. The user would issue the SENSE INTERRUPT STATUS command which would return an invalid command error condition.

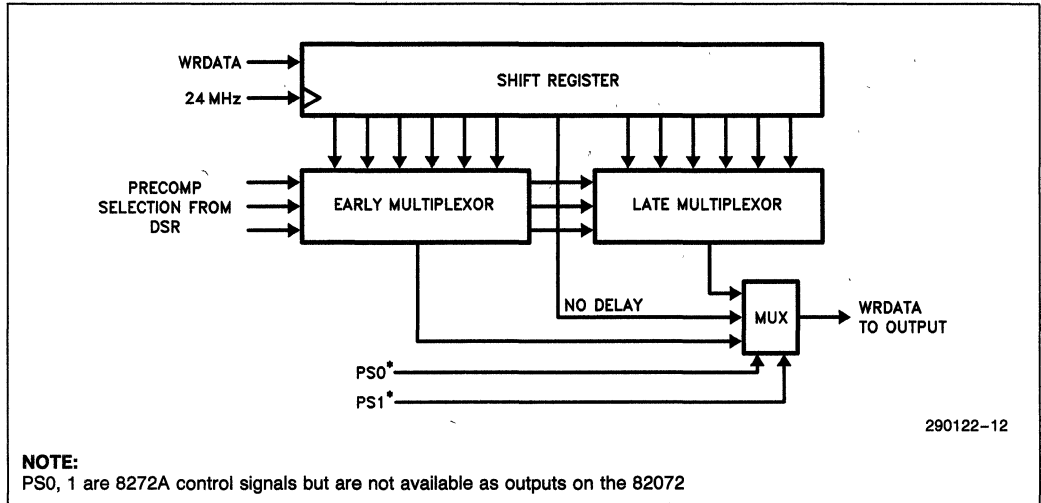
DATA TRANSFER COMMANDS

The READ DATA, READ DELETED DATA, WRITE DATA, WRITE DELETED DATA and READ TRACK require and return the same parameter and status bytes respectively. The only differences being the coding of bits 0-4 in the first byte sent to the 82072.

Table 6. Command Summary

Data Transfer	Control	
Read Data	Recalibrate	Configure *
Read Deleted Data	Sense Interrupt Status	Motor On/Off *
Write Data	Specify	Relative Seek *
Write Deleted Data	Sense Drive Status	Dumpreg *
Read Track	Seek	
Format Track	Read ID	

***NOTE:**
 These commands are not present in the 8272A.



NOTE:
PS0, 1 are 8272A control signals but are not available as outputs on the 82072

Figure 11. Precompensation Block Diagram

Table 7. Data Transfer Command Summary

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
Command	W	MT	MFM	SK	0	X	X	X	X	Command Code
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	-----				C	-----			
	W	-----				H	-----			
	W	-----				R	-----			
	W	-----				N	-----			
	W	-----				EOT	-----			
	W	-----				GPL	-----			
Execution	W	-----				DTL	-----			Data Transfer Occurs.
Result	R	-----			ST0	-----			Status After Command Has Completed. Disk Status After The Command Has Completed.	
	R	-----			ST1	-----				
	R	-----			ST2	-----				
	R	-----			C	-----				
	R	-----			H	-----				
	R	-----			R	-----				
R	-----			N	-----					

Table 8. Command Encoding

Command	D3	D2	D1	D0
Read Data	0	1	1	0
Read Deleted Data	1	1	0	0
Write Data	0	1	0	1
Write Deleted Data	1	0	0	1
Read Track	0	0	1	0

IMPLIED SEEK FOR READ AND WRITE COMMANDS

The 82072 will execute an implied seek for any command that specifies the cylinder number. The implied seek execution is transparent to the user. This means that the user would issue a READ DATA command, send all of the parameter bytes, see step pulses issued and start reading data.

After the parameter bytes are received, a comparison is made between PCN and NCN. If stepping is required and the Implied Seek bit in the CONFIGURE command was set, then a SEEK command is issued. This would be indicated by the DRIVE BUSY bits in the Main Status Register being set. If the stepping completes successfully, DRIVE BUSY in the MSR will go inactive and then the READ DATA command starts. If the SEEK failed, the status bytes of the READ DATA command are sent to the FIFO. ST0 contains the cause of the error and C would contain the cylinder number on which the stepping failed.

READ DATA COMMAND

A set of nine (9) bytes is required to place the 82072 into the Read Data Mode. After the READ DATA command has been issued, the 82072 loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the SPECIFY command), and begins reading ID Address Marks and ID fields. When the sector address read off the diskette matches with the sector address specified in the command, the 82072 reads the sector's data field and transfers the data to the FIFO.

After completion of the read operation from the current sector, the sector address is incremented by one, and the data from the next logical sector is read and output via the FIFO. This continuous read function is called "Multi-Sector Read Operation". Upon receipt of TC, or an implied TC (FIFO overrun/underrun), the 82072 stops sending data, but will continue to read data from the current sector, check the CRC bytes, and at the end of the sector terminate the READ DATA Command.

N determines the number of bytes per sector (See Table 8A Below). If N is set to zero, the sector size is set to 128. The DTL value determines the number of bytes to be transferred. If DTL is less than 128, the 82072 transfers the specified number of bytes to the host. For reads, it continues to read the entire 128 byte sector and checks for CRC errors. For writes it completes the 128 byte sector by filling in zeroes. If N is not set to 00 Hex, DTL should be set to FF Hex, and has no impact on the number of bytes transferred.

Table 8A. Sector Sizes

N	Sector Size
00	128 bytes
01	256 bytes
02	512 bytes
03	1024 bytes
..	...
07	16 Kbytes

The amount of data which can be handled with a single command to the 82072 depends upon MT (multi-track) and N (Number of bytes/sector).

MT	N	Max. Transfer Capacity	Final Sector Read from Disk
0	1	256 × 26 = 6,656	26 at side 0 or 1
1	1	256 × 52 = 13,312	26 at side 1
0	2	512 × 15 = 7,680	15 at side 0 or 1
1	2	512 × 30 = 15,360	15 at side 1
0	3	1024 × 8 = 8,192	8 at side 0 or 1
1	3	1024 × 16 = 16,384	16 at side 1

The Multi-Track function (MT) allows the 82072 to read data from both sides of the diskette. For a particular cylinder, data will be transferred starting at Sector 1, Side 0 and completing at the last sector of the same track at Side 1.

At the completion of the DATA Command, the head is not unloaded until after the Head Unload Time Interval (specified in the SPECIFY command) has elapsed. If the host issues another command before the head unloads then the head settling time may be saved between subsequent reads.

If the 82072 detects a pulse on the IDX pin twice without finding the specified sector (meaning that the diskette's index hole passes through index detect logic in the drive twice), the 82072 sets the IC code in Status Register 0 to "01" (Abnormal termination), and sets the ND bit in Status Register 1 to "1", and terminates the DATA Command.

After reading the ID and Data Fields in each sector, the 82072 checks the CRC bytes. If a CRC error occurs in the ID or data field, the 82072 sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the DE bit flag in Status Register 1 to "1", sets the DD bit in Status Register 2 to "1" if CRC is incorrect in the ID field, and terminates the READ DATA Command.

If the SK bit is "0" and the 82072 reads a Deleted Data Address Mark from the diskette, it sets the CM bit in Status Register 2 to "1", and terminates the READ DATA Command, after reading all the data in the sector. If SK is "1", the 82072 skips the sector with the Deleted Data Address Mark, sets the CM bit in Status Register 2 to a "1", and continues to read the next sector.

WRITE DATA

After the WRITE DATA command has been issued, the 82072 loads the head (if it is in the unloaded

state), waits the specified head load time if unloaded (defined in the SPECIFY command), and begins reading ID Fields. When the sector address read from the diskette matches the sector address specified in the command, the 82072 reads the data from the host via the FIFO, and writes it to the sector's data field.

After writing data into the current sector, the Sector Number stored in "R" is incremented by one, and the 82072 continues writing to the next data field. The 82072 continues this "Multi-Sector Write Operation". Upon receipt of a terminal count signal or if a FIFO over/under run occurs while a data field is being written, then the remainder of the data field is filled with zeros.

The 82072 reads the ID field of each sector and checks the CRC bytes. If it detects a CRC error in one of the ID Fields, it sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the DE bit of Status Register 1 to "1", and terminates the WRITE DATA command.

The WRITE DATA command operates in much the same manner as the READ DATA command. The following items are the same. Please refer to the READ DATA Command for details:

- Transfer Capacity
- EN (End of Cylinder) bit
- ND (No Data) bit
- Head Load, Unload Time Interval
- ID information when the host terminates the command.
- Definition of DTL when $N = 0$ and when N does not = 0.

READ DELETED DATA

This command is the same as the READ DATA command except that when the 82072 detects a Deleted Data Address Mark at the beginning of a Data Field and $SK = "0"$, it will read all the data in the sector and set the CM bit in Status Register 2 to "1", and terminate the command. If $SK = "1"$, then the 82072 skips the sector with the Deleted Data Address Mark and reads the next sector. Thus the SK

bit can be used to cause the 82072 to read only the deleted data sectors during a read operation.

WRITE DELETED DATA

This command is almost the same as the WRITE DATA command except that a Deleted Data Address Mark is written at the beginning of the Data Field instead of the normal Data Address Mark. This command is typically used to mark a bad sector containing an error on the floppy disk.

READ TRACK

This command is similar to the READ DATA command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering a pulse on the IDX pin, the 82072 starts to read all data fields on the track as continuous blocks of data without regard to logical sector numbers. If the 82072 finds an error in the ID or DATA CRC check bytes, it continues to read data from the track and sets the appropriate error bits at the end of the command. The 82072 compares the ID information read from each sector with the specified value in the command, and sets the ND flag of Status Register 1 to a "1" if there is no comparison. Multi-track or skip operations are not allowed with this command.

This command terminates when the EOT specified number of sectors have been read. If the 82072 does not find an ID Address Mark on the diskette after the second occurrence of a pulse on the IDX pin, then it sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the MA bit in Status Register 1 to "1", and terminates the command.

REGISTER INFORMATION FOR HOST TERMINATION

If the host terminates a read or write operation in the 82072, then the ID information in the result phase is dependent upon the state of the MT bit and EOT byte.

MT	Head	Final Sector Transferred to Host	ID Information at Result Phase			
			C	H	R	N
0	0	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	NC	01	NC
	1	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	NC	01	NC
1	0	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	NC	LSB	01	NC
	1	Less than EOT	NC	NC	R+1	NC
		Equal to EOT	C+1	LSB	01	NC

NC: no change, the same value as the one at the beginning of command execution.
 LSB: least significant bit, the LSB of H is complemented.

FORMAT COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks		
Command	W	0	MFM	0	0	1	1	0	1	Command Code		
	W	0	0	0	0	0	HDS	DS1	DS0			
	W	-----				N	-----					
	W	-----				SC	-----					
	W	-----				GPL	-----					
Execution	W	-----				D	-----				Data Transfer Occurs	
	Result	R	-----				ST0	-----				
		R	-----				ST1	-----				
		R	-----				ST2	-----				
R		-----				C	-----					
Result	R	-----				H	-----					
	R	-----				R	-----					
	R	-----				N	-----					

The FORMAT command allows an entire track to be formatted. After a pulse from the IDX pin is detected, the 82072 starts writing data on the disk including Gaps, Address Marks, ID Fields and Data Fields, per the IBM System 34 or 3740 format (MFM or FM respectively). The particular values that will be written to the gap and data field are controlled by the values programmed into N, SC, GPL, and D which are specified by the host during the command phase. The data field of the sector is filled with the data byte specified by D. The ID Field for each sector is supplied by the host; that is, four data bytes per sector are needed by the 82072 for C, H, R, and N (cylinder, head, sector number and sector size respectively).

After formatting each sector, the host must send new values for C, H, R and N to the 82072 for the next sector on the track. The R value (sector number) is the only value that must be changed by the host after each sector is formatted. This allows the disk to be formatted with nonsequential sector addresses (interleaving). This incrementing and formatting continues for the whole track until the 82072 encounters a pulse on the IDX pin again and it terminates the command.

Table 9 contains typical values for gap fields which are dependent upon the size of the sector and the number of sectors on each track. Actual values can vary due to drive electronics.

Table 9. Typical Values for Formatting

	Sector Size	N	SC	GPL1	GPL2
8" Drives	128	0	1A	07	1B
	256	1	0F	0E	2A
	512	2	08	1B	3A
	1024	3	04	47	8A
	2048	4	02	C8	FF
	4096	5	01	C8	FF
5.25" Drives	256	1	1A	0E	36
	512	2	0F	1B	54
	1024	3	08	35	74
	2048	4	04	99	FF
	4096	5	02	C8	FF
3.5" Drives	128	00	12	07	09
	128	00	10	10	19
	512	02	08	18	30
	1024	03	04	46	87
	2048	04	02	C8	FF
	4096	05	01	C8	FF
			
	256	01	12	0A	0C
	256	01	10	20	32
	512	02	08	2A	50
1024	03	04	80	F0	
2048	04	02	C8	FF	
4096	05	01	C8	FF	
...	...				
8" Drives	128	0	0F	07	1B
	256	1	09	0F	2A
	512	2	05	1B	3A
5.25" Drives	256	1	0F	0E	36
	512	2	09	1B	54
	1024	3	05	35	74

GPL1 = suggested GPL values in read and write commands to avoid splice point between data field and ID field of contiguous sections.

GPL2 = suggested GPL value in FORMAT TRACK command.

NOTE:

All values are in Hex.

GAP 4a	SYNC	IAM		GAP 1	SYNC	IDAM		C	H	S	N	C	GAP 2	SYNC	DATA AM		DATA	C	GAP 3	GAP 4b
80x 4E	12x 00	3x C2	FC	50x 4E	12x 00	3x A1	FE	Y	D	E	O	R	22x 4E	12x 00	3x A1	FB				

Figure 12. System 34 Format Double Density

GAP 4a	SYNC	IAM		GAP 1	SYNC	IDAM		C	H	S	N	C	GAP 2	SYNC	DATA AM		DATA	C	GAP 3	GAP 4b
40x FF	6x 00	3x C2	FC	26x FF	6x 00	3x A1	FE	Y	D	E	O	R	11x FF	6x 00	3x A1	FB or F8				

Figure 13. System 3740 Format Single Density

CONTROL COMMANDS

Each of the control commands has a unique set of parameters that are sent and returned. Control commands differ in that they do not have a data transfer in the execution phase.

READ ID COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks	
Command	W	0	MFM	0	0	1	0	1	0	Command Code	
	W	0	0	0	0	0	HDS	DS1	DS0		
Execution										1ST ID Field Located.	
Result	R	-----				ST0	-----				Status After Command Has Completed.
	R	-----				ST1	-----				Disk Status After The Command Has Completed.
	R	-----				ST2	-----				
	R	-----				C	-----				
	R	-----				H	-----				
	R	-----				R	-----				
	R	-----				N	-----				

The READ ID command is used to find the present position of the recording heads. The 82072 stores the values from the first ID Field it is able to read into its registers. If the 82072 does not find an ID Address Mark on the diskette after the second occurrence of a pulse on the IDX pin, it then sets the IC code in Status Register 0 to "01" (Abnormal termination), sets the MA bit in Status Register 1 to "1", and terminates the command.

CONTROL COMMANDS WITH INTERRUPTS

The following commands will generate an interrupt upon completion. They do not return any result bytes. It is highly recommended that control commands be followed by the SENSE INTERRUPT STATUS command. Otherwise, valuable interrupt status information will be lost.

RECALIBRATE COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
Command	W	0	0	0	0	0	1	1	1	Command Code
	W	0	0	0	0	0	0	DS1	DS0	
Execution										Head Retracts To Track 0.
										Interrupt

This command causes the read/write head within the FDD to retract to the track 0 position. The 82072 clears the contents of the PCN counter, and checks the status of the TRK0 pin from the FDD. As long as the TRK0 pin is low, the DIR pin remains "1" and step pulses are issued. When the TRK0 pin goes high, the SE bit in Status Register 0 is set to "1", and the command is terminated. If the TRK0 pin is still low after 256 step pulses have been issued, the 82072 sets the SE and the EC bits of Status Register 0 to "1", and terminates the command.

The RECALIBRATE command does not have a result phase. SENSE INTERRUPT STATUS command must be issued after the RECALIBRATE command to effectively terminate it and to provide verification of the head position (PCN). During the command phase of the recalibrate operation, the 82072 is in the BUSY state, but during the execution phase it is in a NON BUSY state. At this time another RECALIBRATE command may be issued, and in this manner, parallel RECALIBRATE operations may be done on up to 4 drives at once.

Upon power up, the software must issue a RECALIBRATE command to properly initialize all drives and the controller.

SEEK COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
Command	W	0	0	0	0	1	1	1	1	Command Code
	W	0	0	0	0	0	HDS	DS1	DS0	
Execution	W	----- NCN -----				-----				Head is Moved to Specified Track.

The read/write head within the FDD is moved from track to track under the control of the SEEK Command. The 82072 compares the PCN which is the current head position with the NCN and performs the following operation if there is a difference:

- PCN < NCN: Direction signal to FDD set to "1" (step in), and issues step pulses.
- PCN > NCN: Direction signal to FDD set to "0" (step out), and issues step pulses.

The rate at which step pulses are issued is controlled by SRT (Stepping Rate Time) in the SPECIFY command. After each step pulse is issued, NCN is compared against PCN, and when NCN = PCN, then the SE bit in Status Register 0 is set to "1", and the command is terminated.

During the command phase of the seek or recalibrate operation, the 82072 is in the BUSY state, but during the execution phase it is in the NON BUSY state. At this time another SEEK or RECALIBRATE command may be issued, and in this manner, parallel seek operations may be done on up to 4 drives at once.

If an FDD is in a NOT READY state at the beginning of the command execution phase or during the seek operation, then the NR bit in Status Register 0 is set to "1", and the command is terminated.

Note that if implied seek is not enabled, the read and write commands should be preceded by:

- 1) SEEK command; Step to the proper track
- 2) SENSE INTERRUPT STATUS command; Terminate the Seek command
- 3) READ ID. Verify head is on proper track
- 4) Issue READ/WRITE command.

The SEEK command does not have a result phase. Therefore, it is highly recommended that the SENSE INTERRUPT STATUS Command after the SEEK command be issued to effectively terminate it and to provide verification of the head position (PCN). When exiting POWERDOWN mode, the 82072 saves the current PCN values. The status information, however, is cleared. Prior to issuing the POWERDOWN command, it is highly recommended that the user service all pending interrupts through the SENSE INTERRUPT STATUS command.

CONTROL COMMANDS WITHOUT INTERRUPTS

These commands have no execution phase and do not generate an interrupt upon completion. Their primary use is to follow one of the commands that do not have an execution phase, although they may be issued whenever the 82072 is in the command phase.

SENSE INTERRUPT STATUS COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
Command	W	0	0	0	0	1	0	0	0	Command Code
Result	R	-----				ST0	-----			
	R	-----				PCN	-----			

An interrupt signal on INT pin is generated by the 82072 for one of the following reasons:

1. Upon entering the Result Phase of:
 - a. READ DATA Command
 - b. READ TRACK Command
 - c. READ ID Command
 - d. READ DELETED DATA Command
 - e. WRITE DATA Command
 - f. FORMAT TRACK Command
 - g. WRITE DELETED DATA Command
2. RDY pin changes state
3. End of SEEK or RECALIBRATE Command
4. 82072 requires a data transfer during the execution phase in the non-DMA Mode

The SENSE INTERRUPT STATUS command resets the interrupt signal and via the IC code and SE bit of Status Register 0, identifies the cause of the interrupt.

Table 10. Interrupt Identification

SE	IC	Interrupt Due To
0	11	RDY pin changes state
1	00	Normal Termination of SEEK or RECALIBRATE command
1	01	Abnormal Termination of SEEK or RECALIBRATE command

Both the SEEK and the RECALIBRATE commands have no result phase. SENSE INTERRUPT STATUS command must be issued immediately after these commands to terminate them and to provide verification of the head position (PCN). If a SENSE INTERRUPT STATUS is not issued, the drive, will continue to be BUSY and may effect the operation of the next command.

SENSE DRIVE STATUS

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks	
Command	W	0	0	0	0	0	1	0	0	Command Code	
	W	MOT	0	0	0	0	HD	DSI	DS0		
Result	R	-----				ST3	-----				

SENSE DRIVE STATUS obtains drive status information. It has no execution phase and goes directly to the result phase from the command phase. STATUS REGISTER 3 contains the drive status information. If Bit 7 of the second command byte is set (MOT), there is no delay before the drive is accessed for status information. If Bit 7 is not set, the 82072 waits the motor ON/OFF delay period before accessing the drive.

SPECIFY COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks	
Command	W	0	0	0	0	0	0	1	1	Command Code	
	W	<-----SRT----->				<-----HUT----->					
Execution	W	<-----				HLT	>-----				ND
										Values Placed Into Regs.	

The SPECIFY command sets the initial values for each of the three internal timers. The HUT (Head Unload Time) defines the time from the end of the execution phase of one of the read/write commands to the head unload state. The SRT (Step Rate Time) defines the time interval between adjacent step pulses. The HLT (Head Load Time) defines the time between the Head Load signal goes high and the read, write operation starts. The values change with the data rate speed selection and are documented in Table 11. The values are the same for MFM and FM.

Table 11. Drive Control Delays

	HUT				SRT			
	1M	500K	300K	250K	1M	500K	300K	250K
00	128	256	512	1024				
01	8	16	26.7	32	8.0	16	26.7	32
02	16	32	53.3	64	7.5	15	25	30
..
0E	112	224	373	448	1.0	2	3.33	4
0F	120	240	400	480	0.5	1	1.67	2

	HLT			
	1M	500K	300K	250K
00	128	256	512	1024
01	1	2	3.3	4
02	2	4	6.7	8
..
7F	126	252	420	504
7F	127	254	423	508

The choice of DMA or NON-DMA operations is made by the ND bit. When this bit is "1", the NON-DMA mode is selected, and when ND is "0", the DMA mode is selected. In DMA mode, data transfers are signalled by the DRQ pin. Non-DMA mode uses the RQM bit and the INT pin to signal data transfers.

NEW CONTROL COMMANDS

CONFIGURE COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
Command	W	0	0	0	1	0	0	1	1	Configuration Information
	W	HSDA	<--- MOFF --->			<--- MON --->				
	W	0	EIS	EFIFO	POLL	<--- FIFOTHR --->				
	W	<--- PRETRK --->								

Issued to select the special features of the 82072. A CONFIGURE command need not be issued if the default values of the 82072 meet the system requirements.

CONFIGURE DEFAULT VALUES:

- EIS —No Implied Seeks
- EFIFO —FIFO Enabled
- POLL —Polling Disabled
- FIFOTHR —FIFO Threshold Set to 1 Byte
- PRETRK —Pre-Compensation Set to Track 0
- HSDA —Disabled
- MOFF —Set to Motor Off Delay of 5.2 Seconds
- MON —Set to 1.0 Second

EIS

Enable implied seek. When set to "1", the 82072 will perform a SEEK operation before executing a read or write command. Defaults to no implied seek.

EFIFO

A "1" puts the FIFO into the 8272A compatible mode where the FIFO is disabled. This means data transfers are asked for on a byte by byte basis. Defaults to "0", FIFO enabled. The threshold defaults to one.

POLL

Disable polling of the drives. Defaults to "1", polling disabled. When enabled, each of the drives is selected for a period of time, and its RDY pin sampled. After a delay, the next drive is selected. This sequence occurs whenever the 82072 is in the idle state. An interrupt is generated if the 82072 detects a change in the drive RDY signal. No polling is performed while the drive head is loaded or until the unload head delay has expired.

FIFOTHR

The FIFO threshold in the execution phase of read or write commands. This is programmable from 1 to 16 bytes. Defaults to one byte.

00 = 1 byte
01 = 2 bytes
02 = 3 bytes
...
0F = 16 bytes

PRETRK

Pre-compensation start track number. Programmable from track 0 to 255. Defaults to track 0.

00 = TRACK 0
01 = TRACK 1
02 = TRACK 2
...
FE = TRACK 254
FF = TRACK 255

HSDA

High Speed Disk Adjust causes the motor on/off delays to be doubled. This is necessary for disks that rotate at high rates (i.e., 600 RPM vs 300 RPM). Defaults to a "0" which is disabled.

MOFF

Programs the number of index pulses to be counted before the MOTOR pin is deactivated (if high). All 82072 commands restart the motor off timer upon their completion. The MOFF bits increment the count value by 4. The HSDA bit doubles that value for disks that spin extremely fast. This chart is simplified to illustrate the count delays. All combinations are legal and no checking is done by the 82072.

The motor off delay is ignored while any drive is BUSY.

Defaults to 110 (5.2 sec. at 300 RPM).

MOFF Bits	HSDA	Revolutions	Delay			SEC.
			300 RPM	360 RPM	600 RPM	
000	0	2	0.4	0.33	0.2	SEC.
000	1	4	—	—	0.4	
001	0	6	1.2	1.0	0.6	
001	1	12	—	—	1.2	
010	0	10	2.0	1.67	1.0	
010	1	20	—	—	2.0	
•	•	•	•	•	•	
•	•	•	•	•	•	
110	0	26	5.2	4.33	2.6	
110	1	52	—	—	5.2	
111	0	30	6.0	5.0	3.0	
111	1	60	—	—	6.0	

MON

Programs the number of Index pulses that are counted after the MOTOR pin goes active before a command can proceed into the execution phase. This is to give the floppy drive time to spin up and stabilize at the proper speed. If MOTOR is already active, no delay will be generated even if the drive selection is changed. The MON bits increment the index count by one and the HSDA bit will double that value if active. All combinations are legal and no checking is done by the 82072.

Defaults to 101 (1.0 sec. at 300 RPM).

MON	HSDA	Motor on to Command Execution Delay				SECS.
		Revolutions	300 RPM	360 RPM	600 RPM	
0000	NO Delay to Command, MOTOR OFF is Infinite MOTOR Pin must be given a Command to go On/Off.					SECS.
0001	0	1	0.2	0.17	0.1	
0001	1	2	—	—	0.2	
0010	0	2	0.4	0.33	0.2	
0010	1	4	—	—	0.4	
0011	0	3	0.6	0.5	0.3	
0011	1	6	—	—	0.6	
:	:	:	:	:	:	
1111	0	15	3.0	2.5	1.5	
1111	1	30	—	—	3.0	

NOTE:

Actual index counts depend upon the position of the disk and when the command begins processing. There is an error rate of -1/+0 revolutions.

The MOTOR logic provides a programmable motor enable signal to the FDDs. This is to allow the drive time to spin up the floppy and stabilize before attempting a read or write. Most floppy drives do not require the drive to be selected (through the DSx line) or to be ready. The application should qualify the MOTOR pin with the drive select outputs. The 82072 will activate the appropriate drive select outputs along with the MOTOR pin. Figure 14 is an example of the interface logic. If a drive is not currently selected or the MOTOR signal is not on, DS and MOTOR are activated the programmed time before the drive is accessed.

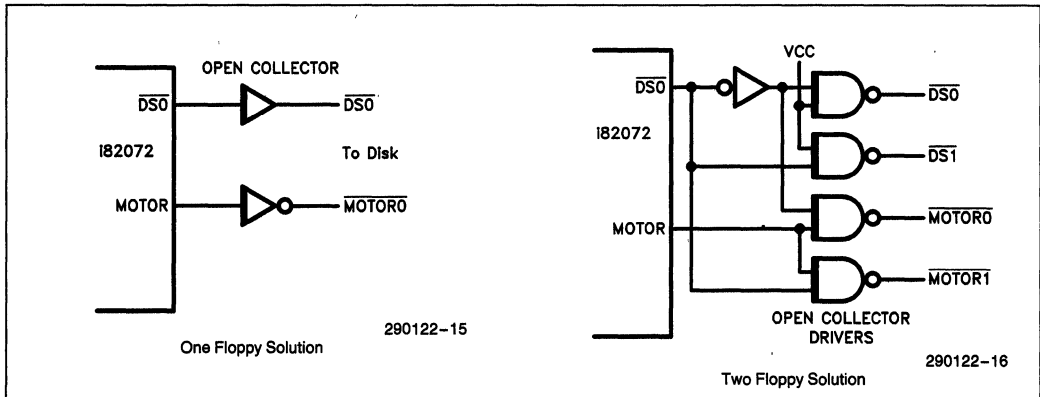


Figure 14. MOTOR and Drive Select Interface

MOTOR ON/OFF COMMAND

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
Command	W	ON/OFF	DS1	DS0	0	1	0	1	1	Command Code

ON/OFF

Determines whether the command sets the MOTOR pin high or low independent of the internal timer. A "0" takes the MOTOR pin low immediately and a "1" takes the pin to a high. The 82072 returns to the command phase.

The internal timer takes drive changes into account. When switching between drives, the MOFF time will be ignored so that the newly selected drive can be accessed as soon as the MOTOR ON delay has expired. The MOTOR ON/OFF command allows the user to override the programmed delays. The typical implementation would not use the MOTOR ON/OFF command to enable the drive motor because the delays would be the responsibility of the host system.

RELATIVE SEEK

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks
Command	W	1	DIR	0	0	1	1	1	1	
	W	0	0	0	0	0	HDS	DS1	DS0	
	W	<----- RCN ----->								

NOTE:

The command is coded the same as for SEEK, except for the MSB of the first byte and the DIR bit.

DIR Head Step Direction Control.

DIR	Action
0	Step Head Out
1	Step Head In

RCN Relative Cylinder Number that determines how many tracks to step the head in or out from the current track number.

The RELATIVE SEEK command differs from the SEEK command in that it steps the head the absolute number of tracks specified in the command instead of making a comparison against an internal register. The SEEK command is good for drives that have a maximum of 256 tracks. RELATIVE SEEKS cannot be overlapped with other RELATIVE SEEKS. Only one RELATIVE SEEK can be active at a time. RELATIVE SEEKS may be overlapped with SEEKS and RECALIBRATES. Bit 4 of Status Register 0 (EC) will be set if RELATIVE SEEK attempts to step outward beyond Track 0.

As an example, assume that a floppy drive has 300 useable tracks and that the host needs to read track 300 and the head is on any track (0-255). If a SEEK command was issued, the head would stop at track 255. If a RELATIVE SEEK command was issued, the 82072 would move the head the specified number of tracks, regardless of the internal cylinder position register (but would increment the register). If the head had been on track 40 (D), the maximum track that the 82072 could position the head on using RELATIVE SEEK, would be 296 (D), the initial track, + 256 (D). The maximum count that the head can be moved with a single RELATIVE SEEK command is 256 (D).

The internal register, PCN, would overflow as the cylinder number crossed track 255 and would contain 40 (D). Functionally, the 82072 starts counting from 0 again as the track number goes above 255(D). It is the users responsibility to compensate 82072 functions (precompensation track number) when accessing tracks greater than 255. The 82072 does not keep track that it is working in an "extended track area" (greater than 255). Any command issued would use the current PCN value except for the RECALIBRATE command which only looks for the TRACK0 signal. RECALIBRATE would return an error if the head was farther than 255 due to its limitation of issuing a maximum 256 step pulses. The user simply needs to issue a second RECALIBRATE command. The SEEK command and implied seeks will function correctly within the 44 (D) track (299-255) area of the "extended track area". It is the users responsibility not to issue a new track position that would exceed the maximum track that is present in the extended area.

To return to the standard floppy range (0-255) of tracks, a RELATIVE SEEK would be issued to cross the track 255 boundary.

A RELATIVE SEEK can be used instead of the normal SEEK but the host is required to calculate the difference between the current head location and the new (target) head location. This may require the host to issue a READ ID command to ensure that the head is physically on the track that software assumes it to be. Different 82072 commands will return different cylinder results which may be difficult to keep track of with software without the SCAN ID command.

DUMPREG

Phase	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Remarks		
Command	W	0	0	0	0	1	1	1	0	*Note		
Execution										Registers Placed into FIFO		
Result	R	_____				PCN-Drive 0	_____				See Note	
	R	_____				PCN-Drive 1	_____					
	R	_____				PCN-Drive 2	_____					
	R	_____				PCN-Drive 3	_____					
	R	< --- SRT --- >					< --- HUT --- >					
	R	< --- HLT ---					> ND					
	R	< --- SC/EOT ---				>						
	R	HSDA < --- MOFF --- >		< --- MON --- >								
R	< --- PRETRK ---				>							
R		0	EIS	EFIFO	POLL	< --- FIFOTHR --- >						

NOTE:

SC is returned if the last command that was issued was the Format command. EOT is returned if the last command issued was a Read or Write.

The DUMPREG command is designed to support system run-time diagnostics and application software development and debug.

STATUS REGISTER ENCODING

The contents of these registers are available only through a command sequence.

Status Register 0

Bit No.	Symbol	Name	Description
7, 6	IC	Interrupt Code	00-Normal termination of command. The specified command was properly executed and completed without error. 01-Abnormal termination of command. Command execution was started, but was not successfully completed. 10-Invalid command. The requested command could not be executed. 11-Abnormal termination. During the command execution, the RDY signal changed state.
	SE	Seek End	The 82072 completed a SEEK command or a READ or WRITE with implied seek command.
4	EC	Equipment Check	The TRK0 pin failed to become a "1" after 256 step pulses in the RECALIBRATE command.
3	NR	Not Ready	RDY pin became a "0" while executing a read, write, or seek command.
2	H	Head Address	The current head address.
1, 0	DS1, 0	Drive Select	The current selected drive.

Status Register 1

Bit No.	Symbol	Name	Description
7	EN	End of Cylinder	The 82072 tried to access a sector beyond the final sector of the track (255D).
6	—	—	Unused. This bit is always "0".
5	DE	Data Error	The 82072 detected a CRC error in either the ID field or the data field of a sector.
4	OR	Overrun/Underrun	Becomes set if the 82072 does not receive CPU or DMA service within the required time interval, resulting in data overrun or underrun.
3	—	—	Unused. This bit is always "0".
2	ND	No Data	Any one of the following: 1. READ DATA, READ DELETED DATA command, the 82072 did not find the specified sector. 2. READ ID command, the 82072 cannot read the ID field without an error. 3. READ TRACK command, the 82072 cannot find the starting sector specified.
1	NW	Not Writable	WP pin became a "1" while the 82072 is executing a WRITE DATA, WRITE DELETED DATA, or FORMAT TRACK command.
0	MA	Missing Address Mark	Any one of the following: 1. The 82072 did not detect an ID address mark at the specified track after encountering the index pulse from the IDX pin twice. 2. The 82072 cannot detect a data address mark or a deleted data address mark on the specified track.

Status Register 2

Bit No.	Symbol	Name	Description
7	—	—	Unused. This bit is always "0".
6	CM	Control Mark	Any one of the following: 1. READ DATA command, the 82072 encounters a deleted data address mark. 2. READ DELETED DATA command, the 82072 encounters a data address mark.
5	DD	Data Error in Data Field.	The 82072 detected a CRC error in the data field.
4	WC	Wrong Cylinder	The track address from the sector ID field is different from the track address maintained inside the 82072. With implied seek, this bit will not be set.
3	—	—	Unused. This bit is always "0".
2	—	—	Unused. This bit is always "0".
1	BC	Bad Cylinder	The track address from the sector ID field is different from the track address maintained inside the 82072 and is equal to FF hex which indicates a bad track with a hard error according to the IBM soft-sectored format.
0	MD	Missing Data Address Mark	The 82072 cannot detect a data address mark or a deleted data address mark.

Status Register 3

Bit No.	Symbol	Name	Description
7	—	—	Unused. This bit is always "0".
6	WP	Write Protected	Indicates the status of the WP pin.
5	RDY	Ready	Indicates the status of the RDY pin.
4	T0	TRACK 0	Indicates the status of the TRK0 pin.
3	—	—	Unused. This bit is always "1".
2	HD	Head Address	Indicates the status of the HDSEL pin.
1, 0	DS1, 0	Direct Select	Indicates the status of the DS1, DS0 pins.

D.C. SPECIFICATION

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65°C to +150°C
Supply Voltage	-0.5 to +8.0V
Supply Voltage	-0.5 to +8.0V
Voltage on Any Input	GND - 2V to 6.5V
Voltage on Any Output	..GND -	0.5V to VCC+0.5V
Power Dissipation	1 Watt

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

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$ to $= 70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 10\%$, $V_{SS} = AV_{SS} = 0\text{V}$

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Input Low Voltage, X1	-0.5	0.8	V	
V_{IHC}	Input High Voltage, X1	3.9	$V_{CC} + 0.5$	V	
V_{IL}	Input Low Voltage (all pins except X1)	-0.5	0.8	V	
V_{IH}	Input High Voltage (all pins except X1)	2.0	$V_{CC} + 0.5$	V	
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = 2.5\text{ mA}$
V_{OH}	Output High Voltage	3.0 $V_{CC} - 0.4$		V V	$I_{OH} = -2.5\text{ mA}$ $I_{OH} = 100\ \mu\text{A}$
I_{CC}	V_{CC} Supply Current (Digital and Analog)		tbd tbd 20 35	mA mA mA mA	(Note 1, 3, 5) (Note 1, 4, 5) (Note 2, 3, 5) (Note 2, 3, 5)
I_{SSA}	Analog Supply Current		5	mA	
I_{CCSB}	I_{CC} in Powerdown		125	μA	(Note 3, 5)
I_{SSASB}	Analog I_{CC} in Powerdown		25	μA	(Note 3)
I_{IL}	Input Load Current (all input pins)		10 -10	μA μA	$V_{IN} = V_{CC}$ $V_{IN} = 0\text{V}$
I_{OFL}	Data Bus Output Float Leakage		± 10	μA	$0.45 < V_{OUT} < V_{CC}$

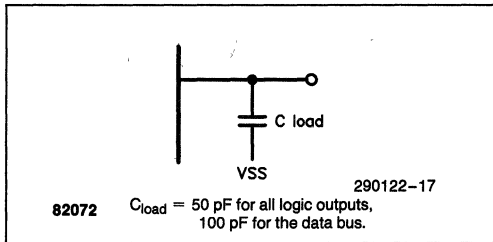
NOTES:

1. Tested at the 1 Mbps data rate.
2. Tested at the 500 kbps data rate.
3. $V_{IL} = V_{SS}$, $V_{IH} = V_{CC}$; Outputs not connected to D.C. loads.
4. $V_{IL} = 0.8$, $V_{IH} = 2.0$; Outputs not connected to D.C. loads.
5. Each data bus input that is floated can increase the I_{CC} by about 1.5 mA if the 82072 is not in the POWER DOWN mode. These are the only inputs that may be floated.

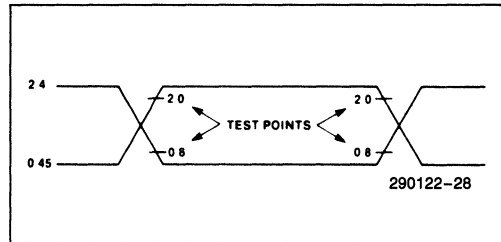
Capacitance

C_{IN}	Input Capacitance	10	pF	All Pins Except
C_{IN1}	Clock Input Capacitance	20	pF	Pins Under Test
$C_{I/O}$	Input/Output Capacitance	20	pF	Tied to AC Ground

LOAD CIRCUIT



A. C. TESTING INPUT, OUTPUT WAVEFORM



A.C. SPECIFICATIONS

$T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = +5V \pm 10\%$, $V_{SS} = AV_{SS} = 0V$

Symbol	Parameter	Min	Max	Unit	Notes
CLOCK TIMINGS					
t_1	Clock Rise Time		10	ns	
	Clock Fall Time		10	ns	
t_2	Clock High Time	14	27	ns	
t_3	Clock Low Time	14	27	ns	
t_4	Clock Period	40	43	ns	
t_5	Internal Clock Period				(Note 3)
HOST READ CYCLES					
t_6	Deleted Specification				
t_7	Address Setup to \overline{RD}	5		ns	
t_8	\overline{RD} Pulse Width	100		ns	
t_9	Address Hold From RD	0		ns	
t_{10}	Data Valid From \overline{RD}		60	ns	
t_{11}	Command Inactive	60		ns	
t_{12}	Output Float Delay		35	ns	
t_{13}	INT Delay From RD		$t_5 + 125$	ns	
t_{14}	Data Hold From RD	5		ns	

A.C. SPECIFICATIONS (Continued)
 $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 10\%$, $V_{SS} = AV_{SS} = 0\text{V}$

Symbol	Parameter	Min	Max	Unit	Notes
HOST WRITE CYCLES					
t15	Address Setup to $\overline{\text{WR}}$	5		ns	
t16	$\overline{\text{WR}}$ Pulse Width	100		ns	
t17	Address Hold from WR	0		ns	
t18	Command Inactive	60		ns	
t19	Data Setup to WR	70		ns	
t20	Data Hold from WR	0		ns	
t21	INT Delay from WR		t5 + 125	ns	
DMA CYCLES					
t22	DRQ Cycle Period	6.5		μs	(Note 1)
t23	DACK to DRQ Inactive		100	ns	(Note 1)
t24	RD to DRQ Inactive		150	ns	(Note 4)
t25	DACK Setup to $\overline{\text{RD}}$, $\overline{\text{WR}}$	5		ns	
t26	DACK Hold from RD, WR	0		ns	
t27	DRQ to $\overline{\text{RD}}$, $\overline{\text{WR}}$ Active	0	6	μs	(Note 1)
t28	Terminal Count Width	2		t5	
t29	TC to DRQ Inactive		2.5 t5 + 100	ns	
RESET					
t30	Reset Width	170		t4	(Note 5)
t31	Reset to Control Inactive		2	μs	
WRITE DATA TIMING					
t32	Write Data Width	40		ns	82072-1
	Write Data Width	80		ns	82072
DRIVE CONTROL					
t33	DS0, 1 Setup to DIR	6	20	μs	
t34	DS0, 1 Hold from DIR	8		μs	
t35	DIR Setup to STEP	0.5	2	μs	
t36	DIR Hold from STEP	10		μs	
t37	STEP Active Time (High)	2.5		μs	
t38	STEP Cycle Time			μs	(Note 2)
t39	INDEX Pulse Width	5		t5	

A.C. SPECIFICATIONS (Continued)

 $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = +5V \pm 10\%$, $V_{SS} = AV_{SS} = 0V$

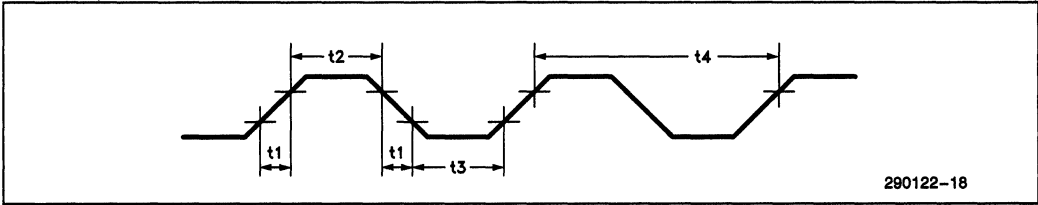
Symbol	Parameter	Min	Max	Unit	Notes
READ DATA TIMING					
t40	Read Data Pulse Width	40		ns	
t41	Window Setup to RDDATA	15		ns	
t42	Window Hold from RDDATA	15			
t43	Window Cycle Time	2		μs	FM
	Window Cycle Time	0.5		μs	MFM
f44	PLL Data Rate		1M	bit/s	82072-1
			500K	bit/s	82072
t44	Data Rate Period				1/f44
tLOCK	Lockup Time		48	t44	
tRANGE	Frequency Range	-6.0%	+6.0%	f44	
tJITTER	Noise Error	-17.5%	+17.5%	t44	

NOTES:

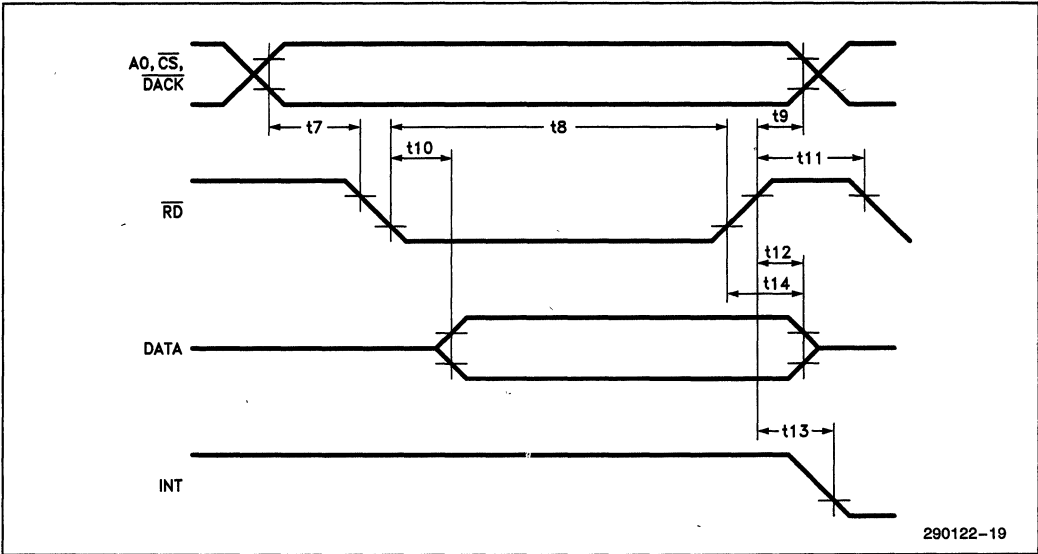
- This timing is for FIFO threshold = 1. When FIFO threshold is N bytes, the value of t22 should be multiplied by N and subtract 1.5 μs .
- This value can be ranged from 0.5 ms to 8.0 ms and is dependent upon data rate.
- Many timings are a function of the selected data rate. The nominal values for the internal clock period (t5) for the various data rates are:

1 Mbps	3 x oscillator period = 125 ns
500 Kbps	6 x oscillator period = 250 ns
300 Kbps	10 x oscillator period = 420 ns
250 Kbps	12 x oscillator period = 500 ns
- If $\overline{\text{DACK}}$ transitions before $\overline{\text{RD}}$, then this specification is ignored. If there is no transition on $\overline{\text{DACK}}$, then this becomes the DRQ inactive delay.
- Reset requires a stable oscillator to meet the minimum active period.

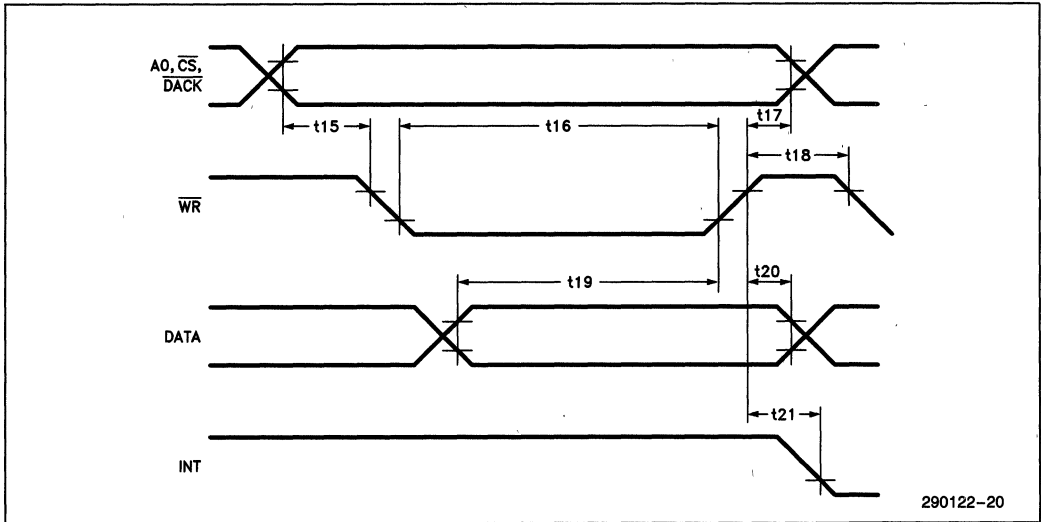
CLOCK TIMING



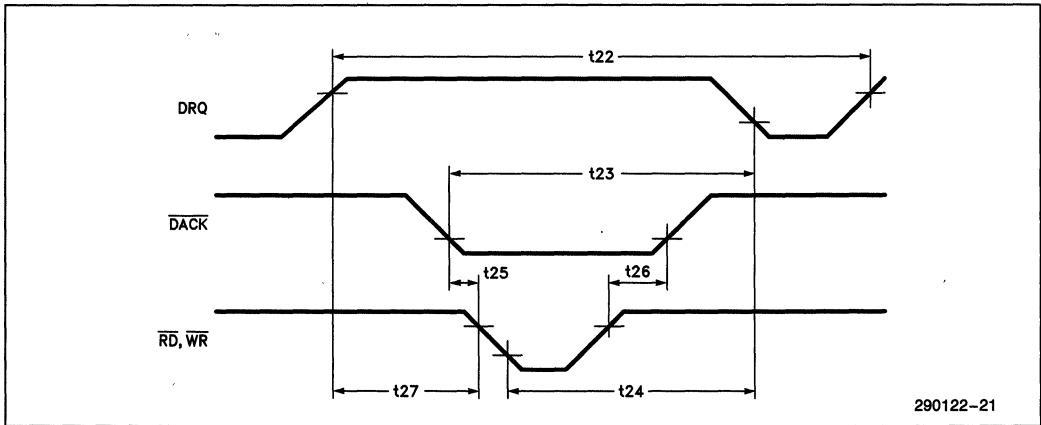
HOST READ CYCLES



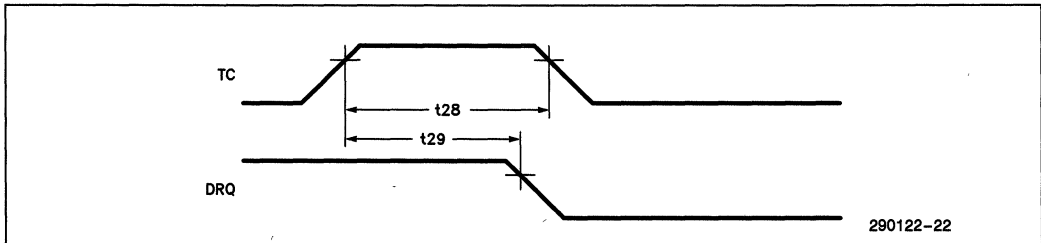
HOST WRITE CYCLES



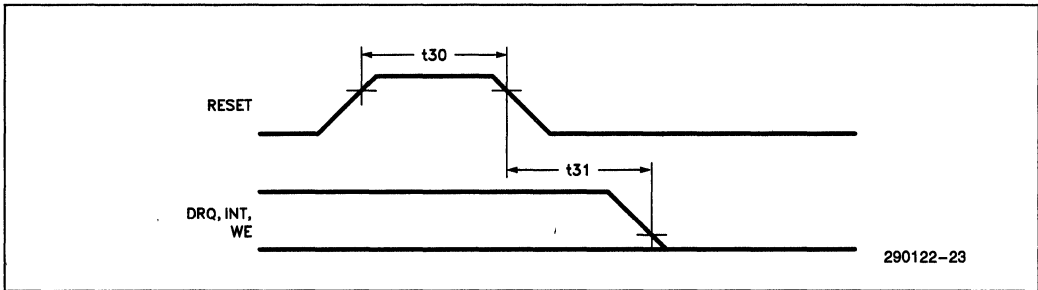
DMA CYCLES



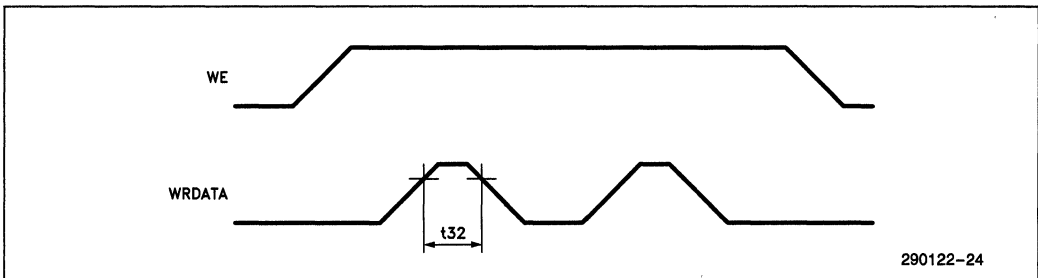
TERMINAL COUNT



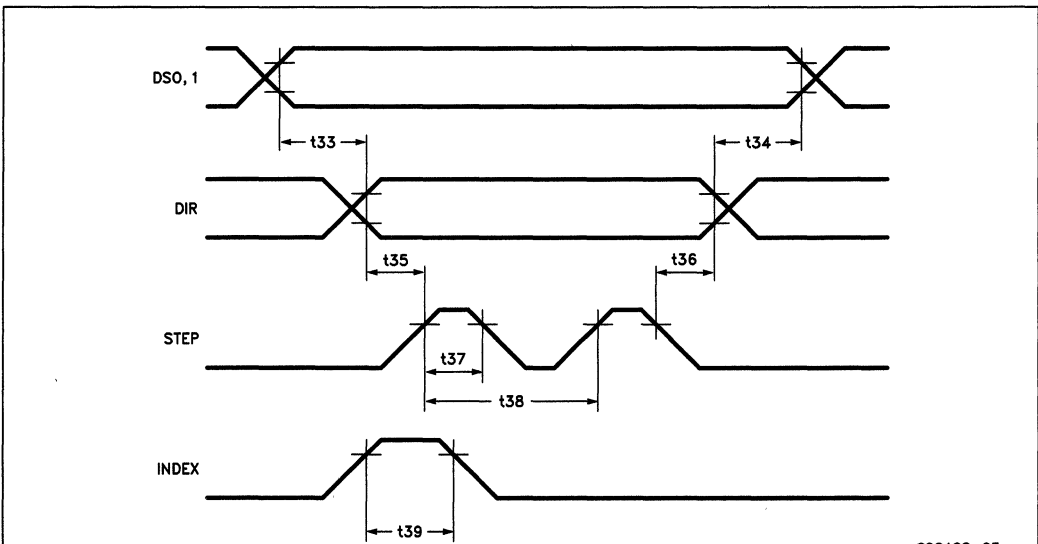
RESET



WRITE DATA TIMING



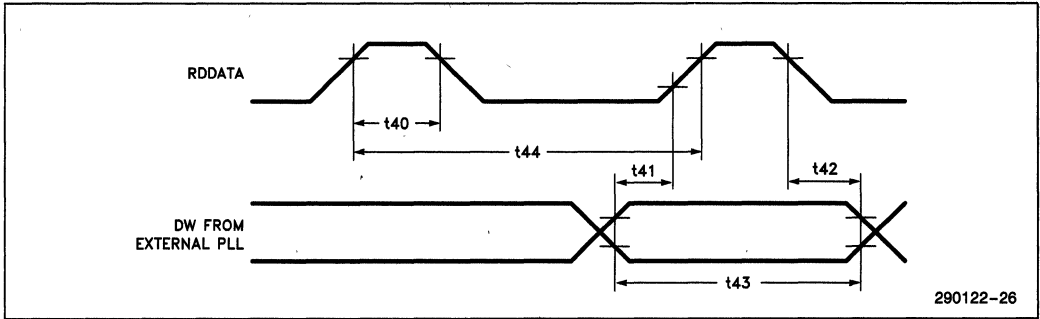
DRIVE CONTROL



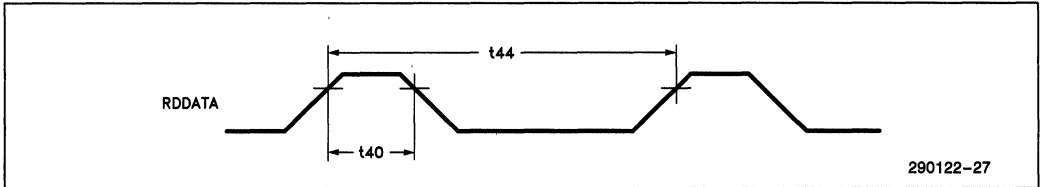
NOTE:

For overlapped seeks, only one step pulse per drive selection is issued. Non-overlapped seeks will issue all programmed step pulses.

READ DATA TIMING-EXTERNAL PLL



INTERNAL PLL





**APPLICATION
NOTE**

AP-116

November 1986

**An Intelligent Data Base System
Using the 8272**

TOM ROSSI
PERIPHERALS APPLICATIONS MANAGER

Order Number: 207875-001

1.0 INTRODUCTION

Most microcomputer systems in use today require low-cost, high-density removable magnetic media for information storage. In the area of removable media, a designer's choice is limited to magnetic types and floppy disks (flexible diskettes), both of which offer non-volatile data storage. The choice between these two technologies is relatively straight-forward for a given application. Since disk drives are designed to permit random access to stored information, they are significantly faster than tape units. For example, locating information on a disk requires less than a second, while tape movement (even at the fastest rewind or fast-forward speed) often requires several minutes. This random access ability permits the use of floppy disks in on-line storage applications (where information must be located, read, and modified/updated in real-time under program or operator control). Tapes, on the other hand, are ideally suited to archival or back-up storage due to their large storage capacities (more than 10 million bytes of data can be archived on a cartridge tape).

A sophisticated controller is required to capitalize on the abilities of the disk storage unit. In the past, disk controller designs have required upwards of 150 ICs. Today, the single-chip 8272 Floppy Disk Controller (FDC) plus approximately 30 support devices can handle up to four million bytes of on-line data storage on four floppy disk drives.

The Floppy Disk

A floppy disk is a circular piece of thin plastic material covered with a magnetic coating and enclosed in a pro-

TECTIVE JACKET (Figure 1). The circular piece of plastic revolves at a fixed speed (approximately 360 rpm) within its jacket in much the same manner that a record revolves at a fixed speed on a stereo turntable. Disks are manufactured in a variety of configurations for various storage capacities. Two standard physical disk sizes are commonly used. The 8-inch disk (8 inches square) is the larger of the two sizes; the smaller size (5¼ inches square) is often referred to as a mini-floppy. Single-sided disks can record information on only one side of the disk, while double-sided disks increase the storage capacity by recording on both sides. In addition, disks are classified as single-density or double-density. Double-density disks use a modified recording method to store twice as much information in the same disk area as can be stored on a single-density disk. Table 1 lists storage capacities for standard floppy disk media.

A magnetic head assembly (in contact with the disk) writes information onto the disk surface and subsequently reads the data back. This head assembly can

Table 1. Formatted Disk Capacities

Single-Density Format				
Byte/Sector	128	256	512	1024
Sectors/Track	26	15	8	4
Track/Disk	77	77	77	77
Bytes/Disk	256,256	295,680	315,392	315,392
Double-Density Format				
Byte/Sector	128	256	512	1024
Sectors/Track	52	30	16	8
Track/Disk	77	77	77	77
Bytes/Disk	512,512	591,360	630,784	630,784

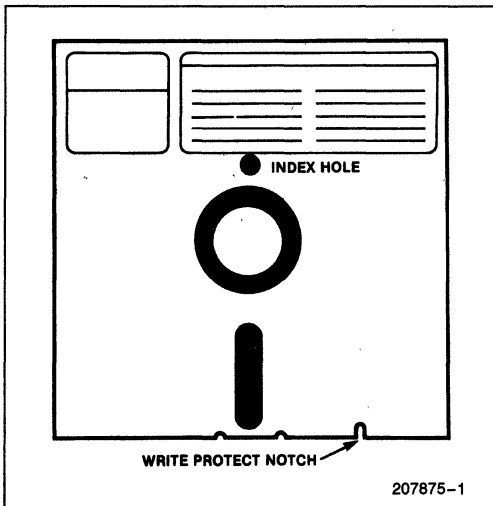


Figure 1. A Floppy Diskette

move from the outside edge of the disk toward the center in fixed increments. Once the head assembly is positioned at one of these fixed positions, the head can read or write information in a circular path as the disk revolves beneath the head assembly. This method divides the surface into a fixed number of cylinders (as shown in Figure 2). There are normally 77 cylinders on a standard disk. Once the head assembly is positioned at a given cylinder, data may be read or written on either side of the disk. The appropriate side of the disk is selected by the read/write head address (zero or one). Of course, a single-sided disk can only use head zero. The combination of cylinder address and head address uniquely specifies a single circular track on the disk. The physical beginning of a track is located by means of a small hole (physical index mark) punched through the plastic near the center of the disk. This hole is optically sensed by the drive on every revolution of the disk.

Each track is subdivided into a number of sectors (see detailed discussion in section 3). Sectors are generally

128, 256, 512, or 1024 data bytes in length. This track sectoring may be accomplished by one of two techniques: hard sectoring or soft sectoring. Hard sectored disks divide each track into a maximum of 32 sectors. The beginning of each sector is indicated by a sector hole punched in the disk plastic. Soft sectoring, the IBM standard method, allows software selection of sector sizes. With this technique, each data sector is preceded by a unique sector identifier that is read/written by the disk controller.

A floppy disk may also contain a write protect notch punched at the edge of the outer jacket of the disk. This notch is detected by the drive and passed to the controller as a write protect signal.

The Floppy Disk Drive

The floppy disk drive is an electromechanical device that records data on, or reads data from, the surface of a floppy disk. The disk drive contains head control electronics that move the head assembly one increment (step) forward (toward the center of the disk) or backward (toward the edge of the disk). Since the recording head must be in contact with the disk material in order to read or write information, the disk drive also contains head-load electronics. Normally the read/write head is unloaded until it is necessary to read or write information on the floppy disk. Once the head assembly has been positioned over the correct track on the disk, the head is loaded (brought into contact with the disk). This sequence prevents excessive disk wear. A small time penalty is paid when the head is loaded. Approximately thirty to fifty milliseconds are needed before data may be reliably read from, or written to, the disk. This time is known as the head load time. If desired, the head may be moved from cylinder to cylinder while loaded. In this manner, only a small time interval (head settling time) is required before data may be read from the new cylinder. The head settling time is often shorter than the head load time. Typically, disk drives also contain drive select logic that allows more than one physical drive to be connected to the same interface cable (from the controller). By means of a jumper on the drive, the drive number may be selected by the OEM or end user. The drive is enabled only when selected; when not selected, all control signals on the cable are ignored.

Finally, the drive provides additional signals to the system controller regarding the status of the drive and disk. These signals include:

Drive Ready—Signals the system that the drive door is closed and that a floppy disk is inserted into the drive.

Track Zero—Indicates that the head assembly is located over the outermost track of the disk. This signal may be used for calibration of the disk drive at system initialization and after an error condition.

Write Protect—Indicates that the floppy disk loaded into the drive is write protected.

Dual Sided—Indicates that the floppy disk in the drive is dual-sided.

Write Fault—Indicates that an error occurred during a recording operation.

Index—Informs the system that the physical index mark of the floppy disk (signifying the start of a data track) has been sensed.

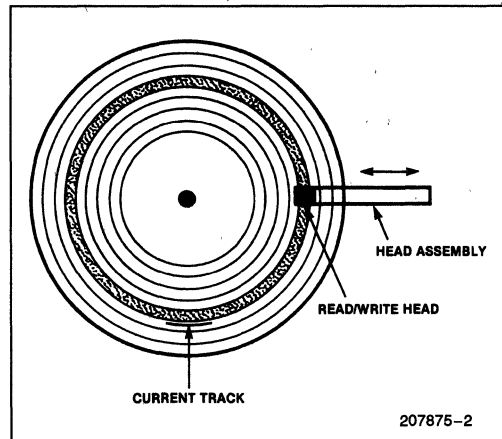


Figure 2. Concentric Cylinders on a Floppy Diskette

2.0 SUBSYSTEM OVERVIEW

A disk subsystem consists of the following functional electronic units:

- 1) Disk Controller Electronics
- 2) Disk Drive Electronics
- 3) Controller/Disk Interface (cables, drivers, terminators)
- 4) Controller/Microprocessor System Interface

The operation of these functional units is discussed in the following paragraphs.

Controller Electronics

The disk controller is responsible for converting high-level disk commands (normally issued by software executing on the system processor) into disk drive commands. This function includes:

- 1) Disk Drive Selection—Disk controllers typically manage the operations of multiple floppy disk drives. This controller function permits the system processor to specify which drive is to be used in a particular operation.
- 2) Track Selection—The controller issues a timed sequence of step pulses to move the head from its current location to the proper disk cylinder from which data is to be read or to which data is to be written. The controller stores the current cylinder number and computes the stepping distance from the current cylinder to the specified cylinder. The controller also manages the head select signal to select the correct side of the floppy disk.
- 3) Sector Selection—The controller monitors the data on a track until the requested sector is sensed.
- 4) Head Loading—The disk controller determines the times at which the head assembly is to be brought in contact with the disk surface in order to read or write data. The controller is also responsible for waiting until the head has settled before reading or writing information. Often the controller maintains the head loaded condition for up to 16 disk revolutions (approximately 2 seconds) after a read or write

operation has been completed. This feature eliminates the head load time during periods of heavy disk I/O activity.

- 5) Data Separation—The actual signal recorded on a floppy disk is a combination of timing information (clock) and data. The serial READ DATA input (from the disk drive) must be converted into two signal streams: clock and data. (The READ DATA input operates at 250K bits/second for single-density disks and 500K bits/second for double-density disks.) The serial data must also be assembled into 8-bit bytes for transfer to system memory. A byte must be assembled and transferred every 32 microseconds for single-density disks and every 16 microseconds for double-density.
- 6) Error-Checking—Information recorded on a floppy disk is subject to both hard and soft errors. Hard (permanent) errors are caused by media defects. Soft errors, on the other hand, are temporary errors caused by electromagnetic noise or mechanical interference. Disk controllers use a standard error checking technique known as a Cyclic Redundancy Check (CRC). As data is written to a disk, a 16-bit CRC character is computed and also stored on the disk. When the data is subsequently read, the CRC character allows the controller to detect data errors. Typically, when CRC errors are detected, the controlling software retries the failed operation (attempting to recover from a soft error). If data cannot reliably be read or written after a number of retries, the system software normally reports the error to the operator. Multiple CRC errors normally indicate unrecoverable media error on the current disk track. Subsequent recovery attempts must be defined by the system designers and tailored to meet system interfacing requirements.

Today, single-chip digital LSI floppy disk controllers such as the 8272 perform all the above functions with the exception of data separation. A data separation circuit (a combination of digital and analog electronics) synchronizes itself to the actual data rate of the disk drive. This data rate varies from drive to drive (due to mechanical factors such as motor tolerances) and varies from disk to disk (due to temperature effects). In order to operate reliably with both single- and double-density storage, the data separation circuit must be based on phase-locked loop (PLL) technology. The phase-locked loop data separation logic is described in section 5. The

separation logic, after synchronizing with the data stream, supplies a data window to the LSI disk controller. This window differentiates data information from clock information within the serial stream. The controller uses this window to reconstruct the data previously recorded on the floppy disk.

Drive Electronics

Each floppy disk drive contains digital electronic circuits that translate TTL-compatible command signals into electromechanical operations (such as drive selection and head movement/loading) and that sense and report disk or drive status to the controller (e.g., drive ready, write fault, and write protect). In addition, the drive electronics contain analog components to sense, amplify, and shape data pulses read from, or written to, the floppy disk surface by the read/write head.

Controller/Drive Interface

The controller/drive interface consists of high-current line drivers, Schmitt triggered input gates, and flat or twisted pair cable(s) to connect the disk drive electronics to the controller electronics. Each interface signal line is resistively terminated at the end of the cable farthest from the line drivers. Eight-inch drives may be directly interfaced by means of 50-conductor flat-cable. Generally, cable lengths should be less than ten feet in order to maintain noise immunity.

Normally, provisions are made for up to four disk drives to share the same interface cable. The controller may operate as many cable assemblies as practical. LSI floppy disk controllers typically operate one to four drives on a single cable.

Processor/Memory Interface

The disk controller must interface to the system processor and memory for two distinct purposes. First, the processor must specify disk control and command parameters to the controller. These parameters include the selection of the recording density and specification of disk formatting information (discussed in section 3). In addition to disk parameter specification, the processor must also send commands (e.g., read, write, seek, and scan) to the controller. These commands require the specification of the command code, drive number, cylinder address, sector address, and head address. Most LSI controllers receive commands and parameters by means of processor I/O instructions.

In addition to this I/O interface, the controller must also be designed for high-speed data transfer between memory and the disk drive. Two implementation methods may be used to coordinate this data transfer. The lower-cost method requires direct processor interven-

tion in the transfer. With this method, the controller issues an interrupt to the processor for each data transfer. (An equivalent method allows the processor to poll an interrupt flag in the controller status word.) In the case of a disk write operation, the processor writes a data byte (to be encoded into the serial output stream) to the disk controller following the receipt of each controller interrupt. During a disk read operation, the processor reads a data byte (previously assembled from the input data stream) from the controller after each interrupt. The processor must transfer a data byte from the controller to memory or transfer a data byte from memory to the disk controller within 16 or 32 microseconds after each interrupt (double-density and single-density response times, respectively).

If the system processor must service a variety of other interrupt sources, this interrupt method may not be practical, especially in double-density systems. In this case, the disk controller may be interfaced to a Direct Memory Access (DMA) controller. When the disk controller requires the transfer of a data byte, it simply activates the DMA request line. The DMA controller interfaces to the processor and, in response to the disk controller's request, gains control of the memory interface for a short period of time—long enough to transfer the requested data byte to/from memory. See section 6 for a detailed DMA interface description.

3.0 DISK FORMAT

New floppy disks must be written with a fixed format by the controller before these disks may be used to store data. Formatting is a method of taking raw media and adding the necessary information to permit the controller to read and write data without error. All formatting is performed by the disk controller on a track-by-track basis under the direction of the system processor. Generally, a track may be formatted at any time. However, since formatting "initializes" a complete disk track, all previously written data is lost (after a format operation). A format operation is normally used only when initializing new floppy disks. Since soft-sectoring in such a predominant formatting technique (due to IBM's influence), the following discussion will limit itself to self-sectored formats.

Data Recording Techniques

Two standard data recording techniques are used to combine clock and data information for storage on a floppy disk. The single-density technique is referred to as FM encoding. In FM encoding (see Figure 3), a double frequency encoding technique is used that inserts a data bit between two adjacent clock bits. (The presence of a data bit represents a binary "one" while the absence of a data bit represents a binary "zero.") The two adjacent clock bits are referred to as a bit cell, and

except for a unique field identifiers, all clock bits written on the disk are binary "ones." In FM encoding, each data bit is written at the center of the bit cell and the clock bits are written at the leading edge of the bit cell.

The encoding used for double-density recording is terms MFM encoding (for "Modified FM"). In MFM encoding (Figure 3) the data bits are again written at the center of the bit cell. However, a clock bit is written at the leading edge of the bit cell only if no data bit was written in the previous bit cell and no data bit will be written in the present bit cell.

Sectors

Soft-sectored floppy disks divide each track into a number of data sectors. Typically, sector sizes of 128, 256, 512, or 1024 data bytes are permitted. The sector size is specified when the track initially formatted by the controller. Table 1 lists the single- and double-density data storage capacities for each of the four sector sizes. Each sector within a track is composed of the following four fields (illustrated in Figure 4):

- 1) Sector ID Field—This field, consisting a seven bytes, is written only when the track is formatted. The ID field provides the sector identification that is used by the controller when a sector must be read or written. The first byte of the field is the ID address mark, a unique coding that specifies the beginning of the ID field. The second, third, and fourth bytes are the cylinder, head, and sector addresses, respectively, and the fifth byte is the sector length code. The last two bytes are the 16-bit CRC character for the ID field. During formatting, the controller supplies the address mark. The cylinder, head, and sector addresses and the sector length code are supplied to the controller by the processor software. The CRC character is derived by the controller from the data in the first five bytes.

- 2) Post ID Field Gap—The post ID field gap (gap 2) is written initially when the track is formatted. During subsequent write operations, the drive's write circuitry is enabled within the gap and the trailing bytes of the gap are rewritten each time the sector is updated (written). During subsequent read operations, the trailing bytes of the gap are used to synchronize the data separator logic with the upcoming data field.
- 3) Data Field—The length (number of data bytes) of the data field is determined by software when the track is formatted. The first byte of the data field is the data address mark, a unique coding that specifies the beginning of the data field. When a sector is to be deleted, (e.g., a hard error on the disk), a deleted data address mark is written in place of the data address mark. The last two bytes of the data field comprise the CRC character.
- 4) Post Data Field Gap—The post data field gap (gap 3) is written when the track is formatted and separates the preceding data field from the next physical ID field on the track. Note that a post data field gap is not written following the last physical sector on a track. The gap itself contains a program-selectable number of bytes. Following a sector update (write) operation, the drive's write logic is disabled during the gap. The actual size of gap 3 is determined by the maximum number of data bits that can be recorded on a track, the number of sectors per track and the total sector size (data plus overhead information). The gap size must be adjusted so that it is large enough to contain the discontinuity generated on the floppy disk when the write current is turned on or off (at the start or completion of a disk write operation) and to contain a synchronization field for the upcoming ID field (of the next sector). On the other hand, the gaps must be small enough so that the total number of data bits required on the track (sectors plus gaps) is less than the maximum number of data bits that can be recorded on the track. The gap

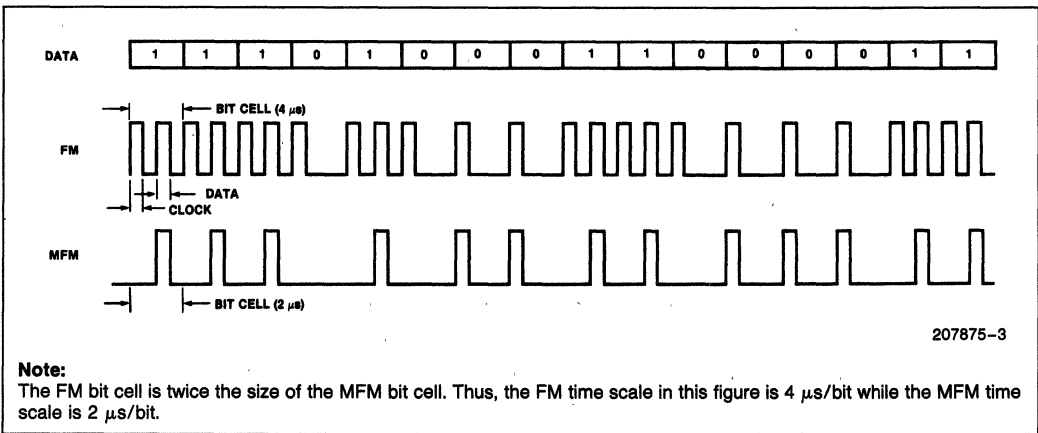


Figure 3. FM and MFM Encoding

size must be specified for all read, write, and format operation. The gap size used during disk reads and writes must be smaller than the size used to format the disk to avoid the splice points between contiguous physical sectors. Suggested gap sizes are listed in Table 9.

Tracks

The overall format for a track is illustrated in Figure 4. Each track consists of the following fields:

1) Pre-Index Gap—The pre-index gap (gap 5) is written only when the track is formatted.

- 2) Index Address Mark—The index address mark consists of a unique code that indicates the beginning of a data track. One index mark is written on each track when the track is formatted.
- 3) Post Index Gap—The post index gap (gap 1) is used during disk read and write operations to synchronize the data separator logic with the data to be read from the ID field (of the first sector). The post index gap is written only when the disk is formatted.
- 4) Sectors—The sector information (discussed above) is repeated once for each sector on this track.

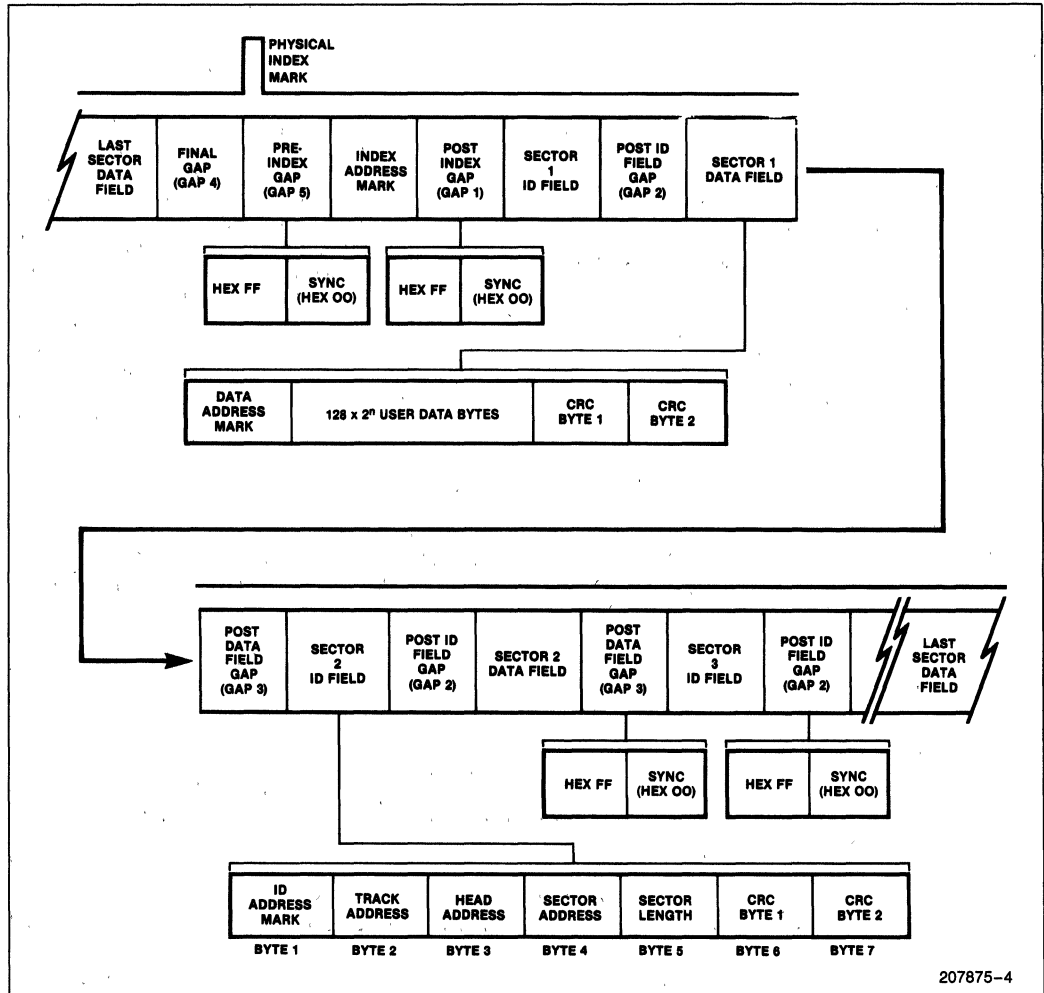


Figure 4. Standard Floppy Diskette Track Format (from SBC 204 Manual)

5) Final Gap—The final gap (gap 4) is written when the track is formatted and extends from the last physical data field on the track to the physical index mark. The length of this gap is dependent on the number of bytes per sector specified, the lengths of the program-selectable gaps specified, and the drive speed.

Sector Interleaving

The initial formatting of a floppy disk determines where sectors are located within a track. It is not necessary to allocate sectors sequentially around the track (i.e., 1,2,3, . . .,26). In fact, it is often advantageous to place the sectors on the track in non-sequential order. Sequential sector ordering optimizes sector access times during multi-sectors transfers (e.g., when a program is loaded) by permitting the number of sector specified (up to an entire track) to be transferred within a single revolution of the disk. A technique known as sector interleaving optimizes access times when, although sectors are accessed sequentially, a small amount of processing must be performed between sector reads/writes. For example, an editing program performing a text search reads sectors sequentially, and after each sector is read, performs a software search. If a match is not found, the software issues a read request for the next sector. Since the floppy disk continues to rotate during the time that the software executes, the next physical sector is already passing under the read/write head when the read request is issued, and the processor must wait for another complete revolution of the disk (approximately 166 milliseconds) before the data may actually be input. With interleaving, the sectors are not stored sequentially on a track; rather, each sector is physically removed from the previous sector by some number (known as the interleave factor) of physical sectors as shown in Figure 5. This method of sector allocation provides the processor additional execution time between sectors on the disk. For example, with a 26 sector/track format, an interleave factor of 2 provides 6.4 milliseconds of processing time between sequential 128 byte sector accesses.

To calculate the correct interleave factor, the maximum processor time between sector operations must be divided by the time required for a complete sector to pass under the disk read/write head. After determining the interleave factor, the correct sector numbers are passed to the disk controller (in the exact order that they are to physically appear on the track) during the execution of a format operation.

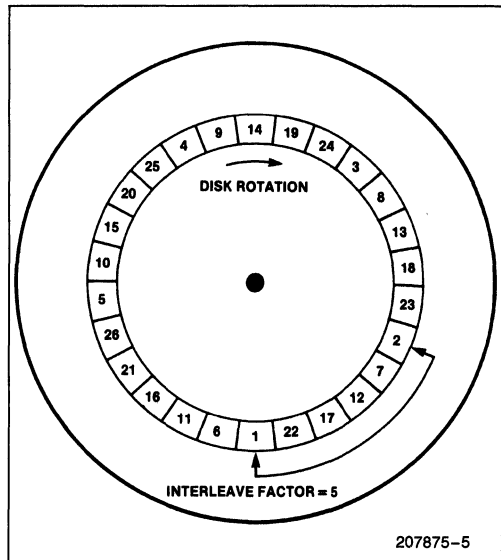


Figure 5. Interleaved Sector Allocation within a Track

4.0 THE 8272 FLEXIBLE DISKETTE CONTROLLER

The 8272 is a single-chip LSI Floppy Disk Controller (FDC) that contains the circuitry necessary to implement both single- and double-density floppy disk storage subsystems (with up to four dual-sided disk drives per FDC). The 8272 supports the IBM 3740 single-density recording format (FM) and the IBM System 34 double-density recording format (MFMM). With the 8272, less than 30 ICs are needed to implement a complete disk subsystem. The 8272 accepts and executes high-level disk commands such as format track, seek, read sector, write sector, and read track. All data synchronization and error checking is automatically performed by the FDC to ensure reliable data storage and subsequent retrieval. External logic is required only for the generation of the FDC master clock and write clock (see Section 6) and for data separation (Section 5). The FDC provides signals that control the startup and base frequency selection of the data separator. These signals greatly ease the design of a phase-locked loop data separator.

In addition to the data separator interface signals, the 8272 also provides the necessary signals to interface to

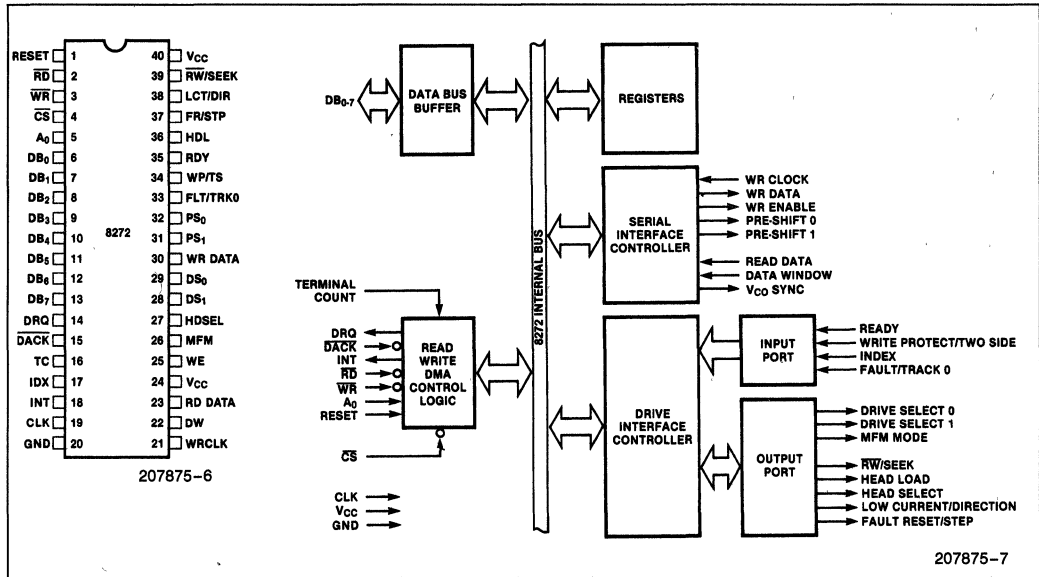


Figure 6. 8272 Pin Configuration and Internal Block Diagram

microprocessor systems with or without Direct Memory Access (DMA) capabilities. In order to interface to a large number of commercially available floppy disk drives, the FDC permits software specification of the track stepping rate, the head load time, and the head unload time.

The pin configuration and internal block diagram of the 8272 is shown in Figure 6. Table 2 contains a description for each FDC interface pin.

Floppy Disk Commands

The 8272 executes fifteen high-level disk interface commands:

- | | |
|------------------------|--------------------|
| Specify | Write Data |
| Sense Drive Status | Write Deleted Data |
| Sense Interrupt Status | Read Track |
| Seek | Read ID |
| Recalibrate | Scan Equal |
| Format Track | Scan High or Equal |
| Read Data | Scan Low or Equal |
| Read Deleted Data | |

Each command is initiated by a multi-byte transfer from the processor to the FDC (the transferred bytes contain command and parameter information). After complete command specification, the FDC automatically executes the command. The command result data (after execution of the command) may require a multi-byte transfer of status information back to the processor. It is convenient to consider each FDC command as consisting of the following three phases:

COMMAND PHASE: The executing program transfers to the FDC all the information required to perform a particular disk operation. The 8272 automatically enters the command phase after RESET and following the completion of the result phase (if any) of a previous command.

EXECUTION PHASE: The FDC performs the operation as instructed. The execution phase is entered immediately after the last command parameter is written to the FDC in the preceding command phase. The execution phase normally ends when the last data byte is transferred to/from the disk (signalled by the TC input to the FDC) or when an error occurs.

RESULT PHASE: After completion of the disk operation, status and other housekeeping information are made available to the processor. After the processor reads this information, the FDC re-enters the command phase and is ready to accept another command.

Table 2. 8272 FDC Pin Description

Number	Pin Symbol	I/O	To/From	Description
1	RST	I	μ P	RESET: Active-high signal that places the FDC in the "idle" state and all disk drive output signals are forced inactive (low). This input must be held active during power on reset while the RD and WR input are active.
2	$\overline{\text{RD}}$	I*	μ P	READ: Active-low control signal that enables data transfer from the FDC to the data bus.
3	$\overline{\text{WR}}$	I*	μ P	WRITE: Active-low control signal that enables data transfer from the data bus into the FDC.
4	$\overline{\text{CS}}$	I	μ P	CHIP SELECT: Active-low control signal that selects the FDC. No reading of writing will occur unless the FDC is selected.
5	A ₀	I*	μ P	ADDRESS: Selects the Data Register or Main Status Register for input/output in conjunction with the RD and WR inputs. (See Table 3.)
6–13	DB ₀ –DB ₇	I/O*	μ P	DATA BUS: Bidirectional three-state 8-bit data bus.
14	DRQ	O	DMA	DMA REQUEST: Active-high output that indicates an FDC request for DMA services.
15	$\overline{\text{DACK}}$	1	DMA	DMA ACKNOWLEDGE: Active-low control signal indicating that the requested DMA transfer is in progress.
16	TC	I	DMA	TERMINAL COUNT: Active-high signal that causes the termination of a command. Normally, the terminal count input is directly connected to the TC/EOP output from the DMA controller, signalling that the DMA transfer has been completed. In a non-DMA environment, the processor must count data transfers and supply a TC signal to the FDC.
17	IDX	I	Drive	INDEX: Indicates detection of the physical index mark (the beginning of a track) on the selected disk drive.
18	INT	O	μ P	INTERRUPT REQUEST: Active-high signal indicating an 8272 interrupt service request.
19	CLK	I		CLOCK: Signal phase 8 MHz clock (50% duty cycle).
20	GND			GROUND: DC power return.
21	WR CLK	I		WRITE CLOCK: 500 kHz (FM) or 1 MHz (MFM) write clock with a constant pulse width of 250 ns (for both FM and MFM recording). The write clock must be present at all times.
22	DW	I	PLL	DATA WINDOW: Data sample signal from the phase-locked loop indicating that the FDC should sample input data from the disk drive.
23	RD DATA	I	Drive	READ DATA: FDC input data from the selected disk drive.
24	VCO	O	PLL	VCO SYNC: Active-high output that enables the phase-locked loop to synchronize with the input data from the disk drive.
25	WE	O	Drive	WRITE ENABLE: Active-high output that enables the disk drive write gate.
26	MFM	O	PLL	MFM MODE: Active-high output used by external logic to enable the MFM double-density recording mode. When the MFM output is low, single-density FM recording is indicated.
27	HDSEL	O	Drive	HEAD SELECT: Selects head 0 or head 1 on a dual-sided disk.

Table 2. 8272 FDC Pin Description (Continued)

Number	Pin Symbol	I/O	To/From	Description
28, 29	DS ₁ , DS ₀	O	Drive	DRIVE SELECT: Selects one of four disk drives.
30	WR DATA	O	Drive	WRITE DATA: Serial data stream (combination of clock and data bits) to be written on the disk.
31, 32	PS ₁ , PS ₀	O	Drive	PRECOMPENSATION (PRE-SHIFT) CONTROL: Write precompensation output control during MFM mode. Specifies early, late, and normal timing signals. See the discussion in Section 5.
33	FLT/TRKO	I	Drive	FAULT/TRACK 0: Senses the disk drive fault condition in the Read/Write mode and the Track 0 condition in the Seek mode.
34	WP/TS	I	Drive	WRITE PROTECT/TWO-SIDED: Sense the disk write protect status in the Read/Write mode and the dual-sided media status in the Seek mode.
35	RDY	I	Drive	READY: Senses the disk drive ready status.
36	HDL	O	Drive	HEAD LOAD: Loads the disk drive read/write head. (The head is placed in contact with the disk.)
37	FR/STP	O	Drive	FAULT RESET/STEP: Resets the fault flip-flop in the disk drive when operating in the Read/Write mode. Provides head step pulses (to move the head from one cylinder to another cylinder) in the Seek mode.
38	LCT/DIR	O	Drive	LOW CURRENT/DIRECTION: Signals that the recording head has been positioned over the inner cylinders (44–77) of the floppy disk in the Read/Write mode. (The write current must be lowered when recording on the physically shorter inner cylinders of the disk. Most drives do not track the actual head position and require that the FDC supply this signal.) Determines the head step direction in the Seek mode. In the Seek mode, a high level on this pin steps the read/write head toward the spindle (step-in); a low level steps the head away from the spindle (step-out).
39	RW/SEEK	O	Drive	READ, WRITE/SEEK MODE SELECTOR: A high level selects the Seek mode; a low level selects the Read/Write mode.
40	V _{CC}			+5V DC Power.

*Disabled when CS is high.

Interface Registers

To support information transfer between the FDC and the system processor, the 8272 contains two 8-bit registers: the Main Status Register and the Data Register. The Main Status Register (read only) contains FDC status information and may be accessed at any time. The Main Status Register (Table 4) provides the system processor with the status of each disk drive, the status of the FDC, and the status of the processor interface. The Data Register (read/write) stores data, commands, parameters, and disk drive status information. The Data Register is used to program the FDC during the command phase and to obtain result information after completion of FDC operations. Data is read from, or written to, the FDC registers by the combination of the A₀, RD, WR, and CS signals, as described in Table 3.

In addition to the Main Status Register, the FDC contains four additional status registers (ST₀, ST₁, ST₂, and ST₃). These registers are only available during the result phase of a command.

Table 3. FDC Read/Write Interface

CS	A ₀	RD	WR	Function
0	0	0	1	Read Main Status Register
0	0	1	0	Illegal
0	0	0	0	Illegal
0	1	0	0	Illegal
0	1	0	1	Read from Data Register
0	1	1	0	Write into Data Register
1	X	X	X	Data Bus is three-stated

Table 4. Main Status Register Bit Definitions

Bit Number	Symbol	Description
0	D ₀ B	DISK DRIVE 0 BUSY: Disk Drive 0 is in the Seek mode.
1	D ₁ B	DISK DRIVE 1 BUSY: Disk Drive 1 is in the Seek mode.
2	D ₂ B	DISK DRIVE 2 BUSY: Disk Drive 2 is in the Seek mode.
3	D ₃ B	DISK DRIVE 3 BUSY: Disk Drive 3 is in the Seek mode.
4	CB	FDC BUSY: A read or write command is in process.
5	NDM	NON-DMA MODE: The FDC is in the non-DMA mode when this bit is high. This bit is set only during the execution phase of commands in the non-DMA mode. Transition to a low level indicates that the execution phase has ended.
6	DIO	DATA INPUT/OUTPUT: Indicates the direction of a data transfer between the FDC and the Data Register. When DIO is high, data is read from the Data Register by the processor; when DIO is low, data is written from the processor to the Data Register.
7	RQM	REQUEST FOR MASTER: Indicates that the Data Register is read to send data to, or receive data from, the processor.

Command/Result Phases

Table 5 lists the 8272 command set. For each of the fifteen commands, command and result phase data transfers are listed. A list of abbreviations used in the table is given in Table 6, and the contents of the result status registers (ST0-ST3) are illustrated in Table 7.

The bytes of data which are sent to the 8272 during the command phase, and are read out of the 8272 in the result phase, must occur in the order shown in Table 5. That is, the command code must be sent first and the other bytes sent in the prescribed sequence. All bytes of the command and result phases must be read/written as described. After the last byte of data in the command

phase is sent to the 8272 the execution phase automatically starts. In a similar fashion, when the last byte of data is read from the 8272 in the result phase, the command is automatically ended and the 8272 is ready for a new command. A command may be aborted by simply raising the terminal count signal (pin 16). This is a convenient means of ensuring that the processor may always gain control of the 8272 (even if the disk system hangs up in an abnormal manner).

It is important to note that during the result phase all bytes shown in Table 5 must be read. The Read Data command, for example, has seven bytes of data in the result phase. All seven bytes must be read in order to successfully complete the Read Data command. The 8272 will not accept a new command until all seven bytes have been read. The number of command and result bytes varies from command-to-command.

In order to read data from, or write data to, the Data Register during the command and result phases, the system processor must examine the Main Status Register to determine if the Data Register is available. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command word may be written into the 8272. Many of the commands require multiple bytes, and as a result, the Main Status Register must be read prior to each byte transfer to the 8272. To read status bytes during the result phase, DIO and RQM in the Main Status Register must both be high. Note, checking the Main Status Register in this manner before each byte transfer to/from the 8272 is required only in the command and result phases, and is NOT required during the execution phase.

Execution Phase

All data transfers to (or from) the floppy drive occur during the execution phase. The 8272 has two primary modes of operation for data transfers (selected by the specify command):

1. DMA mode
2. non-DMA mode

In the DMA mode, DRQ (DMA Request) is activated for each transfers request. The DMA controller responds to DRQ with $\overline{\text{DACK}}$ (DMA Acknowledge) and RD (for read commands) or WR (for write commands). DRQ is reset by the FDC during the transfer. INT is activated after the last data transfer, indicating the completion of the execution phase, and the beginning of the result phase. In the DMA mode, the terminal count (TC/EOP) output of the DMA controller should be connected to the 8272 TC input to properly terminate disk data transfer commands.

Table 5. 8272 Command Set

PHASE	R/W	DATA BUS								REMARKS	PHASE	R/W	DATA BUS								REMARKS
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀				D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
READ DATA																					
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes	Sector ID information prior to Command execution										
	W	0	0	0	0	0	HDS	DS1	DS0												
	W						C														
	W						H														
	W						R														
	W						N														
	W						EOT														
Execution	W					GPL			Data transfer between the FDD and the main-system												
	W					DTL															
	R					ST 0															
	R					ST 1															
Result	R					ST 2			Status information after Command execution												
	R					C															
	R					H															
	R					R															
	R					N															
	R																				
READ DELETED DATA																					
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes	Sector ID information prior to Command execution										
	W	0	0	0	0	0	HDS	DS1	DS0												
	W						C														
	W						H														
	W						R														
	W						N														
	W						EC 1														
Execution	W					GPL			Data transfer between the FDD and the main-system												
	W					DTL															
	R					ST 0															
	R					ST 1															
Result	R					ST 2			Status information after Command execution												
	R					C															
	R					H															
	R					R															
	R					N															
	R																				
READ ID																					
Command	W	0	MFM	0	0	1	0	1	0	Command Codes											
	W	0	0	0	0	0	HDS	DS1	DS0												
Execution	W									The first correct ID information on the track is stored in Data Register											
	R																				
	R						ST 0														
	R						ST 1														
	R						ST 2														
	R						C														
	R						H														
Result	R					R			Sector ID information during Execution Phase												
	R					N															
	R																				
	R																				
WRITE DATA																					
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes	Sector ID information prior to Command execution										
	W	0	0	0	0	0	HDS	DS1	DS0												
	W						C														
	W						H														
	W						R														
	W						N														
	W						EOT														
Execution	W					GPL			Data transfer between the main-system and the FDD												
	W					DTL															
	R					ST 0															
	R					ST 1															
Result	R					ST 2			Status information after Command execution												
	R					C															
	R					H															
	R					R															
	R					N															
	R																				
WRITE DELETED DATA																					
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes	Sector ID information prior to Command execution										
	W	0	0	0	0	0	HDS	DS1	DS0												
	W						C														
	W						H														
	W						R														
	W						N														
	W						EOT														
Execution	W					GPL			Data transfer between the FDD and the main-system												
	W					DTL															
	R					ST 0															
	R					ST 1															
Result	R					ST 2			Status information after Command execution												
	R					C															
	R					H															
	R					R															
	R					N															
	R																				
FORMAT A TRACK																					
Command	W	0	MFM	0	0	1	1	0	1	Command Codes											
	W	0	0	0	0	0	HDS	DS1	DS0												
Execution	W									Bytes/Sector Sectors/Track Gap 3 Filter Byte											
	W						N														
	W						SC														
	W						GPL														
	W						D														
	R																				
	R						ST 0														
Result	R					ST 1			FDC formats an entire track												
	R					ST 2															
	R					C															
	R					H															
	R					R															
	R					N															
Result	R								In this case, the ID information has no meaning												
	R																				
	R																				
	R																				
SCAN EQUAL																					
Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes											
	W	0	0	0	0	0	HDS	DS1	DS0												
Execution	W									Sector ID information prior to Command execution											
	W						C														
	W						H														
	W						R														
	W						N														
	W						EOT														
	W						GPL														
Execution	W					STP			Data compared between the FDD and the main-system												
	R					ST 0															
	R					ST 1															
	R					ST 2															
Result	R					C			Status information after Command execution												
	R					H															
	R					R															
	R					N															
	R																				
	R																				

NOTE:
1. A₀ = 1 for all operations.

Table 5. 8272 Command Set (Continued)

PHASE	R/W	DATA BUS								REMARKS	PHASE	R/W	DATA BUS								REMARKS	
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀				D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		
SCAN LOW OR EQUAL																						
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes	Command	W	0	0	0	0	0	1	1	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior Command execution	Execution	W	0	0	0	0	0	0	DS1	DS0	Head retracted to Track 0	
	W						C			Data compared between the FDD and the main-system												
	W						H															
	W						R															
	W						N															
	W						EOT															
	W						GPL															
	W						STP															
Execution																						
Result	R						ST 0			Status information after Command execution	Command	W	0	0	0	0	1	0	0	0	Command Codes	
	R						ST 1				Result	R					ST 0				Status information at the end of each seek operation about the FDC	
	R						ST 2					R					C					
	R						C															
	R						H			Sector ID information after Command execution												
	R						R															
	R						N															
SCAN HIGH OR EQUAL																						
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes	Command	W	0	0	0	0	0	0	1	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior Command execution	Execution	W	0	0	0	0	0	0	1	1	Timer Settings	
	W						C			Data compared between the FDD and the main-system												
	W						H															
	W						R															
	W						N															
	W						EOT															
	W						GPL															
	W						STP															
Execution																						
Result	R						ST 0			Status information after Command execution	Command	W	0	0	0	0	0	0	1	1	Command Codes	
	R						ST 1				Result	W									Status information about the FDD	
	R						ST 2					W										
	R						C					W										
	R						H			Sector ID information after Command execution												
	R						R															
	R						N															
SCAN LOW OR EQUAL																						
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes	Command	W	0	0	0	0	0	1	0	0	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior Command execution	Execution	W	0	0	0	0	0	0	0	0	Status information at the end of each seek operation about the FDC	
	W						C			Data compared between the FDD and the main-system												
	W						H															
	W						R															
	W						N															
	W						EOT															
	W						GPL															
	W						STP															
Execution																						
Result	R						ST 0			Status information after Command execution	Command	W	0	0	0	0	0	1	0	0	Command Codes	
	R						ST 1				Result	W									Status information about the FDD	
	R						ST 2					W										
	R						C					W										
	R						H			Sector ID information after Command execution												
	R						R															
	R						N															
SCAN HIGH OR EQUAL																						
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes	Command	W	0	0	0	0	0	1	0	0	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior Command execution	Execution	W	0	0	0	0	0	1	0	0	Status information about the FDD	
	W						C			Data compared between the FDD and the main-system												
	W						H															
	W						R															
	W						N															
	W						EOT															
	W						GPL															
	W						STP															
Execution																						
Result	R						ST 0			Status information after Command execution	Command	W	0	0	0	0	0	1	0	0	Command Codes	
	R						ST 1				Result	W									Status information about the FDD	
	R						ST 2					W										
	R						C					W										
	R						H			Sector ID information after Command execution												
	R						R															
	R						N															

207875-19

Table 6. Command/Result Parameter Abbreviations

Symbol	Description
C	CYLINDER ADDRESS: The currently selected cylinder address (0 to 76) on the disk.
D	DATA PATTERN: The pattern to be written in each sector data field during formatting.
DS0,DS1	DISK DRIVE SELECT:
	DS1 DS0 Description
	0 0 Drive 0
	0 1 Drive 1
	1 0 Drive 2
1 1 Drive 3	
DTL	SPECIAL SECTOR SIZE: During the execution of disk read/write commands, this parameter is used to temporarily alter the effective disk sector size. By setting N to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the diskette) is larger than DTL specifies, the remainder of the actual sector is not passed to the system during read commands; during write commands, the remainder of the actual sector is written with all-zeroes bytes. DTL should be set to FF hexadecimal when N is not zero.
EOT	END OF TRACK: The final sector number of the current track.

Table 6. Command/Result Parameter Abbreviations (Continued)

Symbol	Description
GPL	GAP LENGTH: The gap 3 size. (Gap 3 is the space between sectors excluding the VCO synchronization field as defined in section 3.)
H	HEAD ADDRESS: Selected head: 0 or 1 (disk side 0 or 1, respectively) as encoded in the sector ID field.
HLT	HEAD LOAD TIME: Defines the time interval that the FDC waits after loading the head before initiating a read or write operation. Programmable from 2 to 254 milliseconds (in increments of 2 ms).
HUT	HEAD UNLOAD TIME: Defines the time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Programmable from 16 to 240 milliseconds (in increments of 16 ms).
MFM	MFM/FM MODE SELECTOR: Selects MFM double-density recording mode when high, FM single-density mode when low.
MT	MULTI-TRACK SELECTOR: When set, this flag selects the multi-track operating mode. In this mode (used only with dual-sided disks), the FDC treats a complete cylinder (under both read/write head 0 and read/write head 1) as a single track. The FDC operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set (high), a multi-sector read operation will automatically continue to the first sector under head 1 when FDC finishes operating on the last sector under head 0.
N	SECTOR SIZE: The number of data bytes within a sector. (See Table 9.)
ND	NON-DMA MODE FLAG: When set (high), this flag indicates that the FDC is to operate in the non-DMA mode. In this mode, the processor is interrupted for each data transfer. When low, the FDC interfaces to a DMA controller by means of the DRQ and DACK signals.
R	SECTOR ADDRESS: Specifies the sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of first sector to be read or written.
SC	NUMBER OF SECTORS PER TRACK: Specifies the number of sectors per track to be initialized by the Format Track command.
SK	SKIP FLAG: When this flag is set, sectors containing deleted data address marks will automatically be skipped during the execution of multi-sector Read Data or Scan commands. In the same manner, a sector containing a data address mark will automatically be skipped during the execution of a multi-sector Read Deleted Data command.
SRT	STEP RATE INTERVAL: Defines the time interval between step pulses issued by the FDC (track-to-track access time). Programmable from 1 to 16 milliseconds (in increments of 1 ms).

Table 6. Command/Result Parameter Abbreviations (Continued)

Symbol	Description
ST0,ST3 ST2,ST3	STATUS REGISTER 0-3: Registers within the FDC that store status information after a command has been executed. This status information is available to the processor during the Result Phase after command execution. These registers may only be read after a command has been executed (in the exact order shown in Table 5 for each command). These registers should not be confused with the Main Status Register.
STP	SCAN SECTOR INCREMENT: During Scan operations, this parameter is added to the current sector number in order to determine the next sector to be scanned.

Table 7. Status Register Definitions

Bit Number	Symbol	Description
Status Register 0		
7,6	IC	INTERRUPT CODE: 00— Normal termination of command. The specified command was properly executed and completed without error. 01— Abnormal termination of command. Command execution was started but could not be successfully completed. 10— Invalid command. The requested command could not be executed. 11— Abnormal termination. During command execution, the disk drive ready signal changed state.
5	SE	SEEK END: This flag is set (high) when the FDC has completed the Seek command and the read/write head is positioned over the correct cylinder.
4	EC	EQUIPMENT CHECK ERROR: This flag is set (high) if a fault signal is received from the disk drive or if the track 0 signal fails to become active after 77 step pulses (Recalibrate command).
3	NR	NOT READY ERROR: This flag is set if a read or write command is issued and either the drive is not ready or the command specifies side 1 (head 1) of a single-sided disk.
2	H	HEAD ADDRESS: The head address at the time of the interrupt.
1,0	DS1,DS0	DRIVE SELECT: The number of the drive selected at the time of the interrupt.
Status Register 1		
7	EN	END OF TRACK ERROR: This flag is set if the FDC attempts to access a sector beyond the final sector of the track.
6		Not used. This bit is always low.
5	DE	DATA ERROR: Set when the FDC detects a CRC error in either the ID field or the data field of a sector.

Table 7. Status Register Definitions (Continued)

Bit Number	Symbol	Description
Status Register 1 (Continued)		
4	OR	OVERRUN ERROR: Set (during data transfers) if the FDC does not receive DMA or processor service within the specified time interval.
3		Not used. This bit is always low.
2	ND	SECTOR NOT FOUND ERROR: This flag is set by any of the following conditions. a) The FDC cannot locate the sector specified in the Read Data, Read Deleted Data, or Scan command. b) The FDC cannot locate the starting sector specified in the Read Track command. c) The FDC cannot read the ID field without error during a Read ID command.
1	NW	WRITE PROTECT ERROR: This flag is set if the FDC detects a write protect signal from the disk drive during the execution of a Write Data, Write Deleted Data, or Format Track command.
0	MA	MISSING ADDRESS MARK ERROR: This flag is set by either of the following conditions: a) The FDC cannot detect the ID address mark on the specified track (after two occurrences of the physical index mark). b) The FDC cannot detect the data address mark or deleted data address mark on the specified track. (See also the MD bit of Status Register 2.)
Status Register 2		
7		NOT USED: This bit is always low.
6	CM	CONTROL MARK: This flag is set when the FDC encounters one of the following conditions: a) A deleted data address mark during the execution of a Read Data or Scan command. b) A data address mark during the execution of a Read Deleted Data command.
5	DD	DATA ERROR: Set (high) when the FDC detects a CRC error in a sector data field. This flag is not set when a CRC error is detected in the ID field.
4	WC	CYLINDER ADDRESS ERROR: Set when the cylinder address from the disk sector ID field is different from the current cylinder address maintained within the FDC.
3	SH	SCAN HIT: Set during the execution of the Scan command if the scan condition is satisfied.
2	SN	SCAN NOT SATISFIED: Set during execution of the Scan command if the FDC cannot locate a sector on the specified cylinder that satisfies the scan condition.
1	BC	BAD TRACK ERROR: Set when the cylinder address from the disk sector ID field is FF hexadecimal and this cylinder address is different from the current cylinder address maintained within the FDC. This all "ones" cylinder number indicates a bad track (one containing hard errors) according to the IBM soft-sectored format specifications.
0	MD	MISSING DATA ADDRESS MARK ERROR: Set if the FDC cannot detect a data address mark or deleted data address mark on the specified track.

Table 7. Status Register Definitions (Continued)

Bit Number	Symbol	Description
Status Register 3		
7	FT	FAULT: This flag indicates the status of the fault signal from the selected disk drive.
6	WP	WRITE PROTECTED: This flag indicates the status of the write protect signal from the selected disk drive.
5	RDY	READY: This flag indicates the status of the ready signal from the selected disk drive.
4	TO	TRACK 0: This flag indicates the status of the track 0 signal from the selected disk drive.
3	TS	TWO-SIDED: This flag indicates the status of the two-sided signal from the selected disk drive.
2	H	HEAD ADDRESS: This flag indicates the status of the side select signal for the currently selected disk drive.
1,0	DS1,DS0	DRIVE SELECT: Indicates the currently selected disk drive number.

In the non-DMA mode, transfers requests are indicated by activation of both the INT output signal and the RQM flag (bit 7) in the Main Status Register. INT can be used for interrupt-driven systems and RQM can be used for polled systems. The system processor must respond to the transfer request by reading data from (activating RD), or writing data to (activating WR), the FDC. This response removes the transfer request (INT and RQM are set inactive). After completing the last transfer, the 8272 activates the INT output to indicate the beginning of the result phase. In the non-DMA mode, the processor must activate the TC signal to the FDC (normally by means of an I/O port) after the transfer request for the last data byte has been received (by the processor) and before the appropriate data byte has been read from (or written to) the FDC.

In either mode of operation (DMA or non-DMA), the execution phase ends when a terminal count signal is sensed or when the last sector on a track (the EOT parameter—Table 5) has been read or written. In addition, if the disk drive is in a “not ready” state at the beginning of the execution phase, the “not ready” flag (bit 3 in Status Register 0) is set (high) and the command is terminated.

If a fault signal is received from the disk drive at the end of a write operation (Write Data, Write Deleted Data, or Format), the FDC sets the “equipment check” flag (bit 4 in Status Register 0), and terminates the command after setting the interrupt code (bits 7 and 6 of Status Register 0) to “01” (bit 7 low, bit 6 high).

Multi-Sector and Multi-Track Transfers

During disk read/write transfers (Read Data, Write Data, Read Deleted Data, and Write Deleted Data), the FDC will continue to transfer data from sequential sectors until the TC input is sensed. In the DMA mode, the TC input is normally connected to the TC/EOP (terminal count) output of the DMA controller. In the non-DMA mode, the processor directly controls the FDC TC input as previously described. Once the TC input is received, the FDC stops requesting data transfers (from the system processor or DMA controller). The FDC, however, continues to read data from, or write data to, the floppy disk until the end of the current disk sector. During a disk read operation, the data read from the disk (after reception of the TC input) is discarded, but the data CRC is checked for errors; during a disk write operation, the remainder of the sector is filled with all-zero bytes.

If the TC signal is not received before the last byte of the current sector has been transferred to/from the system, the FDC increments the sector number by one and initiates a read or write command for this new disk sector.

The FDC is also designed to operate in a multi-track mode for dual-sided disks. In the multi-track mode (specified by means of the MT flag in the command byte—Table 5) the FDC will automatically increment the head address (from 0 to 1) when the last sector (on the track under head 0) has been read or written. Reading or writing is then continued on the first sector (sector 1) of head 1.

Drive Status Polling

After the power-on reset, the 8272 automatically enters a drive status polling mode. If a change in drive status is detected (all drives are assumed to be "not ready" at power-on), an interrupt is generated. The 8272 continues this status polling between command executions (and between step pulses in the Seek command). In this manner, the 8272 automatically notifies the system processor when a floppy disk is inserted, removed, or changed by the operator.

Command Details

During the command phase, the Main Status Register must be polled by the CPU before each byte is written into the Data Register. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command may be written into the 8272. The beginning of the execution phase for any of these commands will cause DIO to be set high and RQM to be set low.

The following paragraphs describe the fifteen FDC commands in detail.

Specify

The Specify command is used prior to performing any disk operations (including the formatting of a new disk) to define drive/FDC operating characteristics. The Specify command parameters set the values for three internal timers:

1. **Head Load Time (HLT)**—This seven-bit value defines the time interval that the FDC waits after loading the head before initiating a read or write operation. This timer is programmable from 2 to 254 milliseconds in increments of 2 ms.
2. **Head Unload Time (HUT)**—This four-bit value defines the time from the end of the execution phase (of a read or write command) until the head is unloaded. This timer is programmable from 16 to 240 milliseconds in increments of 16 ms. If the processor issues another command before the head unloads, the head will remain loaded and the head load wait will be eliminated.
3. **Step Rate Time (SRT)**—This four-bit value defines the time interval between step pulses issued by the FDC (track-to-track access time). This timer is programmable from 1 to 16 milliseconds in increments of 1 ms.

The time intervals mentioned above are a direct function of the FDC clock (CLK on pin 19). Times indicated above are for an 8 MHz clock.

The Specify command also indicates the choice of DMA or non-DMA operation (by means of the ND

bit). When this bit is high the non-DMA mode is selected; when ND is low, the DMA mode is selected.

Sense Drive Status

This command may be used by the processor whenever it wishes to obtain the status of the disk drives. Status Register 3 (returned during the result phase) contains the drive status information as described in Table 7.

Sense Interrupt Status

An interrupt signal is generated by the FDC when one or more of the following events occurs:

- 1) The FDC enters the result phase for:
 - a) Read Data command
 - b) Read Track command
 - c) Read ID command
 - d) Read Deleted Data command
 - e) Write Data command
 - f) Format Track command
 - g) Write Deleted Data command
 - h) Scan commands
- 2) The ready signal from one of the disk drives changes state.
- 3) A Seek or Recalibrate command completes operation.
- 4) The FDC requires a data transfer during the execution phase of a command in the non-DMA mode.

Interrupts caused by reasons (1) and (4) above occur during normal command operations and are easily discernible by the processor. However, interrupts caused by reasons (2) and (3) above are uniquely identified with the aid of the Sense Interrupt Status command. This command, when issued, resets the interrupt signal and by means of bits 5, 6, and 7 of Status Register 0 (returned during the result phase) identifies the cause of the interrupt (see Table 8).

Table 8. Interrupt Codes

Seek End Bit 5	Interrupt Code		Cause
	Bit 6	Bit 7	
0	1	1	Read Line changed state, either polarity
1	0	0	Normal Termination of Seek or Recalibrate Command
1	1	0	Abnormal Termination of Seek or Recalibrate Command

Neither the Seek nor the Recalibrate command has a result phase. Therefore, it is mandatory to use the Sense Interrupt Status Command after these commands to effectively terminate them and to provide verification of the disk head position.

When an interrupt is received by the processor, the FDC busy flag (bit 4) and the non-DMA (bit 5) may be used to distinguish the above interrupt causes:

bit 5	bit 4	
0	0	Asynchronous event-(2) or (3) above
0	1	Result phase-(1) above
1	1	Data transfer required-(4) above

A single interrupt request to the processor may, in fact, be caused by more than one of the above events. The processor should continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are serviced.

Seek

The Seek command causes the drive's read/write head to be positioned over the specified cylinder. The FDC determines the difference between the current cylinder address and the desired (specified) address, and issues the appropriate number of step pulses. If the desired cylinder address is larger than the current address, the direction signal (LCT/DIR, pin 38) is set high (step-in); the direction signal is set low (step-out) if the desired cylinder address is less than the current address. No head movement occurs (no step pulses are issued) if the desired cylinder is the same as the current cylinder.

The rate at which step pulses are issued is controlled by the step rate time (SRT) in the Specify command. After each step pulse is issued, the desired cylinder address is compared against the current cylinder address. When the cylinder addresses are equal, the "seek end" flag (bit 5 in Status Register 0) is set (high) and the command is terminated. If the disk drive becomes "not ready" during the seek operation, the "not ready" flag (in Status Register 0) is set (high) and the command is terminated.

During the command phase of the Seek operation the FDC is in the FDC busy state, but during the execution phase it is in the non-busy state. While the FDC is in the non-busy state, another Seek command may be issued. In this manner parallel seek operations may be in operation on up to four floppy disk drives at once. The Main Status Register contains a flag for each drive (Table 4) that indicates whether the associated drive is currently operating in the seek mode. When a drive has completed a seek operation, the FDC generates an interrupt. In response to this interrupt, the system software must issue a Sense Interrupt Status command. During the result phase of this command, Status Register 0 (containing the drive number in bits 0 and 1) is read by the processor.

Recalibrate

This command causes the read/write head of the disk drive to retract to the track 0 position. The FDC clears the contents of its internal cylinder counter, and checks the status of the track 0 signal from the disk drive. As long as the track 0 signal is low, the direction signal remains high and step pulses are issued. When the track 0 signal goes high, the seek end flag (in Status Register 0) is set (high) and the command is terminated. If the track 0 signal is still low after 77 step pulses have been issued, the seek end and equipment check flags (in Status Register 0) are both set and the Recalibrate command is terminated.

Recalibrate commands for multiple drives can be overlapped in the same manner that Seek commands are overlapped.

Format Track

The Format Track command formats or "initializes" a track on a floppy disk by writing the ID field, gaps, and address marks for each sector. Before issuing the Format command, the Seek command must be used to position the read/write head over the correct cylinder. In addition, a table of ID field values (cylinder, head, and sector addresses and sector length code) must be prepared before the command is executed. During command execution, the FDC accesses the table and, using the values supplied, writes each sector on the track. The ID field address mark originates from the FDC and is written automatically as the first byte of each sector's ID field. The cylinder, head, and sector addresses are taken, in order, from the table. The ID field CRC character (derived from the data written in the first five bytes) is written as the last two bytes of the ID field. Gaps are written automatically by the FDC, with the length of the variable gap determined by one of the Format command parameters.

The data field address mark is generated by the FDC and is written automatically as the first byte of the data field. The data pattern specified in the command phase is written into each data byte of each sector. A CRC character is derived from the data address mark and the data written in the sector's data field. The two CRC bytes are appended to the last data byte.

The formatting of a track begins at the physical index mark. As previously mentioned, the order of sector assignment is taken directly from the formatting table. Four entries are required for each sector: a cylinder address, a head address, a sector address, and a sector length code. The cylinder address in the ID field should be equal to the cylinder address of the track currently being formatted.

The sector addresses must be unique (no two equal). The order of the sector entries in the table is the sequence in which sector numbers appear on the track when it is formatted. The number of entry sets (cylinder, head, and sector address and sector length code) must equal the number of sectors allocated to the track (specified in the command phase).

Since the sector address is supplied, in order, for each sector, tracks can be formatted sequentially (the first sector following the index mark is assigned sector address 1, the adjacent sector is assigned sector address 2, and so on) or sector numbers can be interleaved (see section 3) on a track.

Table 9 lists recommended gap sizes and sectors/track for various sector sizes.

Read Data

Nine (9) bytes are required to complete the command phase specification for the Read Data command. During the execution phase, the FDC loads the head (if it is in the unloaded state), waits the specified head load time (defined in the Specify command), and begins reading ID address marks and ID fields. When the requested sector address compares with the sector address read from the disk, the FDC outputs data (from the data field) byte-by-byte to the system. The Read Data command automatically operates in the multi-sector mode described earlier. In addition, multi-track operation may be specified by means of the MT command flag (Table 5). The amount of data that can be transferred with a single command to the FDC depends on the multi-track flag, the recording density flag, and the number of bytes per sector.

During the execution of read and write commands, the special sector size parameter (DTL) is used to temporarily alter the effective disk sector size. By setting the sector size code (N) to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the

actual sector (on the disk) is larger than DTL specifies, only the number of bytes specified by the DTL parameter are passed to the system; the remainder of the actual disk sector is not transferred (although the data is checked for CRC errors). Multi-sector read operations are performed in the same manner as they are when the sector size code is non-zero. (The N and DTL parameters are always present in the command sequence. DTL should be set to FF hexadecimal when N is not zero.)

If the FDC detects the physical index mark twice without finding the requested sector, the FDC sets the "sector not found error" flag (bit 2 in Status Register 1) and terminates the Read Data command. The interrupt code (bits 7 and 6 of Status Register 0) is set to "01." Note that the FDC searches for each sector in a multi-sector operation. Therefore, a "sector not found" error may occur after successful transfer of one or more preceding sectors. This error could occur if a particular sector number was not included when the track was first formatted or if a hard error on the disk has invalidated a sector ID field.

After reading the ID field and data field in each sector, the FDC checks the CRC bytes. If a read error is detected (incorrect CRC in the ID field), the FDC sets the "data error" flag in Status Register 1; if a CRC error occurs in the data field, the FDC sets the "data error" flag in Status Register 2. In either error condition, the FDC terminates the Read Data command. The interrupt code (bits 7 and 6 in Status Register 0) is set to "01."

If the FDC reads a deleted data address mark from the disk, and the skip flag (specified during the command phase) is not set, the FDC sets the "control mark" flag (bit 6 in Status Register 2) and terminates the Read Data command (after reading all the data in the sector). If the skip flag is set, the FDC skips the sector with the deleted data address mark and reads the next sector. Thus, the skip flag may be used to cause the FDC to ignore deleted data sectors during a multi-sector read operation.

Table 9. Sector Size Relationships

Format	Sector Size	N Sector Size Code	SC Sectors/ Track	GPL(1) Gap 3 Length	GPL(2) Gap 3 Length	Remarks
FM Mode	128 bytes/Sector	00	1A ₍₁₆₎	07 ₍₁₆₎	1B ₍₁₆₎	IBM Diskette 1
	256	01	0F ₍₁₆₎	0E ₍₁₆₎	2A ₍₁₆₎	IBM Diskette 2
	512	02	08	1B ₍₁₆₎	3A ₍₁₆₎	
MFM Mode	256	01	1A ₍₁₆₎	0E ₍₁₆₎	36 ₍₁₆₎	IBM Diskette 2D
	512	02	0F ₍₁₆₎	1B ₍₁₆₎	54 ₍₁₆₎	
	1024	03	08	35 ₍₁₆₎	74 ₍₁₆₎	IBM Diskette 2D

NOTES:

1. Suggested values of GPL in Read or Write commands to avoid splice point between data field and ID field of contiguous sectors.
2. Suggested values of GPL in Format command.

During disk data transfers between the FDC and the system, the FDC must be serviced by the system (processor or DMA controller) every 27 μs in the FM mode, and every 13 μs in the MFM mode. If the FDC is not serviced within this interval, the "overrun error" flag (bit 4 in Status Register 1) is set and the Read Data command is terminated.

If the processor terminates a read (or write) operation in the FDC, the ID information in the result phase is dependent upon the state of the multi-track flag and end of track byte. Table 11 shows the values for C, H, R, and N, when the processor terminates the command.

Write Data

Nine (9) bytes are required to complete the command phase specification for the Write Data command. During the execution phase the FDC loads the head (if it is in the unloaded state), waits the specified head load time (defined by the Specify command), and begins reading sector ID fields. When the requested sector address compares with the sector address read from the disk, the FDC reads data from the processor one byte at a time via the data bus and outputs the data to the data field of that sector. The CRC is computed on this data and two CRC bytes are written at the end of the data field.

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (incorrect CRC) in one of the ID fields, it sets the "data error" flag (bit 5 in Status Register 1) and terminates the Write Data command. The interrupt code (bits 7 and 6 in Status Register 0) is set to "01."

The Write Data command operates in much the same manner as the Read Data command. The following items are the same; refer to the Read Data command for details:

- Multi-sector and Multi-track operation
- Data transfer capacity
- "End of track error" flag
- "Sector not found error" flag
- "Data error" flag
- Head unload time interval
- ID information when the processor terminates the command (see Table 11)
- Definition of DTL when $N = 0$ and when $N \neq 0$

During the Write Data execution phase, data transfers between the processor and FDC must occur every 31 μs in the FM mode, and every 15 μs in the MFM mode. If the time interval between data transfers is

longer than this, the FDC sets the "overrun error" flag (bit 4 in Status Register 1) and terminates the Write Data command.

Read Deleted Data

This command operates in almost the same manner as the Read Data command operates. The only difference involves the treatment of the data address mark and the skip flag. When the FDC detects a data address mark at the beginning of a data field (and the skip flag is not set), the FDC reads all the data in the sector, sets the "control mark" flag (bit 6 in Status Register 2), and terminates the command. If the skip flag is set, the FDC skips the sector with the data address mark and continues reading at the next sector. Thus, the skip flag may be used to cause the FDC to read only deleted data sectors during a multi-sector read operation.

Write Deleted Data

This command operates in the same manner as the Write Data command operates except that a deleted data address mark is written at the beginning of the data field instead of the normal data address mark. This command is used to mark a bad sector (containing a hard error) on the floppy disk.

Read Track

The Read Track command is similar to the Read Data command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering the physical index mark, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID field or data field CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the values specified during the command phase. If the specified ID field information is not found on the track, the "sector not found error" flag (in Status Register 1) is set. Multitrack and skip operations are not allowed with this command.

This command terminates when the last sector on the track has been read. (The number of sectors on the track is specified by the end of track parameter byte during the command phase.) If the FDC does not find an ID address mark on the disk after it encounters the physical index mark for the second time, it sets the "missing address mark error" flag (bit 0 in Status Register 1) and terminates the command. The interrupt code (bits 7 and 6 of Status Register 0) is set to "01."

Read ID

The Read ID command transfers (reads) the first correct ID field from the current disk track (following the physical index mark) to the processor. If no correct ID address mark is found on the track, the "missing address mark error" flag is set (bit 0 in Status Register 1). If no data mark is found on the track, the "sector not found error" flag is also set (bit 2 in Status Register 1). Either error condition causes the command to be terminated.

Scan Commands

The Scan commands allow the data being read from the disk to be compared against data supplied by the system (by the processor in non-DMA mode, and by the DMA controller in DMA mode). The FDC compares the data on a byte-by-byte basis, and searches for a sector of data that meets the conditions of "disk data equal to system data", "disk data less than or equal to system data", or "disk data greater than or equal to system data". Simple binary (ones complement) arithmetic is used for comparison (FF = largest number, 00 = smallest number). If, after a complete sector of data is compared, the conditions are not met, the sector number is incremented by the scan sector increment (specified in the command phase), and the scan operation is continued. The scan operation continues until one of the following conditions occurs; the conditions for scan are met (equal, low, or high), the last sector on the track is reached, or the terminal count signal is received.

If the conditions for scan are met, the FDC sets the "scan hit" flag (bit 3 in Status Register 2) and terminates the Scan command. If the conditions for scan are not met between the starting sector and the last sector on the track (specified in the command phase), the FDC sets the "scan not satisfied" flag (bit 2 in Status Register 2) and terminates the Scan command. The receipt of a terminal count signal from the processor or DMA controller during the scan operation will cause the FDC to complete the comparison of the particular byte which is in process, and to terminate the command. Table 10 shows the status of the "scan hit" and "scan not satisfied" flags under various scan termination conditions.

If the FDC encounters a deleted data address mark in one of the sectors and the skip flag is low, it regards the sector as the last sector on the cylinder, sets the "control mark" flag (bit 6 in Status Register 2) and terminates the command. If the skip flag is high, the FDC skips the sector with the deleted address mark, and reads the next sector. In this case, the FDC also sets the

Table 10. Scan Status Codes

Command	Status Register 2		Comments
	Bit 2 = SN	Bit 3 = SH	
Scan Equal	0	1	DFDD = DProcessor
	1	0	DFDD ≠ DProcessor
Scan Low or Equal	0	1	DFDD = DProcessor
	0	0	DFDD < DProcessor
	1	0	DFDD > DProcessor
Scan High or Equal	0	1	DFDD = DProcessor
	0	0	DFDD > DProcessor
	1	0	DFDD < DProcessor

"control mark" flag (bit 6 in Status Register 2) in order to show that a deleted sector had been encountered.

NOTE:

During scan command execution, the last sector on the track must be read for the command to terminate properly. For example if the scan sector increment is set to 2, the end of track parameter is set to 26, and the scan begins at sector 21, sectors 21, 23, and 25 will be scanned. The next sector, 27 will not be found on the track and an abnormal command termination will occur. The command would be completed in a normal manner if either a) the scan had started at sector 20 or b) the end of track parameter had been set to 25.

During the Scan command, data is supplied by the processor or DMA controller for comparison against the data read from the disk. In order to avoid having the "overrun error" flag set (bit 4 in Status Register 1), it is necessary to have the data available in less than 27 μ s (FM Mode) or 13 μ s (MFM Mode). If an overrun error occurs, the FDC terminates the command.

Invalid Commands

If an invalid (undefined) command is sent to the FDC, the FDC will terminate the command. No interrupt is generated by the 8272 during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both set indicating to the processor that the 8272 is in the result phase and the contents of Status Register 0 must be read. When the processor reads Status Register 0 it will find an 80H code indicating that an invalid command was received.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate interrupt; otherwise the FDC will consider the next command to be an invalid command. Also, when the last "hidden" interrupt has been serviced, further Sense Interrupt Status commands will result in invalid command codes.

Table 11. ID Information When Processor Terminates Command

MT	EOT	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	C + 1	NC	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	NC	R = 01	NC
1	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	NC	LSB	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	LSB	R = 01	NC

NOTES:

1. NC (No Change): The same value as the one at the beginning of command execution.
2. LSB (Least Significant Bit): The least significant bit of H is complemented.

In some applications the user may wish to use this command as a No-Op command to place the FDC in a stand-by or no operation state.

5.0 THE DATA SEPARATOR

As briefly discussed in Section 2, LSI disk controllers such as the 8272 require external circuitry to generate a data window signal. This signal is used within the FDC to isolate the data bits contained within the READ DATA input signal from the disk drive. (The disk READ DATA signal is a composite signal constructed from both clock and data information.) After isolating the data bits from this input signal, the FDC assembles the data bits into 8-bit bytes for transfer to the system processor or memory.

Single Density

In a single-density (FM) recording (Figure 3), the bit cell is 4 microseconds wide. Each bit cell contains a clock bit at the leading edge of the cell. The data bit (if present) is always located at the center of the cell. The job of data separation is relatively straightforward for single-density; simply generate a data window 2 μ s wide starting 1 μ s after each clock bit. Since every cell has a clock bit, a fixed window reference is available for every data bit and because the window is 2 μ s wide, a slightly shifted data bit will still remain within the data window.

A single-density data separator with these specifications may be easily generated using a digital or analog one-shot triggered by the clock bit.

Double-Density

Double-density (MFM) bit cells are reduced to $2 \mu\text{s}$ (in order to double the disk data storage capacity). Clock bits are inserted into the data stream only if data bits are not present in both the current and preceding bit cells (Figure 3). The data bit (if present) still occurs at the center of the bit cell and the clock bit (if present) still occurs at the leading edge of the bit cell.

MFM data separation has two problems. First, only some bit cells contain a clock bit. In this manner, MFM encoding loses the fixed bit cell reference pulse present in FM encoding. Second, the bit cell for MFM is one-half the size of the bit data for FM. This shorter bit cell means that MRM cannot tolerate as large a playback data-shift (as FM can tolerate) without errors.

Since most playback data-shift is predictable, the FDC can precompensate the write data stream so that data/clock pulses will be correctly positioned for subsequent playback. This function is completely controlled by the FDC and is only required for MFM recording. During write operations, the FDC specifies an early, normal, or late bit positioning. This timing information is specified with respect to the FDC write clock. Early and late timing is typically 125 ns to 250 ns before or after the write clock transition (depending on disk drive requirements).

The data separator circuitry for double-density recording must continuously analyze the total READ DATA stream, synchronizing its operation (window generation) with the actual clock/data bits of the data stream. The data separation circuit must track the disk input

data frequency very closely—unpredictable bit shifts leave less than 50 ns margin to the window edges.

Phase-Locked Loop

Only an analog phase-locked loop (PLL) can provide the reliability required for a double-density data separation circuit. (A phase-locked loop is an electronic circuit that constantly analyzes the frequency of an input signal and locks another oscillator to that frequency.) Using analog PLL techniques, a data separator can be designed with ± 1 ns resolution (this would require a 100 MHz clock in a digital phase-locked loop). The analog PLL determines the clock and data bit positions by sampling each bit in the serial data stream. The phase relationship between a data bit and the PLL generated data window is constantly fed back to adjust the position of the data window, enabling the PLL to track input data frequency changes, and thereby reliably read previously recorded data from a floppy disk.

PLL Design

A block diagram of the phase-locked loop described in this application note is shown in Figure 7. Basically, the phase-locked loop operates by comparing the frequency of the input data (from the disk drive) against the frequency of a local oscillator. The difference of these frequencies is used to increase or decrease the frequency of the local oscillator in order to bring its frequency closer to that of the input. The PLL synchronizes the local

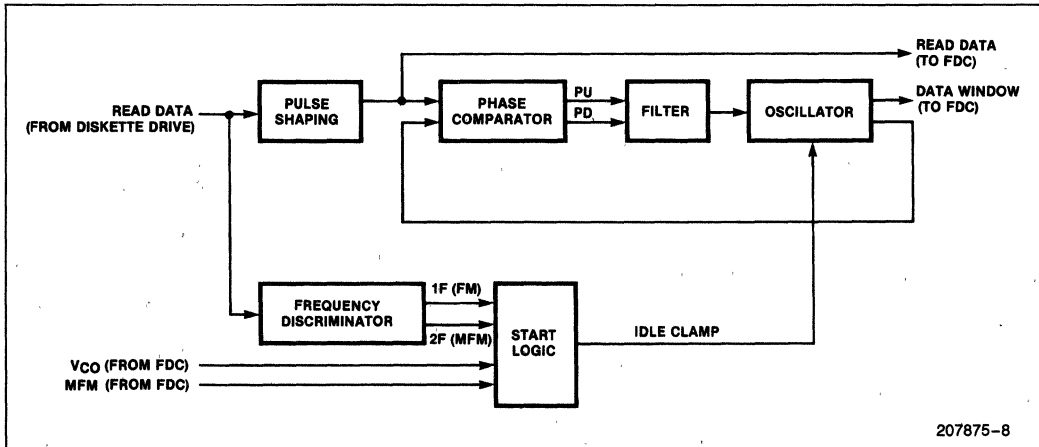


Figure 7. Phase-Locked Loop Data Separator

oscillator to the frequency of the input during the all “zeroes” synchronization field on the floppy disk (immediately preceding both the ID field and the data field.

The PLL consists of nine ICs and is located on page 3 of the schematics in the Appendix. The 8272 VCO output essentially turns the PLL circuitry on and off. When the PLL is off, it “idles” at its center frequency. The VCO turns the PLL on only when valid data is being received from the disk drive. The VCO turns the PLL on after the read/write head has been loaded and the head load time has elapsed. The PLL is turned off in the gap between the ID field and the data field and in the gap after the data field (before the next sector ID field). The GPL parameter in the FDC read and write commands specifies the elapsed time (number of data bytes) that the PLL is turned off in order to blank out discontinuities that appear in the gaps when the write current is turned on and off. The PLL operates with either MFM or FM input data. The MFM output from the FDC controls the PLL operation frequency.

The PLL consists of six functional blocks as follows:

- 1) Pulse Shaping—A 96LS02 senses a READ DATA pulse and provides a clean output signal to the FDC and to the PLL Phase Comparator and Frequency Discriminator circuitry.
- 2) Phase Comparator—The phase difference between the PLL oscillator and the READ DATA input is compared. Pump up (PU) and pump down (PD) error signals are derived from this phase difference and output to the filter. If there is no phase difference between the PLL oscillator and the READ DATA input, the PU and PD pulse widths are equal. If the READ DATA pulse occurs early, the PU duration is shorter than the PD duration. If the data pulse occurs late, the PU duration is longer than the PD duration.
- 3) Filter—This analog circuit filters the PU and PD pulses into an error voltage. This error voltage is buffered by an LM358 operational amplifier.
- 4) PLL Oscillator—This oscillator is composed of a 74LS393, 74LS74, and 96LS02. The oscillator frequency is controlled by the error voltage output by the filter. This oscillator also generates the data window signal to the FDC.
- 5) Frequency Discriminator—This logic tracks the READ DATA input from the disk drive and discriminates between the synchronization gap for FM recording (250 KHz) and the gap for MFM recording (500 KHz). Synchronization gaps immediately precede address marks.
6. Start Logic—The function of this logic is to clamp the PLL oscillator to its center frequency (2 MHz) until the FDC VCO signal is enabled and a valid data pattern is sensed by the frequency discriminator. The start logic (consisting of a 74LS393 and 74LS74) ensures that the PLL oscillator is started with zero phase error.

PLL Adjustments

The PLL must be initially adjusted to operate at its center frequency with the VCO output off and the adjustment jumper removed. The 5K timepot should be adjusted until the frequency at the test point (Q output of the 96LS02) is 2 MHz. The jumper should then be replaced for normal operation.

PLL Design Details

The following paragraphs describe the operational and design details of the phase-locked loop data separator illustrated in the appendix. Note that the analog section is operated from a separately filtered +5V supply.

Initialization

As long as the 8272 maintains a low VCO signal, the data separator logic is “turned off”. In this state, the PLL oscillator (96LS02) is not oscillating and therefore the 2XBR signal is constantly low. In addition, the pump up (PU) and pump down (PD) signals are inactive (PU low and PD high), the CNT8 signal is inactive (low), and the filter input voltage is held at 2.5 volts by two 1 M Ω resistors between ground and +5 volts.

Floppy Disk Data

The data separator frequency discriminator, the input pulse shaping circuitry, and the start logic are always enabled and respond to rising edges of the READ DATA signal. The rising edge of every data bit from the disk drive triggers two pulse shaping one-shots. The first pulse shaper generates a stable and well-defined 200 ns read data pulse for input to the 8272 and other portions of the data separator logic. The second one-shot generates a 2.5 μ s data pulse that is used for input data frequency discrimination.

The frequency discriminator operates as illustrated in Figure 8. The 2F output signal is active (high) during reception of valid MFM (double-density) sync fields on the disk while the 1F signal is active (high) during reception of valid FM (single-density) sync fields. A multiplexer (controlled by the 8272 MFM signal) selects the appropriate 1F or 2F signal depending on the programmed mode.

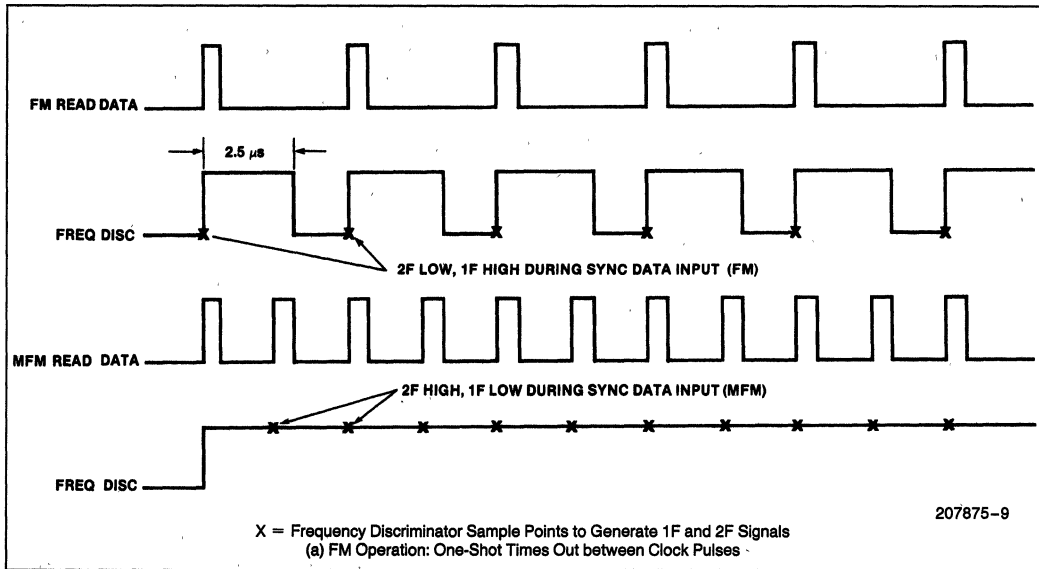


Figure 8. Input Data Frequency Discrimination

Startup

The data separator is designed to require reception of eight valid sync bits (one sync byte) before enabling the PLL oscillator and attempting to synchronize with the input data stream (see Figure 9). This delay ensures that the PLL will not erroneously synchronize outside a valid sync field in the data stream if the VCO signal is enabled slightly early. The sync bit counter is asynchronously reset by the CNTEN signal when valid sync data is not being received by the drive.

Once the VCO signal is active and eight sync bits have been counted, the CNT8 signal is enabled. This signal turns on the PLL oscillator. Note that this oscillator starts synchronously with the rising edge of the disk input data (because CNT8 is synchronous with the data rising edge) and the oscillator also starts at its center frequency of 2 MHz (because the LM348 filter input is held at its center voltage of approximately 2.5 volts). This frequency is divided by two and four to generate the 2XBR signal (1 MHz for MFM and 500 KHz for FM).

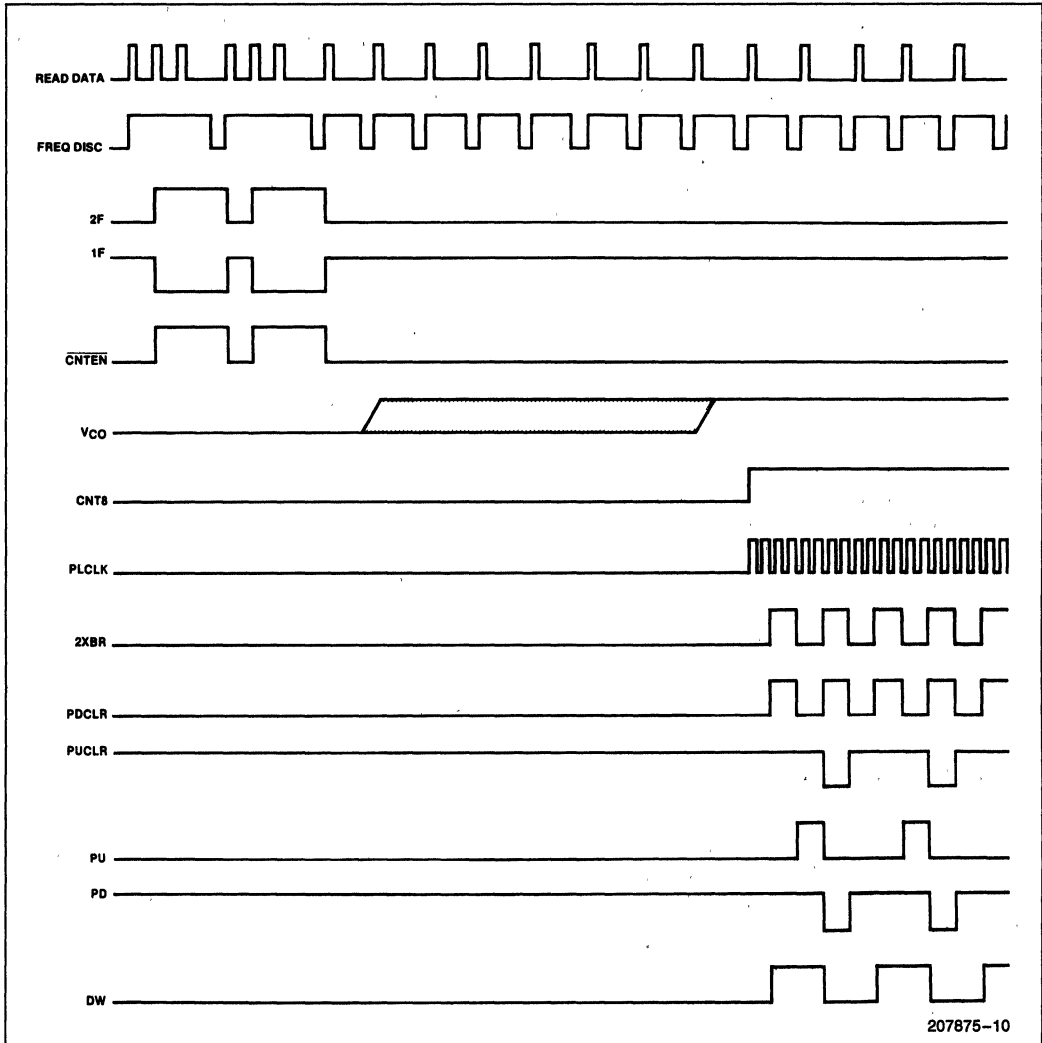


Figure 9. Typical Data Separator Startup Timing Diagram

PLL Synchronization

At this point, the PLL is enabled and begins to synchronize with the input data stream. This synchronization is accomplished very simply in the following manner. The pump up (PU) signal is enabled on the rising edge of the READ DATA from the disk drive. (When the PLL is synchronized with the data stream, this point will occur at the same time as the falling edge of the 2XBR signal as shown in Figure 9). The PU Signal is turned off and the PD signal is activated on the next rising edge of the 2XBR clock. With this scheme, the difference between PU active time and the PD active time is equal to the difference between the input bit rate

and the PLL clock rate. Thus, if PU is turned on longer than PD is on, the input bit rate is faster than the PLL clock.

As long as PU and PD are both inactive, no charge is transferred to or from the LM358 input holding capacitor, and the PLL output frequency is maintained (the LM358 operational amplifier has a very high input impedance). Whenever PU is turned on, current flows from the +5 volt supply through a 20K resistor into the holding capacitor. When the PD signal is turned on, current flows from the holding capacitor to ground through a 20K resistor. In this manner, both the pump up and pump down charging rates are balanced.

The change in capacitor charge (and therefore voltage) after a complete PU/PD cycle is proportional to the difference between the PU and PD pulse widths and is also proportional to the frequency difference between the incoming data stream and the PLL oscillator. As the capacitor voltage is raised (PU active longer than PD), the PLL oscillator time constant (RC of the 96LS02) is modified by the filter output (LM358) to raise the oscillator frequency. As the capacitor voltage is lowered (PD active longer than PD), the oscillator frequency is lowered. If both frequencies are equal, the voltage on the holding capacitor does not change, and the PLL oscillator frequency remains constant.

6.0 AN INTELLIGENT DISKETTE DATA BASE SYSTEM

The system described in this application note is designed to function as an intelligent data base controller. The schematics for this data base unit are presented in Appendix A; a block diagram of the unit is illustrated in Figure 10. As designed, the unit can access over four million bytes of mass storage on four floppy disk drives (using a single 8272 FDC); the system can easily be expanded to four FDC devices (and 16 megabytes of on-line disk storage). Three serial data links are also included. These data links may be used by CRT terminals or other microprocessor systems to access the data base.

Processor and Memory

A high-performance 8088 eight-bit microprocessor (operating at 5 MHz with no wait states) controls system operation. The 8088 was selected because of its memory addressing capabilities and its sophisticated string handling instructions. These features improve the speed of data base search operations. In addition, these capabilities allow the system to be easily upgraded with additional memory, disk drives, and if required, a bubble memory or Winchester disk unit.

The schematics for the basic design provide 8K bytes of 2732A high-speed EPROM program storage and 8K bytes of disk directory and file buffer RAM. This memory can easily be expanded to 1 megabyte for performance upgrades.

An 8259A Programmable Interrupt Controller (PIC) is also included in the design to field interrupts from both the serial port and the FDC. This interrupt controller provides a large degree of programming flexibility for the implementation of data base function in an asynchronous, demand driven environment. The PIC allows the system to accumulate asynchronous data base requests from all serial I/O ports while previously specified data base operations are currently in progress. This feature is made possible by the ability of the 8251A RXRDY signal to cause a processor interrupt. After receiving this interrupt, the processor can temporarily halt work on existing requests and enter the incoming information into a data base request buffer. Once the information has been entered into the buffer, the system can resume its previous processing.

In addition, the PIC permits some portions of data base requests to be processed in parallel. For example, once a disk record has been loaded into a memory buffer, a memory search can proceed in parallel with the loading of the next record. After the FDC completes the record transfer, the memory search will be interrupted and the processor can begin another disk transfer before resuming the memory search.

The bus structure of the system is split into three functional buffered units. A 20-bit address from the processor is latched by three-state transparent 74LS373 devices. When the processor is in control of the address and data busses, these devices are output enabled to the system buffered address bus. All I/O devices are placed directly on the local data bus. Finally, the memory data bus is isolated from the local data bus by an 8286 octal transceiver. The direction of this transceiver is determined by the Memory Read signal, while its output enable is activated by a Memory Read or Memory Write command.

no wait states when 2114-3 (150 ns) or faster static RAM is used. In operation, the 8272 presents a DMA request to the 8237 for every byte of data to be transferred. This request causes the 8273 to present a HOLD request to the 8088. As soon as the 8088 is able to relinquish data/address bus control, the processor signals a HOLD acknowledge to the 8237. The 8237 then assumes control over the data and address busses. After latching the address for the DMA transfer, the 8237 generates simultaneous I/O Read and Memory Write commands (for a disk read) or simultaneous I/O Write and Memory Read commands (for a disk write). At the same time, the 8272 is selected as the I/O device by means of the DMA acknowledge signal from the 8237. After this single byte has been transferred between the FDC and memory, the DMA controller releases the data/address busses to the 8088 by deactivating the HOLD request. In a short period of time (13 μ s for double-density and 27 μ s for single-density) the FDC requests a subsequent data transfer. This transfer occurs in exactly the same manner as the previous transfer. After all data transfers have been completed (specified by the word count programmed into the 8237 before the FDC operation was initiated), the 8237 signals a terminal count (EOP pin). This terminal count signal informs the 8272 that the data transfer is complete. Upon reception of this terminal count signal, the 8272 halts DMA requests and initiates an "operation complete" interrupt.

Since the system is designed for a 20-bit addressing, a four-bit DMA-address latch is included as a processor addressable I/O port. The processor writes the upper four DMA address bits before a data transfer. When the DMA controller assumes bus control, the contents of this latch are output enabled on the upper four bits of the address bus. The only restriction in the use of this address latch is that a single disk read or write transfer cannot cross a 64K memory boundary.

Disk Drive Interface

The 8272 FDC may be interfaced to a maximum of four eight-inch floppy disk drives. Both single- and double-density drives are accommodated using the data separation circuit described in section 5. In addition, single- or dual-sided disk drives may be used. The 8272 is driven by an 8 MHz crystal controller clock produced by an 8224 clock generator.

Drive select signals are decoded by means of a 74LS139 from the DS0, DS1 outputs of the FDC. The fault reset, step, low current, and direction outputs to the disk drives are generated from the FR/STEP, LCT/DIR, and RW/SEEK FDC output signals by means of a 74LS240. The other half of the 74LS240 functions as an input multiplexer for the disk write protect, two-sided, fault, and track zero status signals. These signals are multiplexed into the WP/TS and FLT/TRK0 inputs to the 8272.

The 8272 write clock (WR CLK) is generated by a ring counter/multiplexer combination. The write clock frequency is 1 MHz for MFM recording and 500 KHz for FM recording (selected by the MFM output of the 8272). The pulse width is a constant 250 ns. The write clock is constantly generated and input to the FDC (during both read and write operations). The FDC write enable output (WE) is transmitted directly to the write gate disk drive input.

Write data to the disk drive is preshifted (according to the PS0, PS1 FDC outputs) by the combination of a 74LS175 four-bit latch and a 74LS153 multiplexer. The amount of preshift is completely controlled within the 8272 FDC. Three cases are possible: the data may be written one clock cycle early, one clock cycle late, or with no preshift. The data preshift circuit is activated by the FDC only in the double-density mode. The preshift is required to cancel predictable playback data shifts when recorded data is later read from the floppy disk.

A single 50-conductor flat cable connects the board to the floppy disk drives. FDC outputs are driven by 7438 open collector high-current line-drivers. These drivers are resistively terminated on the last disk drive by means of a 150 Ω resistor to +5V. The line receivers are 7414 Schmitt triggered inverters with 150 Ω pull-up resistors on board.

7.0 SPECIAL CONSIDERATIONS

This section contains a quick review of key features and issues, most of which have been mentioned in other sections of this application note. Before designing with the 8272 FDC, it is advisable that the information in this section be completely understood.

7.1 Multi-Sector Transfers

The 8272 always operates in a multi-sector transfer mode. The 8272 continues to transfer data until the TC input is activated. In a DMA configuration, the TC input of the 8272 must always be connected to the EOP/TC output of the DMA controller. When multiple DMA channels are used on a single DMA controller, EOP must be gated with the select signal for the proper FDC. If the TCD signal is not gated, a terminal count on another channel will abort FDC operation.

In a processor driven configuration with no DMA controller, the system must count the transfers and supply a TC signal to the FDC. In a DMA environment, OR-ing a programmable TC with the TC from the DMA controller is a convenient means of ensuring that the processor may always gain control of the FDC (even if the diskette system hangs up in an abnormal manner).

7.2 Processor Command/Result Phase Interface

In the command phase, the processor must write the exact number of parameters in the exact order shown in Table 5. During the result phase, the processor must read the complete result status. For example, the Format Track command requires six command bytes and presents seven result bytes. The 8272 will not accept a new command until all result bytes are read. Note that the number of command and result bytes varies from command-to-command. Command and result phases cannot be shortened.

During both the command and result phases, the Main Status Register must be read by the processor before each byte of information is read from, or written to, the FDC Data Register. Before each command byte is written, DIO (bit 6) must be low (indicating a data transfer from the processor) and RQM (bit 7) must be high (indicating that the FDC is ready for data). During the result phase, DIO must be high (indicating a data transfer to the processor) and RQM must also be high (indicating that data is ready for the processor).

NOTE:

After the 8272 receives a command byte, the RQM flag may remain set for 12 microseconds (with an 8 MHz clock). Software should not attempt to read the Main Status Register before this time interval has elapsed; otherwise, the software will erroneously assume that the FDC is ready to accept the next byte.

7.3 Sector Sizes

The 8272 does not support 128 byte sectors in the MFM (double-density) mode.

7.4 Write Clock

The FDC Write Clock input (WR CLK) must be present at all times.

7.5 Reset

The FDC Reset input (RST) must be held active during power-on reset while the RD and WR inputs are active. If the reset input becomes inactive while RD and WR are still active, the 8272 enters the test mode. Once activated, the test mode can only be deactivated by a power-down condition.

7.6 Drive Status

The 8272 constantly polls (starting after the power-on reset) all drives for changes in the drive ready status. At

power-on, the FDC assumes that all drives are not ready. If a drive application requires that the ready line be strapped active, the FDC will generate an interrupt immediately after power is applied.

7.7 Gap Length

Only the gap 3 size is software programmable. All other gap sizes are fixed. In addition, different gap 3 sizes must be specified in format, read, write, and scan commands. Refer to Section 3 and Table 9 for gap size recommendations.

7.8 Seek Command

The drive busy flag in the Main Status Register remains set after a Seek command is issued until the Sense Interrupt Status command is issued (following reception of the seek complete interrupt).

The FDC does not perform implied seeks. Before issuing data read or write commands, the read/write head must be positioned over the correct cylinder. If the head is not positioned correctly, a cylinder address error is generated.

After issuing a step pulse, the 8272 resumes drive status polling. For correct stepper operation in this mode, the stepper motor must be constantly enabled. (Most drives provide a jumper to permit the stepper motor to be constantly enabled.)

7.9 Step Rate

The 8272 can emit a step pulse that is one millisecond faster than the rate programmed by the SRT parameter in the Specify command. This action may cause subsequent sector not found errors. The step rate time should be programmed to be 1 ms longer than the step rate time required by the drive.

7.10 Cable Length

A cable length of less than 10 feet is recommended for drive interfacing.

7.11 Scan Commands

The current 8272 has several problems when using the scan commands. These commands should not be used at this time.

7.12 Interrupts

When the processor receives an interrupt from the FDC, the FDC may be reporting one of two distinct events:

- a) The beginning of the result phase of a previously requested read, write, or scan command.
- b) An asynchronous event such as a seek/recalibrate completion, an attention, an abnormal command termination, or an invalid command.

These two cases are distinguished by the FDC busy flag (bit 4) in the Main Status Register. If the FDC busy flag is high, the interrupt is of type (a). If the FDC busy flag is low, the interrupt was caused by an asynchronous event (b).

A single interrupt from the FDC may signal more than one of the above events. After receiving an interrupt, the processor must continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are ferreted out and serviced.

7.13 Skip Flag (SK)

The skip flag is used during the execution of Read Data, Read Deleted Data, Read Track, and various Scan commands. This flag permits the FDC to skip unwanted sectors on a disk track.

When performing a Read Data, Read Track, or Scan command, a high SK flag indicates that the FDC is to skip over (not transfer) any sector containing a deleted data address mark. A low SK flag indicates that the FDC is to terminate the command (after reading all the data in the sector) when a deleted data address mark is encountered.

When performing a Read Deleted Data command, a high SK flag indicates that sectors containing normal data address marks are to be skipped. Note that this is just the opposite situation from that described in the last paragraph. When a data address mark is encountered during a Read Deleted Data command (and the SK flag is low), the FDC terminates the command after reading all the data in the sector.

7.14 Bad Track Maintenance

The 8272 does not internally maintain bad track information. The maintenance of this information must be performed by system software. As an example of typical bad track operation, assume that a media test determines that track 31 and track 66 of a given floppy disk are bad. When the disk is formatted for use, the system software formats physical track 0 as logical cylinder 0 ($C = 0$ in the command phase parameters), physical track 1 as logical track 1 ($C = 1$), and so on, until physical track 30 is formatted as logical cylinder 30 ($C = 30$). Physical track 31 is bad and should be formatted as logical cylinder FF (indicating a bad track). Next, physical track 32 is formatted as logical cylinder 31, and so on, until physical track 67 is formatted as logical cylinder 64. Next, bad physical track 66 is formatted as logical cylinder FF (another bad track marker), and physical track 67 is formatted as logical cylinder 65. This formatting continues until the last physical track (77) is formatted as logical cylinder 75. Normally, after this formatting is complete, the bad track information is stored in a prespecified area on the floppy disk (typically in a sector on track 0) so that the system will be able to recreate the bad track information when the disk is removed from the drive and reinserted at some later time.

To illustrate how the system software performs a transfer operation disk with bad tracks, assume that the disk drive head is positioned at track 0 and the disk described above is loaded into the drive. If a command to read track 36 is issued by an application program, the system software translates this read command into a seek to physical track 37 (since there is one bad track between 0 and 36, namely 31) followed by a read of logical cylinder 36. Thus, the cylinder parameter C is set to 37 for the Seek command and 36 for the Read Sector command.

7.15 Head Load versus Head Settle Times

The 8272 does not permit separate specification of the head load time and the head settle time. When the Specify command is issued for a given disk drive, the proper value for the HLT parameter is the maximum of the head load time and the head settle time.

APPENDIX A SCHEMATICS

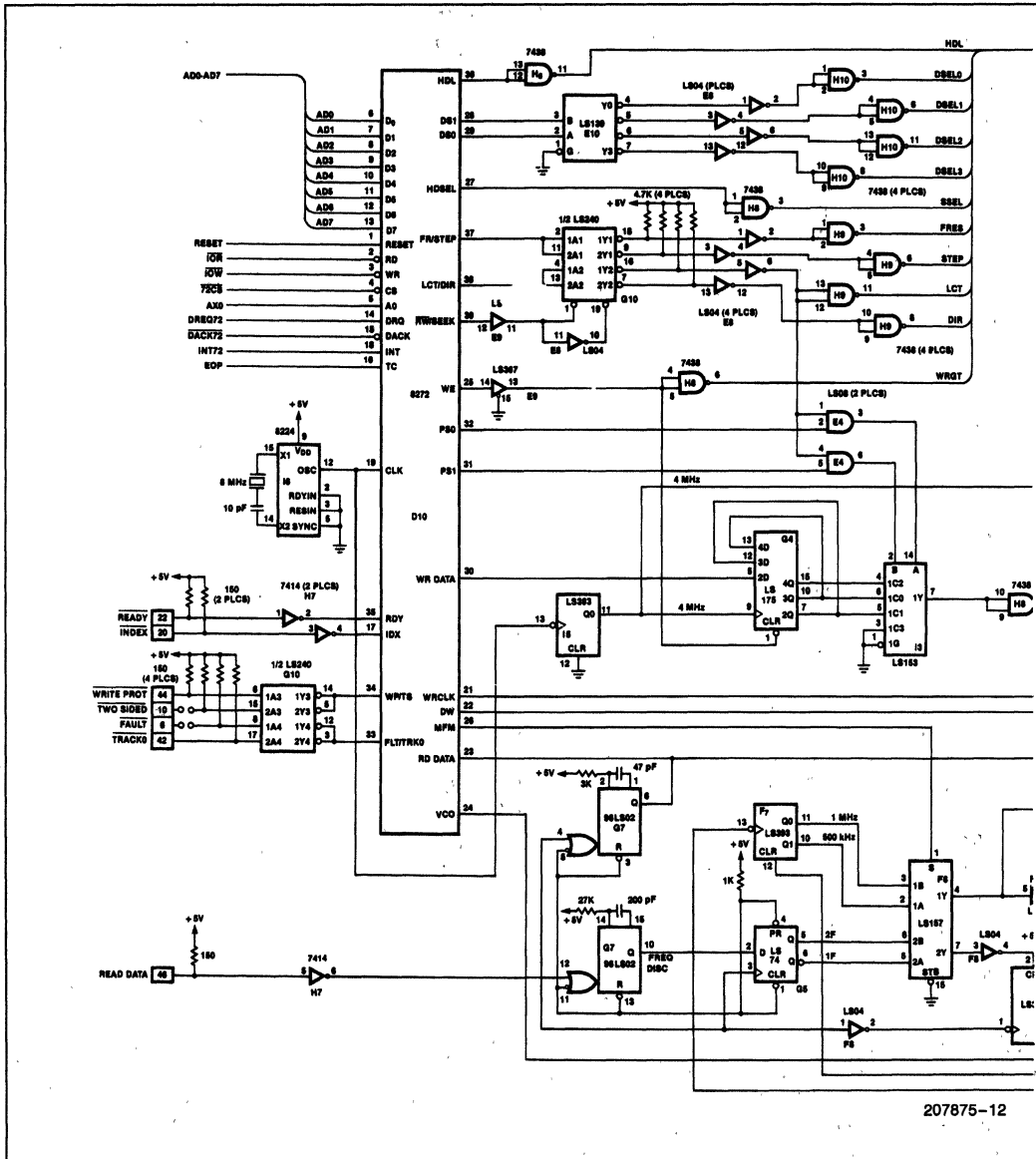
Power Distribution

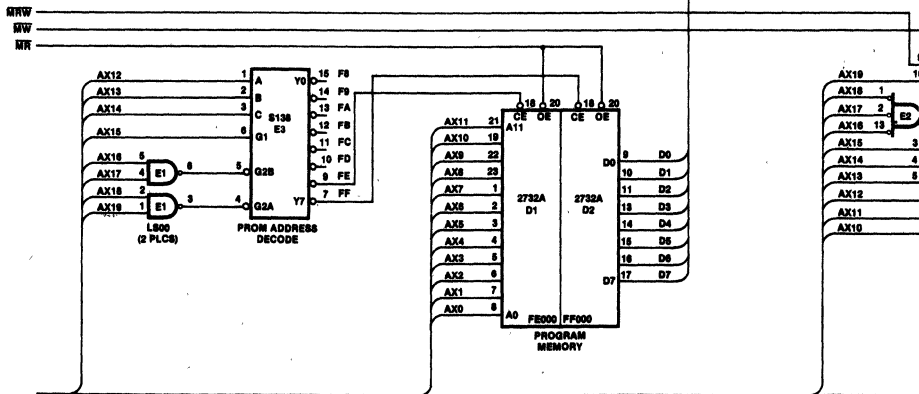
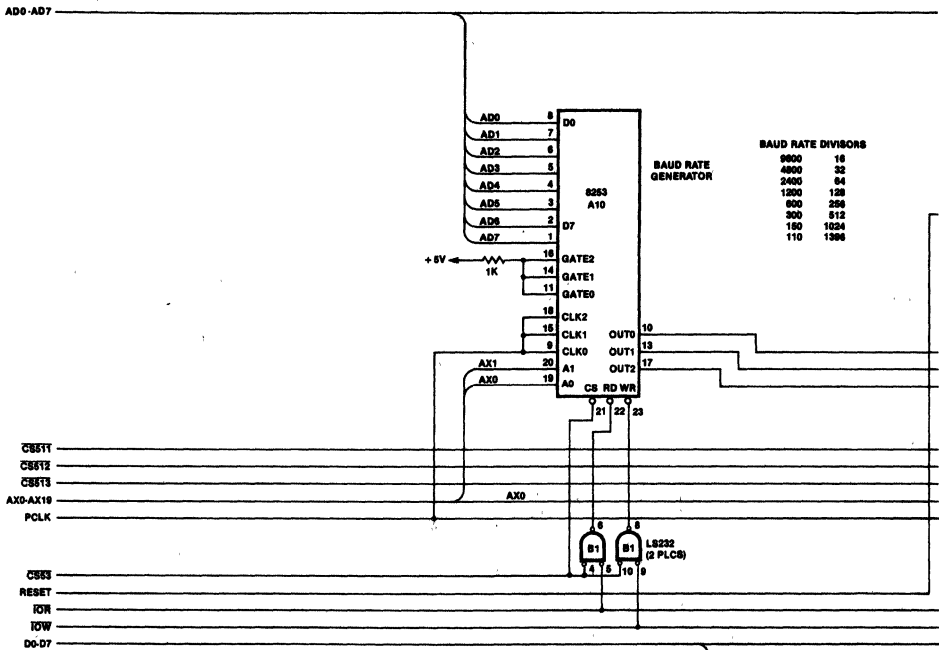
Part	Ref Desig	+ 5	GND	+ 12	- 12
8088	A2	40	1, 20		
8224	I6	9, 16	8		
8237-2	A6	31	20		
8251A	A9, B9, C9	26	4		
8253-5	A10	24	12		
8259A	B10	28	14		
8272	D10	40	20		
8284	A1	18	9		
8286	B6, F4	20	10		
2114	F1, F2, G1, G2, H1, H2, I1, I2	18	9		
2732A	D1, D2	24	12		
74LS00	E1	14	7		
74LS04	B2, E6, E8, F8	14	7		
74LS27	E2, E5	14	7		
74LS32	B1	14	7		
74LS74	A4, G5, H6	14	7		
74LS138	F3	16	8		
74LS139	E10	16	8		
74LS153	I3	16	8		
74LS157	F6	16	8		
74LS164	F5	14	7		
74LS173	G3	16	8		
74LS175	G4	16	8		
74LS240	G10	20	10		
74LS257	D3	16	8		
74LS367	C3, E9	16	8		
74LS373	B4, C4, D4, C6	20	10		
74LS393	I5, F7	14	7		
74S08	E4	14	7		
74S138	D6, E3	16	8		
7414	H7	14	7		
7438	H8, H9, H10	14	7		
1488	H3		7	14	1
1489	H4	14	7		
96LS02	G7	16	8		
96LS02	G6			16	8
LM358	H5			8	4

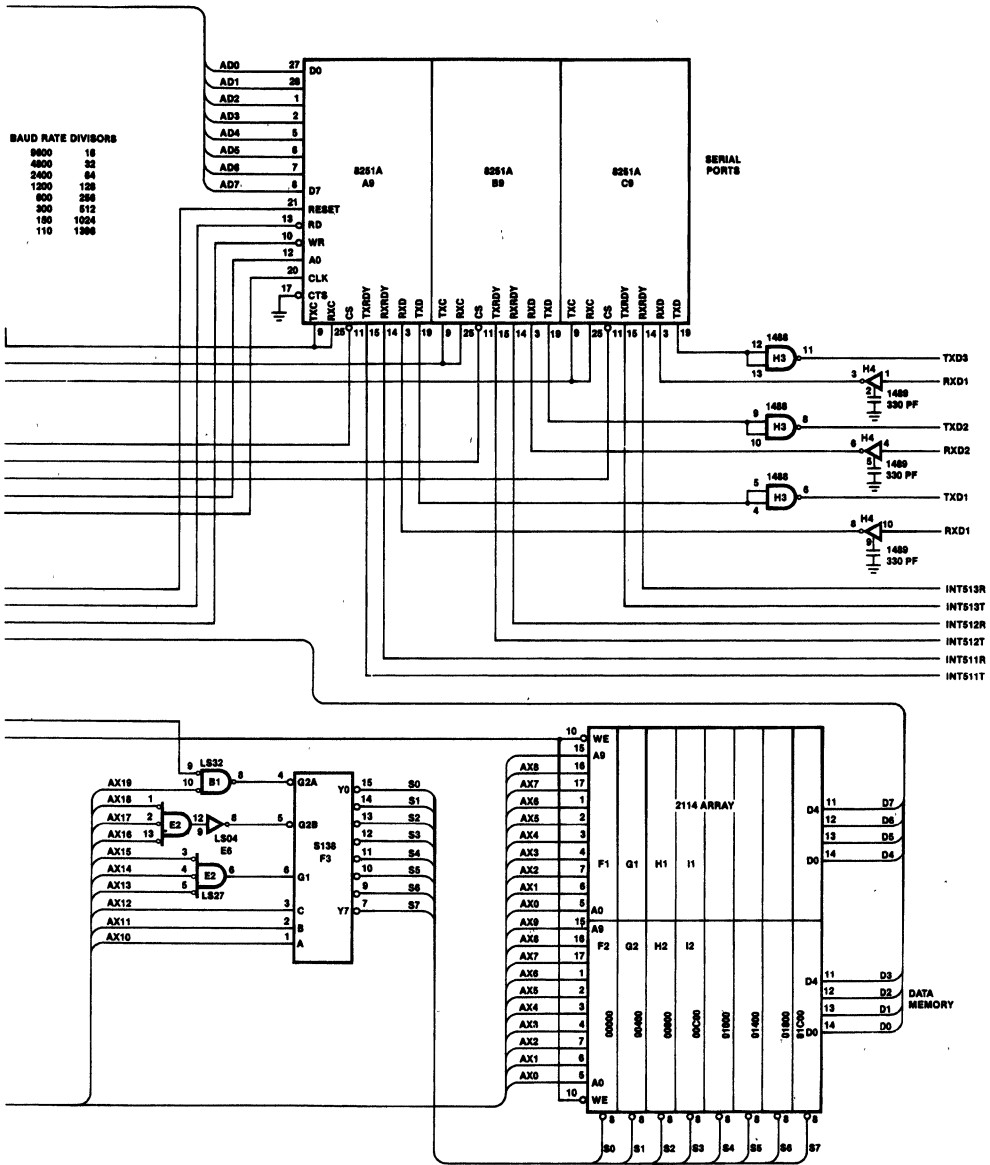
REFERENCES

- 1) Intel, "8272 Single/Double Density Floppy Disk Controller Data Sheet," Intel Corporation, 1980.
- 2) Intel, *iSBC 208 Hardware Reference Manual*, Manual Order No. 143078, Intel Corporation, 1980.
- 3) Intel, *iSBC 204 Flexible Diskette Controller Hardware Reference Manual*, Manual Order No. 9800568A, Intel Corporation, 1978.
- 4) Shugart, *SA800/801 Diskette Storage Drive OEM Manual*, Part No. 50574, Shugart Associates, 1977.
- 5) Shugart, *SA800/801 Diskette Storage Drive Theory of Operations*, Part No. 50664, Shugart Associates, 1977.
- 6) Shugart, *SA800 Series Diskette Storage Drive Double Density Design Guide*, Part No. 39000, Shugart Associates, 1977.

- 7) Shugart, "Application Notes for Shugart Dual VFO," Part No. 39101, Shugart Associates, 1980.
- 8) Pertec, "Soft-sector Formatting for PERTEC Flexible Disk Drives," Pertec Application Note, 1977.
- 9) Ausing Lesea and Rodney Zaks, "Floppy-disc Controller Design Must Begin With the Basics," EDN, May 20, 1978.
- 10) John Hoepfner and Larry Wall, "Encoding/Decoding Techniques Double Floppy Disc Capacity," Computer Design, Feb 1980.
- 11) John Zarrella, *System Architecture*, Microcomputer Applications, 1980.







207875-15



November 1986

**Software Design and
Implementation of Floppy Disk
Subsystems**

Order Number: 207885-001

1.0 INTRODUCTION

Disk interface software is a major contributor to the efficient and reliable operation of a floppy disk subsystem. This software must be a well-designed compromise between the needs of the application software modules and the capabilities of the floppy disk controller (FDC). In an effort to meet these requirements, the implementation of disk interface software is often divided into several levels of abstraction. The purpose of this application note is to define these software interface levels and describe the design and implementation of a modular and flexible software driver for the 8272 FDC. This note is a companion to AP-116, "An Intelligent Data Base System Using the 8272".

The Physical Interface Level

The software interface level closest to the FDC hardware is referred to as the physical interface level. At this level, interface modules (often called disk drivers or disk handlers) communicate directly with the FDC device. Disk drivers accept floppy disk commands from other software modules, control and monitor the FDC execution of the commands, and finally return operational status information (at command termination) to the requesting modules.

In order to perform these functions, the drivers must support the bit/byte level FDC interface for status and data transfers. In addition, the drivers must field, classify, and service a variety of FDC interrupts.

The Logical Interface Level

System and application software modules often specify disk operation parameters that are not directly compatible with the FDC device. This software incompatibility is typically caused by one of the following:

- 1) The change from an existing FDC to a functionally equivalent design. Replacing a TTL based controller with an LSI device is an example of a change that may result in software incompatibilities.
- 2) The upgrade of an existing FDC subsystem to a higher capability design. An expansion from a single-sided, single-density system to a dual-sided, double-density system to increase data storage capacity is an example of such a system change.
- 3) The abstraction of the disk software interface to avoid redundancy. Many FDC parameters (in particular the density, gap size, number of sectors per track and number of bytes per sector) are fixed for a floppy disk (after formatting). In fact, in many systems these parameters are never changed during the life of the system.
- 4) The requirement to support a software interface that is independent of the type of disk attached to the system. In this case, a system generated ("logical") disk address (drive, head, cylinder, and sector numbers) must be mapped into a physical floppy disk address. For example, to switch between single- and dual-sided disks, it may be easier and more cost-effective for the software to treat the dual-sided disk as containing twice as many sectors per track (52) rather than as having two sides. With this technique, accesses to sectors 1 through 26 are mapped onto head 0 while accesses to sectors 27 through 52 are mapped onto head 1.
- 5) The necessity of supporting a bad track map. Since bad tracks depend on the disk media, the bad track mapping varies from disk to disk. In general, the system and application software should not be concerned with calculating bad track parameters. Instead, these software modules should refer to cylinders logically (0 through 76). The logical interface level procedures must map these cylinders into physical cylinder positions in order to avoid the bad tracks.

The key to logical interface software design is the mapping of the "logical disk interface" (as seen by the application software) into the "physical disk interface" (as implemented by the floppy disk drivers). This logical to physical mapping is tightly coupled to system software design and the mapping serves to isolate both applications and system software from the peculiarities of the FDC device. Typical logical interface procedures are described in Table 1.

The File System Interface Level

The file system typically comprises the highest level of disk interface software used by application programs. The file system is designed to treat the disk as a collection of named data areas (known as files). These files are cataloged in the disk directory. File system interface software permits the creation of new files and the deletion of existing files under software control. When a file is created, its name and disk address are entered into the directory; when a file is deleted, its name is removed from the directory. Application software requests the use of a file by executing an OPEN function. Once opened, a file is normally reserved for use by the requesting program or task and the file cannot be reopened by other tasks. When a task no longer needs to use an open file, the task closes the file, releasing it for use by other tasks.

Most file systems also support a set of file attributes that can be specified for each file. File attributes may be used to protect files (e.g., the WRITE PROTECT attri-

bute ensures that an existing file cannot accidentally be overwritten) and to supply system configuration information (e.g., a `FORMAT` attribute may specify that a file should automatically be created on a new disk when the disk is formatted).

At the file system interface level, application programs need not be explicitly aware of disk storage allocation techniques, block sizes, or file coding strategies. Only a "file name" must be presented in order to open, read or write, and subsequently close a file. Typical file system functions are listed in Table 2.

Table 1. Examples of Logical Interface Procedures

Name	Description
FORMAT DISK	Controls physical disk formatting for all tracks on a disk. Formatting adds FDC recognized cylinder, head, and sector addresses as well as address marks and data synchronization fields (gaps) to the floppy disk media.
RECALIBRATE	Moves the disk read/write head to track 0 (at the outside edge of the disk).
SEEK	Moves the disk read/write head to a specified logical cylinder. The logical and physical cylinder numbers may be different if bad track mapping is used.
READ STATUS	Indicates the status of the floppy disk drive and media. One important use of this procedure is to determine whether a floppy disk is dual-sided.
READ SECTOR	Reads one or more complete sectors starting at a specified disk address (drive, head, cylinder, and sector).
WRITE SECTOR	Writes one or more complete sectors starting at a specified disk address (drive, head, cylinder, and sector).

Table 2. Disk File System Functions

Name	Description
OPEN	Prepare a file for processing. If the file is to be opened for input and the file name is not found in the directory, an error is generated. If the file is opened for output and the file name is not found in the directory, the file is automatically created.
CLOSE	Terminate processing of an open file.
READ	Transfer Data from an open file to memory. The READ function is often designed to buffer one or more sectors of data from the disk drive and supply this data to the requesting program, as required.
WRITE	Transfer data from memory to an open file. The WRITE function is often designed to buffer data from the application program until enough data is available to fill a disk sector.
CREATE	Initialize a file and enter its name and attributes into the file directory.
DELETE	Remove a file from the directory and release its storage space.
RENAME	Change the name of a file in the directory.
ATTRIBUTE	Change the attributes of a file.
LOAD	Read a file of executable code into memory.
INITDISK	Initialize a disk by formatting the media and establishing the directory file, the bit map file, and other system files.

Scope of this Note

This application note directly addresses the logical and physical interface levels. A complete 8272 driver (including interrupt service software) is listed in Appendix A. In addition, examples of recalibrate, seek, format, read, and write logical interface level procedures are included as part of the exerciser program found in Appendix B. Wherever possible, specific hardware configuration dependencies are parameterized to provide maximum flexibility without requiring major software changes.

2.0 Disk I/O Techniques

One of the most important software aspects of disk interfacing is the fixed sector size. (Sector sizes are fixed when the disk is formatted.) Individual bytes of disk storage cannot be read/written; instead, complete sectors must be transferred between the floppy disk and system memory.

Selection of the appropriate sector size involves a trade-off between memory size, disk storage efficiency, and disk transfer efficiency. Basically, the following factors must be weighed:

- 1) Memory size. The larger the sector size, the larger the memory area that must be reserved for use during disk I/O transfers. For example, a 1K byte disk sector size requires that at least one 1K memory block be reserved for disk I/O.
- 2) Disk Storage efficiency. Both very large and very small sectors can waste disk storage space as follows. In disk file systems, space must be allocated somewhere on the disk to link the sectors of each file together. If most files are composed of many small sectors, a large amount of linkage overhead information is required. At the other extreme, when most files are smaller than a single disk sector, a large amount of space is wasted at the end of each sector.
- 3) Disk transfer efficiency. A file composed of a few large sectors can be transferred to/from memory more efficiently (faster and with less overhead) than a file composed of many small sectors.

Balancing these considerations requires knowledge of the intended system applications. Typically, for general purpose systems, sector sizes from 128 bytes to 1K bytes are used. For compatibility between single-density and double-density recording with the 8272 floppy disk controller, 256 byte sectors or 512 byte sectors are most useful.

FDC Data Transfer Interface

Three distinct software interface techniques may be used to interface system memory to the FDC device during sector data transfers:

- 1) DMA—In a DMA implementation, the software is only required to set up the DMA controller memory address and transfer count, and to initiate the data transfer. The DMA controller hardware handshakes with the processor/system bus in order to perform each data transfer.
- 2) Interrupt Driven—The FDC generates an interrupt when a data byte is ready to be transferred to memory, or when a data byte is needed from memory. It is the software's responsibility to perform appropriate memory reads/writes in order to transfer data from/to the FDC upon receipt of the interrupt.
- 3) Polling—Software responsibilities in the polling mode are identical to the responsibilities in the interrupt driven mode. The polling mode, however, is used when interrupt service overhead (context switching) is too large to support the disk data rate. In this mode, the software determines when to transfer data by continually polling a data request status flag in the FDC status register.

The DMA mode has the advantage of permitting the processor to continue executing instructions while a disk transfer is in progress. (This capability is especially useful in multiprogramming environments when the operating system is designed to permit other tasks to execute while a program is waiting for I/O.) Modes 2 and 3 are often combined and described as non-DMA operating modes. Non-DMA modes have the advantage of significantly lower system cost, but are often performance limited for double-density systems (where data bytes must be transferred to/from the FDC every 16 microseconds).

Overlapped Operations

Some FDC devices support simultaneous disk operations on more than one disk drive. Normally seek and recalibrate operations can be overlapped in this manner. Since seek operations on most floppy drives are extremely slow, this mode of operation can often be used by the system software to reduce overall disk access times.

Buffers

The buffer concept is an extremely important element in advanced disk I/O strategies. A buffer is nothing more than a memory area containing the same amount of data as a disk sector contains. Generally, when an application program requests data from a disk, the system software allocates a buffer (memory area) and transfers the data from the appropriate disk sector into the buffer. The address of the buffer is then returned to the application software. In the same manner, after the application program has filled a buffer for output, the

buffer address is passed to the system software, which writes data from the buffer into a disk sector. In multi-tasking systems, multiple buffers may be allocated from a buffer pool. In these systems, the disk controller is often requested to read ahead and fill additional data buffers while the application software is processing a previous buffer. Using this technique, system software attempts to fill buffers before they are needed by the application programs, thereby eliminating program waits during I/O transfers. Figure 1 illustrates the use of multiple buffers in a ring configuration.

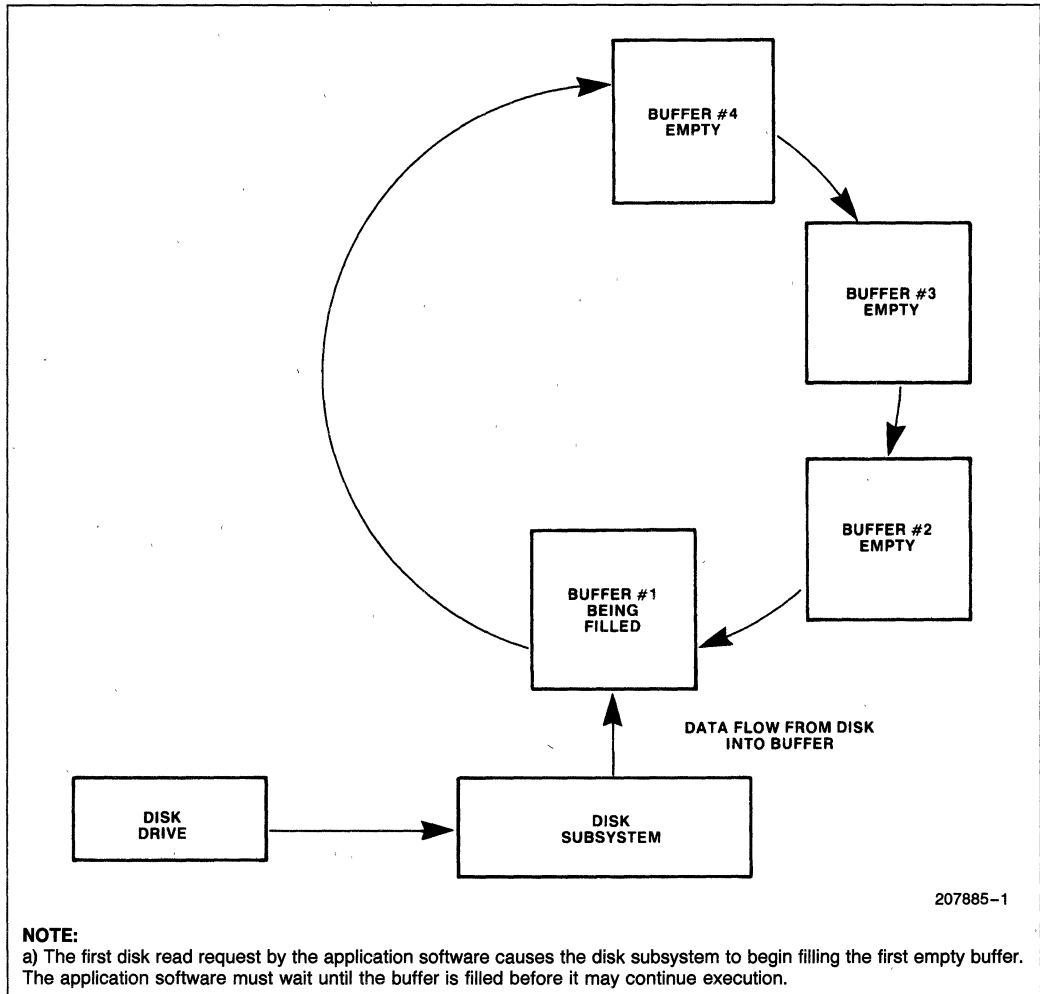


Figure 1. Using Multiple Memory Buffers for Disk I/O

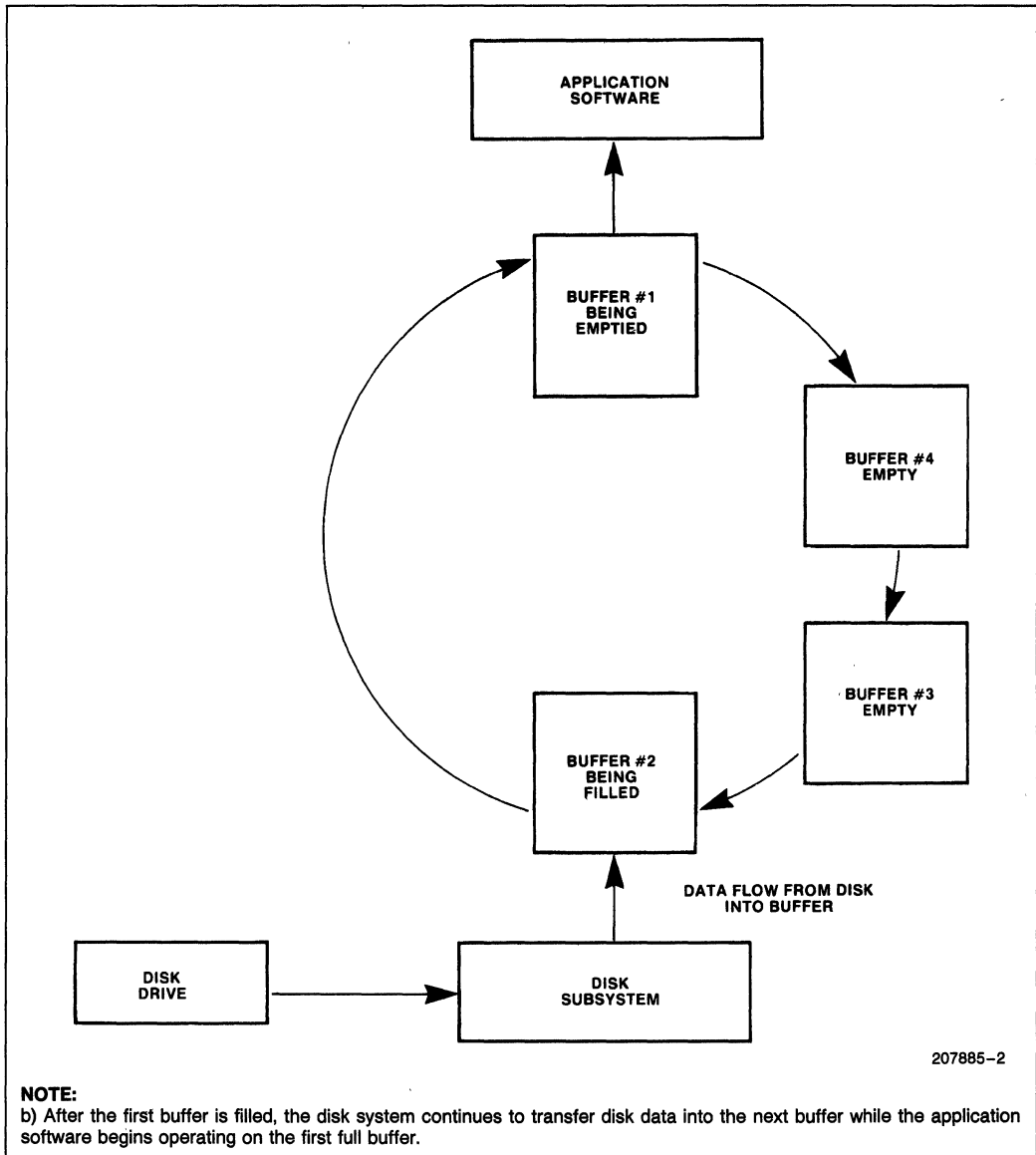
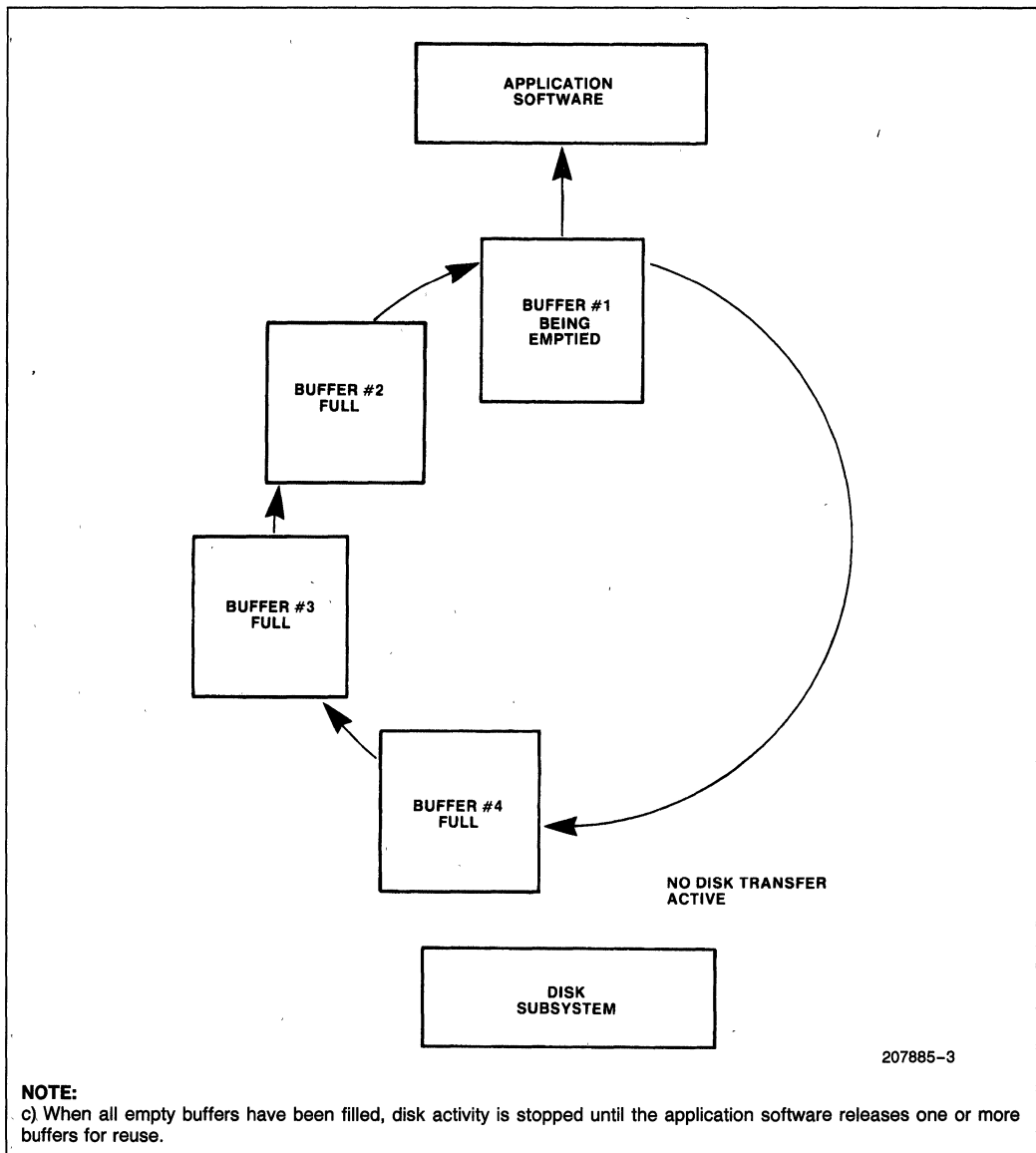


Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)



207885-3

Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)

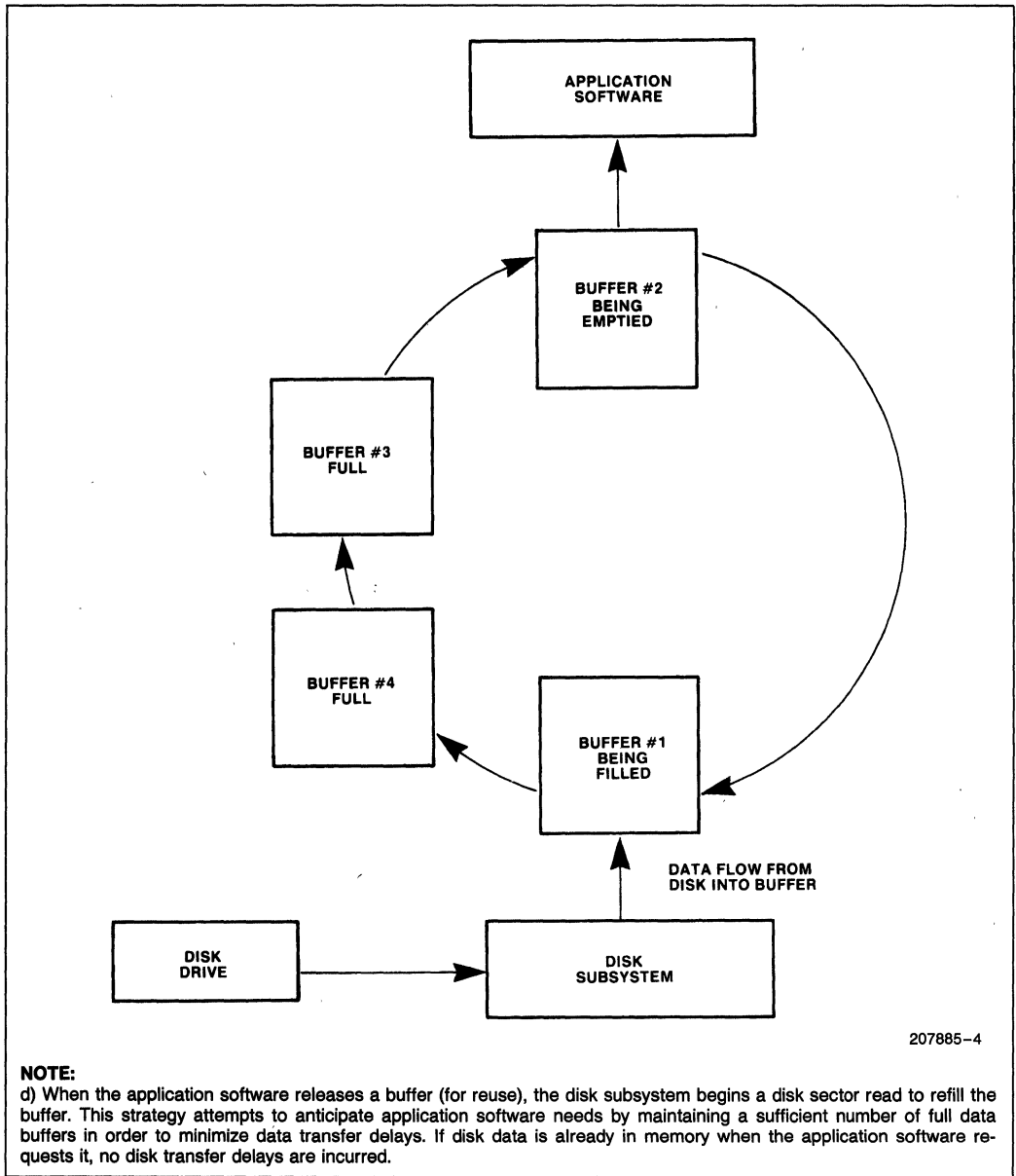


Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)

3.0 THE 8272 FLOPPY DISK CONTROLLER

The 8272 is a single-chip LSI Floppy Disk Controller (FDC) that implements both single- and double-density floppy disk storage subsystems (with up to four dual-sided disk drives per FDC). The 8272 supports the IBM 3740 single-density recording format (FM) and the IBM System 34 double-density recording format (MFM). The 8272 accepts and executes high-level disk commands such as format track, seek, read sector, and write sector. All data synchronization and error checking is automatically performed by the FDC to ensure reliable data storage and subsequent retrieval. The 8272 interfaces to microprocessor systems with or without Direct Memory Access (DMA) capabilities and also interfaces to a large number of commercially available floppy disk drives.

Floppy Disk Commands

The 8272 executes fifteen high-level disk interface commands:

Specify	Write Data
Sense Drive Status	Write Deleted Data
Sense Interrupt Status	Read Track
Seek	Read ID
Recalibrate	Scan Equal
Format Track	Scan High or Equal
Read Data	Scan Low or Equal
Read Deleted Data	

Each command is initiated by a multi-byte transfer from the driver software to the FDC (the transferred bytes contain command and parameter information). After complete command specification, the FDC automatically executes the command. The command result data (after execution of the command) may require a multi-byte transfer of status information back to the driver. It is convenient to consider each FDC command as consisting of the following three phases:

Command Phase: The driver transfers to the FDC all the information required to perform a particular disk operation. The 8272 automatically enters the command phase after RESET and following the completion of the result phase (if any) of a previous command.

Execution Phase: The FDC performs the operation as instructed. The execution phase is entered immediately after the last command parameter is written to the

FDC in the preceding command phase. The execution phase normally ends when the last data byte is transferred to/from the disk or when an error occurs.

Result Phase: After completion of the disk operation, status and other housekeeping information are made available to the driver software. After this information is read, the FDC reenters the command phase and is ready to accept another command.

Interface Registers

To support information transfer between the FDC and the system software, the 8272 contains two 8-bit registers: the Main Status Register and the Data Register. The Main Status Register (read only) contains FDC status information and may be accessed at any time. The Main Status Register (Table 3) provides the system processor with the status of each disk drive, the status of the FDC, and the status of the processor interface. The Data Register (read/write) stores data, commands, parameters, and disk drive status information. The Data Register is used to program the FDC during the command phase and to obtain result information after completion of FDC operations.

In addition to the Main Status Register, the FDC contains four additional status registers (ST0, ST1, ST2, and ST3). These registers are only available during the result phase of a command.

Command/Result Phases

Table 4 lists the 8272 command set. For each of the fifteen commands, command and result phase data transfers are listed. A list of abbreviations used in the table is given in Table 5, and the contents of the result status registers (ST0-ST3) are illustrated in Table 6.

The bytes of data which are sent to the 8272 by the drivers during the command phase, and are read out of the 8272 in the result phase, must occur in the order shown in Table 4. That is, the command code must be sent first and the other bytes sent in the prescribed sequence. All bytes of the command and result phases must be read/written as described. After the last byte of data in the command phase is sent to the 8272 the execution phase automatically starts. In a similar fashion, when the last byte of data is read from the 8272 in the result phase, the result phase is automatically ended and the 8272 reenters the command phase.

It is important to note that during the result phase all bytes shown in Table 4 must be read. The Read Data

command, for example, has seven bytes of data in the result phase. All seven bytes must be read in order to successfully complete the Read Data command. The 8272 will not accept a new command until all seven bytes have been read. The number of command and result bytes varies from command-to-command.

In order to read data from, or write data to, the Data Register during the command and result phases, the software driver must examine the Main Status Register to determine if the Data Register is available. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Regis-

ter must be low and high, respectively, before each byte of the command word may be written into the 8272. Many of the commands require multiple bytes, and as a result, the Main Status Register must be read prior to each byte transfer to the 8272. To read status bytes during the result phase, DIO and RQM in the Main Status Register must both be high. Note, checking the Main Status Register in this manner before each byte transfer to/from the 8272 is required only in the command and result phases, and is NOT required during the execution phase.

Table 3. Main Status Register Bit Definitions

Bit Number	Symbol	Description
0	D ₀ B	Disk Drive 0 Busy. Disk Drive 0 is seeking.
1	D ₁ B	Disk Drive 1 Busy. Disk Drive 1 is seeking.
2	D ₂ B	Disk Drive 2 Busy. Disk Drive 2 is seeking.
3	D ₃ B	Disk Drive 3 Busy. Disk Drive 3 is seeking.
4	CB	FDC Busy. A read or write command is in progress.
5	NDM	Non-DMA Mode. The FDC is in the non-DMA mode when this flag is set (1). This flag is set only during the execution phase of commands in the non-DMA mode. Transition of this flag to a zero (0) indicates that the execution phase has ended.
6	DIO	Data Input/Output. Indicates the direction of a data transfer between the FDC and the Data Register. When DIO is set (1), data is read from the Data Register by the processor; when DIO is reset (0), data is written from the processor to the Data Register.
7	RQM	Request for Master. When set (1), this flag indicates that the Data Register is ready to send data to, or receive data from, the processor.

Table 4. 8272 Command Set

PHASE	R/W	DATA BUS							REMARKS	PHASE	R/W	DATA BUS							REMARKS
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁				D ₀	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	
READ DATA																			
Command	W	MT	MFM	SK	0	0	1	1	0	0	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution									
	W						C												
	W						H												
	W						R												
	W						N												
	W						EOT												
Execution	W						GPL												
	W						DTL												
	W							Data transfer between the FDD and the main-system											
Result	R						ST 0	Status information after Command execution											
	R						ST 1												
	R						ST 2												
	R						C												
	R						H	Sector ID information after command execution											
	R						R												
	R						N												
READ DELETED DATA																			
Command	W	MT	MFM	SK	0	1	1	0	0	0	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution									
	W						C												
	W						H												
	W						R												
	W						N												
	W						EC 1												
Execution	W						GPL												
	W						DTL												
	W							Data transfer between the FDD and the main-system											
Result	R						ST 0	Status information after Command execution											
	R						ST 1												
	R						ST 2												
	R						C												
	R						H	Sector ID information after Command execution											
	R						R												
	R						N												
READ ID																			
Command	W	0	MFM	0	0	1	0	1	0	0	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	The first correct ID information on the track is stored in Data Register									
Execution	W																		
	W																		
	W																		
	W																		
	W																		
	W																		
	W																		
Result	R						ST 0	Status information after Command execution											
	R						ST 1												
	R						ST 2												
	R						C												
	R						H	Sector ID information during Execution Phase											
	R						R												
	R						N												
FORMAT A TRACK																			
Command	W	0	MFM	0	0	1	0	1	0	1	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	Bytes/Sector Sectors/Track Gap 3 Filter Byte									
Execution	W																		
	W																		
	W																		
	W																		
	W																		
	W																		
	W																		
Result	R						ST 0	Status information after Command execution											
	R						ST 1												
	R						ST 2												
	R						C												
	R						H	In this case, the ID information has no meaning											
	R						R												
	R						N												
SCAN EQUAL																			
Command	W	MT	MFM	SK	1	0	0	0	1	0	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution									
Execution	W						C												
	W						H												
	W						R												
	W						N												
	W						EOT												
	W						GPL												
	W						STP												
Result	R						ST 0	Status information after Command execution											
	R						ST 1												
	R						ST 2												
	R						C												
	R						H	Sector ID information after Command execution											
	R						R												
	R						N												
WRITE DELETED DATA																			
Command	W	MT	MFM	0	0	1	0	0	1	0	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution									
	W						C												
	W						H												
	W						R												
	W						N												
	W						EOT												
Execution	W						GPL												
	W						DTL												
	W							Data transfer between the FDD and the main-system											
Result	R						ST 0	Status information after Command execution											
	R						ST 1												
	R						ST 2												
	R						C												
	R						H	Sector ID information after Command execution											
	R						R												
	R						N												

207885-34

NOTE:

1. A₀ = 1 for all operations.

Table 4. 8272 Command Set (Continued)

PHASE	R/W	DATA BUS								REMARKS
		D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
SCAN LOW OR EQUAL										
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes
W	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior Command execution
W	W						C			
W	W						H			
W	W						R			
W	W						N			
W	W						EOT			
W	W						GPL			
W	W						STP			
Execution										Data compared between the FDD and the main-system
Result	R						ST 0			Status information after Command execution
R	R						ST 1			
R	R						ST 2			
R	R						C			Sector ID information after Command execution
R	R						H			
R	R						R			
R	R						N			
SCAN HIGH OR EQUAL										
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes
W	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior Command execution
W	W						C			
W	W						H			
W	W						R			
W	W						N			
W	W						EOT			
W	W						GPL			
W	W						STP			
Execution										Data compared between the FDD and the main-system
Result	R						ST 0			Status information after Command execution
R	R						ST 1			
R	R						ST 2			
R	R						C			Sector ID information after Command execution
R	R						H			
R	R						R			
R	R						N			
RECALIBRATE										
Command	W	0	0	0	0	0	1	1	1	Command Codes
Execution	W	0	0	0	0	0	0	0	DS1 DS0	Head retracted to Track 0
SENSE INTERRUPT STATUS										
Command	W	0	0	0	0	1	0	0	0	Command Codes
Result	R					ST 0				Status information at the end of each seek operation about the FDC
R	R					C				
SPECIFY										
Command	W	0	0	0	0	0	0	1	1	Command Codes
W	W		SPT				HUT			Timer Settings
W	W		HLT				ND			
SENSE DRIVE STATUS										
Command	W	0	0	0	0	1	0	0		Command Codes
Result	W	0	0	0	0	0	HDS	DS1	DS0	Status information about the FDD
R	R					ST 3				
SEEK										
Command	W	0	0	0	0	1	1	1	1	Command Codes
W	W	0	0	0	0	0	HDS	DS1	DS0	
W	W					C				
Execution										Head is positioned over proper Cylinder on Diskette
INVALID										
Command	W	Invalid Codes								Invalid Command Codes (NoOp - FDC goes into Standby State)
Result	R					ST 0				ST 0 = 80 (16)

207885-35

NOTE:

1. A₀ = 1 for all operations.

Table 5. Command/Result Parameter Abbreviations

Symbol	Description
C	CYLINDER ADDRESS. The currently selected cylinder address (0 to 76) on the disk.
D	DATA PATTERN. The pattern to be written in each sector data field during formatting.
DS0, DS1	DISK DRIVE SELECT. DS1 DS0 0 0 Drive 0 0 1 Drive 1 1 0 Drive 2 1 1 Drive 3
DTL	SPECIAL SECTOR SIZE. During the execution of disk read/write commands, this parameter is used to temporarily alter the effective disk sector size. By setting N to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the disk) is larger than DTL specifies, the remainder of the actual sector is not passed to the system during read commands; during write commands, the remainder of the actual sector is written with all-zeroes bytes. DTL should be set to FF hexadecimal when N is not zero.
EOT	END OF TRACK. The final sector number of the current track.
GPL	GAP LENGTH. The gap 3 size. (Gap 3 is the space between sectors).
H	HEAD ADDRESS. Selected head: 0 or 1 (disk side 0 or 1, respectively) as encoded in the sector ID field.
HLT	HEAD LOAD TIME. Defines the time interval that the FDC waits after loading the head before initiating the read or write operation. Programmable from 2 to 254 milliseconds (in increments of 2 ms).
HUT	HEAD UNLOAD TIME. Defines the time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Programmable from 16 to 240 milliseconds (in increments of 16 ms).
MFM	MFM/FM MODE SELECTOR. Selects MFM double-density recording mode when high, FM single-density mode when low.
MT	MULTI-TRACK SELECTOR. When set, this flag selects the multi-track operating mode. In this mode (used only with dual-sided disks), the FDC treats a complete cylinder (under both read/write head 0 and read/write head 1) as a single track. The FDC operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set (high), a multi-sector read operation will automatically continue to the first sector under head 1 when the FDC finishes operating on the last sector under head 0.
N	SECTOR SIZE CODE. The number of data bytes within a sector.
ND	NON-DMA MODE FLAG. When set (1), this flag indicates that the FDC is to operate in the non-DMA mode. In this mode, the processor participates in each data transfer (by means of an interrupt or by polling the RQM flag in the Main Status Register). When reset (0), the FDC interfaces to a DMA controller.
R	SECTOR ADDRESS. Specifies the sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of the first sector to be read or written.
SC	NUMBER OF SECTORS PER TRACK. Specifies the number of sectors per track to be initialized by the Format Track command.
SK	SKIP FLAG. When this flag is set, sectors containing deleted data address marks will automatically be skipped during the execution of multi-sector Read Data or Scan commands. In the same manner, a sector containing a data address mark will automatically be skipped during the execution of a multi-sector Read Deleted Data command.
SRT	STEP RATE INTERVAL. Defines the time interval between step pulses issued by the FDC (track-to-track access time). Programmable from 1 to 16 milliseconds (in increments of 1 ms).
ST0, ST1, ST2, ST3	STATUS REGISTER 0-3. Registers within the FDC that store status information after a command has been executed. This status information is available to the processor during the Result Phase after command execution. These registers may only be read after a command has been executed (in the exact order shown in Table 4 for each command). These registers should not be confused with the Main Status Register.
STP	SCAN SECTOR INCREMENT. During Scan operations, this parameter is added to the current sector number in order to determine the next sector to be scanned.

Table 6. Status Register Definitions

Bit Number	Symbol	Description
STATUS REGISTER 0		
7, 6	IC	INTERRUPT CODE. 00— Normal termination of command. The specified command was properly executed and completed without error. 01— Abnormal termination of command. Command execution was started but could not be successfully completed. 10— Invalid command. The requested command could not be executed. 11— Abnormal termination. During command execution, the disk drive ready signal changed state.
5	SE	SEEK END. This flag is set (1) when the FDC has completed the Seek command and the read/write head is positioned over the correct cylinder.
4	EC	EQUIPMENT CHECK ERROR. This flag is set (1) if a fault signal is received from the disk drive or if the track 0 signal is not received from the disk drive after 77 step pulses (Recalibrate command).
3	NR	NOT READY ERROR. This flag is set if a read or write command is issued and either the drive is not ready or the command specifies side 1 (head 1) of a single-sided disk.
2	H	HEAD ADDRESS. The head address at the time of the interrupt.
1, 0	DS1, DS0	DRIVE SELECT. The number of the drive selected at the time of the interrupt.
STATUS REGISTER 1		
7	EN	END OF TRACK ERROR. This flag is set if the FDC attempts to access a sector beyond the final sector of the track.
6		UNDEFINED
5	DE	DATA ERROR. Set when the FDC detects a CRC error in either the ID field or the data field of a sector.
4	OR	OVERRUN ERROR. Set (during data transfers) if the FDC does not receive DMA or processor service within the specified time interval.
3		UNDEFINED
2	ND	SECTOR NOT FOUND ERROR. This flag is set by any of the following conditions. a) The FDC cannot locate the sector specified in the Read Data, Read Deleted Data, or Scan command. b) The FDC cannot locate the starting sector specified in the Read Track command. c) The FDC cannot read the ID field without error during a Read ID command.
1	NW	WRITE PROTECT ERROR. This flag is set if the FDC detects a write protect signal from the disk drive during the execution of a Write Data, Write Deleted Data, or Format Track command.
0	MA	MISSING ADDRESS MARK ERROR. This flag is set by either of the following conditions: a) The FDC cannot detect the ID address mark on the specified track (after two rotations of the disk). b) The FDC cannot detect the data address mark or deleted data address mark on the specified track. (See also the MD bit of Status Register 2.)

Table 6. Status Register Definitions (Continued)

Bit Number	Symbol	Description
STATUS REGISTER 2		
7		UNDEFINED
6	CM	CONTROL MARK. This flag is set when the FDC encounters one of the following conditions: a) A deleted data address mark during the execution of a Read Data or Scan command. b) A data address mark during the execution of a Read Deleted Data command.
5	DD	DATA ERROR. Set (1) when the FDC detects a CRC error in a sector data field. This flag is not set when a CRC error is detected in the ID field.
4	WC	CYLINDER ADDRESS ERROR. Set when the cylinder address from the disk sector ID field is different from the current cylinder address maintained within the FDC.
3	SH	SCAN HIT. Set during the execution of the Scan command if the scan condition is satisfied.
2	SN	SCAN NOT SATISFIED. Set during execution of the Scan command if the FDC cannot locate a sector on the specified cylinder that satisfies the scan condition.
1	BC	BAD TRACK ERROR. Set when the cylinder address from the disk sector ID field is FF hexadecimal and this cylinder address is different from the current cylinder address maintained within the FDC. This all "ones" cylinder number indicates a bad track (one containing hard errors) according to the IBM soft-sectored format specifications.
0	MD	MISSING DATA ADDRESS MARK ERROR. Set if the FDC cannot detect a data address mark or deleted data address mark on the specified track.
STATUS REGISTER 3		
7	FT	FAULT. This flag indicates the status of the fault signal from the selected disk drive.
6	WP	WRITE PROTECTED. This flag indicates the status of the write protect signal from the selected disk drive.
5	RDY	READY. This flag indicates the status of the ready signal from the selected disk drive.
4	T0	TRACK 0. This flag indicates the status of the track 0 signal from the selected disk drive.
3	TS	TWO-SIDED. This flag indicates the status of the two-sided signal from the selected disk drive.
2	H	HEAD ADDRESS. This flag indicates the status of the side select signal for the currently selected disk drive.
1, 0	DS1, DS0	DRIVE SELECT. Indicates the currently selected disk drive number.

Execution Phase

All data transfers to (or from) the floppy drive occur during the execution phase. The 8272 has two primary modes of operation for data transfers (selected by the specify command):

- 1) DMA mode
- 2) non-DMA mode

In the DMA mode, execution phase data transfers are handled by the DMA controller hardware (invisible to the driver software). The driver software, however, must set all appropriate DMA controller registers prior to the beginning of the disk operation. An interrupt is generated by the 8272 after the last data transfer, indicating the completion of the execution phase, and the beginning of the result phase.

In the non-DMA mode, transfer requests are indicated by generation of an interrupt and by activation of the RQM flag (bit 7 in the Main Status Register). The interrupt signal can be used for interrupt-driven systems and RQM can be used for polled systems. The driver software must respond to the transfer request by reading data from, or writing data to, the FDC. After completing the last transfer, the 8272 generates an interrupt to indicate the beginning of the result phase. In the non-DMA mode, the processor must activate the "terminal count" (TC) signal to the FDC (normally by means of an I/O port) after the transfer request for the last data byte has been received (by the driver) and before the appropriate data byte has been read from (or written to) the FDC.

In either mode of operation (DMA or non-DMA), the execution phase ends when a "terminal count" signal is sensed by the FDC, when the last sector on a track (the EOT parameter—Table 4) has been read or written, or when an error occurs.

Multi-Sector and Multi-Track Transfers

During disk read/write transfers (Read Data, Write Data, Read Deleted Data, and Write Deleted Data), the FDC will continue to transfer data from sequential sectors until the TC input is sensed. In the DMA mode,

the TC input is normally set by the DMA controller. In the non-DMA mode, the processor directly controls the FDC TC input as previously described. Once the TC input is received, the FDC stops requesting data transfers (from the system software or DMA controller). The FDC, however, continues to read data from, or write data to, the floppy disk until the end of the current disk sector. During a disk read operation, the data read from the disk (after reception of the TC input) is discarded, but the data CRC is checked for errors; during a disk write operation, the remainder of the sector is filled with all-zero bytes.

If the TC signal is not received before the last byte of the current sector has been transferred to/from the system, the FDC increments the sector number by one and initiates a read or write command for this new disk sector.

The FDC is also designed to operate in a multi-track mode for dual-sided disks. In the multi-track mode (specified by means of the MT flag in the command byte—Table 4) the FDC will automatically increment the head address (from 0 to 1) when the last sector (on the track under head 0) has been read or written. Reading or writing is then continued on the first sector (sector 1) of head 1.

Drive Status Polling

After the power-on reset, the 8272 automatically enters a drive status polling mode. If a change in drive status is detected (all drives are assumed to be "not ready" at power-on), an interrupt is generated. The 8272 continues this status polling between command executions (and between step pulses in the Seek command). In this manner, the 8272 automatically notifies the system software whenever a floppy disk is inserted, removed, or changed by the operator.

Command Details

During the command phase, the Main Status Register must be polled by the driver software before each byte is written into the Data Register. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be

low and high, respectively, before each byte of the command may be written into the 8272. The beginning of the execution phase for any of these commands will cause DIO to be set high and RQM to be set low.

Operation of the FDC commands is described in detail in Application Note AP-116, "An Intelligent Data Base System Using the 8272".

Invalid Commands

If an invalid (undefined) command is sent to the FDC, the FDC will terminate the command. No interrupt is generated by the 8272 during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both set indicating to the processor that the 8272 is in the result phase and the contents of Status Register 0 must be read. When the processor reads Status Register 0 it will find an 80H code indicating that an invalid command was received. The driver software in Appendix B checks each requested command and will not issue an invalid command to the 8272.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate interrupt; otherwise the FDC will consider the next command to be an invalid command. Also, when the last "hidden" interrupt has been serviced, further Sense Interrupt Status commands will result in invalid command codes.

4.0 8272 PHYSICAL INTERFACE SOFTWARE

PL/M software driver listings for the 8272 FDC are contained in Appendix A. These drivers have been designed to operate in a DMA environment (as described in Application Note AP-116, "An Intelligent Data Base System Using the 8272"). In the following paragraphs, each driver procedure is described. (A description of the driver data base variables is given in Table 7.) In addition, the modifications necessary to reconfig-

ure the drivers for operation in a polled environment are discussed.

INITIALIZE\$DRIVERS

This initialization procedure must be called before any FDC operations are attempted. This module initializes the DRIVE\$READY, DRIVE\$STATUS\$CHANGE, OPERATION\$IN\$PROGRESS, and OPERATION\$COMPLETE arrays as well as the GLOBAL\$DRIVE\$NO variable.

EXECUTE\$DOCB

This procedure contains the main 8272 driver control software and handles the execution of a complete FDC command. EXECUTE\$DOCB is called with two parameters: a) a pointer to a disk operation control block and b) a pointer to a result status byte. The format of the disk operation control block is illustrated in Figure 2 and the result status codes are described in Table 8.

Before starting the command phase for the specified disk operation, the command is checked for validity and to determine whether the FDC is busy. (For an overlapped operation, if the FDC BUSY flag is set—in the Main Status Register—the command cannot be started; non-overlapped operations cannot be started if the FDC BUSY flag is set, if any drive is in the process of seeking/recalibrating, or if an operation is currently in progress on the specified drive.)

After these checks are made, interrupts are disabled in order to set the OPERATION\$IN\$PROGRESS flag, reset the OPERATION\$COMPLETE flag, load a pointer to the current operation control block into the OPERATION\$DOCB\$PTR array and set GLOBAL\$DRIVE\$NO (if a non-overlapped operation is to be started).

At this point, parameters from the operation control block are output to the DMA controller and the FDC

command phase is initiated. After completion of the command phase, a test is made to determine the type of result phase required for the current operation. If no result phase is needed, control is immediately returned to the calling program. If an immediate result phase is required, the result bytes are input from the FDC. Otherwise, the CPU waits until the OPERA-

TION\$COMPLETE flag is set (by the interrupt service procedure).

Finally, if an error is detected in the result status code (from the FDC), an FDC operation error is reported to the calling program.

Table 7. Driver Data Base

Name	Description
DRIVE\$READY	A public array containing the current "ready" status of each drive.
DRIVE\$STATUS\$CHANGE	A public array containing a flag for each drive. The appropriate flag is set whenever the ready status of a drive changes.
OPERATION\$DOCB\$PTR	An internal array of pointers to the operation control block currently in progress for each drive.
OPERATION\$IN\$PROGRESS	An internal array used by the driver procedures to determine if a disk operation is in progress on a given drive.
OPERATION\$COMPLETE	An internal array used by the driver procedures to determine when the execution phase of a disk operation is complete.
GLOBAL\$DRIVE\$NO	A data byte that records the current drive number for non-overlapped disk operations.
VALID\$COMMAND	A constant flag array that indicates whether a specified FDC command code is valid.
COMMAND\$LENGTH	A constant byte array specifying the number of command/parameter bytes to be transferred to the FDC during the command phase.
DRIVE\$NO\$PRESENT	A constant flag array that indicates whether a drive number is encoded into an FDC command.
OVERLAP\$OPERATION	A constant flag array that indicates whether an FDC command can be overlapped with other commands.
NO\$RESULT	A constant flag array that is used to determine when an FDC operation does not have a result phase.
IMMED\$RESULT	A constant flag array that indicates that an FDC operation has a result phase beginning immediately after the command phase is complete.
POSSIBLE\$ERROR	A constant flag array that indicates if an FDC operation should be checked for an error status indication during the result phase.

Address Offset	Disk Operation Control Block (DOCB)
0	DMA\$OP
1	DMA\$ADDR
3	DMA\$ADDR\$EXT
4	DMA\$COUNT
6	DISK\$COMMAND (0)
7	DISK\$COMMAND (1)
8	DISK\$COMMAND (2)
9	DISK\$COMMAND (3)
10	DISK\$COMMAND (4)
11	DISK\$COMMAND (5)
12	DISK\$COMMAND (6)
13	DISK\$COMMAND (7)
14	DISK\$COMMAND (8)
15	DISK\$RESULT (0)
16	DISK\$RESULT (1)
17	DISK\$RESULT (2)
18	DISK\$RESULT (3)
19	DISK\$RESULT (4)
20	DISK\$RESULT (5)
21	DISK\$RESULT (6)
22	MISC

Figure 2. Disk Operation Control Block (DOCB) Format

Table 8. EXECUTE\$DOCB Return Status Codes

Code	Description
0	NO ERRORS. The specified operation was completed without error.
1	FDC BUSY. The requested operation cannot be started. This error occurs if an attempt is made to start an operation before the previous operation is completed.
2	FDC ERROR. An error was detected by the FDC during the execution phase of a disk operation. Additional error information is contained in the result data portion of the disk operation control block (DOCB.DISK\$RESULT) as described in the 8272 data sheet. This error occurs whenever the 8272 reports an execution phase error (e.g., missing address mark).
3	8272 COMMAND INTERFACE ERROR. An 8272 interfacing error was detected during the command phase. This error occurs when the command phase of a disk operation cannot be successfully completed (e.g., incorrect setting of the DIO flag in the Main Status Register).
4	8272 RESULT INTERFACE ERROR. An 8272 interfacing error was detected during the result phase. This error occurs when the result phase of a disk operation cannot be successfully completed (e.g., incorrect setting of the DIO flag in the Main Status Register).
5	INVALID FDC COMMAND.

FDCINT

This procedure performs all interrupt processing for the 8272 interface drivers. Basically, two types of interrupts are generated by the 8272: (a) an interrupt that signals the end of a command execution phase and the beginning of the result phase and (b) an interrupt that signals the completion of an overlapped operation or the occurrence of an unexpected event (e.g., change in the drive "ready" status).

An interrupt of type (a) is indicated when the FDC BUSY flag is set (in the Main Status Register). When a type (a) interrupt is sensed, the result bytes are read from the 8272 and placed in the result portion of the disk operation control block, the appropriate OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.

When an interrupt of type (b) is indicated (FDC not busy), a sense interrupt status command is issued (to the FDC). The upper two bits of the result status register (Status Register Zero—ST0) are used to determine the cause of the interrupt. The following four cases are possible:

- 1) Operation Complete. An overlapped operation is complete. The drive number is found in the lower two bits of ST0. The ST0 data is transferred to the active operation control block, the OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.
- 2) Abnormal Termination. A disk operation has abnormally terminated. The drive number is found in the lower two bits of ST0. The ST0 data is transferred to the active control block, the OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.
- 3) Invalid Command. The execution of an invalid command (i.e., a sense interrupt command with no interrupt pending) has been attempted. This interrupt signals the successful completion of all interrupt processing.
- 4) Drive Status Change. A change has occurred in the "ready" status of a disk drive. The drive number is found in the lower two bits of ST0. The DRIVE\$READY flag for this disk drive is set to the new drive "ready" status and the DRIVE\$STATUS\$CHANGE flag for the drive is also set. In addition, if a command is currently in progress, the ST0 data is transferred to the active control block, the OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.

After processing a type (b) interrupt, additional sense interrupt status commands must be issued and processed until an "invalid command" result is returned from the FDC. This action guarantees that all "hidden" interrupts are serviced.

In addition to the major driver procedures described above, a number of support procedures are required. These support routines are briefly described in the following paragraphs.

OUTPUT\$CONTROLS\$TO\$DMA

This procedure outputs the DMA mode, the DMA address, and the DMA word count to the 8237 DMA controller. In addition, the upper four bits of the 20-bit DMA address are output to the address extension latch. Finally, the disk DMA channel is started.

OUTPUT\$COMMAND\$TO\$FDC

This software module outputs a complete disk command to the 8272 FDC. The number of required command/parameter bytes is found in the COMMAND\$LENGTH table. The appropriate bytes are output one at a time (by calls to OUTPUT\$BYTE\$TO\$FDC) from the command portion of the disk operation control block.

INPUT\$RESULT\$FROM\$FDC

This procedure is used to read result phase status information from the disk controller. At most, seven bytes are read. In order to read each byte, a call is made to INPUT\$BYTE\$FROM\$FDC. When the last byte has been read, a check is made to insure that the FDC is no longer busy.

OUTPUT\$BYTE\$TO\$FDC

This software is used to output a single command/parameter byte to the FDC. This procedure waits until the FDC is ready for a command byte and then outputs the byte to the FDC data port.

INPUT\$BYTE\$FROM\$FDC

This procedure inputs a single result byte from the FDC. The software waits until the FDC is ready to transfer a result byte and then reads the byte from the FDC data port.

FDC\$READY\$FOR\$COMMAND

This procedure assures that the FDC is ready to accept a command/parameter byte by performing the following three steps. First, a small time interval (more than 20 microseconds) is inserted to assure that the RQM flag has time to become valid (after the last byte transfer). Second, the master request flag (RQM) is polled until it is activated by the FDC. Finally, the DIO flag is checked to ensure that it is properly set for FDC input (from the processor).

FDC\$READY\$FOR\$RESULT

The operation of this procedure is similar to the FDC\$READY\$FOR\$COMMAND with the following exception. If the FDC BUSY flag (in the Main Status Register) is not set, the result phase is complete and no more data is available from the FDC. Otherwise, the procedure waits for the RQM flag and checks the DIO flag for FDC output (to the processor).

OPERATION\$CLEAN\$UP

This procedure is called after the execution of a disk operation that has no result phase. OPERATION\$CLEAN\$UP resets the OPERATION\$IN\$PROGRESS flag and the GLOBAL\$DRIVE\$NO variable if appropriate. This procedure is also called to clean up after some disk operation errors.

Modifications for Polling Operation

To operate in the polling mode, the following modifications should be made to the previous routines:

- 1) The OUTPUT\$CONTROLS\$TO\$DMA routine should be deleted.
- 2) In EXECUTE\$DOCB, immediately prior to WAIT\$FOR\$OP\$COMPLETE, a polling loop should be inserted into the code. The loop should test the RQM flag (in the Main Status Register). When RQM is set, a data byte should be written to, or read from, the 8272. The buffer address may be computed from the base address contained in DOCB.DMA\$ADDR and DOCB.DMA\$ADDR\$EXT. After the correct number of bytes have been transferred, an operation complete interrupt will be issued by the FDC. During data transfer in the non-DMA mode, the NON-DMA MODE flag (bit 5 of the Main Status Register) will be set. This flag will remain set for the complete execution phase. When the transfer is finished, the NON-DMA MODE flag is reset and the result phase interrupt is issued by the FDC.

5.0 8272 LOGICAL INTERFACE SOFTWARE

Appendix B of this Application Note contains a PL/M listing of an exerciser program for the 8272 drivers. This program illustrates the design of logical interface level procedures to specify disk parameters, recalibrate a drive, seek to a cylinder, format a disk, read data, and write data.

The exerciser program is written to operate a standard single-sided 8" floppy disk drive in either the single- or double-density recording mode. Only the eight parameters listed in Table 9 must be specified. All other parameters are derived from these 8 basic variables.

Each of these logical interface procedures is described in the following paragraphs (refer to the listing in Appendix B).

SPECIFY

This procedure sets the FDC signal timing so that the FDC will interface correctly to the attached disk drive. The SPECIFY procedure requires four parameters, the step rate (SRT), head load time (HLT), head unload time (HUT), and the non-DMA mode flag (ND). This procedure builds a disk operation control block (SPECIFY\$DOCB) and passes the control block to the FDC driver module (EXECUTE\$DOCB) for execution. (Note carefully the computation required to transform the step rate (SRT) into the correct 8272 parameter byte.)

RECALIBRATE

This procedure causes the floppy disk read/write head to retract to track 0. The RECALIBRATE procedure requires only one parameter—the drive number on which the recalibrate operation is to be performed. This procedure builds a disk operation control block (RECALIBRATE\$DOCB) and passes the control block to the FDC driver for execution.

SEEK

This procedure causes the disk read/write head (on the selected drive) to move to the desired cylinder position. The SEEK procedure is called with three parameters: drive number (DRV), head/side number (HD), and cylinder number (CYL). This software module builds a disk operation control block (SEEK\$DOCB) that is executed by the FDC driver.

FORMAT

The FORMAT procedure is designed to initialize a complete floppy disk so that sectors can subsequently be read and written by system and application programs. Three parameters must be supplied to this procedure: the drive number (DRV), the recording density (DENS), and the interleave factor (INTLVE). The FORMAT procedure generates a data block (FMTBLK) and a disk operation control block (FORMAT\$DOCB) for each track on the floppy disk (normally 77).

Table 9. Basic Disk Parameters

Name	Description
DENSITY	The recording mode (FM or MFM).
FILLER\$BYTE	The data byte to be written in all sectors during formatting.
TRACKS\$PER\$DISK	The number of cylinders on the floppy disk.
BYTES\$PER\$SECTOR	The number of bytes in each disk sector. The exerciser accepts 128, 256, and 512 in FM mode, and 256, 512, and 1024 in MFM mode.
INTERLEAVE	The sector interleave factor for each disk track.
STEP\$RATE	The disk drive step rate (1–16 milliseconds).
HEAD\$LOAD\$TIME	The disk drive head load time (2–254 milliseconds).
HEAD\$UNLOAD\$TIME	The head unload time (16–240 milliseconds).

The format data block specifies the four sector ID field parameters (cylinder, head, sector, and bytes per sector) for each sector on the track. The sector numbers need not be sequential; the interleave factor (INTLVE parameter) is used to compute the logical to physical sector mapping.

After both the format data block and the operation control block are generated for a given cylinder, control is passed to the 8272 drivers for execution. After the format operation is complete, a SEEK to the next cylinder is performed, a new format table is generated, and another track formatting operation is executed by the drivers. This track formatting continues until all tracks on the diskette are formatted.

In some systems, bad tracks must also be specified when a disk is formatted. For these systems, the existing FORMAT procedure should be modified to format bad tracks with a cylinder number of OFFH.

Write

The WRITE procedure transfers a complete sector of data to the disk drive. Five parameters must be supplied to this software module: the drive number (DRV), the cylinder number (CYL), the head/side number (HD), the sector number (SEC) and the recording density (DENS). This procedure generates a disk operation control block (WRITE\$DOCB) from these parameters and passes the control block to the 8272 driver for execution. When control returns to the calling program, the data has been transferred to disk.

Read

This procedure is identical to the WRITE procedure except the direction of data transfer is reversed. The READ procedure transfers a sector of data from the floppy disk to system memory.

Coping with Errors

In actual practice all logical disk interface routines would contain error processing mechanisms. (Errors have been ignored for the sake of simplicity in the exerciser programs listed in Appendix B.) A typical error recovery technique consists of a two-stage procedure. First, when an error is detected, a recalibrate operation is performed followed by a retry of the failed operation. This procedure forces the drive to seek directly to the requested cylinder (lowering the probability of a seek error) and attempts to perform the requested operation an additional time. Soft (temporary) errors caused by mechanical or electrical interference do not normally recur during the retry operation; hard errors (caused by media or drive failures), on the other hand, will continue to occur during retry operations. If, after a number of retries (approximately 10), the operation continues to fail, an error message is displayed to the system operator. This error message lists the drive number, type of operation, and failure status (from the FDC). It is the operator's responsibility to take additional action as required.

6.0 FILE SYSTEMS

The file system provides the disk I/O interface level most familiar to users of interactive microcomputer and minicomputer systems. In a file system, all data is stored in named disk areas called files. The user and applications programs need not be concerned with the exact location of a file on the disk—the disk file system automatically determines the file location from the file name. Files may be created, read, written, modified, and finally deleted (destroyed) when they are no longer needed. Each floppy disk typically contains a directory that lists all the files existing on the disk. A directory entry for a file contains information such as file name, file size, and the disk address (track and sector) of the beginning of the file.

File Allocation

File storage is actually allocated on the disk (by the file system) in fixed size areas called blocks. Normally a block is the same size as a disk sector. Files are created by finding and reserving enough unused blocks to contain the data in the file. Two file allocation methods are currently in widespread use. The first method allocates blocks (for a file) from a sequential pool of unused blocks. Thus, a file is always contained in a set of sequential blocks on the disk. Unfortunately, as files are created, updated, and deleted, these free-block pools become fragmented (separated from one another). When this fragmentation occurs, it often becomes impossible for the file system to create a file even though there is a sufficient number of free blocks on the disk. At this point, special programs must be run to "squeeze" or compact the disk, in order to re-create a single contiguous free-block pool.

The second file allocation method uses a more flexible technique in which individual data blocks may be located anywhere on the disk (with no restrictions). With this technique, a file directory entry contains the disk address of a file pointer block rather than the disk address of the first data block of the file. This file pointer block contains pointers (disk addresses) for each data block in the file. For example, the first pointer in the file pointer block contains the track and sector address of the first data block in the file; the second pointer contains the disk address of the second data block, etc.

In practice, pointer blocks are usually the same size as data blocks. Therefore, some files will require multiple pointer blocks. To accommodate this requirement without loss of flexibility, pointer blocks are linked together, that is, each pointer block contains the disk address of the following pointer block. The last pointer block of the file is signaled by an illegal disk address (e.g., track 0, sector 0 or track OFFH, sector OFFH).

The Intel File System

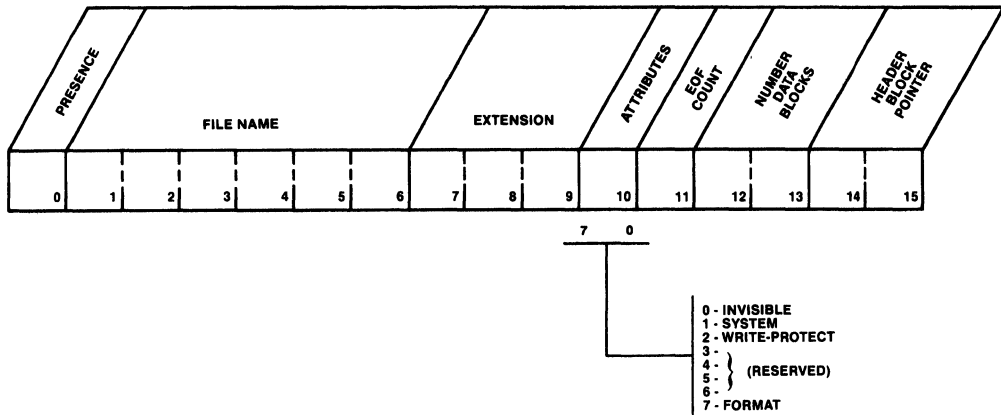
The Intel file system (described in detail in the RMX-80 Users Guide) uses the second disk file allocation method (previously discussed). In order to lower the system overhead involved in finding free data blocks, the Intel file system incorporates a free space management data structure known as a bit map. Each disk sector is represented by a single bit in the bit map. If a bit in the bit map is set to 1, the corresponding disk sector has been allocated. A zero in the bit map indicates that the corresponding sector is free. With this technique, the process of allocating or freeing a sector is accomplished by simply altering the bit map.

File names consist of a basic file name (up to six characters) and a file extension (up to three characters). The basic file name and the file extension are separated by a period (.). Examples of valid file names are: DRIV72.OBJ, XX.TMP, and FILE.CS. In addition, four file attributes are supported (see Figure 3 for attribute definitions).

The bit map and the file directory are placed on pre-specified disk tracks (reserved for system use) beginning at track zero.

Disk File System Functions

Table 2 illustrates the typical functions implemented by a disk file system. As an example, the disk directory function (DIR) lists disk file information on the console display terminal. Figure 3 details the contents of a display entry in the Intel file system. The PL/M procedure outlined in Figure 4 illustrates a disk directory algorithm that displays the file name, the file attributes, and the file size (in blocks) for each file in the directory.



207885-5

Directory Entry

Presence is a flag that can contain one of three values:

- 000H— The file associated with this entry is present on the disk.
- 07FH— No file is associated with this entry; the content of the rest of the entry is undefined. The first entry with its flag set to 07FH marks the current logical end of the directory and directory searches stop at this entry.
- 0FFH— The file named in this entry once existed on the disk but is currently deleted. The next file added to the directory will be placed in the first entry marked 0FFH. This flag cannot, therefore, be used to (reliably) find a file that has been deleted. A value of 0FFH should be thought of as simply marking an open directory entry.

File Name is a string of up to 6 non-blank ASCII characters specifying the name of the file associated with the directory entry. If the file name is shorter than six characters, the remaining bytes contain binary zeros. For example, the name ALPHA would be stored as: 414C50484100H.

Extension is a string of up to 3 non-blank ASCII characters that specifies an extension to the file name. Extensions often identify the type of data in the file such as OBJ (object module), or PLM (PL/M source module). As with the file name, unused positions in the extension field are filled with binary zeros.

Attributes are bits that identify certain characteristics of the file. A 1 bit indicates that the file has the attribute, while a 0 bit means that the file does not have the attribute. The bit positions and their corresponding attributes are listed below (bit 0 is the low-order or rightmost bit, bit 7 is the leftmost bit):

- 0: **Invisible.** Files with this attribute are not listed by the ISIS-II DIR command unless the I switch is used. All system files are invisible.
- 1: **System.** Files with this attribute are copied to the disk in drive 1 when the S switch is specified with the ISIS-II FORMAT command.
- 2: **Write-Protect.** Files with this attribute cannot be opened for output or update, nor can they be deleted or renamed.
- 3-6: These positions are reserved for future use.
- 7: **Format.** Files with this attribute are treated as though they are write-protected. In addition, these files are created on a new diskette when the ISIS-II FORMAT command is issued. The system files all have the FORMAT attribute and it should not be given to any other files.

Figure 3. Intel Directory Entry Format

EOF Count contains the number of the last byte in the last data block of the file. If the value of this field is 080H, for example, the last byte in the file is byte number 128 in the last data block (the last block is full).

Number of Data Blocks is an address variable that indicates the number of data blocks currently used by the file. ISIS-II and the RMX/80 Disk File system both maintain a counter called LENGTH that is the current number of bytes in the file. This is calculated as:

$$(\text{Number of Data Blocks} - 1) \times 128 + \text{EOF Count.}$$

Header Block Pointer is the address of the file's header block. The high byte of the field is the sector number and the low byte is the track number. The system "finds" a disk file by searching the directory for the name and then using the header block pointer to seek to the beginning of the file.

Figure 3. Intel Directory Entry Format (Continued)

```

dir: procedure(drv,dens)   public;
  declare drv             byte,
         dens             byte,
         sector          byte,
         i                byte,
         dir$ptr         byte,
         dir$entry       based rdbptr structure (presence byte,
         filename(6) byte,extension(3) byte,
         attribute byte,eof$count byte,
         data$blocks address,header$ptr address),
         size (5)         byte,

         invisible$flag  literally '1',
         system$flag    literally '2',
         protected$flag literally '4',
         format$flag    literally '80H';

/* The disk directory starts at cylinder 1, sector 2 */
call seek(drv,1,0);
do sector=2 to 26;
  call read(drv,1,0,sector,dens);
  do dir$ptr=0 to 112 by 4;
    if dir$entry.presence=7FH then return;
    if dir$entry.presence=0
      then do;
        do i=0 to 5; call co(dir$entry.file$name(i)); end;
        call co(period);
        do i=0 to 2; call co(dir$entry.extension(i)); end;
        do i=0 to 4; call co(space); end;
        call convert$to$decimal(@size,dir$entry.data$blocks);
        do i=0 to 4; call co(size(i)); end;
        If (dir$entry.attribute and invisible$flag) <> 0 then call co("I");
        If (dir$entry.attribute and system$flag) <> 0 then call co("S");
        If (dir$entry.attribute and protected$flag) <> 0 then call co("W");
        If (dir$entry.attribute and format$flag) <> 0 then call co("F");
      end;
  end;
end;
end dir;

```

207885-6

Figure 4. Sample PL/M Directory Procedure

7.0 KEY 8272 SOFTWARE INTERFACING CONSIDERATIONS

This section contains a quick review of Key 8272 Software design features and issues. (Most items have been mentioned in other sections of this application note.) Before designing 8272 software drivers, it is advisable that the information in this section be thoroughly understood.

1. Non-DMA Transfers

In systems that operate without a DMA controller (in the polled or interrupt driven mode), the system software is responsible for counting data transfers to/from the 8272 and generating a TC signal to the FDC when the transfer is complete.

2. Processor Command/Result Phase Interface

In the command phase, the driver software must write the exact number of parameters in the exact order shown in Table 5. During the result phase, the driver must read the complete result status. For example, the Format Track command requires six command bytes and presents seven result bytes. The 8272 will not accept a new command until all result bytes are read. Note that the number of command and result bytes varies from command-to-command. **Command and result phases cannot be shortened.**

During both the command and result phases, the Main Status Register must be read by the driver before each byte of information is read from, or written to, the FDC Data Register. Before each command byte is written, DIO (bit 6) must be low (indicating a data transfer from the processor) and RQM (bit 7) must be high (indicating that the FDC is ready for data). During the result phase, DIO must be high (indicating a data transfer to the processor).

NOTE:

After the 8272 receives a command byte, the RQM flag may remain set for approximately 16 microseconds (with an 8 MHz clock). The driver should not attempt to read the Main Status Register before this time interval has elapsed; otherwise, the driver may erroneously assume that the FDC is ready to accept the next byte.

3. Sector Sizes

The 8272 does not support 128 byte sectors in the MFM (double-density) mode.

4. Drive Status Changes

The 8272 constantly polls all drives for changes in the drive ready status. This polling begins immediately fol-

lowing RESET. An interrupt is generated every time the FDC senses a change in the drive ready status. After reset, the FDC assumes that all drives are "not ready". If a drive is ready immediately after reset, the 8272 generates a drive status change interrupt.

5. Seek Commands

The 8272 FDC does not perform implied seeks. Before issuing a data read or write command, the read/write head must be positioned over the correct cylinder by means of an explicit seek command. If the head is not positioned correctly, a cylinder address error is generated.

6. Interrupt Processing

When the processor receives an interrupt from the FDC, the FDC may be reporting one of two distinct events:

- a) The beginning of the result phase of a previously requested read, write, or scan command.
- b) An asynchronous event such as a seek/recalibrate completion, an attention, an abnormal command termination, or an invalid command.

These two cases are distinguished by the FDC BUSY flag (bit 4) in the Main Status Register. If the FDC BUSY flag is high, the interrupt is of type (a). If the FDC BUSY flag is low, the interrupt was caused by an asynchronous event (b).

A single interrupt from the FDC may signal more than one of the above events. After receiving an interrupt, the processor must continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are ferreted out and serviced.

7. Skip Flag (SK)

The skip flag is used during the execution of Read Data, Read Deleted Data, Read Track, and various Scan commands. This flag permits the FDC to skip unwanted sectors on a disk track.

When performing a Read Data, Read Track, or Scan command, a high SK flag indicates that the FDC is to skip over (not transfer) any sector containing a deleted data address mark. A low SK flag indicates that the FDC is to terminate the command (after reading all the data in the sector) when a deleted data address mark is encountered.

When performing a Read Deleted Data command, a high SK flag indicates that sectors containing normal

data address marks are to be skipped. Note that this is just the opposite situation from that described in the last paragraph. When a data address mark is encountered during a Read Deleted Data command (and the SK flag is low), the FDC terminates the command after reading all the data in the sector.

8. Bad Track Maintenance

The 8272 does not internally maintain bad track information. The maintenance of this information must be performed by system software. As an example of typical bad track operation, assume that a media test determines that track 31 and track 66 of a given floppy disk are bad. When the disk is formatted for use, the system software formats physical track 0 as logical cylinder 0 ($C = 0$ in the command phase parameters), physical track 1 as logical track 1 ($C = 1$), and so on, until physical track 30 is formatted as logical cylinder 30 ($C = 30$). Physical track 31 is bad and should be formatted as logical cylinder FF (indicating a bad track). Next, physical track 32 is formatted as logical cylinder 31, and so on, until physical track 65 is formatted as logical cylinder 64. Next, bad physical track 66 is formatted as logical cylinder FF (another bad track marker), and physical track 67 is formatted as logical cylinder 65. This formatting continues until the last physical track (77) is formatted as logical cylinder 75. Normally, after this formatting is complete, the bad track information is stored in a prespecified area on the floppy disk (typically in a sector on track 0) so that the system will be able to recreate the bad track information when the disk is removed from the drive and reinserted at some later time.

To illustrate how the system software performs a transfer operation on a disk with bad tracks, assume that the

disk drive head is positioned at track 0 and the disk described above is loaded into the drive. If a command to read track 36 is issued by an application program, the system software translates this read command into a seek to physical track 37 (since there is one bad track between 0 and 36, namely 31) followed by a read of logical cylinder 36. Thus, the cylinder parameter C is set to 37 for the Seek command and 36 for the Read Sector command.

REFERENCES

1. Intel, "8272 Single/Double Density Floppy Disk Controller Data Sheet," Intel Corporation, 1980.
2. Intel, "An Intelligent Data Base System Using the 8272," Intel Application Note," AP-116, 1981.
3. Intel, *iSBC 208 Hardware Reference Manual*, Manual Order No. 143078, Intel Corporation, 1980.
4. Intel, *RMX/80 User's Guide*, Manual Order No. 9800522, Intel Corporation, 1978.
5. Brinch Hansen, P., *Operating System Principles*, Prentice-Hall, Inc., New Jersey, 1973.
6. Flores, I., *Computer Software: Programming Systems for Digital Computers*, Prentice-Hall, Inc., New Jersey, 1965.
7. Knuth, D. E., *Fundamental Algorithms*, Addison-Wesley Publishing Company, Massachusetts, 1975.
8. Shaw, A. C., *The Logical Design of Operating Systems*, Prentice-Hall, Inc., New Jersey, 1974.
9. Watson, R. W., *Time Sharing System Design Concepts*, McGraw-Hill, Inc., New York, 1970.
10. Zarrella, J., *Operating Systems: Concepts and Principles*, Microcomputer Applications, California, 1979.

APPENDIX A

8272 FDC DEVICE DRIVER SOFTWARE

PL/M-86 COMPILER 8272 FLOPPY DISK CONTROLLER DEVICE DRIVERS

ISIS-II PL/M-86 V1.2 COMPILATION OF MODULE DRIVERS

OBJECT MODULE PLACED IN :Fl:driv72.OBJ

COMPILER INVOKED BY: plm86 :Fl:driv72.p86 DEBUG

```

$title('8272 floppy disk controller device drivers')
$nointvector
$optimize(2)
$large

1   drivers: do;
2   1   declare
      /* floppy disk port definitions */
      fdc$status$port   literally '30H',      /* 8272 status port */
      fdc$data$port     literally '31H';      /* 8272 data port */

3   1   declare
      /* floppy disk commands */
      sense$int$status  literally '08H';

4   1   declare
      /* interrupt definitions */
      fdc$int$level     literally '33';      /* fdc interrupt level */

5   1   declare
      /* return status and error codes */
      error              literally '0',
      ok                 literally '1',
      complete           literally '3',
      false              literally '0',
      true               literally '1',
      error$in           literally 'not',
      propagate$error    literally 'return error',

      stat$ok            literally '0',      /* fdc operation completed without errors */
      stat$busy          literally '1',      /* fdc is busy, operation cannot be started */
      stat$error         literally '2',      /* fdc operation error */
      stat$command$error literally '3',      /* fdc not ready for command phase */
      stat$result$error  literally '4',      /* fdc not ready for result phase */
      stat$invalid       literally '5';      /* invalid fdc command */

6   1   declare
      /* masks */
      busy$mask          literally '10H',
      DIO$mask           literally '40H',
      RQM$mask           literally '80H',
      seek$mask          literally '0FH',
      result$error$mask  literally '0C0H',
      result$drive$mask  literally '03H',
      result$ready$mask  literally '08H';

7   1   declare
      /* drive numbers */
      max$no$drives      literally '3',
      fdc$general        literally '4';

8   1   declare
      /* miscellaneous control */
      any$drive$seeking  literally '((input(fdc$status$port) and seek$mask) <> 0)',
      command$code       literally '(docb.disk$command(0) and LPH)',
      DIO$set$for$input  literally '((input(fdc$status$port) and DIO$mask)=0)',
      DIO$set$for$output literally '((input(fdc$status$port) and DIO$mask)<>0)',
      extract$drive$no   literally '(docb.disk$command(1) and 03H)',
      fdc$busy           literally '((input(fdc$status$port) and busy$mask) <> 0)',
      no$fdc$error       literally '(possible$error(command$code) and ((docb.disk$result(0)
                                and result$error$mask) = 0)',
      wait$for$op$complete literally 'do while not operation$complete(drive$no); end',
      wait$for$RQM       literally 'do while (input(fdc$status$port) and RQM$mask) = 0; end;';

9   1   declare
      /* structures */
      docb$type          literally          /* disk operation control block */
      (dma$op byte,dma$addr word, dma$addr$ext byte,dma$count word,
       disk$command(9) byte,disk$result(7) byte,misc byte);

$seject
10  1   declare
      drive$status$change(4) byte public, /* when set - indicates that drive status changed */
      drive$ready(4) byte public;        /* current status of drives */

```

207885-7


```

38 2      /* wait for "master request" flag */
      wait$for$RQM;

41 2      /* check data direction flag */
      if DIO$set$for$output
43 2          then return ok;
          else return error;

44 2      end fdc$ready$for$result;

      /***** output a single command/parameter byte to the 8272 fdc. The "data$byte"
      parameter is the byte to be output to the fdc. *****/

45 1      output$byte$to$fdc: procedure(data$byte) byte;
46 2          declare data$byte byte;

          /* check to see if fdc is ready for command */
47 2          if not fdc$ready$for$command
              then propagate$error;

49 2          output(fdc$data$port)=data$byte;

50 2          return ok;
51 2      end output$byte$to$fdc;

      /***** input a single result byte from the 8272 fdc. The "data$byte$ptr"
      parameter is a pointer to the memory location that is to contain
      the input byte. *****/

52 1      input$byte$from$fdc: procedure(data$byte$ptr) byte;
53 2          declare data$byte$ptr pointer;
54 2          declare
              data$byte based data$byte$ptr byte,
              status byte;

          /* check to see if fdc is ready */
55 2          status=fdc$ready$for$result;
56 2          if error$in status
              then propagate$error;

          /* check for result phase complete */
58 2          if status=complete
              then return complete;

60 2          data$byte=input(fdc$data$port);
61 2          return ok;
62 2      end input$byte$from$fdc;

      Seject

      /***** output the dma mode, the dma address, and the dma word count to the
      8237 dma controller. Also output the high order four bits of the
      address to the address extension latch. Finally, start the disk
      dma channel. The "docb$ptr" parameter is a pointer to the appropriate
      disk operation control block. *****/

63 1      output$controls$to$dma: procedure(docb$ptr);
64 2          declare docb$ptr pointer;
65 2          declare docb based docb$ptr structure docbtype;

66 2          declare
              /* dma port definitions */
              dma$upper$addr$port    literally '10H',      /* upper 4 bits of current address */
              dma$disk$addr$port     literally '00H',      /* current address port */
              dma$disk$word$count    literally '01H',      /* word count port */
              dma$command$port       literally '08H',      /* command port */
              dma$mode$port           literally '0BH',      /* mode port */
              dma$mask$sr$port        literally '0AR',      /* mask set/reset port */
              dma$clear$fff$port      literally '0CH',      /* clear first/last flip-flop port */
              dma$master$clear$port   literally '0DH',      /* dma master clear port */
              dma$mask$port           literally '0FH',      /* parallel mask set port */

              dma$disk$chan$start     literally '00H',      /* dma mask to start disk channel */
              dma$extended$write      literally 'shl(1,5)', /* extended write flag */
              dma$single$transfer     literally 'shl(1,6)'; /* single transfer flag */

67 2          if docb.dma$op < 3
              then do;
                  /* set dma mode and clear first/last flip-flop */
69 3                  output(dma$mode$port)=shl(docb.dma$op,2) or 40H;
70 3                  output(dma$clear$fff$port)=0;

```

207885-9

```

71 3      /* set dma address */
72 3      output(dma$disk$addr$port)=low(docb.dma$addr);
73 3      output(dma$disk$addr$port)=high(docb.dma$addr);
74 3      output(dma$supper$addr$port)=docb.dma$addr$ext;

74 3      /* output disk transfer word count to dma controller */
75 3      output(dma$disk$word$count)=low(docb.dma$count);
76 3      output(dma$disk$word$count)=high(docb.dma$count);

76 3      /* start dma channel 0 for fdc */
77 3      output(dma$mask$sr$port)=dma$disk$chan$start;
78 2      end;

78 2      end output$controls$to$dma;

      /**** output a high-level disk command to the 8272 fdc. The number of bytes
      required for each command is contained in the "command$length" table.
      The "docb$ptr" parameter is a pointer to the appropriate disk operation
      control block. ****/

79 1      output$command$to$fdc: procedure(docb$ptr) byte;
80 2      declare docb$ptr pointer;

81 2      declare
82 2      docb based docb$ptr structure docb$type,
      cmd$byte$no byte;

82 2      disable;

83 2      /* output all command bytes to the fdc */
84 3      do cmd$byte$no=0 to command$length(command$code)-1;
85 3      if error$in output$byte$to$fdc(docb.disk$command(cmd$byte$no))
86 3      then do; enable; propagate$error; end;
87 3      end;

88 2      enable;
89 2      return ok;
90 2      end output$command$to$fdc;

      /**** input the result data from the 8272 fdc during the result phase (after
      command execution). The "docb$ptr" parameter is a pointer to the
      appropriate disk operation control block. ****/

93 1      input$result$from$fdc: procedure(docb$ptr) byte;
94 2      declare docb$ptr pointer;
95 2      declare
96 2      docb based docb$ptr structure docb$type,
      result$byte$no byte,
      temp byte,
      status byte;

96 2      disable;

97 2      do result$byte$no=0 to 7;
98 3      status=input$byte$from$fdc(@temp);
99 3      if error$in status
100 3      then do; enable; propagate$error; end;
101 3      if status=complete
102 3      then do; enable; return ok; end;
103 3      docb.disk$result(result$byte$no)=temp;
104 3      end;

105 2      enable;
106 2      if fdc$busy
107 2      then return error;
108 2      else return ok;
109 2      end input$result$from$fdc;

      /**** cleans up after the execution of a disk operation that has no result
      phase. The procedure is also used after some disk operation errors.
      "drv" is the drive number, and "cc" is the command code for the
      disk operation. ****/

116 1      operation$clean$up: procedure(drv,cc);
117 2      declare (drv,cc) byte;

118 2      disable;
119 2      operation$in$progress(drv)=false;

```

207885-10

```

120 2      if not overlap$operation(cc)
122 2          then global$drive$no=0;
           enable;
123 2      end operation$clean$up;

$reject

/**** execute the disk operation control block specified by the pointer
parameter "docb$ptr". The "status$ptr" parameter is a pointer to
a byte variable that is to contain the status of the requested
operation when it has been completed. Six status conditions are
possible on return:

    0  The specified operation was completed without error.
    1  The fdc is busy and the requested operation cannot be started.
    2  Fdc error (further information is contained in the result
       storage portion of the disk operation control block - as
       described in the 8272 data sheet).
    3  Transfer error during output of the command bytes to the fdc.
    4  Transfer error during input of the result bytes from the fdc.
    5  Invalid fdc command. *****/

124 1      execute$docb: procedure(docb$ptr,status$ptr) public;
           /* execute a disk operation control block */

125 2      declare docb$ptr pointer, status$ptr pointer;
126 2      declare
           docb based docb$ptr structure docb$type,
           status based status$ptr byte,
           drive$no byte;

           /* check command validity */
127 2      if not valid$command(command$code)
           then do; status=stat$invalid; return; end;

           /* determine if command has a drive number field - if not, set the drive
           number for a general fdc command */
132 2      if drive$no$present(command$code)
           then drive$no=extract$drive$no;
134 2      else drive$no=fdc$general;

           /* an overlapped operation can not be performed if the fdc is busy */
135 2      if overlap$operation(command$code) and fdc$busy
           then do; status=stat$busy; return; end;

           /* for a non-overlapped operation, check fdc busy or any drive seeking */
140 2      if not overlap$operation(command$code) and (fdc$busy or any$drive$seeking)
           then do; status=stat$busy; return; end;

           /* check for drive operation in progress - if none, set flag and start operation */
145 2      disable;
146 2      if operation$in$progress(drive$no)
           then do; enable; status=stat$busy; return; end;
152 2      else operation$in$progress(drive$no)=true;

           /* at this point, an fdc operation is about to begin, so:
           1. reset the operation complete flag
           2. set the docb pointer for the current operation
           3. if this is not an overlapped operation, set the global drive
           number for the subsequent result phase interrupt. */
153 2      operation$complete(drive$no)=0;
154 2      operation$docb$ptr(drive$no)=docb$ptr;

155 2      if not overlap$operation(command$code)
           then global$drive$no=drive$no+1;
           enable;

158 2      call output$control$to$dma(docb$ptr);
159 2      if error$in output$command$to$fdc(docb$ptr)
           then do;
161 3          call operation$clean$up(drive$no,command$code);
162 3          status=stat$command$error;
163 3          return;
164 3      end;

           /* return immediately if the command has no result phase or completion interrupt - specify */
165 2      if no$result(command$code)
           then do;
167 3          call operation$clean$up(drive$no,command$code);
168 3          status=stat$ok;
169 3          return;
170 3      end;

```

207885-11

```

171 2      if immed$result(command$code)
173 3          then do;
175 4              if error$in input$result$from$fdc(docb$ptr)
176 4                  then do;
177 4                      call operation$clean$up(drive$no,command$code);
178 4                      status=stat$result$error;
179 4                      return;
180 2                  end;
181 3              end do;
182 3              else do;
183 3                  wait$for$op$complete;
184 3                  if docb.misc = error
185 3                      then do; status=stat$result$error; return; end;
186 3                  end;
187 3              end do;
188 3          end;
189 2      if no$fdc$error
190 2          then status=stat$ok;
191 2          else status=stat$error;
192 2      end execute$docb;
$ject
/**** copy disk command results from the interrupt control block to the
currently active disk operation control block if a disk operation is
in progress. ****/
193 1      copy$int$result: procedure(drv);
194 2          declare drv byte;
195 2          declare
196 2              i byte,
197 2              docb$ptr pointer,
198 2              docb based docb$ptr structure docb$type;
199 2          if operation$in$progress(drv)
200 2              then do;
201 3              docb$ptr=operation$docb$ptr(drv);
202 3              do i=1 to 6; docb.disk$result(i)=interrupt$docb.disk$result(i); end;
203 3              docb.misc=ok;
204 3              operation$in$progress(drv)=false;
205 3              operation$complete(drv)=true;
206 3              end;
207 2          end copy$int$result;

```

/**** interrupt processing for 8272 fdc drivers. Basically, two types of interrupts are generated by the 8272: (a) when the execution phase of an operation has been completed, an interrupt is generated to signal the beginning of the result phase (the fdc busy flag is set when this interrupt is received), and (b) when an overlapped operation is completed or an unexpected interrupt is received (the fdc busy flag is not set when this interrupt is received).

When interrupt type (a) is received, the result bytes from the operation are read from the 8272 and the operation complete flag is set.

When an interrupt of type (b) is received, the interrupt result code is examined to determine which of the following four actions are indicated:

1. An overlapped option (recalibrate or seek) has been completed. The result data is read from the 8272 and placed in the currently active disk operation control block.
2. An abnormal termination of an operation has occurred. The result data is read and placed in the currently active disk operation control block.
3. The execution of an invalid command has been attempted. This signals the successful completion of all interrupt processing.
4. The ready status of a drive has changed. The "drive\$ready" and "drive\$ready\$status" change tables are updated. If an operation is currently in progress on the affected drive, the result data is placed in the currently active disk operation control block.

After an interrupt is processed, additional sense interrupt status commands must be issued and processed until an invalid command result is returned from the fdc. This action guarantees that all "hidden" interrupts are serviced. ****/

207885-12

```

207 1   fdcint: procedure public interrupt fdc$int$level;
208 2   declare
        invalid byte,
        drive$no byte,
        docb$ptr pointer,
        docb based docb$ptr structure docb$type;

209 2   declare
        /* interrupt port definitions */
        ocw2           literally "70H",
        nseci          literally "shl(1,5)";

210 2   declare
        /* miscellaneous flags */
        result$code   literally "shr(interrupt$docb.disk$result(0) and result$error$mask,6)",
        result$drive$ready   literally "{(interrupt$docb.disk$result(0) and result$ready$mask) = 0}",
        extract$result$drive$no   literally "{(interrupt$docb.disk$result(0) and result$drive$mask)",
        end$of$interrupt   literally "output(ocw2)=nseci";

        /* if the fdc is busy when an interrupt is received, then the result
        phase of the previous non-overlapped operation has begun */
211 2   if fdc$busy
        then do;
213 3       /* process interrupt if operation in progress */
        if global$drive$no <> 0
        then do;
215 4           docb$ptr=operation$docb$ptr(global$drive$no-1);
216 4           if error$in input$result$from$fdc(docb$ptr)
        then docb.misc=error;
218 4           else docb.misc=ok;
219 4           operation$in$progress(global$drive$no-1)=false;
220 4           operation$complete(global$drive$no-1)=true;
221 4           global$drive$no=0;
222 4           end;
223 3       end;

        /* if the fdc is not busy, then either an overlapped operation has been
        completed or an unexpected interrupt has occurred (e.g., drive status
        change) */
224 2   else do;
225 3       invalid=false;
226 3       do while not invalid;

                /* perform a sense interrupt status operation - if errors are detected,
                in the actual fdc interface, interrupt processing is discontinued */
227 4       if error$in output$byte$to$fdc(sense$int$status) then go to ignore;
229 4       if error$in input$result$from$fdc(@interrupt$docb) then go to ignore;

231 4       do case result$code;

                /* case 0 - operation complete */
232 5       do;
233 6           drive$no=extract$result$drive$no;
234 6           call copy$int$result(drive$no);
235 6           end;

                /* case 1 - abnormal termination */
236 5       do;
237 6           drive$no=extract$result$drive$no;
238 6           call copy$int$result(drive$no);
239 6           end;

                /* case 2 - invalid command */
240 5       invalid=true;

                /* case 3 - drive ready change */
241 5       do;
242 6           drive$no=extract$result$drive$no;
243 6           call copy$int$result(drive$no);
244 6           drive$status$change(drive$no)=true;
245 6           if result$drive$ready
        then drive$ready(drive$no)=true;
        else drive$ready(drive$no)=false;

247 6           end;
248 6       end;
249 5       end;
250 4       end;
251 3       end;

252 2       ignore: end$of$interrupt;
253 2       end fdcint;

254 1   end drivers;

```

207885-13

```

MODULE INFORMATION:
CODE AREA SIZE      = 0615H  1557D
CONSTANT AREA SIZE  = 0000H   0D
VARIABLE AREA SIZE  = 0050H   80D
MAXIMUM STACK SIZE  = 0032H   50D
564 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-86 COMPILATION

207885-14

APPENDIX B

8272 FDC EXERCISER PROGRAM

PL/M-86 COMPILER 8272 FLOPPY DISK DRIVER EXERCISE PROGRAM

ISIS-II PL/M-86 V1.2 COMPILATION OF MODULE RUN72
 OBJECT MODULE PLACED IN :Pl:run72.OBJ
 COMPILER INVOKED BY: plm86 :Pl:run72.p86 DEBUG

```

$title ('8272 floppy disk driver exercise program')
$nointvector
$optimize(2)
$large
run72: do;

1
2 1 declare
  docb$type          literally          /* disk operation control block */
  ('dma$op byte,dma$addr word,dma$addr$ext byte,dma$count word,
   disk$command(9) byte,disk$result(7) byte,misc byte');

3 1 declare
  /* 8272 fdc commands */
  fm          literally '0',
  mfm         literally '1',
  dma$mode    literally '0',
  non$dma$mode literally '1',
  recalibrate$command literally '7',
  specify$command literally '3',
  read$command literally '6',
  write$command literally '5',
  format$command literally '0FH',
  seek$command  literally '0FH';

4 1 declare
  dma$verify    literally '0',
  dma$read      literally '1',
  dma$write     literally '2',
  dma$snoop     literally '3';

5 1 declare
  /* disk operation control blocks */
  format$docb   structure docb$type,
  seek$docb     structure docb$type,
  recalibrate$docb structure docb$type,
  specify$docb  structure docb$type,
  read$docb     structure docb$type,
  write$docb    structure docb$type;

6 1 declare
  step$rate     byte,
  head$load$time byte,
  head$unload$time byte,
  filler$byte   byte,
  operation$status byte,
  interleave    byte,
  format$gap    byte,
  read$write$gap byte,
  index         byte,
  drive         byte,
  density       byte,
  multitrack    byte,
  sector        byte,
  cylinder      byte,
  head          byte,
  tracks$per$disk byte,
  sectors$per$track byte,
  bytes$per$sector$code byte,
  bytes$per$sector word;
  /* disk drive head */
  /* number of bytes in a sector on the disk */

7 1 declare
  /* read and write buffers */
  fmbblk(1024) byte public,
  wrbuf(1024)  byte public,
  rdbuf(1024)  byte public;

8 1 declare
  /* disk format initialization tables */
  sec$trk$table(3) byte data(26,15,8),
  fmt$gap$table(8) byte data(1BH,2AH,3AH,0,0,36H,54H,74H),
  rd$wr$gap$table(8) byte data(07H,0EH,1BH,0,0,0EH,1BH,35H);

```

207885-15


```

9 1 declare
/* external pointer tables and interrupt vector */
   rdbptr (2)          word external,
   wrbptr (2)         word external,
   fbp (2)           word external,
   intptr (2)        word external,
   intvec(80H)       word external;

10 1 execute$dccb: procedure(dccb$ptr,status$ptr) external;
11 2   declare dccb$ptr pointer, status$ptr pointer;
12 2   end execute$dccb;

13 1 initialize$drivers: procedure external;
14 2   end initialize$drivers;

$seject

/**** specify step rate ("srt"), head load time ("hlt"), head unload time ("hut"),
and dma or non-dma operation ("nd"). ****/

15 1 specify: procedure(srt, hlt, hut, nd);
16 2   declare (srt, hlt, hut, nd) byte;

17 2   specify$dccb.dma$op=dma$noop;
18 2   specify$dccb.disk$command(0)=specify$command;
19 2   specify$dccb.disk$command(1)=shl(not srt)+1,4) or shr(hut,4);
20 2   specify$dccb.disk$command(2)=(hlt and OFEH) or (nd and 1);
21 2   call execute$dccb(@specify$dccb,@operation$status);

22 2   end specify;

/**** recalibrate disk drive
8272 automatically steps out until the track 0 signal is activated
by the disk drive. ****/

23 1 recalibrate: procedure(drv);
24 2   declare drv byte;

25 2   recalibrate$dccb.dma$op=dma$noop;
26 2   recalibrate$dccb.disk$command(0)=recalibrate$command;
27 2   recalibrate$dccb.disk$command(1)=drv;
28 2   call execute$dccb(@recalibrate$dccb,@operation$status);

29 2   end recalibrate;

/**** seek drive "drv", head (side) "hd" to cylinder "cyl". ****/

30 1 seek: procedure(drv, cyl, hd);
31 2   declare (drv, cyl, hd) byte;

32 2   seek$dccb.dma$op=dma$noop;
33 2   seek$dccb.disk$command(0)=seek$command;
34 2   seek$dccb.disk$command(1)=drv or shl(hd,2);
35 2   seek$dccb.disk$command(2)=cyl;
36 2   call execute$dccb(@seek$dccb,@operation$status);

37 2   end seek;

/**** format a complete side ("head") of a single floppy disk in drive "drv". The density,
(single or double) is specified by flag "dens". ****/

38 1 format: procedure(drv, dens, intlve);
/* format disk */
39 2   declare (drv, dens, intlve) byte;
40 2   declare physical$sector byte;

41 2   call recalibrate(drv);
42 2   do cylinder=0 to tracks$per$disk-1;
/* set sector numbers in format block to zero before computing interleave */
43 3     do physical$sector=1 to sectors$per$track; fmbblk((physical$sector-1)*4+2)=0; end;
/* physical sector 1 equals logical sector 1 */
46 3     physical$sector=1;

/* assign interleaved sectors */
47 3     do sector=1 to sectors$per$track;
48 4       index=(physical$sector-1)*4;

```

207885-16

```

49 4      /* change sector and index if sector has already been assigned */
        do while fmtblk(index+2) <> 0; index=index+4; physical$sector=physical$sector+1; end;

53 4      /* set cylinder, head, sector, and size code for current sector into table */
54 4      fmtblk(index)=cylinder;
55 4      fmtblk(index+1)=head;
56 4      fmtblk(index+2)=sector;
        fmtblk(index+3)=bytes$per$sector$code;

57 4      /* update physical sector number by interleave */
58 4      physical$sector=physical$sector+intlv;
        if physical$sector > sectors$per$track
60 4      then physical$sector=physical$sector-sectors$per$track;
        end;

61 3      /* seek to next cylinder */
        call seek(drv,cylinder,head);

62 3      /* set up format control block */
63 3      format$dccb.dma$op=dma$write;
64 3      format$dccb.dma$addr=fbptr(0)+shl(fbptr(1),4);
65 3      format$dccb.dma$addr$ext=0;
66 3      format$dccb.dma$count=sectors$per$track*4-1;
67 3      format$dccb.disk$command(0)=format$command or shl(dens,6);
68 3      format$dccb.disk$command(1)=drv or shl(head,2);
69 3      format$dccb.disk$command(2)=bytes$per$sector$code;
70 3      format$dccb.disk$command(3)=sectors$per$track;
71 3      format$dccb.disk$command(4)=format$gap;
72 3      format$dccb.disk$command(5)=filler$byte;
73 3      call execute$dccb(@format$dccb,@operation$status);
        end;

74 2      end format;

/**** write sector "sec" on drive "drv" at head "hd" and cylinder "cyl". The
disk recording density is specified by the "dens" flag. Data is expected to be
in the global write buffer ("wrbuf"). ****/

75 1      write: procedure(drv,cyl,hd,sec,dens);
76 2      declare (drv,cyl,hd,sec,dens) byte;

77 2      write$dccb.dma$op=dma$write;
78 2      write$dccb.dma$addr=wrbptr(0)+shl(wrbptr(1),4);
79 2      write$dccb.dma$addr$ext=0;
80 2      write$dccb.dma$count=bytes$per$sector-1;
81 2      write$dccb.disk$command(0)=write$command or shl(dens,6) or shl(multitrack,7);
82 2      write$dccb.disk$command(1)=drv or shl(hd,2);
83 2      write$dccb.disk$command(2)=cyl;
84 2      write$dccb.disk$command(3)=hd;
85 2      write$dccb.disk$command(4)=sec;
86 2      write$dccb.disk$command(5)=bytes$per$sector$code;
87 2      write$dccb.disk$command(6)=sectors$per$track;
88 2      write$dccb.disk$command(7)=read$write$gap;
89 2      if bytes$per$sector$code = 0
        then write$dccb.disk$command(8)=bytes$per$sector;
        else write$dccb.disk$command(8)=OFFH;
91 2      call execute$dccb(@write$dccb,@operation$status);
92 2

93 2      end write;

/**** read sector "sec" on drive "drv" at head "hd" and cylinder "cyl". The
disk recording density is defined by the "dens" flag. Data is read into
the global read buffer ("rdbuf"). ****/

94 1      read: procedure(drv,cyl,hd,sec,dens);
95 2      declare (drv,cyl,hd,sec,dens) byte;

96 2      read$dccb.dma$op=dma$read;
97 2      read$dccb.dma$addr=rdbptr(0)+shl(rdbptr(1),4);
98 2      read$dccb.dma$addr$ext=0;
99 2      read$dccb.dma$count=bytes$per$sector-1;
100 2      read$dccb.disk$command(0)=read$command or shl(dens,6) or shl(multitrack,7);
101 2      read$dccb.disk$command(1)=drv or shl(hd,2);
102 2      read$dccb.disk$command(2)=cyl;
103 2      read$dccb.disk$command(3)=hd;
104 2      read$dccb.disk$command(4)=sec;
105 2      read$dccb.disk$command(5)=bytes$per$sector$code;
106 2      read$dccb.disk$command(6)=sectors$per$track;
107 2      read$dccb.disk$command(7)=read$write$gap;

```

207885-17

```

108 2      if bytes$per$sector$code = 0
110 2          then read$dock.disk$command(8)=bytes$per$sector;
111 2          else read$dock.disk$command(8)=0FFH;
112 2          call execute$dock(@read$dock,@operation$status);

112 2      end read;

Seject

/**** initialize system by setting up 8237 dma controller and 8259A interrupt
controller. ****/

113 1      initializeSystem: procedure;
114 2      declare
          /* I/O ports */
          dma$disk$addr$port      literally  '00H',      /* current address port */
          dma$disk$word$count$port literally  '01H',      /* word count port */
          dma$command$port        literally  '08H',      /* command port */
          dma$mode$port           literally  '0BH',      /* mode port */
          dma$mask$ar$port         literally  '0AH',      /* mask set/reset port */
          dma$clear$ff$port        literally  '0CH',      /* clear first/last flip-flop port */
          dma$master$clear$port    literally  '0DH',      /* dma master clear port */
          dma$mask$port           literally  '0FH',      /* parallel mask set port*/
          dma$c1$addr$port         literally  '02H',
          dma$c1$word$count$port   literally  '03H',
          dma$c2$addr$port         literally  '04H',
          dma$c2$word$count$port   literally  '05H',
          dma$c3$addr$port         literally  '06H',
          dma$c3$word$count$port   literally  '07H',
          icw1                     literally  '70H',
          icw2                     literally  '71H',
          icw4                     literally  '71H',
          ocw1                     literally  '71H',
          ocw2                     literally  '70H',
          ocw3                     literally  '70H';

115 2      declare
          /* misc masks and literals */
          dma$extended$write      literally  'shl(1,5)', /* extended write flag */
          dma$single$transfer      literally  'shl(1,6)', /* single transfer flag */
          dma$disk$mode           literally  '40H',
          dma$c1$mode             literally  '41H',
          dma$c2$mode             literally  '42H',
          dma$c3$mode             literally  '43H',
          mode$8088               literally  '1',
          interrupt$base          literally  '20H',
          single$controller       literally  'shl(1,1)',
          level$sensitive         literally  'shl(1,3)',
          control$word$4$required literally  '1',
          base$icw1               literally  '10H',
          mask$all                literally  '0FFH',
          disk$interrupt$mask     literally  '1';

116 2      output(dma$master$clear$port)=0; /* master reset */
117 2      output(dma$mode$port)=dma$extended$write; /* set dma command mode */

          /* set all dma registers to valid values */
118 2      output(dma$mask$port)=mask$all; /* mask all channels */

          /* set all addresses to zero */
119 2      output(dma$clear$ff$port)=0; /* reset first/last flip-flop */
120 2      output(dma$disk$addr$port)=0;
121 2      output(dma$disk$addr$port)=0;
122 2      output(dma$c1$addr$port)=0;
123 2      output(dma$c1$addr$port)=0;
124 2      output(dma$c2$addr$port)=0;
125 2      output(dma$c2$addr$port)=0;
126 2      output(dma$c3$addr$port)=0;
127 2      output(dma$c3$addr$port)=0;

          /* set all word counts to valid values */
128 2      output(dma$clear$ff$port)=0; /* reset first/last flip-flop */
129 2      output(dma$disk$word$count$port)=1;
130 2      output(dma$disk$word$count$port)=1;
131 2      output(dma$c1$word$count$port)=1;
132 2      output(dma$c1$word$count$port)=1;
133 2      output(dma$c2$word$count$port)=1;
134 2      output(dma$c2$word$count$port)=1;
135 2      output(dma$c3$word$count$port)=1;
136 2      output(dma$c3$word$count$port)=1;

```

207885-18

```

137 2      /* initialize all dma channel modes */
138 2      output(dma$mode$port)=dma$disk$mode;
139 2      output(dma$mode$port)=dma$c1$mode;
140 2      output(dma$mode$port)=dma$c2$mode;
141 2      output(dma$mode$port)=dma$c3$mode;

141 2      /* initialize 8259A interrupt controller */
142 2      output(icw1)=single$controller or level$sensitive or control$word4$required or base$icw1;
143 2      output(icw2)=interrupt$base;
144 2      output(icw4)=mode$8088; /* set 8088 interrupt mode */
144 2      output(ocw1)=not disk$interrupt$mask; /* mask all interrupts except disk */

145 2      /* initialize interrupt vector for fdc */
146 2      intvec(40H)=intptr(0);
146 2      intvec(41H)=intptr(1);

147 2      end initialize$system;

Seject

/**** main program: first format disk (all tracks on side (head) 0. Then
read each sector on every track of the disk forever. ****/

148 1      declare drive$ready(4) byte external;

149 1      /* disable until interrupt vector setup and initialization complete */
149 1      disable;

150 1      /* set initial floppy disk parameters */
151 1      density=mfm; /* double-density */
152 1      head=0; /* single sided */
153 1      multitrack=0; /* no multitrack operation */
154 1      filler$byte=55H; /* for format */
155 1      tracks$per$disk=77; /* normal floppy disk drive */
156 1      bytes$per$sector=1024; /* 1024 bytes in each sector */
157 1      interleave=6; /* set track interleave factor */
158 1      step$rate=11; /* 10ms for SA800 plus 1 for uncertainty */
159 1      head$load$time=40; /* 40ms head load for SA800 */
159 1      head$unload$time=240; /* keep head loaded as long as possible */

160 1      /* derive dependent parameters from those above */
161 1      bytes$per$sector$code=shr(bytes$per$sector,7);
162 2      do index=0 to 3;
162 2          if (bytes$per$sector$code and 1) <> 0
167 2              then do; bytes$per$sector$code=index; go to donebc; end;
168 2              else bytes$per$sector$code=shr(bytes$per$sector$code,1);
169 1          donebc;
170 1          sectors$per$track=sec$trk$stable(bytes$per$sector$code-density);
171 1          format$gap=fmt$gap$stable(shl(density,2)+bytes$per$sector$code);
172 1          read$write$gap=rd$wr$gap$stable(shl(density,2)+bytes$per$sector$code);

172 1      /* initialize system and drivers */
173 1      call initialize$system;
173 1      call initialize$drivers;

174 1      /* reenable interrupts and give 8272 a chance to report on drive status
175 1      before proceeding */
175 1      enable;
175 1      call time(10);

176 1      /* specify disk drive parameters */
176 1      call specify(step$rate,head$load$time,head$unload$time,dma$mode);

177 1      drive=0; /* run single disk drive #0 */

178 1      /* wait until drive ready */
179 2      do while 1;
179 2          if drive$ready(drive)
181 2              then go to start;
182 1          end;

182 1      start;
183 1      call format(drive,density,interleave);
183 1      do while 1;
184 2          do cylinder=0 to tracks$per$disk-1;
185 3              call seek(drive,cylinder,head);
186 3              do sector=1 to sectors$per$track;

187 4          /* set up write buffer */
187 4          do index=0 to bytes$per$sector-1; wrbuf(index)=index+sector+cylinder; end;

```

207885-19

```
190 4      call write(drive,cylinder,head,sector,density);
191 4      call read(drive,cylinder,head,sector,density);

192 4      /* check read buffer against write buffer */
          if cmpw(@wrbuf,@rdbuf,shr(bytes$per$sector,1)) <> 0FFFFH
          then halt;
194 4      end;
195 3      end;
196 2      end;

197 1      end run72;
```

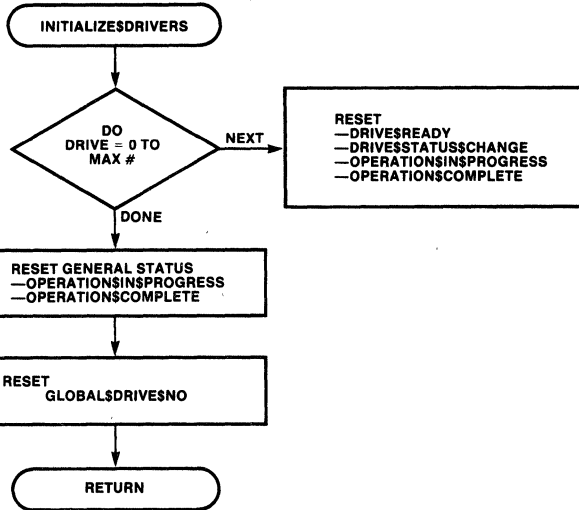
MODULE INFORMATION:

```
CODE AREA SIZE      = 0570H   1392D
CONSTANT AREA SIZE  = 0000H     0D
VARIABLE AREA SIZE  = 0907H   2311D
MAXIMUM STACK SIZE  = 0022H     34D
412 LINES READ
0 PROGRAM ERROR(S)
```

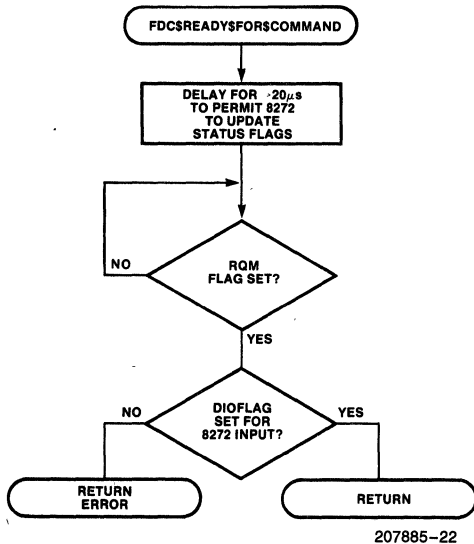
END OF PL/M-86 COMPILATION

207885-20

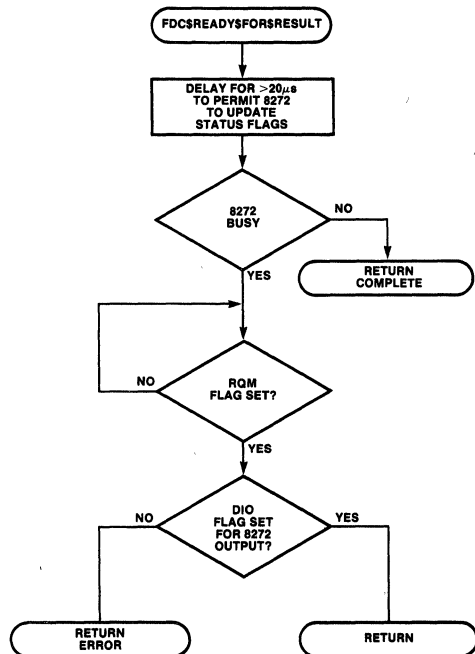
APPENDIX C 8272 DRIVER FLOWCHARTS



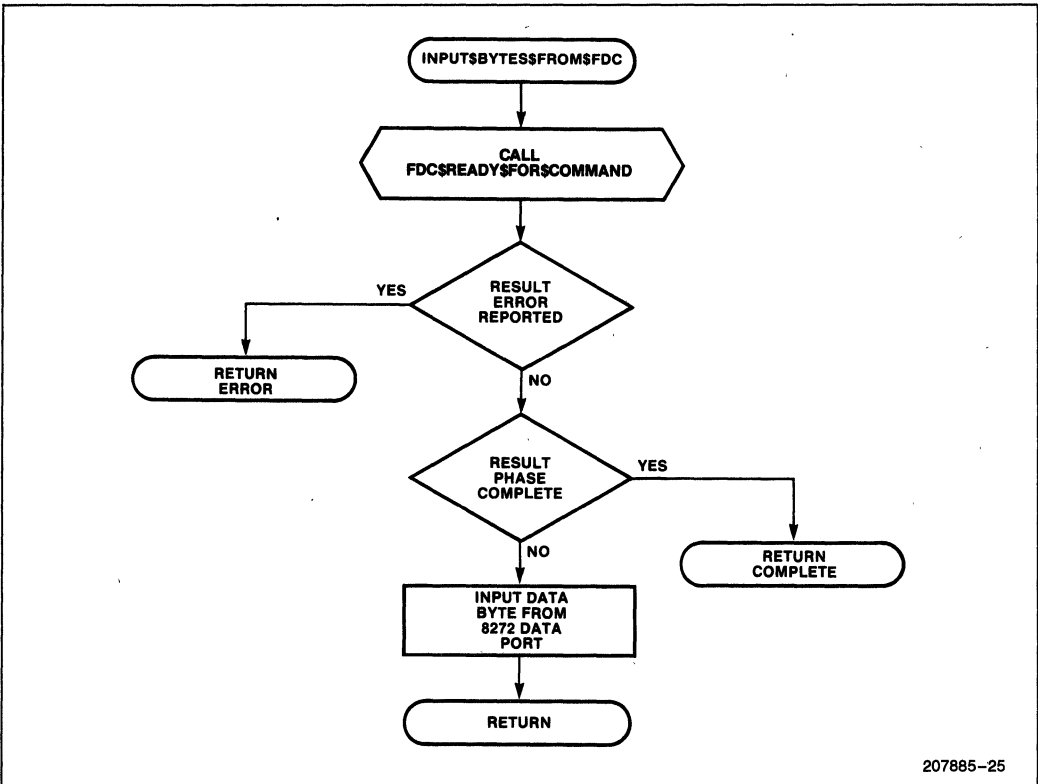
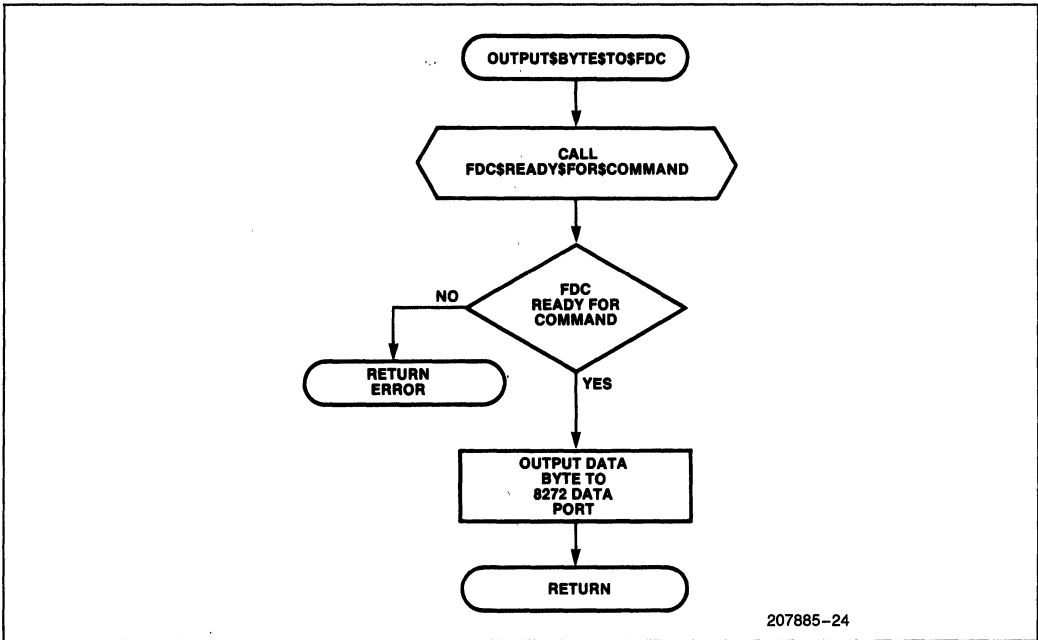
207885-21

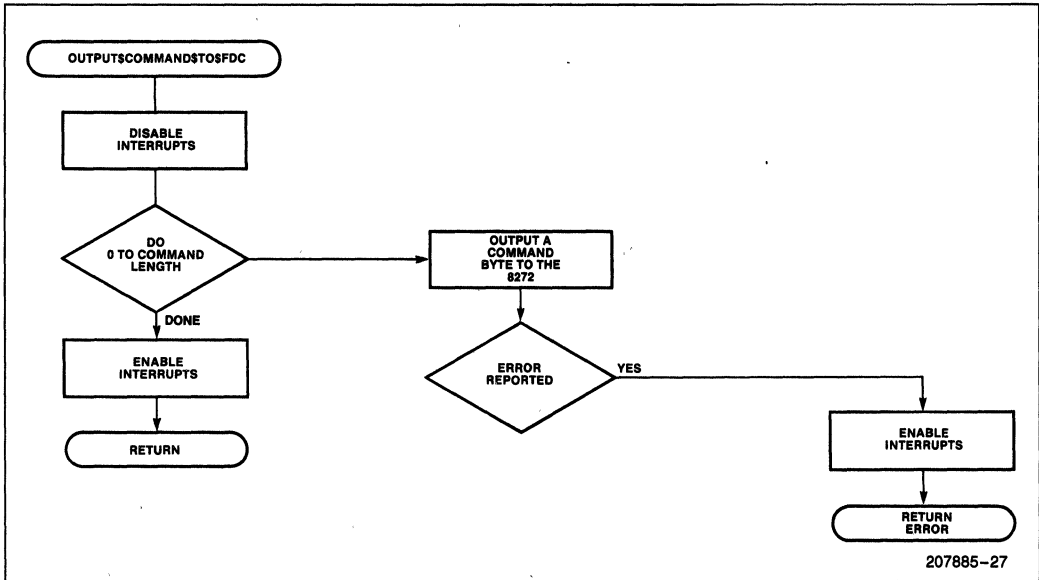
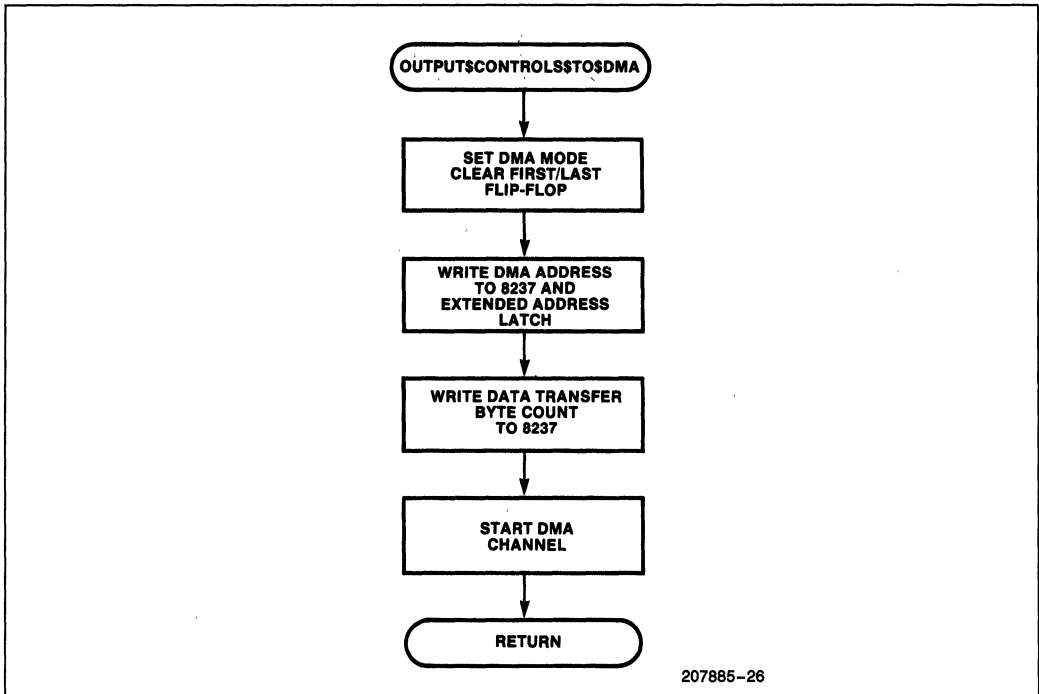


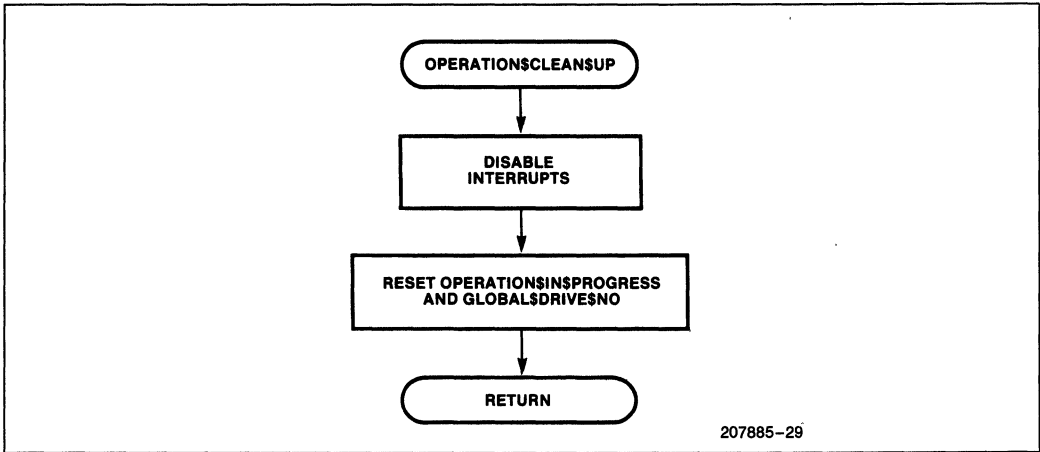
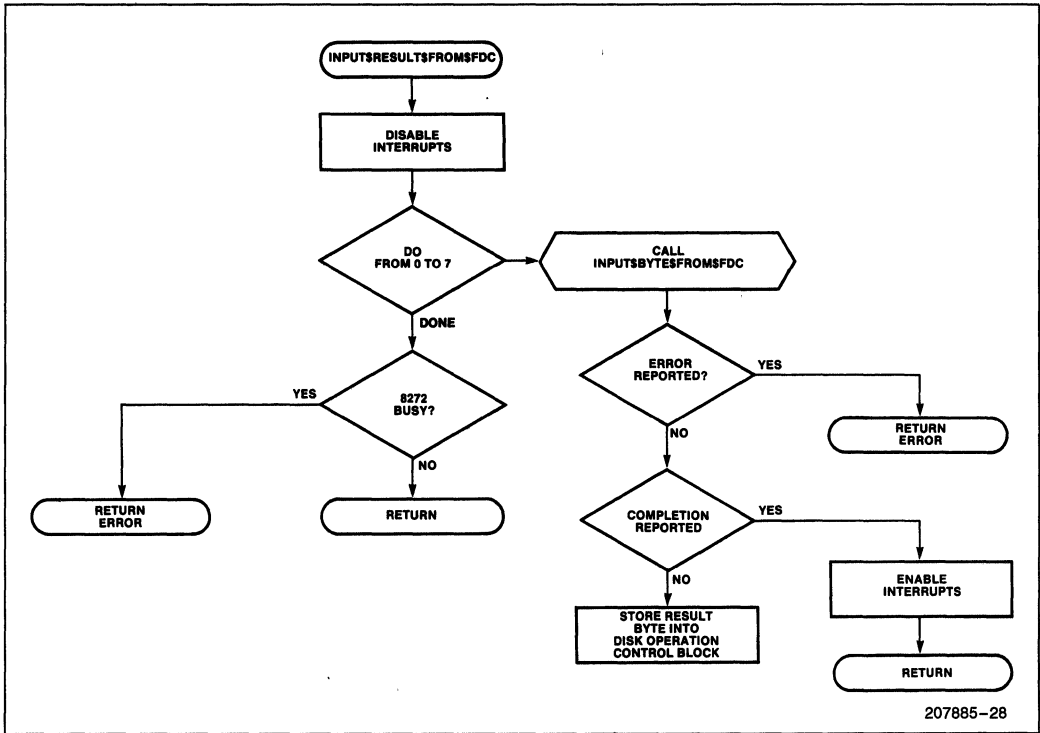
207885-22

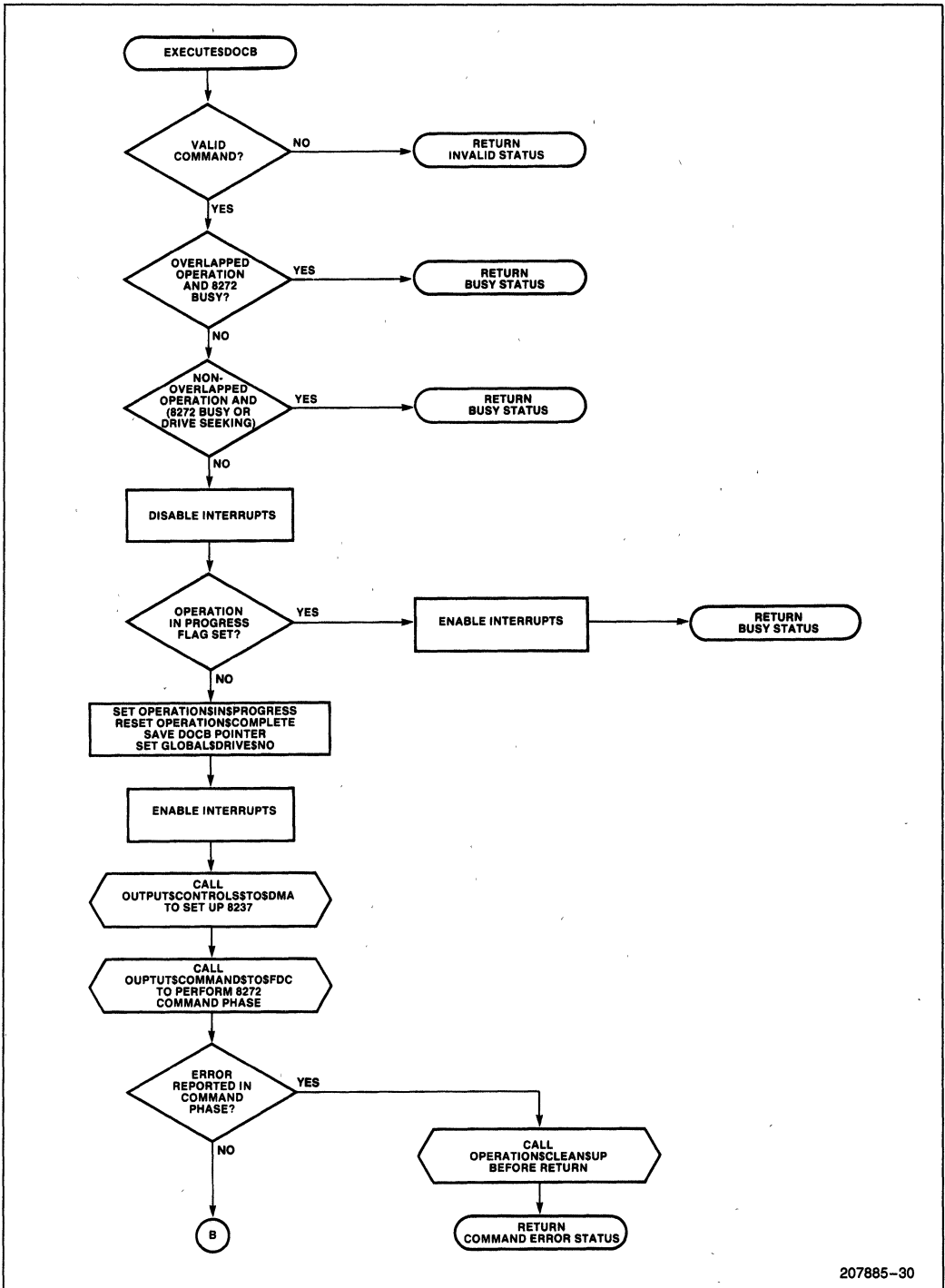


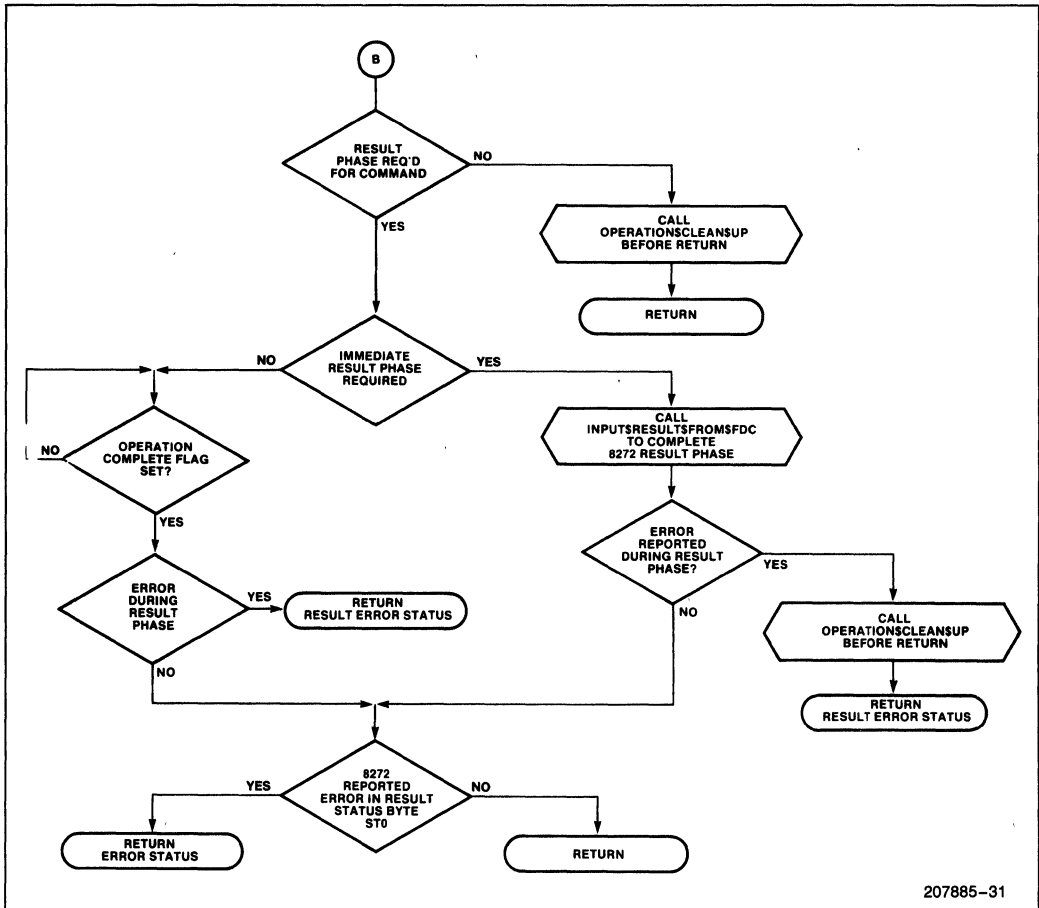
207885-23



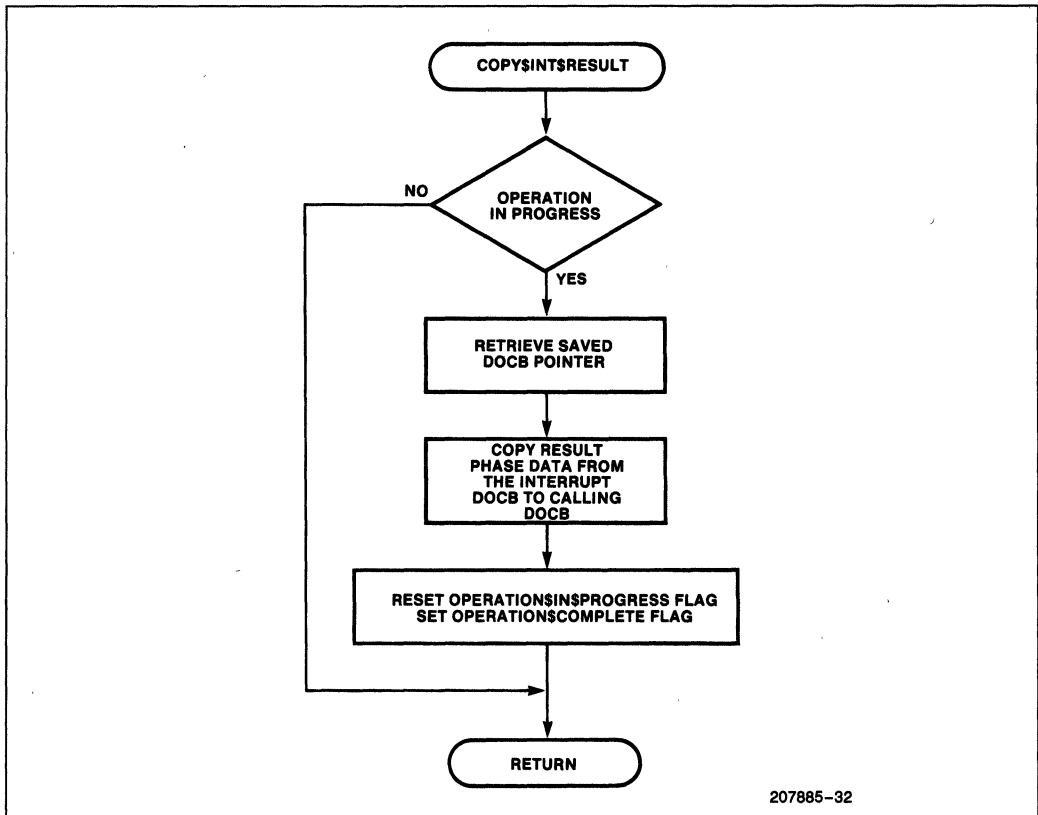




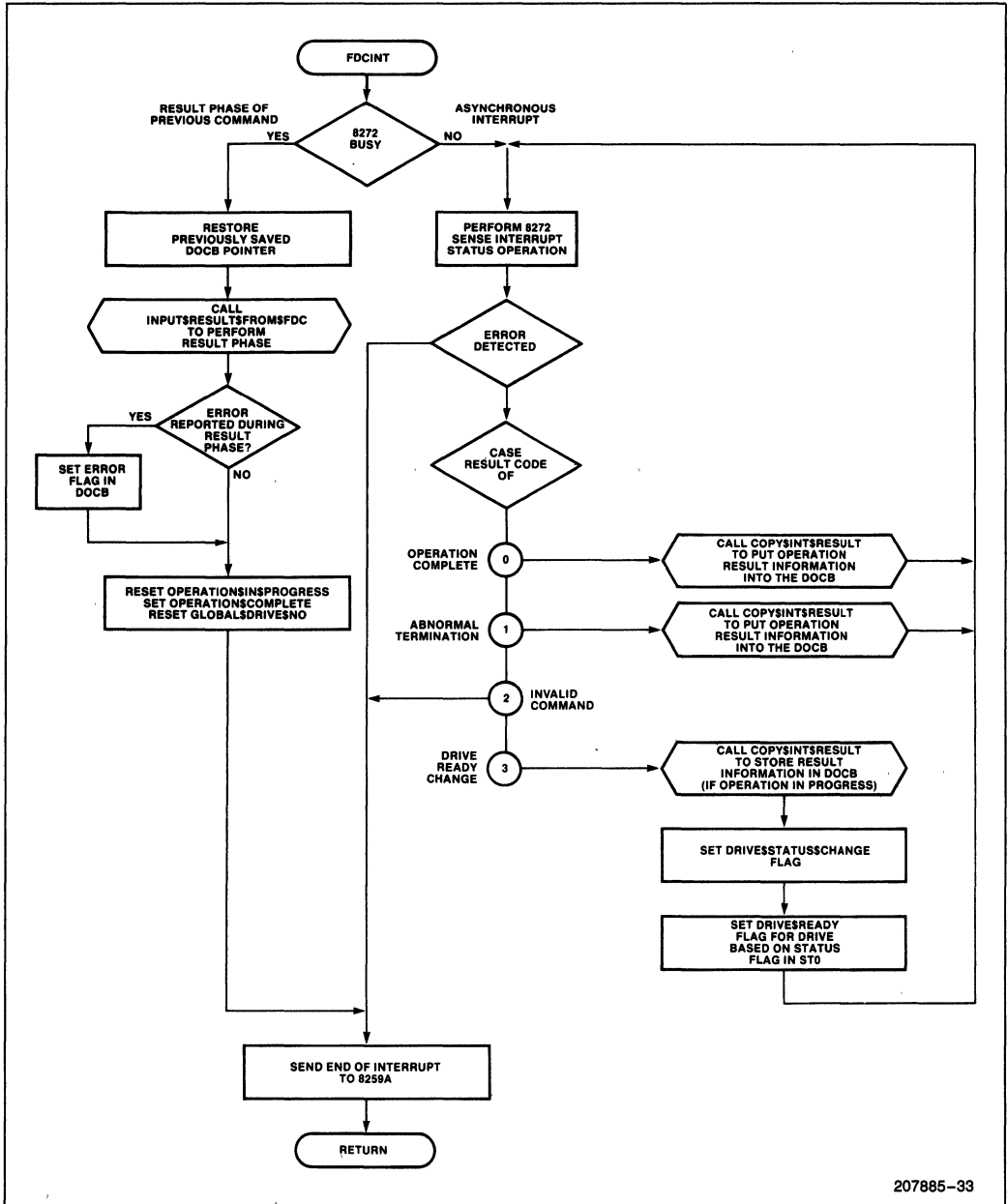




207885-31



207885-32



207885-33



**APPLICATION
NOTE**

AP-289

November 1986

**Designing with the 82072 CMOS
Single-Chip Floppy Disk Controller**

SRIDHAR BEGUR
APPLICATIONS ENGINEER

Order Number: 292022-002

CHAPTER 1

INTRODUCTION TO THE 82072 FLOPPY DISK CONTROLLER

1.1 ORGANIZATION

This Application Note provides a detailed description of the features and capabilities of the 82072 Floppy Disk Controller. Chapter 1 gives a brief overview of floppy disk concepts. The second chapter "INTERNAL ARCHITECTURE" describes the internal functions of the 82072 in block diagram form. Operation of the internal Data Separator and the Write Pre-Compensation circuits are also included. Chapter 3 presents a functional description of the 82072. Chapter 4 describes the software requirements of the 82072. Chapter 5 describes the technique of interfacing floppy disk drives to the PC/PC-XT, or the PC-AT, using the 82072 floppy disk controller. Appendix A provides detailed flow charts for the various commands, to aid software designers.

Pin functions, electrical and timing characteristics are contained in the 82072 Data Sheet.

1.2 THE 82072 FLOPPY DISK CONTROLLER

The Intel CHMOS 82072 is a fully integrated floppy disk controller designed for use in real time, on-line storage applications. The 82072 has been developed to allow designers to take full advantage of the emerging high capacity, quad density, double sided diskette drive capabilities. The 82072 performs all functions associated with data transfers between disks and user's memory.

The 82072 is a software compatible upgrade of its predecessor, the 8272A floppy disk controller. The on-chip enhancements include a self-compensating analog data separator with software selectable data rates (that require no external components), programmable write pre-compensation circuitry and write clock generation logic. These enhancements greatly reduce the number of components required to interface high capacity floppy disk drives to processors. The on-chip sixteen byte FIFO, with programmable threshold, minimizes data transfer times and overcomes the I/O bottleneck for fast disk access. Additionally, the 82072 has a power down mode which makes it ideal for low power, battery backed, portable applications.

1.3 OVERVIEW

The following sections provide the reader with a simplified overview of floppy disk controller concepts. These sections are particularly appropriate for those who are unfamiliar with floppy disk operations.

1.3.1 Data Separator

The actual signal recorded on a floppy disk is a combination of timing information and data. The serial read data from the disk must be converted into two signal streams: clock and data. The serial data must also be assembled into eight bit bytes for transfer to the host. The Data Separator generates a Data Window signal that is active during the data bit cell interval. This Data Window signal is used by the READ hardware to extract data from the serial read data stream.

The job of data separation is relatively straightforward for the single density encoding scheme. In this scheme, each bit cell contains a clock bit at the leading edge of the cell. The bit cell is 4 microseconds wide (at 125 Kbps data transfer rate). The data bit (if present) is always located at the center of the cell. The job of the data separator is to generate a data window 2 microseconds wide starting one microsecond after the clock pulse. Since every cell has a clock bit, a fixed window reference is available for every bit, and as the window is 2 microsecond wide, a slightly shifted data bit will still remain within the data window.

When MFM (Modified Frequency Modulation) encoding is used, data separation has two problems. First, MFM encoding loses the fixed cell reference pulse present in FM encoding as only some bit cells contain a clock bit. Secondly, the bit cell in MFM is one half the size of the bit cell for FM. This shorter bit cell implies that MFM cannot tolerate as large a playback data shift (as FM can) without errors.

The data separator for MFM must continuously monitor the read data stream, synchronizing its operation with the actual clock/data bits of the data stream. The data separation circuit must track the input Read Data very closely as unpredictable bit shifts leave less than 50 ns margin to decode data reliably.

1.3.1.2 PHASE LOCK LOOP (PLL)

Only an analog PLL can provide the reliability required for double density data separation. A PLL is an electronic circuit that continuously analyzes the input frequency and locks another reference oscillator to that frequency. The PLL determines the clock and data bit position by sampling each bit in the serial data stream. The phase difference between the PLL generated Data Window and the data bit is constantly fed back to adjust the position of the data window, enabling the PLL to track input data frequency changes. This allows the data separator to reliably read back previously recorded data.

The 82072 has an on-chip self-compensating analog data separator. The data separator is a fully integrated design with all passive and active components on chip.

The internal data separator consists of two analog Phase Lock Loops (PLL): a Reference PLL and a Data PLL. Analog Phase Locked Loops were chosen to provide the reliability and resolution required for double density data separation. The Reference PLL stabilizes the Data PLL. The Data PLL performs the actual data separation function by synchronizing its operation to the digital read data input from the disk drive. The synchronized signal (Data Window), generated by the Data PLL, is used to separate the clock and data bits from the read data signal. The data bits are then assembled into bytes and transferred to the system memory.

The PLL consists of three main components: a phase comparator, a second order loop filter and a Voltage Controlled Oscillator. The block diagram of the PLL is shown in Figure 1.0. Basically, the PLL compares the phase difference between the VCO output and the read data input from the disk drive. The difference in phase is used to increase or decrease the frequency of the voltage controlled oscillator in order to bring its frequency closer to that of the input. The PLL is locked when the frequency of the oscillator is exactly the same as the frequency of the read data input.

1.3.2 Data Rate

The 8272A allows the user to vary the data transfer rate by changing the frequency of the write clock. To maintain this flexibility, the 82072 provides an on-chip prescaler that divides the 24 MHz clock to rates that are equivalent to those used with the 8272A. By appropriately setting the DRATSEL bits in the Data Rate Select (DSR) register, the prescaler value can be changed to obtain the desired data transfer rate. The supported data transfer rates are 250, 300, 500 Kbps and 1 Mbps with MFM encoding, and 125, 150 and 250 Kbps with FM encoding. Most of the internal circuitry of the 82072 is driven with a 2-phase clock (PH1, PH2) that is synchronous with the chosen data rate.

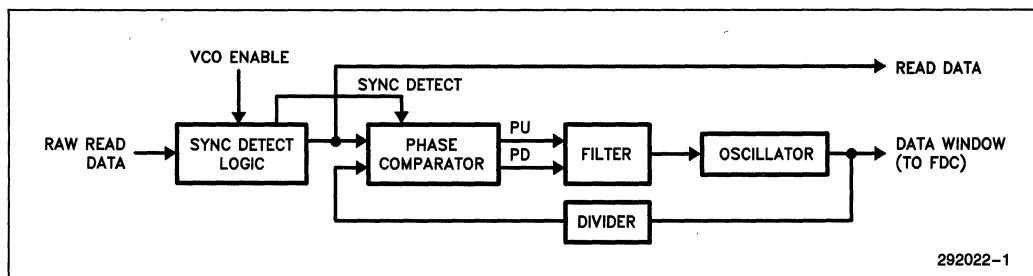


Figure 1.0. Phase Locked Loop Data Separator

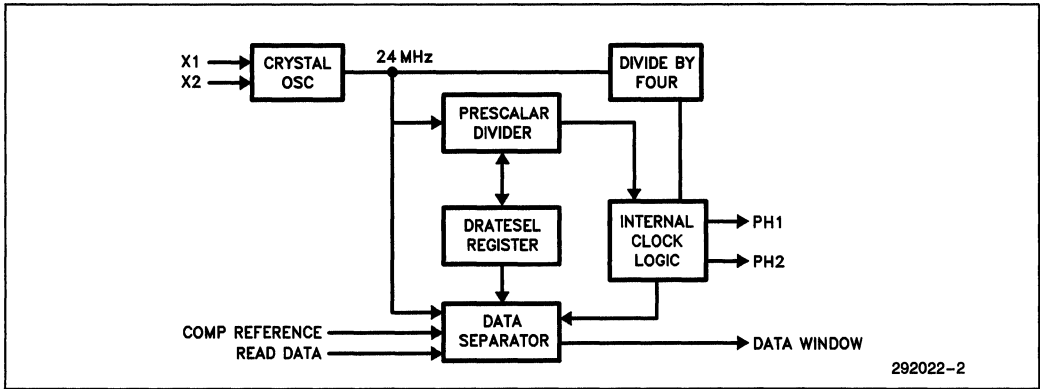


Figure 1.1. 82072 Clock Generation Logic

Table 1.1. Supported Data Transfer Rate

DRATESEL		PH1/PH2	WRCLK	
Bit 1	Bit 0		MFM	FM
1	1	8.0 MHz	1 MHz	—
0	0	4.0 MHz	500 Kbps	250 Kbps
0	1	2.4 MHz	300 Kbps	150 Kbps
1	0	2.0 MHz	250 Kbps	125 Kbps

1.3.3 Write Pre-Compensation

Data is recorded on the magnetic media by pulsing a recording head to produce a flux change on the surface of the diskette for each logic one. Lack of such a magnetic pulse indicates a logic zero. Ideally, the flux reversal induced by the Read/Write head would be instantaneous, as shown in Figure 1.1. Current would immediately switch from one polarity to the other. But in reality, it takes time for the current to reverse and the fields to decay and build up in the opposite direction. The resulting read back voltage is more or less sinusoidal with peaks less defined in time or amplitude.

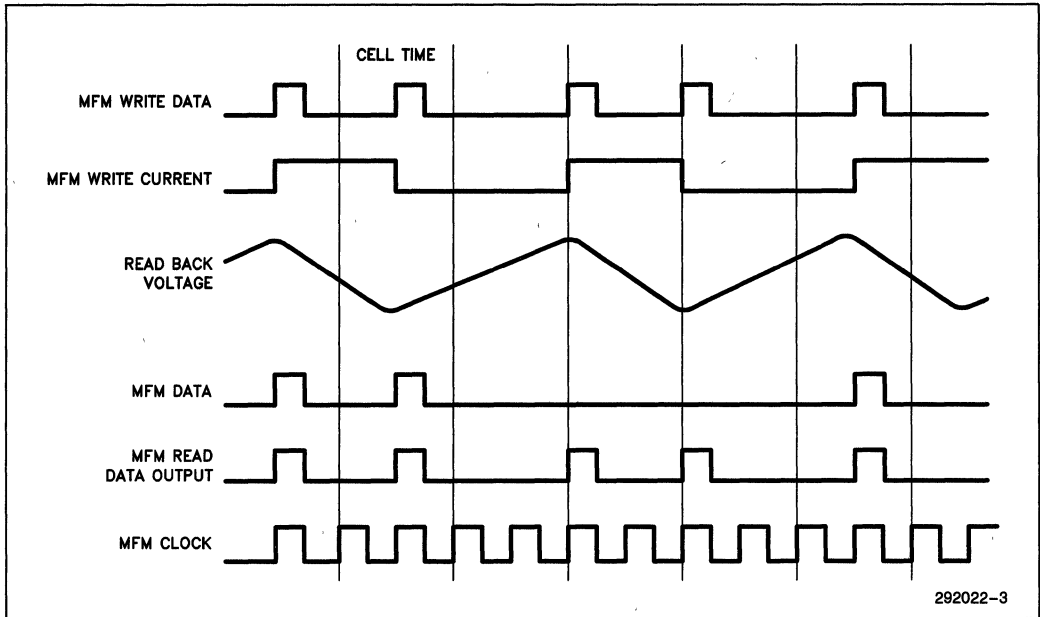
With current recording techniques, adjacent clock/data pulses are close enough to interact with each other. This phenomenon is particularly noticeable on the inner tracks. Peak shift is the result of the interaction of these pulses. Because the two pulses tend to have a portion of their individual signals superimposed on each other, the actual read back voltage is an algebraic summation of the pulses, giving rise to overlap errors.

When all "1's" or all "0's" are being recorded, the data frequency remains constant and the pulses are uniformly spaced. As a result, the positive going and the negative going errors tend to be equal and opposite and hence cancel each other. This is a "zero peak shift" condition.

Peak shift occurs when there is a change in frequency. A "011" pattern represents a frequency increase since there is a delay of about 1.5 cells between the "01" and only 1 cell spacing between the "11". As a result, the squeezing of cells cause the mathematical average (the actual read back voltage) to shift the apparent peak to the left. This is known as "early peak" shift.

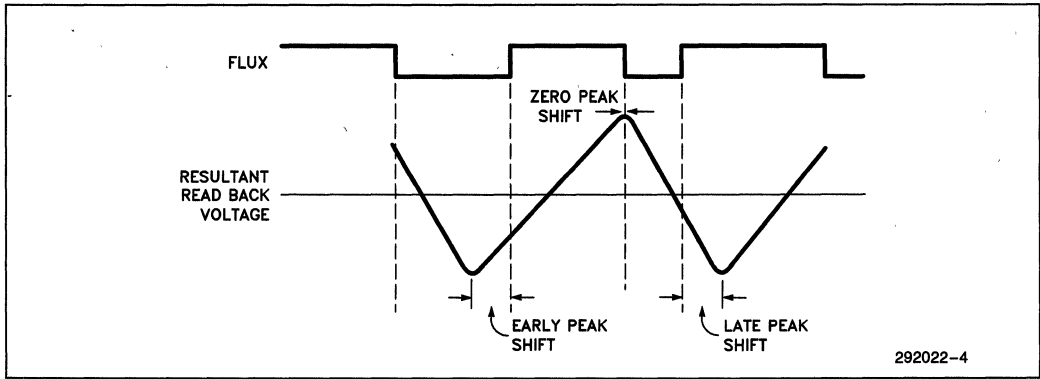
A "110" pattern represents a frequency decrease since a pulse is not written at all in the third cell. This causes the peak to be shifted to the right. This is known as "late peak shift."

The effect of peak shift is to displace the flux changes and create a timing uncertainty which can cause errors when data is read back. MFM encoding is much more susceptible to peak shift errors due to the shorter data window. The data window is defined as the time that is allowed for the data bit to appear and be recognized. The data window of the MFM encoding is 1 microsecond (at 500 Kbps) as opposed to the data window of FM encoding where the Data Window is 2 microseconds. The MFM encoding cannot tolerate as large a play-back data shift (as compared to FM encoding) without errors, due to the smaller data window.



292022-3

Figure 1.2. MFM Recording



292022-4

Figure 1.3. Peak Shift

To overcome this problem, the WRITE DATA stream is pre-compensated, i.e., the direction of each bit shift is anticipated and the bit is moved in the opposite direction before being written.

The 82072 has on-chip write pre-compensation circuitry. This circuitry pre-compensates for head and media caused peak shifts. The amount of pre-compensation applied is programmable, and the value should be chosen to compensate for the media peak shifts.

The older 8" drives required a pre-compensation value of 250 ns. The newer 5¼" and 3½" mini floppy disk drives require a pre-compensation value of 125 ns. The 82072 supports both pre-compensation values.

In order to support tomorrow's higher capacity disk drives (1 Mbps data transfer rate), the 82072 allows the user to select lower pre-compensation values. The pre-compensation values supported by the 82072 are summarized in Table 1.2.

Table 1.2. Write Pre-Compensation Values

Pre-Comp	Amount of Pre-Comp	Frequency
111	0 ns—Disabled	0
001	41.67 ns	1/24 MHz
010	83.34 ns	2/24 MHz
011	125.00 ns	3/24 MHz
100	166.67 ns	4.24 MHz
101	208.33 ns	5.24 MHz
110	250.00 ns	6.24 MHz
000	Default	

If the 82072 is programmed with the default pre-compensation value (000), a default pre-compensation value matching the PC-AT is automatically chosen based on the data rate selected. The default values corresponding to the data rate values are illustrated below in Table 1.2.1.

Table 1.2.1. Default Write Pre-Compensation Delays

Data Rate	Pre-Compensation Delay
1 Mbps	41.67 ns
500 Kbps	125 ns
300 Kbps	125 ns
250 Kbps	125 ns

1.3.4 Seek

There are three ways to position the disk Read/Write head over the desired cylinder:

1. Normal Seek
2. Implied Seek
3. Relative Seek

1. Normal Seek

This uses the SEEK command. During the execution phase of this command, the track number to seek to (NCN) is compared with the present track number (PCN). If there is a difference, the following operation is performed:

1. PCN < NCN: The DIRECTION signal to the disk drive is set to "1" (step-in) and step pulses are issued to position the head over the specified cylinder.
2. PCN ≥ NCN: The DIRECTION signal is set to "0" (step-out) and the step pulses are issued to retract the read/write head to the cylinder specified.

2. Implied Seek

This method uses the implied seek mode. When enabled, the 82072 automatically performs a seek operation before executing a read or write command.

With the 8272A, a READ operation required the following steps:

1. Issue SEEK command
2. Issue Sense Interrupt Status Command
3. Issue Read Data Command.

The new "IMPLIED SEEK" feature is enabled by issuing a "CONFIGURE" command. When enabled, only the data transfer command needs to be issued. The 82072 automatically performs the SEEK and the Sense Interrupt Status commands before executing the data transfer command.

3. Relative Seek

Standard Double Density diskettes (360 Kbytes) have 48 tracks per inch (tpi) while high capacity diskettes (1.2 Mbytes) have 96 tracks per inch. The emerging high capacity floppy disk drives can support up to 10 Mbytes and have more than 300 tracks.

The IBM System 34 format, which the 82072 follows, allows only 8 bits for the specification of track address, limiting the number of tracks that can be directly accessed to 256. To support the high capacity diskettes that have more than 256 tracks, the 82072 allows the diskette to be partitioned into pages, with each page containing 256 tracks.

To support this paging mechanism, the 82072 provides a "Relative Seek" mode. The Relative Seek command differs from the Seek Track command in that it steps the head the absolute number of tracks specified in the command, instead of making a comparison against an internal register.

The paging mechanism can be better understood by considering an example. Let us assume that a diskette has 1024 tracks. This would be divided into four pages with each page containing 256 tracks. Let us also assume that the CPU needs to access track 800 (which is contained in Page 4). If the Read/Write head is currently on track 400 (contained in Page 2); the PCN register contains $400 - 256 = 144$, then a total of $800 - 400 = 400$ 400 step pulses have to be issued, to position the head over the desired track. The maximum number of step pulses that can be issued by a single seek command is 256 pulses (as the relative cylinder number register is 8 bits wide). The internal register, PCN (which is an 8-bit register that stores the track that the R/W head is currently on), will overflow as the cylinder number crosses the track 255 and will count up from 0.

The approach would be to issue two successive Relative seek commands. The first Relative Seek is issued with the Relative Seek Number (RCN) register initialized to 255 (which will result in 256 step pulses). When this command is executed, the Read/Write head will be positioned over the absolute track 656 ($400 + 256$) (absolute track address). The PCN register will now contain $(144 + 256 - 256) 144$. The second relative seek is issued with the RCN register initialized to 143 (which will result in 144 Step pulses). This will position the Read/Write head over track 800 and the PCN register will contain $(144 + 144 - 256 = 32) 32$.

When operating within a page all commands function correctly. The user's software should keep a record of the Page that the Read/Write head is currently on, and determine the number of times the Relative Seek command is executed, to position the Read/Write head over the desired cylinder.

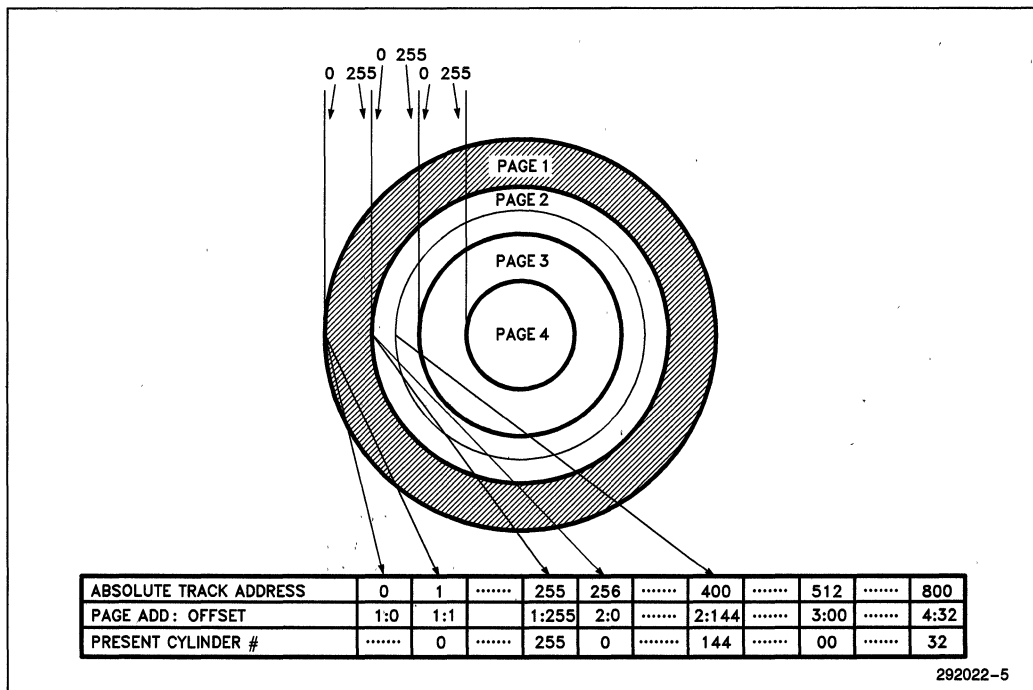


Figure 1.4. Disk Paging

1.3.5 FIFO

The microprocessor interface was enhanced by adding a 16 byte FIFO to reduce the timing constraints that most floppy disk controllers impose upon a system. The 16 Byte FIFO with programmable threshold eliminates the extremely critical timing constraints by permitting burst data transfers between the 82072 and the system bus. The point at which the 82072 generates a request for data transfers is selectable within the 16 byte range. The interface was further enhanced to support today's faster microprocessors (i.e., 10 MHz 80186, 10 MHz 80286) without incurring waitstates.

1.3.6 Power Down Mode

The 82072 power down mode allows the internal oscillator to be turned off by the user's software. Because the 82072 is fabricated in CHMOS, the current consumption occurs only when the transistors change state. In power down, the controller consumes less than 125 microamps.

Table 1.3. 82072 Feature Summary

Features	PC-AT Implementation	FDC 82072
Write Pre-Comp	External/Programmable	On-Chip/Programmable
Write Clock	External/Programmable	On-Chip/Programmable
Drive Select	External	On-Chip
Motor On	External	On-Chip
Data Separator	External/Programmable	On-Chip/Programmable
Software Reset	External/Software Contr	On-Chip/Software Contr
Power Down	Not Supported	Supported/Software Contr
Low Density Signal	External	On-Chip

Table 1.4. 82072 Supported Data Rates

		Values	PC-AT	FDC 82072
Supported Data Rates	MFM	250 Kbps	yes	yes
		360 Kbps	yes	yes
		500 Kbps	yes	yes
		1 Mbps	no	yes
	FM	125 Kbps	—	yes
		150 Kbps	—	yes
250 Kbps		—	yes	
Supported Pre-Comp Values		0 ns	no	yes
		41.67 ns	no	yes
		83.34 ns	no	yes
		125.00 ns	yes	yes
		166.67 ns	no	yes
		208.33 ns	no	yes
		250 ns	no	yes

CHAPTER 2 ARCHITECTURAL OVERVIEW

2.1 ARCHITECTURE

The 82072 provides the data interface between the floppy disk drive and the processor. The 82072 is a software compatible upgrade of the 8272A, with many on-chip enhancements. Figure 2.1 shows the block diagram of the 82072. The highlighted blocks represent areas that are completely new or highly enhanced. All the new features were adopted with the requirement of being software compatible with the 8272A. The new features of the 82072 are:

1. A crystal controlled reference oscillator
2. Pre-compensation circuits for the write operation
3. Analog Data Separator for data detection in the read mode
4. Motor enable and drive select logic
5. FIFO (Chapter 3)
6. Enhanced Processor Interface (Chapter 3)

2.1.1 Crystal Controlled Oscillator

The 82072 requires an external 24 MHz clock that can either be supplied by a 24 MHz crystal or a MOS level

square wave input. All the internal timings are derived from this clock input. It provides the reference timings for recording of data, generation of Write Clock, pre-compensation circuitry and the analog data separator.

The 82072 provides the Write Clock for synchronization of control signals and Write Data, during the write operation. To retain the flexibility of the 8272A, where the user could vary the data transfer rates by changing the frequency of the Write Clock, a prescaler was added to the Write Clock generation logic, which divides the 24 MHz clock to generate the Write Clock. By appropriately setting the data rate select bits in the DSR the prescalar value can be changed to obtain the desired Write Clock frequency.

The oscillator also serves as a reference to the read circuits. While waiting for the synchronization field, preceding the data field, the PLL is locked to a reference clock frequency (derived from the 24 MHz clock input) that is 2 or 4 times the selected data rate for MFM and FM respectively. This allows a faster acquisition time when a valid data stream arrives.

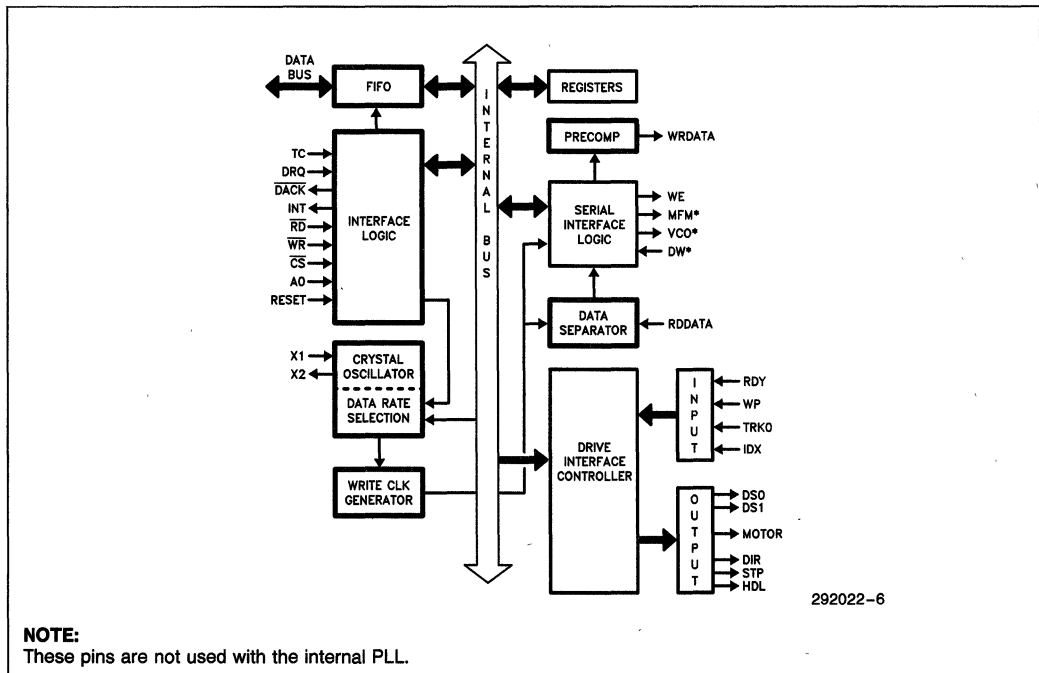


Figure 2.1. 80272 Block Diagram

2.1.2 Write Pre-Compensation

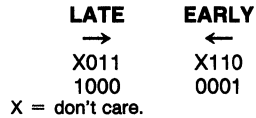
To overcome the undesirable effects of peak shifts (the cause and effect of peak shift is described in CHAPTER 1), the 82072 pre-compensates the Write Data stream before sending it to the disk drive.

The amount of pre-compensation applied depends upon the data transfer rate and the characteristics of the recording media and is typically 125 ns (for 500 Kbps). This pre-compensation is applied to data patterns that result in large peak shifts. The block diagram of the pre-compensation logic is shown in Figure 2.2. The Write Data is fed into a 13 bit serial in-parallel out shift register. The shift register is clocked at the main clock rate (24 MHz). The no-delay tap is at the center (bit 7) of the shift register. This allows six levels of early and late shifting (in increments of 41.67 ns) with respect to the no-delay tap. The first six outputs are fed into an Early Multiplexer and the last six outputs are fed into a Late Multiplexer. The multiplexers select one out of the six inputs depending upon the programming of the Data Rate Select register. A final multiplexer combines the early, late and normal signals to generate the Write Data to the disk drive. The final multiplexer is controlled, in turn, by PSO and PS1 (8272A equivalent signals), which is internally generated by the 82072.

Table 2.1. PSO, PS1 decoding

PS0	PS1	Select
0	0	Normal
0	1	Late
1	0	Early
1	1	Invalid

The circuit examines four bits and determines whether to shift or not. The following patterns are compensated (bit shifted) in the direction of the arrow.



When a flux transition pattern of "110" is written on to the disk the second "1" is pulled towards the "0". The write pre-compensation logic shifts this bit in the opposite direction by an amount equal to the expected shift.

Since the play-back shift is predictable, pre-compensation can be applied to the write data stream, such that the clock and data bits can be correctly positioned for subsequent reads. This function is completely controlled by the 82072 and is required only for MFM encoding. During write operations, the 82072 specifies an early, late or normal bit positioning depending upon the data pattern. This timing is specified with respect to the 82072 generated write clock. The track on which write pre-compensation is to be enabled is programmable.

2.1.3 82072 Data Separator

The 82072 data separator consists of two analog PLLs: a Data PLL and a Reference PLL. The Data PLL performs the actual data separation. The Reference PLL stabilizes the Data PLL. Unlike conventional PLLs which are based on a Voltage Controlled Oscillator, the Reference PLL uses an analog mono-stable multivibrator. The frequency input to the Reference PLL is a derivative of the 24 MHz crystal oscillator input. In lock, the output pulse width of the mono-stable multivibrator matches the pulse width of the input reference frequency.

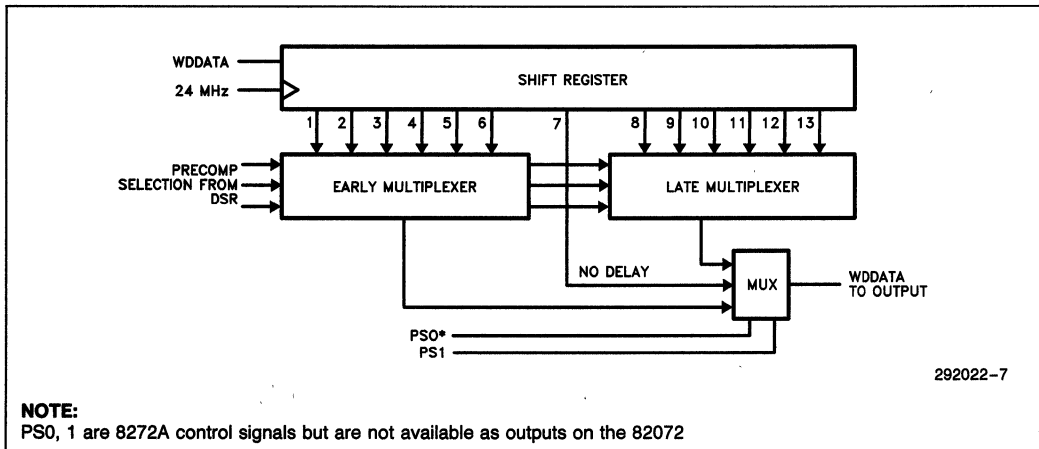


Figure 2.2. Precompensation Block Diagram

The Reference PLL controls the Data PLL's center frequency, charge pump current and the transfer function of the loop filter.

The Data PLL in conjunction with the Reference PLL provides a Data Separator that is immune to temperature, voltage, process and data rate variations. The analog data separators available today cannot provide this immunity.

2.1.3.1 FUNCTIONAL OVERVIEW

The 82072 uses two signals to select the transfer rate of the Data Separator:

1. MFM/FM sets the coding mode. This bit directly controls the reference frequency of the Data PLL.
2. HF/LF (500 Kbps/300 or 250 Kbps) which is controlled by the data rate signals. The source of the data rate signals is the DSR which is written to by the user's software.

The reference frequency to the Data loop is controlled by a combination of the Reference PLL's analog information and the Data PLL's response to the reference signal input. Table 2.2 shows the relationship between the data rates and the input reference frequency to the Data PLL.

Table 2.2 Frequency Chart

Data Rate	Ref Freq	VCO Freq	DW Freq MFM	DW Freq FM
1 Mbps	2 MHz	4 MHz	1 MHz	500 KHz
500 Kbps	1 MHz	2 MHz	500 KHz	250 KHz
300 Kbps	600 KHz	1.2 MHz	300 KHz	150 KHz
250 Kbps	500 KHz	1 MHz	250 KHz	125 KHz

During idle, while the VCO enable signal is inactive, the Data PLL is locked on to the input reference frequency. When the 82072 receives a data transfer command from the host, it internally activates the VCO enable signal (if the internal data separator is being used) and enables the sync detection logic. When a data transfer command is received, the sync detection logic samples the Read Data input for eight consecutive ones. When the eight sync pulses are detected, the Data PLL's input is switched from the reference frequency to the Read Data input. The Data PLL's VCO is halted and restarted in phase with the incoming Read Data input. The Data PLL generates the Data Window signal that is used by the read hardware to separate clock and data information.

2.1.3.2 Reference PLL Description

The primary function of the Reference PLL is to stabilize the Data PLL. The reference PLL is not directly involved in the data separation process.

The block diagram of the Reference PLL is shown in Figure 2.3. The principle elements of the Reference PLL are a phase comparator, charge pump, transconductance amplifier, analog mono-stable multivibrator and a reference signal generator. The pulse width of the mono-stable multivibrator (one shot) is controlled by a current source. The current source is, in turn, controlled by a signal called IFQ which is generated by the

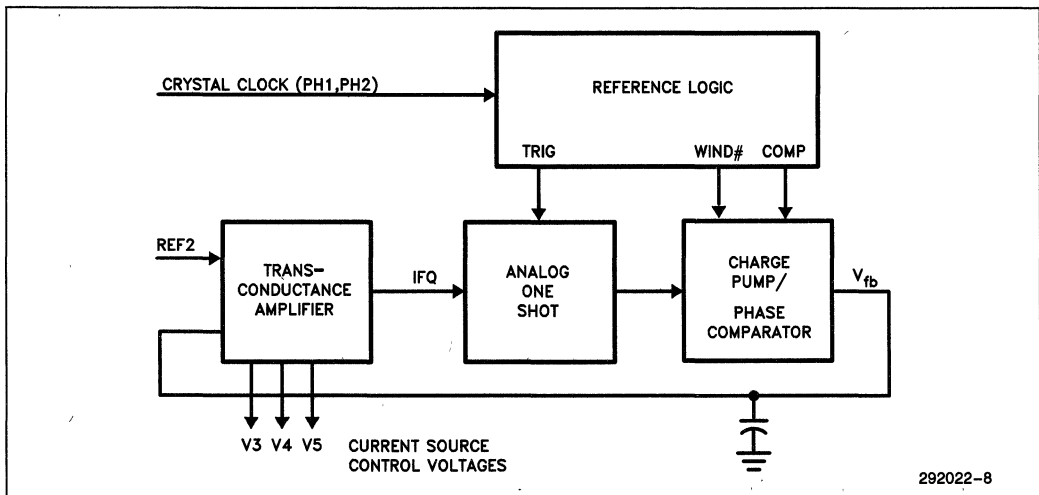


Figure 2.3. Reference Phase Locked Loop Block Diagram

transconductance amplifier. The higher the current, the shorter the pulse width. The transconductance amplifier also generates current source control voltages V3, V4, V5. The V3, V4, and V5 signals control the center frequency of the Data PLL's VCO. The levels of these signals are dependent on Vfb (the Reference PLL's feed-back voltage-Vfb). As Vfb increases the current source's current increase. The transconductance amplifier provides a voltage to current conversion with Vfb being the reference input voltage.

As mentioned earlier, the reference clock (\overline{WIND}) is a derivative of the 24 MHz crystal clock input. The circuitry that generates the reference clock provides two additional signals—TRIG and \overline{COMP} . TRIG triggers the one shot every time the reference clock makes a low to high transition. When TRIG is active, the \overline{COMP} signal disables the phase comparator of the Reference PLL.

The Reference PLL is locked onto the stable reference clock (for proper operation, the oscillator clock should be a stable signal). When the Reference PLL is locked to the reference clock frequency input, it generates the signals that control the Data PLL's center frequency. The signals generated by the Reference PLL are Vfb, V3, V4 and V5.

The Reference PLL's phase comparator compares the pulse width of the output of the one shot and the high time of the reference clock (\overline{WIND}). If the pulse width of the reference input and the output of the one shot are equal, the Reference PLL is in "Lock". If the pulse width of the reference clock is more than the width of the one shot, the phase comparator activates the pump-down signal. If the width of the one shot is greater than the high time of the reference clock, then it activates the pump-up signal (higher the current, shorter the pulse width). The charge pump charges or discharges the loop capacitor, depending upon the state of the pump-up and pump-down signals, to reduce the error.

The data separator design is based on the assumption that the signal input to the reference logic is stable, and not affected by temperature, voltage and process variations. When the Reference PLL is locked, a known relationship exists between the reference signal and the effective "RC" time constant of the one-shot. This "standard" RC is then used in the Data PLL. Figure 2.4 shows the relationship between \overline{WIND} , TRIG and the output of the one shot.

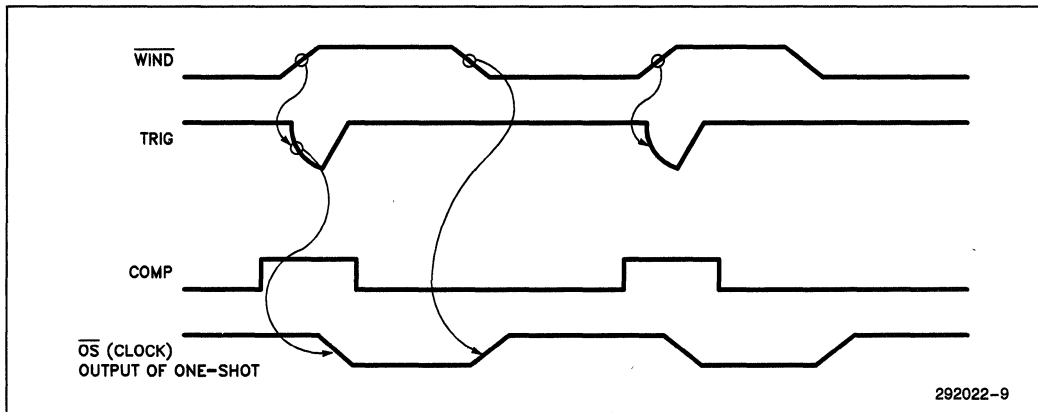


Figure 2.4. Relationship Between \overline{WIND} , TRIG, COMP

292022-9

2.1.3.3 DATA PLL CIRCUIT DESCRIPTION

The Data PLL is the section of the data separator that actually performs the data separation. The data PLL is made up of two major blocks—an analog PLL and sync detect logic. The analog PLL is a classic second order closed loop system.

The elements of the Data PLL are a phase comparator, charge pump, loop filter, transconductance amplifier, current mirror, VCO, frequency divider and some delay logic. The block diagram of the Data PLL is shown in Figure 2.5.

The VCO used in the Data PLL is very similar to the analog one shot used in the Reference PLL. The VCO normally operates at four times the data rate in the MFM mode and eight times the data rate in the FM mode. The center frequency of the VCO is controlled by the Reference loop (V_3 , V_4 , and V_5) PLL. The output of the VCO is fed into a divider (see the block diagram) which generates the Data Window signal and the V_{fbo} signal. The divider circuit divides the VCO input by four or two, depending upon whether it is FM or MFM, to generate the Data Window signal. The V_{fbo} signal is always twice the frequency of the Data Window signal.

The comparator used in the Data PLL, operates either as a phase comparator or a frequency comparator, de-

pending upon the input. If the input to the comparator is the reference frequency, then the frequency comparator mode is selected. The phase comparator mode is selected when reading data. The sync detector logic controls the comparator's mode of operation. The sync detector activates the phase comparator mode when it recognizes the valid sync pulses.

The rising edge of the Read Data input enables the circuit for phase comparison. The phase comparator (reading data) compares the phase difference between the delayed read data and the V_{fbo} . If the rising edge of delayed read data occurs first, the pump-up signal is activated and the VCO frequency is increased. If the V_{fbo} edge occurs before the Delayed read data, then the pump-down signal is activated and the VCO frequency is decreased. When a low to high transition of both the delayed read data and the V_{fbo} occurs at the same time, the frequency of the VCO is maintained.

The charge pump controls the current flow (in or out) of the loop filter. When the pump-up (PU) signal is active, the current source charges the loop capacitor. When pump-down (PD) is active, the current source discharges the loop capacitor. When both the PU and PD are inactive, the current source is turned off. The current source and the charge pump is controlled by the Reference PLL. The current changes as a function of the data rate, to maintain the loop gain.

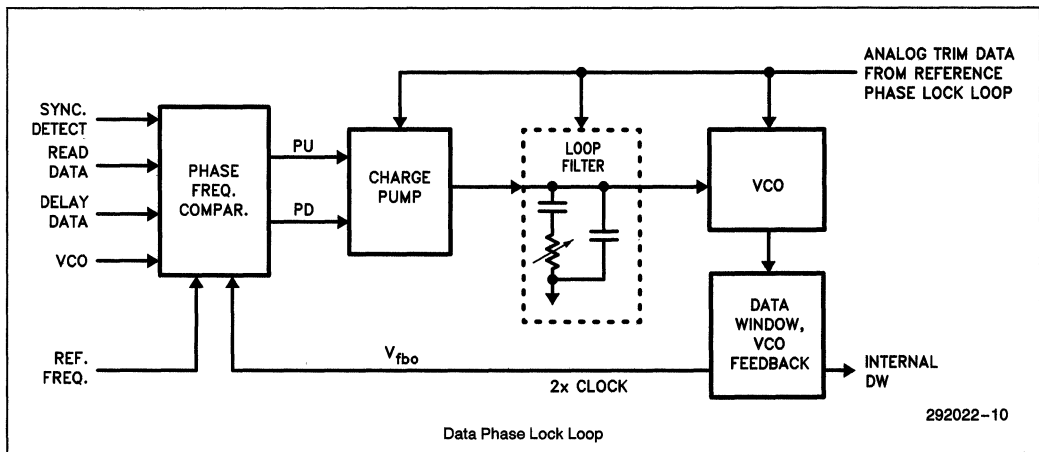


Figure 2.5. Data PLL

During the idle state, the Data PLL is locked to the reference frequency. When the sync detect logic detects eight valid sync bits, the data loop input is switched to the incoming data stream and the VCO is halted and restarted in phase with the read data signal. This reduces the time required to achieve lock.

The VCO requires 5 microseconds (48 bit times) to achieve the final lock. After processing the read data, the Data PLL switches back to the reference input frequency.

2.1.4 Drive Select and Motor Enable Circuits

Figure 2.6 shows the circuitry required for generating the drive select and motor enable signals. The 74LS156 is used to decode the DS0, DS1 and the MOTOR signals from the 82072 to generate four drive select and motor enable signals. The 74LS156 can source approximately 30 milliamps, which is sufficient to support small cable assemblies (less than 2 feet). If longer cable lengths have to be supported, then open collector drivers of the type 74LS38 have to be used instead.

The FDC's MOTOR logic provides a programmable motor enable signal. If a drive is not currently selected or the MOTOR signal is not on, the drive select and the MOTOR signals are activated the programmed time before the drive is accessed.

2.1.5 Drive Status Polling

Following hardware reset, the 82072 defaults to polling disabled mode. When disabled, the 82072 does not check for changing READY status except on drives currently involved in some disk operation. The polling mode may be enabled by issuing a CONFIGURE command (see Chapter 4). When polling is enabled the disk controller polls the drives periodically while in the idle state. The idle state is after the last byte in the result phase has been read and before the first byte for the command phase has been written. The disk controller selects each drive and checks the ready signal. If the drive has changed its ready signal, it generates an interrupt. The host must respond to this interrupt by executing a Sense Interrupt Status command.

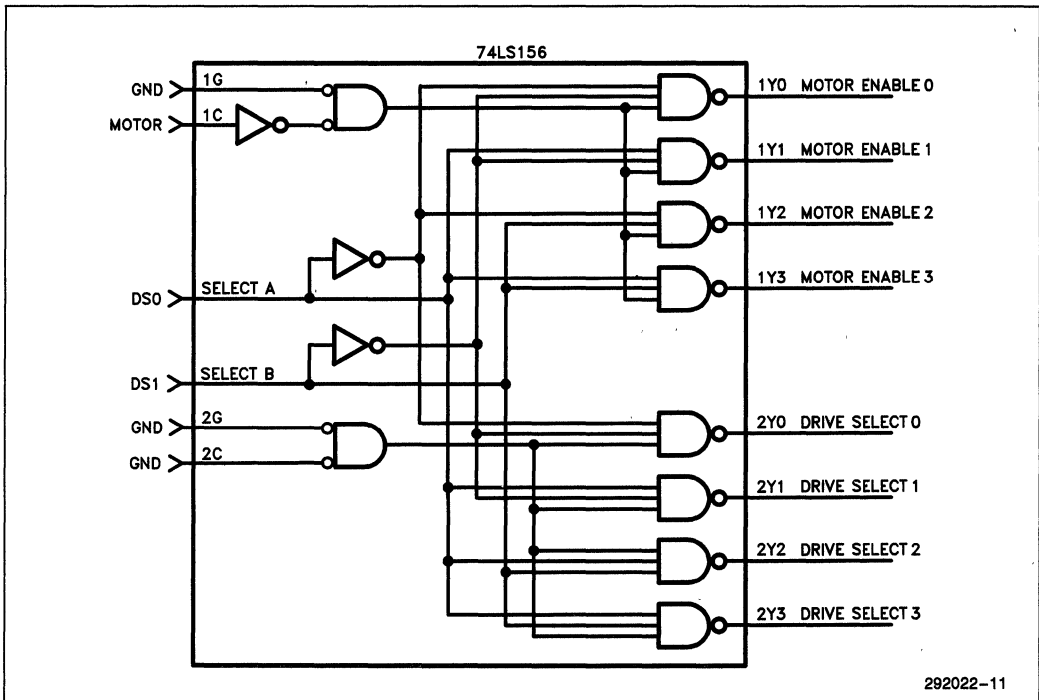


Figure 2.6. Drive Select and Motor Enable Circuitry

CHAPTER 3 FUNCTIONAL DESCRIPTION

The CHMOS III 82072's bus interface has been optimized to interface to the 10 MHz 80186 and the 10 MHz 80286 microprocessors without waitstates. Due to the conventional nature of the hardware signals between the 82072 and the processor, it can easily interface to other microprocessors. This chapter describes the processor interface and the data transfer mechanism between the processor and the disk drive. The emphasis is not on any particular command, but on the data transfer process.

3.1 PROCESSOR INTERFACE

The 82072 system interface consists of a Parallel Interface Unit (PIU) and several registers. The PIU has an 8 bit bi-directional data bus and handles all of the data transfers to/from the system bus. Handshaking signals are provided by the 82072 which makes it easy to interface to the DMA controller. The 82072 contains three registers:

1. Main Register (MSR)
2. Data Rate Select Register (DSR)
3. FIFO

1. Main Status Register (MSR)

The Main Status Register is an 8 bit read only register. This register may be accessed at any time. It provides the system processor with the status of the disk drives, the status of the 82072 and the status of the processor interface.

2. Data Rate Select Register (DSR)

The Data Rate Select register is a write only register. It allows users to specify the data transfer rate between the controller and the disk drive. The user can select between internal and external Data Separator, the data transfer rate and the write precompensation delays.

When the Data Rate Select register is accessed by the processor, data is loaded into it after an internal synchronization delay. The processor should not try to perform successive writes to the Data Rate Select register until the synchronization time has elapsed (24 clock periods at 24 MHz or 1 microsecond). The DSR defaults to a 250 Kbps data transfer rate with 125 ns of write precompensation delay. If this value does not meet the user's needs, the DSR should be initialized before executing data transfer commands. Following the initialization of the DSR, the internal PLL requires 2 milliseconds to stabilize at the new frequency (if you

are selecting a new data transfer rate). During this stabilization delay, only head positioning commands should be issued. This register should not be altered during the execution phase of data transfer commands, otherwise data errors would occur. The contents of the DSR register are not affected by software reset. The functions of the various bits in the DSR register are explained below:

SWR	PD	EPLL	PRE-COMP			DRATESEL	
7	6	5	4	3	2	1	0

SWR: (Software Reset) When this bit is set to '1', it enables the software reset of the 82072. Software reset is similar to the Hardware reset, but does not affect the contents of the DSR. When enabled, the 82072 internally holds the reset active for 12 to 15 clock cycles (~4 microseconds) depending upon the state of the internal state machine. The processor must wait for the "Request for Master" (RQM bit in the Main status Register) bit to be set before issuing any commands. During the initialization period following a software reset, the 82072 does not accept commands from the processor.

A command may be aborted by setting the software reset bit. This ensures that the host will get control of the 82072 even if the disk system hangs up in an abnormal fashion. After a software reset, the SPECIFY command has to be re-issued as the software reset clears the parameters set by this command. The SPECIFY command initializes the Step Rate Time, the Head Load Time and the Head Unload Time.

PD: (Power Down) When the Data Rate Select register is programmed with the "PD" bit set to a 1, the 82072 goes into its power down mode. A software reset sequence is performed by the 82072, before entering the power down state. During powerdown, the 82072 shuts off the clock oscillator. The design should ensure that all input signals are held in a valid high or low state. Only write operations to the Data Rate Select Register are allowed during power down.

EPLL: (Enable) When set to a zero, this bit enables the internal PLL data separator. When set (the internal PLL is enabled following reset), the external Data Window signal is ignored. It internally generates the data window signal

that is used to extract data bits from the serial clock and data bit stream. If this bit is set to 1, external data separation circuitry is required and should provide the Data Window input.

DRATSEL: Data Rate Select programs both the read (Data Rate Select) and write data rates. For single density (FM) mode, the data rates are one half the values stated for double density (MFM) mode.

Table 3.1. Supported Data Rates

DRATSEL	DATA RATE	
	MFM	FM
11	1 Mbps	ND*
00	500 Kbps	250 Kbps
01	300 Kbps	150 Kbps
10	250 Kbps	125 Kbps

* Not defined

PRECOMP: The 82072 has write pre-compensation circuitry that internally adjusts the write data pulses before sending it to the disk drive.

The 82072 defaults to PC-AT compatible pre-compensation values when the DSR register is programmed with PRE-COMP bits (bits 2–4) set to zeroes. The pre-compensation value chosen depends upon the programmed data transfer rate (bits 0–1). The default pre-compensation values are illustrated in Table 3.1.1 below.

Table 3.1.1. Default Write Pre-Compensation Delays

Data Rate	Pre-Compensation Delay
1 Mbps	41.67 ns
500 Kbps	125 ns
300 Kbps	125 ns
250 Kbps	125 ns

Upon hardware reset, the contents of the Data Rate Select Register defaults to the values shown below:

Table 3.2 DSR Default Values

SWR	PD	EPLL	PRE-COMP			DRATSEL	
			4	3	2	1	0
0	0	0	0	0	0	1	0

Default Values:

- 250 Kbps
- Internal PLL
- Pre-comp 125 ns

3. FIFO

The 82072 has a sixteen byte FIFO with a programmable threshold which greatly improves the data transfer mechanism while providing buffering between the serial channel and the system bus. During the command phase, the FIFO is disabled to retain compatibility with the 8272A and to provide proper handling of the “invalid command” condition. A command/parameter byte must be sent after polling the Main Status register to ensure that the controller is ready to accept a byte of data. During the execution phase, the FIFO improves the data transfer rate in two ways. First, the DMA requests to the processor are minimized by fine tuning the FIFO threshold (threshold can be set with the CONFIGURE command. See Chapter 4 for details on the CONFIGURE command) to match the system bus latencies. Second, the host side of the chip does not have to wait for serial side which transfers data at a much slower rate.

The Main Status register, the Data Rate Select register (DSR) and the FIFO can be selected with different combinations of the RD, WR and A0 pins as shown below.

Table 3.3. 82072 Register Set

A0	RD	WR	Function
0	0	0	Illegal
0	0	1	Read Main Status Register
0	1	0	Write to Data Rate Select Register
0	1	1	Data Bus is Tri-stated
1	0	0	Illegal
1	0	1	Read Data from FIFO
1	1	0	Write Data to FIFO
1	1	1	Data Bus is Tri-stated

3.2 PRINCIPLE OF OPERATION

Each command is initiated by a multi-byte transfer from the processor to the 82072 (the transferred bytes contain command and parameter information). After completion of the command specification, the 82072 automatically executes the command. The results after the command execution is also a multi-byte transfer from the controller to the processor. Because of the multi-byte interchange of information between the processor and the controller, it is convenient to consider the command execution as having three phases:

1. **Command Phase:** The CPU transfers all the information required to perform the command. The 82072 automatically enters the command phase after Reset or following the completion of the result phase of the previous command.

2. Execution Phase: The 82072 executes the command. It enters the execution phase immediately after receiving the last command parameter. The execution phase normally ends after the last byte of data is transferred or if an error occurs.
3. Result Phase: Upon completion of the command, status information is made available to the host processor. After the host reads all the result bytes, the 82072 re-enters the command phase and is ready to accept the next command.

3.2.1. Command Phase

The 82072 is in the command phase after all of the result bytes of the previous command have been read or following RESET. Each command is initiated by a multi-byte transfer from the processor to the 82072. The command and parameter bytes have to be sent to the 82072 in the order shown in the COMMAND SET (82072 Data Sheet). That is, the command code must be sent first and the other parameter bytes must be sent in the prescribed sequence.

During the command phase, bytes are transferred to the 82072 using the Main Status Register to control the direction and timing. Bit 6 (DIO) of the MSR should be cleared and Bit 7 (RQM) should be set before a byte can be written to the FIFO. Many of the commands require multiple bytes, and as a result, the MSR must be checked prior to each byte transfer.

The RQM bit (the MSB in the MSR) goes low following the receipt of the first command byte. The duration that RQM is low, is dependent upon the command being executed. The 82072 holds RQM low and internally sets up the operation to be performed. It sets RQM high when it is ready to accept the next parameter byte. After the last parameter byte is received the 82072 automatically enters the next phase as defined by the command definition.

3.2.2 Execution Phase

All data transfers occur during the execution phase. The data transfer can be performed either in the DMA or the non-DMA mode, as specified in the SPECIFY command.

1. NON-DMA MODE, Transfers from the FIFO to memory:

When programmed to operate in Non-DMA mode, the 82072 activates the INT pin and sets the RQM bit in

the main status register, when the FIFO contains the (16 - threshold) number of bytes or after the last bytes of a full sector transfer has been placed in the FIFO. The MSR should be read to verify that the interrupt is for the data transfer. Bits 5 and 7 of the MSR will be set. The INT pin is used by interrupt driven systems, and the "RQM" bit is used by polled systems. The processor must respond to the interrupt and read the data bytes from the FIFO. The 82072 deactivates the INT pin and RQM bit when the FIFO is empty. This process is repeated until the last byte is transferred out of the FIFO.

2. NON-DMA MODE, Transfers from memory to FIFO:

At the start of the data transfer, the INT pin and the RQM bit in the Main Status register are activated sixteen-byte times before the 82072 enters the execution phase (if the selected data transfer rate is 500 Kbps, then the time from the generation of interrupt to the time the first byte is to be written to the FIFO to prevent underrun errors, is $(16 \times 16 - 1.5) = 254.5$ microseconds. 1.5 microseconds is the time required for serial to parallel conversion). This allows a greater tolerance to interrupt service latency. The CPU must respond to the interrupt by writing data bytes to the FIFO. The serial unit empties the FIFO at the serial data rate. If the interrupt service latency is greater than the (serial data rate X FIFO threshold), an underrun will occur. The Interrupt pin stays active until the FIFO becomes full. The INT pin and the RQM bit of the Main Status register are set again when the FIFO has (Threshold) number of bytes remaining in the FIFO. This process is repeated until the last byte is transferred to the FIFO, as indicated by the activation terminal count. The interrupt pin is deactivated upon receiving terminal count. The 82072 enters the result phase when the serial unit has emptied the last byte from the FIFO.

3. DMA MODE, Transfer from FIFO to memory:

When programmed in DMA mode, the 82072 activates the DRQ pin (DMA Request) when the number of bytes in FIFO equals (16 - threshold) number of bytes, or the last bytes of a sector has been placed in the FIFO. The system DMA has to respond to the request by reading data from the FIFO. The 82072 deactivates the DRQ pin, when the FIFO is emptied by the DMA controller.

4. DMA MODE, Transfers from memory to FIFO:

The 82072 activates the DRQ pin upon entering the execution phase of the data transfer command. The DMA controller must respond to this DMA request by activating its DACK and WR signals, and placing data

in the FIFO. DRQ remains active till the FIFO becomes full. DRQ is activated again when the FIFO contains (16-FIFO threshold) number of bytes. The 82072 also deactivates the DRQ pin when the TC pin is activated, indicating that no more data transfers are required by the command. The 82072 enters the result phase after the last byte has been transferred from the FIFO.

In either mode of operation (DMA or the Non-DMA mode), the execution phase ends when a terminal count signal is sensed, or when the last track has been read or written. In addition, if the disk drive is in a not-ready state at the beginning of the execution phase, the not ready flag is set in the status register 3 and the command is terminated. If a fault signal is received at the end of the execution phase, the FDC sets the equipment check flag in the status register 0, and terminates the command.

3.2.3 Result Phase

Following the completion of the disk operation, status information is made available to the processor. The result phase is indicated by the generation of an interrupt. All of the result bytes have to be read to properly terminate a command.

Bit 6 (DIO) and 7 (RQM) must be set before a byte can be read from the FIFO. The RQM and DIO bits will remain set until the last result byte has been read by the processor. When the last byte of data is read in the result phase, the command is automatically ended and the 82072 is ready to accept a new command.

3.3 SETTING THE FIFO THRESHOLD

The transaction involved in bus acquisition and release imply overhead resulting in losing system clocks due to signal propagation delays and arbitration time requirements. For many short DMA bursts, up to 5 clocks are lost due to this switching. On the other hand, long burst transfers imply that other masters will experience long delays in bus acquisition. An optimal FIFO threshold must be selected that improves system performance and, at the same time, ensures that OVER-RUN and UNDERRUN errors are avoided.

A. OVERRUN ERROR: An overrun error occurs if the FDC is not serviced before the FIFO overflows. This error occurs only during read data transfers. The timing requirements for preventing overrun errors can be understood by considering an example.

Let us assume that the FIFO threshold has been set to 10. During the execution phase of a read data command, the DRQ line (when programmed in the

DMA mode) is active when the FIFO contains (16 - threshold) = 6 bytes. The DMA controller must respond by reading data from the FIFO before the serial unit adds 10 additional bytes to the FIFO. If the DMA response latency is greater than the time taken by the serial unit to assemble the 10 additional bytes, then an overflow problem occurs.

To prevent overrun error, the DMA latency time must be less than

$$\text{Threshold} \times \text{the serial data transfer rate.}$$

At 500 Kbps (High Capacity Quad Density Drives) data transfer rate:

$$\begin{aligned} &= 10 \times 16 \mu\text{s} = 160 \mu\text{s} \\ &= 1.28 \times 10^3 \text{ bus cycles at 8 MHz.} \end{aligned}$$

B. UNDERRUN ERRORS: An underrun error occurs if the FDC is not serviced before the FIFO is completely emptied by the serial unit. This error is encountered only during write data transfers. At the start of write data transfers, during the execution phase, the FDC 82072 activates the DRQ (DMA request) line sixteen byte times before it actually requires a byte of data. This gives ample time for the DMA controller to respond to the DMA request and place data bytes into the FIFO.

The underrun error timing requirements can be understood by extrapolating the overflow example given above.

Let us again assume that the FIFO threshold has been set to 10. During the execution phase of write data transfers, the DRQ line is activated when the FIFO contains less than the Threshold number of bytes. The DMA controller must respond to the DMA request and write data bytes to the FIFO before the serial unit empties the FIFO.

To prevent underrun errors, the DMA response latency must be less than

$$\text{Threshold} \times \text{serial data transfer rate.}$$

At 500 Kbps (High Capacity Quad Density Drives) data transfer rate:

$$\begin{aligned} &= 10 \times 16 \mu\text{s} = 160 \mu\text{s} \\ &= 1.28 \times 10^3 \text{ bus cycles at 8 MHz.} \end{aligned}$$

The 82072's CONFIGURE command provide a means for choosing the optimal setting required by the application.

If the FIFO is set too low, there are longer periods of time between DMA requests, but the DMA controller must be very responsive to DMA requests. This is

the desired operating mode when interfacing to a "fast" system, and typically implies the FDC must be given a high DMA priority.

A high threshold value is used with a "slow" system. This allows for a long latency period after a DMA request, but results in more DMA service requests. The optimal setting should be a compromise between the number of times the bus is accessed, and ensuring that there is no overflow/underflow problem. Typically, a low threshold value is chosen when the 82072 is given a lower DMA priority.

The parameters that determine the optimal threshold value are the parallel bus frequency (F_p), serial data transfer rate (F_s), and the bus latency time (in the range between N_{Amax} and N_{Amin}). The ratio of F_p/F_s determines the relationship between how fast the FIFO is filled and how fast it can be emptied. The bus latency time determines how long the 82072 has to wait before being serviced. If F_s is the serial data rate, then $F_s/8$ is the serial data rate in bytes per second. The data rate of the system depends on the system clock frequency and the number of clock cycles required to transfer one byte of data (n).

In order for the parallel side to keep up with the serial side, the following must hold:

$$(F_p/n)/(F_s/8) > 1$$

This equation assumes that the parallel interface section will read/write data at a faster rate when compared to the serial section.

To arrive at the optimal threshold value, the designer must take into account that it takes a number of clock cycles to acquire the system bus (defining the range to be between N_{Amin} and N_{Amax} , where N_{Amin} is the minimum time required to acquire the system bus and N_{Amax} is the maximum time). To prevent underrun or overrun, the maximum bus acquisition time must be less than the time to fill the FIFO from the threshold limit (for write operation, the maximum bus acquisition should be less than the time to empty the FIFO from the threshold limit).

$$(N_{Amax}/F_p) < (\text{The number of bytes from FIFO threshold to Full (for read)})/(F_s/8)$$

$$(N_{Amax}/F_p) < (LIMIT)/(F_s)/8;$$

where $LIMIT =$ FIFO threshold for read/write data transfers.

$$8(N_{Amax}/F_p)/F_s < LIMIT$$

$$LIMIT > F_s(N_{Amax})/8F_p.$$

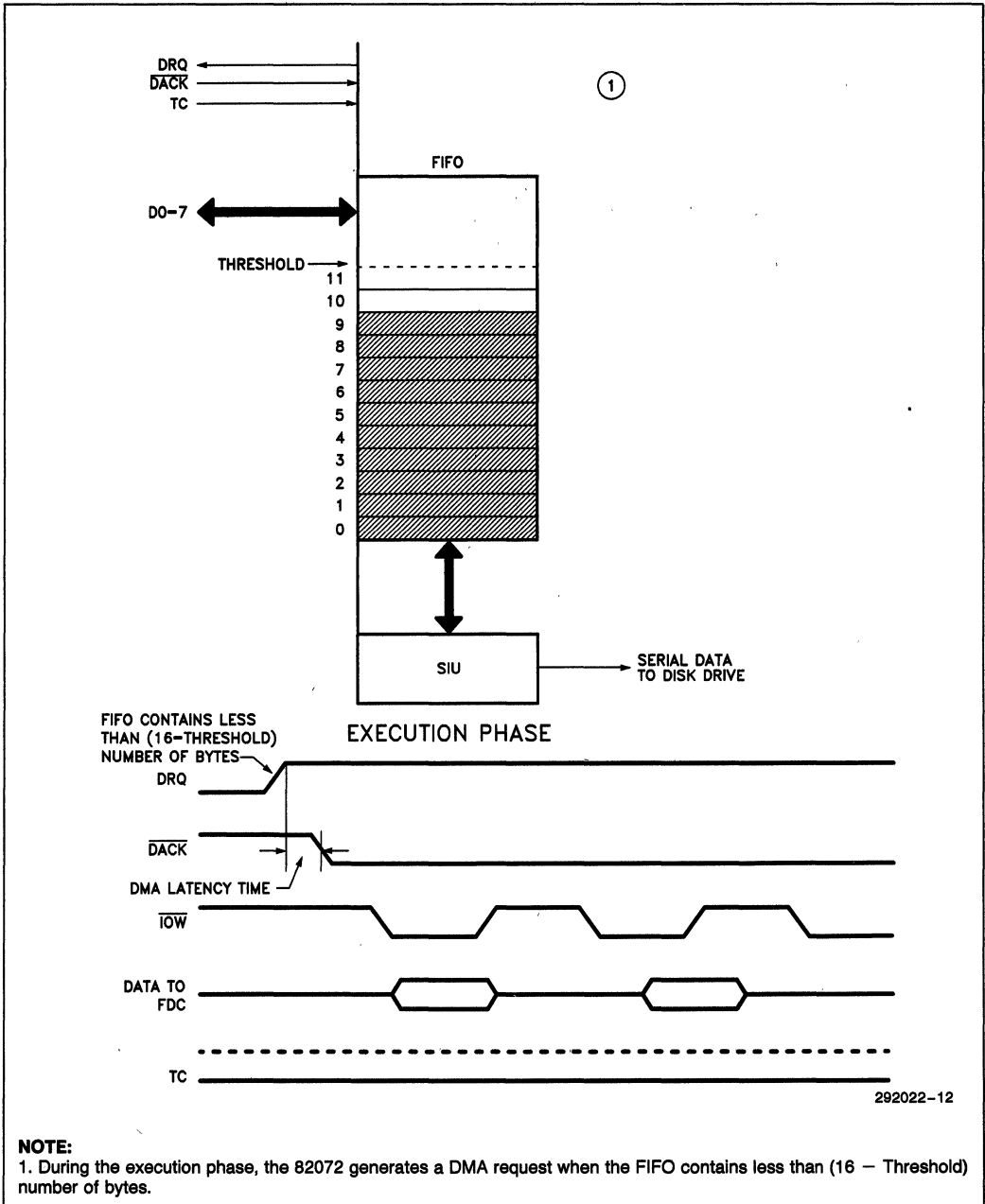


Figure 3.1. FIFO Operation During a Write Command

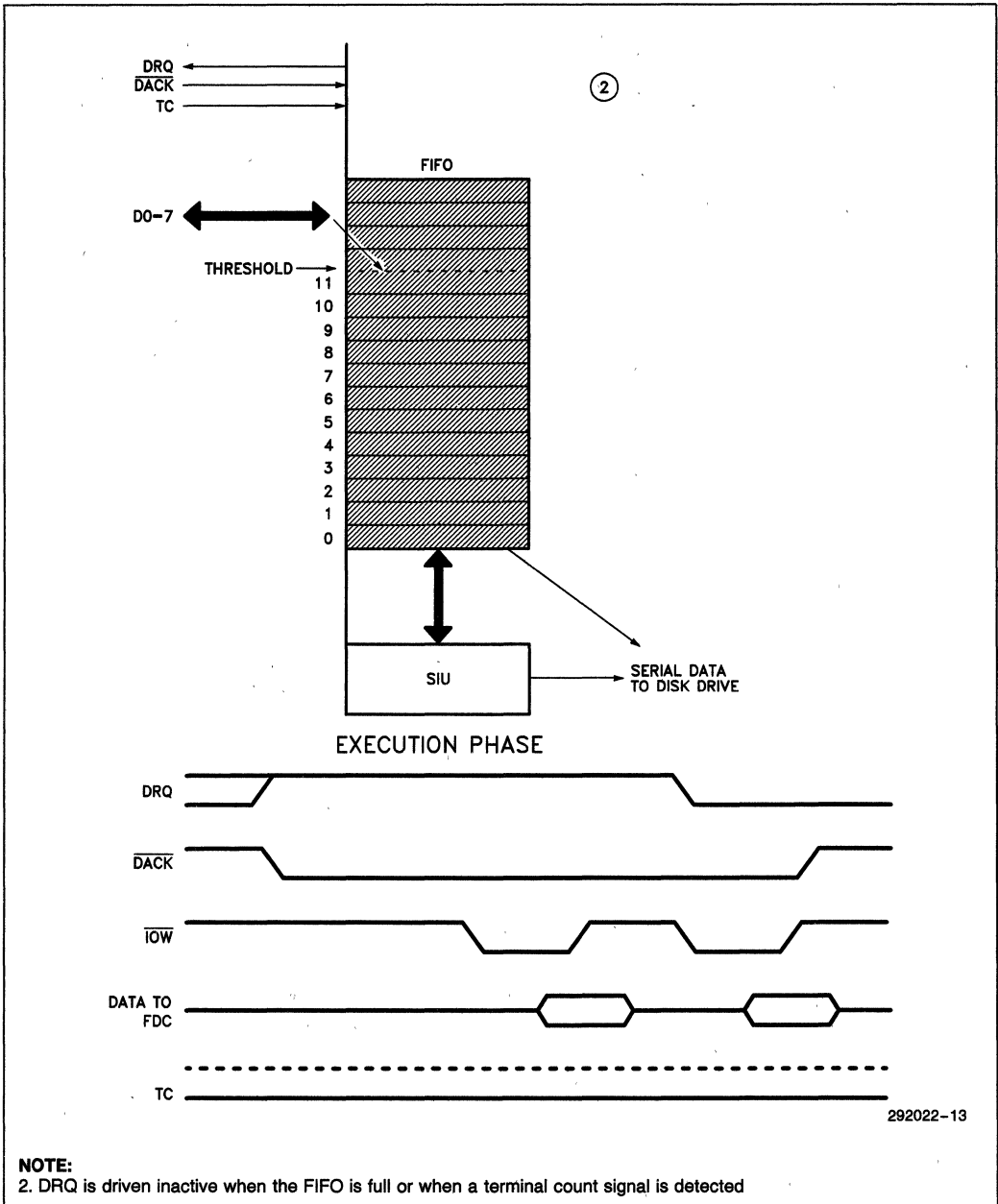


Figure 3.1. FIFO Operation During a Write Command (Continued)

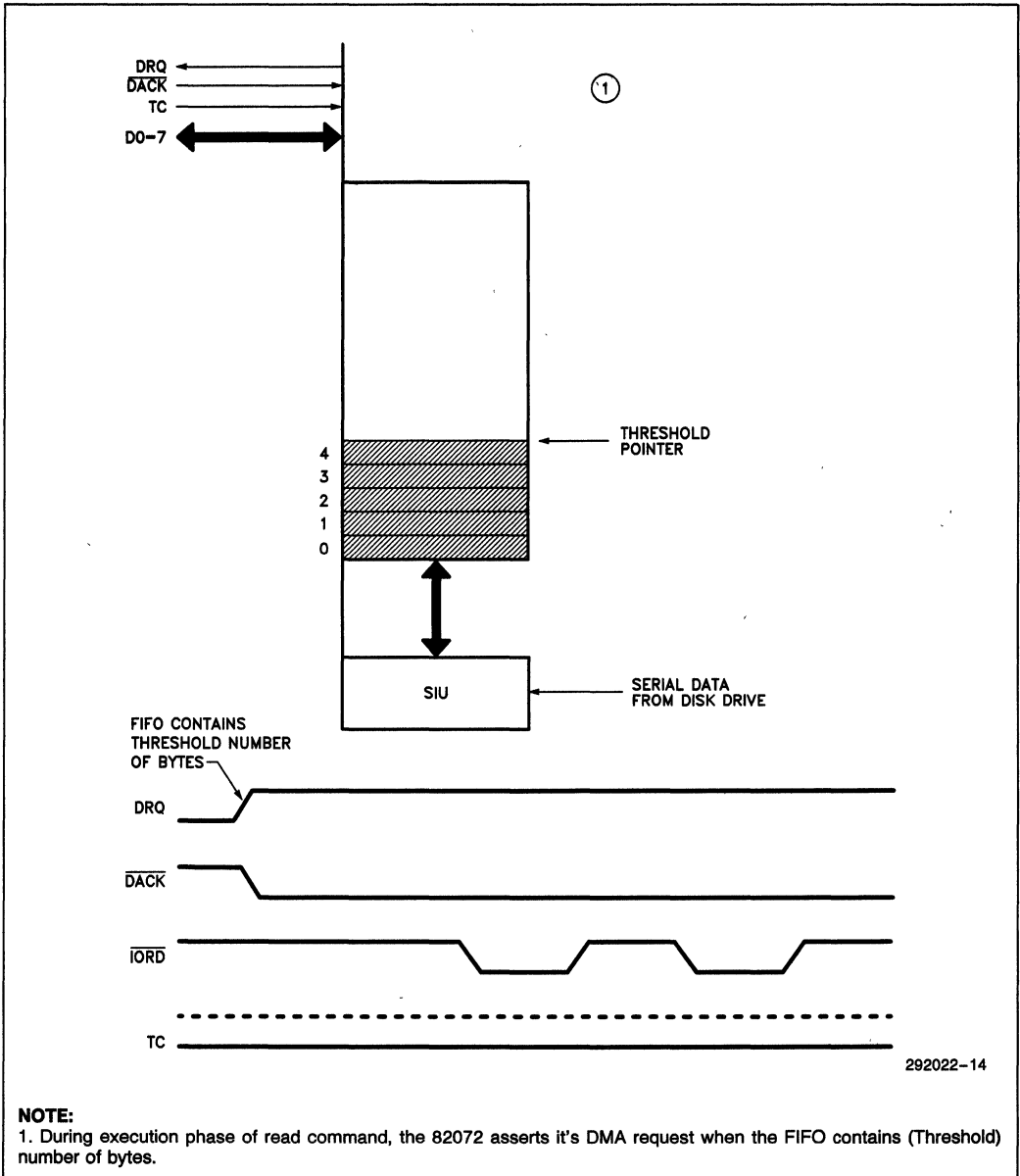


Figure 3.2. FIFO Operation During a Read Command

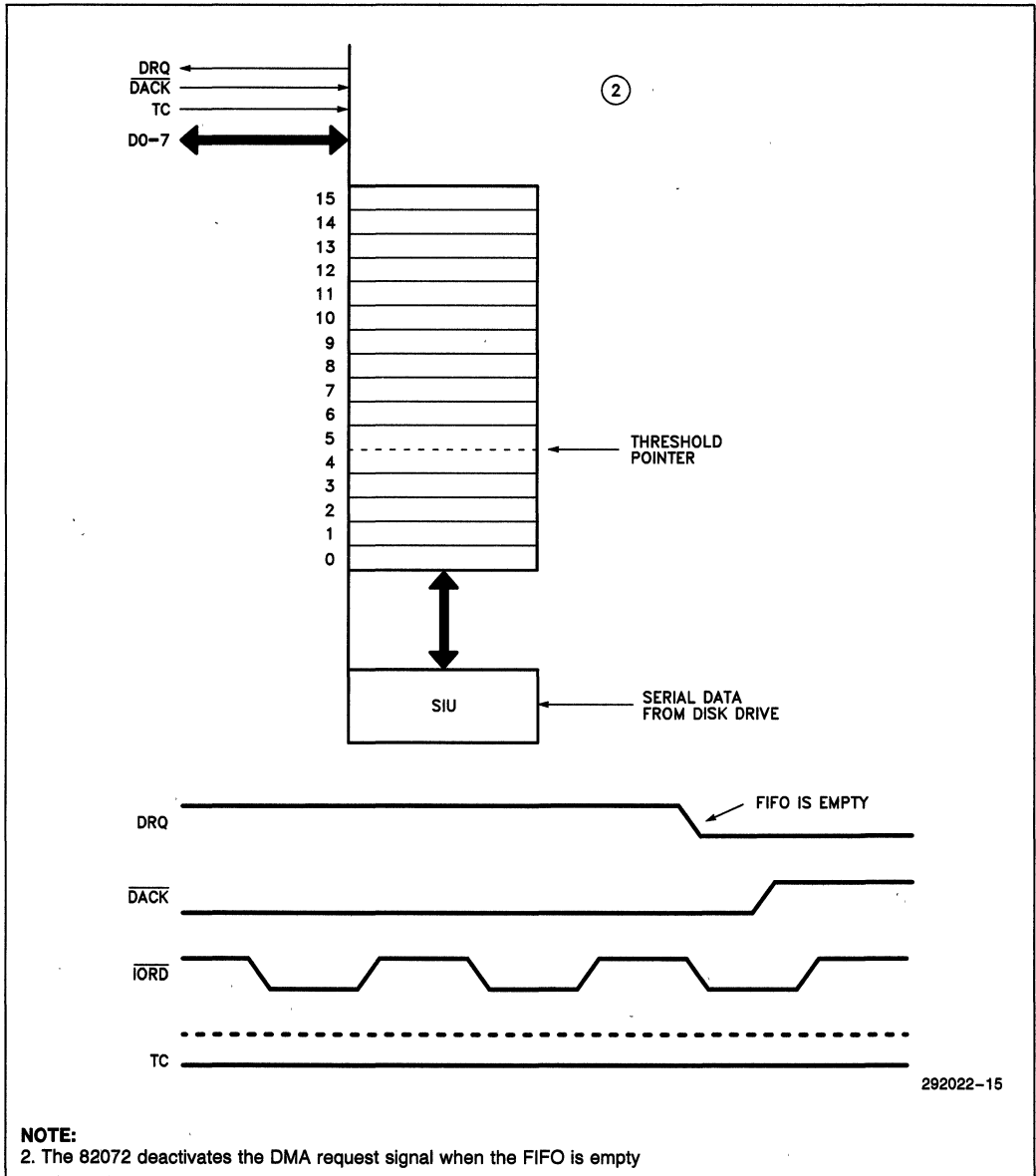


Figure 3.2. FIFO Operation During a Read Command (Continued)

Table 3.4. Minimum Safe FIFO Limit

Fp/Fs	Clocks Between DMA Bursts N1	NAm _{ax}	NAm _{in}	FIFO Threshold	
				MIn	Max
8	500	100	20	2	8
16	500	100	20	1	12
26.67	500	100	20	1	13
32	500	100	20	1	14

In order to assure that the CPU can have N1 clock cycles between burst cycles, the following conditions must hold: the time it takes the serial unit to fill the FIFO till the programmed threshold plus the minimum bus acquisition time must be greater than the time interval between burst cycles.

$$(16 - \text{LIMIT}) / (\text{Fs} / 8) + \text{NAm}_{in} / (\text{Fp}) > \text{N1} / \text{Fp}$$

$$\text{LIMIT} < 16 - \text{Fs}(\text{N1} - \text{NAm}_{in}) / (8\text{Fp})$$

Note that this analysis assumes steady state conditions, and should only be used as a rough guideline in arriving at the optimal FIFO threshold.

3.3.1 Bus Utilization

Bus utilization is defined as the fraction of time that the bus is not busy servicing the 82072 during data trans-

fers. The following analysis also assumes steady state conditions, and should be regarded as a guideline only.

The CPU utilization time is the compliment of the time taken by the CPU to empty the FIFO (tF) divided by the cycle time from FIFO empty to the time it is empty again.

Definition of terms used in the equation

tF = time it takes the system bus to fill the FIFO

tA = bus acquisition time.

FIFO Threshold = tF

$$\text{BUS UTILIZATION} = 1 - \frac{tF / (tF / (fp/8))}{tF + tA}$$

3.4 CONTROLLER/DRIVE INTERFACE

Figure 3.3 is a block diagram of the floppy drive interface required to interface the controller to four high capacity disk drives. Either single or double density disk drives can be supported. No external circuitry is required to generate the write clock. The only external logic needed are a decoder to generate the drive select signals, high current line drivers and Schmitt triggered input gates.

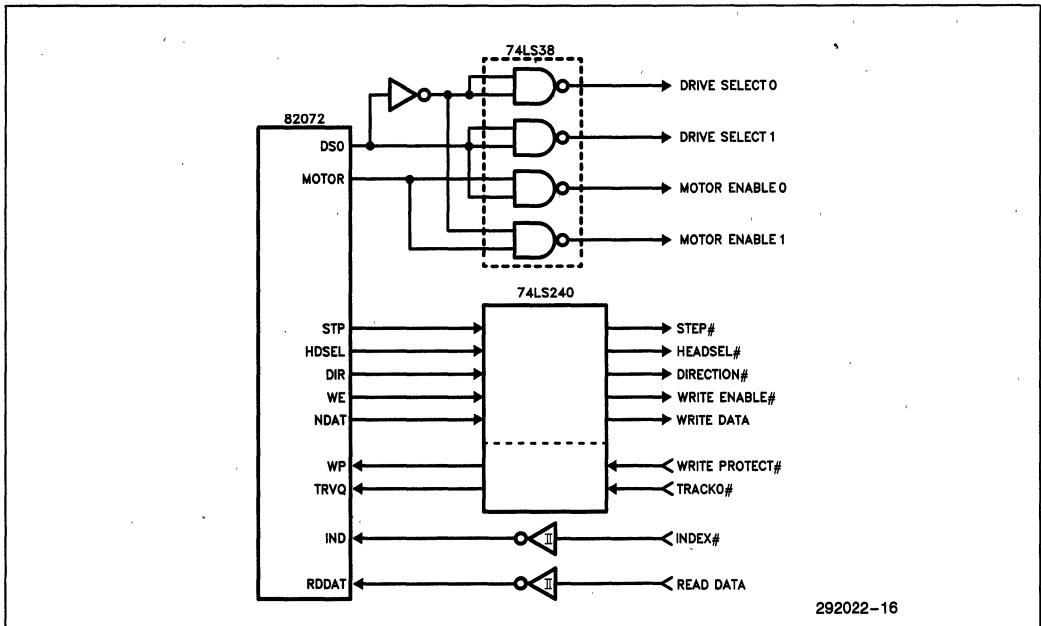


Figure 3.3. Drive Interface Logic

CHAPTER 4 SOFTWARE OVERVIEW

The 82072 is a highly integrated floppy disk controller that supports both single and double density disk storage subsystems. The 82072 supports the IBM 3740 single-density recording format and IBM System 34 double-density recording format. The 82072 accepts and executes high level disk commands such as format a track, read sector and write sector. All data synchronization and error checking is automatically performed by the 82072 to ensure reliable data storage and subsequent retrieval.

The software required by the 82072 consists of input/output drivers whose functions are to:

1. Initialize the 82072 at power on.
2. Issue commands to the 82072.
3. Handle completion interrupts from the 82072.

4.1 INPUT/OUTPUT PARAMETER BLOCK

Most disk operations require multiple byte transfers to the 82072 before the command can be executed. For example, the program must specify the drive number, the track number, the sector number and the side for double density diskettes from which data is to be read or written. I/O driver routines communicate this data to the 82072 in the requisite order, reading them out of a user programmed Input/Output parameter block (IOPB). The Input/Output parameter block would be an array in memory. Figure 4.1 shows the format of this array.

Fourteen of the fifteen commands require multiple bytes before the command can be successfully executed. Regardless of the command or the number of bytes it uses, the user program must initialize the IOPB before calling the driver routine. Some of the parameters of the IOPB are dynamic (e.g., cylinder number, sector number, etc.) and must be written into the IOPB buffer, before calling the I/O driver. Other parameters remain fixed (e.g., gap lengths, number of bytes per sector, etc.) during program operation and need only be written once.

4.2 POWER ON INITIALIZATION

When power is first applied to the FDC, it is not programmed. The first step is to reset the device, which can be accomplished either by a hardware or a software reset. This should then be followed by a SPECIFY command to set the 82072 signal timings so that it will interface correctly with the attached disk drives.

4.2.1 Initialization Procedure

Flow Chart 4.1 illustrates the initialization procedure following system reset.

ADDRESS OFFSET

0	DMA OPERATION
1	DMA PAGE REGISTER
2	DMA ADDRESS LOW BYTE
3	DMA ADDRESS HIGH BYTE
4	DMA COUNT LOW BYTE
5	DMA COUNT HIGH BYTE
6	DISK COMMAND 0
7	DISK COMMAND 1
8	DISK COMMAND 2
9	DISK COMMAND 3
10	DISK COMMAND 4
11	DISK COMMAND 5
12	DISK COMMAND 6
13	DISK COMMAND 7
14	DISK COMMAND 8
15	DISK RESULT 0
16	DISK RESULT 1
17	DISK RESULT 2
18	DISK RESULT 3
19	DISK RESULT 4
20	DISK RESULT 5
21	DISK RESULT 6
22	MISC

Figure 4.1. Input/Output Control Clock Format

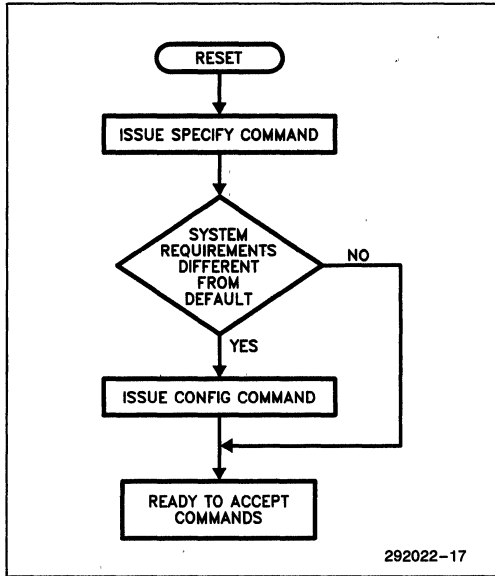


Figure 4.1. Initialization Flowchart

1. SPECIFY COMMAND

This command sets the 82072's signal timings so that the controller will interface correctly to the attached disk drive. The Specify command requires four parameters:

1. Step Rate time (SRT): The 'SRT' defines the time interval between step pulses. Step pulses are used by the disk drive to position the read/write head over the desired cylinder. The step pulse causes the head to move forward or backwards (depending upon the polarity of DIRECTION pin) by one track.
2. Head Load time (HLT): The 'HLT' defines the time interval that the 82072 waits after loading the head before initiating the read or write operation.
3. Head Unload time (HUT): The 'HUT' defines the time interval after the execution phase (of a read or write command) until the head is unloaded.
4. Non-DMA Mode Flag (ND): The 'ND' bit determines if the controller is programmed to operate in the DMA or the non-DMA mode. In the non-DMA mode, the processor is interrupted for each data byte to be transferred. In the DMA mode, the 82072 interfaces to the DMA controller via the DRQ and DACK hand shake signals.

The specify command must be issued prior to performing any disk operations, to define the drive operating characteristics (Refer to the 82072 Data Sheet for timing information). SPECIFY is typically performed following power on initialization. Specify has neither an execution phase nor a result phase.

2. CONFIGURE COMMAND

After system reset, the 82072 defaults to the '8272A' compatible mode. The bits set by the CONFIGURE command default to:

- FIFOTHR = 01. Enable FIFO (FIFO threshold is set to "1").
- MOFF = Infinite motor off delay.
- PRETRK = 00. The track number beyond which write pre-comp is enabled. Defaults to pre-compensation always enabled.
- EIS = 0. No implied seek.
- EFIFO = 0. 8272A transfer mode is enabled (i.e., data transfer is performed on a byte by byte basis). The FIFO is enabled with the FIFO threshold set to "1".
- POLL = 0. disable drive polling.

The default mode enables the 82072 to be software compatible with the 8272A. After receiving a CONFIGURE command, the 82072 will proceed to initialize its internal registers to enable the desired features.

PRETRACK

The PRETRACK variable allows the programmer to select the track number beyond which the write pre-compensation is enabled.

FIFOTHR

The 82072 provides a sixteen byte FIFO for a more efficient CPU interface. The FIFO threshold is programmable from 1 to 16 bytes.

MOFF, MON

The motor logic provides a programmable motor signal to the floppy disk drives. If the drives is not currently selected or the MOTOR signal is not on, the drive se-

lect and the MOTOR signal are activated the programmed time before the drive is accessed. MON is programmable in increments of rotation times of the disk (0 to 15) for a range of 0 to 3 seconds. Counting rotations makes the delay independent of the data rates. MOFF is programmed in the same way as MON, giving a range of about 0 to 6 seconds.

If the CPU requests an overlapped SEEK or RECALIBRATE operation, the programmed MON time will be ignored and the step pulses will be issued once the SRT time has elapsed. Refer to the 82072 Data Sheet for additional details.

EIS

Setting EIS to "1" enables implied seek for the READ, WRITE commands. When the implied seek mode is selected, a SEEK command will be executed, before executing the read or write operation. If the seek operation fails, an error flag is set, the 82072 generates an interrupt and enters the result phase of the issued read or write command. The parameter C in the result phase indicates the track number at the time of the abnormal termination. All the remaining status bits are set to zeroes.

EFIFO

This bit controls the mode of data transfers to and from the host system. When this bit is to a "1", the FIFO is disabled, and the 82072 is in the "8272A compatible" mode. In this mode, the data transfers are performed on a byte by byte basis. The 82072 FIFO threshold defaults to one.

POLL

When set to a "1", POLL disables drive polling. The 82072 comes out of reset with polling disabled. When enabled, the 82072 scans all the drives to check if the READY has changed status. Any change of the ready status on any drive will cause an interrupt.

An indepth discussion of the CONFIGURE command can be found in the 82072 Data Sheet.

The CONFIGURE command need not be issued if the system requirements are satisfied by the default mode. If the system requirements are different, this command should be issued before executing data transfer commands.

4.3 CONTROL AND DATA TRANSFER COMMANDS

1. CONTROL COMMANDS

These commands are used to position the head on the desired track and to get information regarding the status of the disk drives. The control commands are summarized below:

1. SEEK
2. RECALIBRATE
3. RELATIVE SEEK
4. MOTOR ON/OFF
5. SENSE INTERRUPT STATUS
6. SENSE DRIVE STATUS
7. DUMPREG
8. READ ID

The SENSE DRIVE STATUS (SDS) command can be performed between other commands to obtain the status of any one of the drives. The status of the drive is available immediately. The SDS has no execution phase and no interrupts are generated.

The SEEK and RECALIBRATE commands are performed prior to a data transfer command, to position the Read/Write head over the desired cylinder. The Host processor is not involved during the execution phase of these commands. At the end of the execution phase of these commands, the 82072 generates an interrupt. A SENSE INTERRUPT STATUS command has to be issued in response to the interrupt.

The MON/MOFF command provides software control of the MOTOR pin. This command has no execution phase and no interrupts are generated.

The DUMPREG command is designed to support system run-time diagnostics and application software development and debug. The DUMPREG command can be issued following the SPECIFY or the CONFIGURE command, to determine if there is proper communication between the Host and the floppy disk controller. This command has no execution phase and no interrupts are generated.

A detailed description of these commands are contained in the 82072 Data Sheet.

2. DATA TRANSFER COMMANDS

The 82072 supports six data transfer commands. They are:

1. Read Data Command
2. Read Delete Data Command
3. Read a Track Command
4. Write Data Command
5. Write Deleted Data Command
6. Format a Track Command

The data transfer commands all require the same parameter bytes and return the same status bytes. The only difference between the data transfer commands are the coding of bits 0-4 (D3-D0) in the first command byte sent to the 82072.

Table 4.1. Data Transfer coding

Command	D3	D2	D1	D0
Read Data	0	1	1	0
Read Deleted Data	1	1	0	0
Write Data	0	1	0	1
Write Deleted Data	1	0	0	1
Read Track	0	0	1	0

The format common to all the data transfer commands and the algorithm for beginning and completing the execution is described below.

The general format of the Command Parameter Block includes the following fields:

Byte #	Symbol	Description
1	HDS & DS0, 1	Head select and drive select.
2	C	Cylinder address. The currently selected cylinder address.
3	H	Selected head address.
4	R	Sector address. Specifies the sector number to read or written.
5	N	Sector size code. The number of data bytes within a sector.
6	EOT	The End of Track. The final sector number of the current track.
7	GPL	Gap Length. Gap 3 size for read or write operation. Gap are regions introduced during disk format. The gaps are used to turn the drives Read/Write head off. This prevents the corruption of the data recorded on the disks.
8	DTL	Special sector size. This parameter is used to temporarily alter the sector size. By setting N to zero, the DTL may be used to specify a sector size from 1 to 256 bytes.

EXECUTING A DATA TRANSFER COMMAND

As mentioned earlier in section 2.4, each command can be considered to have three phases.

1. Command Phase: The processor issues all the command/parameter bytes required to perform the command.

2. Execution Phase: The 82072 performs the operation as instructed.

3. Result Phase: The controller provides the CPU with status information.

The paragraphs to follow describe the three phases with reference to data transfer commands. Figure 4.4 shows the flow chart for the Data Transfer commands.

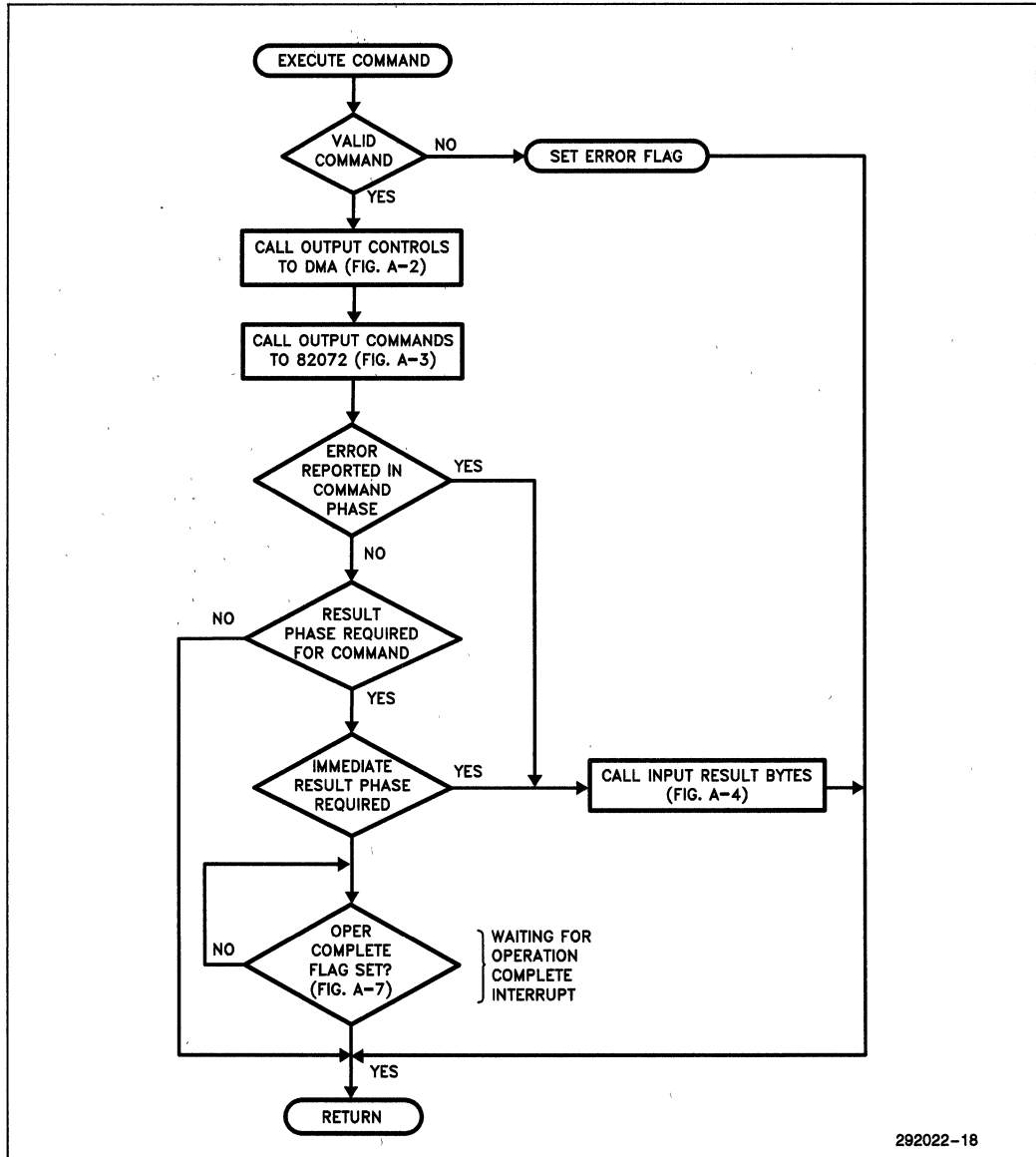


Figure 4.4. Flow Chart for Data Transfer Commands

1. COMMAND PHASE OF THE DATA TRANSFER COMMAND

The data transfer commands consist of a command byte and eight parameter bytes. (Except for FORMAT command, which requires one command byte and five parameter bytes.) The host processor must transfer the command and parameter bytes required to perform the data transfer operation, in the prescribed sequence (refer to the 82072 Data Sheet for the command sequence). The command bytes must be sent after polling the Main Status Register to determine if the FIFO register is available. The handshake mechanism to transfer command bytes is explained in Section 2.4.1. After the last byte of data is sent to the 82072 in the command phase, the 82072 automatically enters the execution phase. In a similar fashion, after the last byte of result is read from the 82072 in the result phase, the result phase is ended and the 82072 reenters the command phase.

The 82072 determines if the first byte sent is a valid command byte. If valid, it continues to request additional bytes. If, on the other hand, the first byte was invalid, the 82072 issues an interrupt and enters the result phase. All the result bytes must be read to successfully terminate the data transfer command.

2. EXECUTION PHASE OF THE DATA TRANSFER COMMANDS

Upon receiving the command byte and 8 parameter bytes, the 82072 loads the head, waits the specified head settling time, and begins reading the ID address marks and the ID fields. When the sector number stored in the ID register matches the sector number read from the diskette, the 82072 starts the actual data transfer. If the current data transfer command being executed is a read from the disk drive, then the controller assembles the serial data bits sent by the disk drive into eight-bit bytes and loads the FIFO. When the FIFO contains the programmed threshold number of bytes, the controller activates its 'DRQ' line (if programmed to operate in the DMA mode) indicating it requires DMA Controller's attention. The DMA controller transfers the data from the FIFO to the system memory. The DRQ line is deactivated when the FIFO is emptied.

The data transfer commands automatically operates in the multi-sector mode, i.e., after completion of read from the current sector, the sector address is incremented by one and the data from the next sector is transferred. Additionally, multi-track operation may be specified (which allows data to be transferred from both sides of the diskette) by setting the 'MT' flag in the command word. The amount of data that can be transferred with a single command depends upon the multi-track flag, the recording density and the number of bytes per sector.

The data transfer is terminated upon receiving a terminal count (TC) through its TC pin or if an attempt is

made to read past the 'EOT' (End of Track) specified in the parameter field. Upon receiving the terminal count, the 82072 stops outputting data to the FIFO, but continues to read data from the current sector to check for CRC errors.

ERRORS DURING EXECUTION

The data transfer commands will be terminated if any of the following conditions occur:

1. If the 82072 detects the Index mark twice without finding the requested sector, the 82072 sets the 'sector not found error' flag and terminates the data transfer command. The bits 7 and 6 of the status register ST0 are set to "01".

NOTE:

The controller searches for each sector in a multi-sector operation. Therefore a 'sector not found error' may occur after successful transfer of one or more sectors.

2. If the 82072 reads the deleted data address mark from the disk and the skip flag is not set, the 82072 sets the "control mark" flag (bit 6 in status register 2) and terminates the data transfer command. If the skip flag is set, the 82072 skips the sector with the deleted data address marks, sets the "control mark" flag and reads the next sector. Thus, the 82072 could be forced to skip the sectors with the deleted data address marks, during multi-sector transfers.
3. After reading the ID fields and the data field in each sector, the 82072 checks for CRC bytes. If it finds an incorrect CRC in the ID field, it sets the "data error" flag in the status register 2 and aborts the data transfer. Bits 7 and 6 in the status register ST0 will be set to a '01' respectively.
4. During data transfers between the 82072 and the system, the 82072 must be serviced by the system before an underrun or an overflow condition occurs. These error conditions can be avoided by choosing an optimal FIFO threshold (Chapter 2 shows how to arrive at an optimum FIFO threshold).

3. RESULT PHASE OF THE DATA TRANSFER COMMANDS

After the completion of the data transfer, the 82072 enters the result phase. The 82072 generates an interrupt to the processor. The processor must respond to the interrupt and read the seven result bytes to successfully terminate the data transfer command. The result bytes are read by the "handshake" mechanism described in chapter 2.4.1, and contain status information.

Following the data transfer command, the head is not unloaded until after the head unload time has elapsed. If the host issues another command before the head unload time has elapsed, the head remains loaded. This feature allows subsequent commands to bypass the head load time delay.

CHAPTER 5 APPLICATIONS

This chapter presents three 82072 design examples. The first example illustrates an 82072 in the popular PC environment. It shows a generic floppy disk controller board design that is compatible with the PC/PC-XT. The board can be used as a direct replacement for the existing PC/PC-XT floppy disk controller board. The second design covers the PC-AT. By adding one additional TTL package to the PC/PC-XT design, a 100% PC-AT compatible design can be realized. No software modifications are required.

Additionally, the BIOS can be modified to support the enhanced features of the 82072 such as:

1. Burst data transfers
2. Implied Seek
3. Power Down Mode
4. Relative seek and disk paging
5. 1 Mbps data transfer rate

The third example describes the hardware required to interface the 82072 to the high integration iAPX 186 microprocessor.

Although specific microprocessors are used in each of these examples, the applications can be applied to either MCS85, iAPX86, iAPX88, iAPX 386 or any other processors, provided the necessary hardware changes are made. These applications should serve as a useful guide in illustrating the various procedures in using the 82072.

5.1 EXAMPLE 1. INTERFACING TO THE PC/PC-XT

This example illustrates a generic floppy disk controller board design for the PC/PC-XT.

5.1.1 Interface Requirements

The 82072 based floppy disk controller board occupies a single slot on the I/O channel of the IBM PCs. For clarity, the I/O channel configuration of the PC/PC-XT is used in this example. While the PC-AT's I/O channel configuration differs from the PC/PC-XT, the sub-set of the signals used by the floppy disk controller board is identical.

The floppy disk controller board requires the resources of the processor, DMA and the interrupt controller on the system board. A block diagram of the design is shown in Figure 5.1. As can be seen from the block diagram, the floppy disk sub-system consists of five functional units:

1. Interface between the 82072 card and the I/O Channel
2. Chip-select generation
3. Clock generation
4. DMA interface
5. Drive interface

The implementation of these functional units is discussed in the following paragraphs.

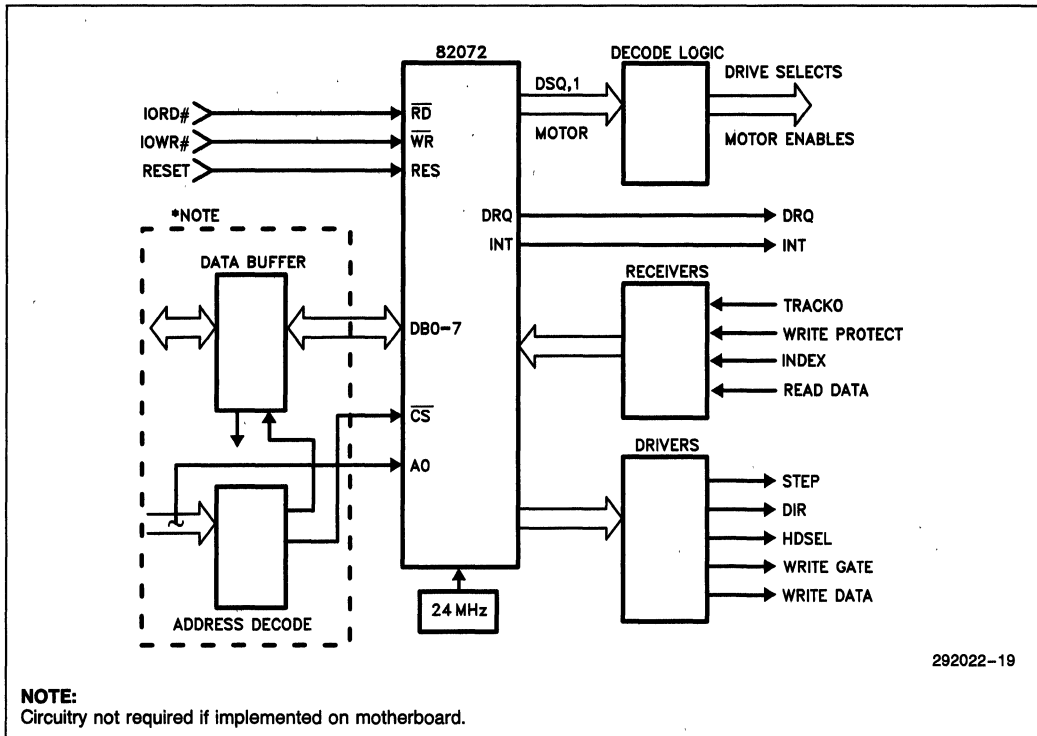


Figure 5.1. Floppy Disk Controller Block Diagram

1. INTERFACING TO THE I/O CHANNEL

The IBM PC/PC-XT has 8 slots on the system board, which allows expansion of the basic system. The expansion slots are electrically identical. The I/O channel contains an 8-bit bi-directional data bus, 20 address lines, 6 levels of interrupt, 3 DMA channels and other control signals required to perform I/O and memory read/write operations. Figure 5.2 shows the signals and the pin assignments for the I/O channel. The existing disk controller boards designed for the PC/PC-XT use interrupt level 6 to get the attention of the processor, and channel 2 of the DMA controller for disk data transfers. To maintain compatibility with the existing floppy disk controller boards, this design example used interrupt level 6 and DMA channel 2.

2. CHIP SELECT AND DATA BUS INTERFACE

Table 5.1. shows the IBM PC's I/O address map. To maintain compatibility with the existing floppy disk

controller boards for the IBM PC, the 82072 based floppy disk controller board's I/O port address space was chosen to be 3F0-3F7H. Signals A1 through A9 and AEN (Address Enable) are decoded to generate the chip-select for the 82072. This design uses a 50C60 EPLD (Erasable Programmable Logic Device) to generate the chip-select for the 82072 and the control signals for the transceiver (74LS245).

The PC/PC-XT supports 8 expansion slots. However, because of the fan out constraints, each I/O channel slot signal loading must be limited to only one TTL load. Consequently, to meet this specification, the data bus is buffered through a transceiver. The DT/R (data transmit/receive) line of the transceiver is controlled by a $\overline{\text{IORD}}$ signal. The $\overline{\text{DEN}}$ signal is controlled by "BDSEL" Signal. (Please refer to the EPLD equations in Figures 5.4, 5.5). The transceiver is enabled only when the floppy disk controller board is accessed, thereby preventing potential data bus contention.

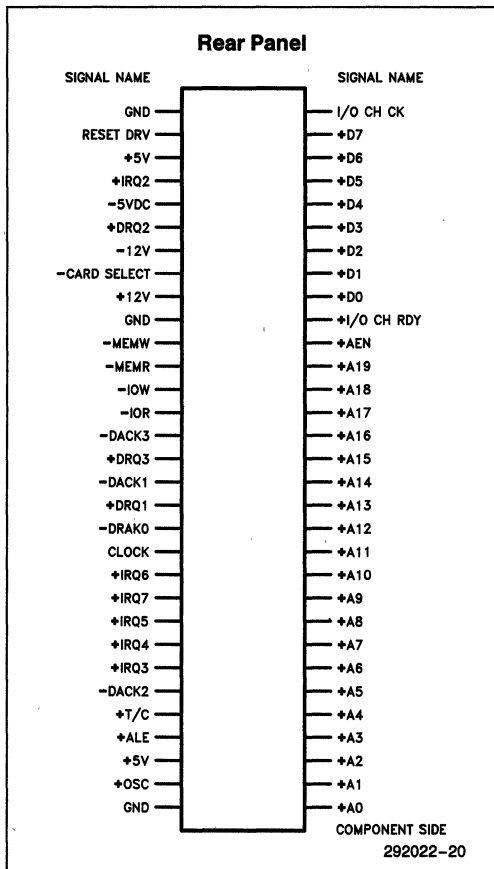


Figure 5.2. I/O Channel Diagram

3. GENERATION

The 82072 requires an external 24 MHz system clock for its operation. The clock input controls the internal operations of the 82072 and its data transfer rate. A 24 MHz fundamental parallel resonant crystal can be attached to the X1 and X2 input pins of the 82072 or the X1 (X2 floats) input can be driven directly by a MOS level clock. This design example uses an external 24 MHz crystal.

4. DISK DRIVE INTERFACE

The 82072 provides all the signals required to interface the host processor to four high capacity disk drives. The only additional logic required are high current drivers to drive the cable and drive select/motor enable decode logic. If the length of the cable connecting the disk drives to the controller is relatively small, the high current drivers can be replaced with CMOS equivalent TTL circuits.

Table 5.1. PC/PC-XT Address Map

Hex Range	Usage
000-00F	DMA Controller 1 8237A-5
020-021	Interrupt Controller, 8259A
040-043	Timer, 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Registers
0AX*	NMI Mask Register
200-20F	Game Control
210-217	Expansion Unit
2F8-2FF	Asynchronous Communications (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Printer
380-38C**	SDLC Communications
380-389**	Binary Synchronous Communications (Second)
390-393	Cluster
3A0-3AF	Binary Synch Communications (Primary)
3B0-3BF	Monochrome Display and Printer Adapter
3D0-3DF	Colour/Graphics Monitor Adapter
3F0-3F7	Diskette Controller
3F8-3FF	Asynchronous Communications (Primary)

*Upon reset, the NMI into the 8088 is masked off. The mask bit can be set and reset with system programs as follows:

MASK ON: Write I/O address 070H, with data bit 7 equal 0

MASK OFF: Write I/O address 070H, with data bit 7 equal to 1.

**SDLC Communications and secondary Binary Synchronous communications cannot be used together as their hex addresses overlap.

Motor and Drive Select Logic

This design example uses the internal Motor enable and drive select logic. The internal Motor logic provides a programmable motor enable signal to the disk drives. It activates the motor enable signal and waits the pre-programmed time (time required for the drive to spin up and stabilize), before executing the command. Upon reset, the 82072 defaults to support a spin up time of 1 second (which is the value used by the PC/PC-XT). Since the 82072 provides a single motor enable signal, it should be gated by the decoded drive select signals to generate the motor enable signals for the appropriate drives. The 82072 activates the appropriate drive select outputs along with the Motor enable signal.

Figure 5.3 shows the circuitry required for generating the drive select and motor enable signals. Drive Select 0 (DS0) and MOTOR signals from the 82072 are decoded, using an 74LS14 inverter and 74LS38 open collector NAND gates, to generate two drive select and motor enable signals. The design can easily be upgraded to support four drives by using a decoder/driver of the type 74LS156. The limitation of the 74LS156 is that it can only source 30 mA. This is sufficient to support small cable assemblies. If longer cable lengths (maximum of 10 feet) have to be supported, then open collector drivers like the 74LS38 have to be used instead. For low power applications, the TTL gates can be replaced by CMOS gates (e.g., HCT 14, HCT 240 ...).

Head Positioning and Drive Status Circuitry

The step, head select, direction, write enable and write data signal outputs to the disk drive are derived by buffering the STP, HDSEL, DIR, WE and WRDATA from the 82072 through a 74LS240 driver. The other half of the 74LS240 acts as a buffer between the Track 0 and Write Protect signals from the disk drive and the 82072. The Read Data input is inverted by a Schmitt triggered inverter and fed directly into the RDDATA input of the 82072. The 82072's internal data separator generates the data window signal which is used by the read hardware to extract data from the serial Read Data stream.

A single 34-conductor cable connects the floppy disk controller board to the disk drives. The last disk drive should be resistively terminated by means of 150Ω resistors pulled-up to +5V.

5. DMA INTERFACE

The 82072 requires a single DMA channel for its operation. The 82072 interfaces to system memory by means of the 8237A-5 DMA controller. Each channel on the 8237A-5 can transfer data throughout the sixteen Megabyte system address space in 64 KB blocks. To support the 20 bit addressing, the IBM PC has a page register for each of the DMA channels. Table 5.2 shows the address generation for the PC DMA channels.

Table 5.2. PC DMA Address Generation

Source	DMA Page Register	8237-5
Address	A19<----->A16	A15<----->A0

The processor writes the upper four address bits (A20–A16) to the page register before initiating a data transfer command. When the DMA controller assumes control of the system bus, the contents of the page register are enabled onto the upper four bits of the address bus. The only restriction in the use of this page register is that a single read or write transfer should not cross the 64K memory boundary (FFFFH).

1. OVERVIEW OF THE PC/PC-XT DMA

The IBM PC system board has one 8237A-5 DMA controller. Channel 0 is used for doing the refresh of DRAMs. Channels 1, 2 and 3 are available for add-on boards. The DMA channel assignments for the PC/PC-XT are as follows:

Table 5.3. DMA Channel Assignments

Controller
Chn0-DRAM REFRESH
Chn1-Spare
Chn2-Spare
Chn3-Spare

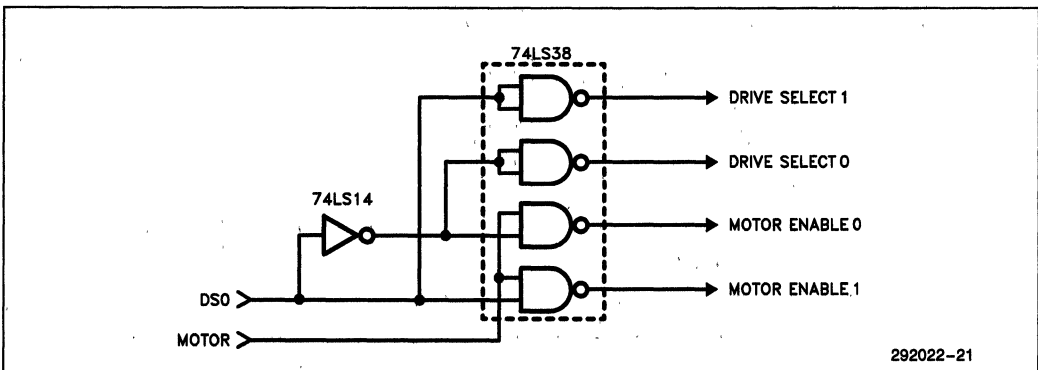
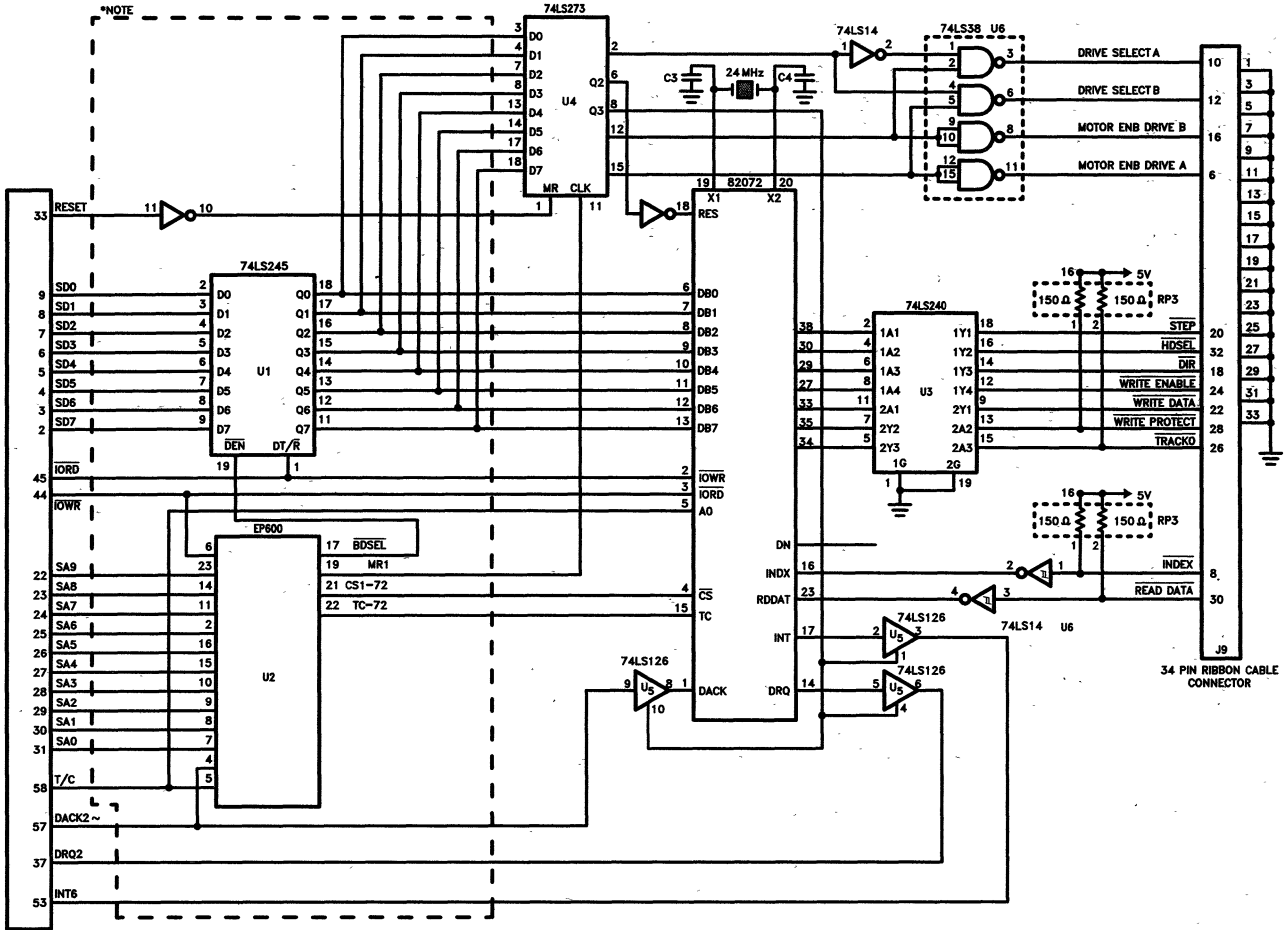


Figure 5.3. Drive Select and Motor Enable Circuit



NOTE:
*Circuitry not required if implemented on motherboard.

Figure 5.4. SBX 072 Floppy Disk Controller Board for PC/PC-XT


```

PART: SC060
INPUTS: SA9, SA8, SA7, SA6, SA5, SA4, SA3, SA2, SA1, SA0, IOW, TC, DACK, AEN
OUTPUTS: BDSEL, TRIGO, TRIG1, CS072, CS172, TC72

```

NETWORK:

```

AEN= INP(AEN)    % INPUT PIN DEFINITIONS %
SA9= INP(SA9)
SA8= INP(SA8)
SA7= INP(SA7)
SA6= INP(SA6)
SA5= INP(SA5)
SA4= INP(SA4)
SA3= INP(SA3)
SA2= INP(SA2)
SA1= INP(SA1)
SA0= INP(SA0)
IOW= INP(IOW)
TC=  INP(TC)
DACK=INP(DACK)

```

```

TRIGO = CONF(TRIGOC, VCC) % OUTPUT PIN DEFINITIONS %
CS072 = CONF(CS072C, VCC)
TC72  = CONF(TC72C, VCC)
BDSEL = CONF(BDSEL, VCC)

```

EQUATIONS

```

CS072C= !(AEN * SA9 * SA8 * SA7 * SA6 * SA5 * SA4 * SA3 * SA2 * SA1);
TRIGOC= (!IOW & AEN & SA9 & SA8 & SA7 & SA6 & SA5 & SA4 & SA3 & SA2 & SA1 & SA0);
BDSEL= !(SA9 * SA8 * SA7 * SA6 * SA5 * SA4 * SA3);
TC72C= (!DACK * TC);

```

Figure 5.5. EPLD Equations for PC/PC-XT Compatible FDC Board

Since channel 0 of the 8237A-5 DMA controller is used to do the refresh of the DRAMs, all of the remaining channels should be operated in a single-byte transfer mode. In this mode, the channel with the highest priority gets the DMA access. After the DMA cycle is granted, the next channel with the next highest priority gains DMA access. This priority scheme ensures that no single channel can "hog" the bus. More importantly, it ensures that the DRAMs are refreshed once every 15 microseconds, as the refresh channel has the highest priority.

This "byte-by-byte" transfer mode of operation is very slow, as HOLD is dropped after every DMA cycle, and then asserted again for the next cycle. The HOLD/HLDA handshake mechanism or byte-by-byte transfer results in reduced bus bandwidth due to the overhead resulting from latencies in bus acquisition. Burst mode operation is a more desirable approach when using the 82072 but cannot be used here due to the DRAM refresh requirements.

CONCLUSION

The 82072 based floppy disk controller design is 100% compatible with the existing disk controller boards for the PC/PC-XT. Only eight components are required for implementing a disk controller board as opposed to the existing boards that use more than 40 components. Furthermore, the 82072 can interface to the 10 MHz 8088 microprocessors without waitstates (the 8272A introduces two waitstates).

5.2 DESIGNING FOR THE PC-AT

The previous section illustrated the design of a floppy disk controller board for the PC/PC-XT. With the addition of one TTL package, the same design can also be used for the PC-AT.

This example provides a brief overview of the current PC-AT floppy disk controller implementation. It is followed by a discussion of the compatibility issues that arise when interfacing the PC-AT to the 82072 based floppy disk controller board. The section concludes with a discussion of the enhancements provided by the 82072 based disk controller board.

5.2.1 Overview Of The PC-AT

1. Disk Formats

The PC/PC-XT supports four disk formats: single side—8 sectors per track, single side—9 sectors per track, double side—8 sectors per track, and double side—9 sectors per track. The four formats are derived from the number of sides and the number of sectors on each track. The PC-AT has added an additional quad density format, double side—15 sectors per track. Table 5.3.1 shows the data transfer rates associated with the different capacity drives.

Table 5.3.1. The Standard Floppy Disk DOS Formats

Capacity	Density	Drive Speed	Data Rate	Sectors	Tracks
160 K	Single	300 rpm	250 Kbps	8	40
160 K	Single	360 rpm	300 Kbps*	8	40
180 K	Single	300 rpm	250 Kbps	9	40
320 K	Double	300 rpm	250 Kbps	8	40
320 K	Double	360 rpm	300 Kbps*	8	40
360 K	Double	300 rpm	250 Kbps	9	40
1.2 K	Quad	360 rpm	500 Kbps	15	80

* 160K/320K diskettes in 1.2 Mbyte Drives.

2. PC-AT Register Set

To support different capacity drives and the different data transfer rates, the current PC-AT disk controller

board incorporates three additional registers: the Data Rate Register (DRR) and the Digital Input Register (DIR) and the Digital Output Register (DOR). These registers are shown in the PC-AT floppy controller Block Diagram, Figure 5.6. The I/O address map for the registers, along with a brief description, are given below:

Table 5.4. I/O Address Map for the PC-AT

Hex Address	Access Type	Description
3F0H	-----	Unused
3F1H	-----	Unused
3F2H	Write	Digital Output Reg
3F3H	-----	Unused
3F4H	Read	Main Status Reg
3F5H	Read/Write	Data Register
3F6H	-----	Unused
3F7H	Write	Data Rate Register
3F7H	Read	Digital Input Reg

PC-AT Data Rate Register (DRR)

The Data Rate Register is used to specify the type of disk drive and the diskette media (if the diskette is double density or quad density diskette). This is a write only register selected on address 3F7H. This value is used by PC-AT floppy disk controller board's data separator circuit to select the data transfer rate and the disk drive spindle speed. The data rate is selected by the BIOS. The decoding for the Data Rate Register is shown below:

Table 5.5. Drive Decoding

D1	D0	Description
1	1	Reserved
1	0	DD Drive DD Diskette
0	1	QD Drive DD Diskette
0	0	QD Drive QD Diskette

NOTES:

- DD = Double Density
- QD = Quad Density

Additionally, the DRR controls external hardware which generates the "Low Density" (LD) signal. LD is an active high signal that occurs whenever a data transfer rate of 300 Kbps is selected. The PC-AT quad density disk drive was LD internally to vary the Read/Write head and data channel characteristics. This feature allows the PC-AT quad density disk drive to support double density diskettes.

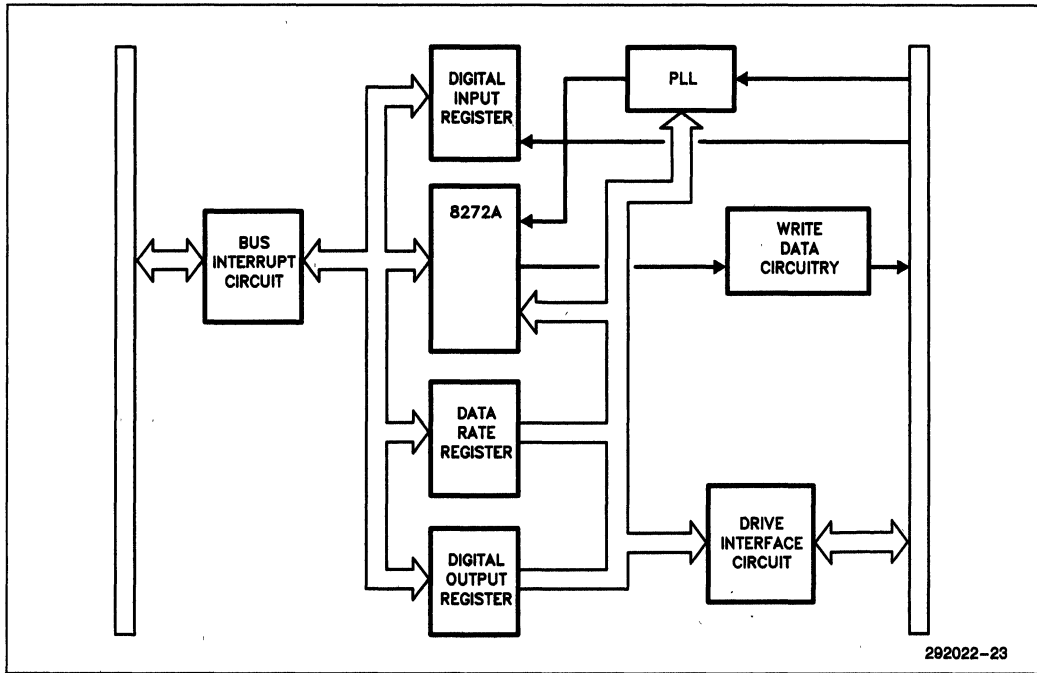


Figure 5.6. PC-AT Floppy Disk Controller Block Diagram

PC-AT Digital Output Register (DOR)

The Digital Output Register is a write only register located at I/O address 3F2H. The individual data bits have the following meanings.

Bits	Definition
0	Drive Select: 0 on this bit indicates that drive A is selected, 1 indicates that drive B is selected.
1	Reserved: This bit is not used by the hardware. It can be set to a 1 or a 0.
2	Diskette Function Reset: When this bit is set to a 1, the diskette reset function is disabled.
3	Enable Diskette DMA and Interrupts: When this bit is set to a 0, the DMA and Interrupt lines are enabled.
4	Drive A Motor Enable: When this bit is set to a 1, the Motor A enable signal is activated. A timer is dedicated to perform the motor disable function. The timer is initialized by the BIOS, based on the command selected.
5	Drive B motor Enable; same as 4.
6	Reserved; same as 1.
7	Reserved; same as 1.

PC-AT Digital Input Register (DIR)

The Digital Input Register is a read only register located at I/O address 3F7H. Bit 7 is the only bit used by the floppy disk controller. The other bits are used by the hard disk controller. Bit 7 is set to a one when a quad density drive is accessed. Since the PC may have multiple floppy disk drives, the appropriate drive must first be selected, before reading this register. This signal is reset after the drive is de-selected.

3. PC-AT OPERATION

The PC-AT uses a quad density drive that supports both quad density and double density floppy diskettes. Upon power-up or when the quad density disk drive is loaded, the PC-AT performs a routine to determine the type of media inserted. Initially the BIOS assumes that the diskette inserted is a quad density diskette and defaults to a data transfer rate of 500 Kbps. It reads Track 0, Head 0, sector 1 to determine the diskette type. If successful, it then proceeds to read track 0, Sector 15. A successful read operation indicates that the media is a quad density diskette. An unsuccessful operation implies that the media is a double density diskette. If the diskette is a double density diskette, the PC-AT BIOS programs the hardware to support a data transfer rate of 300 Kbps.

When interfacing to standard double density disk drives, the selected data transfer rate is 250 Kbps.

Table 5.7. Data Rate Select Register Default Values

	SWR	PD	EPLL	PRE-COMP			DRATSEL	
BIT	7	6	5	4	3	2	1	0
CONTENT	0	0	0	0	0	0	1	0

DATA RATE = 250 Kbps
 PRE-COMP = 125 ns
 INTERNAL PLL ENABLED
 POWER DOWN MODE DISABLED
 SOFTWARE RESET DISABLED

292022-24

5.2.2 Compatibility Issues

The 82072 has integrated a software selectable data separator, programmable write pre-compensation logic, Write Clock generation logic and motor on/off logic on-chip. The 82072 has an on-chip clock prescaler that internally divides the 24 MHz clock input to generate the Write Clock. By appropriately setting the data rate selection bits in the DRS, the prescaler value can be changed to obtain the desired Write Clock rate. Additionally, the 82072 has a built-in "Software Reset" feature. This allows the 82072 based disk controller board to support all the features of the PC-AT, without requiring the external data separator or its associated registers. These enhancements were implemented with the objective of maintaining 100% PC-AT compatibility.

Table 5.6. Comparison Between the PC-AT and the 82072

#	Features	82072	PC-AT
1.	Motor Control and Drive Select Ckt	On-chip	External
2.	Programmable Data Transfer Rate	On-chip	External
3.	Software Reset	On-chip	External
4.	DMA and Interrupt enable	External	External
5.	Diskette Change Signal	External	External

COMPATIBILITY CHECK

This section compares the features supported by the PC-AT registers to the on-chip features of the 82072.

PC-AT Data Rate Register (DRR)

The 82072's DRS register is compatible with the PC-AT's DRR register. A comparison of the DRR and the DSR register is shown in Table 5.8.

Table 5.8. Decoding of Data Rate Select Bits

Data Rates MFM	PC-AT		82072	
	Bit1	Bit0	Bit3	Bit4
1 Mbps	-----	-----	1	1
500 Kbps	0	0	0	0
300 Kbps	0	1	0	1
250 Kbps	1	0	1	0

The 82072 Data Rate Select (DSR) register is bit compatible with the PC-AT's DRR register. The only incompatibility is in the address location of the registers. The DRR of the PC-AT is located at I/O address 3F7 Hex. The DSR of the 82072 is located at I/O address 3F4 Hex. This requires that accesses to 3F7 Hex be translated to 3F4 Hex to maintain compatibility. This can be easily implemented using one AND gate as shown in Figure 5.7.

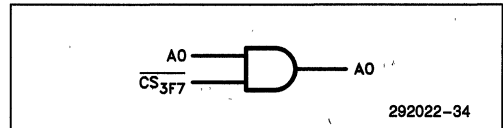


Figure 5.7

The 82072 supports a multiplexed V_{CO}/LD pin. When the internal PLL is selected by setting the EPLL bit, this pin behaves as the Low Density (LD) output to the disk drive. The 82072 activates the LD signal whenever a data transfer rate of 300 Kbps is selected (identical to the PC-AT). When the internal PLL is not used, this pin provides the V_{CO} signal to enable an external PLL.

The overall PC-AT compatible design is illustrated in Figure 5.8. The PAL equations must be modified to provide the additional chip select signal for 3F7 Hex. These equations are provided in Figure 5.9.

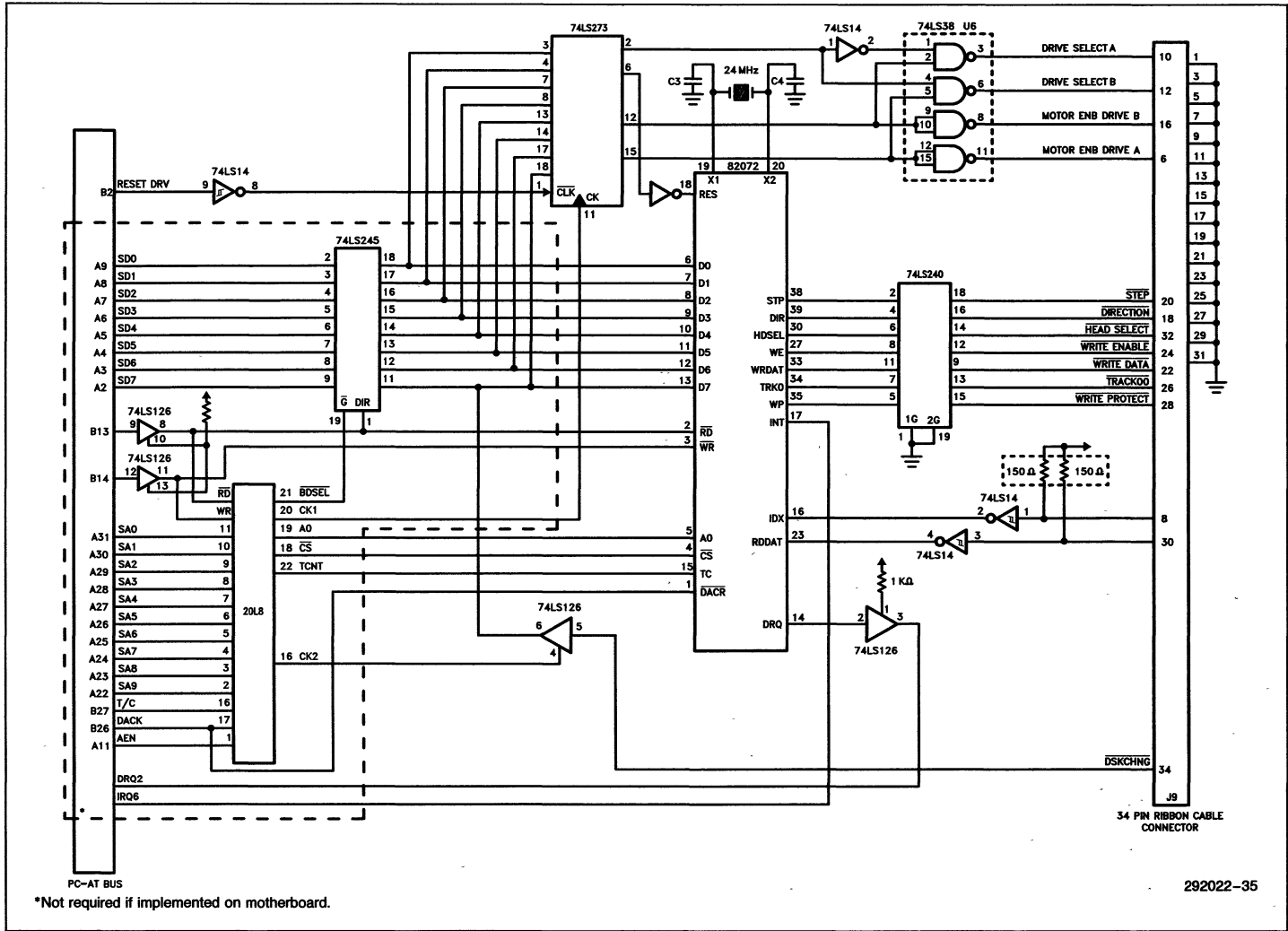


Figure 5.8. SBX 072 Floppy Disk Controller Board for PC-AT

3-203

*Not required if implemented on motherboard.

```

MODULE DECODE_LOGIC;

TITLE 'SBX 82072 PC-AT COMPATIBLE BOARD DECODE LOGIC';
DEC_LOG DEVICE 'P20LS';

VCC,GND                PIN 12,24;
SA9,SA8,SA7,SA6,SA5   PIN 2,3,4,5,6;
SA4,SA3,SA2,SA1,SA0   PIN 7,8,9,10,11;
AEN,IOR_,IOW_,TC,DACK_ PIN 1,13,14,16,17;

BDSEL_,CK1,AO,CS_     PIN 21,20,19,18;
TCNT,CK2              PIN 22,15;

EQUATIONS

BDSEL_ = !(AEN & SA9 & SA8 & SA7 & SA6 & SA5 & SA4 & !SA3);
CK1    = (!BDSEL_ & !IOW_ & !SA2 & SA1 & !SA0);
AO     = (!BDSEL_ & !IOW_ & SA2 & SA1 & SA0) & SA0;
CS_    = !((!BDSEL_ & !IOW_ & SA2 & SA1 & SA0)+(!BDSEL_ & SA2 & !SA1));
TCNT   = (!DACK_ & TC);
CK2    = (!BDSEL_ & !IOR_ & SA2 & SA1 & SA0);

END DECODE_LOGIC

```

Figure 5.9. PAL Equations

PC-AT Digital Output Register (DOR)

There is no equivalent register to the DOR on the 82072. Instead, its functions have been integrated into the enhanced command set. A bit definition of the DOR register, along with its 82072 counterpart is summarized below:

Bits	Definition	82072 Counterpart
0	Drive Select	Automatically generated by the 82072
1	Reserved	
2	Diskette Function Reset	Bit 7 of 82072's DRS register
3	Enable Diskette DMA and Interrupts	Not supported. Requires external logic.
4	Drive A Motor Enable	Automatically generated by the 82072 as a part of the command execution sequence
5	Drive B Motor Enable	Same as 4
6	Reserved;	
7	Reserved;	

PC-AT Digital Input Register (DIR)

The quad density disk drives provide a "Disk Change" signal. This signal informs the system that the diskette in the disk drive has been changed. The "Disk Change" signal sets bit 7 in the DIR register. When set, the PC-AT BIOS performs a routine to determine the type of media inserted in the quad density disk drive.

This feature is not supported by the 82072. It can be implemented by either a.) An external register or b.) Modifying the BIOS to support this feature.

5.3 NEW FEATURES

If it is the objective of the designer to minimize changes to the BIOS, the above design example provides three advantages over the current implementation with minimal effort. It provides the user with:

1. Reduced board space. Only 8 components are required for an 82072 based floppy disk controller board. The current 8272A implementation requires up to 38 components.

2. Reduced power consumption
3. No wait-state interface to the 10 MHz 80286 micro-processor. The 8272A imposes two wait states.

With additional effort, the user can reduce the BIOS and support enhanced features of the 82072 such as:

1. Burst data transfers
2. Implied Seek
3. Power Down mode
4. Relative seek and disk paging
5. 1 Mbps data transfer rate

Burst Data Transfers

During disk transfers between the floppy disk controller and the system, today's floppy disk controllers must be serviced every 13 μ s (MFM mode-500 Kbps). This imposes severe timing constraints on the system. During data transfers, the processor is dedicated to the data transfer process. The 82072 overcomes this timing constraint by providing a 16 byte FIFO. The FIFO improves the data transfer mechanism by:

1. The DMA requests to the processor are minimized by fine tuning the FIFO threshold to match system bus latencies.
2. The host side of the 82072 does not have to wait for the serial side which transfers data at a much slower data rate.

Following reset, the FIFO is enabled, with the FIFO threshold set to a "1". The read data transfers are performed on a byte by byte basis. When write data transfers are performed, the DMA request line is held active until the FIFO is full. The DRQ is activated again when the Serial Unit transfers a byte to the disk drive, and the 82072 reverts back to the byte by byte transfer mode.

To improve system performance, the CONFIGURE command may be issued to select an optimal FIFO threshold. This permits burst data transfers between the 82072 and the system memory. To support burst data transfers, the 8237A-5 mode register has to be reprogrammed to operate in the Demand Transfer Mode (refer to 8237 data sheet for a description of Demand Transfer Mode).

Implied Seek

An added advantage of using the CONFIGURE command is that the user can also enable the Implied Seek mode. When enabled, the 82072 automatically performs the seek and the sense interrupt status commands before executing the data transfer command, thereby reducing the software overhead required to perform data transfers.

Power Down Mode

The 82072 has a power down mode. If there are known periods of time when the disk controller is not being accessed, the 82072 can be programmed to go into the power down mode. This mode is entered by setting Bit 6 of the DSR register. In this mode, the 82072 consumes less than 125 microamps of current.

The present cylinder number is maintained while in Power Down mode. However, the status information is cleared. Prior to issuing a Power Down command, it is highly recommended that the user service all pending interrupts.

Relative Seek

This is a new command which enables the 82072 to access more than 256 tracks. The command can be used to partition the diskette into an unlimited number of pages, each page consisting of 256 tracks. This allows the user to exceed the 256 track limit imposed by the IBM format, while maintaining compatibility. Refer to the 82072 Data Sheet or Chapter 1 for further details.

1 Mbps Data Transfer Rate

The 82072 supports data transfer rates up to 1 Mbps. This enables the 82072 to interface to tomorrow's high-capacity disk drives.

5.4 INTERFACING THE 82072 TO THE IAPX 186 DMA CONTROLLER

Although the 82072 interfaces easily to almost any processor, no processor offers as much of the needed functionality as the 80186 or its 8-bit cousin, the 80188. The 80186 is an 8088 object code compatible processor with integrated DMA controller, timers, interrupt controller, chip-select logic, wait state generator, ready logic and clock generation logic on chip.

Figure 5.10 shows a typical 82072/iAPX 186 microprocessor interface. The data lines of the 82072 are connected through buffers to the 80186 AD0-AD7 lines. The 82072, following a read cycle, does not float its output drivers quickly enough to prevent contention with the 186's generated address for the next cycle. To prevent this data bus contention, the data lines of the 82072 are connected through buffers to the 186's AD0-AD7 lines.

The 80186 DMA controller does not provide an explicit DMA Acknowledge or Terminal Count signal required by the 82072. Instead, the 80186 performs a read or write directly to the DMA requesting device. The DMA Acknowledge ($\overline{\text{DACK}}$) signal can be generated by decoding an address or merely by using one of the 186 generated chip-select lines. The generation of $\overline{\text{DACK}}$ and terminal count (TC) signals are discussed in the following paragraph.

1. $\overline{\text{DACK}}$ Generation

The 80186 can generate chip selects for up to seven peripheral devices (PCS0-PCS7). These chip selects are active for seven contiguous blocks of 128 bytes above a programmed base address. The base address is programmable and can only be a multiple of 1 Kbyte.

Let us consider an example to illustrate the generation of $\overline{\text{DACK}}$ and TC signals. For this example, assume that PCS4 is connected to the chip-select input of the 82072. The chip-select is activated when an access is made to any I/O location between 3F0-3FFH.

CASE 1 $\overline{\text{DACK}}$ Generation Using a Single PCS line

The 80186 activates PCS4 when an access to an I/O location is between 0300H and 03FFH. The $\overline{\text{DACK}}$ signal for the 82072 can be generated by an address decode within the region assigned to PCS4. Let us assume for this example that $\overline{\text{DACK}}$ is activated when an access is made to the I/O location 0380H.

To generate $\overline{\text{DACK}}$ during DMA transfers, the DMA source pointer (for a read transfer) or the destination pointer (for a write transfer) should be initialized to 0380H. The "INC" and "DEC" bits in the 80186 control register must both be set, to prevent the contents of the source or destination pointer from changing.

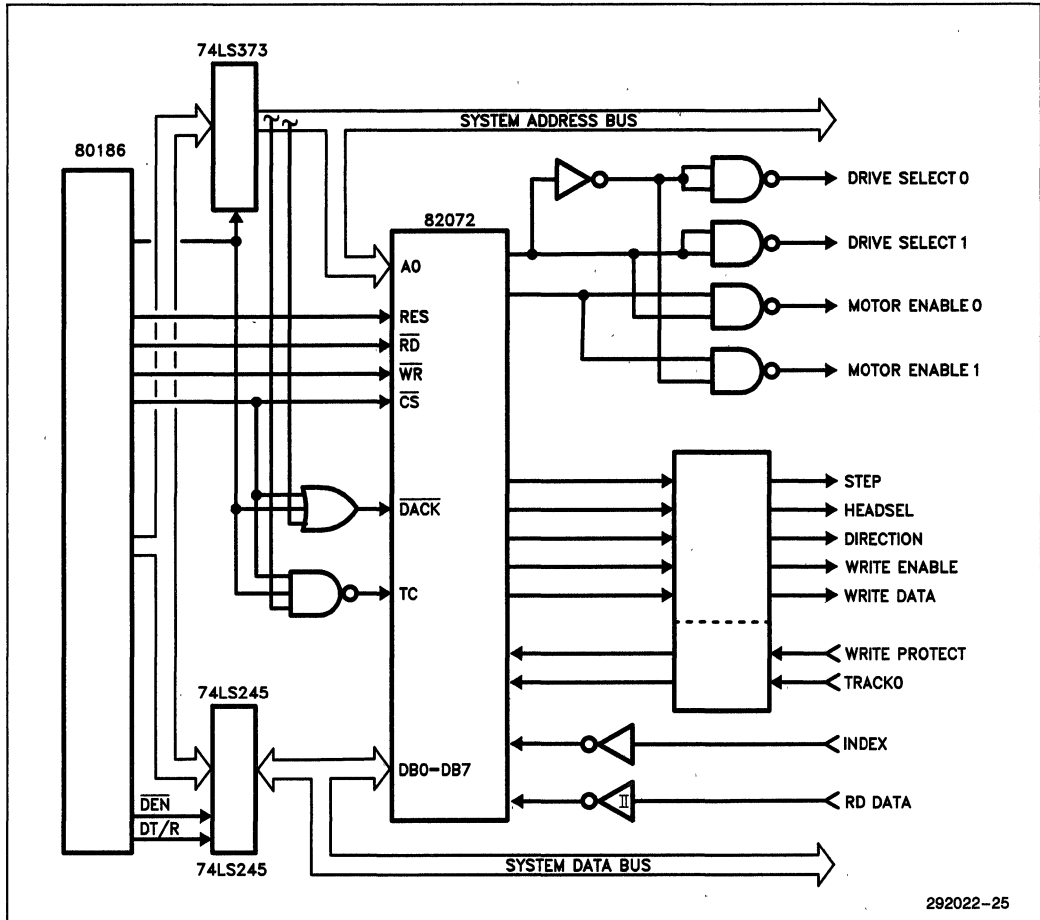


Figure 5.10. INTERFACING to the IAPX 186

“ALE” (Address Latch Enable) must be factored into the $\overline{\text{DACK}}$ generation circuitry as the addresses are not stable when PCS goes active. This could cause glitches at the output of the $\overline{\text{DACK}}$ generation circuitry, as the address lines may change state. The circuitry required to generate $\overline{\text{DACK}}$ is shown in Figure 5.4.

CASE 2 Using an independent chip select line to generate $\overline{\text{DACK}}$.

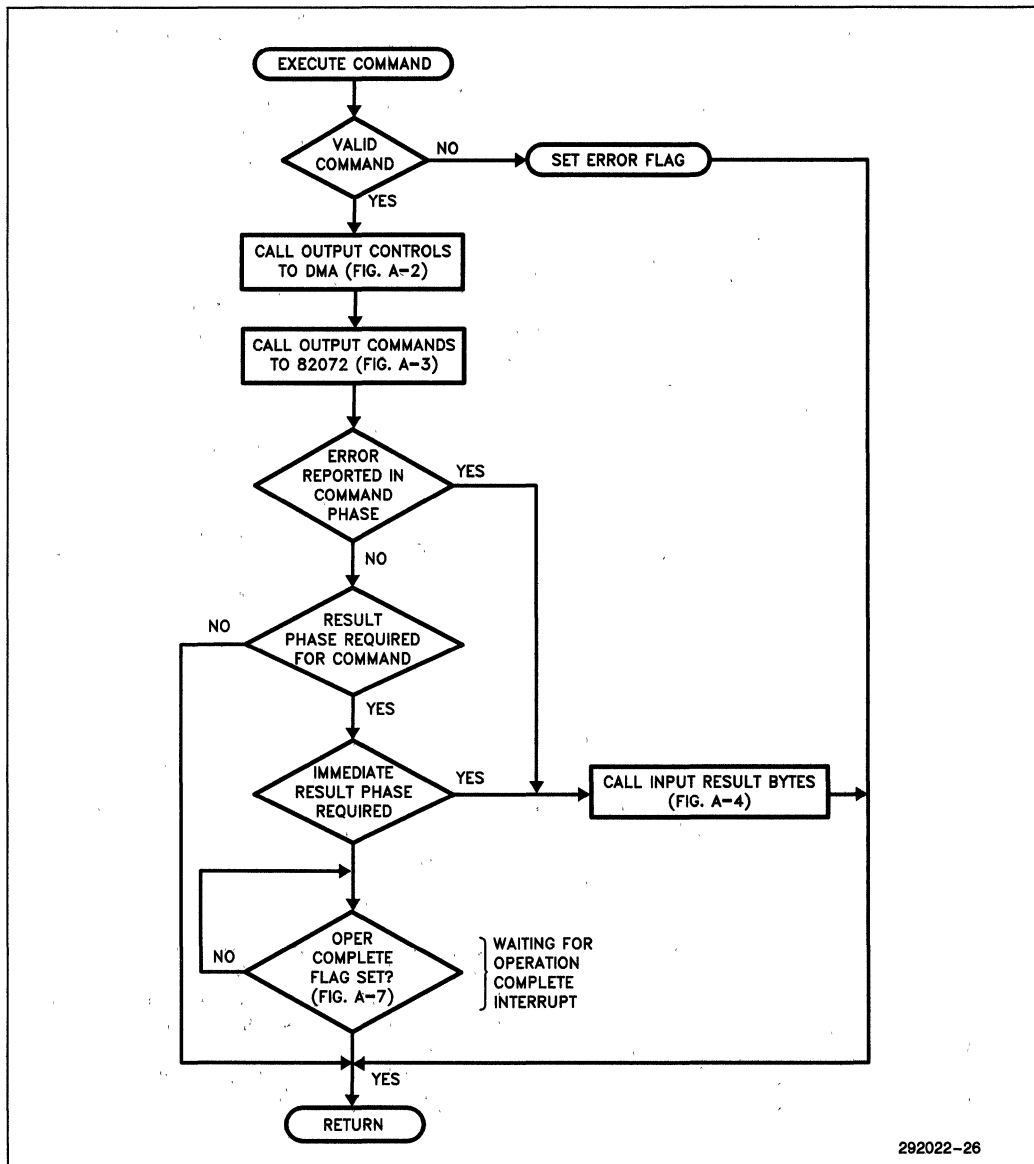
If the system is not using all of the chip select lines provided by the 80186, then the spare chip select line can be used to generate the $\overline{\text{DACK}}$ signal for the 82072. This eliminates the extra decode logic required

when using a common chip select line. In this type of configuration, PCS4 can be used to generate the chip select for the 82072 and PCS5 can be used to generate the $\overline{\text{DACK}}$ signal.

2. Terminal Count Generation

The TC line of the 82072 is driven by a circuit similar to the $\overline{\text{DACK}}$ generation circuit. The TC line is used to inform the 82072 to terminate the data transfer. Another method of generating a terminal count signal is by connecting the $\overline{\text{DACK}}$ signal to one of the 80186 timers and program the timer to output a pulse after the execution of a certain number of DMA cycles.

APPENDIX A 82072 FLOWCHARTS



292022-26

Figure A.1. Generic Command Execution

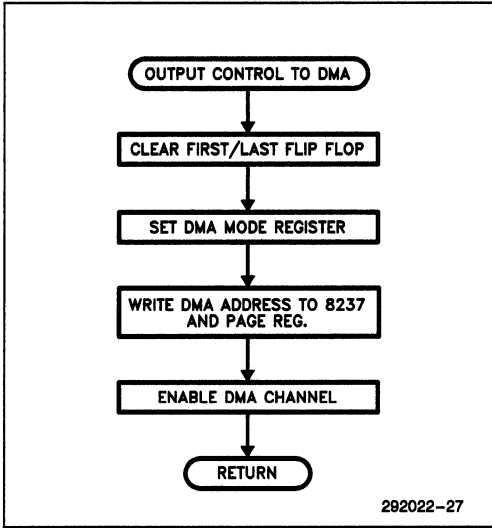


Figure A.2. DMA Initialization

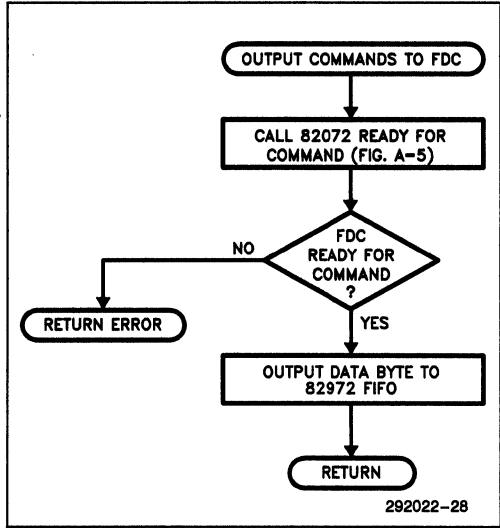


Figure A.3. Output Commands to FDC

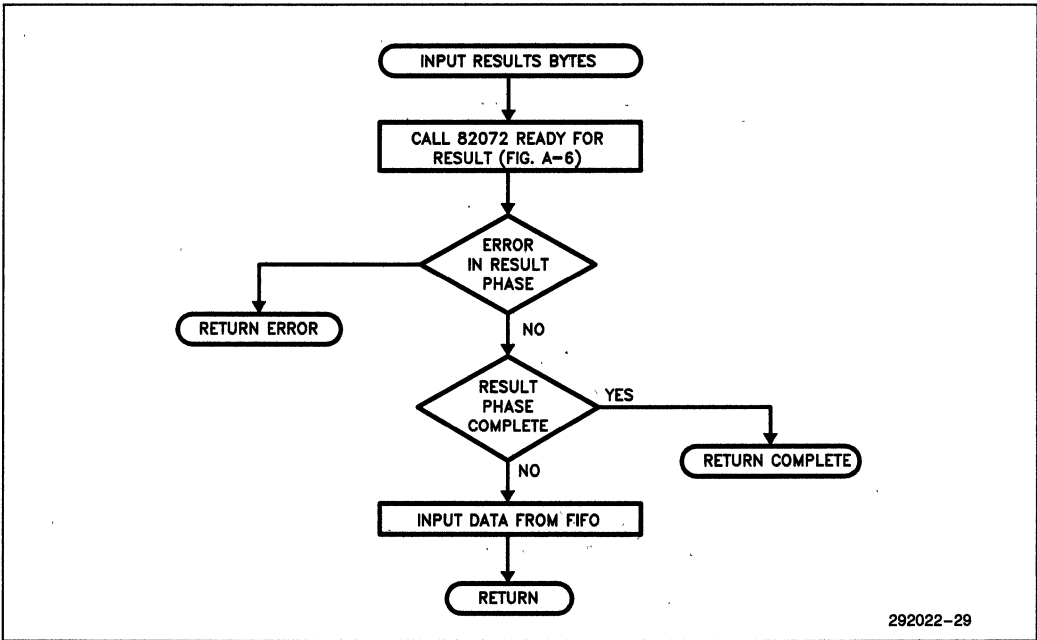


Figure A.4. Input Result Bytes

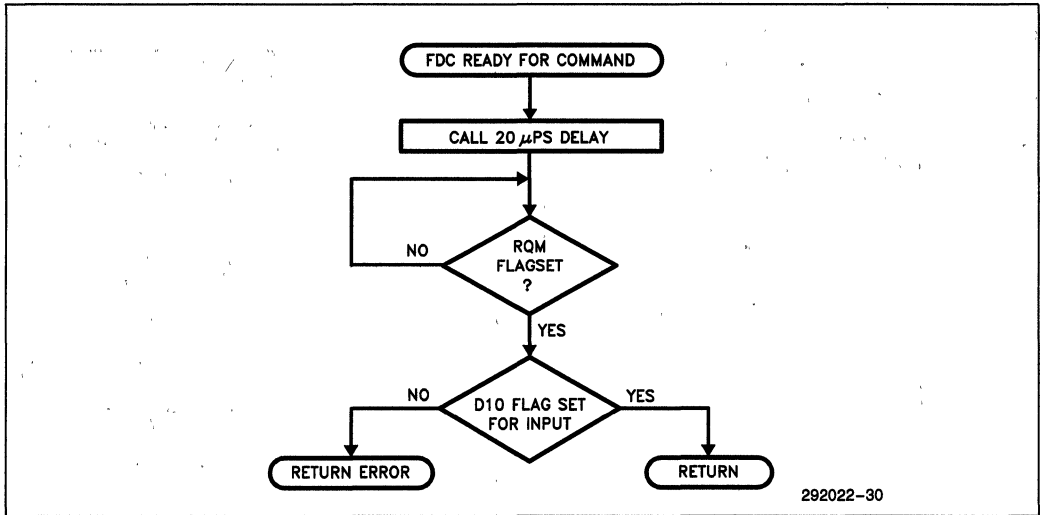


Figure A.5. FDC Ready For Command

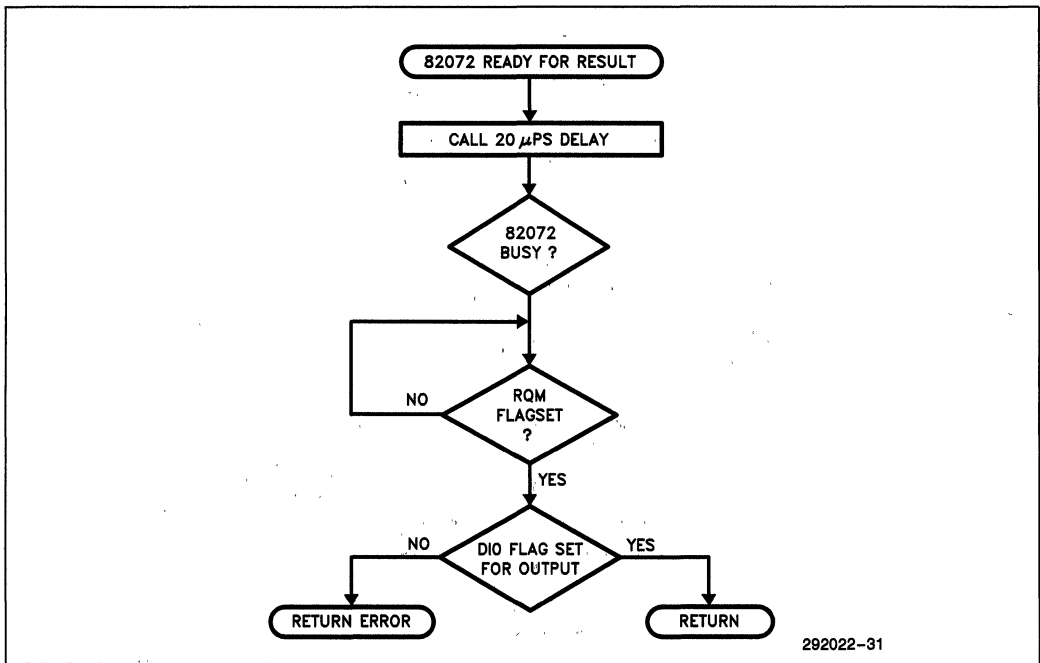


Figure A.6. FDC Ready for Result

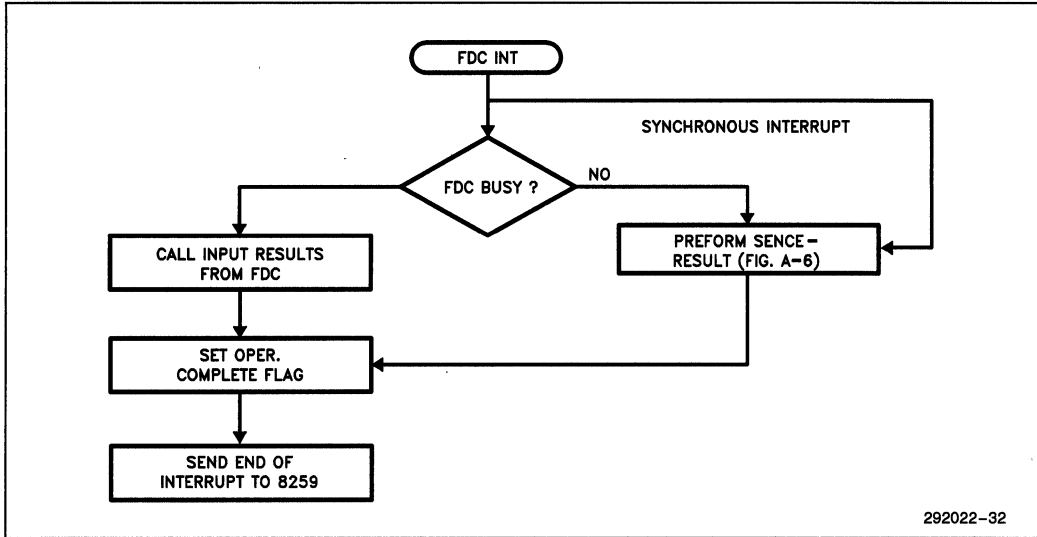


Figure A.7. Interrupt Service Routine

Hard Disk Controllers

4

82064

CHMOS WINCHESTER DISK CONTROLLER WITH ON-CHIP ERROR DETECTION AND CORRECTION

- Controls ST506/ST412 Interface Winchester Disk Drives
- 5 Mbit/sec Data Transfer Rate
- Compatible with All Intel and Most Other Microprocessors
- High Speed Operation
 - “Zero Wait State” Operation with 8 MHz 80286 and 10 MHz 80186/188
 - “One Wait State” Operation with 10 MHz 80286
- Eight High-Level Commands: Restore, Seek, Read Sector, Write Sector, Scan ID, Write Format, Compute Correction, Set Parameter
- Low Power CHMOS III
- On-Chip ECC Unit Automatically Corrects Errors
- 5 or 11-Bit Correction—Span Software Selectable
- Implied Seeks with Read/Write Commands
- Multiple Sector Transfer Capability
- 128, 256, 512 and 1024 Byte Sector Lengths
- Available in 40-Lead Ceramic Dual In-Line, 40-Lead Plastic Dual In-Line, and 44-Lead Plastic Chip Carrier Packages

(See Packaging Spec., Order #231369)

The 82064 Winchester Disk Controller (WDC) with on-chip error detection and correction circuitry interfaces microprocessor systems to 5¼" Winchester disk drives. The 82064 is a CHMOS version of the Western Digital WD2010A-05. It is also socket and software compatible with the Western Digital WD1010A-05 Winchester Disk Controller, and additionally includes on-chip ECC, support for drives with up to 2k tracks, and has an additional control signal which eliminates an external decoder.

The 82064 is fabricated on Intel's advanced CHMOS III technology and is available in 40-lead CERDIP, plastic DIP, and 44-lead plastic leaded chip carrier packages.

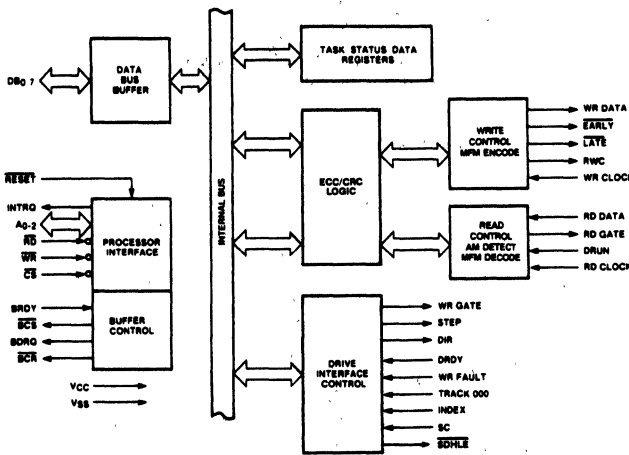
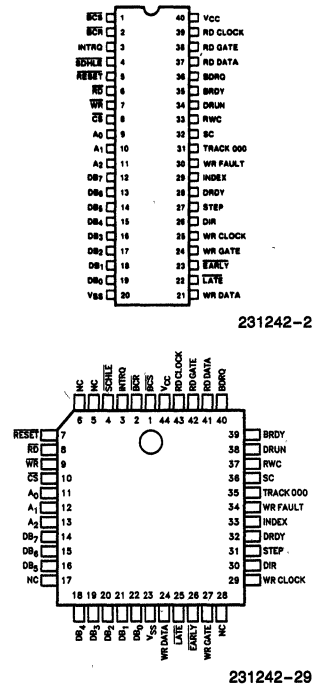


Figure 1. 82064 Block Diagram

231242-1



231242-2

231242-29

Figure 2. 82064 Pinouts

Table 1. Pin Description

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
\overline{BCS}	1	1	O	BUFFER CHIP SELECT: Output used to enable reading or writing of the external sector buffer by the 82064. When low, the host should not be able to drive the 82064 data bus, \overline{RD} , or \overline{WR} lines.
\overline{BCR}	2	2	O	BUFFER COUNTER RESET: Output that is asserted by the 82064 prior to read/write operation. This pin is asserted whenever \overline{BCS} changes state. Used to reset the address counter of the buffer memory.
INTRQ	3	3	O	INTERRUPT REQUEST: Interrupt generated by the 82064 upon command termination. It is reset when the STATUS register is read, or a new command is written to the COMMAND register. Optionally signifies when a data transfer is required on Read Sector commands.
\overline{SDHLE}	4	4	O	\overline{SDHLE} is asserted when the SDH register is written by the host.
\overline{RESET}	5	7	I	RESET: Initializes the controller and clears all status flags. Does not clear the Task Register File.
\overline{RD}	6	8	I/O	READ: Tri-state, bi-directional signal. As an input, \overline{RD} controls the transfer of information from the 82064 registers to the host. \overline{RD} is an output when the 82064 is reading data from the sector buffer (\overline{BCS} low).
\overline{WR}	7	9	I/O	WRITE: Tri-state, bi-directional signal. As an input, \overline{WR} controls the transfer of command or task information into the 82064 registers. \overline{WR} is an output when the 82064 is writing data to the sector buffer (\overline{BCS} low).
\overline{CS}	8	10	I	CHIP SELECT: Enables \overline{RD} and \overline{WR} as inputs for access to the Task Registers. It has no effect once a disk command starts.
A_{0-2}	9-11	11-13	I	ADDRESS: Used to select a register from the task register file.
DB_{0-7}	12-19	14-16 18-22	I/O	DATA BUS: Tri-state, bi-directional 8-bit Data Bus with control determined by \overline{BCS} . When \overline{BCS} is high the microprocessor has full control of the data bus for reading and writing the Task Register File. When \overline{BCS} is low the 82064 controls the data bus to transfer to or from the buffer.
V_{SS}	20	23		Ground
WR DATA	21	24	O	WRITE DATA: Output that shifts out MFM data at a rate determined by Write Clock. Requires an external D flip-flop clocked at 10 MHz. The output has an active pullup and pulldown that can sink 4.8 mA.
LATE	22	25	O	LATE: Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.
EARLY	23	26	O	EARLY: Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
WR GATE	24	27	O	WRITE GATE: High when write data is valid. WR GATE goes low if the WR FAULT input is active. This output is used by the drive to enable head write current.
WR CLOCK	25	29	I	WRITE CLOCK: Clock input used to derive the write data rate. Frequency = 5 MHz for the ST506 interface.
DIR	26	30	O	DIRECTION: High level on this output tells the drive to move the head inward (increasing cylinder number). The state of this signal is determined by the 82064's internal comparison of actual cylinder location vs. desired cylinder.
STEP	27	31	O	STEP: This signal is used to move the drive head to another cylinder at a programmable frequency. Pulse width = 1.6 μ s for a step rate of 3.2 μ s/step, and 8.4 μ s for all other step rates.
DRDY	28	32	I	DRIVE READY: If DRDY from the drive goes low, the command will be terminated.
INDEX	29	33	I	INDEX: Signal from the drive indicating the beginning of a track. It is used by the 82064 during formatting, and for counting retries. Index is edge triggered. Only the rising edge is valid.
WR FAULT	30	34	I	WRITE FAULT: An error input to the 82064 which indicates a fault condition at the drive. If WR FAULT from the drive goes high, the command will be terminated.
TRACK 000	31	35	I	TRACK ZERO: Signal from the drive which indicates that the head is at the outermost cylinder. Used to verify proper completion of a RESTORE command.
SC	32	36	I	SEEK COMPLETE: Signal from the drive indicating to the 82064 that the drive head has settled and that reads or writes can be made. SC is edge triggered. Only the rising edge is valid.
RWC	33	37	O	REDUCED WRITE CURRENT: Signal goes high for all cylinder numbers above the value programmed in the Write Precomp Cylinder register. It is used by the precompensation logic and by the drive to reduce the effects of bit shifting.
DRUN	34	38	I	DATA RUN: This signal informs the 82064 when a field of all ones or all zeroes has been detected in the read data stream by an external one-shot. This indicates the beginning of an ID field. RD GATE is brought high when DRUN is sampled high for 16 clock periods.
BRDY	35	39	I	BUFFER READY: Input used to signal the controller that the buffer is ready for reading (full), or writing (empty), by the host μ P. Only the rising edge indicates the condition.

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
BDRQ	36	40	O	BUFFER DATA REQUEST: Activated during Read or Write commands when a data transfer between the host and the 82064's sector buffer is required. Typically used as a DMA request line.
RD DATA	37	41	I	READ DATA: Single ended input that accepts MFM data from the drive.
RD GATE	38	42	O	READ GATE: Output that is asserted when a search for an address mark is initiated. It remains asserted until the end of the ID or data field.
RD CLOCK	39	43	I	READ CLOCK: Clock input derived from the external data recovery circuits.
V _{CC}	40	44	I	D.C. POWER: +5V.
NC	—	5, 6 17, 28		No Connects

FUNCTIONAL DESCRIPTION

The Intel 82064 CHMOS Winchester Disk Controller (WDC) interfaces microprocessor systems to Winchester disk drives that use the Seagate Technology ST506/ST412 interface. The device translates parallel data from the microprocessor to a 5 Mbit/sec, MFM-encoded serial bit stream. It provides all of the drive control logic and control signals which simplify the design of external data separation and write pre-compensation circuitry. The 82064 is designed to interface to the host processor through an external sector buffer.

On-chip error detection algorithms include the CRC/CCITT and a 32-bit computer generated ECC polynomial. If the ECC code is selected, the 82064 provides three possible error handling techniques if an error is detected during a read operation:

1. Automatically correct the data in the sector buffer, providing the host with good information.
2. Provide the host with the error location and pattern, allowing the host to correct the error.
3. Take no action other than setting the error flag.

The Intel 82064 is an enhanced version of the Western Digital WD2010A-05 Winchester Disk Controller. The 82064 has been completely redesigned for Intel's advanced CHMOS III fabrication process, allowing Intel to offer a high quality, low power device while at the same time maintaining complete compatibility with the WD2010A-05.

Enhancements to the basic design include:

Conversion to a CHMOS III fabrication process for low power consumption.

Improvements to the processor interface to provide high-speed "zero wait state" operation with 10 MHz 80186/188 and 8 MHz 80286. High-speed "one wait state" operation with 10 MHz 80286.

The 82064 is socket and software compatible with the Western Digital WD1010A-05 and WD2010A-05 Winchester Disk Controllers.

INTERNAL ARCHITECTURE

The internal architecture of the 82064 is shown in more detail in Figure 3. It is made up of seven major blocks as described below.

PLA Controller

The PLA interprets commands and provides all control functions. It is synchronized with WR CLOCK.

Magnitude Comparator

An 11-bit magnitude comparator is used to calculate the direction and number of steps needed to move the heads from the present to the desired cylinder position. It compares the cylinder number in the task file to the internal "present position" cylinder number.

A separate high-speed equivalence comparator is used to compare ID field bytes when searching for a sector ID field.

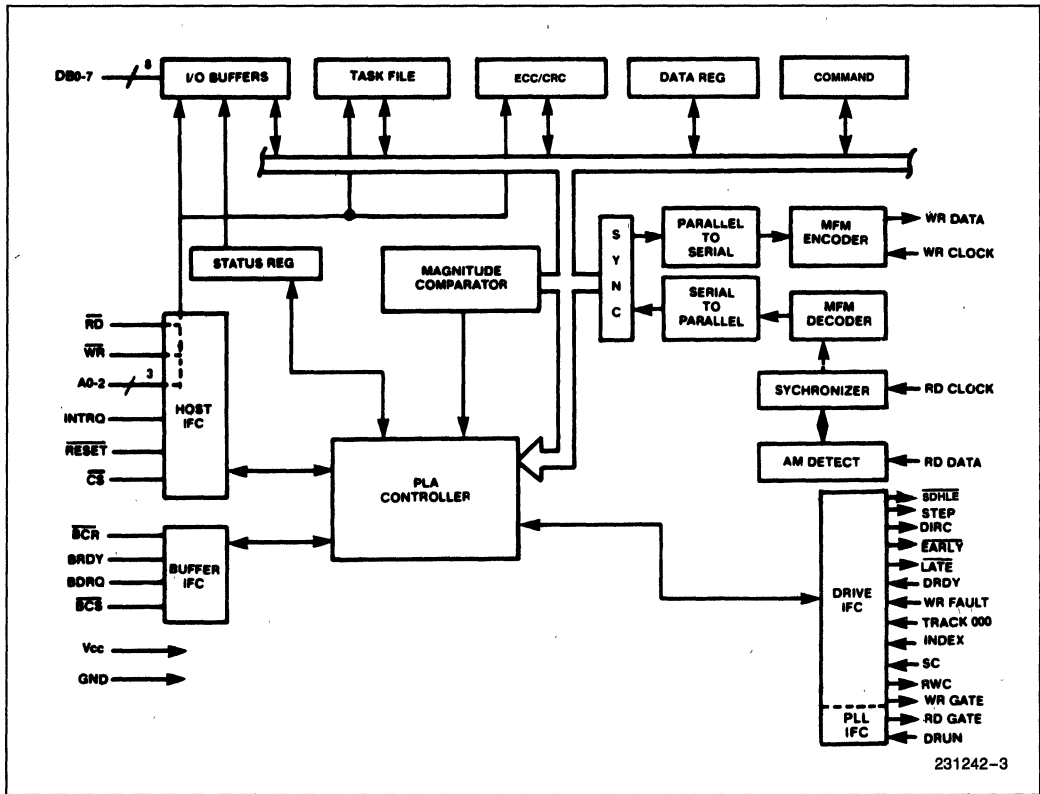


Figure 3. 82064 Detailed Block Diagram

CRC and ECC Generator and Checker

The 82064 provides two options for protecting the integrity of the data field. The data field may have either a CRC (SDH register, bit 7=0), or a 32-bit ECC (SDH register, bit 7=1) appended to it. The ID field is always protected by a CRC.

The CRC mode provides a means of verifying the accuracy of the data read from the disk, but does not attempt to correct it. The CRC generator computes and checks cyclic redundancy check characters that are written and read from the disk after ID and data fields. The polynomial used is:

$$X^{16} + X^{12} + X^5 + 1$$

The CRC register is preset to all one's before computation starts.

If the CRC character generated while reading the data does not equal the one previously written an error exists. If an ID field CRC error occurs the "ID not found" bit in the error register will be set. If a

data field CRC error occurs the "ECC/CRC" bit in the error register will be set.

The ECC mode is only applicable to the data field. It provides the user with the ability to detect and correct errors in the data field automatically. The commands and registers which must be considered when ECC is used are:

1. SDH Register, bit 7 (CRC/ECC)
2. READ SECTOR Command, bit 0 (T)
3. READ SECTOR and WRITE SECTOR Commands, bit 1 (L)
4. COMPUTE CORRECTION Command
5. SET PARAMETER Command
6. STATUS Register, bit 2 - error correction successful
7. STATUS Register, bit 0 - error occurred
8. ERROR Register, bit 6 - uncorrectable error

To enable the ECC mode, bit 7 of the SDH register must be set to one.

Bit 0 (T) of the READ Command controls whether or not error correction is attempted. When T = 0 and an error is detected, the 82064 tries up to 10 times to correct the error. If the error is successfully corrected, bit 2 of the STATUS Register is set. The host can interrogate the status register and detect that an error occurred and was corrected. If the error was not correctable, bit 6 of the ERROR Register is set. If the correction span was set to 5 bits, the host may now execute the SET PARAMETER Command to change the correction span to 11 bits, and attempt the read again. If the error persists, the host can read the data, but it will contain errors.

When T = 1 and an error is detected, no attempt is made to correct it. Bit 0 of the STATUS Register and bit 6 of the ERROR Register are set. The user now has two choices:

1. Ignore the error and make no attempt to correct it.
2. Use the COMPUTE CORRECTION Command to determine the location and pattern of the error, and correct it within the user's program.

When the COMPUTE CORRECTION Command is implemented, it must be done before executing any command which can alter the contents of the ECC Register. The READ SECTOR, WRITE SECTOR, SCAN ID, and FORMAT Commands will alter this register and correction will be impossible. The COMPUTE CORRECTION Command may determine that the error is uncorrectable, at which point the error bits in the STATUS and ERROR Registers are set.

Although ECC generation starts with the first bit of the F8H byte in the data ID field, the actual ECC bytes written will be the same as if the A1H byte was included. The ECC polynomial used is:

$$X^{32} + X^{28} + X^{26} + X^{19} + X^{17} + X^{10} + X^6 + X^2 + 1$$

For automatic error correction, the external sector buffer must be implemented with a static RAM and counter, not with a FIFO.

The SET PARAMETER Command is used to select a 5-bit or 11-bit correction span.

When the L Bit (bit 1) of the READ SECTOR and WRITE SECTOR commands is set to one, they are referred to as READ LONG and WRITE LONG commands. For these commands, no CRC or ECC characters are generated or checked by the 82064. In effect, the data field is extended by 4 bytes which are passed to/from the sector buffer.

With proper use of the WRITE SECTOR, READ LONG, WRITE LONG, and READ SECTOR Commands, a diagnostic routine may be developed to test the accuracy of the error correction process.

MFM ENCODER/DECODER

Encodes and decodes MFM data to be written/read from the drive. The MFM encoder operates from WR CLOCK, a clock having a frequency equal to the bit rate. The MFM decoder operates from RD CLOCK, a bit rate clock generated by the external data separator. RD CLOCK and WR CLOCK need not be synchronous.

The MFM encoder also generates the write precompensation control signals. Depending on the bit pattern of the data, EARLY or LATE may be asserted. External circuitry uses these signals to compensate for drift caused by the influence one bit has over another. More information on the use of the EARLY and LATE control signals can be found in the section which describes the drive interface.

Address Mark (AM) Detection

An address mark is comprised of two unique bytes preceding both the ID field and the data field. The first byte is used for resynchronization. The second byte indicates whether it is an ID field or a data field.

The first byte, A1H, normally has a clock pattern of 0EH; however, one clock pulse has been suppressed, making it 0AH. With this pattern, the AM detector knows it is looking at an address mark. It now examines the next byte to determine if it is an ID or data field. If this byte is 111101XX or 111111XX it is an ID field. Bits 3, 1, and 0 are the high order cylinder number bits. If the second byte is F8H, it is a data field.

Host/Buffer Interface Control

The primary interface between the host processor and the 82064 is an 8-bit bi-directional bus. This bus is used to transmit and receive data for both the 82064 and the sector buffer. The sector buffer consists of a static RAM and counter. Since the 82064 makes the bus active when accessing the sector buffer, a transceiver must be used to isolate the host during this time. Figure 4 illustrates a typical interface with a sector buffer. Whenever the 82064 is not using the sector buffer, the BUFFER CHIP SELECT (\overline{BCS}) is high (disabled). This allows the host access to the 82064's Task Register File and to the sector buffer. A decoder is used to generate \overline{BCS} when A₀₋₂ is '000', an unused address in the 82064. A binary counter is enabled whenever \overline{RD} or \overline{WR} go active. The location within the sector buffer which is addressed by the counter will be accessed. The counter will be incremented by the trailing edge of the \overline{RD} or \overline{WR} . This allows the host to access se-

When INTRQ is asserted, the host is signaled that execution of a command has terminated (either a normal termination or an aborted command). For the READ SECTOR command, interrupts may be programmed to be asserted either at the termination of the command, or when BDRQ is asserted. INTRQ will remain active until the host reads the STATUS register to determine the cause of the termination, or writes a new command into the COMMAND register.

The 82064 asserts \overline{SDHLE} whenever the SDH register is being written. This signal can be used to latch the drive and head select information in an external register for decoding. Figure 5 illustrates one method.

Drive Interface

The drive side of the 82064 WDC requires three sections of external logic. These are the control line buffer/receivers, data separator, and write precompensation. Figure 5 illustrates a drive interface.

The buffer/receivers condition the control lines to be driven down the cable to the drive. The control lines are typically single-ended, resistor terminated, TTL levels. The data lines to and from the drive also require buffering. This is typically done with differential RS-422 drivers. The interface specification for the drive will be found in the drive manufacturer's OEM

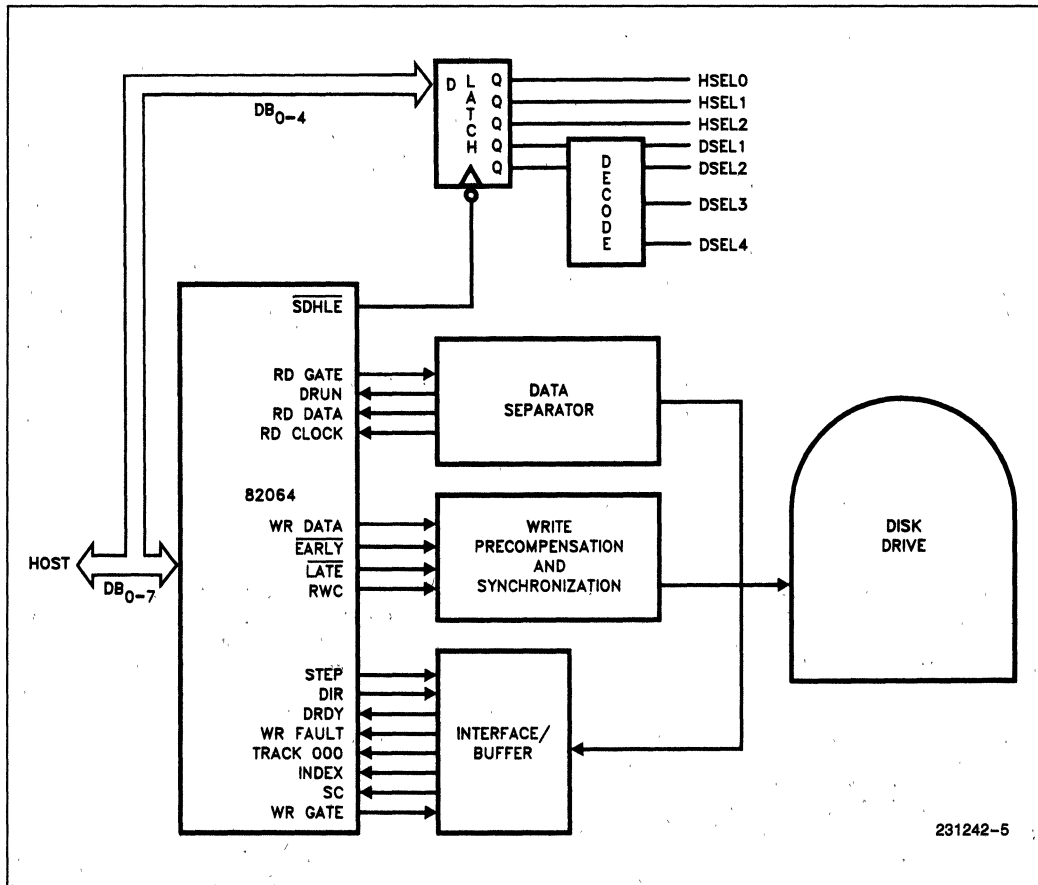


Figure 5. Drive Interface Block Diagram

manual. The 82064 supplies TTL compatible signals, and will interface to most buffer/driver devices.

The data recovery circuits consist of a phase locked loop, data separator, and associated components. The 82064 interacts with the data separator through the DATA RUN (DRUN) and RD GATE signals. A block diagram of a typical data separator circuit is shown in Figure 6. Read data from the drive is presented to the RD DATA input of the 82064, the reference multiplexor, and a retriggerable one shot. The PLL should remain locked to the reference clock.

When any READ or WRITE command is initiated and a search for an address mark begins, the DRUN input is examined. The DRUN one-shot is set for slightly longer than one bit time, allowing it to retrigger constantly on a field of all ones or all zeroes. An internal counter times out to see that DRUN is asserted for two byte times. RD GATE is asserted by the 82064, switching the data separator to lock on to the incoming data stream. If DRUN is deasserted prior to an additional seven byte times, RD GATE is deasserted and the process is repeated. RD GATE will remain asserted until a non-zero, non-address mark byte is detected. The 82064 will then deassert RD GATE for two byte times to allow the PLL to lock back on the reference clock, and start the DRUN search again. If an address mark is detected, RD GATE remains asserted and the command will continue searching for the proper ID field. This sequence is shown in the flow chart in Figure 7.

The write precompensation circuitry is designed to reduce the drift in the data caused by interaction between bits. It is divided into two parts, REDUCED WRITE CURRENT (RWC) and EARLY/LATE writing of bits. A block diagram of a typical write precompensation circuit is shown in Figure 8.

The cylinder in which the RWC line becomes active is controlled by the REDUCE WRITE CURRENT register in the Task Register File. When a cylinder is written which has a cylinder number greater than or equal to the contents of this register, the write current will be reduced. This will decrease the interaction between the bits.

Drift may also be caused by the bit pattern. With certain combinations of ones and zeroes some of the bits can drift far enough apart to be difficult to read without error. This phenomenon can be minimized by using EARLY and LATE as described below. The 82064 examines three bits, the last one written, the one being written, and the next one to be written. From this, it determines whether to assert EARLY or LATE. Since the bit leaving the 82064 has already been written, it is too late to make it early. Therefore, the external delay circuit must be as follows:

EARLY asserted and LATE deasserted = no delay

EARLY deasserted and LATE deasserted = one unit delay (typically 12-15 ns)

EARLY deasserted and LATE asserted = two units delay (typically 24-30 ns)

EARLY and LATE are always active, and should be gated externally by the RWC signal. Figure 8 illustrates one method of using these signals.

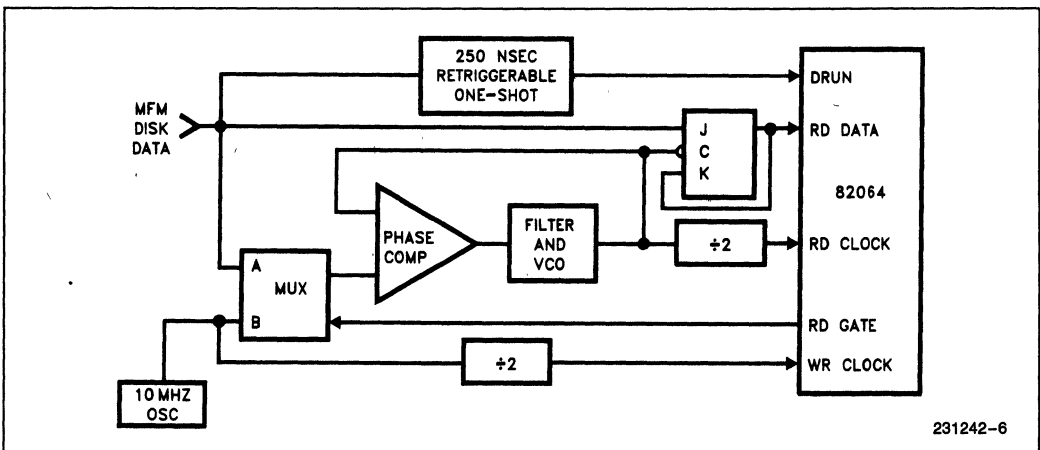
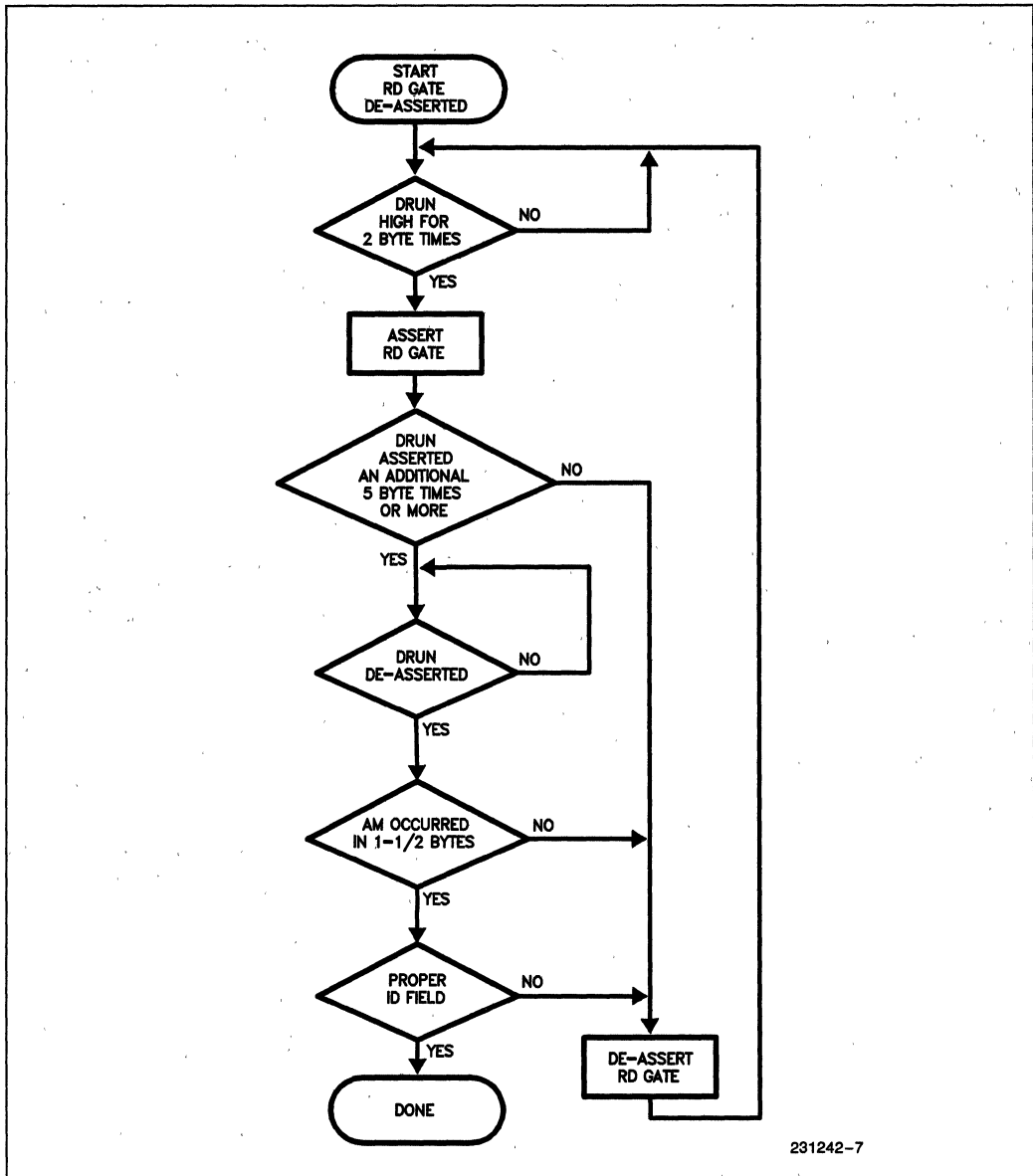


Figure 6. Data Separator Circuit



231242-7

Figure 7. PLL Control Sequence

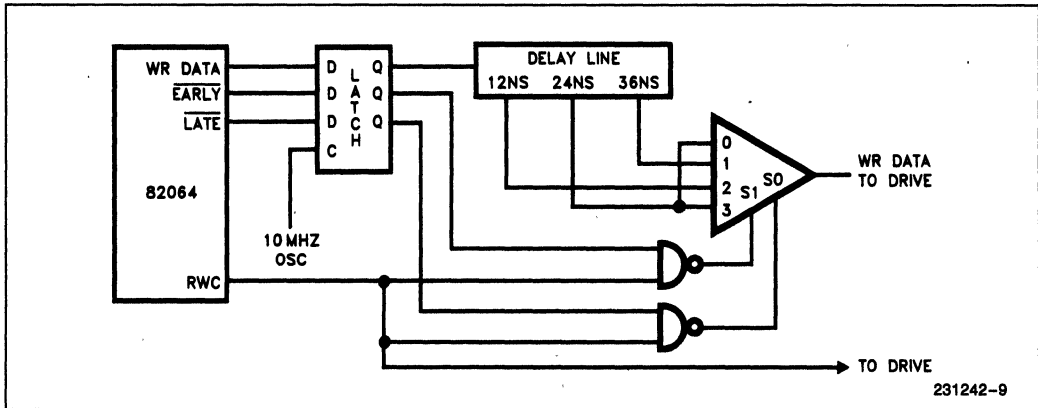


Figure 8. Write Precompensation Circuit

231242-9

TASK REGISTER FILE

The Task Register File is a bank of nine registers used to hold parameter information pertaining to each command, status information, and the command itself. These registers and their addresses are:

A2	A1	A0	READ	WRITE
0	0	0	BUS TRI-STATED	BUS TRI-STATED
0	0	1	ERROR REGISTER	REDUCE WRITE CURRENT
0	1	0	SECTOR COUNT	SECTOR COUNT
0	1	1	SECTOR NUMBER	SECTOR NUMBER
1	0	0	CYLINDER LOW	CYLINDER LOW
1	0	1	CYLINDER HIGH	CYLINDER HIGH
1	1	0	SDH	SDH
1	1	1	STATUS	COMMAND

NOTE:

These registers are not cleared by RESET being asserted.

ERROR REGISTER

This read only register contains specific error information after the termination of a command. The bits are defined as follows:

7	6	5	4	3	2	1	0
BB	CRC/ECC	0	ID	0	AC	TK000	DAM

Bit 7 - Bad Block Detect (BB)

This bit is set when an ID field has been encountered that contains a bad block mark. It is used for bad sector mapping.

Bit 6 - CRC/ECC Data Field Error (CRC/ECC)

When in the CRC mode (SDH register, bit 7 = 0), this bit is set when a CRC error occurs in the data field. When retries are enabled, ten more attempts are made to read the sector correctly. If none of these attempts are successful bit 0 in the STATUS register is also set. If one of the attempts is successful, the CRC/ECC error bit remains set to inform the host that a marginal condition exists; however, bit 0 in the STATUS register is not set.

When in the ECC mode (SDH register, bit 7 = 1), this bit is set when the first non-zero syndrome is detected. When retries are enabled, up to ten attempts are made to correct the error. If the error is successfully corrected, this bit remains set; however, bit 2 of the STATUS register is also set to inform the host that the error has been corrected. If the error is not correctable, the CRC/ECC error bit remains set and bit 0 of the STATUS register is also set.

The data may be read even if uncorrectable errors exist.

NOTE: If the long mode (L) bit is set in the READ or WRITE command, no error checking is performed.

Bit 5 - Reserved

Not used. Forced to zero.

Bit 4 - ID Not Found (ID)

This bit is set to indicate that the correct cylinder, head, sector, or size parameter could not be found, or that a CRC error occurred in the ID field. This bit is set on the first failure and remains set even if the error is recovered on a retry. When recovery is unsuccessful, the Error bit (bit 0) of the STATUS register is also set.

For a SCAN ID command with retries enabled (T = 0), the Error bit in the STATUS register is set after ten unsuccessful attempts have been made to find the correct ID. With retries disabled (T = 1), only two attempts are made before setting the Error bit.

For a READ or WRITE command with retries enabled (T = 0), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan and auto-seek are performed. Then ten more retries are made before setting the Error bit. When retries are disabled (T = 1), only two tries are made. No auto-scan or auto-seek operations are performed.

Bit 3 - Reserved

Not used. Forced to zero.

Bit 2 - Aborted Command (AC)

Command execution is aborted and this bit is set if a command was issued while DRDY is deasserted or WR FAULT is asserted. This bit will also be set if an undefined command is written to the COMMAND register; however, an implied seek will be executed.

Bit 1 - Track 000 Error (TK000)

This bit is set during the execution of a RESTORE command if the TRACK 000 pin has not gone active after the issuance of 2047 step pulses.

Bit 0 - Data Address Mark (DAM) Not Found

This bit is set during the execution of a READ SECTOR command if the DAM is not found following the proper sector ID.

REDUCE WRITE CURRENT REGISTER

This register is used to define the cylinder number where the RWC output (Pin 33) is asserted.

7	6	5	4	3	2	1	0
CYLINDER NUMBER ÷ 4							

The value (00-FFH) loaded into this cylinder is internally multiplied by four to specify the actual cylinder where RWC is asserted. Thus a value of 01H will cause RWC to be asserted on cylinder 04H, 02H on cylinder 08H, . . . , 9CH on cylinder 270H, 9DH on cylinder 274H, and so on. RWC will be asserted when the present cylinder is greater than or equal to four times the value of this register. For example, the ST506 interface requires precomp on cylinder 80H and above. Therefore, the REDUCE WRITE CURRENT register should be loaded with 20H.

A value of FFH causes RWC to remain deasserted, regardless of the actual cylinder number.

SECTOR COUNT REGISTER

This register is used to define the number of sectors that need to be transferred to the buffer during a READ MULTIPLE SECTOR or WRITE MULTIPLE SECTOR command.

7	6	5	4	3	2	1	0
NUMBER OF SECTORS							

The value contained in the register is decremented after each sector is transferred to/from the sector buffer. A zero represents a 256 sector transfer, a one a one sector transfer, etc. This register is a "don't care" when single sector commands are specified.

SECTOR NUMBER REGISTER

This register holds the sector number of the desired sector.

7	6	5	4	3	2	1	0
SECTOR NUMBER							

For a multiple sector command, it specifies the first sector to be transferred. It is incremented after each sector is transferred to/from the sector buffer. The SECTOR NUMBER register may contain any value from 0 to 255.

The SECTOR NUMBER register is also used to program the Gap 1 and Gap 3 lengths to be used when formatting a disk. See the WRITE FORMAT command description for further explanation.

CYLINDER NUMBER LOW REGISTER

This register holds the lower byte of the desired cylinder number.

7	6	5	4	3	2	1	0
LS BYTE OF CYL. NUMBER							

It is used with the CYLINDER NUMBER HIGH register to specify the desired cylinder number over a range of 0 to 2047.

CYLINDER NUMBER HIGH REGISTER

This register holds the three most significant bits of the desired cylinder number.

7	6	5	4	3	2	1	0
x	x	x	x	x	#	#	#

The CYLINDER NUMBER LOW/HIGH register pair determine where the R/W heads are to be positioned. The host writes the desired cylinder number into these registers. Internal to the 82064 is another pair of registers that hold the present head location. When any command other than a RESTORE is executed, the internal head location registers are compared to the CYLINDER NUMBER registers to determine how many cylinders to move the heads and in what direction.

The internal head location registers are updated to equal the CYLINDER NUMBER registers after the completion of the seek.

When a RESTORE command is executed, the internal head location registers are reset to zero while DIR and STEP move the heads to track zero.

SECTOR/DRIVE/HEAD (SDH) REGISTER

The SDH register contains the desired sector size, drive number, and head parameters. The format is shown in Figure 9. The EXT bit (bit 7) is used to select between the CRC or ECC mode. When bit 7 = 1 the ECC mode is selected for the data field. When bit 7 = 0 the CRC mode is selected.

The SDH byte written in the ID field of the disk by the FORMAT command is different than the SDH register contents. The recorded SDH byte does not have

the drive number recorded, but does have the bad block mark written. The format of the SDH byte written on the disk is:

7	6	5	4	3	2	1	0
BAD B.	SIZE		0	0	HEAD		

STATUS REGISTER

The status register is used to inform the host of certain events performed by the 82064, as well as reporting status from the drive control lines. Reading the STATUS register deasserts INTRQ. The format is:

7	6	5	4	3	2	1	0
BUSY	READY	WF	SC	DRQ	DWC	CIP	ERR

Bit 7 - Busy

This bit is asserted when a command is written into the COMMAND register and, except for the READ command, is deasserted at the end of the command. When executing a READ command, Busy will be deasserted when the sector buffer is full. Commands should not be loaded into the COMMAND register when Busy is set. When the Busy bit is set, no other bits in the STATUS or ERROR registers are valid.

Bit 6 - Ready

This bit reflects the status of DRDY (pin 28). When this bit equals zero, the command is aborted and the status of this bit is latched.

Bit 5 - Write Fault (WF)

This bit reflects the status of WR FAULT (pin 30). When this bit equals one the command is aborted, INTRQ is asserted, and the status of this bit is latched.

Bit 4 - Seek Complete (SC)

This bit reflects the status of SC (pin 32). When a seek or implied seek has been initiated by a command, execution of the command pauses until the seek is complete. This bit is latched after an aborted command error.

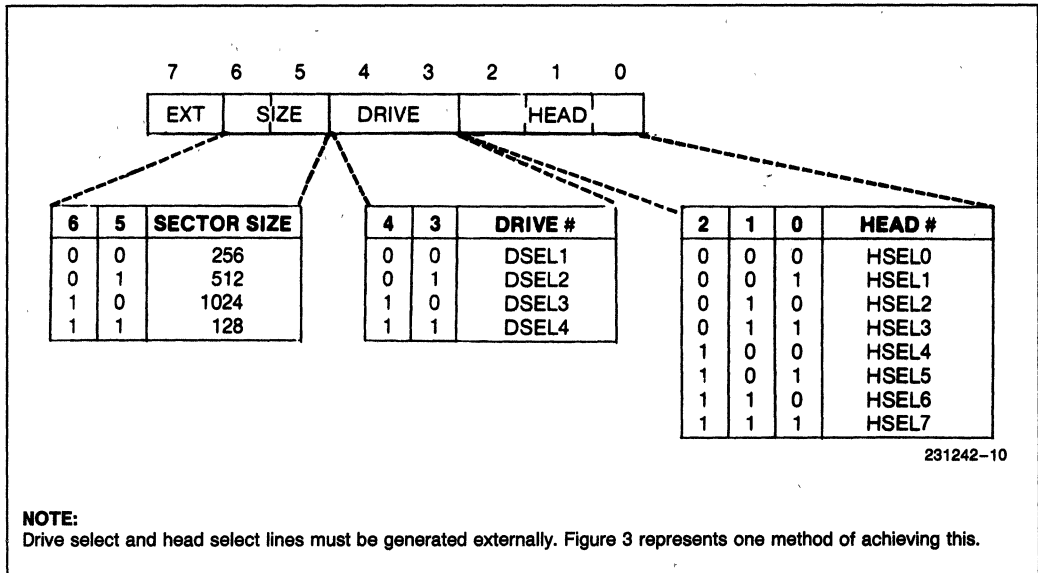


Figure 9. SDH Register Format

Bit 3 - Data Request (DRQ)

The DRQ bit reflects the status of BDRQ (pin 36). It is asserted when the sector buffer must be written into or read from. DRQ and BDRQ remain asserted until BRDY indicates that the sector buffer has been filled or emptied, depending upon the command. BDRQ can be used for DMA interfacing, while DRQ is used in a programmed I/O environment.

Bit 2 - Data Was Corrected (DWC)

When set, this bit indicates that an ECC error has been detected during a read operation, and that the data in the sector buffer has been corrected. This provides the user with an indication that there may be a marginal condition within the drive before the errors become uncorrectable. This bit is forced to zero when not in the ECC mode.

Bit 1 - Command In Progress (CIP)

When this bit is set a command is being executed and a new command should not be loaded. Although a command is being executed, the sector buffer is still available for access by the host. When the 82064 is no longer Busy (bit 7 = 0) the STATUS register can be read. If other registers are read while CIP is set the contents of the STATUS register will be returned.

Bit 0 - Error

This bit is set whenever any bits in the ERROR register are set. It is the logical 'or' of the bits in the ERROR register and may be used by the host processor to quickly check for nonrecoverable errors. The host must read the ERROR register to determine what type of error occurred. This bit is reset when a new command is written into the COMMAND register.

COMMAND REGISTER

The command to be executed is written into this write-only register:

7	6	5	4	3	2	1	0
COMMAND							

The command sets Busy and CIP, and begins to execute as soon as it is written into this register. Therefore, all necessary information should be loaded into the Task Register File prior to entering the command. Any attempt to write a register will be ignored until command execution has terminated, as indicated by the CIP bit being cleared. INTRQ is deasserted when the COMMAND register is written.

COMMAND	7	6	5	4	3	2	1	0
RESTORE	0	0	0	1	R3	R2	R1	R0
SEEK	0	1	1	1	R3	R2	R1	R0
READ SECTOR	0	0	1	0	I	M	L	T
WRITE SECTOR	0	0	1	1	0	M	L	T
SCAN ID	0	1	0	0	0	0	0	T
WRITE FORMAT	0	1	0	1	0	0	0	0
COMPUTE CORRECTION	0	0	0	0	1	0	0	0
SET PARAMETER	0	0	0	0	0	0	0	S

R₃₋₀ = Stepping Rate Field

For 5 MHz WR CLOCK:

R ₃₋₀ = 0000	35 μs
0001	0.5 ms
0010	1.0 ms
0011	1.5 ms
0100	2.0 ms
0101	2.5 ms
0110	3.0 ms
0111	3.5 ms
1000	4.0 ms
1001	4.5 ms
1010	5.0 ms
1011	5.5 ms
1100	6.0 ms
1101	6.5 ms
1110	3.2 μs
1111	16 μs

I = Interrupt Control

I = 0 INTRQ occurs with BDRQ/DRQ indicating the sector buffer is full. Valid only when M = 0.

I = 1 INTRQ occurs when the command is completed and the host has read the sector buffer.

M = Multiple Sector Flag

M = 0 Transfer one sector. Ignore the SECTOR COUNT register.

M = 1 Transfer multiple sectors.

L = Long Mode

L = 0 Normal mode. Normal CRC or ECC functions are performed.

L = 1 Long mode. No CRC or ECC bytes are developed or error checking performed on the data field. The 82064 appends the four additional bytes supplied by the host or disk to the data field.

T = Retry Enable

T = 0 Enable retries.

T = 1 Disable retries.

S = Error Correction Span

S = 0 5-bit span.

S = 1 11-bit span.

RESTORE COMMAND

The RESTORE command is used to position the R/W heads over track zero. It is usually issued by the host when a drive has just been turned on. The 82064 forces an auto-restore when a FORMAT command has been issued following a drive number change.

The actual step rate used for the RESTORE command is determined by the seek complete time. A step pulse is issued and the 82064 waits for a rising edge on the SC line before issuing the next pulse. If the rising edge of SC has not occurred within ten revolutions (INDEX pulses) the 82064 switches to sensing the level of SC. If after 2047 step pulses the TRACK 000 line does not go active the 82064 will set the TRACK 000 bit in the ERROR register, assert INTRQ, and terminate execution of the command. An interrupt will also occur if WR FAULT is asserted on DRDY is deasserted at any time during execution.

The rate field specified (R₃₋₀) is stored in an internal register for future use in commands with implied seeks.

A flowchart of the RESTORE command is shown in Figure 10.

SEEK COMMAND

The SEEK command can be used for overlapping seeks on multiple drives. The step rate used is taken from the Rate Field of the command, and is stored in an internal register for future use by those commands with implied seek capability.

The direction and number of step pulses needed are calculated by comparing the contents of the CYLINDER NUMBER registers in the Task Register File to the present cylinder position stored internally. After all the step pulses have been issued the present cylinder position is updated, INTRQ is asserted, and the command terminated.

If DRDY is deasserted or WR FAULT is asserted during the execution of the command, INTRQ is asserted and the command aborts setting the AC bit in the ERROR register.

If an implied seek is performed, the step rate indicated by the rate field is used for all but the last step pulse. On the last pulse, the command execution continues until the rising edge of SC is detected. If 10 INDEX pulses are received without a rising edge of SC, the 82064 will switch to sensing the level of SC.

A flowchart of the SEEK command flow is shown in Figure 11.

READ SECTOR

The READ SECTOR command is used to transfer one or more sectors of data from the disk to the sector buffer. Upon receipt of the command, the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against the internal cylinder position register to see if they are equal. If not, the direction and number of steps calculation takes place, and a seek is initiated. As stated in the description of the SEEK command, if an implied seek occurs, the step rate specified by the rate field is used for all but the last step pulse. On the last step pulse the seek continues until the rising edge of SC is detected.

If the 82064 detects a change in the drive number since the last command, an auto-scan ID is performed. This updates the internal cylinder position register to reflect the current drive before the seek begins.

After the 82064 senses SC (with or without an implied seek) it must find an ID field with the correct cylinder number, head, sector size, and CRC. If retries are enabled (T = 0), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan ID and auto-seek are performed. Then ten more retries are attempted before setting the ID Not Found error bit. When retries are disabled (T = 1) only two tries are made. No auto-scan or auto-seek operations are performed.

When the data address mark (DAM) is found, the 82064 is ready to transfer data into the sector buffer. When the disk has filled the sector buffer, the 82064 asserts BDRQ and DRQ and then checks the I flag. If I = 0, INTRQ is asserted, signaling the host to read the contents of the sector buffer. If I = 1, INTRQ occurs after the host has read the sector buffer and the command has terminated. If after successfully reading the ID field, the DAM is not found the DAM Not Found bit in the ERROR register is set.

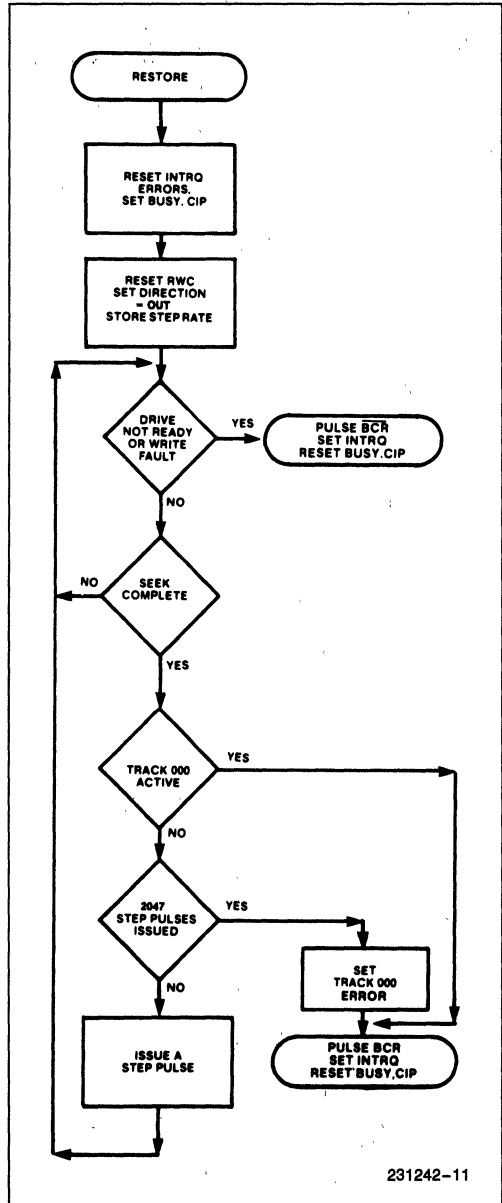
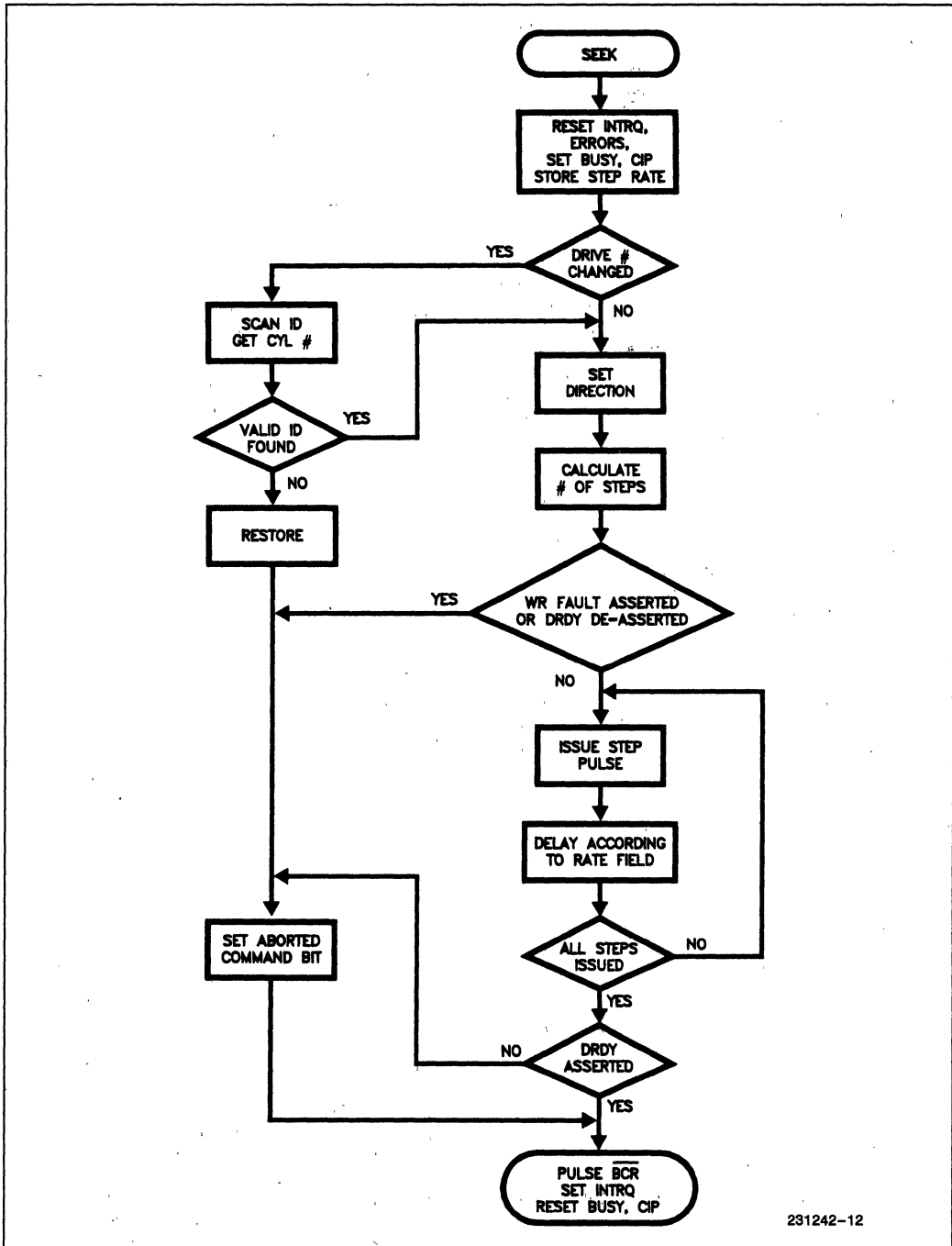


Figure 10. Restore Command Flow

231242-11



231242-12

Figure 11. Seek Command Flow

An optional M flag can be set for multiple sector transfers. When $M = 0$, one sector is transferred and the SECTOR COUNT register is ignored. When $M = 1$, multiple sectors are transferred. After each sector is transferred, the 82064 decrements the SECTOR COUNT register and increments the SECTOR NUMBER register. The next logical sector is transferred regardless of any interleave. Sectors are numbered during the FORMAT command by a byte in the ID field.

For the 82064 to make multiple sector transfers to the sector buffer, the BRDY signal must be toggled from low to high for each sector. The transfers continue until the SECTOR COUNT register equal zero. If the SECTOR COUNT is not zero (indicating more sectors remain to be read), and the sector buffer is full, BDRQ will be asserted and the host must unload the sector buffer. Once this occurs, the sector buffer is free to accept the next sector.

WR FAULT and DRDY are monitored throughout the command execution. If WR FAULT is asserted or DRDY is deasserted, the command will terminate and the Aborted Command bit in the ERROR register will be set. For a description of the error checking procedure on the data field see the explanation in the section entitled "CRC and ECC Generator and Checker."

Both the READ and WRITE commands feature a "simulated completion" to ease programming. BDRQ, DRQ, and INTRQ are generated in a normal manner upon detection of an error condition. This allows the same program flow for successful or unsuccessful completion of a command.

In summary then, the READ SECTOR operation is as follows:

When $M = 0$ (Single Sector Read)

1. HOST: Sets up parameters. Issues READ SECTOR command.
2. 82064: Asserts \overline{BCR} .
3. 82064: Finds sector specified. Asserts \overline{BCR} and \overline{BCS} . Transfers data to sector buffer.
4. 82064: Asserts \overline{BCR} . Deasserts \overline{BCS} .
5. 82064: Asserts BDRQ and DRQ.
6. 82064: If $I = 1$ then go to 9.
7. HOST: Read contents of sector buffer.
8. 82064: Wait for BRDY, then assert INTRQ. End.
9. 82064: Assert INTRQ.
10. HOST: Read contents of sector buffer. End.

When $M = 1$ (Multiple Sector Read)

1. HOST: Sets up parameters. Issues READ SECTOR command.
2. 82064: Asserts \overline{BCR} .
3. 82064: Finds sector specified. Asserts \overline{BCR} and \overline{BCS} . Transfers data to sector buffer.
4. 82064: Asserts \overline{BCR} . Deasserts \overline{BCS} .
5. 82064: Asserts BDRQ and DRQ.
6. HOST: Reads contents of sector buffer.
7. SECTOR BUFFER: Indicates data has been transferred by asserting BRDY.
8. 82064: When BRDY is asserted, decrement SECTOR COUNT, increment SECTOR NUMBER. If SECTOR COUNT = 0, go to 10.
9. 82064: Go to 2.
10. 82064: Assert INTRQ.

A flowchart of the READ SECTOR command is shown in Figure 12.

WRITE SECTOR

The WRITE SECTOR command is used to write one or more sectors of data from the sector buffer to the disk. Upon receipt of the command, the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against the internal cylinder position register to see if they are equal. If not, the direction and number of steps calculation takes place, and a seek is initiated. As stated in the description of the SEEK command, if an implied seek occurs, the step rate specified by the rate field is used for all but the last step pulse. On the last step pulse the seek continues until the rising edge of SC is detected.

If the 82064 detects a change in the drive number since the last command, an auto-scan ID is performed. This updates the internal cylinder position register to reflect the current drive before the seek begins.

After the 82064 senses SC (with or without an implied seek) BDRQ and DRQ are asserted and the host begins filling the sector buffer with data. When BRDY is asserted, a search for the ID field with the correct cylinder number, head, sector size, and CRC is initiated. If retries are enabled ($T = 0$), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan ID and auto-seek are performed. Then ten more retries are attempted before setting the ID Not Found error bit. When retries are disabled ($T = 1$) only two tries are made. No auto-scan or auto-seek operations are performed.

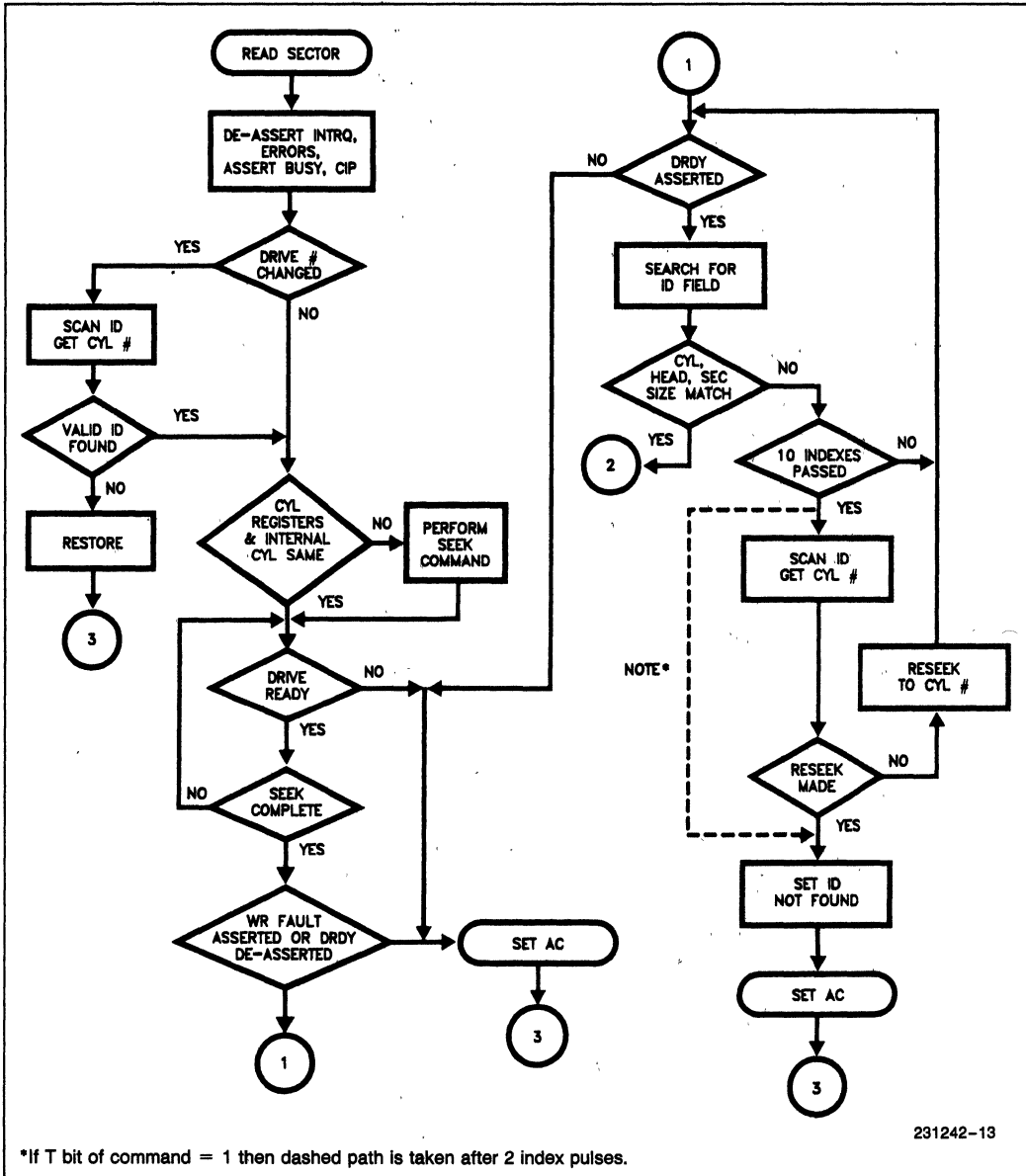


Figure 12a. Read Sector Command Flow

When the correct ID is found, WR GATE is asserted and data is written to the disk. When the CRC/ECC bit (SDH Register, bit 7) is zero, the 82064 generates a two byte CRC character to be appended to the data. When the CRC/ECC bit is one, four ECC bytes replace the CRC character. When $L = 1$, the polynomial generator is inhibited and neither CRC or ECC bytes are generated. Instead four bytes of data supplied by the host are written.

During a WRITE MULTIPLE SECTOR command ($M = 1$), the SECTOR NUMBER register is incremented and the SECTOR COUNT register is decremented. If BRDY is asserted after the first sector is read from the sector buffer, the 82064 continues to read data from the sector buffer for the next sector. If BRDY is deasserted, the 82064 asserts BDRQ and waits for the host to place more data in the sector buffer.

In summary then, the WRITE SECTOR operation is as follows:

When $M = 0,1$

1. HOST: Sets up parameters. Issues WRITE SECTOR command.
2. 82064: Asserts BDRQ and DRQ.
3. HOST: Loads sector buffer with data.
4. 82064: Waits for rising edge of BRDY.
5. 82064: Finds specified ID field. Writes sector to disk.
6. 82064: If $M = 0$, asserts INTRQ. End.
7. 82064: Increments SECTOR NUMBER. Decrements SECTOR COUNT.
8. 82064: IF SECTOR COUNT = 0, assert INTRQ. End.
9. 82064: Go to 2.

A flowchart of the WRITE SECTOR command is shown in Figure 13.

SCAN ID

The SCAN ID command is used to update the SDH, SECTOR NUMBER, and CYLINDER NUMBER LOW/HIGH registers.

After the command is loaded, the SC line is sampled until it is valid. The DRDY and WR FAULT lines are also monitored throughout execution of the command. If a fault occurs the command is aborted and the appropriate error bits are set. When the first ID field is found, the ID information is loaded into the SDH, SECTOR NUMBER, and CYLINDER NUMBER registers. The internal cylinder position register is also updated. If this is an auto-scan caused by a

change in drive numbers, only the internal position register is updated. If a bad block is detected, the BAD BLOCK bit will also be set.

If an ID field is not found, or if a CRC error occurs, and if retries are enabled ($T = 0$), ten attempts are made to read it. If retries are disabled ($T = 1$), only two tries are made. There is no auto-seek in this command and the sector buffer is not disturbed.

A flowchart of the SCAN ID command is shown in Figure 14.

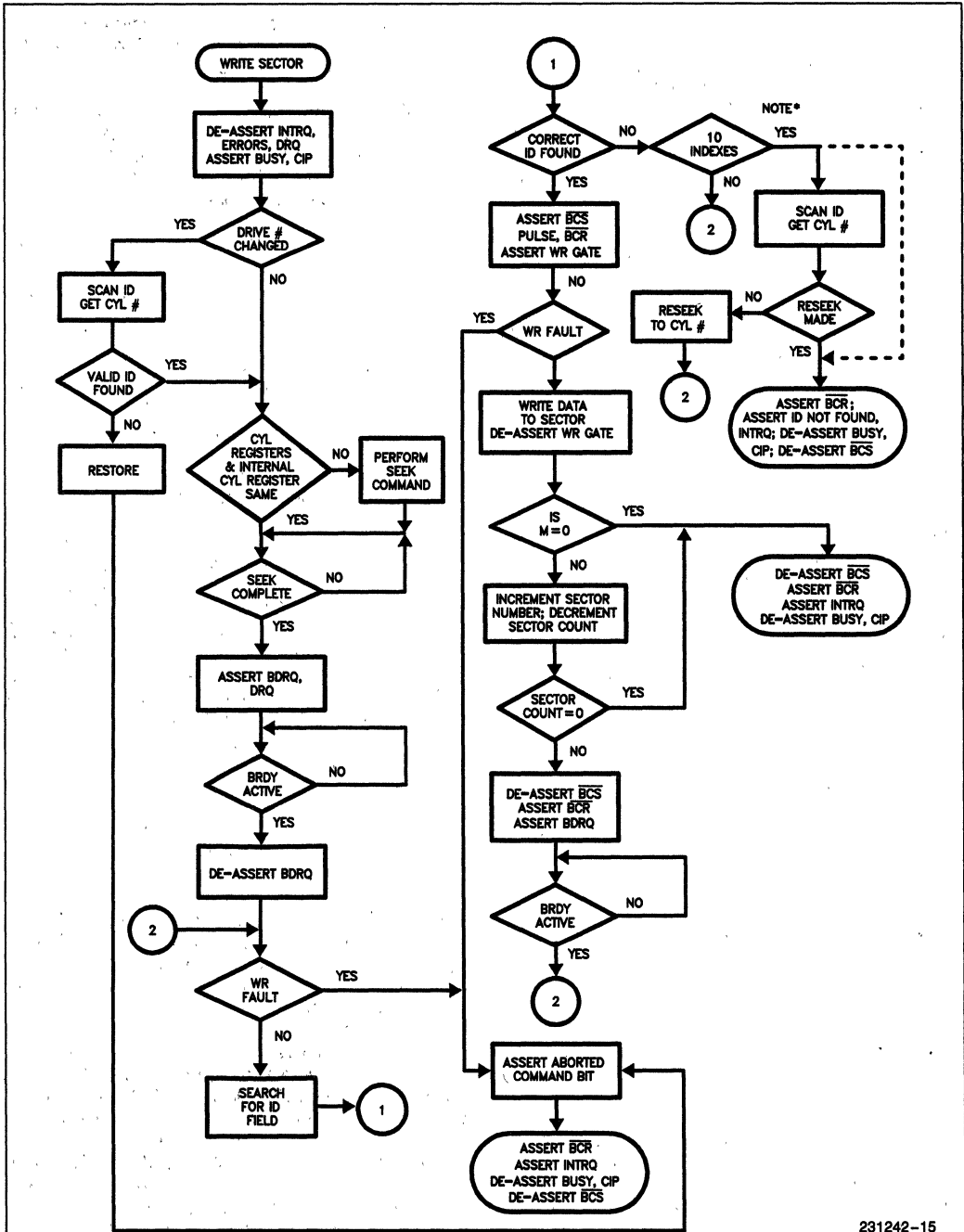
WRITE FORMAT

The WRITE FORMAT command is used to format one track using information in the Task Register File and the sector buffer. During execution of this command, the sector buffer is used for additional parameter information instead of data. Shown in Figure 15 is the contents of a sector buffer for a 32 sector track with an interleave factor of two.

Each sector requires a two byte sequence. The first byte designates whether a bad block mark is to be recorded in the sector's ID field. An 00H is normal; an 80H indicates a bad block mark for that sector. In the example of Figure 15, sector 04 will get a bad block mark recorded. The second byte indicates the logical sector number to be recorded. This allows sectors to be recorded with any interleave factor desired. The remaining memory in the sector buffer may be filled with any value; its only purpose is to generate a BRDY to tell the 82064 to begin formatting the track.

If the drive number has been changed since the last command, an auto-restore is initiated, positioning the heads to track 000. The internal cylinder position register is set to zero and the heads seek to the track specified in the Task Register File CYLINDER NUMBER register. This prevents an ID Not Found error from occurring due to an incompatible format, or the track having been erased. A normal implied seek is also in effect for this command.

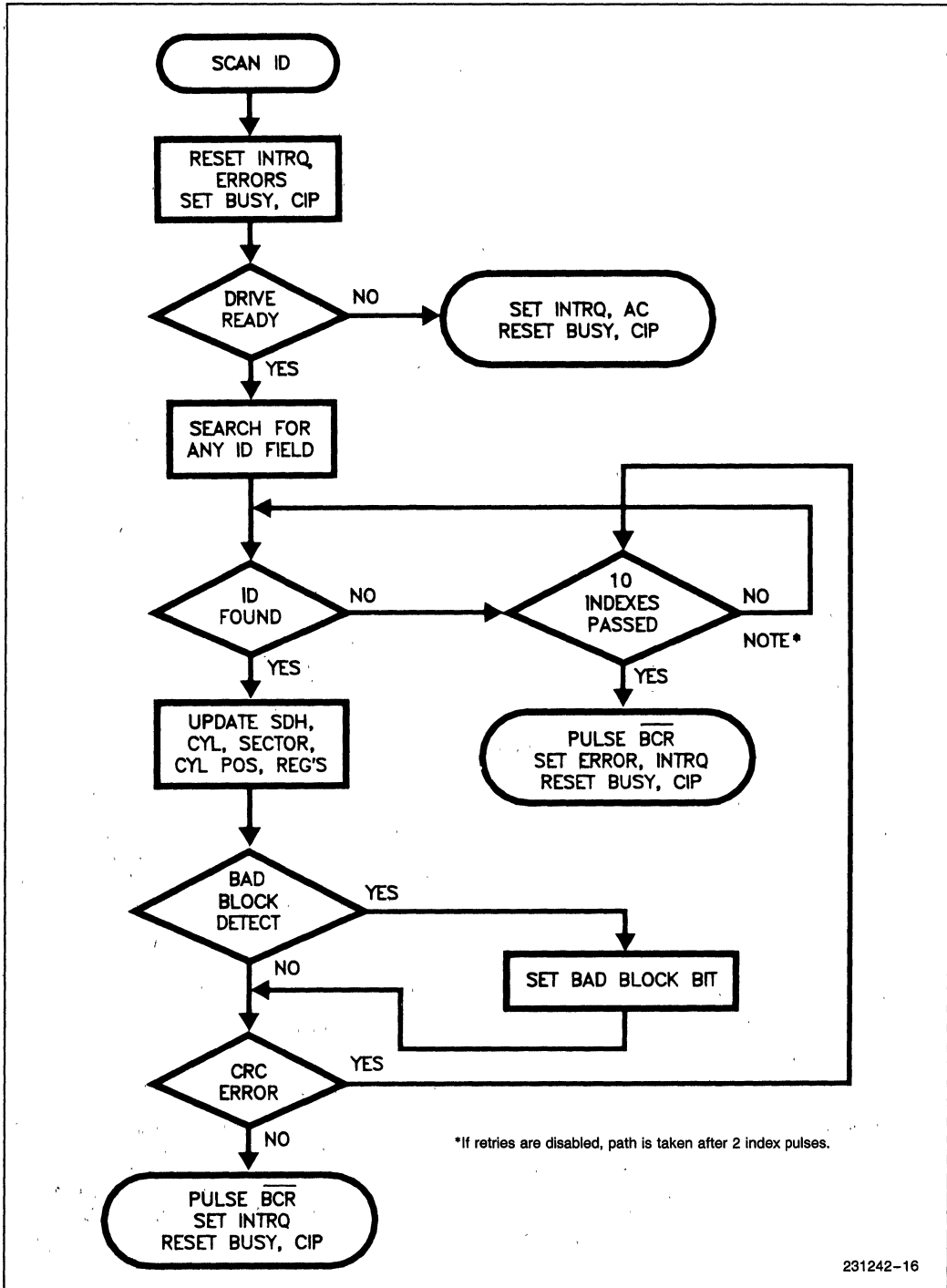
The SECTOR COUNT register is used to hold the total number of sectors to be formatted ($FFH = 255$ sectors), while the SECTOR NUMBER register holds the number of bytes, minus three, to be used for Gap 1 and Gap 3. If, for example, the SECTOR COUNT register value is 02H and the SECTOR NUMBER register value is 00H, then 2 sectors are formatted and 3 bytes of 4EH are written for Gap 1 and Gap 3. The data fields are filled with FFH and the CRC or ECC is automatically generated and appended. After the last sector is written the track is filled with 4EH.



231242-15

*If retries disabled then dashed path is taken after 2 index pulses.

Figure 13. Write Sector Command Flow



*If retries are disabled, path is taken after 2 index pulses.

Figure 14. Scan ID Command Flow

ADDR	DATA								
	0	1	2	3	4	5	6	7	
00	00	00	00	10	00	01	00	11	
08	00	02	00	12	00	03	00	13	
10	80	04	00	14	00	05	00	15	
18	00	06	00	16	00	07	00	17	
20	00	08	00	18	00	09	00	19	
28	00	0A	00	1A	00	0B	00	1B	
30	00	0C	00	1C	00	0D	00	1D	
38	00	0E	00	1E	00	0F	00	1F	
40	FF	FF	FF	FF	FF	FF	FF	FF	
				⋮					
F0	FF	FF	FF	FF	FF	FF	FF	FF	

Figure 15. Format Command Buffer Contents

The Gap 3 value is determined by the drive motor speed variation, data sector length, and the interleave factor. The interleave factor is only important when 1:1 interleave is used. The formula for determining the minimum Gap 3 length is:

$$\text{Gap 3} = (2 * M * S) + K + E$$

where:

- M = motor speed variation (e.g., 0.03 for + 3%)
- S = sector length in bytes
- K = 18 for an interleave factor of 1
0 for any other interleave factor
- E = 2 if ECC is enabled (SDH register, bit 7 = 1)

As for all commands, if WR FAULT is asserted or DRDY is deasserted during execution of the command, the command terminates and the Aborted Command bit in the ERROR register is set.

Figure 16 shows the format which the 82064 will write on the disk.

A flowchart of the WRITE FORMAT command is shown in Figure 17.

COMPUTE CORRECTION

The COMPUTE CORRECTION command determines the location and pattern of a single burst error, but does not correct it. The host, using the data provided by the 82064, must perform the actual correction. The COMPUTE CORRECTION command is used following a data field ECC error. The command initiating the read must specify no retries (T = 1).

The COMPUTE CORRECTION command first writes the four syndrome bytes from the internal ECC register to the sector buffer. Then the ECC register is clocked. With each clock, a counter is incremented

and the pattern examined. If the pattern is correctable, the procedure is stopped and the count and pattern are written to the sector buffer, following the syndrome. The process is also stopped if the count exceeds the sector size before a correctable pattern is found.

When the command terminates the sector buffer contains the following data:

- Syndrome MSB
- Syndrome
- Syndrome
- Syndrome LSB
- Error Pattern Offset
- Error Pattern Offset
- Error Pattern MSB
- Error Pattern
- Error Pattern LSB

As an example, when the Error Pattern Offset is zero the following procedure may correct the error. The first data byte of the sector is exclusive OR'd with the MSB of the Error Pattern, the second data byte with the second byte of the Error Pattern, and the third data byte with the LSB of the Error Pattern.

If the sector buffer count exceeds the sector size, or if the error burst length is greater than that selected by the Set Parameter command, the ECC/CRC error in the ERROR register and the Error bit in the STATUS register is set.

SET PARAMETER

This command selects the correction span to be used for the error correction process. A 5-bit span is selected when bit zero of the command equals 0, and an 11-bit span when bit zero equals 1. The 82064 defaults to a 5-bit span after a RESET.

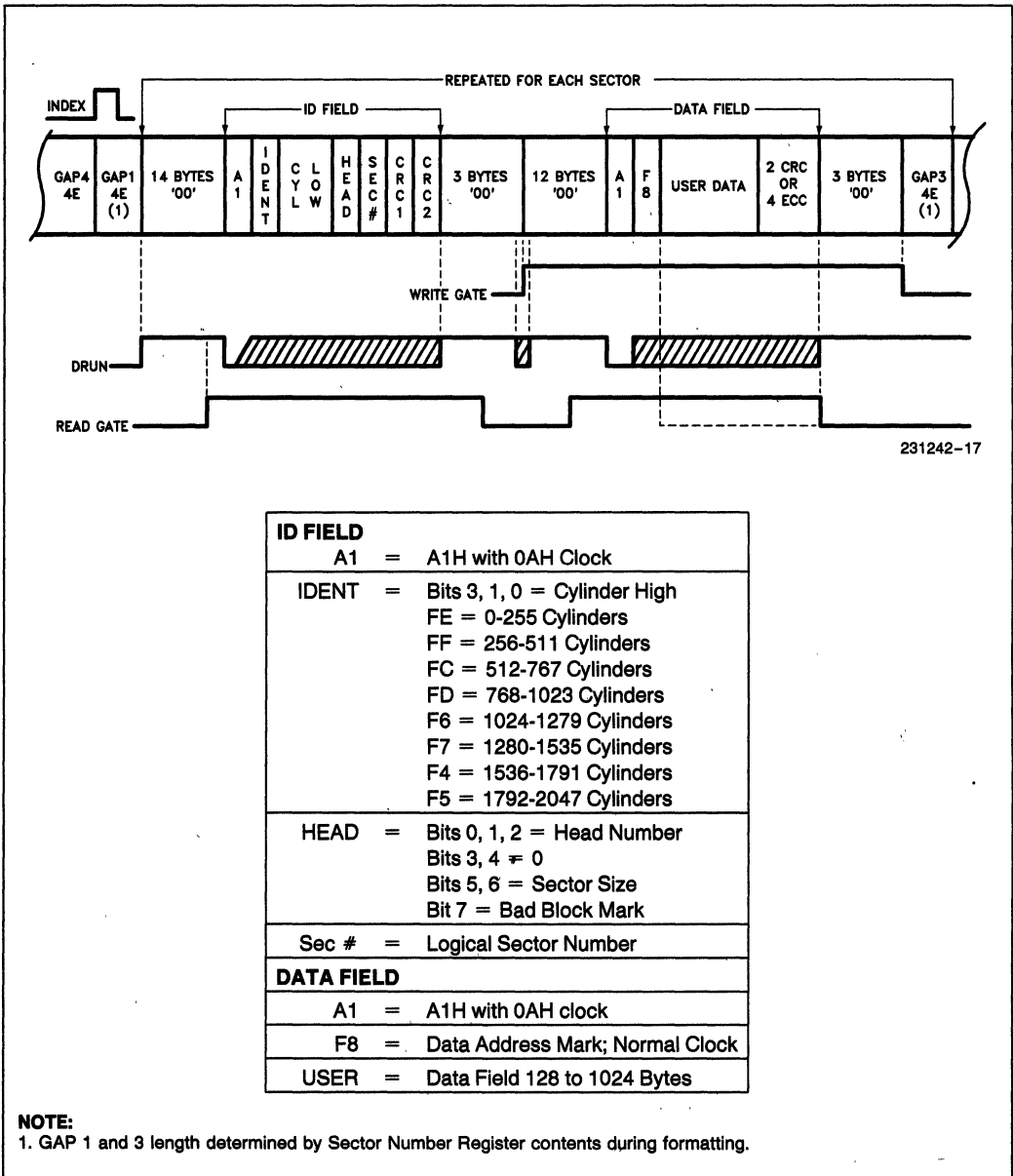


Figure 16. Track Format

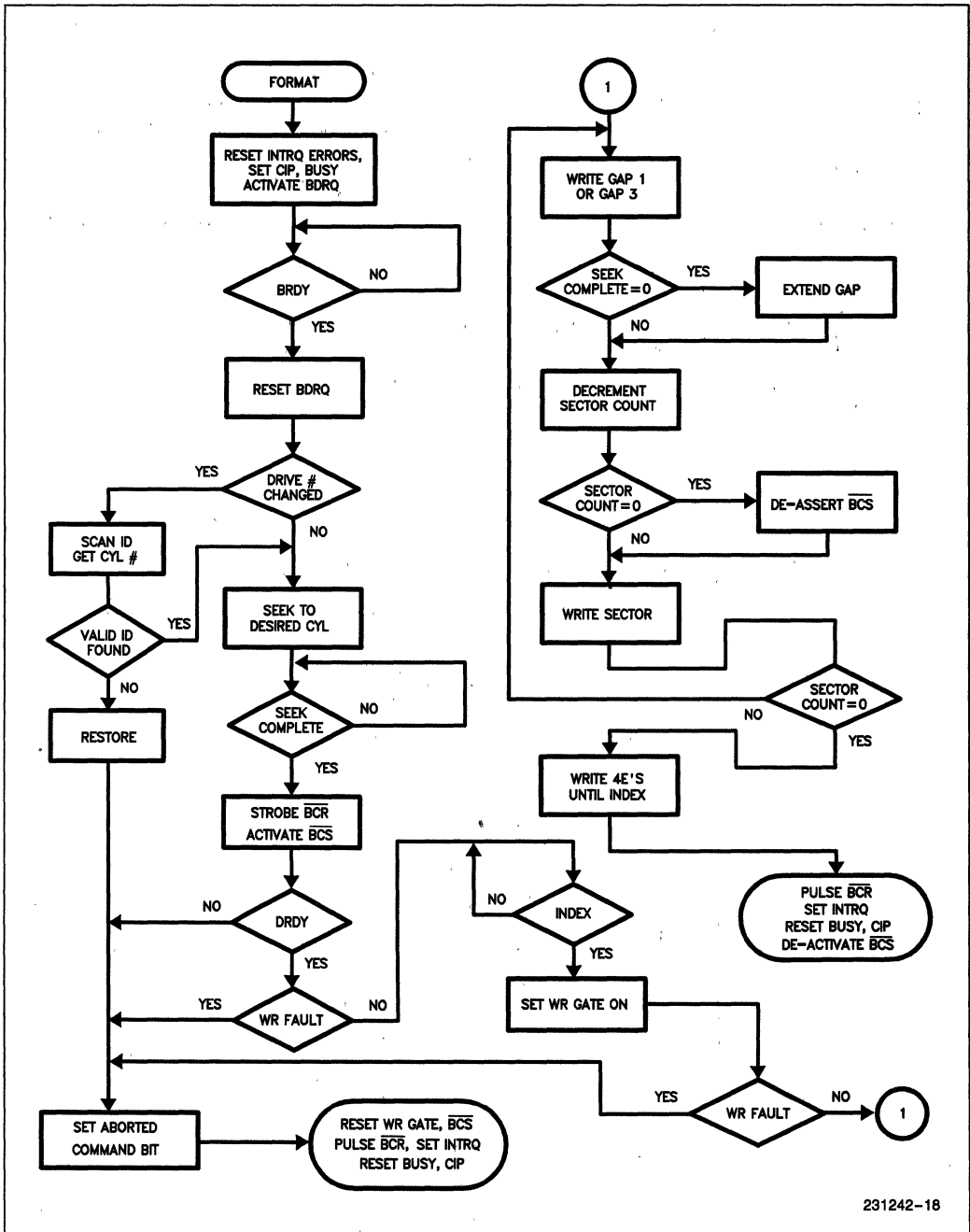


Figure 17. Write Format Command Flow

**ELECTRICAL CHARACTERISTICS
ABSOLUTE MAXIMUM RATINGS***

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Supply Voltage -0.5V to +8V
 Voltage on Any Input GND - 2V to +6.5V
 Voltage on Any Output . GND - 0.5V to V_{CC} + 0.5V
 Power Dissipation 1 Watt

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

NOTICE: Specifications contained within the following tables are subject to change.

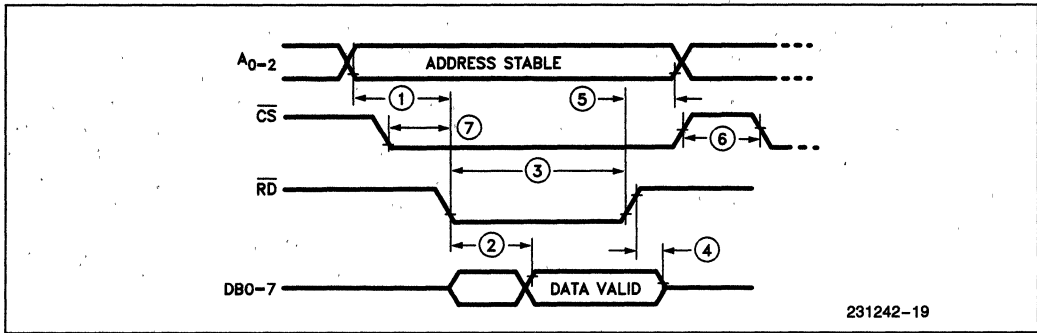
D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = +5V ± 10%; GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
I _{IL}	Input Leakage Current		± 10	μA	V _{IN} = V _{CC} to 0V
I _{OFL}	Output Leakage Current		± 10	μA	V _{OUT} = V _{CC} to 0.45V
V _{IH}	Input High Voltage	2.0	V _{CC} + 0.5	V	
V _{IL}	Input Low Voltage	-0.5	0.8	V	
V _{OH}	Output High Voltage	V _{CC} - 0.4 3.0		V	I _{OH} = -100 μA I _{OH} = -2.5 mA
V _{OL}	Output Low Voltage		0.4 0.45	V	I _{OL} = 2.5 mA 6.0 mA P21, 22, 23
I _{CC}	Supply Current		20 45	mA	See Note 10 See Note 11
I _{CCSB}	Standby Supply Current		50	μA	See Note 12
C _{IN}	Input Capacitance		10	pF	f _c = 1 MHz
C _{I/O}	I/O Capacitance		20	pF	Unmeasured pins returned to GND
For Pins 25, 34, 37, 39 (WR CLOCK, DRUN, READ DATA, READ CLOCK)					
TR _S	Rise Time		30	ns	0.9V to 4.2V

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

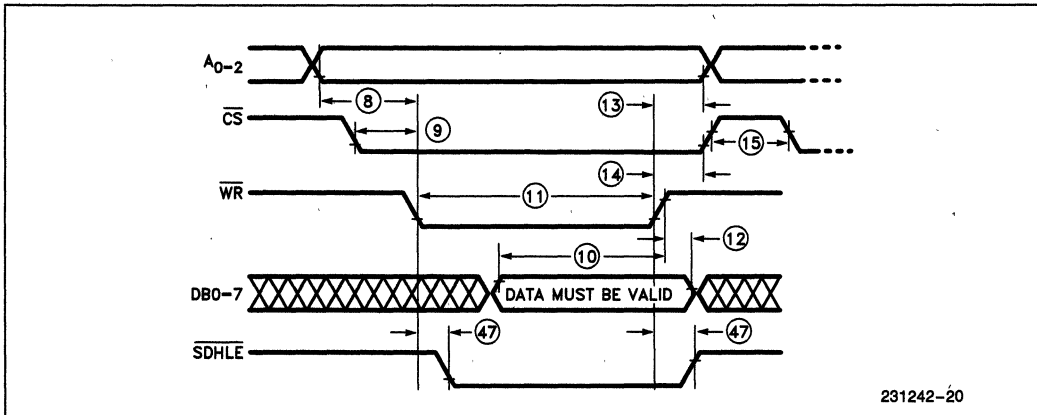
HOST READ TIMING WR CLOCK = 5.0 MHz

Symbol	Parameter	Min	Max	Units	Test Conditions
1	Address Stable Before \overline{RD} ↓	0		ns	
2	Data Delay from \overline{RD} ↓		150	ns	
3	\overline{RD} Pulse Width	100		ns	
4	Data Valid after \overline{RD} ↑	10	100	ns	
5	Address Hold Time after \overline{RD} ↑	0		ns	
6	Read Recovery Time	300		ns	
7	\overline{CS} Stable before \overline{RD} ↓	0		ns	See Note 6



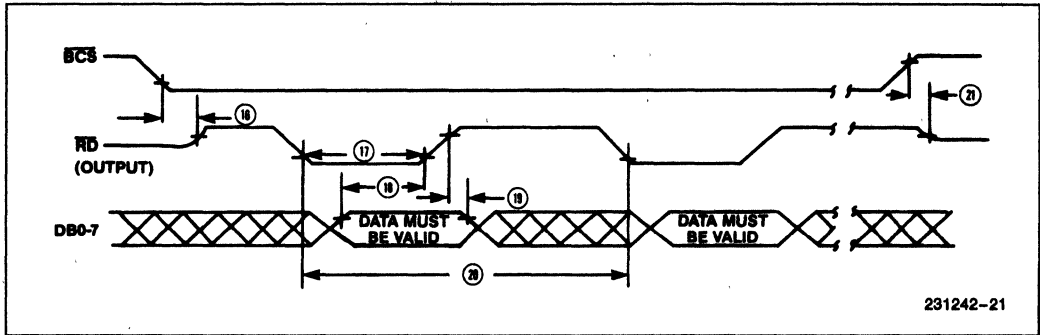
HOST WRITE TIMING WR CLOCK = 5.0 MHz

Symbol	Parameter	Min	Max	Units	Test Conditions
8	Address Stable Before $\overline{WR} \downarrow$	0		ns	
9	\overline{CS} Stable Before $\overline{WR} \downarrow$	0		ns	
10	Data Setup Time Before $\overline{WR} \uparrow$	75		ns	
11	\overline{WR} Pulse Width	100	10000	ns	
12	Data Hold Time After $\overline{WR} \uparrow$	0		ns	
13	Address Hold Time After $\overline{WR} \uparrow$	0		ns	
14	\overline{CS} Hold Time After $\overline{WR} \uparrow$	0		ns	See Note 7
15	Write Recovery Time	300		ns	
47	\overline{SDHLE} Propagation Delay	20	150	ns	



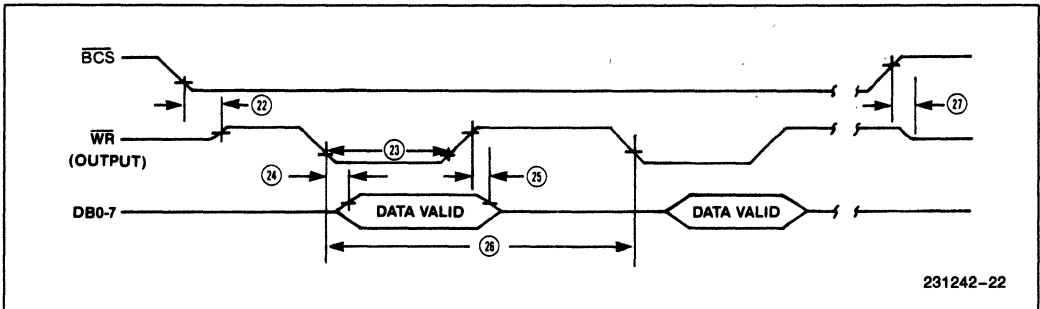
BUFFER READ TIMING (WRITE SECTOR COMMAND) WR CLOCK = 5.0 MHz

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
16	BCS ↓ to RD Valid	0		100	ns	
17	RD Output Pulse Width	300	400	500	ns	See Note 3
18	Data Setup to RD ↑	140			ns	
19	Data Hold from RD ↑	0			ns	
20	RD Repetition Rate	1.2	1.6	2.0	μs	See Note 8
21	RD Float from BCS ↑	0		100	ns	



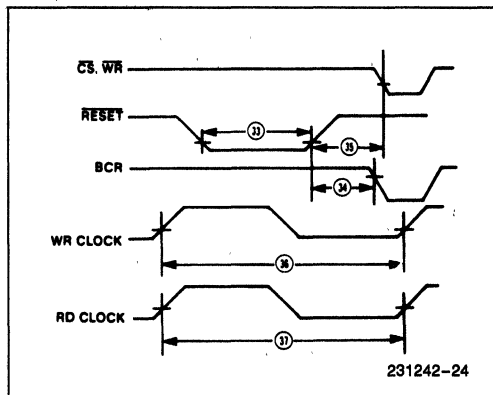
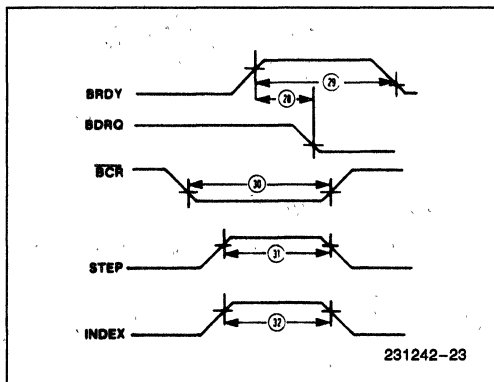
BUFFER WRITE TIMING (READ SECTOR COMMAND) WR CLOCK = 5.0 MHz

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
22	BCS ↓ to WR Valid	0		100	ns	
23	WR Output Pulse Width	300	400	500	ns	See Note 3
24	Data Valid from WR ↓			150	ns	
25	Data Hold from WR ↑	60		200	ns	
26	WR Repetition Rate	1.2	1.6	2.0	μs	See Note 8
27	WR Float from BCS ↑	0		100	ns	



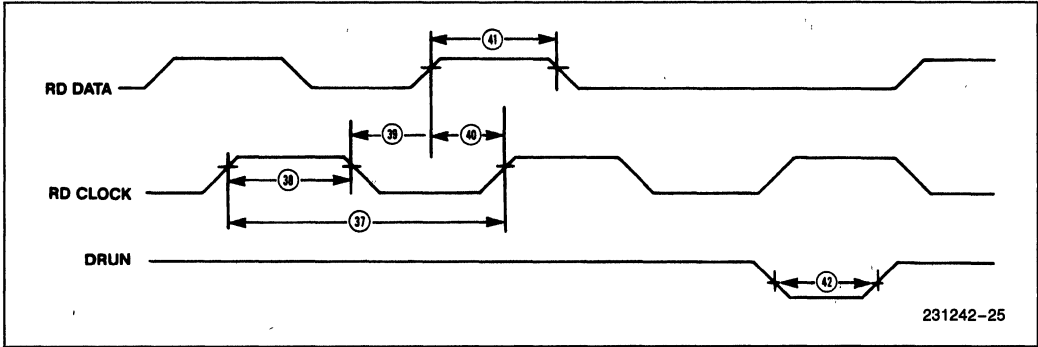
MISCELLANEOUS TIMING

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
28	BDRQ Reset from BRDY	20		200	ns	
29	BRDY Pulse Width	400			ns	See Note 4
30	BCR Pulse Width	1.4	1.6	1.8	μ s	See Note 9
31	STEP Pulse Width	1.5	1.6	1.7	μ s	Step Rate = 3.2 μ s/step
		7.9	8.4	8.7	μ s	All other step rates
32	INDEX Pulse Width	500			ns	
33	RESET Pulse Width	24			WR CLK	See Note 2
34	RESET \uparrow to BCR	0	3.2	6.4	μ s	See Note 1
35	RESET \uparrow to WR, CS \downarrow	6.4			μ s	See Note 1
36	WR CLOCK Frequency	0.25	5.0	5.25	MHz	50% Duty Cycle
37	RD CLOCK Frequency	0.25	5.0	5.25	MHz	See Note 5



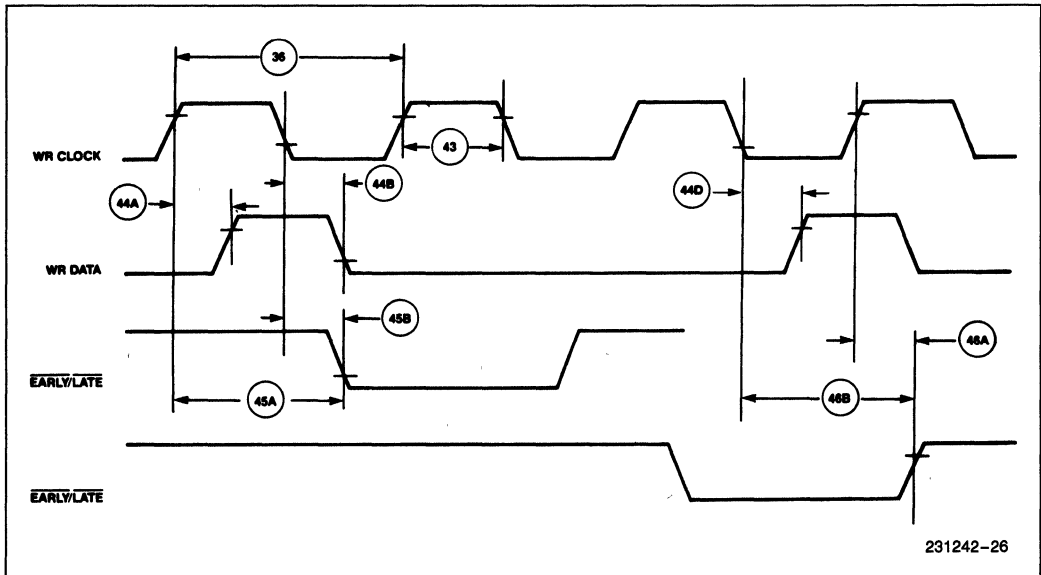
READ DATA TIMING WR CLOCK = 5.0 MHz

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
38	RD CLOCK Pulse Width	95		2000	ns	50% Duty Cycle
39	RD DATA after RD CLOCK \downarrow	10			ns	
40	RD DATA before RD CLOCK \uparrow	20			ns	
41	RD DATA Pulse Width	40		T38/2	ns	
42	DRUN Pulse Width	30			ns	

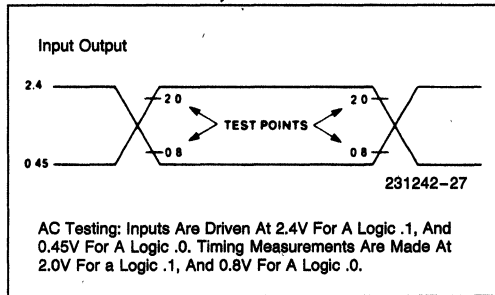


WRITE DATA TIMING WR CLOCK = 5.0 MHZ

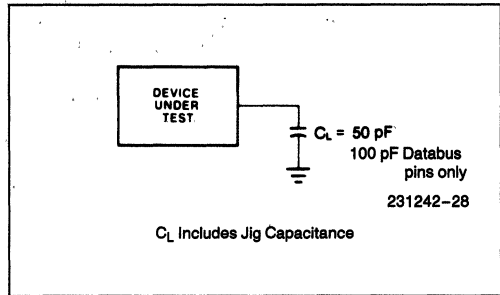
Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
43	WR CLOCK Pulse Width	95		2000	ns	50% Duty Cycle
	Propagation Delay					
44A	WR CLOCK ↑ to WR DATA ↑	10		65	ns	
44B	WR CLOCK ↓ to WR DATA ↓					
44D	WR CLOCK ↓ to WR DATA ↑					
45A	WR CLOCK ↑ to $\overline{\text{EARLY/LATE}} \downarrow$	10		65	ns	
45B	WR CLOCK ↓ to $\overline{\text{EARLY/LATE}} \downarrow$					
46A	WR CLOCK ↑ to $\overline{\text{EARLY/LATE}} \uparrow$	10		65	ns	
46B	WR CLOCK ↓ to $\overline{\text{EARLY/LATE}} \uparrow$					



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



NOTES

1. Based on WR CLOCK = 5.0 MHz
2. 24 WR CLOCK periods = 4.8 μs at 5.0 MHz.
3. 2 WR CLOCK periods \pm 100 ns.
4. Previous restrictions on BRDY no longer apply. There are no restrictions on when BRDY may come. BRDY may be connected directly to BDRQ.
5. WR CLOCK Frequency = RD CLOCK Frequency \pm 15%.
6. $\overline{\text{RD}}$ may be asserted before $\overline{\text{CS}}$ as long as it remains active for at least the minimum. T3 pulse width after $\overline{\text{CS}}$ is asserted.
7. $\overline{\text{WR}}$ may be asserted before $\overline{\text{CS}}$ as long as it remains active for at least the minimum T11 pulse width after $\overline{\text{CS}}$ is asserted.
8. 8 WR CLOCK periods \pm 2 WR CLOCK periods.
9. 8 WR CLOCK periods \pm 1 WR CLOCK period.
10. $V_{IL} = \text{GND}$, $V_{IH} = V_{CC}$, Outputs Open.
11. $V_{IL} = 0.8\text{V}$, $V_{IH} = 2.0\text{V}$, Outputs Open.
12. WR CLOCK = DC, $V_{IL} = 0\text{V}$, $V_{IH} = V_{CC}$, all output open, $\overline{\text{CS}}$ inactive.



November 1986

**Multimodule™
Winchester Controller
Using the CHMOS 82064**

J. SLEEZER
TECHNICAL MARKETING

Order Number: 231927-001

1.0 INTRODUCTION

The 82064 Winchester Disk Controller (WDC) was developed to ease the complex task of interfacing Winchester disk drives to microprocessor systems. Specifically, the 82064 WDC interfaces to drives that conform to the ST506 specification, which is the dominant interface for 5¼ inch drives. This Application Note provides some background on the 82064 WDC, the drive interfaces and general software routines. It concludes with a design example using the 82064 WDC interfaced to the SBX™ bus. Appendix B contains the listing of the software necessary to operate this controller board.

1.1 ST506 Winchester Drive Overview

Since the 82064 WDC interfaces only to drives conforming to the ST506 specification, this overview will limit itself to those drives. A summary of the ST506 specification is shown in Appendix A for those who are not familiar with it. The ST506 Winchester Disk contains from 1 to 8 hard disks (or platters) with the average being 2 to 3 disks. These disks are made from aluminum (hence the term hard disk) and are coated with some type of recording media. The recording media is typically made of magnetic-oxide, which is similar to the material used on floppy disks and cassette tapes. Each side of a hard disk is coated with recording media and each side can store data. Each surface of a disk has its own read/write head.

Hard disk drives are sealed units because the R/W heads actually fly above the disk surface at about 8 to 20 microinches. A piece of dust or dirt, which appears as a boulder to the gap between the heads and the disk surface, will wreak havoc upon the disk media.

The R/W heads are mechanically connected together and move as a single unit across the surface of the disk. There are 2 basic methods for positioning the heads. The first is with stepper motors, which is the most common method and is also used on most floppy disk drives. These positioners are used mainly because of their low cost.

The second method of positioning the heads is to use a voice-coil mechanism. These units do not move in steps but swing across the disk. These mechanisms generally permit greater track density than steppers, but also require complex feedback electronics which increases the cost of the drive. Generally, voice-coil head positioners use closed loop servo positioning, as compared to the open loop positioning used with stepper motors.

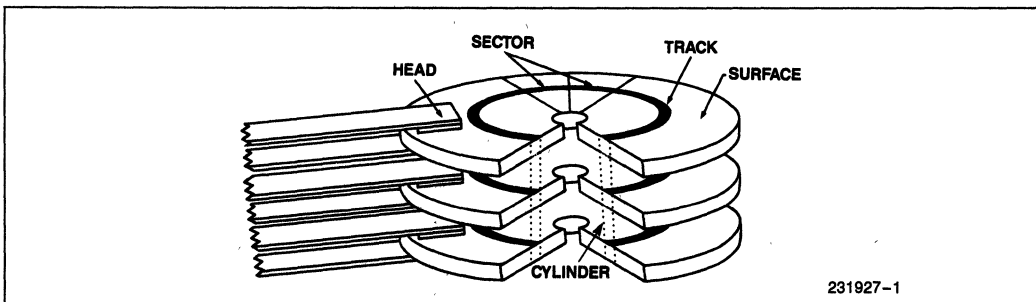
The surface of a disk is divided logically into concentric circles radiating from the center as shown in Figure 1. Each concentric circle is called a track.

The group of tracks, all in the same position, on all of the disks (platters) in the drive is collectively called a cylinder. The number of tracks on a surface (which affects storage density) is determined by the head positioners. Typically, stepper head positioners have fewer tracks than drives that use a voice coil positioner. Which type of positioner is used is irrelevant to the 82064 as positioners are part of the drive electronics. The 82064 can access up to 2048 tracks per surface.

Once the surface is divided into cylinders it is further divided radially (as with a pie). The area between the radial spokes is referred to as a sector. The number of sectors per track is determined by many variables, but is basically determined by the number of data bytes and the length of the ID field (which locates a sector). Figure 2 shows one manufacturer's specifications for their drive. The manufacturer formats the drive with 32–256 byte sectors per track. Alternatively, the drive could be reformatted to contain 17–512 byte sectors per track. This second option has fewer sectors per track but stores more data. Determining how many bytes each sector contains is done by extensive analysis of the hardware and operating system. The 82064 WDC is programmable for sector size during formatting.

The order in which sectors are logically numbered on the track is called interleaving. An interleave factor of four would have three sectors separating logically sequential sectors. Starting at the index pulse, an example of four way interleaving is:

Sector 1, Sector X, Sector Y, Sector Z, Sector 2, Sector . . .



231927-1

Figure 1

Capacity	
Unformatted	
Per Drive	6.38 Megabytes
Per Surface	1.59 Megabytes
Per Track	10416 Bytes
Formatted	
Per Drive	5.0 Megabytes
Per Surface	1.25 Megabytes
Per Track	8192 Bytes
Per Sector	256 Bytes
Sectors per Track	32
Transfer Rate	5.0 Megabits per second
Access Time	
Track to Track	3 ms
Average (Inc. Settle)	170 ms
Maximum (Inc. Settle)	500 ms
Settling Time	15 ms
Average Latency	8.33 ms
Functional Specifications	
Rotational speed	3600 rpm \pm 1%
Recording density	7690 bpi max
Flux density	7690 fci
Track density	255 tpi
Cylinders	153
Tracks	612
R/W Heads	4
Disks	2

Figure 2. A Typical Drive Specification

Interleaving is used primarily because one sector at a time is transferred from disk to sector buffer to system RAM. This transferring of data takes time, and causes a delay between the first sector transferred and sectors that follow it. Without interleaving, the delay in transferring data would result in sectors on the disk rotating past the heads before they could be read. The operating system would then have to wait one disk revolution to get to the next sector (a 16.7 msec delay). With interleaved sectors, the next logical sector would be positioned beneath the heads after the previous sector of data had been transferred to the system RAM. Interleaving unfortunately slows down the overall transfer rate from the disk. A 5 Mbit/second transfer rate averages out to a 1.25 Mbit/second transfer rate when many sectors are transferred with four way interleaving. Again, how much interleaving to use is determined by extensive hardware/software benchmarking.

Whenever data is stored on a multiple platter disk drive, the same track on all surfaces would be used before repositioning the heads to another track. Repositioning the heads generates a longer delay due to the mechanical delay of moving the heads. Switching to another head incurs no mechanical positioning delay. Only one head can be selected at a time.

Hard disk drives tend to be faster than floppies for two reasons. The speed at which the disk spins is about 10 times faster than the floppy (a floppy spins at 360 rpm for the popular double density disk drives). This yields an immediate one-tenth reduction in access times for the same size drive. While both ST506 drives and floppies use stepper motors, the steppers utilized by the hard disk drives are approximately twice as fast as those used by floppies.

2.0 82064 WINCHESTER DISK CONTROLLER

The 82064 WDC provides most of the functions necessary to interface between a microprocessor and an ST506 compatible disk drive. The 82064 converts the high level commands and parallel data of a microprocessor bus into ST506 compatible disk control signals and serial MFM encoded data. This section presents a detailed description of the 82064 and a discussion of various techniques which can be used to interface the 82064 to a microprocessor.

The internal structure of the 82064 is divided into several sections as shown in Figure 3. They are:

1. the microprocessor interface which includes the status and task registers;
2. sector buffer control;
3. the drive interface;
4. the data transfer section, which includes the MFM encoding/decoding of microprocessor data;
5. and CRC/ECC generation and checker.

2.1 Clock Inputs

The 82064 has two clock inputs: read clock (RD CLOCK) and write clock (WR CLOCK). The PLA controller, the processor interface, buffer control and MFM encoding sections operate off the WR CLOCK input. The RD CLOCK input is used only for decoding the MFM data stream. The clocks may be asynchronous to one another. Both clocks have non-TTL compatible inputs. The easiest method to interface to TTL requires a pull-up resistor to satisfy their input voltage needs. The resistor's value must be compatible with the VIL specification of these pins. See the Pin Descriptions Section for more specific information.

2.2 Microprocessor Interface

The microprocessor interface of the 82064 contains the control logic which permits commands and data to be transferred between the host and the 82064. The interface consists of an 8 bit, tri-state, bidirectional data bus; the task registers; a 3 to 8 address decoder for selecting one of the seven registers; and the general read, write, and chip select logic. Externally, the 82064 expects a buffer equal in size to a sector on the disk, and tri-state

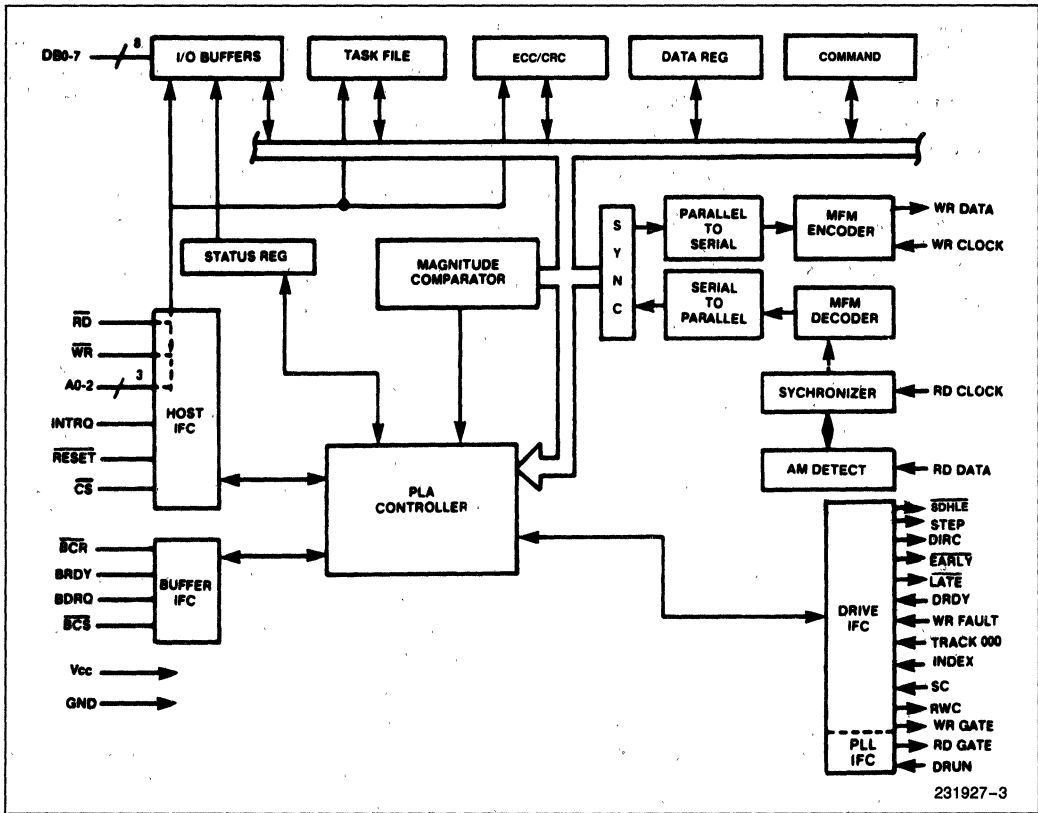


Figure 3. 82064 Internal Block Diagram

transceivers between the sector buffer and the microprocessors data bus in order to isolate itself from the microprocessor during disk data transfers.

A0-A2, Data Bus

These three address lines are active high signals and select one of the seven register locations in the 82064. They are not latched internally. If the three addresses are equal to 0 and the 82064 is selected, the data bus is kept tri-stated to ease interfacing to a sector buffer. The 82064's data bus is controlled by both the microprocessor and the 82064. The microprocessor has control for loading the registers and command. During disk reads or writes, control switches to the 82064 so that it may access the local sector buffer when transferring data between the disk and the buffer.

RD, WR, CS

The chip select (\overline{CS}) is typically decoded from the higher order address lines. \overline{CS} only permits data to be placed into, or read from, the 82064's task registers.

Once a disk operation starts, \overline{CS} no longer effects the 82064. \overline{RD} and \overline{WR} are bidirectional lines and are used to read or write the 82064's registers by the host microprocessor and are valid only if \overline{CS} is present. The 82064 will drive \overline{RD} and \overline{WR} when transferring data between the sector buffer and the disk. A signal is provided to tri-state the \overline{RD} and \overline{WR} lines from the host during a buffer access. This is covered in the Sector Buffer Control Section.

Interrupts

An interrupt is issued at the end of all commands, and the interrupt is cleared by reading any register. For the Read Sector command only, the 82064 allows the user the option of an interrupt either at the termination of the command, as is the case with all other commands, or when data needs to be transferred to the host from the sector buffer. This is discussed further in the Interrupt Mode Section. When selecting the data transfer option, the interrupt line will go active at the same time as the BDRQ line and the interrupt will be removed only when the proper handshake occurs with the sector buffer.

Task Registers

The Task Register File contains the command, status, track number, sector number, and other information necessary to properly execute a command. These registers are accessed with A0-A2, RD (or WR), and CS being valid and are not cleared by a reset. The registers are covered in detail in the Task Register File Section.

2.3 Sector Buffer Control

The 82064 was designed to operate with an external buffer equal in size to one sector. To ease the design-in of this buffer, the 82064 provides all of the control signals it needs to operate the buffer. This buffer must be isolated from the system bus, using tri-state buffers, during disk transfers to prevent contention during the period that the 82064 is accessing the buffer. A sector buffer is generally used to ease interfacing to the system due to the high disk data rates (625 kbytes/sec), although it is not required.

BCS

The Buffer Chip Select (\overline{BCS}) line goes active whenever the 82064 is accessing the sector buffer. This signal should remove the microprocessors ability to access the 82064 and sector buffer and must enable the sector buffer for use by the 82064.

At a 5 Mbit/sec disk data rate, the 82064 will access the buffer every 1.6 microseconds ($8 \text{ bits} \times 200 \text{ ns/bit}$). \overline{BCS} will remain low the entire time the 82064 is accessing the buffer. The 82064 will pulse the appropriate \overline{RD} or \overline{WR} line for each byte transferred.

BCR

Buffer Counter Reset (\overline{BCR}) goes active each time that \overline{BCS} changes state. Its purpose is to reset the address counter of the sector buffer back to zero before and after the 82064 uses the sector buffer. Its function is optimized for single sector transfers. Multiple sector transfers should use a software controlled buffer counter reset and not use \overline{BCR} as the sector buffer will be reset to the beginning after each sector is transferred.

BDRQ, BRDY

Buffer Data Request (BDRQ) and Buffer Ready (BRDY) provide the handshake needed to transfer data between the sector buffer and the host. BDRQ signals that data must be moved to/from the sector buffer and the host. BRDY has two functions. Once the transfer signaled by BDRQ is finished, asserting BRDY will inform the 82064 that the transfer is completed and that it may finish executing the command. BRDY is also used in multiple sector commands. BRDY going

high during a multiple sector transfer indicates that the buffer is full (or empty—depending upon the command) and the transfer should wait until the buffer is serviced. The sector that was being transferred will finish and the 82064 will deactivate \overline{BCS} and activate BDRQ. The host microprocessor must then transfer the data between the buffer and system memory. When this transfer is finished, asserting BRDY will cause the 82064 to resume the command.

The handshaking between BDRQ and BRDY occurs only in full sector increments—not on a byte basis. A high on BDRQ indicates a full sector's worth of data is required; BRDY going high indicates a full sector of data is available to the 82064 without interruption.

Only the rising edge of BRDY is valid. A falling edge may occur at any time without effect. \overline{BCR} will pulse and \overline{BCS} will go active eight byte times ($8 \text{ bytes} \times 8 \text{ bits/byte} \times 200 \text{ ns/bit} = 12.8 \text{ microseconds}$) before the first data byte is transferred from the sector buffer to the disk.

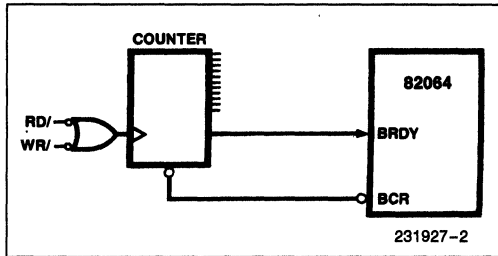


Figure 4. BRDY Generation Logic

2.4 Data Transfer Logic

This section of the 82064 is responsible for conversion of serial disk data to parallel data (and vice versa); encoding/decoding of the disk's MFM serial bit stream; and detecting the address mark.

Polled Interface

Since the 82064 isolates itself from the host during several commands, the host cannot read the status register during some periods to determine what course should be taken. In Figure 10, trying to read the status register when \overline{BCS} is active will return indeterminate data. To prevent the microprocessor from reading this indeterminate data, a hardware generated "Busy" pattern should be driven onto the data bus if \overline{BCS} is active. This is shown in Figure 11. The status register contains a data request (DRQ) bit whose timing is equal to the BDRQ output signal, thus making a polled operation possible. DRQ will stay set in the status register until a BRDY is generated.

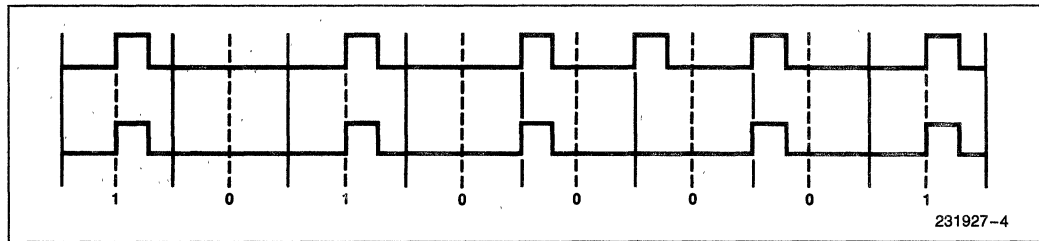


Figure 5. Data Address Mark

One design issue with the polled interface occurs when the microprocessor is polling the status and the 82064 deactivates \overline{BCS} . The microprocessor would normally read the hardware busy pattern. If \overline{BCS} is deasserted, the hardware pattern is disabled and the microprocessor will start to read the real status register. The read cycle may almost be finished, and the read access period of the 82064 will not be satisfied. The data returned to the microprocessor will be invalid.

Interrupt Interface

There are cases where the designer does not want to tie up the microprocessor with polling. The typical 82064 design will need two interrupts per command. One for a data transfer and one for the completion of the command. The 82064 has an output to issue an interrupt when the command has finished. However for data transfers an interrupt must be generated from the BDRQ line as shown in Figure 12 (whether a DMA controller is used or not). When a data transfer is needed, the 82064 will activate the BDRQ line. The microprocessor will be interrupted and do the data transfer function. BDRQ will stay active until BRDY is generated, so the system must either use edge triggered interrupts or must not write the end-of-interrupt byte until BDRQ is removed (this is true of Intel's 8259A).

MFM Encoding/Decoding

The MFM encoding section will receive 8 bit parallel data when a valid command has been recognized and BRDY has gone high. The parallel data is first serialized and converted to an intermediate, NRZ encoded, data stream. The serial NRZ data is sent to the MFM encoding section and then transferred to the disk. Decoding of the MFM bit stream (during disk reads) happens in reverse order.

The control logic operates off the write clock (WR CLOCK) running at a frequency of the desired transfer rate. The MFM decoding portion operates off of the read clock (RD CLOCK) input, which is supplied by an external phase lock loop. The two clocks need not be synchronized to each other. Data is written (and hence read) with the most significant bit first.

Address Mark Detector

The address mark is a unique 2 byte code written at the beginning of each ID field and data field. This address mark serves two purposes. It tells the controller what type of data is about to be received so that internal computations can be performed, and to ensure that ID fields are not sent to the host. The second purpose is to align the serial data back to the original 8 bit boundaries that existed when data was written (there are no byte boundaries on a disk).

An address mark is always preceded by the all zeros synchronization field. The 82064 starts comparing the incoming data stream when the synchronization field ends. A high speed comparator is used since the 82064 does not yet know where the proper byte boundaries are. When a proper comparison of the address mark is made the controller starts assembling bytes, starting with the second byte of the address mark.

The first byte of the address mark is an "A1" Hex, but purposely violates the MFM encoding rules by removing a clock pulse. In Figure 5, the first example is of a normal MFM encoded A1H. The second example is of the address mark and shows the missing clock pulse. The non-MFM compatible A1 is to prevent the host from issuing a similar data byte and possibly confusing detection logic.

The second byte specifies either an ID or data field and is encoded according to normal MFM rules. It is either an "F8" Hex for a data field, or "FC" through "FF" for an ID field. The different values correspond to a range of cylinders on the drive in increments of 256 tracks. The 82064 makes no use of this information, but writes it for compatibility with the ST506 specification during formatting.

PLA Control

The PLA Controller interprets command sent by the microprocessor. Its operation is synchronized to the WR CLOCK input. The PLA controller is started when a command is written into the command register. It generates control signals and operates in a handshake mode when communicating with the MFM decoding block.

Magnitude Comparator

A 10 bit magnitude comparator is used to calculate the direction and number of step pulses needed to move the head from the present cylinder position to the desired position. A separate high speed equivalence comparator is used to compare ID field bytes when searching for a sector ID field.

2.5 CRC/ECC Generator and Checker

The 82064 provides two options for protecting the integrity of the data field. The data field may have either a CRC (SDH register, bit 7 = 0), or a 32-bit ECC (SDH register, bit 7 = 1) appended to it. The ID field is always protected by a CRC.

CRC Generation/Checking

The CRC generator computes and checks the cyclic redundancy check bytes that are appended to the ID and data fields. CRC generation/checking is always done on ID fields. Data fields have a choice between 82064 CRC, internal ECC or externally supplied ECC. The CRC mode is chosen by setting bit 7 of the SDH register low. The CRC mode provides a means of verifying the accuracy of the data read from the disk, but does not attempt to correct it. The CRC generator computes and checks cyclic redundancy check characters that are written and read from the disk after the ID and data fields.

The generator polynomial for the CRC-CCITT (CRC-16) code is:

$$x^{16} + x^{12} + x^5 + 1 = (x + 1)(x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1)$$

The code's capability is as follows:

- Detects all occurrences of an odd number of bits in error.
- Detects all single, double, and triple bit errors if the record length (including check bits) is less than 32,767 bits.
- Detects all single-burst errors of sixteen bits or less.
- Detects 99.99695% of all possible 17 bit burst errors, and 99.99847% of all possible longer burst, assuming all errors are possible and equally probable.

The CRC code has some double-burst capability when used with short records (sectors). For a 256 byte sector the code will detect double-bursts as long as the total number of bits in error does not exceed 7.

If the CRC character generated while reading the data does not equal the one previously written, an error exists. If an ID field CRC error occurs the "ID not found" bit in the error register will be set. If a data field CRC error occurs the "ECC/CRC" bit in the error register will be set.

ECC Generation/Checking

The ECC mode is only applicable to the data field. It provides the user with the ability to detect and correct errors in the data field automatically. The commands and registers which must be considered when ECC is used are:

- SDH Register, bit 7 (CRC/ECC)
- READ SECTOR Command, bit 0 (T)
- READ SECTOR and WRITE SECTOR Commands, bit 1 (L)
- COMPUTE CORRECTION Command
- SET PARAMETER Command
- STATUS Register, bit 2—error correction successful
- STATUS Register, bit 0—error occurred
- ERROR Register, bit 6—uncorrectable error

To enable the ECC mode, bit 7 of the SDH register must be set to one.

Bit 0 (T) of the READ Command controls whether or not error correction is attempted. When T = 0 and an error is detected, the 82064 tries up to 10 times to correct the error. If the error is successfully corrected, bit 2 of the STATUS Register is set. The host can interrogate the status register and detect that an error occurred and was corrected. If the error was not correctable, bit 6 of the ERROR Register is set. If the correction span was set to 5 bits, the host may now execute the SET PARAMETER Command to change the correction span to 11 bits, and attempt the read again. If the error persists, the host can read the data, but it will contain errors.

When T = 1 and an error is detected, no attempt is made to correct it. Bit 0 of the STATUS Register and bit 6 of the ERROR Register are set. The user now has two choices:

- Ignore the error and make no attempt to correct it.
- Use the COMPUTER CORRECTION Command to determine the location and pattern of the error, and correct it within the user's program.

When the COMPUTE CORRECTION Command is implemented, it must be done before executing any command which can alter the contents of the ECC Register. The READ SECTOR, WRITE SECTOR,

SCAN ID, and FORMAT Commands will alter this register and correction will be impossible. The COMPUTE CORRECTION Command may determine that the error is uncorrectable, at which point the error bits in the STATUS and ERROR Registers are set.

Although ECC generation starts with the first bit of the F8H byte in the data ID field, the actual ECC bytes written will be the same as if the A1H byte was included. The ECC polynomial used is:

$$X^{32} + X^{28} + X^{26} + X^{19} + X^{17} + X^{10} + X^6 + X^2 + 1$$

For automatic error correction, the external sector buffer must be implemented with a static RAM and counter, not with a FIFO.

The SET PARAMETER Command is used to select a 5-bit or 11-bit correction span.

2.6 Drive Interface

The drive interface of the 82064 contains the logic that makes possible the storage and reliable recovery of data. This interface consists of the drive and head select logic, the disk control signals, and read and write data logic as shown in Figure 6. This section describes the external circuitry which is required to complete the 82064's drive interface.

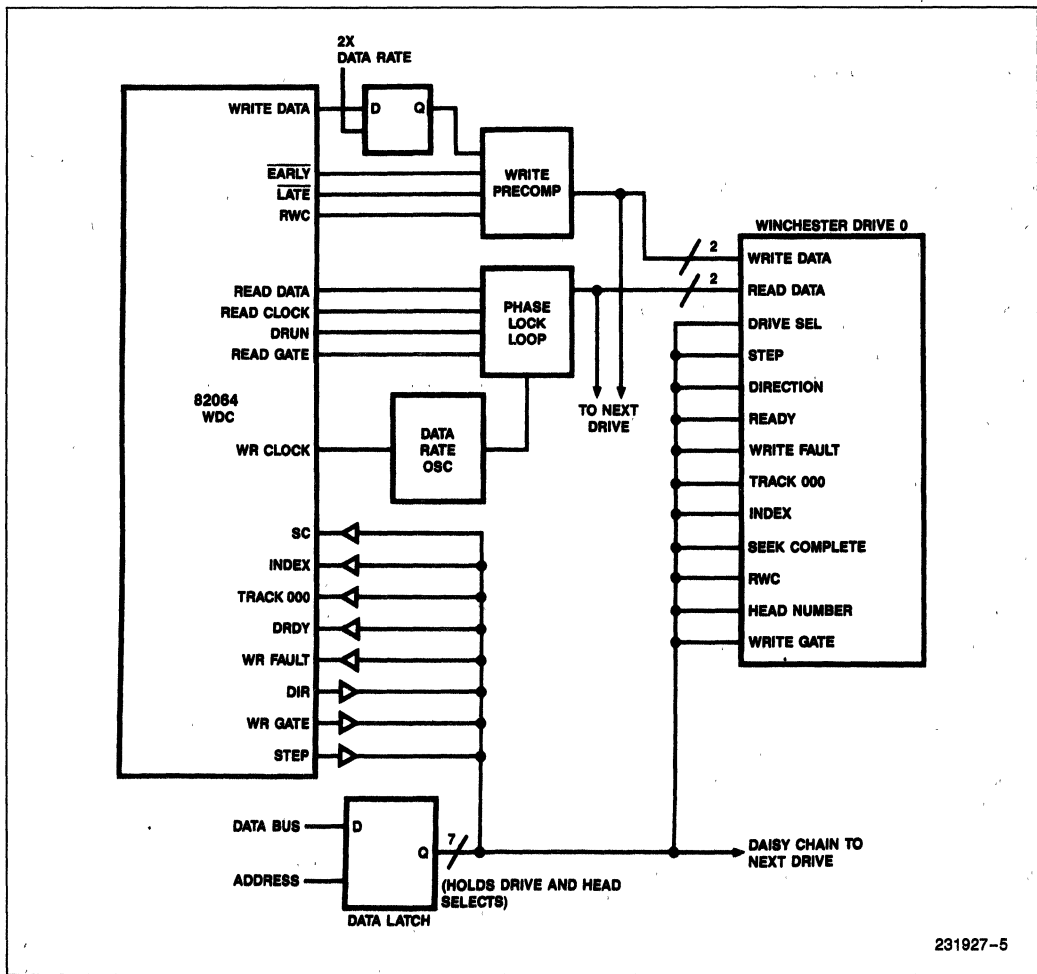


Figure 6. Drive Interface

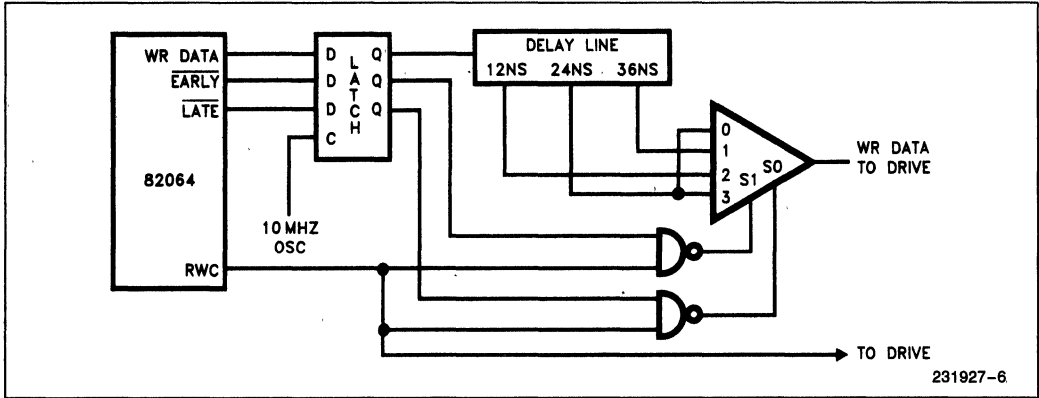


Figure 7. Write Precompensation Logic

Drive/Head Select

The 82064 has no outputs for selecting the head or drive. Therefore these signals must be generated by the user as shown in Figure 6. Data bits 0-4 should be latched whenever the SDH register is written. Bits 0-2 would then be driven onto the drive cable with open collector buffers. Bits 3 and 4 would be decoded after being latched, then buffered for the cable. The head information written to the 82064's SDH register is used to write the proper ID fields during formatting. Changing the drive bits in the SDH register will cause a Scan ID to be performed by the 82064 to update non user accessible registers.

Drive Control

The drive control (STEP, DIR, WR FAULT, TRACK 000, INDEX, SC, RWC, and WR GATE) signals are merely conditioned for transmission over the drive

cable. The purpose of each pin can be found in the section on Pin Descriptions and their use in the Command Section.

WR DATA, EARLY, LATE

Figure 7 is a diagram of the interface required on the write data line. The final stage of the MFM encoding requires applying the WR DATA to an external flip-flop clocked at 10 MHz. The 82064 monitors the serial write data output for particular bit patterns that require precompensation to prevent bit shifting. EARLY and LATE are active on all cylinders and will normally require that RWC be factored into them to activate the data precompensation on the proper cylinder.

A delay line is required to generate the delayed data for precompensation since the actual delay varies between drive manufacturers. EARLY and LATE go active in the same clock period that generates the data bit to be shifted.

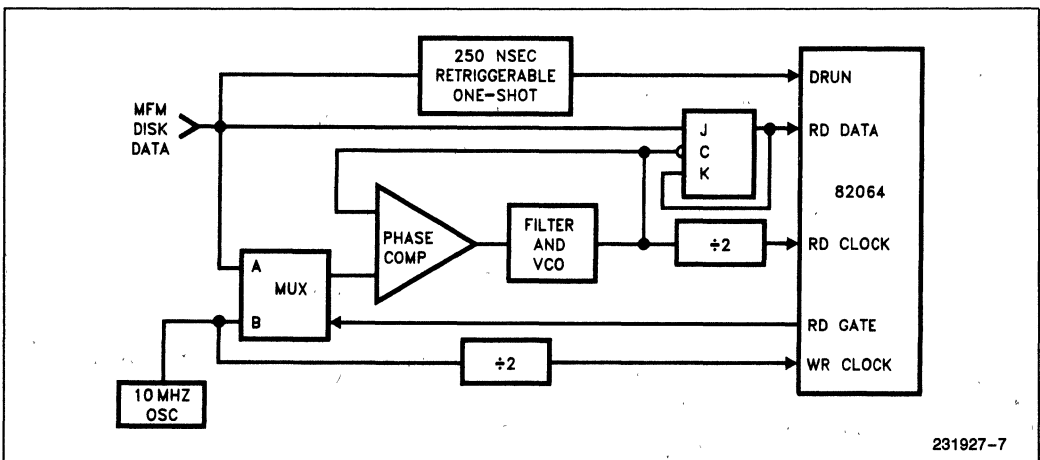


Figure 8. Data Separator Circuit

RD Data, DRUN, RD Gate

The read data interface is shown in Figure 8, and consists of the data run (DRUN) signal and a phase lock loop to generate the RD CLOCK input to decode the serial data. DRUN is generated from a retriggerable one-shot with a period just exceeding one bit cell. A sync field consisting of a string of clock pulses will continually retrigger the one-shot producing a steady high level on DRUN. The 82064 counts off 16 clock pulses internally, and if DRUN is still active, will make RD GATE active. Any byte other than an address mark will deactivate RD GATE and the sequence starts over.

The phase lock loop generates RD CLOCK which is used to decode the incoming serial data. Until RD GATE is activated by the 82064, the phase lock loop (PLL) should be locked onto a local 10 MHz clock to minimize PLL lock-up times. When RD GATE is activated, the PLL starts locking onto the incoming data stream, which should consist of the all zeros sync field. Once the PLL locks onto this synch field, the 82064 will start examining the serial data for a non-zero byte. A non-zero byte will be indicated by DRUN dropping since the address mark follows the sync field and is an "A1" Hex. This sequence is shown in Figure 9. If the address mark is detected, and if it was preceded by at least 9 bytes of zeros, RD GATE will stay active. The 82064 will then assemble bytes of data, and ensure the proper ID field is found. If a non-zero or non-address mark byte was detected, RD GATE will go inactive for a minimum of 2 byte times. If a data field or the wrong ID field is detected, or the ID field was not preceded by 8 bytes of zeros, then RD GATE goes inactive and the sequence starts over with the 82064 examining the DRUN input.

2.7 Microprocessor Interfaces

This section shows the general 82064 interfaces to a microprocessor system. There are essentially four interfaces which consist of a combination of polled, DMA, and interrupts. While the 82064 was designed to interface directly to one type, it accommodates all with minor additional logic.

DMA Interface

The 82064 is designed to use a DMA controller for data transfer between its sector buffer and the host system, and to interrupt the host when the command has finished. This interface is shown in Figure 10.

When the 82064 determines that a transfer is needed between the sector buffer and the host (either at the beginning of a command or through BRDY going active in a multiple sector transfer), it will assert BDRQ. BDRQ will initiate a DMA transfer via the DMA re-

quest input. The DMA controller will generate reads or writes which will increment an address counter. BRDY indicates that the data transfer has finished and is issued off the carry-out line (or high order address line) of the counter. The 82064 will assert BDRQ at this point and activate BCS to prevent the host from interfering with disk/buffer transfers. There can be no polling for a data transfer or a register read without an interrupt in this scheme.

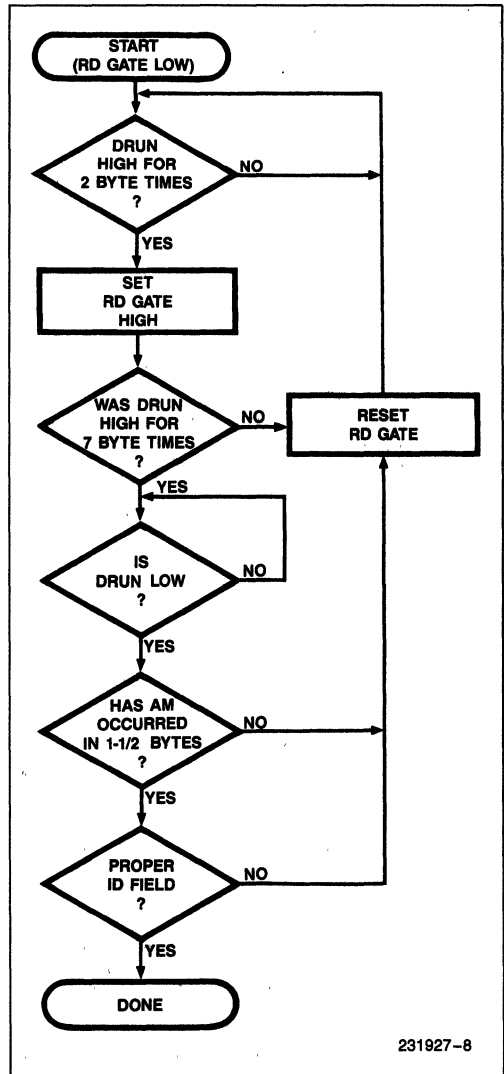


Figure 9. PLL Control Sequence

231927-8

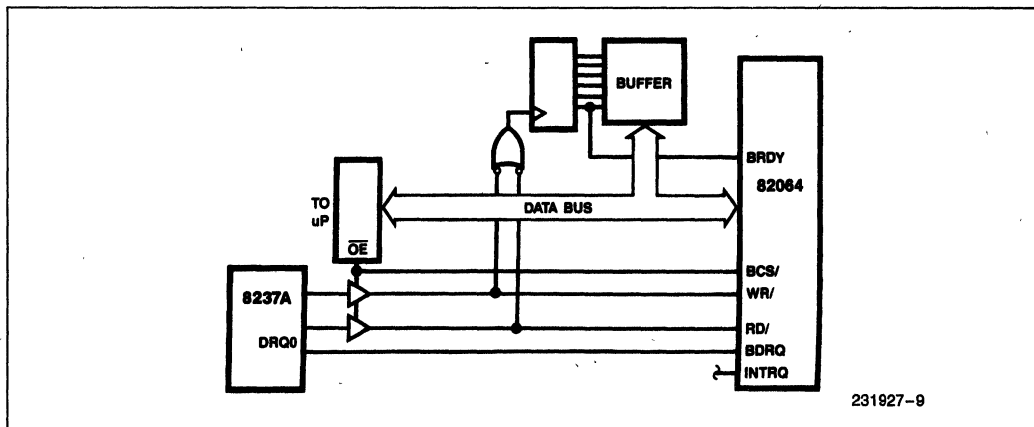


Figure 10. 82064 DMA Interface

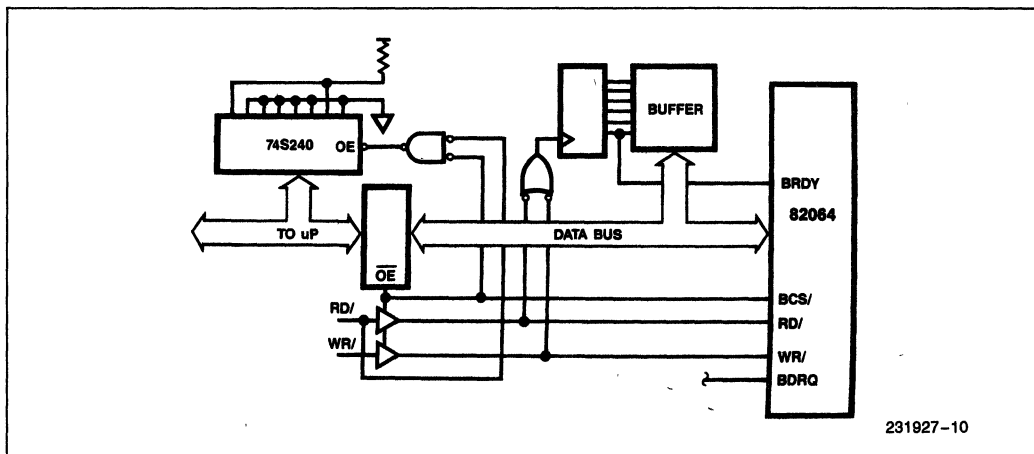


Figure 11. 82064 Polled Interface

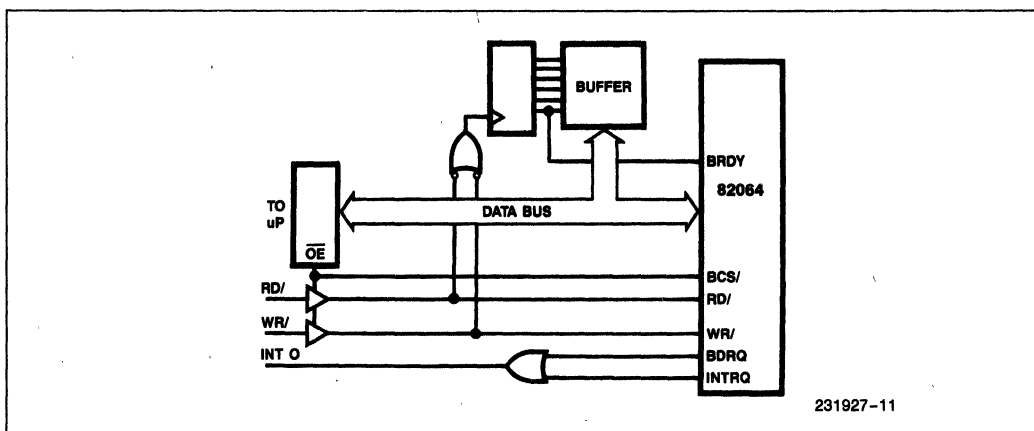


Figure 12. 82064 Interrupt Interface

3.0 PIN DESCRIPTIONS

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
\overline{BCS}	1	1	O	BUFFER CHIP SELECT: Output used to enable reading or writing of the external sector buffer by the 82064. When low, the host should not be able to drive the 82064 data bus, \overline{RD} , or \overline{WR} lines.
\overline{BCR}	2	2	O	BUFFER COUNTER RESET: Output that is asserted by the 82064 prior to read/write operation. This pin is asserted whenever \overline{BCS} changes state. Used to reset the address counter of the buffer memory.
INTRQ	3	3	O	INTERRUPT REQUEST: Interrupt generated by the 82064 upon command termination. It is reset when the STATUS register is read, or a new command is written to the COMMAND register. Optionally signifies when a data transfer is required on Read Sector commands.
\overline{SDHLE}	4	4	O	\overline{SDHLE} is asserted when the SDH register is written by the host.
\overline{RESET}	5	7	I	RESET: Initializes the controller and clears all status flags. Does not clear the Task Register File.
\overline{RD}	6	8	I/O	READ: Tri-state, bi-directional signal. As an input, \overline{RD} controls the transfer of information from the 82064 registers to the host. \overline{RD} is an output when the 82064 is reading data from the sector buffer (\overline{BCS} low).
\overline{WR}	7	9	I/O	WRITE: Tri-state, bi-directional signal. As an input, \overline{WR} controls the transfer of command or task information into the 82064 registers. \overline{WR} is an output when the 82064 is writing data to the sector buffer (\overline{BCS} low).
\overline{CS}	8	10	I	CHIP SELECT: Enables \overline{RD} and \overline{WR} as inputs for access to the Task Registers. It has no effect once a disk command starts.
A_0-A_2	9-11	11-13	I	ADDRESS: Used to select a register from the task register file.
DB_0-DB_7	12-19	14-16 18-22	I/O	DATA BUS: Tri-state, bi-directional 8-bit Data Bus with control determined by \overline{BCS} . When \overline{BCS} is high the microprocessor has full control of the data bus for reading and writing the Task Register File. When \overline{BCS} is low the 82064 controls the data bus to transfer data to or from the buffer.
V_{SS}	20	23		Ground
WR DATA	21	24	O	WRITE DATA: Output that shifts out MFM data at a rate determined by Write Clock. Requires an external D flip-flop clocked at 10 MHz. The output has an active pullup and pulldown that can sink 4.8 mA.
LATE	22	25	O	LATE: Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.
EARLY	23	26	O	EARLY: Output used to derive a delay value for write precompensation. Valid when WR GATE is high. Active on all cylinders.
WR GATE	24	27	O	WRITE GATE: High when write data is valid. WR GATE goes low if the WR FAULT input is active. This output is used by the drive to enable head write current.
WR CLOCK	25	29	I	WRITE CLOCK: Clock input used to derive the write data rate. Frequency = 5 MHz for the ST506 interface.

3.0 PIN DESCRIPTIONS (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
DIR	26	30	O	DIRECTION: High level on this output tells the drive to move the head inward (increasing cylinder number). The state of this signal is determined by the 82064's internal comparison of actual cylinder location vs. desired cylinder.
STEP	27	31	O	STEP: This signal is used to move the drive head to another cylinder at a programmable frequency. Pulse width = 1.6 μ s for a step rate of 3.2 μ s/step, and 8.4 μ s for all other step rates.
DRDY	28	32	I	DRIVE READY: If DRDY from the drive goes low, the command will be terminated.
INDEX	29	33	I	INDEX: Signal from the drive indicating the beginning of a track. It is used by the 82064 during formatting, and for counting retries. Index is edge triggered. Only the rising edge is valid.
WR FAULT	30	34	I	WRITE FAULT: An error input to the 82064 which indicates a fault condition at the drive. If WR FAULT from the drive goes high, the command will be terminated.
TRACK 000	31	35	I	TRACK ZERO: Signal from the drive which indicates that the head is at the outermost cylinder. Used to verify proper completion of a RESTORE command.
SC	32	36	I	SEEK COMPLETE: Signal from the drive indicating to the 82064 that the drive head has settled and that reads or writes can be made. SC is edge triggered. Only the rising edge is valid.
RWC	33	37	O	REDUCED WRITE CURRENT: Signal goes high for all cylinder numbers above the value programmed in the Write Precomp Cylinder register. It is used by the precompensation logic and by the drive to reduce the effects of bit shifting.
DRUN	34	38	I	DATA RUN: This signal informs the 82064 when a field of all ones or all zeroes has been detected in the read data stream by an external one-shot. This indicates the beginning of an ID field. RD GATE is brought high when DRUN is sampled high for 16 clock periods.
BRDY	35	39	I	BUFFER READY: Input used to signal the controller that the buffer is ready for reading (full), or writing (empty), by the host μ P. Only the rising edge indicates the condition.
BDRQ	36	40	O	BUFFER DATA REQUEST: Activated during Read or Write commands when a data transfer between the host and the 82064's sector buffer is required. Typically used as a DMA request line.
RD DATA	37	41	I	READ DATA: Single ended input that accepts MFM data from the drive.
RD GATE	38	42	O	READ GATE: Output that is asserted when a search for an address mark is initiated. It remains asserted until the end of the ID or data field.
RD CLOCK	39	43	I	READ CLOCK: Clock input derived from the external data recovery circuits.
V _{CC}	40	44	I	D.C. POWER: +5V.
NC	—	5, 6, 17, 28		NO CONNECTS

4.0 TASK REGISTER FILE

The Task Register File is a bank of registers used to hold parameter information pertaining to each command. These registers and their addresses are:

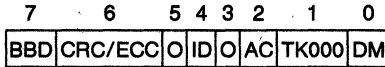
A2	A1	A0	READ	WRITE
0	0	0	(Bus Tri-Stated)	(Bus Tri-Stated)
0	0	1	Error Flags	Reduce Write Current
0	1	0	Sector Count	Sector Count
0	1	1	Sector Number	Sector Number
1	0	0	Cylinder Low	Cylinder Low
1	0	1	Cylinder High	Cylinder High
1	1	0	SDH	SDH
1	1	1	Status Register	Command Register

NOTE:

Registers are not cleared by RESET

4.1 Error Register

This read-only register contains specific error status after the completion of a command. If any bit in this register is set, then the Error bit in the Status Register will also be set. The bits are defined as follows:



Bit 7 - Bad Block Detect (BBD)

This bit is set when an ID field has been encountered that contains a bad block mark. The bad block bit is set only during formatting. The 82064 will terminate a command if an attempt is made to read a sector that contains this bit.

Bit 6 - CRC/ECC Data Field Error (CRC/ECC)

When in the CRC mode (SDH register, bit 7 = 0), this bit is set when a CRC error occurs in the data field. When retries are enabled, ten more attempts are made to read the sector correctly. If none of these attempts are successful bit 0 in the STATUS register is also set. If one of the attempts is successful, the CRC/ECC error bit remains set to inform the host that a marginal condition exists; however, bit 0 in the STATUS register is not set.

When in the ECC mode (SDH register, bit 7 = 1), this bit is set when the first non-zero syndrome is detected. When retries are enabled, up to ten attempts are made to correct the error. If the error is successfully corrected, this bit remains set; however, bit 2 of the STATUS register is also set to inform the host that the error has been corrected. If the error is not correctable, the CRC/ECC error bit remains set and bit 0 of the STATUS register is also set.

The data may be read even if uncorrectable errors exist.

NOTE:

If the long mode (L) bit is set in the READ or WRITE command, no error checking is performed.

Bit 5 - Reserved.

Not used. Set to zero.

Bit 4 - ID Not Found

This bit is set to indicate that the correct cylinder, head, sector, or size parameter could not be found, or that a CRC error occurred in the ID field. This bit is set on the first failure and remains set even if the error is recovered on a retry. When recovery is unsuccessful, the Error bit (bit 0) of the STATUS register is also set.

For a SCAN ID command with retries enabled (T = 0), the Error bit in the STATUS register is set after ten unsuccessful attempts have been made to find the correct ID. With retries disabled (T = 1), only two attempts are made before setting the Error bit.

For a READ or WRITE command with retries enabled (T = 0), ten attempts are made to find the correct ID field. If there is still an error on the tenth try, an auto-scan and auto-seek are performed. Then ten more retries are made before setting the Error bit. When retries are disabled (T = 1), only two tries are made. No auto-scan or auto-seek operations are performed.

Bit 3 - Reserved.

Not used. Set to zero.

Bit 2 - Aborted Command

This bit is set if a command was issued or in progress while DRDY (Pin 28) was deasserted or WR FAULT (Pin 30) was asserted. The Aborted Command bit will also be set if an undefined command is written into the COMMAND register, but an implied seek will be executed.

Bit 1 - TRACK 000 Error (TK000)

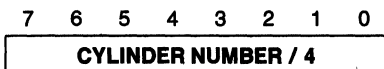
This bit is set only by the RESTORE command. It indicates that TRACK 000 (Pin 31) has not gone active after the issuance of 2048 stepping pulses.

Bit 0 - Data Address Mark

This bit is set during a READ SECTOR command if the Data Address Mark is not found after the proper Sector ID is read.

4.2 Reduce Write Current Register

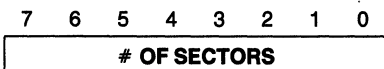
This register is used to define the cylinder number where RWC (Pin 33) is asserted:



The value (0–255) written into this register is internally multiplied by 4 to specify the actual cylinder where RWC is asserted. Thus a value of 01H will cause RWC to activate on cylinder 4, 02H on cylinder 8 and so on. RWC will be asserted when the present cylinder is greater than or equal to the cylinder indicated by this register. For example, one ST506 compatible drive requires precompensation on cylinder 128 (80H) and above. Therefore the REDUCE WRITE CURRENT register should be loaded with 32 (20H). A value of FFH will keep the RWC output inactive regardless of the actual cylinder number.

4.3 Sector Count Register

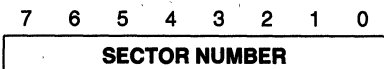
This register is used to define the number of sectors that need to be transferred to the buffer during a READ MULTIPLE SECTOR or WRITE MULTIPLE SECTOR command.



The value contained in the register is decremented after each sector is transferred to/from the sector buffer. A zero represents a 256 sector transfer, a one a 1 sector transfer, etc. This register is ignored when single sector commands are specified in the Command register.

4.4 Sector Number

This register holds the sector number of the desired sector:

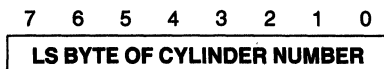


For a multiple sector command it specifies the first sector to be transferred. It is decremented after each sector is transferred to/from the sector buffer. The SECTOR NUMBER register may contain any value from 0 to 255. The ID Not Found bit will be set if the desired sector cannot be located on the track.

The SECTOR NUMBER register is also used to program the Gap 1 and Gap 3 lengths to be used when formatting a disk. See the WRITE FORMAT command description for further explanation.

4.5 Cylinder Number Low Register

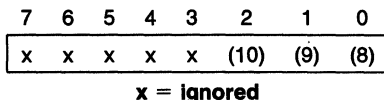
This register holds the lower byte of the desired cylinder number:



It is used in conjunction with the CYLINDER NUMBER HIGH register to specify a range of 0 to 2048 tracks.

4.6 Cylinder Number High Register

This register holds the three most significant bits of the desired cylinder number:

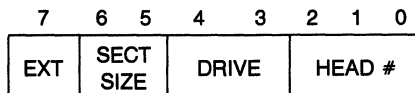


The 82064 contains a pair of registers that store the actual position where the R/W head are located. The CYLINDER NUMBER HIGH and LOW registers are considered the cylinder destination registers for seeks and other commands. The 82064 compares its internal registers to the destination registers and issues the number of steps in the right direction to make both sets of registers equal. After a command is executed, the internal cylinder position registers' contents are equal to the cylinder high/low registers. If a drive number change is detected on a new command, the 82064 automatically reads an ID field to update its internal cylinder position registers. This affects all commands except a RESTORE.

When a RESTORE command is executed, the internal head location registers are reset to zero while DIR and STEP move the heads to track zero.

4.7 Sector/Drive/Head (SDH) Register

The SDH register contains the desired sector size, drive number, and head number parameters. The format is shown below.



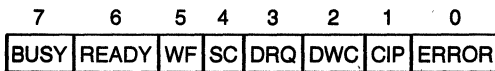
Both head number and sector size are compared against the disk's ID field. Head select and drive select lines are not available as outputs from the 82064 and must be generated externally.

Bit 7, the extension bit (EXT), is used to select between the CRC or ECC mode. When bit 7 = 1, the ECC mode is selected for the data field. When bit 7 = 0, the CRC mode is selected. The CRC is checked on the ID field regardless of the state of EXT. The SDH byte written into the ID field is different than the SDH Register contents. The recorded SDH byte does not have the drive number (DRIVE) written but does have the BAD BLOCK mark written.

Note that use of the extension bit requires the gap lengths to be modified as described in the WRITE FORMAT command description.

4.8 Status Register

The status register is a read-only register which informs the host of certain events. This register is a flow-through latch until the microprocessor reads it at which point the drive status lines are latched. The INTRQ line will be reset when this register is read. The format is:



Bit 7 - Busy

This bit is asserted when a command is written into the COMMAND register and, except for the READ command, is deasserted at the end of the command. When executing a READ command, Busy will be deasserted when the sector buffer is full. Commands should not be loaded into the COMMAND register when Busy is set. When the Busy bit is set, no other bits in the STATUS or ERROR registers are valid.

During other non-data transfer commands, Busy should be ignored as it will go active for short periods.

Bit 6 - Ready

This bit reflects the state of the DRDY (Pin 28) line at the time the microprocessor reads the status register. Transitions on the DRDY line will abort a command and set the aborted command bit in the error register.

Bit 5 - Write Fault

This bit reflects the state of the WR FAULT (Pin 30) line. Transitions on this line will abort a command and set the aborted command bit in the error register.

Bit 4 - Seek Complete

This bit reflects the state of the SC (Pin 32) line. Commands which initiate a seek will pause until Seek Complete is set. This bit is latched after an aborted command error.

Bit 3 - Data Request

The Data request bit (DRQ) reflects the state of the BDRQ (Pin 36) line. It is set when the sector buffer should be loaded with data or read by the host processor, depending upon the command. The DRQ bit and the BDRQ line remain high until BRDY indicates that the sector buffer has been filled or emptied, depending upon the command. BRDQ can be used for DMA.

Bit 2 - Data Was Corrected (DWC)

When set, this bit indicates that an ECC error has been detected during a read operation, and that the data in the sector buffer has been corrected. This provides the user with an indication that there may be a marginal condition within the drive before the errors become uncorrectable. This bit is forced to zero when not in the ECC mode.

Bit 1 - Command in Progress

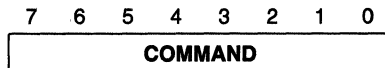
When this bit is set, a command is being executed and a new command should not be loaded until it is cleared. Although a command may be executing, the sector buffer is still available for access by the host processor. When the 82064 is no longer busy (bit 7 = 0) the status register can be read. If CIP is set, only the status register can be read regardless of which register is selected.

Bit 0 - Error

This bit is a logical OR of the contents of the error register. Any bit being set in the error register sets this bit. The host must read the ERROR register to determine what type of error occurred. This bit is cleared when a new command is loaded.

4.9 Command Register

This write-only register is loaded with the desired command:



The 82064 begins to execute immediately upon loading any value into this register. This register should not be written while the Busy or Command in Progress bits are set in the STATUS register. The INTRQ line (Pin 3) if set, will be cleared by a write to the COMMAND register.

Instruction Set

The 82064 WDC instruction set contains six commands. Prior to loading the command register, the host processor must first set up the Task Register File with the information needed for the command. Except for

the COMMAND register, the registers may be loaded in any order. If a command is in progress, a subsequent write to the COMMAND register will be ignored. A command is finished when the command in progress (CIP) bit in the STATUS register is cleared. See the Command Section for an explanation of each command.

COMMAND	7	6	5	4	3	2	1
RESTORE	0	0	0	1	R3	R2	R1 R0
SEEK	0	1	1	1	R3	R2	R1 R0
READ SECTOR	0	0	1	0	I	M	0 T
WRITE SECTOR	0	0	1	1	0	M	0 T
SCAN ID	0	1	0	0	0	0	0 T
WRITE FORMAT	0	1	0	1	0	0	0 0
COMPUTE CORRECTION	0	0	0	0	1	0	0 0
SET PARAMETER	0	0	0	0	0	0	0 S

R 3 - 0 = Rate Field

For 5 MHz WR Clock:

0000	—	≈ 35 μs
0001	—	0.5 ms
0010	—	1.0 ms
0011	—	1.5 ms
0100	—	2.0 ms
0101	—	2.5 ms
0110	—	3.0 ms
0111	—	3.5 ms
1000	—	4.0 ms
1001	—	4.5 ms
1010	—	5.0 ms
1011	—	5.5 ms
1100	—	6.0 ms
1101	—	6.5 ms
1110	—	3.2 μs
1111	—	16 μs

COMMAND	7	6	5	4	3	2	1
T =	Retry Enable						
T = 0	Enable Retries						
T = 1	Disable Retries						
M =	Multiple Sector Flag						
M = 0	Transfer 1 Sector						
M = 1	Transfer Multiple Sectors						
I =	Interrupt Enable						
I = 0	Interrupt at BDRQ time						
I = 1	Interrupt at end of command						
S =	Error Correction Span						
S = 0	5-bit Span						
S = 1	11-bit Span						

5.0 PROGRAMMING THE 82064

This section consists of two parts. The first part gives an explanation of each command, a flowchart showing the 82064's sequence of events, and the commands' sequence of events as seen by the host microprocessor. The second section shows flowcharts of general software routines and their PLM equivalent, for both polled and interrupt driven software.

The designer must remember that the 82064 expects a full sector buffer that can be isolated from the host during data transfers between the 82064 and the disk. Since the 82064 assumes a full sector buffer is available, it does not check for data overrun or underrun error conditions. If such a condition occurs, corruption of data will happen and the host will have no indication of an error. The design must guarantee against over-run and under-run conditions when not using the sector buffer approach.

5.1 Commands

A command is placed into the command register only after the Task Registers have been written with proper values. The Task Registers may be loaded in any order. A command, once started, can only be terminated by a hardware reset to the 82064. This may corrupt data on the disk by removing necessary control signals out of sequence.

The general sequence of a command is as follows:

- The host loads the Task Registers
- The host loads the Command Register
- The 82064 locates the correct cylinder
- Data transfer takes place
- The 82064 issues an interrupt

Restore Command - 0 0 0 1 R3 R2 R1 R0

The Restore command is used to position the heads to track 0. This command is usually issued to the 82064 on power-up to initialize internal registers. The user specified rate field (R3-R0) is stored internally for FUTURE use in commands with implied seeks.

The step rate value is not used with this command. The actual stepping rate used is dependent upon the handshake delay between the 82064 issuing a step pulse and the drive returning a seek complete for each track. After each step pulse is issued, the 82064 waits for a rising edge on the Seek Complete (SC) line before issuing the next pulse. If 8 index pulses are received without a rising edge on SC, the 82064 will switch to sampling the level of the SC line. If after 2048 step pulses the Track 00 signal has not gone active, the 82064 will terminate

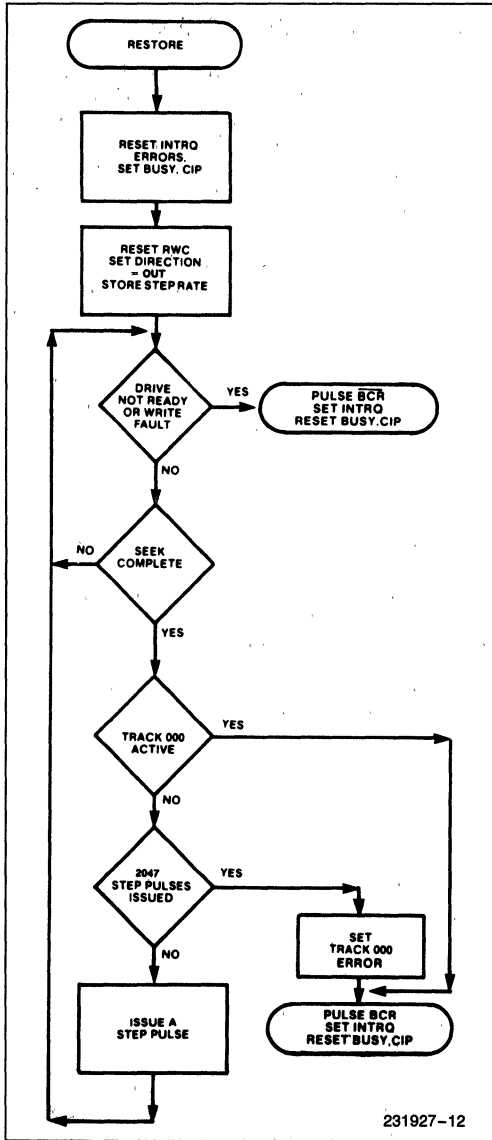


Figure 13. Restore Command Flow

the command, assert INTRQ and set the TRACK 000 bit in the Error Register. The command will terminate if WR Fault goes active or DRDY goes inactive at any time. Figure 13 is a flow chart of the command.

Seek Command - 0 1 1 1 R3 R2 R1 R0

The Seek command positions the heads to the cylinder specified in the Task Registers. The direction and num-

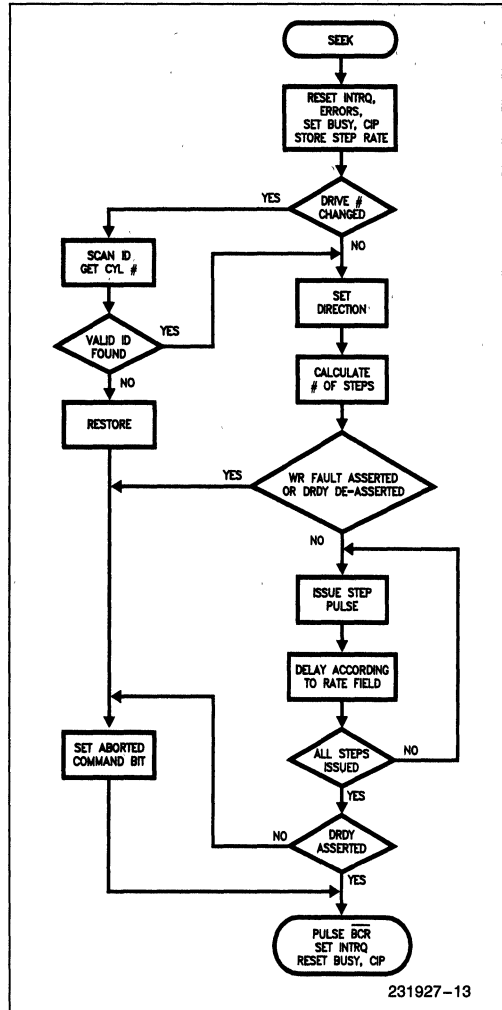


Figure 14. Seek Command Flow

ber of step pulses issued is calculated by comparing the cylinder high/low registers to an internal "present position" cylinder register. The present position register is updated after all step pulses are issued and the command is terminated.

The actual stepping rate is taken from the rate field bits (R3-R0) and stored for future use. The command terminates at once if WR FAULT goes active or DRDY goes inactive at any time. Figure 14 is a flowchart of the command.

Since the data transfer commands feature implied seeks, this command is of use mainly to those using multiple drives and software that can take advantage of overlapped seeks.

**Scan ID Command -
010000T**

The Scan ID command is used by both the 82064 and the host to update the SDH, the Sector Number, Cylinder and internal present position registers. Once the command is issued, the Seek Complete line is sampled until valid. The first ID field found, as indicated by the address mark, is loaded into the previously mentioned registers. The Bad Block bit will be set if detected, and the command will terminate. ID CRC errors will start the search sequence over for a maximum of 10 index pulses, but the registers will be loaded with whatever data the 82064 had perceived as ID information. Improper states on WR Fault on DRDY will terminate the command. Figure 15 is the flow chart of the command.

The main use for this command is to determine where the heads are currently located and what size the sectors are (i.e. 256, 512 etc.). Without this command, it would be necessary to recall the heads to track zero and then step out to the desired cylinder each time a drive was changed. Specifying the wrong sector size would yield an ID not found error. This command enables the system to read the disk drive to determine what size sectors were recorded.

**Read Sector Command -
00101MOT**

The READ SECTOR command is used to transfer one or more sectors of data from the disk to the sector buffer. Upon receipt of the READ SECTOR command, the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against an internal cylinder position register to see if they are equal. If not, the direction and number of steps are calculated and a seek takes place. If an implied seek is performed, the 82064 will search until a rising edge of SC is received. The WR FAULT and DRDY lines are monitored throughout the command.

Once the Seek Complete (SC) line is high (with or without an implied seek having occurred), the search for an ID field begins. If $T = 0$ (retries enabled), the 82064 must find an ID with the correct cylinder number, head, sector size, and CRC within 10 revolutions, or a Scan ID and re-Seek will be performed. The search for the proper ID will again be tried for up to 10 revolutions. If the correct sector is still not found, the appropriate error bits will be set and the command terminated. Data CRC errors will also be retried for up to 10 revolutions (if $T = 0$).

If $T = 1$ (retries disabled), the ID search must find the correct sector within 2 revolutions or the appropriate error bits will be set and the command terminated.

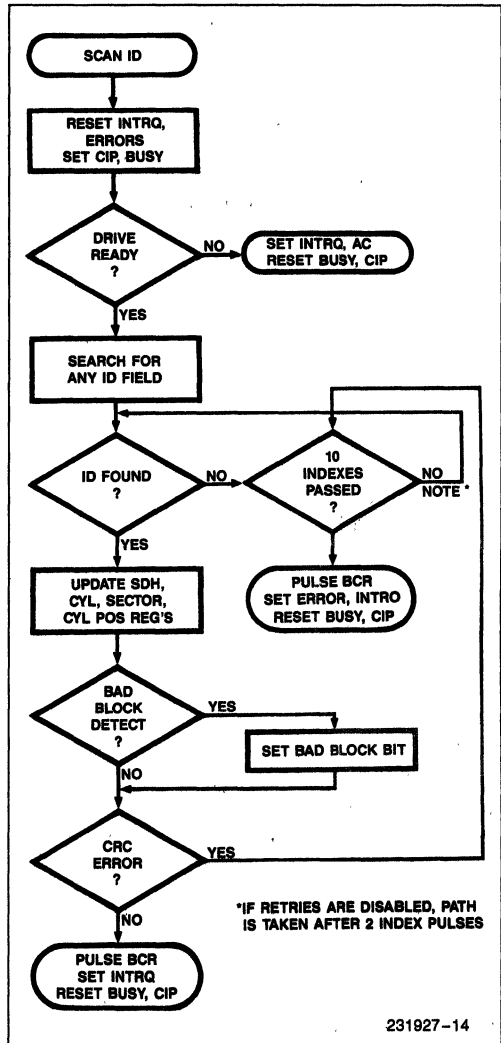


Figure 15. Scan ID Command Flow

Both the READ SECTOR and WRITE SECTOR commands feature a "simulated completion" to ease programming. DRQ/BDRQ will be generated upon detecting an error condition. This allows the same program flow for successful or unsuccessful completion of a command.

When the data address mark is found, the 82064 is ready to transfer data to the sector buffer. After the data has been transferred, the I bit is checked. If $I = 0$, INTRQ is made active coincident with BDRQ, indicating that a transfer of data from the buffer to the host processor is required. If $I = 1$, INTRQ will occur at the end of the command, i.e. after the buffer is unloaded by the host.

In summary then, READ SECTOR operation is as follows:

When M = 0 (READ SECTOR)

- (1) Host: Sets up parameters; issues READ SECTOR command.
- (2) 82064: Strobes $\overline{\text{BCR}}$.
- (3) 82064: Finds sector specified; asserts $\overline{\text{BCR}}$ and $\overline{\text{BCS}}$; transfers data to buffer.
- (4) 82064: Sets $\overline{\text{BCR}} = 1$, $\overline{\text{BCS}} = 0$.
- (5) 82064: Sets BDRQ = 1; DRQ = 1.
- (6) 82064: If I bit = 1 go to (9).
- (7) Host: Reads contents of sector buffer.
- (8) 82064: Waits for BRDY, then sets INTRQ = 1: END.
- (9) 82064: Sets INTRQ = 1.
- (10) Host: Reads out contents of buffer; END.

When M = 1 (READ MULTIPLE SECTOR)

- (1) Host: Sets up parameters; issues READ SECTOR command.
- (2) 82064: Assert $\overline{\text{BCR}}$.
- (3) 82064: Finds sector specified; asserts $\overline{\text{BCR}}$ and $\overline{\text{BCS}}$; transfers data to buffer.
- (4) 82064: Strobes $\overline{\text{BCR}}$; sets $\overline{\text{BCS}} = 0$.
- (5) 82064: Sets BDRQ = 1; DRQ = 1.
- (6) Host: Reads out contents of buffer.
- (7) 82064: Waits for BRDY; Decrements SECTOR COUNT; increments SECTOR NUMBER.
- (8) 82064: When BRDY = 1, if Sector Count = 0 then go to (10).
- (9) 82064: Go to (2).
- (10) 82064: Set INTRQ = 1; End.

A flowchart of the READ SECTOR command is shown in Figures 16A and 16B.

The M bit is set for multiple sector transfers. When M = 0, one sector is transferred and the SECTOR COUNT register is ignored. When M = 1, multiple sectors are transferred. After each sector is transferred, the 82064 decrements the SECTOR COUNT register and increments the SECTOR NUMBER register. The next logical sector will be transferred regardless of any interleave. Sectors are numbered at format time.

Multiple sector transfers continue until the SECTOR COUNT register equals zero, or the BRDY line goes active (low to high). If the SECTOR COUNT register is non-zero (indicating more sectors are to be transferred but the buffer is full), BDRQ will be made active and the host must unload the buffer. After this occurs, the buffer will again be free to accept the remaining sectors from the 82064. This scheme enables the user to transfer more sectors than the buffer memory has capacity for.

Write Sector Command – 0 1 1 1 0 M 0 T

The WRITE SECTOR command is used to write one or more sectors of data to the disk from the sector buffer. Upon receipt of a WRITE SECTOR command the 82064 checks the CYLINDER NUMBER LOW/HIGH register pair against the internal cylinder position register to see if they are equal. If not, the direction

and number of steps calculation is performed and a seek takes place. The WR FAULT and DRDY lines are checked throughout the command.

When the Seek Complete (SC) line is found to be true (with or without an implied seek having occurred), the BDRQ signal is made active and the host proceeds to load the buffer. Once BRDY goes high, the ID field with the specified cylinder number, head, and sector size is searched for. Once found, WR GATE is made active and the data is written to the disk. If retries are enabled (T = 0), and if the ID field cannot be found within 10 revolutions, a Scan ID and re-Seek are performed. If the correct ID field is not found within 10 additional revolutions, the ID Not Found error bit is set and the command is terminated. If retries are disabled, (T = 1) and if the ID field cannot be found within 2 revolutions, the ID Not Found error bit is set and the command is terminated.

During a WRITE MULTIPLE SECTOR command (M = 1), the SECTOR NUMBER register is decremented and the SECTOR COUNT register is incremented after the transfer to the disk takes place. During multiple sector transfers if BRDY is asserted after the first sector is transferred from the buffer, the 82064 will transfer the next sector before issuing BDRQ. The 82064 will set BDRQ and wait for the host processor to place more data in the buffer.

In summary then, the WRITE SECTOR operation is as follows:

When M = 0, 1 (WRITE SECTOR)

- (1) Host: Sets up parameters; issues WRITE SECTOR command.
- (2) 82064: Sets BDRQ = 1, DRQ = 1.
- (3) Host: Loads sector buffer with data.
- (4) 82064: Waits for BRDY = 0 to 1.
- (5) 82064: Finds specified ID field; writes sector to disk.
- (6) 82064: If M = 0, then set INTRQ = 1; END.
- (7) 82064: Increment SECTOR NUMBER register; decrement SECTOR COUNT register.
- (8) 82064: If SECTOR = 0, then set INTRQ = 1; END.
- (9) 82064: Go to (2).

A flowchart of the WRITE SECTOR command is shown in Figure 17.

Write Format Command 0 1 0 1 0 0 0

The WRITE FORMAT command is used to format one track using the Task Register File and the sector buffer. During execution of this command, the sector buffer is used for additional parameter information instead of sector data. Shown in Figure 18 is the contents of the sector buffer for a 32 sector/track format with an interleave factor of two. Each sector requires a two byte sequence. The first byte designates whether a bad block mark is to be recorded in the sector's ID field. A 00 Hex is normal; an 80H indicates a bad block mark for the sector. In the example of Figure 18, sector 04 will get a bad block mark recorded. Any attempt to access sector 4 in the future will terminate the command.

The second byte indicates the logical sector number to be recorded. This allows sectors to be recorded with any interleave factor desired. The remaining memory in the sector buffer may contain any value. Its only purpose is to generate a BRDY to tell the 82064 to begin formatting the track. An implied seek is in effect on this command. As for other commands, if the drive number has been changed an ID field will be scanned for cylinder position information before the implied seek is performed. If no ID field can be read (because the track had been erased or because an incomplete format had been used), an ID Not Found error will result and the WRITE FORMAT command will be aborted. This can be avoided by issuing a RESTORE command before formatting.

The SECTOR COUNT register is used to hold the total number of sectors to be formatted (01H = 1 sector; 00H = 256 sectors), while the SECTOR NUMBER

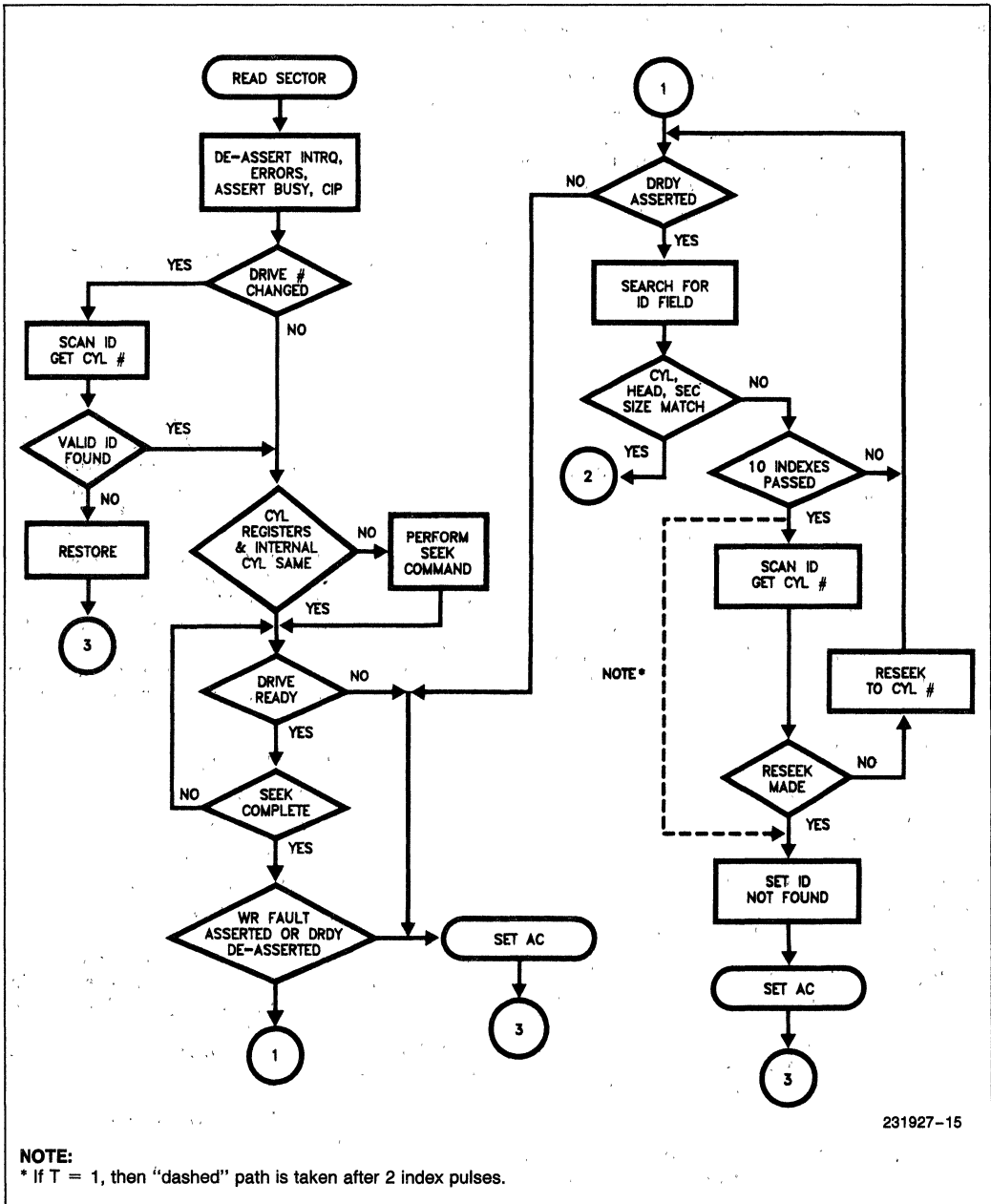
register holds the number of bytes (minus three) to be used for Gap 1 and Gap 3. For instance, if the SECTOR COUNT register value is 02H and the SECTOR NUMBER register value is 00H, then 2 sectors are written on a track and 3 bytes of 4EH are written for Gap 1 and Gap 3. The data fields are filled with FFH and the CRC is automatically generated and appended. All gaps are filled with 4EH. After the last sector is written, the track is filled with 4EH until the index pulse terminates the write. The Gap 3 value is determined by the drive motor speed variation, data sector length, and the interleave factor. The interleave factor is only important when 1:1 (no) interleave is used. The formula for determining the minimum Gap 3 length value is:

$$\text{Gap 3} = (2 * M * S) + K + E$$

- M = motor speed variation (e.g., 0.03 for ±3%)
- S = sector length in bytes
- K = 25 for interleave factor of 1
- K = 0 for any other interleave factor
- E = 7 if the sector is to be extended

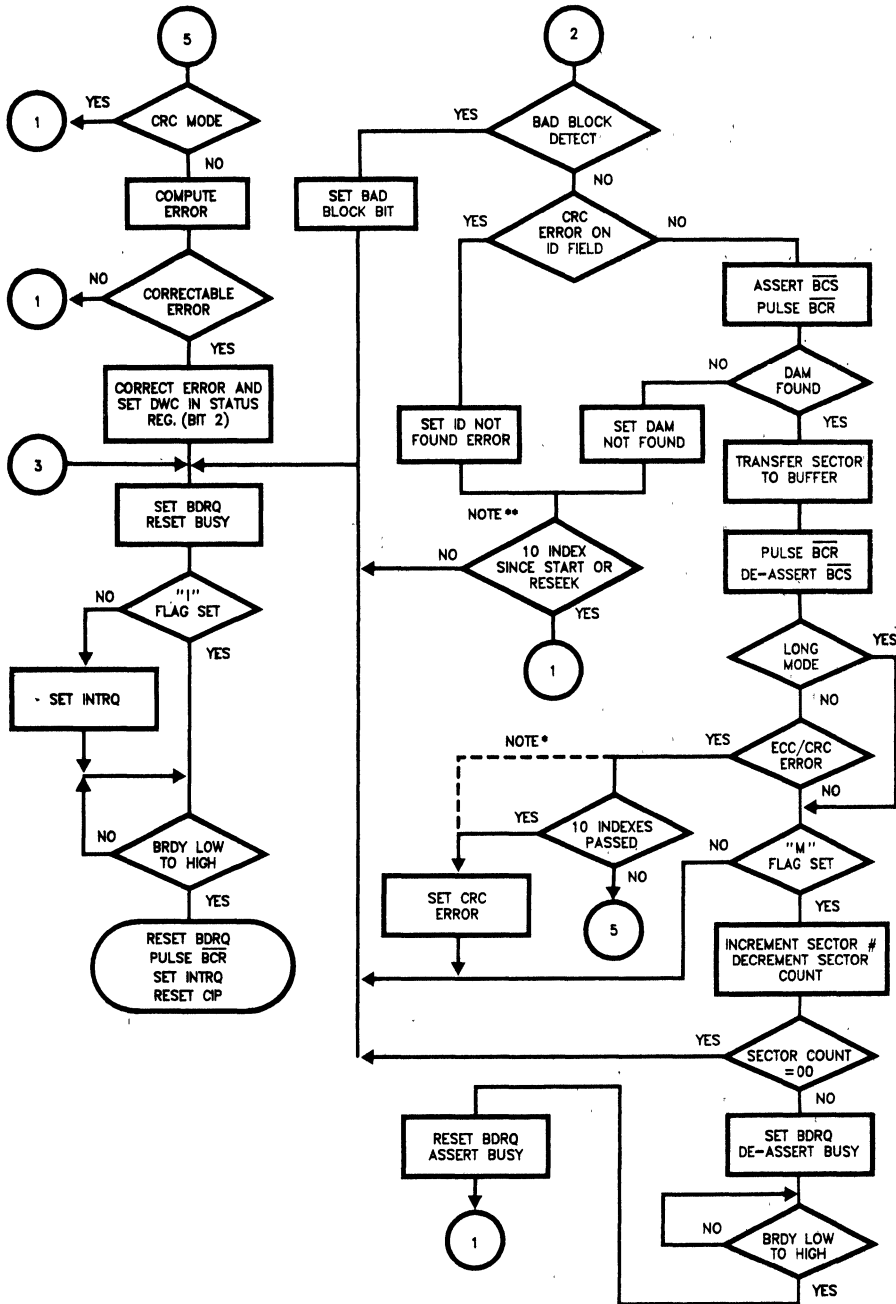
As with all commands, a WR FAULT or drive not ready condition, will terminate execution of the WRITE FORMAT command. Figure 19 shows the format that the 82064 will write on the disk. The extend bit in the SDH register must not be set during the Format command.

A flowchart of the WRITE FORMAT command is shown in Figure 20.



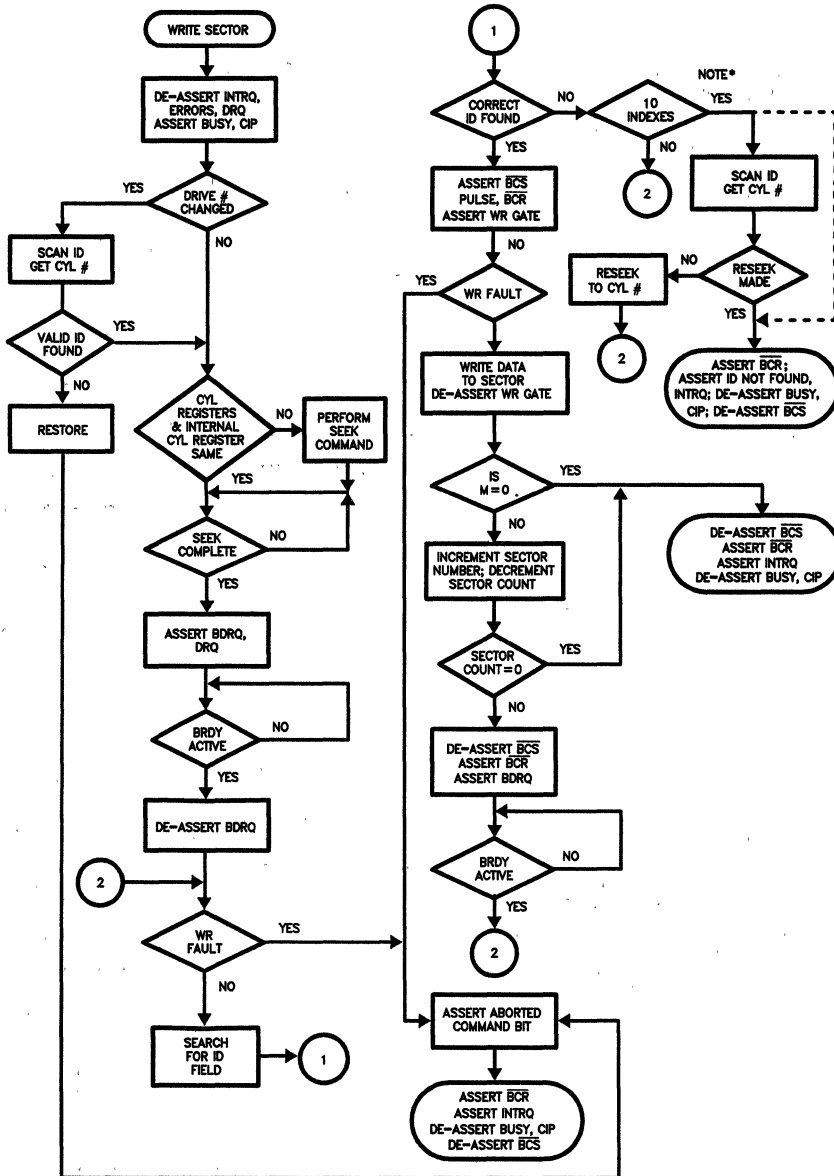
231927-15

Figure 16A. Read Sector Command Flow



* If T bit of command = 1, then dashed path is taken.
 ** If T bit of command = 1, then test is for 2 index pulses.

Figure 16B. Read Sector Command Flow (Continued)



231927-17

NOTE:
 *If retries are disabled, the "dashed" path is taken after 2 index pulses.

Figure 17. Write Sector Command Flow

00	00	00	10	00	01	00	11	00	02	00	12	00	03	00	13
80	04	00	14	00	05	00	15	00	06	00	15	00	07	00	17
00	08	00	18	00	09	00	19	00	0A	00	19	00	0B	00	1B
00	0C	00	1C	00	0D	00	1D	00	0E	00	1E	00	0F	00	1F
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA	AA
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figure 18. Sector Buffer Contents For Format

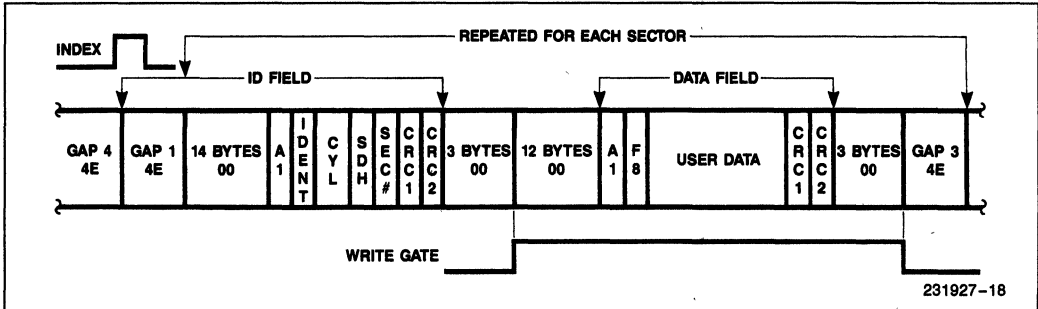


Figure 19. 82064 Sector Format

5.2 Software Section: General Programming

This section describes the software needed to communicate with the 82064 in order to store and retrieve data. This chapter describes the software in a general manner and Appendix B contains the actual implementation used to exercise the 82064 SBX board.

Polled Mode

As discussed in the Polled Interface Section, the 82064 does not directly support polled operation for data transfers without the addition of hardware. This section is based upon the polled interface as described in the Polled Interface Section.

The six 82064 commands can be divided into two groups, those with data transfers and those without. The commands that do not use the sector buffer are: Restore, Seek and Scan ID. The functions of each command are explained in the Commands Section. Figure 21 is a flowchart of a polled operation and a PLM example.

The last status that was read will contain any error conditions that might have occurred during the command.

For commands that do make use of the sector buffer, the size of the sector buffer will affect the software. If the sector buffer is equal in size to one sector, then a carry out of an address counter (for the sector buffer) as the buffer is being filled will indicate to the 82064 that the command should continue. If the sector buffer size is equal to two or more disk sectors, and only one sector is being transferred, then the carry out signal would not go active, and the 82064 will be forever waiting for BRDY. In this case an I/O port would have to be used to generate this signal for the 82064 so that command execution can finish. Figure 22 is a flowchart of the READ SECTOR command, and its PLM representation. The WRITE SECTOR and FORMAT TRACK commands are equivalent in terms of software interfacing. Their flowcharts and their PLM equivalents are shown in Figure 23.

Once the command register is written the 82064 requests a data transfer before locating the proper track. Once the buffer is filled and BRDY is asserted, the 82064 will locate the target track and sector. If the ID is not located before the selected number of retries have occurred, the 82064 will terminate the command. The data transferred to the sector buffer will not have been used. Once the command has finished (i.e., CIP = 0), the status and error registers will inform the host of an error.

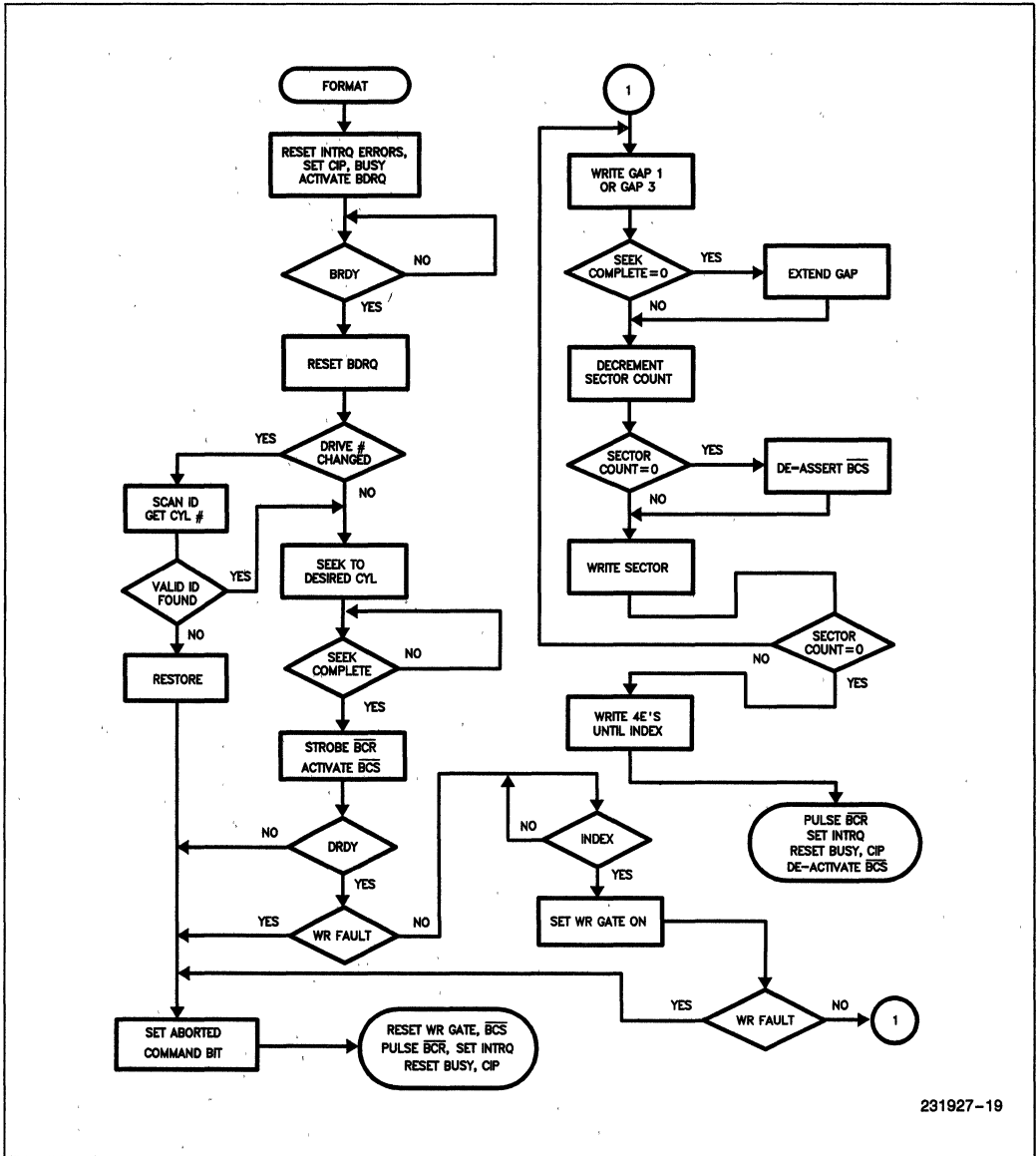
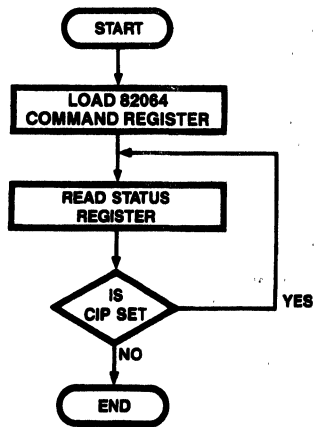


Figure 20. Write Format Command Flow

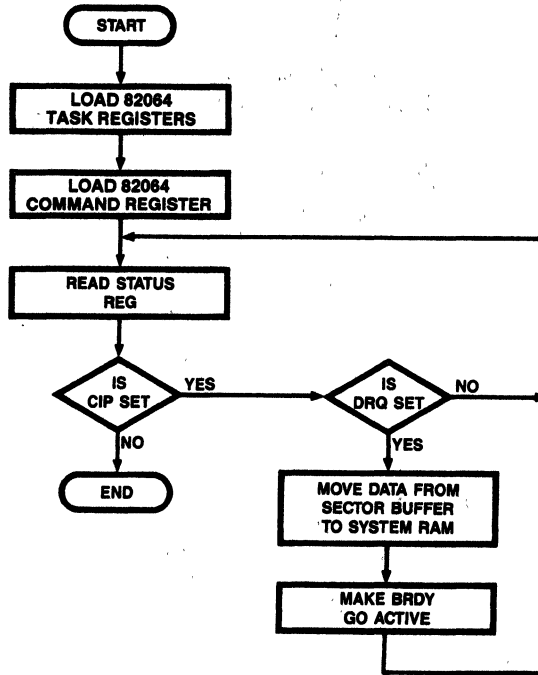


231927-20

```
Disk$Operation: Procedure;  
  Call Write$$82064$Task$Reg's; /* Write Task Registers */  
  Output (Command$Reg) = Command;  
  Status = Input (Status$Reg); /* Read Status Reg */  
  Do while Status and CIP = CIP; /* Wait until command finishes */  
    Status = Input (Status$Reg);  
  End;  
End Disk$Operation;
```

Figure 21. Polling Status

READ SECTOR COMMAND



231927-21

```

Disk$Operation: Procedure;
  Call Write$82064$Task$Regs;
  Output (Command $ Reg) = Command;
  Status = Input (Status$Reg);
  Do while Status and CIP = CIP;
    If Status and DRQ = DRQ then Do;
      Call Read$Data$From$Buffer;
      Output (BRDY$PORT) = 01;
    End;
    Status = Input (Status$Port)
  End;
End Disk$Operation;
  
```

Figure 22. Polling For Read Data

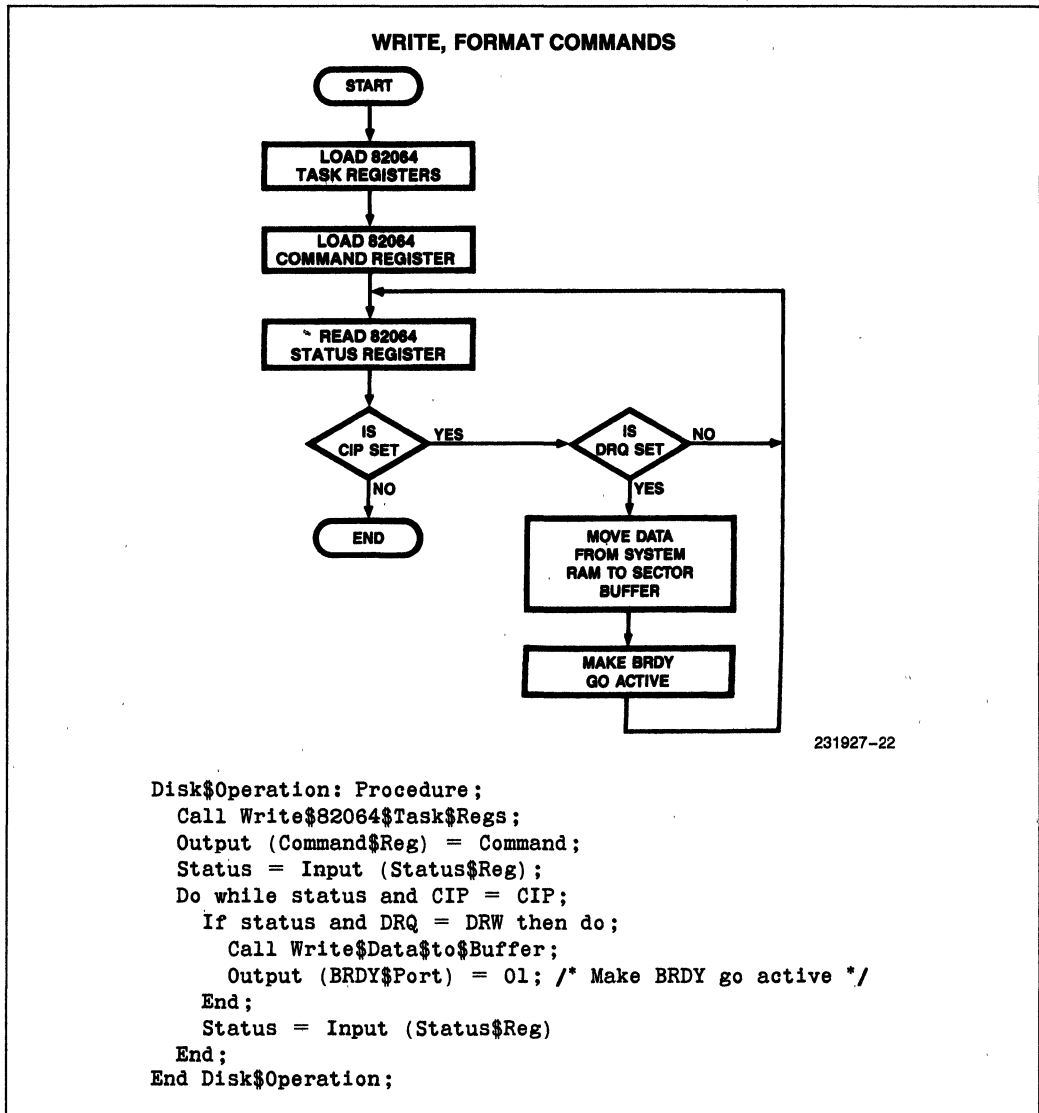


Figure 23. Polling For Write Data

```

Disk$Operation: Procedure;
  Call Write$82064$Task$Regs; /* Write registers */
  Output (Command$Reg) = Command; /* Start command */
  Status = Input (Status$Reg); /* Read status */
  Do while status and CIP = CIP; /* Is a command in progress */
    If status and DRQ = DRQ then do; /* Data transfer? = yes */
      If command = Read$Sector then
        Call Read$Data$From$Buffer; /* Remove data */
      Else Call Write$Data$to$Buffer; /* Send data */
      Output (BRDY$PORT) = 01; /* Toggle BRDY 0 to 1 */
    End;
  End Disk$Operation;

```

Figure 24. Complete Polled Flow

```

Start$Disk$Operation: Procedure;
  Call Write$82064$Task$Reg's;
  Output (Command $ Reg) = Command;
  End Start$Disk$Operation;

```

Figure 25. Interrupt Mode; Starting a Disk Transfer

Figure 24 is the PLM routine that allows for all six of the commands. It differs from the READ and WRITE routines in that the direction that data is to be transferred is determined by the command.

Figure 24 also works for multiple sector transfers. However, the BRDY signal must be generated in hardware (the carry-out of an address counter).

Interrupt Mode

Interrupt driven software is chosen when the microprocessor must execute other tasks and cannot sit waiting for the disk to reposition its heads, as in a polled environment. The delay in repositioning heads can be anything from a couple of milliseconds to a second or more.

The 82064's interrupt (INTRQ) pin goes active to indicate that the command has finished. The READ SECTOR command provides the programmable choice of having the interrupt occur at the end of the data transfer or the normal end of the command. The reason for this option is that when the 82064 signals that a data transfer is required (via BDRQ, DRQ) the disk has been read and the data has been placed in the buffer. The host would remove the data and issue BRDY. The 82064 would then issue an interrupt indicating that the command has finished. The interrupt procedure would

only have to read the status register. If the interrupt is issued at BDRQ the host would remove the buffer data and generate BRDY. At this point the status and error registers contain valid information. Generating an interrupt at BDRQ time may save some systems some software overhead.

The WRITE SECTOR and FORMAT commands do not have this option because the sector buffer is filled before the track and sector are located. Hence, there can be significant delays between asking for data and the command terminating.

In an interrupt driven environment, the 82064 can interface to a DMA controller for data transfers between the sector buffer and the host's RAM. If a DMA controller is not available an interrupt must be generated via the BDRQ line. However, BDRQ can stay active for long periods of time (until BRDY is generated). The interrupt sensing logic must take this into account to avoid being retriggered constantly. Intel's 8259A Interrupt Controller 8259A provides that capability. It should be programmed for edge triggered interrupts or the end of interrupt byte must not be issued until BDRQ is removed to prevent retriggering.

Figure 25 is a PLM example of starting a disk operation in an interrupt driven environment. The command starts, and some indefinite amount of time later an interrupt would be generated, indicating service is required.

```

End$of$Transfer: Procedure Interrupt;
    Status = Input (Status$Register);
    Output (8259A PIC) = End$of$Interrupt;
End End$of$Transfer;

```

Figure 26. Checking Status via Interrupt

```

Service$Disk$Controller: Procedure Interrupt;
    Status = Input (Status$Port);
    If Status and DRQ = DRQ then
        Call Transfer$Data$To/From$Buffer; /* Enable DMAC */
    Output (8259A PIC) = End$of$Interrupt;
End Service$Disk$Controller;

```

Figure 27. Complete Interrupt Procedure

If a DMA controller is used, it would have to be programmed and initialized before the command is issued to the 82064. Recall that once a data transfer between the microprocessor and 82064 has finished, BRDY must be set high. As long as BRDY is generated from hardware, no microprocessor intervention is needed. If BRDY is generated by an I/O port the microprocessor will have to perform this function (this will be the case with any system that has a sector buffer larger than one sector). (One option could be to generate an interrupt from the terminal count pin of the DMA controller. The microprocessor would then issue a BRDY.) Data transfers between host RAM and the sector buffer would be handled without microprocessor intervention. The interrupt would then signal that the command has finished as shown in Figure 26. The only operation the host processor would perform is to check the status register of the 82064 for any error conditions.

If BDRQ is used to generate an interrupt in addition to the normal interrupt, then the routines shown in Figure 27 will check the status register to see if a data transfer should be executed or if the command is finished. If DRQ is not set, the command has finished and any error conditions would be in the status register.

Another possibility would be to have separate interrupt routines for the two possible sources of interrupts (INTRQ, BRDQ). There would then be no need to test the status to see which interrupt had occurred.

6.0 APPLICATION EXAMPLE

This section shows an application using the 82064 interfaced to the SBX bus. A quick overview of the SBX bus is provided (pin descriptions, general wave forms) as a background for the application. Designing the 82064 onto an SBX Multimodule board was chosen to highlight the size and complexity differences between earlier TTL, MSI, LSI-based disk controller boards and what is possible using the 82064. Both the hardware and software sections will be applicable to most other designs using the 82064. This design example is called SBX82064 and does not represent a real product offered by Intel Corporation. Appendix C contains the schematic of the SBX board.

The advantage of the SBX Multimodule is that it permits the system to be tailored for specific needs with a minimum of effort. The advantage of an SBX based disk controller is that a current system can make use of the capacity, reliability and speed of a hard disk with no (or minimal) hardware redesign.

6.1 iSBX Bus Multimodule Boards

The iSBX Multimodule boards are small, specialized, I/O mapped boards which plug onto base boards. The iSBX boards connect to the iSBX bus connector and convert the iSBX bus signals to a defined I/O interface.

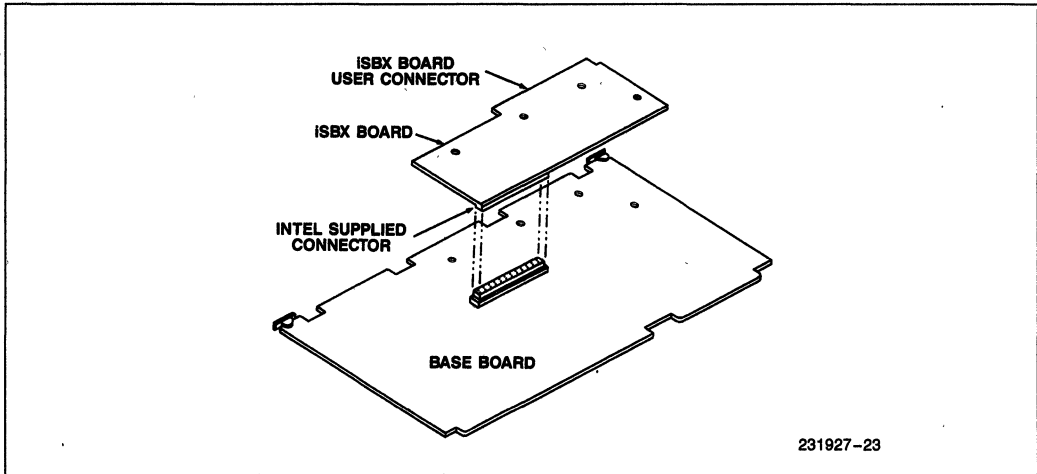


Figure 28. ISBX Multimodule Board Concept (Double Wide)

Base Boards

The base board decodes I/O addresses and generates the chip selects for the iSBX Multimodule boards. In 8-bit systems, the base board decodes all but the lower three addresses in generating the iSBX Multimodule board chip selects. In 16-bit systems, the base board decodes all but the lower order four addresses in generating the iSBX Multimodule board chip selects. Thus, a base board would normally reserve two blocks of 8 I/O ports for each iSBX socket it provides.

There are two classes of base boards, those with Direct Memory Access (DMA) support and those without. Base boards with DMA support are boards with DMA controllers on them. These boards, in conjunction with an iSBX Multimodule board (with DMA capability), can perform direct I/O to memory or memory to I/O operations.

ISBX Bus Interface

The iSBX bus interface can be grouped into six functional classes:

1. Control Lines
2. Address and Chip Select Lines
3. Data Lines
4. Interrupt Lines
5. Option Lines
6. Power Lines

Control Lines

The following signals are classified as control lines:

COMMANDS:

$\overline{\text{IORD}}$ (I/O Read)
 $\overline{\text{IOWRT}}$ (I/O Write)

DMA:

$\overline{\text{MDRQT}}$ (DMA Request)
 $\overline{\text{MDACK}}$ (DMA Acknowledge)
 $\overline{\text{TDMA}}$ (Terminate DMA)

INITIALIZE:

$\overline{\text{RESET}}$

CLOCK:

$\overline{\text{MCLK}}$ (iSBX Multimodule Clock)

SYSTEM CONTROL:

$\overline{\text{MWAIT}}$
 $\overline{\text{MPST}}$ (iSBX Multimodule Board Present)

Command Lines ($\overline{\text{IORD}}$, $\overline{\text{IOWRT}}$)

The command lines are active low signals which provide the communication link between the base board and the iSBX Multimodule board. An active command line, conditioned by chip select, indicates to the iSBX Multimodule board that the address lines are valid and the iSBX Multimodule board should perform the specified operation.

DMA Lines ($\overline{\text{MDRQT}}$, $\overline{\text{MDACK}}$, $\overline{\text{TDMA}}$)

The DMA lines are the communication link between the DMA controller device on the base board and the iSBX Multimodule board. $\overline{\text{MDRQT}}$ is an active high output signal from the iSBX Multimodule board to the base board's DMA device requesting a DMA cycle. $\overline{\text{MDACK}}$ is an active low input signal to the iSBX Multimodule board from the base board DMA device acknowledging that the requested DMA cycle has been granted. $\overline{\text{TDMA}}$ is an active high output signal from the iSBX Multimodule board to the base board. $\overline{\text{TDMA}}$ is used by the iSBX Multimodule board to terminate DMA activity. The use of the DMA lines is optional as not all base boards will provide DMA channels and not all iSBX Multimodule boards will be capable of supporting a DMA channel.

Initialize Lines (Reset)

This input line to the iSBX Multimodule board is generated by the base board to put the iSBX Multimodule board into a known internal state.

Clock Lines ($\overline{\text{MCLK}}$)

This input to the iSBX Multimodule board is a timing signal. The 10 MHz (+0%, -10%) frequency can vary from base board to base board. This clock is asynchronous from all other iSBX bus signals.

System Control Lines ($\overline{\text{MWAIT}}$, $\overline{\text{MPST}}$)

These output signals from the iSBX Multimodule board control the state of the system.

An active $\overline{\text{MWAIT}}$ (Active Low) will put the CPU on the board into wait states providing additional time for the iSBX Multimodule board to perform the requested operation. $\overline{\text{MWAIT}}$ must be generated from address (address plus chip select) information only. If $\overline{\text{MWAIT}}$ is driven active due to a glitch on the CS line during address transitions, $\overline{\text{MWAIT}}$ must be driven inactive in less than 75 ns.

The iSBX Multimodule board present $\overline{\text{MPST}}$ is an active low signal (tied to signal ground) that informs the base board I/O decode logic that an iSBX Multimodule board has been installed.

Address and Chip Select Lines

The address and chip select lines are made up of two groups of signals.

Address Lines: MA0-MA2

Chip Select Lines: $\overline{\text{MCS0}}$ - $\overline{\text{MCS1}}$

The base board decodes I/O addresses and generates the chip selects for the iSBX Multimodule boards. The base board decodes all but the lower order three addresses in generating the iSBX Multimodule board chip selects.

Address Lines (MA0-MA2)

These positive true input lines to the iSBX Multimodule boards are generally the least three significant bits of the I/O address. In conjunction with the command and chip select lines, they establish the I/O port address being accessed. In 16-bit systems, MA0-MA2 may be connected to ADR1-ADR3 of the base board address lines.

Chip Select Lines ($\overline{\text{MCS0}}$ - $\overline{\text{MCS1}}$)

In an 8-bit system, these input lines to the iSBX Multimodule board are the result of the base board I/O decode logic. $\overline{\text{MCS}}$ is an active low signal which conditions the I/O command signals and thus enables communication with the iSBX Multimodule boards.

6.2 The SBX82064 Design Example

The SBX82064 Multimodule board will interface an ST506 compatible drive to any host board having an SBX connector. Two restrictions on the disk drive are that there is a maximum of 2048 cylinders and/or 8 heads. The SBX connector cannot supply the power-up current requirements of the drive. The drive must be connected directly to the power supply. The SBX82064 in Appendix C does not support DMA transfers. The version in Appendix D does support DMA transfers. Since this multimodule has a 2 kbyte sector buffer, the host microprocessor must generate a BRDY by accessing an I/O port during data transfers.

The software for communicating to the SBX board is intended to be interrupt driven. Polling for data transfers is not supported. Reading the status without an interrupt is not recommended. During the times the 82064 is accessing the sector buffer, the SBX82064 will isolate itself from the host. To support polling, a hardware generated busy pattern should be driven onto the host's data bus as is shown in the Polled Interface section. The sector buffer stores up to 2 kbytes of disk data, for multiple sector transfers. The SBX board only interfaces to one drive (for space reasons), but four drives could be used with the addition of a read data multiplexor (one IC) and the drive data cables.

MWAIT is deactivated after allowing for the delayed RD and the access period of the 82064. This delay is accomplished with a 500 ns delay line. The first tap at 100 ns generates the read request to allow for the address setup margin. The next tap 400 ns later removes MWAIT to allow the host to continue.

Sector Buffer

The sector buffer consists of an address counter (using '1s393's) and a 2 kbyte static RAM. The address counter is incremented on the trailing edge of a valid RD or WR cycle, either host microprocessor or 82064 initiated. The counter is reset by a hardware reset, the 82064 buffer reset BCR, or by accessing an I/O port to provide software control. The 82064 will issue BCR each time BCS changes state (i.e. twice per sector). Resetting the buffer counter can be put under software control for multiple sector transfers. BRDY going high tells the 82064 that the buffer is available for its use. BRDY is generated by the address counter, by filling or emptying the entire buffer in multiple sector transfers, or from an I/O port when single sector transfers are done (since single sectors won't use all 2 kbytes of the buffer, the hardware signal will not be generated). When the 82064 is using the buffer, BCS will be low, and the RD or WR line will be pulsed every 1.6 microseconds.

When the 82064 is using the buffer it prevents access by the host by tristating the read, write, select and data lines with a low on BCS.

SDH Register Logic

The drive and head select bits must be latched externally to the 82064, since these outputs are not provided. An 8 bit latch is strobed on the trailing edge of the WR pulse when the SDH register is selected. The two drive select bits are then demultiplexed to provide a one of four drive select line. If multiple drives are used then these outputs would also be used to select which disk's read data line would be gated into the PLL.

Interrupts

While the interrupt line is programmable (to notify of an end of command or data transfer request for the Read Sector command only), software will ensure that the interrupt from the 82064 signifies command termination. The BDRQ line is OR'ed with the 82064's INTRQ line or BDRQ can generate its own interrupt. BDRQ is also gated off-board for a DMA controller.

Disk Interface

Figure 30 is a block diagram of the interface between the 82064 and the disk drive. The functional blocks are:

- Write Data Logic
- Read Data Logic (PLL)
- Drive Control

Write Data Logic

The WR DATA output requires a D flip-flop clocked at 10 MHz to complete the conversion of data to MFM. The output of this D flip-flop is true MFM and is sent to a delay line. A delay line determines the amount of delay for precompensation. No delay corresponds to shifting the data bit early; the first tap is approximately 12 ns of delay and is the "normal", or no delay and the second tap provides 12 ns of delay, referenced to the "normal" write data. Which output is selected is determined by the states on RWC, Early and Late. This function was generated with a 74s151 multiplexer. When RWC is inactive EARLY and LATE only select "normal" data since they are always active. The pre-compensated write data is then driven onto the data cable by an RS-422 driver.

Read Data Logic

The PLL generates the RD CLOCK that is used to decode the serial MFM data from the drive. A selected drive issues read data, unless WR GATE is active. A one-shot generates a pulse of 220–270 ns to provide the DRUN input. Only during an all zero's or one's field will the DRUN input stay high, as it will be retriggered every 200 ns (the minimum distance that separates continuous clock and data bits). As soon as DRUN is determined to be valid, the RD GATE output will go active, switching the PLL from the 10 MHz local clock input to disk data. The PLL will synchronize to the incoming serial data and generate a Read Clock of the proper timing and phase. The 82064 will then start to search for the address mark which is indicated by DRUN going low at the address mark.

No detail is provided herein on PLL design, as it is beyond the scope of this document. PLL design should be left to experienced designers, since minute changes in temperature and component values will drastically affect the soft error rate. As an alternative, several companies manufacture very high speed PLL chips for MFM encoded disk drives. Besides being fairly easy to design in, they reduce the number of components and board area needed for the sophisticated PLL.

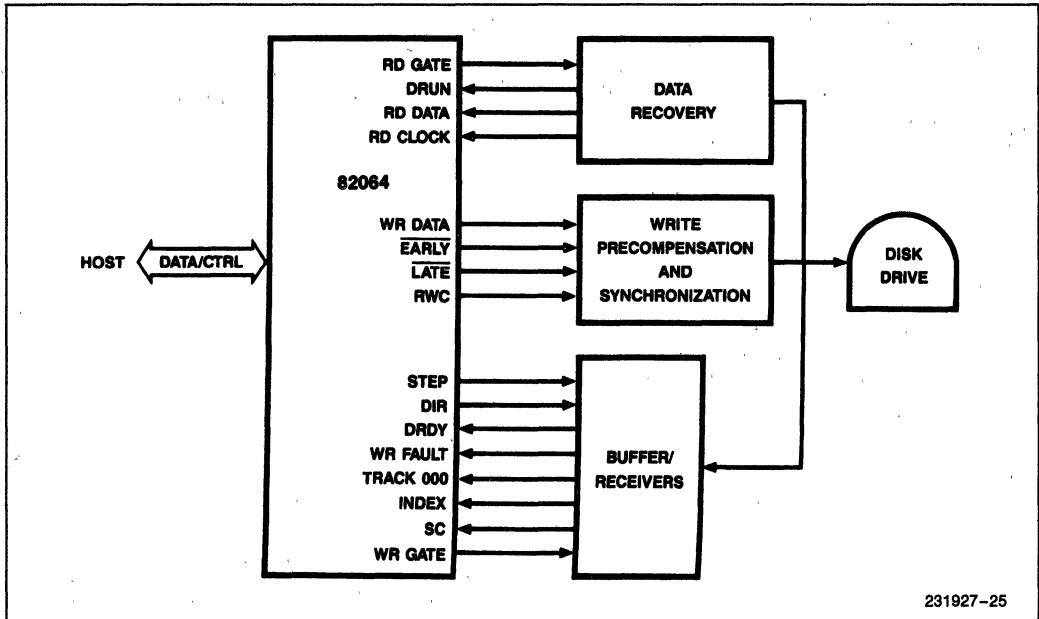


Figure 30. 82064 Disk Interface Block Diagram

6.3 Software Driver Overview

Presented in Appendix B is a listing of the software used to exercise the SBX 82064 board. Communication between the host software and the SBX driver routine is done through a structure located in system RAM. The host routine fills in required parameters, then passes the address of this communication block to the SBX driver routine. The driver routine pulls necessary values from this command block (CBL), executes a disk operation, then fills the CBL with the 82064's register contents, plus status and error information. The command block structure is shown in Figure 31.

Command	Byte
Rwc Reg	Byte
Sector Cnt.	Byte
Sector Num.	Byte
Cyl Low	Byte
Cyl High	Byte
SDH Reg	Byte
Status Reg	Byte
Error Reg	Byte
Host Buffer	Pointer

Figure 31

The host board did not have a DMA controller available, so an interrupt is issued from the BDRQ line and OR'ed with the 82064's interrupt line as interrupt sources were limited by the host. When an interrupt occurs, the interrupt procedure checks for either a data transfer, and executes it, or the completion of the command. If the interrupt signifies command completion, the interrupt procedure fills the command block with the 82064's task, status and error registers.

In this example, the host software examines one byte in the command block and until this byte is changed to a 00, no other command blocks will be passed to the disk driver routine. An alternative would be to issue a software interrupt to notify the microprocessor that the disk operation has finished and the command block contains parameters from the last operation and that a new disk command could start.

The driver for this example allows polling for non-data transfer commands, and must use interrupts for data transfers. As mentioned earlier, microprocessor intervention is required since the sector buffer is much larger than one sector and will not generate a BRDY. The microprocessor must write to an I/O port, which sets BRDY, after each host to sector buffer transfer. An

actual software implementation would not include the polling and interrupt routines together, as only one method would generally be used.

The calling routine, which would normally be a directory program, places the values for which sector, number of sectors, etc., in the CBL. The disk routine is called and the address of this structure is passed on the stack. The disk driver places these parameters in the 82064's Task registers and initiates a command.

If the interrupt driven method was chosen, the disk driver routine returns to the calling routine. This permits other processing to be performed while the disk is executing a command. At some point, an interrupt will be generated, either from BRDY or INTRQ. Control will pass to the driver and the status register will be checked. If a data transfer is needed, either the microprocessor can transfer data or a DMA controller can perform the function. Once the transfer of data to the buffer is finished the microprocessor must set BRDY through an I/O port.

APPENDIX A ST506 INTERFACE

THE ST506 INTERFACE

The ST506 interface is a modified version of Shugarts floppy disk drive interface and has been promoted by Seagate Technology. This interface is intended to be easy and low in cost to implement, yet provide a medium level of performance. The interface rigidly defines several areas: the hardware interconnects, the data transfer rate, the data encoding method, and how the disk is formatted.

Data Transfer Rate

The data transfer rate depends upon the linear bit density of the disk media and the speed at which the disk spins. ST506 specifies a 5 Mbit/second transfer rate. The typical ST506 drive has a nominal linear density of 10,416 bytes and a disk speed of 3600 rpm, which yields a 5 Mbit/second data transfer rate. No deviation from 5 M/bits second is allowed.

Increasing the linear density to increase storage capacity would require a decrease in disk speed. Otherwise, the data rate would increase. This decrease in disk speed would cause access times to increase, which many would deem unacceptable. To increase storage capacity, and remain ST506 compatible, either the number of cylinders and/or the number of platters can increase.

Data Encoding

ST506 requires that the serial data, sent between the drive and the controller, be encoded according to MFM rules. The basic unit of storage is a bit cell, which stores one bit information. This bit cell is divided into two halves, consisting of a clock bit and a data bit (see Figure A-1).

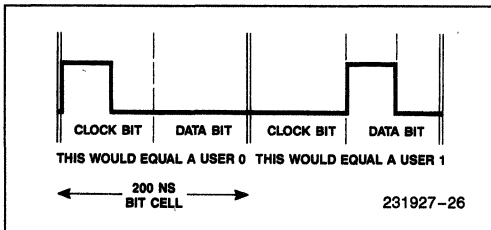


Figure A-1

The encoding rules for MFM are fairly simple:

1. A clock bit is written when the previous and the current bit cell does not contain a data bit.
2. A data bit is written whenever there is a "one" from the user.

Sync fields are composed of zeroes which generates a series of clock bits in the bit cell's. A phase lock loop locks on to the data stream during this period and generates a signal of the proper phase and frequency which is used to decode the combined clock and data serial data stream.

Disk Format

All disk media must be written with a specified format so that data may be reliably stored and retrieved. The smallest unit of controller accessible data is the sector which typically contains sync fields, ID fields, and a data field, and buffer fields.

The format of the disk required by ST506 is shown in Figure A-2. It should be noted that this format is fixed in the 82064. The user has options only for GAP1 and 3 length (when changing sector size or ECC) and whether to have 82064 CRC checking or user supplied ECC syndrome bits.

Gap 1 - Index Gap

Gap 1 serves two purposes. The first is to allow for variations in the index pulse timing due to motor speed variations. The second purpose is to allow a small delay to permit a different head to be selected without missing a sector. This is more of a data transfer optimization function and requires the disk controller to know which head is to be selected, when the last sector of a track has been read, and the next logical sector in the file exists on another platter. The 82064 does not switch heads automatically. Whether this scheme can be used or not depends upon the μ P being able to alter one register in the 82064, before the next sector passes beneath the heads.

This gap is typically 12 bytes long and is written by the 82064 as 4E Hex.

Gap 2 - Write Splice Gap

This gap follows the CRC bytes of the ID field and continues up to the data field address mark. When updating a previously written sector, motor speed variations could turn on the write coil, as the head was passing over the ID field. This gap prevents this from occurring. The value written is OOH and also serves as the PLL sync field for the data field. The minimum value is determined by the "lock up" performance of the PLL. The 82064 writes sixteen bytes for this field once WG is activated. The user has no control over this field.

Gap 3 - Post Data Field Gap

Gap 3 is very similar to Gap 2 as it is used as a speed tolerance buffer also. Without this gap, and with the motor speed varying slightly, it would be possible for the upcoming sector's sync field and ID field to be overwritten. This value is '4E' H and is typically 15 bytes long. The 82064's Gap 3 length is programmable. The exact value is dependent upon several factors. Refer to 82064 Format command, Software Section: General Programming Section.

Gap 4 - Track Buffer Gap

This gap follows the last sector on a track and is written until an index pulse is received. Its purpose is to prevent the last sector from overflowing past the index gap, and absorb track length variations when ECC is used (ECC uses more bytes than CRC). The value is

'4E' H and is about 320 bytes when CRC and 256 byte sectors are used. The 82064 writes this field only during formatting. The user has no control over the number of bytes written with the 82064.

ID Fields

The controller uses ID fields to locate any individual sector. An address mark of two bytes precedes the ID field and the data field in a sector. An address mark tells the controller the nature of the upcoming information. ID fields are used by the disk controller and are not passed to the host.

Sector Interleaving

Sector interleaving occurs when logical sectors are in a non-sequential order, which is determined during formatting. An advantage is that there is a delay between logically sequential sectors. This delay can be used for data processing and then deciding if the next sector should be read. Without interleaving, the next sector could slip by, imposing a one revolution delay (approx. 16.7 ms). An additional benefit to this delay is that bus utilization is reduced by spreading the data transfer over a greater amount of time. The delay between sectors can be determined as follows:

$$\frac{1 \text{ Revolution Period}}{\text{Sectors/Track}} \times (\text{Interleave factor} - 1) = \text{Delay}$$

For the typical ST506 drive with four-way interleaving this yields 1.57 ms of delay.

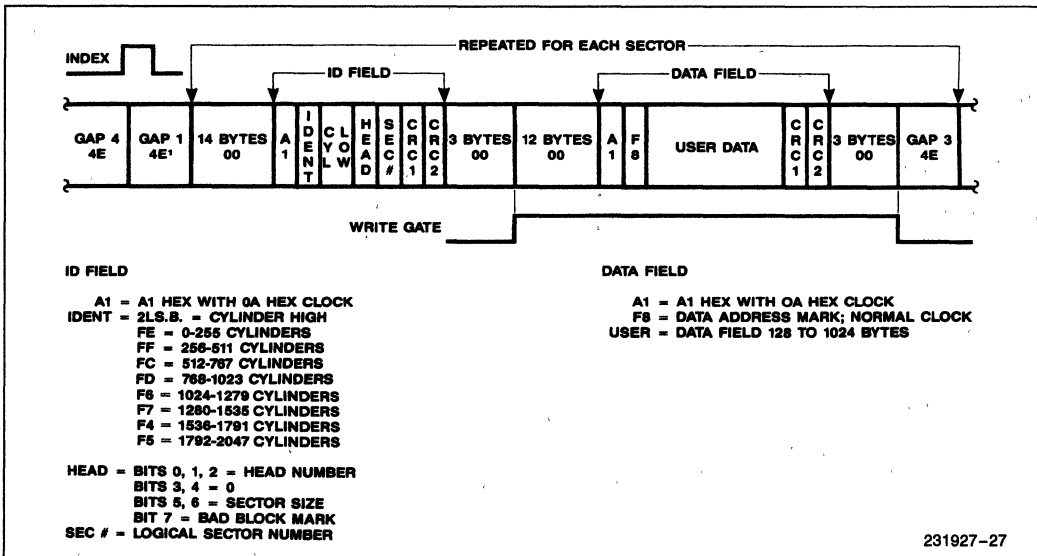
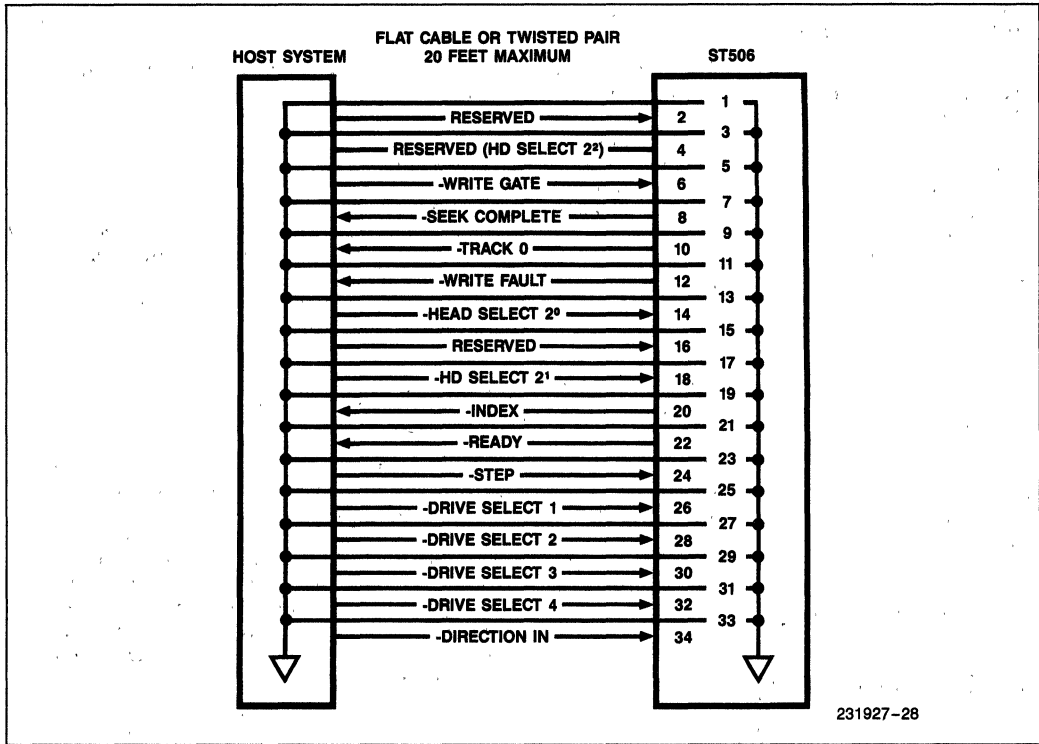


Figure A-2. Format Field



231927-28

Figure A-3

The disadvantage to interleaving is that file transfers take longer, which may slow down the overall system. A four-way interleaved disk will have the transfer rate reduced to an average of 1.25 Mbit/sec.

The 82064 leaves the logical sector sequence to the user.

ELECTRICAL INTERFACE

The interface to the ST506 drive is divided into three categories and they are:

- 1. control signals,
- 2. data signals,
- 3. power.

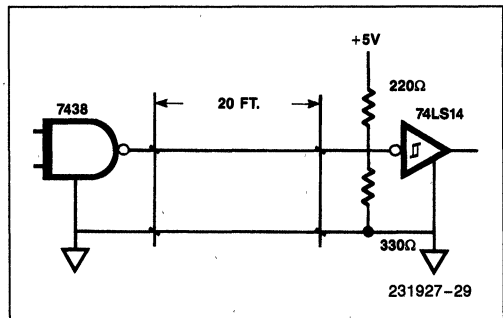
Control Signals

The functions of the control signals are not covered in detail here. Their purpose can be found in the pin descriptions section. All control lines are digital in nature and either provide signals to the drive or inform the

host of certain conditions. A diagram of the 34 pin control connector is shown in Figure A-3.

The driver/receiver logic diagram is shown in Figure A-4 and the electrical characteristics are:

	Voltage	Current
True	0.0 VDC to 0.4 VDC	-40 mA (IOL max.)
False	2.5 VDC to 5.25 VDC	250 μ A (IOH open)



231927-29

Figure A-4

Data Signals

The lines associated with the transfer of read/write data between the host and the drive are differential in nature and may not be multiplexed between drives. There is one pair of balanced lines for each read and write data line per drive and must conform to the RS-422 specification. Figure A-5 shows the receiver/transmitter combination.

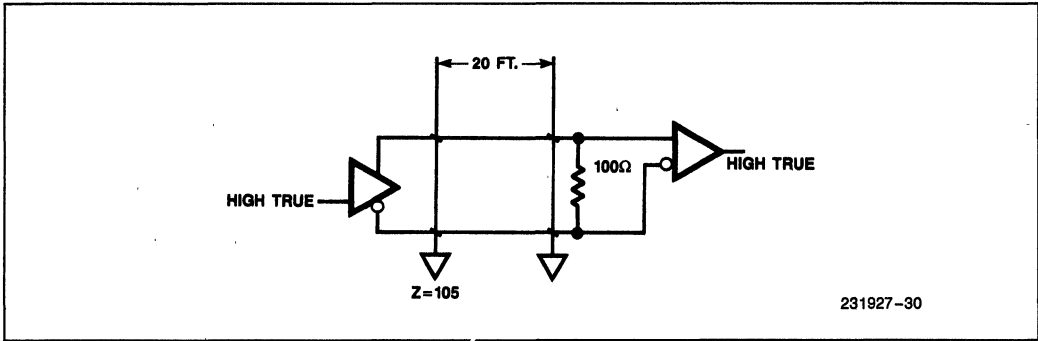
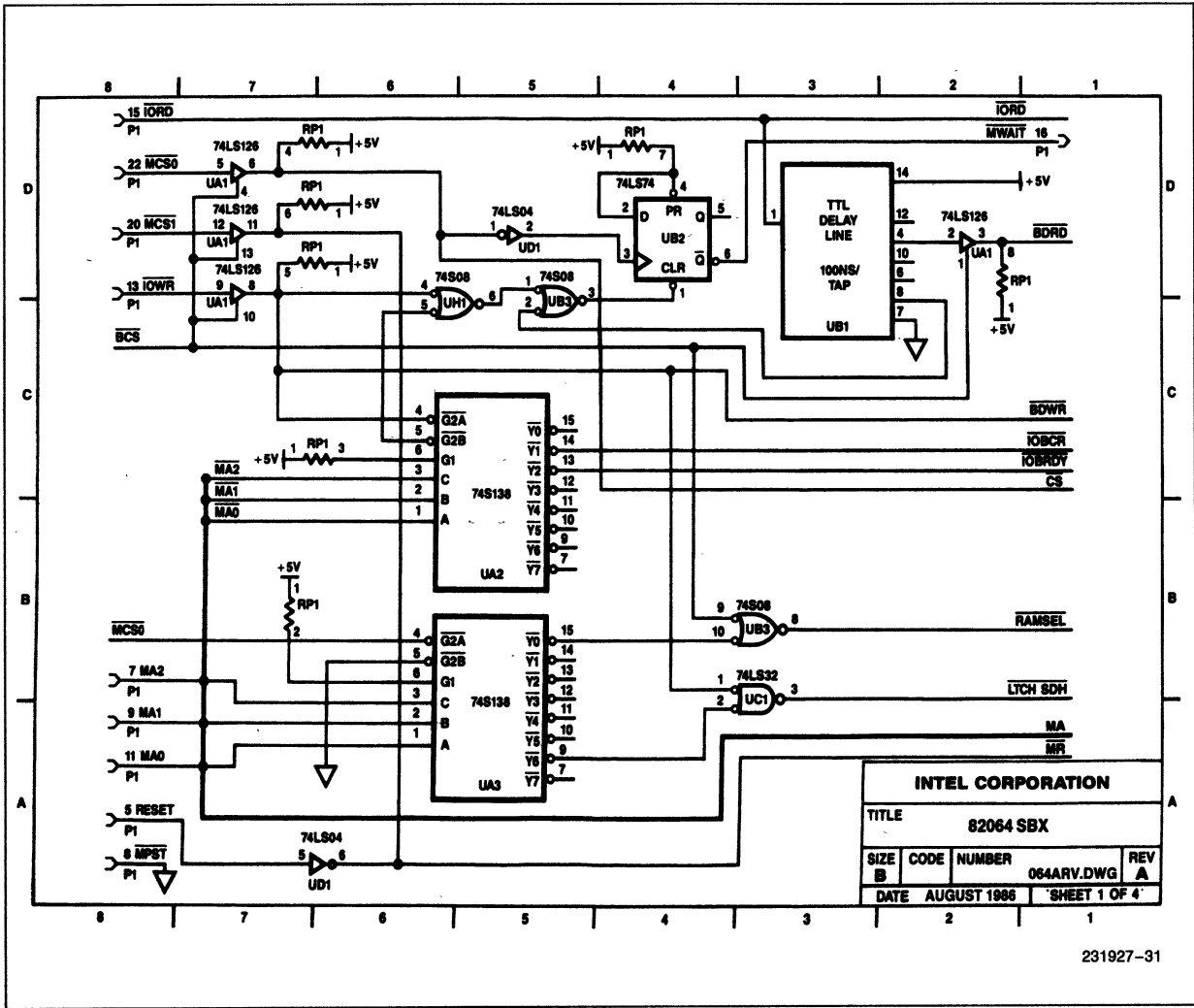


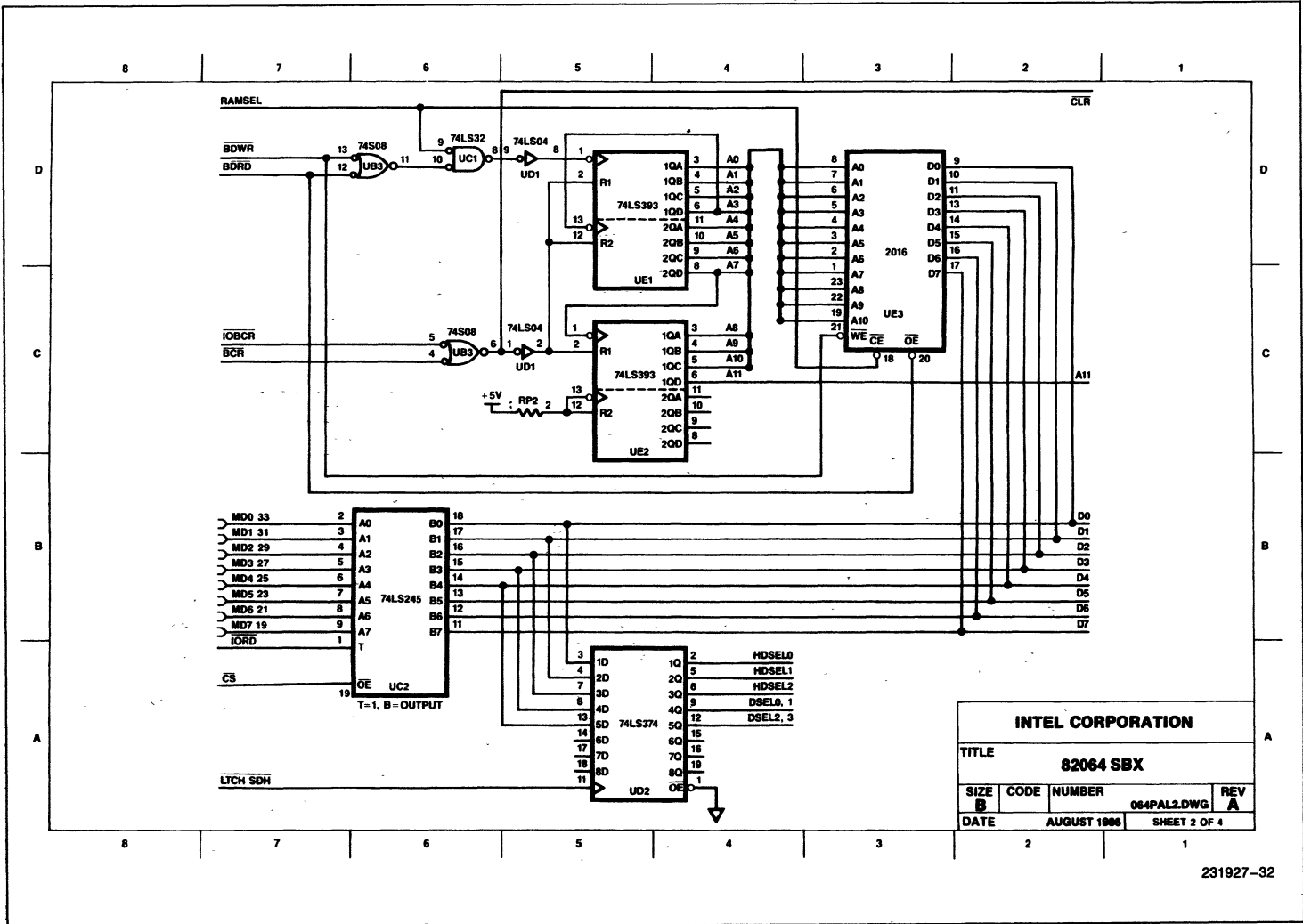
Figure A-5. E1A RS22 Driver/Receiver Pair Flat Ribbon or Twisted Pair

APPENDIX B
SCHEMATICS

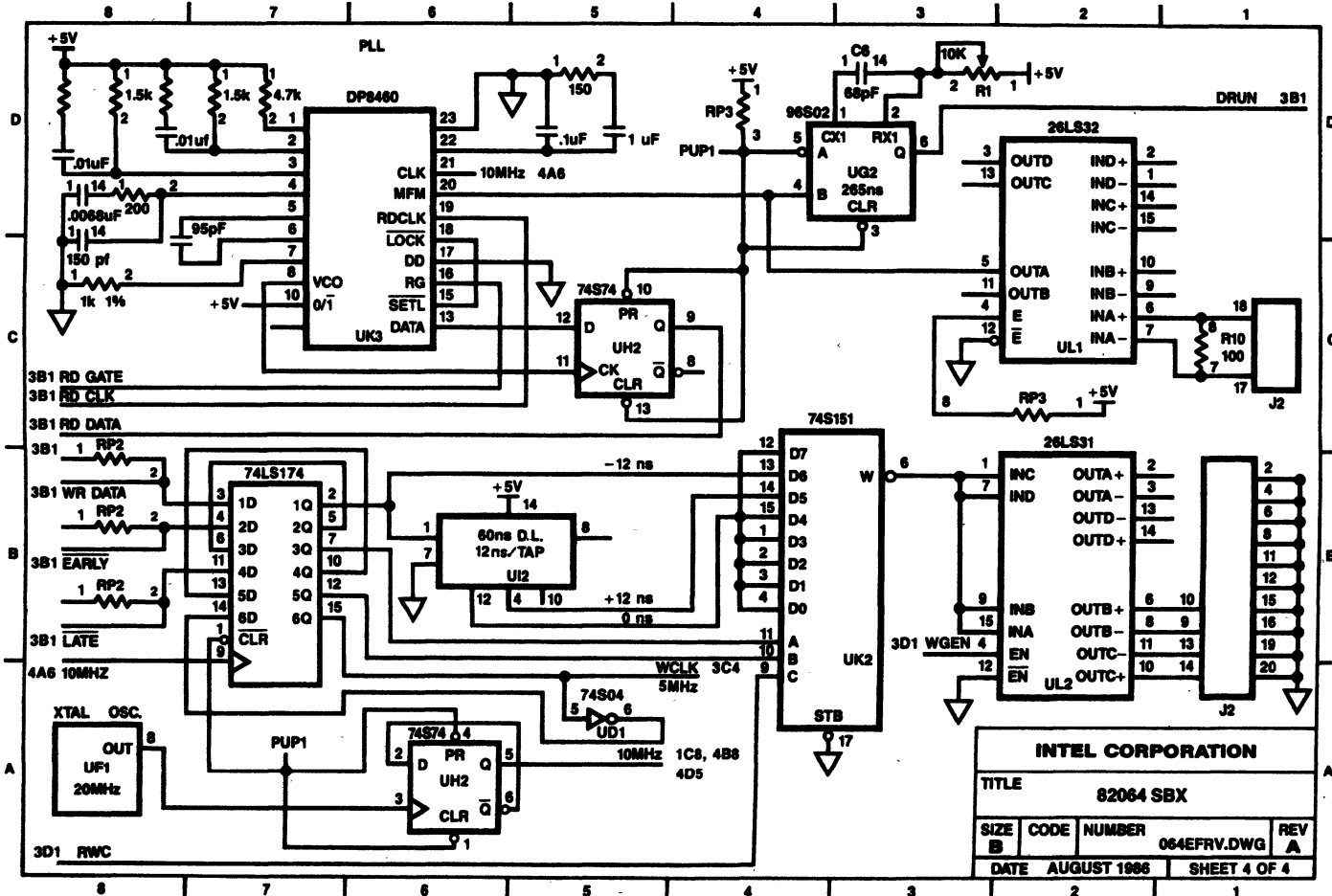


INTEL CORPORATION				
TITLE				
82064 SBX				
SIZE	CODE	NUMBER	REV	
B		064ARV.DWG	A	
DATE		AUGUST 1986		
SHEET 1 OF 4				

231927-31



INTEL CORPORATION			
TITLE		82064 SBX	
SIZE	CODE	NUMBER	REV
B		064PAL2.DWG	A
DATE	AUGUST 1986		SHEET 2 OF 4



INTEL CORPORATION				
TITLE				
82064 SBX				
SIZE	CODE	NUMBER	REV	
B		064EFRV.DWG	A	
DATE	AUGUST 1986	SHEET 4 OF 4		

4-77

APPENDIX C

This appendix contains a schematic of the previous design using PAL's to replace the random logic. The previous design could not do DMA transfers and inserted a large delay when transferring data from buffer RAM to the system. The PAL version does do DMA transfers and buffer reads happen at full SBX bus speed. One other minor change was to replace the 500 ns delay line with a 74LS164, which is a more cost effective solution.

This schematic is only a paper design since only random logic was replaced with the PAL's.

PAL Equation's

PAL - Page 1:

$$\text{BDRD/} = (\text{IORD/} * \text{MDACK/}) + (\text{IORD/} * \text{MCSO/} * \text{MAO} * \text{MA1} * \text{MA2}) + (\text{DELAYED-READ/} * \text{CLK}) \text{ IF BCS}$$

$$\text{LTCHSDH/} = (\text{MCSO/} * \text{MAO/} * \text{MA1} * \text{MA2} * \text{IOWR/})$$

$$\text{RAMSEL/} = (\text{MCSO} * \text{MAO} * \text{MA1} * \text{MA2}) + (\text{BCS/}) + (\text{MDACK/})$$

$$\text{IOBRDY/} = (\text{MCS1/} * \text{MAO/} * \text{MA1} * \text{MA2/} * \text{IOWR/})$$

$$\text{IOBCR/} = (\text{MCS1/} * \text{MAO} * \text{MA1/} * \text{MA2/} * \text{IOWR/})$$

$$\text{BDWR/} = (\text{IOWR/}) \text{ IF BCS}$$

$$\text{CS/} = (\text{MCSO/}) \text{ IF BCS}$$

$$\text{CLK} = (\text{MCSO/} * \text{MAO} * \text{MA1/} * \text{MA2/}) + (\text{MCSO/} * \text{MAO/} * \text{MA1} * \text{MA2/}) + (\text{MCSO/} * \text{MAO} * \text{MA1} * \text{MA2/}) + (\text{MCSO/} * \text{MAO/} * \text{MA1/} * \text{MA2}) + (\text{MCSO/} * \text{MAO} * \text{MA1/} * \text{MA2}) + (\text{MCSO/} * \text{MAO/} * \text{MA1} * \text{MA2}) + (\text{MCSO/} * \text{MAO} * \text{MA1} * \text{MA2})$$

PAL - Page 2:

$$\text{MINTR1/MDRQT} = (\text{PIN1})$$

$$\text{MINTRO} = (\text{PIN2}) + (\text{INTRQ})$$

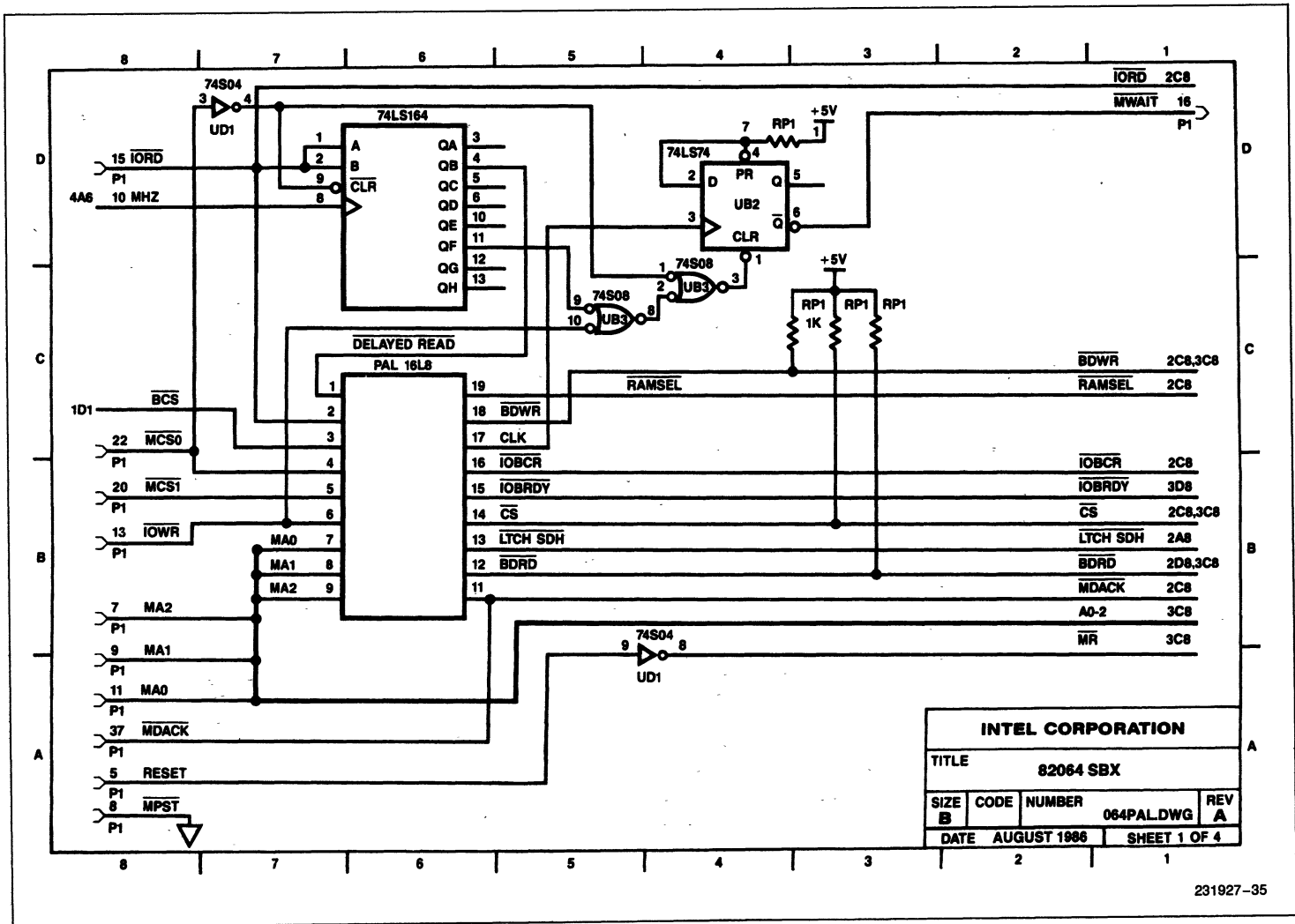
$$\text{COUNT} = (\text{BDWR/} + \text{BDRD/}) * (\text{RAMSEL/})$$

$$\text{RSTCOUNT} = (\text{IOBCR/}) + (\text{BCR/})$$

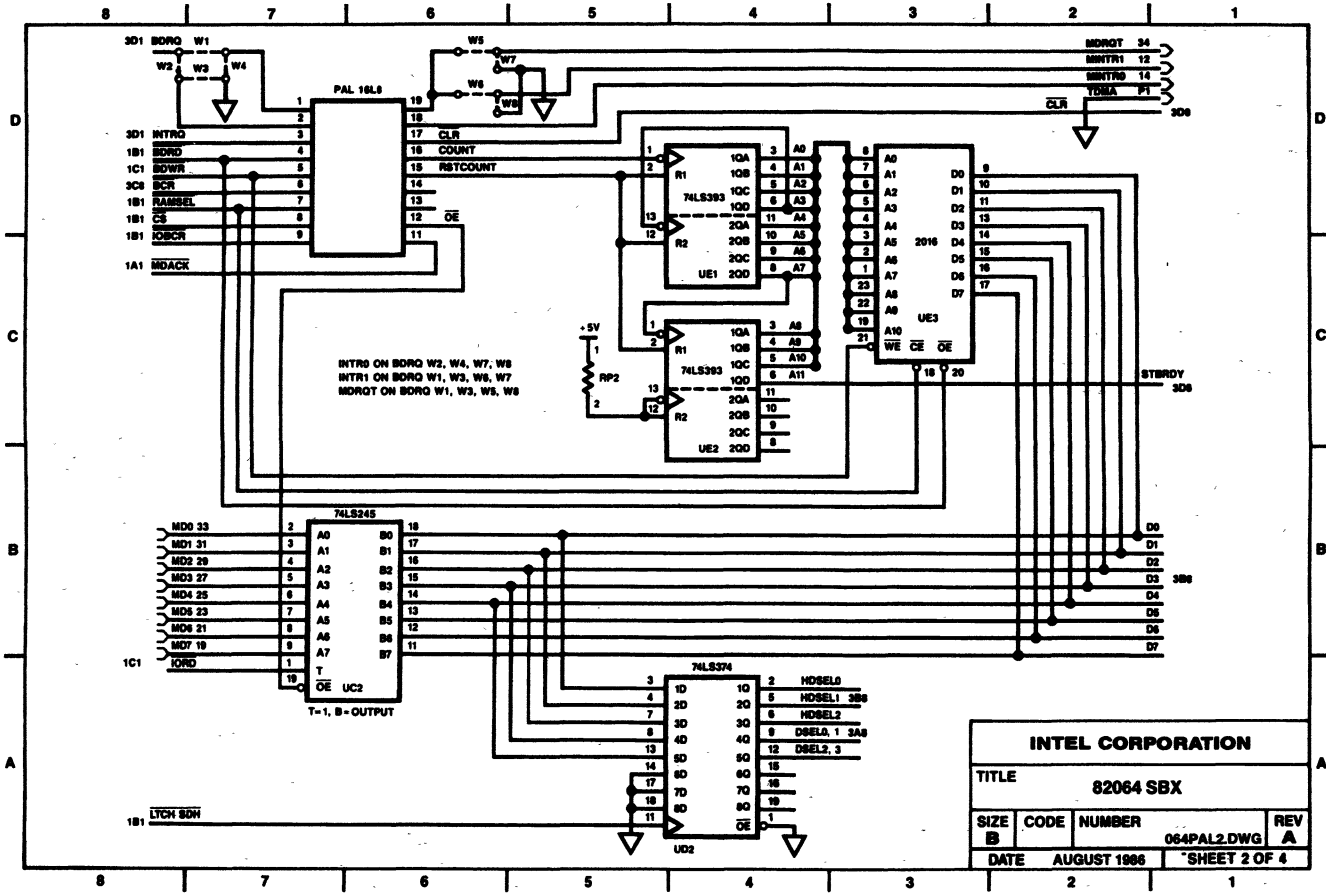
$$\text{OE/} = (\text{MDACK/}) + (\text{CS/})$$

$$\text{CLR/} = (\text{IOBCR/}) + (\text{BCR/})$$

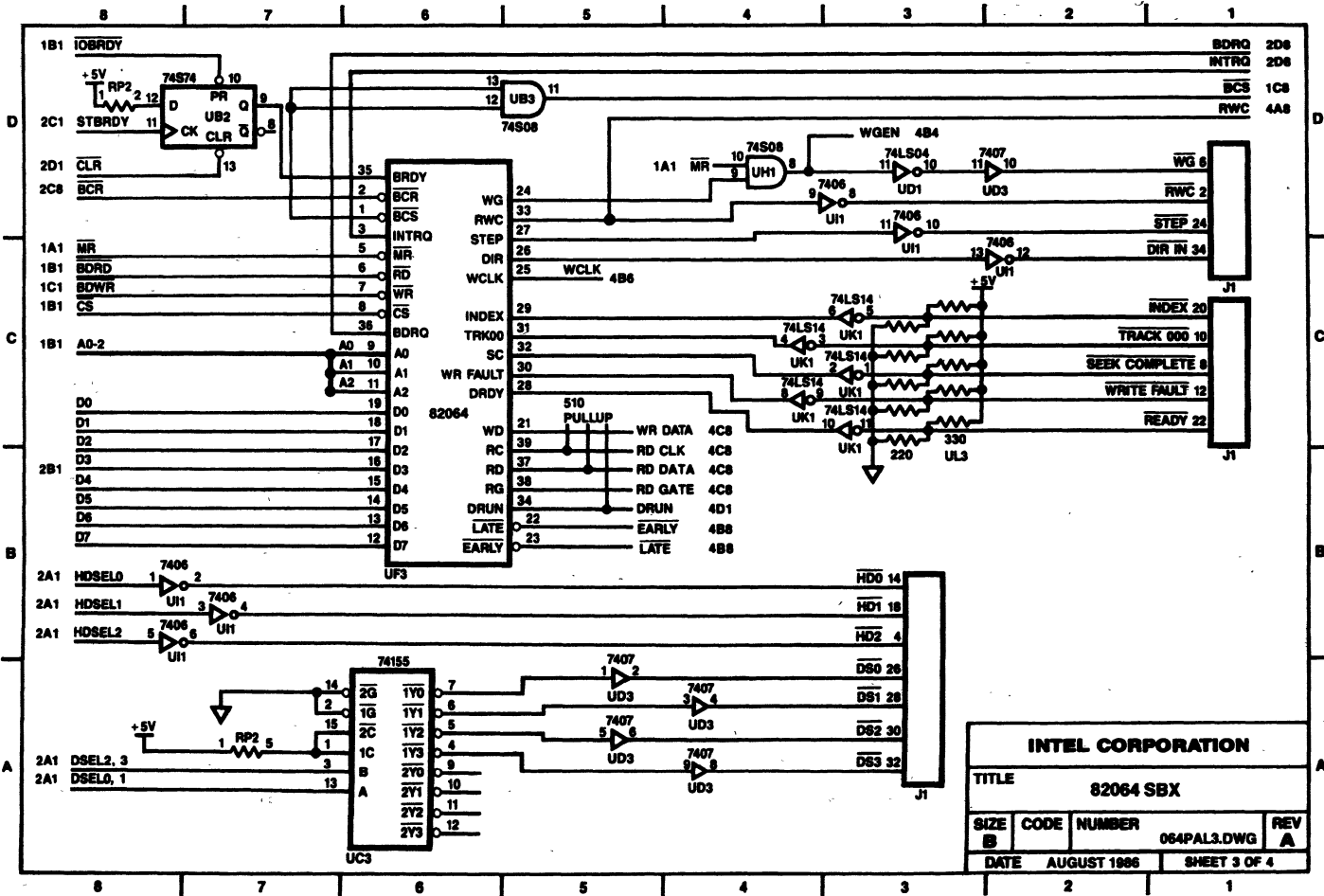
4-79



INTEL CORPORATION			
TITLE		82064 SBX	
SIZE	CODE	NUMBER	REV
B		064PAL.DWG	A
DATE		AUGUST 1986	
		SHEET 1 OF 4	

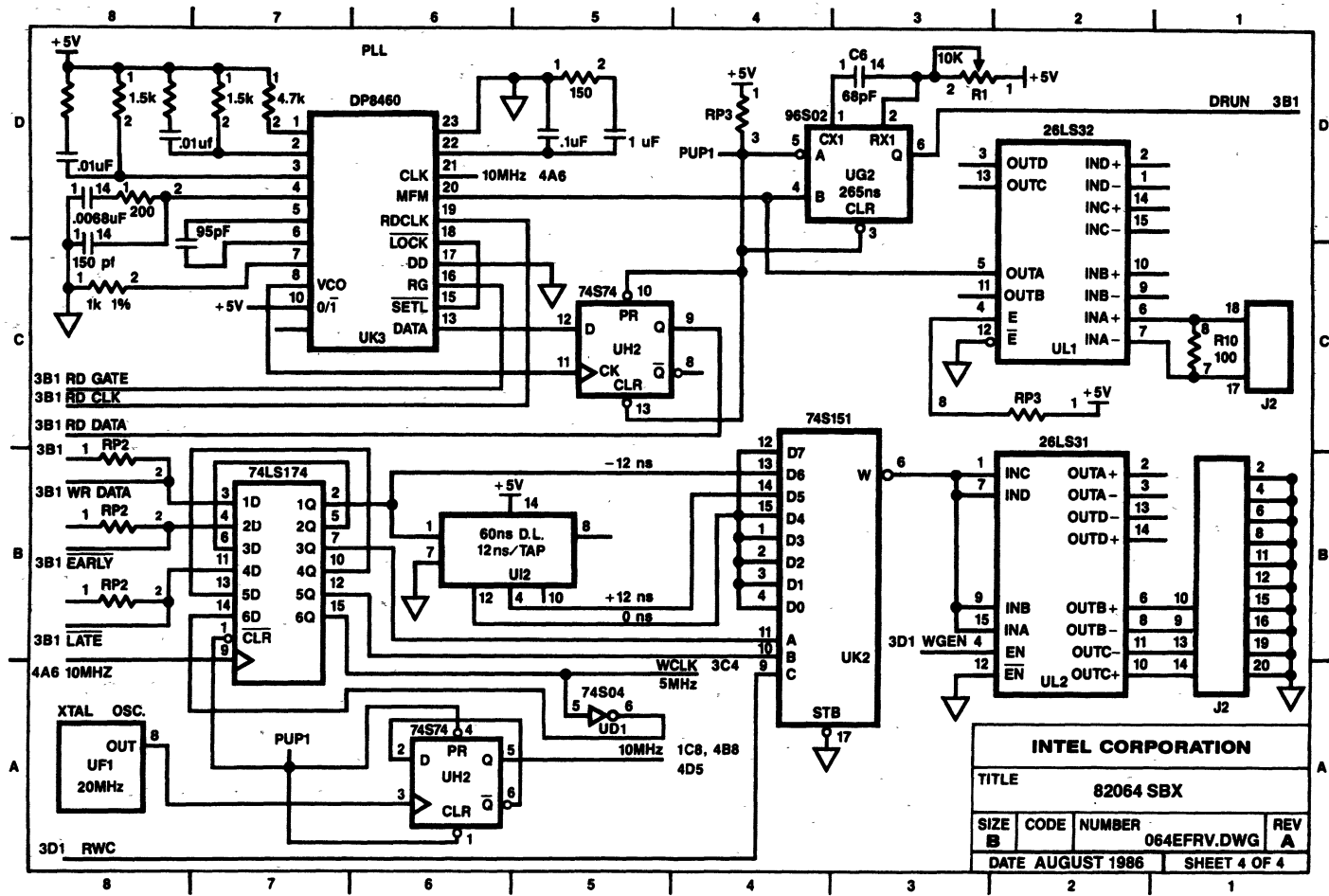


4-80



INTEL CORPORATION				
TITLE 82064 SBX				
SIZE B	CODE	NUMBER 064PAL3.DWG	REV A	
DATE	AUGUST 1986	SHEET 3 OF 4		

4-81



INTEL CORPORATION			
TITLE		82064 SBX	
SIZE	CODE	NUMBER	REV
B		064EFRV.DWG	A
DATE		AUGUST 1986	
		SHEET 4 OF 4	

Universal Peripheral Interface Slave Microcontrollers

5



UPI-452 CHMOS PROGRAMMABLE I/O PROCESSOR

83C452 - 8K × 8 Mask Programmable Internal ROM

87C452P - 8K × 8 Piggyback EPROM

80C452 - External ROM/EPROM

- **83C452/87C452P/80C452:3.5 to 16 MHz Clock Rate**
- **Software Compatible with the MCS-51 Family**
- **128-Byte Bi-Directional FIFO Slave Interface**
- **Two DMA Channels**
- **256 × 8-Bit Internal RAM**
- **34 Additional Special Function Registers**
- **40 Programmable I/O Lines**
- **Two 16-Bit Timer/Counters**
- **Boolean Processor**
- **Bit Addressable RAM**
- **8 Interrupt Sources**
- **Programmable Full Duplex Serial Channel**
- **64K Program Memory Space**
- **64K Data Memory Space**
- **68-Pin PGA**

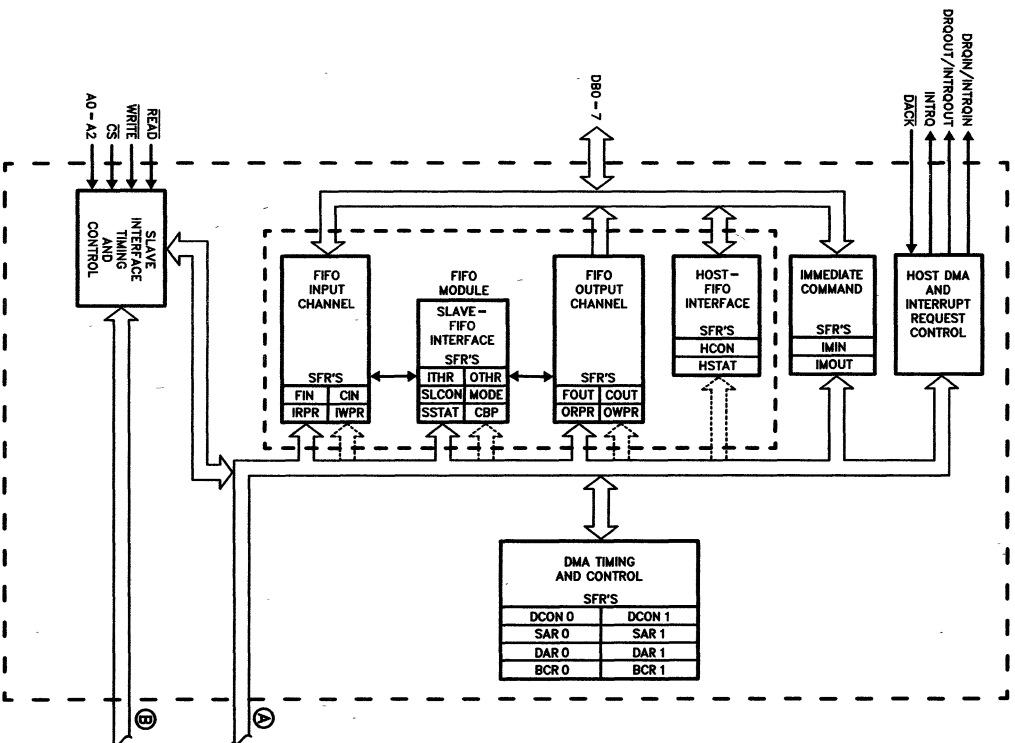
(See Packaging Spec., Order: #231369)

The Intel UPI-452 (Universal Peripheral Interface) is a 68 pin CHMOS Slave I/O Processor with a sophisticated bi-directional FIFO buffer interface on the slave bus and a two channel DMA processor on-chip. The UPI-452 is the newest member of Intel's UPI family of products. It is a general-purpose slave I/O Processor that allows the designer to grow a customized interface solution.

The UPI-452 contains a complete 80C51 with twice the on-chip data and program memory. The sophisticated slave FIFO module acts as a buffer between the UPI-452 internal CPU and the external host CPU. To both the external host and the internal CPU, the FIFO module looks like a bi-directional bottomless buffer that can both read and write data. The FIFO manages the transfer of data independent of the UPI-452 core CPU and generates an interrupt or DMA request to either CPU, host or internal, as a FIFO service request.

The FIFO consists of two channels: the Input FIFO and the Output FIFO. The division of the FIFO module array, 128 bytes, between Input channel and Output channel is programmable by the user. Each FIFO byte has an additional logical ninth bit to distinguish between a data byte and a Data Stream Command byte. Additionally, Immediate Commands allow direct, interrupt driven, bi-directional communication between the UPI-452 internal CPU and external host CPU, bypassing the FIFO.

The on-chip DMA processor allows high speed data transfers from one writeable memory space to another. As many as 64K bytes can be transferred in a single DMA operation. Three distinct memory spaces may be used in DMA operations; Internal Data Memory, External Data Memory, and the Special Function Registers (including the FIFO IN, FIFO OUT, and Serial Channel Special Functions Registers).



231428-1

Figure 1. Architectural Block Diagram

TABLE OF CONTENTS

Introduction	1
Table of Contents	4
List of Tables and Figures	5
Pin Description	7
Architectural Overview	10
Introduction	10
FIFO Buffer Interface	11
FIFO Programmable Features	11
Immediate Commands	12
DMA	12
FIFO/Slave Interface Functional Description	12
Overview	12
Input FIFO Channel	12
Output FIFO Channel	14
Immediate Commands	15
Host & Slave Interface Special Function Registers	17
Slave Interface Special Function Registers	17
External Host Interface Special Function Registers	19
FIFO Module—External Host Interface	21
Overview	21
Slave Interface Address Decoding	21
Interrupts to the Host	21
DMA Requests to the Host	23
FIFO Module—Internal CPU Interface	23
Overview	23
Internal CPU Access to FIFO via Software Instructions	23
General Purpose DMA Channels	24
Overview	24
Architecture	24
DMA Special Function Registers	25
DMA Transfer Modes	26
External Memory DMA	27
Latency	28
DMA Interrupt Vectors	28
Interrupts When DMA is Active	28
DMA Arbitration	29
Interrupts	31
Overview	31
FIFO Module Interrupts to Internal CPU	31
Interrupt Enabling and Priority	32
FIFO—External Host Interface FIFO DMA Freeze Mode	34
Overview	34
Initialization	34
Invoking FIFO DMA Freeze Mode During Normal Operation	35
FIFO Module Special Function Register Operation During FIFO DMA Freeze Mode	36
Internal CPU Read & Write of the FIFO During FIFO DMA Freeze Mode	40
Memory Organization	40
Accessing External Memory	40
Miscellaneous Special Function Register Descriptions	41

LIST OF TABLES AND FIGURES**Figures:**

1. Architectural Block Diagram	2
2. UPI-452 68-Pin PGA Pinout Diagram	6
3. UPI-452 Conceptual Block Diagram	10
4. UPI-452 Functional Block Diagram	11
5. Input FIFO Channel Functional Block Diagram	13
6. Output FIFO Channel Functional Block Diagram	14
7a. Handshake Mechanisms for Handling Immediate Command IN Flowchart	16
7b. Handshake Mechanisms for Handling Immediate Command OUT Flowchart	16
8. DMA Transfer from: External to External Memory	29
9. DMA Transfer from: External to Internal Memory	30
10. DMA Transfer from: Internal to External Memory	30
11. DMA Transfer Waveform: Internal to Internal Memory	31
12. Disabling FIFO to Host Slave Interface Timing Diagram	35

Tables:

1. Input FIFO Channel Registers	12
2. Output FIFO Channel Registers	14
3. UPI-452 Address Decoding	22
4. DMA Accessible Special Function Registers	25
5. DMA Mode Control - PCON SFR	28
6. Interrupt Priority	31
7. Interrupt Vector Addresses	31
8. Slave Bus Interface Status During FIFO DMA Freeze Mode	34
9. FIFO SFR's Characteristics During FIFO DMA Freeze Mode	37
10. Threshold SFRs Range of Values and Number of Bytes to be Transferred	38
11a. 80C51 Special Function Registers	40
11b. UPI-452 Additional Special Function Registers	41
12. Program Status Word (PSW)	42
13. PCON Special Function Register	42

UPI MICROCONTROLLER FAMILY

The UPI-452 joins the current members of the UPI microcontroller family. UPI's are derivatives of the MCS™ family of microcontrollers. Because of their on-chip system bus interface, UPI's are designed to be system bus "slaves", while their microcontroller counterparts are intended as system bus "masters".

These UPI Microcontrollers are fully supported by Intel's EPROM programmers (IUP-201) and development tools (ICE, ASM and PLM).

Packaging

The 80C452 comes in a 68-pin PGA (Pin Grid Array) package, while the 87C452P will be offered in a piggyback package. This piggyback package will consist of the standard 68-pin PGA package with a 2764A EPROM soldered on top. These two packages allow designers to use either on-chip EPROM or external memory for their initial designs. The 83C452 (ROM version) will come in the standard 68-pin PGA package. A complete description of 87C452P programming can be found at the end of this data sheet.

UPI Family (Slave Configuration)	MCS Family (Master Configuration)	Speed	RAM (Bytes)	ROM (Bytes)	EPROM (Bytes)
80C452	80C51	12 MHz	256	—	—
83C452	80C51	12 MHz	256	8K	—
87C452P	80C51	12 MHz	256	—	8K
80C452-1	80C51	16 MHz	256	—	—
83C452-1	80C51	16 MHz	256	8K	—
87C452P-1	80C51	16 MHz	256	—	8K

UPI-452 PIN DESCRIPTIONS

Symbol	Pin #	Type	Name and Function
V _{SS}	9/43	I	Circuit Ground.
V _{CC}	60	I	+ 5V power supply during normal, idle, programming and verification operation.
XTAL1	38	I	Input to the oscillator's high gain amplifier. A crystal or external source can be used.
XTAL2	39	O	Output from the high gain amplifier.
Port 0 (AD0-AD7) P0.0 .1 .2 .3 .4 .5 .6 P0.7	8 10 11 12 13 14 15 16	I/O	Port 0 is an 8-bit open drain bi-directional I/O port. It is used for data input and output during programming and verification. External pullups are required during program verification. Port 0 can sink eight LS TTL inputs. It is also the multiplexed low-order address and data local expansion bus during accesses to external memory.

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
Port 1 (A0–A7) (HLD, HLDA) P1.0 .1 .2 .3 .4 .5 .6 P1.7	 7 6 5 4 3 2 1 68	I/O	Port 1 is an 8-bit quasi-bi-directional I/O port. It is used for low-order address byte during programming and verification. Port 1 can sink four LS TTL inputs. Pins P1.5 and P1.6 are multiplexed with HLD and HLDA respectively whose functions are defined as below: Port Pin Alternate Function P1.5 HLD — Local bus hold input/output signal P1.6 HLDA — Local bus hold acknowledge output
Port 2 (A8–A15) P2.0 .1 .2 .3 .4 .5 .6 .7	 29 28 27 25 24 23 22 21	I/O	Port 2 is an 8-bit quasi-bi-directional I/O port. It also emits the high-order 8 bits of address when accessing local expansion bus external memory (or during 87C452P programming and verification). Port 2 can sink four LS TTL inputs.
Port 3 P3.0 .1 .2 .3 .4 .5 .6 P3.7	 67 66 65 64 63 62 61 59	I/O	Port 3 is an 8-bit quasi-bi-directional I/O port. It is also multiplexed with the interrupt, timer, local serial channel, RD/ and WR/ functions that are used by various options. The alternate functions can only be activated if the corresponding bit latch in the port SFR contains a 1. Otherwise, the port pin is stuck at 0. Port 3 can sink four LS TTL inputs. The alternate functions assigned to the pins of Port 3 are as follows: Port Pin Alternate Function P3.0 RxD — Serial input port P3.1 TxD — Serial output port P3.2 INT0 — Interrupt 0 Input P3.3 INT1 — Interrupt 1 Input P3.4 T0 — Input to counter 0 P3.5 T1 — Input to counter 1 P3.6 WR/ — The write control signal latches the data from Port 0 outputs into the External Data Memory on the local bus. P3.7 RD/ — The read control signal latches the data from Port 0 outputs on the local bus.

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
Port 4 P4.0 .1 .2 .3 .4 .5 .6 .7	30 31 32 33 34 35 36 37	I/O	Port 4 is an 8-bit quasi-bi-directional I/O port. Port 4 can sink/source four TTL inputs. It is also used as the control signals during EPROM programming and verification drive pins as follows: Port Pin Alternate Function P4.5 '1' during program and verify P4.6 '0' during program and verify P4.7 '0' during verify - used as output enable '1' during programming w/ ALE = 0 Note: see Programming and Verification Characteristics in AC/DC Specification section.
RST	20	I	A high level on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown resistor permits Power-on reset using only a capacitor connected to V_{CC} . This pin does not receive the power down voltage as is the case for HMOS MCS-51 family members. This function has been transferred to the V_{CC} pin.
ALE/PGM	18	I/O	Provides Address Latch Enable output used for latching the address into external memory during normal operation. Receives the program pulse input during EPROM programming. ALE can sink/source eight LS TTL inputs.
PSEN	19	O	The Program Store Enable output is a control signal that enables the external Program Memory to the bus during normal fetch operation. PSEN can sink/source eight LS TTL inputs.
\overline{EA}	17	I	When held at TTL high level, the UPI-452 executes instructions from the internal ROM/EPROM when the PC is less than 8192 (8K, 200H). When held at a TTL low level, the UPI-452 fetches all instructions from external Program Memory.
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7	58 57 56 55 54 53 52 51	I/O	Host Bus Interface is an 8-bit bi-directional bus. It is used to transfer data and commands between the UPI-452 and the host processor. This bus can sink/source eight LS TTL inputs.
\overline{CS}	44	I	This pin is the Chip Select of the UPI-452.
A0 A1 A2	40 41 42	I	These three address lines are used to interface with the host system. They define the UPI-452 operations. The interface is compatible with the Intel microprocessors and the MULTIBUS.
READ	46	I	This pin is the read strobe from the host CPU. Activating this pin causes the UPI-452 to place the contents of the Output FIFO (either a command or data) or the Host Status/Control Special Function Register on the Slave Data Bus.
WRITE	47	I	This pin is the write strobe from the host. Activating this pin will cause the value on the Slave Data Bus to be written into the register specified by A0-A2.
DRQIN/ INTRQIN	49	O	This pin requests an input transfer from the host system whenever the Input Channel requires data.
DRQOUT/ INTRQOUT	48	O	This output pin requests an output transfer whenever the Output Channel requires service. If the external host to UPI-452 DMA is enabled, and a Data Stream Command is at the Output FIFO, DRQOUT is deactivated and INTRQ is activated (see 'GENERAL PURPOSE DMA CHANNELS' section).

UPI-452 PIN DESCRIPTIONS (Continued)

Symbol	Pin #	Type	Name and Function
INTRQ	50	O	This output pin is used to interrupt the host processor when an Immediate Command Out or an error condition is encountered. It is also used to interrupt the host processor when the FIFO requests service if the DMA is disabled and INTRQIN and INTRQOUT are not used.
DACK	45	I	This pin is the DMA acknowledge for the host bus interface Input and Output Channels. When activated, a write command will cause the data on the Slave Data Bus to be written as data to the Input Channel (to the Input FIFO). A read command will cause the Output Channel to output data (from the Output FIFO) on to the Slave Data Bus. This pin should be driven high (+ 5V) in systems which do not have a DMA controller (see Address Decoding).
V _{CC} /V _{PP}	26	I	+ 5V power supply during operation. The V _{CC} pin receives the + 12V EPROM programming and verification supply voltage. It is also the standby power pin for power down mode.

ARCHITECTURAL OVERVIEW

Introduction

The UPI-452 slave microcontroller incorporates an 80C51 with double the program and data memory, a slave interface which allows it to be connected directly to the host system bus as a peripheral, a FIFO buffer module, a two channel DMA processor, and a fifth I/O port (Figure 3). The UPI-452 retains all of the 80C51 architecture, and is fully compatible with the MCS-51 instruction set.

The Special Function Register (SFR) interface concept introduced in the MCS-51 family of microcontrollers has been expanded in the UPI-452. To the 20 Special Function Registers of the MCS-51, the UPI-452 adds 34 more. These additional Special Function Registers, like those of the MCS-51, provide access to the UPI-452 functional elements including the FIFO, DMA and added interrupt capabilities. Several of the 80C51 core Special Function Registers have also been expanded to support added features of the UPI-452.

This data sheet describes the unique features of the UPI-452. Refer to the 80C51 data sheet for a description of the UPI-452's core CPU functional blocks including;

- Timers/Counters
- I/O Ports
- Interrupt timing and control (other than FIFO and DMA interrupts)
- Serial Channel
- Local Expansion Bus
- Program/Data Memory structure
- Power-Saving Modes of Operation *
- CHMOS Features
- Instruction Set

* except 87C452P piggyback package

Figure 3 contains a conceptual block diagram of the UPI-452. Figure 4 provides a functional block diagram.

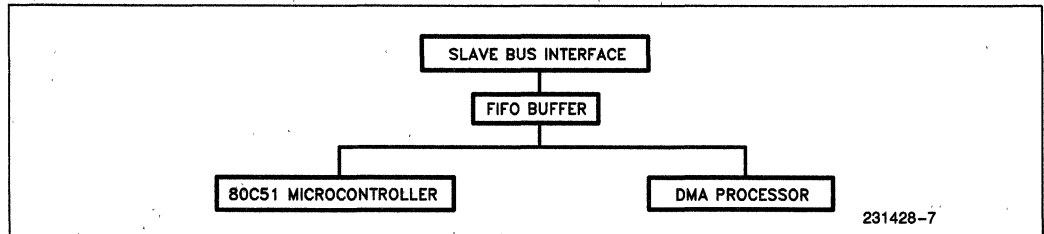


Figure 3. UPI-452 Conceptual Block Diagram

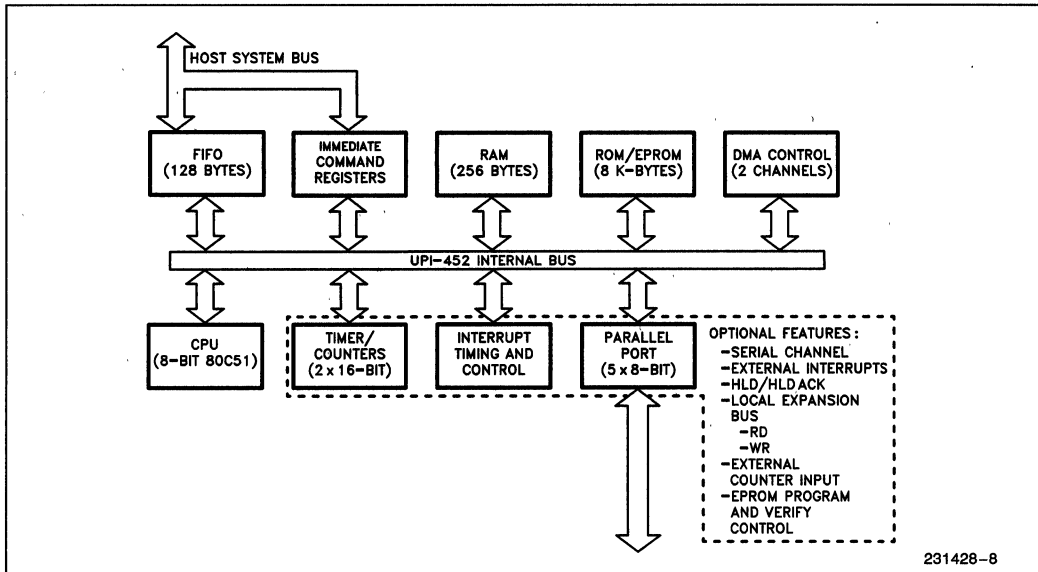


Figure 4. UPI-452 Functional Block Diagram

FIFO Buffer Interface

A unique feature of the UPI-452 is the incorporation of a 128 byte FIFO array at the host-slave interface. The FIFO allows asynchronous bi-directional transfers between the host CPU and the internal CPU. The division of the 128 bytes between Input and Output channels is user programmable allowing maximum flexibility. If the entire 128 byte FIFO is allocated to the Input channel, a high performance Host can transfer up to 128 bytes at one time, then dedicate its resources to other functions while the internal CPU processes the data in the FIFO. Various handshake signals allow the external Host to operate independently and without frequent monitoring of the UPI-452 internal CPU. The FIFO Buffer insures that the slave processor receives data in the same order that it was sent by the host without the need to keep track of addresses. Three slave bus interface handshake methods are supported by the UPI-452: DMA, Interrupt and Polled.

The FIFO is nine bits wide. The ninth bit acts as a command/data flag. Commands written to the FIFO by either the host or internal CPU are called Data Stream Commands or DSCs. DSCs are written to the input FIFO by the Host via a unique external address. DSCs are written to the output FIFO by the internal CPU via the COMMAND OUT Special Function Register (SFR). When encountered by the host or internal CPU a Data Stream Command can be used as an address vector to user defined service routines. DSCs provide synchronization of data and commands between the Host and internal CPU.

FIFO PROGRAMMABLE FEATURES

Size of Input/Output Channels

The 128 bytes of FIFO space can be allocated between the Input and Output channels via the Channel Boundary Pointer (CBP) SFR. This register contains the number of address locations assigned to the Input channel. The remaining address locations are automatically assigned to the Output FIFO. The CBP SFR can only be programmed by the internal CPU during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description). The CBP is initialized to 40H (64 bytes) upon reset.

The number in the Channel Boundary Pointer SFR is actually the first address location of the Output FIFO. Writing to the CBP SFR reassigns the Input and Output FIFO address space. Whenever the CBP is written, the Input FIFO pointers are reset to zero and the Output FIFO pointers are set to the value in the CBP SFR.

All of the FIFO space may be assigned to one channel. In such a situation the other channel's data path consists of a single SFR (FIFO IN/COMMAND IN or FIFO OUT/COMMAND OUT SFR) location.

FIFO Read/Write Pointers

These normally operate in auto-increment (and auto-rollover) mode, but can be reassigned by the internal CPU during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description).

Threshold Register

The Input FIFO Threshold SFR contains the number of empty bytes that must be available in the Input FIFO to generate a Host interrupt. The Output FIFO Threshold SFR contains the number of bytes, data and/or DSC(s), that must be in the FIFO before an interrupt is generated. The Threshold feature prevents the Host from being interrupted each time the FIFO needs to load or unload one byte of data. The thresholds, therefore, allow the FIFO's operation to be adjusted to the speed of the Host, optimizing the overall interface performance.

Immediate Commands

The UPI-452 provides, in addition to data and DSCs, a third direct means of communication between the external Host and internal CPU called Immediate Commands. As the name implies, an Immediate Command is available to the receiving CPU immediately, via an interrupt, without being entered into the FIFO as are Data Stream Commands. Like Data Stream Commands, Immediate Commands are written either via a unique external address by the host CPU, or via dedicated SFR by the internal CPU.

The DSC and/or Immediate Command interface may be defined as either Interrupt or Polled under user program control via the Interrupt Enable (IE), Slave Control Register (SLCON), and Interrupt Enable Priority (IEP) Special Function Registers, for the internal CPU and via the Host Control SFR for the external Host CPU.

DMA

The UPI-452 contains a two channel internal DMA controller which allows transfer of data between any

of the three writeable memory spaces: Internal Data Memory, External Load Expansion Bus Data Memory and the Special Function Register array. The Special Function Register array appears as a set of unique dedicated memory addresses which may be used as either the source or destination address of a DMA transfer. Each DMA channel is independently programmable via dedicated Special Function Registers for mode, source and destination addresses, and byte count to be transferred. Each DMA channel has four programmable modes:

- Alternate Cycle Mode
- Burst Mode
- FIFO or Serial Channel Demand Mode
- External Demand Mode

A complete description of each mode and DMA operation may be found in the section titled "General Purpose DMA Channels".

FIFO/SLAVE INTERFACE FUNCTIONAL DESCRIPTION

Overview

The FIFO is a 128 Byte RAM array with recirculating pointers to manage the read and write accesses. The FIFO consists of an Input and an Output channel. Access cycles to the FIFO by the internal CPU and external Host are interleaved and appear to be occurring concurrently to both the internal CPU and external Host. Interleaving access cycles ensures efficient use of this shared resource. The internal CPU accesses the FIFO in the same way it would access any of the Special Function Registers e.g., direct and register indirect addressing as well as arithmetic and logical instructions.

Input FIFO Channel

The Input FIFO Channel provides for data transfer from the external Host to the internal CPU (Figure 5). The registers associated with the Input Channel during normal operation are listed in Table 1*.

Table 1. Input FIFO Channel Registers*

	Register Name	Description
1)	Input Buffer Latch	Host CPU Write only
2)	FIFO IN SFR	Internal CPU Read only
3)	COMMAND IN SFR	Internal CPU Read only
4)	Input FIFO Read Pointer SFR	Internal CPU Read only
5)	Input FIFO Write Pointer SFR	Internal CPU Read only
6)	Input FIFO Threshold SFR	Internal CPU Read only

*See "FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE" section for FIFO DMA Freeze Mode SFR characteristics description.

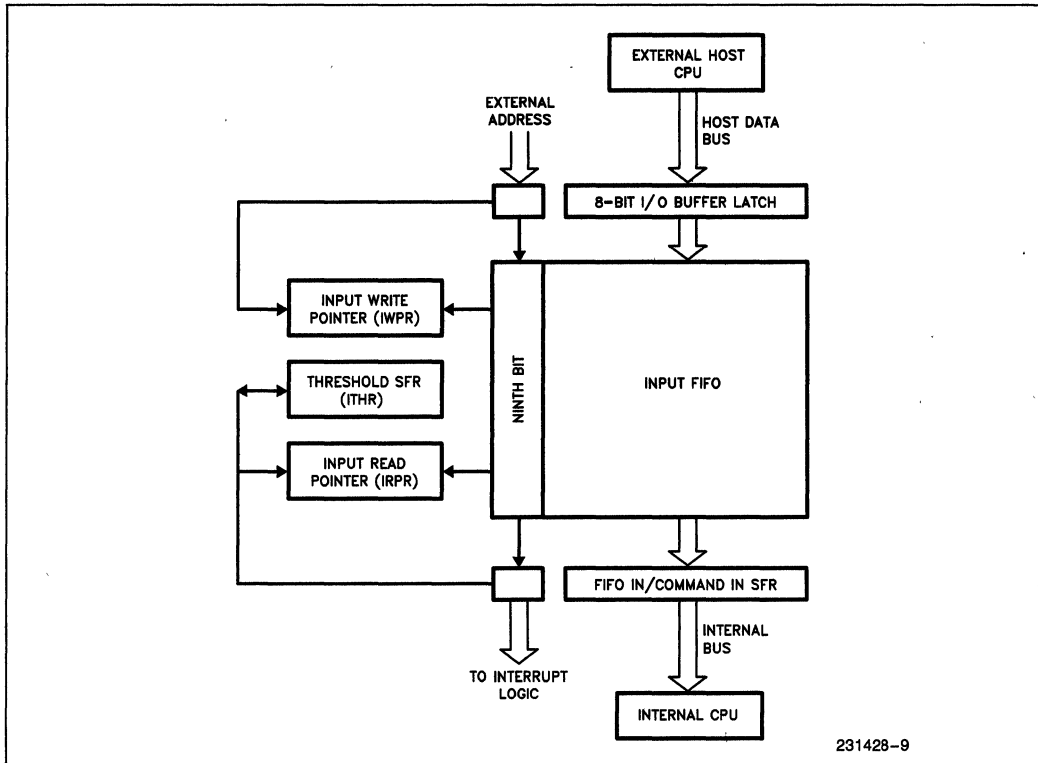


Figure 5. Input FIFO Channel Functional Block Diagram

The host CPU writes data and Data Stream Commands into the Input Buffer Latch on the rising edge of the external WR signal. External addressing determines whether the byte is a data byte or Data Stream Command and the FIFO logic sets the ninth bit of the FIFO accordingly as the byte is moved from the Input Buffer Latch into the FIFO. A "1" in the ninth bit indicates that the incoming byte is a Data Stream Command. The internal CPU reads data bytes via the FIFO IN SFR, and Data Stream Commands via the COMMAND IN SFR.

A Data Stream Command will generate an interrupt to the internal CPU prior to being read and after completion of the previous operation. The DSC can then be read via the COMMAND IN SFR. Data can only be read via the FIFO IN SFR and Data Stream Commands via the COMMAND IN SFR. Attempting to read Data Stream Commands as data by addressing the FIFO IN SFR will result in "OFFH" being read, and the Input FIFO Read Pointer will remain intact. (This prevents accidental misreading of Data Stream Commands.) Attempting to read data as Data Stream Commands will have the same consequence.

The Input FIFO Channel addressing is controlled by the Input FIFO Read and Write Pointer SFRs. These SFRs are read only registers during normal operation. However, during FIFO DMA Freeze Mode (See FIFO-External Host Interface FIFO DMA Freeze Mode description), the internal CPU has write access to them. Any write to these registers in normal mode will have no effect. The Input Write Pointer SFR contains the address location to which data/commands are written from the Slave Bus Input/Slave Bus Command registers. The write pointer is automatically incremented after each write and is reset to zero if equal to the CBP, as the Input FIFO operates as a circular buffer.

If a write is performed on an empty FIFO, the first byte is also written into the FIFO IN or COMMAND IN SFR. If the Host continues writing while the Input FIFO is full, an external interrupt, if enabled, is sent to the host to signal the overrun condition. The writes are ignored by the FIFO control logic. Similarly, an internal CPU read of an empty FIFO will cause an underrun error interrupt to be generated to the internal CPU and a value of "OFFH" will be read by the internal CPU.

The Read Pointer SFR holds the address of the next byte to be read from the Input FIFO. An Input FIFO read operation post-increments the Input Read Pointer SFR and loads a new data byte into the FIFO IN SFR or a Data Stream Command into the COMMAND IN SFR at the end of the read cycle.

An Input FIFO Request for Service (via DMA, Interrupt or a flag) is generated to the Host whenever more data can be written into the Input FIFO. For efficient utilization of the Host, a "threshold" value can be programmed into the Input FIFO Threshold SFR. The range of values of the Input FIFO Threshold SFR can be from 0 to (CBP-2). The Request for Service Interrupt is generated only after the Input FIFO has room to accommodate a threshold number of bytes or more. The threshold is equal to the total

number of bytes in the Input FIFO minus the number of bytes programmed in the Input FIFO Threshold SFR. With this feature the Host is assured that it can write at least a threshold number of bytes to the Input FIFO channel without worrying about an overrun condition. Once the Request for Service is generated it remains active until the Input FIFO becomes full.

Output FIFO Channel

The Output FIFO Channel provides data transfer from the UPI-452 internal CPU to the external Host (Figure 6).

The registers associated with the Output Channel during normal operation are listed in Table 2*.

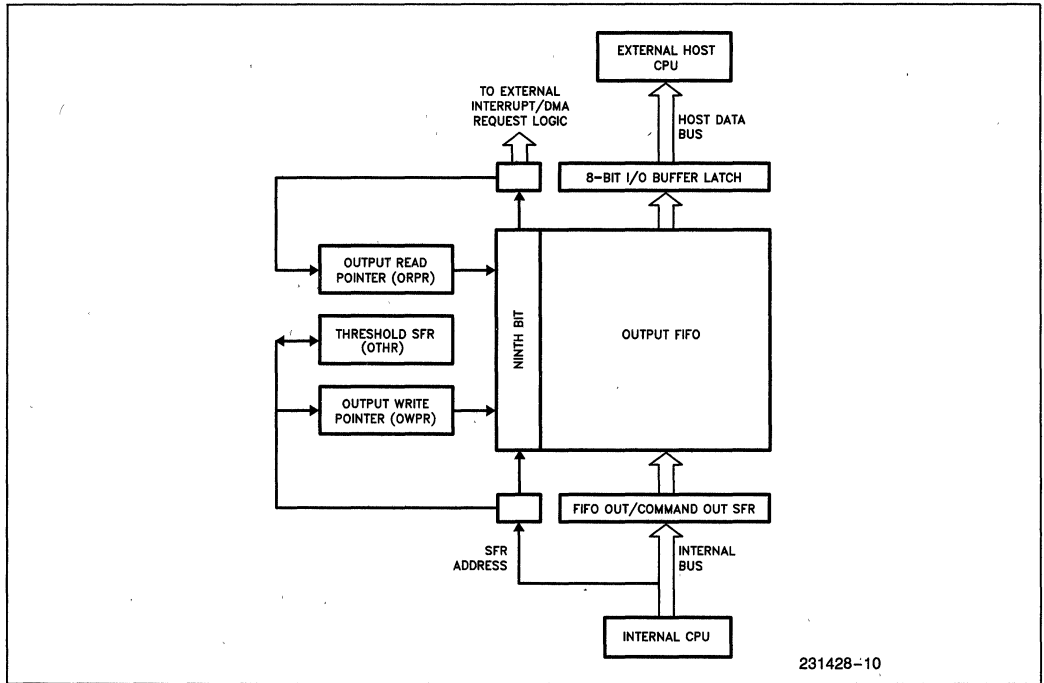


Figure 6. Output FIFO Channel Functional Block Diagram

Table 2. Output FIFO Channel Registers

	Register Name	Description
1)	Output Buffer Latch	Host CPU Read only
2)	FIFO OUT SFR	Internal CPU Read and Write
3)	COMMAND OUT SFR	Internal CPU Read and Write
4)	Output FIFO Read Pointer SFR	Internal CPU Read only
5)	Output FIFO Write Pointer SFR	Internal CPU Read only
6)	Output FIFO Threshold SFR	Internal CPU Read only

*See "FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE" section for FIFO DMA Freeze Mode register characteristics description.

The UPI-452 internal CPU transfers data to the Output FIFO via the FIFO OUT SFR and commands via the COMMAND OUT SFR. If the byte is written to the COMMAND OUT SFR, the ninth bit is automatically set (= 1) to indicate a Data Stream Command. If the byte is written to the FIFO OUT SFR the ninth bit is cleared (= 0). Thus the FIFO OUT and COMMAND OUT SFRs are the same but the address determines whether the byte entered in the FIFO is a DSC or data byte.

The Output FIFO preloads a byte into the Output Buffer Latch. When the Host issues a RD/ signal, the data is immediately read from the Output Buffer Latch. The next data byte is then loaded into the Output Buffer Latch, a flag is set and an interrupt, if enabled, is generated if the byte is a DSC (ninth bit is set). The operation is carefully timed such that an interrupt can be generated in time for it to be recognized by the Host before its next read instruction. Internal CPU write and external Host read operations are interleaved at the FIFO so that they appear to be occurring concurrently.

The Output FIFO read and write pointer operation is the same as for the Input Channel. Writing to the FIFO OUT or COMMAND OUT SFRs will increment the Output Write Pointer SFR but reading from it will leave the write pointer unchanged. A rollover of the Output FIFO Write Pointer causes the pointer to be reset to the value in the Channel Boundary Pointer (CBP) SFR.

If the external host attempts to read a Data Stream Command as a data byte it will result in invalid data being read. The DSC is not lost because the invalid read does not increment the pointer. Similarly attempting to read a data byte as a Data Stream Command has the same result.

A Request for Service is generated to the external Host under the following two conditions:

- 1.) Whenever the internal CPU has written a threshold number of bytes or more into the Output FIFO (threshold = (OTHR) + 1). The threshold number should be chosen such that the bus latency time for the external Host does not result in a FIFO overrun error condition on the internal CPU side. The threshold limit should be large enough to make a bus request by the UPI-452 to the external host CPU worthwhile. Once a request for service is generated, the request remains active until the Output FIFO becomes empty. The range of values of the FIFO Output Threshold (OTHR) SFR is from 1 to the Output FIFO Size. The threshold number can be programmed via the OTHR SFR.

- 2.) The second type of Request for Service is called "Flush Mode" and occurs when the internal CPU writes a Data Stream Command into the Output FIFO. Its purpose is to ensure that a data block entered into the Output FIFO, which is less than the programmed threshold, will generate a Request for Service interrupt, if enabled, and be read, or "Flushed" from the Output FIFO, by the external host CPU regardless of the status of the OTHR SFR.

Immediate Commands

Immediate Commands provide direct communication between the external Host and UPI-452. Unlike Data Stream Commands which are entered into the FIFO, the Immediate Command is available to the receiving CPU directly, bypassing the FIFO. The Immediate Command can serve as a program vector pointing into a jump table in the recipient's software. Immediate Command Interrupts are generated, if enabled, and a bit in the appropriate Status Register is set when an Immediate Command is input or output. A similar bit is provided to acknowledge when an Immediate Command has been read and whether the register is available to receive another command. The bits are reset when the Immediate Commands are read. Two Special Function Registers are dedicated to the Immediate Command interface. External addressing determines whether the Host is accessing the Input FIFO or the Immediate Command IN (IMIN) SFR. The internal CPU writes Immediate Commands to the Immediate Command OUT (IMOUT) SFR.

Both processors have the ability to enable or disable Immediate Command Interrupts. By disabling the interrupt, the recipient of the Immediate Command can poll the status SFR and read the Immediate Command at its convenience. Immediate Commands should only be written when the appropriate Immediate Command SFR is empty (as indicated in the appropriate status SFR: HSTAT/SSTAT). Similarly, the Immediate Command SFR should only be read when there is data in the Register.

The flowcharts in Figure 7a and 7b illustrate the proper handshake mechanisms between the external Host and internal CPU when handling Immediate Commands.

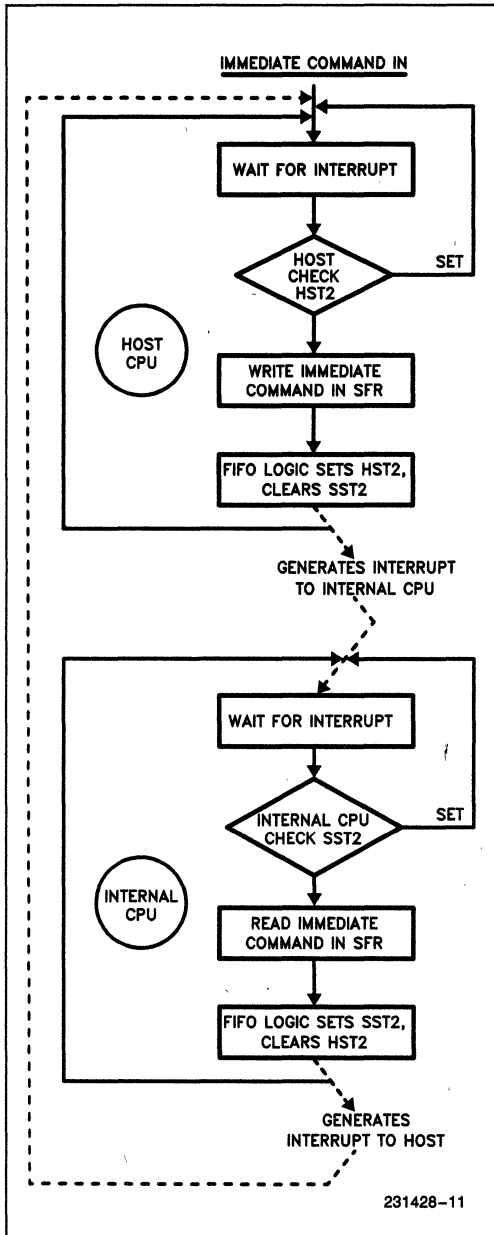


Figure 7a. Handshake Mechanisms for Handling Immediate Command IN Flowchart

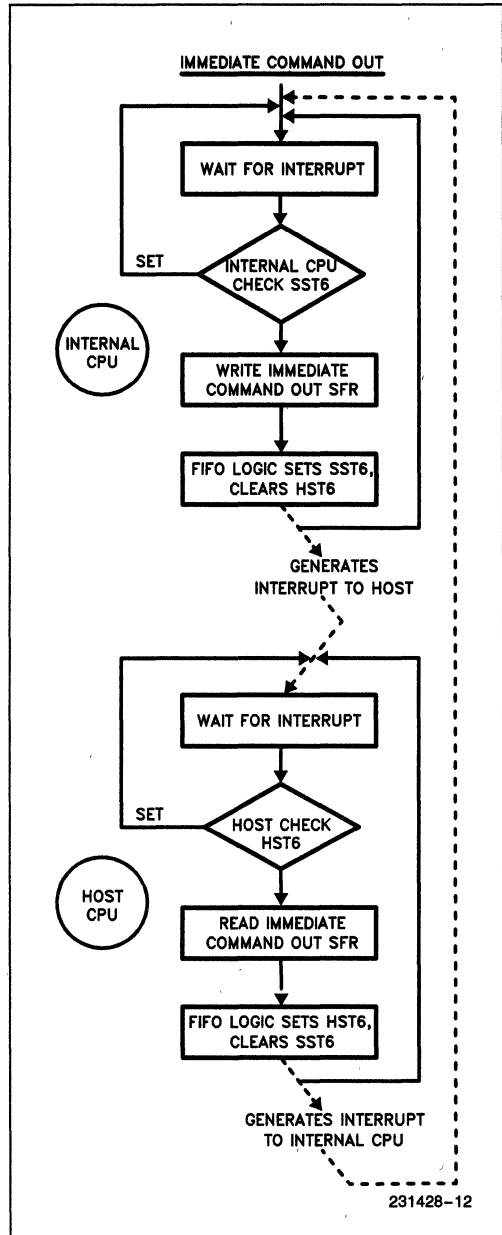


Figure 7b. Handshake Mechanisms for Handling Immediate Command OUT Flowchart

HOST & SLAVE INTERFACE SPECIAL FUNCTION REGISTERS

Slave Interface Special Function Registers

The Internal CPU interfaces with the FIFO slave module via the following registers:

- 1) Mode Special Function Register (MODE)
- 2) Slave Control Special Function Register (SLCON)
- 3) Slave Status Special Function Register (SSTAT)

Each register resides in the SFR Array and is accessible via all direct addressing modes except bit. Only the Slave Control Register (SLCON) is bit addressable.

1) MODE Special Function Register (MODE)

The MODE SFR provides the primary control of the external host-FIFO interface. It is included in the SFR Array so that the internal CPU can configure the external host-FIFO interface should the user decide that the UPI-452 slave initialize itself independent of the external host CPU.

The MODE SFR can be directly modified by the internal CPU through direct address instructions. It can also be indirectly modified by the external host CPU by setting up a MODE SFR service routine in the UPI-452 program memory and having the host issue a Command, either Immediate or DSC, to vector to that routine.

Symbolic Address									Physical Address
MODE	—	MD6	MD5	MD4	—	—	—	—	0F9H
	(MSB)				(LSB)				
	Status On Reset:								
	1*	0	0	0	1*	1*	1*	1*	

MD7 (reserved)**

MD6 Request for Service to external CPU via;

1 = DMA (DRQIN/DRQOUT) request to external host when the Input or Output FIFO channel requests service

0 = Interrupt (INTRQIN/INTRQOUT or INTRQ) to external host when the Input or Output FIFO channel requests service or a DSC is encountered in the I/O Buffer Latch

MD5 Configure DRQIN/INTRQIN and DRQOUT/INTRQOUT to be either;

1 = Enable (Actively driven)

0 = Disable (Tri-state)

MD4 Configure INTRQ to be either;

1 = Enable (Actively driven)

0 = Disable (Tri-state)

MD3 (reserved) **

MD2 (reserved) **

MD1 (reserved) **

MD0 (reserved) **

2) Slave Control SFR (SLCON)

The Slave Control SFR is used to configure the FIFO-internal CPU interface. All interrupts are to the internal CPU.

Symbolic Address **Physical Address**

SLCON	IFI	OFI	ICII	ICOI	FRZ	—	IFRS	OFRS	0E8H
	(MSB)							(LSB)	

Status On Reset:

0	0	0	0	0	1*	0	0
---	---	---	---	---	----	---	---

IFI Enable Input FIFO Interrupt (due to Underrun Error Condition, Data Stream Command or Request Service)

1 = Enable

0 = Disable

OFI Enable Output FIFO Interrupt (due to Overrun Error Condition or Request Service)

1 = Enable

0 = Disable

Note: If the DMA is configured to service a FIFO demand, then the Request for Service Interrupt is not generated.

ICII Generate Interrupt when a command is written to the Immediate Command in Register

1 = Enable

0 = Disable

ICOI Generate Interrupt when Immediate Command Out Register is Available

1 = Enable

0 = Disable

FRZ Enable FIFO DMA Freeze Mode

1 = Normal operation

0 = FIFO DMA Freeze Mode

SC2 (reserved) **

IFRS Input FIFO Channel Request for Service

1 = Request when Input FIFO not empty

0 = Request when Input FIFO full

OFRS Output FIFO Channel Request for Service

1 = Request when Output FIFO not full

0 = Channel Request when Output FIFO empty

NOTES:

*A '1' will be read from all SFR reserved locations except HCON, SFR, HC0 and HC2.

**'reserved'—these locations are reserved for future use by Intel Corporation.

3) Slave Status SFR (SSTAT)

The bits in the Slave Status SFR reflect the status of the FIFO-internal CPU interface. It can be read during an internal interrupt service routine to determine the nature of the interrupt or read during a polling sequence to determine a course of action.

Symbolic Address **Physical Address**

SSTAT	SST7	SST6	SST5	SST4	SST3	SST2	SST1	SST0	0E9H
	← Output FIFO Status →				← Input FIFO Status →				

Status On Reset:

1	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---

(MSB)

(LSB)

- SST7 Output FIFO Overrun Error Condition
 - 1 = No Error
 - 0 = Error (latched until Slave Status SFR is read)
- SST6 Immediate Command Out Register Status
 - 1 = Full (i.e. Host CPU has not read previous Immediate Command Out sent by internal CPU)
 - 0 = Available
- SST5 FIFO DMA Freeze Mode Status
 - 1 = Normal Operation
 - 0 = FIFO DMA Freeze Mode in Progress
- SST4 Output FIFO Request for Service Flag
 - 1 = Output FIFO does not request service
 - 0 = Output FIFO requests service
- SST3 Input FIFO Underrun Error Condition Flag
 - 1 = No Underrun Error
 - 0 = Underrun Error (latched until Slave Status SFR is read)
- SST2 Immediate Command In SFR Status
 - 1 = Empty
 - 0 = Immediate Command received from host CPU
- SST1 Data Stream Command/Data at Input FIFO Flag
 - 1 = Data (not DSC)
 - 0 = DSC (at COMMAND IN SFR)
- SST0 Input FIFO Request For Service Flag
 - 1 = Input FIFO Does Not Request Service
 - 0 = Input FIFO Request for Service

EXTERNAL HOST INTERFACE SPECIAL FUNCTION REGISTERS

The external host CPU has direct access to the following SFRs:

- 1) Host Control Special Function Register
- 2) Host Status Special Function Register

It can also access other SFRs by commanding the internal CPU to change them accordingly via Data Stream Commands or Immediate Commands. The protocol for implementing this is entirely determined by the user.

1) Host Control SFR (HCON)

By writing to the Host Control SFR, the host can enable or disable FIFO interrupts and DMA requests and can reset the UPI-452.

Symbolic Address									Physical Address
HCON	HC7	HC6	HC5	HC4	HC3	—	HC1	—	0E7H
	(MSB)				(LSB)				
	Status On Reset:								
	0	0	0	0	0	0*	0	0*	

- HC7 Enable Output FIFO Interrupt due to Underrun Error Condition, Data Stream Command or Service Request
 1 = Enable
 0 = Disable
- HC6 Enable Input FIFO Interrupt due to Overrun Error Condition, or Service Request
 1 = Enable
 0 = Disable
- HC5 Enable the generation of the Interrupt due to Immediate Command Out being present
 1 = Enable
 0 = Disable
- HC4 Enable the Interrupt due to the Immediate Command in Register being Available for a new Immediate Command byte
 1 = Enable
 0 = Disable
- HC3 Reset UPI-452
 1 = Software RESET
 0 = Normal Operation
- HC2 (reserved) **
- HC1 Select between INTRQ and INTRQIN/INTRQOUT as Request for Service interrupt signal when DMA is disabled
 1 = INTRQ
 0 = INTRQIN or INTRQOUT
- HC0 (reserved) **

NOTES:

*A '1' will be read from all SFR reserved locations except HCON, SFR, HC0 and HC2.
 ***'reserved'—these locations are reserved for future use by Intel Corporation.

2) Host Status SFR (HSTAT)

The Host Status SFR provides information on the FIFO-Host Interface and can be used to determine the source of an external interrupt during polling. Like the Slave Status SFR, the Host Status SFR reflects the current status of the FIFO-external host interface.

Symbolic Address

Physical Address

HSTAT	HST7	HST6	HST5	HST4	HST3	HST2	HST1	HST0	0E6H
	← Output FIFO Status →				← Input FIFO Status →				
	Status On Reset:								
	1	1	1	1	1	0/1*	1	1	
	(MSB)				(LSB)				

- HST7** Output FIFO Underrun Error Condition
 1 = No Underrun Error
 0 = Underrun Error (latched until Host Status Register is read)
- HST6** Immediate Command Out SFR Status
 1 = Empty
 0 = Immediate Command Present
- HST5** Data Stream Command/Data at Output FIFO Status
 1 = Data (not DSC)
 0 = DSC (present at Output FIFO COMMAND OUT SFR)
 (Note: Only if HST4=0, if HST4=1 then undetermined)
- HST4** Output FIFO Request for Service Status
 1 = No Request for Service
 0 = Output FIFO Request for Service due to:
 a. Output FIFO containing the threshold number of bytes or more
 b. Internal CPU sending a block of data terminated by a DSC (DSC alone clears upon being read)
- HST3** Input FIFO Overrun Error Condition
 1 = No Overrun Error
 0 = Overrun Error (latched until Host Status Register is read)
- HST2** Immediate Command In SFR Status
 1 = Full (i.e. Internal CPU has not read previous Immediate Command sent by Host)
 0 = Empty
 Reset value;
 '1' — if read by the external Host
 '0' — if read by internal CPU (reads shadow latch - see FIFO DMA Freeze Mode description)
- HST1** FIFO DMA Freeze Mode Status
 1 = Freeze Mode in progress.
 (In Freeze Mode, the bits of the Host Status SFR are forced to a '1' initially to prevent the external Host from attempting to access the FIFO. The definition of the Host Status SFR bits during FIFO DMA Freeze Mode can be found in FIFO DMA Freeze Mode description)
 0 = Normal Operation
- HST0** Input FIFO Request Service Status
 1 = Input FIFO does not request service
 0 = Input FIFO request service due to the input FIFO containing enough space for the host to write the threshold number of bytes or more

FIFO MODULE - EXTERNAL HOST INTERFACE

Overview

The FIFO-external Host interface supports high speed asynchronous bi-directional 8-bit data transfers. The host interface is fully compatible with Intel microprocessor local busses and with MULTIBUS. The FIFO has two specialized DMA request pins for Input and Output FIFO channel DMA requests. These are multiplexed to provide a dedicated Request for Service interrupt (DRQIN/INTRQIN, DRQOUT/INTRQOUT).

The external Host can program, under user defined protocol, thresholds into the FIFO Input and Output Threshold SFRs which determine when the FIFO Request for Service interrupt is generated to the Host CPU. The FIFO module external Host interface is configured by the internal CPU via the MODE SFR. "The external Host can enable and disable Host interface interrupts via the Host Control SFR." Data Stream Commands in the Input FIFO channel allow the Host to influence the processing of data blocks and are sent with the data flow to maintain synchronization. Data Stream Commands in the Output FIFO Channel allow the internal CPU to perform the same function, and also to set the Output FIFO Request Service status logic to the host CPU regardless of the programmed value in the Threshold SFR.

Slave Interface Address Decoding

The UPI-452 determines the desired Host function through address decoding. The lower three bits of the address as well as the READ, WRITE, Chip Select (CS) and DMA Acknowledge (DACK) are used for decoding. Table 3 shows the pin states and the Read or Write operations associated with each configuration.

Interrupts to the Host

The UPI-452 interrupts the external Host via the INTRQ pin. In addition, the DRQIN and DRQOUT pins can be multiplexed as interrupt request lines, INTRQIN and INTRQOUT respectively, when DMA is disabled. This provides two special FIFO "Request for Service" interrupts.

There are eight FIFO-related interrupt sources; two from The Input FIFO; three from The Output FIFO; one from the Immediate Command Out SFR; one from the Immediate Command IN SFR; and one due to FIFO DMA Freeze Mode.

INPUT FIFO: The Input FIFO interrupt is generated whenever:

- a. The Input FIFO contains space for a threshold number of bytes.

Table 3. UPI-452 Address Decoding

DACK	CS	A2	A1	A0	Read	Write
1	1	X	X	X	No Operation	No Operation
1	0	0	0	0	Data or DMA from Output FIFO Channel	Data or DMA to Input FIFO Channel
1	0	0	0	1	Data Stream Command from Output FIFO Channel	Data Stream Command to Input FIFO Channel
1	0	0	1	0	Host Status SFR Read	Reserved
1	0	0	1	1	Host Control SFR Read	Host Control SFR Write
1	0	1	0	0	Immediate Command SFR Read	Immediate Command to SFR Write
1	0	1	1	X	Reserved	Reserved
0	X	X	X	X	DMA Data from Output FIFO Channel	DMA Data to Input FIFO Channel
1	0	1	0	1	Reserved	Reserved

NOTES:

1. Attempting to read a DSC as a data byte will result in invalid data being read. The read pointers are not incremented so that the DSC is not lost. Attempting to read a data byte as a DSC has the same result.
2. If DACK is active the UPI-452 will attempt a DMA operation when RD or WR becomes active regardless of the DMA enable bit (MD6) in the MODE SFR. Care should be taken when using DACK. For proper operation, DACK must be driven high (+5V) when not using DMA.

b. When an Input FIFO overrun error condition exists. The appropriate bits in the Host Status SFR are set and the interrupt is generated only if enabled.

OUTPUT FIFO: The Output FIFO Request for Service Interrupt operates in a similar manner as the Input FIFO interrupt:

- a. When the FIFO contains the threshold number of bytes or more.
- b. Output FIFO error condition interrupts are generated when the Output FIFO is underrun.
- c. Data Stream Command present in the Output Buffer Latch.

A Data Stream Command interrupt is used to halt normal processing, using the command as a vector to a service routine. When DMA is disabled, the user may program (through HC1) INTRQ to include FIFO Request for Service Interrupts or use INTRQIN and INTRQOUT as Request for Service Interrupts.

IMMEDIATE COMMAND INTERRUPTS:

a. An Immediate Command Out Interrupt is generated, if enabled, to the Host and the corresponding Host Status SFR bit (HSTAT HST6) is set, when the internal CPU writes to the Immediate Command OUT (IMOUT) SFR. When the Host reads the Immediate Command OUT (IMOUT) SFR the corresponding bit in the Host Status (HSTAT) SFR is cleared. This causes the Slave Status Immediate Command OUT Status bit (SSTAT SST6) to be set indicating that the Immediate Command OUT (IMOUT) SFR is empty. If enabled, a FIFO-Slave Interface will also be generated to the internal CPU. (See Figure 7b, Immediate Command OUT Flowchart.)

b. An Immediate Command IN interrupt is generated, if enabled, to the Host when the internal CPU has read a byte from the Immediate Command IN (IMIN) SFR. The read operation clears the Host Status SFR Immediate Command IN Status bit (HSTAT HST2) indicating that the Immediate Command IN SFR is empty. The corresponding Slave Status (SSTAT) SFR bit is also set to indicate an empty status. Setting the Slave Status SFR bit generates a FIFO-Slave Interface interrupt, if enabled, to the internal CPU. (See Figure 7a, Immediate Command IN Flowchart.)

NOTE:

Immediate Command IN and OUT interrupts are actually specific Request For Service interrupts to the Host.

FIFO DMA FREEZE MODE: When the internal CPU invokes FIFO DMA Freeze Mode, for example at reset or to reconfigure the FIFO interface, INTRQ is activated. The INTRQ can only be deactivated by the external Host reading the Host Status SFR (HST1 remains active until FIFO DMA Freeze Mode is disabled by the internal CPU).

Once an interrupt is generated, INTRQ will remain high until no interrupt generating condition exists. For a FIFO underrun/overrun error interrupt, the interrupt condition is deactivated by the external Host reading the Host Status SFR. An interrupt is serviced by reading the Host Status SFR to determine the source of the interrupt and vectoring the appropriate service routine.

DMA Requests to the Host

The UPI-452 generates two DMA requests, DRQIN and DRQOUT, to facilitate data transfer between the Host and the Input and Output FIFO channels. A DMA acknowledge, \overline{DACK} , is used as a chip select and initiates a data transfer. The external READ and WRITE signals select the Input and Output FIFO respectively. The \overline{CS} and address lines can also be used as a DMA acknowledge for processors with onboard DMA controllers which do not generate a \overline{DACK} signal.

The internal CPU can configure the UPI-452 to request service from the external host via DMA or interrupts by programming Mode SFR MD6 bit. In addition the external Host enables DMA requests through bits 6 and 7 of the Host Control SFR. When a DMA request is invoked the number of bytes transferred to the Input FIFO is the total number of bytes in the Input FIFO (as determined by the CBP SFR) minus the value programmed in the Input FIFO Threshold SFR. The DMA request line is activated only when the Input FIFO has a threshold number of bytes that can be transferred.

The Output FIFO DMA request is activated when a DSC is written by the internal CPU at the end of a less than threshold size block of data (Flush Mode) or when the Output FIFO threshold is reached. The request remains active until the Input FIFO becomes full or the Output FIFO becomes empty. If a DSC is encountered during an Output FIFO DMA transfer, the DMA request is dropped until the DSC is read. The DMA request will be reactivated after the DSC is read and remains active until the Output FIFO becomes empty or another DSC is encountered.

FIFO MODULE - INTERNAL CPU INTERFACE

Overview

The Input and Output FIFOs are accessed by the internal CPU through direct addressing of the FIFO IN/COMMAND IN and FIFO OUT/COMMAND OUT Special Function Registers. All of the 80C51 instructions involving direct addressing may be used to access the FIFO's SFRs. The FIFO IN, COMMAND IN and Immediate Command In SFRs are actually read only registers, and their Output counterparts are write only. Internal DMA transfers data between Internal memory, External Memory and the Special Function Registers. The Special Function Registers appear as another group of dedicated memory addresses and are programmed as the source or desti-

nation via the DMA0/DMA1 Source Address or Destination Address Special Function Registers. The FIFO module manages the transfer of data between the external host and FIFO SFRs.

Internal CPU Access to FIFO Via Software Instructions

The internal CPU has access to the Input and Output FIFOs via the FIFO IN/COMMAND IN and FIFO OUT/COMMAND OUT SFRs which reside in the Special Function Register Array. At the end of every instruction that involves a read of the FIFO IN/COMMAND IN SFR, the SFR is written over by a new byte from the Input FIFO channel when available. At the end of every instruction that involves a write to the FIFO OUT/COMMAND OUT SFR, the new byte is written into the Output FIFO channel and the write pointer is incremented after the write operation (post incremented).

The internal CPU reads the Input FIFO by using the FIFO IN/COMMAND IN SFR as the source register in an instruction. Those instructions which read the Input FIFO are listed below:

```
ADD A,FIFO IN/COMMAND IN
ADDC A,FIFO IN/COMMAND IN
PUSH FIFO IN/COMMAND IN
ANL A,FIFO IN/COMMAND IN
ORL A,FIFO IN/COMMAND IN
XRL A,FIFO IN/COMMAND IN
CJNE A,FIFO IN/COMMAND IN, rel
SUBB A,FIFO IN/COMMAND IN
MOV direct,FIFO IN/COMMAND IN
MOV @Ri,FIFO IN/COMMAND IN
MOV Rn,FIFO IN/COMMAND IN
MOV A,FIFO IN/COMMAND IN
```

After each access to these registers, they are overwritten by a new byte from the FIFO.

NOTE:

Instructions which use the FIFO IN or COMMAND IN SFR as both a source and destination register will have the data destroyed as the next data byte is rewritten into the FIFO IN register at the end of the instruction. These instructions are not supported by the UPI-452 FIFO. Data can only be read through the FIFO IN SFR and DSCs through the COMMAND IN SFR. Data read through the COMMAND IN SFR will be read as 0FFH, and DSCs read through the FIFO IN SFR will be read as 0FFH. The Immediate Command in SFR is read with the same instructions as the FIFO IN and COMMAND IN SFRs.

The FIFO IN, COMMAND IN and Immediate Command In SFRs are read only registers. Any write operation performed on these registers will be ignored and the FIFO pointers will remain intact.

The internal CPU uses the FIFO OUT SFR to write to the Output FIFO and any instruction which uses the FIFO OUT or COMMAND OUT SFR as a destination will invoke a FIFO write. DSCs are differentiated from data by writing to the COMMAND OUT SFR. In the FIFO, Data Stream Commands have the ninth bit associated with the command byte set to "1". The instructions used to write to the Output FIFO are listed below:

```
MOV FIFO OUT/COMMOUT, A
MOV FIFO OUT/COMMOUT, direct
MOV FIFO OUT/COMMOUT, Rn
POP FIFO OUT/COMMOUT
MOV FIFO OUT/COMMOUT, #data
MOV FIFO OUT/COMMOUNT, @Ri
```

NOTE:

Instructions which use the FIFO OUT/COMMAND OUT SFRs as both a source and destination register cause invalid data to be written into the Output FIFO. These instructions are not supported by the UPI-452 FIFO.

GENERAL PURPOSE DMA CHANNELS

Overview

There are two identical General Purpose DMA Channels on the UPI-452 which allow high speed data transfer from one writeable memory space to another. As many as 64K bytes can be transferred in a single DMA operation. The following memory spaces can be used with DMA channels:

- Internal Data Memory
- External Data Memory
- Special Function Registers

The Special Function Register array appears as a limited group of dedicated memory addresses. The Special Function Registers may be used in DMA transfer operations by specifying the SFR as the source or destination address. The Special Function Registers which may be used in DMA transfers are listed in Table 4. Table 4 also shows whether the SFR may be used as Source or Destination only, or both.

The FIFO can be accessed during DMA by using the FIFO IN SFR as the DMA Source Address Register (SAR) or the FIFO OUT SFR as the Destination Ad-

dress Register (DAR). (Note: Since the FIFO IN SFR is a read only register, the DMA transfer will be ignored if it is used as a DMA DAR. This is also true if the FIFO OUT SFR is used as a DMA SAR.)

Each DMA channel is software programmable to operate in either Block Mode or Demand Mode. In the Block Mode, DMA transfers can be further programmed to take place in Burst Mode or Alternate Cycle mode. In Burst Mode, the processor halts its execution and dedicates its resources to the DMA transfer. In Alternate Cycle Mode, DMA cycles and instruction cycles occur alternately.

In Demand Mode, a DMA transfer occurs only when it is demanded. Demands can be accepted from an external device (through External Interrupt pins, EXT0/EXT1) or from either the Serial Channel or FIFO flags. In this way, a DMA transfer can be synchronized to an external device, the FIFO or the Serial Port. If the External Interrupt is configured in Edge Mode, a single byte transfer occurs per transition. The external interrupt itself will occur if enabled. If the External Interrupt is configured in Level Mode, DMA transfers continue until the External Interrupt request goes inactive or the byte count becomes zero. The following flags activate Demand Mode transfers of one byte to/from the FIFO or Serial Channel:

RI - Serial Channel Receiver Buffer Full

TI - Serial Channel Transmitter Buffer Empty

Architecture

There are three 16 bit and one 8 bit Special Function Registers associated with each DMA channel.

- The 16 bit Source Address SFR (SAR) points to the source byte.
- The 16 bit Destination Address SFR (DAR) points to the destination.
- The 16 bit Byte Count SFR (BCR) contains the number of bytes to be transferred and is decremented when a byte transfer is accomplished.
- The DMA Control SFR (DCON) is eight bits wide and specifies the source memory space, destination memory space and the mode of operation.

In Auto Increment mode, the Source Address and/or Destination Address is incremented when a byte is transferred. When a DMA transfer is complete (BCR = 0), the DONE bit is set and a maskable interrupt is generated. The GO bit must be set to start any DMA transfer (also, the Slave Control SFR FRZ bit must be set to disable FIFO DMA Freeze Mode). The two DMA channels are designated as DMA0 and DMA1, and their corresponding registers are suffixed by 0 or 1; e.g. SAR0, DAR1, etc.

Table 4. DMA Accessible Special Function Registers

SFR	Symbol	Address	Source Only	Destination Only	Either
Accumulator	A/ACC	0E0H			Y
B Register	B	0F0H			Y
FIFO IN	FIN	0EEH	Y		
COMMAND IN	CIN	0EFH	Y		
FIFO OUT	FOUT	0FEH		Y	
COMMAND OUT	COUT	0FFH		Y	
Serial Data Buffer	SBUF	099H			Y
Port 0	P0	080H			Y
Port 1	P1	090H			Y
Port 2	P2	0A0H			Y
Port 3	P3	0B0H			Y
Port 4	P4	0C0H			Y

DMA Special Function Registers

DMA Control SFR: DCON0, DCON1

**Symbolic
Address**
**Physical
Address**

DCON0	DAS	IDA	SAS	ISA	DM	TM	DONE	GO	092H
DCON1	DAS	IDA	SAS	ISA	DM	TM	DONE	GO	093H
	(MSB)						(LSB)		

Reset Status: DCON0 and DCON1 = 00H

Bit Definition:

DAS	IDA	Destination Address Space
0	0	External Data Memory without Auto-Increment
0	1	External Data Memory with Auto-Increment
1	0	Special Function Register
1	1	Internal Data Memory

SAS	ISA	Source Address Space
0	0	External Data Memory without Auto-Increment
0	1	External Data Memory with Auto-Increment
1	0	Special Function Register
1	1	Internal Data Memory

DM	TM	DMA Transfer Mode
0	0	Alternate-Cycle Transfer Mode
0	1	Burst Transfer Mode
1	0	FIFO or Serial Channel Demand Mode
1	1	External Demand Mode

DONE DMA transfer Flag:

- 0 DMA transfer is not completed.
- 1 DMA transfer is complete.

NOTE:

This flag is set when contents of the Byte Count SFR decrements to zero. It is reset automatically when the DMA vectors to its interrupt routine.

GO Enable DMA Transfer:

- 0 Disable DMA transfer (in all modes).
- 1 Enable DMA transfer. If the DMA is in the Block mode, start DMA transfer if possible. If it is in the Demand mode, enable the channel and wait for a demand.

NOTE:

The GO bit is reset when the BCR decrements to zero.

DMA Transfer Modes

The following four modes of DMA operation are possible in the UPI-452.

1. ALTERNATE-CYCLE MODE

General

Alternate cycle mode is useful when CPU processing must occur during the DMA transfers. In this mode, a DMA cycle and an instruction cycle occur alternately. The interrupt request is generated (if enabled) at the end of the process, i.e. when BCR decrements to zero. The transfer is initiated by setting the GO bit in the DCON SFR.

Alternate-Cycle FIFO Demand Mode

Alternate cycle demand mode is useful for FIFO transfers of a less urgent nature. As mentioned before, CPU instruction cycles are interleaved with DMA transfer cycles, allowing true parallel processing.

This mode differs from FIFO Demand Mode in that CPU instruction cycles must be interleaved with DMA transfers, even if the FIFO is demanding DMA. In FIFO Demand Mode, CPU cycles would never occur if the FIFO demand was present.

Input Channel

The DMA is configured as in FIFO Demand Mode and transfers are initiated whenever an input FIFO

service request is generated. DMA transfer cycles are alternated with instruction execution cycles. DMA transfers are terminated as in FIFO Demand Mode.

Output Channel

The DMA is configured as in FIFO Demand Mode and transfers are initiated whenever an Output FIFO requests service. DMA transfer cycles are alternated with instruction execution cycles. DMA transfers are terminated as in FIFO Demand Mode.

The FIFO logic resets the interrupt flag after transferring the byte, so the interrupt is never generated.

2. BURST MODE

In BURST mode the DMA is initiated by setting the GO bit in the DCON SFR. The DMA operation continues until BCR decrements to zero (zero byte count), then an interrupt is generated (if enabled). No interrupts are recognized during a DMA operation once started.

Input Channel

The FIFO Input Channel can be used in burst mode by specifying the FIFO IN SFR as the DMA Source Address. DMA transfers begin when the GO bit in the DMA Control SFR is set. The number of bytes to be transferred must be specified in the Byte Count SFR (BCR) and auto-incrementing of the SAR must be disabled. Once the GO bit is set nothing can interrupt the transfer of data until the BCR is zero. In this mode, a Data Stream Command encountered in the FIFO will be held in the COMMAND IN SFR with the pointers frozen, and invalid data (FFH) will be read through the FIFO IN SFR. If the input FIFO becomes empty during the block transfer, an 0FFH will be read until BCR decrements to zero.

Output Channel

The Output FIFO Channel can be used in burst mode by specifying the FIFO OUT or COMMAND OUT SFR as the DMA Destination Address. DMA transfers begin when the GO bit is set. This mode can be used to send a block of data or a block of Data Stream Commands. If the FIFO becomes full during the block transfer, the remaining data will be lost.

NOTE:

All interrupts including FIFO interrupts are not recognized in Burst Mode. Burst Mode transfers should be used to service the FIFO only when the user is certain that no Data Stream Commands are in the block to be transferred (Input FIFO) and that the FIFO contains enough space to store the block to be transferred. In all other cases Alternate Cycle or Demand Mode should be used.

3. FIFO AND SERIAL CHANNEL DEMAND MODES

FIFO Demand Mode

Although any DMA mode is possible using the FIFO buffer, only FIFO Demand and Alternate Cycle FIFO Demand Modes are recommended. FIFO Demand Mode DMA transfers using the input FIFO Channel are set-up by setting the GO bit and specifying the FIFO IN register as the DMA Source Address Register. The BCR should be set to the maximum number of expected transfers. The user must also program bit 1 of the Slave Control Register (SC1) to determine whether the Slave Status (SSTAT) SFR FIFO Request For Service Flag will be set when the FIFO becomes not empty or full. Once the Request For Service Flag is set by the FIFO, the DMA transfer begins, and continues until the request flag is deactivated. While the request is active, nothing can interrupt the DMA (i.e. it behaves like burst mode). The DMA Request is held active until one of the following occurs:

- 1) The FIFO becomes empty.
- 2) A Data Stream Command is encountered (this generates a FIFO interrupt and DMA operation resumes after the Data Stream Command is read).
- 3) BCR = 0 (this generates a DMA interrupt and sets the DONE bit).

DMA transfers to the Output FIFO Channel are similar. The FIFO OUT or COMMAND OUT SFR is the DMA Destination Address SFR and a transfer is started by setting the GO bit. The user programs bit 0 of the Slave Control SFR (SC0) to determine whether a demand occurs when the Output FIFO is not full or empty. DMA transfers begin when the Request For Service Flag is set by the FIFO logic and continue as long as the flag is set. The Flag remains set until one of the following occurs:

- 1) The FIFO becomes full
- 2) BCR = 0 (this generates a DMA interrupt and sets the DONE bit).

As in Alternate Cycle FIFO Demand Mode, the FIFO logic resets the interrupt flag after transferring the byte, so the interrupt is never generated.

Serial Channel Demand Mode

Serial Channel Demand Mode is the logical choice when using the Serial Port. The DMAs can be activated by one of the Serial Channel Flags. Receiver interrupt (RI) or Transmitter interrupt (TI).

After the GO bit is set, the DMA is activated if one of the following conditions takes place:

SAR(0/1) = SBUF and RI flag is set
 DAR(0/1) = SBUF and TI flag is set
 SAR(0/1) = FIFO IN and IFRS flag is set
 DAR(0/1) = FIFO OUT and OFRS flag is set

NOTE:

TI flag must be set by software to initiate the first transfer.

When the DMA transfer begins, only one byte is transferred at a time. The serial port hardware automatically resets the flag after completion of the transfer, so an interrupt will not be generated until BCR = 0.

4. EXTERNAL DEMAND MODE

The DMA can be initiated by an external device via External interrupt 0 and 1 (INT0/INT1) pins. The INT0 pin demands DMA0 (Channel 0) and INT1 demands DMA1 (Channel 1). If the interrupts are configured in edge mode, a single byte transfer is accomplished for every request. Interrupts also result (INT0 and INT1) after every byte transfer (if enabled). If the interrupts are configured in level mode, the DMA transfer continues until the request goes inactive or BCR = 0. In either case, a DMA interrupt is generated (if enabled) when BCR = 0. The GO bit must be set for the transfer to begin.

EXTERNAL MEMORY DMA

When transferring data to or from external memory via DMA, the HOLD (HLD) and HOLD-ACKNOWLEDGE (HLDA) signals are used for handshaking. The HOLD and HOLD-ACKNOWLEDGE are active low signals which arbitrate control of the local bus. The UPI-452 can be used in a system where multi-masters are connected to a single parallel Address/Data bus. The HLD/HLDA signals are used to share

resources (memory, peripherals, etc.) among all the processors on the local bus. The UPI-452 can be configured in any of three different External Memory Modes controlled by bits 5 and 6 (REQ & ARB) in the PCON SFR (Table 5). Each mode is described below:

REQUESTER MODE: In this mode, the UPI-452 is not the bus master, but must request the bus from another device. The UPI-452 configures port pin P1.5 as a HLD output and pin P1.6 as a HLDA input. The UPI-452 issues a HLD signal when it needs external access for a DMA channel. It uses the local bus after receiving the HLDA signal from the bus master, and will not release the bus until its DMA operation is complete.

ARBITER MODE: In this mode, the UPI-452 is the bus master. It configures port pin P1.5 as HLD input and pin P1.6 as HLDA output. When a device asserts the HLD signal to use the local bus, the UPI-452 asserts the HLDA signal after current instruction execution is complete. If the UPI-452 needs an external access via a DMA channel, it waits until the requester releases the bus, HLD goes inactive.

DISABLE (NON-DMA) MODE: When external program memory is accessed by an instruction or by program counter overflow beyond the internal ROM address or external data memory is accessed by MOVX instructions, it is a local memory access and the HLD/HLDA logic is not initiated. When a DMA channel attempts data transfer to/from the external data memory, the HLD/HLDA logic is initiated as described below. DMA transfer from internal memory space to internal memory space does not initiate the HLD/HLDA logic.

The balance of the PCON SFR bits are described in the "80C51 Register Description: Power Control SFR" section below.

Latency

When the GO bit is set, the UPI-452 finishes the current instruction before starting the DMA operation. Thus the maximum latency is 3.0 microseconds (at 16 MHz).

DMA Interrupt Vectors

Each DMA channel has a unique vectored interrupt associated with it. There are two vectored interrupts associated with the two DMA channels. The DMA interrupts are enabled and priorities set via the Interrupt Enable and Priority SFR (see "Interrupts" section). The interrupt priority scheme is similar to the scheme in 80C51.

When a DMA operation is complete (BCR decrements to zero), the DONE flag in the respective DCON (DCON0 or DCON1) SFR is set. If the DMA interrupt is enabled, the DONE flag is reset automatically upon vectoring to the interrupt routine.

Interrupts When DMA is Active

If a Burst Mode DMA transfer is in progress, the interrupts are not serviced until the DMA transfer is complete. This is also true for level activated External Demand DMA transfers. During Alternate Cycle DMA transfers, however, the interrupts are serviced at the end of the DMA cycle. After that, DMA cycles and instruction execution cycles occur alternately. In the case of edge activated External Demand Mode DMA transfers, the interrupt is serviced at the end of DMA transfer of that single byte.

Table 5. DMA MODE CONTROL - PCON SFR

Symbolic Address								Physical Address
PCON	—*	ARB	REQ	—*	—*	—*	—*	87H
	(MSB)						(LSB)	
	*Defined as per MLS-51 Data Sheet							
	Reset Status: 00H							

Definition:

ARB	REQ	
0	0	HLD/HLDA logic is disabled.
0	1	The UPI-452 is in the Requester Mode.
1	0	The UPI-452 is in the Arbiter Mode.
1	1	Invalid

DMA Arbitration

Only one of the two DMA channels is active at a time, except when both are configured in the Alternate Cycle mode. In this case, the DMA cycles and Instruction Execution cycles occur in the following order:

1. DMA Cycle 0.
2. Instruction execution.
3. DMA Cycle 1.
4. Instruction execution.

DMA0 has priority over DMA1 during simultaneous activation of the two DMA channels. If one DMA channel is active, the other DMA channel, if activated, waits until the first one is complete.

If DMA0 is already in the Alternate Cycle mode and DMA1 is activated in Alternate Cycle Mode, it will take two instruction cycles before DMA1 is activated (due to the priority of DMA0). Once DMA1 becomes active, the execution will follow the normal sequence.

If DMA0 is already in the Alternate Cycle mode and DMA1 is activated in Burst Mode, the DMA1 Burst transfer will follow the DMA0 Alternate Cycle transfer (after the completion of the next instruction).

If the UPI-452 (as a Requester) asserts a HLD signal to request a DMA transfer (see "External Memory DMA") and its other DMA Channel requests a transfer before the HLDA signal is received, the channel having higher priority is activated first. A Burst Mode transfer on channel 0 can not be interrupted since DMA0 has the highest priority. A Demand Mode transfer on channel 0 is the only type of activity that can interrupt a block transfer on DMA1.

If, while executing a DMA transfer, the Arbiter receives a HLD signal, and then before it can acknowledge, its other DMA Channel requests a transfer, it then completes the second DMA transfer before sending the HLDA signal to release the bus to the HLD request.

The DMA Transfer waveforms are in Figures 8-11.

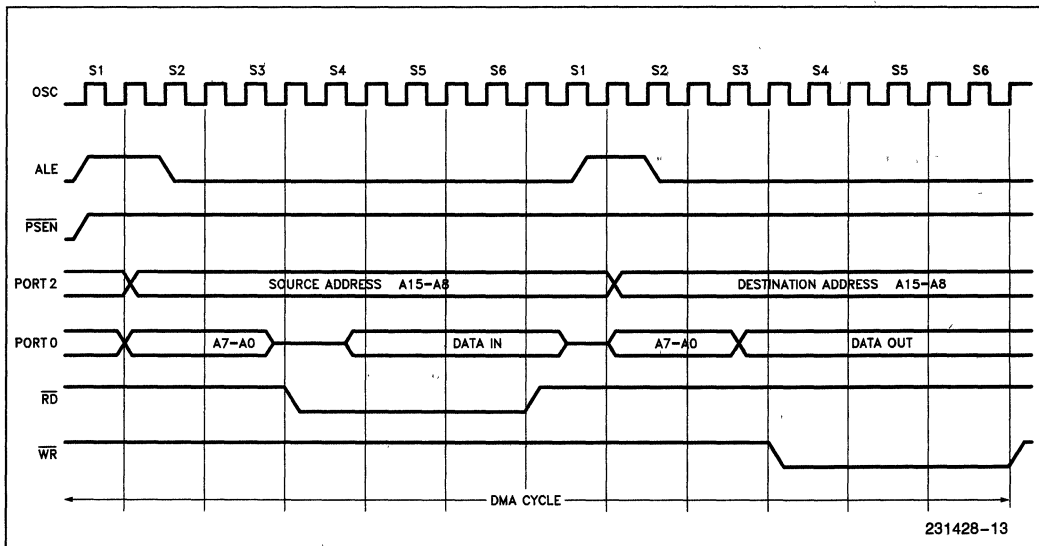


Figure 8. DMA Transfer from External Memory to External Memory

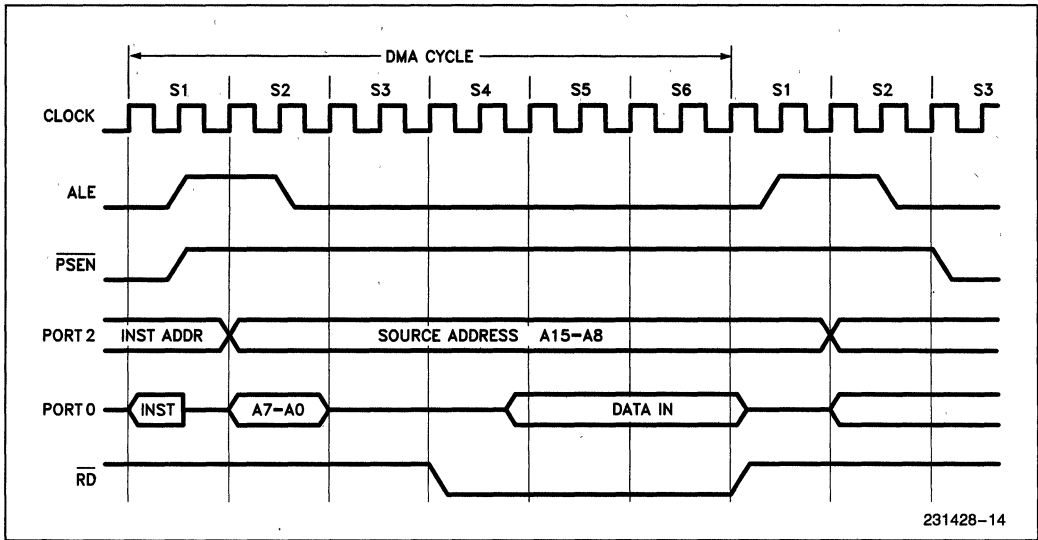


Figure 9. DMA Transfer from External Memory to Internal Memory

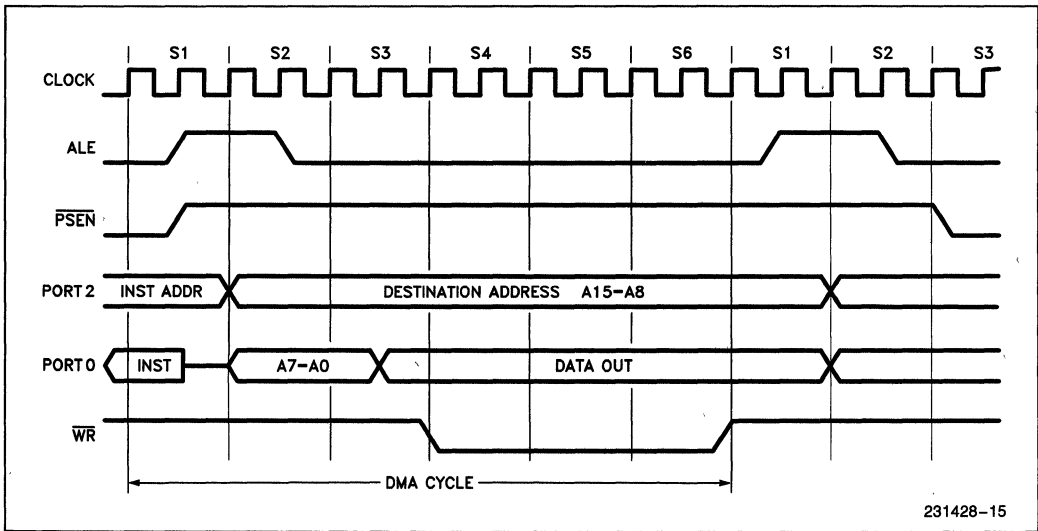


Figure 10. DMA Transfer from Internal Memory to External Memory

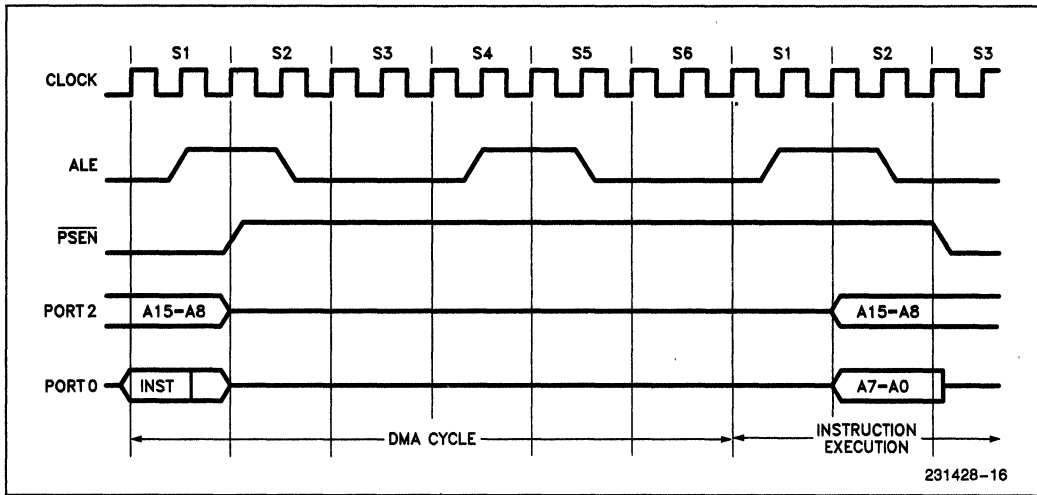


Figure 11. DMA Transfer from Internal Memory to Internal Memory

INTERNAL INTERRUPTS

Overview

The UPI-452 provides a total of eight interrupt sources (Table 6). Their operation is the same as in the 80C51, with the addition of three new interrupt sources for the UPI-452 FIFO and DMA features. These added interrupts have their enable and priority bits in the Interrupt Enable and Priority (IEP) SFR. The IEP SFR is in addition to the 80C51 Interrupt Enable (IE) and Interrupt Priority (IP) SFRs. The added interrupt sources are also globally enabled or disabled by the EA bit in the Interrupt Enable SFR. Table 6 lists the eight interrupt sources in order of priority. Table 7 lists the eight interrupt sources and their respective address vector location in program memory. (DMA interrupts are discussed in the "General Purpose DMA Channels" section. Additional interrupt information for Timer/Counter, Serial Channel, External Interrupt may be found in the Microcontroller Handbook for the 80C51.)

FIFO Module Interrupts to Internal CPU

The FIFO module generates interrupts to the internal CPU whenever the FIFO requests service or when a Data Stream Command is in the COMMAND IN SFR. The Input FIFO will request service whenever it becomes full or not empty depending on bit 1 of the Slave Control SFR (IFRS). Similarly, the Output

Table 6. Interrupt Priority
Interrupt Source **Priority Level**
 (highest)

External Interrupt 0	0
Internal Timer/Counter 0	1
DMA Channel 0 Request	2
External Interrupt 1	3
DMA Channel 1 Request	4
Internal Timer/Counter 1	5
FIFO - Slave Bus Interface	6
Serial Channel	7
	(lowest)

Table 7. Interrupt Vector Addresses

Interrupt Source	Starting Address
External Interrupt 0	3 (003H)
Internal Timer/Counter 0	11 (00BH)
External Interrupt 1	19 (013H)
Internal Timer/Counter 1	27 (01BH)
Serial Channel	35 (023H)
FIFO - Slave Bus Interface	43 (02BH)
DMA Channel 0 Request	51 (033H)
DMA Channel 1 Request	59 (03BH)

FIFO requests service when it becomes empty or not full as determined by bit 0 of the Slave Control SFR (OFRS). Request for Service interrupts are generated only if enabled by the internal CPU via the Interrupt Enable SFR, and the Slave Control Register.

A Data Stream Command Interrupt is generated whenever there is a Data Stream Command in the COMMAND IN SFR. The interrupt is generated to ensure that the internal interrupt is recognized before another instruction is executed.

Immediate Command Interrupts

- a. An Immediate Command IN interrupt is generated, if enabled, to the internal CPU when the Host has written to the Immediate Command IN (IMIN) SFR. The write operation clears the Slave Status SFR bit (SSTAT SST2) and sets the Host Status SFR bit (HSTAT HST2) to indicate that a byte is present in the Immediate Command IN SFR. When the internal CPU reads the Immediate Command IN (IMIN) SFR the Slave Status SFR status bit is set, and the Host Status SFR status bit is cleared indicating the IMIN SFR is empty. Clearing the Host Status SFR bit will cause a Request For Service (INTRQ) interrupt, if enabled, to signal the Host that the IMIN SFR is empty. (See Figure 7a, Immediate Command IN Flowchart.)
- b. An Immediate Command OUT interrupt is generated, if enabled, to the internal CPU when the Host has read the Immediate Command OUT SFR. The Host read causes the Slave Status

Immediate Command OUT bit (SSTAT SST6) to be set and the corresponding Host Status bit (HSTAT HST6) to be cleared indicating the SFR is empty. When the internal CPU writes to the Immediate Command OUT SFR, the Host Status bit is set and Slave Status bit is cleared to indicate the SFR is full. (See Figure 7b, Immediate Command OUT Flowchart.)

NOTE:

Immediate Command IN and OUT interrupts are actually specific FIFO-Slave Interface interrupts to the internal CPU.

One instruction from the main program is executed between two consecutive interrupt service routines as in the 80C51. However, if the second interrupt service routine is due to a Data Stream Command Interrupt, the main program instruction is not executed (to prevent misreading of invalid data).

Interrupt Enabling and Priority

Each of the three interrupt special function registers (IE, IP and IEP) is listed below with its corresponding bit definitions.

Interrupt Enable SFR (IE)

Symbolic Address

Physical Address

IE	EA	—	—	ES	ET1	EX1	ET0	EX0	0A8H
	(MSB)								(LSB)

Symbol	Position	Function
EA	IE.7	Enables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	(reserved)
—	IE.5	(reserved)
ES	IE.4	Serial Channel interrupt enable
ET1	IE.3	Internal Timer/Counter 1 Overflow Interrupt
EX1	IE.2	External Interrupt Request 1.
ET0	IE.1	Internal Timer/Counter 0 Overflow Interrupt
EX0	IE.0	External Interrupt Request 0.

Interrupt Priority SFR (IP)

A priority level of 0 or 1 may be assigned to each interrupt source, with 1 being higher priority level, through the IPC and the IEP (Interrupt Enable and Priority) SFR. A priority level of 1 interrupt can interrupt a priority level 0 service routine to allow nesting of interrupts.

**Symbolic
Address**

**Physical
Address**

IP

—	—	—	PSC	PT1	PX1	PT0	PX0
(MSB)				(LSB)			

0B8H

Symbol	Position	Function	Priority Within A Level
—	IP.7	(reserved)	(lowest)
—	IP.6	(reserved)	—
—	IP.5	(reserved)	—
PSC	IP.4	Local Serial Channel	0.7
PT1	IP.3	Internal Timer/Counter 1	0.5
PX1	IP.2	External Interrupt Request 1	0.3
PT0	IP.1	Internal Timer/Counter 0	0.1
PX0	IP.0	External Interrupt Request 0	0.0 (highest)

Interrupt Enable and Priority SFR (IEP)

The Interrupt Enable and Priority Register establishes the enabling and priority of those resources not covered in the Interrupt Enable and Interrupt Priority SFRs.

**Symbolic
Address**

**Physical
Address**

IEP

—	—	PFIFO	EDMA0	EDMA1	PDMA0	PDMA1	EFIFO
(MSB)				(LSB)			

0F8H

Symbol	Position	Function	Priority Within a Level
—	IEP.7	(reserved)	
—	IEP.6	(reserved)	
PFIFO	IEP.5	FIFO Slave Bus Interface Interrupt Priority	0.6
EDMA0	IEP.4	DMA Channel 0 Interrupt Enable	
EDMA1	IEP.3	DMA Channel 1 Interrupt Enable	
PDMA0	IEP.2	DMA Channel 0 Priority	0.2
PDMA1	IEP.1	DMA Channel 1 Priority	0.4
EFIFO	IEP.0	FIFO Slave Bus Interface Interrupt Enable	

FIFO-EXTERNAL HOST INTERFACE FIFO DMA FREEZE MODE

Overview

During FIFO DMA Freeze Mode the internal CPU can reconfigure the FIFO interface. FIFO DMA Freeze Mode is provided to prevent the Host from accessing the FIFO during a reconfiguration sequence. The internal CPU invokes FIFO DMA Freeze Mode by clearing bit 3 of the Slave Control SFR (SC3). INTRQ becomes active whenever FIFO DMA Freeze Mode is invoked to indicate the freeze status. The interrupt can only be deactivated by the Host reading the Host Status SFR.

During FIFO DMA Freeze Mode only two operations are possible by the Host to the UPI-452 slave, the balance are disabled, as shown in Table 8. The internal DMA is disabled during FIFO DMA Freeze Mode, and the internal CPU has write access to all of the FIFO control SFRs (Table 9).

Initialization

At power on reset the FIFO Host interface is automatically frozen. The Slave Control Enable FIFO

DMA Freeze Mode bit defaults to FIFO DMA Freeze Mode (SLCON FRZ=0). Below is a list of the FIFO Special Function Registers and their default power on reset values;

SFR Name	Label	Value
Channel Boundary Pointer	CBP	40H / 64D
Output Channel Read Pointers	ORPR	40H / 64D
Output Channel Write Pointers	OWPR	40H / 64D
Input Channel Read Pointers	IRPR	00H / 00D
Input Channel Write Pointers	IWPR	00H / 00D
Input Threshold	ITH	00H / 00D
Output Threshold	OTH	01H / 1D

The Input and Output FIFO channels may be reconfigured by programming any of these SFRs while the FIFO Host interface is in FIFO DMA Freeze Mode. The UPI-452 also notifies the Host that FIFO DMA Freeze Mode is in progress by setting the Host Status SFR FIFO DMA Freeze Mode Status bit, FIFO DMA Freeze Mode In Progress. The Host interrogates the Host Status SFR to determine the

Table 8. Slave Bus Interface Status During FIFO DMA Freeze Mode

Interface Pins; DACK	CS	A2	A1	A0	READ	WRITE	Operation In Normal Mode	Status In FIFO DMA Freeze Mode
1	0	0	1	0	0	1	Read Host Status SFR	Operational
1	0	0	1	1	0	1	Read Host Control SFR	Operational
1	0	0	1	1	1	0	Write Host Control SFR	Disabled
1	0	0	0	0	0	1	Data or DMA Data from Output Channel	Disabled
1	0	0	0	0	1	0	Data or DMA Data to Input Channel	Disabled
1	0	0	0	1	0	1	Data Stream Command from Output Channel	Disabled
1	0	0	0	1	1	0	Data Stream Command to Input Channel	Disabled
1	0	1	0	0	0	1	Read Immediate Command Out from Output Channel	Disabled
1	0	1	0	0	1	0	Write Immediate Command In to Input Channel	Disabled
0	X	X	X	X	0	1	DMA Data from Output Channel	Disabled
0	X	X	X	X	1	0	DMA Data to Input Channel	Disabled

status of the FIFO Host interface following reset before attempting to read from or write to the UPI-452 FIFO buffer.

The UPI-452 can also be programmed to interrupt the Host following power on reset in order to indicate to the Host that FIFO DMA Freeze Mode is in progress. This is done by enabling the INTRQ interrupt output pin via the MODE SFR (MD4) before the Slave Control SFR Enable FIFO DMA Freeze Mode bit is set to Normal Mode. At power on reset the Mode SFR is forced to zero. This disables all interrupt and DMA output pins (INTRQ, DRQIN/INTRQIN and DRQOUT/INTRQOUT). Because the Host Status SFR FIFO DMA Freeze Mode In Progress bit is set, a Request For Service, INTRQ, interrupt is pending. This is because the FIFO DMA Freeze Mode interrupt is always enabled. If the Slave Control FIFO DMA Freeze Mode bit (SLCON FRZ) is set to Normal Mode before the MODE SFR INTRQ bit is enabled, the INTRQ output will not go active when the MODE SFR INTRQ bit is enabled.

The default values for the FIFO and Slave Interface represents minimum UPI-452 internal initialization. No specific Special Function Register initialization is required to begin operation of the FIFO Slave Interface. The last initialization instruction must always set the UPI-452 to Normal Mode. This causes the UPI-452 to exit FIFO DMA Freeze Mode and enables Host read/write access of the FIFO.

Following reset, either hardware (via the RST pin) or software (via HCON SFR bit HC3) the UPI-452 requires 2 internal machine cycles (24 TCLCL) to update all internal registers.

Invoking FIFO DMA Freeze Mode During Normal Operation

When the UPI-452 is in normal operation, FIFO DMA Freeze Mode should not be arbitrarily invoked by

clearing SC3 (SC3=0) because the external Host runs asynchronously to the internal CPU. Invoking FIFO DMA Freeze Mode without first stopping the external Host from accessing the UPI-452 will not guarantee a clean break with the external Host.

The proper way to invoke FIFO DMA Freeze Mode is by issuing an Immediate Command to the external host indicating that FIFO DMA Freeze Mode will be invoked. Upon receiving the Immediate Command, the external Host should complete servicing all pending interrupts and DMA requests, then send an Immediate Command back to the UPI-452 acknowledging the FIFO DMA Freeze Mode request. After issuing the first Immediate Command, the internal CPU should not perform any action on the FIFO until FIFO DMA Freeze Mode is invoked.

If FIFO DMA Freeze Mode is invoked without stopping the Host, only the last two bytes of data written into or read from the FIFO will be valid. The timing diagram for disabling the FIFO module to the external Host interface is illustrated in Figure 12. Due to this synchronization sequence, the UPI-452 might not go into FIFO DMA Freeze Mode immediately after SC3 is cleared. A special bit in the Slave Status Register (SST5) is provided to indicate the status of the FIFO DMA Freeze Mode. The FIFO DMA Freeze Mode operations described in this section are only valid after SST5 is cleared.

As FIFO DMA Freeze Mode is invoked, the DRQIN or DRQOUT will be deactivated (stopping the transferring of data), bit 1 of the Host Status SFR will be set (HST1 = 1), and SST5 will be cleared (SST5 = 0) to indicate to the external Host and internal CPU that the slave interface has been frozen. After the freeze becomes effective, any attempt by the external Host to access the FIFO will cause the overrun and underrun bits to be activated (bits HST7 (for reads) or HST3 (for writes)). These two bits, HST3 and HST7, will be set (deactivated) after the Host Status SFR has been read.

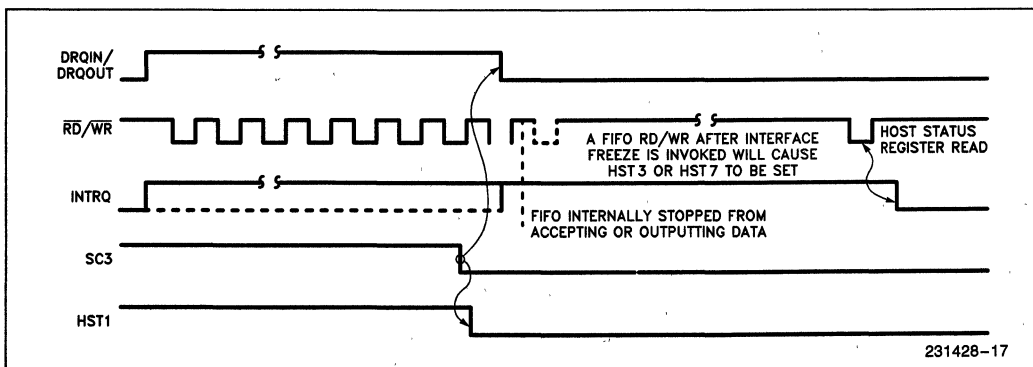


Figure 12. Disabling FIFO to Host Slave Interface Timing Diagram

External Host writing to the Immediate Command In SFR and the Host Control SFR is also inhibited when the slave bus interface is frozen. Writing to these two registers after FIFO DMA Freeze Mode is invoked will also cause HST3 (overrun) to be activated (HST3=0). Similarly, reading the Immediate Command Out Register by the external Host is disabled during FIFO DMA Freeze Mode, and any attempt to do so will cause the clearing (deactivating, "0") of HST7 bit (underrun).

After the slave bus interface is frozen, the internal CPU can perform the following operations on the FIFO Special Function Registers (these operations are allowed only during FIFO DMA Freeze Mode).

For FIFO	1. Changing the Channel Boundary Pointer SFR.
Reconfiguration	2. Changing the Input and Output Threshold SFR.
To Enhance the Testability	3. Writing to the read and write pointers of the Input and Output FIFO's. 4. Writing to and reading the Host Control SFRs. 5. Controlling some bits of Host and Slave Status SFRs. 6. Reading the Immediate Command Out SFR and Writing to the Immediate Command In SFR.

Description of each of these special functions are as follows:

FIFO Module SFRs During FIFO DMA Freeze Mode

Table 9 summarizes the characteristics of all the FIFO Special Function Registers during normal and FIFO DMA Freeze Modes. The registers that require special treatment in FIFO DMA Freeze Mode are: HCON, IWPR, IRPR, OWPR, ORPR, HSTST, SSTAT, IMMIN & IMMOUT SFRs. They can be described in detail as follows:

Host Control SFR (HCON)

During normal operation, this register is written to or read by the external Host. However, in FIFO DMA Freeze Mode (i.e. SST5=0) the UPI-452 internal CPU has write access to the Host Control SFR and write operations to this SFR by the external Host will not be accepted. If the Host attempts to write to

HCON, the Input Channel error condition flag (HST3) will be cleared.

Input FIFO Pointer Registers (IRPR & IWPR)

Once the FIFO module is in FIFO DMA Freeze Mode, error flags due to overrun and underrun of the Input FIFO pointers will be disabled. Any attempt to create an overrun or underrun condition by changing the Input FIFO pointers would result in an inconsistency in performance between the status flag and the threshold counter.

To enhance the speed of the UPI-452, read operations on the Input FIFO will look ahead by two bytes. Hence, every time the IRPR is changed during FIFO DMA Freeze Mode, two NOPs need to be executed so that the two byte pipeline can be updated with the new data bytes pointed to by the new IRPR. The Threshold Counter SFR also needs to change by the same number of bytes as the IRPR (increase Threshold Counter if IRPR goes forward or decrease if IRPR goes backward). This will ensure that future interrupts will still be generated only after a threshold number of bytes are available. (See "Input and Output FIFO Threshold SFR" section below.)

In FIFO DMA Freeze Mode, the internal CPU can also change the content of IWPR, and each change of IWPR also requires an update of the Threshold Counter SFR.

Normally, the internal CPU cannot write into the Input FIFO. It can, however, during FIFO DMA Freeze Mode by first reconfiguring the FIFO as an Output FIFO (Refer to "Input and Output FIFO Threshold SFR" section below). Changing the IRPR to be equal to IWPR generates a full condition while changing IWPR to be equal to IRPR generates an empty condition. The order in which the pointers are written determines whether a full or empty condition is generated.

Output FIFO Pointer SFR (ORPR and OWPR)

In FIFO DMA Freeze Mode the contents of OWPR can be changed by the internal CPU, but each change of OWPR or ORPR requires the Threshold Counter SFR to be updated as described in the next section. A NOP must be executed whenever a new value is written into ORPR, as just described for changes to IRPR. As before, changing ORPR to be equal to OWPR will generate a full condition, Output FIFO overrun or underrun condition cannot be generated though. The FIFO pointers should not be set to a value outside of its range.

Table 9. FIFO SFR's Characteristics During FIFO DMA Freeze Mode

Label	Name	Normal Operation (SST5 = 1)	FIFO DMA Freeze Mode Operation (SST5 = 0)
HCON	Host Control	Not Accessible	Read & Write
HSTAT	Host Status	Read Only	Read & Write 4
SLCON	Slave Control	Read & Write	Read & Write
SSTAT	Slave Status	Read Only	Read & Write 4
IEP	Interrupt Enable & Priority	Read & Write	Read & Write
MODE	Mode Register	Read & Write	Read & Write
IWPR	Input FIFO Write Pointer	Read Only	Read & Write 5
IRPR	Input FIFO Read Pointer	Read Only	Read & Write 1, 5
OWPR	Output FIFO Write Pointer	Read Only	Read & Write 6
ORPR	Output FIFO Read Pointer	Read Only	Read & Write 2, 6
CBP	Channel Boundary Pointer	Read Only	Read & Write 3
IMIN	Immediate Command In	Read Only	Read & Write
IMOUT	Immediate Command Out	Read & Write	Read & Write
FIN	FIFO IN	Read Only	Read Only
CIN	COMMAND IN	Read Only	Read Only
FOUT	FIFO OUT	Read & Write	Read & Write
COUT	COMMAND OUT	Read & Write	Read & Write
ITHR	Input FIFO Threshold	Read Only	Read & Write
OTHR	Output FIFO Threshold	Read Only	Read & Write

NOTES:

1. Writing of IRPR will automatically cause the FIFO IN SFR to load the contents of the Input FIFO from that location.
2. Writing to ORPR will automatically cause the IOBL SFR to load the contents of the Output FIFO at that ORPR address.
3. Writing to the CBP SFR will cause automatic reset of the four pointers of the Input and Output FIFO channels.
4. The internal CPU cannot directly change the status of these registers. However, by changing the status of the FIFO channels, the internal CPU can indirectly change the contents of the status registers.
5. Changing the Input FIFO Read/Write Pointers also requires that a consistent update of the Input FIFO Threshold Counter SFR.
6. Changing the Output FIFO Read/Write Pointers also requires that a consistent update of the Output FIFO Threshold Counter SFR.

Input and Output FIFO Threshold SFR (ITHR & OTHR)

The Input and Output FIFO Threshold SFRs are also programmable by the internal CPU during FIFO DMA Freeze Mode. For proper operation of the Threshold feature, the Threshold SFR should be changed only when the Input and Output FIFO channels are empty, since they reflect the current number of bytes available to read/write before an interrupt is generated.

Table 10 illustrates the Threshold SFRs range of values and the number of bytes to be transferred when the Request For Service Flag is activated:

Table 10. Threshold SFRs Range of Values and Number of Bytes to be Transferred

ITHR (lower seven bits)	No. of Bytes Available to be Written	OTHR (lower seven bits)	No. of Bytes Available to be Read
0	CBP	1	2
1	CBP-1	2	3
2	CBP-2	3	4
•	•	•	•
•	•	•	•
•	•	•	•
CBP-3	3	(80H-CBP)-3	(80H-CBP)-2
CBP-2	2	(80H-CBP)-2	(80H-CBP)-1
		(80H-CBP)-1	(80H-CBP)

The eighth bit of the Input and Output FIFO Threshold SFR indicates the status of the service requests regardless of the freeze condition. If the eighth bit is a "1", the FIFO is requesting service from the external Host. In other words, when the Threshold SFR value goes below zero (2's complement), a service request is generated. Normally the ITHR SFR is decremented after each external Host write to the Input FIFO and incremented after each internal CPU read of the Input FIFO. The OTHR SFR is decremented by internal CPU writes and incremented by external Host reads. Thus if the pointers are moved when the FIFO's are not empty, these relationships can be used to calculate the offset for the Threshold SFRs. It is best to change the Threshold SFRs only when the FIFO's are empty to avoid this complication. The threshold registers should also be updated after the pointers have been manipulated.

NOTE:

When programming the ITHR SFR, the eighth bit should be set to 1 (OR'd with 80H). This causes HSTAT SFR HST0 = 0, Input FIFO Request For Service. If ITHR bit 7 = 0 then HSTAT HST0 = 1, Input FIFO Does Not Request Service, and no interrupt will be generated.

Host Status SFR (HSTAT)

When in FIFO DMA Freeze Mode, some bits in the Host Status SFR are forced high and will not reflect the new status until the system returns to normal operation. The definition of the register in FIFO DMA Freeze Mode is as follows:

NOTE:

The internal CPU reads this shadow latch value when reading the Host Status SFR. The shadow latch will keep the information for these bits so normal operation can be resumed with the right status. The following bits are cleared (=0) when FIFO DMA Freeze Mode is invoked;

HST7 Output FIFO Error Condition Flag

1 = No error.

0 = An invalid read has been done on the output FIFO or the Immediate Command Out Register by the host CPU.

NOTE:

The normal underrun error condition status is disabled. If an Immediate Command Out (IMOUT) SFR read is attempted during FIFO DMA Freeze Mode, the contents of the IMOUT SFR is output on the Data Buffer and the error status is set (= 1).

HST6 Immediate Command Out SFR Status

During normal operation, this bit is cleared (=0) when the IMOUT SFR is written by the UPI-452 internal CPU and set (= 1) when the IMOUT SFR is read by the external Host. Once the host-slave interface is frozen (i.e. SST5 = 0), this bit will be read as a 1 by the host CPU. A shadow latch will keep the information for this bit so normal operation can be resumed with the correct status.

Shadow latch:

1 = Internal CPU reads the IMOUT SFR

0 = Internal CPU writes to the IMOUT SFR

HST5 Data Stream Command at Output FIFO

This bit is forced to a "1" during FIFO DMA Freeze Mode to prevent the external host CPU from trying to read the DSC. Once normal operation is resumed, HST5 will reflect the Data/Command status of the current byte in the Output FIFO.

Shadow Latch (read by the internal CPU):

1 = No Data Stream Command (DSC)

0 = Data Stream Command at Output FIFO

HST4 Output FIFO Service Request Status

When FIFO DMA Freeze Mode is invoked, this bit no longer reflects the Output FIFO Request Service Status. This bit will be forced to a "1".

HST3 Input FIFO Error Condition Flag

- 1 = No error.
- 0 = One of the following operations has been attempted by the external host and is invalid:
 - 1) Write into the Input FIFO
 - 2) Write into the Host Control SFR
 - 3) Write into the Immediate Command In SFR

NOTE:

The normal Input FIFO overrun condition is disabled.

HST2 Immediate Command In SFR Status

This bit is normally cleared when the internal CPU reads the IMIN SFR and set when the external host CPU writes into the IMIN SFR. When the host-slave interface is frozen, reading and writing of the IMIN will change the shadow latch of this bit. This bit will be read as a "1" by the external Host.

Shadow latch.

- 1 = Internal CPU writes into IMIN SFR
- 0 = Internal CPU reads the IMIN SFR

HST1 FIFO DMA Freeze Mode Status

- 1 = FIFO DMA Freeze Mode.
- 0 = Normal Operation (non-FIFO DMA Freeze Mode).

NOTE:

This bit is used to indicate to the external Host that the host-slave interface has been frozen and hence the external Host functions are now reduced as shown in Table 8.

HST0 Input FIFO Request Service Status

When slave interface is frozen this bit no longer reflects the Input FIFO Request Service Status. This bit will be forced to a "1".

Slave Status SFR (SSTAT)

The Slave Status SFR is a read-only SFR. However, once the slave interface is frozen, most of the bits of this SFR can be changed by the internal CPU by reconfiguring the FIFO and accessing the FIFO Special Function Registers.

SST7 Output FIFO Overrun Error Flag

Inoperative in FIFO DMA Freeze Mode.

SST6 Immediate Command Out SFR Status

In FIFO DMA Freeze Mode, this bit will be cleared when the internal CPU reads the Immediate Command Out SFR and set when the internal CPU writes to the Immediate Command Out Register.

SST5 FIFO-External Interface FIFO DMA Freeze Mode Status

This bit indicates to the internal CPU that FIFO DMA Freeze Mode is in progress and that it has write access to the FIFO Control, Host control and Immediate Command SFRs.

SST4 Output FIFO Request Service Status

During normal operation, this bit indicates to the internal CPU that the Output FIFO is ready for more data. The status of this bit reflects the position of the Output FIFO read and write pointers. Hence, in FIFO DMA Freeze Mode, this flag can be changed by the internal CPU indirectly as the read and write pointers change.

SST3 Input FIFO Underrun Flag

Inoperative during FIFO DMA Freeze Mode.

During normal operation, a read operation clears (=0) this bit when there are no data bytes in the Input FIFO and deactivated (=1) when the Slave Status SFR is read. In FIFO DMA Freeze Mode, this bit will not be cleared by an Input FIFO read underrun error condition, nor will it be reset by the reading of the Slave Status SFR.

SST2 Immediate Command In SFR Status

This bit is normally activated (=0) when the external host CPU writes into the Immediate Command In SFR and deactivated (=1) when it is read by the internal CPU. In FIFO DMA Freeze Mode, this bit will not be activated (=0) by the external Host's writing of the Immediate Command IN SFR since this function is disabled. However, this bit will be cleared (=0) if the internal CPU writes to the Immediate Command In SFR and it will be set (=1) if it reads from the register.

SST1 Data Stream Command at Input FIFO Flag

In FIFO DMA Freeze Mode, this bit operates normally. It indicates whether the next byte of data from the Input FIFO is a DSC or data byte. If it is a DSC byte, reading from FIFO IN SFR will result in reading invalid data (FFH) and vice versa. In FIFO DMA Freeze Mode, this bit still reflects the type of data byte available from the Input FIFO.

SST0 Input FIFO Service Request Flag

During normal operation, this bit is activated (=0) when the Input FIFO contains bytes that can be read by the internal CPU and deactivated (=1) when the Input FIFO does not need any service from the internal CPU. In FIFO DMA Freeze Mode, the status of this bit should not change unless the pointers of the Input FIFO are changed. In this mode, the internal CPU can indirectly change this bit by changing the read and write pointers of the Input FIFO but cannot change it directly.

Immediate Command In/Out SFR (IMIN/IMOUT)

If FIFO DMA Freeze Mode is in progress, writing to the Immediate Command In SFR by the external host will be disabled, and any such attempt will cause HST3 to be cleared (=0). Similarly, the Immediate Command Out SFR read operation (by the host) will be disabled internally and read attempts will cause HST7 to be cleared (=0).

Internal CPU Read and Write of the FIFO During FIFO DMA Freeze Mode

In normal operation, the Input FIFO can only be read by the internal CPU and similarly, the Output FIFO can only be written by the internal CPU. During FIFO DMA Freeze Mode, the internal CPU can read the entire contents of the Input FIFO by programming the CBP SFR to 7FH, setting the IRPR SFR to zero, and then the IWPR SFR to zero. Programming the pointer registers in this order generates a FIFO full signal to the FIFO logic and enables internal CPU read operations. If the IWPR and IRPR are already zero, the write pointer should be changed to a non-zero value to clear the empty status then the pointers can be set to zero. This allows the input FIFO look ahead registers to be updated when the read pointer is changed.

In a similar manner, the internal CPU can write to all 128 bytes of the FIFO by setting the CBP SFR to zero, setting OWPR SFR to zero, and then setting ORPR SFR to zero. This generates a FIFO empty signal and allows internal CPU write operations to all 128 bytes of the FIFO. The Threshold registers also need to be adjusted when the pointers are changed. (See "Input and Output FIFO Threshold SFR" section below.)

MEMORY ORGANIZATION

The UPI-452 has separate address spaces for Program Memory and Data Memory like the 80C51. The

Program Memory can be up to 64K bytes. The lower 8K of Program Memory may reside on-chip. The Data Memory consists of 256 bytes of on-chip RAM, up to 64K bytes of off-chip RAM and a number of "SFRs" (Special Function Registers) which appear as yet another set of unique memory addresses. The 80C51 Special Function Registers are listed in Table 11a, and the additional UPI-452 SFRs are listed in Table 11b. A brief description of the 80C51 core SFRs is also provided below.

Accessing External Memory

As in the 80C51, accesses to external memory are of two types: Accesses to external Program Memory and accesses to external Data Memory.

External Program Memory is accessed under two conditions:

- 1) Whenever signal $\overline{EA} = 0$; or
- 2) Whenever the program counter (PC) contains a number that is larger than 1FFFH.

This requires that the ROMless versions have \overline{EA} wired low to enable the lower 8K program bytes to be fetched from external memory.

External Data Memory is accessed using either the MOVX @DPTR (16 bit address) or the MOVX @Ri (8 bit address) instructions.

Table 11a. 80C51 Special Function Registers;

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer (consisting of DPH and DPL)	82H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
TCON	Timer/Counter Control	88H
TH0	Timer/Counter 0 (high byte)	8CH
TLO	Timer/Counter 0 (low byte)	8AH
TH1	Timer/Counter 1 (high byte)	8DH
TL1	Timer/Counter 1 (low byte)	8BH
*SCON	Serial Control	98H
SBUF	Serial Data Buff	99H
PCON	Power Control	87H

Table 11b. UPI-452 Additional Special Function Registers

ITHR	Input FIFO Threshold	0F6H
OTHR	Output FIFO Threshold	0F7H
*SLCON	Slave Control	0E8H
SSTAT	Slave Status	0E9H
*IEP	Interrupt Enable & Priority	0F8H
MODE	Mode Register	0F9H
IWPR	Input Write Pointer	0EAH
IRPR	Input Read Pointer	0EBH
ORPR	Output Read Pointer	0FAH
OWPR	Output Write Pointer	0FBH
CBP	Channel Boundary Pointer	0ECH
IMIN	Immediate Command In	0FCH
IMOUT	Immediate Command Out	0FDH
FIN	FIFO IN	0EEH
CIN	COMMAND IN	0EFH
FOUT	FIFO OUT	0FEH
COUT	COMMAND OUT	0FFH
*P4	Port 4	0C0H
HSTAT	Host Status	0E6H
HCON	Host Control	0E7H
DCONO	DMA0 Control	92H
DCON1	DMA1 Control	93H
	DMA Source Address	
SARLO	low byte/	0A2H
SARHO	hi byte/ channel 0	0A3H
SARL1	low byte/	0B2H
SARH1	hi byte/ channel 1	0B3H
	DMA Destination Address	
DARLO	low byte/	0C2H
DARHO	hi byte/ channel 0	0C3H
DARL1	low byte/	0D2H
DARH1	hi byte/ channel 1	0D3H
	DMA Byte Count	
BCRLO	low byte/	0E2H
BCRHO	hi byte/ channel 0	0E3H
BCRL1	low byte/	0F2H
BCRH1	hi byte/ channel 1	0F3H

The SFRs marked with an asterisk (*) are both bit- and byte- addressable. The functions of the SFRs are as follows:

Miscellaneous Special Function Register Description

80C51 SFRs

ACCUMULATOR

ACC is the Accumuator SFR. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

B REGISTER

The B SFR is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

PROGRAM STATUS WORD

The PSW SFR contains program status information as detailed in Table 12.

STACK POINTER

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

PORTS 0 TO 4

P0, P1, P2, P3 and P4 are the SFR latches of Ports 0, 1, 2, 3 and 4, respectively.

SERIAL DATA BUFFER

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

TIMER/COUNTER SFR

Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit counting registers for Timer/Counters 0 and 2.

POWER CONTROL SFR (PCON)

The PCON Register (Table 13) controls the power down and idle modes in the UPI-452, as well as providing the ability to double the Serial Channel baud rate. There are also two general purpose flag bits available to the user. Bits 5 and 6 are used to set the DMA mode (see "General Purpose DMA Channels" section), and bit 4 is not used.

Table 12. Program Status Word

Symbolic Address									Physical Address
PSW	CY	AC	FO	RS1	RS0	OV	—	P	0D0H
	(MSB)				(LSB)				

Symbol	Position	Name
CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry (For BCD operations)
FO	PSW.5	Flag 0 (user assignable)
RS1	PSW.4	Register Bank Select bit 1*
RS0	PSW.3	Register Bank Select bit 0*
OV	PSW.2	Overflow Flag
—	PSW.1	(reserved)
P	PSW.0	Parity Flag

*(RS1, RS0) enable internal RAM register banks as follows:

RS1	RS0	Internal RAM Register Bank
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

Table 13. PCON Special Function Register

Symbolic Address									Physical Address
PCON	SMOD	ARB	REQ	—	GF1	GF0	PD	IDL	087H
	(MSB)				(LSB)				

Symbol	Position	Function
SMOD	PCON7	Double Baud rate bit. When set to a 1, the baud rate is doubled when the serial port is being used in either Mode 1, 2 or 3.
ARB	PCON6	DMA Arbiter control bit *
REQ	PCON5	DMA Requestor control bit *
—	PCON4	(reserved)
GF1	PCON3	General-purpose flag bit
GF0	PCON2	General-purpose flag bit
PD	PCON1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON0	Idle Mode bit. Setting this bit activates idle mode operation.

*See "DMA Transfer Mode" description.

NOTE:

If 1's are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (000X0000).

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias0°C to 70°C†
 Storage Temperature -65°C to +150°C
 Voltage on Any
 Pin to V_{SS} -0.5V to V_{CC} + 0.5V
 Voltage on V_{CC} to V_{SS} -0.5V to +6.5V
 Power Dissipation 1.0W**
 V_{CC}/V_{PP} Supply Voltage with
 Respect to Ground
 During Programming -0.6V to +14.0V

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

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5V ± 10%; V_{SS} = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V _{IL}	Input Low Voltage (except EA)	-0.5	0.2 V _{CC} - 0.1	V	
V _{IL1}	Input Low Voltage (EA)	-0.5	0.2 V _{CC} - 0.3	V	
V _{IH}	Input High Voltage (except XTAL1, RST)	0.2 V _{CC} + 0.9	V _{CC} + 0.5	V	
V _{IH1}	Input High Voltage (XTAL1, RST)	0.7 V _{CC}	V _{CC} + 0.5	V	
V _{OL}	Output Low Voltage (Ports 1, 2, 3, 4)		0.45	V	I _{OL} = 1.6 mA (Note 1)
V _{OL1}	Output Low Voltage (except Ports 1, 2, 3, 4)		0.45	V	I _{OL} = 3.2 mA (Note 1)
V _{OH}	Output High Voltage (Ports 1, 2, 3, 4)	2.4		V	I _{OH} = -60 μA, V _{CC} = 5V ± 10%
		0.75 V _{CC}		V	I _{OH} = -25 μA
		0.9 V _{CC}		V	I _{OH} = -10 μA
V _{OH1}	Output High Voltage (except Ports 1, 2, 3, 4 and Host Interface (Slave) Port)	2.4		V	I _{OH} = -400 μA, V _{CC} = 5V ± 10%
		0.75 V _{CC}		V	I _{OH} = -150 μA
		0.9 V _{CC}		V	I _{OH} = -40 μA (Note 2)
V _{OH2}	Output High Voltage (Host Interface (Slave) Port)	2.4		V	I _{OH} = -400 μA, V _{CC} = 5V ± 10%
		3.0		V	I _{OH} = -40 μA
		V _{CC} - 0.4		V	I _{OH} = -10 μA
I _{IL}	Logical 0 Input Current (Ports 1, 2, 3, 4)		-50	μA	V _{IN} = 0.45V
I _{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3, 4)		-650	μA	V _{IN} = 2V

† Ambient Temperature under Bias for 87C452P is 0°C to 50°C.

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to 70°C ; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$ (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{LI}	Input Leakage Current (except Ports 1, 2, 3, 4)		± 10	μA	$0.45\text{V} < V_{IN} < V_{CC}$
I_{OZ}	Output Leakage Current (except Ports 1, 2, 3, 4)		± 10	μA	$V_{OUT} = 0.45\text{V}$, and 3.0V
I_{CC1}	Operating Current (Note 6)		15	mA	$V_{CC} = 5.5\text{V}$, 16 MHz
I_{CC}	Operating Current (Note 7)		50	mA	$V_{CC} = 5.5\text{V}$, 16 MHz (Note 4)
I_{CCI}	Idle Mode Current		25	mA	$V_{CC} = 5.5\text{V}$, 16 MHz (Note 5)
I_{PD}	Power Down Current		100	μA	$V_{CC} = 2\text{V}$ (Note 3)
RRST	Reset Pulldown Resistor	50	150	$\text{K}\Omega$	
CIO	Pin Capacitance		20	pF	1 MHz, $T_A = 25^\circ\text{C}$ (sampled, not tested on all parts)

NOTES:

- Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OLS} of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.
- Capacitive loading on Ports 0 and 2 may cause the V_{OH} on ALE and PSEN to momentarily fall before the 0.9 V_{CC} specification when the address bits are stabilizing.
- Power DOWN I_{CC} is measured with all output pins disconnected; EA = Port 0 = V_{CC} ; XTAL2 N.C.; RST = V_{SS} ; DB = V_{CC} ; $\overline{WR} = \overline{RD} = \overline{DACK} = \overline{CS} = A0 = A1 = A2 = V_{CC}$. Power Down Mode is not supported on the 87C452P.
- I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, $V_{IL} = V_{SS} + 0.5\text{V}$, $V_{IH} = V_{CC} - 0.5\text{V}$; XTAL2 N.C.; EA = RST = Port 0 = V_{CC} ; $\overline{WR} = \overline{RD} = \overline{DACK} = \overline{CS} = A0 = A1 = A2 = V_{CC}$. I_{CC} would be slightly higher if a crystal oscillator is used.
- Idle I_{CC} is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, $V_{IL} = V_{SS} + 0.5\text{V}$, $V_{IH} = V_{CC} - 0.5\text{V}$; XTAL2 N.C.; Port 0 = V_{CC} ; EA = RST = V_{SS} ; $\overline{WR} = \overline{RD} = \overline{DACK} = \overline{CS} = A0 = A1 = A2 = V_{CC}$.
- 87C452P Piggyback EPROM only.
- 80C452 and 83C452 only.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list of all the characters and what they stand for:

A: Address.

C: Clock.

D: Input data.

H: Logic level HIGH.

I: Instruction (program memory contents).

L: Logic level LOW, or ALE.

P: PSEN.

Q: Output data.

R: READ signal.

T: Time.

V: Valid.

W: WRITE signal.

X: No longer a valid logic level.

Z: Float.

EXAMPLE

TAVLL = Time for Address Valid to ALE Low.

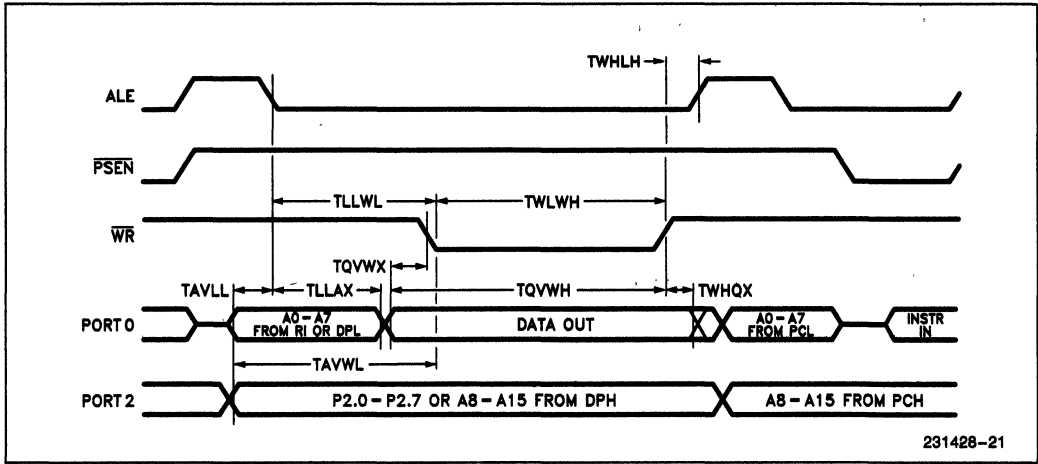
TLLPL = Time for ALE Low to PSEN Low.

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$, Load Capacitance for Port 0, ALE, and PSEN = 100 pF, Load Capacitance for All Other Outputs = 80 pF

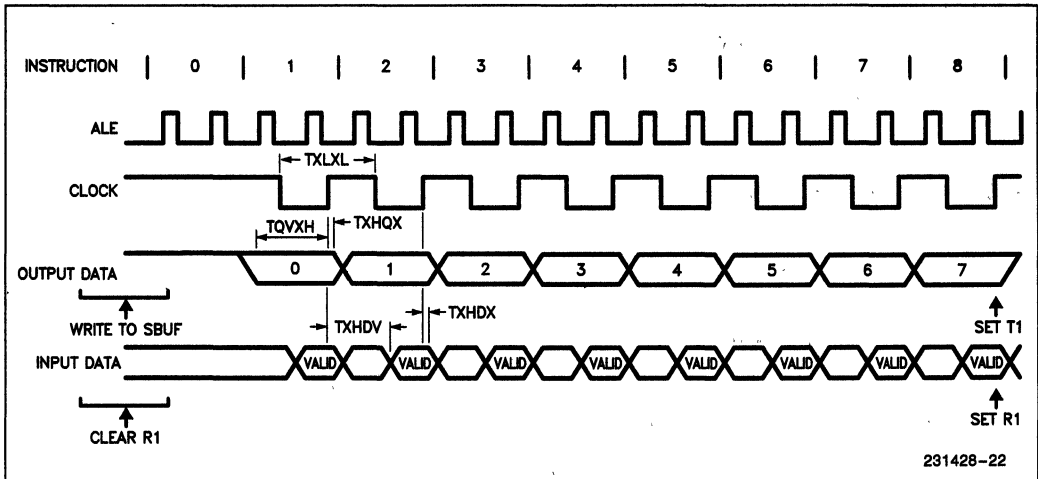
EXTERNAL PROGRAM AND DATA MEMORY CHARACTERISTICS

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency	3.5	16			MHz
TLHLL	ALE Pulse Width	85		2TCLCL - 40		ns
TAVLL	Address Valid to ALE Low	7		TCLCL - 55		ns
TLLAX	Address Hold after ALE Low	28		TCLCL - 35		ns
TLLIV	ALE Low to Valid Instr In		150		4TCLCL - 100	ns
TLLPL	ALE Low to PSEN Low	22		TCLCL - 40		ns
TPLPH	PSEN Pulse Width	142		3TCLCL - 45		ns
TPLIV	PSEN Low to Valid Instr In		82		3TCLCL - 105	ns
TPXIX	Input Instr Hold after PSEN	0		0		ns
TPXIZ	Input Instr Float after PSEN		37		TCLCL - 25	ns
TAVIV	Address to Valid Instr In		207		5TCLCL - 105	ns
TPLAZ	PSEN Low to Address Float		10		10	ns
TRLRH	RD Pulse Width	275		6TCLCL - 100		ns
TWLWH	WR Pulse Width	275		6TCLCL - 100		ns
TRLDV	RD Low to Valid Data In		148		5TCLCL - 165	ns
TRHDX	Data Hold after RD	0		0		ns
TRHDZ	Data Float after RD				2TCLCL - 70	ns
TLLDV	ALE Low to Valid Data In		350		8TCLCL - 150	ns
TAVDV	Address to Valid Data In		398		9TCLCL - 165	ns
TLLWL	ALE Low to RD or WR Low	137	237	3TCLCL - 50	3TCLCL + 50	ns
TAVWL	Address Valid to Read or Write Low	120		4TCLCL - 130		ns
TQVWX	Data Valid to \overline{WR} Transition	2		TCLCL - 60		ns
TWHQX	Data Hold after \overline{WR}	12		TCLCL - 50		ns
TRLAZ	\overline{RD} Low to Address Float		0		0	ns
TWHLH	\overline{RD} or \overline{WR} High to ALE High	23	103	TCLCL - 40	TCLCL + 40	ns
TPXAV	PSEN High to Address Valid	55		TCLCL - 8		ns
TQVWH	Data Valid to \overline{WR} (Setup Time)	288		7TCLCL - 150		ns

EXTERNAL DATA MEMORY WRITE CYCLE



SHIFT REGISTER MODE TIMING WAVEFORMS



EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	16	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

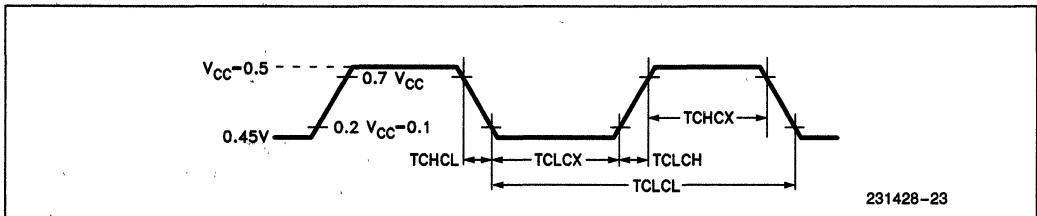
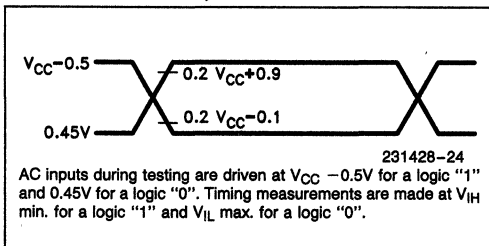
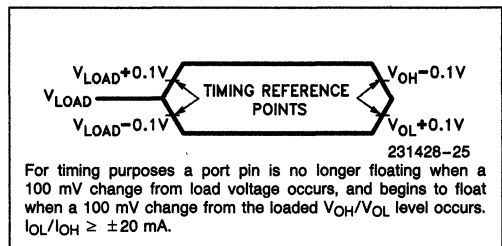
NOTE:

External clock timings are sampled, not tested on all parts.

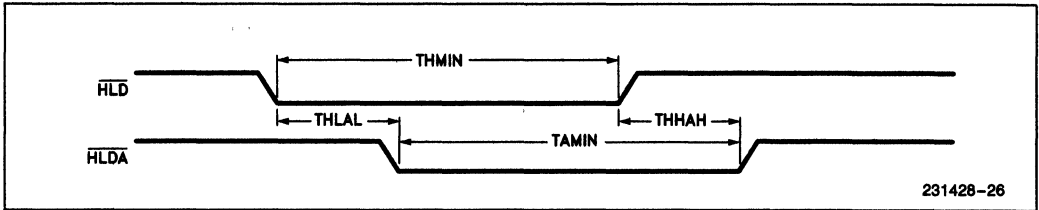
SERIAL PORT TIMING—SHIFT REGISTER MODE

 Test Conditions: $T_A = 0^\circ\text{C}$ to 70°C ; $V_{CC} = 5\text{V} \pm 10\%$; $V_{SS} = 0\text{V}$; Load Capacitance = 80 pF

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	750		12TCLCL		ns
TQVXH	Output Data Setup to Clock Rising Edge	492		10TCLCL - 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	8		2TCLCL - 117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		492		10TCLCL - 133	ns

EXTERNAL CLOCK DRIVE WAVEFORM

AC TESTING INPUT, OUTPUT WAVEFORMS

FLOAT WAVEFORMS


HLD/HLDA WAVEFORMS

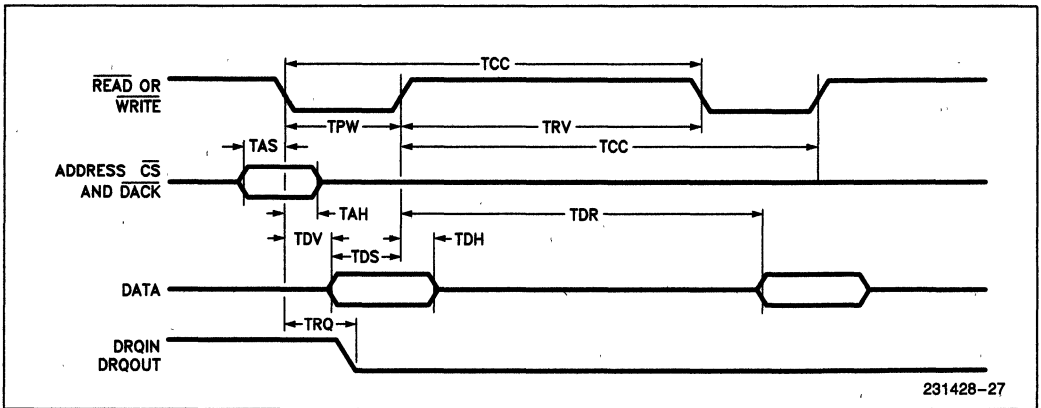


HLD/HLDA TIMINGS

Test Conditions: $T_A = 0^\circ\text{C to } 70^\circ\text{C}$; $V_{CC} = 5V \pm 10\%$, $V_{SS} = 0V$; Load Capacitance = 80 pF

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
THMIN	HLD Pulse Width	350		$4TCLCL + 100$		ns
TAMIN	HLDA Pulse Width	350		$4TCLCL + 100$		ns
THHAH	HLD to HLDA Delay if HLDA is Granted	350		$4TCLCL + 100$		ns
THLAL	HLD to HLDA Delay	350		$4TCLCL + 100$		ns

HOST PORT WAVEFORMS



HOST PORT TIMINGS

Test Conditions: $T_A = 0^\circ\text{C}$ to 70°C ; $V_{CC} = 5V \pm 10\%$; $V_{SS} = 0V$; Load Capacitance = 80 pF

Symbol	Parameter	16 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
TCC	Cycle Time	375		6TCLCL		ns
TPW	Command Pulse Width	100		100		ns
TRV	Recovery Time	60		60		ns
TAS	Address Setup Time	5		5		ns
TAH	Address Hold Time	30		30		ns
TDS	Write Data Setup Time	30		30		ns
TDHW	Write Data Hold Time	0		0		ns
TDHR	Read Data Hold Time	5	40	5	40	ns
TDV	$\overline{\text{READ}}$ Active to Read Data Valid Delay	85		85		ns
TDR	$\overline{\text{WRITE}}$ Inactive to Read Data Valid Delay (Applies only to Host Control SFR)		300		4.8TCLCL	ns
TRQ	$\overline{\text{READ}}$ or $\overline{\text{WRITE}}$ Active to DRQIN or DRQOUT Delay		150		150	ns

PROGRAMMING MODES

Caution: Exceeding 14V on V_{PP} will permanently damage the device.

Initially, all bits of the EPROM are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light exposure (Cerdip EPROMs).

This device is in the programming mode when V_{PP} is raised to its programming voltage and ALE/PGM are both at TTL-low. The data to be programmed is applied 8 bits in parallel to the Port 0 pins. The levels required for the address and data inputs are TTL. The address is applied to Port 1 and 2.

Program Verify

A verify should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with $\overline{\text{OE}}$ at V_{IL} , PGM at V_{IH} and V_{PP} and V_{CC} at their programming voltages.

intelligent Identifier™ Mode

intelligent Identifier™ Mode is not supported on the 87C452P piggyback EPROM device.

ERASURE CHARACTERISTICS

The erasure characteristics are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase the EPROM in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the EPROM is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the window to prevent unintentional erasure.

The recommended erasure procedure is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity \times exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 $\mu\text{W}/\text{cm}^2$ power rating. The EPROM should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose an EPROM can be exposed to without damage is 7258 Wsec/cm² (1 week @ 12000 $\mu\text{W}/\text{cm}^2$). Exposure of the EPROM to high intensity UV light for longer periods may cause permanent damage.

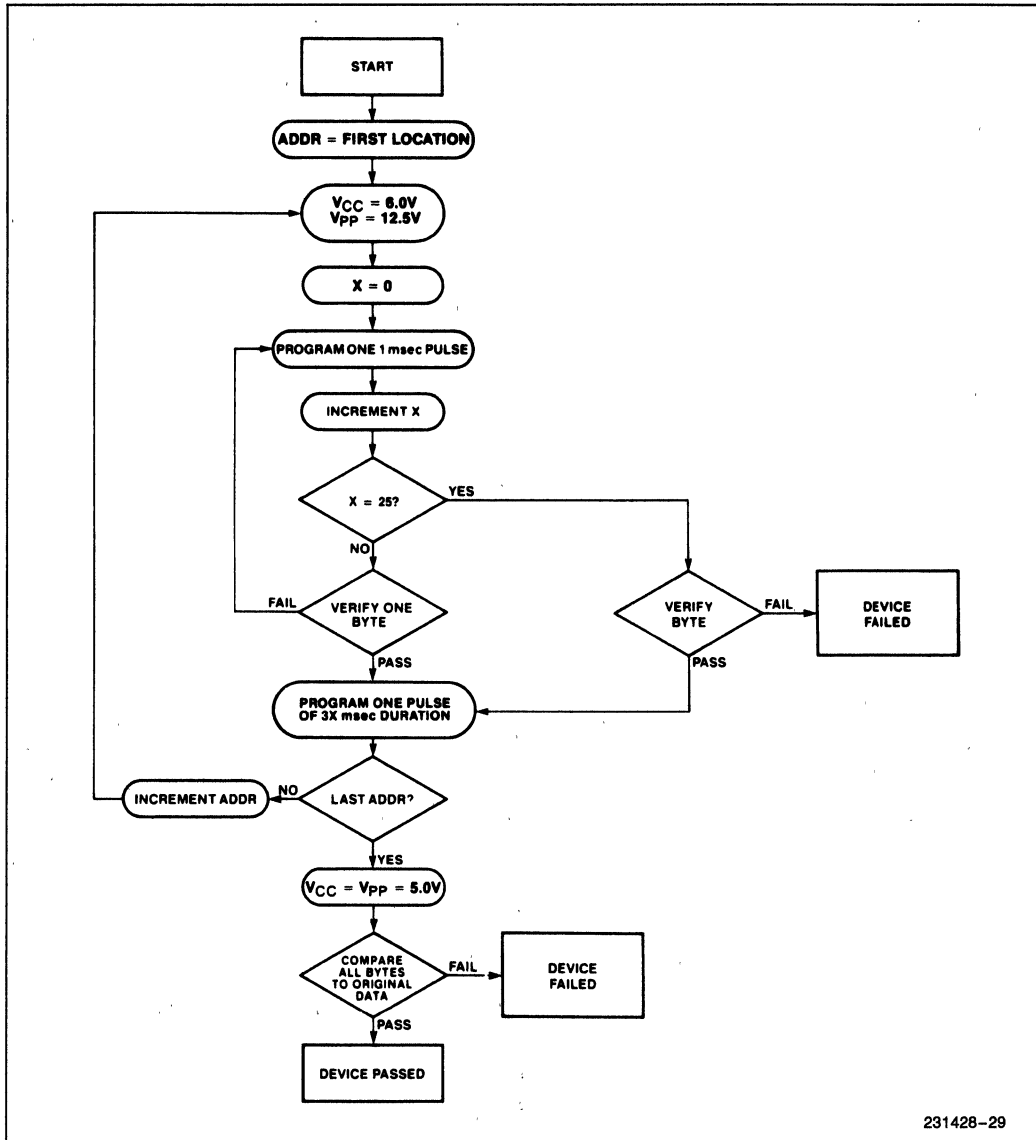


Figure 4. Intelligent Programming™ Flowchart

Intelligent Programming™ Algorithm

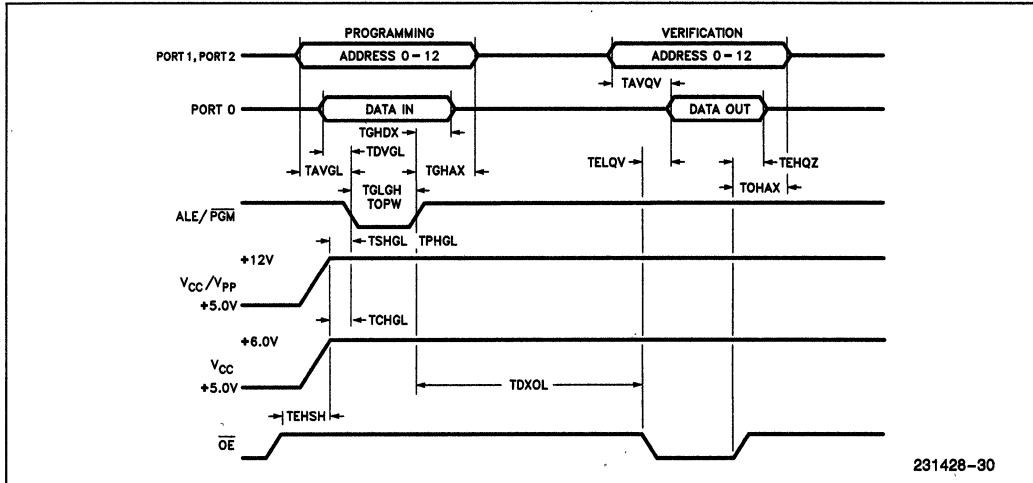
The Intelligent Programming Algorithm, a standard in the industry for the past few years, is required for all of Intel's 12.5V DERDEP EPROMs. Plastic and PLCC EPROMs may also be programmed using this method. A flowchart of the Intelligent Programming Algorithm is shown in Figure 4.

duration of the initial \overline{PGM} pulse(s) is one millisecond, which will then be followed by a longer overprogram pulse of length $3X$ ms. X is an iteration counter and is equal to the number of the initial one millisecond pulses applied to a particular location, before a correct verify occurs. Up to 25 one-millisecond pulses per byte are provided for before the overprogram pulse is applied.

The Intelligent Programming Algorithm utilizes two different pulse types: initial and overprogram. The

The entire sequence of program pulses and byte verifications is performed at $V_{CC} = 6.0V$ and $V_{PP} = 12.5V$. When the intelligent Programming cycle has been completed, all bytes should be compared to the original data with $V_{CC} = V_{PP} = 5.0V$.

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



231428-30

A.C. PROGRAMMING CHARACTERISTICS

$T_A = 25^\circ C \pm 5^\circ C$ (see EPROM PROGRAMMING D.C. CHARACTERISTICS for V_{CC} and V_{PP} Voltages)

Symbol	Parameter	12 MHz OSC			Test Conditions*(1)
		Min	Max	Unit	
TAVGL	Address Setup to \overline{PGM}		2.9	μs	
TDXOL	\overline{OE} Setup from Data Float	2.0		μs	
TDVGL	Data Setup to \overline{PGM}	3.8		μs	
TOHAX	Address Hold after \overline{OE}	0		μs	
TGHDX	Data Hold after \overline{PGM}	2.0		ns	
TEHQZ	Data Float after \overline{OE}	0	1.6	μs	(Note 3)
TOHAX	Address Hold after \overline{PGM}	0		μs	
TPHGL	V_{PP} Setup to \overline{PGM}	2.0		μs	
TCHGL	V_{CC} Setup to \overline{PGM}	2.0		μs	
TEHSH	\overline{OE} Setup to V_{PP} and V_{CC}	2.0		μs	
TGLGH	\overline{PGM} Pulse Width	0.95	1.05	ms	intelligent Programming
TAVQV	Address to Data Valid		3.0	μs	
TOPW	\overline{PGM} Overprogram Pulse Width	2.85	78.75	ms	(Note 2)
TELQV	Data Valid from \overline{OE}		2.0	μs	

NOTES:

- V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP} .
- The length of the overprogram pulse may vary from 2.85 ms to 78.75 ms as a function of the iteration counter value X (intelligent Programming Algorithm only).
- This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.
- The maximum current value is with Port 0 unloaded.

D.C. PROGRAMMING CHARACTERISTICS $T_A = 25^\circ\text{C} \pm 5\%$

Symbol	Parameter	Limits			Test Conditions*(1)
		Min	Max	Unit	
$I_{CC2}^{(4)}$	V_{CC} Supply Current (Program and Verify)		150	mA	
$I_{PP2}^{(4)}$	V_{PP} Supply Current (Program)		50	mA	$\overline{CE} = V_{IL}$
V_{ID}	A_9 Intelligent Identifier Voltage	11.5	12.5	V	
V_{PP}	intelligent Programming Algorithm	12.0	13.0	V	$\overline{CE} = \overline{PGM} = V_{IL}$
V_{CC}	intelligent Programming Algorithm	5.75	6.25	V	

UPI™-41, 42: 8041AH/8042AH/8741AH/8742AH UNIVERSAL PERIPHERAL INTERFACE 8-BIT SLAVE MICROCONTROLLER

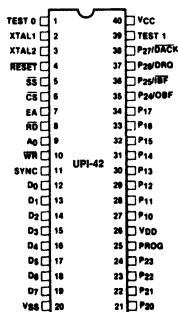
- UPI-41: 6 MHz; UPI-42: 12 MHz
- Pin, Software and Architecturally Compatible with all UPI-41 and UPI-42 Products
- 8-Bit CPU plus ROM/EPROM, RAM, I/O, Timer/Counter and Clock in a Single Package
- 2048 x 8 ROM/EPROM, 256 x 8 RAM on UPI-42, 1024 x 8 ROM/EPROM, 128 x 8 RAM on UPI-41, 8-Bit Timer/Counter, 18 Programmable I/O Pins
- One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface
- DMA, Interrupt, or Polled Operation Supported
- Fully Compatible with all Intel and Most Other Microprocessor Families
- Interchangeable ROM and EPROM Versions
- Expandable I/O
- Sync Mode Available
- Over 90 Instructions: 70% Single Byte
- Available in EXPRESS — Standard Temperature Range
- intelligent Programming™ Algorithm — Fast EPROM Programming
- Available in 40-Lead Cerdip, 40-Lead Plastic and 44-Lead Plastic Leaded Chip Carrier Packages
(See Packaging Spec., Order #231369)

The Intel UPI-41 and UPI-42 are general-purpose Universal Peripheral Interfaces that allow the designer to develop customized solutions for peripheral device control.

They are essentially "slave" microcontrollers, or microcontrollers with a slave interface included on the chip. Interface registers are included to enable the UPI device to function as a slave peripheral controller in the MCST™ Modules and iAPX family, as well as other 8-, 16-bit systems.

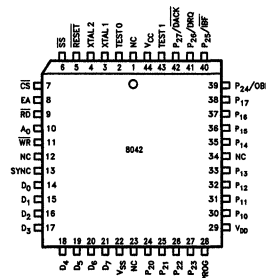
To allow full user flexibility, the program memory is available in ROM, One-Time Programmable EPROM (OTP) and UV-erasable EPROM. All UPI-41 and UPI-42 devices are fully pin compatible for easy transition from prototype to production level designs. These are the memory configurations available.

UPI Device	ROM	EPROM	RAM	Programming Voltage
8042AH	2K	—	256	—
8742AH	—	2K	256	12.5V
8041AH	1K	—	128	—
8741AH	—	1K	128	12.5V



210393-2

Figure 1. DIP Pin Configuration



210393-3

Figure 2. PLCC Pin Configuration

Table 1. Pin Description

Symbol	DIP Pin No.	PLCC Pin No.	Type	Name and Function
TEST 0, TEST 1	1 39	2 43	I	TEST INPUTS: Input pins which can be directly tested using conditional branch instructions. FREQUENCY REFERENCE: TEST 1 (T_1) also functions as the event timer input (under software control). TEST 0 (T_0) is used during PROM programming and ROM/EPROM verification. It is also used during Sync Mode to reset the instruction state to S1 and synchronize the internal clock to PH1. See the Sync Mode Section.
XTAL 1, XTAL 2	2 3	3 4	I	INPUTS: Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
RESET	4	5	I	RESET: Input used to reset status flip-flops and to set the program counter to zero. RESET is also used during EPROM programming and verification.
SS	5	6	I	SINGLE STEP: Single step input used in conjunction with the SYNC output to step the program through each instruction (EPROM). This should be tied to +5V when not used. This pin is also used to put the device in Sync Mode by applying 12.5V to it.
CS	6	7	I	CHIP SELECT: Chip select input used to select one UPI microcomputer out of several connected to a common data bus.
EA	7	8	I	EXTERNAL ACCESS: External access input which allows emulation, testing and ROM/EPROM verification. This pin should be tied low if unused.
RD	8	9	I	READ: I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
A ₀	9	10	I	COMMAND/DATA SELECT: Address Input used by the master processor to indicate whether byte transfer is data ($A_0 = 0$, F1 is reset) or command ($A_0 = 1$, F1 is set). $A_0 = 0$ during program and verify operations.
WR	10	11	I	WRITE: I/O write input which enables the master CPU to write data and command words to the UPI INPUT DATA BUS BUFFER.
SYNC	11	13	O	OUTPUT CLOCK: Output signal which occurs once per UPI instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
D ₀ -D ₇ (BUS)	12-19	14-21	I/O	DATA BUS: Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI microcomputer to an 8-bit master system data bus.
P ₁₀ -P ₁₇	27-34	30-33 35-38	I/O	PORT 1: 8-bit, PORT 1 quasi-bidirectional I/O lines. P ₁₀ -P ₁₇ access the signature row and security bit.
P ₂₀ -P ₂₇	21-24 35-38	24-27 39-42	I/O	PORT 2: 8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P ₂₀ -P ₂₃) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits (P ₂₄ -P ₂₇) can be programmed to provide interrupt Request and DMA Handshake capability. Software control can configure P ₂₄ as Output Buffer Full (OBF) interrupt, P ₂₅ as Input Buffer Full (IBF) interrupt, P ₂₆ as DMA Request (DRQ), and P ₂₇ as DMA Acknowledge (DACK).
PROG	25	28	I/O	PROGRAM: Multifunction pin used as the program pulse input during PROM programming. During I/O expander access the PROG pin acts as an address/data strobe to the 8243. This pin should be tied high if unused.
V _{CC}	40	44		POWER: +5V main power supply pin.
V _{DD}	26	29		POWER: +5V during normal operation. +12.5V during programming operation. Low power standby supply pin.
V _{SS}	20	22		GROUND: Circuit ground potential.

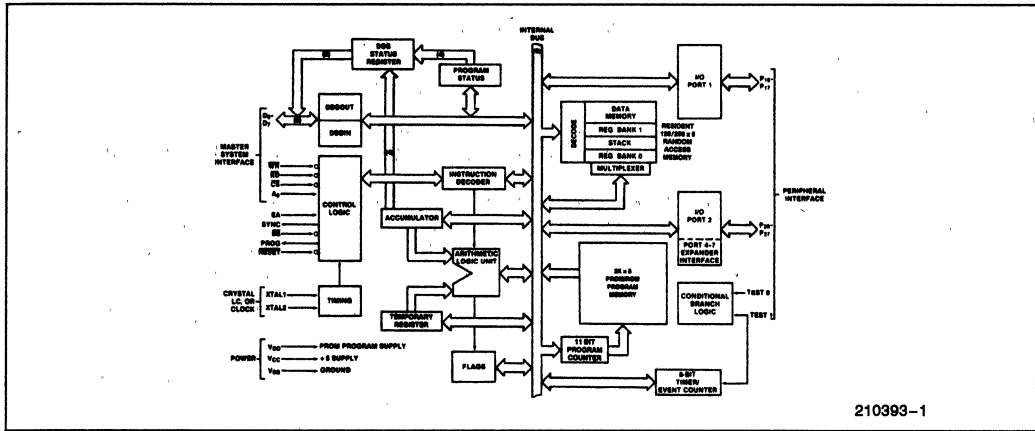
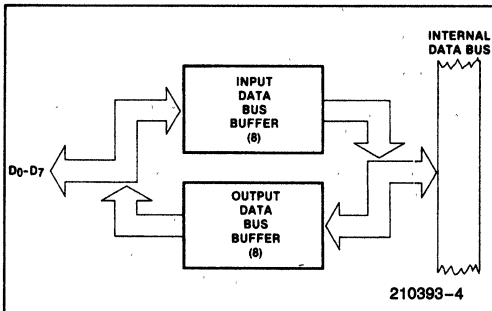


Figure 3. Block Diagram

UPI-41 and UPI-42 FEATURES

1. Two Data Bus Buffers, one for input and one for output. This allows a much cleaner Master/Slave protocol.



2. 8 Bits of Status

ST ₇	ST ₆	ST ₅	ST ₄	F ₁	F ₀	IBF	OBF
-----------------	-----------------	-----------------	-----------------	----------------	----------------	-----	-----

D₇ D₆ D₅ D₄ D₃ D₂ D₁ D₀

ST₄-ST₇ are user definable status bits. These bits are defined by the "MOV STS, A" single byte, single cycle instruction. Bits 4-7 of the accumulator are moved to bits 4-7 of the status register. Bits 0-3 of the status register are not affected.

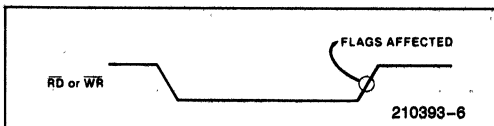
MOV STS, A Op Code: 90H

1	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---

D₇

D₀

3. RD and WR are edge triggered. IBF, OBF, F₁ and INT change internally after the trailing edge of RD or WR.

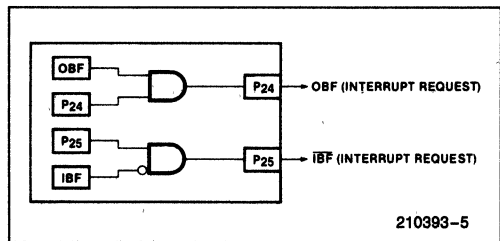


During the time that the host CPU is reading the status register, the UPI is prevented from updating this register or is 'locked out.'

4. P₂₄ and P₂₅ are port pins or Buffer Flag pins which can be used to interrupt a master processor. These pins default to port pins on Reset.

If the "EN FLAGS" instruction has been executed, P₂₄ becomes the OBF (Output Buffer Full) pin. A "1" written to P₂₄ enables the OBF pin (the pin outputs the OBF Status Bit). A "0" written to P₂₄ disables the OBF pin (the pin remains low). This pin can be used to indicate that valid data is available from the UPI (in Output Data Bus Buffer).

If "EN FLAGS" has been executed, P₂₅ becomes the IBF (Input Buffer Full) pin. A "1" written to P₂₅ enables the IBF pin (the pin outputs the inverse of the IBF Status Bit). A "0" written to P₂₅ disables the IBF pin (the pin remains low). This pin can be used to indicate that the UPI is ready for data.



Data Bus Buffer Interrupt Capability

EN FLAGS Op Code: 0F5H

1	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

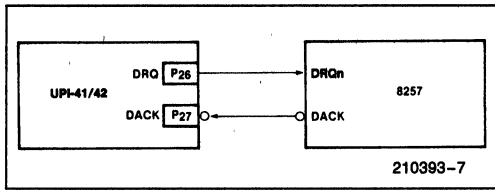
D₇

D₀

5. P₂₆ and P₂₇ are port pins or DMA handshake pins for use with a DMA controller. These pins default to port pins on Reset.

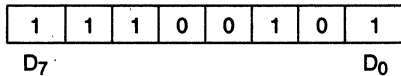
If the "EN DMA" instruction has been executed, P₂₆ becomes the DRQ (DMA Request) pin. A "1" written to P₂₆ causes a DMA request (DRQ is activated). DRQ is deactivated by DACK•RD, DACK•WR, or execution of the "EN DMA" instruction.

If "EN DMA" has been executed, P₂₇ becomes the DACK (DMA Acknowledge) pin. This pin acts as a chip select input for the Data Bus Buffer registers during DMA transfers.



DMA Handshake Capability

EN DMA Op Code: 0E5H



6. When EA is enabled on the UPI, the program counter is placed on Port 1 and the lower three bits of Port 2 (MSB = P₂₂, LSB = P₁₀). On the UPI this information is multiplexed with PORT DATA (see port timing diagrams at end of this data sheet).

7. The 8741AH and 8742AH support the intelligent Programming Algorithm. (See the Programming Section.)

APPLICATIONS

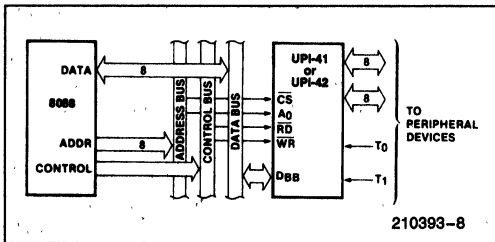


Figure 3. 8088-UPI-41/42 interface

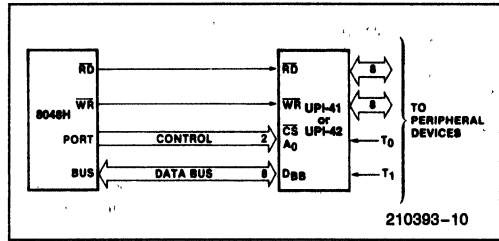


Figure 4. 8048H-UPI-41/42 Interface

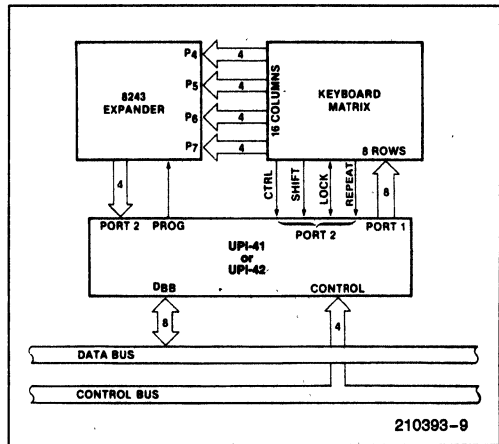


Figure 5. UPI-41/42-8243 Keyboard Scanner

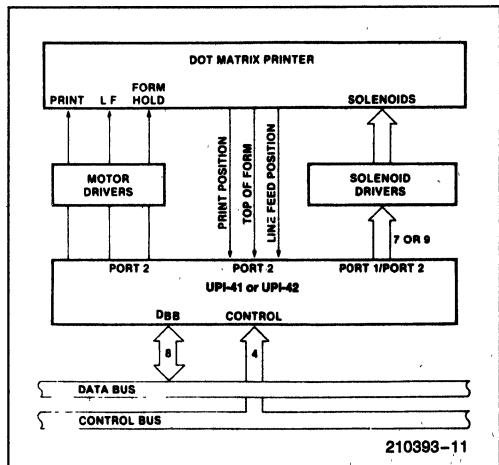


Figure 6. UPI-41/42 80-Column Matrix Printer Interface

PROGRAMMING, VERIFYING, AND ERASING THE 8741AH and 8742AH EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	2 Clock Inputs
Reset	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Signature Row/Security Bit Modes
BUS	Address and Data Input Data Output During Verify
P ₂₀₋₂₂	Address Input
V _{DD}	Programming Power Supply
PROG	Program Pulse Input

WARNING

An attempt to program a missocketed 8741AH or 8742AH will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1. CS = 5V, V_{CC} = 5V, V_{DD} = 5V, RESET = 0V, A₀ = 0V, TEST 0 = 5V, clock applied or internal oscillator operating, BUS floating, PROG = 5V.
2. Insert 8741AH or 8742AH in programming socket
3. TEST 0 = 0V (select program mode)
4. EA = 12.5V (active program mode)
5. V_{CC} = 6V (programming supply)
6. V_{DD} = 12.5V (programming power)
7. Address applied to BUS and P₂₀₋₂₂
8. RESET = 5V (latch address)
9. Data applied to BUS
10. PROG = 5V followed by one 1 ms pulse to 0V
11. TEST 0 = 5V (verify mode)
12. Read and verify data on BUS
13. TEST 0 = 0V
14. Apply overprogram pulse
15. RESET = 0V and repeat from step 6
16. Programmer should be at conditions of step 1 when 8741AH or 8742AH is removed from socket

Please follow the intelligent Programming flow chart for proper programming procedure.

8741AH/8742AH Erasure Characteristics

The erasure characteristics of the 8741AH/8742AH are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8741AH/8742AH in approximately 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 8741AH/8742AH is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8741AH/8742AH window to prevent unintentional erasure.

The recommended erasure procedure for the 8741AH/8742AH is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 15 w-sec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm² power rating. The 8741AH/8742AH should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. Exposure of the 8741AH/8742AH to high intensity UV light for long periods may cause permanent damage.

intelligent Programming™ Algorithm

The intelligent Programming Algorithm rapidly programs Intel 8741AH/8742AH EPROMs using an efficient and reliable method particularly suited to the production programming environment. Typical programming time for individual devices is on the order of 10 seconds. Programming reliability is also ensured as the incremental program margin of each byte is continually monitored to determine when it has been successfully programmed. A flowchart of the 8741AH/8742AH intelligent Programming Algorithm is shown in Figure 7.

The intelligent Programming Algorithm utilizes two different pulse types: initial and overprogram. The duration of the initial PROG pulse(s) is one millisecond, which will then be followed by a longer overprogram pulse of length 3X msec. X is an iteration counter and is equal to the number of the initial one millisecond pulses applied to a particular 8741AH/8742AH location, before a correct verify occurs. Up to 25 one-millisecond pulses per byte are provided for before the overprogram pulse is applied.

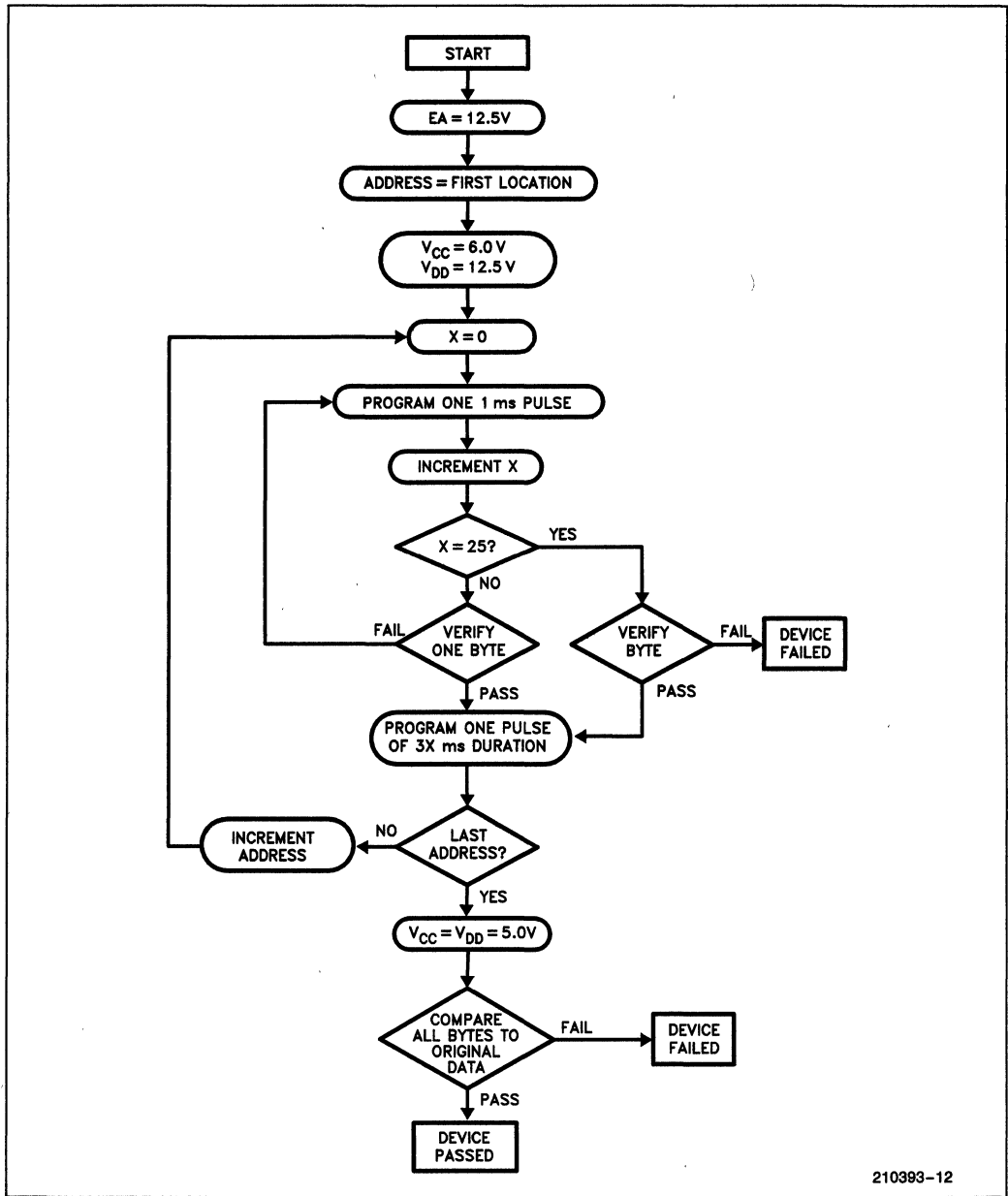


Figure 7

210393-12

The entire sequence of program pulses and byte verifications is performed at $V_{CC} = 6.0V$ and $V_{DD} = 12.5V$. When the intelligent Programming cycle has been completed, all bytes should be compared to the original data with $V_{CC} = 5.0$, $V_{DD} = 5V$.

Verify

A verify should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with $T_0 = 5V$, $V_{DD} = 5V$, $EA = 12.5V$, $SS = 5V$, $PROG = 5V$, $A_0 = 0V$, and $\overline{CS} = 5V$.

SECURITY BIT

The security bit is a single EPROM cell outside the EPROM array. The user can program this bit with the appropriate access code and the normal programming procedure, to inhibit any external access to the EPROM contents. Thus the user's resident program is protected. There is no direct external access to this bit. However, the security byte in the signature mode has the same address and can be used to check indirectly whether the security bit has been programmed or not. The security bit has no effect on the signature mode, so the security byte can always be examined.

SECURITY BIT PROGRAMMING/ VERIFICATION

Programming

- a. Read the security byte of the signature mode. Make sure it is 00H.

- b. Apply access code to appropriate inputs to put the device into security mode.
- c. Apply high voltage to EA and V_{DD} pins.
- d. Follow the programming procedure as per the intelligent Programming Algorithm with known data on the databus. Not only the security bit, but also the security byte of the signature mode is programmed.
- e. Verify that the security byte of the signature mode contains the same data as appeared on the data bus. (If $DB_0-DB_7 = \text{high}$, the security byte will contain FFH.)
- f. Read two consecutive known bytes from the EPROM array and verify that the wrong data are retrieved in at least one verification. If the EPROM can still be read, the security bit may have not been fully programmed though the security byte in the signature mode has.

Verification

Since the security bit address overlaps the address of the security byte of the signature mode, it can be used to check indirectly whether the security bit has been programmed or not. Therefore, the security bit verification is a mere read operation of the security byte of the signature row (1 = security bit programmed; 0 = security bit unprogrammed). Note that during the security bit programming, the reading of the security byte does not necessarily indicate that the security bit has been successfully programmed. Thus, it is recommended that two consecutive known bytes in the EPROM array be read and the wrong data should be read at least once, because it is highly improbable that random data coincides with the correct ones twice.

SIGNATURE MODE

The UPI-41/42 has an additional 32 bytes of EPROM available for Intel and user signatures and miscellaneous purposes. The 32 bytes are partitioned as follows:

- A. **Test code/checksum**—This can accommodate up to 25 bytes of code for testing the internal nodes that are not testable by executing from the external memory. The test code/checksum is present on ROMs, and OTPs.
 - B. **Intel signature**—This allows the programmer to read from the UPI-41/42 the manufacturer of the device and the exact product name. It facilitates automatic device identification and will be present in the ROM and OTP versions. Location 10H contains the manufacturer code. For Intel, it is 89H. Location 11H contains the device code.
- The code is 43H and 42H for the 8042AH and OTP 8742AH, and 41H and 40H for the 8041AH and OTP 8741AH, respectively.
- C. **User signature**—The user signature memory is implemented in the EPROM and consists of 2 bytes for the customer to program his own signature code (for identification purposes and quick sorting of previously programmed materials).
 - D. **Test signature**—This memory is used to store testing information such as: test data, bin number, etc. (for use in quality and manufacturing control).
 - E. **Security byte**—This byte is used to check whether the security bit has been programmed (see the security bit section).

The signature mode can be accessed by setting P10 = 0, P11–P17 = 1, and then following the programming and/or verification procedures. The location of the various address partitions are as follows:

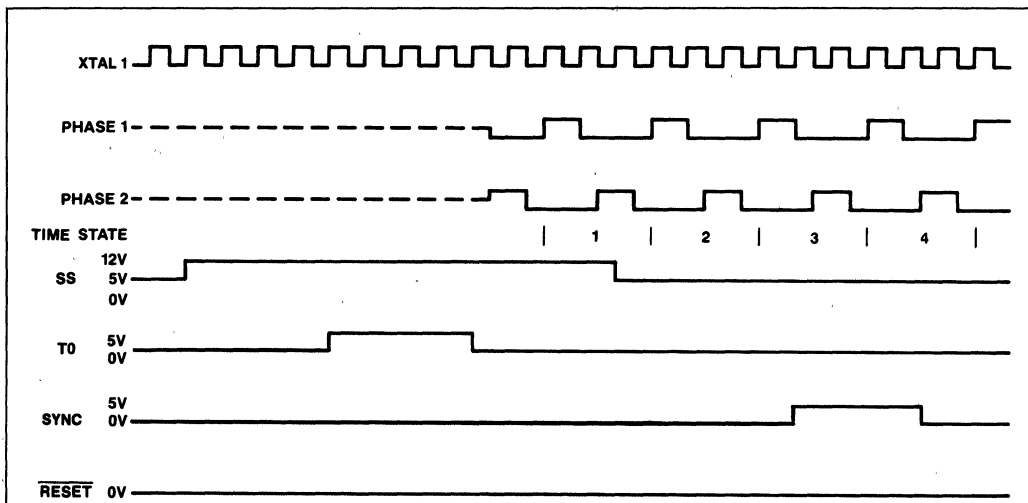
	Address		Device Type	No. of Bytes
Test Code/Checksum	0 16H	0FH 1EH	ROM/OTP	25
Intel Signature	10H	11H	ROM/OTP	2
User Signature	12H	13H	EPROM/OTP	2
Test Signature	14H	15H	ROM/OTP	2
Security Byte	1FH		EPROM/OTP	1

SYNC MODE

The UPI-41/42 has incorporated a new mode. The Sync Mode is provided to ease the design of multiple controller circuits by allowing the designer to force the device into known phase and state time. The Sync Mode may also be utilized by automatic test equipment (ATE) for quick, easy, and efficient synchronizing between the tester and the DUT (device under test).

Sync Mode is enabled when \overline{SS} pin is raised to high voltage level of +12 volts. To begin synchronization, T0 is raised to 5 volts at least four clock cycles after \overline{SS} . T0 must be high for at least four X1 clock cycles to fully reset the prescaler and time state generators. T0 may then be brought down during low state of X1. Two clock cycles later, with the rising edge of X1, the device enters into Time State 1, Phase 1. \overline{SS} is then brought down to 5 volts 4 clocks later after T0. RESET is allowed to go high 5 tCY (75 clocks) later for normal execution of code.

SYNC MODE TIMING DIAGRAMS



210393-28

Minimum Specifications

SYNC Operation Time, t_{SYNC} = 3.5 XTAL 1 Clock cycles. Reset Time, t_{RS} = 4 tCY.

NOTE:

The rising and falling edges of T0 should occur during low state of XTAL1 clock.

ACCESS CODE

The following table summarizes the access codes required to invoke the Sync Mode, Signature Mode, and the Security Bit, respectively. Also, the programming and verification modes are included for comparison.

Modes	Control Signals							Data Bus							Access Code												
	TO	RST	SS	EA	PROG	V _{DD}	V _{CC}	0	1	2	3	4	5	6	7	Port 2			Port 1								
																0	1	2	0	1	2	3	4	5	6	7	
Programming Mode	0	0	1	HV	1	V _{DDH}	V _{CC}	Address							Addr			a ₀	a ₁	X	X	X	X	X	X	X	
	0	1	1	HV	STB	V _{DDH}	V _{CC}	Data In							Addr												
Verification Mode	0	0	1	HV	1	V _{CC}	V _{CC}	Address							Addr			a ₀	a ₁	X	X	X	X	X	X	X	
	1	1	1	HV	1	V _{CC}	V _{CC}	Data Out							Addr												
Sync Mode	STB High	0	HV	0	X	V _{CC}	V _{CC}	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Signature Mode	Prog	0	0	1	HV	1	V _{DDH}	V _{CC}	Addr. (see Sig Mode Table)							0 0 0			0	1	1	1	1	X	X	1	
		0	1	1	HV	STB	V _{DDH}	V _{CC}	Data In							0 0 0											
	Verify	0	0	1	HV	1	V _{CC}	V _{CC}	Addr. (see Sig Mode Table)							0 0 0											
		1	1	1	HV	1	V _{CC}	V _{CC}	Data Out							0 0 0											
Security Bit/Byte	Prog	0	0	1	HV	1	V _{DDH}	V _{CC}	Address							0 0 0											
		0	1	1	HV	STB	V _{DDH}	V _{CC}	Data In							0 0 0											
	Verify	0	0	1	HV	1	V _{CC}	V _{CC}	Address							0 0 0											
		1	1	1	HV	1	V _{CC}	V _{CC}	Data Out							0 0 0											

NOTES:

1. a₀ = 0 or 1; a₁ = 0 or 1. a₀ must = a₁.

ABSOLUTE MAXIMUM RATINGS*

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

Storage Temperature -65°C to +150°C

Voltage on Any Pin with

Respect to Ground -0.5V to +7V

Power Dissipation 1.5 W

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

NOTICE: Specifications contained within the following tables are subject to change.

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

Symbol	Parameter	8042/8742		Units	Notes
		Min	Max		
V _{IL}	Input Low Voltage (Except XTAL1, XTAL2, RESET)	-0.5	0.8	V	
V _{IL1}	Input Low Voltage (XTAL1, XTAL2, RESET)	-0.5	0.6	V	
V _{IH}	Input High Voltage (Except XTAL1, XTAL2, RESET)	2.0	V _{CC}	V	
V _{IH1}	Input High Voltage (XTAL1, RESET)	3.5	V _{CC}	V	
V _{IH2}	Input High Voltage (XTAL2)	2.2	V _{CC}	V	
V _{OL}	Output Low Voltage (D ₀ -D ₇)		0.45	V	I _{OL} = 2.0 mA

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}$, $V_{CC} = V_{DD} = +5\text{V} \pm 10\%$ (Continued)

Symbol	Parameter	8042/8742		Units	Notes
		Min	Max		
V_{OL1}	Output Low Voltage ($P_{10}P_{17}$, $P_{20}P_{27}$, Sync)		0.45	V	$I_{OL} = 1.6\text{ mA}$
V_{OL2}	Output Low Voltage (PROG)		0.45	V	$I_{OL} = 1.0\text{ mA}$
V_{OH}	Output High Voltage (D_0 – D_7)	2.4		V	$I_{OH} = -400\ \mu\text{A}$
V_{OH1}	Output High Voltage (All Other Outputs)	2.4			$I_{OH} = -50\ \mu\text{A}$
I_{IL}	Input Leakage Current (T_0 , T_1 , RD, WR, CS, A_0 , EA)		± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OFL}	Output Leakage Current (D_0 – D_7 , High Z State)		± 10	μA	$V_{SS} + 0.45 \leq V_{OUT} \leq V_{CC}$
I_{LI}	Low Input Load Current ($P_{10}P_{17}$, $P_{20}P_{27}$)		0.3	mA	$V_{IL} = 0.8\text{V}$
I_{LI1}	Low Input Load Current (RESET, SS)		0.2	mA	$V_{IL} = 0.8\text{V}$
I_{DD}	V_{DD} Supply Current		20	mA	Typical = 8 mA
$I_{CC} + I_{DD}$	Total Supply Current		135	mA	Typical = 80 mA
I_{DD} Standby	Power Down Supply Current		20	mA	Typical = 8 mA
I_{IH}	Input Leakage Current (P_{10} – P_{17} , P_{20} – P_{27})		100	μA	$V_{IN} = V_{CC}$
C_{IN}	Input Capacitance		10	pF	
C_{IO}	I/O Capacitance		20	pF	

D.C. CHARACTERISTICS—PROGRAMMING
 $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$, $V_{CC} = 6\text{V} \pm 0.25\text{V}$, $V_{DD} = 12.5\text{V} \pm 0.5\text{V}$

Symbol	Parameter	Min	Max	Units
V_{DDH}	V_{DD} Program Voltage High Level	12	13	V(1)
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V
V_{PH}	PROG Program Voltage High Level	2.0	5.5	V
V_{PL}	PROG Voltage Low Level	-0.5	0.8	V
V_{EAH}	Input High Voltage for EA	12.0	13.0	V(2)
V_{EAL}	EA Voltage Low Level	-0.5	5.25	V
I_{DD}	V_{DD} High Voltage Supply Current		30.0	mA
I_{EA}	EA High Voltage Supply Current		1.0	mA

NOTES:

1. Voltages over 13V applied to pin V_{DD} will permanently damage the device.
2. V_{EAH} must be applied to EA before V_{DDH} and removed after V_{DDL} .
3. V_{CC} must be applied simultaneously or before V_{DD} and must be removed simultaneously or after V_{DD} .

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$, $V_{SS} = 0\text{V}$, $V_{CC} = V_{DD} = +5\text{V} \pm 10\%$
DBB READ

Symbol	Parameter	Min	Max	Units
t_{AR}	CS, A_0 Setup to RD \downarrow	0		ns
t_{RA}	CS, A_0 Hold After RD \uparrow	0		ns
t_{RR}	RD Pulse Width	160		ns
t_{AD}	CS, A_0 to Data Out Delay		130	ns
t_{RD}	RD \downarrow to Data Out Delay	0	130	ns
t_{DF}	RD \uparrow to Data Float Delay		85	ns

DBB WRITE

Symbol	Parameter	Min	Max	Units
t_{AW}	CS, A_0 Setup to WR \downarrow	0		ns
t_{WA}	CS, A_0 Hold After WR \uparrow	0		ns
t_{WW}	WR Pulse Width	160		ns
t_{DW}	Data Setup to WR \uparrow	130		ns
t_{WD}	Data Hold After WR \uparrow	0		ns

CLOCK

Symbol	Parameter	Min	Max	Units
t_{CY} (UPI-41)	Cycle Time	2.5	9.20	$\mu\text{s}^{(1)}$
t_{CYC} (UPI-41)	Clock Period	167	613	ns
t_{CY} (UPI-42)	Cycle Time	1.25	9.20	$\mu\text{s}^{(1)}$
t_{CYC} (UPI-42)	Clock Period	83.3	613	ns
t_{PWH}	Clock High Time	33		ns
t_{PWL}	Clock Low Time	33		ns
t_R	Clock Rise Time		10	ns
t_F	Clock Fall Time		10	ns

NOTE:

- $t_{CY} = 15/f(\text{XTAL})$

A.C. CHARACTERISTICS DMA

Symbol	Parameter	Min	Max	Units
t_{ACC}	DACK to WR or RD	0		ns
t_{CAC}	RD or WR to DACK	0		ns
t_{ACD}	DACK to Data Valid	0	130	ns
t_{CRQ}	RD or WR to DRQ Cleared		110	ns ⁽¹⁾

NOTE:

- $C_L = 150\text{ pF}$.

A.C. CHARACTERISTICS—PROGRAMMING
 $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$, $V_{CC} = 6\text{V} \pm 0.25\text{V}$, $V_{DDL} = +5\text{V} \pm 0.25\text{V}$, $V_{DDH} = 12.5\text{V} \pm 0.5\text{V}$
(8741AH/8742AH ONLY)

Symbol	Parameter	Min	Max	Units
t_{AW}	Address Setup Time to RESET \uparrow	$4t_{CY}$		
t_{WA}	Address Hold Time After RESET \uparrow	$4t_{CY}$		
t_{DW}	Data in Setup Time to PROG \downarrow	$4t_{CY}$		
t_{WD}	Data in Hold Time After PROG \uparrow	$4t_{CY}$		
t_{PW}	Initial Program Pulse Width	0.95	1.05	ms ⁽¹⁾
t_{TW}	Test 0 Setup Time for Program Mode	$4t_{CY}$		
t_{WT}	Test 0 Hold Time After Program Mode	$4t_{CY}$		
t_{DO}	Test 0 to Data Out Delay		$4t_{CY}$	
t_{WW}	RESET Pulse Width to Latch Address	$4t_{CY}$		
t_r, t_f	PROG Rise and Fall Times	0.5	100	μs
t_{CY}	CPU Operation Cycle Time	2.5	3.75	μs
t_{RE}	RESET Setup Time Before EA \uparrow	$4t_{CY}$		
t_{OPW}	Overprogram Pulse Width	2.85	78.75	ms ⁽²⁾
t_{DE}	EA High to V_{DD} High	$1t_{CY}$		

NOTES:

1. Typical Initial Program Pulse width tolerance = 1 ms \pm 5%.
2. This variation is a function of the iteration counter value, X.
3. If TEST 0 is high, t_{DO} can be triggered by RESET \uparrow .

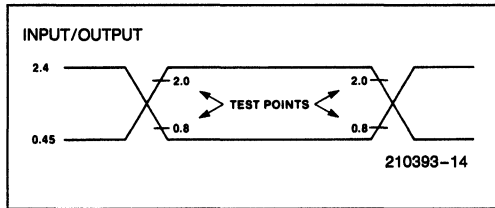
A.C. CHARACTERISTICS PORT 2 $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 10\%$

Symbol	Parameter	$f(t_{CY})^{(3)}$	Min	Max	Units
t_{CP}	Port Control Setup Before Falling Edge of PROG	$1/15 t_{CY} - 28$	55		ns ⁽¹⁾
t_{PC}	Port Control Hold After Falling Edge of PROG	$1/10 t_{CY}$	125		ns ⁽²⁾
t_{PR}	PROG to Time P2 Input Must Be Valid	$8/15 t_{CY} - 16$		650	ns ⁽¹⁾
t_{PF}	Input Data Hold Time		0	150	ns ⁽²⁾
t_{DP}	Output Data Setup Time	$2/10 t_{CY}$	250		ns ⁽¹⁾
t_{PD}	Output Data Hold Time	$1/10 t_{CY} - 80$	45		ns ⁽²⁾
t_{PP}	PROG Pulse Width	$6/10 t_{CY}$	750		ns

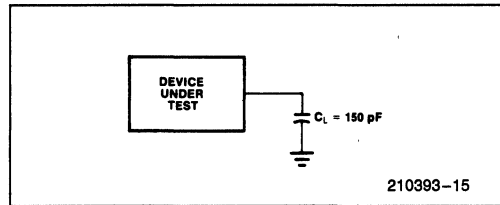
NOTES:

1. $C_L = 80$ pF.
2. $C_L = 20$ pF.
3. $t_{CY} = 1.25$ μs .

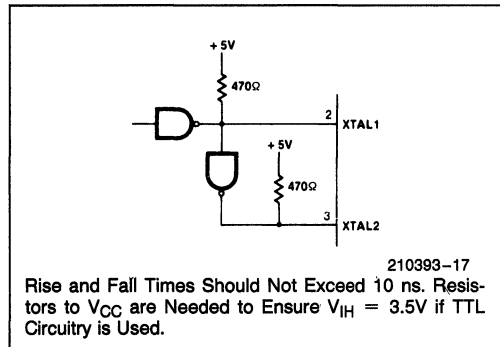
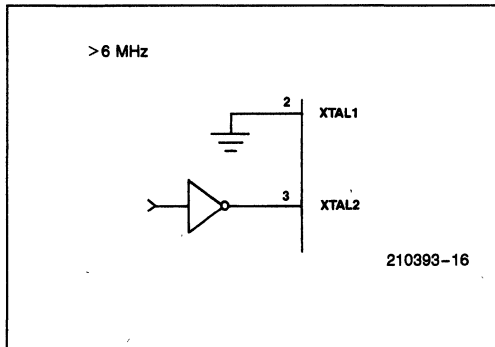
A.C. TESTING INPUT/OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT



DRIVING FROM EXTERNAL SOURCE-TWO OPTIONS



LC OSCILLATOR MODE

L	C	NOMINAL
45 H	20 pF	5.2 MHz
120 H	20 pF	3.2 MHz

$$f = \frac{1}{2\pi\sqrt{LC}}$$

$$C' = \frac{C + 3C_{pp}}{2}$$

$C_{pp} \approx 5-10 \text{ pF}$
Pin-to-Pin Capacitance

210393-18

Each C should be Approximately 20 pF, including Stray Capacitance.

CRYSTAL OSCILLATOR MODE

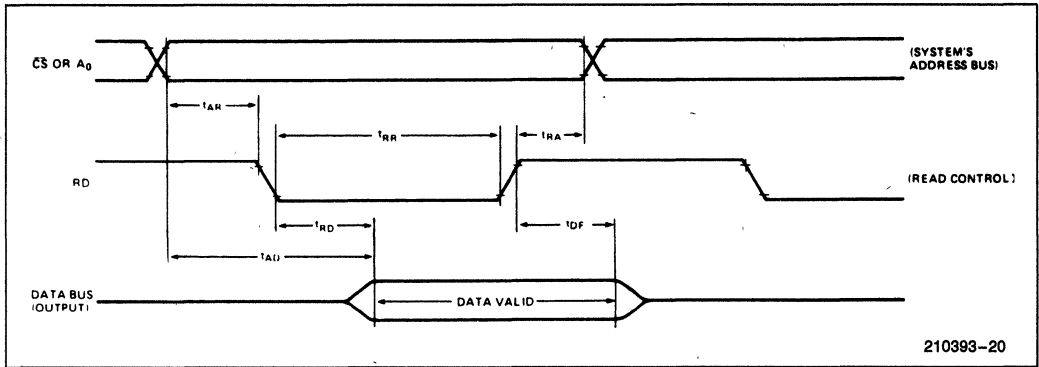
210393-19

C1 5 pF (STRAY 5 pF)
C2 (CRYSTAL + STRAY) 8 pF
C3 20-30 pF INCLUDING STRAY

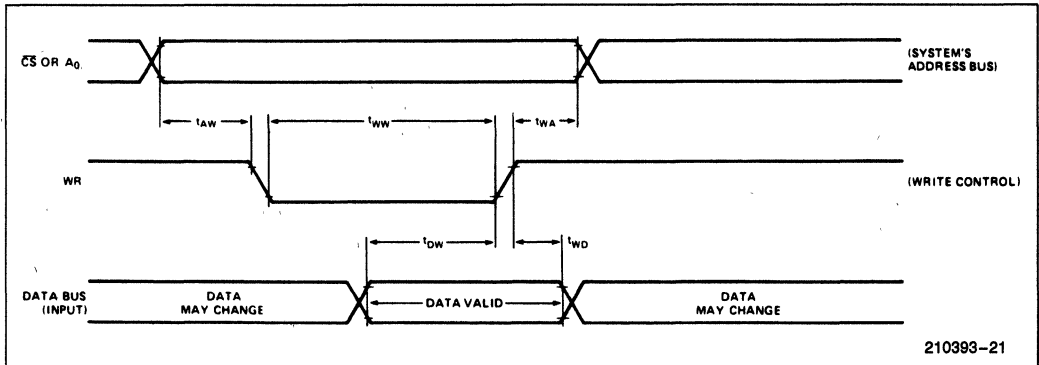
Crystal Series Resistance Should be Less Than 30Ω at 12 MHz; Less Than 75Ω at 6 MHz; Less Than 180Ω at 3.6 MHz.

WAVEFORMS

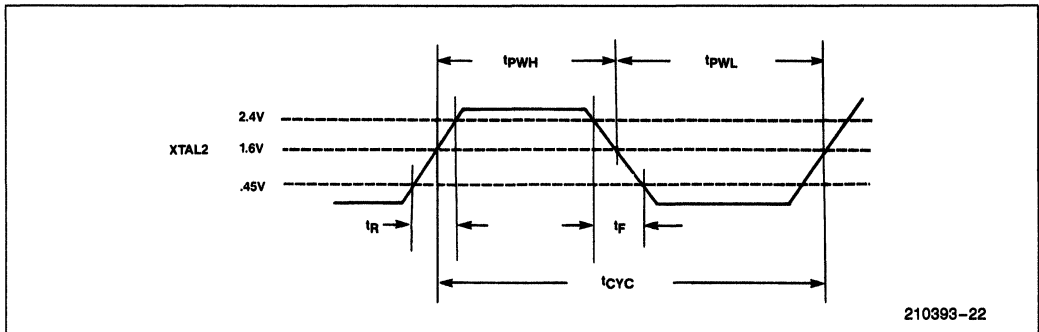
READ OPERATION—DATA BUS BUFFER REGISTER



WRITE OPERATION—DATA BUS BUFFER REGISTER

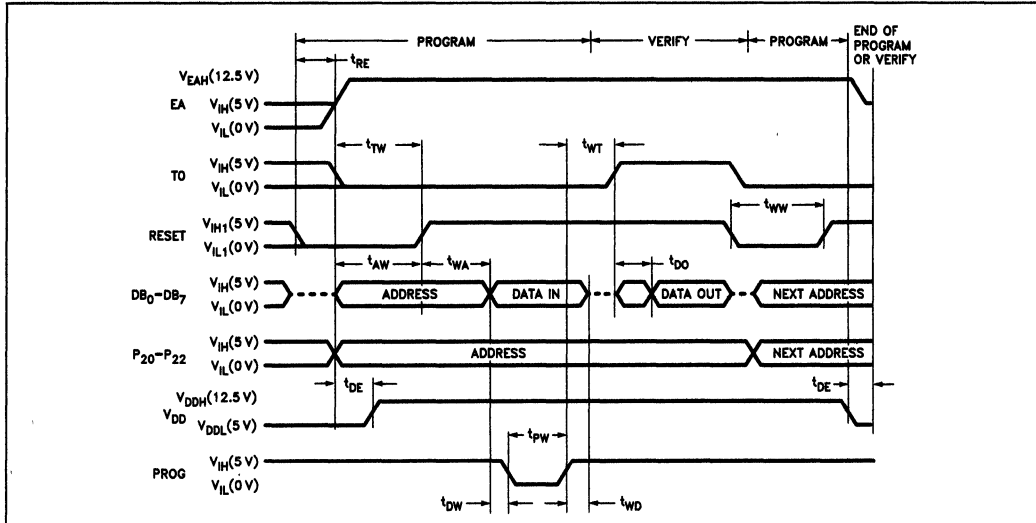


CLOCK TIMING



WAVEFORMS (Continued)

COMBINATION PROGRAM/VERIFY MODE

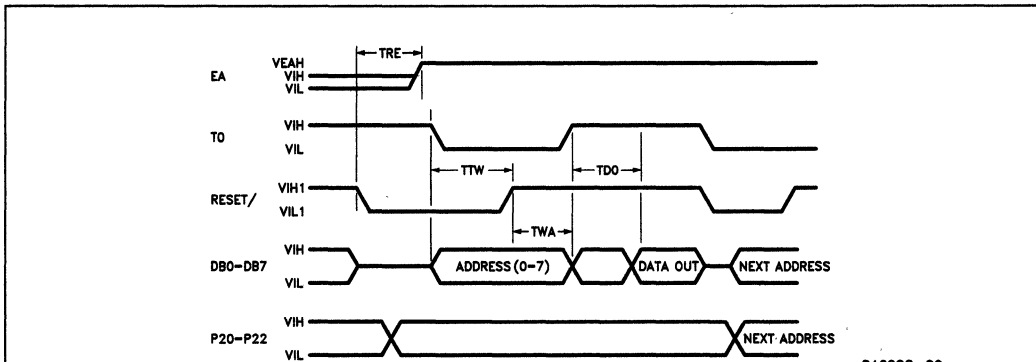


210393-23

NOTES:

1. A₀ must be held low (0V) during program/verify modes.
2. For V_{IH}, V_{IH1}, V_{IL}, V_{IL1}, V_{DDH}, and V_{DDL}, please consult the D.C. Characteristics Table.
3. When programming the 8741AH/8742AH, a 0.1 μF capacitor is required across V_{DD} and ground to suppress spurious voltage transients which can damage the device.

VERIFY MODE



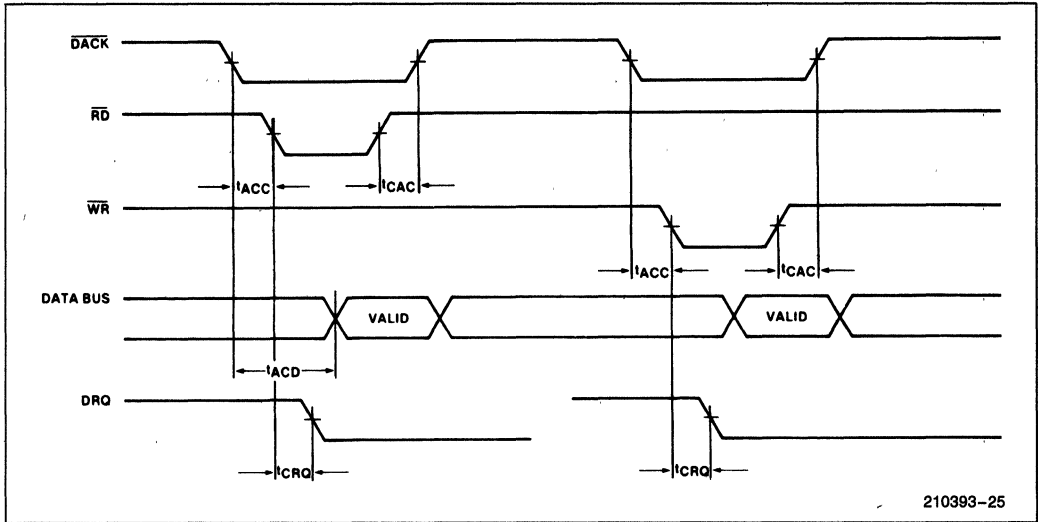
210393-29

NOTES:

1. PROG must float if EA is low.
2. PROG must always float.
3. P₁₀-P₁₇ = 5V or must float.
4. P₂₄-P₂₇ = 5V or must float.
5. A₀ must be held low during programming/verify modes.

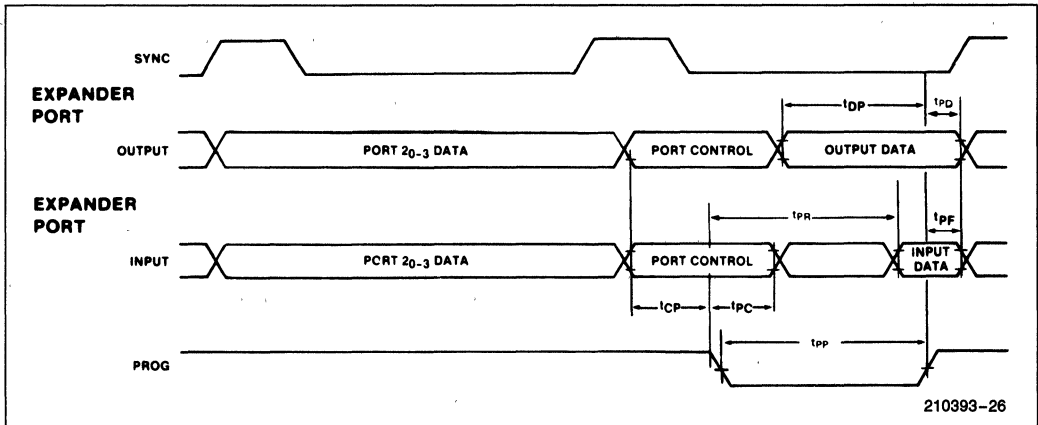
WAVEFORMS (Continued)

DMA



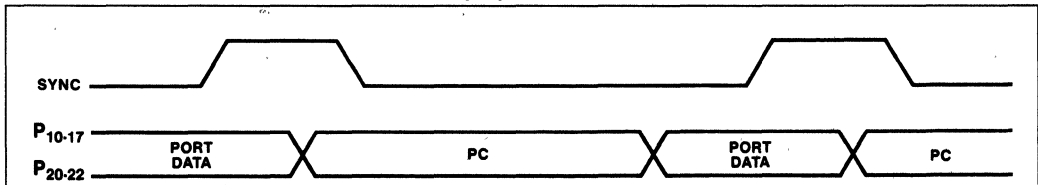
210393-25

PORT 2



210393-26

PORT TIMING DURING EXTERNAL ACCESS (EA)



210393-27

On the Rising Edge of SYNC and EA is Enabled, Port Data is Valid and can be Strobed on the Trailing Edge of Sync the Program Counter Contents are Available.

Table 2. UPITM Instruction Set

Mnemonic	Description	Bytes	Cycles
ACCUMULATOR			
ADD A, Rr	Add register to A	1	1
ADD A, @Rr	Add data memory to A	1	1
ADD A, #data	Add immediate to A	2	2
ADDC A, Rr	Add register to A with carry	1	1
ADDC A, @Rr	Add data memory to A with carry	1	1
ADDC A, #data	Add immediate to A with carry	2	2
ANL A, Rr	AND register to A	1	1
ANL A, @Rr	AND data memory to A	1	1
ANL A, #data	AND immediate to A	2	2
ORL A, Rr	OR register to A	1	1
ORL A, @Rr	OR data memory to A	1	1
ORL A, #data	OR immediate to A	2	2
XRL A, Rr	Exclusive OR register to A	1	1
XRL A, @Rr	Exclusive OR data memory to A	1	1
XRL A, #data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
INPUT/OUTPUT			
IN A, Pp	Input port to A	1	2
OUTL Pp, A	Output A to port	1	2
ANL Pp, #data	AND immediate to port	2	2
ORL Pp, #data	OR immediate to port	2	2
IN A, DBB	Input DBB to A, clear IBF	1	1
OUT DBB, A	Output A to DBB, set OBF	1	1
MOV STS, A	A ₄ -A ₇ to Bits 4-7 of Status	1	1
MOVD A, Pp	Input Expander port to A	1	2
MOVD Pp, A	Output A to Expander port	1	2
ANLD Pp, A	AND A to Expander port	1	2
ORLD Pp, A	OR A to Expander port	1	2
DATA MOVES			
MOV A, Rr	Move register to A	1	1
MOV A, @Rr	Move data memory to A	1	1
MOV A, #data	Move immediate to A	2	2
MOV Rr, A	Move A to register	1	1
MOV @Rr, A	Move A to data memory	1	1
MOV Rr, #data	Move immediate to register	2	2
MOV @Rr, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, Rr	Exchange A and register	1	1
XCH A, @Rr	Exchange A and data memory	1	1
XCHD A, @Rr	Exchange digit of A and register	1	1
MOVP A, @A	Move to A from current page	1	2
MOVP3, A, @A	Move to A from page 3	1	2
TIMER/COUNTER			
MOV A, T	Read Timer/Counter	1	1
MOV T, A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1
CONTROL			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
REGISTERS			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1

Table 2. UPITM Instruction Set (Continued)

Mnemonic	Description	Bytes	Cycles
SUBROUTINE			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
FLAGS			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
BRANCH			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ Rr, addr	Decrement register and jump	2	2
JC addr	Jump on Carry = 1	2	2
JNC addr	Jump on Carry = 0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0 = 1	2	2
JNT0 addr	Jump on T0 = 0	2	2
JT1 addr	Jump on T1 = 1	2	2
JNT1 addr	Jump on T1 = 0	2	2
JF0 addr	Jump on F0 Flag = 1	2	2
JF1 addr	Jump on F1 Flag = 1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag = 0	2	2
JNIBF addr	Jump on IBF Flag = 0	2	2
JOBF addr	Jump on OBF Flag = 1	2	2
JBb addr	Jump on Accumulator Bit	2	2



8243 MCS-48® INPUT/OUTPUT EXPANDER

- Low Cost
- Simple Interface to MCS-48® Microcomputers
- Four 4-Bit I/O Ports
- AND and OR Directly to Ports
- 24-Pin DIP
- Single 5V Supply
- High Output Drive
- Direct Extension of Resident 8048 I/O Ports

The Intel® 8243 is an input/output expander designed specifically to provide a low cost means of I/O expansion for the MCS-48 family of single chip microcomputers. Fabricated in 5 volts NMOS, the 8243 combines low cost, single supply voltage and high drive current capability.

The 8243 consists of four 4-bit bidirectional static I/O ports and one 4-bit port which serves as an interface to the MCS-48 microcomputers. The 4-bit interface requires that only 4 I/O lines of the 8048 be used for I/O expansion, and also allows multiple 8243's to be added to the same bus.

The I/O ports of the 8243 serve as a direct extension of the resident I/O facilities of the MCS-48 microcomputers and are accessed by their own MOV, ANL, and ORL instructions.

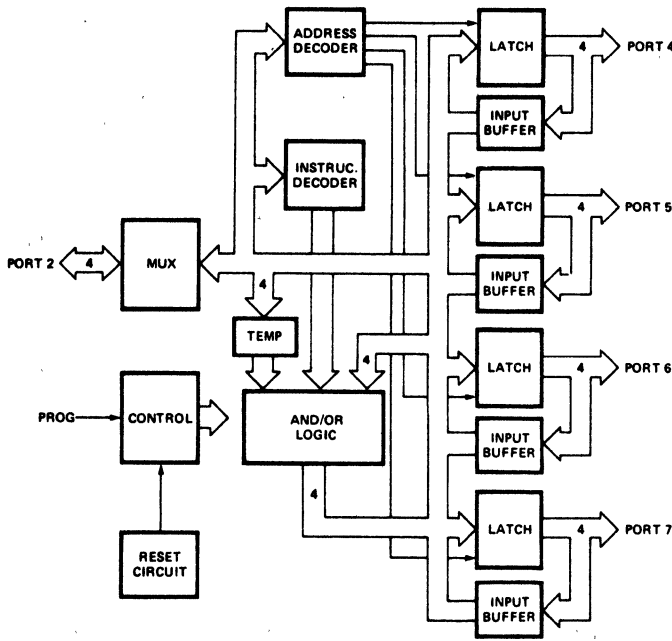


Figure 1. 8243 Block Diagram

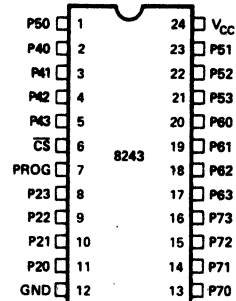


Figure 2. 8243
Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Function
PROG	7	CLOCK INPUT. A high to low transition on PROG signifies that address and control are available on P20–P23, and a low to high transition signifies that data is available on P20–P23.
\overline{CS}	6	CHIP SELECT INPUT. A high on CS inhibits any change of output or internal status.
P20–P23	11–8	Four (4) bit bi-directional port contains the address and control bits on a high to low transition of PROG. During a low to high transition, contains the data for a selected output port if a write operation, or the data from a selected port before the low to high transition if a read operation.
GND	12	0 volt supply.
P40–P43 P50–P53 P60–P63 P70–P73	2–5 1, 23–21 20–17 13–16	Four (4) bit bi-directional I/O ports. May be programmed to be input (during read), low impedance latched output (after write), or a tri-state (after read). Data on pins P20–P23 may be directly written, ANDed or ORed with previous data.
V _{CC}	24	+5 volt supply.

FUNCTIONAL DESCRIPTION

General Operation

The 8243 contains four 4-bit I/O ports which serve as an extension of the on-chip I/O and are addressed as ports 4–7. The following operations may be performed on these ports:

- Transfer Accumulator to Port
- Transfer Port to Accumulator
- AND Accumulator to Port
- OR Accumulator to Port

All communication between the 8048 and the 8243 occurs over Port 2 (P20–P23) with timing provided by an output pulse on the PROG pin of the processor. Each transfer consists of two 4-bit nibbles.

The first containing the “op code” and port address and the second containing the actual 4-bits of data. A high to low transition of the PROG line indicates that address is present while a low to high transition indicates the presence of data. Additional 8243's

may be added to the 4-bit bus and chip selected using additional output lines from the 8048/8748/8035.

Power On Initialization

Initial application of power to the device forces input/output ports 4, 5, 6, and 7 to the tri-state and port 2 to the input mode. The PROG pin may be either high or low when power is applied. The first high to low transition of PROG causes device to exit power on mode. The power on sequence is initiated if V_{CC} drops below 1V.

Address				Instruction	
P21	P20	Code	P23	P22	Code
0	0	Port 4	0	0	Read
0	1	Port 5	0	1	Write
1	0	Port 6	1	0	ORLD
1	1	Port 7	1	1	ANLD

Write Modes

The device has three write modes. MOVD Pi, A directly writes new data into the selected port and old data is lost. ORLD Pi, A takes new data, OR's it with the old data and then writes it to the port. ANLD Pi, A takes new data, AND's it with the old data and then writes it to the port. Operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. On the low to high transition of PROG, data on port 2 is transferred to the logic block of the specified output port.

After the logic manipulation is performed, the data is latched and output. The old data remains latched until new valid outputs are entered.

Read Mode

The device has one read mode. The operation code and port address are latched from the input port 2 on the high to low transition of the PROG pin. As soon as the read operation and port address are decoded, the appropriate outputs are tri-stated, and the input buffers switched on. The read operation is terminated by a low to high transition of the PROG pin. The port (4, 5, 6 or 7) that was selected is switched to the tri-stated mode while port 2 is returned to the input mode.

Normally, a port will be in an output (write mode) or input (read mode). If modes are changed during operation, the first read following a write should be ignored; all following reads are valid. This is to allow the external driver on the port to settle after the first read instruction removes the low impedance drive from the 8243 output. A read of any port will leave that port in a high impedance state.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage on Any Pin
 With Respect to Ground -0.5V to +7V
 Power Dissipation 1W

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

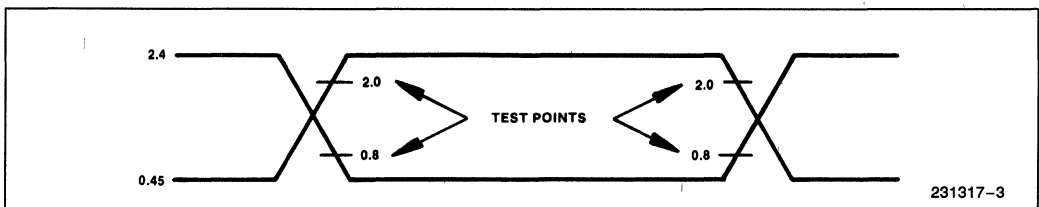
D.C. CHARACTERISTICS $T_A = 0^\circ\text{C to }70^\circ\text{C}$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
V _{IL}	Input Low Voltage	-0.5		0.8	V	
V _{IH}	Input High Voltage	2.0		V _{CC} + 0.5	V	
V _{OL1}	Output Low Voltage Ports 4-7			0.45	V	I _{OL} = 4.5 mA*
V _{OL2}	Output Low Voltage Port 7			1	V	I _{OL} = 20 mA
V _{OH1}	Output High Voltage Ports 4-7	2.4			V	I _{OH} = 240 μA
I _{IL1}	Input Leakage Ports 4-7	-10		20	μA	V _{IN} = V _{CC} to 0V
I _{IL2}	Input Leakage Port 2, CS, PROG	-10		10	μA	V _{IN} = V _{CC} to 0V
V _{OL3}	Output Low Voltage Port 2			0.45	V	I _{OL} = 0.6 mA
I _{CC}	V _{CC} Supply Current		10	20	mA	
V _{OH2}	Output Voltage Port 2	2.4			V	I _{OH} = 100 μA
I _{OL}	Sum of all I _{OL} from 16 Outputs			72	mA	4.5 mA Each Pin

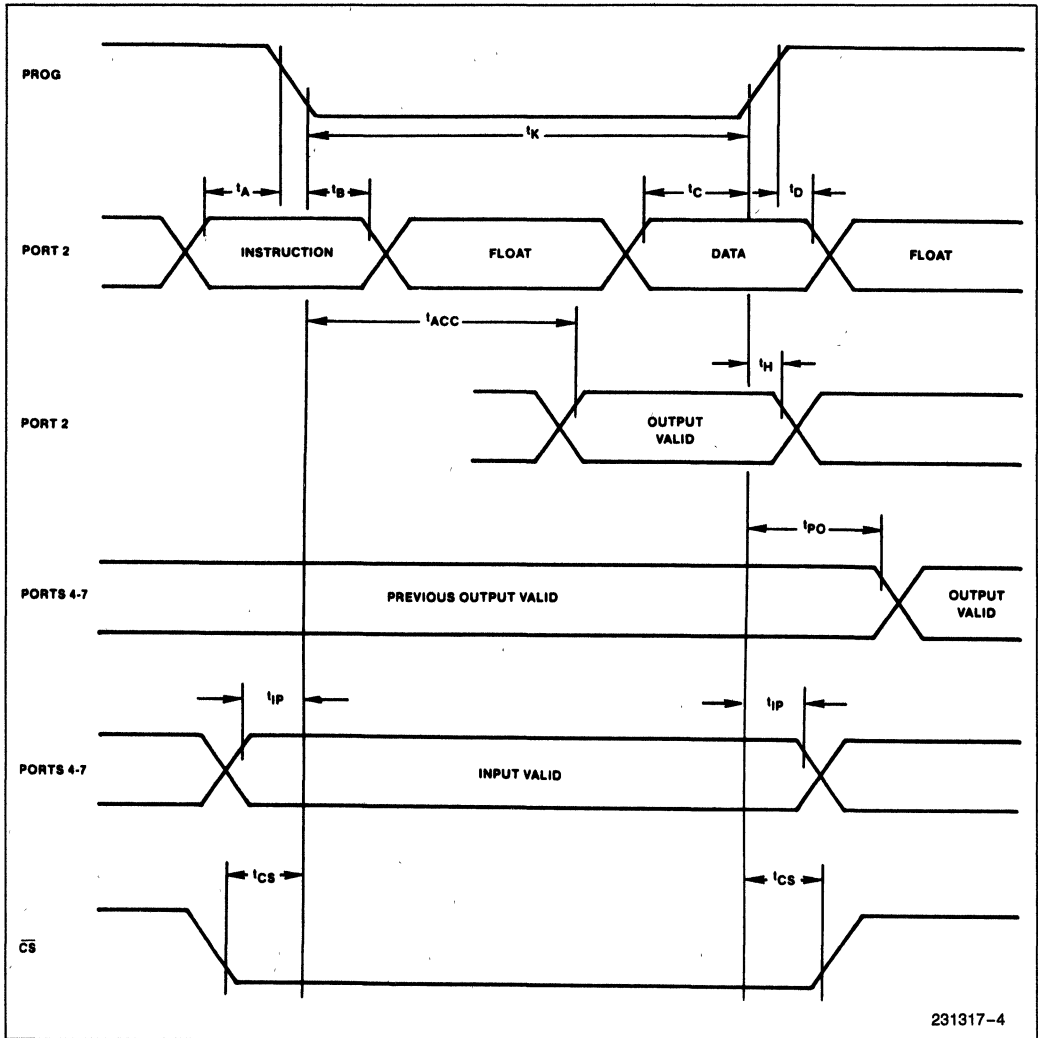
*See following graph for additional sink current capability.

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to }70^\circ\text{C}$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
t _A	Code Valid Before PROG	100		ns	80 pF Load
t _B	Code Valid After PROG	60		ns	20 pF Load
t _C	Data Valid Before PROG	200		ns	80 pF Load
t _D	Data Valid After PROG	20		ns	20 pF Load
t _H	Floating After PROG	0	150	ns	20 pF Load
t _K	PROG Negative Pulse Width	700		ns	
t _{CS}	CS Valid Before/After PROG	50		ns	
t _{PO}	Ports 4-7 Valid After PROG		700	ns	100 pF Load
t _{LP1}	Ports 4-7 Valid Before/After PROG	100		ns	
t _{ACC}	Port 2 Valid After PROG		650	ns	80 pF Load



WAVEFORMS



231317-4

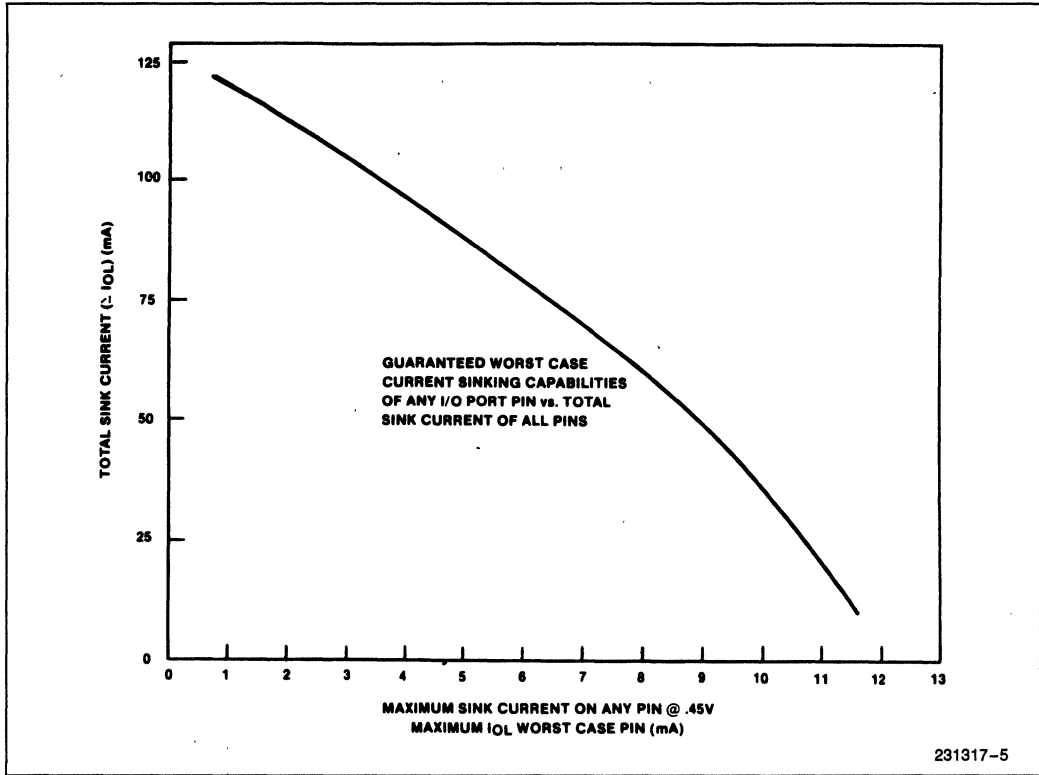


Figure 3

Sink Capability

The 8243 can sink 5 mA @ 0.45V on each of its 16 I/O lines simultaneously. If, however, all lines are not sinking simultaneously or all lines are not fully loaded, the drive capability of any individual line increases as is shown by the accompanying curve.

For example, if only 5 of the 16 lines are to sink current at one time, the curve shows that each of those 5 lines is capable of sinking 9 mA @ 0.45V (if any lines are to sink 9 mA the total I_{OL} must not exceed 45 mA or five 9 mA loads).

Example: How many pins can drive 5 TTL loads (1.6 mA) assuming remaining pins are unloaded?

$$I_{OL} = 5 \times 1.6 \text{ mA} = 8 \text{ mA}$$

$$\epsilon I_{OL} = 60 \text{ mA from curve}$$

$$\# \text{ pins} = 60 \text{ mA} \div 8 \text{ mA/pin} = 7.5 = 7$$

In this case, 7 lines can sink 8 mA for a total of 56 mA. This leaves 4 mA sink current capability which can be divided in any way among the remaining 8 I/O lines of the 8243.

Example: This example shows how the use of the 20 mA sink capability of Port 7 affects the sinking capability of the other I/O lines.

An 8243 will drive the following loads simultaneously.

- 2 loads—20 mA @ 1V (port 7 only)
- 8 loads—4 mA @ 0.45V
- 6 loads—3.2 mA @ 0.45V

Is this within the specified limits?

$$\epsilon I_{OL} = (2 \times 20) + (8 \times 4) + (6 \times 3.2)$$

$$= 91.2 \text{ mA. From the curve: for } I_{OL} = 4 \text{ mA, } \epsilon I_{OL} \approx 93 \text{ mA. Since } 91.2 \text{ mA} < 93 \text{ mA the loads are within specified limits.}$$

Although the 20 mA @ 1V loads are used in calculating ϵI_{OL} , it is the largest current required @ 0.45V which determines the maximum allowable ϵI_{OL} .

NOTE:

A 10 K Ω to 50 K Ω pullup resistor to +5V should be added to 8243 outputs when driving to 5V CMOS directly.

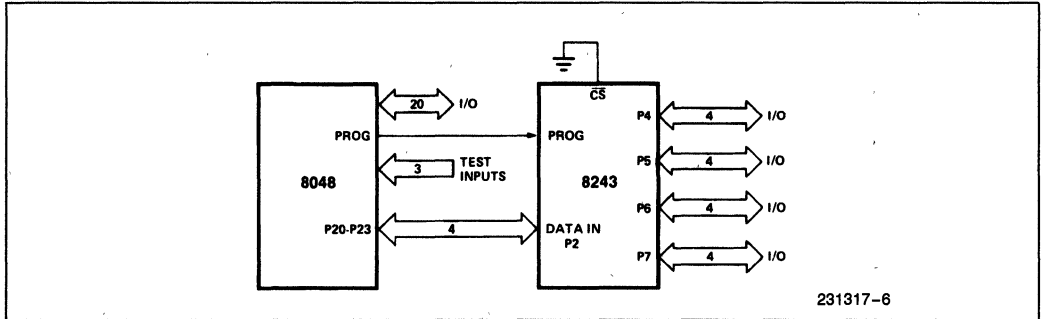


Figure 4. Expander Interface

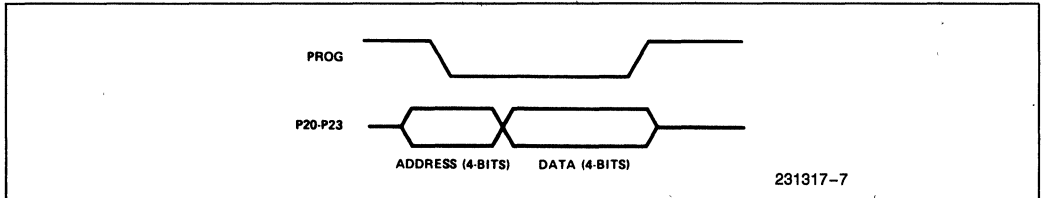


Figure 5. Output Expander Timing

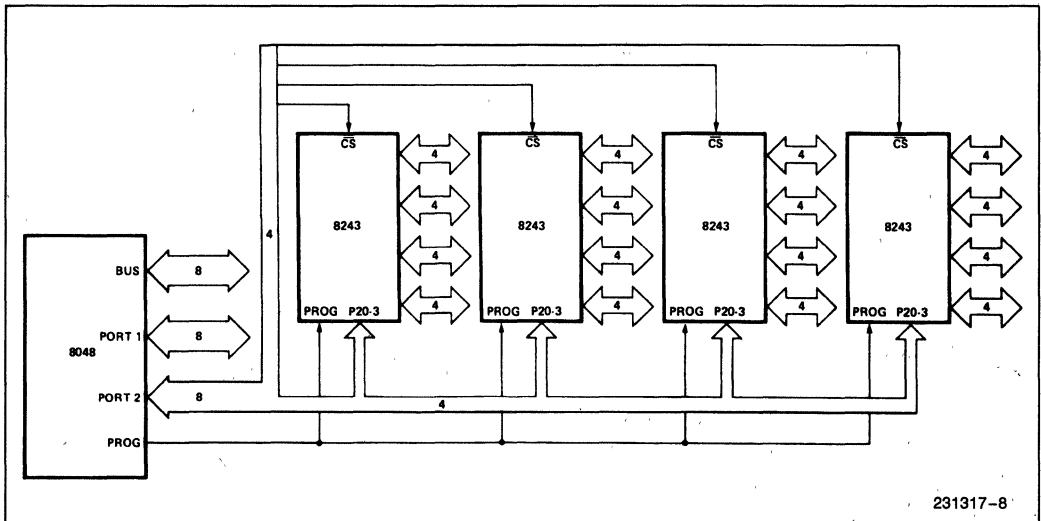


Figure 6. Using Multiple 8243's



November 1986

**Microprocessor Peripherals
UPI™-41AH/42AH
User's Manual**

Order Number: 231318-002

CHAPTER 1 INTRODUCTION

Accompanying the introduction of microprocessors such as the 8088, 8086, 80186 and 80286 there has been a rapid proliferation of intelligent peripheral devices. These special purpose peripherals extend CPU performance and flexibility in a number of important ways.

Table 1-1. Intelligent Peripheral Devices

8255 (GPIO)	Programmable Peripheral Interface
8251A(USART)	Programmable Communication Interface
8253 (TIMER)	Programmable Interval Timer
8257 (DMA)	Programmable DMA Controller
8259	Programmable Interrupt Controller
8271 (SDFDC), 8272 (DDFDC)	Programmable Floppy Disk Controllers
8273 (SDLC)	Programmable Synchronous Data Link Controller
8274	Programmable Multiprotocol-Serial Communications Controller
8275/8276 (CRT)	Programmable CRT Controllers
8279 (PKD)	Programmable Keyboard/Display Controller
8291A, 8292, 8293	Programmable GPIB System Talker, Listener, Controller

Intelligent devices like the 8272 floppy disk controller and 8273 synchronous data link controller (see Table 1-1) can preprocess serial data and perform control tasks which off-load the main system processor. Higher overall system throughput is achieved and software complexity is greatly reduced. The intelligent peripheral chips simplify master processor control tasks by performing many functions externally in peripheral hardware rather than internally in main processor software.

Intelligent peripherals also provide system flexibility. They contain on-chip mode registers which are programmed by the master processor during system initialization. These control registers allow the peripheral to be configured into many different operation modes. The user-defined program for the peripheral is stored in main system memory and is transferred to the peripheral's registers whenever a mode change is required. Of course, this type of flexibility requires software overhead in the master system which tends to limit the benefit derived from the peripheral chip.

In the past, intelligent peripherals were designed to handle very specialized tasks. Separate chips were designed for communication disciplines, parallel I/O, keyboard encoding, interval timing, CRT control, etc. Yet, in spite of the large number of devices available and the increased flexibility built into these chips, there is still a large number of microcomputer peripheral control tasks which are not satisfied.

With the introduction of the Universal Peripheral Interface (UPI) microcomputer, Intel has taken the intelligent peripheral concept a step further by providing an intelligent controller that is fully user programmable. It is a complete single-chip microcomputer which can connect directly to a master processor data bus. It has the same advantages of intelligence and flexibility which previous peripheral chips offered. In addition, UPIs are user-programmable: it has 1K/2K bytes (41AH/42AH respectively) of ROM or EPROM memory for program storage plus 256 bytes of RAM memory for data storage or initialization from the master processor. The UPI device allows a designer to fully specify his control algorithm in the peripheral chip without relying on the master processor. Devices like printer controllers and keyboard scanners can be completely self-contained, relying on the master processor only for data transfer.

The UPI family currently consists of five components:

- 8741AH microcomputer with 1K EPROM memory
- 8041AH microcomputer with 1K ROM memory
- 8042AH microcomputer with 2K ROM memory
- 8243 I/O expander device
- 8742AH microcomputer with 2K EPROM memory

The 8741AH, 8041AH, 8742AH and 8042AH single chip microcomputers are functionally equivalent except for the type and amount of program memory available with each. These devices have the following main features:

- 8-bit CPU
- 8-bit data bus interface registers
- 2K by 8 bit ROM or EPROM memory (1K for 8041AH/8741AH)
- 256 by 8 bit RAM memory
- Interval timer/event counter
- Two 8-bit TTL compatible I/O ports
- Resident clock oscillator
- 12 MHz operation, 1.25 μ sec instruction cycle for 8041AH, 8742AH, 8042AH

HMOS processing has been applied to the UPI family to allow for additional performance and memory capability while reducing costs. The 8041AH, 8741AH, 8042AH, 8742AH are all pin and software compatible. This allows growth in present designs to incorporate new features and add additional performance. For new designs, the additional memory and performance of the 8042AH/8742AH extends the UPI 'grow your own solution' concept to more complex motor control tasks, 80-column printers and process control applications as examples.

The 8243 device is an I/O multiplexer which allows expansion of I/O to over 100 lines (if seven devices are used). All three parts are fabricated with N-channel MOS technology and require a single, 5V supply for operation.

INTERFACE REGISTERS FOR MULTI-PROCESSOR CONFIGURATIONS

In the normal configuration, the 8041AH/8741AH, 8042AH/8742AH interfaces to the system bus, just like any intelligent peripheral device (see Figure 1-1). The host processor and the 8041AH/8741AH, 8042AH/8742AH form a loosely coupled multi-processor system, that is, communications between the two processors are direct. Common resources are three address-

sable registers located physically on the 8041AH/8741AH, 8042AH/8742AH. These registers are the Data Bus Buffer Input (DBBIN), Data Bus Buffer Output (DBBOUT), and Status (STATUS) registers. The host processor may read data from DBBOUT or write commands and data into DBBIN. The status of DBBOUT and DBBIN plus user-defined status is supplied in STATUS. The host may read STATUS at any time. An interrupt to the UPI processor is automatically generated (if enabled) when DBBIN is loaded.

Because the UPI contains a complete microcomputer with program memory, data memory, and CPU it can function as a "Universal" controller. A designer can program the UPI to control printers, tape transports, or multiple serial communication channels. The UPI can also handle off-line arithmetic processing, or any number of other low speed control tasks.

POWERFUL 8-BIT PROCESSOR

The UPI contains a powerful, 8-bit CPU with as fast as 1.25 μ sec cycle time and two single-level interrupts. Its instruction set includes over 90 instructions for easy software development. Most instructions are single byte and single cycle and none are more than two bytes long. The instruction set is optimized for bit manipulation and I/O operations. Special instructions are included to

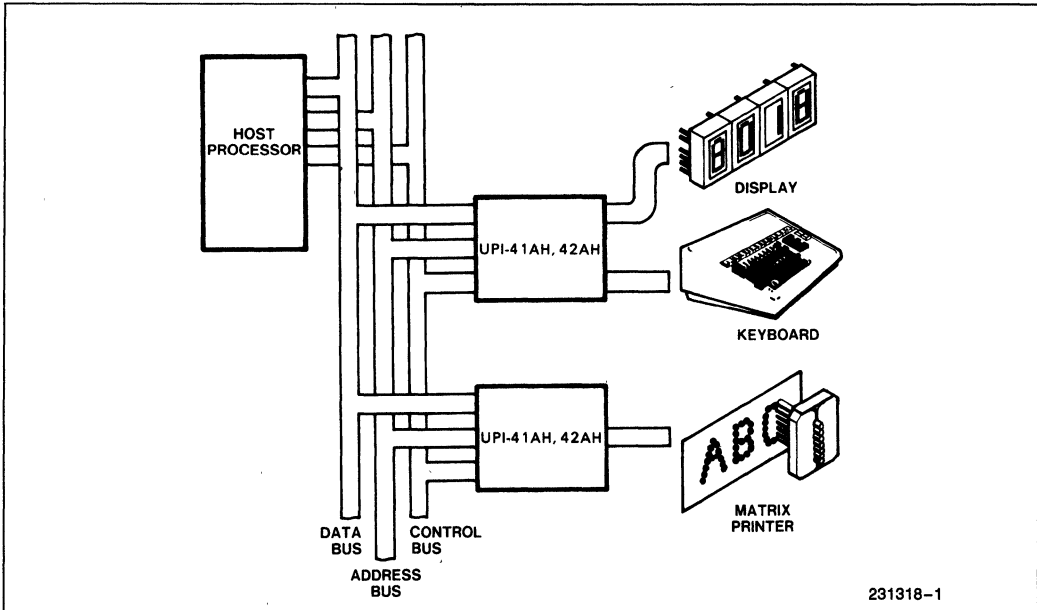


Figure 1-1. Interfacing Peripherals To Microcomputer Systems

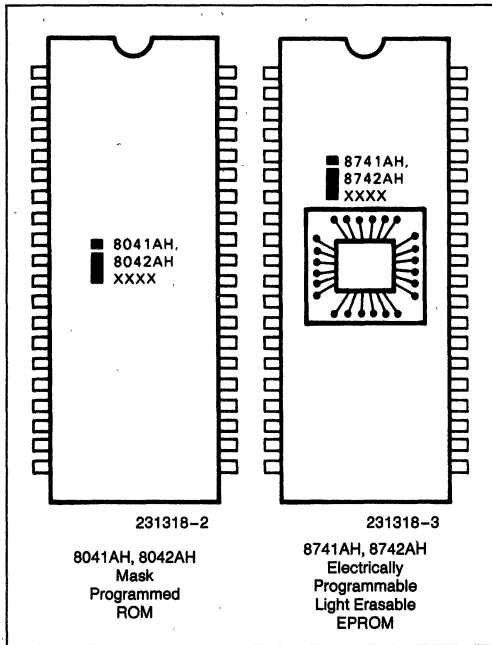


Figure 1-2. Pin Compatible ROM/EPROM Versions

allow binary or BCD arithmetic operations, table look-up routines, loop counters, and N-way branch routines.

SPECIAL INSTRUCTION SET FEATURES

- For Loop Counters:
Decrement Register and Jump if not zero.
- For Bit Manipulation:
AND to A (immediate data or Register)
OR to A (immediate data or Register)
XOR to A (immediate data or Register)
AND to Output Ports (Accumulator)
OR to Output Ports (Accumulator)
Jump Conditionally on any bit in A
- For BDC Arithmetic:
Decimal Adjust A
Swap 4-bit Nibbles of A
Exchange lower nibbles of A and Register
Rotate A left or right with or without Carry
- For Lookup Tables:
Load A from Page of ROM (Address in A)
Load A from Current Page of ROM (Address in A)

Features for Peripheral Control

The UPI 8-bit interval timer/event counter can be used to generate complex timing sequences for control applications or it can count external events such as switch closures and position encoder pulses. Software timing loops can be simplified or eliminated by the interval timer. If enabled, an interrupt to the CPU will occur when the timer overflows.

The UPI I/O complement contains two TTL-compatible 8-bit bidirectional I/O ports and two general-purpose test inputs. Each of the 16 port lines can individually function as either input or output under software control. Four of the port lines can also function as an

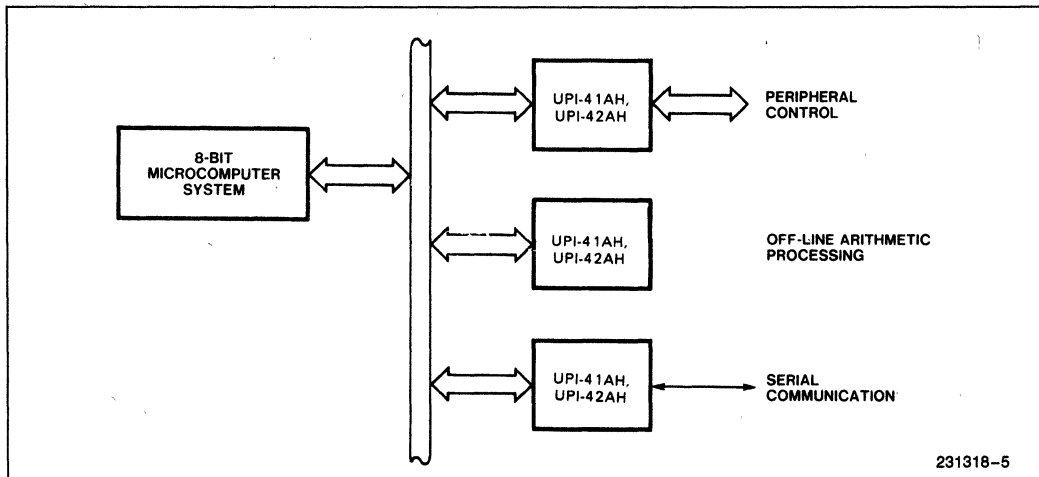


Figure 1-3. Interfaces and Protocols for Multiprocessor Systems

interface for the 8243 I/O expander which provides four additional 4-bit ports that are directly addressable by UPI software. The 8243 expander allows low cost I/O expansion for large control applications while maintaining easy and efficient software port addressing.

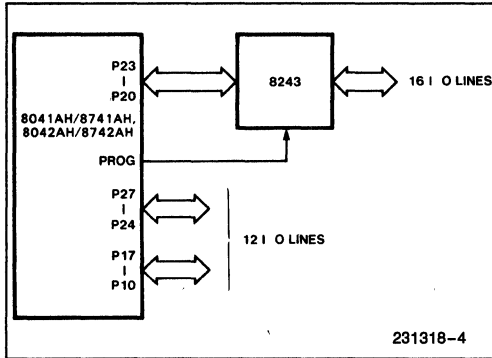


Figure 1-4. 8243 I/O Expander Interface

On-Chip Memory

The UPI's 256 bytes of data memory include dual working register banks and an 8-level program counter stack. Switching between the register banks allows fast response to interrupts. The stack is used to store return addresses and processor status upon entering a subroutine.

The UPI program memory is available in two types to allow flexibility in moving from design to prototype to production with the same PC layout. The 8741AH, 8742AH device with EPROM memory is very economical for initial system design and development. Its program memory can be electrically programmed using the Intel Universal PROM Programmer. When changes are needed, the entire program can be erased using UV lamp and reprogrammed in about 20 minutes. This means the 8741AH/8742AH can be used as a single chip "breadboard" for very complex interface and control problems. After the 8741AH/8742AH is programmed it can be tested in the actual production level PC board and the actual functional environment. Changes required during system debugging can be made in the 8741AH/8742AH program much more easily than they could be made in a random logic design. The system configuration and PC layout can remain fixed during the development process and the turn around time between changes can be reduced to a minimum.

At any point during the development cycle, the 8741AH/8742AH EPROM part can be replaced with the low cost 8041AH, 8042AH respectively with factory mask programmed memory. The transition from system development to mass production is made smoothly because the 8741AH and 8041AH, 8742AH and 8042AH parts are completely pin compatible. 8742AHs or 8042AHs can be used in an 8041AH/8741AH socket. This feature allows extensive testing with the EPROM part, even into initial shipments to customers. Yet, the transition to low-cost ROM is simplified to the point of being merely a package substitution.

PREPROGRAMMED UPI's

The 8292, 8294, and 8295 are 8042AH's that are programmed by Intel and sold as standard peripherals. The 8292 is a GPIB controller, part of a three chip GPIB system. The 8294 is a Data Encryption Unit that implements the National Bureau of Standards data encryption algorithm. The 8295 is a dot matrix printer controller designed especially for the LRC 7040 series dot matrix impact printers. These parts illustrate the great flexibility offered by the UPI family.

DEVELOPMENT SUPPORT

The UPI microcomputer is fully supported by Intel with development tools like the UPP PROM programmer already mentioned. An ICE-42AH in-circuit emulator is also available to allow UPI software and hardware to be developed easily and quickly. The combination of device features and Intel development support make the UPI an ideal component for low-speed peripheral control applications.

UPI DEVELOPMENT SUPPORT

- 8048/8041AH/8042AH Assembler
- Universal PROM Programmer UPP Series
- ICE-41A Module
- MULTI-ICE
- Insite User's Library
- Application Engineers
- Training Courses

CHAPTER 2 FUNCTIONAL DESCRIPTION

The UPI-41AH, 42AH microcomputer is an intelligent peripheral controller designed to operate in iAPX-86, 88, MCS-85, MCS-80, MCS-51 and MCS-48 systems. The UPI's architecture, illustrated in Figure 2-1, is based on a low cost, single-chip microcomputer with program memory, data memory, CPU, I/O, event timer and clock oscillator in a single 40-pin package. Special interface registers are included which enable the UPI to function as a peripheral to an 8-bit master processor.

This chapter provides a basic description of the UPI microcomputer and its system interface registers. Unless otherwise noted the descriptions in this section ap-

ply to both the 8741AH, 8742AH (with UV erasable program memory) and the 8041AH, 8042AH (with factory mask programmed memory). These two devices are so similar that they can be considered identical under most circumstances. All functions described in this chapter apply to the 8041AH, 8042AH, and 8741AH, 8742AH.

PIN DESCRIPTION

The 8041AH/8741AH, 8042AH/8742AH are packaged in 40-pin Dual In-Line (DIP) packages. The pin configuration for both devices is shown in Figure 2-2. Figure 2-3 illustrates the UPI Logic Symbol.

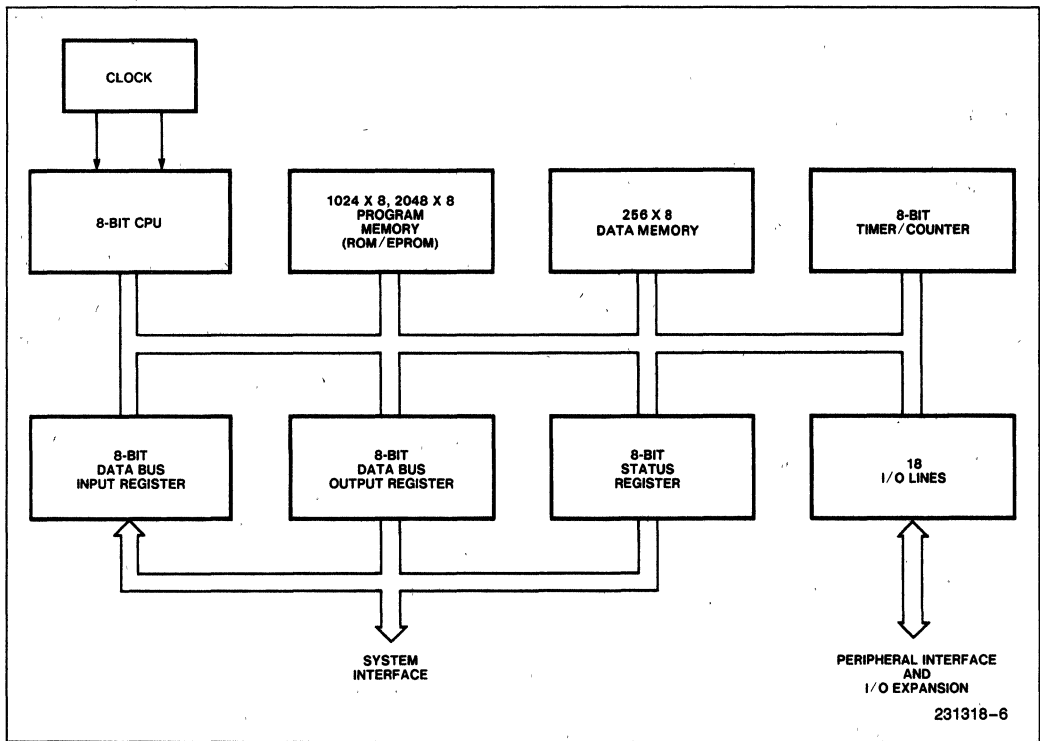
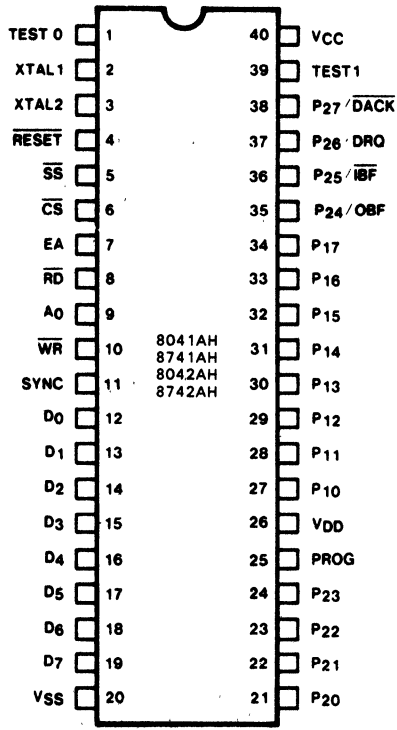
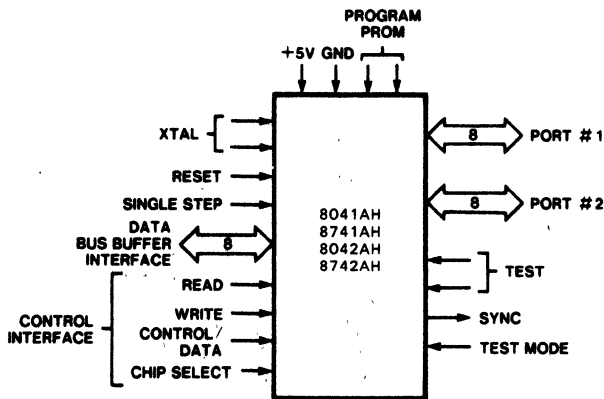


Figure 2-1. UPI-41AH, 42AH Single Chip Microcomputer



231318-7

Figure 2-2. Pin Configuration



231318-8

Figure 2-3. Logic Symbol

The following section summarizes the functions of each UPI-41A pin. NOTE that several pins have two or more functions which are described in separate paragraphs.

Table 2-1. Pin Description

Symbol	Pin No.	Type	Name and Function
D ₀ -D ₇ (BUS)	12-19	I/O	DATA BUS: Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI-41AH, 42AH microcomputer to an 8-bit master system data bus.
P ₁₀ -P ₁₇	27-34	I/O	PORT 1: 8-bit, PORT 1 quasi-bidirectional I/O lines.
P ₂₀ -P ₂₇	21-24 35-38	I/O	PORT 2: 8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P ₂₀ -P ₂₃) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits (P ₂₄ -P ₂₇) can be programmed to provide interrupt Request and DMA Handshake capability. Software control can configure P ₂₄ as Output Buffer Full (OBF) interrupt, P ₂₅ as Input Buffer Full (IBF) interrupt, P ₂₆ as DMA Request (DRQ), and P ₂₇ as DMA ACKnowledge (DACK).
WR	10	I	WRITE: I/O write input which enables the master CPU to write data and command words to the UPI-41AH INPUT DATA BUS BUFFER.
RD	8	I	READ: I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
CS	6	I	CHIP SELECT: Chip select input used to select one UPI-41AH, 42AH microcomputer out of several connected to a common data bus.
A ₀	9	I	COMMAND/DATA SELECT: Address input used by the master processor to indicate whether byte transfer is data (A ₀ = 0) or command (A ₀ = 1).
TEST 0, TEST 1	1 39	I	TEST INPUTS: Input pins can be directly tested using conditional branch instructions. FREQUENCY REFERENCE: TEST 1 (T ₁) also functions as the event timer input (under software control). TEST0 (T ₀) is used during PROM programming and verification in the 8741AH, 8742AH.
XTAL 1, XTAL 2	2 3	I	INPUTS: Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
SYNC	11	O	OUTPUT CLOCK: Output signal which occurs once per UPI-41AH instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
EA	7	I	EXTERNAL ACCESS: External access input which allows emulation, testing and PROM/ROM verification.
PROG	25	I/O	PROGRAM: Multifunction pin used as the program pulse input during PROM programming. During I/O expander access the PROG pin acts as an address/data strobe to the 8243.
RESET	4	I	RESET: Input used to reset status flip-flops and to set the program counter to zero. RESET is also used during PROM programming and verification.
SS	5	I	SINGLE STEP: Single step input used in conjunction with the SYNC output to step the program through each instruction.
V _{CC}	40		POWER: +5V main power supply pin.
V _{DD}	26		POWER: +5V during normal operation. +25V during programming operation, +21V for programming 8742AH. Low power standby pin in ROM version.
V _{SS}	20		GROUND: Circuit ground potential.

The following sections provide a detailed functional description of the UPI microcomputer. Figure 2-4 illustrates the functional blocks within the UPI device.

CPU SECTION

The CPU section of the UPI-41AH, 42AH microcomputer performs basic data manipulations and controls data flow throughout the single chip computer via the internal 8-bit data bus. The CPU section includes the following functional blocks shown in Figure 2-4:

- Arithmetic Logic Unit (ALU)
- Instruction Decoder
- Accumulator
- Flags

Arithmetic Logic Units (ALU)

The ALU is capable of performing the following operations:

- ADD with or without carry
- AND, OR, and EXCLUSIVE OR
- Increment, Decrement
- Bit complement
- Rotate left or right
- Swap
- BCD decimal adjust

In a typical operation data from the accumulator is combined in the ALU with data from some other source on the UPI-41AH, 42AH internal bus (such as a register or an I/O port). The result of an ALU operation can be transferred to the internal bus or back to the accumulator.

If an operation such as an ADD or ROTATE requires more than 8 bits, the CARRY flag is used as an indicator. Likewise, during decimal adjust and other BCD operations the AUXILIARY CARRY flag can be set and acted upon. These flags are part of the Program Status Word (PSW).

Instruction Decoder

During an instruction fetch, the operation code (opcode) portion of each program instruction is stored and decoded by the instruction decoder. The decoder generates outputs used along with various timing signals to control the functions performed in the ALU. Also, the instruction decoder controls the source and destination of ALU data.

Accumulator

The accumulator is the single most important register in the processor. It is the primary source of data to the ALU and is often the destination for results as well. Data to and from the I/O ports and memory normally passes through the accumulator.

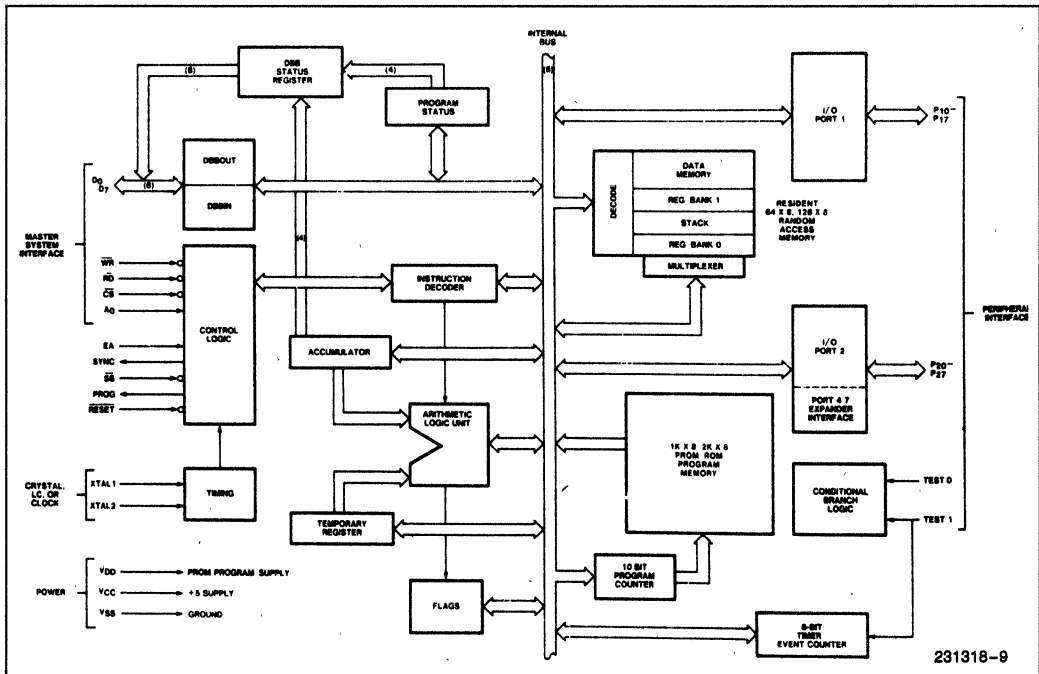


Figure 2-4. UPI-41AH, 42AH™ Block Diagram

PROGRAM MEMORY

The UPI-41AH, 42AH microcomputer has 1024, 2048 8-bit words of resident, read-only memory for program storage. Each of these memory locations is directly addressable by a 10-bit program counter. Depending on the type of application and the number of program changes anticipated, two types of program memory are available:

- 8041AH, 8042AH with mask programmed ROM Memory
- 8741AH, 8742AH with electrically programmable EPROM Memory

The 8041AH and 8741AH, 8042AH and 8742AH are functionally identical parts and are completely pin compatible. The 8742AH and 8042AH can be used in 8041AH, 8741AH sockets. The 8041AH, 8042AH has ROM memory which is mask programmed to user specification during fabrication. The 8741AH/8742AH are electrically programmed by the user using the Universal PROM Programmer (UPP series) with a UPP-848 or UPP-549 Personality Card. It can be erased using ultraviolet light and reprogrammed at any time.

A program memory map is illustrated in Figure 2-5. Memory is divided into 256 location 'pages' and three locations are reserved for special use:

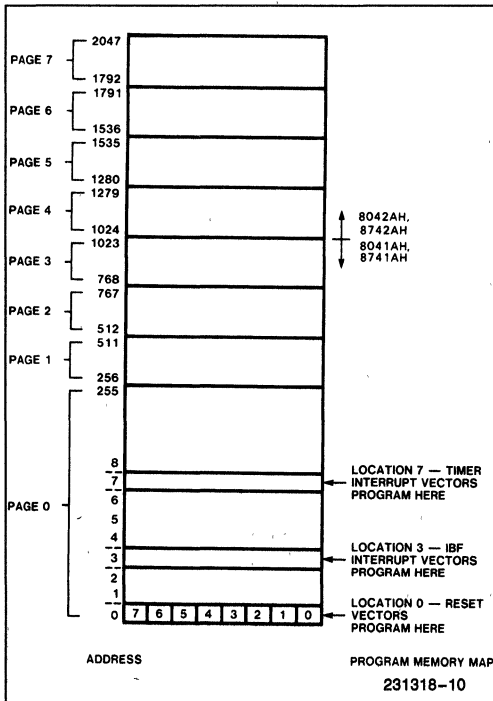


Figure 2-5. Program Memory Map

INTERRUPT VECTORS

1) Location 0

Following a $\overline{\text{RESET}}$ input to the processor, the next instruction is automatically fetched from location 0.

2) Location 3

An interrupt generated by an Input Buffer Full (IBF) condition (when the IBF interrupt is enabled) causes the next instruction to be fetched from location 3.

3) Location 7

A timer overflow interrupt (when enabled) will cause the next instruction to be fetched from location 7.

Following a system $\overline{\text{RESET}}$, program execution begins at location 0. Instructions in program memory are normally executed sequentially. Program control can be transferred out of the main line of code by an input buffer full (IBF) interrupt or a timer interrupt, or when a jump or call instruction is encountered. An IBF interrupt (if enabled) will automatically transfer control to location 3 while a timer interrupt will transfer control to location 7.

All conditional JUMP instructions and the indirect JUMP instruction are limited in range to the current 256-location page (that is, they alter PC bits 0-7 only). If a conditional JUMP or indirect JUMP begins in location 255 of a page, it must reference a destination on the following page.

Program memory can be used to store constants as well as program instructions. The UPI-41AH, 42AH instruction set contains an instruction (MOVP3) designed specifically for efficient transfer of look-up table information from page 3 of memory.

DATA MEMORY

The UPI-41AH, 42AH universal peripheral interface has 256 8-bit words of random access data memory. This memory contains two working register banks, an 8-level program counter stack and a scratch pad memory, as shown in Figure 2-6. The amount of scratch pad memory available is variable depending on the number of addresses nested in the stack and the number of working registers being used.

Addressing Data Memory

The first eight locations in RAM are designated as working registers R₀-R₇. These locations (or registers) can be addressed directly by specifying a register number in the instruction. Since these locations are easily addressed, they are generally used to store frequently accessed intermediate results. Other locations in data memory are addressed indirectly by using R₀ or R₁ to specify the desired address.

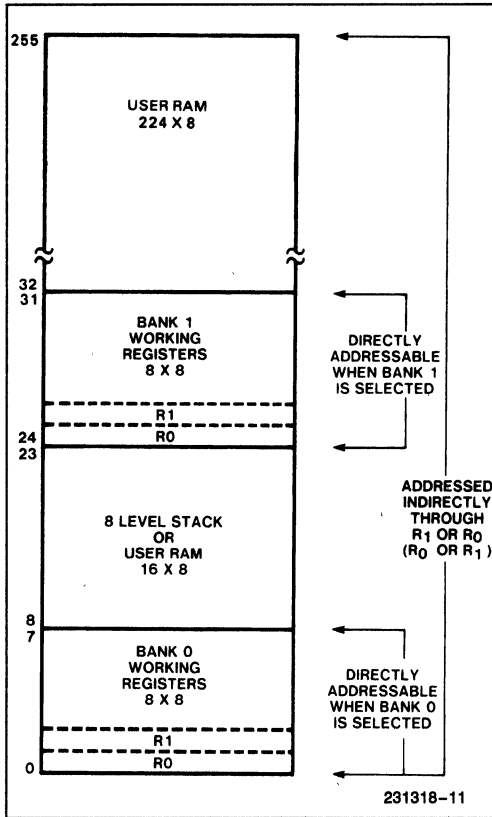


Figure 2-6. Data Memory Map

Working Registers

Dual banks of eight working registers are included in the UPI-41AH, 42AH data memory. Locations 0–7 make up register bank 0 and locations 24–31 form register bank 1. A **RESET** signal automatically selects register bank 0. When bank 0 is selected, references to R_0 – R_7 in UPI-41AH, 42AH instructions operate on locations 0–7 in data memory. A “select register bank” instruction is used to selected between the banks during program execution. If the instruction **SEL RB1** (Select Register Bank 1) is executed, then program references to R_0 – R_7 will operate on locations 24–31. As stated previously, registers 0 and 1 in the active register bank are used as indirect address registers for all locations in data memory.

Register bank 1 is normally reserved for handling interrupt service routines, thereby preserving the contents of the main program registers. The **SEL RB1** instruction can be issued at the beginning of an interrupt service routine. Then, upon return to the main program, an **RETR** (return & restore status) instruction will automatically restore the previously selected bank. During

interrupt processing, registers in bank 0 can be accessed indirectly using R_0 and R_1 .

If register bank 1 is not used, registers 24–31 can still serve as additional scratch pad memory.

Program Counter Stack

RAM locations 8–23 are used as an 8-level program counter stack. When program control is temporarily passed from the main program to a subroutine or interrupt service routine, the 10-bit program counter and bits 4–7 of the program status word (PSW) are stored in two stack locations. When control is returned to the main program via an **RETR** instruction, the program counter and PSW bits 4–7 are restored. Returning via an **RET** instruction does not restore the PSW bits, however. The program counter stack is addressed by three stack pointer bits in the PSW (bits 0–2). Operation of the program counter stack and the program status word is explained in detail in the following sections.

The stack allows up to eight levels of subroutine ‘nesting’; that is, a subroutine may call a second subroutine, which may call a third, etc., up to eight levels. Unused stack locations can be used as scratch pad memory. Each unused level of subroutine nesting provides two additional RAM locations for general use.

The following sections provide a detailed description of the Program Counter Stack and the Program Status Word.

PROGRAM COUNTER

The UPI-41AH, 42AH microcomputer has a 10-bit program counter (PC) which can directly address any of the 1024, 2048 locations in program memory. The program counter always contains the address of the next instruction to be executed and is normally incremented sequentially for each instruction to be executed when each instruction fetches occurs.

When control is temporarily passed from the main program to a subroutine or an interrupt routine, however, the PC contents must be altered to point to the address of the desired routine. The stack is used to save the current PC contents so that, at the end of the routine, main program execution can continue. The program counter is initialized to zero by a **RESET** signal.

PROGRAM COUNTER STACK

The Program Counter Stack is composed of 16 locations in Data Memory as illustrated in Figure 2-7. These RAM locations (8 through 23) are used to store the 10-bit program counter and 4 bits of the program status word.

An interrupt or Call to a subroutine causes the contents of the program counter to be stored in one of the 8 register pairs of the program counter stack.

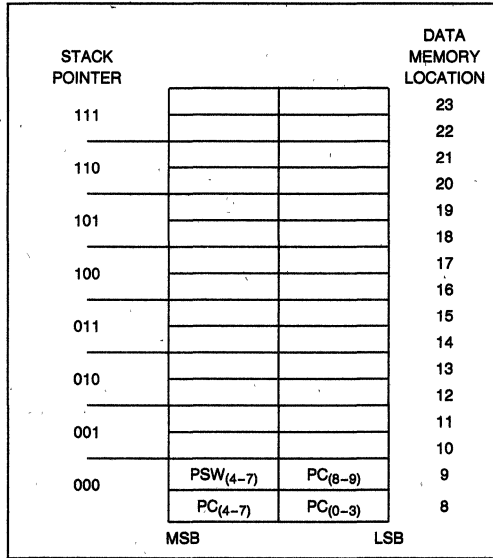


Figure 2-7. Program Counter Stack

A 3-bit Stack Pointer which is part of the Program Status Word (PSW) determines the stack pair to be used at a given time. The stack pointer is initialized by a RESET signal to 00H which corresponds to RAM locations 8 and 9.

The first call or interrupt results in the program counter and PSW contents being transferred to RAM locations 8 and 9 in the format shown in Figure 2-7. The stack pointer is automatically incremented by 1 to point to location 10 and 11 in anticipation of another CALL.

Nesting of subroutines within subroutines can continue up to 8 levels without overflowing the stack. If overflow does occur the deepest address stored (locations 8 and 9) will be overwritten and lost since the stack pointer overflows from 07H to 00H. Likewise, the stack pointer will underflow from 00H to 07H.

The end of a subroutine is signaled by a return instruction, either RET or RETR. Each instruction will automatically decrement the Stack Pointer and transfer the contents of the proper RAM register pair to the Program Counter.

PROGRAM STATUS WORD

The 8-bit program status word illustrated in Figure 2-8 is used to store general information about program execution. In addition to the 3-bit Stack Pointer discussed previously, the PSW includes the following flags:

- CY — Carry
- AC — Auxiliary Carry
- F₀ — Flag 0
- BS — Register Bank Select

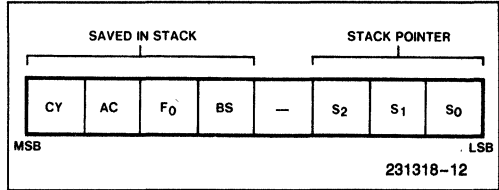


Figure 2-8. Program Status Word

The Program Status Word (PSW) is actually a collection of flip-flops located throughout the machine which are read or written as a whole. The PSW can be loaded to or from the accumulator by the MOV A, PSW or MOV PSW, A instructions. The ability to write directly to the PSW allows easy restoration of machine status after a power-down sequence.

The upper 4 bits of the PSW (bits 4, 5, 6, and 7) are stored in the PC Stack with every subroutine CALL or interrupt vector. Restoring the bits on a return is optional. The bits are restored if an RETR instruction is executed, but not if an RET is executed.

PSW bit definitions are as follows:

- Bits 0–2 Stack Pointer Bits S₀, S₁, S₂
- Bit 3 Not Used
- Bit 4 Working Register Bank
 - 0 = Bank 0
 - 1 = Bank 1
- Bit 5 Flag 0 bit (F₀)
 - This is a general purpose flag which can be cleared or complemented and tested with conditional jump instructions. It may be used during data transfer to an external processor.
- Bit 6 Auxiliary Carry (AC)
 - The flag status is determined by an ADD instruction and is used by the Decimal Adjustment instruction DAA
- Bit 7 Carry (CY)
 - The flag indicates that a previous operation resulted in overflow of the accumulator.

CONDITIONAL BRANCH LOGIC

Conditional Branch Logic in the UPI-41AH, 42AH allows the status of various processor flags, inputs, and other hardware functions to directly affect program execution. The status is sampled in state 3 of the first cycle.

Table 2-2 lists the internal conditions which are testable and indicates the condition which will cause a jump. In all cases, the destination address must be within the page of program memory (256 locations) in which the jump instruction occurs.

OSCILLATOR AND TIMING CIRCUITS

The 8041AH's internal timing generation is controlled by a self-contained oscillator and timing circuit. A choice of crystal, L-C or external clock can be used to derive the basic oscillator frequency.

The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 2-9. Figure 2-10 shows instruction cycle timing.

Oscillator

The on-board oscillator is a series resonant circuit with a frequency range of 1 to 12 (8041AH-2/8042AH/8742AH) MHz. Pins XTAL 1 and XTAL 2 are input

and output (respectively) of a high gain amplifier stage. A crystal or inductor and capacitor connected between XTAL 1 and XTAL 2 provide the feedback and proper phase shift for oscillation. Recommended connections for crystal or L-C are shown in Figure 2-11.

State Counter

The output of the oscillator is divided by 3 in the state counter to generate a signal which defines the state times of the machine.

Each instruction cycle consists of five states as illustrated in Figure 2-10 and Table 2-3. The overlap of address and execution operations illustrated in Figure 2-10 allows fast instruction execution.

Table 2-2. Conditional Branch Instructions

Device	Instruction Mnemonic		Jump Condition Jump if:
Accumulator	JZ	addr	All bits zero
	JNZ	addr	Any bit not zero
Accumulator bit	JBb	addr	Bit "b" = 1
Carry flag	JC	addr	Carry flag = 1
	JNC	addr	Carry flag = 0
User flag	JFO	addr	F ₀ flag = 1
	JF1	addr	F ₁ flag = 1
Timer flag	JTF	addr	Timer flag = 1
Test Input 0	JT0	addr	T ₀ = 1
	JNT0	addr	T ₀ = 0
Test Input 1	JT1	addr	T ₁ = 1
	JNT1	addr	T ₁ = 0
Input Buffer flag	JNIBF	addr	IBF flag = 0
Output Buffer flag	JOBF	addr	OBF flag = 1

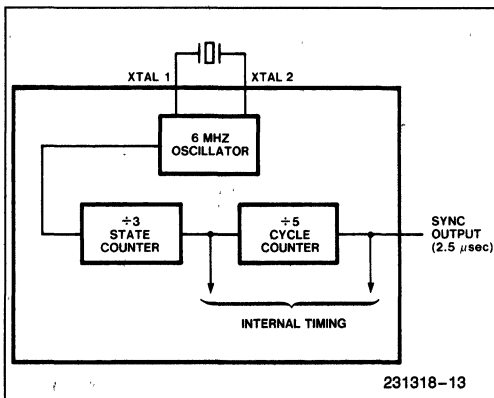


Figure 2-9. Oscillator Configuration

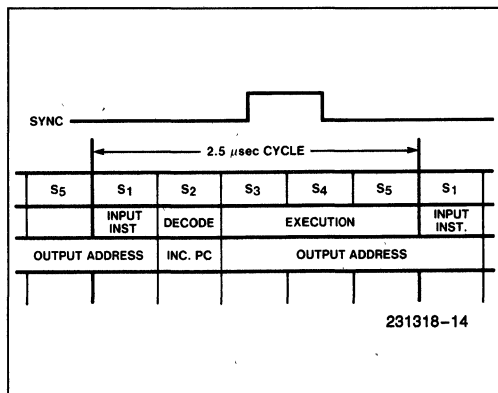


Figure 2-10. Instruction Cycle Timing

Table 2-3. Instruction Timing Diagram

Instruction	CYCLE 1					CYCLE 2				
	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
IN A,Pp	Fetch Instruction	Increment Program Counter	—	Increment Timer	—	—	Read Port	—	—	—
OUTL Pp,A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port	—	—	—	—	—
ANL Pp, DATA	Fetch Instruction	Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	Increment Program Counter	Output To Port	—
ORL Pp, DATA	Fetch Instruction	Increment Program Counter	—	Increment Timer	Read Port	Fetch Immediate Data	—	Increment Program Counter	Output To Port	—
MOVD A,Pp	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	—	—	Read P2 Lower	—	—	—
MOVD Pp, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data To P2 Lower	—	—	—	—	—
D Pp, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	—	—	—
ORLD Pp, A	Fetch Instruction	Increment Program Counter	Output Opcode/Address	Increment Timer	Output Data	—	—	—	—	—
J (Conditional)	Fetch Instruction	Increment Program Counter	Sample Condition	Increment Timer	—	Fetch Immediate Data	—	Update Program Counter	—	—
MOV STS, A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Update Status Register	—	—	—	—	—
IN A, DBB	Fetch Instruction	Increment Program Counter	—	Increment Timer	—	—	—	—	—	—
OUT DBB, A	Fetch Instruction	Increment Program Counter	—	Increment Timer	Output To Port	—	—	—	—	—
STRT T	Fetch Instruction	Increment Program Counter	—	—	Start Counter	—	—	—	—	—
STRT CNT	Fetch Instruction	Increment Program Counter	—	—	Stop Counter	—	—	—	—	—
EN I	Fetch Instruction	Increment Program Counter	—	Enable Interrupt	—	—	—	—	—	—
DIS I	Fetch Instruction	Increment Program Counter	—	Disable Interrupt	—	—	—	—	—	—
EN DMA	Fetch Instruction	Increment Program Counter	—	DMA Enabled DRQ Cleared	—	—	—	—	—	—
EN FLAGS	Fetch Instruction	Increment Program Counter	—	OBF, IBF Output Enabled	—	—	—	—	—	—

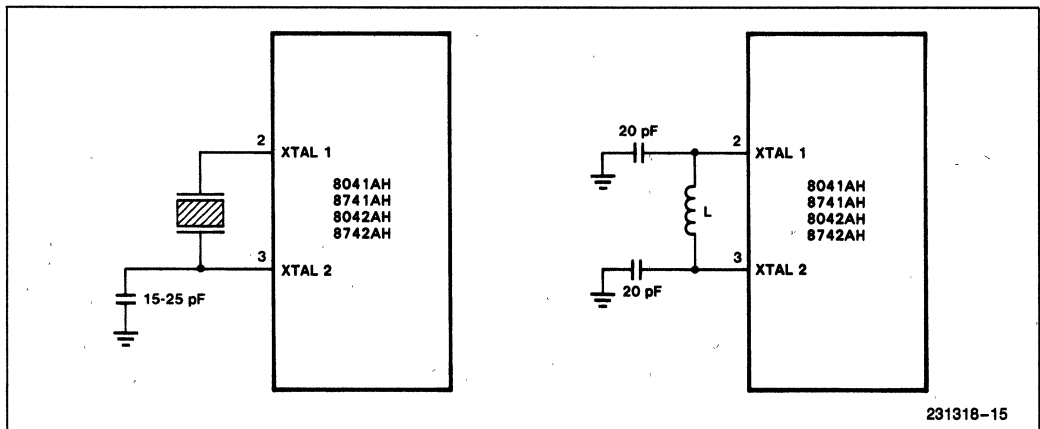


Figure 2-11. Recommended Crystal and L-C Connections

Cycle Counter

The output of the state counter is divided by 5 in the cycle counter to generate a signal which defines a machine cycle. This signal is called SYNC and is available continuously on the SYNC output pin. It can be used to synchronize external circuitry or as a general purpose clock output. It is also used for synchronizing single-step.

Frequency Reference

The external crystal provides high speed and accurate timing generation. A crystal frequency of 5.9904 MHz is useful for generation of standard communication frequencies by the 8041AH/8741AH, 8042AH/8742AH. However, if an accurate frequency reference and maximum processor speed are not required, an inductor and capacitor may be used in place of the crystal as shown in Figure 2-11.

A recommended range of inductance and capacitance combinations is given below:

- $L = 130 \mu\text{H}$ corresponds to 3 MHz
- $L = 45 \mu\text{H}$ corresponds to 5 MHz

An external clock signal can also be used as a frequency reference to the 8741AH, 8741AH, 8742AH or 8042AH; however, the levels are *not* TTL compatible. The signal must be in the 1–12 MHz frequency range and must be connected to pins XTAL 1 and XTAL 2 by buffers with a suitable pull-up resistor to guarantee that a logic "1" is above 3.8 volts. The recommended connection is shown in Figure 2-12.

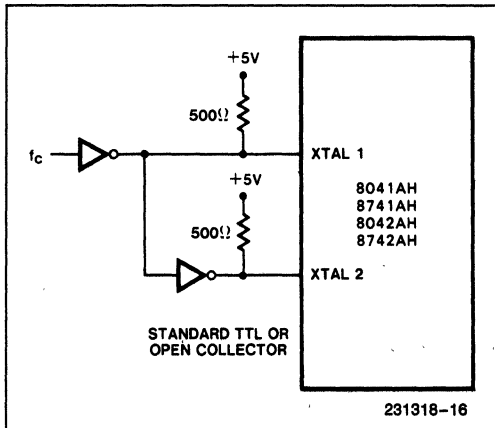


Figure 2-12. Recommended Connection For External Clock Signal

INTERVAL TIMER/EVENT COUNTER

The 8041AH, 8042AH has a resident 8-bit timer/counter which has several software selectable modes of operation. As an interval timer, it can generate accurate delays from 80 microseconds to 20.48 milliseconds without placing undue burden on the processor. In the counter mode, external events such as switch closures or tachometer pulses can be counted and used to direct program flow.

Timer Configuration

Figure 2-13 illustrates the basic timer/counter configuration. An 8-bit register is used to count pulses from either the internal clock and prescaler or from an external source. The counter is presettable and readable with two MOV instructions which transfer the contents of the accumulator to the counter and vice-versa (i.e. MOV T, A and MOV A, T). The counter is stopped by a RESET or STOP TCNT instruction and remains stopped until restarted either as a timer (START T instruction) or as a counter (START CNT instruction). Once started, the counter will increment to its maximum count (FFH) and overflow to zero continuing its count until stopped by a STOP TCNT instruction or RESET.

The increment from maximum count to zero (overflow) results in setting the Timer Flag (TF) and generating an interrupt request. The state of the overflow flag is testable with the conditional jump instruction, JTF. The flag is reset by executing a JTF or by a RESET signal.

The timer interrupt request is stored in a latch and ORed with the input buffer full interrupt request. The timer interrupt can be enabled or disabled independent of the IBF interrupt by the EN TCNTI and DIS TCTNI instructions. If enabled, the counter overflow will cause a subroutine call to location 7 where the timer service routine is stored. If the timer and Input Buffer Full interrupts occur simultaneously, the IBF source will be recognized and the call will be to location 3. Since the timer interrupt is latched, it will remain pending until the DBBIN register has been serviced and will immediately be recognized upon return from the service routine. A pending timer interrupt is reset by the initiation of a timer interrupt service routine.

Event Counter Mode

The STRT CNT instruction connects the TEST 1 input pin to the counter input and enables the counter. Note this instruction does not clear the counter. The counter is incremented on high to low transitions of the TEST 1 input. The TEST 1 input must remain high for a minimum of one state in order to be registered (250 ns at 12 MHz). The maximum count frequency is one count per three instruction cycles (267 kHz at 12 MHz). There is no minimum frequency limit.

Timer Mode

The **STRT T** instruction connects the internal clock to the counter input and enables the counter. The input clock is derived from the **SYNC** signal of the internal oscillator and the divide-by-32 prescaler. The configuration is illustrated in Figure 2-13. Note this instruction does not clear the timer register. Various delays and timing sequences between 40 μ sec and 10.24 msec can easily be generated with a minimum of software timing loops (at 12 MHz).

Times longer than 10.24 msec can be accurately measured by accumulating multiple overflows in a register under software control. For time resolution less than 40 μ sec, an external clock can be applied to the **TEST 1** counter input (see Event Counter Mode). The minimum time resolution with an external clock is 3.75 μ sec (267 kHz at 12 MHz).

TEST 1 Event Counter Input

The **TEST 1** pin is multifunctional. It is automatically initialized as a test input by a **RESET** signal and can be tested using UPI-41A conditional branch instructions.

In the second mode of operation, illustrated in Figure 2-13, the **TEST 1** pin is used as an input to the internal

8-bit event counter. The **Start Counter (STRT CNT)** instruction controls an internal switch which connects **TEST 1** through an edge detector to the 8-bit internal counter. Note that this instruction does not inhibit the testing of **TEST 1** via conditional Jump instructions.

In the counter mode the **TEST 1** input is sampled once per instruction cycle. After a high level is detected, the next occurrence of a low level at **TEST 1** will cause the counter to increment by one.

The event counter functions can be stopped by the **Stop Timer/Counter (STOP TCNT)** instruction. When this instruction is executed the **TEST 1** pin becomes a test input and functions as previously described.

TEST INPUTS

There are two multifunction pins designated as Test Inputs, **TEST 0** and **TEST 1**. In the normal mode of operation, status of each of these lines can be directly tested using the following conditional Jump instructions:

- **JT0** Jump if **TEST 0** = 1
- **JNT0** Jump if **TEST 0** = 0
- **JT1** Jump if **TEST 1** = 1
- **JNT1** Jump if **TEST 1** = 0

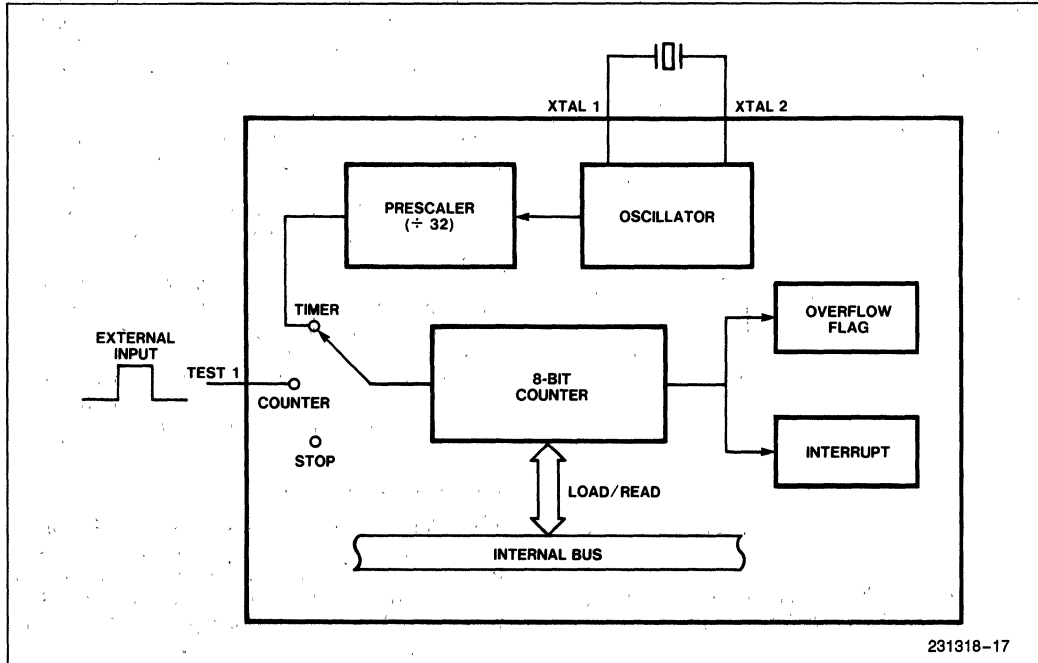


Figure 2-13. Timer Counter

The test inputs are TTL compatible. An external logic signal connected to one of the test inputs will be sampled at the time the appropriate conditional jump instruction is executed. The path of program execution will be altered depending on the state of the external signal when sampled.

INTERRUPTS

The 8041AH/8741AH, 8042AH/8742AH has the following internal interrupts:

- Input Buffer Full (IBF) interrupt
- Timer Overflow interrupt

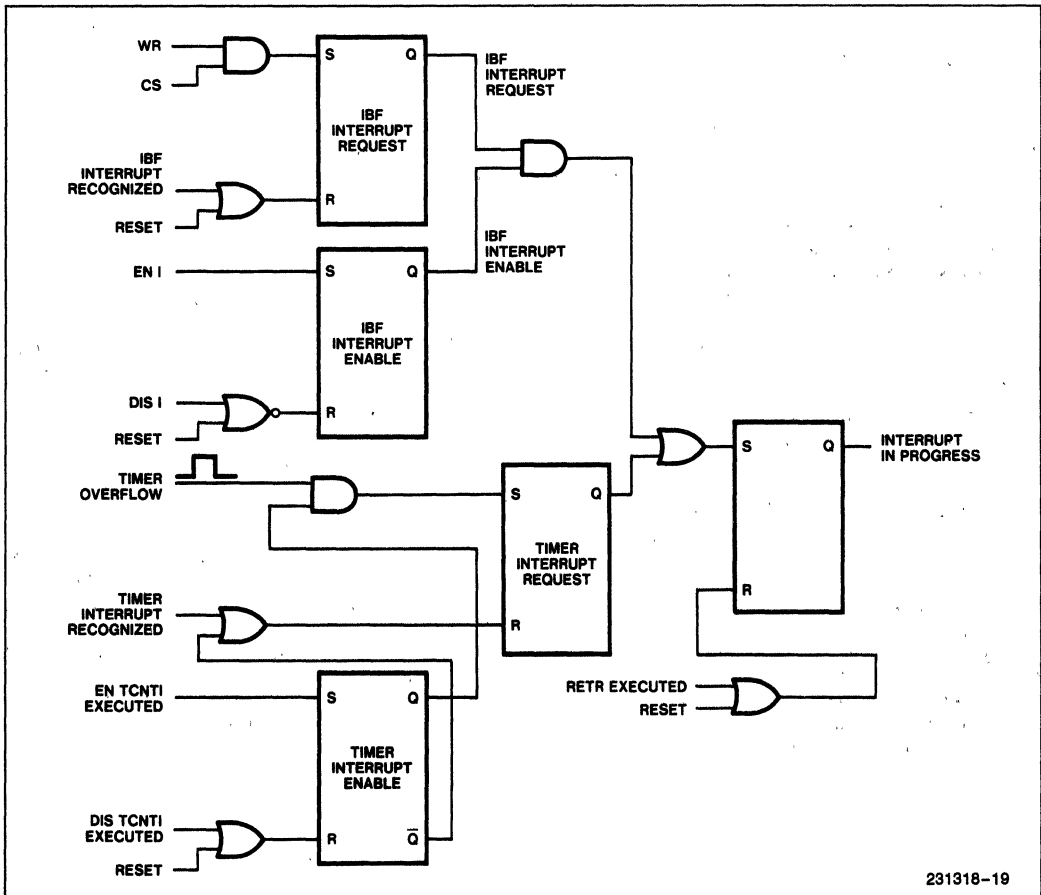
The IBF interrupt forces a CALL to location 3 in program memory; a timer-overflow interrupts forces a CALL to location 7. The IBF interrupt is enabled by the EN I instruction and disabled by the DIS I instruction. The timer-overflow interrupt is enabled and disabled by the EN TNCTI and DIS TCNTI instructions, respectively.

Figure 2-14 illustrates the internal interrupt logic. An IBF interrupt request is generated whenever WR and CS are both low, regardless of whether interrupts are enabled. The interrupt request is cleared upon entering the IBF service routine only. That is, the DIS I instruction does not clear a pending IBF interrupt.

Interrupt Timing Latency

When the IBF interrupt is enabled and an IBF interrupt request occurs, an interrupt sequence is initiated as soon as the currently executing instruction is completed. The following sequence occurs:

- A CALL to location 3 is forced.
- The program counter and bits 4-7 of the Program Status Word are stored in the stack.
- The stack pointer is incremented.



231318-19

Figure 2-14. Interrupt Logic

Location 3 in program memory should contain an unconditional jump to the beginning of the IBF interrupt service routine elsewhere in program memory. At the end of the service routine, an RETR (Return and Restore Status) instruction is used to return control to the main program. This instruction will restore the program counter and PSW bits 4–7, providing automatic restoration of the previously active register bank as well. RETR also re-enables interrupts.

A timer-overflow interrupt is enabled by the EN TCNTI instruction and disabled by the DIS TCNTI instruction. If enabled, this interrupt occurs when the timer/counter register overflows. A CALL to location 7 is forced and the interrupt routine proceeds as described above.

The interrupt service latency is the sum of current instruction time, interrupt recognition time, and the internal call to the interrupt vector address. The worst case latency time for servicing an interrupt is 7 clock cycles. Best case latency is 4 clock cycles.

Interrupt Timing

Interrupt inputs may be enabled or disabled under program control using EN I, DIS I, EN TCNTI and DIS TCNTI instructions. Also, a $\overline{\text{RESET}}$ input will disable interrupts. An interrupt request must be removed before the RETR instruction is executed to return from the service routine, otherwise the processor will re-enter the service routine immediately. Thus, the $\overline{\text{WR}}$ and $\overline{\text{CS}}$ inputs should not be held low longer than the duration of the interrupt service routine.

The interrupt system is single level. Once an interrupt is detected, all further interrupt requests are latched but are not acted upon until execution of an RETR instruction re-enables the interrupt input logic. This occurs at the beginning of the second cycle of the RETR instruction. If an IBF interrupt and a timer-overflow interrupt occur simultaneously, the IBF interrupt will be recognized first and the timer-overflow interrupt will remain pending until the end of the interrupt service routine.

External Interrupts

An external interrupt can be created using the UPI-41AH, 42AH timer/counter in the event counter mode. The counter is first preset to FFH and the EN TCNTI instruction is executed. A timer-overflow interrupt is generated by the first high to low transition of the

TEST 1 input pin. Also, if an IBF interrupt occurs during servicing of the timer/counter interrupt, it will remain pending until the end of the service routine.

Host Interrupts And DMA

If needed, two external interrupts to the host system can be created using the EN FLAGS instruction. This instruction allocates two I/O lines on PORT 2 (P₂₄ and P₂₅). P₂₄ is the Output Buffer Full interrupt request line to the host system; P₂₅ is the Input Buffer empty interrupt request line. These interrupt outputs reflect the internal status of the OBF flag and the IBF inverted flag. Note, these outputs may be inhibited by writing a "0" to these pins. Reenabling interrupts is done by writing a "1" to these port pins. Interrupts are typically enabled after power on since the I/O ports are set in a "1" condition. The EN FLAG's effect is only cancelled by a device RESET.

DMA handshaking controls are available from two pins on PORT 2 of the UPI-41AH microcomputer. These lines (P₂₆ and P₂₇) are enabled by the EN DMA instruction. P₂₆ becomes DMA request (DRQ) and P₂₇ becomes DMA acknowledge (DACK). The UPI program initiates a DMA request by writing a "1" to P₂₆. The DMA controller transfers the data into the DBBIN data register using DACK which acts as a chip select. The EN DMA instruction can only be cancelled by a chip RESET.

RESET

The $\overline{\text{RESET}}$ input provides a means for internal initialization of the processor. An automatic initialization pulse can be generated at power-on by simply connecting a 1 μfd capacitor between the $\overline{\text{RESET}}$ input and ground as shown in Figure 2-15. It has an internal pull-up resistor to charge the capacitor and a Schmitt-trigger circuit to generate a clean transition. A 2-stage synchronizer has been added to support reliable operation up to 12 MHz.

If automatic initialization is used, $\overline{\text{RESET}}$ should be held low for at least 10 milliseconds to allow the power supply to stabilize. If an external RESET signal is used, $\overline{\text{RESET}}$ may be held low for a minimum of 8 instruction cycles. Figure 2-15 illustrates a configuration using an external TTL gate to generate the $\overline{\text{RESET}}$ input. This configuration can be used to derive the $\overline{\text{RESET}}$ signal from the 8224 clock generator in an 8080 system.

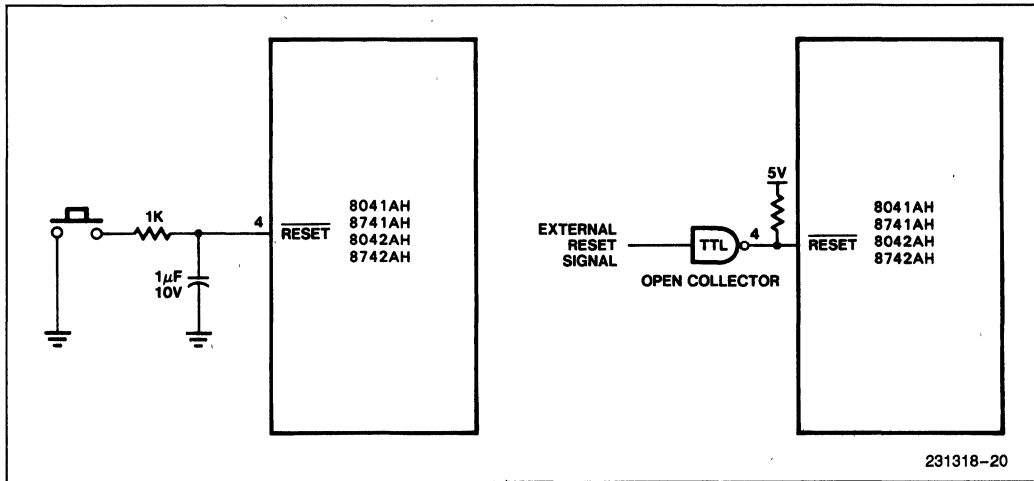


Figure 2-15. External Reset Configuration

The $\overline{\text{RESET}}$ input performs the following functions:

- Disables Interrupts
- Clears Program Counter to Zero
- Clears Stack Pointer
- Clears Status Register and Flags
- Clears Timer and Timer Flag
- Stops Timer
- Selects Register Bank 0
- Sets PORTS 1 and 2 to Input Mode

DATA BUS BUFFER

Two 8-bit data bus buffer registers, $\overline{\text{DBBIN}}$ and $\overline{\text{DBBOUT}}$, serve as temporary buffers for commands and data flowing between it and the master processor. Externally, data is transmitted or received by the $\overline{\text{DBB}}$ registers upon execution of an $\overline{\text{IN}}\text{PUT}$ or $\overline{\text{OUT}}\text{PUT}$ instruction by the master processor. Four control signals are used:

- A_0 Address input signifying control or data
- $\overline{\text{CS}}$ Chip Select
- $\overline{\text{RD}}$ Read Strobe
- $\overline{\text{WR}}$ Write Strobe

Transfer can be implemented with or without UPI program interference by enabling or disabling an internal UPI interrupt. Internally, data transfer between the $\overline{\text{DBB}}$ and the UPI accumulator is under software con-

trol and is completely asynchronous to the external processor timing. This allows the UPI software to handle peripheral control tasks independent of the main processor while still maintaining a data interface with the master system.

Configuration

Figure 2-16 illustrates the internal configuration of the $\overline{\text{DBB}}$ registers. Data is stored in two 8-bit buffer registers, $\overline{\text{DBBIN}}$ and $\overline{\text{DBBOUT}}$. $\overline{\text{DBBIN}}$ and $\overline{\text{DBBOUT}}$ may be accessed by the external processor using the $\overline{\text{WR}}$ line and the $\overline{\text{RD}}$ line, respectively. The data bus is a bidirectional, three-state bus which can be connected directly to an 8-bit microprocessor system. Four control lines ($\overline{\text{WR}}$, $\overline{\text{RD}}$, $\overline{\text{CS}}$, A_0) are used by the external processor to transfer data to and from the $\overline{\text{DBBIN}}$ and $\overline{\text{DBBOUT}}$ registers.

An 8-bit register containing status flags is used to indicate the status of the $\overline{\text{DBB}}$ registers. The eight status flags are defined as follows:

- **OB $\overline{\text{F}}$ Output Buffer Full**
This flag is automatically set when the UPI-Microcomputer loads the $\overline{\text{DBBOUT}}$ register and is cleared when the master processor reads the data register.
- **IB $\overline{\text{F}}$ Input Buffer Full**
This flag is set when the master processor writes a character to the $\overline{\text{DBBIN}}$ register and is cleared when the UPI $\overline{\text{IN}}\text{PUT}$ s the data register contents to its accumulator.

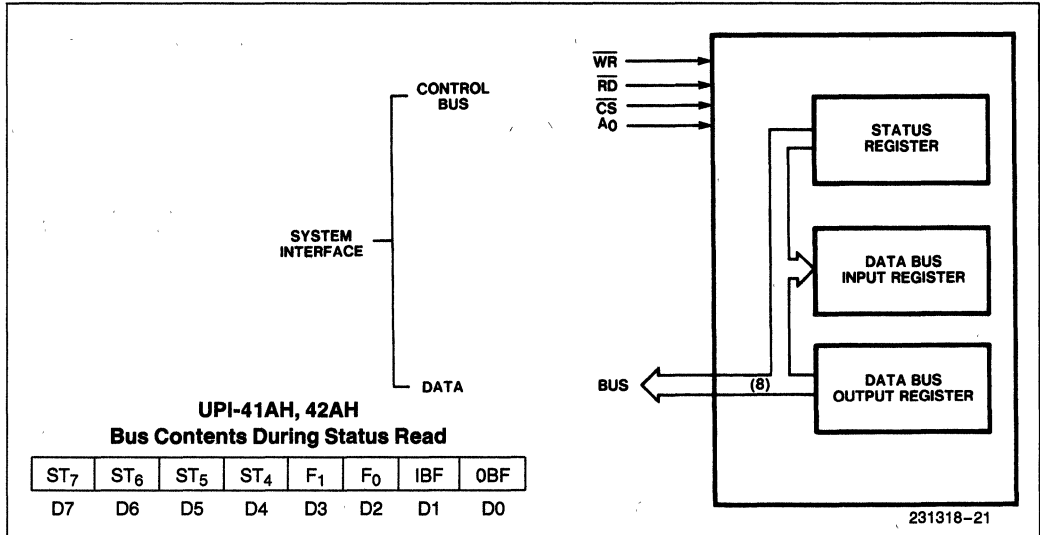


Figure 2-16. Data Bus Buffer Configuration

- **F₀**
This is a general purpose flag which can be cleared or toggled under UPI software control. The flag is used to transfer UPI status information to the master processor.
- **F₁ Command/Data**
This flag is set to the condition of the A₀ input line when the master processor writes a character to the data register. The F₁ flag can also be cleared or toggled under UPI-Microcomputer program control.
- **ST₄ through ST₇**
These bits are user defined status bits. They are defined by the MOV STS,A instruction.

SYSTEM INTERFACE

Figure 2-17 illustrates how a UPI-Microcomputer can be connected to a standard 8080-type bus system. Data lines D₀-D₇ form a three-state, bidirectional port which can be connected directly to the system data bus. The UPI bus interface has sufficient drive capability (400 μA) for small systems, however, a larger system may require buffers.

Four control signals are required to handle the data and status information transfer:

- **\overline{WR}**
I/O WRITE signal used to transfer data from the system bus to the UPI DBBIN register and set the F₁ flag in the status register.
- **\overline{RD}**
I/O READ signal used to transfer data from the DBBOUT register or status register to the system data bus.

- **\overline{CS}**
CHIP SELECT signal used to enable one 8041AH out of several connected to a common bus.
- **$\overline{A_0}$**
Address input used to select either the 8-bit status register or DBBOUT register during an I/O READ. Also, the signal is used to set the F₁ flag in the status register during an I/O WRITE.

The \overline{WR} and \overline{RD} signals are active low and are standard MCS-80 peripheral control signals used to synchronize data transfer between the system bus and peripheral devices.

The \overline{CS} and A₀ signals are decoded from the address bus of the master system. In a system with few I/O devices a linear addressing configuration can be used where A₀ and A₁ lines are connected directly to A₀ and \overline{CS} inputs (see Figure 2-17).

Data Read

Table 2-4 illustrates the relative timing of a DBBOUT Read. When \overline{CS} , A₀, and \overline{RD} are low, the contents of the DBBOUT register is placed on the three-state Data lines D₀-D₇ and the OBF flag is cleared.

The master processor uses \overline{CS} , A₀, \overline{WR} , and \overline{RD} to control data transfer between the DBBOUT register and the master system. The following operations are under master processor control:

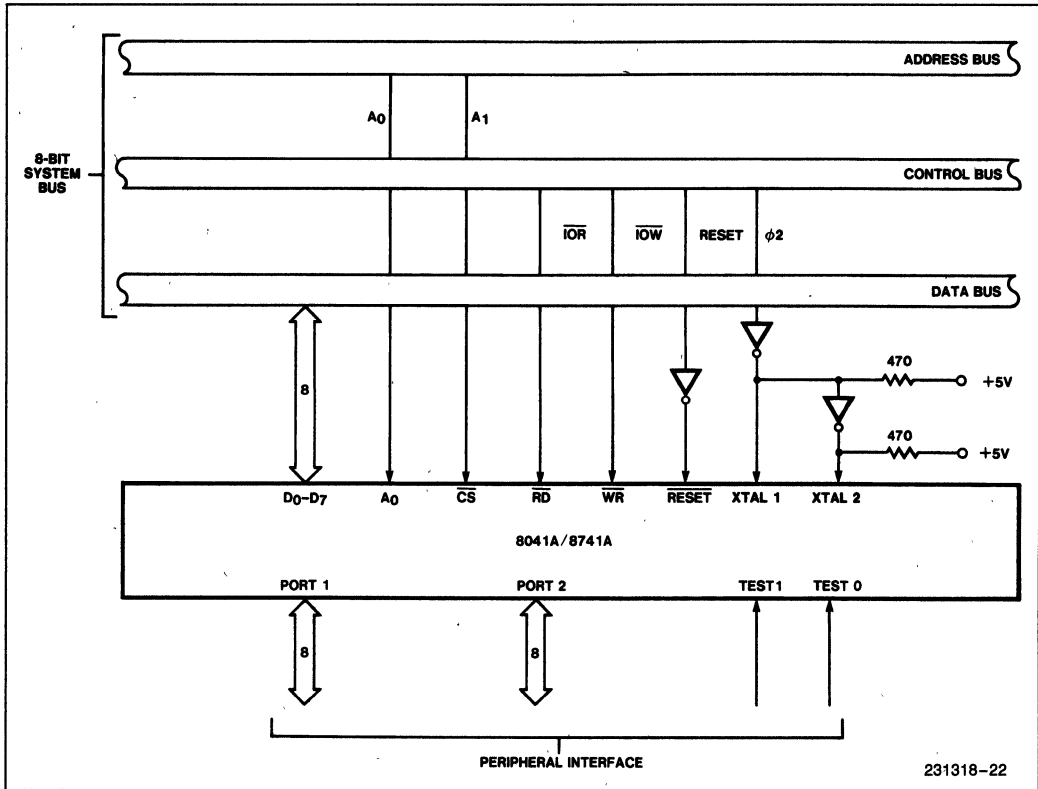


Figure 2-17. Interface to 8080 System Bus

Table 2-4. Data Transfer Controls

CS	RD	WR	A ₀	
0	0	1	0	Read DBBOUT register
0	0	1	1	Read STATUS register
0	1	0	0	Write DBBIN data register
0	1	0	1	Write DBBIN command register
1	x	x	x	Disable DBB

Status Read

Table 2-4 shows the logic sequence required for a STATUS register read. When CS and RD are low with A₀ high, the contents of the 8-bit status register appears on Data lines D₀-D₇.

Data Write

Table 2-4 shows the sequence for writing information to the DBBIN register. When CS and WR are low, the contents of the system data bus is latched into DBBIN. Also, the IBF flag is set and an interrupt is generated, if enabled.

Command Write

During any write (Table 2-4), the state of the A₀ input is latched into the status register in the F₁ (command/data) flag location. This additional bit is used to signal whether DBBIN contents are command (A₀ = 1) or data (A₀ = 0) information.

INPUT/OUTPUT INTERFACE

The UPI-41A has 16 lines for input and output functions. These I/O lines are grouped as two 8-bit TTL compatible ports: PORTS 1 and 2. The port lines can individually function as either inputs or outputs under software control. In addition, the lower 4 lines of PORT 2 can be used to interface to an 8243 I/O expander device to increase I/O capacity to 28 or more lines. The additional lines are grouped as 4-bit ports: PORTS 4, 5, 6, and 7.

PORTS 1 and 2

PORTS 1 and 2 are each 8 bits wide and have the same I/O characteristics. Data written to these ports by an

OUTL Pp,A instruction is latched and remains unchanged until it is rewritten. Input data is sampled at the time the IN, A, Pp instruction is executed. Therefore, input data must be present at the PORT until read by an INput instruction. PORT 1 and 2 inputs are fully TTL compatible and outputs will drive one standard TTL load.

Circuit Configuration

The PORT 1 and 2 lines have a special output structure (shown in Figure 2-18) that allows each line to serve as an input, an output, or both, even though outputs are statically latched.

Each line has a permanent high impedance pull-up (50 K Ω) which is sufficient to provide source current for a TTL high level, yet can be pulled low by a standard TTL gate drive. Whenever a "1" is written to a line, a low impedance pull-up (5K) is switched in momentarily (500 ns) to provide a fast transition from 0 to 1. When a "0" is written to the line, a low impedance pull-down (300 Ω) is active to provide TTL current sinking capability.

To use a particular PORT pin as an input, a logic "1" must first be written to that pin.

NOTE:

A RESET initializes all PORT pins to the high impedance logic "1" state.

An external TTL device connected to the pin has sufficient current sinking capability to pull-down the pin to the low state. An IN A, Pp instruction will sample the status of PORT pin and will input the proper logic level. With no external input connected, the IN A,Pp instruction inputs the previous output status.

This structure allows input and output information on the same pin and also allows any mix of input and output lines on the same port. However, when inputs and outputs are mixed on one PORT, a PORT write will cause the strong internal pull-ups to turn on at all inputs. If a switch or other low impedance device is connected to an input, a PORT write ("1" to an input) could cause current limits on internal lines to be exceeded. Figure 2-19 illustrates the recommended connection when inputs and outputs are mixed on one PORT.

The bidirectional port structure in combination with the UPI-41AH, 42AH logical AND and OR instructions provide an efficient means for handling single line inputs and outputs within an 8-bit processor.

PORTS 4, 5, 6, and 7

By using an 8243 I/O expander, 16 additional I/O lines can be connected to the UPI-41AH, 42AH and directly addressed as 4-bit I/O ports using UPI-41AH, 42AH

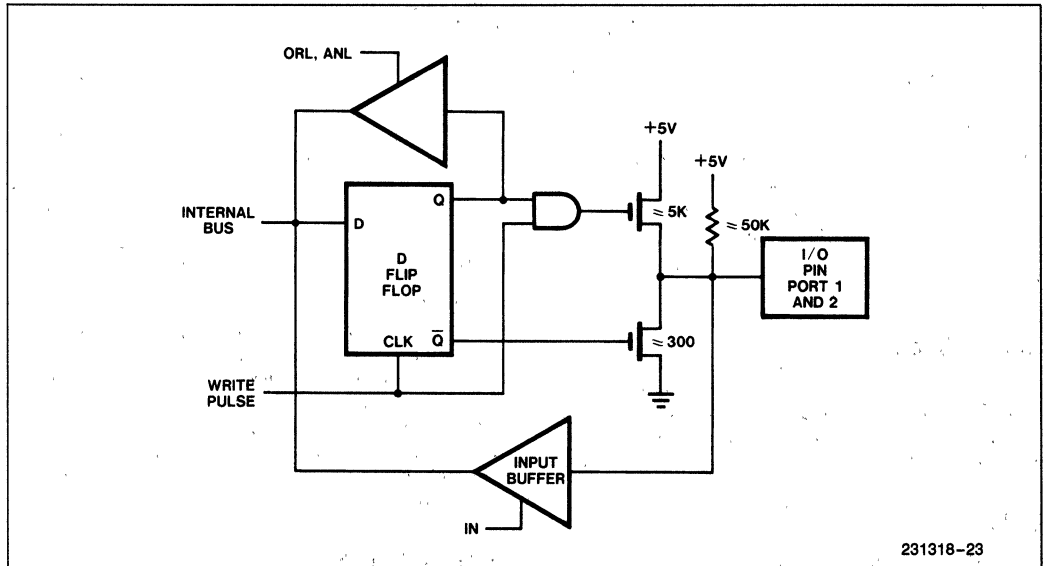


Figure 2-18. Quasi-Bidirectional Port Structure

231318-23

instructions. This feature saves program space and design time, and improves the bit handling capability of the UPI-41AH, 42AH.

The lower half of PORT 2 provides an interface to the 8243 as illustrated in Figure 2-20. The PROG pin is used as a strobe to clock address and data information via the PORT 2 interface. The extra 16 I/O lines are referred to in UPI software as PORTS 4, 5, 6, and 7. Each PORT can be directly addressed and can be ANDed and ORed with an immediate data mask. Data can be moved directly to the accumulator from the expander PORTS (or vice-versa).

The 8243 I/O ports, PORTS 4, 5, 6, and 7, provide more drive capability than the UPI-41AH, 42AH bidi-

rectional ports. The 8243 output is capable of driving about 5 standard TTL loads.

Multiple 8243's can be connected to the PORT 2 interface. In normal operation, only one of the 8243's would be active at the time an Input or Output command is executed. The upper half of PORT 2 is used to provide chip select signals to the 8043's. Figure 2-21 shows how four 8243's could be connected. Software is needed to select and set the proper PORT 2 pin before an INPUT or OUTPUT command to PORTS 4-7 is executed. In general, the software overhead required is very minor compared to the added flexibility of having a large number of I/O pins available.

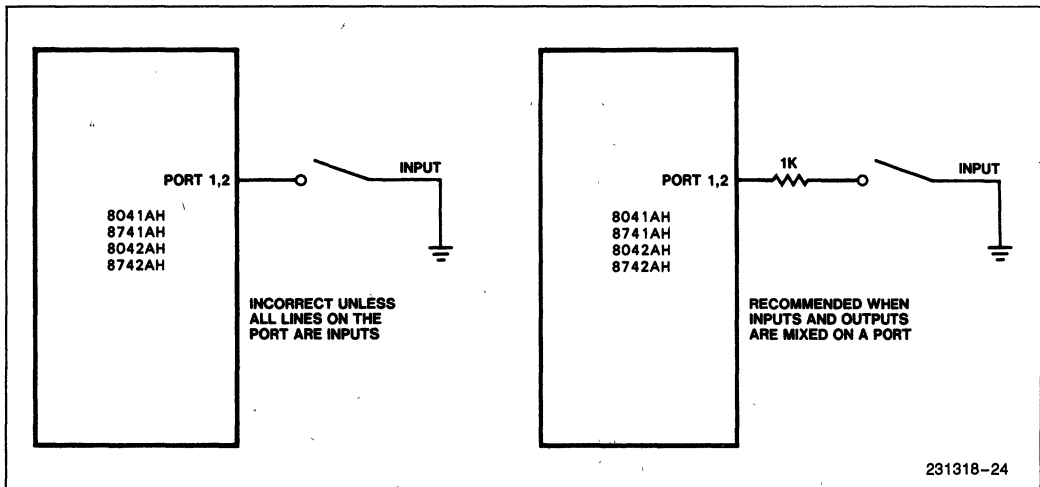


Figure 2-19. Recommended PORT Input Connections

CHAPTER 3 INSTRUCTION SET

The UPI-41AH, 42AH Instruction Set is opcode-compatible with the MCS-48 set except for the elimination of external program and data memory instructions and the addition of the data bus buffer instructions. It is very straightforward and efficient in its use of program memory. All instructions are either 1 or 2 bytes in length (over 70% are only 1 byte long) and over half of the instructions execute in one machine cycle. The remainder require only two cycles and include Branch, Immediate, and I/O operations.

The UPI-41AH, 42AH Instruction Set efficiently handles the single-bit operations required in control applications. Special instructions allow port bits to be set or cleared individually. Also, any accumulator bit can be directly tested via conditional branch instructions. Additional instructions are included to simplify loop counters, table look-up routines and N-way branch routines.

The UPI-41AH, 42AH Microcomputer handles arithmetic operations in both binary and BCD for efficient interface to peripherals such as keyboards and displays.

The instruction set can be divided into the following groups:

- Data Moves
- Accumulator Operations
- Flags
- Register Operations
- Branch Instructions
- Control
- Timer Operations
- Subroutines
- Input/Output Instructions

Data Moves (See Instruction Summary)

The 8-bit accumulator is the control point for all data transfers within the UPI-41AH, 42AH. Data can be transferred between the 8 registers of each working register bank and the accumulator directly (i.e., with a source or destination register specified by 3 bits in the instruction). The remaining locations in the RAM array are addressed either by R_0 or R_1 of the active register bank. Transfers to and from RAM require one cycle.

Constants stored in Program Memory can be loaded directly into the accumulator or the eight working registers. Data can also be transferred directly between the accumulator and the on-board timer/counter, the

Status Register (STS), or the Program Status Word (PSW). Transfers to the STS register alter bits 4–7 only. Transfers to the PSW alter machine status accordingly and provide a means of restoring status after an interrupt or of altering the stack pointer if necessary.

Accumulator Operations

Immediate data, data memory, or the working registers can be added (with or without carry) to the accumulator. These sources can also be ANDed, ORed, or exclusive ORed to the accumulator. Data may be moved to or from the accumulator and working registers or data memory. The two values can also be exchanged in a single operation.

The lower 4 bits of the accumulator can be exchanged with the lower 4 bits of any of the internal RAM locations. This operation, along with an instruction which swaps the upper and lower 4-bit halves of the accumulator, provides easy handling of BCD numbers and other 4-bit quantities. To facilitate BCD arithmetic a Decimal Adjust instruction is also included. This instruction is used to correct the result of the binary addition of two 2-digit BCD numbers. Performing a decimal adjust on the result in the accumulator produces the desired BCD result.

The accumulator can be incremented, decremented, cleared, or complemented and can be rotated left or right 1 bit at a time with or without carry.

A subtract operation can be easily implemented in UPI-41AH, 42AH software using three single-byte, single-cycle instructions. A value can be subtracted from the accumulator by using the following instructions:

- Complement the accumulator
- Add the value to the accumulator
- Complement the accumulator

Flags

There are four user accessible flags:

- Carry
- Auxiliary Carry
- F_0
- F_1

The Carry flag indicates overflow of the accumulator, while the Auxiliary Carry flag indicates overflow between BCD digits and is used during decimal adjust

operations. Both Carry and Auxiliary Carry are part of the Program Status Word (PSW) and are stored in the stack during subroutine calls. The F_0 and F_1 flags are general-purpose flags which can be cleared or complemented by UPI instructions. F_0 is accessible via the Program Status Word and is stored in the stack with the Carry flags. F_1 reflects the condition of the A_0 line, and caution must be used when setting or clearing it.

Register Operations

The working registers can be accessed via the accumulator as explained above, or they can be loaded with immediate data constants from program memory. In addition, they can be incremented or decremented directly, or they can be used as loop counters as explained in the section on branch instructions.

Additional Data Memory locations can be accessed with indirect instructions via R_0 and R_1 .

Branch Instructions

The UPI-41AH, 42AH Instruction Set includes 17 jump instructions. The unconditional allows jumps anywhere in the 1K words of program memory. All other jump instructions are limited to the current page (256 words) of program memory.

Conditional jump instructions can test the following inputs and matching flags:

- TEST 0 input pin
- TEST 1 input pin
- Input Buffer Full flag
- Output Buffer Full flag
- Timer flag
- Accumulator zero
- Accumulator bit
- Carry flag
- F_0 flag
- F_1 flag

The conditions tested by these instructions are the instantaneous values at the time the conditional jump instruction is executed. For instance, the jump on accumulator zero instruction tests the accumulator itself, not an intermediate flag.

The decrement register and jump if not zero (DJNZ) instruction combines decrement and branch operations

in a single instruction which is useful in implementing a loop counter. This instruction can designate any of the 8 working registers as a counter and can effect a branch to any address within the current page of execution.

A special indirect jump instruction (JMPP @A) allows the program to be vectored to any one of several different locations based on the contents of the accumulator. The contents of the accumulator point to a location in program memory which contains the jump address. As an example, this instruction could be used to vector to any one of several routines based on an ASCII character which has been loaded into the accumulator. In this way, ASCII inputs can be used to initiate various routines.

Control

The UPI-41AH, 42AH Instruction Set has six instructions for control of the DMA, interrupts, and selection of working registers banks.

The UPI-41AH, 42AH provides two instructions for control of the external microcomputer system. IBF and OBF flags can be routed to PORT 2 allowing interrupts of the external processor. DMA handshaking signals can also be enabled using lines from PORT 2.

The IBF interrupt can be enabled and disabled using two instructions. Also, the interrupt is automatically disabled following a RESET input or during an interrupt service routine.

The working register bank switch instructions allow the programmer to immediately substitute a second 8 register bank for the one in use. This effectively provides either 16 working registers or the means for quickly saving the contents of the first 8 registers in response to an interrupt. The user has the option of switching register banks when an interrupt occurs. However, if the banks are switched, the original bank will automatically be restored upon execution of a return and restore status (RETR) instruction at the end of the interrupt service routine.

Timer

The 8-bit on-board timer/counter can be loaded or read via the accumulator while the counter is stopped or while counting.

The counter can be started as a timer with an internal clock source or as an event counter or timer with an

external clock applied to the TEST 1 pin. The instruction executed determines which clock source is used. A single instruction stops the counter whether it is operating with an internal or an external clock source. In addition, two instructions allow the timer interrupt to be enabled or disabled.

Subroutines

Subroutines are entered by executing a call instruction. Calls can be made to any address in the 1K word program memory. Two separate return instructions determine whether or not status (i.e., the upper 4 bits of the PSW) is restored upon return from a subroutine.

Input/Output Instructions

Two 8-bit data bus buffer registers (DBBIN and DBBOUT) and an 8-bit status register (STS) enable the UPI-41A universal peripheral interface to communicate with the external microcomputer system. Data can be INputted from the DBBIN register to the accumulator. Data can be OUTputted from the accumulator to the DBBOUT register.

The STS register contains four user-definable bits (ST₄-ST₇) plus four reserved status bits (IBF, OBF, F₀ and F₁). The user-definable bits are set from the accumulator.

The UPI-41AH, 42AH peripheral interface has two 8-bit static I/O ports which can be loaded to and from the accumulator. Outputs are statically latched but inputs to the ports are sampled at the time an IN instruction is executed. In addition, immediate data from program memory can be ANDed and ORed directly to PORTS 1 and 2 with the result remaining on the port. This allows "masks" stored in program memory to be used to set or reset individual bits on the I/O ports. PORTS 1 and 2 are configured to allow input on a given pin by first writing a "1" to the pin.

Four additional 4-bit ports are available through the 8243 I/O expander device. The 8243 interfaces to the

UPI-41AH, 42AH peripheral interface via four PORT 2 lines which form an expander bus. The 8243 ports have their own AND and OR instructions like the on-board ports, as well as move instructions to transfer data in or out. The expander AND or OR instructions, however, combine the contents of the accumulator with the selected port rather than with immediate data as is done with the on-board ports.

INSTRUCTION SET DESCRIPTION

The following section provides a detailed description of each UPI instruction and illustrates how the instructions are used.

For further information about programming the UPI, consult the *8048/8041AH Assembly Language Manual*.

Table 3-1. Symbols and Abbreviations Used

Symbol	Definition
A	Accumulator
C	Carry
DBBIN	Data Bus Buffer Input
DBBOUT	Data Bus Buffer Output
F ₀ , F ₁	FLAG 0, FLAG 1 (C/D flag)
I	Interrupt
P	Mnemonic for "in-page" operation
PC	Program Counter
Pp	Port designator (p = 1, 2, or 4-7)
PSW	Program Status Word
Rr	Register designator (r = 0-7)
SP	Stack Pointer
STS	Status register
T	Timer
TF	Timer Flag
T ₀ , T ₁	TEST 0, TEST 1
#	Immediate data prefix
@	Indirect address prefix
(())	Double parentheses show the effect of @, that is @RO is shown as ((RO)).
()	Contents of

Table 3-2. Instruction Set Summary

Mnemonic	Description	Bytes	Cycle	Mnemonic	Description	Bytes	Cycle
ACCUMULATOR				DATA MOVES (Continued)			
ADD A, Rr	Add register to A	1	1	MOVP A, @A	Move to A from current page	1	2
ADD A, @Rr	Add data memory to A	1	1	MOVP3 A, @A	Move to A from page 3	1	2
ADD A, #data	Add immediate to A	2	2	TIMER/COUNTER			
ADDC A, Rr	Add register to A with carry	1	1	MOV A, T	Read Timer/Counter	1	1
ADDC A, @Rr	Add data memory to A with carry	1	1	MOV T, A	Load Timer/Counter	1	1
ADDC A, #data	Add immediate to A with carry	2	2	STRT T	Start Timer	1	1
ANL A, Rr	And register to A	1	1	STRT CNT	Start Counter	1	1
ANL A, @Rr	And data memory to A	1	1	STOP TCNT	Stop Timer/Counter	1	1
ANL A, #data	And immediate to A	2	2	EN TCNTI	Enable Timer/Counter	1	1
ORL A, Rr	Or register to A	1	1	DIS TCNTI	Disable Timer/Counter	1	1
ORL A, @Rr	Or data memory to A	1	1	CONTROL			
ORL A, #data	Or immediate to A	2	2	EN DMA	Enable DMA Handshake Lines	1	1
XRL A, Rr	Exclusive Or register to A	1	1	EN I	Enable IBF interrupt	1	1
XRL A, @Rr	Exclusive Or data memory to A	1	1	DIS I	Disable IBF interrupt	1	1
XRL A, #data	Exclusive Or immediate to A	2	2	EN FLAGS	Enable Master Interrupts	1	1
INC A	Increment A	1	1	SEL RB0	Select register bank 0	1	1
DEC A	Decrement A	1	1	SEL RB1	Select register bank 1	1	1
CLR A	Clear A	1	1	NOP	No Operation	1	1
CPL A	Complement A	1	1	REGISTERS			
DA A	Decimal Adjust A	1	1	INC Rr	Increment register	1	1
SWAP A	Swap nibbles of A	1	1	INC @Rr	Increment data memory	1	1
RL A	Rotate A left	1	1	DEC Rr	Decrement register	1	1
RLC A	Rotate A left through carry	1	1	SUBROUTINE			
RR A	Rotate A right	1	1	CALL addr	Jump to subroutine	2	2
RRC A	Rotate A right through carry	1	1	RET	Return	1	2
INPUT/OUTPUT				RETR	Return and restore status	1	2
IN A, Pp	Input port to A	1	2	FLAGS			
OUTL Pp, A	Output A to port	1	2	CLR C	Clear Carry	1	1
ANL Pp, #data	And immediate to port	2	2	CPL C	Complement Carry	1	1
ORL Pp, #data	Or immediate to port	2	2	CLR F0	Clear Flag 0	1	1
IN A, DBB	Input DDB to A, clear IBF	1	1	CPL F0	Complement Flag 0	1	1
OUT DBB, A	Output A to DBB, Set OBF	1	1	CLR F1	Clear F ₁ Flag	1	1
MOV STS, A	A ₄ -A ₇ to bits 4-7 of status	1	1	CPL F1	Complement F ₁ Flag	1	1
MOVD A, Pp	Input Expander port to A	1	2	BRANCH			
MOVD Pp, A	Output A to Expander port	1	2	JMP addr	Jump unconditional	2	2
ANLD Pp, A	And A to Expander port	1	2	JMPP @A	Jump indirect	1	2
ORLD Pp, A	Or A to Expander port	1	2	DJNZ Rr, addr	Decrement register and jump on non-zero	2	2
DATA MOVES				JC addr	Jump on Carry = 1	2	2
MOV A, Rr	Move register to A	1	1	JNC addr.	Jump on Carry = 0	2	2
MOV A, @Rr	Move data memory to A	1	1	JZ addr	Jump on A zero	2	2
MOV A, #data	Move immediate to A	2	2	JNZ addr	Jump on A not zero	2	2
MOV Rr, A	Move A to register	1	1	JT0 addr	Jump on T ₀ = 1	2	2
MOV @Rr, A	Move A to data memory	1	1	JNT0 addr	Jump on T ₀ = 0	2	2
MOV Rr, #data	Move immediate to register	2	2	JT1 addr	Jump on T ₁ = 1	2	2
MOV @Rr, #data	Move immediate to data memory	2	2	JNT1 addr	Jump on T ₁ = 0	2	2
MOV A, PSW	Move PSW to A	1	1	JF0 addr	Jump on F ₀ Flag = 1	2	2
MOV PSW, A	Move A to PSW	1	1	JF1 addr	Jump on F ₁ Flag = 1	2	2
XCH A, Rr	Exchange A and registers	1	1	JTF addr	Jump on Timer Flag = 1	2	2
XCH A, @Rr	Exchange A and data memory	1	1	JNIBF addr	Jump on IBF Flag = 0	2	2
XCHD A, @Rr	Exchange digit of A and register	1	1	JOBFB addr	Jump on OBF Flag = 1	2	2
				JBb addr	Jump on Accumulator Bit	2	2

CALL address Subroutine Call

Opcode:

0	a ₉	a ₈	1	0	1	0	0
---	----------------	----------------	---	---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄	a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

This is a 2-cycle instruction. The program counter and PSW bits 4–7 are saved in the stack. The stack pointer (PSW bits 0–2) is updated. Program control is then passed to the location specified by 'address'.

Execution continues at the instruction following the CALL upon return from the subroutine.
 $((SP)) \leftarrow (PC), (PSW_{4-7})$
 $(SP) \leftarrow (SP) + 1$
 $(PC_{8-9}) \leftarrow (addr_{8-9})$
 $(PC_{0-7}) \leftarrow (addr_{0-7})$

Example: Add three groups of two numbers. Put subtotals in locations 50, 51 and total in location 52.

```

MOV R0, #50          ;MOVE '50' DEC TO ADDRESS
                    ;REG 0
BEGADD: MOV A,R1     ;MOVE CONTENTS OF REG 1
                    ;TO ACC
          ADD A,R2    ;ADD REG 2 TO ACC
          CALL SUBTOT ;CALL SUBROUTINE 'SUBTOT'
          ADD A,R3    ;ADD REG 3 TO ACC
          ADD A,R4    ;ADD REG 4 TO ACC
          CALL SUBTOT ;CALL SUBROUTINE 'SUBTOT'
          ADD A,R5    ;ADD REG 5 TO ACC
          ADD A,R6    ;ADD REG 6 TO ACC
          CALL SUBTOT ;CALL SUBROUTINE 'SUBTOT'
          .
          .
          .
SUBTOT:  MOV @R0,A   ;MOVE CONTENTS OF ACC TO
                    ;LOCATION ADDRESSED BY
                    ;REG 0
          INC R0     ;INCREMENT REG 0
          RET       ;RETURN TO MAIN PROGRAM
    
```

CLR A Clear Accumulator

Opcode:

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

The contents of the accumulator are cleared to zero.
 $(A) \leftarrow 00H$

CLR C Clear Carry Bit

Opcode:

1	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---

During normal program execution, the carry bit can be set to one by the ADD, ADDC, RLC, CPLC, RRC, and DAA instructions. This instruction resets the carry bit to zero.
 $(C) \leftarrow 0$

CLR F1 Clear Flag 1

Opcode:

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

The F₁ flag is cleared to zero.
 $(F_1) \leftarrow 0$

CLR F0 Clear Flag 0

Opcode:

1 0 0 0	0 1 0 1
---------	---------

F₀ flag is cleared to zero.
(F₀) ← 0

CPL A Complement Accumulator

Opcode:

0 0 1 1	0 1 1 1
---------	---------

The contents of the accumulator are complemented. This is strictly a one's complement. Each one is changed to zero and vice-versa.
(A) ← NOT (A)

Example: Assume accumulator contains 01101010.
CPLA: CPL A ;ACC CONTENTS ARE COMPLE-
;MENTED TO 10010101

CPL C Complement Carry Bit

Opcode:

1 0 1 0	0 1 1 1
---------	---------

The setting of the carry bit is complemented; one is changed to zero, and zero is changed to one.
(C) ← NOT (C)

Example: Set C to one; current setting is unknown.
CT01: CLR C ;C IS CLEARED TO ZERO
CPL C ;C IS SET TO ONE

CPL F0 COMPLEMENT FLAG 0

Opcode:

1 0 0 1	0 1 0 1
---------	---------

The setting of Flag 0 is complemented; one is changed to zero, and zero is changed to one.
F₀ ← NOT (F₀)

CPL F1 Complement Flag 1

Opcode:

1 0 1 1	0 1 0 1
---------	---------

The setting of the F₁ Flag is complemented; one is changed to zero, and zero is changed to one.
(F₁) ← NOT (F₁)

DA A Decimal Adjust Accumulator

Opcode:

0 1 0 1	0 1 1 1
---------	---------

The 8-bit accumulator value is adjusted to form two 4-bit Binary Coded Decimal (BCD) digits following the binary addition of BCD numbers. The carry bit C is affected. If the contents of bits 0–3 are greater than nine, or if AC is one, the accumulator is incremented by six.

The four high-order bits are then checked. If bits 4–7 exceed nine, or if C is one, these bits are increased by six. If an overflow occurs, C is set to one; otherwise, it is cleared to zero.

Example: Assume accumulator contains 9AH.

	DA A		;ACC ADJUSTED TO 01H with C set
	C AC		ACC
	0 0		9AH INITIAL CONTENTS
			06H ADD SIX TO LOW DIGIT
	0 0		A1H
			60H ADD SIX TO HIGH DIGIT
	1 0		01H RESULT

DEC A Decrement Accumulator

Opcode:

0 0 0 0	0 1 1 1
---------	---------

The contents of the accumulator are decremented by one.
 $(A) \leftarrow (A) - 1$

Example: Decrement contents of data memory location 63.

	MOV R0, #3FH		;MOVE '3F' HEX TO REG 0
	MOV A, @R0		;MOVE CONTENTS OF LOCATION 63
			;TO ACC
	DEC A		;DECREMENT ACC
	MOV @R0, A		;MOVE CONTENTS OF ACC TO
			;LOCATION 63

DEC Rr Decrement Register

Opcode:

1 1 0 0	1 r ₂ r ₁ r ₀
---------	--

The contents of working register 'r' are decremented by one.
 $(Rr) \leftarrow (Rr) - 1$ $r = 0-7$

Example: DEC R1 ;DECREMENT ADDRESS REG 1

DIS I Disable IBF Interrupt

Opcode:

0 0 0 1	0 1 0 1
---------	---------

The input Buffer Full interrupt is disabled. The interrupt sequence is not initiated by \overline{WR} and CS, however, an IBF interrupt request is latched and remains pending until an EN I (enable IBF interrupt) instruction is executed.

Note: The IBF flag is set and cleared independent of the IBF interrupt request so that handshaking protocol can continue normally.

DIS TCNTI Disable Timer/Counter Interrupt

Opcode:

0	0	1	1
---	---	---	---

0	1	0	1
---	---	---	---

The timer/counter interrupt is disabled. Any pending timer interrupt request is cleared. The interrupt sequence is not initiated by an overflow, but the timer flag is set and time accumulation continues.

DJNZ Rr, address Decrement Register and Test

Opcode:

1	1	1	1
---	---	---	---

1	r ₂	r ₁	r ₀
---	----------------	----------------	----------------

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Register 'r' is decremented and tested for zero. If the register contains all zeros, program control falls through to the next instruction. If the register contents are not zero, control jumps to the specified address within the current page.

$(Rr) \leftarrow (Rr) - 1$
 If $R \neq 0$, then;
 $(PC_{0-7}) \leftarrow \text{addr}$

Note: A 10-bit address specification does not cause an error if the DJNZ instruction and the jump target are on the same page. If the DJNZ instruction begins in location 255 of a page, it will jump to a target address on the following page. Otherwise, it is limited to a jump within the current page.

Example: Increment values in data memory locations 50–54.

```

MOV R0, #50           ;MOVE '50' DEC TO ADDRESS
                     ;REG 0
MOV R3, #05           ;MOVE '5' DEC TO COUNTER
                     ;REG 3
INCR: INC @R0         ;INCREMENT CONTENTS OF
                     ;LOCATION ADDRESSED BY
                     ;REG 0
INC R0                ;INCREMENT ADDRESS IN REG 0
DJNZ R3, INCR         ;DECREMENT REG 3—JUMP TO
                     ;'INCR' IF REG 3 NONZERO
NEXT—                 ;'NEXT' ROUTINE EXECUTED
                     ;IF R3 IS ZERO
    
```

EN DMA Enable DMA Handshake Lines

Opcode:

1	1	1	0
---	---	---	---

0	1	0	1
---	---	---	---

DMA handshaking is enabled using P₂₆ as DMA request (DRQ) and P₂₇ as DMA acknowledge (DACK). The DACK lines forces CS and A₀ low internally and clears DRQ.

EN FLAGS Enable Master Interrupts

Opcode:

1	1	1	1
---	---	---	---

0	1	0	1
---	---	---	---

The Output Buffer Full (OBF) and the Input Buffer Full (IBF) flags (IBF is inverted) are routed to P₂₄ and P₂₅. For proper operation, a "1" should be written to P₂₅ and P₂₄ before the EN FLAGS instruction. A "0" written to P₂₄ or P₂₅ disables the pin.

EN I Enable IBF Interrupt

Opcode:

0 0 0 0	0 1 0 1
---------	---------

The Input Buffer Full interrupt is enabled. A low signal on \overline{WR} and \overline{CS} initiates the interrupt sequence.

EN TCNTI Enable Timer/Counter Interrupt

Opcode:

0 0 1 0	0 1 0 1
---------	---------

The timer/counter interrupt is enabled. An overflow of this register initiates the interrupt sequence.

IN A,DBB Input Data Bus Buffer Contents to Accumulator

Opcode:

0 0 1 0	0 0 1 0
---------	---------

Data in the DBBIN register is transferred to the accumulator and the Input Buffer Full (IBF) flag is set to zero.

(A) ← (DBB)
(IBF) ← 0

Example: INDBB: IN A,DBB ;INPUT DBBIN CONTENTS TO
;ACCUMULATOR

IN A,Pp Input Port 1-2 Data to Accumulator

Opcode:

0 0 0 0	1 0 p ₁ p ₀
---------	-----------------------------------

This is a 2-cycle instruction. Data present on port 'p' is transferred (read) to the accumulator.
(A) ← (Pp) p = 1-2 (see ANL instruction)

Example: INP 12: IN A,P1 ;INPUT PORT 1 CONTENTS
;TO ACC
MOV R6,A ;MOVE ACC CONTENTS TO
;REG 6
IN A,P2 ;INPUT PORT 2 CONTENTS
;TO ACC
MOV R7,A ;MOVE ACC CONTENTS TO REG 7

INC A Increment Accumulator

Opcode:

0 0 0 1	0 1 1 1
---------	---------

The contents of the accumulator are incremented by one.
(A) ← (A) + 1

Example: Increment contents of location 10 in data memory.
INCA: MOV R0,#10 ;MOV '10' DEC TO ADDRESS
;REG 0
MOV A,@R0 ;MOVE CONTENTS OF LOCATION
;10 TO ACC
INC A ;INCREMENT ACC
MOV @R0,A ;MOVE ACC CONTENTS TO
;LOCATION 10

INC Rr Increment Register
Opcode:

0	0	0	1
---	---	---	---

1	r ₂	r ₁	r ₀
---	----------------	----------------	----------------

The contents of working register 'r' are incremented by one.
 $(Rr) \leftarrow (Rr) + 1$ r = 0-7

Example: INCR0: INC R0 ;INCREMENT ADDRESS REG 0

INC @Rr Increment Data Memory Location
Opcode:

0	0	0	1
---	---	---	---

0	0	0	r
---	---	---	---

The contents of the resident data memory location addressed by register 'r' bits 0-5 are incremented by one.
 $((Rr)) \leftarrow ((Rr)) + 1$ r = 0-1

Example: INCDM: MOV R1, #OFFH ;MOVE ONES TO REG 1
;INCREMENT LOCATION 63

JBb address Jump If Accumulator Bit Is Set
Opcode:

b ₂	b ₁	b ₀	1
----------------	----------------	----------------	---

0	0	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if accumulator bit 'b' is set to one.
 $(PC_{0-7}) \text{ addr}$ if b = 1
 $(PC) \leftarrow (PC) + 2$ if b = 0

Example: JB4IS1: JB4 NEXT ;JUMP TO 'NEXT' ROUTINE
;IF ACC BIT 4 = 1

JC address Jump If Carry Is Set
Opcode:

1	1	1	1
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the carry bit is set to one.
 $(PC_{0-7}) \leftarrow \text{addr}$ if C = 1
 $(PC) \leftarrow (PC) + 2$ if C = 0

Example: JC1: JC OVERFLOW ;JUMP TO 'OVFLOW' ROUTINE
;IF C = 1

JF0 address Jump If Flag 0 Is Set
Opcode:

1	0	1	1
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if flag 0 is set to one.
 $(PC_{0-7}) \leftarrow \text{addr}$ if F₀ = 1

Example: JF0IS1: JF0 TOTAL ;JUMP TO 'TOTAL' ROUTINE
;IF F₀ = 1

JF1 address Jump If C/D Flag (F1) Is Set

Opcode:

0	1	1	1
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the C/D flag (F₁) is set to one.

(PC₀₋₇) ← addr if F₁ = 1

Example: JF 11S1: JF1 FILBUF ;JUMP TO 'FILBUF'
;ROUTINE IF F₁ = 1

JMP address Direct Jump Within 1K Block

Opcode:

a ₁₀	a ₉	a ₈	0
-----------------	----------------	----------------	---

0	1	0	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Bits 0-9 of the program counter are replaced with the directly-specified address.

(PC₈₋₉) ← addr 8-9

(PC₀₋₇) ← addr 0-7

Example: JMP SUBTOT ;JUMP TO SUBROUTINE 'SUBTOT'
JMP \$-6 ;JUMP TO INSTRUCTION SIX LOCATIONS
;BEFORE CURRENT LOCATION
JMP 2FH ;JUMP TO ADDRESS '2F' HEX

JMPP @A Indirect Jump Within Page

Opcode:

1	0	1	1
---	---	---	---

0	0	1	1
---	---	---	---

This is a 2-cycle instruction. The contents of the program memory location pointed to by the accumulator are substituted for the 'page' portion of the program counter (PC 0-7).

(PC₀₋₇) ← ((A))

Example: Assume accumulator contains OFH
JMPPAG: JMPP @A ;JMP TO ADDRESS STORED IN
;LOCATION 15 IN CURRENT PAGE

JNC address Jump If Carry Is Not Set

Opcode:

1	1	1	0
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the carry bit is not set, that is, equals zero.

(PC₀₋₇) ← addr if C = 0

Example: JCO: JNC NOVFLO ;JUMP TO 'NOVFLO' ROUTINE
;IF C = 0

JNIBF address Jump If Input Buffer Full Flag Is Low

Opcode:

1	1	0	1
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the Input Buffer Full flag is low (IBF = 0).

(PC₀₋₇) ← addr if IBF = 0

Example: LOC 3: JNIBF LOC 3 ;JUMP TO SELF IF IBF = 0
;OTHERWISE CONTINUE

JNTO address Jump If TEST 0 Is Low

Opcode:

0	0	1	0
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address, if the TEST 0 signal is low. Pin is sampled during SYNC.

(PC₀₋₇) ← addr if T₀ = 0

Example: JTOLOW: JNTO 60 ;JUMP TO LOCATION 60 DEC
;IF T₀ = 0

JNT1 address Jump If TEST 1 Is Low

Opcode:

0	1	0	0
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the TEST 1 signal is low. Pin is sampled during SYNC.

(PC₀₋₇) ← addr if T₁ = 0

Example: JT1LOW: JNT1 OBBH ;JUMP TO LOCATION 'BB' HEX
;IF T₁ = 0

JNZ address Jump If Accumulator Is Not Zero

Opcode:

1	0	0	1
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the accumulator contents are nonzero at the time this instruction is executed.

(PC₀₋₇) ← addr if A ≠ 0

Example: JACCNO: JNZ OABH ;JUMP TO LOCATION 'AB' HEX
;IF ACC VALUE IS NONZERO

JOBF Address Jump If Output Buffer Full Flag Is Set

Opcode:

1	0	0	0
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the Output Buffer Full (OBF) flag is set (= 1) at the time this instruction is executed.

(PC₀₋₇) ← addr if OBF = 1

Example: JOBFHI: JOBF OAAH ;JUMP TO LOCATION 'AA' HEX
;IF OBF = 1

JTF address Jump If Timer Flag Is Set

Opcode:

0	0	0	1
---	---	---	---

0	1	1	0
---	---	---	---

 •

a ₇	a ₆	a ₅	a ₄
----------------	----------------	----------------	----------------

a ₃	a ₂	a ₁	a ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Control passes to the specified address if the timer flag is set to one, that is, the timer/counter register overflows to zero. The timer flag is cleared upon execution of this instruction. (This overflow initiates an interrupt service sequence if the timer-overflow interrupt is enabled.)

(PC₀₋₇) ← addr if TF = 1

Example: JTF1: JTF TIMER ;JUMP TO 'TIMER' ROUTINE
;IF TF = 1

JT0 address Jump If TEST 0 Is High

Opcode:

0 0 1 1	0 1 1 0
---------	---------

 •

a ₇ a ₆ a ₅ a ₄	a ₃ a ₂ a ₁ a ₀
---	---

This is a 2-cycle instruction. Control passes to the specified address if the TEST 0 signal is high (= 1). Pin is sampled during SYNC.
(PC₀₋₇) ← addr if T₀ = 1

Example: JT0HI: JT0 53 ;JUMP TO LOCATION 53 DEC
;IF T₀ = 1

JT1 address Jump If TEST 1 Is High

Opcode:

0 1 0 1	0 1 1 0
---------	---------

 •

a ₇ a ₆ a ₅ a ₄	a ₃ a ₂ a ₁ a ₀
---	---

This is a 2-cycle instruction. Control passes to the specified address if the TEST 1 signal is high (= 1). Pin is sampled during SYNC.
(PC₀₋₇) ← addr if T₁ = 1

Example: JT1HI: JT1 COUNT ;JUMP TO 'COUNT' ROUTINE
;IF T₁ = 1

JZ address Jump If Accumulator Is Zero

Opcode:

1 1 0 0	0 1 1 0
---------	---------

 •

a ₇ a ₆ a ₅ a ₄	a ₃ a ₂ a ₁ a ₀
---	---

This is a 2-cycle instruction. Control passes to the specified address if the accumulator contains all zeros at the time this instruction is executed.
(PC₀₋₇) ← addr if A = 0

Example: JACCO: JZ OA3H ;JUMP TO LOCATION 'A3' HEX
;IF ACC VALUE IS ZERO

MOV A, #data Move Immediate Data to Accumulator

Opcode:

0 0 1 0	0 0 1 1
---------	---------

 •

d ₇ d ₆ d ₅ d ₄	d ₃ d ₂ d ₁ d ₀
---	---

This is a 2-cycle instruction. The 8-bit value specified by 'data' is loaded in the accumulator.
(A) ← data

Example: MOV A, #OA3H ;MOV 'A3' HEX TO ACC

MOV A,PSW Move PSW Contents to Accumulator

Opcode:

1 1 0 0	0 1 1 1
---------	---------

The contents of the program status word are moved to the accumulator.
(A) ← (PSW)

Example: Jump to 'RB1SET' routine if bank switch, PSW bit 4, is set.
BSCHK: MOV A,PSW ;MOV PSW CONTENTS TO ACC
JB4 RB1 SET ;JUMP TO 'RB1SET' IF ACC
;BIT 4 = 1

MOV A,Rr Move Register Contents to Accumulator

Opcode:

1	1	1	1
---	---	---	---

1	r_2	r_1	r_0
---	-------	-------	-------

Eight bits of data are moved from working register 'r' into the accumulator.

(A) ← (Rr) r = 0-7

Example: MAR: MOV A,R3 ;MOVE CONTENTS OF REG 3
;TO ACC

MOV A,@Rr Move Data Memory Contents to Accumulator

Opcode:

1	1	1	1
---	---	---	---

0	0	0	r
---	---	---	---

The contents of the data memory location addressed by bits 0-5 of register 'r' are moved to the accumulator. Register 'r' contents are unaffected.

(A) ← ((Rr)) r = 0-1

Example: Assume R1 contains 00110110.
MADM: MOV A,@R1 ;MOVE CONTENTS OF DATA MEM
;LOCATION 54 TO ACC

MOV A,T Move Timer/Counter Contents to Accumulator

Opcode:

0	1	0	0
---	---	---	---

0	0	1	0
---	---	---	---

The contents of the timer/event-counter register are moved to the accumulator. The timer/event-counter is not stopped.

(A) ← (T)

Example: Jump to "Exit" routine when timer reaches '64', that is, when bit 6 is set—assuming initialization to zero.
TIMCHK: MOV A,T ;MOVE TIMER CONTENTS TO
;ACC
;JUMP TO 'EXIT' IF ACC BIT
;6 = 1
JB6 EXIT

MOV PSW,A Move Accumulator Contents to PSW

Opcode:

1	1	0	1
---	---	---	---

0	1	1	1
---	---	---	---

The contents of the accumulator are moved into the program status word. All condition bits and the stack pointer are affected by this move.

(PSW) ← (A)

Example: Move up stack pointer by two memory locations, that is, increment the pointer by one.
INCPTR: MOV A,PSW ;MOVE PSW CONTENTS TO ACC
;INCREMENT ACC BY ONE
;MOVE ACC CONTENTS TO PSW
INC A
MOV PSW,A

MOV Rr,A Move Accumulator Contents to Register

Opcode:

1	0	1	0
---	---	---	---

1	r ₂	r ₁	r ₀
---	----------------	----------------	----------------

The contents of the accumulator are moved to register 'r'
 $(Rr) \leftarrow (A)$ r = 0-7

Example: MRA MOV R0,A ;MOVE CONTENTS OF ACC TO
 ;REG 0

MOV Rr,#data Move Immediate Data to Register

Opcode:

1	0	1	1
---	---	---	---

1	r ₂	r ₁	r ₀
---	----------------	----------------	----------------

 •

d ₇	d ₆	d ₅	d ₄
----------------	----------------	----------------	----------------

d ₃	d ₂	d ₁	d ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. The 8-bit value specified by 'data' is moved to register 'r'.
 $(Rr) \leftarrow \text{data}$ r = 0-7

Example: MIR4: MOV R4,#HEXTEN ;THE VALUE OF THE SYMBOL
 ;'HEXTEN' IS MOVED INTO
 ;REG 4
 MIR5: MOV R5,#PI*(R*R) ;THE VAUE OF THE
 ;EXPRESSION 'PI*(R*R)'
 ;IS MOVED INTO REG 5
 MIR6: MOV R6,#OADH ;'AD' HEX IS MOVED INTO
 REG 6

MOV @Rr,A Move Accumulator Contents to Data Memory

Opcode:

1	0	1	0
---	---	---	---

0	0	0	r
---	---	---	---

The contents of the accumulator are moved to the data memory location whose address is specified by bits 0-5 of register 'r'. Register 'r' contents are unaffected.
 $((Rr)) \leftarrow (A)$ r = 0-1

Example: Assume R0 contains 11000111.
 MDMA: MOV @R,A ;MOVE CONTENTS OF ACC TO
 ;LOCATION 7 (REG)

MOV @Rr,#data Move Immediate Data to Data Memory

Opcode:

1	0	1	1
---	---	---	---

0	0	0	r
---	---	---	---

 •

d ₇	d ₆	d ₅	d ₄
----------------	----------------	----------------	----------------

d ₃	d ₂	d ₁	d ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. The 8-bit value specified by 'data' is moved to the standard data memory location addressed by register 'r', bit 0-5.

Example: Move the hexadecimal value AC3F to locations 62-63.
 MIDM: MOV R0,#62 ;MOVE '62' DEC TO ADDR REG0
 MOV @R0,#OACH ;MOVE 'AC' HEX TO LOCATION 62
 INC R0 ;INCREMENT REG 0 TO '63'
 MOV @R0,#3FH ;MOVE '3F' HEX TO LOCATION 63

MOV STS,A Move Accumulator Contents to STS Register

Opcode:

1 0 0 1	0 0 0 0
---------	---------

The contents of the accumulator are moved into the status register. Only bits 4–7 are affected.
 $(STS_{4-7}) \leftarrow (A_{4-7})$

Example: Set ST_4-ST_7 to "1".
MSTS: MOV A, #0FOH ;SET ACC
MOV STS,A ;MOVE TO STS

MOV T,A Move Accumulator Contents to Timer/Counter

Opcode:

0 1 1 0	0 0 1 0
---------	---------

The contents of the accumulator are moved to the timer/event-counter register.
 $(T) \leftarrow (A)$

Example: Initialize and start event counter.
INITEC: CLR A ;CLEAR ACC TO ZEROS
MOV T,A ;MOVE ZEROS TO EVENT COUNTER
STRT CNT ;START COUNTER

MOVD A,Pp Move Port 4–7 Data to Accumulator

Opcode:

0 0 0 0	1 1 p ₁ p ₀
---------	-----------------------------------

This is a 2-cycle instruction. Data on 8243 port 'p' is moved (read) to accumulator bits 0–3. Accumulator bits 4–7 are zeroed.
 $(A_{0-3}) \leftarrow Pp$ $p = 4-7$
 $(A_{4-7}) \leftarrow 0$

Note: Bits 0–1 of the opcode are used to represent PORTS 4–7. If you are coding in binary rather than assembly language, the mapping is as follows:

Bits		Port
P ₁	P ₀	
0	0	4
0	1	5
1	0	6
1	1	7

Example: INPPT5: MOVD A,P5 ;MOVE PORT 5 DATA TO ACC
;BITS 0–3, ZERO ACC BITS 4–7

MOVD Pp,A Move Accumulator Data to Port 4, 5, 6 and 7

Opcode:

0 0 1 1	1 1 p ₁ p ₀
---------	-----------------------------------

This is a 2-cycle instruction. Data in accumulator bits 0–3 is moved (written) to 8243 port 'p'. Accumulator bits 4–7 are unaffected. (See NOTE above regarding port mapping.)

Example: Move data in accumulator to ports 4 and 5.
OUTP45: MOVD P4,A ;MOVE ACC BITS 0–3 TO PORT 4
SWAP A ;EXCHANGE ACC BITS 0–3 AND 4–7
MOVD P5,A ;MOVE ACC BITS 0–3 TO PORT 5

OUTL Pp,A Output Accumulator Data to Port 1 and 2

Opcode:

0 0 1 1	1 0 p ₁ p ₀
---------	-----------------------------------

This is a 2-cycle instruction. Data residing in the accumulator is transferred (written) to port 'p' and latched.

$$(Pp) \leftarrow (A) \qquad P = 1-2$$

Note: Bits 0-1 of the opcode are used to represent PORT 1 and PORT 2. If you are coding in binary rather than assembly language, the mapping is as follows:

Bits		Port
P ₁	P ₀	
0	0	X
0	1	1
1	0	2
1	1	X

Example: `OUTLP; MOV A,R7` ;MOVE REG 7 CONTENTS TO ACC
`OUTL P2,A` ;OUTPUT ACC CONTENTS TO PORT2
`MOV A,R6` ;MOVE REG 6 CONTENTS TO ACC
`OUTL P1,A` ;OUTPUT ACC CONTENTS TO PORT 1

RET Return Without PSW Restore

Opcode:

1 0 0 0	0 0 1 1
---------	---------

This is a 2-cycle instruction. The stack pointer (PSW bits 0-2) is decremented. The program counter is then restored from the stack. PSW bits 4-7 are not restored.

$$(SP) \leftarrow (SP) - 1$$

$$(PC) \leftarrow ((SP))$$

RETR Return With PSW Restore

Opcode:

1 0 0 1	0 0 1 1
---------	---------

This is a 2-cycle instruction. The stack pointer is decremented. The program counter and bits 4-7 of the PSW are then restored from the stack. Note that RETR should be used to return from an interrupt, but should not be used within the interrupt service routine as it signals the end of an interrupt routine.

$$(SP) \leftarrow (SP) - 1$$

$$(PC) \leftarrow ((SP))$$

$$(PSW_{4-7}) \leftarrow ((SP))$$

RL A Rotate Left Without Carry

Opcode:

1 1 1 0	0 1 1 1
---------	---------

The contents of the accumulator are rotated left one bit. Bit 7 is rotated into the bit 0 position.

$$(A_{n+1}) \leftarrow (A_n) \qquad n = 0-6$$

$$(A_0) \leftarrow (A_7)$$

Example: Assume accumulator contains 10110001.
`RLNC: RL A` ;NEW ACC CONTENTS ARE 01100011

SEL RB1 Select Register Bank 1**Opcode:**

1 1 0 1	0 1 0 1
---------	---------

PSW bit 4 is set to one. References to working registers 0–7 address data memory locations 24–31. This is the recommended setting for interrupt service routines, since locations 0–7 are left intact. The setting of PSW bit 4 in effect at the time of an interrupt is restored by the RETR instruction when the interrupt service routine is completed.

Example: Assume an IBF interrupt has occurred, control has passed to program memory location 3, and PSW bit 4 was zero before the interrupt.

```
LOC3: JMP INIT                ;JUMP TO ROUTINE 'INIT'
      .
      .
INIT:  MOV R7,A                ;MOV ACC CONTENTS TO
      .                        ;LOCATION 7
      SEL RB1                  ;SELECT REG BANK 1
      MOV R7,#OFAH            ;MOVE 'FA' HEX TO LOCATION 31
      .
      .
      SEL RBO                  ;SELECT REG BANK 0
      MOV A,R7                ;RESTORE ACC FROM LOCATION 7
      RETR                     ;RETURN—RESTORE PC AND PSW
```

STOP TCNT Stop Timer/Event Counter**Opcode:**

0 1 1 0	0 1 0 1
---------	---------

This instruction is used to stop both time accumulation and event counting.

Example: Disable interrupt, but jump to interrupt routine after eight overflows and stop timer. Count overflows in register 7.

```
START: DIS TCNTI              ;DISABLE TIMER INTERRUPT
      CLR A                    ;CLEAR ACC TO ZERO
      MOV T,A                  ;MOV ZERO TO TIMER
      MOV R7,A                 ;MOVE ZERO TO REG 7
      STRT T                   ;START TIMER
MAIN:  JTF COUNT               ;JUMP TO ROUTINE 'COUNT'
      .                        ;IF TF = 1 AND CLEAR TIMER FLAG
      JMP MAIN                 ;CLOSE LOOP
COUNT: INC R7                 ;INCREMENT REG 7
      MOV A,R7                 ;MOVE REG 7 CONTENTS TO ACC
      JB3 INT                  ;JUMP TO ROUTINE 'INT' IF ACC
      .                        ;BIT 3 IS SET (REG 7 = 8)
      JMP MAIN                 ;OTHERWISE RETURN TO ROUTINE
      .                        ;MAIN
      .
INT:   STOP TCNT               ;STOP TIMER
      JMP 7H                   ;JUMP TO LOCATION 7 (TIMER
      .                        ;INTERRUPT ROUTINE)
```

STRT CNT Start Event Counter

Opcode:

0 1 0 0	0 1 0 1
---------	---------

The TEST 1 (T₁) pin is enabled as the event-counter input and the counter is started. The event-counter register is incremented with each high to low transition on the T₁ pin.

Example: Initialize and start event counter. Assume overflow is desired with first T₁ input.

```

STARTC: EN TCNTI           ;ENABLE COUNTER INTERRUPT
        MOV A,#OFFH       ;MOVE 'FF' HEX (ONES) TO
                           ;ACC
        MOV T,A           ;MOVE ONES TO COUNTER
        STRT CNT          ;INPUT AND START

```

STRT T Start Timer

Opcode:

0 1 0 1	0 1 0 1
---------	---------

Timer accumulation is initiated in the timer register. The register is incremented every 32 instruction cycles. The prescaler which counts the 32 cycles is cleared but the timer register is not.

Example: Initialize and start timer.

```

STARTT: EN TCNTI           ;ENABLE TIMER INTERRUPT
        CLR A              ;CLEAR ACC TO ZEROS
        MOV T,A           ;MOVE ZEROS TO TIMER
        STRT T            ;START TIMER

```

SWAP A Swap Nibbles Within Accumulator

Opcode:

0 1 0 0	0 1 1 1
---------	---------

Bits 0–3 of the accumulator are swapped with bits 4–7 of the accumulator.
(A_{4–7}) ↔ (A_{0–3})

Example: Pack bits 0–3 of locations 50–51 into location 50.

```

PCKDIG: MOV R0,#50        ;MOVE '50' DEC TO REG 0
        MOV R1,#51        ;MOVE '51' DEC TO REG 1
        XCHD A,@R0        ;EXCHANGE BIT 0–3 OF ACC
                           ;AND LOCATION 50
        SWAP A            ;SWAP BITS 0–3 AND 4–7 OF ACC
        XCHD A,@ R1       ;EXCHANGE BITS 0–3 OF ACC AND
                           ;LOCATION 51
        MOV @R0,A         ;MOVE CONTENTS OF ACC TO
                           ;LOCATION 51

```

XCH ARr Exchange Accumulator-Register Contents

Opcode:

0 0 1 0	1 r ₂ r ₁ r ₀
---------	--

The contents of the accumulator and the contents of working register 'r' are exchanged.
(A) ↔ (Rr) r = 0–7

Example: Move PSW contents to Reg 7 without losing accumulator contents.

```

XCHAR7: XCH A,R7         ;EXCHANGE CONTENTS OF REG 7
                           ;AND ACC
        MOV A,PSW        ;MOVE PSW CONTENTS TO ACC
        XCH A,R7         ;EXCHANGE CONTENTS OF REG 7
                           ;AND ACC AGAIN

```

XCH A,@Rr Exchange Accumulator and Data Memory Contents
Opcode:

0 0 1 0	0 0 0 r
---------	---------

The contents of the accumulator and the contents of the data memory location addressed by bits 0-5 of register 'r' are exchanged. Register 'r' contents are unaffected.

(A) \longleftrightarrow ((Rr)) r = 0-1

Example: Decrement contents of location 52.

```

DEC 52: MOV R0, #52           ;MOVE '52' DEC TO ADDRESS
                                ;REG 0
                                ;EXCHANGE CONTENTS OF ACC
                                ;AND LOCATION 52
                                ;DECREMENT ACC CONTENTS
                                ;EXCHANGE CONTENTS OF ACC
                                ;AND LOCATION 52 AGAIN

                                XCH A,@R0
                                DEC A
                                XCH A,@R0
    
```

XCHD A,@Rr Exchange Accumulator and Data Memory 4-bit Data
Opcode:

0 0 1 1	0 0 0 r
---------	---------

This instruction exchanges bits 0-3 of the accumulator with bits 0-3 of the data memory location addressed by bits 0-5 of register 'r'. Bits 4-7 of the accumulator, bits 4-7 of the data memory location, and the contents of register 'r' are unaffected.

(A₀₋₃) \longleftrightarrow ((Rr₀₋₃)) r = 0-1

Example: Assume program counter contents have been stacked in locations 22-23.

```

XCHNIB: MOV R0, #23          ;MOVE '23' DEC TO REG 0
                                ;CLEAR ACC TO ZEROS
                                ;EXCHANGE BITS 0-3 OF ACC
                                ;AND LOCATION 23 (BITS 8-11
                                ;OF PC ARE ZEROED, ADDRESS
                                ;REFERS TO PAGE 0)

                                CLR A
                                XCHD A,@R0
    
```

XRL A,Rr Logical XOR Accumulator With Register Mask
Opcode:

1 1 0 1	1 r ₂ r ₁ r ₀
---------	--

Data in the accumulator is EXCLUSIVE ORed with the mask contained in working register 'r'.

(A) \longleftrightarrow (A) XOR (Rr) r = 0-7

Example: XORREG: XRL A,R5 ;'XOR' ACC CONTENTS WITH
;MASK IN REG 5

XRL A,@Rr Logical XOR Accumulator With Memory Mask
Opcode:

1 1 0 1	0 0 0 r
---------	---------

Data in the accumulator is EXCLUSIVE ORed with the mask contained in the data memory location address by register 'r', bits 0-5.

(A) \leftarrow (A) XOR ((Rr)) r = 0-1

Example: XORDM: MOV R1, #20H ;MOVE '20' HEX TO REG 1
XRL A,@R1 ;'XOR' ACC CONTENTS WITH MASK
;IN LOCATION 32

XRL A, #data, Logical XOR Accumulator With Immediate Mask

Opcode:

1	1	0	1
---	---	---	---

0	0	1	1
---	---	---	---

 •

d ₇	d ₆	d ₅	d ₄
----------------	----------------	----------------	----------------

d ₃	d ₂	d ₁	d ₀
----------------	----------------	----------------	----------------

This is a 2-cycle instruction. Data in the accumulator is EXCLUSIVE Ored with an immediately-specified mask.

(A) ← (A) XOR data

Example: XORID: XOR A, #HEXTEN ;XOR CONTENTS OF ACC WITH
;MASK EQUAL VALUE OF SYMBOL
;HEXTEN'

CHAPTER 4 SINGLE-STEP AND POWER-DOWN MODES

SINGLE-STEP

The UPI family has a single-step mode which allows the user to manually step through his program one instruction at a time. While stopped, the address of the next instruction to be fetched is available on PORT 1 and the lower 2 bits of PORT 2. The single-step feature simplifies program debugging by allowing the user to easily follow program execution.

Figure 4-1 illustrates a recommended circuit for single-step operation, while Figure 4-2 shows the timing relationship between the SYNC output and the \overline{SS} input. During single-step operation, PORT 1 and part of PORT 2 are used to output address information. In order to retain the normal I/O functions of PORTS 1 and 2, a separate latch can be used as shown in Figure 4-3.

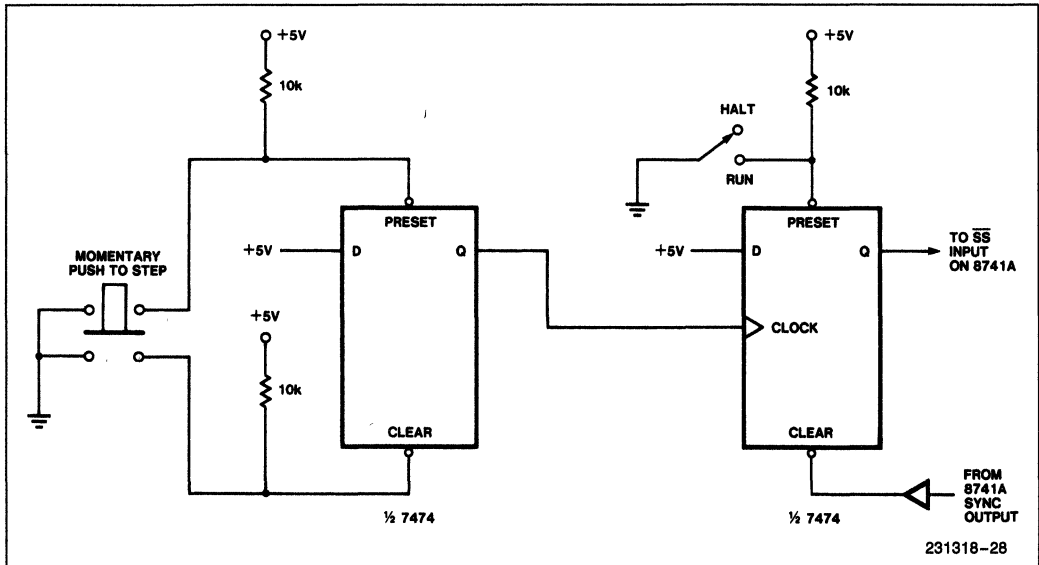


Figure 4-1. Single-Step Circuit

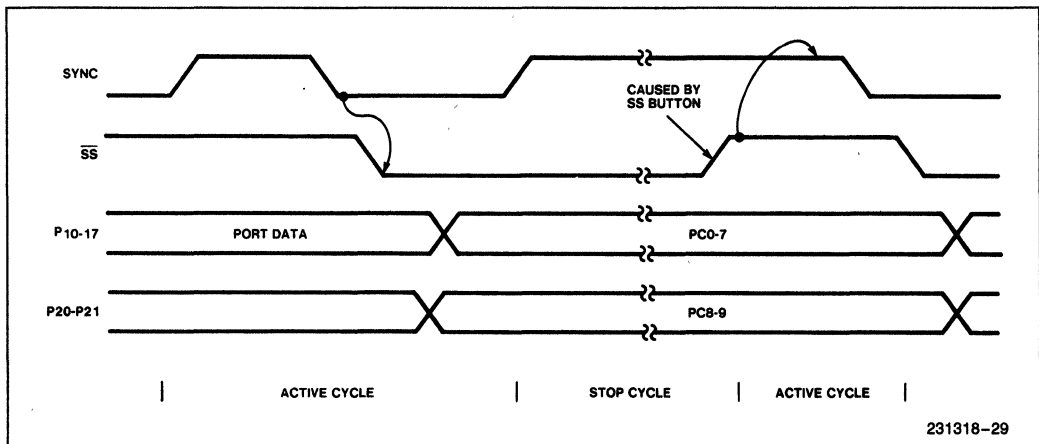


Figure 4-2. Single-Step Timing

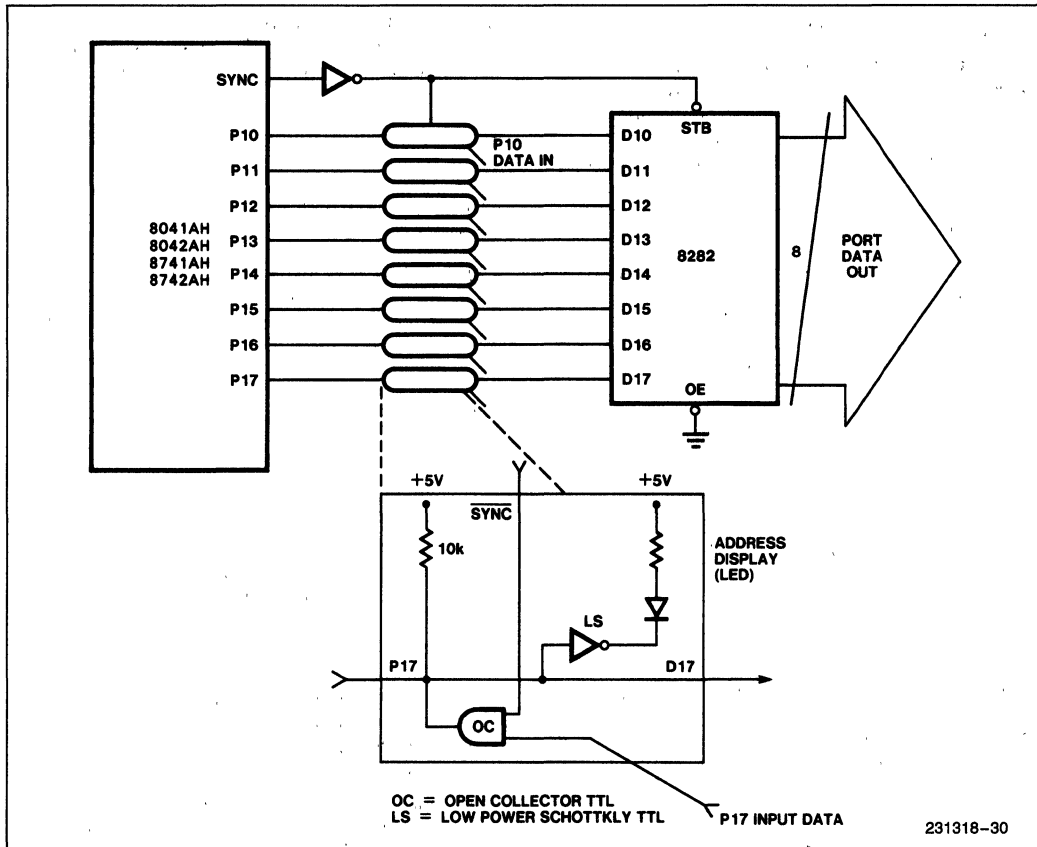


Figure 4-3. Latching Port Data

Timing

The sequence of single-step operation is as follows:

- 1) The processor is requested to stop by applying a low level on \overline{SS} . The \overline{SS} input should not be brought low while SYNC is high. (The UPI samples the \overline{SS} pin in the middle of the SYNC pulse).
- 2) The processor responds to the request by stopping during the instruction fetch portion of the next instruction. If a double cycle instruction is in progress when the single-step command is received, both cycles will be completed before stopping.
- 3) The processor acknowledges it has entered the stopped state by raising SYNC high. In this state, which can be maintained indefinitely, the 10-bit address of the next instruction to be fetched is preset on PORT 1 and the lower 2 bits of PORT 2.
- 4) \overline{SS} is then raised high to bring the processor out of the stopped mode allowing it to fetch the next instruction. The exit from stop is indicated by the processor bringing SYNC low.

- 5) To stop the processor at the next instruction \overline{SS} must be brought low again before the next SYNC pulse—the circuit in Figure 4-1 uses the trailing edge of the previous pulse. If \overline{SS} is left high, the processor remains in the “RUN” mode.

Figure 4-1 shows a schematic for implementing single-step. A single D-type flip-flop with preset and clear is used to generate \overline{SS} . In the RUN mode \overline{SS} is held high by keeping the flip-flop preset (preset has precedence over the clear input). To enter single-step, preset is removed allowing SYNC to bring \overline{SS} low via the clear input. Note that SYNC must be buffered since the SN7474 is equivalent to 3 TTL loads.

The processor is now in the stopped state. The next instruction is initiated by stoppe state. The next instruction is initiated by clocking “1” the flip-flop. This “1” will not appear on \overline{SS} unless SYNC is high (I.e., clear must be removed from the flip-flop). In response to \overline{SS} going high, the processor begins an instruction fetch which brings SYNC low. \overline{SS} is then reset through the clear input and the processor again enters the stopped state.

EXTERNAL ACCESS

The UPI family has an External Access mode (EA) which puts the processor into a test mode. This mode allows the user to disable the internal program memory and execute from external memory. External Access mode is useful in testing because it allows the user to test the processor's functions directly. It is only useful for testing since this mode uses D₀-D₇, PORTS 10-17 and PORTS 20-22.

This mode is invoked by connecting the EA pin to 5V. The 11-bit current program counter contents then come out on PORTS 10-17 and PORTS 20-22 after the SYNC output goes high. (PORT 10 is the least significant bit.) The desired instruction opcode is placed on D₀-D₇ before the start of state S₁. During state S₁, the opcode is sampled from D₀-D₇ and subsequently executed in place of the internal program memory contents.

The program counter contents are multiplexed with the I/O port data on PORTS 10-17 and PORTS 20-22. The I/O port data may be demultiplexed using an external latch on the rising edge of SYNC. The program counter contents may be demultiplexed similarly using the trailing edge of SYNC.

Reading and/or writing the Data Bus Buffer registers is still allowed although only when D₀-D₇ are not being sampled for opcode data. In practice, since this sampling time is not known externally, reads or writes on the system bus are done during SYNC high time. Approximately 600 ns are available for each read or write cycle.

POWER DOWN MODE (8041AH/8042AH ROM ONLY)

Extra circuitry is included in the ROM version to allow low-power, standby operation. Power is removed from

all system elements except the internal data RAM in the low-power mode. Thus the contents of RAM can be maintained and the device draws only 10 to 15% of its normal power.

The V_{CC} pin serves as the 5V power supply pin for all of the ROM version's circuitry except the data RAM array. The V_{DD} pin supplies only the RAM array. In normal operation, both V_{CC} and V_{DD} are connected to the same 5V power supply.

To enter the Power-Down mode, the $\overline{\text{RESET}}$ signal to the UPI is asserted. This ensures the memory will not be inadvertently altered by the UPI during power-down. The V_{CC} pin is then grounded while V_{DD} is maintained at 5V. Figure 4-4 illustrates a recommended Power-Down sequence. The sequence typically occurs as follows:

- 1) Imminent power supply failure is detected by user defined circuitry. The signal must occur early enough to guarantee the 8041AH or 8042AH can save all necessary data before V_{CC} falls outside normal operating tolerance.
- 2) A "Power Failure" signal is used to interrupt the processor (via a timer overflow interrupt, for instance) and call a Power Failure service routine.
- 3) The Power Failure routine saves all important data and machine status in the RAM array. The routine may also initiate transfer of a backup supply to the V_{DD} pin and indicate to external circuitry that the Power Failure routine is complete.
- 4) A $\overline{\text{RESET}}$ signal is applied by external hardware to guarantee data will not be altered as the power supply falls out of limits. $\overline{\text{RESET}}$ must be low until V_{CC} reaches ground potential.

Recovery from the Power-Down mode can occur as any other power-on sequence. An external 1 μfd capacitor on the $\overline{\text{RESET}}$ input will provide the necessary initialization pulse.

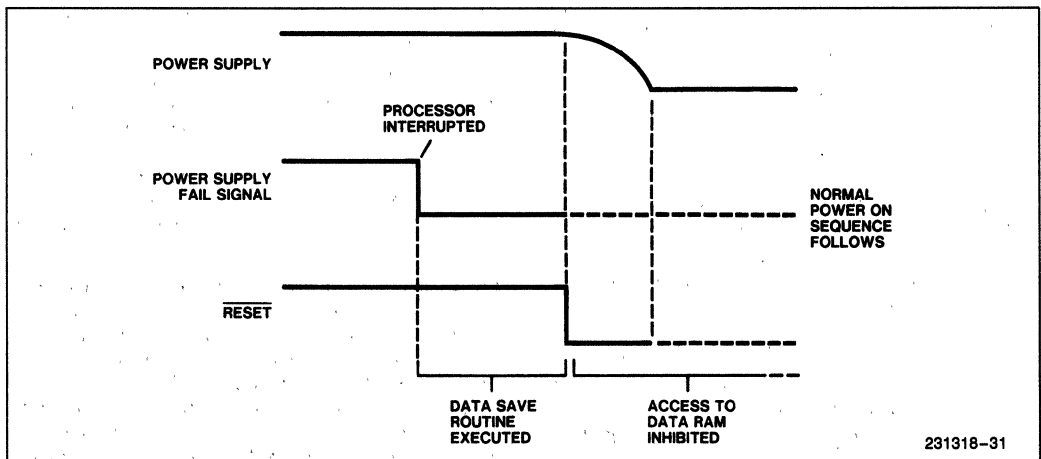


Figure 4-4. Power-Down Sequence

CHAPTER 5 SYSTEM OPERATION

BUS INTERFACE

The UPI-41AH, 42AH Microcomputer functions as a peripheral to a master processor by using the data bus buffer registers to handle data transfers. The DBB configuration is illustrated in Figure 5-1. The UPI-41AH, 42AH Microcomputer's 8 three-state data lines (D₇-D₀) connect directly to the master processor's data bus. Data transfer to the master is controlled by 4 external inputs to the UPI:

- A₀ Address Input signifying command or data
- \overline{CS} Chip Select
- \overline{RD} Read strobe
- \overline{WR} Write strobe

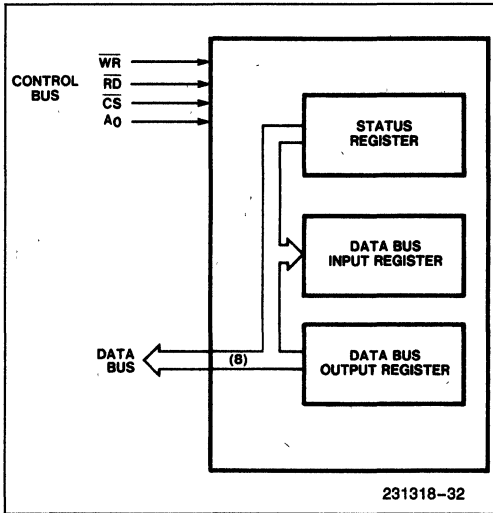


Figure 5-1. Data Bus Register Configuration

The master processor addresses the UPI-41AH, 42AH Microcomputer as a standard peripheral device. Table 5-1 shows the conditions for data transfer:

Table 5-1. Data Transfer Controls

CS	A ₀	RD	WR	Condition
0	0	0	1	Read DBBOUT
0	1	0	1	Read STATUS
0	0	1	0	Write DBBIN data, set F ₁ = 0
0	1	1	0	Write DBBIN command set F ₁ = 1
1	x	x	x	Disable DBB

Reading the DBBOUT Register

The sequence for reading the DBBOUT register is shown in Figure 5-2. This operation causes the 8-bit contents of the DBBOUT register to be placed on the system Data Bus. The OBF flag is cleared automatically.

Reading STATUS

The sequence for reading the UPI-41AH, 42AH Microcomputer's 8 STATUS bits is shown in Figure 5-3. This operation causes the 8-bit STATUS register contents to be placed on the system Data Bus as shown.

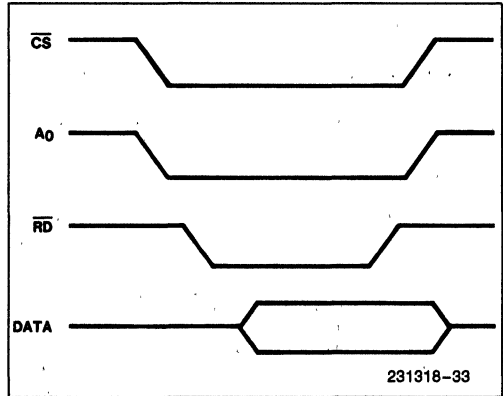


Figure 5-2. DBBOUT Read

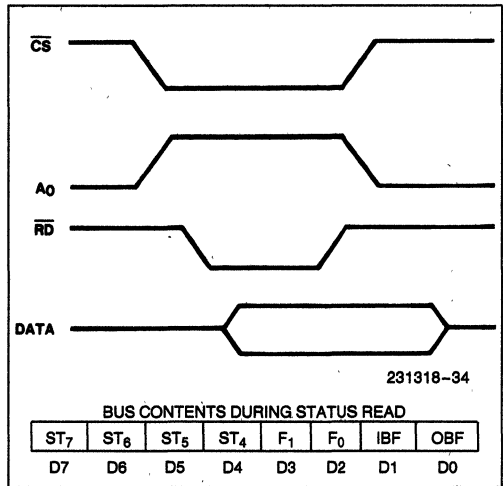


Figure 5-3. Status Read

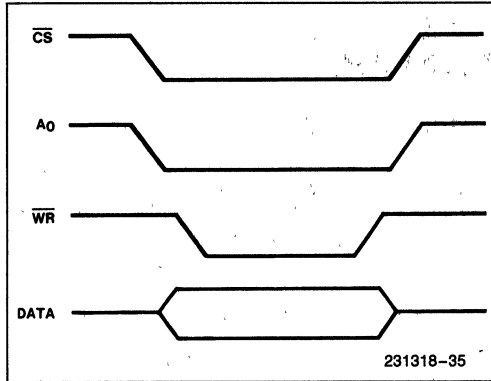


Figure 5-4. Writing Data to DBBIN

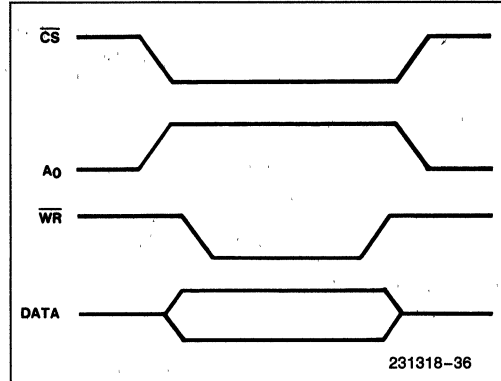


Figure 5-5. Writing Commands to DBBIN

Write Data to DBBIN

The sequence for writing data to the DBBIN register is shown in Figure 5-4. This operation causes the system Data Bus contents to be transferred to the DBBIN register and the IBF flag is set. Also, the F_1 flag is cleared ($F_1 = 0$) and an interrupt request is generated. When the IBF interrupt is enabled, a jump to location 3 will occur. The interrupt request is cleared upon entering the IBF service routine or by a system RESET input.

Writing Commands to DBBIN

The sequence for writing commands to the DBBIN register is shown in Figure 5-5. This sequence is identical to a data write except that the A_0 input is latched in the F_1 flag ($F_1 = 1$). The IBF flag is set and an interrupt request is generated when the master writes a command to DBB.

Operations of Data Bus Registers

The UPI-41AH, 42AH Microcomputer controls the transfer of DBB data to its accumulator by executing INput and OUTput instructions. An IN A,DBB instruction causes the contents to be transferred to the UPI accumulator and the IBF flag is cleared.

The OUT DBB,A instruction causes the contents of the accumulator to be transferred to the DBBOUT register. The OBF flag is set.

The UPI's data bus buffer interface is applicable to a variety of microprocessors including the 8086, 8088, 8085AH, 8080, and 8048.

A description of the interface to each of these processors follows.

DESIGN EXAMPLES

8085AH Interface

Figure 5-6 illustrates an 8085AH system using a UPI-41AH, 42AH. The 8085AH system uses a multiplexed address and data bus. During I/O the 8 upper address lines (A_8-A_{15}) contain the same I/O address as the lower 8 address/data lines (A_0-A_7); therefore I/O address decoding is done using only the upper 8 lines to eliminate latching of the address. An 8205 decoder provides address decoding for both the UPI-41AH, 42AH and the 8237. Data is transferred using the two DMA handshaking lines of PORT 2. The 8237 performs the actual bus transfer operation. Using the UPI-41AH, 42AH's OBF master interrupt, the UPI-41A notifies the 8085AH upon transfer completion using the RST 5.5 interrupt input. The $\overline{\text{IBF}}$ master interrupt is not used in this example.

8088 Interface

Figure 5-7 illustrates a UPI-41AH, 42AH interface to an 8088 minimum mode system. Two 8-bit latches are used to demultiplex the address and data bus. The address bus is 20-lines wide. For I/O only, the lower 16 address lines are used, providing an addressing range of 64K. UPI address selection is accomplished using an 8205 decoder. The A_0 address line of the bus is connected to the corresponding UPI input for register selection. Since the UPI-41A is polled by the 8088, neither DMA nor master interrupt capabilities of the UPI-41AH, 42AH are used in the figure.

8086 Interface

The UPI-41AH, 42AH can be used on an 8086 maximum mode system as shown in Figure 5-8. The address and data bus is demultiplexed using three 8282 latches providing separate address and data buses. The address

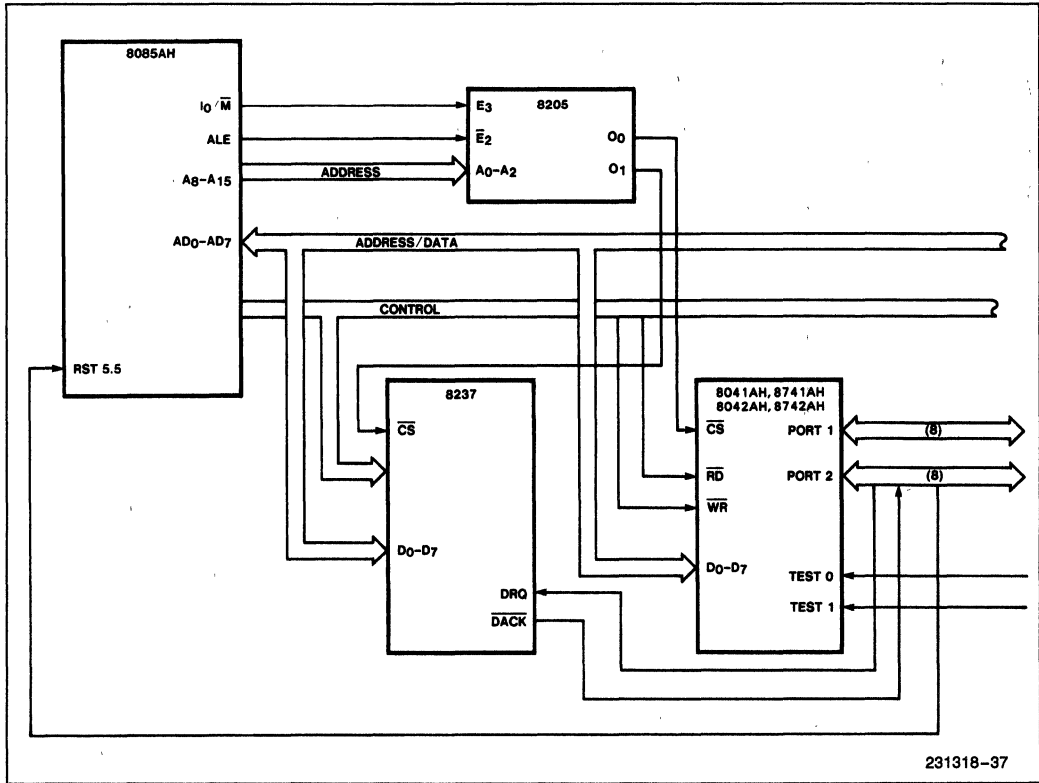


Figure 5-6. 8085AH-UPI System

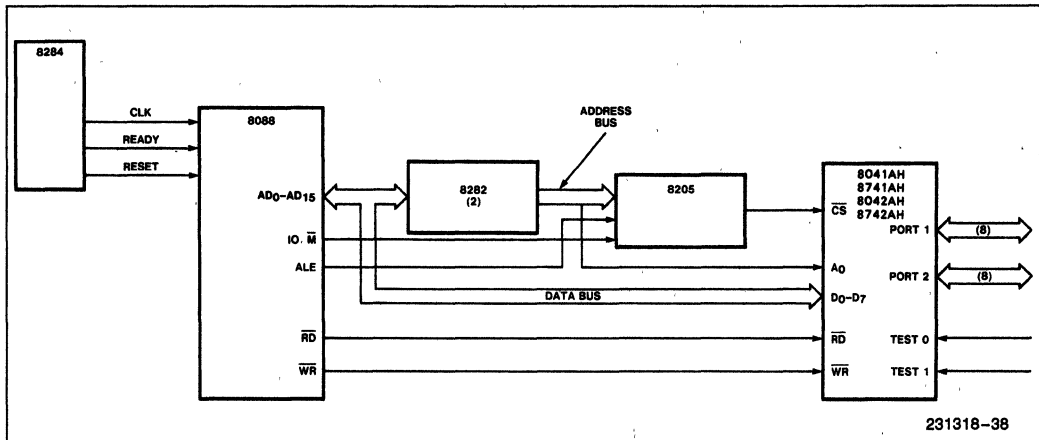


Figure 5-7. 8088-UPI Minimum Mode System

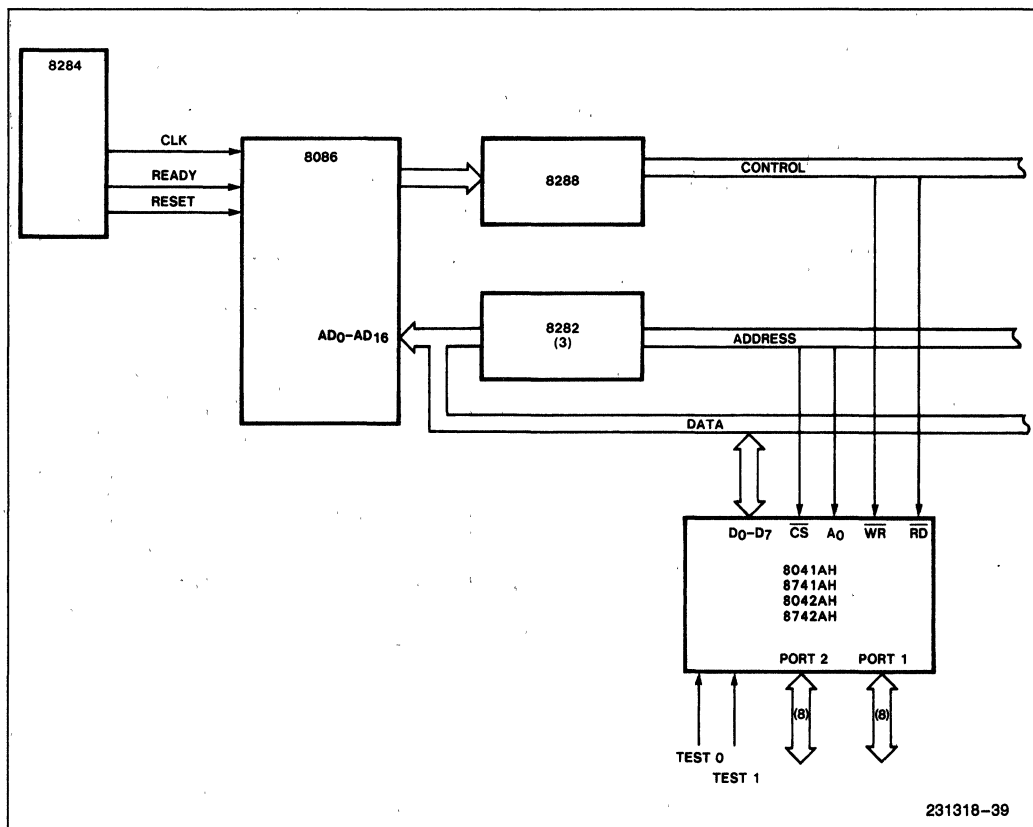


Figure 5-8. 8086-UPI Maximum Mode Systems

bus is 20-lines wide and the data bus is 16-lines wide. Multiplexed control lines are decoded by the 8288. The UPI's \overline{CS} input is provided by linear selection. Note that the UPI-41AH, 42AH is both I/O mapped and memory mapped as a result of the linear addressing technique. An address decoder may be used to limit the UPI-41AH, 42AH to a specific I/O mapped address. Address line A_1 is connected to the UPI's A_0 input. This insures that the registers of the UPI will have even I/O addresses. Data will be transferred on D_0 - D_7 lines only. This allows the I/O registers to be accessed using byte manipulation instructions.

8080 Interface

Figure 5-9 illustrates the interface to an 8080A system. In this example, a crystal and capacitor are used for UPI-41AH, 42AH timing reference and power-on RESET. If the 2-MHz 8080A 2-phase clock were used instead of the crystal, the UPI-41AH, UPI-42AH would run at only 16% full speed.

The A_0 and \overline{CS} inputs are direct connections to the 8080 address bus. In larger systems, however, either of these inputs may be decoded from the 16 address lines.

The \overline{RD} and \overline{WR} inputs to the UPI can be either the \overline{IOR} and \overline{IOW} or the \overline{MEMR} and \overline{MEMW} signals depending on the I/O mapping technique to be used.

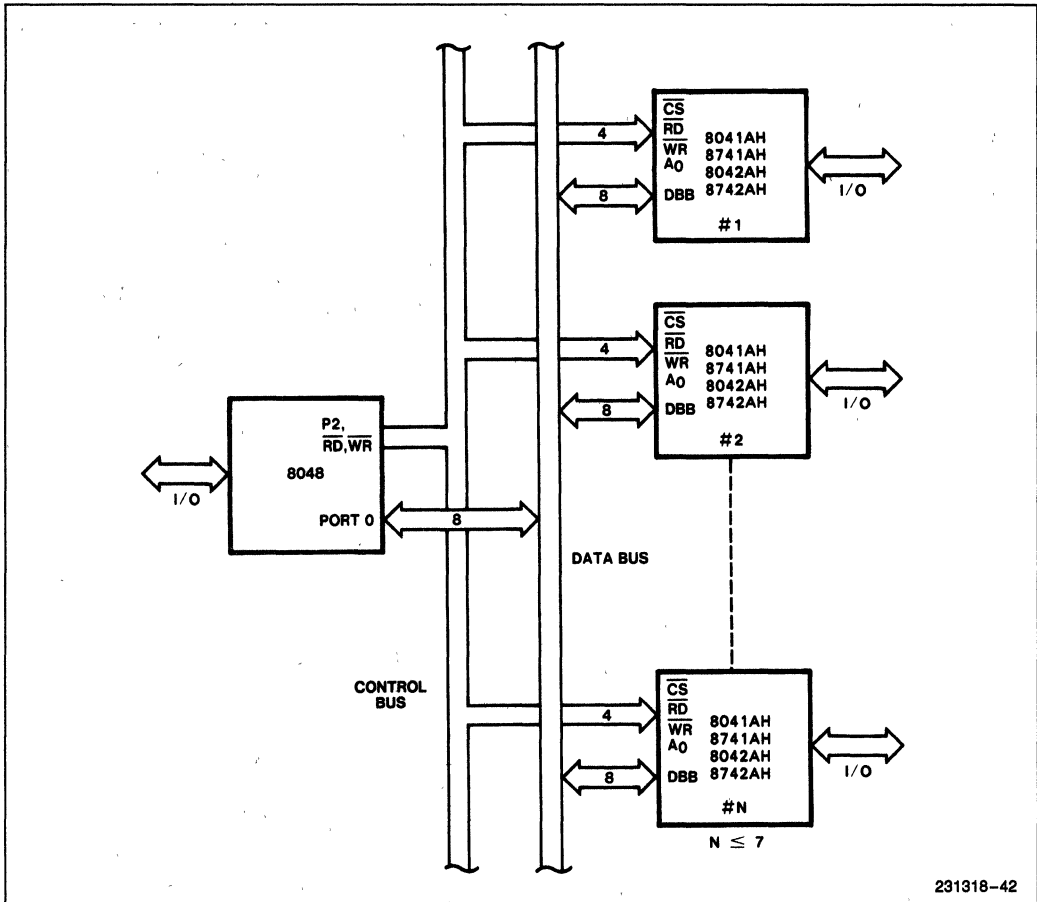
The UPI can be addressed as an I/O device using INPUT and OUTPUT instructions in 8080 software.

8048 Interface

Figure 5-10 shows the UPI interface to an 8048 master processor.

The 8048 \overline{RD} and \overline{WR} outputs are directly compatible with the UPI. Figure 5-11 shows a distributed processing system with up to seven UPI's connected to a single 8048 master processor.

In this configuration the 8048 uses PORT 0 as a data bus. I/O PORT 2 is used to select one of the seven



231318-42

Figure 5-11. Distributed Processor System

Chapter 6 APPLICATIONS

ABSTRACTS

The UPI-41A is designed to fill a wide variety of low to medium speed peripheral interface applications where flexibility and easy implementation are important considerations. The following examples illustrate some typical applications.

Keyboard Encoder

Figure 6-1 illustrates a keyboard encoder configuration using the UPI and the 8243 I/O expander to scan a 128-key matrix. The encoder has switch matrix scanning logic, N-key rollover logic, ROM look-up table, FIFO character buffer, and additional outputs for display functions, control keys or other special functions.

PORT 1 and PORTS 4-7 provide the interface to the keyboard. PORT 1 lines are set one at a time to select the various key matrix rows.

When a row is energized all 16 columns (i.e., PORTS 4-7 inputs) are sampled to determine if any switch in the row is closed. The scanning software is code effi-

cient because the UPI instruction set includes individual bit set/clear operations and expander PORTS 4-7 can be directly addressed with single, 2-byte instructions. Also, accumulator bits can be tested in a single operation. Scan time for 128 keys is about 10 ms. Each matrix point has a unique binary code which is used to address ROM when a key closure is detected. Page 3 of ROM contains a look-up table with useable codes (i.e., ASCII, EBCDIC, etc.) which correspond to each key. When a valid key closure is detected the ROM code corresponding to that key is stored in a FIFO buffer in data memory for transfer to the master processor. To avoid stray noise and switch bounce, a key closure must be detected on two consecutive scans before it is considered valid and loaded into the FIFO buffer. The FIFO buffer allows multiple keys to be processed as they are depressed without regard to when they are released, a condition known as N-key rollover.

The basic features of this encoder are fairly standard and require only about 500 bytes of memory. Since the UPI is programmable and has additional memory capacity it can handle a number of other functions. For example, special keys can be programmed to give an entry on closing as well as opening. Also, I/O lines are

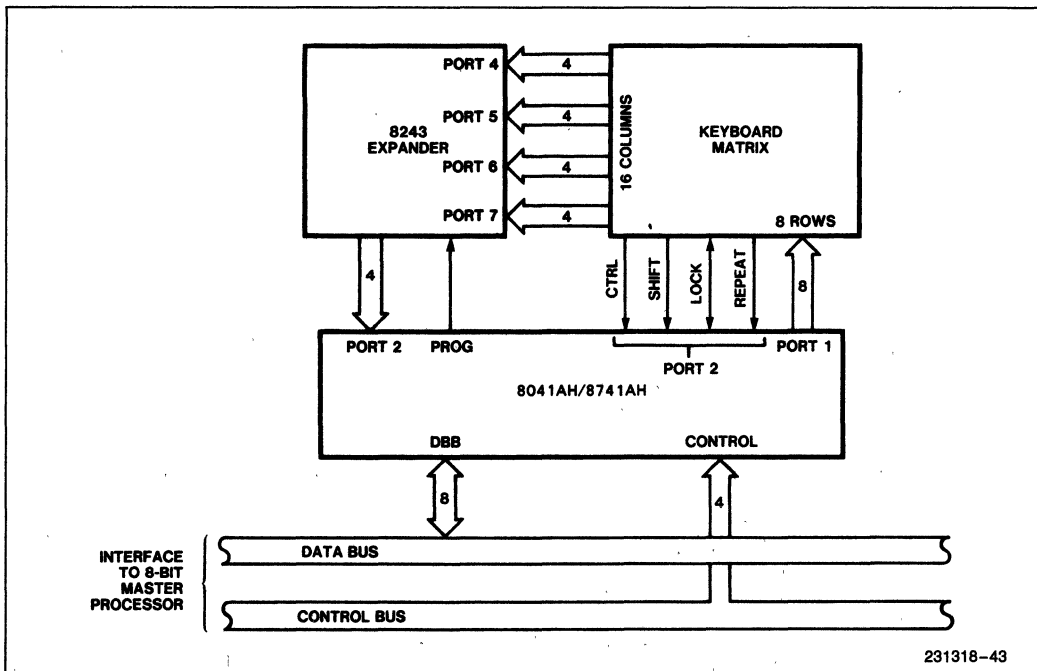


Figure 6-1. Keyboard Encoder Configuration

available to control a 16-digit, 7-segment display. The UPI can also be programmed to recognize special combinations of characters such as commands, then transfer only the decoded information to the master processor.

A complete keyboard application has been developed for the UPI-41A. A description is included in this section. The code for the application is available in the Intel Insite Library (program AB 147).

Matrix Printer Interface

The matrix printer interface illustrated in Figure 6-2 is a typical application for the UPI-41A. The actual printer mechanism could be any of the numerous dot-matrix types and similar configurations can be shown for drum, spherical head, daisy wheel or chain type printers.

The bus structure shown represents a generalized, 8-bit system bus configuration. The UPI's three-state interface port and asynchronous data buffer registers allow it to connect directly to this type of system for efficient, two-way data transfer.

The UPI's two on-board I/O ports provide up to 16 input and output signals to control the printer mechanism. The timer/event counter is used for generating a timing sequence to control print head position, line feed, carriage return, and other sequences. The on-board program memory provides character generation for 5 x 7, 7 x 9, or other dot matrix formats. As an added feature a portion of the 64 x 8-bit data memory can be used as a FIFO buffer so that the master processor can send a block of data at a high rate. The UPI can then output characters from the buffer at a rate the printer can accept while the master processor returns to other tasks.

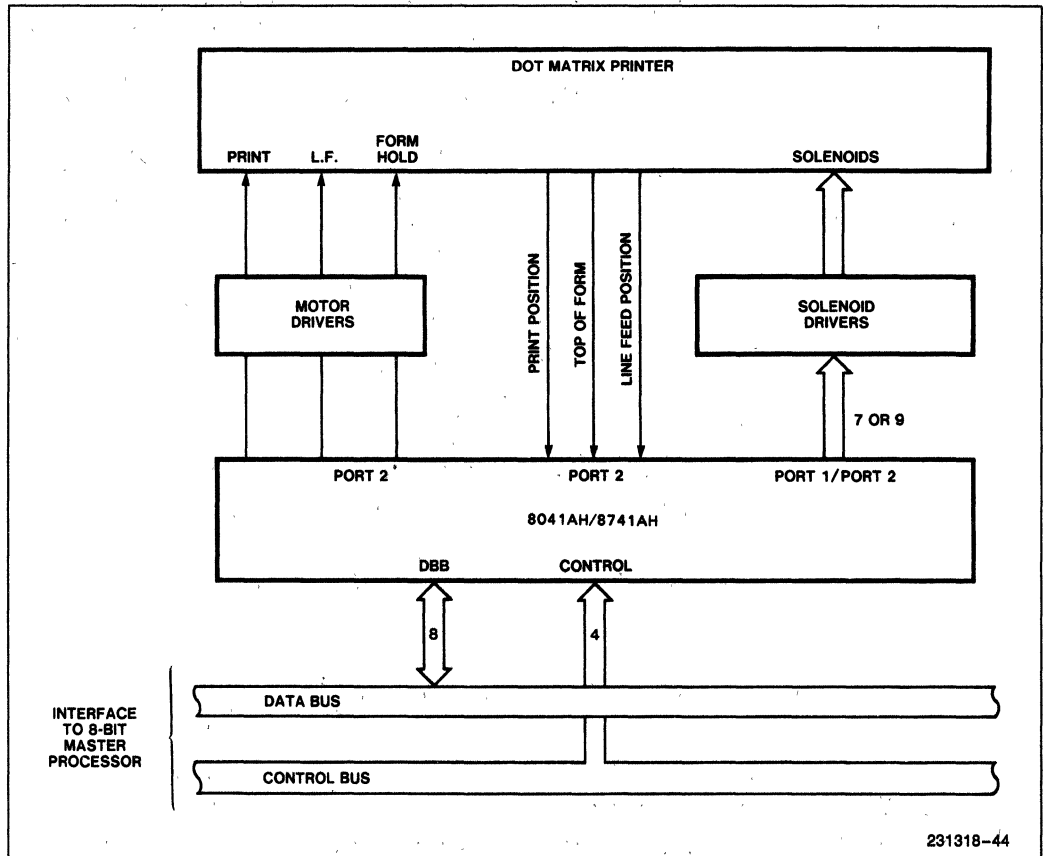


Figure 6-2. Matrix Printer Controller

231318-44

The 8295 Printer Controller is an example of an 8041A preprogrammed as a dot matrix printer interface.

Tape Cassette Controller

Figure 6-3 illustrates a digital cassette interface which can be implemented with the UPI-41A. Two sections of the tape transport are controlled by the UPI: digital data/command logic, and motor servo control.

The motor servo requires a speed reference in the form of a monostable pulse whose width is proportional to the desired speed. The UPI monitors a prerecorded clock from the tape and uses its on-board interval timer to generate the required speed reference pulses at each clock transition.

Recorded data from the tape is supplied serially by the data/command logic and is converted to 8-bit words by the UPI, then transferred to the master processor. At 10 ips tape speed the UPI can easily handle the 8000 bps data rate. To record data, the UPI uses the two input lines to the data/command logic which control the flux direction in the recording head. The UPI also monitors 4 status lines from the tape transport including: end of tape, cassette inserted, busy, and write permit. All control signals can be handled by the UPI's two I/O ports.

Universal I/O Interface

Figure 6-4 shows an I/O interface design based on the UPI. This configuration includes 12 parallel I/O lines and a serial (RS232C) interface for full duplex data transfer up to 1200 baud. This type of design can be used to interface a master processor to a broad spectrum of peripheral devices as well as to a serial communication channel.

PORT 1 is used strictly for I/O in this example while PORT 2 lines provide five functions:

- P₂₃-P₂₀ I/O lines (bidirectional)
- P₂₄ Request to send (RTS)
- P₂₅ Clear to send (CTS)
- P₂₆ Interrupt to master
- P₂₇ Serial data out

The parallel I/O lines make use of the bidirectional port structure of the UPI. Any line can function as an input or output. All port lines are automatically initialized to 1 by a system RESET pulse and remain latched. An external TTL signal connected to a port line will override the UPI's 50 KΩ internal pull-up so that an INPUT instruction will correctly sample the TTL signal.

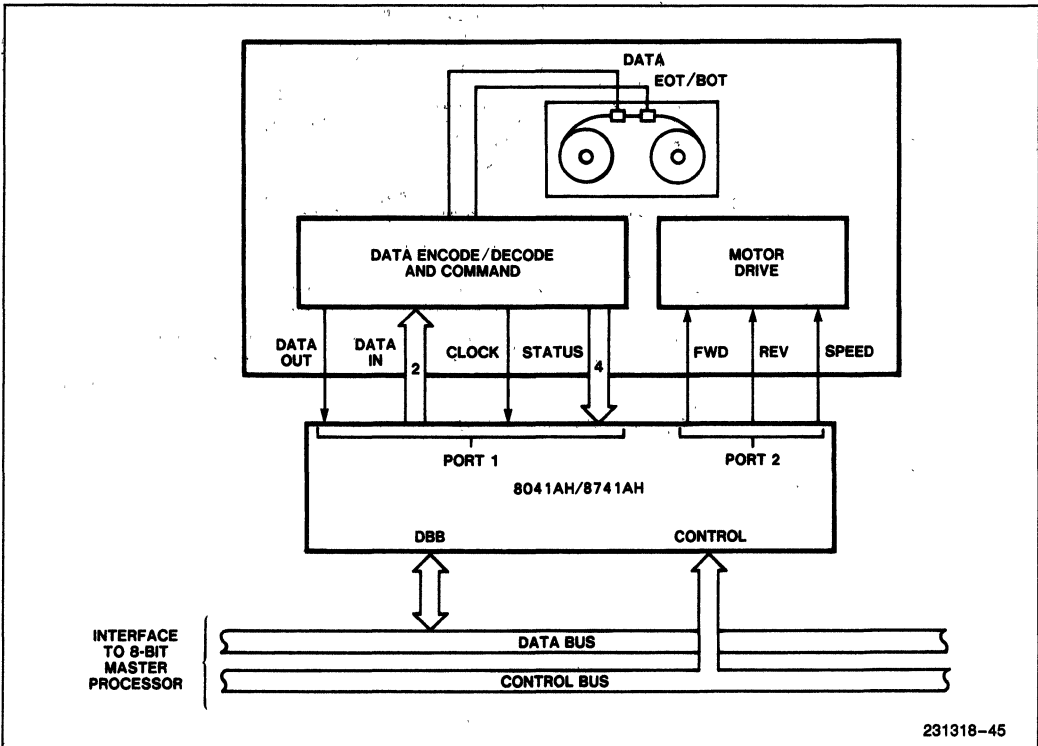


Figure 6-3. Tape Transport Controller

Four PORT 2 lines function as general I/O similar to PORT 1. Also, the RTS signal is generated on PORT 2 under software control when the UPI has serial data to send. The CTS signal is monitored via PORT 2 as an enable to the UPI to send serial data. A PORT 2 line is also used as a software generated interrupt to the master processor. The interrupt functions as a service request when the UPI has a byte of data to transfer or when it is ready to receive. Alternatively, the EN FLAGS instruction could be used to create the OBF and IBF interrupts on P₂₄ and P₂₅.

The RS232C interface is implemented using the TEST 0 pin as a receive input and a PORT 2 pin as a transmit output. External packages (A₀, A₁) are used to provide RS232C drive requirements. The serial receive software is interrupt driven and uses the on-chip timer to perform time critical serial control. After a start bit is detected the interval timer can be preset to generate an interrupt at the proper time for sampling the serial bit stream. This eliminates the need for software timing

loops and allows the processor to proceed to other tasks (i.e., parallel I/O operations) between serial bit samples. Software flags are used so the main program can determine when the interrupt driven receive program has a character assembled for it.

This type of configuration allows system designers flexibility in designing custom I/O interfaces for specific serial and parallel I/O applications. For instance, a second or third serial channel could be substituted in place of the parallel I/O if required. The UPI's data memory can buffer data and commands for up to 4 low-speed channels (110 baud teletypewriter, etc.)

Application Notes

The following application notes illustrate the various applications of the UPI family. Other related publications including the *Microcontroller Handbook* are available through the Intel Literature Department.

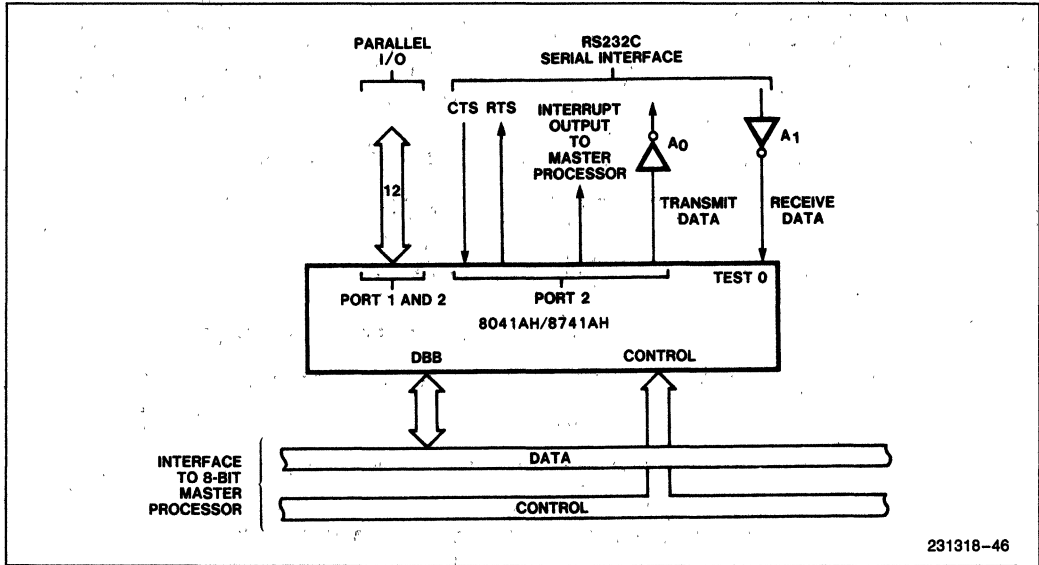


Figure 6-4. Universal I/O Interface



June 1985

Applications Using the 8042 UPI™ Microcontroller

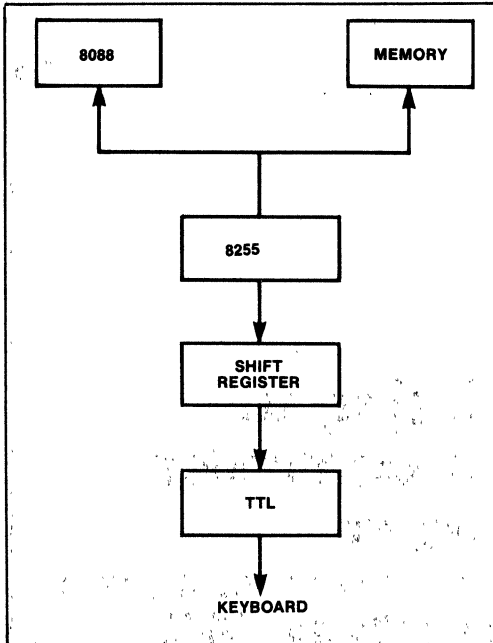
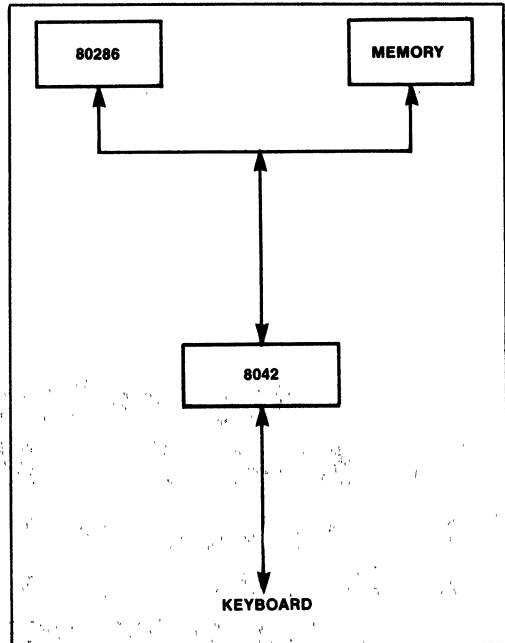
- 1. The 8042 in the IBM PC/AT**
- 2. Using the 8042 vs. using microcontrollers**
- 3. Custom serial protocol with the 8042**

Joe Froelich

Order Number: 231600-001

APPLICATION #1 THE 8042 UPI™ MICROCONTROLLER IN THE IBM PC/AT

The following example is an important application of UPIs but there are many more. It is truly a universal device that can be *customized* to all those “non-standard” peripheral control problems. Applications are limited only by imagination. Think UPIs for non-standard peripherals!!

IBM PC/AT (BEFORE) . . .

. . . IBM PC/AT (AFTER)


NEW FUNCTIONS

The 8042 also brings new functions to the PC/AT:

- Keyboard lockup security (front panel key)
- CRT type input to the system
- Diagnostics/self testing of keyboard interface
- Parity check and retry
- PC and PC/AT keyboard interchangeability
- Reset CPU to compatible mode
- Address wrap-around protect in compatible mode

THE FUTURE IS THE KEY

Modifications and upgrades are easy because of the 8042's programmable flexibility and power:

- Change keyboard scan codes (in 8042 ROM)
- Increase functionality of keyboard interface through software and/or unused I/O lines on 8042
- Control other PC/AT functions with these I/O lines

Summary

In short, IBM used the 8042 since it:

- Offloads housekeeping details from the CPU
- Facilitates modular system design
- Offers a customized peripheral
- Provides a clean, efficient upgrade path

These benefits can apply to many of your applications also.

APPLICATION #2 USING THE 8042 VS. USING MICROCONTROLLERS

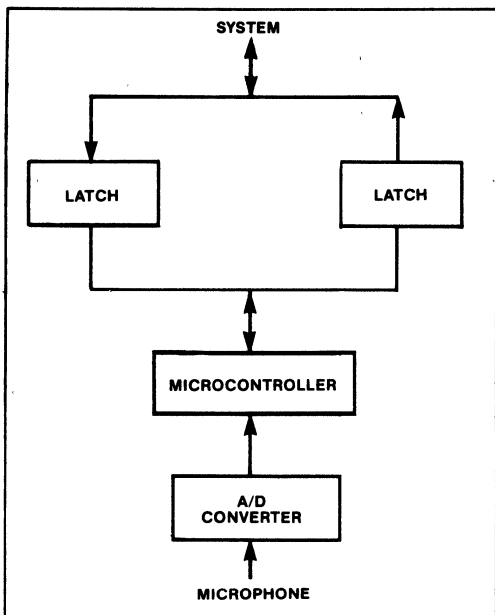
PROBLEM

What do you do when you're making SBX and VME modules for a voice digitizing board and you need:

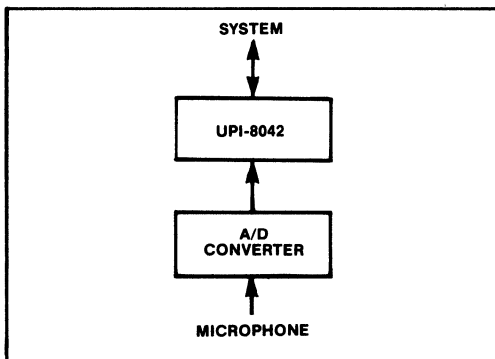
- 1) an interface to an A/D Converter with
- 2) 12 MHz operation,
- 3) an absolute minimum chip count, and
- 4) very low cost (for the PC market).

A leading vendor was faced with exactly this problem. Here is what they started with, and the bottom figure shows how they improved things with the 8042 UPI™ microcontroller.

SOLUTION BEFORE . . .



AFTER . . .



The 8042 integrates two latches and the microcontroller into a single-chip solution.

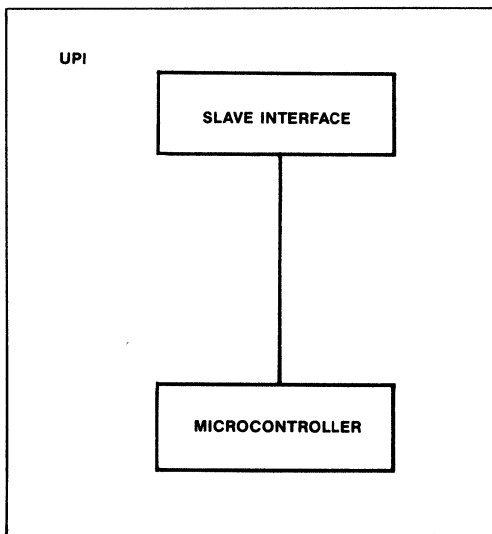
WHY THE SWITCH

After studying the four requirements for this module, it is easy to see why they switched. The first two (A/D interface and 12 MHz) were met by both solutions. However, it is clear the second alternative is much better on board space and on overall cost. There are fewer chips, so they could avoid a multi-layer board and thus save a lot in total cost. Actual chip costs are within 10% of each other (a typical microcontroller like a Z8 or 6801 plus 2 latches compared to an 8042), and they do the same thing.

WHAT'S THE DIFFERENCE

People tend to think of microcontrollers whenever there is a "non-standard" device to control. CRTs, disk drives and DRAMs all have dedicated controllers, but printers, front panels, displays and keyboards don't, because they are all "non-standard" devices. Microcontrollers can be customized to these applications.

The problem is when the device is a "slave" or a peripheral, regular microcontrollers need the extra circuitry shown previously. That's why we invented UPIs. They are simply microcontrollers with the slave interface built in. They are, therefore, more efficient to use in peripheral-type configurations.



UPIs may be misunderstood because of the funny name. They shouldn't be. It's really simple. When faced with non-standard device control, think microcontrollers.

If it's a master-only configuration, think regular microcontrollers. If it's a slave/peripheral configuration, think UPIs.

APPLICATION #3 CUSTOM SERIAL PROTOCOL WITH THE 8042 UPI™ MICROCONTROLLER

BACKGROUND

The 8042 UPI Microcontroller, because of its programmability is being used everywhere, and here is another example. A leading board vendor was designing a communications concentrator board. They wanted a way to:

- 1) interface 8 serial channels to a minicomputer bus
- 2) operate these at 4800 baud
- 3) use one board
- 4) provide a custom serial protocol that
 - communicated commands and data packets
 - assembled the data packets
 - provided handshaking signals
 - checked for framing, timing, parity, noise, modem and synchronization errors
 - provided self-test diagnostics

THE 8042 SOLUTION

There certainly wasn't an "off-the-shelf" chip that would satisfy the above requirements, and using the main

CPU would have caused tremendous system performance degradation. They needed all of these features to offer a competitive product, so they looked to the 8042 UPI Microcontroller. Since the speed requirements were not too great (4800 baud), they could implement the protocol in software. The 8042's programmability gave them all the flexibility needed to incorporate the formatting, handshaking and error checking desired. Moreover, the on-chip slave interface made communication with the minicomputer's bus a snap.

SUMMARY

In short, the 8042 allowed this company to implement a custom serial communication protocol that in turn allowed them to offer a customized, competitive interface board to their customers.

◆ **Don't some of your applications need customized interfaces?**



**APPLICATION
NOTE**

AP-161

November 1986

**Complex Peripheral Control with
the UPI-42AH**

**CHRISTOPHER SCOTT
TECHNICAL MARKETING ENGINEER**

Order Number: 230795-002

INTRODUCTION

The UPI-42AH represents a significant growth in UPI capabilities resulting in a broader spectrum of applications. The UPI-42AH incorporates twice the EPROM/ROM of the UPI-41AH, 2048 vs 1024 bytes, 256 bytes of RAM, and operates at a maximum speed twice that of the UPI-41AH, i.e., 12MHz vs 6MHz. The ROM based 8042 and the EPROM based 8742AH provide more highly integrated solutions for complex stepping motor and dot matrix printer applications. Those applications previously requiring a microprocessor plus a UPI chip can now be implemented entirely with the UPI-42AH.

The software features of the UPI-42AH, such as indirect Data and Program Memory addressing, two independent and selectable 8 byte register banks, and directly software testable I/O pins, greatly simplify the external interface and software flow. The software and hardware design of the UPI-42AH allows a complex peripheral to be controlled with a minimum of external hardware.

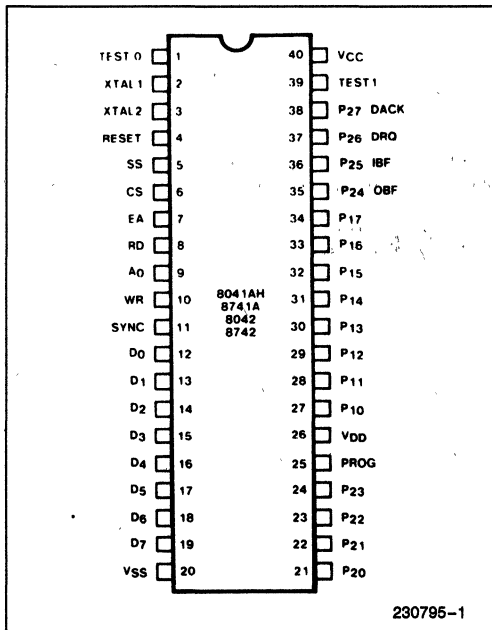


Figure 1. UPI-42AH Pin Configuration

Many microcomputer systems need real time control of peripheral devices such as a printer, keyboard, complex motor control or process control. These medium speed but still time consuming tasks require a fair amount of system software overhead. This processing burden can be reduced by using a dedicated peripheral control processor.

Until recently, the dedicated control processor approach was usually not cost effective due to the large number of components needed; CPU, RAM, ROM, I/O, and Timer/Counters. To help make the approach more cost effective, in 1977 Intel introduced the UPI-41 family of Universal Peripheral Interface controllers consisting of an 8041AH (ROM) device and an 8741AH (EPROM) device. These devices integrated the common microprocessor system functions into one 40 pin package. The UPI-42AH family, consisting of the 8042 and 8742AH, further extends the UPI's cost effectiveness through more memory and higher speed.

Another member of the UPI family is the Intel 8243 Input/Output Expander chip. This chip provides the UPI-41AH and UPI-42AH with up to 16 additional independently programmable I/O lines, and interfaces directly to the UPI-41AH/42AH. Up to seven 8243s can be cascaded to provide over 100 I/O lines.

The UPI is a single chip microcomputer with a standard microprocessor interface. The UPI's architecture, illustrated in Figure 3, features on-chip program memory, ROM (8041A/8042) or EPROM (8741A/8742AH), data memory (RAM), CPU, timer/counter, and I/O. Special interface registers are provided which enable the UPI to function as a peripheral to an 8-bit central processor.

Using one of the UPI devices, the designer simply codes his proprietary peripheral control algorithm into the UPI device itself, rather than into the main system software. The UPI device then performs the peripheral control task while the host processor simply issues commands and transfers data. With the proliferation of microcomputer systems, the use of UPIs or slave microprocessors to off load the main system microprocessor has become quite common.

This Application Note describes how the UPI-42AH can be used to control dot matrix printing and the printer mechanism, using stepper motors for carriage/print head assembly and paper feed motion. Previous Intel Application Notes AP-27, AP-54, and AP-91 describe using intelligent processors and peripherals to control single solenoid driven printer mechanisms with 80 character line buffering and bidirectional printing. This Application Note expands on these previous themes and extends the concept of complex device control by incorporating full 80 character line buffering, bidirectional printing, as well as drive and feedback control of two four phase stepper motors.

The Application Note assumes that the reader is familiar with the 8042/8742AH and 8243 Data Sheets, and UPI-41AH/42AH Assembly Language. Although some background information is included, it also assumes a basic understanding of stepper motors and dot matrix printer mechanisms. A complete software listing is included in Appendix A.

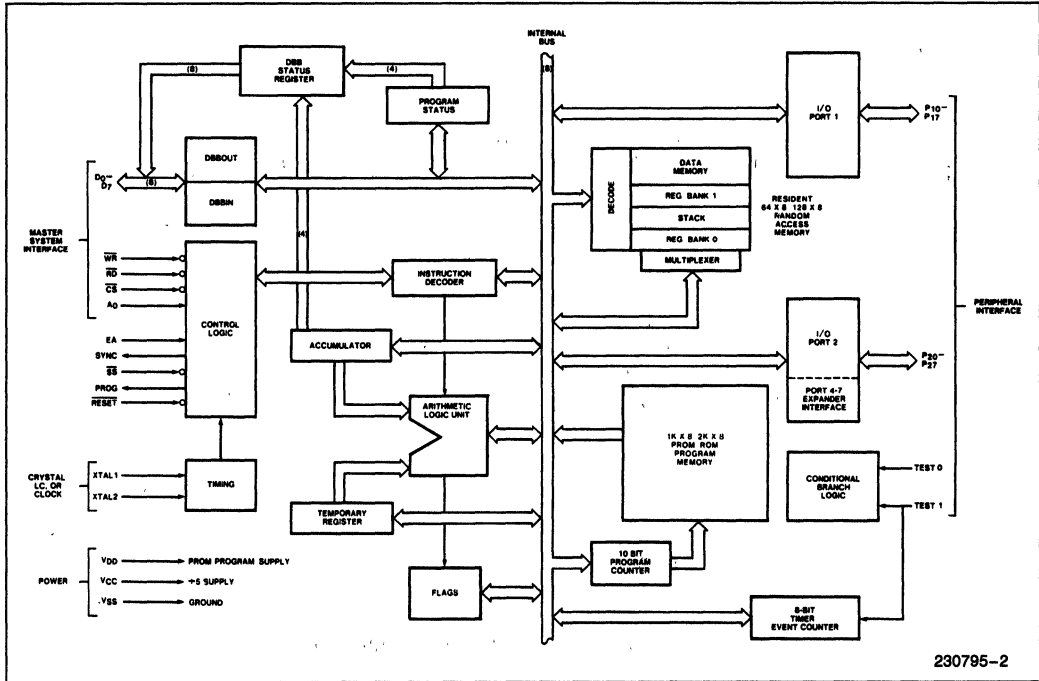


Figure 2. UPI-42AH Block Diagram

DOT MATRIX PRINTING

A dot matrix printer print head typically consists of seven to nine solenoids, each of which drives a stiff wire, or hammer, to impact the paper through an inked ribbon. Characters are formed by firing the solenoids to form a matrix of "dots" (impacts of the wires). Figure 4 shows how the character "E" is formed using a 5 × 7 matrix. The columns are labeled C1 through C5, and the rows R1 through R7. The print head moves left-to-right across the paper, so that at time T1 the head is over column C1. The character is formed by activating the proper solenoid as the print head sweeps across the character position.

Dot matrix printers are a cost effective way of providing good quality hard copy output for microcomputer systems. There is an ever increasing demand for the moderately priced printer to provide more functionality with improved cost and performance. Using stepper motors to control the paper feed and carriage/print head assembly motion is one way of enabling the dot matrix printer to provide more capabilities, such as expanded or contracted characters, dot or line graphics, variable line and character spacing, and subscript or superscript printing.

However, stepper motors require fairly complex control algorithms. Previous solutions involved the use of a

main CPU, UPI, RAM, ROM, and I/O onboard the peripheral. The CPU acted as supervisor and used parallel processing to achieve accurate stepper motor control via a UPI, character buffering via the I/O device, RAM, and ROM. The CPU performed real-time decoding of each character into a dot matrix pattern. This Application Note demonstrates that the increased memory and performance of the UPI-42AH facilitates integrating these control functions to reduce the cost and component count.

THE PRINTER MECHANISM

The printer mechanism used in this application is the Epson Model 3210. It consists of four basic sub-assemblies; the chassis or frame, the paper feed mechanism and stepper motor, the carriage motion mechanism and stepper motor, and the print head assembly.

The paper feed mechanism is a tractor feed type. It accommodates up to 8.5 inch wide paper (not including tractor feed portion). There is no platen as such; the paper is moved through the paper guide by two sprocketed wheels mounted on a center sprocket shaft. The sprocket shaft is driven by a four phase stepper motor. The rotation of the stepper motor is transmitted to the sprocket shaft through a series of four reduction gears.

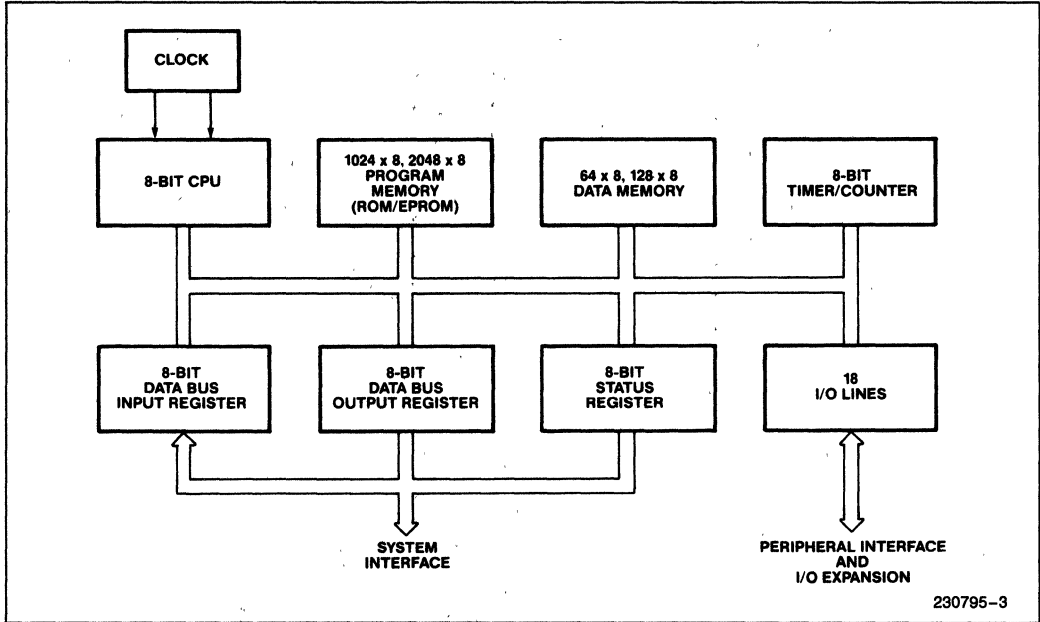


Figure 3. UPI-41AH, 42AH Functional Block Diagram

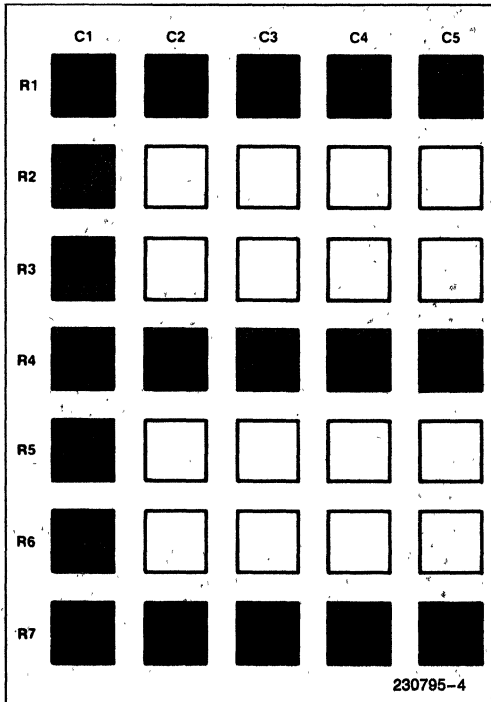


Figure 4. Character E in 5 x 7 Dot Matrix Format

The carriage motion mechanism consists of another four phase stepper motor which controls the left-to-right or right-to-left print head assembly motion. The print speed is 80 CPS maximum. Both the speed of the stepper motor and the movement of the print head assembly are independently controllable in either direction. The rotation of the stepper motor is converted to the linear motion of the print head assembly via a series of reduction gears and a toothed drive belt. The drive belt also controls a second set of reduction gears which advances the print ribbon as the print head assembly moves.

Two optical sensors provide feedback information on the carriage assembly position and speed. The first of these optical sensors, called the 'HOME RESET' or HR, is mounted near the left-most physical position to which the print head assembly can move. As the print head assembly approaches the left-most position, a flange on the print head assembly interferes with the light source and sensor, causing the output of the sensor to shift from a logic level one to zero. The right-most printer position is monitored in software rather than by another optical sensor. The right-most print position is a function of the number of characters printed and the distance required to print them.

The second optical sensor, called the 'PRINT TIMING SIGNAL' or PTS, provides feedback on carriage stepper motor velocity and relative position within a

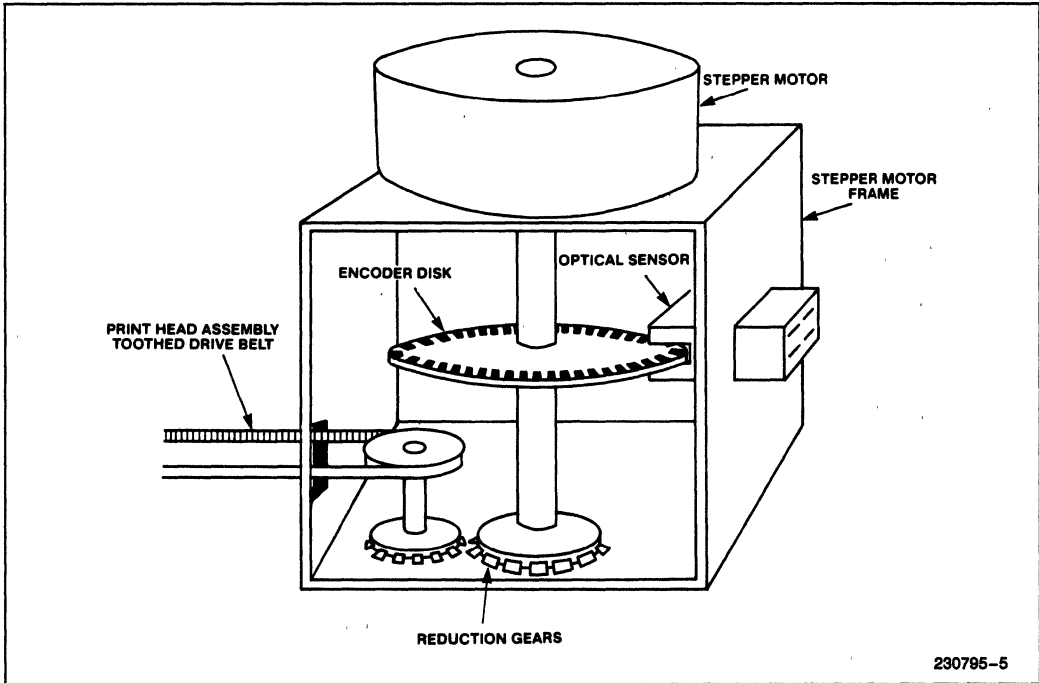


Figure 5. Carriage Stepper Motor Assembly

given step of the motor. The feedback is generated by the optical sensor as an “encoder disk” moves across it. Figure 5 illustrates the carriage stepper motor, optical sensor, encoder disk and reduction gears, and drive belt assembly. The optical sensor outputs a pulse train with the same period as the phase shift signal used to drive the stepper, but slightly out of phase with it when the motor is at a constant speed (see Software Functional Block: Phase Shift Data for additional details). The disk acts as a timing wheel, providing feedback to the UPI software of the carriage speed, position, and optimum position for energizing the print head solenoids. The two optical sensors are monitored under software and provide the critical feedback needed to control the print head assembly and paper feed motion accurately. The process of stepper motor drive and control via feedback signals is called “closed loop” stepper motor control, and is covered in more detail in the software discussion.

The print head assembly consists of nine solenoids and nine wires or hammers. Figure 6 illustrates a print head assembly. The available dot matrix measures 9 x 9. This large matrix enables the Epson 3210 print mechanism

to print a variety of character fonts, such as expanded or contracted characters, as well as line or block graphics (see Appendix B, Printer Enhancements). It also facilitates printing lower case ASCII characters with “lower case descenders.” That is to say, certain lower case letters (e.g., y, p, etc.) will print below the bottom part of all upper case letters.

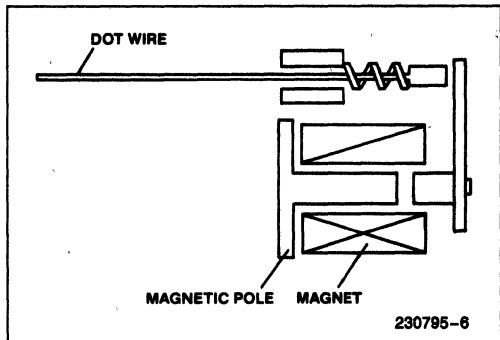


Figure 6. Print Head Solenoid Assembly

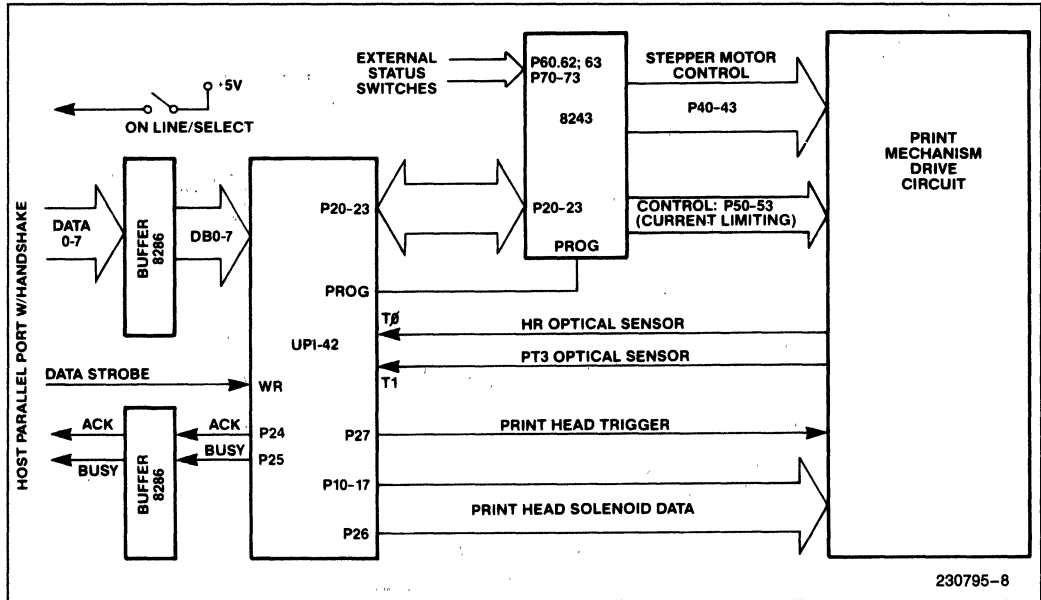


Figure 7. Hardware Interface Block Diagram

HARDWARE DESCRIPTION

Figure 7 shows a block diagram of the UPI-42AH and 8243 interface to the printer mechanism drive circuit. A complete schematic is shown in Figure 8. The UPI-42AH provides all signals necessary to control character buffering and handshaking, paperfeed and carriage motion stepper motor timing, print head solenoid activation, and monitoring of external status switches.

The Epson 3210 printer mechanism manual recommends a specific interface circuit to provide proper drive levels to the stepper motors windings and print head solenoids. The hardware interface used for this

Application Note followed those recommendations exactly (see Appendix C, Printer Mechanism Drive Circuit Schematics).

I/O Ports

The lower half of the UPI-42AH Port 2, pins 0-3, provides an interface to the 8243 I/O expander. The PROG pin of the UPI-42AH is used as a strobe to clock address and data information via the Port 2 interface. The extra 16 I/O lines of the 8243 become PORTS 4, 5, 6, and 7 to the UPI software. Combined, the UPI-42AH and 8243 provide a total of 28 independently programmable I/O line. These lines are used as shown in Figure 9.

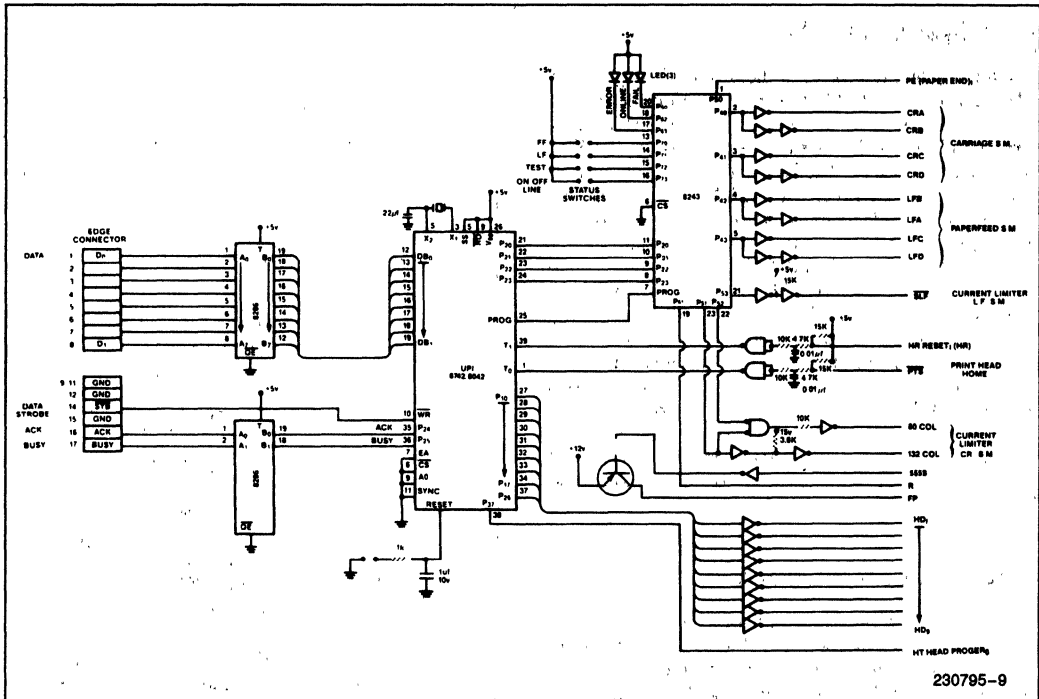


Figure 8. Hardware Interface Schematic

Port	No. of lines	Bits	I/O	Description
1	8	0-7	O	Character dot column data to print head solenoids (same)
2	1	6	O	Print head solenoid trigger
2	1	7	O	Host system data transfer handshaking (ACK/BUSY)
4	2	4, 5	O	Carriage & paper feed stepper motors
4	4	0-3	O	Stepper motor select and current limiting
5	3	1-3	O	Paper End sense
5	1	0	I	Paper End sense
6	1	1	O	Print head trigger reset
6	3	0, 2, 3	-	(unused)
7	5	0-3	I	External status switches; (LF, FF, TEST, ON/OFF Line)

Figure 9. UPI-42AH and 8243 I/O Port Map

NOTE:

The notation used in the balance of this Application Note, when referring to a port number and a particular pin or bit, is Port 23 rather than Port 2 bit 3.

The two printer mechanism optical sensors, discussed in the Printer Mechanism description are tied to the two "Test Input" pins, T0 and T1, of the UPI-42AH through a buffer circuit for noise suppression. These inputs are directly testable in software.

Host System Interface

The host system interfaces to the printer through a parallel port to the UPI-42AH Data Bus. Four handshaking signals are used to control data transfer; Data Strobe (STB/), Acknowledge (ACK), Busy (BUSY), and Online or Select. The Data Strobe line of the host parallel port is tied directly to the UPI-42AH WR/ pin. This provides a low going pulse on the UPI-42AH WR/ pin whenever a data byte is written to the UPI-42AH. The ACK and BUSY handshake signals are tied to two UPI-42AH I/O port lines for software control of data transfer. The "On Line" handshake signal is tied to a single-pole single-throw fixed position switch, which externally enables or disables character transfer from the host system. Characters transmitted to the UPI-42AH by the host are loaded into the UPI-42AH Data Bus Buffer In (DBBIN) register, and the Input Buffer Full (IBF) interrupt and UPI-42AH status flag are set (see Figure 9. UPI-42AH and 8243 I/O Ports).

Stepper Motor Interface

Port 4 (41-43) of the 8243, provides both carriage and paper feed stepper motor phase shift signals to the printer mechanism drive circuit. Each of the two stepper motors is driven by 2 two phase excitation signals (4 phases). Figure 10 shows the wave form for each stepper motor. Each signal consists of two components (Sig. 1 A/B & Sig. 2 C/D) 180 degrees out of phase with the other. Each of these signal pairs (A/B & C/D) is 90 degrees out of phase with the other pair. For each signal pair, one port line supplies both halves by using an inverter.

Each of the resulting eight stepper motor drive signals is interfaced to a discrete drive transistor through an inverter. The emitter of the drive transistor is tied to the open collector of the inverter to provide high current sinking capability for the drive transistor. Each half of the motor winding is tied to the collector of the drive transistor (see Appendix C, Printer Mechanism Drive Circuit Schematic).

Each stepper motor requires two current levels for operation. These levels are called "Rush" current and "Hold" current. Rush current refers to the high current required to cause the rotor to rotate within its windings as the polarity of the power applied to the windings is changing. Each change in the polarity of the power applied to the motor windings is called a step of phase shift. Hold current refers to the low level of current required to stabilize and maintain the rotor in a fixed position when the polarity is applied to the windings is not changing. Hold current is simply Rush current with a current limiting transistor switched in. Switching from Hold to Rush current "selects" or enables that

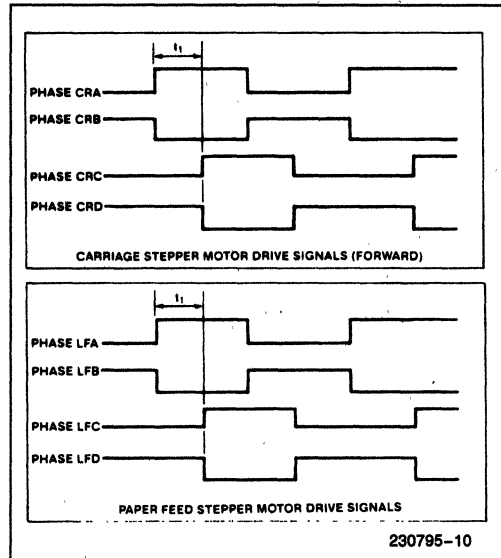


Figure 10. Stepper Motor Step Sequence Waveforms

stepper motor to move with the next step signal output. In the balance of this Application Note, the term "select" will be used to refer to turning on Rush current, and "deselect" will refer to switching to Hold current.

Three 8243 port lines are dedicated to the select/deselect control of the two stepper motors. One line is for the paper feed stepper motor (80 and 132 column). These lines are labeled SLF, 80Col, and 132Col, and are 8243 PORT 53, 52, and 51, respectively.

By varying the voltage applied to the stepper motor biasing circuit and the current, it is possible to vary the distance the motor moves the print head assembly with each step. Enabling one of two different voltage biasing levels, and changing the timing rate at which the motor is stepped, facilitates either 80 or 132 character column printing. Only 80 character column printing is implemented in the software design. Appendix B, Printer Enhancements, details the software algorithm for handling 132 character printing.

Print Head Interface

A total of eleven I/O lines are used to control the print head solenoids and solenoid firing (see Figure 9). Nine are used for character dot data, one for the Print Head Trigger, and one for Reset of the Print Head Trigger circuit. Each of the nine character dot data lines is buffered by an open collector hex inverter.

The Print Head Trigger output is tied to the Trigger input of a 555 Monostable Multivibrator. The output pulse generated by the 555 triggers the print head solenoids to fire. The 555 Output pulse width is independent of the input trigger waveform. The pulse width is determined by an RC network across the 555 inputs and the voltage level applied to the Control Voltage 555 input. The 555 Output is tied to the base of a PNP transistor through an inverter, biased in a normally off configuration. The PNP transistor supplies enough drive to pull up the open collector inverter on each print head solenoid line, Port 10-17 and 26. The 555 output pulse momentarily enables the print head solenoid line open collector inverter output, turning on the solenoid drive transistor, and firing the print head hammer. The 555 Output pulse width is approximately 400 us. Further details of the print head firing operation can be found in the software description below.

Miscellaneous Interface Signals

The 8243 Port 5 pin 0 is tied to the Paper End Detector, a reed switch located on the printer paper guide. This sensor detects when the paper is nearly exhausted.

Three LED status lights complete the hardware interface design. One status light is used for each of the following: Power ON/OFF, On/Off Line, and Out of Paper.

BACKGROUND

Before a detailed discussion of the software begins, a few terms and software functions reference throughout the software need introduction.

A. What is a Stepper Motor?

A stepper motor has the ability to rotate in either direction as well as start and stop at predetermined angular positions. The stepper motor's shaft (rotor) moves in precise angular increments for each input step. The displacement is repeated for each input step command, accurately positioning the rotor for a given number and sequence of steps.

The stepper motor controls position, velocity, and direction. The accuracy of stepper motors is generally 5 percent of one step. The number of steps in each revolution of the shaft varies, depending on the intended application.

B. Open/Closed Loop Stepper Motor Drive and Control

The carriage stepper motor is closed loop driven. The paper feed stepper motor is open loop driven.

There are two major types of stepper motor control known by the broad headings of open and closed loop. Open loop is simply continuous pulses to drive the motor at a predetermined rate based on the voltage, current, and the timing of the step pulses applied. Closed loop control is characterized by continuous monitoring of the stepper motor, through feedback signals, and adjusting the motor's operation based upon the feedback received.

C. Stepper Motor Drive Phase Shift of Step Sequence

Each change in the polarity of the power applied to the motor windings is called a step or phase shift. The sequence of the steps or phase shifts, and the pattern of polarity changes output to the stepper motor, determines the direction of rotation.

Figure 10 shows the waveforms for each of the two stepper motors. Figure 11 lists the step sequence for carriage motor clockwise rotation, which moves the print head assembly Left-to-Right. Figure 11 also lists the step sequence for counterclockwise rotations; the print head assembly moves Right-to-Left. Figure 12 lists the step sequence for the paper feed stepper motor clockwise drive. The phase sequence, for either stepper motor, may begin at any point within the sequence list, but must then continue in order.

Step No.	A-Step	B-Step	C-Step	D-Step
1	On	Off	Off	On
2	On	Off	On	Off
3	Off	On	On	Off
4	Off	On	Off	On

Carriage stepper motor rotates clockwise Print head assembly moves from left to right

Step No.	A-Step	B-Step	C-Step	D-Step
1	On	Off	On	Off
2	On	Off	Off	On
3	Off	On	Off	On
4	Off	On	On	Off

Carriage stepper motor rotates counter clockwise Print head assembly moves from right to left

Figure 11. Carriage Stepper Motor Step Sequence

Step No.	A-Step	B-Step	C-Step	D-Step
1	On	Off	On	Off
2	On	Off	Off	On
3	Off	On	Off	On
4	Off	On	On	Off

Figure 12. Paper Feed Stepper Motor Step Sequence

D. Acceleration and Deceleration of Stepper Motors

The carriage stepper motor starts from a fixed position, accelerates to a constant speed, maintains constant speed, and then decelerates to a fixed position. Printing may occur from the time and position the print head assembly reaches constant speed, until the time and position the print head assembly begins to decelerate from constant speed. Whether printing occurs during any carriage stepper motor drive sequence is controlled by software. Figure 18, below, illustrates these components of print head assembly line motion.

Due to inertia, a finite time interval and angular displacement is required to accelerate a stepper motor to

its full speed. Conversely, deceleration must begin some time before the final angular position. The time interval and angular displacement of the carriage stepper motor translates into the distance the print head assembly travels before it reaches a constant speed. The distance traveled during acceleration is constant. The distance the print head assembly travels during deceleration must be the same as the distance traveled during acceleration in order to accurately align the character dot columns from one line to the next.

E. Stepper Motor Predetermined Time Constant

Whenever the stepper motor is stepped, or energized, the angular velocity of the rotor is greater than the constant speed which is ultimately required. This is called "overshoot." The frictional load of the carriage assembly (motor rotor, reduction gears, drive belt and print head assembly, or paper feed sprocket shaft and wheels) provides damping or frictional load. Damping slows the motor to less than the required constant speed and is called "undershoot" (see Figure 13, Carriage Stepper Motor Drive Timing). A constant rate of speed is achieved through the averaging of the overshoot and undershoot within each step.

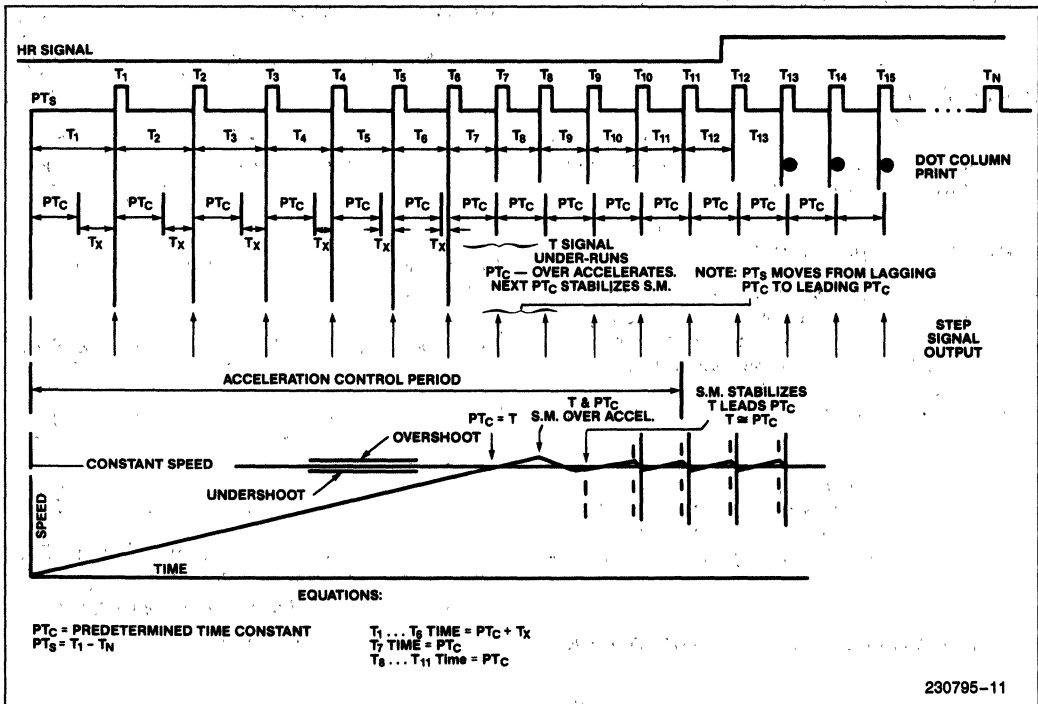


Figure 13. Carriage Stepper Motor Drive Timing.

The Predetermined Time (PT) Constant is the time required to average the overshoot and undershoot of the particular stepper motor for a desired constant rate of speed. The PT also is the time required to move the print head assembly a specific distance, accounting for both overshoot and undershoot of the stepper motor.

Changing the Predetermined Time Constant changes the angular displacement of the stepper motor rotor, this in turn changes the output. Figure 14 lists the Time Constants for both standard and condensed character printing. Figure 15 lists the paper feed stepper motor Time Constants used for various line spacing formats. This Application Note implements standard character print and paper feed (6 lines per inch) Time Constants. See Appendix B, Printer Enhancement, for details on implementing non-standard Time Constants.

Character mode	Predetermined time	
Standard or Enlarged Character	2.08 ms	+ 10% - 4%
Condensed Character	4.16 ms	+ 10% - 4%

Figure 14. Carriage Stepper Motor Predetermined Time Constants

Paper feed pitch	0.12 mm (1/216")/1 pulse
	4.23 mm (1/16")/36 pulses
	3.18 mm (1/8")/27 pulses
	2.82 mm (1/9")/24 pulses
Paper feed time	
150 ms/4.23 mm	Approx. 6.6 lines/s (continuous feed)
113 ms/3.18 mm	Approx. 8.8 lines/s (continuous feed)
100 ms/2.82 mm	Approx. 10 lines/s (continuous feed)

Figure 15. Paper Feed Stepper Motor Predetermined Time Constants

F. Relationship Between PTS and PT

Figure 13 illustrates how PTS lags PT at the start of acceleration, and moves to lead PT as the motor achieves constant speed. Figure 13 also illustrates the relationship between HR, PTS, PT, acceleration, constant speed, and printing. Figures 16 and 17 illustrate the relationship between PTS and PT during acceleration and at constant speed.

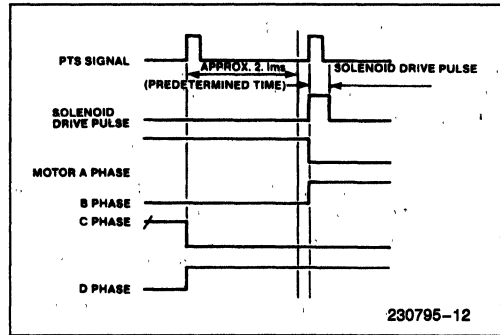


Figure 16. PTS Lags PT Timing

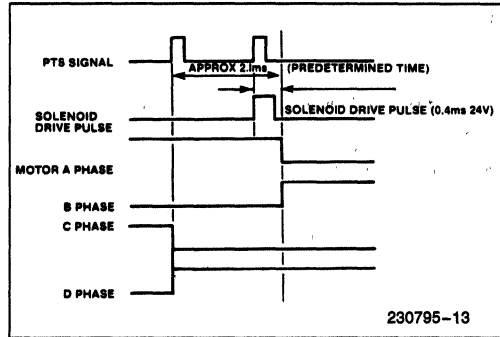


Figure 17. PTS Leads PT Timing

PTS is the point of peak angular velocity within a step of the motor. PTS is a function of the slot spacing on the encoder disk, shown in Figure 5. The spacing is determined by the mechanics of the printer mechanism.

When the carriage stepper motor is accelerated from a fixed position, the effects of damping slows the angular velocity of energizing the stepper motor. This causes PTS to occur after the PT, or PTS lags PT. When PTS lags PT, the next step signal is output at PTS rather than at PT. If the step signal is outputted at PTS, the rotor could be midway through a rotation. Energizing the motor at PT could cause it to bind or shift in the wrong direction. When the carriage stepper motor is at a constant rate of speed, PTS leads PT and the step signal is output at PT (see Figure 13).

G. Stored Time Constants

The time between each step, for a constant number of steps, required for the motor to reach a constant speed, is calculated and stored in Data Memory during acceleration. The values stored are used, in reverse order, during deceleration as the Predetermined Time (PT) Constants. This ensures that the acceleration and deceleration distance traveled by the print head assembly is the same, and that it accurately aligns character dot columns from one line to the next during printing. The time values stored are called "Stored Time Constants." Steps T1 through T11 in Figure 13, represent the Stored Time Constants.

The equations for the Stored Time Constants are given at the bottom of Figure 13, Carriage Stepper Motor Drive Timing.

H. Print Head Assembly "Home" Position

The "logical" Home position for the print head assembly is the left-most position at which printing begins

(for L-to-R motion) or ends (for R-to-L motion). The "physical" Home position is the logical HOME position, plus the distance required by the carriage stepper motor to fully accelerate the print head assembly to a constant speed. Printing can only occur when the print head is moving at a constant speed. The printer mechanism manual stipulates eleven step time periods are required to ensure the print head assembly is at a constant speed. These eleven step time periods are the Stored Time Constants described above. Figure 18 illustrates the components of print head assembly line motion and character printing.

SOFTWARE

Introduction

The software description is presented in three sections. First, a brief overview of the software to familiarize the reader with the interdependencies and overall program flow. Second, data and program memory allocation and status register flag definitions. And third, each of the ten software blocks is presented with its own flowchart.

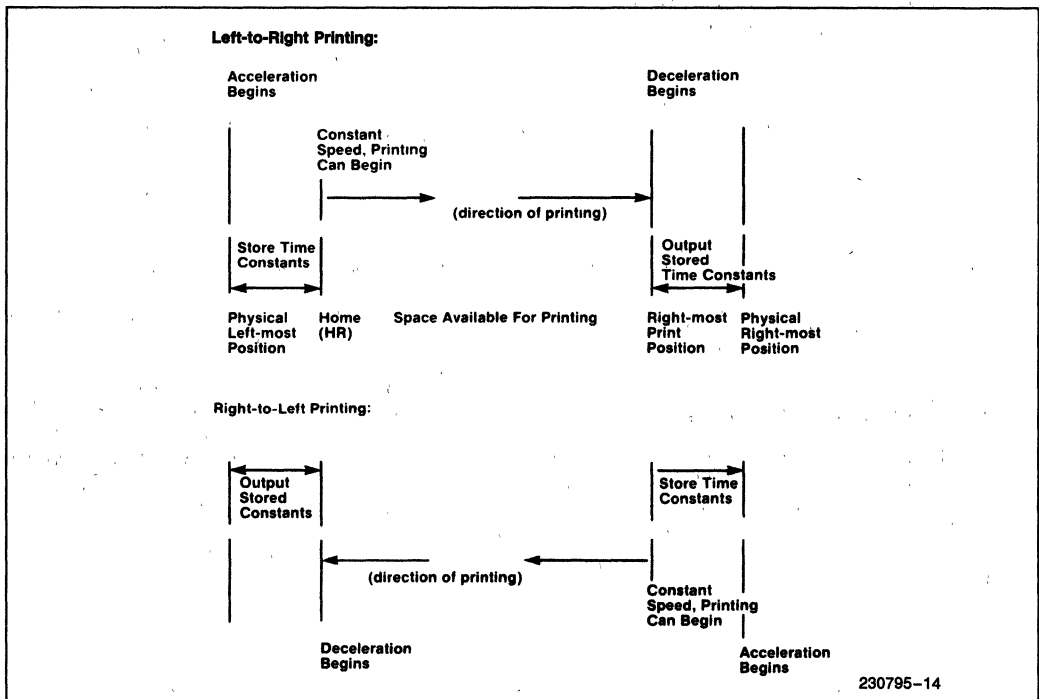


Figure 18. Components of Print Head Assembly Line Motion and Printing

Software Overview

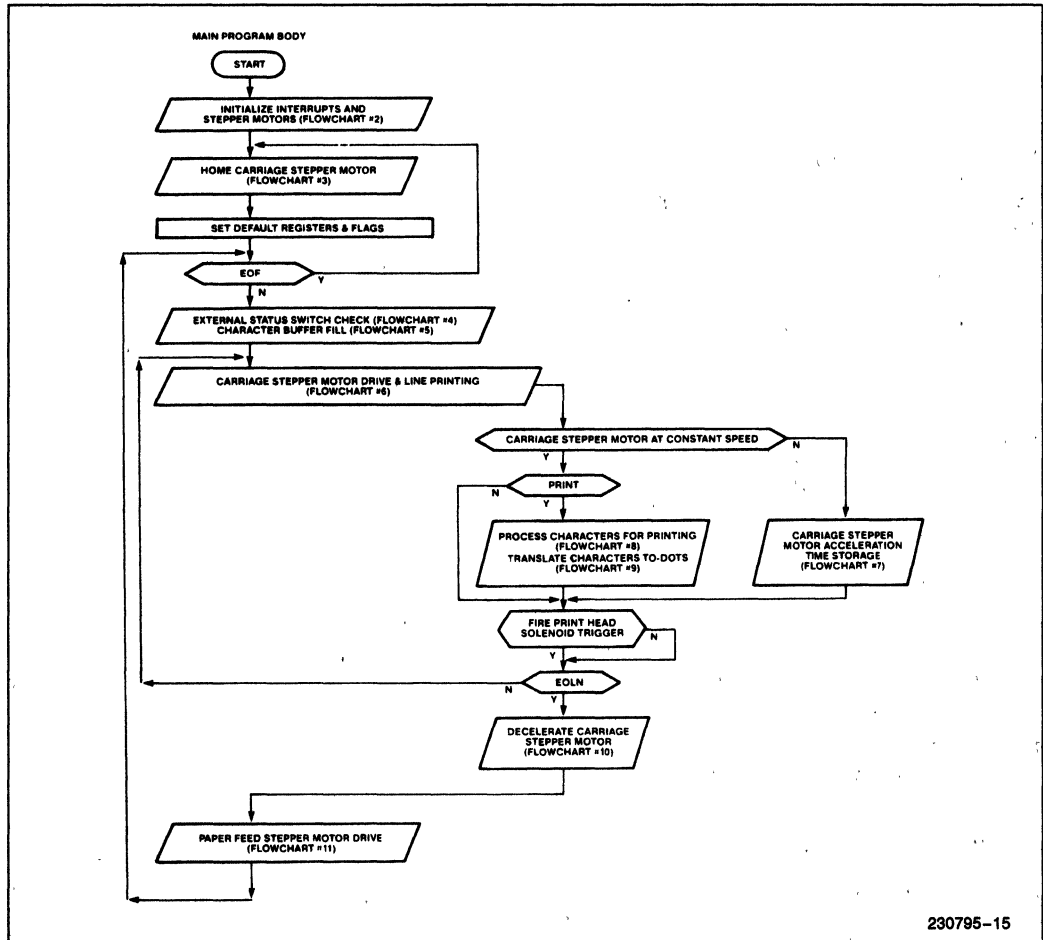
The software is written in Intel UPI-41AH/42AH Assembly Language. A block structure approach is used for ease of development, maintenance, and comprehension. The software is divided into five principal parts.

1. Initialization
2. Character Buffering or Input
3. Stepper Motor Drive and Control
4. Character Processing
5. Character Printing or Output

The five principal parts are incorporated into ten software blocks, listed below.

1. Power On/Reset Initialization
2. Home Print Head Assembly
3. External Status Switch Check
4. Character Buffer Fill
5. Carriage Stepper Motor Drive and Line Printing
6. Accelerate Stepper Motor Time Storage
7. Process Characters for Printing
8. Translate Character-to-Dots
9. Decelerate Carriage Stepper Motor
10. Paperfeed Stepper Motor Drive

Flow Chart No. 1 illustrates the overall software algorithm. Below, is a description of the algorithm.



230795-15

Flow Chart No. 1. Main Program Body.

Upon power-on reset, a software and hardware initialization is performed. This stabilizes and sets inactive the printer hardware and electronics. The print head assembly is then moved to establish its HOME position. The default status registers are set for character buffering, carriage, and paper feed stepper motor drive. The External Status switches are checked; FORMFEED, LINEFEED, ON/OFF LINE, and Character Print TEST. If the printer is ON LINE, the software will loop on filling the Data Memory Character Buffer.

Character or data input to the UPI-42AH is interrupt driven. Characters sent by the host system set the Input Buffer Full (IBF) interrupt and the IBF Program Status flag. Character input servicing (completed during the paper feed and carriage stepper motor drive end Delay subroutine) tests for various ASCII character codes, loads characters into the Character Buffer (CB), and repeats until one of several conditions sets the CB Full status flag. Once the CB Full flag is set, further character transmission by the host system is inhibited and printing can begin.

The carriage stepper motor is initialized, and drive begins for the direction indicated. The motor is accelerated to constant speed, printable character codes are translated to dot patterns and printed (if printing is enabled), and the motor is decelerated to a stop. Two timing loops guarantee both constant speed and protection (Failsafe Time) against stepper motor burn out due to high current overload. The two optical sensors, described in the Printer Mechanism section above, are constantly monitored to maintain constant speed, and trigger print head solenoid firing.

Once the line is printed and the carriage stepper motor drive routine has been completed, a Linefeed is forced. The paper feed stepper motor drive subroutine tests the number of lines to move, and energizes the paper feed stepper motor for the required distance. The lines per page default is 66; if 66 lines have been received, a Formfeed to Top-of-Next-Page is performed. The Top-Of-Page is set at Power On/Reset.

When the EOF code is received, the EOF status flag is set. When the last line has been printed, the EOF check will force the print head assembly to the HOME position. The EOF flag is tested following each paperfeed stepper motor drive. The next entry to the External Status Check subroutine begins a loop which waits for input from either the external status switches or the host system.

The software character dot matrix used in this application is 5 x 7 of the available 9 x 9 print head solenoid matrix. Although lower case descenders and block/line graphics characters are not implemented, Appendix B, Printer Enhancements, discusses how and where these enhancements could be added. The software implements the full 95 ASCII printable characters set.

Memory and Register Allocation

DATA MEMORY ALLOCATION (RAM)

The UPI-42AH has 256 bytes of Data Memory. Sixteen bytes are used by the two 8 byte register banks (RB0 and RB1). Sixteen additional bytes are used for the Program Stack. The Stored Time Constants utilize 11 bytes, while the stepper motor phase storage requires 4 bytes. Below is a detailed description of Data and Program Memory.

Hex Address	Description
2F-7FH	80 Character Line Buffer (80 Bytes)
25-2EH	Stored Time Constants Buffer (11 Bytes)
24H	Unused
23H	Character Print Test ASCII Code Start Temporary Storage
22H	Pseudo Register: Paperfeed Stepper Motor Last Phase Indirect Address
21H	Pseudo Register: Carriage Stepper Motor Forward/Reverse Last Phase
20H	Pseudo Register: Last Phase of Stepper Motor Not Being Driven
18-1FH	Register Bank 1: Character Processing
8-17H	8 Level Stack
0-07H	Register Bank 0: Stepper Motor Forward/Reverse Acceleration/Drive

Figure 19. Data Memory Allocation Map

Register Bank 0 is used for stepper motor drive functions. Register Bank 1 is used for character processing. Each register bank's register assignments is listed in Figures 20 and 22, respectively. Each register bank has one register allocated as a Status Register. Figures 21 and 23 detail the Status Register flag assignments. Note

that bit 7 of each Status Byte is used as a print head assembly motion direction flag. This saves coding of the Select Register Bank (SEL RBn) instruction at each point the flag is checked.

Register Bank 0		
Register	Program Label	Description
R0	TmpR00	RB0 Temporary Register
R1	TStrR0	Store Time Register
R2	GStR20	General Status Register
R3	PhzR30	Stepper Motor Step Register
R4	CntR40	Count Register
R5	TConR0	Time Constant Register
R6	LnCtR0	Line Count Register
R7	OpnR70	Available, Scratch

Figure 20. Register Bank 0 Register Assignment

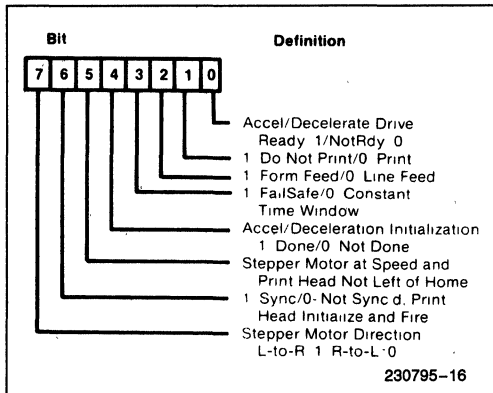


Figure 21. Register Bank 0 Status Byte Flag Assignments

Register	Program Label	Description
R0	TmpR10	RB0 Temporary Register
R1	CAdrR1	Character Data Memory Address Register
R2	ChStr1	Character Processing Status Byte Register
R3	CDtCR1	Character Dot Count Register
R4	CDotR1	Character Dot Temporary Storage Register
R5	CCntR1	Character Count Temporary Register
R6	StrCR1	Store Character Register
R7	OpnR71	Available/Scratch

Figure 22. Register Bank 1 Register Assignment

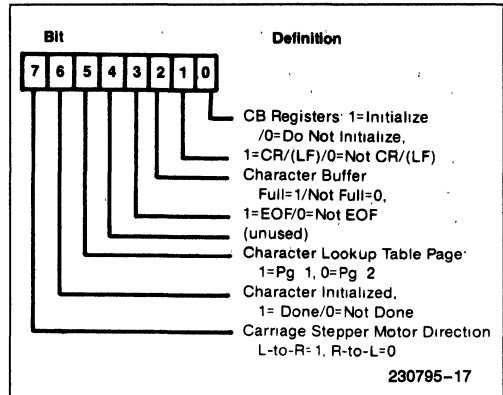


Figure 23. Register Bank 1 Status Byte Flag Assignments

PROGRAM MEMORY ALLOCATION (EPROM/ROM)

The UPI-42AH has 2048 bytes of Program Memory divided into eight pages, each 256 bytes. Figure 24 illustrates the Program Memory allocation map by page.

Page	Hex Address	Description
Page 7	1792-2047	Character to Dot Pattern Lookup Table; Page 2: ASCII 50H-7EH
Page 6	1536-1791	Character to Dot Pattern Lookup Table; Page 1: ASCII 20H-4FH (sp-M)
Page 5	1280-1535	Miscellaneous Subroutines: InitAI/AllOff Clear Data Memory Home Print Head Assembly Character Print Test Initialize Carriage Stepper Motor Delay Stepper Motor Deselect
Page 4	1024-1279	Paper Feed Stepper Motor Drive
Page 3	768-1023	Stepper Motor Step Look Up Table (Indexed) Character Processing and Translation Print Head Firing
Page 2	51-767	Carriage Stepper Motor Acceleration Time Calculation and Storage Stepper Motor Deceleration
Page 1	256-511	Carriage Stepper Motor Drive
Page 0	0-255	Initialization-Jump-on-Reset Main Program Body External Status Switch Check Character Buffer Fill

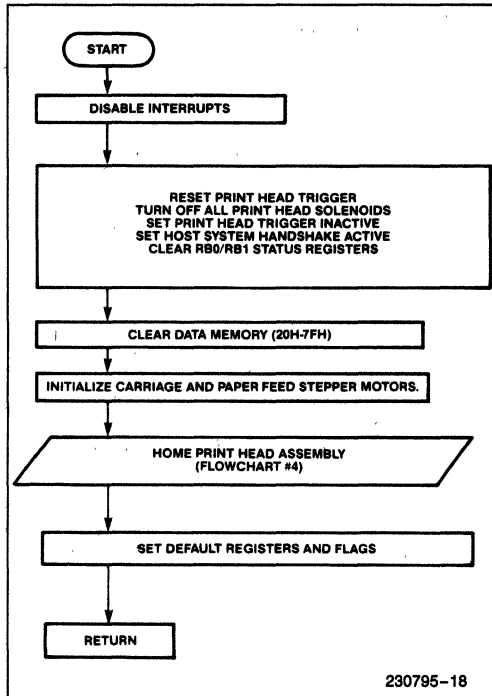
Figure 24. Program Memory Allocation Map

Software Functional Blocks

Below is a description and flow chart for each of the ten software blocks listed.

1. Power-On/Reset Initialization

The first operational part in Flow Chart No. 1 is the Power-On or Reset Initialization. Flowchart No. 2 illustrates the Initialization sequence in detail.



Flow Chart No. 2. Power-On/Reset Initialization

Initialization first disables both interrupts. This is done as a precaution to prevent the system software from hanging-up should an interrupt occur before the proper registers and Data Memory values are initialized.

Initialization then deactivates the system electronics. This is also a precaution to protect the printer mechanism and includes the print head solenoid (trigger and data) lines and the stepper motor select lines. The host system handshake signals are activated to inhibit data transfer from the host until the printer is ready to accept data.

Next, Data Memory is cleared from 20H to 7FH. This includes the 80 byte Character Buffer, the 11 byte Stored Time Constants buffer, and the 4 bytes used as pseudo registers. The pseudo registers are Data Memory locations used as if they were registers. They serve as

storage locations for step data used in accurately reversing the direction of the carriage stepper motor, and stabilizing either of the stepper motors not being driven.

The Data Memory locations 00H through 1FH are not cleared. These locations are Register Bank 0 (00H-07H), Program Stack (08H-17H), and Register Bank 1 (18H-1FH) (see Figure 19). Clearing the Program Registers or Stack would cause the initialization subroutine to become lost. The registers are used from the beginning of the program. Care is taken to initialize the registers and stack accurately prior to each program subroutine as required.

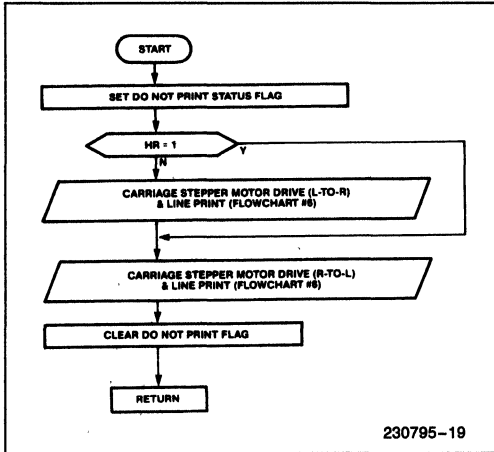
Upon power-on, it is necessary to initialize the two stepper motors, verify their operation, and locate the print head assembly in the left-most 'HOME' position. This sequence serves as a system checkout. If a failure occurs, the motors are deselected and the external status light is turned on. Each stepper motor is selected and energized for a sequence of four steps. This serves to align and stabilize each stepper motor's rotor position, preventing the rotor from skipping or binding when the first drive sequence begins.

At the end of each stepper motor's initialization, the last step data address is stored in one of the Data Memory pseudo registers. The last step data address is recalled at the beginning of the next corresponding stepper motor drive sequence, and used as the basis of the next step sequence. This ensures that the stepper motor always receives the exact next step data, in sequence, to guarantee smooth stepper motor motion. This also guarantees the motor never skips or jerks, which would misalign the start, stop, and character dot column positions. A stepper motor not being driven has its last phase data output held constant to stabilize it.

Following any stepper motor drive sequence of either motor, a delay of 30-60 ms occurs by switching the current to Hold Current, stabilizing the motor before it is deselected.

2. Home Print Head Assembly

At the end of the carriage stepper motor four step initialization, the output of the HR optical sensor is tested. The level of the HR signal indicates which drive sequence will be required to 'HOME' the print head assembly. If the print head assembly is to the right of HR, HR is high, the print head assembly need only be moved to from Right-to-Left until HR is low, then decelerated to locate the physical home position. If HR is low, the print head assembly must be moved first Left-to-Right until HR is high, then Right-to-Left to locate both the logical and physical 'HOME' positions. In each case, the software accelerates the carriage stepper motor, generating the Stored Time Constants then decelerates the stepper motor using the Stored Time Con-



Flow Chart No. 3. HOME Print Head Assembly

starts (see Background section above). Flow Chart No. 3 details the HOME print head assembly subroutine. Figures 13 and 18 illustrate the components of acceleration and print head assembly line motion.

The carriage stepper motor drive subroutines used to HOME the print head assembly and to print, are the same. A status flag, called Do-Not-Print, determines whether the Character Processing subroutine is called.

The flag is set by the subroutine which calls the Carriage Stepper Motor Drive subroutine. Details of the carriage and paper feed stepper motor drive and character processing subroutines are covered separately below.

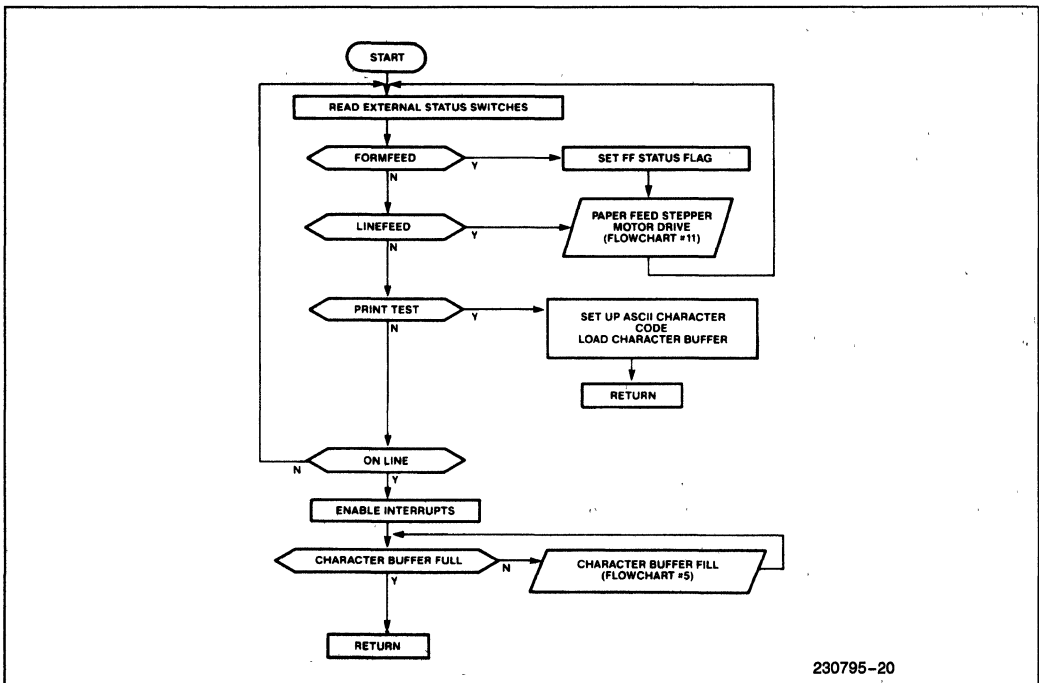
3. External Status Switch Check

Once the system is initialized and the print head is at the HOME position, the software enters a loop which continually monitors the four external status switches, and exits if any one is active. Flow Chart No. 4 details the External Status Switch Check subroutine.

Flow Chart No. 4. External Status Switch Check

If the LINEFEED or FORMFEED switch is set, the Paper Feed subroutine is called. The Paper Feed subroutine is discussed in detail below. If the ONLINE switch is set, the Character Buffer (CB) Fill subroutine is called.

If the Character Print TEST switch is set, the Data Memory Character Buffer (CB) is automatically loaded with the ASCII code sequence, beginning at 20H (a Space character), the first ASCII printable character code. The software then proceeds as if the CB had been filled by characters received from the host system. The



Flow Chart No. 4. External Status Switch Check

External Status Switch Check subroutine is exited and character printing begins. When the line has finished printing, a linefeed occurs (as shown in the main program Flow Chart No. 1) and the program returns to the External Status Switch Check subroutine. If the TEST switch remains active, the ASCII character code is incremented and program continues as before. This will eventually print all 95 ASCII printable characters. An example of the TEST printer output, the complete ASCII character code printed, is shown in Figure 25.

4. Character Buffer Fill

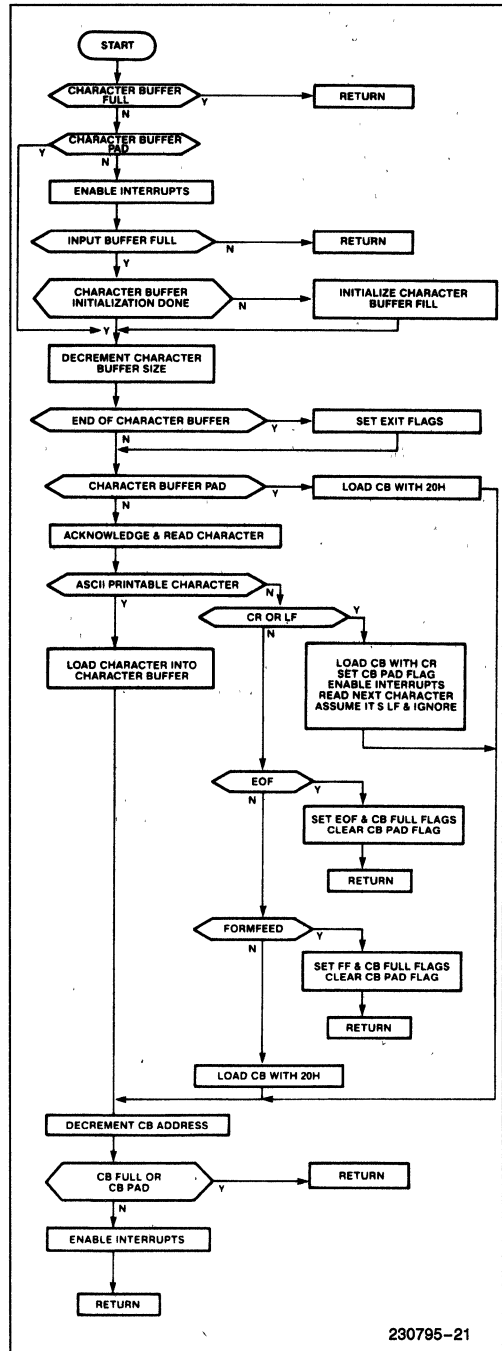
The Character Buffer (CB) Fill subroutine is called from three points within the main program; External Status Switch subroutine, and the Delay subroutine following the carriage and paper feed stepper motor drive subroutines. Flowchart No. 5 details the Character Buffer Fill subroutine operation.

The approximate 80 ms total pre-deselect delay at the end of each stepper motor drive sequence, 40 ms carriage and 40 ms paper feed stepper motor pre-deselect delay, is sufficient to load an entire 80 character line. Half the CB is filled at the end of printing the current line, and the second half is filled at the end of a paper feed. There is no time lost in printing throughput due to filling the character buffer.

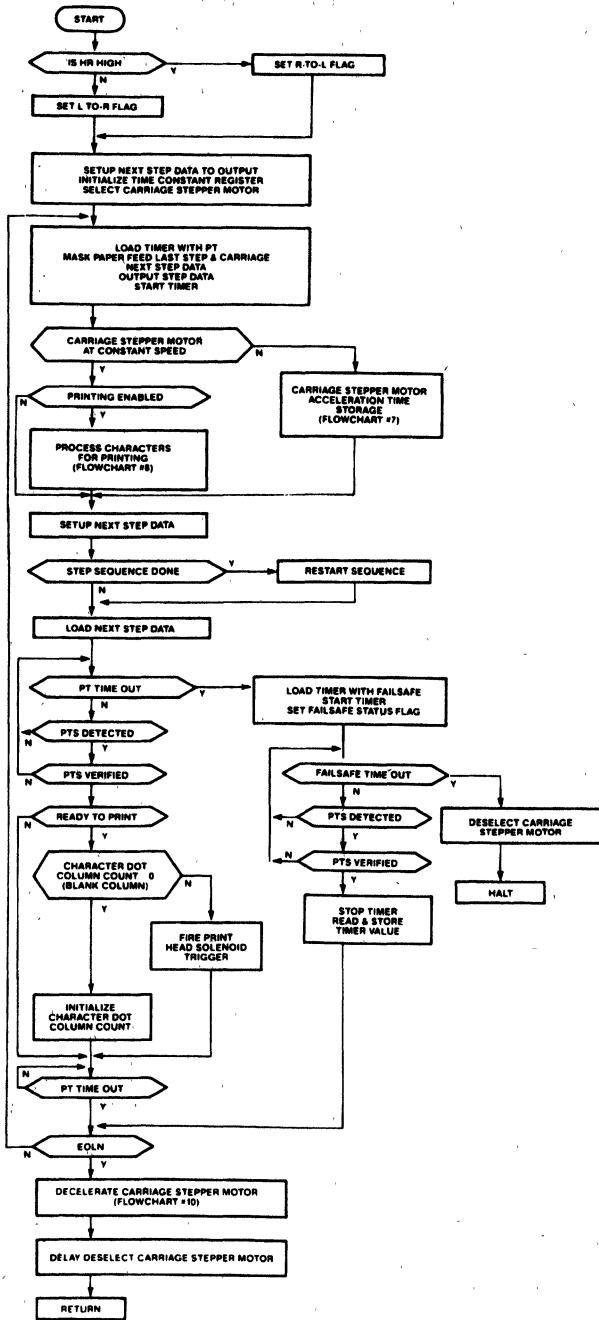
Character input is interrupt driven. When the IBF interrupt is enabled, a transmitted character sets the IBF interrupt and IBF Program Status flag. Three instructions make up the IBF interrupt service routine. This short routine disables further interrupts, sets the BUSY handshake line active, inhibiting further transmission by the host, and returns. The subroutine can be executed at virtually any point in the software flow without affecting the printer mechanism operation. Processing of the received character takes place during one of the three program segments mentioned above. The BUSY line remains active until the character is processed by the CB Fill subroutine.

The CB is 80 bytes from the top of Data Memory (30H-7FH). It is a FIFO for forward, left-to-right printing, and a LIFO for reverse, right-to-left, printing. Loading the CB always begins at the top, 7FH. One character may be loaded into the buffer each time the CB Fill subroutine is called.

The CB is always filled with 80 bytes of data prior to printing. If the total number of characters input up to a Carriage Return (CR)/Linefeed (LF), does not completely fill the CB, the CR code is loaded into the CB and the balance of the CB is padded with 20H (Space Character) until the CB is full. A Linefeed (LF) character following a Carriage Return is ignored. A LF is always forced at the end of a printed line. When the CB is full, the CB Full status byte flag is set and printing can begin.



Flow Chart No. 5. Character Buffer Fill



230795-23

Flow Chart No. 6. Carriage Stepper Motor Drive/Line Printing

Next, the carriage and paper feed stepper motor step data is initialized. The last step data output to the paper feed stepper motor is loaded into the Last Phase pseudo register. This data is masked with each step data output to the carriage stepper motor. Masking the step data in this manner guarantees the paper feed motor signals do not change as the carriage stepper motor is being driven.

Figure 26 illustrates the carriage stepper motor step sequence versus the actual step data output for clockwise rotation, Left-to-Right motion, and counterclockwise rotation, Right-to-Left print head assembly motion. An eight step sequence is depicted in the figure. Note that the sequence for Right-to-Left motion is the reverse of the sequence for Left-to-Right motion. Note also, that for the L-to-R sequence step 4 is the same as step 0, step 5 the same as step 1, etc., through step 7 matching step 3. The four step sequence simply repeats itself until the motor is stopped via the Deceleration subroutine.

L-to-R Motion Sequence	Phase/Step Data (3 2 1 0)	R-to-L Motion Sequence	BCD (3 2 1 0)
0	1 0 0 1	7	0 0 0 0
1	1 0 1 0	6	0 0 0 1
2	0 1 1 0	5	0 0 1 0
3	0 1 0 1	4	0 0 1 1
4	1 0 0 1	3	0 1 0 0
5	1 0 1 0	2	0 1 0 1
6	0 1 1 0	1	0 1 1 0
7	0 1 0 1	0	0 1 1 1

**Figure 26. Carriage Stepper Motor
Phase/Step Data**

When the carriage stepper motor is driven for a specific direction of print head assembly motion, the step sequence must be consistent for the motion to be smooth and accurate. The same holds true for the transition from one direction of motion to the other. Since the sequence for one direction is the opposite for the other direction, incrementing the sequence for L-to-R and decrementing for R-to-L provides the needed step data flow. For example, referring to Figure 26, if the print head assembly moved L-to-R and the last step output was #1, the first step for R-to-L motion would be #7. Thus, when the carriage stepper motor is initialized for a clockwise (L-to-R) or counterclockwise (R-to-L) rotation, the last step sequence number is incremented or decremented to obtain the proper next step. In this way, the smooth motion of the stepper motors is assured.

The step data is referenced indirectly via the step sequence number. The step data is stored in a Program Memory look-up table whose addresses correspond to the step sequence numbers. For example, as shown in

Figure 26, at location 0 the step data "1001" is stored. This method is particularly well suited to the UPI-42AH software. The UPI-42AH features a number of instructions which perform an indirect move or data handling operation. One of these instructions, MOVP3A,@A, unlike the others, allows data to be moved from Page 3 of Program Memory to any other page of Program Memory. This instruction allows the step data to be centrally located on Page 3 of Program Memory and accessed by various subroutines.

Each time the carriage stepper motor step data is output, the step data lookup table address is incremented or decremented, depending upon the direction of rotation, and tested for restart of the sequence. The address is tested because the actual step data, Figure 26, is not a linear sequence and thus is not an easily testable condition for restarting the sequence. The sequence number is tested for rollover of the sequence count from 03H to 04H and clockwise motor rotation via the Jump on Accumulator Bit instruction (JbN), with 00H loaded to restart the sequence. The same bit is tested when decrementing the sequence count for counterclockwise motor rotation, R-to-L motion, because the count rolls over from 00H to 0FFH, with 03H loaded to restart the sequence.

At this point the UPI-42AH Timer/Counter is loaded, the step signal is output, and the timer started. The next step data to be output has been determined and the At-Speed flag is tested for entry to one of two subroutines; Stepper Motor Acceleration Time Storage or Character Processing.

The first entry to the Acceleration Time Storage subroutine initializes the subroutine and returns. All other entries to one of the two subroutines perform the necessary operations, detailed below (Blocks 6 and 7), and returns. The program loops until the PT times out or the PTS level change is detected. PTS is tied to T0 of the UPI-42AH. The level present on T0 is directly tested via conditional jump instructions. The software loops on polling the timer Time Out Program Status flag and the T0 input level.

As described in the Background section above (shown in Figure 13), if PT times out before PTS is detected, the software waits for PTS before outputting the next step signal. If PT times out before PTS, a second timer count value is loaded into the UPI-42AH timer. The timer value is called "Failsafe." This is the maximum time the stepper motor can be selected, with no rotor motion, and not damage the motor. If PTS is not detected, either the carriage stepper motor is not rotating or the optical sensor is defective. In either case, program execution halts, the motor is deselected, and the external status light is turned on to indicate a malfunction. A system reset is required to recover from this condition. The Failsafe time is approximately 20 milliseconds, including PT.

The Failsafe time loop also serves as a means of tracking the elapsed time between PT time out and PTS. Entry to the Failsafe time loop sets the Failsafe/Constant Time Window status flag. This flag is tested by the Acceleration Time Storage subroutine for branching to the proper time storage calculation to be performed (see Figure 13 and Block 6 below for further description).

During the Failsafe timer loop, if PTS is detected and verified as true, the Failsafe timer value is read and stored in the Time Storage register. This value is used during the next Acceleration Time Storage subroutine call to calculate the Stored Time Constant (see Block 6 below). If PTS is invalid, the flow returns to the timer loop just exited, again waiting for PTS or Failsafe time out.

During the PT time loop, if PTS is detected and verified, the Sync flag is tested for entry to the print head solenoid firing subroutine. This flag is set by the first entry to the Character Processing subroutine. The flag synchronizes the solenoid firing with character processing. Only if characters are processed for printing will the solenoids be enabled, via the Sync flag, for firing. This prevents the solenoids from being fired without valid character dot data present.

As described in the Background section "Relationship Between PTS and PT," PTS is the point of peak angular velocity within a step of the motor. After PTS is detected the motor speed ramps down, compensating for the overshoot of the rotor motion. PTS is the optimum time for print head solenoid firing, as shown in Figure 13. This is the stable point of motor rotation and, thus, the print head assembly motion. If PTS is detected during PT, printing is enabled, the Sync flag is set, and the solenoid trigger is fired.

The firing of the solenoid trigger, following PTS, is very time critical. The time between PTS and solenoid firing must be consistent for accurate dot column alignment throughout the printed line. The software is designed to meet this requirement by placing all character processing and motor control overhead before the solenoid firing subroutine is called. The actual instruction sequence which fires the print head solenoid trigger is plus or minus one instruction for any call to the subroutine.

Once the timer loop is complete, the software tests for Exit conditions. If the Exit conditions fail, the software loops to output the next step signal, starts the PT timer, and continues to accelerate the carriage stepper motor, or process, and print characters. If the Exit test is true, the carriage stepper motor is decelerated to a fixed position, and the program returns to the main program flow (see Flowchart 1).

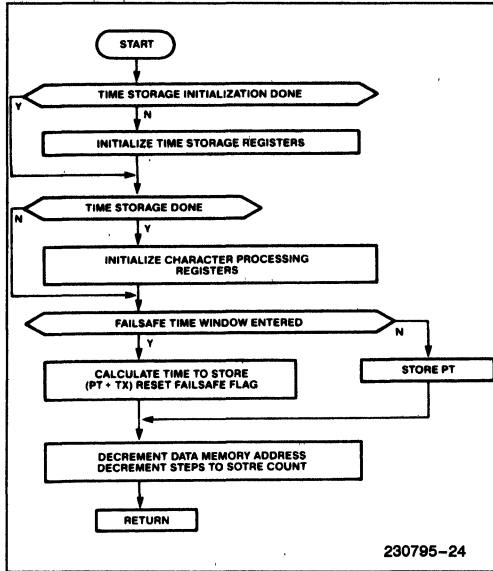
The exit conditions are different for the two directions of print head assembly motion. For L-to-R printing, if a Carriage Return (CR) character code is read from CB, the carriage stepper motor drive terminates and the motor is decelerated to a fixed position. There are two conditions for terminating carriage stepper motor drive upon detecting a CR during L-to-R motion. If less than half a character line (40 characters) has been printed, the print head assembly returns to the HOME position to start the next printed line. Otherwise, the print head assembly continues to the right-most position for a full 80 character line, and then begins printing the next line from R-to-L. R-to-L printing always returns the print head assembly to the HOME position before the next line is printed L-to-R. When HR is high, character printing always stops and the carriage stepper motor drive subroutine exits to the deceleration subroutine.

6. Accelerate Stepper Motor Time Storage

As described above, when the carriage stepper motor is accelerated the step time required to guarantee the motor is at a constant rate of speed translates to a specific distance traveled by the print head assembly (see Figure 18). In order to position the print head assembly accurately for bi-directional printing, the distance traveled during deceleration must be the same as during acceleration. The Carriage Motor Acceleration Time Storage subroutine calculates the step times needed to accelerate the carriage stepper motor, and stores them in Data Memory for use as PT during deceleration.

The first call of the Carriage Stepper Motor Acceleration Time Storage subroutine initializes the required registers and status flags. The time calculation begins with the second carriage stepper motor step signal output. The program returns to the carriage stepper motor drive subroutine and loops on PT. Each subsequent call of the Acceleration Time Storage subroutine tests the Failsafe/Constant flag and branches accordingly (see Flow Chart 7). The Acceleration Time Storage subroutine has two parts which correspond to PTS leading or PTS lagging PT.

If the Failsafe/Constant flag is set, PTS lagged PT. The time from PT time out to PTS, Tx (see Figure 13), must be added to the PT and stored in Data Memory. As described above, if PT lagged PT, the Failsafe time is loaded and PTS is again polled during the time loop. When PTS occurs within the Failsafe time, the timer is stopped and the timer value stored. The UPI-42AH timer is an up timer, which means that the value stored is the time remaining of the Failsafe time when PTS occurred. The elapsed time must be calculated by subtracting the time remaining (the value stored) from the Failsafe time constant. This is done in software by using two's complement arithmetic. If the Failsafe flag is not set PTS led PT, and PT is the Stored Time Constant stored.



Flow Chart No. 7. Carriage Stepper Motor Acceleration Time Storage

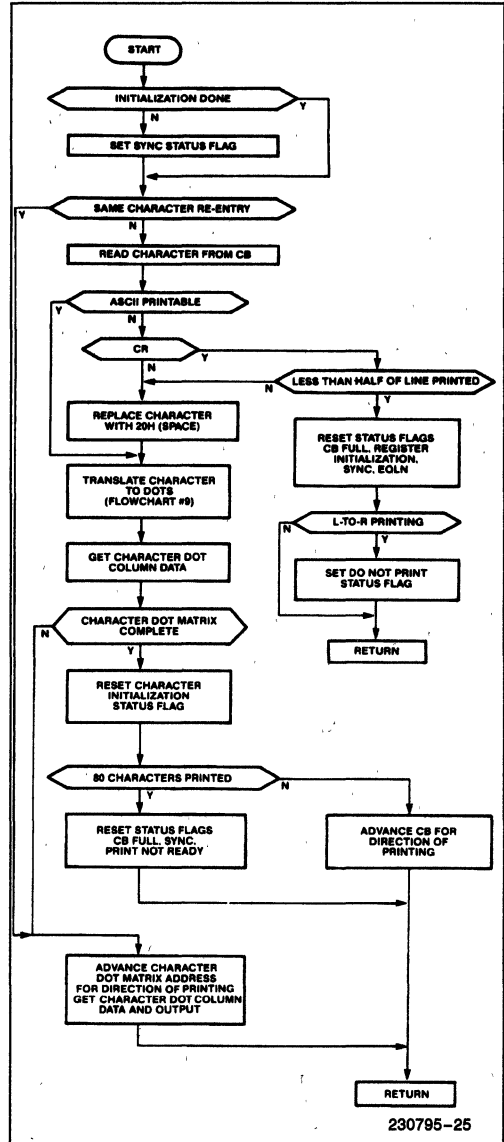
Indirect addressing of Data Memory is used to reference the Stored Time Constant Data Memory location. The Data Memory location address is decremented each time the Acceleration Time Storage subroutine is exited and a Stored Time Constant has been generated.

The last Acceleration Time Storage subroutine exit sets the At-Speed status flag and initializes the character processing registers and flags.

7. Process Characters for Printing

The Character Processing subroutine is entered only if the Home Reset (HR) optical sensor signal is high and printing is enabled. Otherwise, the software simply returns to the Carriage Stepper Motor Drive subroutine. There are two cases when printing is not enabled; during the HOME subroutine operation, and when the print head assembly returns to the HOME position after printing less than half an 80 character line. If printing is enabled, the Sync status flag is set.

All character processing operations use the second UPI-42AH Data Memory Register Bank, RB1. Register Bank 1 is independent of Data Memory Register Bank 0, used for stepper motor control. The use of two independent register banks greatly simplifies the software flow, and helps to ensure the accuracy of event sequences that must be handled in parallel. Each register bank must be initialized only once for any entry to either the Carriage Stepper Motor Drive or Character Processing subroutines. A single UPI-42AH Assembly Language instruction selects the appropriate register



Flow Chart No. 8. Process Characters for Printing

bank. Initializing the character processing registers includes loading the maximum character count (80), dot matrix size count (6), and CB start address. The CB start address is print direction dependent, as described in Block 4, above.

Character processing reads a character from the CB, tests for control codes, translates the character to dots,

and conditionally exits, returning to the Carriage Stepper Motor Drive subroutine. Flow Chart 8 details the character processing subroutine.

Each character requires six steps of the carriage stepper motor to print; five for the 5 character dot columns and 1 for the blank dot column between each character. Reading a character from the CB and character-to-dot pattern translation takes place during the last character dot column, or blank column, time.

The first character line entry to the Character Processing subroutine appears to the software as if a last character dot column (blank column) had been entered. The next character, in this case the first character in the line, is translated and printing can begin. This method of initializing the Character Processing subroutine utilizes the same software for both start-up and normal character flow. Once a character code has been translated to a dot matrix pattern starting address in the look-up table, all subsequent entries to the Character Processing subroutine simply advance the dot column data address and outputs the data.

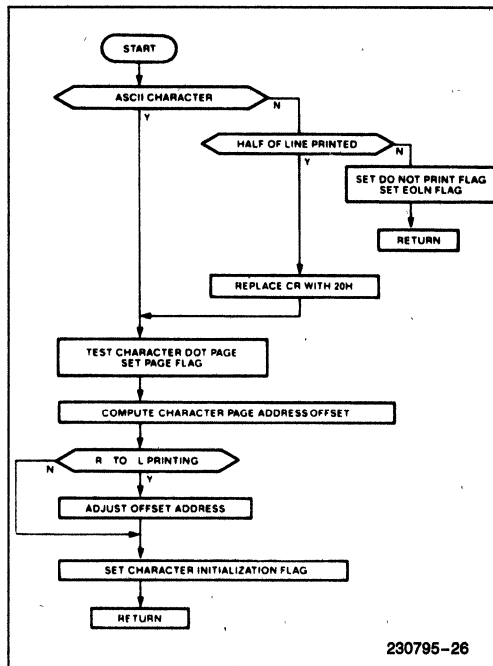
The decision to translate the character to dots during the blank column time was an arbitrary one. As was the choice of the blank column following rather than preceding the actual character dot matrix printing.

8. Translate Character-to-Dots

Characters-to-dot pattern translation involves converting the ASCII code into a look-up table address, where the first of the five bytes of character dot column data is stored. The address is then incremented for the next column of dot pattern data until the full character has been printed.

The dot pattern look-up table occupies two pages, or approximately 512 bytes of Program Memory. A printable ASCII character is tested for its dot pattern location page and the offset address, from zero, on that page. Both the page test and page offset calculations use two's complement arithmetic, with a jump on carry or not carry causing the appropriate branching. Once the pattern page and address are determined the indirect addressing and data move instructions are used to read and output the data to the print head solenoids. Flow-chart 9 details the Character-to-Dots Translation subroutine.

In the case of R-to-L printing, although the translation operation is the same, the character is printed in reverse. This requires that the character dot pattern address be incremented by five, before printing begins, so that the first dot column data output is the last dot column data of the character. The dot pattern look-up table address is then decremented rather than incremented; as in L-to-R printing, for the balance of the character. Translation still takes place during the last



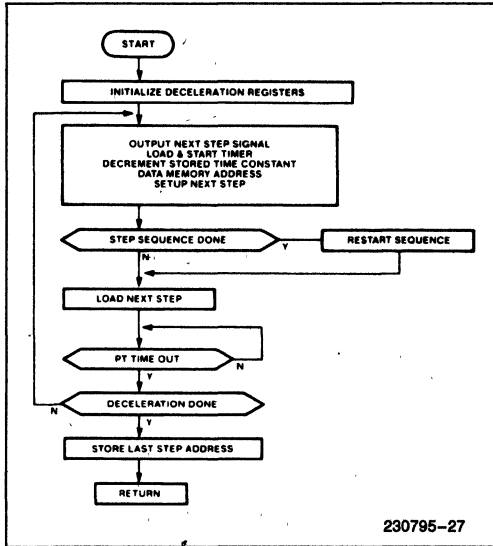
Flow Chart No. 9. Translate Character-to-Dots

character dot column, the blank column, and the blank column follows the character matrix.

Only one control code, a Carriage Return (CR), is encountered by the character translation subroutine. Linefeed (LF) characters are stripped off by the CB Fill subroutine. If a CR code is detected the software tests for a mid-line exit condition; less than half the line printed exits the stepper motor drive subroutine and HOMEs the print head assembly before printing the next line. If the test fails, more than half the line has been printed, the CR is replaced by a 20H (Space character) and printing continues for the balance of the line; the space characters padding the CB are printed.

As mentioned above, the character dots are printed and the print head trigger is fired when the PTS signal is detected and verified and the carriage stepper motor is At Speed.

When the character to print test fails the CB Buffer size count equals zero, the Carriage Stepper Motor Drive subroutine exit flags are set, and the flow passes to the Deceleration and Delay subroutines and programs returns to the main program flow.



Flow Chart No. 10. Decelerate Carriage Stepper Motor

9. Decelerate Carriage Stepper Motor

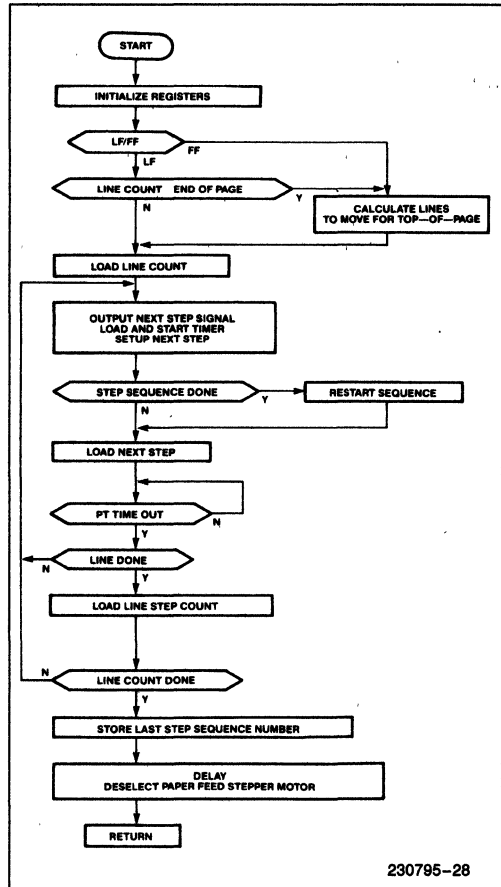
The transition from the Carriage Stepper Motor Drive subroutine to the Deceleration subroutine outputs the next step signal in sequence, and then initializes the Deceleration subroutine registers; Stored Time Constants Data Memory buffer end address and size. The Stored Time Constant Buffer is a LIFO for deceleration of the carriage stepper motor. The buffer size is used as the step count. When the step count decrements to zero, the step signal output is terminated, and the last step sequence number is stored in the carriage stepper motor Next Step pseudo register. The last step sequence number is recalled, during initialization of the next carriage stepper motor drive, as the basis of the next step data signal to be output. See Flow Chart 10.

When the carriage stepper motor is decelerated, Failsafe protection and PTS monitoring are not necessary. The Deceleration subroutine acts as its own failsafe mechanism. Should the stepper motor hang-up, the subroutine would exit and deselect the motor in sufficient time to protect the motor from burnout. Since neither Failsafe nor print head solenoid firing take place during deceleration, PTS is not needed. PT is replaced by the Stored Time Constant values in Data Memory. The Deceleration subroutine determines the next step signal to output, loads the Timer with the Stored Time Constant, starts the UPI-42AH Timer, and loops until time out. The subroutine loops to output the next step until all of the Stored Time Constants have been used. The program returns to the Carriage Stepper Motor Drive subroutine and the motor is deselected following the Delay subroutine execution. The

Delay subroutine is called to stabilize the stepper motor before it is deselected. During the DELAY subroutine, the IBF interrupt is enabled and characters are processed. A paperfeed is forced following the carriage stepper motor being deselected.

10. Paper Feed Stepper Motor Drive

The paper feed stepper motor subroutine outputs a predefined number of step signals to advance the paper, in one line increments, for the required number of lines. The number of step signals per line increment is a function of the defined number of lines per inch, given the distance the paper moves in one step. Figure 16 lists three step (or pulse) count and line spacing configurations, as well as the distance the paper moves in one step. Standard 6 lines per inch spacing has been implemented in this Application Note (Appendix B details how variable line spacing could be implemented). Flow Chart 11 illustrates the Paper Feed subroutine.



Flow Chart No. 11. Paper Feed Stepper Motor Drive

The number of lines the paper is to be moved is called the "Line Count." The Line Count defaults to one unless the Formfeed flag is set, or the total number of lines previously moved equals a full page. The default total lines per page for this application is 66. When the total number of lines moved equals 66, the paper is moved to the top of the next page. The Top-of-Page is set at power-on or reset.

If the Formfeed flag has been set in the Character Buffer Fill subroutine, the software calculates the number of lines needed for a top of next page paper feed. The resulting line count is loaded in the Line Count Register. The Paper Feed subroutine loops on the line count until done and then returns to the main program body.

Once the Paper Feed subroutine is complete, the software loops to test the End of File (EOF) Flag (see Flow Chart 1). If EOF is set, the print head assembly is moved to the HOME position, the program again enters the External Status Switch Test subroutine, and begins polling the external status switches. If EOF is not set, the program directly calls the External Status Switch Check subroutine, and the program repeats for the next line.

CONCLUSION

Although the full speed, 12 MHz, of the UPI-42AH was used, the actual speed required is approximately 8-9 MHz. 1400 bytes of the available 2k bytes of Program

Memory were used; 500 bytes for the 95 character ASCII code dot pattern look-up table, 900 bytes for operational software. This means that the UPI-42AH has excess processing power and memory space for implementing the additional functions such as those listed below and discussed in Appendix B.

- Special Characters or Symbols
- Lower Case Descenders
- Inline Control Codes
- Different Character Formats
- Variable Line Spacing

The software developed for this Application Note was not fully optimized and could be further packed by combining functions. This would require creating another status register, which could also serve to implement some of the features listed above. Since the full 16 byte stack is not used for subroutine nesting, there are 6-8 bytes of Program Stack Data Memory that could be used for this purpose. In several places, extra code was added for clarity of the Application Note. For example, each status byte flag is set with a separate instruction, using an equate label, rather than setting several flags simultaneously at the same point in the code.

This Application Note has demonstrated that the UPI-42AH is easily capable of independently controlling a complex peripheral device requiring real time event monitoring. The moderate size of the program required to implement this application attests to the effectiveness of the UPI-42AH for peripheral control.

APPENDIX A SOFTWARE LISTING

```

1 $MOD42 TITLE('UPI 42 APP NOTE');
2 $MACROFILE NOSYMBOLS NOGEN DEBUG
3
4 $INCLUDE('F1: ANECD. OV1)
= 5 ; PG
= 6
= 7 ; *****
= 8 ;           Complex Peripheral Control With the UPI-42
= 9
= 10 ;          Intel Corporation
= 11 ;          3065 Bowers Avenue
= 12 ;          Santa Clara, Ca 95051
= 13
= 14
= 15 ;          Written By      Christopher Scott
= 16
= 17 ; *****
= 18 ;
= 19 ;
= 20 ;          Notes and Comments
= 21
= 22 ;          Three Assembly Language files comprise the full Application
= 23 ;          Note source code:
= 24 ;
= 25 ;          1. ANECD. OV1   App Note Equates, Constants, Declarations . Overlay
= 26 ;
= 27 ;          2. 42ANC. SRC   UPI-42 App Note Code Source
= 28 ;
= 29 ;          3. CHRTBL. OV1  Character Table . Overlay (Character Lookup Tables)
= 30 ;
= 31 ;
= 32 ;
= 33 ;
= 34 ; PG
= 35 ; *****
= 36 ;          Equates, Constants and System Definitions
= 37 ; *****
= 38 ;
= 39 ;          Data & Program Memory Allocations
= 40 ;
= 41 ;          Program Memory
= 42 ;
= 43 ;          Page No.  Hex Addr  Description
= 44 ;          -----
= 45 ;
= 46 ;          Page 7  1792-2047  Char to Dot pattern lookup table
= 47 ;          Page 2: ASCII 50H-7FH (N-*)
= 48 ;          Page 6  1536-1791  Char to Dot pattern lookup table
= 49 ;          Page 1: ASCII 20H-4FH (sp-M)
= 50 ;          Page 5  1280-1535  Misc called routines:
= 51 ;          InitAI/AllOff
= 52 ;          Clear Data Memory
= 53 ;          CR Home
= 54 ;          Char Print Test - load Ascii char codes
= 55 ;          Initialize CR Stpr Mtr
= 56 ;          Delay: short/long/very long
= 57 ;          Stpr Mtr deselect
= 58 ;          Page 4  1024-1279  PaperFeed Stpr Mtr Init and Drive
= 59 ;          Page 3  768-1023  Stpr Mtr Phase LookUp Table - Indexed
= 60 ;          Character Translation and processing
= 61 ;          PrintHead firing
= 62 ;          Page 2  512-767   Stpr Mtr Accel. Time calc. and memorization
= 63 ;          Stpr Mtr Deceleration
= 64 ;          Page 1  256-511   SMDriv (FAccel/RAccel) - Forward & Reverse
= 65 ;          Stpr Mtr acceleration & drive
= 66 ;          Page 0  0-255    Initialization Jmp-on-Reset
= 67 ;          Program Body - all calls
= 68 ;          Character Input test and Char Buffer fill loop
= 69 ;          Interrupt service routines
= 70 ;

```

```

= 71 ; PG
= 72 ;
= 73 ;      Data Memory
-----
= 74 ;      Dec.      Hex      Description
= 75 ;
= 76 ;
= 77 ;      TOP
= 78 ;
= 79 ;      48-127     2F-7FH     80 Character Line Buffer
= 80 ;      37-47     25-2EH     Stpr Mtr Accel/Decel time, memorization
= 81 ;      36        24H        Unused
= 82 ;      35        23H        Char Print test ASCII code start tmp store
= 83 ;      34        22H        LF SM last Phz Indereect Addr psuedo reg
= 84 ;      33        21H        CR SM Forward/Reverse last Phz psuedo reg
= 85 ;      32        20H        Psuedo Reg: Last Phase of stpr mtr not
= 86 ;                          being driven
= 87 ;      24-31     18-1FH     Register Bank 1: Character Handling
= 88 ;      8-23      8-17H     8 Level Stack
= 89 ;      0-7       0-07H     Register Bank 0: Stpr Mtr F/R Accel/Drive
= 90 ;
= 91 ;      BOTTOM
= 92 ;
= 93 ;      Data Memory Equates
-----
= 94 ;
= 95 ;
0050 = 96 CHBFSZ EQU 50H ;char buffer size 0-79 = 80
00D9 = 97 H1fCpl Equ OD9H ;Cpl(1/2 CbBfSz) => cpl of 27H = OD9H
= 98 ;
007F = 99 FCBfSt Equ 7fH ;start of char buffer
0080 = 100 FCBfIS Equ 80H ;init CB strt-allows xtra Dec by 1
002F = 101 RCBfIS Equ 2FH ;init CB strt-allows xtra Inc by 1
0051 = 102 CbBfIS Equ 81 ;load char cnt reg w/char bufr Init Size
= 103 ;
002F = 104 ENDBUF EQU 2FH ;END OF CHAR BUFFER
0008 = 105 ASBfSz EQU 08H ;Accelerate stpr mtr buf count
000A = 106 DSBfSz Equ 0AH ;Decelerate stpr mtr buf count
002F = 107 SMBFST EQU 2FH ;STPR MTR BUFFER START
0025 = 108 SMBfEnd Equ 25H ;Stpr Mtr Data Memory Address end
007F = 109 DMTop Equ 7FH ;Data Memory Top
005D = 110 DMSize Equ 93 ;Data Memory Size (less two working reg's)
= 111 ;
0020 = 112 LastPh Equ 20H ;last phz psuedo reg addr
0021 = 113 CPsAdr EQU 21H ;CR phz psuedo reg
0022 = 114 LPSAdr Equ 22H ;LF phz psuedo reg
0023 = 115 PTAscS Equ 23H ;Char Print Test code start tmp store
= 116 ;
= 117 ; PG
= 118 ; * * * * *
= 119 ;      Register allocation
= 120 ; * * * * *
= 121 ;
= 122 ;      All Indirect Data Memory Addressing via @Rn Inst must use
= 123 ;      only registers 0 & 1 of either register bank. Any other will
= 124 ;      be rejected by the Assembler
= 125 ;      Last character in table indicates Register Bank referenced
= 126 ;
= 127 ;      Register Bank 0
-----
0000 = 129 TmpR00 EQU R0 ;RBO Temporary Register
0001 = 130 TStrR0 EQU R1 ;Store Time Register RBO
0002 = 131 GStrR20 EQU R2 ;General Status Register RBO
0003 = 132 PhzR30 EQU R3 ;Stpr Mtr Phase Register RBO
0004 = 133 CntrR40 EQU R4 ;Count Reg. Phase count-Stpr Mtr loops
= 134 ;      Accel/Decel Count
0005 = 135 TConR0 EQU R5 ;Time constant reg RBO
0006 = 136 LnCtR0 EQU R6 ;Line count
= 137 ;
0007 = 138 OpnR70 EQU R7 ;available
= 139 ;
= 140 ;      Register Bank 0 Data Memory Address
= 141 ;
-----

```

```

0000 = 142 TmpA00 Equ 00H ; Temporary Register DM address
0001 = 143 TStrA0 EQU 01H ; Time Store Register DM address
0002 = 144 QStrAd Equ 02H ; RBO Char Status Reg DM address
0003 = 145 PhzA20 EQU 03H ; Stpr Mtr Phase Register DM address
0004 = 146 CntrA0 Equ 04H ; Count Reg. Phase count-Stpr Mtr loops
      = 147 ; Accel/Decel Count DM address
0005 = 148 TConA0 Equ 05H ; Time constant reg DM address
0006 = 149 LnCtA0 Equ 06H ; Line Count Register DM address
      = 150
0007 = 151 OpnA70 EQU 07H ; available
      = 152
      = 153
      = 154 ; PG
      = 155 ; -----
      = 156 ; RBO Status Byte Bit Definition
      = 157 ; -----
      = 158 ;
      = 159 ; Bit Definition
      = 160 ;
      = 161 ; 7 Stpr Mtr Direction: L-to-R = 1, R-to-L = 0
      = 162 ; 6 1 = Sink / 0 = Not Sinking, Print Head Init and Fire
      = 163 ; 5 Stpr Mtr at speed and CR not left of Home
      = 164 ; 4 Accel/Decel Init, 1 = Done / 0 = Not Done
      = 165 ;
      = 166 ; 3 1 = FailSafe / 0 = Constant, Time Window
      = 167 ; 2 1 = Form Feed / 0 = Line Feed
      = 168 ; 1 1 = Do Not Print / 0 = Print
      = 169 ; 0 FAccel/DAccel drive Ready = 1/NotRdy = 0 (exit
      = 170 ; drive & decel stpr mtr)
      = 171 ;
      = 172 ; Bit Masks: RBO
      = 173 ; Stepper Motor control bit masks function on QStr10
      = 174 ; -----
      = 175 ;
      = 176 LRPPrnt Equ 80H ; Left to Right Printing (ORL)
      = 177 RLPPrnt Equ 7FH ; Right to Left Printing (ANL)
      = 178 SnkStt Equ 40H ; Ready Print flag
      = 179 ClrSnk Equ 0BFH ; clear Ready to Print Bit
      = 180 AtSpdF Equ 20H ; Stpr Mtr at constant speed
      = 181 NAtSpd Equ 0DFH ; Stpr Mtr Not at speed
      = 182 ADIntD Equ 10H ; Accel/Decel Init Done
      = 183 ADIntN Equ 0EFH ; Accel/Decel Init Not Done
      = 184 ;
      = 185 FsCtm Equ 08H ; FailSafe/Constant Time
      = 186 ClrFsC Equ 0F7H ; Clear FailSafe/Const time flag
      = 187 FrmFd Equ 04H ; do formfeed
      = 188 LineFd Equ 0FBH ; do line feed
      = 189 DoNotP Equ 02H ; set Do Not Print Stat bit
      = 190 OkPrnt Equ 0FDH ; Reset - Ok to Print
      = 191 Ready Equ 01H ; Ready drive Stpr Mtr
      = 192 NotRdy Equ 0FEH ; Not Ready exit Stpr Mtr drive
      = 193 ;
      = 194 ;
      = 195 ; PG
      = 196 ; * * * * *
      = 197 ; Register allocation (cont)
      = 198 ; * * * * *
      = 199 ;
      = 200 ;
      = 201 ; Register Bank 1
      = 202 ; -----
0000 = 203 TmpR10 Equ R0
0001 = 204 CAdrR1 EQU R1 ; char data memory addr register
0002 = 205 ChStr1 EQU R2 ; char processing status byte register
0003 = 206 CDbtCR1 EQU R3 ; Char Dot count register
0004 = 207 CDbtR1 EQU R4 ; Char dot temp storage register
0005 = 208 CCntR1 EQU R5 ; Char count temp register
0006 = 209 StrCR1 EQU R6 ; Store Char Register
      = 210
0007 = 211 OpnR71 EQU R7 ; Available
      = 212
      = 213 ; Register Bank 1 Data Memory Address
      = 214 ; -----

```

```

0018 = 215 TmpA10 Equ 24 ;temporary/scratch register
0019 = 216 ChARR1 EQU 25 ;char data memory addr register
001A = 217 ChStAd EQU 26 ;RB1 Char Status Reg address
001B = 218 CdtCA1 EQU 27 ;Char Dot count register
001C = 219 CdotA1 EQU 28 ;Char dot temp storage register
001D = 220 CCntA1 EQU 29 ;Char count temp register
001E = 221 StrCA1 EQU 30 ;Store Char Register
    = 222
001F = 223 OpnA71 EQU 31 ;Available
    = 224
    = 225
    = 226 ; PG
    = 227
    = 228 ; -----
    ; RB1 Status Byte Bit Definition
    ; -----
    = 229
    = 230
    = 231 ; Bit Definition
    = 232 ; -----
    = 233 ; 7 Stpr Mtr Direction: L-to-R = 1, R-to-L = 0
    = 234 ; 6 Char Init, 1 = Done / 0 = Not Done
    = 235 ; 5 Char Lookup Table Page: 1 = Pgt, 0 = Pg2
    = 236 ; 4 1 = Test / 0 = Normal char print/input
    = 237
    = 238 ; 3 1 = EOF / 0 = Not EOF
    = 239 ; 2 Full = 1/Not Full = 0, Line in Char Buffer
    = 240 ; 1 1 = CR/(LF) / 0 = Not CR/(LF)
    = 241 ; 0 1 = Init / 0 = Do Not Init, CB registers done
    = 242
    = 243 ; Bit Masks: RB1
    = 244 ; Character printing bit masks function on ChStr1
    = 245
0080 = 246 ChrPrn Equ 80H ;Stpr Mtr Direction: L-to-R = 1
007F = 247 ClrCPr Equ 7FH ;Stpr Mtr Direction: R-to-L = 0
0040 = 248 ChIntD Equ 040H ;Set Char Init Done
00BF = 249 CIntND Equ 0BFH ;Reset Char Init Not Done
0020 = 250 ChOnP1 Equ 20H ;Page 1 char, set reentry bit (ORL)
00DF = 251 ChOnP2 Equ 0DFH ;Page 2 char, reset reentry bit (ANL)
0010 = 252 TstPrn Equ 10H ;Char print test
00EF = 253 NrmPrn Equ 0EFH ;Normal char input
    = 254
0008 = 255 EOF Equ 08H ;set EOF flag
00F7 = 256 ClrEOF Equ 0F7H ;clear EOF flag - Not EOF
0004 = 257 CRLF Equ 04H ;CR/LF
00FB = 258 ClrCR Equ 0FBH ;Clear CR/LF
0002 = 259 CBFLn Equ 02H ;Full Line in Char Buffer
00FD = 260 NCBFLn Equ 0FDH ;Not Full Line in Char Buffer
0001 = 261 IntCBR Equ 01H ;Init of CB registers done
00FE = 262 CIICBR Equ 0FEH ;Init of CB registers not done
    = 263
    = 264
    = 265 ; PG
    = 266 ; * * * * *
    = 267 ; Equates (cont)
    = 268 ; * * * * *
    = 269
    = 270 ; Misc
    = 271 ; -----
0004 = 272 RLPShf Equ 04H ;R-to-L print lookup table addr shift
    = 273
0020 = 274 Ascii Equ 20H ;hex nbr of first Ascii Char
007F = 275 AscLst Equ 7FH ;hex nbr of last Ascii Char
    = 276
00F3 = 277 CRCpl Equ 0F3H ;ASCII control code 2's complement
00F6 = 278 LFCpl Equ 0F6H ; "
00F4 = 279 FFCpl Equ 0F4H ; "
00E5 = 280 EscCpl Equ 0E5H ; "
00E0 = 281 AscCpl Equ 0E0H ; "
00CB = 282 FTCpl Equ 0CBH ; "
000D = 283 CR Equ 0DH ;Ascii code (hex)
0020 = 284 Space Equ 20H ;Ascii code (hex)
    = 285
00B1 = 286 LAsEnd Equ 81H ;Ascii End 2's cpl - test line start
00B2 = 287 PAsEnd Equ 82H ;Ascii End 2's cpl - within line print
007F = 288 AscStp Equ 7FH ;Ascii mask, strip off MSB
0042 = 289 PgLnCnt Equ 66 ;Page Line Count: Default = 66
00C4 = 290 PgLCpl Equ 0C4H ;Printed lines per page test
001B = 291 EOFcpl Equ 1BH ;EOF ascii code cpl
    = 292
    = 293 ; Loop count values
    = 294 ; -----

```

```

0006 = 295 NDtCct Equ 06H ;Normal Dot Column Count
000A = 296 EDtCct Equ 0AH ;Expanded Dot Column Count
0004 = 297 PHCnt1 EGU 04H ;NUMBER OF 8M PHASES ON INIT
    = 298
0004 = 299 ILFCnt Equ 04 ;Init LF step/phz count
0024 = 300 LPI6p6 Equ 36 ;Lines Per Inch 6.6
0018 = 301 LPI8p8 Equ 27 ;Lines Per Inch 8.8
0018 = 302 LPI10 Equ 24 ;Lines Per Inch 10
    = 303
0001 = 304 LineCt Equ 01 ;linefeed count
0042 = 305 FmFdCt Equ 66 ;lines per formfeed count
0003 = 306 Status EGU 03H ;SEE BELOW FOR STATUS BYTE DEF.
    = 307 ; TEST: SET FOR CR STPR MTR CONTROL
    = 308
    = 309
= 310 ; PG

= 311 ; * * * * *
= 312 ; TIMER VALUES - UPI Timer/Counter is UP Counter
= 313 ; * * * * *
= 314 ; 12 MHz Clk timings
= 315 ; -----
0080 = 316 DLYCL EGU 80H ;DELAY COUNT Long
0030 = 317 DLYCS EGU 30H ;DELAY COUNT Short
00CC = 318 DlyTim EGU 256-52 ;TIME DELAY constant ~2.0ms
0000 = 319 FailTm EGU 256-256 ;FailSafe TIME = ~17.0ms
00CC = 320 CrTmr1 EGU 256-52 ;CR Stpr Mtr Phase TIME = ~2.08ms
008A = 321 CrTmr2 EGU 256-70 ;CR Stpr Mtr Phase TIME = ~2.40ms
0092 = 322 CrTmr3 EGU 256-110 ;CR Stpr Mtr Phase TIME = ~4.16ms
00C0 = 323 IntTmr2 EGU 256-64 ;Init Stpr Mtr Phase TIME = ~2.40ms
0098 = 324 LFTMR1 EGU 256-104 ;LF Stpr Mtr Phase TIME = ~4.16ms
    = 325
    = 326 ; I/O port bit masks
00DF = 327 NotBsy Equ 0DFH ;Not Busy
0020 = 328 Busy Equ 20H ;Busy
00EF = 329 Ack Equ 0EFH ;Ack
0010 = 330 ReSAck Equ 10H ;ReSet Ack
    = 331
    = 332 ; Misc bit Masks
000C = 333 StrpLF EGU 0CH ;Strip off all bits but LF Stpr Mtr
0003 = 334 StrpCR EGU 03H ;Strip off all bits but CR Stpr Mtr
    = 335
    = 336 ; Print Head fires on low going edge of Trigger
    = 337 ; bit #9 in dot column is masked off, always: P2, bit 6
0040 = 338 PTRGLO EGU 40H ;PH TRIGGER BIT - LOW
00C0 = 339 PTRGHI EGU 0C0H ;PH TRIGGER BIT - HIGH
    = 340
= 341 ; * * * * *
= 342 ; Stepper Motor Phase State Equates
= 343 ; * * * * *
= 344
= 345 ; Stepper Motor Phase Shift Index Offset Offset
0000 = 346 FStCRP EGU 00H ;F CR stpr mtr phase data start addr
0003 = 347 RStCRP EGU 03H ;R CR stpr mtr phase data start addr
0008 = 348 STLFF EGU 08H ;Paper feed stpr mtr phase data start addr
    = 349
= 350 ; CARRIAGE STEPPER MOTOR PHASE EQUATES
= 351 ; Forward (1 thru 4) & Reverse (4 thru 1) :
0001 = 352 CRMFP1 EGU 01B ;CR STPR MTR PHASE 1
0003 = 353 CRMFP2 EGU 11B ;CR STPR MTR PHASE 2
0002 = 354 CRMFP3 EGU 10B ;CR STPR MTR PHASE 3
0000 = 355 CRMFP4 EGU 00B ;CR STPR MTR PHASE 4
    = 356
= 357 ; LINE FEED STEPPER MOTOR PHASE EQUATES
= 358 ; Forward:
0004 = 359 LFMFP1 EGU 0100B ;LF STPR MTR PHASE 1
000C = 360 LFMFP2 EGU 1100B ;LF STPR MTR PHASE 2
0008 = 361 LFMFP3 EGU 1000B ;LF STPR MTR PHASE 3
0000 = 362 LFMFP4 EGU 0000B ;LF STPR MTR PHASE 4
    = 363
= 364 ; PG

```

```

= 365 ; * * * * *
= 366 ; STEPPER MOTOR SELECT & CONTROL [CURRENT LIMITING]
= 367 ; * * * * *
= 368 ;
= 369 ; PORT BIT ASSIGNMENT.
= 370 ;
= 371 ; \ \ \
= 372 ; S S S -
= 373 ; L C C
= 374 ; F R R
= 375 ; B 1
= 376 ; O 3
= 377 ; 2
= 378 ;
= 379 ; -----
= 380 ; 5 5 5 5
= 381 ; 3 2 1 0
= 382 ; CODING:
= 383 ; SLF 0 1 1 0 06H
= 384 ; SCR80 1 0 0 0 0AH
= 385 ; SCR132 1 1 0 0 0CH
= 386 ; SMOFF 1 1 1 0 0EH
= 387 ; W/SCR80 & SCR132 '0' [BOTH SELECTED]
= 388 ; DEFAULT IS TO 80 COL.
= 389 ; [DO NOT KNOW WHETHER SCR80='0' WILL
= 390 ; SELECT 80 COL ONLY] - REQUIRES TEST.
= 391 ;
000B = 392 SCR80 EGU 08H ;SELECT CR STPR MTR - 80 COL
= 393 ; w/LF STPR MTR OFF
000C = 394 SCR132 EGU 0CH ;SELECT CR STPR MTR - 132 COL
= 395 ; w/LF STPR MTR OFF
0006 = 396 SLF EGU 06H ;SELECT LF STPR MTR ON
= 397 ; w/CR STPR MTR OFF
000E = 398 SMOFF EGU 0EH ;SELECT CR & LF STPR MTR OFF
= 399
= 400
401 ; PG
402 ; * * * * *
403 ; MAIN PROGRAM BODY
404 ; * * * * *
405 ;
406 ; Power On / Reset Program Entry
407 ;
408 ; PROGRAM START
409 ;
0000 = 410 Org 00H
= 411
0000 040B = 412 START. JMP RESET
= 413
414 ; INPUT BUFFER FULL INTERRUPT CALL ENTRY AND VECTOR
0003 = 415 DRG 03H
0003 1425 = 416 IBFIV: Call IBFIS
0005 93 = 417 RETR
418 ; TIMER OVERFLOW INTERRUPT CALL ENTRY AND VECTOR
0007 = 419 DRG 07H
0007 1429 = 420 TMRIV: Call TMRIS
0009 C5 = 421 SEL R80
000A 83 = 422 Ret
= 423
424 ; INITIALIZATION
425 ;
000B 15 = 426 ResT: Dis I
000C 35 = 427 Dis TCntI
000D B40F = 428 Call InitAl ;set all critical outputs inactive
000F B42F = 429 Call ClrDM ;clear all data memory - 93H to 7FH
= 430 ; do not clear R80, R81 or Stack
0011 B44B = 431 Call InitCR ;CALL CR SM POWER ON INIT
0013 9400 = 432 Call InitLF ;CALL LF SM POWER ON INIT
= 433
434 ; MAIN PROGRAM LOOP
435 ; All program segments are called from here
436 ;
0015 B422 = 437 Home: Call CRHome ;Call Home CR routine -
= 438 ; fixes logical and physical CR Home
0017 B400 = 439 Call Defalt ;set default register values
0019 142C = 440 CBInpt: Call ESCBFF ;Stat Switch / CB Input Service Test
= 441 ; test for: CB full/fill, LF, FF.
= 442 ; Char Prnt Test

```

```

001B 3400      443 Repeat. Call  BMDriv      ;Call Forward Stpr Mtr Drive
001D 940D      444          Call  LFDriv      ;Call Linefeed Stpr Mtr Drive
001F D5        445          SEL    RB1
0020 FA        446          Mov    A, ChStR1      ;get the Char Status Register RB1
0021 7215      447          JBC    Home        ;jump to CR SM Home if EDF bit set
0023 0419      448          Jmp    CBInpt     ;loop to Char Buffer Input test
449
450 ; PG
451 ; *****
452 ;          Interrupt Service Routine
453 ; *****
454
455 -----
456 ;          Input Buffer Full Interrupt Service Routine
457 ; -----
458
459 IBFIS:
460 ; -----
461          Acknowledge Char input and set Hold/Busy Active
0025 8A20      462          ORL    P2, #Busy      ;get & set DBB ACK/Busy Bits
0027 15        463          Dis    I          ;disable IBF interrupts
0028 B3        464          Ret
465
466 -----
467 ;          Timer / Counter Interrupt Service Routine
468 ; -----
469          ITF interrupt service routine disables all intr during
470          stpr mtr phase shifting
0029 15        471 TMRIS: Dis    I          ;disable IBF interrupts
002A 35        472          Dis    TCnt1     ;dicable ITF interrupts
002B B3        473          Ret
474
475 ; PG
476 ; *****
477 ;          External Status Switch Check/Char Buffer Fill
478 ; *****
479 EBCBFF:
480          SEL    RB1          ;Prep for normal character handling/input
002C D5        481          Mov    A, ChStR1      ;get the character stat reg byte
002E 53EF      482          ANL    A, #NmPrn     ;set normal character input
0030 AA        483          Mov    ChStR1, A      ;store the stat byte
0031 C5        484          SEL    RBO
485
486 ;          Test External Status Port
0032 0F        487          MovD   A, P7          ;get the stat switch port bits
0033 123D      488          FormFd  FormFd      ; service Formfeed
0035 3245      489          JBC    LinFd      ; service Linefeed
0037 5249      490          JBC    ChrTst     ; service Character TEST
0039 725E      491          JBC    DnLine     ; service Char Buffer Check/Fill
003B 042C      492          Jmp    EBCBFF     ;Loop
493
494 FormFd: Mov    A, GStR20      ;get the status byte
003E 4304      495          ORL    A, #FrmFd     ;set the formfeed stat flag
0040 AA        496          Mov    GStR20, A      ;store trhe status byte
0041 940D      497          Call  LFDriv      ;do a formfeed
0043 042C      498          Jmp    EBCBFF
499
0045 940D      500 LinFd: Call  LFDriv      ;do a line drive
0047 042C      501          Jmp    EBCBFF
502 ; -----
0049 D5        503 ChrTst: SEL    RB1
004A FA        504          Mov    A, ChStR1      ;get the character stat reg byte
004B 4310      505          ORL    A, #TstPrn     ;set character test flag
004D AA        506          Mov    ChStR1, A      ;store the stat byte
004E BB23      507          Mov    TmpR10, #PTASCS ;load the psuedo Ascii code tmp reg addr
0050 F0        508          Mov    A, @TmpR10     ;get the inc'd ascii code
0051 03B1      509          ADD    A, #LAsEnd       ;test for code end
0053 9657      510          JNZ    AscCLd      ;if not code end jmp to load
511          ;if end restart ascii at beginning
0055 8020      512          Mov    @TmpR10, #Ascii ;store the ascii code start
0057 F0        513 AscCLd: Mov    A, @TmpR10     ;get the ascii code again
0058 AF        514          Mov    OpnR71, A          ;place in the empty register
0059 10        515          Inc    @TmpR10         ;Inc start ASCII char in data memory
005A B439      516          Call  PrnTst         ;call the DM load procedure
005C C5        517          SEL    RBO          ;reselect reg bank 0
005D B3        518          Ret
519 ; -----

```

```

005E D5      520 OnLine: SEL   RB1      ;select char buffer registers
005F 05      521      EN      I          ;enable interrupts
0060 FA      522 CbFCk1: Mov  A,ChStR1 ;get the Char Stat Byte
0061 3267    523      JB1      CBCKEx  ;if Chr Buf has full line exit
0063 146D    524 IBFCk: Call  CBFill   ;read a char into Char Buffer
0065 0460    525      Jmp     CBFCK1  ;loop to Char Buf Ful test
0067 C5      526 CBCKEx: SEL   RBO
0068 83      527      Ret
528
529 ; PG
530 ;
531 ; Character Input
532 ;
533 ; Input Buffer Full service routine. test for Char buffer full-exit
534 ; else load char into char buffer
0069 D5      535 IBFSrv: SEL   RB1
006A FA      536      Mov    A,ChStR1 ;get the RBO stat byte
006B 32EC    537      JB1      CBFULL  ;if Do Not Print Bit Set - EXIT
006D 527C    538 CBFill: JB2      CBPad   ;test for CB padding flag
539 ; if not pad enable char input
540 ; tell the host to send char's
006F 05      541      EN      I
0070 D6EC    542      JNIBF  CBF1Ex
543 ;
544 ; Acknowledge Char input and set Hold/Busy Active
0072 FA      545      Mov    A,ChStR1 ;get the RB1 Char Stat Byte
0073 127C    546      JB0      SkpInt  ;test for CB has been Initialized
547 ;
548 ; Init of all Char handling registers
0075 4301    549      ORL    A,#IntCBr ;set CB Reg skip Initialization stat bit
0077 AA      550      Mov    ChStR1,A ;save the altered stat byte
0078 B97F    551      Mov    CAdR1,#FCBFSt ;load char reg w/char bufr strt
007A BD50    552      Mov    CCnTR1,#ChBFSt ;load char cnt reg w/char bufr size
553 CBPad:
007C ED86    554 SkpInt: DJNZ   CCnTR1,LdChar ;DECREMENT BUFFER SIZE
007E FA      555      Mov    A,ChStR1 ;get the status byte
007F 4302    556      ORL    A,#CBFLn  ;set Char Buffer Full Line stat bit
0081 53FB    557      ANL   A,#ClCr   ;clear the CR/(LF) stat bit
0083 53FE    558      ANL   A,#ClCBr  ;reset CB Init bit: init CB reg on entry
0085 AA      559      Mov    ChStR1,A ;store the status byte
0086 FA      560 LdChar: Mov  A,ChStR1 ;get the status byte
0087 52E1    561      JB2      CBPad1  ;CB not full but CR/LF previously
562 ; received so pad CB
0089 9AEF    563      ANL   P2,#Ack   ;output DBB Ack low
008B 22      564      In     A,DBB     ;read the Char
008C 537F    565      ANL   A,#AscStp ;strip off MSB
008E AB      566      Mov    TmpR10,A ;temp save char
008F 8A10    567      ORL   P2,#RSAck ;output DBB ACK High
568 ;
569 ; test for ASCII printable character
0091 03E0    570      ADD   A,#ASCCpl  ;test for Carriage Return
0093 F697    571      JC   AsciiC    ;jmp to service
0095 049C    572      Jmp  AsciiC    ;
0097 97      573 AsciiC: C!r     C          ;clear carry flag
0098 FB      574      Mov    A,TmpR10 ;get the char back
0099 A1      575      Mov    @AdR1,A  ;load data memory w/Char
009A 04E3    576      Jmp  IBFSrE
577 ;
578 ; test for CR/LF: if CR/LF Strip off LF and exit setting
579 ; Char Buffer Init Stat bit
009C FB      580 ChrChk: Mov  A,TmpR10 ;get the char back
009D 03F3    581      ADD   A,#CRCpl  ;test for Carriage Return
009F C6C3    582      JZ   CRChr    ; if CR go service it
00A1 FB      583      Mov  A,TmpR10 ;get the char back
00A2 031B    584      ADD   A,#EODFCpl ;test for End Of File
00A4 96AA    585      JNZ  ChrCk1   ;if not EOF jmp to CB Pad
00A6 FB      586      Mov  A,TmpR10 ;if EOF, place it in CB
00A7 A1      587      Mov  @AdR1,A  ;load data memory w/CR Char
00A8 04B9    588      Jmp  ExtSet   ;Exit
00AA FB      589 ChrCk1: Mov  A,TmpR10 ;get the status byte
00AB 03F4    590      ADD   A,#FFCpl  ;test for FormFeed
00AD 96E1    591      JNZ  CBPad1   ;if not FF Pad the CB
00AF C5      592      SEL  RBO
00B0 FA      593      Mov  A,GBTR20 ;get the status byte
00B1 4304    594      ORL  A,#FrmFd  ;set the formfeed flag
00B3 AA      595      Mov  A,GBTR20 ;store the status byte
00B4 D5      596      SEL  RB1
00B5 FA      597      Mov  A,ChStR1 ;get the status byte
00B6 4304    598      ORL  A,#CRLF  ;set CRLF stat bit: pad balance of CB
599 ; with Spaces until fill
00BB AA      600      Mov  ChStR1,A ;store the status byte
00B9 FA      601 ExtSet: Mov  A,ChStR1 ;get the status byte

```



```

00BA 4302      602      ORL      A,#CBFLn      ;set Char Buffer Full Line stat bit
00BC 53FB      603      ANL      A,#C1rCr      ;clear the CR/(LF) stat bit
00BE 53FE      604      ANL      A,#C1ICBR     ;reset CB Init bit: init CB reg on entry
00C0 AA        605      Mov      ChStR1,A      ;store the status byte
00C1 04EC      606      Jmp      CBF1Ex       ;Exit
607 ;-----
00C3 FB        608 ; Store CR char read in LF char (assume its always there) and ignor it
00C4 A1        609 CRChr:  Mov      A,TmpR10     ;get the char back
610          610          Mov      @CAdrR1,A     ;load data memory w/CR Char
00C5 C5        611          SEL      RBO
00C6 1E        612          INC      LnCtRO       ;inc the line count
00C7 FE        613          Mov      A,LnCtRO     ;get the line count
00CB 03C4      614          Add      A,#PgLcPl     ;test for page feed ln cnt
615          615          ; if LnCt => PgLcPl set formfeed flag
00CA E6D0      616          JNC      NoFmFd        ;if not at end of page skip
00CC FA        617          Mov      A,GStR2O     ;get the status byte
00CD 4304      618          ORL      A,#FrmFd     ;set the form feed status flag
00CF AA        619          Mov      GStR2O,A     ;save the status byte
00D0 D5        620 NoFmFd: SEL      RB1
00D1 05        621          EN      I            ;enable the IBF service
00D2 9ADF      622          ANL      P2,#NotBsy  ;output a not busy to Host
00D4 D6D4      623 LFTest: JNIBF      LFTest   ;loop to next char
00D6 9AEF      624          ANL      P2,#Ack     ;output DBB Ack low
00DB 22        625          IN      A,DBB       ;get next Char - assume it's a LF
626          626          ; and ignor it (LF is forced upon
627          627          ; detection of CR at print time)
00D9 FA        628 SetPad:  Mov      A,ChStR1   ;get the status byte
00DA 4304      629          ORL      A,#CRLF     ;set CRLF stat bit: pad balance of CB
630          630          ; with Spaces until fill
00DC AA        631          Mov      ChStR1,A     ;store the status byte
00DD 8A10      632          ORL      P2,#ReSack   ;output DBB ACK High
00DF 04E3      633          Jmp      IBFSBE       ;jmp to addr step & exit
634 ;-----
00E1 B120      635 ; fill Char Buffer with space
636          636          CBPad:  Mov      @CAdrR1,#Space ;load data memory w/Char
637 ;-----
00E3 C9        638 ; step the char address test for CB full &/or pad
00E4 FA        639 IBFSBE:  DEC      CAdrR1     ;Decrement dat memory location
00E5 32EC      640          Mov      A,ChStR1   ;get the status byte
00E7 52EC      641          JB1      CBFu1l     ;test for CB Full
642          642          JB2      CBF1Ex     ;test for CB pad - exit w/Busy set
643 ;-----
00E9 05        644          Set Busy Line Low - Not Busy
00EA 9ADF      645          EN      I            ;output a not busy to Host
646          646          ANL      P2,#NotBsy
647 ;-----
648          648          ;exit w/ Busy Still set high
00EC 83        649 CBFu1l:
650          650          CBF1Ex: Ret
651 ;-----
652 ; PG
653 ; * * * * *
654 ; L-to-R/R-to-L Carriage Stepper Motor Drive
655 ; and Line Printing
656 ; * * * * *
0100          657
0100 3622      658          DRQ      100H
659 ;-----
0100          660 SMDriv:  JTO      RAccel   ;if Print Head at left drive right
661          661          ; else drive left
662 ; F-----
663          663          FAccel: ;L-to-R Accelerate Stepper Motor
664          664          ; Set the Forward acceleration/drive Entry status bits
0102 FA        665          Mov      A,GStR2O     ;get the status byte
0103 53BF      666          ANL      A,#C1rSnk   ;set not at speed flag = 0
0105 53DF      667          ANL      A,#NAtSpd   ;set Not At Speed flag = 0
0107 4380      668          ORL      A,#LRPrnt   ;set L-to-R prnt stat bit = 1
0109 4301      669          ORL      A,#Ready     ;set stpr mtr ready - Drive On
010B 53EF      670          ANL      A,#ADIntN   ;set A/D Init Not Done
010D AA        671          Mov      GStR2O,A     ;store the status byte
010E D5        672 CBRDir:  SEL      RB1
010F FA        673          Mov      A,ChStR1   ;get the Char Stat Reg Data Mem Addr
0110 4380      674          ORL      A,#LRPrnt   ;Set L-to-R prnt bit
0112 AA        675          Mov      ChStR1,A     ;save the Char Stat byte
0113 C5        676          SEL      RBO
677 ;-----
0114 8B21      678 ; Restore the phase register index addresses
0116 FO        679          Mov      TmpROO,#CPSAdr ;get Phz Storage Addr psuedo reg
0117 AB        680          Mov      A,@TmpROO   ; get stored CR last phase index addr
681          681          Mov      PhzR3O,A   ;place last LF phase index addr in Phz Reg
682

```

```

683 ; Set up for next phase bit output before entering timing loops
0118 1B 684 INC PhzR30 ;STEP PHASE DB ADDRESS
0119 FB 685 MOV A,PhzR30 ;CHECK THE PHASE COUNT REG
011A 521E 686 JB2 IAFzrP ;CHK FOR COUNT BIT ROLLOVER
011C 2440 687 JMP SMDf1t ;skip adr index reset
011E 8B00 688 IAFzrP MOV PhzR30,#FStCRP ;ZERO CR SM PHASE REGISTER
0120 2440 689 Jmp SMDf1t
690
691 ; R=====
692 RAccel ;R-to-L Accelerate Stepper Motor
693 ;-----
694 ; Set the Reverse acceleration/drive Entry status bits
0122 FA 695 Mov A,GSTR20 ;get the status byte
0123 53BF 696 ANL A,#ClrSnk ;clear Print Ready bit
0125 53DF 697 ANL A,#NAtSpd ;set Not At Speed flag = 0
0127 537F 698 ANL A,#RLPrnt ;set R-to-L prnt status bit
0129 4301 699 ORL A,#Ready ;set stpr mtr ready - Drive On
012B 53EF 700 ANL A,#ADIntN ;set A/D Init Not Done
012D AA 701 Mov GSTR20,A ;store the status byte
012E D5 702 RCBDR: SEL RB1
012F FA 703 Mov A,ChSTR1 ;get the Char Stat Reg Data Mem Addr
0130 537F 704 ANL A,#RLPrnt ;Set R-to-L print bit
0132 AA 705 Mov ChSTR1,A ;save the Char Stat byte
0133 C5 706 SEL RBO
707 ;-----
708 ; Restore the phase register index address
0134 8B21 709 Mov TmpROO,#CPSAdr ;get Phz Storage Addr psuedo reg
0136 F0 710 Mov A,@TmpROO ; get stored CR last phase index addr
0137 AB 711 Mov PhzR30,A ;place last LF phase index addr in Phz Reg
712 ; Set up for next phase bit output before entering timing loops
0138 CB 713 Dec PhzR30 ;STEP PHASE DB ADDRESS
0139 FB 714 MOV A,PhzR30 ;CHECK THE PHASE COUNT REG
013A 523E 715 JB2 IARzrP ;CHK FOR COUNT BIT ROLLOVER
013C 2440 716 JMP SMDf1t
013E 8B03 717 IARzrP: MOV PhzR30,#RStCRP ;ZERO CR SM PHASE REGISTER
718
719 SMDf1t.
720 ;-----
721 ; for stablization of unused stpr mtr during CR stpr mtr drive,
722 ; store the unused stpr mtr current phase bits
0140 8B22 723 Mov TmpROO,#LPSAdr ;get the CR phz storage addr
0142 F0 724 Mov A,@TmpROO ;get the byte stored there
0143 E3 725 MovP3 A,@A ;get the phz data byte
0144 8B20 726 Mov TmpROO,#LastPh ;load Last Phz psuedo reg to Temp Reg
0146 A0 727 Mov @TmpROO,A ;store Last Phase bits - indirect
728
729 ; SetUp Stpr Mtr Time Constant
0147 BDBA 730 MOV TConRO,#CrTmr2 ;Load time constant Reg
731
732 Select. ;Select the Stpr Mtr
0149 2308 733 MOV A,#SCRBO ;GET CR SM SELECT BITS
014B 3D 734 MOVD P5,A ;SELECT SM [SCRBO]
735 ;-----
736 ; SetUp Stpr Mtr Phase Shift index address register
737 ; Output next phase and init timer to Std Time constant
014C FD 738 STRTT: MOV A,TConRO ;get time constant from reg
014D 62 739 MOV T,A ;load the timer
014E FB 740 MOV A,PhzR30 ;get the phz reg indirect addr index
014F E3 741 MovP3 A,@A ;do indirect get of phz bits
742
743 ;-----
744 ; patch together the CR last and LF next phase bits
0150 8B20 745 Mov TmpROO,#LastPh ;load Last Phz psuedo reg to Temp Reg
0152 40 746 ORL A,@TmpROO ;patch together CR existing & new LF
0153 3C 747 MOVD P4,A ;OUTPUT BITS
0154 55 748 STRT T ;START TIMER
749
750 ; At start of timing loop do all Stpr Mtr Accel/Decel or
0155 740C 751 ; Character SetUp overhead:
752 Call ADPTst ;call Accel/Decel/Print Test
753
754 ; Set up for next phase bit output before entering timing loops
755 PNRdy1: ;test for forward / reverse phase start indirect index to load
0157 FA 756 Mov A,GSTR20 ;store stat byte
0158 F264 757 JB2 Ac1F2
758
759 ; reverse:
760 ; Set up for next phase bit output before entering timing loops
015A CB 761 Dec PhzR30 ;STEP PHASE DB ADDRESS
015B FB 762 MOV A,PhzR30 ;CHECK THE PHASE COUNT REG
015C 5260 763 JB2 ARzrP ;CHK FOR COUNT BIT ROLLOVER
015E 2462 764 JMP ARNxtP
0160 8B03 765 ARzrP: MOV PhzR30,#RStCRP ;ZERO CR SM PHASE REGISTER
0162 246C 766 ARNxtP: Jmp ANxtPh

```

```

767
768 ; forward
769 ; Set up for next phase bit output before entering timing loops
0164 1B 770 AC1F2: INC PhzR30 ;STEP PHASE DB ADDRESS
0165 FB 771 MOV A,PhzR30 ;CHECK THE PHASE COUNT REQ
0166 526A 772 JB2 AFZroP ;CHK FOR COUNT BIT ROLLOVER
0168 246C 773 JMP ANxtPh ;skip adr index reset
016A B800 774 AFZroP: MOV PhzR30,#FStCRP ;ZERO CR SM PHASE REGISTER
775 ANxtPh:
776 ; -----
777 ; stage one timer loop - T occurs before Std timeout
778 ; wait for time out
016C 1682 779 TLOOP2: JTF FAILSF ;JMP ON TIME OUT-t DOES NOT OCCUR 1ST
016E 5672 780 JT1 tCHK1 ;IS T HIGH-JMP TO tCHK
0170 246C 781 JMP TLOOP2 ;LOOP FOR JT1 OR JTF
0172 00 782 tCHK1: NOP ;delay, then double check T signal
0173 5677 783 JT1 tTruW1 ;JUMP T TEST TRUE-WAIT FOR JTF
0175 246C 784 JMP TLOOP2
785 tTruW1:
786 ; test for Print Ready bit - was Print Head Fire Setup Done?
787 ; insert acceleration time/store time count done/notdone flag bit
0177 FA 788 Mov A,QStr20 ;get the status byte - prep for prnt
0178 D27C 789 JB6 RdyPr2 ;if Ready Print bit set call PHFire
017A 247E 790 JMP SkpPHF ; else skip Print Head Fire
017C 74CA 791 RdyPr2: Call PHFire ;print head solenoid fire routine
792 PNRdy2:
793 SkpPHF:
017E 1698 794 tTruW2: JTF NXTPHZ ;JUMP TO SM ERROR
0180 247E 795 JMP tTruW2 ; LOOP TO TLOOP3
796 ; -----
797 ; Step into failsafe/startup timer setup - T does not
798 ; occurs before Std Time timeout, load failsafe SM protection
799 ; time and wait for failsafe timeout or T. If T occurs
800 ; output phase immediately after T verify.
0182 2300 801 FAILSF: MOV A,#FailTm ;LOAD TIMER W/~15.0ms
0184 62 802 MOV T,A ; SM PROTECTION TIMEOUT
0185 55 803 STRT T ;START TIMER
804 ; -----
805 ; set the Status bit for Store time test
0186 FA 806 Mov A,QStr20 ;get the status byte
0187 4308 807 ORL A,#FSCtm ;set Failsafe/constant time flag
0189 AA 808 Mov QStr20,A ;store the status byte
018A 5690 809 TLOOP3: JT1 tCHK2 ;IS T HIGH
018C 16AC 810 JTF DSLECT ;IF TIME OUT GD SM ERROR
018E 248A 811 JMP TLOOP3 ;LOOP UNTIL T HIGH OR T-OUT
0190 00 812 tCHK2: NOP ;WAIT
0191 5695 813 JT1 StrTm1 ;jump out and store elapsed time
0193 248A 814 JMP TLOOP3 ; JMP TO FAILSF LOOP
0195 65 815 StrTm1: Stop TCnt ;stop the failSafe Timer
0196 42 816 Mov A,T ;read the timer
0197 A1 817 MOV @TStrRO,A ;Store the time read in indexed addr
818 ; - next entry to A/D Memorize Time
819 ; routine will add time constant to it
820
821 ; Test is CR Stpr Mtr Drive is finished prior to next phase output
822 ; -----
823 NXTPHZ:
824 ; test for forward / reverse phase start indirect index to load
0198 FA 825 Mov A,QStr20 ;store stat byte
0199 F2A7 826 JB7 FDrive
827 ; Reverse -- test for Reverse Stpr Mtr Drive procedure exit
828 ; ALWAYS drive the CR to the left most HOME position
019B 26AC 829 JNTO EOLn ;test if home position jmp stop
019D FA 830 Mov A,QStr20 ;get the status byte
019E 124C 831 JBO StrtT ;test Ready stat bit:
832 ; if bit 0 = 1 then Print More
01A0 4302 833 ORL A,#DoNotP ;set the do not print flag
01A2 53BF 834 ANL A,#ClrBnk ;clear Print Ready bit
01A4 AA 835 Mov QStr20,A ;save the status byte
01A5 244C 836 JMP StrtT ;continue CR SM drive
837 ; - only exit is HR
838 ; Forward -- test for Forward Stpr Mtr Drive procedure exit
839 FDrive:
01A7 FA 840 Mov A,QStr20 ;get the status byte
01A8 124C 841 JBO StrtT ;test Ready stat bit
842 ; if bit 0 = 1 then Print More
01AA 244C 843 JMP EOLn ; else jmp to End Of Line exit
844 ;jump to start timer again
845 DSLECT:
01AC 5437 846 EOLn: Call DecISM ;call Sptr Mtr Deceleration
847 ; -----
848 ; test for forward / reverse phase start indirect index to load
01AE FA 849 Mov A,QStr20 ;store stat byte
01AF F2B3 850 JB7 FDrvFS ;jmp to f drive flag set

```

```

01B1 53FD      851      ANL      A,#OkPrnt      ;reset print flag - Ok Print
                        852                        ; only if printing R-to-L
                        853
                        854      ; update the status byte
01B3 53BF      855 FDrVFS ANL      A,#ClrSnk      ;clear Print Ready bit
                        856                        ;set the Status bit for Store time test
01B5 53DF      857      ANL      A,#NAtSpd     ;Clear At Print Speed Bit
01B7 AA        858      Mov      GStR20,A        ;save the status byte
01B8 83        859      RET
                        860
                        861      ; PG
                        862      ; *****
                        863      ; Stepper Motor Accel. Time Storagee
                        864      ; *****
0200          864      ORG      200H
                        865
0200 920C      866 ADMmTS: JB4      DADInt      ; Entry has Gen Stat Byte in A
                        867                        ;is A/D init done - then jmp
                        868
                        869
                        870      ; 1st Entry initializes the A/D Time store working registers
0202 892F      871      Mov      TStrRO,#SMB#St ;Load the Stpr Mtr Buffer Start Addr
0204 8C08      872      Mov      Cntr40,#ASB#Sz ;Load the Buffer Size
0206 FA        873      Mov      A,GStR20      ;get the status byte
0207 4310      874      ORL      A,#ADIntD    ;set not 1st Accel Entry Flag
0209 AA        875      Mov      GStR20,A        ;store the status byte
020A 4436      876      Jmp      ADExit      ;exit - 1st entry has not generated
                        877                        ; a closed time window
                        878
                        879      ; Step the A/D Store count
020C EC26      880 DADInt: DJNZ     Cntr40,StorCt ;dec Times to store count
                        881                        ;if not 0 store the count
                        882                        ;else at end-set done flag
020E FA        883      Mov      A,GStR20      ;get the status byte
020F 4320      884      ORL      A,#AtSpdF    ;set at speed/no more to store flag
0211 AA        885      Mov      GStR20,A        ;store the status byte
                        886
                        887      ; Initialize Char Print Registers if printing enabled
0212 3226      888      JB1      StorCt      ;if Do Not Print stat bit set
                        889                        ; Skip the Char register init
                        890
                        891      ; Initialize all Char Reg's
                        892      ; Test for L-to-R (forward) or R-to-L (reverse) printing
0214 D5        893      SEL      RB1
0215 FA        894      Mov      A,ChStR1      ;get the status byte
0216 4340      895      ORL      A,#CHIntD    ;set Char Init Done flag - bypass
0218 AA        896      Mov      ChStR1,A        ;save the status byte
0219 F21F      897      JB7      LdCBR1      ;test Chr Stat Byte Returned
                        898                        ; if bit 7 = 1 then Prnt L-to-R
021B 892F      899 LdCBR: Mov      CAdrR1,#RCB#IS ;load char reg w/char bufr strt R-to-L
021D 4421      900      Jmp      LdCBR2
                        901
021F 8980      902 LdCBR1: Mov     CAdrR1,#FCB#IS ;load char reg w/char bufr strt L-to-R
                        903
0221 8D51      904 LdCBR2: Mov     CCntr1,#ChB#IS ;load char cnt reg w/char bufr size
0223 BB01      905      Mov      CDtCR1,#01      ;set the chr dot column cnt
0225 C5        906      SEL      R80
                        907
                        908      ; Test for t > Tc or t < Tc
0226 722C      909 StorCt: JB3      FailST      ;test for failsafe time switch
                        910
                        911      ; t < Tc = store Time Constant in use
0228 FD        912      Mov      A,TConRO      ;Get time constant currently in use
0229 A1        913      Mov      @TStrRO,A        ;Memorize/Store the time - indirect addr
022A 4435      914      Jmp      ADPRet
                        915
                        916      ; t > Tc = store Time Constant + FailSafe Time Elapsed
                        917      ; [see Accel/Cnst Speed/Decel WaveForm]
                        918      ; equation is: Trd - FailSafe Time = Tx
                        919      ; => Trd + Cpl(FailSafe Time) = Tx
                        920      ; Tx + Tcnst = T
                        921      ; Store/Memorize T
                        922
022C F1        923 FailST: Mov     A,@TStrRO ;get the stored time
022D 03C8      924      Add      A,#FTCp1      ;2's cpl add
022F 6D        925      Add      A,TConRO      ;Add: Time stored + Time constant
                        926                        ; currently in use
0230 A1        927      Mov      @TStrRO,A        ;Memorize/Store the time
                        928      ; Reset the Status bit for Store time test
                        929
0231 FA        930      Mov      A,GStR20      ;get the status byte
0232 83F7      931      ANL      A,#ClrFBC    ;reset Failsafe/constant time flag
                        932                        ; assumes entry via constant time

```

```

0234 AA      933      Mov      QStr20,A      ;store the status byte
0235 C9      934 ADPRet: Dec      TStrR0      ;step the A/D time data store addr
0236 B3      935 ADExit: Ret
          936
          937 ; PG
          938 ; * * * * *
939 ;      Carriage Stepper Motor Deceleration
          940 ; * * * * *
          941
          942 DeclSM:
          943 ;      SetUp the Deceleration registers
0237 B925    944      Mov      TStrR0,#SMBEnd ;Load the Strp Mtr Buffer End Addr
0239 BC0A    945      Mov      Cntr40,#DSBfSz ;Load the Buffer Size
0238 FB      946      MOV      A,PhzR30      ;get phase index address
023C E3      947      MovP3   A,@A          ;get phase from indexed address
          948 ;      patch together the CR last and LF next phase bits
023D B820    949      Mov      TmpR00,#LastPh ;load Last Phz psuedo reg to Temp Reg
023F 40      950      ORL      A,@TmpR00     ;patch together CR existing & new LF
0240 3C      951      MOVVD   P4,A          ;OUTPUT BITS
0241 F1      952 StrttD: MOV      A,@TStrR0   ;get time from indexed data memory
0242 62      953      MOV      T,A          ;load timer
0243 55      954      STRT   T            ;START TIMER
0244 19      955      Inc      TStrR0      ;step the Memorized time addr index reg
          956 ;      test for forward / reverse phase start indirect index to load
0245 FA      957      Mov      A,QStr20     ;store stat byte
0246 F252    958      JB7      DclF2
          959
          960 ; reverse:
          961 ;      Set up for next phase bit output before entering timing loops
0248 CB      962      Dec      PhzR30      ;decrement the phase addr
0249 FB      963      MOV      A,PhzR30     ;Get the phz data addr
024A 524E    964      JB2      DRzroP      ;CHK FOR COUNT BIT ROLLOVER
024C 445A    965      JMP      DNxtPh      ;skip adr index reset
024E B803    966 DRzroP: MOV      PhzR30,#RStCRP ;ZERO CR SM PHASE REGISTER
0250 445A    967      Jmp      DclR2
          968
          969 ; forward:
          970 ;      Set up for next phase bit output before entering timing loops
0252 18      971 DclF2: Inc      PhzR30      ;increment the phase addr
0253 FB      972      MOV      A,PhzR30     ;Get the phz data addr
0254 525B    973      JB2      DZroPh      ;CHK FOR COUNT BIT ROLLOVER
0256 445A    974      JMP      DNxtPh      ;skip adr index reset
0258 B800    975 DZroPh: MOV      PhzR30,#FStCRP ;ZERO CR SM PHASE REGISTER
          976 DNxtPh: ;set up for next phase shift
025A FB      977 DclR2: MOV      A,PhzR30     ;get phase index address
025B E3      978      MovP3   A,@A          ;get phase from indexed address
          979 ;      patch together the CR last and LF next phase bits
025C B820    980      Mov      TmpR00,#LastPh ;load Last Phz psuedo reg to Temp Reg
025E 40      981      ORL      A,@TmpR00     ;patch together CR existing & new LF
025F 1663    982 TLoopD: JTF      NxtPD2     ;JMP ON TIME OUT TO NEXT PH
0261 445F    983      JMP      TLoopD       ;LOOP UNTIL TIME OUT
0263 3C      984 NxtPD2: MOVVD   P4,A          ;OUTPUT BITS
0264 EC41    985      DJNZ   Cntr40,StrttD ;Exit Test
          986
          987 ;      Set Storage of next phase data in psuedo addr. This insures
          988 ;      next phase is sequence correct for strp mtr drive direction
0266 B821    989 SetRN: Mov      TmpR00,#CPSAdr ;get Phz Storage Addr psuedo reg
0268 FB      990      MOV      A,PhzR30     ;get Phz data
0269 AD      991      Mov      @TmpR00,A      ;store CR Next phase index addr
026A B47B    992 DMExit: Call   DlyLng
026C B490    993      Call   DeSISM
026E B3      994      RET
          995
          996 ; PG
          997 ; * * * * *
998 ;      Stepper Motor Phase Shift Definitions
          999 ;      All program procedures call this data.
1000 ; * * * * *
1001
1002      ORG      300H
1003
1004 ;      DEFINE PHASE ADDRESSES:
1005 ;      THE PHASE DATA IS ENCODED TO THE ADDRESS CALLED DURING THE
1006 ;      STPR MTR ENERGIZE SEQUENCE CORRESPONDING TO THE NEXT PHASE
1007 ;      OF THE SEQUENCE REQUIRED.
1008
1009 ;      CARRAGE MOTOR ENCODING: FORWARD - LEFT-to-RIGHT
1010 ;      REVERSE - RIGHT-to-LEFT
1011

```

```

1012 Reverse direction ENCODING is the same bytes accessed in
1013 reverse direction
1014
0300 01 1015 DB CRMFP1
0301 03 1016 DB CRMFP2
0302 02 1017 DB CRMFP3
0303 00 1018 DB CRMFP4
1019
1020 *****
1021
1022 LF MOTOR PHASE ENCODE & DECODE: FORWARD (CLOCKWISE)
1023 Forward direction ENCODING:
1024 -----
0308 1025 DRG 308H
1026
0308 04 1028 DB LFMFP1
0309 0C 1029 DB LFMFP2
030A 08 1030 DB LFMFP3
0308 00 1031 DB LFMFP4
1032
1033
1034 ; PG
1035 *****
1036 ; Accel/Decel / Character Handling Test
1037 *****
1038 TEST > Is CR Stpr Mtr At Speed ??
1039 Yes - SetUp do Character Processing
1040 No - Calculate / Store the Acceleration Phase Shift Time (11)
1041 -----
030C FA 1043 ADPTst: Mov A,@StR20 ;get the status byte
030D B211 1044 JB5 PHFSet ;test if Stpr Mtr At Speed
1045 ; jmp to Prnt Head Fire Setup
030F 4400 1046 Jmp ADMmTS ;else Call Accel/Decel Memory Time Store
1047
1048 *****
1049 ; Process Characters for Printing
1050 *****
1051
1052 Character dot matrix - normal char
1053 d = Dot Column
1054 b = Blank Column
1055
1056 b d d d d
1057 (Char Matrix)
1058 0 0 0 0 b
1059 0 0 0 1 d
1060 0 0 1 0 d
1061 0 0 1 1 d
1062 0 1 0 0 d
1063 0 1 0 1 d
1064 -----
0311 2668 1066 PHFSet: JNTO Retrn ;if R=0 not ready to print-exit
0313 326A 1067 JB1 NPrnt ;if Do Not Print stat bit set - EXIT
0315 D21B 1068 JB6 SInkSt ;if bit previously set-skip setting it
0317 FA 1069 Mov A,@StR20 ;get the status byte
0318 4340 1070 ORL A,@SInkSt ;set Prnt Ready SInk bit
031A AA 1071 Mov @StR20,A ;save the status byte
031B D5 1072 SInkSt: SEL RB1
031C FA 1073 Mov A,ChStR1 ;get char status register addr
031D D23A 1074 JB6 PageCk ;test Char Init Done, 1 = Print Dot
1075 ; 0 = Get Char
1076
1077 ; PG
1078 -----
1079 Call for Individual character processing: mid line test if CR/(LF)
1080 -----
1081 GetChr:
1082 test for CR/(LF) if it is the test position in the line
031F F1 1083 CRChCk: Mov A,@CAdrR1 ;get character
0320 03F3 1084 ADD A,#CRCPl ;test for Carriage Return
0322 C626 1085 JZ CrLnCk ; if CR go service it
0324 6437 1086 Jmp AscIC1 ;if not CR Insert Space Char
0326 FA 1087 CRLnCh: Mov A,ChStR1 ;get char status register addr
0327 F22B 1088 JB7 HIFLn ;test Chr Stat Byte Returned
1089 ; if bit 7 = 1 then Print L-to-R
0329 6432 1090 Jmp SpFill ;if R-to-L print skip exit upon CR detect
1091 -----

```

```

1092 ; if L-to-R printing exit the line if less than 1/2 line printed
1093 HlfLn: Mov A,CntR1 ;load char cnt reg w/char buf size
1094 ADD A,#HlFCpl ;add the 2's cpl of 1/2 chr buf size
1095 JC LnPad ;if CB>1/2 full set CR/LF stat bit for pad
1096 ;if CB<1/2 set buffer full stat bit
1097 ;mid-line exit
0330 648A ;
1098 SpFill: ;clear carry flag
1099 LnPad: Clr C ;insert a space char
1100 Mov A,#Space ;char inserted jmp over get char
1101 Jmp ChIsrt ;
-----
0337 F1 1103 AscIC1: Mov A,@AdrR1 ;get character
0338 749B 1104 ChIsrt: Call GChar1 ;call the char lookup/trns table
1105 ;
1106 ; fetch the char dot column data
1107 PageCk: ;page test for balance of char
1108 Mov A,ChStR1 ;get the status byte
1109 JB5 FxJmp1 ;fix jmp over page boundaries
1110 Call ChrPg2 ;Ascii char 50 - 7F Hex
1111 Jmp MtxTst ;jump to Matrix Test
1112 FxJmp1: Call ChrPg1 ;Ascii char 20 - 4F Hex
1113 ; fall thru to print matrix
1114 ; and CB count tests
1115 ;
1116 ; PG
-----
1117 ;
1118 ; test the Char dot column print matrix count and Char buffer count
1119 ;
-----
0343 EB61 1120 MtxTst: DJNZ CDtCR1,PrntDt ;test for dot col or blank
1121 ;status byte in A upon entry here
0345 FA 1122 Mov A,ChStR1 ;get the status byte
0346 53BF 1123 ANL A,#CIntND ;set Char Init NotDone stat Flag
0348 AA 1124 Mov ChStR1,A ;store the status byte
0349 ED58 1125 DJNZ CCntR1,NotLCh ;dec char cnt-jmp if Not Last Char
034B 53FD 1126 ANL A,#NCBFln ;if 0 reset stat bit Not CB Full Line
034D 53FE 1127 ANL A,#CLICBR ;reset CB Reg Init Flag - do Init
034F AA 1128 Mov ChStR1,A ;save the status byte
1129 ;
0350 C5 1130 SEL R80 ;
0351 FA 1131 Mov A,GStR20 ;get Gen Status register addr
0352 53FE 1132 ANL A,#NotRdy ;clear the ready bit
0354 AA 1133 Mov GStR20,A ;store the General Status Byte
0355 D5 1134 SEL R81 ;
0356 6468 1135 Jmp Retrn ;EXIT
1136 ;
1137 ; Test for L-to-R (forward) or R-to-L (reverse) printing
1138 ; (see GChar1 ASCII char code translation procedure)
1139 ;
-----
0358 FA 1140 NotLCh: ;A contains LR/RL bit properly set
0359 F25E 1141 Mov A,ChStR1 ;get char status register addr
1142 JB7 StpCh2 ;test Chr Stat Byte Returned
1143 ; if bit 7 = 1 then Print L-to-R
035B 19 1144 StpCh1: Inc CAdrR1 ;Increment char data memory addr.
035C 6468 1145 Jmp Retrn ;
035E C9 1146 StpCh2: Dec CAdrR1 ;Decrement char data memory addr.
035F 6468 1147 Jmp Retrn ; fall thru to Get Char
1148 ;
1149 ;
-----
1150 ;
1151 ; Re-Entry Exit point for same char:
1152 ; (before returning step the matrix)
1153 ;
-----
1154 ; Test for L-to-R (forward) or R-to-L (reverse) printing
1155 ; (see GChar1 ASCII char code translation procedure)
1156 ;
1157 ;
1158 PrntDt: ;
1159 PrnDir: Mov A,ChStR1 ;get char status byte
0361 FA 1160 PrnDir: Mov A,ChStR1 ;get char status byte
0362 F267 1160 JB7 StpCD2 ;test Chr Stat Byte Returned
1161 ; if bit 7 = 1 then Print L-to-R
0364 CC 1162 StpCD1: Dec CDotR1 ;reverse step char dot col index
1163 ; addr if R-to-L print
0365 6468 1164 Jmp Retrn ;skip over L-to-R print addr inc
0367 1C 1165 StpCD2: INC CDotR1 ;forward step char dot col index
1166 ; addr if L-to-R print
1167 ;EXIT
1168 ;
1169 ; PG

```

```

1170 ; -----
1171 ; Character Print SetUp Exit Procedures
1172 ; -----
1173 ; Clean Standard Exit
1174 ; -----
0368 C5 1175 Retrn: SEL RBO
0369 83 1176 Ret ;EXIT - return w/ Reg Bank 0 Reset
1177
1178 ; Do Not Print exit: set Stpr Mtr drive routine count loop
036A D5 1179 NPrEt: SEL RB1
036B FA 1180 Mov A,ChStR1 ;get the status byte
036C F27C 1181 JB7 SkpNPI ;test print direction
1182 ; Reverse
036E C5 1183 SEL RBO
036F FA 1184 Mov A,GSrR20 ;get the status byte
0370 53BF 1185 ANL A,#ClrSnk ;reset the print ready bit- skips PHFire call
0372 83 1186 Ret
1187 ; Forward
0373 D27C 1188 JB6 SkpNPI ;test for first PHFSet entry reg init
1189 ; Initialize register variables upon first entry
1190 ; end of count clears char to print bit in status byte
0375 4340 1191 ORL A,#ChIntD ;set Char Reg Init Done stat bit
0377 AA 1192 Mov ChStR1,A ;save the status byte
0378 8B07 1193 Mov TmpR10,#07H ;load CR stpr mtr count during NoPrnt
037A 6488 1194 Jmp NPExit
037C EB88 1195 SkpNPI: DJNZ TmpR10,NPExit
037E FA 1196 Mov A,ChStR1 ;get the status byte
037F 53BF 1197 ANL A,#CIntND ;reset - char init not done
0381 AA 1198 Mov ChStR1,A ;save the status byte
0382 C5 1199 SEL RBO
0383 FA 1200 Mov A,GSrR20 ;get Gen Status register addr
0384 53FE 1201 ANL A,#NotRdy ;clear the ready bit
0386 AA 1202 Mov GSrR20,A ;store the General Status Byte
0387 83 1203 NSEtEx: Ret
0388 C5 1204 NPExit: SEL RBO
0389 83 1205 Ret
1206
1207 ; Mid-Line Exit
1208 ; -----
1209 ; EXIT - if CR and not > 1/2 line done during L-to-R print
038A FA 1210 MdLnEx: Mov A,ChStR1 ;get the status byte
038B 53FD 1211 ANL A,#NCBF1n ;if 0 reset stat bit Not CB Full Line
038D 53FE 1212 ANL A,#ClICBR ;reset CB Reg Init Flag - do Init
038F AA 1213 Mov ChStR1,A ;save the status byte
0390 C5 1214 SEL RBO
0391 FA 1215 Mov A,GSrR20 ;get the RBO status byte
0392 4302 1216 ORL A,#DoNotP ;set the Do Not Print Flag(for RAccel)
0394 53BF 1217 ANL A,#ClrSnk ;reset the print ready bit-exit FAccel
0396 AA 1218 Mov GSrR20,A ;save the status byte
0397 83 1219 Ret
1220
1221 ; PG
1222 ; -----
1223 ; Character Dot Generator Math
1224 ; Look-up Table Page Vectoring
1225 ; Print Head Firing
1226 ; -----
1227
0398 AE 1228 GCHAR1: MOV StrCR1,A ;STORE THE CHAR
1229
1230 ; screen for printable char [char +(cp1 20 Hex + 1 = EO Hex)]
0399 03E0 1231 ADD A,#EOEH
039B F69F 1232 JC PrntCh
039D 64C9 1233 Jmp Cnt1Ch ;jmp to control char lookup table
039F 97 1234 PrntCh: Clr C ;clear carry flag
03A0 FE 1235 Mov A,StrCR1 ;get the char again
1236
1237 ; screen for char page [char +(cp1 50 Hex + 1 = BO Hex)]
1238 ; if carry char on page 2 else page 1
03A1 0380 1239 ADD A,#OBOH
03A3 F6AE 1240 JC Page2
1241
1242 ; Page 1 Character -- ASCII 20 Hex thru 4F Hex
1243 ; Correct offset for lookup table page
1244 ; ((char + EO Hex)*5 = Page 1 index addr)
1245 ; -----
03A5 FA 1246 Page1: Mov A,ChStR1 ;get the status byte
03A6 4320 1247 OrL A,#ChDnP1 ;set the page reentry flag bit
03AB AA 1248 Mov ChStR1,A ;store the status byte
03A9 FE 1249 Mov A,StrCR1 ;get the char again
03AA 03E0 1250 ADD A,#EOEH ;set page 1 relative 00 offset
03AC 64BB 1251 Jmp Multi5 ;jump to address math function
1252

```



```

1253 ; Page 2 Character -- ASCII 20 Hex thru 4F Hex
1254 ; Correct offset for lookup table page two's complement
1255 ; of ASCII chr code LookUp Table page base char of 50H plus
1256 ; char * 5 ((char + 80 Hex)*5 = Page 2 index addr)
1257 ; -----
03AE 97 1258 Page2: Clr C ;clear carry flag
03AF FA 1259 Mov A,ChStR1 ;get the status byte
03B0 53DF 1260 AnL A,#ChOnP2 ;set the page reentry flag bit
03B2 AA 1261 Mov ChStR1,A ;store the status byte
03B3 FE 1262 Mov A,StrCR1 ;get the char agian
03B4 03B0 1263 ADD A,#O80H ;set page 2 relative 00 offset
03B6 648B 1264 Jmp Multi5 ;fall thru to address math function
1265 ; -----
1266 ; Compute character page offset dot pattern index address
03B8 AE 1267 MULTIS: Mov StrCR1,A ;store the zero offset char
03B9 E7 1268 RL A ;MULTIPLY CHR BY 5 TO
03BA E7 1269 RL A ; FIND THE ADDRESS
03BB 6E 1270 ADD A,StrCR1 ;ADD 1 TO COMPLETE 5X
03BC AC 1271 MOV CDotR1,A ;SAVE THE ADDRESS
1272 ; -----
1273 ; Test for L-to-R (forward) or R-to-L (reverse) printing
1274 ; (see QChar1 ASCII char code translation procedure)
1275 ; -----
03BD FA 1276 Mov A,ChStR1 ;get char status byte
03BE F2C4 1277 JB7 LRPnn ;test Chr Stat Byte Returned
; if bit 7 = 1 then Print L-to-R
03C0 FC 1279 MOV A,CDotR1 ;get the char index addr
03C1 0304 1280 ADD A,#RLPShf ;add char offset - start at end
1281 ; of char, print it R-to-L
03C3 AC 1282 MOV CDotR1,A ;SAVE THE ADDRESS
1283 ; -----
1284 ; Set the status byte for Character SetUp done
1285 ; -----
03C4 FA 1286 LRPnn: Mov A,ChStR1 ;get the status byte
03C5 4340 1287 ORL A,#ChIntD ;set 1st char col test bit = 0
03C7 AA 1288 Mov ChStR1,A ;store the status byte
03C8 83 1289 Ret ;return w/status byte in A
1290 ; test for non printable characters goes here
03C9 83 1291 CntlCh: Ret
1292 ; -----
1293 ; * * * * *
1294 ; Print Head Fire
1295 ; * * * * *
1296 ; -----
1297 ; Entry point for print head solenoid firing
1298 ; - test for status byte for dot/blank column position
03CA D5 1299 PHFire: SEL RB1
03CB FB 1300 Mov A,CDtCR1 ;set the chr dot column cnt
03CC 96D2 1301 JNZ Fire ;if char cnt not 0 - Fire Head Sol.
; if Chr Dot Cnt 0, reset the
1302 ; char dot column count
03CE 8B06 1303 SetCnt: Mov CDtCR1,#NDtCCt ; char dot column count
03D0 64DB 1304 Jmp Retrn1 ;skip PH Fire
03D2 2340 1305 Fire: MOV A,#PTrgLo ;get the Prnt Head Trigger byte
03D4 3A 1306 OUTL P2,A ;FIRE PRINT HEAD
03D5 23C0 1307 MOV A,#PTrgHi ;get the Prnt Head Trigger byte
03D7 3A 1308 OUTL P2,A ;FIRE PRINT HEAD
03D8 C5 1309 Retrn1: SEL RBO
03D9 83 1310 Ret ;EXIT - return w/ Reg Bank 0 Reset
1311 ; -----
1312 ; PG
1313 ; * * * * *
1314 ; PaperFeed Stpr Mtr Drive
1315 ; * * * * *
1316 ; -----
0400 1317 ORG 400H
1318 ; -----
1319 ; Init psuedo register with LF indirect addr start - subsequent
1320 ; exchanges of the psuedo register will yield correct value
0400 8C04 1321 InitLF: MOV Cntr40,#ILFCnt ; INIT PHASE COUNT REG
0402 8B22 1322 Mov TmpR00,#LPSAdr ;get Ph: Indirect Addr psuedo reg
0404 230B 1323 MOV A,#StLFF ;get LF starting addr
0406 A0 1324 Mov @TmpR00,A ;store LF phase index addr start
; in psuedo register
0407 BE01 1326 Mov LnCntR0,#LineCt ;set line count reg for 1 ln
1327 ; enables exit following LF SM init
0409 841B 1328 Jmp LfDrv1 ;jump over line/form feed amd variable
; line spacing tests & setups
1329 ; -----
1330 ; LineFeed / FormFeed Drive
1331 ; -----

```

```

1332 ; -----
1333 ;
1334 ; load step count constant for standard line spacing
1335 ; -----
1336 ; test for various line/inch spacing would go here
1337 ; (and removal of constant setup below)
040B BC1B 1338 MOV CntR40,#LPiBpB ;init cnt reg for standard line feed
1339 ;
1340 ; LineFeed/FormFeed Test
040D FA 1341 Mov A,GStR20 ;get the status byte
040E 5214 1342 JB2 FmFd ;if linefeed jmp to cnt load
0410 BE01 1343 LnCtLd Mov LnCtRO,#LineCt ;set line count reg for 1 line
0412 841B 1344 Jmp LfDrV1 ;jmp to Start of Drive
0414 FE 1345 FmFd: Mov A,LnCtRO ;get the line count
0415 37 1346 Cpl A ;2's cpl Line Count
0416 0301 1347 Add A,#01
0418 0342 1348 Add A,#PgLnCt ;Add 2's cpl for Paging
1349 ; PgLnCt - LnCt = n Lines to move
1350 ; PgLnCt+(cpl(LnCt)) = n lines to move
041A AE 1351 Mov LnCtRO,A ;set the line count for FF
1352 ;
1353 ; for stablization of unused stpr mtr during CR stpr mtr drive,
1354 ; store the unused stpr mtr current phase bits
041B B821 1355 LFDrv1: Mov TmpR00,#CPSAdr ;get the CR phz storage addr
041D F0 1356 Mov A,@TmpR00 ;get the bytes stored there
041E E3 1357 MovP3 A,@A ;get the phz data byte
041F B820 1358 Mov TmpR00,#LastPh ;load Last Phz psuedo reg to Temp Reg
0421 A0 1359 Mov @TmpR00,A ;store Last Phase bits - indirect
1360 ; exchange/store the phase register index addresses
0422 B822 1361 Mov TmpR00,#LPsAdr ;get Phz Indirect Addr psuedo reg
0424 F0 1362 Mov A,@TmpR00 ;get LF last phase index addr
0425 AB 1363 Mov PhzR30,A ;place last LF phase index addr in Phz Reg
0426 BD9B 1364 MOV TConRO,#LFTMR1 ;Load time constant Reg
1365 ;
1366 ; Select the Stpr Mtr
042B 2306 1367 MOV A,#SLF ;GET CR SM SELECT BITS
042A 3D 1368 MOVD P5,A ;SELECT SM [SCR80]
1369 ; -----
1370 ; LineFeed / FormFeed Drive Loop
1371 ; -----
1372 ;
042B FB 1373 MOV A,PhzR30 ;get the phz reg indirect addr index
042C E3 1374 MovP3 A,@A ;do indirect get of phz bits
1375 ; patch together the CR last and LF next phase bits
042D B820 1376 Mov TmpR00,#LastPh ;load Last Phz psuedo reg to Temp Reg
042F 40 1377 ORL A,@TmpR00 ;patch together CR existing & new LF
1378 ; start timer and step motor
0430 3C 1379 MOVD P4,A ;OUTPUT BITS
1380 ;
1381 ; StrtLF:
0431 FD 1382 STRLFT: MOV A,TConRO ;get time constant from reg
0432 62 1383 MOV T,A ;load the timer
0433 55 1384 STRT T ;START TIMER
1385 ; setup the next phase to output
0434 1B 1386 INC PhzR30 ;STEP PHASE DB ADDRESS
0435 FB 1387 MOV A,PhzR30 ;get the phase index address
0436 523A 1388 JB2 ZRPHL ;test phase
0438 843C 1389 JMP NXTPHL
043A B80B 1390 ZRPHL: MOV PhzR30,#STLFF ;re-init phase register
1391 ;
043C FB 1392 NXTPHL: MOV A,PhzR30 ;get the phz reg indirect addr index
043D E3 1393 MovP3 A,@A ;do indirect get of phz bits
1394 ; patch together the CR last and LF next phase bits
043E B820 1395 Mov TmpR00,#LastPh ;load Last Phz psuedo reg to Temp Reg
0440 40 1396 ORL A,@TmpR00 ;patch together CR existing & new LF
1397 ;
0441 1645 1398 TLoopL: JTF NXPHLF ;jmp on time out to output nxt phz
0443 8441 1399 JMP TLoopL ;loop until timer times out
1400 ;
0445 3C 1401 NXPHLF: MOVD P4,A ;step motor - OUTPUT BITS
0446 EC31 1402 DJNZ CntR40,StrLFT ;test for end of phase count for line
1403 ; prep for next line
1404 ;
1405 ; test for various line/inch spacing would go here
044B BC1B 1406 MOV CntR40,#LPiBpB ;init cnt reg for standard line feed
044A EE31 1407 DJNZ LnCtRO,StrtLF ;test for end of line count
1408 ;
044C FA 1409 Mov A,GStR20 ;Get the status byte
044D 53FB 1410 ANL A,#LineFd ;reset for line feed
044F AA 1411 Mov GStR20,A ;save the status byte
1412 ;
1413 ; store the phase register index addresses
1414 ; Set LineFeed Stpr Mtr Next Phase index address
0450 B822 1415 SetLRN: Mov TmpR00,#LPsAdr ;get Phz Storage Addr psuedo reg

```

```

0452 FB      1416      Mov     A,PhzR30      ;get the phase index address
0453 A0      1417      Mov     @TmpR00,A    ;store LF Next phase index addr
0454 B47B    1418      Call    DlyLng
0456 B490    1419      Call    DeS18M
1420
1421 ;        Check if Char Buffer contains full line (80 char or nChar & CR)
1422 ;        exit otherwise continue to read in characters
1423 ;        Mov     A,GStR20      ;get the stat byte
1424 ;        JBI     ByPas1      ;if Do Not Print Bit Set - EXIT
1425 ;        Call    CBFck
0498 B3      1426      ByPas1: Ret
1427
1428 ; PG
1429 ; *****
1430 ; Minor Software Subroutines
1431 ; *****
1432
0500      1433      ORG     500H
1434
1435 -----
1436 ; System initialization subroutines
1437 ; -----
1438 Default.
1439 ; -----
1440 ; reset/set EOF status flag bit = 0
0500 D5      1441      SEL     RB1
0501 FA      1442      Mov     A,ChStR1    ;get the char status byte
0502 53F7    1443      ANL    A,#ClrEOF   ;clear the EOF flag bit
0504 AA      1444      Mov     ChStR1,A    ;store the char status byte
0505 B823    1445      Mov     TmpR10,#PTAscS ;get the Ascii code tmp store addr
0507 B020    1446      Mov     @TmpR10,#Ascii ;load the tmp stor reg w/ascii start
0509 C5      1447      SEL     RBO
1448
1449 ; reset/set Dk-to-Print status flag bit = 0
050A FA      1450      Mov     A,GStR20    ;get the status byte
050B 53DF    1451      ANL    A,#DkPrnt  ;reset print flag - Dk Print
050D AA      1452      Mov     GStR20,A    ;save the status byte
050E B3      1453      RET
1454 InitAl:
1455 AllDff:
1456 ; -----
1457 ; CLEAR all outputs
050F C5      1458      SEL     RBO
0510 230F    1459      MOVD  A,#OFH      ;FORCE PORT HI - R/ OF 555
0512 3E      1460      MOVD  P6,A
0513 23FF    1461      MOVD  A,#OFFH     ;TURN ALL PRNT SOL's OFF
0515 39      1462      OUTL  P1,A
0516 23C0    1463      MOVD  A,#PTRGHI   ;print head fire trigger inactive
0518 3A      1464      OUTL  P2,A
0519 BA03    1465      ORL   P2,#03      ;set comm hdk to ACK hi/Busy hi
051B BA00    1466      Mov     GStR20,#00H ;clear the status registers
051D D5      1467      SEL     RB1
051E BA00    1468      Mov     ChStR1,#00H
0520 C5      1469      SEL     RBO
0521 B3      1470      RET      ;RETURN TO INIT ROUTINE
1471
1472 ; PG
1473 ; *****
1474 ; Home Carriage / Print Head Assembly
1475 ; *****
1476
0522 FA      1477      CRHome: Mov     A,GStR20    ;get the status byte
0523 4302    1478      ORL   A,#DoNotP   ;set the do not print flag
0525 AA      1479      Mov     GStR20,A    ;save the status byte
0526 362A    1480      JTO   RtoL
1481 ;        ; drive accordingly
052B 3402    1482      Call    FAccel    ;drive CR Stpr Mtr
052A 3422    1483      RtoL: Call    RAccel    ;find the logical left home CR position
1484 ;        ; delay a long time before continuing
052C B474    1485      Call    DlyVLg
052E B3      1486      Ret
1487
1488 ; *****
1489 ; Clear Data Memory
1490 ; *****
1491
1492 ; At PowerUp or Reset, following CR & LF Stpr Mtr Init, this
1493 ; procedure clears data memory above RBO, Stack and RB1.
052F B87F    1494      ClrDM: MOV     RO,#DMTop ;GET BUFFER START LOCATION [HEX]
0531 B95D    1495      MOV     R1,#DMSIZE
0533 B000    1496      ClrDMI: MOV     @RO,#00H ;ZERO MEMORY LOCATION

```

```

0535 C8      1497      DEC      RO
0536 E933   1498      DJNZ     R1,C1rDM1      ;dec buffer, loop if not zero[end]
0538 B3     1499      RET
1500
1501 ; PG
1502 ; *****
1503 ; Character Print TEST
1504 ; *****
1505
1506 PrnTst:
1507 ; TEST --- load the char buffer with successive increments of
1508 ; the ascii code start. test for end of ascii
1509 ; printable chars and reinit the char stream loaded.
1510
0539 B97F   1511 CTInt: Mov      CAdrR1,#FCBfSt ;load char reg w/char bufr strt
053B BD50   1512      Mov      CCntR1,#ChBfSz ;load char cnt reg w/char bufr size
1513      ChTst:      ;Test char buffer fill with ASCII Char Code
1514      Mov      A,opnr71 ;get the ascii char
053E A1     1515      @CAdrR1,A ;load data memory w/Char
053F C9     1516      DEC      CAdrR1 ;Decrement dat memory location
0540 1F     1517      INC      opnr71 ;Increment Ascii char number
0541 03B2   1518      ADD      A,#PAsEnd ;test for ascii code end
0543 9647   1519      JNZ     ChrTGo ;if not end jmp over code restart
0545 BF20   1520      Mov      Opnr71,#Ascii
0547 ED3D   1521 ChrTGo: DJNZ     CCntR1,ChTst ;dec buffer, loop if not zero[end]
0549 C5     1522      SEL     RBO
054A B3     1523      RET
1524 ; ELSE RETURN TO INIT ROUTINE
1525 ; PG
1526 ; *****
1527 ; CR Stpr Mtr Power On Initialization and
1528 ; *****
1529 ; This routine drives the CR or LF stpr mtr for four phase
1530 ; shifts for initialization.
1531 INITCR:
054B BC04   1532      MOV      CntR40,#PhCnt1 ;POWER ON INIT STPR MTR
054D 2308   1533      MOV      A,#SCRBO ;GET CR SM SELECT BITS
054F 3D     1534      MOVVD   P5,A ;SELECT SM [SCRBO]
0550 BDCC   1535      MOV      TConR0,#IntTm2 ;Load time constant Reg
0552 BB00   1536      MOV      PhzR30,#FStCRP ;zero SM phase reg - forward
0554 FB     1537      MOV      A,PhzR30 ;get phase index register byte
0555 E3     1538      MovVP3  A,BA ;load indexed phase shift byte
0556 3C     1539      MOVVD   P4,A ;OUTPUT BITS
0557 FD     1540 STRTTR: MOV      A,TConR0 ;GET TIMER CONSTANT
0558 62     1541      MOV      T,A
0559 55     1542      STRT   T ;START TIMER
055A 1B     1543      INC      PhzR30 ;step phase index register
055B FB     1544      MOV      A,PhzR30 ;CHECK THE PHASE COUNT REG
055C 5260   1545      JB2     ZroRg2
055E A462   1546      JMP      NxtPhR
0560 BB00   1547 ZroRg2: MOV      PhzR30,#FStCRP ;zero SM phase reg - forward
1548      NxtPhR:
0562 FB     1549      MOV      A,PhzR30 ;get phase index register byte
0563 E3     1550      MovVP3  A,BA ;load indexed phase shift byte
0564 1669   1551 TLoopR: JTF     NXPHR1 ;JMP ON TIME OUT TO NEXT PH
0566 A464   1552      JMP      TLoopR ;LOOP UNTIL TIME OUT
0568 3C     1553      MOVVD   P4,A ;OUTPUT BITS
0569 EC57   1554      NXPHR1: DJNZ     CntR40,STRTTR
1555 ;
1556 ; store the last phase register index addresses
056B 8B21   1557      Mov      TmpR00,#CPSAdr ;get Phz Storage Addr psuedo reg
056D FB     1558      Mov      A,PhzR30 ;place last CR phase index addr in Phz Reg
056E A0     1559      Mov      @TmpR00,A
056F B47B   1560      Call   DlyLng ; store CR last phase index addr
0571 B490   1561      Call   DeS1SM
0573 B3     1562      RET
1563
1564 ; PG
1565 ; -----
1566 ; Time Delay Subroutines
1567 ; -----
1568
1569 ; Very Long
0574 8B7F   1570 DlyVLg: MOV      TmpR00,#7FH ;LOAD DELAY COUNT IN REG.
0576 A47E   1571      Jmp     DlyST
1572
1573 ; Long
0578 8B80   1574 DlyLNg: MOV      TmpR00,#DlyCL ;LOAD DELAY COUNT IN REG.
057A A47E   1575      Jmp     DlyST
1576

```

```

057C 8B30      1577 ,      Not So Long - Short
1578 DlySht: MOV      TmpROO,#DlyCS ;LOAD DELAY COUNT IN REG.
1579
057E 23CC      1580 ,      Start Delay
1581 DlyST:  MOV      A,#DlyTim ;GET MAX TIMER DELAY
0580 62        1582 NxtTld  MOV      T,A ;LOAD TIMER
0581 55        1583 STRT      T ;START TIMER
1584
0582 168D      1585 DlyLop: JTF      DlyTO ;LOOP
1586
0584 D5        1587 ,      Char buffer fill during time loop:
0585 FA        1588 SEL      RB1
0586 928A      1589 Mov      A,ChStR1 ;get the character stat reg byte
1591          1590 JB4      SkpCI ; test for normal char input
0588 1469      1591          ; or skip if char prnt test
058A C3        1592 Call     IBFSrv ;service the char buffer fill
0588 A482      1593 SkpCI: SEL     RBO
058D E880      1594 JMP      DlyLOP
058F 83        1595 DlyTO:  DUNZ   TmpROO,NxtTld ;dec delay count & test for exit
1596          RET
1597 ;-----
1598 ;      Stpr Mtr Deselect
1599 ;-----
1600          Stepper Motor DeSelect Routine
1601 DESLSM ;DESELECT LF/CR SM
0590 230E      1602 SMEROR MOV     A,#SMOFF ;GET LF/CR SM DE-SELECT BITS
0592 3D        1603 MOVD    P5,A ;DE-SELECT CR SM
0593 83        1604 RET
1605
1606 $INCLUDE(:(F1:CHRTBL.OV1)
=1607
=1608 ; *****
=1609 ;      Character Dot Generator Look-up Table Page 1
=1610 ; *****
=1611
=1612
=1613 ;      Character Table Page 1, contains
=1614
=1615 ;      20H -----> 4FH
=1616
=1617 ;      " (sp)!"##%&'()*+,./0123456789;<=>?@ABCDEFHIJKLM "
=1618
=1619 ;-----
0600          ORG      600H
=1620
=1621
=1622
=1623 ;-----
=1624 ;      Page 1 -- Character Dot Pattern Fetch
=1625 ;      <<< actual assembled character table code not listed >>>
=1626 ;-----
=1627 *NoList
=1676 *List
=1677 ;      Listing below is for reference only, actual code is not listed
=1678 ;      at assembly time.
=1679
=1680 ;-----
=1681 ; asc20: DB 7FH, 7FH, 7FH, 7FH, 7FH ;SPACE
=1682 ; asc21: DB 7FH, 7FH, 20H, 7FH, 7FH ;!
=1683 ; asc22: DB 7FH, 7FH, 78H, 7FH, 78H ;"
=1684 ; asc23: DB 68H, 00H, 68H, 00H, 68H ;#
=1685 ; asc24: DB 58H, 55H, 00H, 55H, 6DH ;$
=1686 ; asc25: DB 5CH, 6CH, 77H, 18H, 1DH ;%
=1687 ; asc26: DB 19H, 26H, 26H, 59H, 2FH ;&
=1688 ; asc27: DB 7FH, 7FH, 7CH, 7FH, 7FH ;'
=1689 ; asc28: DB 63H, 5DH, 3EH, 7FH, 7FH ;(
=1690 ; asc29: DB 7FH, 7FH, 3EH, 5DH, 63H ;)
=1691 ; asc2A: DB 5DH, 68H, 00H, 68H, 5DH ;*
=1692 ; asc2B: DB 77H, 77H, 41H, 77H, 77H ;+
=1693 ; asc2C: DB 7FH, 3FH, 4FH, 7FH, 7FH ;,
=1694 ; asc2D: DB 77H, 77H, 77H, 77H, 77H ;-
=1695 ; asc2E: DB 7FH, 1FH, 1FH, 7FH, 7FH ;.
=1696 ; asc2F: DB 5FH, 6FH, 77H, 78H, 7DH ;/
=1697 ; asc30: DB 41H, 2EH, 36H, 3AH, 41H ;0
=1698 ; asc31: DB 7FH, 3DH, 00H, 3FH, 7FH ;1
=1699 ; asc32: DB 3DH, 1EH, 2EH, 36H, 39H ;2
=1700 ; asc33: DB 5DH, 3EH, 36H, 36H, 49H ;3
=1701 ; asc34: DB 67H, 68H, 6DH, 00H, 6FH ;4
=1702 ; asc35: DB 58H, 3AH, 3AH, 3AH, 46H ;5
=1703 ; asc36: DB 43H, 35H, 36H, 36H, 4EH ;6
=1704 ; asc37: DB 7EH, 0EH, 76H, 7AH, 7CH ;7
=1705 ; asc38: DB 49H, 36H, 36H, 36H, 49H ;8

```

```

=1706 ; asc39: DB 39H, 36H, 36H, 56H, 61H ;9
=1707 ; asc3A: DB 7FH, 7FH, 68H, 7FH, 7FH ;:
=1708 ; asc3B: DB 7FH, 3FH, 48H, 7FH, 7FH ;:
=1709 ; asc3C: DB 77H, 68H, 5DH, 3EH, 7FH ;<
=1710 ; asc3D: DB 68H, 68H, 68H, 68H, 68H ;=
=1711 ; asc3E: DB 7FH, 3EH, 5DH, 68H, 77H ;>
=1712 ; asc3F: DB 79H, 7EH, 26H, 7AH, 7DH ;?
=1713 ; asc40: DB 41H, 3EH, 22H, 36H, 71H ;@
=1714 ; asc41: DB 03H, 6DH, 6EH, 6DH, 03H ;A
=1715 ; asc42: DB 00H, 36H, 36H, 36H, 49H ;B
=1716 ; asc43: DB 41H, 3EH, 3EH, 3EH, 5DH ;C
=1717 ; asc44: DB 00H, 3EH, 3EH, 5DH, 63H ;D
=1718 ; asc45: DB 00H, 36H, 36H, 36H, 36H ;E
=1719 ; asc46: DB 00H, 76H, 76H, 76H, 76H ;F
=1720 ; asc47: DB 41H, 3EH, 3EH, 2EH, 0DH ;G
=1721 ; asc48: DB 00H, 77H, 77H, 77H, 00H ;H
=1722 ; asc49: DB 7FH, 3EH, 00H, 3EH, 7FH ;I
=1723 ; asc4A: DB 5FH, 3FH, 3FH, 3FH, 40H ;J
=1724 ; asc4B: DB 00H, 77H, 68H, 5DH, 3EH ;K
=1725 ; asc4C: DB 00H, 3FH, 3FH, 3FH, 3FH ;L
=1726 ; asc4D: DB 00H, 7DH, 73H, 7DH, 00H ;M
=1727 ; asc4E: DB 0aah, 0d#H, 0e#H, 0f7H, 0aah ;test
=1728 ; asc4F: DB 55H, 0d#H, 0e#H, 0f7H, 55H ;test
=1729 ; asc4E: DB 00H, 78H, 77H, 6FH, 00H ;N
=1730 ; asc4F: DB 41H, 3EH, 3EH, 3EH, 41H ;O
-----
=1732 ; End Page 1 -- Character Dot Pattern Fetch
=1733 ;
=1734 ;
-----
=1735 ; Character Dot Pattern Fetch
-----
=1736 ;
=1737 ;
06F0 FC =1738 ChrPgl: MOV A,CDotR1 ;get char index address offset
06F1 A3 =1739 MOVVP A,@A ;get column dot pattern byte
=1740 ;
=1741 ; this bit fix necessary to not underline each character
=1742 ; this saves fixing each bit in the look up table
=1743 ;
06F2 43B0 =1744 ORL A,#80H ;char bit fix
06F4 39 =1745 OutL P1,A ;output the dot pattern
06F5 B3 =1746 RET ;exit with byte in acc
=1747 ;
-----
=1748 ; END Page 1 -- Character Dot Pattern Fetch
=1749 ;
=1750 ;
=1751 ;
=1752 ;
=1753 ; PAGE 2 -- Character Dot Generator Look-Up Table
-----
=1754 ;
=1755 ;
=1756 ; Character Table Page 2, contains
=1757 ;
=1758 ; 50H -----> 7EH
=1759 ;
=1760 ; " NOPQRSTUVWXYZ[\]^_(?)abcdefghijklmnopqrstuvwxyz{!}~ "
=1761 ;
=1762 ;
-----
0700 =1763 ORG 700H
=1764 ;
=1765 ;
=1766 ;
-----
=1767 ; Page 2 -- Character Dot Pattern Fetch
=1768 ; <<< Actual assembled character table code not listed >>>
=1769 ;
=1770 @NoLIST
=1818 @List
=1819 ; Listing below is for reference only, actual code is not listed
=1820 ; at assembly time
=1821 ;
=1822 ;
-----
=1823 ; asc50: DB 00H, 76H, 76H, 76H, 79H ;P
=1824 ; asc51: DB 41H, 3EH, 2EH, 5EH, 21# ;Q
=1825 ; asc52: DB 00H, 76H, 66H, 56H, 39H ;R
=1826 ; asc53: DB 59H, 36H, 36H, 36H, 4DH ;S
=1827 ; asc54: DB 7EH, 7EH, 00H, 7EH, 7EH ;T
=1828 ; asc55: DB 40H, 3FH, 3FH, 3FH, 40H ;U
=1829 ; asc56: DB 60H, 5FH, 6FH, 5FH, 60H ;V
=1830 ; asc57: DB 00H, 5FH, 6FH, 5FH, 00H ;W
=1831 ; asc58: DB 1CH, 6BH, 77H, 6BH, 1CH ;X
=1832 ; asc59: DB 7CH, 78H, 07H, 78H, 7CH ;Y

```

```

-1833 ; asc5A      DB      1EH, 2EH, 36H, 3AH, 3CH      ;Z
-1834 ; asc5B      DB      00H, 3EH, 3EH, 3EH, 7FH      ;[
-1835 ; asc5C      DB      7DH, 7BH, 77H, 6FH, 5FH      ;\
-1836 ; asc5D      DB      7FH, 3EH, 3EH, 3EH, 00H      ;]
-1837 ; asc5E      DB      6FH, 77H, 7BH, 77H, 6FH      ;^
-1838 ; asc5F      DB      3FH, 3FH, 3FH, 3FH, 3FH      ;_
-1839 ; asc60      DB      7DH, 7BH, 77H, OFFH, OFFH      ;\
-1840 ; asc61      DB      0DFH, 0ABH, 0ABH, 0ABH, 0B7H      ;a
-1841 ; asc62      DB      0B0H, 0B7H, 0B7H, 0B7H, 0CFH      ;b
-1842 ; asc63      DB      0C7H, 0BBH, 0BBH, 0BBH, 0BBH      ;c
-1843 ; asc64      DB      0CFH, 0B7H, 0B7H, 0B7H, 0B0H      ;d
-1844 ; asc65      DB      0C7H, 0ABH, 0ABH, 0ABH, 0B7H      ;e
-1845 ; asc66      DB      0F7H, 0B1H, 0F6H, 0FEH, 0FDH      ;f
-1846 ; asc67      DB      0F7H, 0ABH, 0ABH, 0ABH, 0C3H      ;g
-1847 ; asc68      DB      0B0H, 0F7H, 0FBH, 0FBH, 0B7H      ;h
-1848 ; asc69      DB      0FFH, 0BFH, 0BBH, 0BFH, 0FFH      ;i
-1849 ; asc6A      DB      0DFH, 0BFH, 0BBH, 0C2H, 0FFH      ;j
-1850 ; asc6B      DB      0FFH, 0B0H, 0EFH, 0B7H, 0BBH      ;k
-1851 ; asc6C      DB      0FFH, 0BEH, 0B0H, 0BFH, 0FFH      ;l
-1852 ; asc6D      DB      0B7H, 0FBH, 0E7H, 0FBH, 0B7H      ;m
-1853 ; asc6E      DB      0B3H, 0F7H, 0FBH, 0FBH, 0B7H      ;n
-1854 ; asc6F      DB      0C7H, 0BBH, 0BBH, 0BBH, 0C7H      ;o
-1855 ; asc70      DB      0B4H, 0EBH, 0EBH, 0EBH, 0F7H      ;p
-1856 ; asc71      DB      0F7H, 0EBH, 0EBH, 0EBH, 0B4H      ;q
-1857 ; asc72      DB      0FFH, 0B3H, 0F7H, 0FBH, 0FBH      ;r
-1858 ; asc73      DB      0B7H, 0ABH, 0ABH, 0ABH, 0DBH      ;s
-1859 ; asc74      DB      0FBH, 0C1H, 0BBH, 0DFH, 0FFH      ;t
-1860 ; asc75      DB      0C3H, 0BFH, 0BFH, 0BFH, 0C3H      ;u
-1861 ; asc76      DB      0E3H, 0DFH, 0BFH, 0DFH, 0E3H      ;v
-1862 ; asc77      DB      0C3H, 0BFH, 0CFH, 0BFH, 0C3H      ;w
-1863 ; asc78      DB      0BBH, 0C7H, 0EFH, 0C7H, 0BBH      ;x
-1864 ; asc79      DB      0FFH, 0B3H, 0AFH, 0AFH, 0C3H      ;y
-1865 ; asc7A      DB      0BBH, 09BH, 0ABH, 0B3H, 0BBH      ;z
-1866 ; ASC7B      DB      07FH, 077H; 049H, 03EH, 03EH      ;{
-1867 ; ASC7C      DB      0FFH, 0FFH, 0BBH, 0FFH, 0FFH      ;|
-1868 ; ASC7D      DB      03EH, 03EH, 009H, 077H, 07FH      ;}
-1869 ; ASC7E      DB      067H, 07BH, 067H, 05FH, 067H      ;~
-1870
-1871

```

-1872 ; Character Dot Pattern Fetch

```

-1873 ;
-1874
07EB FC -1875 ChrPg2: MOV      A,CDotR1      ;get char index address offset
07EC A3 -1876         MOVPP   A,@A          ;get column dot pattern byte
-1877
-1878 ;         this bit fix necessary to not underline each character
-1879 ;         this saves fixing each bit in the look up table
-1880
-1881         ORL      A,#0H          ;char bit fix
07ED 43B0 -1882         OutL    P1,A          ;output the dot pattern
07EF 39 -1883         RET          ;exit with byte in acc
07F0 83 -1884
-1885
-1886
-1887 ; *****
-1888 ; Program End
-1889 ; *****
-1890
-1891         END

```

ASSEMBLY COMPLETE, NO ERRORS

APPENDIX B

SOFTWARE PRINTER ENHANCEMENTS

This section describes several software enhancements which could be implemented as additions to the software developed for this Application Note. Space is available for most of the items described. Approximately 5 bytes of Data Memory would be required to implement most of the features. Two bytes would be used for status flags, and two bytes for temporary data or count storage. It is possible to use less than five bytes, but this would require the duplicate use of some flags, or other Data Memory storage, which will significantly complicate the software coding and debug tasks.

Special Characters or Symbols

Dot matrix printing lends itself well to the creation of custom characters and symbols. There are two aspects to implementing special characters. First, a character look-up table, and second, additional software for decoding and processing the special characters or symbols. Special characters might be scientific notation, mathematical symbols, unique language characters, or block and line graphics characters.

The character look-up table could be an additional page of Program Memory dedicated to the special characters, or replace part, or all, of the existing look-up tables. If an additional look-up table is used, a third page test would be needed at the beginning of the Character Translation subroutine. There is fundamentally no difference between the processing of special characters and standard ASCII printable characters. If the characters require the same 5 x 7 dot matrix, the balance of the software would remain the same. If, however, the special characters require a different matrix, or the manipulation of the matrix, the software becomes more complex.

In general, the major software modification required to implement special characters is the size of the dot matrix printed or the dot matrix configuration used. In the case of scientific characters, it would often be necessary to shift the 5 x 7 matrix pattern within the available 9 x 9 matrix. Block or line graphics characters, on-the-other-hand, would require using the entire 9 x 9 print head matrix and printing during normally blank dot columns. This would require suspending the blank column blanking mechanism implemented in this Application Note. This would be the most complex aspect of implementing special characters. It would possibly change the number of required instructions, and thus the timing between PTS detection and print head solenoid trig-

ger firing. This could cause the dot columns to be misaligned within a printed line and between lines.

In the case of a matrix change, two approaches are possible: dynamically changing the matrix, in line, as standard ASCII characters are being printed, or isolating the special characters to a separate processing flow where special characters are handled as a unique and complete line of characters only. A discussion of in line matrix changes for special characters is beyond the scope of this Appendix. It is sufficient to say that the changes would require the conditions setting the EOLN flag, character count, and dot column count software be modified during character processing and printing.

Lower Case Descenders

The general principle of implementing lower case descenders is to shift the 5 x 7 character dot matrix within the available 9 x 9 print head solenoid matrix. Implementing lower case descenders requires two software modifications and the creation of status flag for the purpose. First, the detection of characters needing descenders and setting a dedicated status flag during the character code to dot pattern translation subroutine. Second, the character dot column data output to the print head solenoids must be shifted for each dot column of the character. At the end of the character, the flag would be reset.

Inline Control Codes

Inline control codes are two to three character sequences, which indicate special hardware conditions or software flow control and branching. The first character indicates that the control code sequence is beginning and is typically an ASCII Escape Character (ESC), 1BH. Termination of the inline code sequence would be indicated by a default number of code sequence characters. This would decrease the buffer size available for characters. Full 80 character line buffering would require loading the Character Buffer with a received character as a character is removed from it and processed.

The Inline Control Code test would be performed in two places: in the Character Buffer Fill subroutine and in the Character Processing (translation) subroutine. The test would be performed in the same manner that a Carriage Return (CR) character code test is implemented. Examples are horizontal tabs and expanded or con-

denser character fonts. In the case of horizontal tabs, 20H (Space Character) would have to be placed in the Character Buffer for inline processing during character processing and printing. Unless fixed position tabs are used, a minimum of a nibble of Data Memory would be required to maintain a "spaces-to-tab" count. Fixed tab positions could be set via another inline control code, by default of the print software, or through the use of external hardware switch settings. The control code method of setting the tab positions is the most desirable, but the most complex to implement.

Different Character Formats

Condensed and enlarged characters are variations in either the number of dots and/or the space used to print them. Thus, each character is a variation of the stepper motor and/or print head solenoid trigger timings. It is possible to print each in bold face by printing each dot twice per dot column position. This would require little software modification, but would require a status flag. Again, care must be used to ensure that the delay in retriggering the solenoids is precisely the same for each type of event. Without this precise timing the dot column alignment will not be accurate. The software modifications needed to implement enlarged or condensed characters is essentially the same. The carriage and print head solenoid firing software flow is the same, but the timing for each changes. For condensed characters, the step Time Constant is doubled to approximately 4.08 ms, and the solenoids are fired four times within each step time. The step rate actually becomes a multiple of the solenoid firing time, and a counter incrementing once for each solenoid firing would be needed. At the count of four, the carriage stepper motor is stepped and the counter reset.

In the case of condensed characters, PTS does not play the same roll as in standard or enlarged character printing. PTS is not used to indicate the optimum print head solenoid firing time. Solenoid firing is purely a time function for condensed characters. PTS would only be used for Failsafe protection.

Enlarged characters would require the solenoids be fired twice per dot column data, in two sequential dot columns, at the same rate as standard characters. The character dot column data and dot column count would not be incremented at each output but at every other output. A flag could be used for this purpose.

When printing either condensed or enlarged characters, the maximum character count would have to compensate for the increased or decreased characters per line count. When printing enlarged characters, the maximum characters per line would be 40.

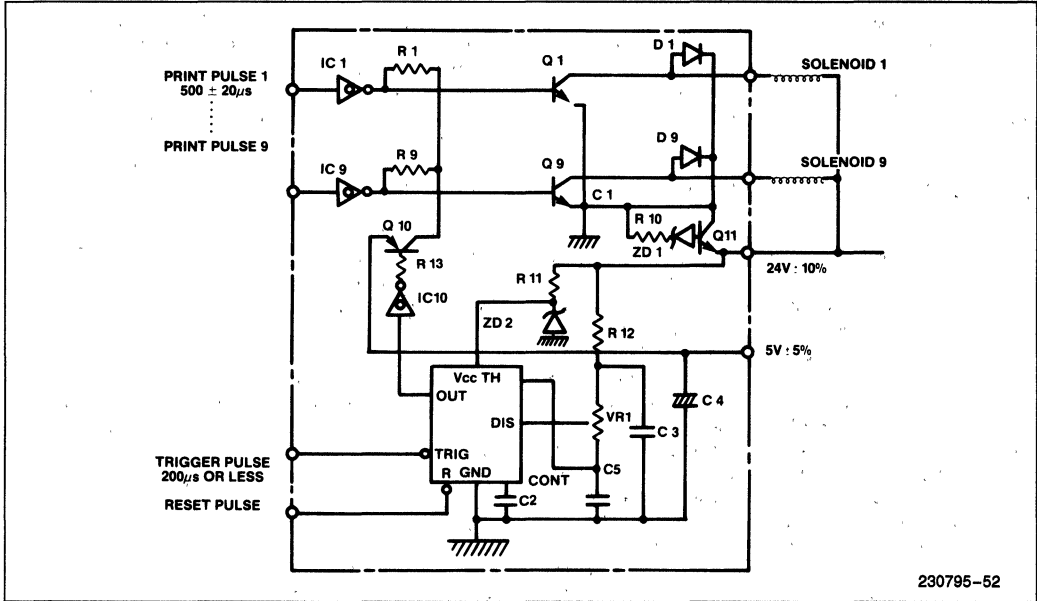
The Character Buffer could hold two complete lines of characters. But, condensed characters presents a quite different situation. The available character per line increases to 132, well beyond the 80 character Character Buffer size. The solution is to re-initialize the Character Buffer Size Count register count during condensed character processing. This will effectively inhibit the carriage stepper motor drive EOLN detection.

Two status flags would be required; one for standard or enlarged characters, and the second for condensed characters. A third status flag would be required to implement bold face printing. Activating one of the alternate character fonts could be either through the use of external status switches or through inline control code sequences, as detailed above. Note, that if the alternate character fonts are implemented in such a way that format changing is to occur dynamically during any single line being printed, the same control code problems described above also apply. In addition, the effect on the timing and dot column alignment must also be investigated.

Variable Line Spacing

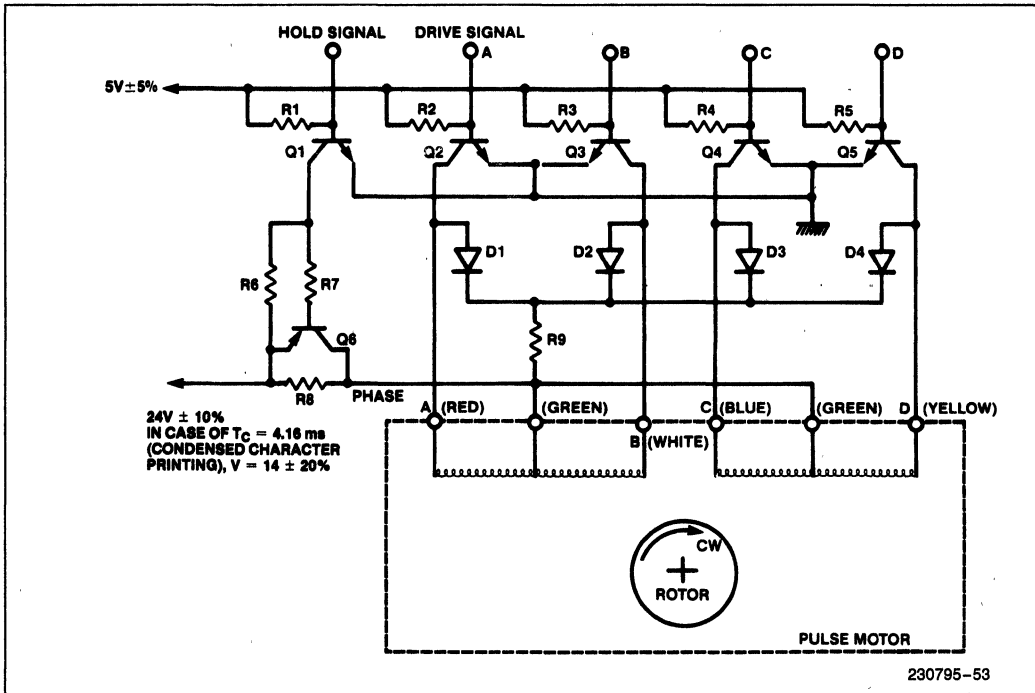
Variable line spacing is another feature which could be implemented either through the use of external status switches or inline control codes. The line spacing is a function of the number of steps the stepper motor rotates for a given line. Figure 15, Paper Feed Stepper Motor Predetermined Time Constants, in the Background section, lists the Time Constants required for three different line spacings; 6, 8, and 10 lines per inch. At the beginning of the Paper Feed Stepper Motor Drive subroutine, the default line step count is loaded. The software required is a conditional load for the line spacing, indicated by a status flag set in the External Status Switch Check subroutine or the Character Buffer Fill subroutine. Implementing the three different line spacings would require two additional status flags.

APPENDIX C PRINTER MECHANISM DRIVE CIRCUIT



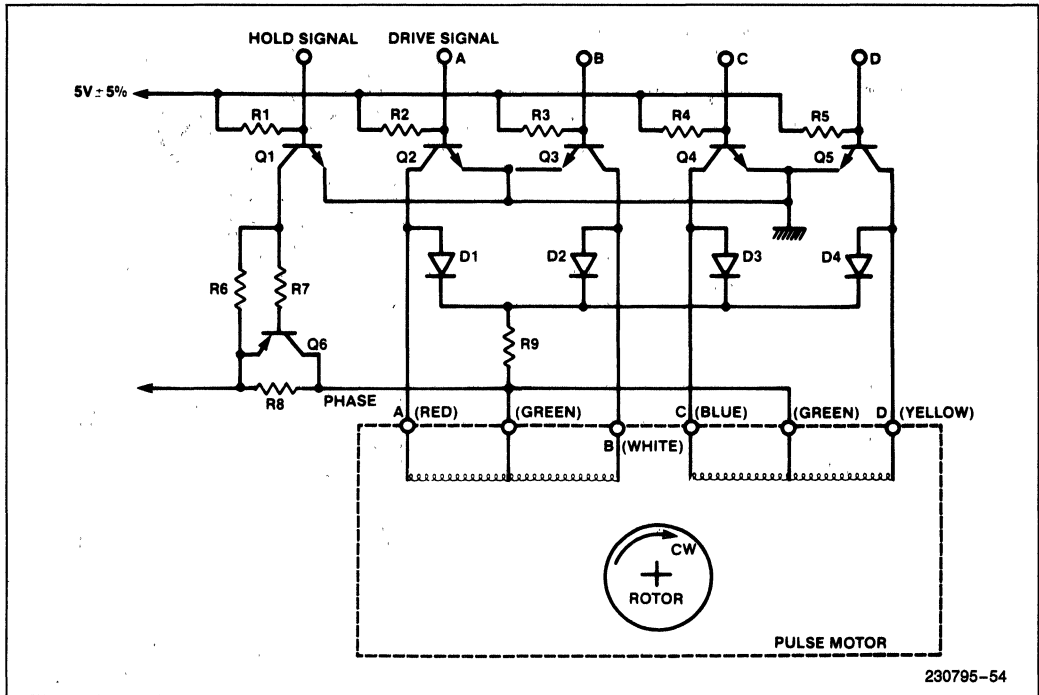
C1. Recommended Solenoid Drive Circuit

Parts No.		Type	Maker
IC1 ~ IC10		SN7406	TI
IC11		μ A555	Fairchild
D1 ~ D9	Diode	S5277B	Toshiba
Q1 ~ 9	Transistor	2SD986	NEC
Q10	Transistor	2SA1015	Toshiba
Q11	Transistor	2SD633	Toshiba
R1 ~ R9	Resistor	1.2 k Ω $\frac{1}{4}$	
R10	Resistor	22 Ω $\frac{1}{4}$	
R11	Resistor	580 Ω 2	
R12	Resistor	15 k Ω $\frac{1}{4}$ Carbon fil =	
R13	Resistor	1.2 k Ω $\frac{1}{4}$	
VR1	Variable Resistor	20 k Ω $\frac{1}{4}$	
C1	Capacitor	1 μ F 100V	
C2	Capacitor	0.01 μ F	
C3	Capacitor	0.001 μ F	
C4	Capacitor	10 μ F 16V	
C5	Capacitor	0.1 μ F fil =	
ZD1	Zenor Diode	HZ24	Hitachi
ZD2	Zenor Diode	HZ5C1	Hitachi



C2. Recommended Carriage Motor Drive Circuit

Parts No.		Type	Maker	Qty
R1	Resistor	1 kΩ ± 10% ¼		1
R2-R5	Resistor	220Ω ± 10% ¼		4
R6	Resistor	10 kΩ ± 10% ¼		1
R7	Resistor	470Ω ± 10% 3		1
R8	Resistor	130Ω ± 10% 7		1
R9	Resistor	330Ω ± 10% 3		1
Q1	Transistor	2SC1815	Toshiba	1
Q2 ~ Q5	Transistor	2SD526-Y	Toshiba	4
Q6	Transistor	2SB669	Matsushita	1
D1 ~ D4	Diode	1S954	NEC	4



C3. Recommended Paper Feed Motor Drive Circuit

Parts No.		Type	Maker	Qty
R1	Resistor	1 kΩ ± 10% 1/4		1
R2-R5	Resistor	220Ω ± 10% 1/4		4
R6	Resistor	10 kΩ ± 10% 1/4		1
R7	Resistor	470Ω ± 10% 3		1
R8	Resistor	130Ω ± 10% 7		1
R9	Resistor	330Ω ± 10% 3		1
Q1	Transistor	2SC1815	Toshiba	1
Q2 ~ Q5	Transistor	2SD526-Y	Toshiba	4
Q6	Transistor	2SB669	Matsushita	1
D1 ~ D4	Diode	1S954	NEC	4



**APPLICATION
NOTE**

AP-90

November 1986

**An 8741AH/8041A Digital
Cassette Controller**

**JOHN BEASTON, JIM KAHN
PERIPHERAL APPLICATIONS**

Order Number: 231314-002

INTRODUCTION

The microcomputer system designer requiring a low-cost, non-volatile storage medium has a difficult choice. His options have been either relatively expensive, as with floppy discs and bubble memories, or non-transportable, like battery backed-up RAMs. The full-size digital cassette option was open but many times it was too expensive for the application. Filling this void of low-cost storage is the recently developed digital mini-cassette. These mini-cassettes are similar to, but not compatible with, dictation cassettes. The mini-cassette transports are inexpensive (well under \$100 in quantity), small (less than 25 cu. in.), low-power (one watt), and their storage capacity is a respectable 200K bytes of unformatted data on a 100-foot tape. These characteristics make the mini-cassette perfect for applications ranging from remote datalogging to program storage for hobbyist systems.

The only problem associated with mini-cassette drives is controlling them. While these drives are relatively easy to interface to a microcomputer system, via a parallel I/O port, they can quickly overburden a CPU if other concurrent or critical real-time I/O is required. The cleanest and probably the least expensive solution

in terms of development cost is to use a dedicated single-chip controller. However, a quick search through the literature turns up no controllers compatible with these new transports. What to do? Enter the UPI-41AH family of Universal Peripheral Interfaces.

The UPI-41AH family is a group of two user-programmable slave microcomputers plus a companion I/O expander. The 8741AH is the "flag-chip" of the line. It is a complete microcomputer with 1024 bytes of EPROM program memory, 64 bytes of RAM data memory, 16 individually programmable I/O lines, an 8-bit event counter and timer, and a complete slave peripheral interface with two interrupts and Direct Memory Access (DMA) control. The 8041A is the masked ROM, pin compatible version of the 8741AH. Figure 2 shows a block diagram common to both parts. The 8243 I/O port expander completes the family. Each 8243 provides 16 programmable I/O lines.

Using the UPI concept, the designer can develop a custom peripheral control processor for his particular I/O problem. The designer simply develops his peripheral control algorithm using the UPI-41AH assembly language and programs the EPROM of the 8741AH. Voila!

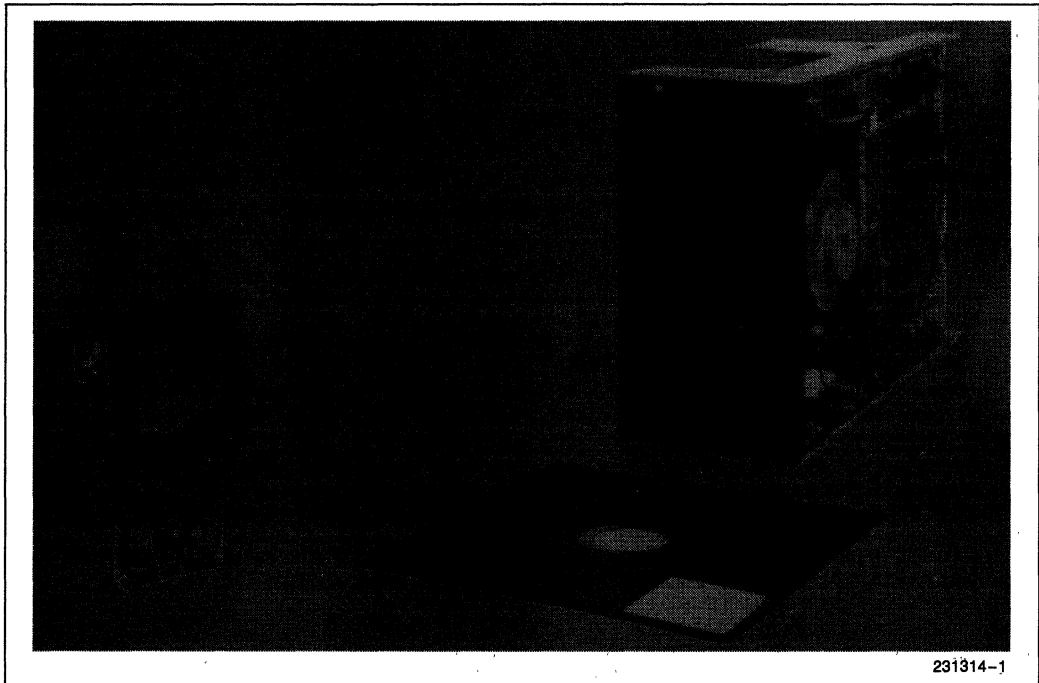


Figure 1. Comparison of Mini-Cassette and Floppy Disk Transports and Media

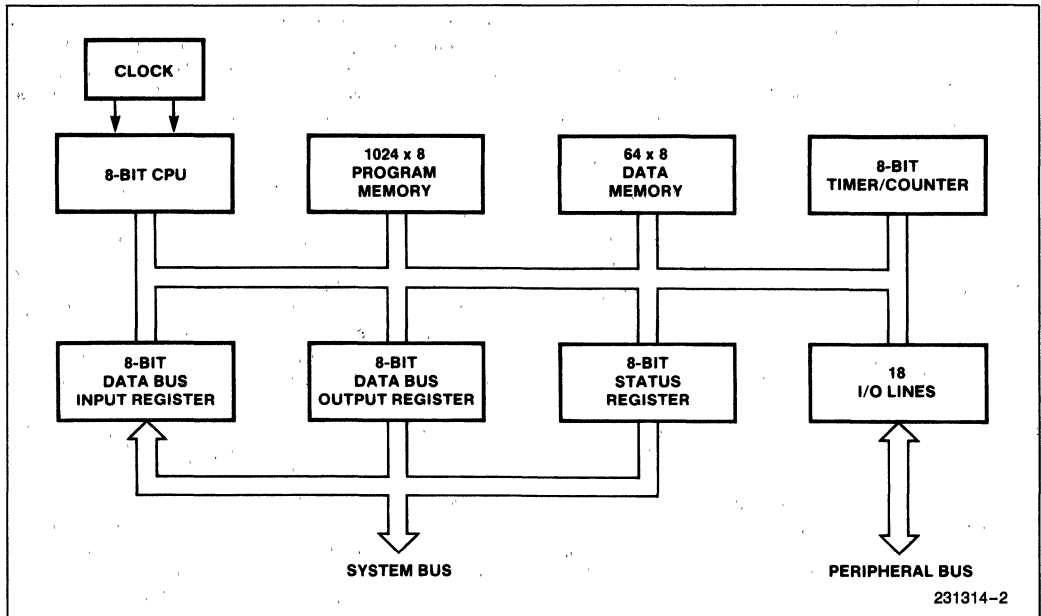


Figure 2. 8741AH/8041A Block Diagram

He has a single-chip dedicated controller. Testing may be accomplished using either an ICE-41A or the Single-Step mode of the 8741AH. UPI-41AH peripheral interfaces are being used to control printers, keyboards, displays, custom serial interfaces, and data encryption units. Of course, the UPI family is perfect for developing a dedicated controller for digital mini-cassette transports. To illustrate this application for the UPI family let's consider the job of controlling the Braemar CM-600 Mini-Dek*.

THE CM-600 MINI-DEK*

The Braemar CM-600 is representative of digital mini-cassette transports. It is a single-head, single-motor transport which operates entirely from a single 5V power supply. Its power requirements, including the motor, are 200 mA for read or write and 700 mA for rewind. Tape speeds are 3 inches per second (IPS) during read or write, 5 IPS fast forward, and 15 IPS rewind. With these speeds and a maximum recording density of 800 bits per inch (BPI), the maximum data rate is 2400 bits per second (BAUD). The data capacity using both sides of a 100-foot tape is 200K bytes. On top of this, the transport occupies only 22.5 cubic inches (3" x3" x2.5").

*MiniDek is a registered trademark of Braemar.

All I/O for the CM-600 is TTL-compatible and can be divided into three groups: motor control, data control, and cassette status. The motor group controls are GO/STOP, FAST/SLOW, and FORWARD/REVERSE. The data controls are READ/WRITE, DATA IN, and DATA OUT. The remaining group of outputs give the transport's status: CLEAR LEADER, CASSETTE PRESENCE, FILE PROTECT, and SIDE SENSOR. These signals, shown schematically in Figure 3 and Table 1, give the pin definition of the CM-600 16-pin I/O connector.

RECORDING FORMAT

The CM-600 does not provide either encoding or decoding of the recorded data; that task is left for the peripheral interface. A multitude of encoding techniques from which the user may choose are available. In this single-chip dedicated controller application, a "self-clocking" phase encoding scheme similar to that used in floppy discs was chosen. This scheme specifies that a logic "0" is a bit cell with no transition; a cell with a transition is a logic "1."

Table 1. CM-600 I/O Pin Definition

Pin	I/O	Function
1	—	Index pin—not used
2	—	Signal ground
3	O	Cassette side (0—side B, 1—side A)
4	I	Data input (0—space, 1—mark)
5	O	Cassette presence (0—cassette, 1—no cassette)
6	I	Read/Write (0—read, 1—write)
7	O	File protect (0—tab present, 1—tab removed)
8	—	+5V motor power
9	—	Power ground
10	—	Chassis ground
11	I	Direction (0—forward, 1—rewind)
12	I	Speed (0—fast, 1—slow)
13	O	Data output (0—space, 1—mark)
14	O	Clear leader (0—clear leader, 1—off clear leader)
15	I	Motion (0—go, 1—stop)
16	—	+5V logic power

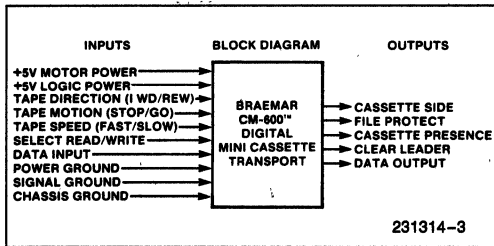


Figure 3. Braemar CM-600 Block Diagram

Figure 4 illustrates the encoding of the character 3AH assuming the previous data ended with the data line high. (The least significant bit is sent first.) Notice that there is always a "clocking" transition at the beginning of each cell. Decoding is simply a matter of triggering on this "clocking" transition, waiting $\frac{3}{4}$ of a bit cell time, and determining whether a mid-cell transition has occurred. Cells with no mid-cell transitions are data 0's; cells with transitions are data 1's. This encoding technique has all the benefits of Manchester encoding with the added advantage that the encoded data may be "decoded by eyeball:" long cells are always 0's, short cells are always 1's.

Besides the encoding scheme, the data format is also up to the user. This controller uses a variable byte length, checksum protected block format. Every block starts and ends with a SYNC character (AAH), and the character immediately preceding the last SYNC is the

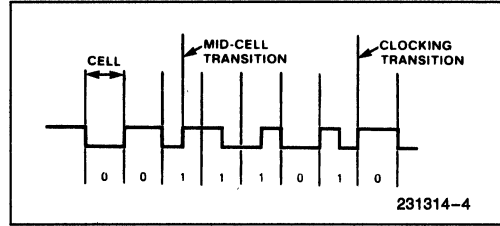


Figure 4. Modified Phase Encoding of Character 3A Hex

checksum. The checksum is capable of catching 2 bit errors. The number of data characters within a block is limited to 64K bytes. Blocks are separated by an Inter-Record Gap (IRG). The IRG is of such a length that the transport can stop and start within an IRG, as illustrated in the data block timing, Figure 5. Braemar specifies a maximum start or stop time of 150 ms for the transport, thus the controller uses 450 ms for the IRG. This gives plenty of margin for controlling the transport and also for detecting IRGs while skipping blocks.

THE UPI-41AHTM CONTROLLER

The goal of the UPI software design for this application was to make the UPI-41AH microcomputer into an intelligent cassette control processor. The host processor (8086, 8088, 8085A, etc.) simply issues a high-level command such as READ-a-block or WRITE-a-block. The 8741AH accepts the command, performs the requested operation, and returns to the host system a result code telling the outcome of the operation, eg. Good-Completion, Sync Error, etc. Table 2 shows the command and result code repertoire. The 8741AH completely manages all the data transfers for reading and writing.

As an example, consider the WRITE-a-block command. When this command is issued, the UPI-41AH expects a 16-bit number from the host telling how many data bytes to write (up to 64K bytes per block). Once this number is supplied in the form of two bytes, the host is free to perform other tasks; a bit in the UPI's STATUS register or an interrupt output will notify the host when a data transfer is required. The 8741AH then checks the transport's status to be sure that a cassette is present and not file protected. If either is false, a result code is returned to the host; otherwise the transport is started. After the peripheral controller checks to make sure that the tape is off the clear leader and past the hole in the tape, it writes a 450 ms IRG, a SYNC character, the block of data, the checksum, and the final SYNC character. (The tape has a clear leader at both ends and a small hole 6 inches from the end of each leader.) The data transfers from the host to the

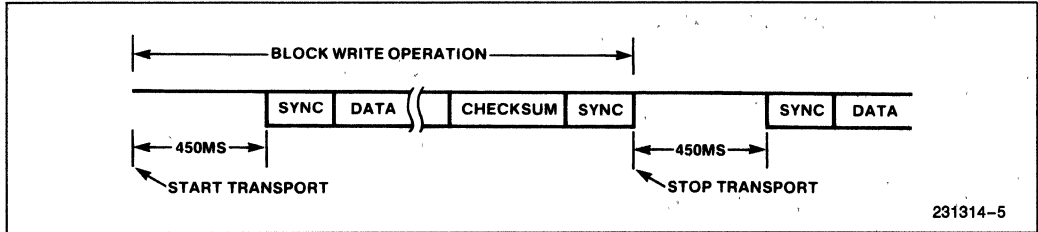


Figure 5. IRG/Block Timing Diagram (not to scale)

Table 2. Controller Command/Result Code Set

Command	Result
Read (01H)	Good-Completion (00H) Buffer Overrun Error (41H) Bad Synch1 Error (42H) Bad Synch2 Error (43H) Checksum Error (44H) Command Error (45H) End of Tape Error (46H)
Rewind (04H)	Good-Completion (00H)
Skip (03H)	Good-Completion (00H) End of Tape Error (47H) Beginning of Tape Error (48H)
Write (02H)	Good-Completion (00H) Buffer Underrun Error (81H) Command Error (82H) End of Tape Error (83H)

UPI-41AH slave microcomputer are double buffered. The controller requests only the desired number of data bytes by keeping track of the count internally.

If nothing unusual happened, such as finding clear leader while writing, it returns a Good-Completion result code to the host. If clear leader was encountered, the transport is stopped immediately and an End-of-Tape result code is returned to the host. Another possible error would be if the host is late in supplying data. If this occurs, the controller writes an IRG, stops the drive, and returns the appropriate Data-Underrun result code.

The READ-a-block command also provides error checking. Once this command is issued by the host, the controller checks for cassette presence. If present, it starts the transport. The data output from the transport is then examined and decoded continuously. If the first

character is not a SYNC, that's an error and the controller returns a Bad-First-SYNC result code (42H) after advancing to the next IRG. If the SYNC is good, the succeeding characters are read into an on-chip 30 character circular buffer. This continues until an IRG is encountered. When this occurs, the transport is stopped. The controller then tests that it is the last character. If it is a SYNC, the controller then compares the accumulated internal checksum to the block's checksum, the second to the last character of the block. If they match, a Good-Completion result code (00H) is returned to the host. If either test is bad, the appropriate error result code is returned. The READ command also checks for the End-of-Tape (EOT) clear leader and returns the appropriate error code if it is found before the read operation is complete.

The 30 character circular buffer allows the host up to 30 character times of response time before the host must collect the data. All data transfers take place thru the UPI-41AH Data Bus Buffer Output register (DBBOUT). The controller continually monitors the status of this register and moves characters from the circular buffer to the register whenever it is empty.

The SKIP-n-blocks command allows the host to skip the transport forward or backward up to 127 blocks. Once the command is issued, the controller expects one data byte specifying the number of blocks to skip. The most significant bit of this byte selects the direction of the skip (0 = forward, 1 = reverse). SKIP is a dual-speed operation in the forward direction. If the number of blocks to skip is greater than 8, the controller uses fast-forward (5 IPS until it is within 8 blocks of the desired location. Once within 8 blocks, the controller switches to the normal read speed (3 IPS) to allow accurate placement of the tape. The reverse skip uses only the rewind speed (15 IPS). Like the READ and WRITE commands, SKIP also checks for EOT and beginning-of-tape (BOT) depending upon the tape's direction. An error result code is returned if either is

encountered before the number of blocks skipped is complete.

The REWIND command simply rewinds the tape to the BOT clear leader. The ABORT command allows the termination of any operation in progress, except a REWIND. All commands, including ABORT, always leave the tape positioned on an IRG.

THE HARDWARE INTERFACE

There's hardly any hardware design effort required for the controller and transport interface in Figure 6. Since the CM-600 is TTL compatible, it connects directly to the I/O ports of the UPI controller. If the two are separated (i.e. on different PC cards), it is recommended that TTL buffers be provided. The only external circuitry needed is an LED driver for the DRIVE ACTIVE status indicator.

The 8741AH-to-host interface is equally straightforward. It has a standard asynchronous peripheral interface: 8 data lines (D₀-D₇), read (RD), write (WR), register select (AO), and chip select (CS). Thus it connects directly to an 8086, 8088, 8085A, 8080, or 8048 bus structure. Two interrupt outputs are provided for data transfer requests if the particular system is interrupt-driven. DMA transfer capability is also available. The clock input can be driven from a crystal directly or with the system clock (6 MHz max). The UPI-41AH clock may be asynchronous with respect to other clocks within the system.

This application was developed on an Intel iSBC 80/30 single board computer. The iSBC 80/30 is controlled by an 8085A microprocessor, contains 16K bytes of dual-ported dynamic RAM and up to 8K bytes of either EPROM or ROM. Its I/O complement consists of an 8255A Programmable Parallel Interface, an 8251A Programmable Communications Interface, an 8253 Programmable Interval Timer, and an 8259A Programmable Interrupt Controller. The iSBC 80/30 is especially convenient for UPI development since it contains an uncommitted socket dedicated to either an 8041A or 8741AH, complete with buffering for its I/O ports. The iSBC 80/30 to 8741AH interface is reflected in Figure 8. (Optionally, an iSBC 569 Digital Controller board could be used. The iSBC 569 board contains three uncommitted UPI sockets with an interface similar to that in Figure 8.)

Looking at the host-to-controller interface, the host sees the 8741AH as three registers in the host's I/O address space: the data register, the command register, and the status register. The decoding of these registers is shown in Figure 7. All data and commands for the controller are written into the Data Bus Buffer Input register (DBBIN). The state of the register select input, AO, determines whether a command or data is written. (Writes with AO set to 1 are commands by convention.) All data and results from the controller are read by the host from the Data Bus Buffer Output register (DBBOU).

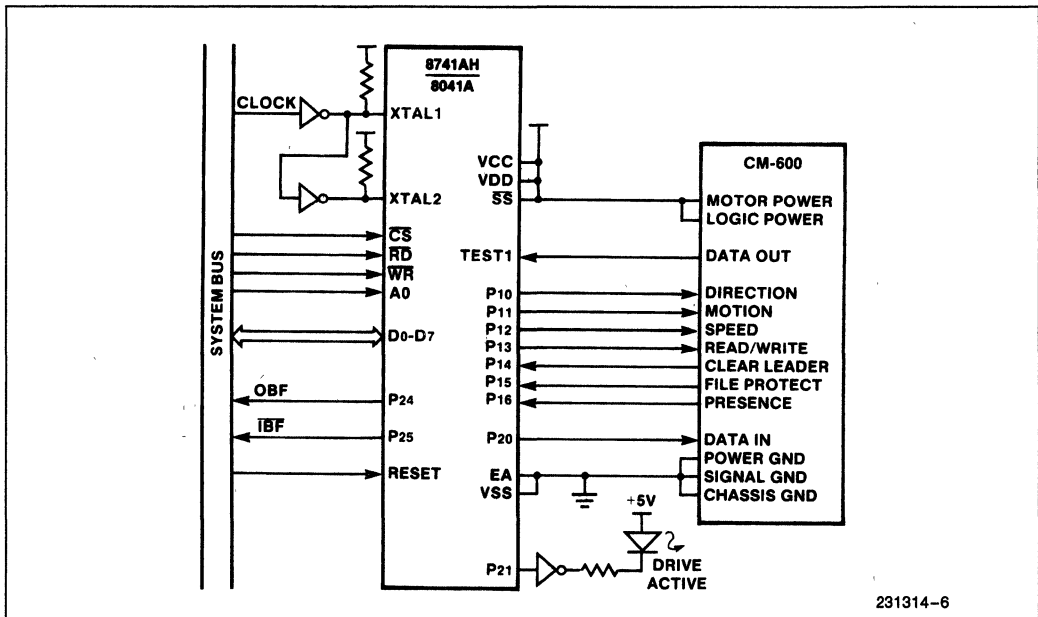
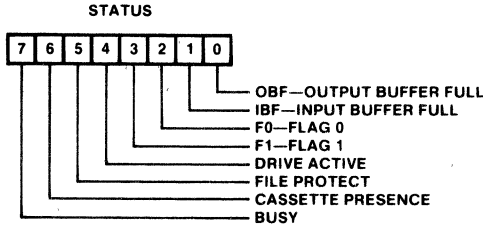


Figure 6. Controller/Transport System Schematic

CS	RD	WR	A0	Register
0	0	1	0	DBBOUT
0	0	1	1	STATUS
0	1	0	0	DBBIN (DATA)
0	1	0	1	DBBIN (COMMAND)
1	X	X	X	NONE

Figure 7. 8741AH/8041A Interface Register Decoding



231314-7

Figure 8. Status Register Bit Definition

The Status register contains flags which give the host the status of various operations within the controller. Its format is given in Figure 8. The Input Buffer Full (IBF) and Output Buffer Full (OBF) flags show the Status of the DBBIN and DBBOUT registers respectively. IBF indicates when the DBBIN register contains data written by the host. The host may write to DBBIN only when IBF is 0. Likewise, the host may read DBBOUT only when OBF is set to a 1. These bits are handled automatically by the UPI-41AH internal hardware. FLAG 0 (F₀) and FLAG 1 (F₁) are general purpose flags used internally by the controller which have no meaning externally.

The remaining four bits are user-definable. For this application they are DRIVE ACTIVE, FILE PROTECT, CASSETTE PRESENCE, and BUSY flags. The FILE PROTECT and CASSETTE PRESENCE flags reflect the state of the corresponding I/O lines from the transport. DRIVE ACTIVE is set whenever the transport motor is on and the controller is performing an operation. The BUSY flag indicates whether the contents of the DBBOUT register is data or a result code. The BUSY flag is set whenever a command is issued by the host and accepted by the controller. As long as BUSY

is set, any character found in DBBOUT is a result code. Thus whenever the host finds OBF set, it should test the BUSY flag to determine whether the character is data or a result code.

Notice the OBF and IBF are available as interrupt outputs to the host processor, Figure 6. These outputs are self-clearing, that is, OBF is set automatically upon the controller loading DBBOUT and cleared automatically by the host reading DBBOUT. Likewise IBF is cleared to a 0 by the host writing into DBBIN: set to a 1 when the controller reads DBBIN into the accumulator.

The flow charts of Figure 9 show the flow of sample host software assuming a polling software interface between the host and the controller. The WRITE command requires two additional count bytes which form the 16-bit byte count. These extra bytes are "hand-shaked" into the controller using the IBF flag in the STATUS register. Once these bytes are written, the host writes data in response to IBF being cleared. This continues until the host finds OBF set indicating that the operation is complete and reads the result code from DBBOUT. No testing of BUSY is needed since only the result code appears in the DBBOUT register.

The READ command does require that BUSY be tested. Once the READ command is written into the controller, the host must test BUSY whenever OBF is set to determine whether the contents of DBBOUT is data from the tape or the result code.

THE CONTROLLER SOFTWARE

The UPI-41AH software to control the cassette can be divided up into various commands such as WRITE, READ and ABORT. In a previous version of this application note (May 1980), software was described that implemented these commands. This code however did not adequately compensate for speed variations of the motor during record and playback nor for data distortion caused by the magnetic media. Since then, a new code has been written to include these effects. This revised software is now available through the INTEL User's Library, INSITE. For more information on this software or INSITE, contact your local INTEL Sales Office.



**APPLICATION
NOTE**

AP-281

July 1986

**UPI-452 Accelerates iAPX 286
Bus Performance**

CHRISTOPHER SCOTT
TECHNICAL MARKETING ENGINEER
INTEL CORPORATION

Order Number: 292018-001

INTRODUCTION

The UPI-452 targets the leading problem in peripheral to host interfacing, the interface of a slow peripheral with a fast Host or "bus utilization". The solution is data buffering to reduce the delay and overhead of transferring data between the Host microprocessor and I/O subsystem. The Intel CMOS UPI-452 solves this problem by combining a sophisticated programmable FIFO buffer and a slave interface with an MSC-51 based microcontroller.

The UPI-452 is Intel's newest Universal Peripheral Interface family member. The UPI-452 FIFO buffer enables Host—peripheral communications to be through streams or bursts of data rather than by individual bytes. In addition the FIFO provides a means of embedding commands within a stream or block of data. This enables the system designer to manage data and commands to further off-load the Host.

The UPI-452 interfaces to the iAPX 286 microprocessor as a standard Intel slave peripheral device. READ, WRITE, CS and address lines from the Host are used to access all of the Host addressable UPI-452 Special Function Registers (SFR).

The UPI-452 combines an MSC-51 microcontroller, with 256 bytes of on-chip RAM and 8K bytes of EPROM/ROM, twice that of the 80C51, a two channel DMA controller and a sophisticated 128 byte, two channel, bidirectional FIFO in a single device. The UPI-452 retains all of the 80C51 architecture, and is fully compatible with the MSC-51 instruction set.

This application note is a description of an iAPX 286 to UPI-452 slave interface. Included is a discussion of the respective timings and design considerations. This application note is meant as a supplement to the UPI-452 Advance Data Sheet. The user should consult the data sheet for additional details on the various UPI-452 functions and features.

UPI-452 IAPX 286 SYSTEM CONFIGURATION

The interface described in this application note is shown in Figure 1, iAPX 286 UPI-452 System Block Diagram. The iAPX 286 system is configured in a local bus architecture design. DMA between the Host and the UPI-452 is supported by the 82258 Advanced DMA Controller. The Host microprocessor accesses all UPI-452 externally addressable registers through address decoding (see Table 3, UPI-452 External Address Decoding). The timings and interface descriptions below are given in equation form with examples of specific calculations. The goal of this application note is a set of interface analysis equations. These equations are the tools a system designer can use to fully utilize the features of the UPI-452 to achieve maximum system performance.

HOST-UPI-452 FIFO SLAVE INTERFACE

The UPI-452 FIFO acts as a buffer between the external Host 80286 and the internal CPU. The FIFO allows the Host - peripheral interface to achieve maximum decoupling of the interface. Each of the two FIFO channels is fully user programmable. The FIFO buffer ensures that the respective CPU, Host or internal CPU, receives data in the same order as transmitted. Three slave bus interface handshake methods are supported by the UPI-452; DMA, Interrupt and Polled.

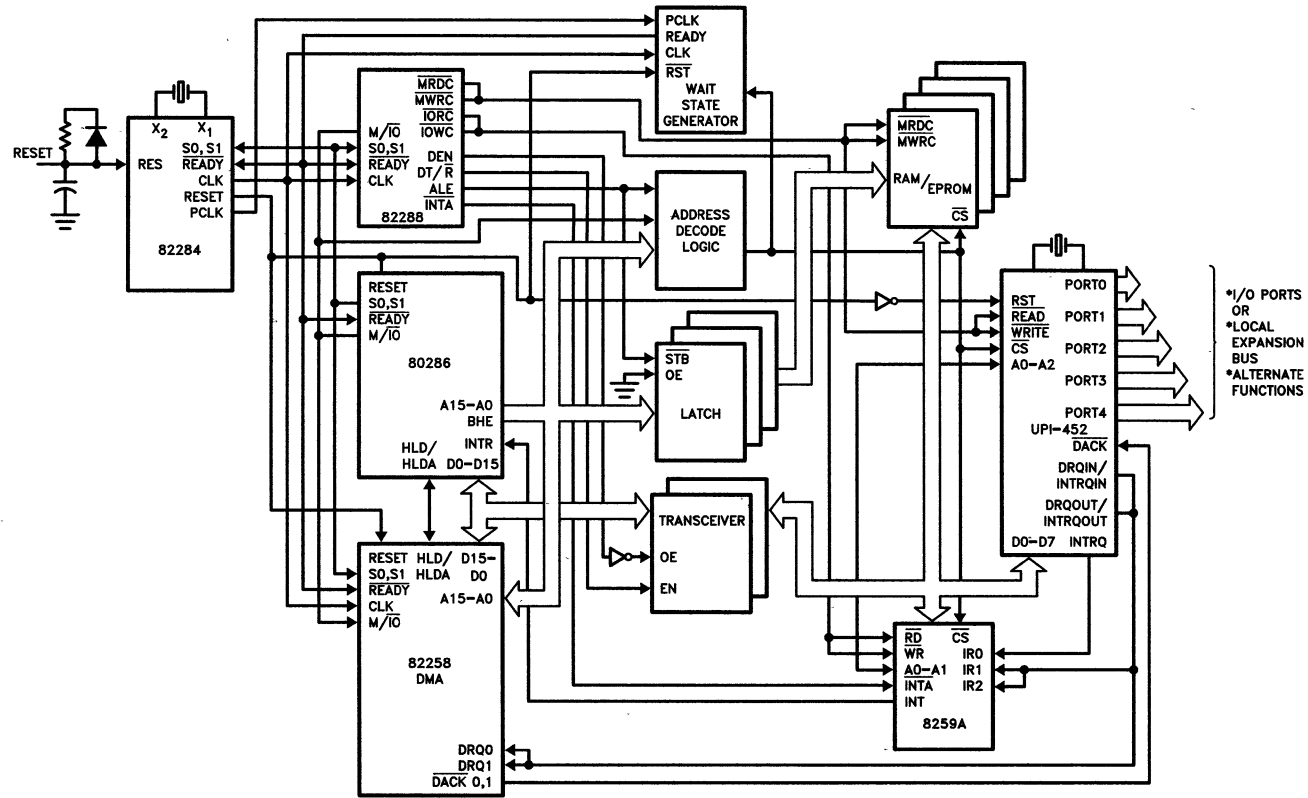
The interface between the Host 80286 and the UPI-452 is accomplished with a minimum of signals. The 8 bit data bus plus READ, WRITE, CS, and AO-2 provide access to all of the externally addressable UPI-452 registers including the two FIFO channels. Interrupt and DMA handshaking pins are tied directly to the interrupt controller and DMA controller respectively.

DMA transfers between the Host and UPI-452 are controlled by the Host processors DMA controller. In the example shown in Figure 1, the Host DMA controller is the 82258 Advanced DMA Controller. An internal DMA transfer to or from the FIFO, as well as between other internal elements, is controlled by the UPI-452 internal DMA processor. The internal DMA processor can also transfer data between Input and Output FIFO channels directly. The description that follows details the UPI-452 interface from both the Host processor's and the UPI-452's internal CPU perspective.

One of the unique features of the UPI-452 FIFO is its ability to distinguish between commands and data embedded in the same data block. Both interrupts and status flags are provided to support this operation in either direction of data transfer. These flags and interrupts are triggered by the FIFO logic independent of, and transparent to either the Host or internal CPUs. Commands embedded in the data block, or stream, are called Data Stream Commands.

Programmable FIFO channel Thresholds are another unique feature of the UPI-452. The Thresholds provide for interrupting the Host only when the Threshold number of bytes can be read or written to the FIFO buffer. This further decouples the Host UPI-452 interface by relieving the Host of polling the buffer to determine the number of bytes that can be read or written. It also reduces the chances of overrun and underrun errors which must be processed.

The UPI-452 also provides a means of bypassing the FIFO, in both directions, for an immediate interrupt of either the Host or internal CPU. These commands are called Immediate Commands. A complete description of the internal FIFO logic operation is given in the FIFO Data Structure section.



292018-1

Figure 1. iAPX 286 UPI-452 System Block Diagram

5-211

UPI-452 INITIALIZATION

The UPI-452 at power-on reset automatically performs a minimum initialization of itself. The UPI-452 notifies the Host that it is in the process of initialization by setting a Host Status SFR bit. The user UPI-452 program must release the UPI-452 from initialization for the FIFO to be accessible by the Host. This is the minimum Host to UPI-452 initialization sequence. All further initialization and configuration of the UPI-452, including the FIFO, is done by the internal CPU under user program control. No interaction or programming is required by the Host 80286 for UPI-452 initialization.

At power-on reset the UPI-452 automatically enters FIFO DMA Freeze Mode by resetting the Slave Control (SLCON) SFR FIFO DMA Freeze/Normal Mode bit to FIFO DMA Freeze Mode (FRZ = "0"). This forces the Slave Status (SSTAT) and Host Status (HSTAT) SFR FIFO DMA Freeze/Normal Mode bits to FIFO DMA Freeze Mode In Progress. FIFO DMA Freeze Mode allows the FIFO interface to be configured, by the internal CPU, while inhibiting Host access to the FIFO.

The MODE SFR is forced to zero at reset. This disables, (tri-states) the DRQIN/INTRQIN, DRQOUT/INTRQOUT and INTRQ output pins. INTRQ is inhibited from going active to reflect the fact that a Host Status SFR bit, FIFO DMA Freeze Mode, is active. If the MODE SFR INTRQ configure bit is enabled (= '1'), before the Slave Control and Host Status SFR FIFO DMA Freeze/Normal Mode bit is set to Normal Mode, INTRQ will go active immediately.

The first action by the Host following reset is to read the UPI-452 Host Status SFR Freeze/Normal Mode bit to determine the status of the interface. This may be done in response to a UPI-452 INTRQ interrupt, or by polling the Host Status SFR. Reading the Host Status SFR resets the INTRQ line low.

Any of the five FIFO interface SFRs, as well as a variety of additional features, may be programmed by the internal CPU following reset. At power-on reset, the five FIFO Special Function Registers are set to their default values as listed in Table 1. All reserved location bits are set to one, all other bits are set to zero in these three SFRs. The FIFO SFRs listed in Table 1 can be programmed only while the UPI-452 is in FIFO DMA Freeze Mode. The balance of the UPI-452 SFRs default values and descriptions are listed in the UPI-452 Advance Data Sheet in the Intel Microsystems Component Handbook Volume II and Microcontroller Handbook.

The above sequence is the minimum UPI-452 internal initialization required. The last initialization instruction must always set the UPI-452 to Normal Mode. This causes the UPI-452 to exit Freeze Mode and enables

Host read/write access of the FIFO. The internal CPU sets the Slave Control (SLCON) SFR FIFO DMA Freeze/Normal Mode (FRZ) bit high (= 1) to activate Normal Mode. This causes the Slave Status (SSTAT) and Host Status (HSTAT) SFR FIFO DMA Freeze Mode bits to be set to Normal Mode. Table 2, UPI-452 Initialization Event Sequence Example, shows a summary of the initialization events described above.

Table 1. FIFO Special Function Register Default Values

SFR Name	Label	Reset Value
Channel Boundary Pointer	CBP	40H/64D
Output Channel Read Pointer	ORPR	40H/64D
Output Channel Write Pointer	OWPR	40H/64D
Input Channel Read Pointer	IRPR	00H/0D
Input Channel Write Pointer	IWPR	00H/0D
Input Threshold	ITH	00H/0D
Output Threshold	OTH	01H/1D

Table 2. UPI-452 Initialization Event Sequence Example

Event Description	SFR/bit
Power-on Reset	
UPI-452 forces FIFO DMA Freeze Mode (Host access to FIFO inhibited)	SLCON FRZ = 0
UPI-452 forces Slave Status and Host Status SFR to FIFO DMA Freeze Mode In Progress	SSTAT SST5 = 0 HSTAT HST1 = 1
UPI-452 forces all SFRs, including FIFO SFRs, to default values.	
* UPI-452 user program enables INTRQ, INTRQ goes active, high	MODE MD4 = 1
* Host READ's UPI-452 Host Status (HSTAT) SFR to determine interrupt source, INTRQ goes low	
* UPI-452 user program initializes any other SFRs; FIFO, Interrupts, Timers/Counters, etc.	
User program sets Slave Control SFR to Normal Mode (Host access to FIFO enabled)	SLCON FRZ = 1
UPI-452 forces Slave and Host Status SFRs bits to Normal Operation	SSTAT SST5 = 1 HSTAT HST1 = 0
* Host polls Host Status SFR to determine when it can access the FIFO - or - * Host waits for UPI-452 Request for Service interrupt to access FIFO	

* user option

FIFO DATA STRUCTURES

Overview

The UPI-452 provides three means of communication between the Host microprocessor and the UPI-452 in either direction;

- Data
- Data Stream Commands
- Immediate Commands

Data and Data Stream Commands (DSC) are transferred between the Host and UPI-452 through the UPI-452 FIFO buffer. The third, Immediate Commands, provides a means of bypassing the FIFO entirely. These three data types are in addition to direct access by either Host or Internal CPU of dedicated Status and Control Special Function Registers (SFR).

The FIFO appears to both the Host 80286 and the internal CPU as 8 bits wide. Internally the FIFO is logically nine bits wide. The ninth bit indicates whether the byte is a data or a Data Stream Command (DSC) byte; 0 = data, 1 = DSC. The ninth bit is set by the FIFO logic in response to the address specified when writing to the FIFO by either Host or internal CPU. The FIFO uses the ninth bit to condition the UPI-452 interrupts and status flags as a byte is made available for a Host or internal CPU read from the FIFO. Figures 2 and 3 show the structure of each FIFO channel and the logical ninth bit.

It is important to note that both data and DSCs are actually entered into the FIFO buffer (see Figures 2 and 3). External addressing of the FIFO determines the state of the internal FIFO logic ninth bit. Table 3 shows the UPI-452 External Address Decoding used by the Host and the corresponding action. The internal CPU interface to the FIFO is essentially identical to the external Host interface. Dedicated internal Special Function Registers provide the interface between the FIFO, internal CPU and the internal two channel DMA processor. FIFO read and write operations by the Host and internal CPU are interleaved by the UPI-452 so they appear to be occurring simultaneously.

The ninth bit provides a means of supporting two data types within the FIFO buffer. This feature enables the Host and UPI-452 to transfer both commands and data while maintaining the decoupled interface a FIFO buffer provides. The logical ninth bit provides both a means of embedding commands within a block of data and a means for the internal CPU, or external Host, to discriminate between data and commands. Data or DSCs may be written in any order desired. Data Stream

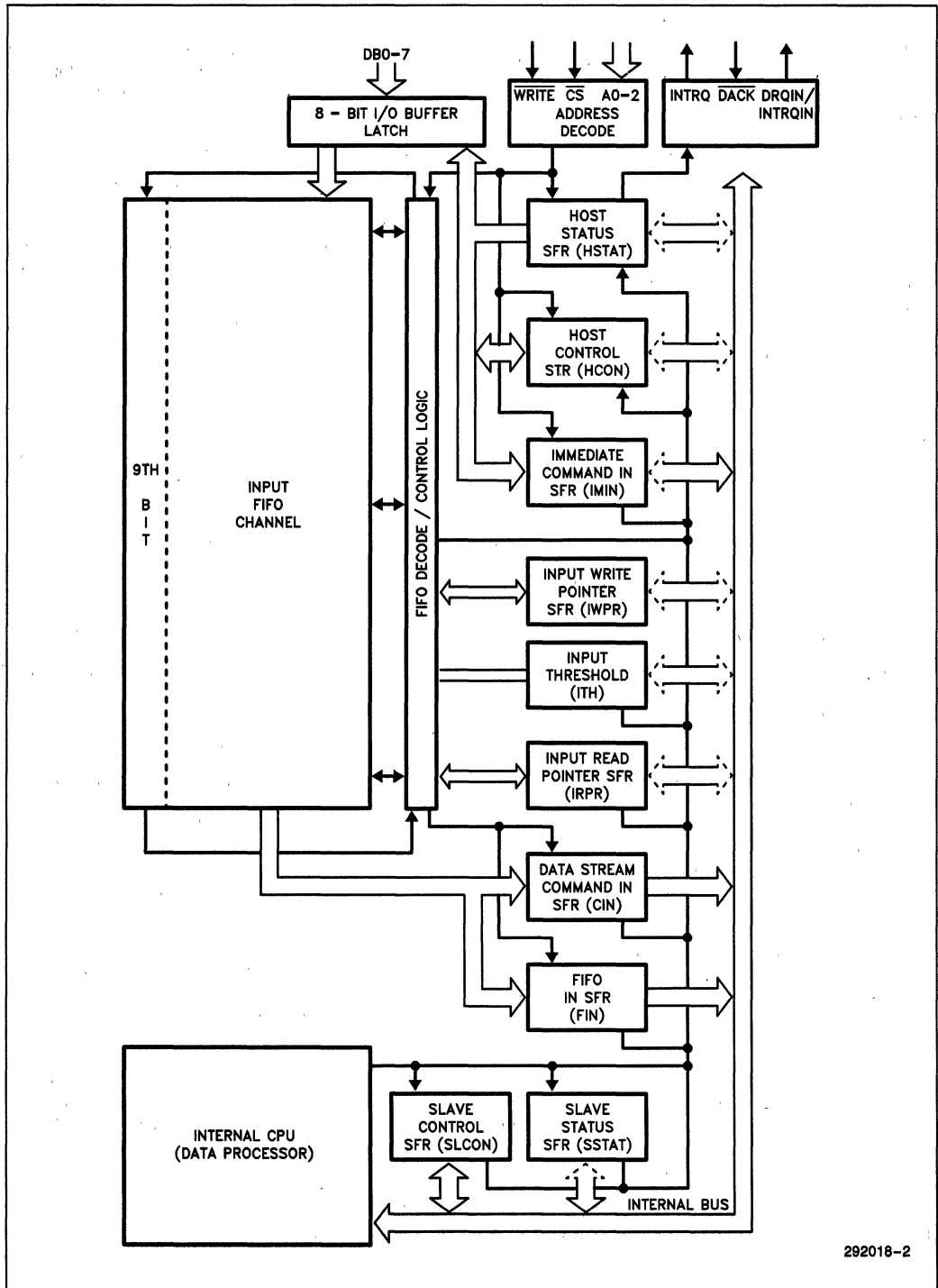
Commands can be used to structure or dispatch the data by defining the start and end of data blocks or packets, or how the data following a DSC is to be processed.

A Data Stream Command (DSC) acts as an internal service routine vector. The DSC generates an interrupt to a service routine which reads the DSC. The DSC byte acts as an address vector to a user defined service routine. The address can be any program or data memory location with no restriction on the number of DSCs or address boundaries.

A Data Stream Command (DSC) can also be used to clear data from the FIFO or "FLUSH" the FIFO. This is done by appending a DSC to the end of a block of data entered in the FIFO which is less than the programmed threshold number of bytes. The DSC will cause an interrupt, if enabled, to the respective receiving CPU. This ensures that a less than Threshold number of bytes in the FIFO will be read. Two conditions force a Request for Service interrupt, if enabled, to the Host. The first is due to a Threshold number of bytes having been written to the FIFO Output channel; the second is if a DSC is written to the Output FIFO channel. If less than the Threshold number of bytes are written to the Output FIFO channel, the Host Status SFR flag will not be set, and a Request for Service interrupt will not be generated, if enabled. By appending a DSC to end of the data block, the FIFO Request for Service flag and/or interrupt will be generated.

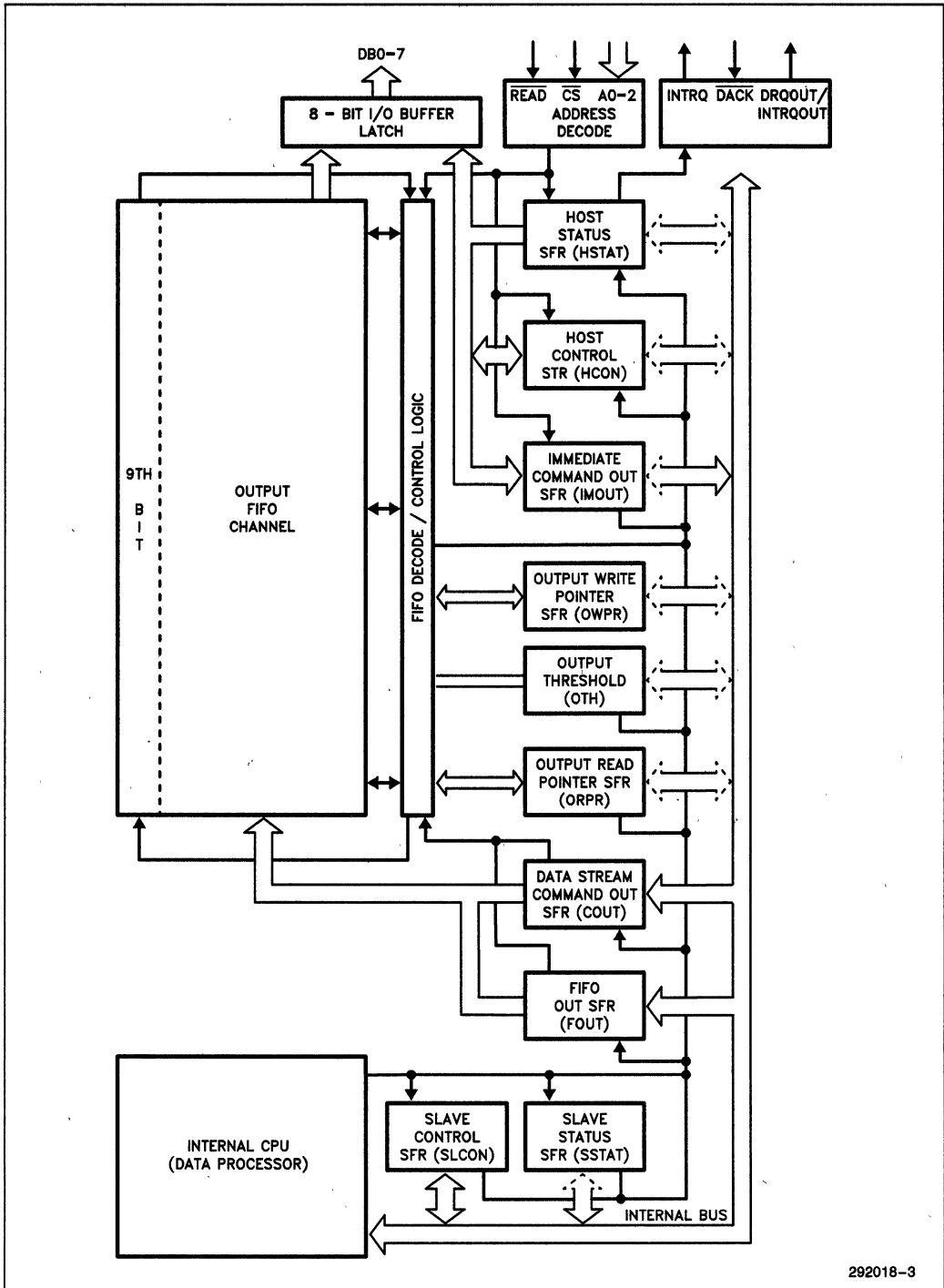
An example of a FIFO Flush application is a mass storage subsystem. The UPI-452 provides the system interface to a subsystem which supports tape and disk storage. The FIFO size is dynamically changed to provide the maximum buffer size for the direction of transfer. Large data blocks are the norm in this application. The FIFO Flush provides a means of purging the FIFO of the last bytes of a transfer. This guarantees that the block, no matter what its size, will be transmitted out of the FIFO.

Immediate Commands allow more direct communication between the Host processor and the UPI-452 by bypassing the FIFO in either direction. The Immediate Command IN and OUT SFRs are two more unique address locations externally and internally addressable. Both DSCs and Immediate Commands have internal interrupts and interrupt priorities associated with their operation. The interrupts are enabled or disabled by setting corresponding bits in the Slave Control (SLCON), Interrupt Enable (IEC), Interrupt Priority (IPC) and Interrupt Enable and Priority (IEP) SFRs. A detailed description of each of these may be found in the UPI-452 Advance Information Data Sheet.



292018-2

Figure 2. Input FIFO Channel Functional Diagram



292018-3

Figure 3. Output FIFO Channel Functional Diagram

Table 3. UPI-452 External Address Decoding

DACK	CS	A2	A1	A0	READ	WRITE
1	1	X	X	X	No Operation	No Operation
1	0	0	0	0	Data or DMA from Output FIFO Channel	Data or DMA to Input FIFO Channel
1	0	0	0	1	Data Stream Command from Output FIFO Channel	Data Stream Command to Input FIFO Channel
1	0	0	1	0	Host Status SFR Read	Reserved
1	0	0	1	1	Host Control SFR Read	Host Control SFR Write
1	0	1	0	0	Immediate Command SFR Read	Immediate Command SFR Write
1	0	1	1	X	Reserved	Reserved
0	X	X	X	X	DMA Data from Output FIFO Channel	DMA Data to Input FIFO Channel

Below is a detailed description of each FIFO channel's operation, including the FIFO logic response to the ninth bit, as a byte moves through the channel. The description covers each of the three data types for each channel. The details below provide a picture of the various FIFO features and operation. By understanding the FIFO structure and operation the user can optimize the interface to meet the requirements of an individual design.

OUTPUT CHANNEL

This section covers the data path from the internal CPU to the HOST. Data Stream Command or Immediate Command processing during Host DMA Operations is covered in the DMA section.

UPI-452 Internal Write to the FIFO

The internal CPU writes data and Data Stream Commands into the FIFO through the FIFO OUT (FOUT) and Command OUT (COUT) SFRs. When a Threshold number of bytes has been written, the Host Status SFR Output FIFO Request for Service bit is set and an interrupt, if enabled, is generated to the Host. Either the INTRQ or DRQOUT/INTRQOUT output pins can be used for this interrupt as determined by the MODE and Host Control (HCON) SFR setting. The Host responds to the Request for Service interrupt by reading the Host Status (HSTAT) SFR to determine the source of the interrupt. The Host then reads the Threshold number of bytes from the FIFO. The internal CPU may continue to write to the FIFO during the Host read of the FIFO Output channel.

Data Stream Commands may be written to the Output FIFO channel at any time during a write of data bytes. The write instruction need only specify the Command Out (COUT) SFR in the direct register instruction used. Immediate Commands may also be written at any time to the Immediate Command OUT (IMOUT) SFR. The Host reads Immediate Commands from the Immediate Command OUT (IMOUT).

The internal CPU can determine the number of bytes to write to the FIFO Output channel in one of three ways. The first, and most efficient, is by utilizing the internal DMA processor which will automatically manage the writing of data to avoid Underrun or Overrun Errors. The second is for the internal CPU to read the Output FIFO channels Read and Write Pointers and compare their values to determine the available space. The third method for determining the available FIFO space is to always write the programmed channel size number of bytes to the Output FIFO. This method would use the Overrun Error flag and interrupt to halt FIFO writing whenever the available space was less than the channel size. The interrupt service routine could read the channel pointers to determine or monitor the available channel space. The time required for the internal CPU to write data to the Output FIFO channel is a function of the individual instruction cycle time and the number of bytes to be written.

Host Read from the FIFO

The Host reads data or Data Stream Commands (DSC) from the FIFO in response to the Host Status (HSTAT) SFR flags and interrupts, if enabled. All Host read operations access the same UPI-452 internal I/O Buffer Latch. At the end of the previous Host FIFO read cycle a byte is loaded from the FIFO into the I/O Buffer Latch and Host Status (HSTAT) SFR bit 5 is set or cleared (1 = DSC, 0 = data) to reflect the state of the byte's FIFO ninth bit. If the FIFO ninth bit is set (= 1) indicating a DSC, an interrupt is generated to the external Host via INTRQ pin or INTRQIN/INTRQOUT pins as determined by Host Control (HCON) SFR bit 1. The Host then reads the Host Status (HSTAT) SFR to determine the source of the interrupt.

The most efficient Host read operation of the FIFO Output channel is through the use of Host DMA. The UPI-452 fully supports external DMA handshaking. The MODE and Host Control SFRs control the configuration of UPI-452 Host DMA handshake outputs. If Host DMA is used the Threshold Request for Service interrupt asserts the UPI-452 DMA Request (DRQOUT) output. The Host DMA processor acknowledges with DACK which acts as a chip select of the FIFO channels. The DMA transfer would stop when either the threshold byte count had been read, as programmed in the Host DMA processor, or when the DRQOUT output is brought inactive by the UPI-452.

INPUT CHANNEL

This section covers the data path from the HOST to the internal CPU or internal DMA processor. The details of Data Stream Command or Immediate Command processing during internal DMA operations are covered in the DMA section below.

Host Write to the FIFO

The Host writes data and Data Stream Commands into the FIFO through the FIFO IN (FIN) and Command IN (CIN) SFRs. When a Threshold number of bytes has been read out of the Input FIFO channel by the internal CPU, the Host Status SFR Input FIFO Request for Service bit is set and an interrupt, if enabled, is generated to the Host. The Input FIFO Threshold interrupt tells the Host that it may write the next block of data into the FIFO. Either the INTRQ or DRQIN/INTRQIN output pins can be used for this interrupt as determined by the MODE and Host Control (HCON) SFR settings. The Host may continue to write to the FIFO Input channel during the internal CPU read of the FIFO. Data Stream Commands may be written to the FIFO Input channel at any time during a write of data bytes. Immediate Commands may also be written at any time to the Immediate Command IN (IMIN) SFR.

The Host also has three methods for determining the available FIFO space. Two are essentially identical to that of the internal CPU. They involve reading the FIFO Input channel pointers and using the Host Status SFR Underrun and Overrun Error flags and Request for Service interrupts these would generate, if enabled in combination. The third involves using the UPI-452 Host DMA controller handshake signals and the programmed Input FIFO Threshold. The Host would receive a Request for Service interrupt when an Input FIFO channel has a Threshold number of bytes able to be written by the Host. The Host service routine would then write the Threshold number of bytes to the FIFO.

If a Host DMA is used to write to the FIFO Input channel, the Threshold Request for Service interrupt could assert the UPI-452 DRQIN output. The Host DMA processor would assert \overline{DACK} and the FIFO write would be completed by Host the DMA processor. The DMA transfer would stop when either the Threshold byte count had been written or the DRQIN output was removed by the UPI-452. Additional details on Host and internal DMA operation is given below.

Internal Read of the FIFO

At the end of an internal CPU read cycle a byte is loaded from the FIFO buffer into the FIFO IN/Command IN SFR and Slave Status (SSTAT) SFR bit 1 is set or cleared (1 = data, 0 = DSC) to reflect the state of the FIFO ninth bit. If the byte is a DSC, the FIFO ninth bit is set (= 1) and an interrupt is generated, if enabled, to the Internal CPU. The internal CPU then reads the Slave Status (SSTAT) SFR to determine the source of the interrupt.

Immediate Commands are written by the Host and read by the internal CPU through the Immediate Command IN (IMIN) SFR. Once written, an Immediate Command sets the Slave Status (SSTAT) SFR flag bit and generates an interrupt, if enabled, to the internal CPU. In response to the interrupt the internal CPU

reads the Slave Status (SSTAT) SFR to determine the source of the interrupt and service the Immediate Command.

FIFO INPUT/OUTPUT CHANNEL SIZE

Host

The Host does not have direct control of the FIFO Input or Output channel sizes or configuration. The Host can, however, issue Data Stream Commands or Immediate Commands to the UPI-452 instructing the UPI-452 to reconfigure the FIFO interface by invoking FIFO DMA Freeze Mode. The Data Stream Command or Immediate Command would be a vector to a service routine which performs the specific reconfiguration.

UPI-452 Internal

The default power-on reset FIFO channel sizes are listed in the "Initialization" section and can be set only by the internal CPU during FIFO DMA Freeze Mode. The FIFO channel size is selected to achieve the optimum application performance. The entire 128 byte FIFO can be allocated to either the Input or Output channel. In this case the other channel consists of a single SFR; FIFO IN/Command IN or FIFO OUT/Command OUT SFR. Figure 4 shows a FIFO division with a portion devoted to each channel. Figure 5 shows a FIFO configuration with all 128 bytes assigned to the Output channel.

The FIFO channel Threshold feature allows the user to match the FIFO channel size and the performance of the internal and Host data transfer rates. The programmed Threshold provides an elasticity to the data transfer operation. An example is if the Host FIFO

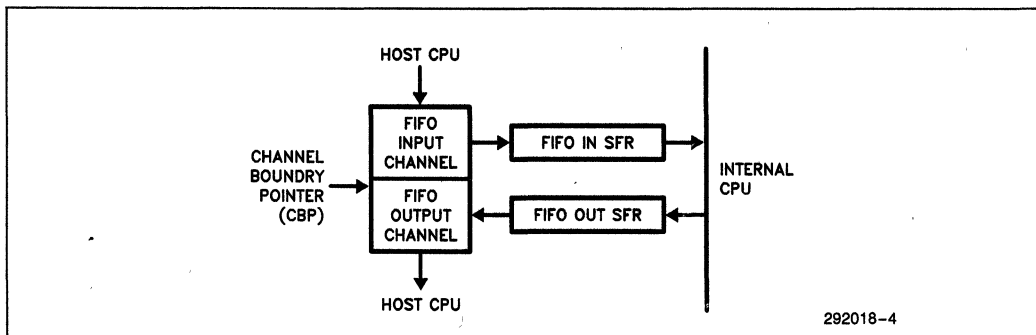


Figure 4. Full Duplex FIFO Operation

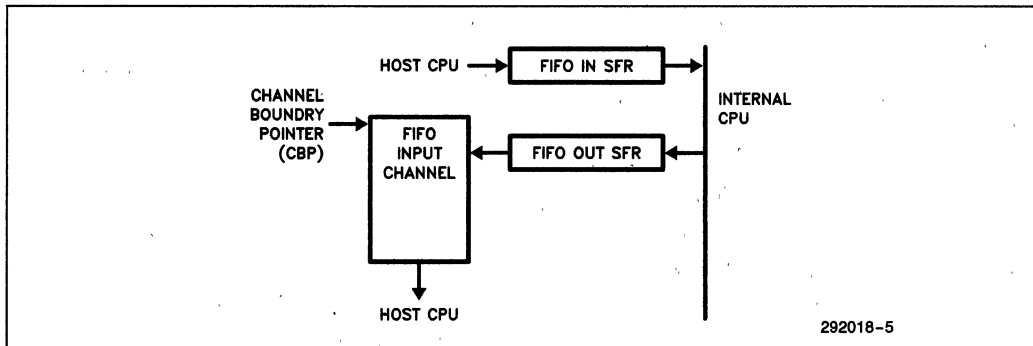


Figure 5. Entire FIFO Assigned to Output Channel

data transfer rate is twice as fast as the internal FIFO DMA data transfer rate. In this example the FIFO Input channel size is programmed to be 64 bytes and the Input channel Threshold is programmed to be 20 bytes. The Host writes the first 64 bytes to the Input FIFO. When the internal DMA processor has read 20 bytes the Threshold interrupt, or DMA request (DRQIN), is generated to signal the Host to begin writing more data to the Input FIFO channel. The internal DMA processor continues to read data from the Input channel as the Host, or Host DMA processor, writes to the FIFO. The Host can write 40 bytes to the FIFO Input channels in the time it takes for the internal DMA processor to read 20 more bytes from it. This will keep both the Host and internal DMA operating at their maximum rates without forcing one to wait for the other.

Two methods of managing the FIFO size are possible; fixed and variable channel size. A fixed channel size is one where the channel is configured at initialization and remains unchanged throughout program execution. In a variable FIFO channel size, the configuration is changed dynamically to meet the data transmission requirements as needed. An example of a variable channel size application is the mass storage subsystem described earlier. To meet the demands of a large data block transfer the FIFO size could be fully allocated to the Input or Output channel as needed. The Thresholds are also reprogrammed to match the respective data transfer rates.

An example of a fixed channel size application might be one which uses the UPI-452 to directly control a series of stepper motors. The UPI-452 manages the motor operation and status as required. This would include pulse train, acceleration, deceleration and feedback. The Host transmits motor commands to the UPI-452 in blocks of 6–10 bytes. Each block of motor command data is preceded by a command to the UPI-452 which selects a specific motor. The UPI-452 transmits blocks of data to the Host which provides motor and overall system status. The data and embedded commands structure, indicating the specific motor, is the same. In

this example the default 64 bytes per channel might be adequate for both channels.

INTERRUPT RESPONSE TIMING

Interrupts enable the Host UPI-452 FIFO buffer interface and the internal CPU FIFO buffer interface to operate with a minimum of overhead on the respective CPU. Each CPU is “interrupted” to service the FIFO on an as needed basis only. In configuring the FIFO buffer Thresholds and choosing the appropriate internal DMA Mode the user must take into account the interrupt response time for both CPUs. These response times will affect the DMA transfer rates for each channel. By choosing FIFO channel Thresholds which reflect both the respective DMA transfer rate and the interrupt response time the user will achieve the maximum data throughput and system bus decoupling. This in turn will mean the overall available system bus bandwidth will increase.

The following section describes the FIFO interrupt interface to the Host and internal CPU. It also describes an analysis of sample interrupt response times for the Host and UPI-452 internal CPU. These equations and the example times shown are then used in the DMA section to further analyze an optimum Host UPI-452 interface.

HOST

Interrupts to the Host processor are supported by the three UPI-452 output pins; INTRQ, DRQIN/INTRQIN and DRQOUT/INTRQOUT. INTRQ is a general purpose Request For Service interrupt output. DRQIN/INTRQIN and DRQOUT/INTRQOUT pins are multiplexed to provide two special “Request for Service” FIFO interrupt request lines when DMA is disabled. A FIFO Input or Output channel Request for Service interrupt is generated based upon the value programmed in the respective channel’s Threshold SFRs; Input Threshold (ITHR), and Output Threshold

(OTHR) SFRs. Additional interrupts are provided for FIFO Underrun and Overrun Errors, Data Stream Commands, and Immediate Commands. Table 4 lists the eight UPI-452 interrupt sources as they appear in the HSTAT SFR to the Host processor.

Table 4. UPI-452 to Host Interrupt Sources

HSTAT SFR Bit	Interrupt Source
HST7	Output FIFO Underrun Error
HST6	Immediate Command Out SFR Status
HST5	Data Stream Command/Data at Output FIFO Status
HST4	Output FIFO Request for Service Status
HST3	Input FIFO Overrun Error Condition
HST2	Immediate Command In SFR Status
HST1	FIFO DMA Freeze/Normal Mode Status
HST0	Input FIFO Request for Service

The interrupt response time required by the Host processor is application and system specific. In general, a typical sequence of Host interrupt response events and the approximate times associated with each are listed in Equation 1.

The example assumes the hardware configuration shown in Figure 1, iAPX 286/UPI-452 Block Diagram, with an 8259A Programmable Interrupt Controller. The timing analysis in Equation 1 also assumes the following; no other interrupt is either in process or pending, nor is the 286 in a LOCK condition. The current instruction completion time is 8 clock cycles (800 ns @ 10 MHz), or 4 bus cycles. The interrupt service routine first executes a PUSHA instruction, PUSH All General Registers, to save all iAPX 286 internal registers. This requires 19 clocks (or 2.0 μ s @ 10 MHz), or 10 bus cycles (rounded to complete bus cycle). The next service routine instruction reads the UPI-452 Host Status SFR to determine the interrupt source.

It is important to note that any UPI-452 INTRQ interrupt service routine should ALWAYS mask for the Freeze Mode bit first. This will insure that Freeze Mode always has the highest priority. This will also save the time required to mask for bits which are forced inactive during Freeze Mode, before checking the Freeze Mode bit. Access to the FIFO channels by the Host is inhibited during Freeze Mode. Freeze Mode is covered in more detail below.

To initiate the interrupt the UPI-452 activates the INTRQ output. The interrupt acknowledge sequence requires two bus cycles, 400 ns (10 MHz iAPX 286), for the two INTA pulse sequence.

Equation 1. Host Interrupt Response Time

Action	Time	Bus Cycles*
Current instruction execution completion	800 ns	4
INTA sequence	400 ns	2
Interrupt service routine (time to host first READ of UPI-452)	2000 ns	10
Total Interrupt Response Time	2.3 μs	16

NOTE:

10 MHz iAPX 286 bus cycle, 200 ns each

UPI-452 Internal

The internal CPU FIFO interrupt interface is essentially identical to that of the Host to the FIFO. Three internal interrupt sources support the FIFO operation; FIFO-Slave bus Interface Buffer, DMA Channel 0 and DMA Channel 1 Requests. These interrupts provide a maximum decoupling of the FIFO buffer and the internal CPU. The four different internal DMA Modes available add flexibility to the interface.

The FIFO-Slave Bus Interface interrupt response is also similar to the Host response to an INTRQ Request for Service interrupt. The internal CPU responds to the interrupt by reading the Slave Status (SSTAT) SFR to determine the source of the interrupt. This allows the user to prioritize the Slave Status flag response to meet the users application needs.

The internal interrupt response time is dependent on the current instruction execution, whether the interrupt is enabled, and the interrupt priority. In general, to finish execution of the current instruction, respond to the interrupt request, push the Program Counter (PC) and vector to the first instruction of the interrupt service routine requires from 38 to 86 oscillator periods (2.38 to 5.38 μ s @ 16 MHz). If the interrupt is due to an Immediate Command or DSC, additional time is required to read the Immediate Command or DSC SFR and vector to the appropriate service routine. This means two service routines back to back. One service routine to read the Slave Status (SSTAT) SFR to determine the source of the Request for Service interrupt, and second the service routine pointed to by the Immediate Command or DSC byte read from the respective SFR.

DMA

DMA is the fastest and most efficient way for the Host or internal CPU to communicate with the FIFO buffer. The UPI-452 provides support for both of these DMA paths. The two DMA paths and operations are fully independent of each other and can function simultaneously. While the Host DMA processor is performing a DMA transfer to or from the FIFO, the UPI-452 internal DMA processor can be doing the same.

Below are descriptions of both the Host and internal DMA operations. Both DMA paths can operate asynchronously and at different transfer rates. In order to make the most of this simultaneous asynchronous operation it is necessary to calculate the two transfer rates and accurately match their operations. Matching the different transfer rates is done by a combination of accurately programmed FIFO channel size and channel Thresholds. This provides the maximum Host and internal CPU to FIFO buffer interface decoupling. Below is a description of each of the two DMA operations and sample calculations for determining transfer rates. The next section of this application note, "Interface Latency", details the considerations involved in analyzing effective transfer rates when the overhead associated with each transfer is considered.

HOST FIFO DMA

DMA transfers between the Host and UPI-452 FIFO buffer are controlled by the Host CPU's DMA controller, and is independent of the UPI-452's internal two channel DMA processor. The UPI-452's internal DMA processor supports data transfers between the UPI-452 internal RAM, external RAM (via the Local Expansion Bus) and the various Special Function Registers including the FIFO Input and Output channel SFRs.

The maximum DMA transfer rate is achieved by the minimum DMA transfer cycle time to accomplish a source to destination move. The minimum Host UPI-452 FIFO DMA cycle time possible is determined by the READ and WRITE pulse widths, UPI-452 command recovery times in relation to the DMA transfer timing and DMA controller transfer mode used. Table 5 shows the relationship between the iAPX-286, iAPX-186 and UPI-452 for various DMA as well as non-DMA byte by byte transfer modes versus processor frequencies.

Host processor speed vs wait states required with UPI-452 running at 16 MHz:

Table 5. Host UPI-452 Data Transfer Performance

Processor & Speed	Wait States: Back to Back READ/WRITE's	DMA: Single Cycle	Two Cycle
iAPX-186* 8 MHz	0	N/A*	0
10 MHz	0	N/A*	0
12.5 MHz	1	N/A*	0
iAPX-286** 6 MHz	0	0	0
8 MHz	1	1	0
10 MHz	2	2	0

NOTES:

- * iAPX 186 On-chip DMA processor is two cycle operation only.
- ** iAPX 286 assumes 82258 ADMA (or other DMA) running 286 bus cycles at 286 clock rate.

In this application note system example, shown in Figure 1, DMA operation is assumed to be two bus cycle I/O to memory or memory to I/O. Two cycle DMA consists of a fetch bus cycle from the source and a store bus cycle to the destination. The data is stored in the DMA controller's registers before being sent to the destination. Single cycle DMA transfers involve a simultaneous fetch from the source and store to the destination. As the most common method of I/O-memory DMA operation, two cycle DMA transfers are the focus of this application note analysis. Equation 2 illustrates a calculation of the overall transfer rate between the FIFO buffer and external Host for a maximum FIFO size transfer. The equation does not account for the latency of initiating the DMA transfer.

Equation 2. Host FIFO DMA Transfer Rate—Input or Output Channel

$$\begin{aligned}
 &2 \text{ Cycle DMA Transfer-I/O (UPI-452) to System Memory} \\
 = & \text{FIFO channel size} * (\text{DMA READ/WRITE FIFO time} + \text{DMA WRITE/READ Memory Time}) \\
 = & 128 \text{ bytes} * (200 \text{ ns} + 200 \text{ ns}) \\
 = & 51.2 \mu\text{s} \\
 = & 256 \text{ bus cycles}^*
 \end{aligned}$$

NOTES:

- *10 MHz iAPX 286, 200 ns bus cycles.

The UPI-452 design is optimized for high speed DMA transfers between the Host and the FIFO buffer. The

UPI-452 internal FIFO buffer control logic provides the necessary synchronization of the external Host event and the internal CPU machine cycle during UPI-452 RD/WR accesses. This internal synchronization is addressed by the TCC AC specification of the UPI-452 shown in Figure 6. TCC is the time from the leading or trailing edge of a UPI-452 RD/WR to the same edge of the next UPI-452 RD/WR. The TCC time is effectively another way of measuring the system bus cycle time with reference to UPI-452 accesses.

In the iAPX-286 10 MHz system depicted in this application note the bus cycle time is 200 ns. Alternate cycle accesses of the UPI-452 during two cycle DMA operation yields a TCC time of 400 ns which is more than the TCC minimum time of 375 ns. Back to back Host UPI-452 READ/WRITE accesses may require wait states as shown in Table 5. The difference between 10 MHz iAPX-186 and 10 MHz iAPX 286 required wait states is due to the number of clock cycles in the respective bus cycle timings. The four clocks in a 10 MHz iAPX 186 bus cycle means a minimum TCC time of 400 ns versus 200 ns for a 10 MHz iAPX 286 with two clock cycle zero wait state bus cycle.

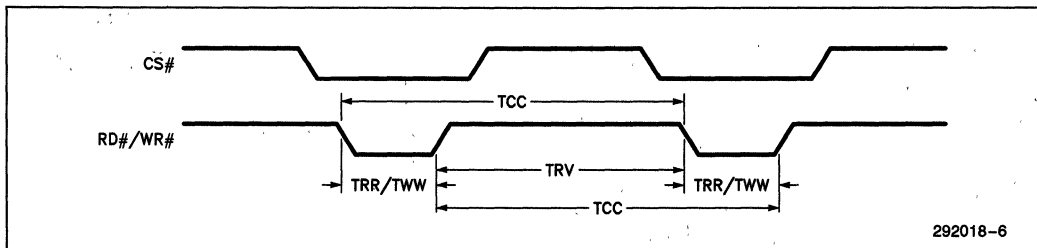
DMA handshaking between the Host DMA controller and the UPI-452 is supported by three pins on the UPI-452; DRQIN/INTRQIN, DRQOUT/INTRQOUT and DACK. The DRQIN/INTRQIN and DRQOUT/INTRQOUT outputs are two multiplexed DMA or interrupt request pins. The function of these pins is controlled by MODE SFR bit 6 (MD6). DRQIN and DRQOUT provide a direct interface to the Host system DMA controller (see Figure 1). In response to a DRQIN or DRQOUT request, the Host DMA controller initiates control of the system bus using HLD/HLDA. The FIFO Input or Output channel transfer is accomplished with a minimum of Host overhead and system bus bandwidth.

The third handshake signal pin is DACK which is used as a chip select during DMA data transfers. The UPI-452 Host READ and WRITE input signals select the respective Input and Output FIFO channel during DMA transfers. The CS and address lines provide DMA acknowledge for processors with onboard DMA controllers which do not generate a DACK signal.

The iAPX 286 Block I/O Instructions provide an alternative to two cycle DMA data transfers with approximately the same data rate. The String Input and Output instructions (INS & OUTS) when combined with the Repeat (REP) prefix, modifies INS and OUTS to provide a means of transferring blocks of data between I/O and Memory. The data transfer rate using REP INS/OUTS instructions is calculated in the same way as two cycle DMA transfer times. Each READ or WRITE would be 200 ns in a 10 MHz iAPX 286 system. The maximum transfer rate possible is 2.5 MBytes/second. The Block I/O FIFO data transfer calculation is the same as that shown in Equation 2 for two cycle DMA data transfers including TCC timing effects.

FIFO Data Structure and Host DMA

During a Host DMA write to the FIFO, if a DSC is to be written, the DMA transfer is stopped, the DSC is written and the DMA restarted. During a Host DMA read from the FIFO, if a DSC is loaded into the I/O Buffer Latch the DMA request, DRQOUT, will be deactivated (see Figure 2 above). The Host Status (HSTAT) SFR Data Stream Command bit is set and the INTRQ interrupt output goes active, if enabled. The Host responds to the interrupt as described above.



Symbol	Description	Var. Osc.	@ 16 MHz
TCC	Command Cycle Time	$6 * T_{cicl}$	375 ns min
.TRV	Command Recovery Time	75	75 ns min

Figure 6. UPI-452 Command Cycle Timing

Once INTRQ is deactivated and the DSC has been read by the Host, the DMA request, DRQOUT, is reasserted by the UPI-452. The DMA request then remains active until the transfer is complete or another DSC is loaded into the I/O Buffer Latch.

An Immediate Command written by the internal CPU during a Host DMA FIFO transfer also causes the Host Status flag and INTRQ to go active if enabled. In this case the Immediate Command would not terminate the DMA transfer unless terminated by the Host. The INTRQ line remains active until the Host reads the Host Status (HSTAT) SFR to determine the source of the interrupt.

The net effect of a Data Stream Command (DSC) on DMA data transfer rates is to add an additional factor to the data transfer rate equation. This added factor is shown in Equation 3. An Immediate Command has the same effect on the data transfer rate if the Immediate Command interrupt is recognized by the Host during a DMA transfer. If the DMA transfer is completed before the Immediate Command interrupt is recognized, the effect on the DMA transfer rate depends on whether the block being transmitted is larger than the FIFO channel size. If the block is larger than the programmed FIFO channel size the transfer rate depends on whether the Immediate Command flag or interrupt is recognized between partial block transfers.

The FIFO configuration shown in Equation 3 is arbitrary since there is no way of predicting the size relative to when a DSC would be loaded into the I/O Buffer Latch. The Host DMA rate shown is for a UPI-452

(Memory Mapped or I/O) to 286 System Memory transfer as described earlier. The equations do not account for the latency of initiating the DMA transfer.

Equation 3. Minimum host FIFO DMA Transfer Rate Including Data Stream Command(s)

$$\begin{aligned}
 &\text{Minimum Host/FIFO DMA Transfer Rate w/ DSC} \\
 &= \text{FIFO size} * \text{Host DMA 2 cycle time transfer rate} \\
 &\quad + \text{iAPX 286 interrupt response time (Eq. \# 1)} \\
 &= (32 \text{ bytes} * (200 \text{ ns} + 200 \text{ ns})) + 2.3 \mu\text{s} \\
 &= 15.1 \mu\text{s} \\
 &= 75.5 \text{ bus cycles (@10 MHz iAPX286, 200 ns bus cycle)}
 \end{aligned}$$

UPI-452 INTERNAL DMA PROCESSOR

The two identical internal DMA channels allow high speed data transfers from one UPI-452 writable memory space to another. The following UPI-452 memory spaces can be used with internal DMA channels:

- Internal Data Memory (RAM)
- External Data Memory (RAM)
- Special Function Registers (SFR)

The FIFO can be accessed during internal DMA operations by specifying the FIFO IN (FIN) SFR as the DMA Source Address (SAR) or the FIFO OUT (FOUT) SFR as the Destination Address (DAR). Table 6 lists the four types of internal DMA transfers and their respective timings.

Table 6. UPI-452 Internal DMA Controller Cycle Timings

Source	Destination	Machine Cycles**	@12 MHz	@16 MHz
Internal Data Mem. or SFR	Internal Data Mem. or SFR	1	1 μs	750 ns
Internal Data Mem. or SFR	External Data Mem.	1	1 μs	750 ns
External Data Mem.	Internal Data Mem. or SFR	1	1 μs	750 ns
*External Data Memory	External Data Memory	2	2 μs	1.5 μs

NOTES:

*External Data Memory DMA transfer applies to UPI-452 Local Bus only.

**MSC-51 Machine cycle = 12 clock cycles (TCLCL).

FIFO Data Structure and Internal DMA

The effect of Data Stream Commands and Immediate Commands on the internal DMA transfers is essentially the same as the effect on Host FIFO DMA transfers. Recognition also depends upon the programmed DMA Mode, the interrupts enabled, and their priorities. The net internal effect is the same for each possible internal case. The time required to respond to the Immediate or Data Stream Command is a function of the instruction time required. This must be calculated by the user based on the instruction cycle time given in the MSC-51 Instruction Set description in the Intel Microcontroller Handbook.

It is important to note that the internal DMA processor modes and the internal FIFO logic work together to automatically manage internal DMA transfers as data moves into and out of the FIFO. The two most appropriate internal DMA processor modes for the FIFO are FIFO Demand Mode and FIFO Alternate Cycle Mode. In FIFO Demand Mode, once the correct Slave Control and DMA Mode bits are set, the internal Input FIFO channel DMA transfer occurs whenever the Slave Control Input FIFO Request for Service flag is set. The DMA transfer continues until the flag is cleared or when the Input FIFO Read Pointer SFR (IRPR) equals zero. If data continues to be entered by the Host, the internal DMA continues until an internal interrupt of higher priority, if enabled, interrupts the DMA transfer, the internal DMA byte count reaches zero or until the Input FIFO Read Pointer equals zero. A complete description of interrupts and DMA Modes can be found in the UPI-452 Data Sheet.

DMA Modes

The internal DMA processor has four modes of operation. Each DMA channel is software programmable to operate in either Block Mode or Demand Mode. Demand Mode may be further programmed to operate in Burst or Alternate Cycle Mode. Burst Mode causes the internal processor to halt its execution and dedicate its resources exclusively to the DMA transfer. Alternate Cycle Mode causes DMA cycles and instruction cycles to occur alternately. A detailed description of each DMA Mode can be found in the UPI-452 Data Sheet.

INTERFACE LATENCY

The interface latency is the time required to accommodate all of the overhead associated with an individual data transfer. Data transfer rates between the Host system and UPI-452 FIFO, with a block size less than or equal to the programmed FIFO channel size, are calculated using the Host system DMA rate. (see Host DMA description above). In this case, the entire block could be transferred in one operation. The total latency is the time required to accomplish the block DMA transfer, the interrupt response or poll of the Host Status SFR response time, and the time required to initiate the Host DMA processor.

A DMA transfer between the Host and UPI-452 FIFO with a block size greater than the programmed FIFO channel size introduces additional overhead. This additional overhead is from three sources; first, is the time to actually perform the DMA transfer. Second, the overhead of initializing the DMA processor, third, the handshaking between each FIFO block required to transfer the entire data block. This could be time to wait for the FIFO to be emptied and/or the interrupt response time to restart the DMA transfer of the next portion of the block. A fourth component may also be the time required to respond to Underrun and Overrun FIFO Errors.

Table 7 shows six typical FIFO Input/Output channel sizes and the Host DMA transfers times for each. The timings shown reflect a 10 MHz system bus two cycle I/O to Memory DMA transfer rate of 2.5 MBytes/second as shown in Equation 1. The times given would be the same for iAPX 286 I/O block move instructions REP INS and REP OUTS as described earlier.

Table 7. Host DMA FIFO Data Transfer Times

FIFO Size:	32	43	64	85	96	128	bytes
Full or Empty	1/4	1/3	1/2	2/3	3/4	Full or Empty	
Time	12.8	17.2	25.6	34.0	38.4	51.2	μs

Table 8 shows six typical FIFO Input/Output channel sizes and the internal DMA processor data transfers times for each. The timings shown are for a UPI-452 single cycle Burst Mode transfer at 16 MHz or 750 ns per machine cycle in or out of the FIFO channels. The

machine cycle time is that of the MSC-51 CPU; 6 states, 2 XTAL2 clock cycles each or 12 clock cycles per machine cycle. Details on the MSC-51 machine cycle timings and operation may be found in the Intel Microcontroller Handbook.

Table 8. UPI-452 Internal DMA FIFO Data Transfer Times

FIFO Size:	32	43	64	85	96	128	bytes
Full or Empty	1/4	1/3	1/2	2/3	3/4	Full or Empty	
Time	24.0	32.3	48.0	64.6	72.0	96.0	μs

A larger than programmed FIFO channel size data block internal DMA transfer requires internal arbitration. The UPI-452 provides a variety of features which support arbitration between the two internal DMA channels and the FIFO. An example is the internal DMA processor FIFO Demand Mode described above. FIFO Demand Mode DMA transfers occur continuously until the Slave Status Request for Service Flag is deactivated. Demand Mode is especially useful for continuous data transfers requiring immediate attention. FIFO Alternate Cycle Mode provides for interleaving DMA transfers and instruction cycles to achieve a maximum of software flexibility. Both internal DMA channels can be used simultaneously to provide continuous transfer for both Input and Output FIFO channels.

Byte by byte transfers between the FIFO and internal CPU timing is a function of the specific instruction cycle time. Of the 111 MCS-51 instructions, 64 require 12 clock cycles, 45 require 24 clock cycles and 2 require 48 clock cycles. Most instructions involving SFRs are 24 clock cycles except accumulator (for example, MOV direct, A) or logical operations (ANL direct, A). Typical instruction and their timings are shown in Table 9.

Oscillator Period: @ 12 MHz = 83.3 ns
 @ 16 MHz = 62.5 ns

Table 9. Typical Instruction Cycle Timings

Instruction	Oscillator Periods	@ 12 MHz	@ 16 MHz
MOV direct†, A	12	1 μs	750 ns
MOV direct, direct	24	2 μs	1.5 μs

NOTE:

† Direct = 8-bit internal data locations address. This could be an Internal Data RAM location (0-255) or a SFR [i.e., I/O port, control register, etc.]

Byte by byte FIFO data transfers introduce an additional overhead factor not found in internal DMA operations. This factor is the FIFO block size to be transferred; the number of empty locations in the Output channel, or the number of bytes in the Input FIFO

channel. As described above in the FIFO Data Structure section, the block size would have to be determined by reading the channel read and write pointer and calculating the space available. Another alternative uses the FIFO Overrun and Underrun Error flags to manage the transfers by accepting error flags. In either case the instructions needed have a significant impact on the internal FIFO data transfer rate latency equation.

A typical effective internal FIFO channel transfer rate using internal DMA is shown in Equation 4. Equation 5 shows the latency using byte by byte transfers with an arbitrary factor added for internal CPU block size calculation. These two equations contrast the effective transfer rates when using internal DMA versus individual instructions to transfer 128 bytes. The effective transfer rate is approximately four times as fast using DMA versus using individual instructions (96 μs with DMA versus 492 μs non-DMA).

Equation 4. Effective Internal FIFO Transfer Time Using Internal DMA

$$\begin{aligned}
 &\text{Effective Internal FIFO Transfer Rate with DMA} \\
 &= \text{FIFO channel size} * \text{Internal DMA Burst Mode} \\
 &\quad \text{Single Cycle DMA Time} \\
 &= 128 \text{ Bytes} * 750 \text{ ns} \\
 &= 96 \mu\text{s}
 \end{aligned}$$

Equation 5. Effective FIFO Transfer Time Using Individual Instructions

$$\begin{aligned}
 &\text{Effective Internal FIFO Transfer Rate without DMA} \\
 &= \text{FIFO channel size} * \text{Instruction Cycle Time} + \\
 &\quad \text{Block size calculation Time} \\
 &= 128 \text{ Bytes} * (24 \text{ oscillator periods @ } 16 \text{ MHz}) + \\
 &\quad 20 \text{ instructions (24 oscillator period each} \\
 &\quad \text{@ } 16 \text{ MHz}) \\
 &= 128 * 1.5 \mu\text{s} + 300 \mu\text{s} \\
 &= 492 \mu\text{s}
 \end{aligned}$$

FIFO DMA FREEZE MODE INTERFACE

FIFO DMA Freeze Mode provides a means of locking the Host out of the FIFO Input and Output channels. FIFO DMA Freeze Mode can be invoked for a variety of reasons, for example, to reconfigure the UPI-452 Local Expansion Bus, or change the baud rate on the serial channel. The primary reason the FIFO DMA Freeze Mode is provided is to ensure that the Host does not read from or write to the FIFO while the FIFO interface is being altered. ONLY the internal CPU has the capability of altering the FIFO Special Function Registers, and these SFRs can ONLY be altered during FIFO DMA Freeze Mode. FIFO DMA Freeze Mode inhibits Host access of the FIFO while the internal CPU reconfigures the FIFO.

FIFO DMA Freeze Mode should not be arbitrarily invoked while the UPI-452 is in normal operation. Because the external CPU runs asynchronously to the internal CPU, invoking freeze mode without first properly resolving the FIFO Host interface may have serious consequences. Freeze Mode may be invoked only by the internal CPU.

The internal CPU invokes Freeze Mode by setting bit 3 of the Slave Control SFR (SC3). This automatically forces the Slave and Host Status SFR FIFO DMA Freeze Mode to In Progress (SSTAT SST5 = 0, HSTAT SFR HST1 = 1). INTRQ goes active, if enabled by MODE SFR bit 4, whenever FIFO DMA Freeze Mode is invoked to notify the Host. The Host reads the Host Status SFR to determine the source of the interrupt. INTRQ and the Slave and Host Status FIFO DMA Freeze Mode bits are reset by the Host READ of the Host Status SFR.

During FIFO DMA Freeze Mode the Host has access to the Host Status and Control SFRs. All other Host FIFO interface access is inhibited. Table 10 lists the FIFO DMA Freeze Mode status of all slave bus interface Special Function Registers. The internal DMA processor is disabled during FIFO DMA Freeze Mode and the internal CPU has write access to all of the FIFO control SFRs (Table 11).

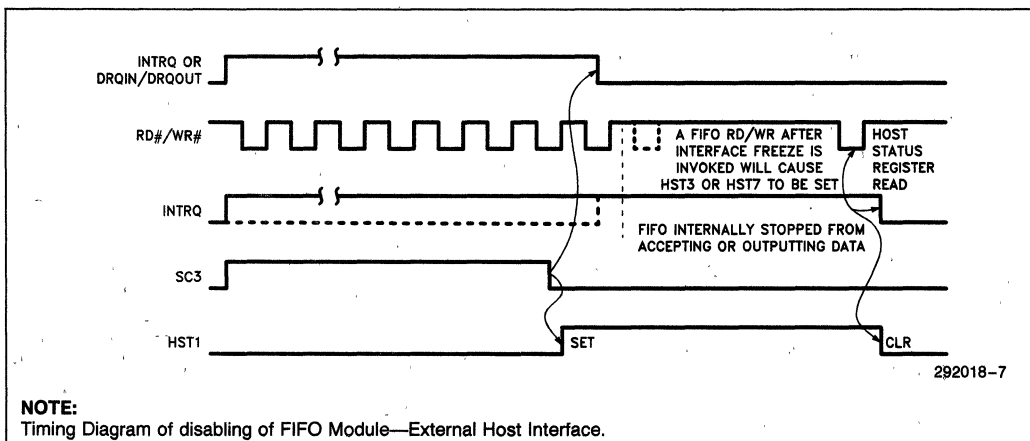
If FIFO DMA Freeze Mode is invoked without stopping the host, only the last two bytes of data written into or read from the FIFO will be valid. The timing diagram for disabling the FIFO module to the external Host interface is illustrated in Figure 7. Due to this synchronization sequence, the UPI-452 might not go into FIFO DMA Freeze Mode immediately after the Slave Control SFR FIFO 7 DMA Freeze Mode bit (SC3) is set = 0. A special bit in the Slave Status SFR (SST5) is provided to indicate the status of the FIFO DMA Freeze Mode. The FIFO DMA Freeze Mode

operations described in this section are only valid after SST5 is cleared.

Either the Host or internal CPU can request FIFO DMA Freeze Mode. The first step is to issue an Immediate Command indicating that FIFO DMA Freeze Mode will be invoked. Upon receiving the Immediate Command, the external CPU should complete servicing all pending interrupts and DMA requests, then send an Immediate Command back to the internal CPU acknowledging the FIFO DMA Freeze Mode request. After issuing the first Immediate Command, the internal CPU should not perform any action on the FIFO until FIFO DMA Freeze Mode is invoked. The hand-shaking is the same in reverse if the HOST CPU initiates FIFO DMA Freeze Mode.

After the slave bus interface is frozen, the internal CPU can perform the operations listed below on the FIFO Special Function Registers. These operations are allowed only during FIFO DMA Freeze Mode. Table 11 summarizes the characteristics of all the FIFO Special Function Registers during Normal and FIFO DMA Freeze Modes.

- | | |
|----------------------------|---|
| For FIFO Reconfiguration | <ol style="list-style-type: none"> 1. Changing the Channel Boundary Pointer SFR. 2. Changing the Input and Output Threshold SFR. |
| To Enhance the testability | <ol style="list-style-type: none"> 3. Writing to the read and write pointers of the Input and Output FIFO's. 4. Writing to and reading the Host Control SFRs. 5. Controlling some bits of Host and Slave Status SFRs. 6. Reading the Immediate Command Out SFR and Writing to the Immediate Command in SFR. |



NOTE:
Timing Diagram of disabling of FIFO Module—External Host Interface.

Figure 7. Disabling FIFO to Host Slave Interface Timing Diagram

The sequence of events for invoking FIFO DMA Freeze Mode are listed in Figure 8.

1. Immediate Command to request FIFO DMA Freeze Mode (interrupt)
2. Host/internal CPU interrupt response/service
3. Host/internal CPU clear/service all pending interrupts and FIFO data
4. Internal CPU sets Slave Control (SLCON) FIFO DMA
Freeze Mode bit = 0, FIFO DMA Freeze Mode, Host Status SFR FIFO DMA Freeze Mode Status bit = 1, INTRQ active (high)
5. Host **READ** Host Status SFR
6. Internal CPU reconfigures FIFO SFRs
7. Internal CPU resets Slave Control (SLCON) FIFO DMA
Freeze Mode bit = 1, Normal Mode, Host Status FIFO DMA Freeze Mode Status bit = 0.
8. Internal CPU issues Immediate Command to Host indicating that FIFO DMA Freeze Mode is complete
or
Host polls Host Status SFR FIFO DMA Freeze Mode bit to determine end of reconfiguration

Figure 8. Sequence of Events to Invoke FIFO DMA Freeze Mode

EXAMPLE CONFIGURATION

An example of the time required to reconfigure the FIFO 180 degrees, for example from 128 bytes Input to 128 bytes Output, is shown in Figure 9. The example approximates the time based on several assumptions;

1. The FIFO Input channel is full-128 bytes of data
2. Output FIFO channel is empty-1 byte
3. No Data Stream Commands in the FIFO.

4. The Immediate Command interrupt is responded to immediately—highest priority—by Host and internal CPU.
5. Respective interrupt response times
 - a. Host (Equation 3 above) = approximately 1.6 μ s
 - b. Internal CPU is 86 oscillator periods or approximately 5.38 μ s worst case.

Event	Time
Immediate Command from Host to UPI-452 to request FIFO DMA Freeze Mode (IAPX286 WRITE)	0.30 μ s
Internal CPU interrupt response/service	5.38 μ s
Internal CPU clears FIFO-128 bytes DMA	96.00 μ s
Internal CPU sets Slave Control Freeze Mode bit	0.75 μ s
Immediate Command to Host-Freeze Mode in progress Host Immediate Command interrupt response	2.3 μ s
Internal CPU reconfigures FIFO SFRs	
Channel Boundary Pointer SFR	0.75 μ s
Input Threshold SFR	0.75 μ s
Output Threshold SFR	0.75 μ s
Internal CPU resets Slave Control (SLCON) Freeze Mode bit = 1, Normal Mode, and automatically resets Host Status FIFO DMA Freeze Mode bit	2.3 μ s
Internal CPU writes Immediate Command Out	0.75 μ s
Host Immediate Command interrupt service	2.3 μ s
Total Minimum Time to Reconfigure FIFO	112.33 μ s

Figure 9. Sequence of Events to Invoke FIFO DMA Freeze Mode and Timings

Table 10. Slave Bus Interface Status During FIFO DMA Freezer Mode

DACK	CS	Interface Pins;			READ	WRITE	Operation In Normal Mode	Status In Freeze Mode
		A2	A1	A0				
1	0	0	1	0	0	1	Read Host Status SFR	Operational
1	0	0	1	1	0	1	Read Host Control SFR	Operational
1	0	0	1	1	1	0	Write Host Control SFR	Disabled
1	0	0	0	0	0	1	Data or DMA data from Output Channel	Disabled
1	0	0	0	0	1	0	Data or DMA data to Input Channel	Disabled
1	0	0	0	1	0	1	Data Stream Command from Output Channel	Disabled
1	0	0	0	1	1	0	Data Stream Command to Input Channel	Disabled
1	0	1	0	0	0	1	Read Immediate Command Out from Output Channel	Disabled
1	0	1	0	0	1	0	Write Immediate Command In to Input Channel	Disabled
0	X	X	X	X	0	1	DMA Data from Output Channel	Disabled
0	X	X	X	X	1	0	DMA Data to Input Channel	Disabled

NOTE:

X = don't care

Table 11. FIFO SFR's Characteristics During FIFO DMA Freeze Mode

Label	Name	Normal Operation (SST5 = 1)	Freeze Mode Operation (SST5 = 0)
HCON	Host Control	Not Accessible	Read & Write
HSTAT	Host Status	Read Only	Read & Write
SLCON	Slave Control	Read & Write	Read & Write
SSTAT	Slave Status	Read Only	Read & Write
IEP	Interrupt Enable & Priority	Read & Write	Read & Write
MODE	Mode Register	Read & Write	Read & Write
IWPR	Input FIFO Write Pointer	Read Only	Read & Write
IRPR	Input FIFO Read Pointer	Read Only	Read & Write
OWPR	Output FIFO Write Pointer	Read Only	Read & Write
ORPR	Output FIFO Read Pointer	Read Only	Read & Write
CBP	Channel Boundary Pointer	Read Only	Read & Write
IMIN	Immediate Command In	Read Only	Read & Write
IMONT	Immediate Command Out	Read & Write	Read & Write
FIN	FIFO IN	Read Only	Read Only
CIN	COMMAND IN	Read Only	Read Only
FOUT	FIFO OUT	Read & Write	Read & Write
COUT	COMMAND OUT	Read & Write	Read & Write
ITHR	Input FIFO Threshold	Read Only	Read & Write
OTHR	Other FIFO Threshold	Read Only	Read & Write



ICE™-42 8042 IN-CIRCUIT EMULATOR

- **Precise, Full-Speed, Real-Time Emulation**
 - Load, Drive, Timing Characteristics
 - Full-Speed Program RAM
 - Parallel Ports
 - Data Bus
- **User-Specified Breakpoints**
- **Execution Trace**
 - User-Specified Qualifier Registers
 - Conditional Trigger
 - Symbolic Groupings and Display
 - Instruction and Frame Modes
- **Emulation Timer**
- **Full Symbolic Debugging**
- **Single-Line Assembly and Disassembly for Program Instruction Changes**
- **Macro Commands and Conditional Block Constructs for Automated Debugging Sessions**
- **HELP Facility: ICE™-42 Command Syntax Reference at the Console**
- **User Confidence Test of ICE-42 Hardware**

The ICE-42 module resides in the Intel Microcomputer Development System and interfaces to any user-designed 8042 or 8041A system through a cable terminating in an 8042 emulator microprocessor and a pin-compatible plug. The emulator processor, together with 2K bytes of user program RAM located in the ICE-42 buffer box, replaces the 8042 device in the user system while maintaining the 8042 electrical and timing characteristics. Powerful Intel debugging functions are thus extended into the user system. Using the ICE-42 module, the designer can emulate the system's 8042 chip in real-time or single-step mode. Breakpoints allow the user to stop emulation on user-specified conditions, and a trace qualifier feature allows the conditional collection of 1000 frames of trace data. Using the single-line 8042 assembler the user may alter program memory using the 8042 assembler mnemonics and symbolic references, without leaving the emulator environment. Frequently used command sequences can be combined into compound commands and identified as macros with user-defined names.



210818-1

FUNCTIONAL DESCRIPTION

Integrated Hardware and Software Development

The ICE-42 emulator allows hardware and software development to proceed interactively. This approach is more effective than the traditional method of independent hardware and software development followed by system integration. With the ICE-42 module, prototype hardware can be added to the system as it is designed. Software and hardware integration occurs while the product is being developed. Figure 1 shows the ICE-42 emulator connected to a user prototype.

The ICE-42 emulator assists four stages of development.

SOFTWARE DEBUGGING

This emulator operates without being connected to the user's system before any of the user's hardware is available. In this stage ICE-42 debugging capabilities can be used in conjunction with the Intellec text editor and 8042 macro-assembler to facilitate program development.

HARDWARE DEVELOPMENT

The ICE-42 module's precise emulation characteristics and full-speed program RAM make it a valuable tool for debugging hardware.

SYSTEM INTEGRATION

Integration of software and hardware begins when any functional element of the user system hardware is connected to the 8042 socket. As each section of the user's hardware is completed, it is added to the prototype. Thus, each section of the hardware and software is "system" tested in real-time operation as it becomes available.

SYSTEM TEST

When the user's prototype is complete, it is tested with the final version of the user system software. The ICE-42 module is then used for real-time emulation of the 8042 chip to debug the system as a completed unit.

The final product verification test may be performed using the 8742 EPROM version of the 8042 micro-

computer. Thus, the ICE-42 module provides the ability to debug a prototype or production system at any stage in its development without introducing extraneous hardware or software test tools.

Symbolic Debugging

The ICE-42 emulator permits the user to define and to use symbolic, rather than absolute, references to program and data memory addresses. Thus, there is no need to recall or look up the addresses of key locations in the program, or to become involved with machine code.

When a symbol is used for memory reference in an ICE-42 emulator command, the emulator supplies the corresponding location as stored in the ICE-42 emulator symbol table. This table can be loaded with the symbol table produced by the assembler during application program assembly. The user obtains the symbol table during software preparation simply by using the "DEBUG" switch in the 8042 macro-assembler. Furthermore, the user interactively modifies the emulator symbol table by adding new symbols or changing or deleting old ones. This feature provides great flexibility in debugging and minimizes the need to work with hexadecimal values.

Through symbolic references in combination with other features of the emulator, the user can easily:

- Interpret the results of emulation activity collected during trace.
- Disassemble program memory to mnemonics, or assemble mnemonic instructions to executable code.
- Reference labels or addresses defined in a user program.

Automated Debugging and Testing

MACRO COMMAND

A macro is a set of commands given a name. A group of commands executed frequently can be defined as a macro. The user executes the group of commands by typing a colon followed by the macro name. Up to ten parameters may be passed to the macro.

Macro commands can be defined at the beginning of a debug session and then used throughout the whole session. One or more macro definitions can be saved on diskette for later use. The Intellec text editor may be used to edit the macro file. The macro definitions are easy to include in any later emulation session.

The power of the development system can be applied to manufacturing testing as well as development by writing test sequences as macros. The macros are stored on diskettes for use during system test.

COMPOUND COMMAND

Compound commands provide conditional execution of commands (IF command) and execution of commands repeatedly until certain conditions are met (COUNT, REPEAT commands).

Compound commands may be nested any number of times, and may be used in macro commands.

Example:

```
*DEFINE .I=0      ;Define symbol .I to 0
*COUNT 100H     ;Repeat the following
                  ;commands 100H times.
.*IF .I AND 1 THEN ;Check if .I is odd
..*CBYTE.I=.I    ;Fill the memory at
                  ;location .I to value .I

..*END

.*.I-.I+1        ;Increment .I by 1.
.*END            ;Command executes
                  ;upon carriage-return
                  ;after END
```

(The asterisks are system prompts; the dots indicate the nesting level of compound commands.)

Operating Modes

The ICE-42 software is an Intellec RAM-based program that provides easy-to-use commands for initiating emulation, defining breakpoints, controlling trace data collection, and displaying and controlling system parameters. ICE-42 commands are configured with a broad range of modifiers that provide maximum flexibility in describing the operation to be performed.

EMULATION

The ICE-42 module can emulate the operation of prototype 8042 system, at real-time speed (up to 12 MHz) or in single steps. Emulation commands to the ICE-42 module control the process of setting up, running, and halting an emulation of the user's 8042-based system. Breakpoints and tracepoints enable the ICE-42 emulator to halt emulation and provide a detailed trace of execution in any part of the user's program. A summary of the emulation commands is shown in Table 1.

Table 1. Major Emulation Commands

Command	Description
GO	Begins real-time emulation and optionally specifies break conditions.
BR0, BR1, BR	Sets or displays either or both Breakpoint Registers used for stopping real-time emulation.
STEP	Performs single-step emulation.
QR0, QR1	Specifies match conditions for qualified trace.
TR	Specifies or displays trace-data collection conditions and optionally sets Qualifier Register (QR0, QR1).
Synchronization Line Commands	Sets and displays status of synchronization line outputs or latched inputs. Used to allow real-time emulation or trace to start and stop synchronously with external events.

Breakpoints

The ICE-42 hardware includes two breakpoint registers that allow halting of emulation when specified conditions are met. The emulator continuously compares the values stored in the breakpoint registers with the status of specified address, opcode, operand, or port values, and halts emulation when this comparison is satisfied. When an instruction initiates a break, that instruction is executed completely before the break takes place. The ICE-42 emulator then regains control of the console and enters the interrogation mode. With the breakpoint feature, the user can request an emulation break when the program:

- Executes an instruction at a specific address or within a range of addresses.
- Executes a particular opcode.
- Receives a specific signal on a port pin.
- Fetches a particular operand from the user program memory
- Fetches an operand from a specific address in program memory.

Trace and Tracepoints

Tracing is used with real-time and single-step emulation to record diagnostic information in the trace buffer as a program is executed. The information collected includes opcodes executed, port values, and memory addresses. The ICE-42 emulator collects 1000 frames of trace data.

If desired this information can be displayed as assembler instruction mnemonics for analysis during interrogation or single-step mode. The trace-collection facility may be set to run conditionally or unconditionally. Two unique trace qualifier registers, specified in the same way as breakpoint registers, govern

conditional trace activity. The qualifiers can be used to condition trace data collection to take place as follows:

- Under all conditions (forever).
- Only while the trace qualifier is satisfied.
- For the frames or instructions preceding the time when a trace qualifier is first satisfied (pre-trigger trace).
- For the frames or instructions after a trace qualifier is first satisfied (post-triggered trace).

Table 2 shows an example of trace display.

Table 2. Trace Display (Instruction Mode)

FRAME	LOC	OBJ	INSTRUCTION	P1	P2	TD	T1	DBVIN	YOUT	YSTS	TOVF
0000:	100H	2355	MOV A,#55H	FFH	FFH	0	0	66H	DFH	02H	0
0004:	102H	39	OUTL P1,A	FFH	FFH	0	0	66H	DFH	02H	0
0008:	103H	3A	OUTL P2,A	55H	FFH	0	0	66H	DFH	02H	0
0012:	104H	22	IN A,DBB	55H	55H	0	0	66H		02H	0
0014:	105H	37	CPL A	55H	55H	0	0		DFH	02H	0
0016:	106H	02	OUT DBB,A	55H	55H	0	0	66H		00H	0
0018:	107H	8A03	MOV R2,#03H	55H	55H	0	0	66H	99H	00H	0
0022:	109H	8840	MOV R0,#.TABLE0	55H	55H	0	0	66H	99H	01H	0
0026:	10BH	8960	MOV R1,#.TABLE1	55H	55H	0	0	66H	99H	01H	0
.LOOP											
0030:	10DH	FD	MOV A,@RD	55H	55H	0	0		99H	01H	0
0032:	10EH	A1	MOV @R1,A	55H	55H	0	0	66H		01H	0
0034:	10FH	18	INC R0	55H	55H	0	0		99H	01H	0
0036:	110H	19	INC R1	55H	55H	0	0	66H		01H	0
0038:	111H	EADD	DJNZ R2,.LOOP	55H	55H	0	0	66H	99H	01H	0
.LOOP											
0042:	10DH	FD	MOV A,@RD	55H	55H	0	0		99H	01H	0
0044:	10EH	A1	MOV @R1,A	55H	55H	0	0	66H		01H	0
0046:	10FH	18	INC R0	55H	55H	0	0		99H	01H	0
0048:	110H	19	INC R1	55H	55H	0	0	66H		01H	0
0050:	111H	EADD	DJNZ R2,.LOOP	55H	55H	0	0	66H	99H	01H	0
.LOOP											
0054:	10DH	FD	MOV A,@RD	55H	55H	0	0		99H	01H	0
0056:	10EH	A1	MOV @R1,A	55H	55H	0	0	66H		01H	0
0058:	10FH	18	INC R0	55H	55H	0	0		99H	01H	0
0060:	110H	19	INC R1	55H	55H	0	0	66H		01H	0
0062:	111H	EADD	DJNZ R2,.LOOP	55H	55H	0	0	66H	99H	01H	0
0066:	113H	00	NOP	55H	55H	0	0		99H	01H	0

210818-2

INTERROGATION AND UTILITY

Interrogation and utility commands give convenient access to detailed information about the user program and the state of the 8042 that is useful in debugging hardware and software. Changes can be made in memory and in the 8042 registers, flags, and port values. Commands are also provided for various utility operations such as loading and saving program files, defining symbols, displaying trace data, controlling system synchronization and returning control to ISIS-II. A summary of the basic interrogation and utility commands is shown in Table 3. Two additional time-saving emulator features are discussed below.

Single-Line Assembler/Disassembler

The single-line assembler/disassembler (ASM and DASM commands) permits the designer to examine and alter program memory using assembly language mnemonics, without leaving the emulator environment or requiring time-consuming program reassembly. When assembling new mnemonic instructions into program memory, previously defined symbolic references (from the original program assembly, or subsequently defined during the emulation session)

Table 3. Major Interrogation and Utility Commands

Command	Description
HELP	Displays help messages for ICE-42 emulator command-entry assistance.
LOAD	Loads user object program (8042 code) into user-program memory, and user symbols into ICE-42 emulator symbol table.
SAVE	Saves ICE-42 emulator symbol table and/or user object program in ISIS-II hexadecimal file.
LIST	Copies all emulator console input and output to ISIS-II file.
EXIT	Terminates ICE-42 emulator operation.
DEFINE	Defines ICE-42 emulator symbol or macro.
REMOVE	Removes ICE-42 emulator symbol or macro.
ASM	Assembles mnemonic instructions into user-program memory.
DASM	Disassembles and displays user-program memory contents.
Change/Display Commands	Change or display value of symbolic reference in ICE-42 emulator symbol table, contents of key-word references (including registers, I/O ports, and status flags), or memory references.
EVALUATE	Evaluates expression and displays resulting value.
MACRO	Displays ICE-42 macro or macros.
INTERRUPT	Displays contents for the Data Bus and timer interrupt registers.
SECONDS	Displays contents of emulation timer, in microseconds.
Trace Commands	Position trace buffer pointer and select format for trace display.
PRINT	Displays trace data pointed to by trace buffer pointer.
MODE	Sets or displays the emulation mode, 8041A or 8042.

Table 4. HELP Command

```

*HELP
Help is available for the following items. Type HELP followed by the item name.
The help items cannot be abbreviated. (For more information, type HELP HELP or
HELP INFO.)
Emulation:      Trace Collection:      Misc:          <address>
GO GR SYD      TR  QR  QRD  QRL  SYL  BASE          <CPU#keyword>
BR BROBR1      DISABLE          <expr>
STEP           Trace Display:  ENABLE        <ICE42 #keyword>
              TRACE  MOVE  PRINT        ERROR        <identifier>
              OLDEST NEWEST  EVALUATE     <instruction>
              HELP          <masked#constant>
              INFO         <match#cond>
Change/        Display/ Define/ Remove:
<CHANGE>      REMOVE  CBYTE  <LIGHTS>
<DISPLAY>    SYMBOL  DBYTE  DASM  LIST
REGISTER     RESET   ASM    LOAD  <string>
              WRITE
SECONDS      STACK          SY  SAVE    <string#constant>
DEFINE       STACK          SY  SUFFIX  <symbol#ref>
              SYMBOLIC <mode>
Macro:        Compound
DEFINE        DIR    Commands:
DISABLE       ENABLE COUNT
INCLUDE       PUT   IF
<MACRO#DISPLAY> REPEAT
<MACRO#INVOCATION>
*
*
*HELP IF
IF - The conditional command allows conditional execution of one or more commands
based on the values of boolean conditions.
    IF <expr> 'THEN <cr>          <true#list> ::= '<command> <cr> @
    <true#list>                  <false#list> ::= '<command> <cr> @
    'ORIF <expr> <cr>          <command> ::= An ICE-42 command.
    <true#list> @
    'ELSE <cr>
    <false#list>
    END
The <expr>s are evaluated in order as 16-bit unsigned integers. If one is
reached whose value has low-order bit 1 (TRUE), all commands in the <true#list>
following that <expr> are then executed and all commands in the other <true#list>
and in the <false#list> are skipped. If all <expr>s have value with low-
order bit 0 (FALSE), then all commands in all <true#list>s are skipped and, if
ELSE is present, all commands in the <false#list> are executed.
    (EX: IF .LOOP=5 THEN
          STEP
          ELSE
          GO
          END)
*
*
*
*EXIT
    
```

210818-3

may be used in the instruction operand field. The emulator supplies the absolute address or data values as stored in the emulator symbol table. These features eliminate user time spent translating to and from machine code and searching for absolute addresses, with a corresponding reduction in transcription errors.

HELP

The HELP file allows display of ICE-42 command syntax information at the Intellec console. By typing "HELP"; a listing of all items for which help messages are available is displayed. Typing "HELP <Item>" then displays relevant information about the item requested, including typical usage examples. Table 4 shows some sample HELP messages.

EMULATION ACCURACY

The speed and interface demands of a high-performance single-chip microcomputer require extremely accurate emulation, including full-speed, real-time operation with the full function of the microcomputer. The ICE-42 module achieves accurate emulation with an 8042 emulator chip, a special configuration of the 8042 microcomputer family, as its emulation processor.

Each of the 40 pins on the user plug is connected directly to the corresponding 8042 pin on the emulator chip. Thus the user system sees the emulator as an 8042 microcomputer at the 8042 socket. The resulting characteristics provide extremely accurate emulation of the 8042 including speed, timing characteristics, load and drive values, and crystal operation. However, the emulator may draw more power from the user system than a standard 8042 family device.

Additional emulator processor pins provide signals such as internal address, data, clock, and control lines to the emulator buffer box. These signals let static RAM in the buffer box substitute for on-chip program ROM or EPROM. The emulator chip also gives the ICE module "back-door" access to internal chip operation, allowing the emulator to break and trace execution without interfering with the values on the user-system pins.



210818-4

Figure 1. A typical 8042 Development Configuration. The host system is an Intellec Series IV. The ICE-42 module is connected to a user prototype system.

SPECIFICATIONS

ICETM-42 Operating Requirements

Intellec Model 800, Series II, Series III, or Series IV Microcomputer Development System (64K RAM required)

System console (Model 800 only)

Intellec Diskette Operating System: ISIS (Version 3.4 or later).

Equipment Supplied

- Printed circuit boards (2)
- Emulation buffer box, Intellec interface cables, and user-interface cable with 8042 emulation processor
- Crystal power accessory
- Operating instructions manuals
- Diskette-based ICE-42 software (single and double density)

Emulation Clock

User's system clock (up to 12 MHz) or ICE-42 crystal power accessory (12 MHz)

Environmental Characteristics

Operating Temperature: 0°C to 40°C
Operating Humidity: Up to 95% relative humidity without condensation.

Physical Characteristics

Printed Circuit Boards

Width: 12.00 in. (30.48 cm)
Height: 6.75 in. (17.15 cm)
Depth: 0.50 in. (1.27 cm)

Buffer Box

Width: 8.00 in. (20.32 cm)
Length: 12.00 in. (30.48 cm)
Depth: 1.75 in. (4.44 cm)
Weight: 4.0 lb (1.81 kg)

Electrical Characteristics

DC Power Requirements (from Intellec® system)

$V_{CC} = +5V, \pm 5\%$
 $I_{CC} = 13.2A \text{ max}; 11.0A \text{ typical}$
 $V_{DD} = +12V, \pm 5\%$
 $I_{DD} = 0.1A \text{ max}; 0.05A \text{ typical}$
 $V_{BB} = -10V, \pm 5\%$
 $I_{BB} = 0.05A \text{ max}; 0.01A \text{ typical}$

User plug characteristics at 8042 socket—Same as 8042 or 8742 except that the user system sees an added load of 25 pF capacitance and 50 μA leakage from the ICE-42 emulator user plug at ports 1, 2, T0, and T1.

ORDERING INFORMATION

Part Number	Description
-------------	-------------

ICE-42	8042 Microcontroller In-Circuit Emulator, cable assembly and interactive diskette software
--------	--



iUP-200A/iUP-201A UNIVERSAL PROM PROGRAMMERS

MAJOR IUP-200A/IUP-201A FEATURES:

- **Personality Module Plug-Ins Provide Industry First Support for Intel and Intel Compatible EPROMs, EEPROMs, KEPROM, Microcontrollers, and other Programmable Devices.**
- **Powerful PROM Programming Software (IPPS) Makes Programming Easy with Inteltec® Development System, INDS-II Networks, iPDS Personal Development System, IBM P/C, X/T, A/T, and PC DOS Compatibles.**
- **New Modules Provide Industry-Fastest, Intelligent Programming Algorithms to Dramatically Shorten Programming Times.**
- **IUP-200A Provides On-Line Operation with a Built-In Serial RS232 Interface and Software for a Growing List of Environments.**
- **IUP-201A Provides Same On-Line Performance and Adds Keyboard and Display for Stand-Alone Use.**
- **iUP-201A Stand-Alone Capability Includes Device Previewing, Editing, Duplication, and Download from any Source Over RS232C Port.**
- **Regular Updates and Add-Ons Have Maintained Even the Earliest IUP-200 and iUP-201 Users at the State-of-Art.**

The iUP-200A and iUP-201A universal programmers program and verify data in all the Intel and Intel compatible, programmable devices (EPROMs and EEPROMs). They can also program the EPROM memory portions of Intel's single-chip microcomputer and peripheral devices. The iUP-200A and iUP-201A universal programmers provide on-line programming and verification in a growing variety of development environments using the Intel PROM programming software (IPPS). In addition, the iUP-201A universal programmer supports off-line, stand-alone program editing, EPROM duplication, and EPROM memory locking. The iUP-200A universal programmer is expandable to an iUP-201A model.



210319-1

FUNCTIONAL DESCRIPTION

The iUP-200A universal programmer operates in on-line mode. The iUP-201A universal programmer operates in both on-line and off-line mode.

On-Line System Hardware

The iUP-200A and iUP-201A universal programmers are free-standing units that, when connected to a host computer with at least 64K bytes of memory, provide on-line EPROM programming and verification of Intel programmable devices. In addition, the universal programmer can read the contents of the ROM versions of these devices.

The universal programmer communicates with the host through a standard RS232C serial data link. Different versions of the iUP-200A and iUP-201A are equipped with different cables, including the cable most commonly used for interfacing to that host. Care should be taken that the version with the correct cable for your particular system is selected, as cable requirements can vary with your host configuration. A serial converter is needed when using the MDS 800 as a host system. (Serial converters are available from other manufacturers).

Each universal programmer contains the CPU, selectable power supply, static RAM, programmable timer, interface for personality modules, RS232C interface for the host system, and control firmware in EPROM. The iUP-201A also has a keyboard and display.

A personality module adapts the universal programmer to a family of EPROM devices; it contains all the hardware and firmware necessary to program either a family of Intel EPROMs or a single Intel device. The user inserts the personality module into the universal programmer front panel.

Figure 1 shows the iUP-200A on-line system configurations, and Figure 2 shows the on-line system data flow.

On-Line System Software

The iUP-200A and iUP201A includes your choice of one copy of Intel's PROM Programming software iPPS, selected from a growing list of versions for different operating systems and hosts. Each version includes the software implementation designed for that host and O.S. and the RS232C cable most commonly used. Additional versions may be purchased separately if you decide to change hosts at a later date. The iPPS software provides user control

through an easy-to-use interactive interface. The iPPS software performs the following functions to make EPROM programming quick and easy:

- Reads EPROMs and ROMs
- Programs EPROMs directly or from a file
- Verifies EPROM data with buffer data
- Locks EPROM memory from unauthorized access (on devices which support this feature)
- Prints EPROM contents on the network or development system printer
- Performs interactive formatting operations such as interleaving, nibble swapping, bit reversal, and block moves
- Programs multiple EPROMs from the source file, prompting the user to insert new EPROMs
- Uses a buffer to change EPROM contents

All iPPS commands, as well as program address and data information, are entered through the host system ASCII keyboard and displayed on the system CRT. Table 1 summarizes the iPPS commands.

The iPPS software lets the user load programs into an EPROM from host system memory or directly from a disk file. Access to the disk lets the user create and manipulate data in a virtual buffer with an address range up to 16M. This large block of data can be formatted into different EPROM word sizes for program storage into several different EPROM types. In addition, a program stored in the target EPROM, the host system memory, or a system disk file can be interleaved with a second program and entered into a specific target EPROM or EPROMs.

The iPPS software supports data manipulation in any Intel format: 8080 hexadecimal ASCII, 8080 absolute object, 8086 hexadecimal ASCII, 8086 absolute object, and 80286 absolute object. Addresses and data can be displayed in binary, octal, decimal, or hexadecimal. The user can easily change default data formats as well as number bases.

The user invokes the iPPS software from the operating system. Inteltec and iPDS Development Systems running ISIS allow running the software under control of ISIS submit files freeing the operator from repetitious command entry.

System Expansion

The iUP-200A universal programmer can be easily upgraded (by the user) to an iUP-201A universal programmer for off-line operation. The upgrade kit (IUP-PAK-A) is available from Intel or your local Intel distributor.

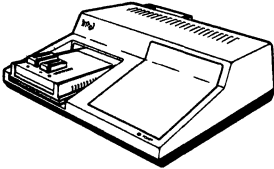
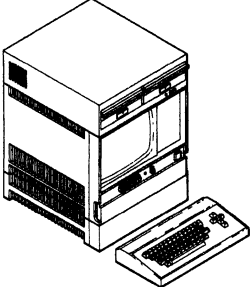
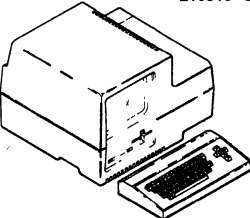


MODEL NUMBER	HOST SUPPORTED	S/W OPERATING SYSTEM ENVIRONMENT	RS232C CABLE INCLUDED*	I/O PORT USED
	 <p>210319-2</p>			
IUP200A211A OR IUP200A212B	 <p>210319-3</p>	ISIS II	INTELLEC-STYLE	CH1 OR 2
IUP200A213C	 <p>210319-4</p>	INDX	INTELLEC-STYLE	CH1 OR 2
IUP200A216D	 <p>210319-5</p>	PC-DOS	PC OR XT STYLE	COM 1 OR 2
IUP200A217D	 <p>210319-6</p>	PC-DOC	PC AT STYLE	COM 1 OR 2
<p>*RS232C CABLES: INTELLEC STYLE—DB 25 PIN MALE TO DB 25 PIN MALE, NULL MODEM CABLE. PC XT STYLE — DB 25 PIN FEMALE TO DB 25 PIN MALE, NULL MODEM CABLE. PC AT STYLE — DB 9 PIN FEMALE TO DB 25 PIN MALE CABLE.</p>				

Figure 1. On-Line System Configurations

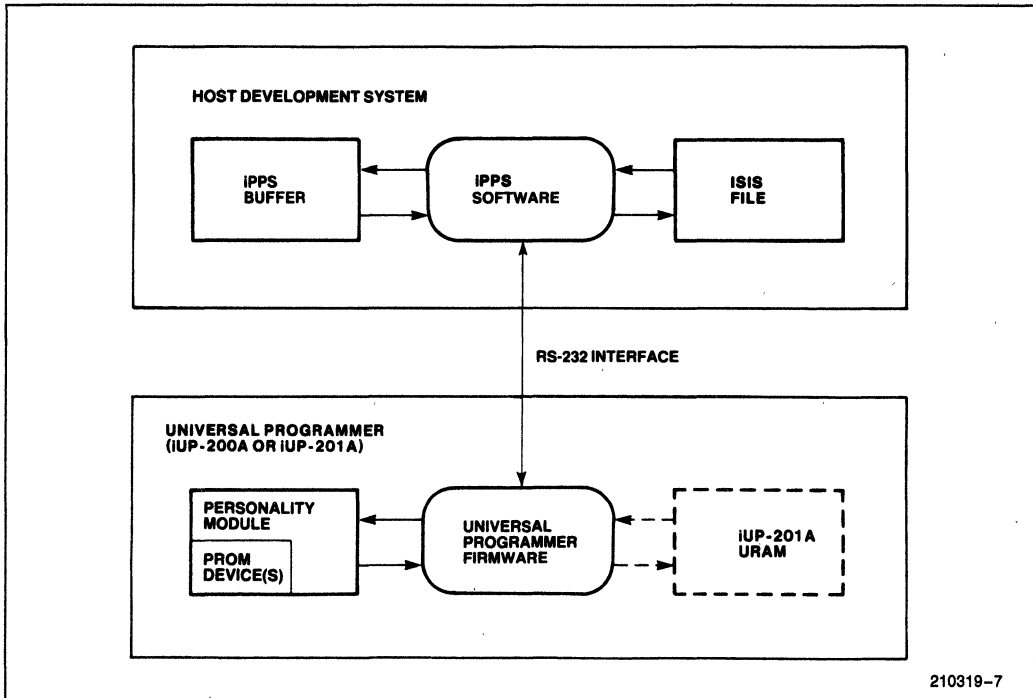


Figure 2. On-Line System Data Flow

Off-Line System

The iUP-201A universal programmer has all the on-line features of the iUP-200A universal programmer plus off-line editing, EPROM duplication, program verification, and locking of EPROM memory independent of the host system. The iUP-201A universal programmer also accepts Intel hexadecimal programs developed on non-Intel development systems. Just a few keystrokes download the program into the iUP RAM for editing and loading into a EPROM.

Off-line commands are entered using the off-line command keys summarized in Table 2.

In addition to the hardware components included as part of the iUP-200A, the iUP-201A contains a 24-character alphanumeric display, full hexadecimal 12-function keypad, and 32K bytes of iUP RAM. Figure 3 illustrates the iUP-201A keyboard and display.

The two logical devices accessible during off-line operation are the EPROM device and the iUP RAM. A typical operation is copying the data from an EPROM (or ROM) into the iUP RAM, modifying this data in iUP RAM, and programming the modified data back into a EPROM device. The address range

of the iUP RAM is automatically determined by the universal programmer when EPROM type selection is made. Figure 4 shows the off-line system data flow.

SYSTEM DIAGNOSTICS

Both the iUP-200A and iUP-201A universal programmers include self-contained system diagnostics that verify system operation and aid the user in fault isolation. Diagnostics are performed on the power supply, CPU internal firmware ROM, internal RAM, timer, the iUP-201A keyboard, and the iUP RAM. In addition, tests are made on any personality module installed in the programmer the first time the module is accessed. The personality module tests include the power select circuitry and module firmware. Straight-forward messages are provided on the development system display in on-line mode and on the iUP-201A display in off-line mode.

PERSONALITY MODULES

A personality module is the interface between the iUP-200A/iUP-201A universal programmer (or an iPDS system) and a selected EPROM (or ROM). Personality modules contain all the hardware and


















Table 1. IPPS Command Summary

Command	Description
PROGRAM CONTROL GROUP EXIT <ESC> REPEAT ALTER	CONTROLS EXECUTION OF THE IPPS SOFTWARE. Exits the IPPS software and returns control to the ISIS operating system. Terminates the current command. Repeats the previous command. Edits and re-executes the previous command.
UTILITY GROUP DISPLAY PRINT QUEUE HELP MAP BLANKCHECK OVERLAY TYPE INITIALIZE WORKFILES	DISPLAYS USER INFORMATION AND STATUS AND SETS DEFAULT VALUES. Displays EPROM, buffer, or file data on the console. Prints EPROM, buffer, or file data on the local printer. Prints EPROM, buffer or file data on the network spooled printer. Displays user assistance information. Displays buffer structure and status. Checks for unprogrammed EPROMs. Checks whether non-blank EPROMs can be programmed. Selects the EPROM type. Initializes the default number base and file type. Specifies the drive device for temporary work files.
BUFFER GROUP SUBSTITUTE LOADDATA VERIFY	EDITS, MODIFIES, AND VERIFIES DATA IN THE BUFFER. Examines and modifies buffer data. Loads a section of the buffer with a constant. Verifies data in the EPROM with buffer data.
FORMATTING GROUP FORMAT	REARRANGES DATA FROM THE EPROM, BUFFER, OR FILE. Formats and interleaves buffer, EPROM, or file data.
COPY GROUP COPY (file to PROM) COPY (PROM to file) COPY (buffer to PROM) COPY (PROM to buffer) COPY (buffer to file) COPY (file to buffer) COPY (file to URAM) COPY (URAM to file) COPY (buffer to URAM) COPY (URAM to buffer)	COPIES DATA FROM ONE DEVICE TO ANOTHER. Programs the EPROM with data in a file on disk. Saves EPROM data in a file on disk. Programs the EPROM with data from the buffer. Loads the buffer with data in the EPROM. Saves the contents of the buffer in a file on disk. Loads the buffer from a file on disk. Loads file data into the iUP RAM (iUP-201A model only). Saves iUP URAM data in a file on disk (iUP-201A model only). Loads the buffer into the iUP URAM (iUP-201A model only). Loads iUP URAM data into the buffer (iUP-201A model only).
SECURITY GROUP KEYLOCK	LOCKS SELECTED DEVICES TO PREVENT UNAUTHORIZED ACCESS. Locks the EPROM from unauthorized access.

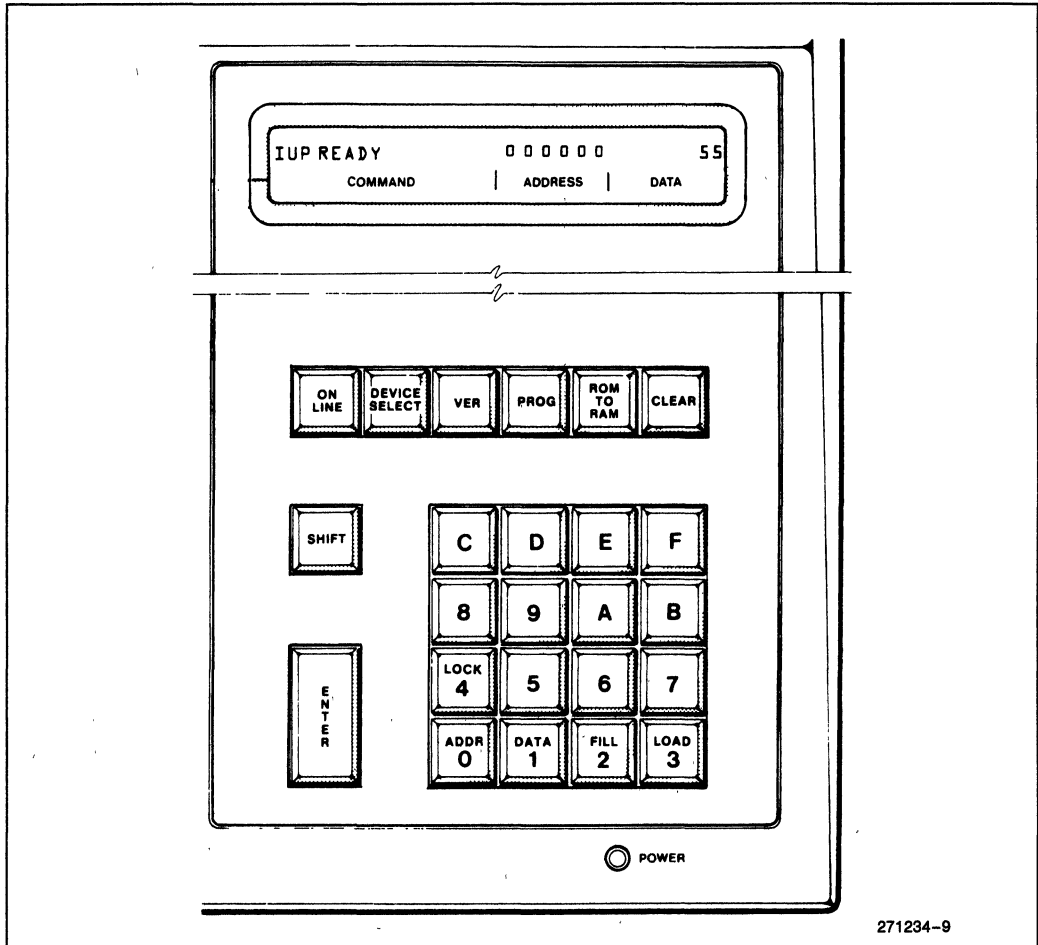
firmware for reading and programming a family of Intel devices. Each personality module is a single molded unit inserted into the front panel of the universal programmer. A wide variety of personality modules and adaptors are available for personal programmable devices. New modules and adaptors allow you to keep abreast of the newest Intel devices, programming algorithms, and device packages while protecting your equipment investment. Refer to the data sheet on "PROM Programming Personality Modules" for a complete list of available support.

Each personality module connects to the universal programmer through a 41-pin connector. Module firmware is uploaded into the iUP RAM and executed by the internal processor. The personality module firmware contains routines necessary to read and program a family of EPROMs. In addition, the personality module sends specific information about the selected EPROM to the universal programmer to help perform EPROM device integrity checks.

Table 2. Off-Line Command Keys Summary

Key	Function	
	<p>Selects either on-line or off-line operation. When on-line, all other function keys are disabled.</p>	
	<p>Selects the EPROM type when using a personality module able to program multiple EPROM devices.</p>	
	<p>Verifies the contents of the installed EPROM device with the contents of the iUP RAM. The universal programmer display indicates the address and the XOR of any mismatches.</p>	
	<p>Performs a device blank check and then programs the target EPROM with data from the iUP RAM. If the blank check fails, pressing PROG again performs an overlay check to verify that non-blank EPROMs can be programmed.</p>	
	<p>Loads the iUP RAM with the data from the EPROM device installed in the personality module.</p>	
	<p>Terminates the current off-line function, clears a user entry, or restores the display after an error.</p>	
	<p>Transfers information from the universal programmer display (addresses, data, or baud rate) into the iUP RAM.</p>	
		<p>Selects an address field for display.</p>
		<p>Selects a data field for keypad editing and entry.</p>
		<p>Loads a contiguous section of iUP RAM locations with a constant.</p>
		<p>Downloads Intel hexadecimal data from any development system which has an RS232C port.</p>
		<p>Locks a EPROM from unauthorized access.</p>

210319-8



271234-9

Figure 3. IUP-201A Keyboard and Display

LEDs on each personality module indicate operational status. On some personality modules a column of LEDs indicate which EPROM device type the user has selected. On some personality modules an LED below the socket indicates which socket is to be used. A red indicator light tells the user when power is being supplied to the selected device. Figure 5 shows a selection for some of the personality modules supported on the universal programmer.

In addition to the testing done by the IUP system self-tests, each personality module contains diagnostic firmware that performs selected EPROM tests and indicates status. These tests are performed in both on-line and off-line modes. The

EPROM installation test verifies that the device is installed in the module correctly and that the ZIF socket is closed. The EPROM blank check determines whether a device is blank. The universal programmer automatically determines whether the blank state is all zeros or all ones. The overlay check (performed when a EPROM is not blank) determines which bits are programmed, compares those bits against the program to be loaded, and allows programming to continue if they match. As with the system self-tests, straight-forward messages are provided. The user can invoke all of the EPROM device integrity checks except the installation test (which occurs automatically any time an operation is selected).

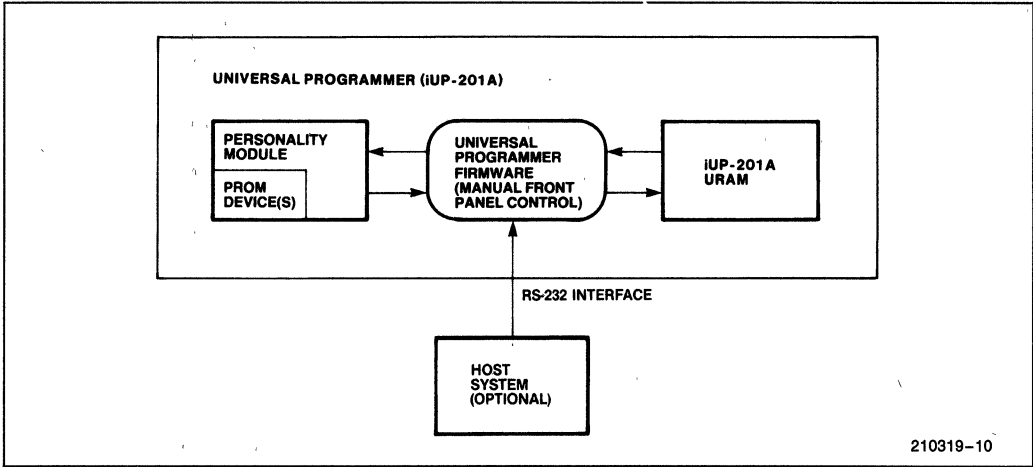


Figure 4. Off-Line System Data Flow

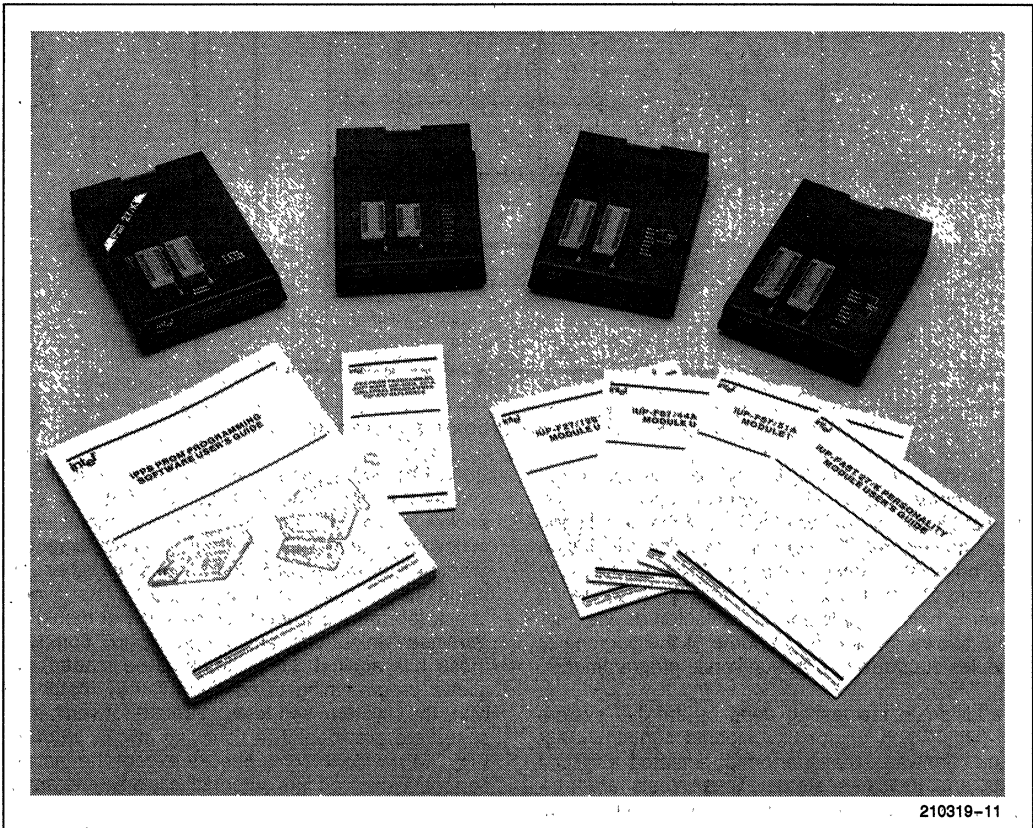


Figure 5. Personality Modules

IUP-200A/iUP201A SPECIFICATIONS

Control Processor

Intel 8085A microprocessor
6.144 MHz clock rate

Memory

RAM—4.3 bytes static
ROM—12K bytes EPROM

Interfaces

Keyboard: 16-character hexadecimal and 12-function keypad (iUP-201A model only)
Display: 24-character alphanumeric (iUP-201A model only)

Software

Monitor—system controller in pre-programmed EPROM
iPPS — Intel PROM programming software on supplied diskette

Physical Characteristics

Depth: 15 inches (38.1 cm)
Width: 15 inches (38.1 cm)
Height: 6 inches (15.2 cm)
Weight: 15 pounds (6.9 kg)

Electrical Characteristics

Selectable 100, 120, 200, or 240 Vac \pm 10%; 50-60 Hz
Maximum power consumption—80 watts

Environmental Characteristics

Reading Temperature: 10°C to 40°C
Programming Temperature: 25°C \pm 5°
Operating Humidity: 10% to 85% relative humidity

Reference Material

166041-001— *iUP-200A/201A Universal Programmer User's Guide.*

166042-001— *Getting Started with the iUP-200A/201A (For ISIS/iNDX Users).*
166043-001— *Getting Started with the iUP-200A/201A (For DOS Users).*
164853 — *iUP-200A/201A Universal Programmer Pocket Reference.*

ORDERING INFORMATION

Product Order Code	Description
iUP-200A 211A	On-Line PROM programmer with iPPS rel 1.4 on Single density ISIS II floppy
iUP-200A 212B	On-Line PROM programmer with iPPS rel 1.4 on Double density ISIS II floppy
iUP-200A 213C	On-Line PROM programmer with iPPS rel 2.0 for Series IV, on mini-floppy
iUP-200A 216D	On-Line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for PC or XT
iUP-200A 217D	On-Line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for AT
iUP-201A 211A	Off-Line and on-line PROM programmer with iPPS rel 1.4 on Single density ISIS II floppy
iUP-201A 212B	Off-Line and on-line PROM programmer with iPPS rel 1.4 on Double density ISIS II floppy
iUP-201A 213C	Off-Line and on-line PROM programmer with iPPS rel 2.0 for Series IV on mini-floppy
iUP-201A 216D	Off-Line and on-line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for PC or XT
iUP-201A 217D	Off-Line and on-line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for AT
iUP-200/201 U1* Upgrade Kit	Upgrades an iUP-200/201 universal programmer to an iUP-200A/201A universal programmer
iUP-PAK-A Upgrade Kit	Upgrades an iUP-200/A universal programmer to an iUP-201A universal programmer

*Most personality modules can be used only with an iUP-200A/201A universal programmer or an iUP-200/iUP201 universal programmer upgraded to an A with the iUP-200/iUP-201 U1 upgrade kit. If used in an iPDS, most personality modules require version 1.4 of the iPPS software.

Product Order Code	Description
iUP-PAK-A Upgrade Kit	Upgrades an iUP-200A universal programmer to an iUP-201A universal programmer

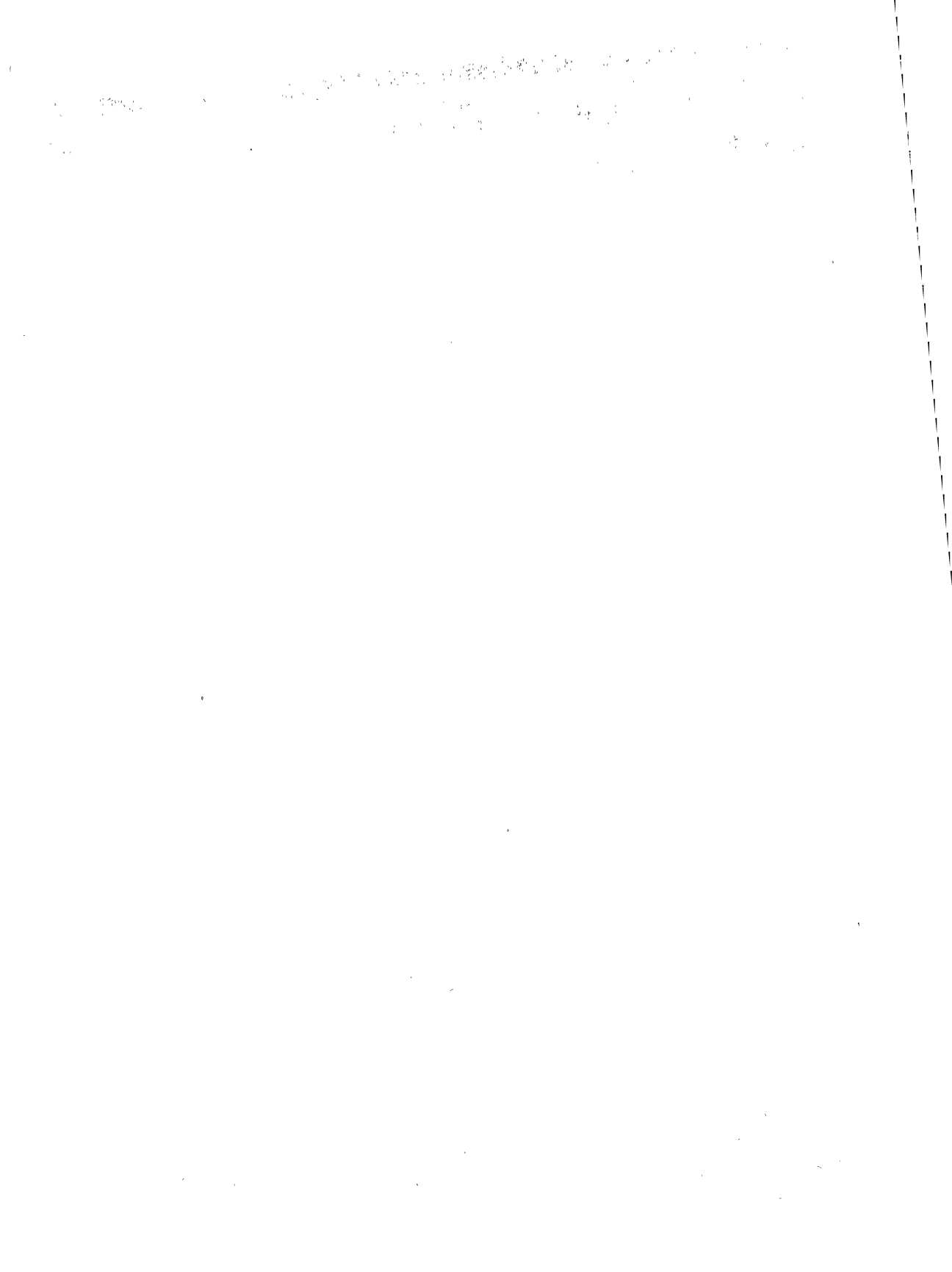
Software Sold Separately

Product Order Code	Description
211A	PROM programming software rel 1.4 on Single density ISIS II floppy
212B	PROM programming software rel 1.4 on Double density ISIS II floppy

Product Order Code	Description
213C	PROM programming software rel 2.0 for Series IV on mini-floppy
216D	PROM programming software rel 2.0 for PC/DOS with cable for PC or PC XT
217D	PROM programming software rel 2.0 for PC/DOS with cable for PC AT
219F	PROM programming software for iPDS on mini-floppy

Alphanumeric Terminal Controllers

6





8275H PROGRAMMABLE CRT CONTROLLER

- Programmable Screen and Character Format
- 6 Independent Visual Field Attributes
- 11 Visual Character Attributes (Graphic Capability)
- Cursor Control (4 Types)
- Light Pen Detection and Registers
- MCS-51®, MCS-85®, IAPX 86, and IAPX 88 Compatible
- Dual Row Buffers
- Programmable DMA Burst Mode
- Single +5V Supply
- High Performance HMOS-II

The Intel® 8275H Programmable CRT Controller is a single chip device to interface CRT raster scan displays with Intel® microcomputer systems. It is manufactured on Intel's advanced HMOS-II process. Its primary function is to refresh the display by buffering the information from main memory and keeping track of the display position of the screen. The flexibility designed in the 8275H will allow simple interface to almost any raster scan CRT display with a minimum of external hardware and software overhead.

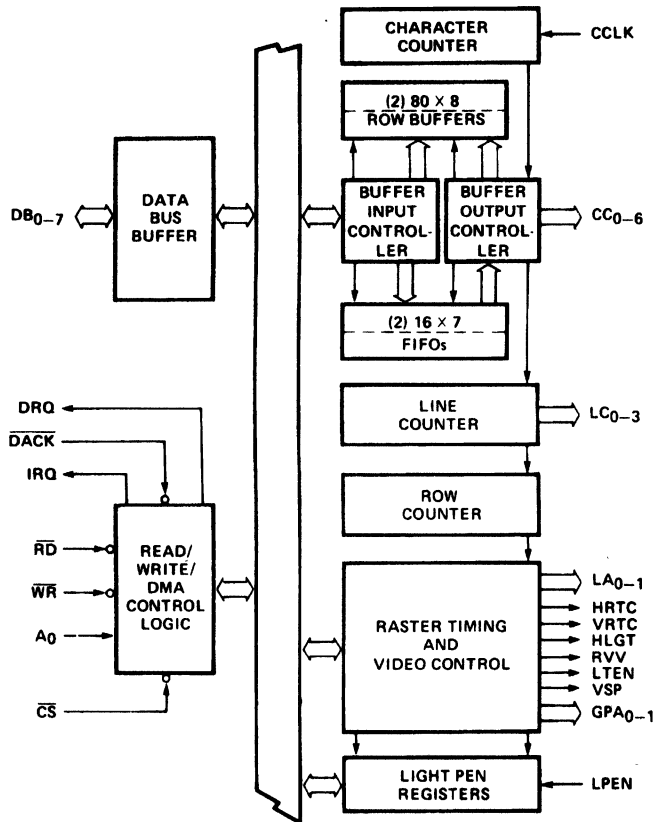


Figure 1. Block Diagram

210464-1

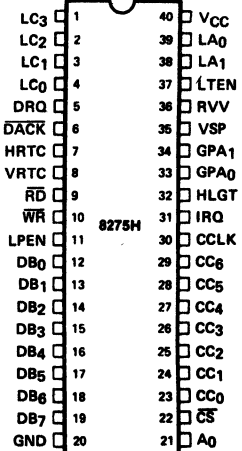


Figure 2. Pin Configuration

210464-2

Table 1. Pin Descriptions

Symbol	Pin No.	Type	Name and Function
LC ₃ LC ₂ LC ₁ LC ₀	1 2 3 4	O	LINE COUNT: Output from the line counter which is used to address the character generator for the line positions on the screen.
DRQ	5	O	DMA REQUEST: Output signal to the 8257 DMA controller requesting a DMA cycle.
$\overline{\text{DACK}}$	6	I	DMA ACKNOWLEDGE: Input signal from the 8257H DMA controller acknowledging that the requested DMA cycle has been granted.
HRTC	7	O	HORIZONTAL RETRACE: Output signal which is active during the programmed horizontal retrace interval. During this period the VSP output is high and the LTEN output is low.
VRTC	8	O	VERTICAL RETRACE: Output signal which is active during the programmed vertical retrace interval. During this period the VSP output is high and the LTEN output is low.
$\overline{\text{RD}}$	9	I	READ INPUT: A control signal to read registers.
$\overline{\text{WR}}$	10	I	WRITE INPUT: A control signal to write commands into the control registers or write data into the row buffers during a DMA cycle.
LPEN	11	I	LIGHT PEN: Input signal from the CRT system signifying that a light pen signal has been detected.
DB ₀ DB ₁ DB ₂ DB ₃ DB ₄ DB ₅ DB ₆ DB ₇	12 13 14 15 16 17 18 19	I/O	BI-DIRECTIONAL THREE-STATE DATA BUS LINES: The outputs are enabled during a read of the C or P ports.
Ground	20		GROUND.
V _{CC}	40		+ 5V POWER SUPPLY.
LA ₀ LA ₁	39 38	O	LINE ATTRIBUTE CODES: These attribute codes have to be decoded externally by the dot/timing logic to generate the horizontal and vertical line combinations for the graphic displays specified by the character attribute codes.

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function
LTEN	37	O	LIGHT ENABLE: Output signal used to enable the video signal to the CRT. This output is active at the programmed underline cursor position, and at positions specified by attribute codes.
RVV	36	O	REVERSE VIDEO: Output signal used to indicate the CRT circuitry to reverse the video signal. This output is active at the cursor position if a reverse video block cursor is programmed or at the positions specified by the field attribute codes.
VSP	35	O	VIDEO SUPPRESSION: Output signal used to blank the video signal to the CRT. This output is active: <ul style="list-style-type: none"> • during the horizontal and vertical retrace intervals. • at the top and bottom lines of rows if underline is programmed to be number 8 or greater. • when an end of row or end of screen code is detected. • when a DMA underrun occurs. • at regular intervals ($1/16$ frame frequency for cursor, $1/32$ frame frequency for character and field attributes)—to create blinking displays as specified by cursor, character attribute, or field attribute programming.
GPA ₁ GPA ₀	34 33	O	GENERAL PURPOSE ATTRIBUTE CODES: Outputs which are enabled by the general purpose field attribute codes.
HLGT	32	O	HIGHLIGHT: Output signal used to intensify the display at particular positions on the screen as specified by the character attribute codes or field attribute codes.
IRQ	31	O	INTERRUPT REQUEST.
CCLK	30	I	CHARACTER CLOCK (from Dot/Timing Logic).
CC ₆ CC ₅ CC ₄ CC ₃ CC ₂ CC ₁ CC ₀	29 28 27 26 25 24 23	O	CHARACTER CODES: Output from the row buffers used for character selection in the character generator.
CS	22	I	CHIP SELECT: The read and write are enabled by CS.
A ₀	21	I	PORT ADDRESS: A high input on A ₀ selects the "C" port or command registers and a low input selects the "P" port or parameter registers.

FUNCTIONAL DESCRIPTION

Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8275H to the system Data Bus.

This functional block accepts inputs from the System Control Bus and generates control signals for overall device operation. It contains the Command, Parameter, and Status Registers that store the various control formats for the device functional definition.

A ₀	Operation	Register
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

A ₀	\overline{RD}	\overline{WR}	\overline{CS}	
0	1	0	0	Write 8275H Parameter
0	0	1	0	Read 8275H Parameter
1	1	0	0	Write 8275H Command
1	0	1	0	Read 8275H Status
X	1	1	0	Three-State
X	X	X	1	Three-State

\overline{RD} (READ)

A "low" on this input informs the 8275H that the CPU is reading data or status information from the 8275H.

\overline{WR} (WRITE)

A "low" on this input informs the 8275H that the CPU is writing data or control words to the 8275H.

\overline{CS} (CHIP SELECT)

A "low" on this input selects the 8275H. No reading or writing will occur unless the device is selected. When \overline{CS} is high, the Data Bus is in the float state and \overline{RD} and \overline{WR} will have no effect on the chip.

DRQ (DMA REQUEST)

A "high" on this output informs the DMA Controller that the 8275H desires a DMA transfer.

\overline{DACK} (DMA ACKNOWLEDGE)

A "low" on this input informs the 8275H that a DMA cycle is in progress.

IRQ (INTERRUPT REQUEST)

A "high" on this output informs the CPU that the 8275H desires interrupt service.

Character Counter

The Character Counter is a programmable counter that is used to determine the number of characters to be displayed per row and the length of the horizontal retrace interval. It is driven by the CCLK (Character Clock) input, which should be a derivative of the external dot clock.

Line Counter

The Line Counter is a programmable counter that is used to determine the number of horizontal lines (Sweeps) per character row. Its outputs are used to address the external character generator ROM.

Row Counter

The Row Counter is a programmable counter that is used to determine the number of character rows to be displayed per frame and length of the vertical retrace interval.

Light Pen Registers

The Light Pen Registers are two registers that store the contents of the character counter and the row counter whenever there is a rising edge on the LPEN (Light Pen) input.

NOTE:

Software correction is required.

Raster Timing and Video Controls

The Raster Timing circuitry controls the timing of the HRTC (Horizontal Retrace) and VRTC (Vertical Retrace) outputs. The Video Control circuitry controls the generation of LA₀₋₁ (Line Attribute), HGLT (Highlight), RVV (Reverse Video), LTEN (Light Enable), VSP (Video Suppress), and GPA₀₋₁ (General Purpose Attribute) outputs.

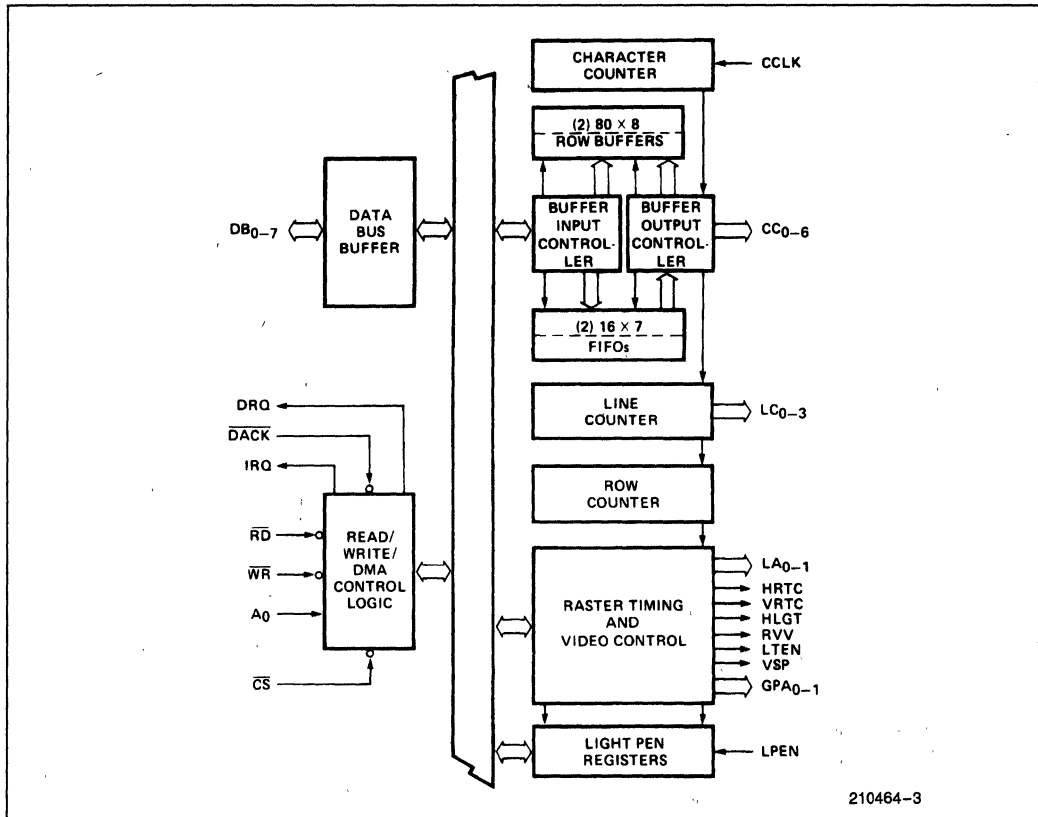


Figure 3. 8275H Block Diagram Showing Counter and Register Functions

Row Buffers

The Row Buffers are two 80 character buffers. They are filled from the microcomputer system memory with the character codes to be displayed. While one row buffer is displaying a row of characters, the other is being filled with the next row of characters.

FIFOs

There are two 16 character FIFOs in the 8275H. They are used to provide extra row buffer length in the Transparent Attribute Mode (see Detailed Operation section).

Buffer Input/Output Controllers

The Buffer Input/Output Controllers decode the characters being placed in the row buffers. If the

character is a character attribute, field attribute or special code, these controllers control the appropriate action. (Examples: An "End of Screen—Stop DMA" special code will cause the Buffer Input Controller to stop further DMA requests. A "Highlight" field attribute will cause the Buffer Output Controller to activate the HGLT output.)

SYSTEM OPERATION

The 8275H is programmable to a large number of different display formats. It provides raster timing, display row buffering, visual attribute decoding, cursor timing, and light pen detection.

It is designed to interface with the 8257 DMA Controller and standard character generator ROMs for dot matrix decoding. Dot level timing must be provided by external circuitry.

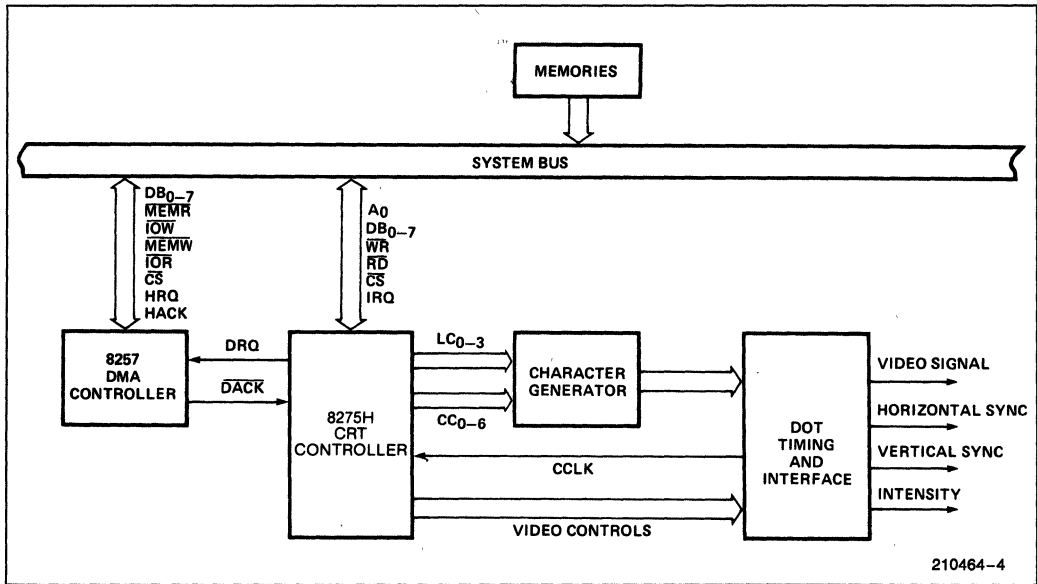


Figure 4. 8275H Systems Block Diagram Showing Systems Operation

General Systems Operational Description

The 8275H provides a "window" into the microcomputer system memory.

Display characters are retrieved from memory and displayed on a row by row basis. The 8275H has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and the number of character rows per frame are software programmable, providing easy interface to most CRT displays. (See Programming Section.)

The 8275H requests DMA to fill the row buffer that is not being used for display. DMA burst length and spacing is programmable. (See Programming Section.)

The 8275H displays character rows one line at a time.

The number of lines per character row, the underline position, and blanking of top and bottom lines are programmable. (See Programming Section.)

The 8275H provides special Control Codes which can be used to minimize DMA or software overhead. It also provides Visual Attribute Codes to cause special action or symbols on the screen without the use of the character generator (See Visual Attributes Section).

The 8275H also controls raster timing. This is done by generating Horizontal Retrace (HRTC) and Vertical Retrace (VRTC) signals. The timing of these signals is programmable.

The 8275H can generate a cursor. Cursor location and format are programmable. (See Programming Section.)

The 8275H has a light pen input and registers. The light pen input is used to load the registers. Light pen registers can be read on command. (See Programming Section.)

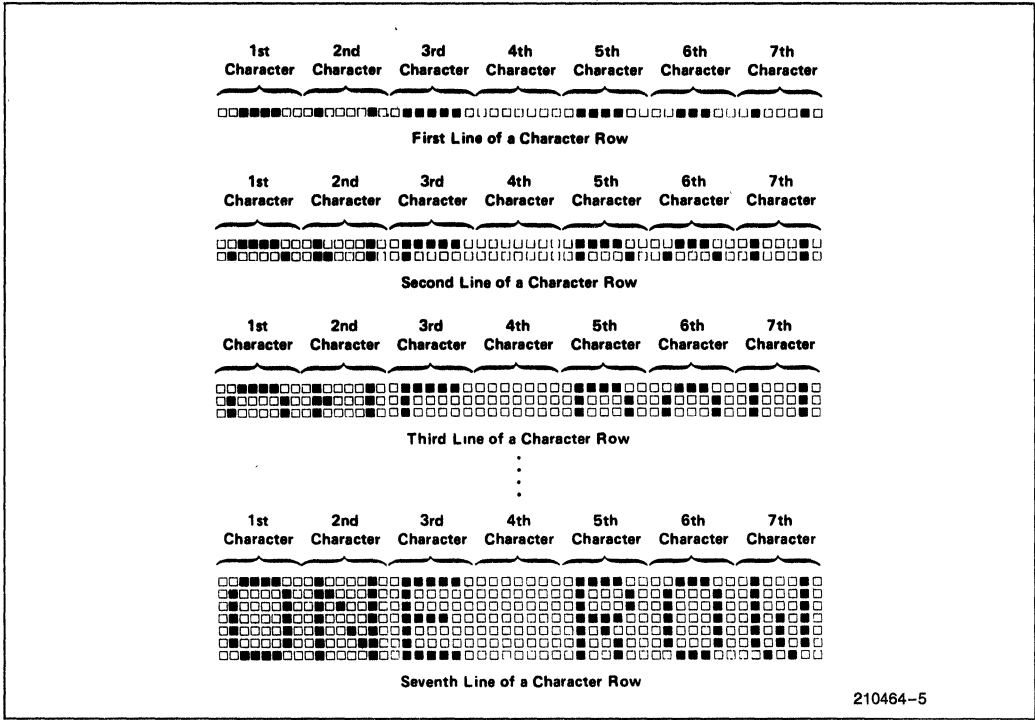


Figure 5. Display of a Character Row

Display Row Buffering

Before the start of a frame, the 8275H requests DMA and one row buffer is filled with characters.

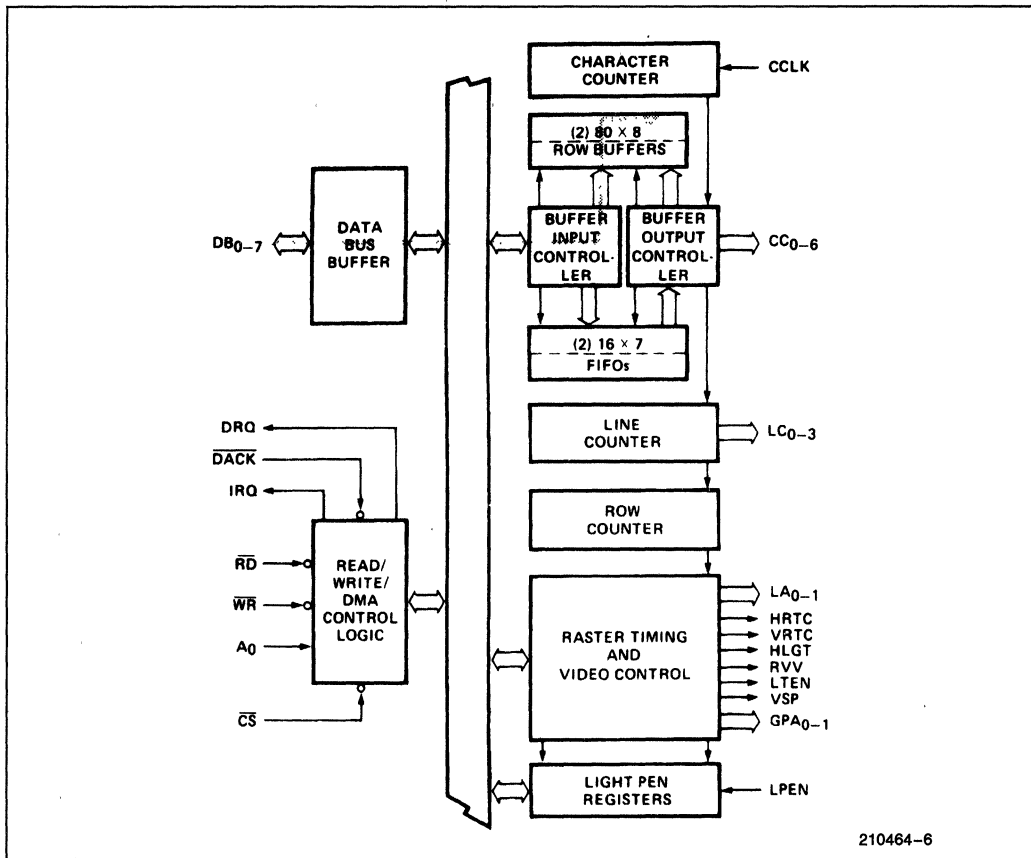
When the first horizontal sweep is started, character codes are output to the character generator from the row buffer just filled. Simultaneously, DMA begins

filling the other row buffer with the next row of characters.

After all the lines of the character row are scanned, the roles of the two row buffers are reversed and the same procedure is followed for the next row.

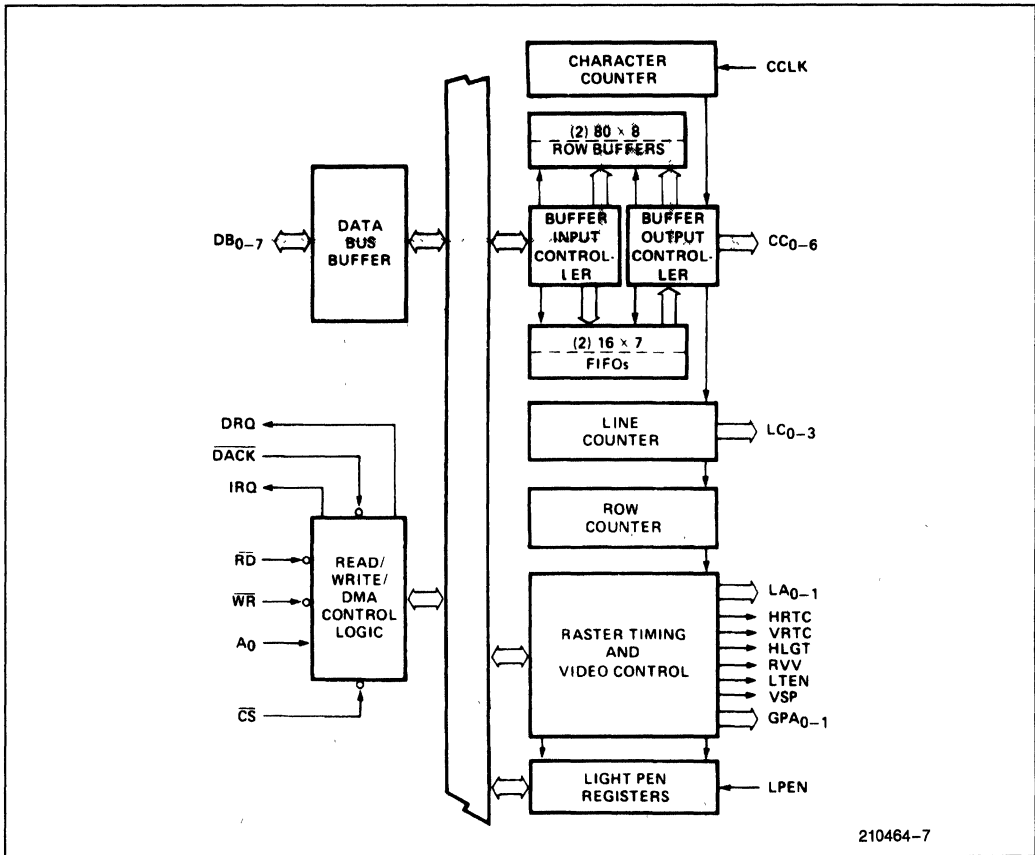
This is repeated until all of the character rows are displayed.

210464-5



210464-6

Figure 6. First Row Buffer Filled



210464-7

Figure 7. Second Buffer Filled, First Row Displayed

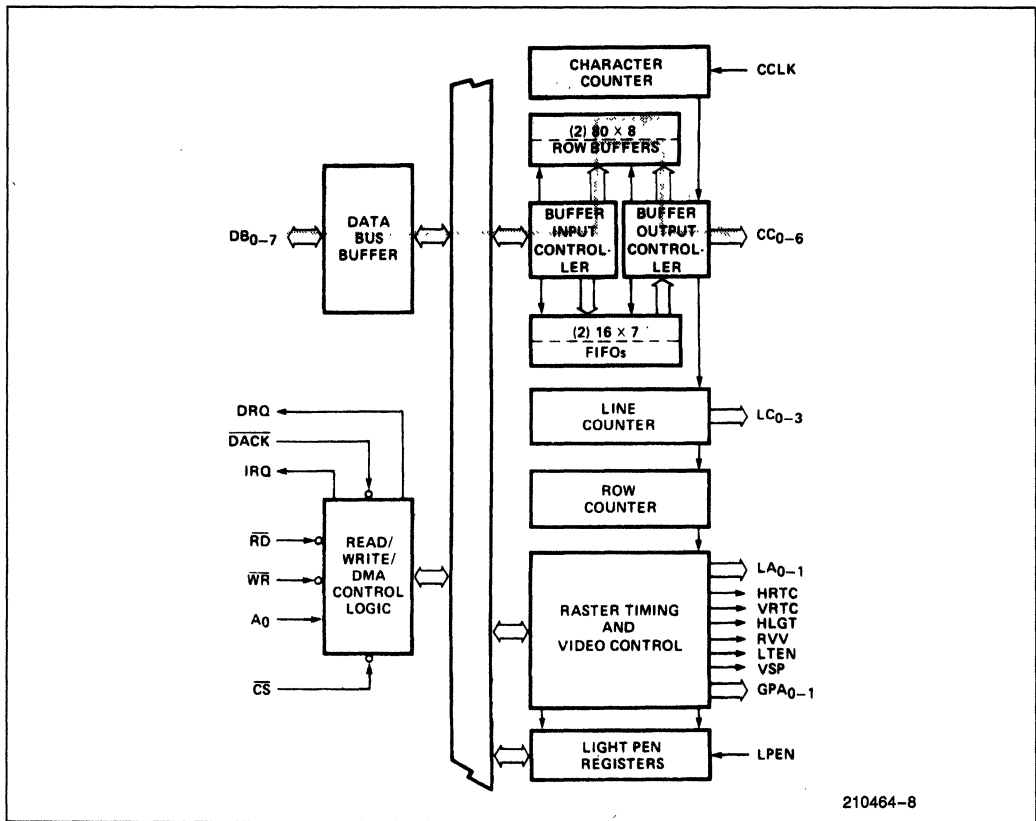


Figure 8. First Buffer Filled with Third Row, Second Row Displayed

Display Format

SCREEN FORMAT

The 8275H can be programmed to generate from 1 to 80 characters per row, and from 1 to 64 rows per frame.

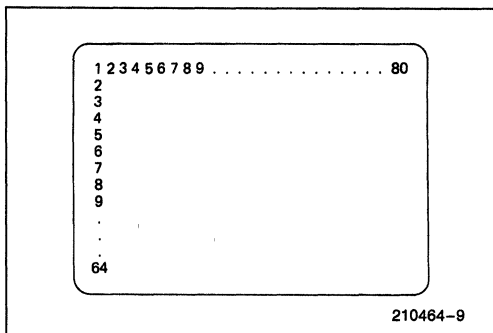


Figure 9. Screen Format

The 8275H can also be programmed to blank alternate rows. In this mode the first row is displayed, the second blanked, the third displayed, etc. DMA is not requested for the blanked rows.

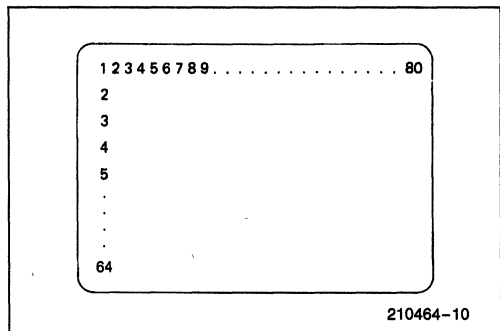


Figure 10. Blank Alternate Rows Mode

ROW FORMAT

The 8275H is designed to hold the line count stable while outputting the appropriate character codes during each horizontal sweep. The line count is incremented during horizontal retrace and the whole row of character codes are output again during the next sweep. This is continued until the whole character row is displayed.

The number of lines (horizontal sweeps) per character row is programmable from 1 to 16.

The output of the line counter can be programmed to be in one of two modes.

In mode 0, the output of the line counter is the same as the line number.

In mode 1, the line counter is offset by one from the line number.

NOTE:

In mode 1, while the first line (line number 0) is being displayed, the last count is output by the line counter (see examples).

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1111
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000
10	1010	1001
11	1011	1010
12	1100	1011
13	1101	1100
14	1110	1101
15	1111	1110

210464-11

Figure 11. Example of a 16-Line Format

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1001
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000

210464-12

Figure 12. Example of a 10-Line Format

Mode 0 is useful for character generators that leave address zero blank and start at address 1. Mode 1 is useful for character generators which start at address zero.

Underline placement is also programmable (from line number 0 to 15). This is independent of the line counter mode.

If the line number of the underline is greater than 7 (line number MSB = 1), then the top and bottom lines will be blanked.

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1011
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000
10	1010	1001
11	1011	1010

Top and Bottom Lines are Blanked

210464-13

Figure 13. Underline in Line Number 10

If the line number of the underline is less than or equal to 7 (line number MSB = 0), then the top and bottom lines will *not* be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ ■ □ □ □	0000	0111
1	□ □ ■ □ ■ □ □ □	0001	0000
2	□ ■ □ □ □ □ ■ □	0010	0001
3	□ □ □ □ □ □ □ □	0011	0010
4	□ ■ □ □ ■ □ ■ □	0100	0011
5	□ ■ □ □ □ □ □ □	0101	0100
6	□ ■ □ □ □ □ □ □	0110	0101
7	■ □ ■ □ ■ □ ■ □	0111	0110

Top and Bottom Lines are not Blanked

210464-14

Figure 14. Underline in Line Number 7

If the line number of the underline is greater than the maximum number of lines, the underline will not appear.

Blanking is accomplished by the VSP (Video Suppression) signal. Underline is accomplished by the LTEN (Light Enable) signal.

DOT FORMAT

Dot width and character width are dependent upon the external timing and control circuitry.

Dot level timing circuitry should be designed to accept the parallel output of the character generator and shift it out serially at the rate required by the CRT display.

Dot width is a function of dot clock frequency

Character width is a function of the character generator width.

Horizontal character spacing is a function of the shift register length.

NOTE:

Video control and timing signals must be synchronized with the video signal due to the character generator access delay.

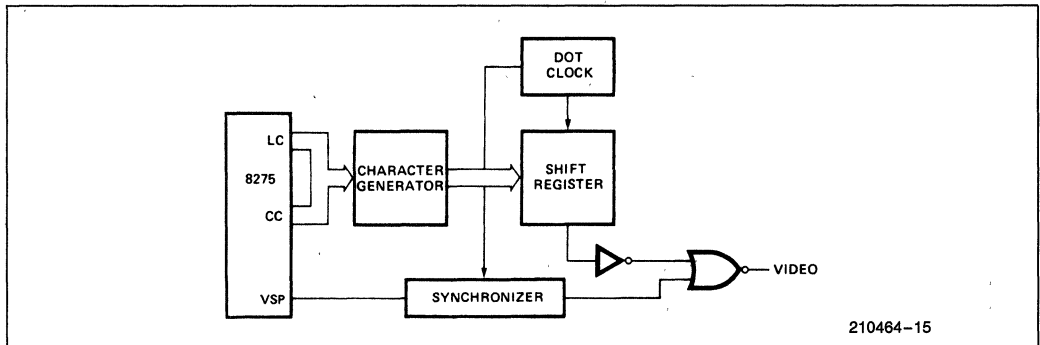


Figure 15. Typical Dot Level Block Diagram

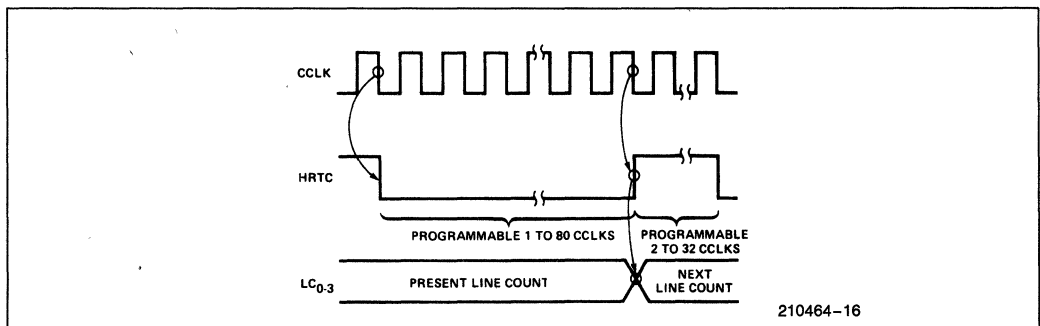


Figure 16. Line Timing

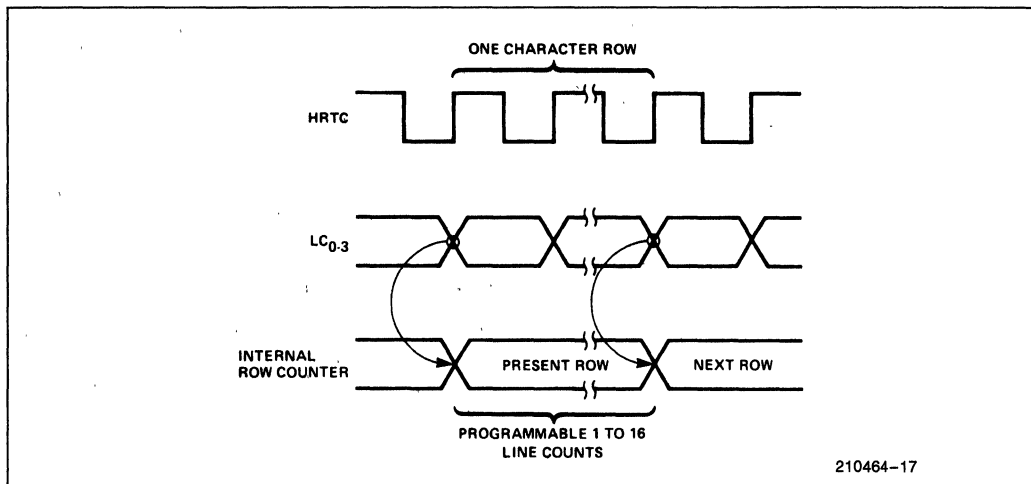


Figure 17. Row Timing

Raster Timing

The character counter is driven by the character clock input (CCLK). It counts out the characters being displayed (programmable from 1 to 80). It then causes the line counter to increment, and it starts counting out the horizontal retrace interval (programmable from 2 to 32). This is constantly repeated.

The line counter is driven by the character counter. It is used to generate the line address outputs (LC₀₋₃) for the character generator. After it counts all of the lines in a character row (programmable from 1 to 16), it increments the row counter, and starts over

again. (See Character Format Section for detailed description of Line Counter functions.)

The row counter is an internal counter driven by the line counter. It controls the functions of the row buffers and counts the number of character rows displayed.

After the row counter counts all of the rows in a frame (programmable from 1 to 64), it starts counting out the vertical retrace interval (programmable from 1 to 4).

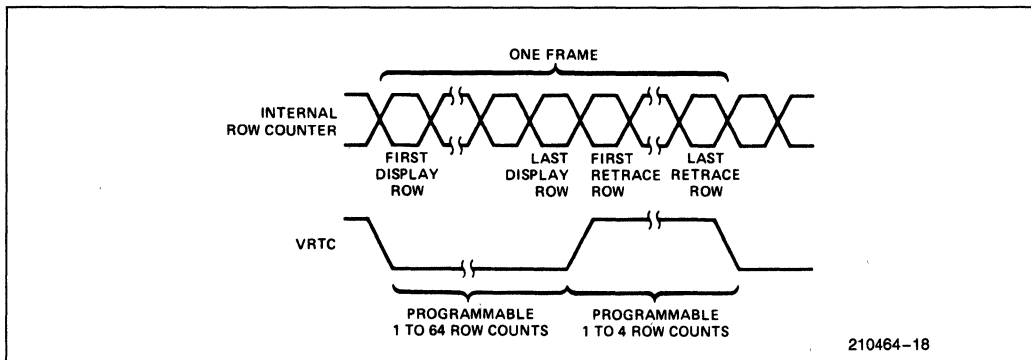


Figure 18. Frame Timing

The Video Suppression Output (VSP) is active during horizontal and vertical retrace intervals.

Dot level timing circuitry must synchronize these outputs with the video signal to the CRT Display.

DMA Timing

The 8275H can be programmed to request burst DMA transfers of 1 to 8 characters. The interval between bursts is also programmable (from 0 to 55 character clock periods ± 1). This allows the user to tailor his DMA overhead to fit his system needs.

The first DMA request of the frame occurs one *row time* before the end of vertical retrace. DMA requests continue as programmed, until the row buffer is filled. If the row buffer is filled in the middle of a burst, the 8275H terminates the burst and resets the burst counter. No more DMA requests will occur until the *beginning* of the *next row*. At that time, DMA requests are activated as programmed until the other buffer is filled.

The first DMA request for a row will start at the first character clock of the preceding row. If the burst

mode is used, the first DMA request may occur a number of character clocks later. This number is equal to the programmed burst space.

If, for any reason, there is a DMA underrun, a flag in the status word will be set.

The DMA controller is typically initialized for the next frame at the end of the current frame.

Interrupt Timing

The 8275H can be programmed to generate an interrupt request at the end of each frame. This can be used to reinitialize the DMA controller. If the 8275H interrupt enable flag is set, an interrupt request will occur at the *beginning* of the *last display row*.

IRQ will go inactive after the status register is read.

A reset command will also cause IRQ to go inactive, but this is not recommended during normal service.

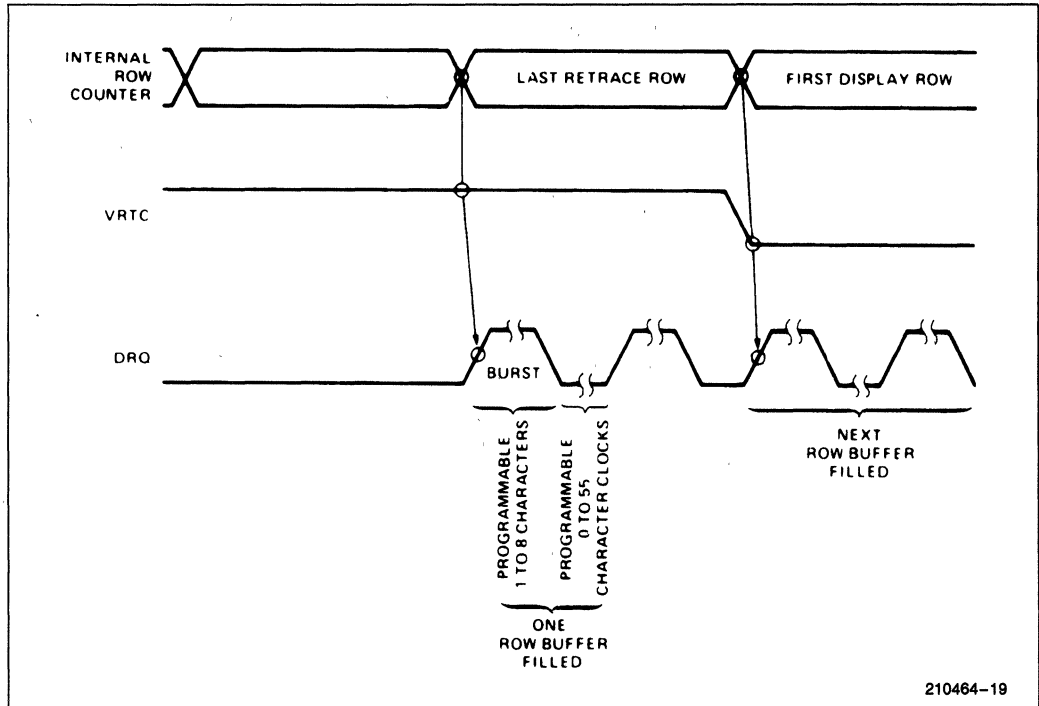


Figure 19. DMA Timing

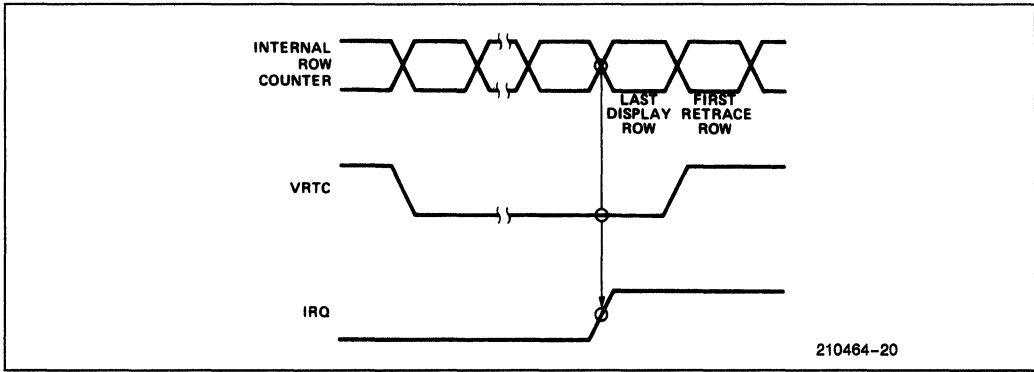


Figure 20. Beginning of Interrupt Request

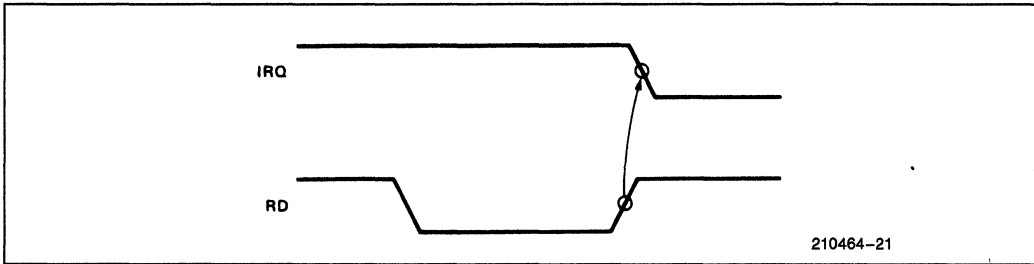


Figure 21. End of Interrupt Request

Another method of reinitializing the DMA controller is to have the DMA controller itself interrupt on terminal count. With this method, the 8275H interrupt enable flag should not be set.

NOTE:

Upon power-up, the 8275H Interrupt Enable Flag may be set. As a result, the user's cold start routine should write a reset command to the 8275H before system interrupts are enabled.

VISUAL ATTRIBUTES AND SPECIAL CODES

The characters processed by the 8275H are 8-bit quantities. The character code outputs provide the character generator with 7 bits of address. The Most Significant Bit is the extra bit and it is used to determine if it is a normal display character (MSB = 0), or if it is a Visual Attribute or Special Code (MSB = 1).

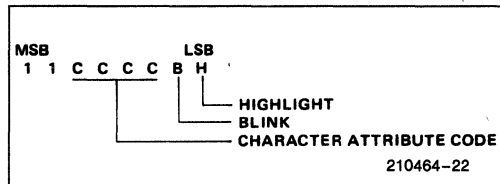
There are two types of Visual Attribute Codes. They are Character Attributes and Field Attributes.

CHARACTER ATTRIBUTE CODES

Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA₀₋₁), the Video Suppression output (VSP), and the Light Enable output. The dot level timing circuitry can use these signals to generate the proper symbols.

Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT).

Character Attributes



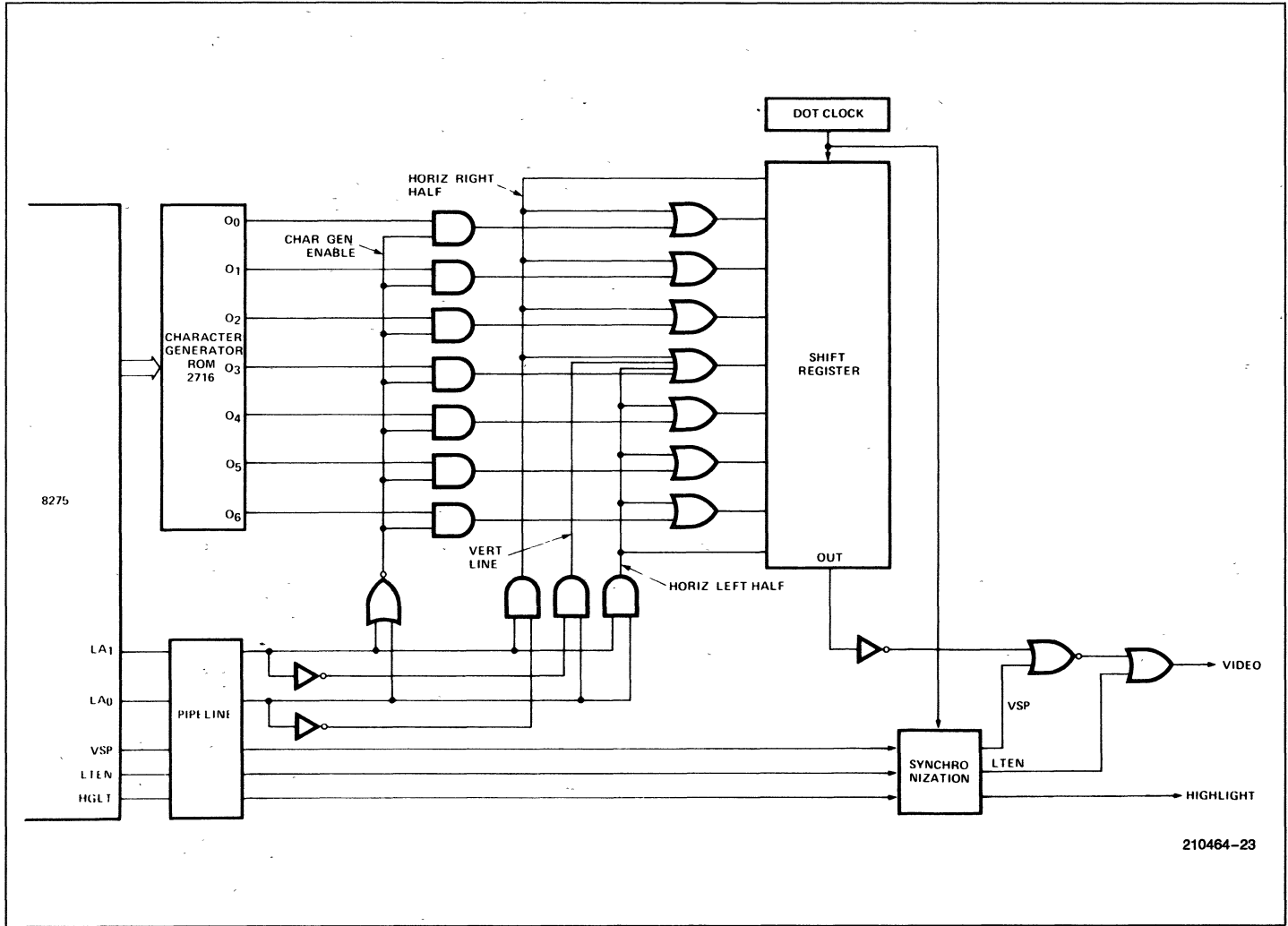















Figure 22. Typical Character Attribute Logic
6-16

Table 2. Character Attributes

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"	OUTPUTS				SYMBOL	DESCRIPTION	
	LA ₁	LA ₀	VSP	LTEN			
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

210464-24

*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

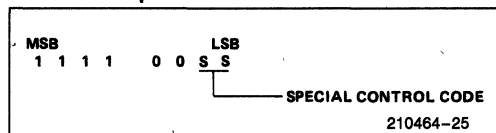
Blinking is active when B = 1.

Highlight is active when H = 1.

SPECIAL CODES

Four special codes are available to help reduce memory, software, or DMA overhead.

Special Control Character



S	S	Function
0	0	End of Row
0	1	End of Row-Stop DMA
1	0	End of Screen
1	1	End of Screen-Stop DMA

The End of Row Code (00) activates VSP and holds it to the end of the line.

The End of Row-Stop DMA Code (01) causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the Row Buffer. It affects the display in the same way as the End of Row Code (00).

The End of Screen Code (10) activates VSP and holds it to the end of the frame.

The End of Screen-Stop DMA Code (11) causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the Row Buffer. It affects the display in the same way as the End of Screen Code (10).

If the Stop DMA feature is not used, all characters after an End of Row character are ignored, except for the End of Screen character, which operates normally. All characters after an End of Screen character are ignored.

NOTE:

If a Stop DMA character is not the last character in a burst or row, DMA is not stopped until after the next character is read. In this situation, a dummy character must be placed in memory after the Stop DMA character.

FIELD ATTRIBUTES

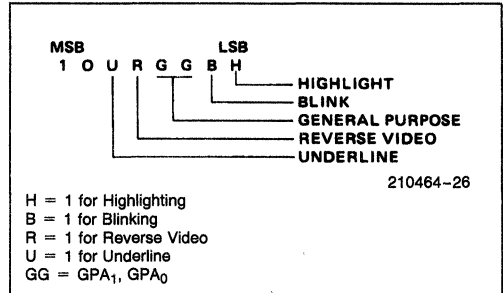
The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the code up to, and including, the character which precedes the *next* field attribute code, or up to the end of the frame. The field attributes are reset during the vertical retrace interval.

There are six field attributes:

- 1) *Blink*—Characters following the code blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
- 2) *Highlight*—Characters following the code are highlighted by activating the Highlight output (HGLT).

- 3) *Reverse Video*—Characters following the code appear with reverse video by activating the Reverse Video output (RVV).
- 4) *Underline*—Characters following the code are underlined by activating the Light Enable output (LTEN).
- 5,6) *General Purpose*—There are two additional 8275H outputs which act as general purpose, independently programmable field attributes. GPA₀₋₁ are active high outputs.

Field Attribute Code



NOTE:

More than one attribute can be enabled at the same time. If the blinking and reverse video attributes are enabled simultaneously, only the reversed characters will blink.

The 8275H can be programmed to provide visible or invisible field attribute characters.

If the 8275H is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character.

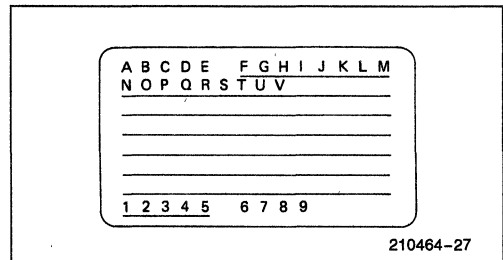


Figure 23. Example of the Visible Field Attribute Mode (Underline Attribute)

If the 8275H is programmed in the invisible field attribute mode, the 8275H FIFO is activated.

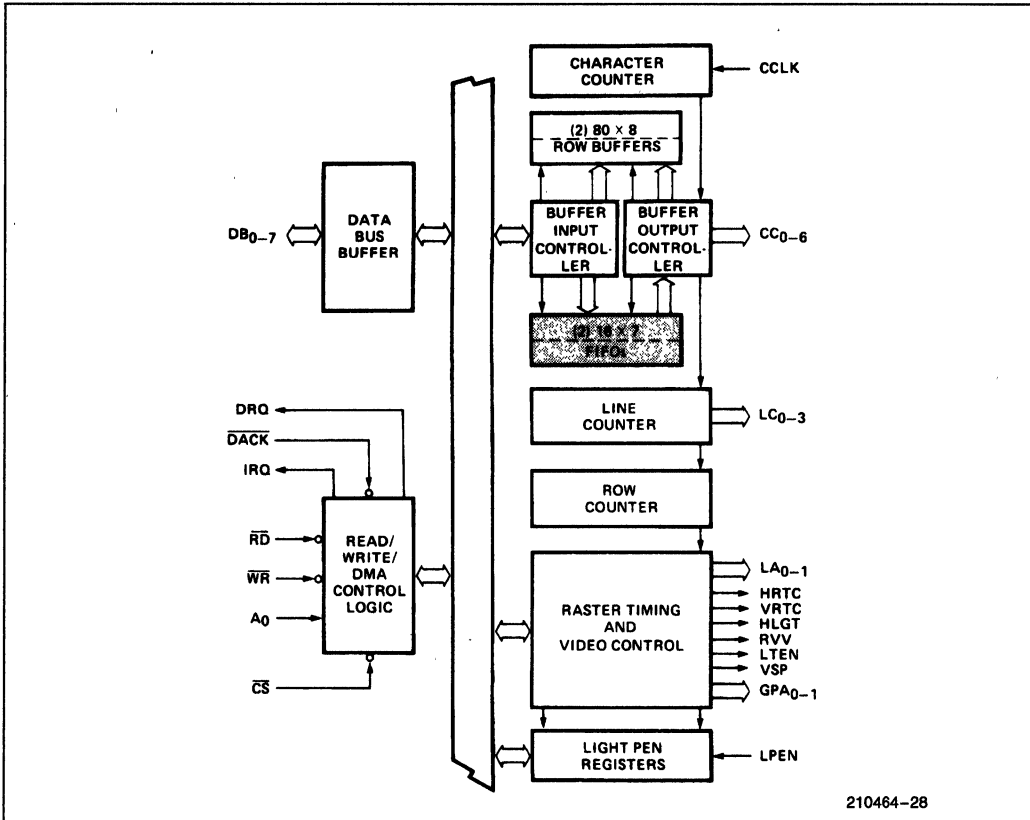


Figure 24. Block Diagram Showing FIFO Activation

Each row buffer has a corresponding FIFO. These FIFOs are 16 characters by 7 bits in size.

When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the *next* character in the proper FIFO.

When a field attribute is placed in the Buffer Output Controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC₀₋₆). The chosen Visual Attributes are also activated.

Since the FIFO is 16 characters long, no more than 16 field attribute characters may be used per line in this mode. If more are used, a bit in the status word is set and the first characters in the FIFO are written over and lost.

NOTE:

Since the FIFO is 7 bits wide, the MSB of any characters put in it are stripped off. Therefore, a Visual

Attribute or Special Code must *not* immediately follow a field attribute code. If this situation does occur, the Visual Attribute or Special Code will be treated as a normal display character.

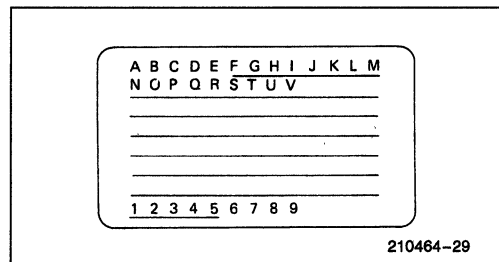


Figure 25. Example of the Invisible Field Attribute Mode (Underline Attribute)

Field and Character Attribute Interaction

Character Attribute Symbols are affected by the Reverse Video (RVV) and General Purpose (GPA₀₋₁) field attributes. They are not affected by Underline, Blink or Highlight field attributes; however, these characteristics can be programmed *individually* for Character Attribute Symbols.

Cursor Timing

The cursor location is determined by a cursor row register and a character position register which are loaded by command to the controller. The cursor can be programmed to appear on the display as:

- 1) a blinking underline
- 2) a blinking reverse video block
- 3) a non-blinking underline
- 4) a non-blinking reverse video block

The cursor blinking frequency is equal to the screen refresh frequency divided by 16.

If a non-blinking reverse video *cursor* appears in a non-blinking reverse video *field*, the cursor will appear as a normal video block.

If a non-blinking underline *cursor* appears in a non-blinking underline *field*, the cursor will not be visible.

Light Pen Detection

A light pen consists of a micro switch and a tiny light sensor. When the light pen is pressed against the CRT screen, the micro switch enables the light sensor. When the raster sweep reaches the light sensor, it triggers the light pen output.

If the output of the light pen is presented to the 8275H LPEN input, the row and character position coordinates are stored in a pair of registers. These registers can be read on command. A bit in the status word is set, indicating that the light pen signal was detected. The LPEN input must be a 0 to 1 transition for proper operation.

1.0 Reset Command

	Operation	A ₀	Description	Data Bus							
				MSB				LSB			
Command	Write	1	Reset Command	0	0	0	0	0	0	0	0
Parameters	Write	0	Screen Comp Byte 1	S	H	H	H	H	H	H	H
	Write	0	Screen Comp Byte 2	V	V	R	R	R	R	R	R
	Write	0	Screen Comp Byte 3	U	U	U	U	L	L	L	L
	Write	0	Screen Comp Byte 4	M	F	C	C	Z	Z	Z	Z

NOTE:

Due to internal and external delays, the character position coordinate will be off by at least three character positions. This has to be corrected in software.

Device Programming

The 8275H has two programming registers, the Command Register (CREG) and the parameter register (PREG). It also has a Status Register (SREG). The Command Register can only be written into and the Status Registers can only be read from. They are addressed as follows:

A ₀	Operation	Register
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

The 8275H expects to receive a command and a sequence of 0 to 4 parameters, depending on the command. If the proper number of parameter bytes are not received before another command is given, a status flag is set, indicating an improper command.

INSTRUCTION SET

The 8275H instruction set consists of 8 commands.

Command	No. of Parameter Bytes
Reset	4
Start Display	0
Stop Display	0
Read Light Pen	2
Load Cursor	2
Enable Interrupt	0
Disable Interrupt	0
Preset Counters	0

In addition, the status of the 8275H (SREG) can be read by the CPU at any time.

Action—After the reset command is written, DMA requests stop, 8275H interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up.

As parameters are written, the screen composition is defined.

Parameter—S Spaced Rows

S	Functions
0	Normal Rows
1	Spaced Rows

Parameter—HHHHHHH

Horizontal Characters/Row

H	H	H	H	H	H	H	No. of Characters Per Row
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	2
0	0	0	0	0	1	0	3
.
1	0	0	1	1	1	1	80
1	0	1	0	0	0	0	Undefined
.
1	1	1	1	1	1	1	Undefined

Parameter—VV Vertical Retrace Row Count

V	V	No. of Row Counts Per VRTC
0	0	1
0	1	2
1	0	3
1	1	4

Parameter—RRRRRR Vertical Rows/Frame

R	R	R	R	R	R	No. of Rows/Frame
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
.
1	1	1	1	1	1	64

Parameter—UUUU Underline Placement

U	U	U	U	Line Number of Underline
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
.
1	1	1	1	16

Parameter—LLLL

Number of Lines per Character Row

L	L	L	L	No. of Lines/Row
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
.
1	1	1	1	16

Parameter—M Line Counter Mode

M	Line Counter Mode
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

Parameter—F Field Attribute Mode

F	Field Attribute Mode
0	Transparent
1	Non-Transparent

Parameter—CC Cursor Format

C	C	Cursor Format
0	0	Blinking reverse video block
0	1	Blinking underline
1	0	Nonblinking reverse video block
1	1	Nonblinking underline

Parameter—ZZZ Horizontal Retrace Count

Z	Z	Z	Z	No. of Character Counts Per HRTC
0	0	0	0	2
0	0	0	1	4
0	0	1	0	6
1	1	1	1	32

NOTE:

uuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

SSS Burst Space Code

S	S	S	No. of Character Clocks Between DMA Requests
0	0	0	0
0	0	1	7
0	1	0	15
0	1	1	23
1	0	0	31
1	0	1	39
1	1	0	47
1	1	1	55

BB Burst Count Code

B	B	No. of DMA Cycles Per Burst
0	0	1
0	1	2
1	0	4
1	1	8

Action—8275H interrupts are enabled, DMA requests begin, video is enabled, Interrupt Enable and Video Enable status flags are set.

2.0 Start Display Command

	Operation	A ₀	Description	Data Bus							
				MSB			LSB				
Command	Write	1	Start Display	0	0	1	S	S	S	B	B
No Parameters											

3.0 Stop Display Command

	Operation	A ₀	Description	Data Bus									
				MSB				LSB					
Command	Write	1	Stop Display	0	1	0	0	0	0	0	0	0	0
No Parameters													

Action—Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the "Start Display" command must be given to re-enable the display.

4.0 Read Light Pen Command

	Operation	A ₀	Description	Data Bus								
				MSB				LSB				
Command	Write	1	Read Light Pen	0	1	1	0	0	0	0	0	0
Parameters	Read	0	Char. Number	(Char. Position in Row)								
	Read	0	Row Number	(Row Number)								

Action—The 8275H is conditioned to supply the contents of the light pen position registers in the next two read cycles of the parameter register. Status flags are not affected.

NOTE:

Software correction of light pen position is required.

5.0 Load Cursor Position

	Operation	A ₀	Description	Data Bus								
				MSB							LSB	
Command	Write	1	Load Cursor	1	0	0	0	0	0	0	0	0
Parameters	Write	0	Char. Number	(Char. Position in Row)								
	Write	0	Row Number	(Row Number)								

Action—The 8275H is conditioned to place the next two parameter bytes into the cursor position registers. Status flags not affected.

6.0 Enable Interrupt Command

	Operation	A ₀	Description	Data Bus								
				MSB							LSB	
Command	Write	1	Enable Interrupt	1	0	1	0	0	0	0	0	0
No Parameters												

Action—The interrupt enable status flag is set and interrupts are enabled.

7.0 Disable Interrupt Command

	Operation	A ₀	Description	Data Bus								
				MSB							LSB	
Command	Write	1	Disable Interrupt	1	1	0	0	0	0	0	0	0
No Parameters												

Action—Interrupts are disabled and the interrupt enable status flag is reset.

8.0 Preset Counters Command

	Operation	A ₀	Description	Data Bus								
				MSB							LSB	
Command	Write	1	Preset Counters	1	1	1	0	0	0	0	0	0
No Parameters												

Action—The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU. After this command, two additional clock cycles are required before the first character of the first row is put out.

STATUS FLAGS

	Operation	A ₀	Description	Data Bus							
				MSB				LSB			
Command	Read	1	Status Word	0	IE	IR	LP	IC	VE	DU	F0

IE — (Interrupt Enable) Set or reset by command. It enables vertical retrace interrupt. It is automatically set by a "Start Display" command and reset with the "Reset" command.

IR — (Interrupt Request) This flag is set at the beginning of display of the last row of the frame if the interrupt enable flag is set. It is reset after a status read operation.

LP — This flag is set when the light pen input (LPEN) is activated and the light pen registers have been loaded. This flag is automatically reset after a status read.

IC — (Improper Command) This flag is set when a command parameter string is too long or

too short. The flag is automatically reset after a status read.

VE — (Video Enable) This flag indicates that video operation of the CRT is enabled. This flag is set on a "Start Display" command, and reset on a "Stop Display" or "Reset" command.

DU — (DMA Underrun) This flag is set whenever a data underrun occurs during DMA transfers. Upon detection of DU, the DMA operation is stopped and the screen is blanked until after the vertical retrace interval. This flag is reset after a status read.

FO — (FIFO Overrun) This flag is set whenever the FIFO is overrun. It is reset on a status read.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C

Storage Temperature -65°C to +150°C

Voltage On Any Pin

With Respect to Ground -0.5V to +7V

Power Dissipation 1 Watt

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

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ± 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} + 0.5V	V	
V _{OL}	Output Low Voltage		0.45	V	I _{OL} = 2.2 mA
V _{OH}	Output High Voltage	2.4		V	I _{OH} = -400 μA
I _{IL}	Input Load Current		± 10	μA	V _{IN} = V _{CC} to 0V
I _{OFL}	Output Float Leakage		± 10	μA	V _{OUT} = V _{CC} to 0.45V
I _{CC}	V _{CC} Supply Current		160	mA	

CAPACITANCE $T_A = 25^\circ\text{C}$, $V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	$f_c = 1\text{ MHz}$ Unmeasured pins returned to V_{SS}
$C_{I/O}$	I/O Capacitance		20	pF	

A.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5.0\text{V} \pm 5\%$, $\text{GND} = 0\text{V}$
Bus Parameters
READ CYCLE

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{AR}	Address Stable before READ	0		ns	
t_{RA}	Address Hold Time for READ	0		ns	
t_{RR}	READ Pulse Width	250		ns	
t_{RD}	Data Delay from READ		200	ns	$C_L = 150\text{ pF}$
t_{DF}	READ to Data Floating		100	ns	$C_L = 150\text{ pF}$

WRITE CYCLE

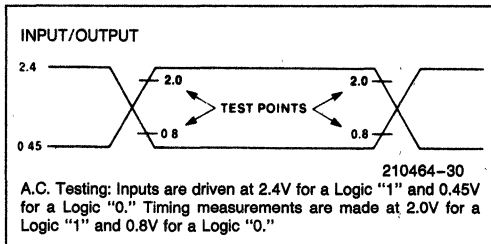
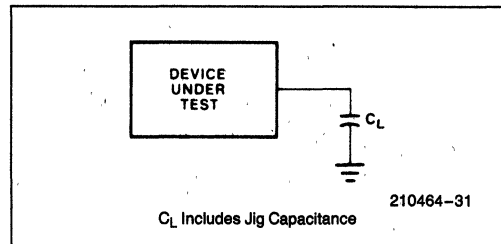
Symbol	Parameter	Min	Max	Units	Test Conditions
t_{AW}	Address Stable before WRITE	0		ns	
t_{WA}	Address Hold Time for WRITE	0		ns	
t_{WW}	WRITE Pulse Width	250		ns	
t_{DW}	Data Setup Time for WRITE	150		ns	
t_{WD}	Data Hold Time for WRITE	0		ns	

CLOCK TIMING

Symbol	Parameter	8275		8275-2		Units	Test Conditions
		Min	Max	Min	Max		
t_{CLK}	Clock Period	480		320		ns	
t_{KH}	Clock High	240		120		ns	
t_{KL}	Clock Low	160		120		ns	
t_{KR}	Clock Rise	5	30	5	30	ns	
t_{KF}	Clock Fall	5	30	5	30	ns	

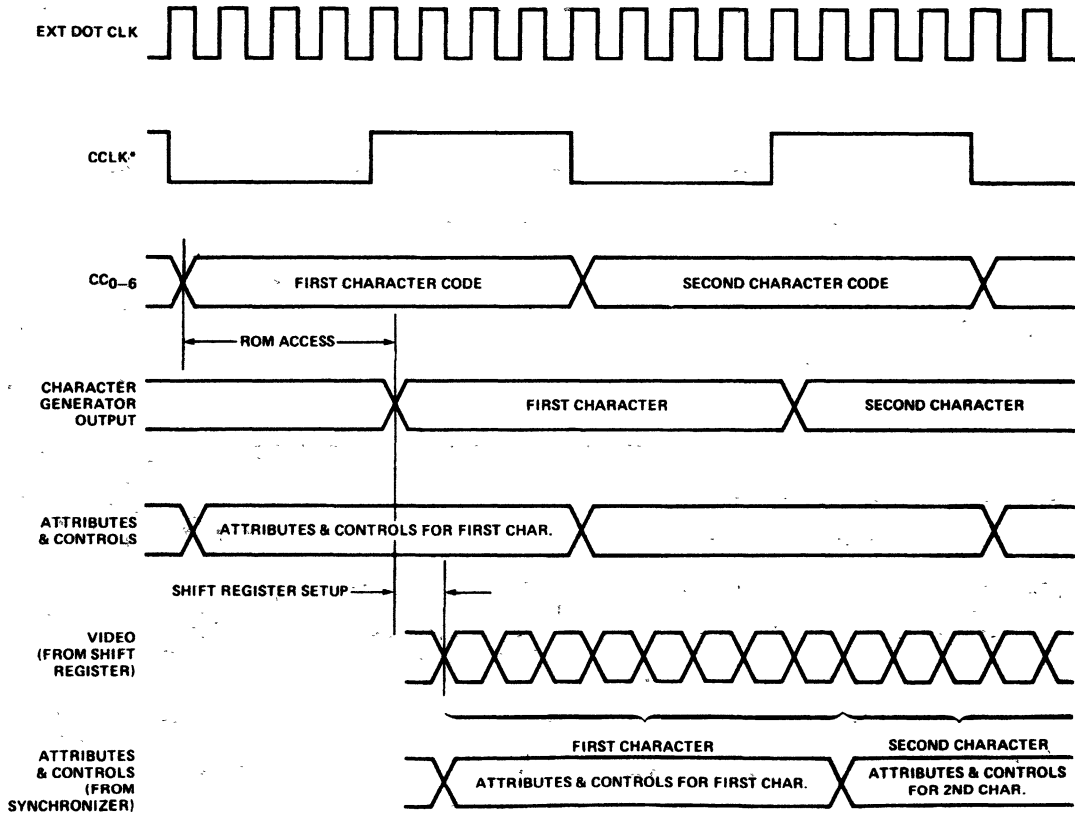
OTHER TIMING

Symbol	Parameter	8275		8275-2		Units	Test Conditions
		Min	Max	Min	Max		
t_{CC}	Character Code Output Delay		150		150	ns	$C_L = 50 \text{ pF}$
t_{HR}	Horizontal Retrace Output Delay		200		150	ns	$C_L = 50 \text{ pF}$
t_{LC}	Line Count Output Delay		400		250	ns	$C_L = 50 \text{ pF}$
t_{AT}	Control/Attribute Output Delay		275		250	ns	$C_L = 50 \text{ pF}$
t_{VR}	Vertical Retrace Output Delay		275		250	ns	$C_L = 50 \text{ pF}$
t_{RI}	IRQ \downarrow from RD \uparrow		250		250	ns	$C_L = 50 \text{ pF}$
t_{WQ}	DRQ \uparrow from WR \uparrow		250		250	ns	$C_L = 50 \text{ pF}$
t_{RQ}	DRQ \downarrow from WR \downarrow		200		200	ns	$C_L = 50 \text{ pF}$
t_{LR}	DACK \downarrow to WR \downarrow	0		0		ns	
t_{RL}	WR \uparrow to DACK \uparrow	0		0		ns	
t_{PR}	LPEN Rise		50		50	ns	
t_{PH}	LPEN Hold	100		100		ns	
t_{DI}	DACK Inactive Period	120				ns	

A.C. TESTING INPUT, OUTPUT WAVEFORM

A.C. TESTING LOAD CIRCUIT


WAVEFORMS

TYPICAL DOT LEVEL TIMING

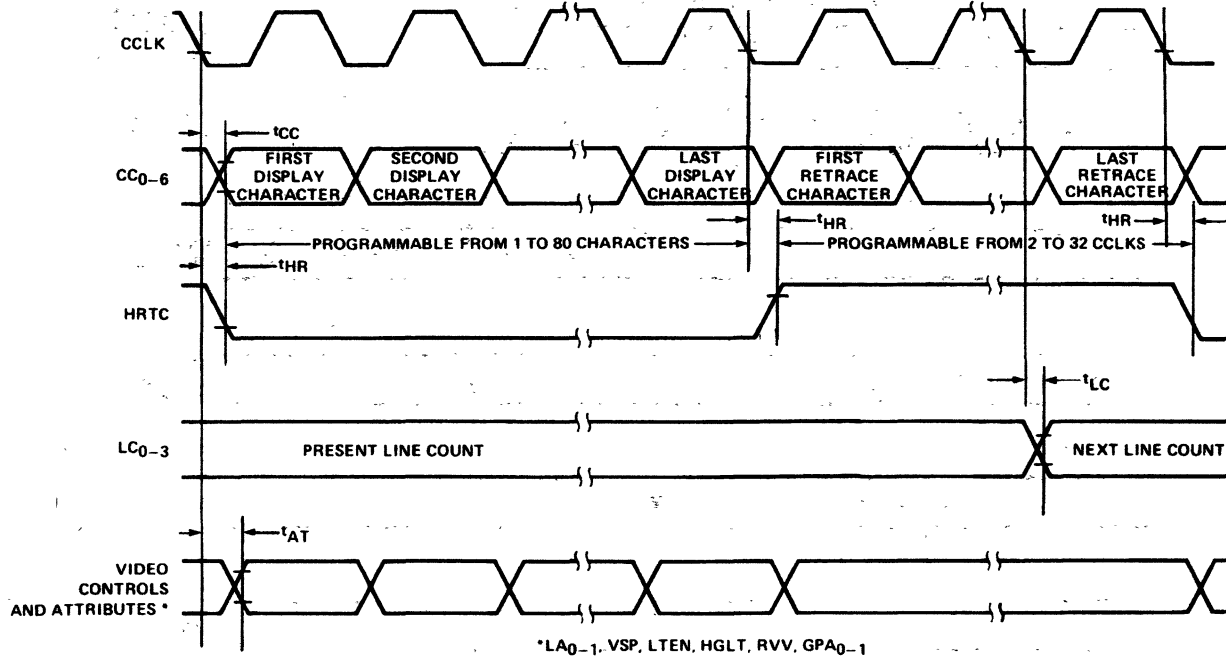


*CCLK IS A MULTIPLE OF THE DOT-CLOCK AND AN INPUT TO THE 8275.

210464-32

WAVEFORMS (Continued)

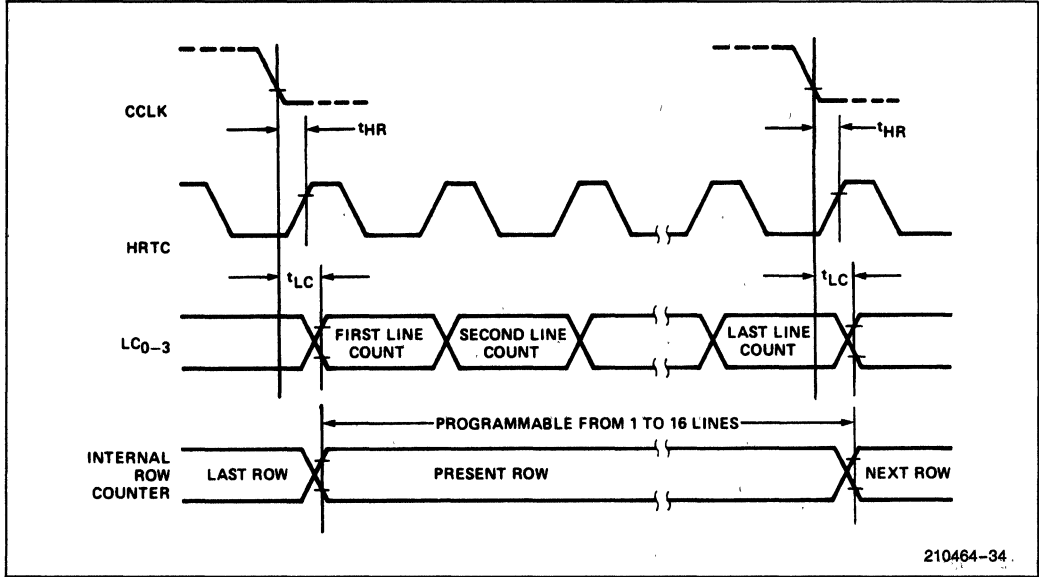
LINE TIMING



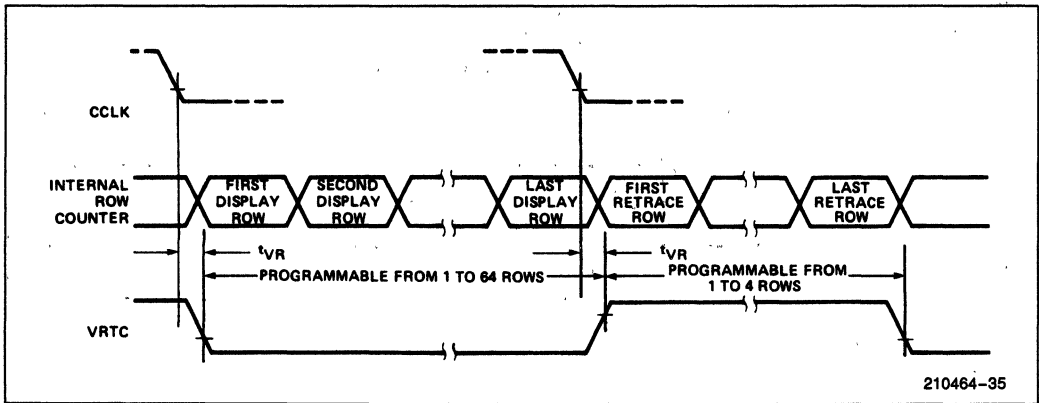
210464-33

WAVEFORMS (Continued)

ROW TIMING

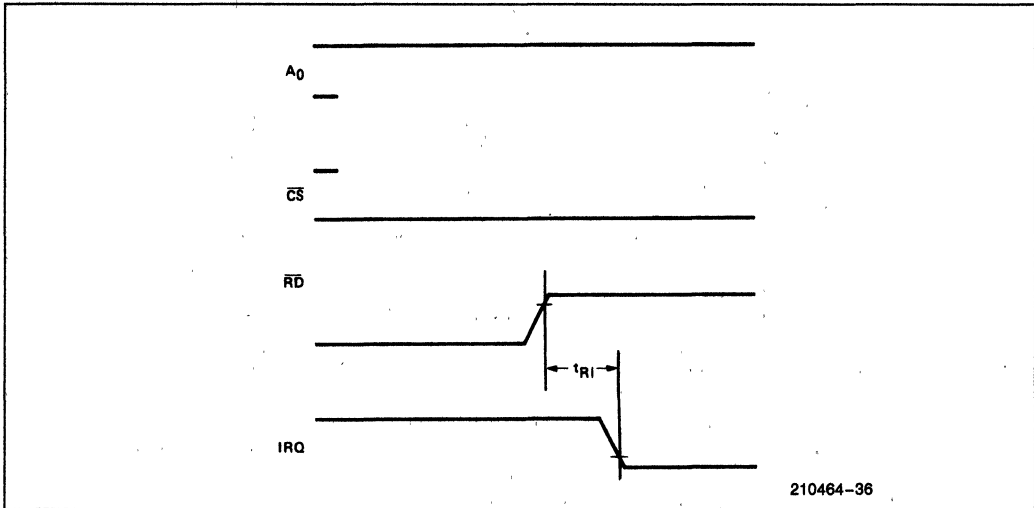


FRAME TIMING

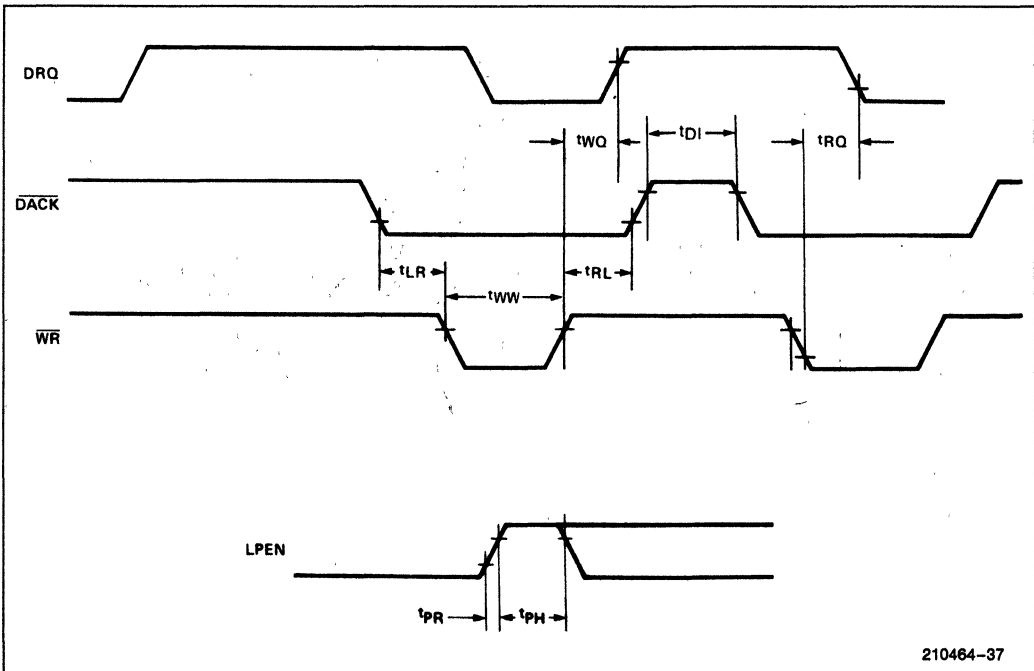


WAVEFORMS (Continued)

INTERRUPT TIMING

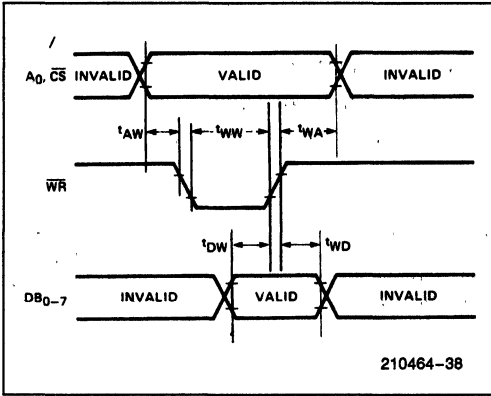


DMA TIMING

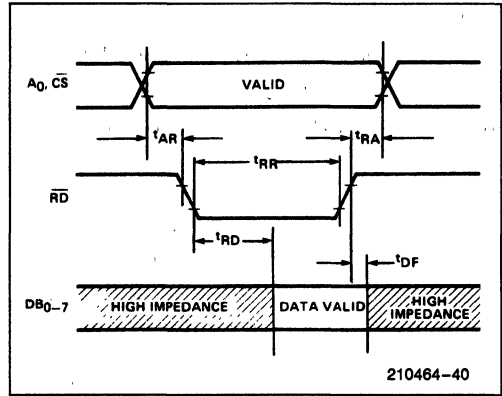


WAVEFORMS (Continued)

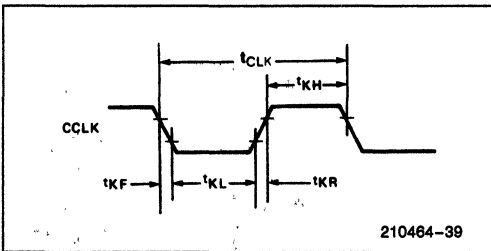
WRITE TIMING



READ TIMING



CLOCK TIMING





8276H SMALL SYSTEM CRT CONTROLLER

- Programmable Screen and Character Format
- 6 Independent Visual Field Attributes
- Cursor Control (4 Types)
- MCS-51®, MCS-85®, iAPX 86, and iAPX 88 Compatible
- Dual Row Buffers
- Single +5V Supply
- 40-Pin Package
- 3 MHz Clock with 8276-2
- High Performance HMOS-II

The Intel 8276H Small System CRT Controller is a single chip device intended to interface CRT raster scan displays with Intel microcomputers in minimum device-count systems. Its primary function is to refresh the display by buffering character information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8276H will allow simple interface to almost any raster scan CRT display. It can be used with the 8051 Single Chip Microcomputer for a minimum IC count design. It is manufactured on Intel's advanced HMOS-II process.

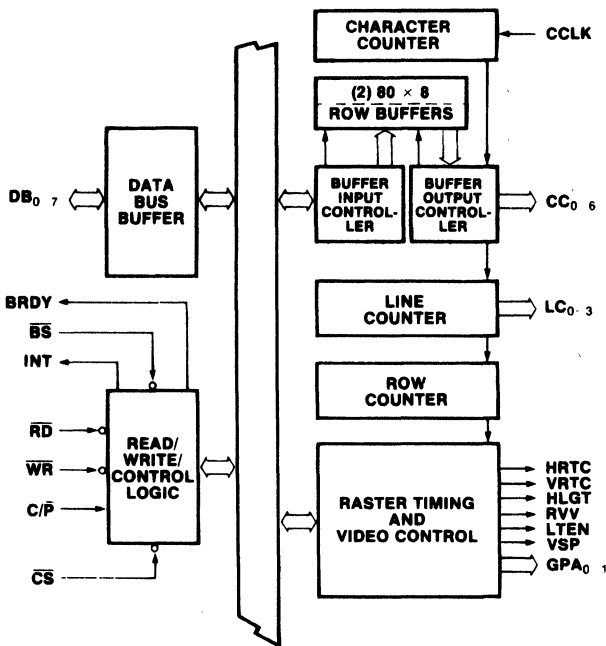


Figure 1. Block Diagram

210668-1

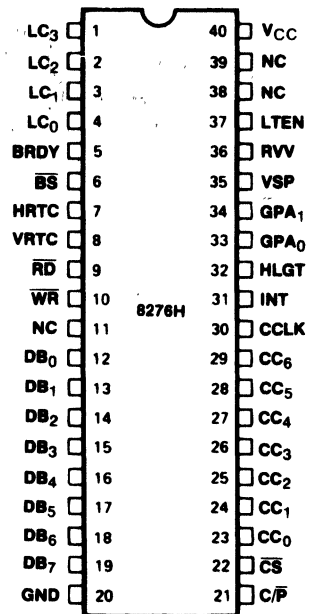


Figure 2. Pin configuration

210668-2

Table 1. Pin Descriptions

Symbol	Pin No.	Type	Name and Function
LC ₃ LC ₂ LC ₁ LC ₀	1 2 3 4	O	LINE COUNT: Output from the line counter which is used to address the character generator for the line positions on the screen.
BRDY	5	O	BUFFER READY: Output signal indicating that a Row Buffer is ready for loading of character data.
\overline{BS}	6	I	BUFFER SELECT: Input signal enabling \overline{WR} for character data into the Row Buffers.
HRTC	7	O	HORIZONTAL RETRACE: Output signal which is active during the programmed horizontal retrace interval. During this period the VSP output is high and the LTEN output is low.
VRTC	8	O	VERTICAL RETRACE: Output signal which is active during the programmed vertical retrace interval. During this period the VSP output is high and the LTEN output is low.
\overline{RD}	9	I	READ INPUT: A control signal to read registers.
\overline{WR}	10	I	WRITE INPUT: A control signal to write commands into the control registers or write data into the row buffers.
NC	11		No Connection.
DB ₀ DB ₁ DB ₂ DB ₃ DB ₄ DB ₅ DB ₆ DB ₇	12 13 14 15 16 17 18 19	I/O	BIDIRECTIONAL DATA BUS: Three-state lines. The outputs are enabled during a read of the C or P ports.
Ground	20		Ground.
V _{CC}	40		+ 5V Power Supply.
NC	39		No Connection.
NC	38		No Connection.
LTN	37	O	LIGHT ENABLE: Output signal used to enable the video signal to the CRT. This output is active at the programmed underline cursor position, and at positions specified by attribute codes.
RVV	36	O	REVERSE VIDEO. Output signal used to activate the CRT circuitry to reverse the video signal. This output is active at the cursor position if a reverse video block cursor is programmed or at the positions specified by the field attribute codes.
VSP	35	O	VIDEO SUPPRESSION. Output signal used to blank the video signal to the CRT. This output is active: —during the horizontal and vertical retrace intervals. —at the top and bottom lines of rows if underline is programmed to be number 8 or greater. —when an end of row or end of screen code is detected. —when a Row Buffer underrun occurs. —at regular intervals ($1/16$ frame frequency for cursor, $1/32$ frame frequency for attributes)—to create blinking displays as specified by cursor or field attribute programming.
GPA ₁ GPA ₀	34 33	O	GENERAL PURPOSE ATTRIBUTE CODES: Outputs which are enabled by the general purpose field attribute codes.
HLGT	32	O	HIGHLIGHT: Output signal used to intensify the display at particular positions on the screen as specified by the field attribute codes.
INT	31	O	INTERRUPT OUTPUT.
CCLK	30	I	CHARACTER CLOCK: (from dot/timing logic).

Table 1. Pin Descriptions (Continued)

Symbol	Pin No.	Type	Name and Function
CC ₆ CC ₅ CC ₄ CC ₃ CC ₂ CC ₁ CC ₀	29 28 27 26 25 24 23	O	CHARACTER CODES: Output from the row buffers used for character selection in the character generator.
\overline{CS}	22	I	CHIP SELECT: Enables \overline{RD} of status or \overline{WR} of command or parameters.
C/P	21	I	PORT ADDRESS: A high input on this pin selects the "C" port or command registers and a low input selects the "P" port or parameter registers.

FUNCTIONAL DESCRIPTION

Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8276H to the system Data Bus.

This functional block accepts inputs from the System Control Bus and generates control signals for overall device operation. It contains the Command, Parameter, and Status Registers that store the various control formats for the device functional definition.

C/P	Operation	Register
0	Read	Reserved
0	Write	Parameter
1	Read	Status
1	Write	Command

\overline{RD} (READ)

A "low" on this input informs the 8276H that the CPU is reading status information from the 8276H.

\overline{WR} (WRITE)

A "low" on this input informs the 8276H that the CPU is writing data or control words to the 8276H.

\overline{CS} (CHIP SELECT)

A "low" on this input selects the 8276H for \overline{RD} or \overline{WR} of Commands, Status, and Parameters.

BRDY (BUFFER READY)

A "high" on this output indicates that the 8276H is ready to receive character data.

\overline{BS} (BUFFER SELECT)

A "low" on this input enables \overline{WR} of character data to the 8276H row buffers.

INT (INTERRUPT)

A "high" on this output informs the CPU that the 8276H needs interrupt service.

C/P	\overline{RD}	\overline{WR}	\overline{CS}	\overline{BS}	
0	0	1	0	1	Reserved
0	1	0	0	1	Write 8276H Parameter
1	0	1	0	1	Read 8276H Status
1	1	0	0	1	Write 8276H Command
X	1	0	1	0	Write 8276H Row Buffer
X	1	1	X	X	High Impedance
X	X	X	1	1	High Impedance

Character Counter

The Character Counter is a programmable counter that is used to determine the number of characters to be displayed per row and the length of the horizontal retrace interval. It is driven by the CCLK (Character Clock) input, which should be derived from the external dot clock.

Line Counter

The Line Counter is a programmable counter that is used to determine the number of horizontal lines (Raster Scans) per character row. Its outputs are used to address the external character generator.

Row Counter

The Row Counter is a programmable counter that is used to determine the number of character rows to be displayed per frame and length of the vertical retrace interval.

Raster Timing and Video Controls

The Raster Timing circuitry controls the timing of the HRTC (Horizontal Retrace) and VRTC (Vertical Retrace) outputs. The Video control circuitry controls the generation of HGLT (Highlight), RVV (Reverse Video), LTEN (Light Enable), VSP (Video Suppress), and GPA₀₋₁ (General Purpose Attribute) outputs.

Row Buffers

The Row Buffers are two 80-character buffers. They are filled from the microcomputer system memory with the character codes to be displayed. While one row buffer is displaying a row of characters, the other is being filled with the next row of characters.

Buffer Input/Output Controllers

The Buffer Input/Output Controllers decode the characters being placed in the row buffers. If the character is a field attribute or special code, they control the appropriate action. (Example: A "High-light" field attribute will cause the Buffer Output Controller to activate the HGLT output.)

SYSTEM OPERATION

The 8276H is programmable to a large number of different display formats. It provides raster timing,

display row buffering, visual attribute decoding and cursor timing.

It is designed to interface with standard character generators for dot matrix decoding. Dot level timing must be provided by external circuitry.

General Systems Operational Description

Display characters are retrieved from memory and displayed on a row-by-row basis. The 8276H has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and the number of character rows per frame are software programmable, providing easy interface to most CRT displays. (See Programming Section.)

The 8276H uses BRDY to request character data to fill the row buffer that is not being used for display.

The 8276H displays character rows one scan line at a time. The number of scan lines per character row, the underline position, and blanking of top and bottom lines are programmable. (See Programming Section.)

The 8276H provides special Control Codes which can be used to minimize overhead. It also provides

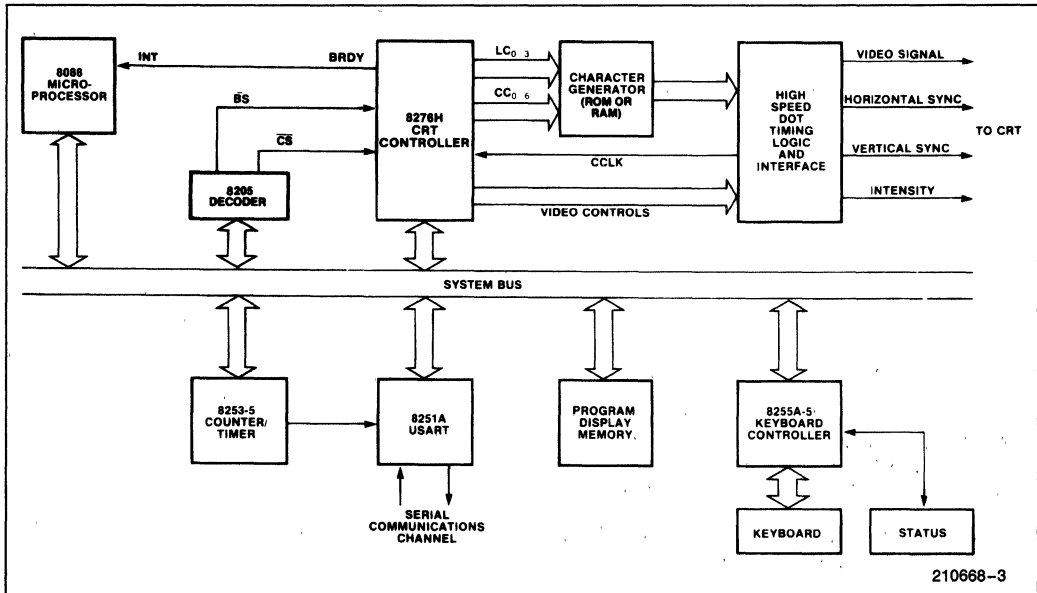


Figure 3. CRT System Block Diagram

210668-3

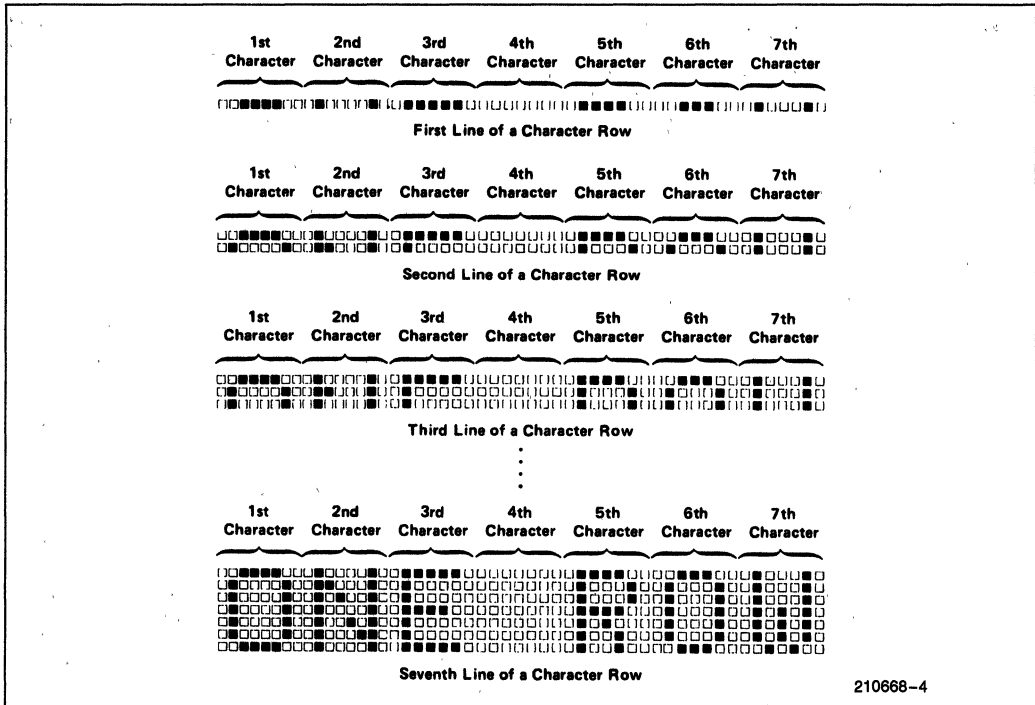


Figure 4. Display Of A Character Row

Visual Attribute Codes to cause special action on the screen without the use of the character generator. (See Visual Attributes Section.)

The 8276H also controls raster timing. This is done by generating Horizontal Retrace (HRTC) and Vertical Retrace (VRTC) signals. The timing of these signals is also programmable.

The 8276H can generate a cursor. Cursor location and format are programmable. (See Programming Section.)

Display Row Buffering

Before the start of a frame, the 8276H uses BRDY and \overline{BS} to fill one row buffer with characters.

When the first horizontal sweep is started, character codes are output to the character generator from the row buffer just filled. Simultaneously, the other row buffer is filled with the next row of characters.

After all the lines of the character row are scanned, the buffers are swapped and the same procedure is followed for the next row.

This process is repeated until all of the character rows are displayed.

Row Buffering allows the CPU access to the display memory at all times except during Buffer Loading (about 25%). This compares favorably to alternative approaches which restrict CPU access to the display memory to occur only during horizontal and vertical retrace intervals (80% of the bus time is used to refresh the display).

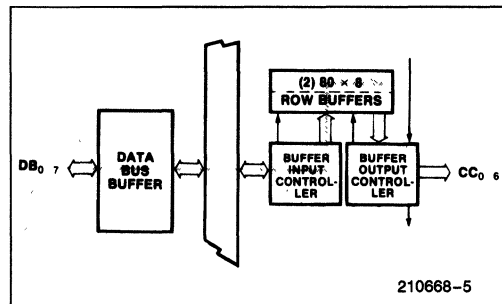


Figure 5. First Row Buffer Filled

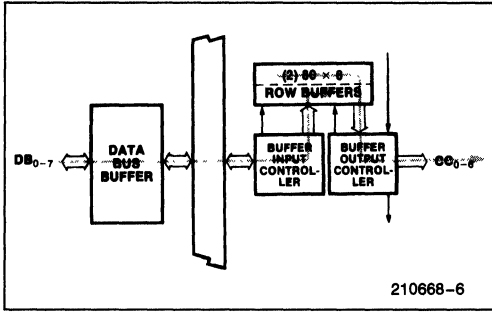


Figure 6. Second Row Buffer Filled, First Row Displayed

The 8276H can also be programmed to blank alternate rows. In this mode, the first row is displayed, the second blanked, the third displayed, etc. Display data is not requested for the blanked rows.

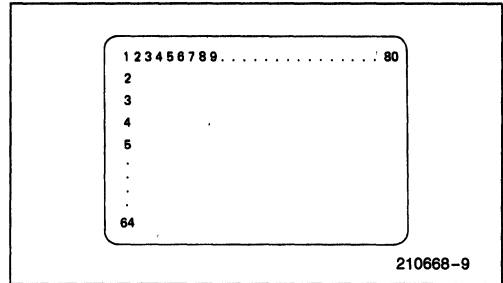


Figure 9. Blank Alternate Rows Mode

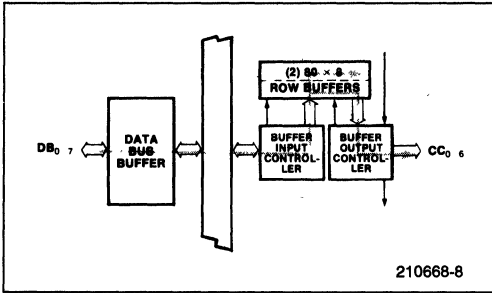


Figure 7. First Row Buffer Filled With Third Row, Second Row Displayed

ROW FORMAT

The 8276H is designed to hold the line count stable while outputting the appropriate character codes during each horizontal sweep. The line count is incremented during horizontal retrace and the whole row of character codes are output again during the next sweep. This is continued until the entire character row is displayed.

The number of lines (horizontal sweeps) per character row is programmable from 1 to 16.

The output of the line counter can be programmed to be in one of two modes.

In mode 0, the output of the line counter is the same as the line number.

In mode 1, the line counter is offset by one from the line number.

NOTE:

In mode 1, while the first line (line number 0) is being displayed, the last count is output by the line counter (see examples).

Display Format

SCREEN FORMAT

The 8276H can be programmed to generate from 1 to 80 characters per row, and from 1 to 64 rows per frame.

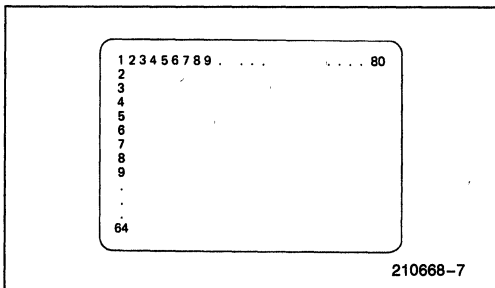


Figure 8. Screen Format

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1111
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000
10	1010	1001
11	1011	1010
12	1100	1011
13	1101	1100
14	1110	1101
15	1111	1110

210668-10

Figure 10. Example of a 16-Line Format

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1001
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000

210668-12

Figure 11. Example of a 10-Line Format

Mode 0 is useful for character generators that leave address zero blank and start at address 1. Mode 1 is useful for character generators which start at address zero.

Underline placement is also programmable (from line number 0 to 15). This is independent of the line counter mode.

If the line number of the underline is greater than 7 (line number MSB = 1), then the top and bottom lines will be blanked.

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1011
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000
10	1010	1001
11	1011	1010

210668-11

Top and Bottom Lines are Blanked

Figure 12. Underline in Line Number 10

If the line number of the underline is less than or equal to 7 (line number MSB = 0), then the top and bottom lines will not be blanked.

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	0111
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110

210668-13

Top and Bottom Lines are not Blanked

Figure 13. Underline in Line Number 7

If the line number of the underline is greater than the maximum number of lines, the underline will not appear.

Blanking is accomplished by the VSP (Video Suppression) signal. Underline is accomplished by the LTEN (Light Enable) signal.

DOT FORMAT

Dot width and character width are dependent upon the external timing and control circuitry.

Dot level timing circuitry should be designed to accept the parallel output of the character generator and shift it out serially at the rate required by the CRT display.

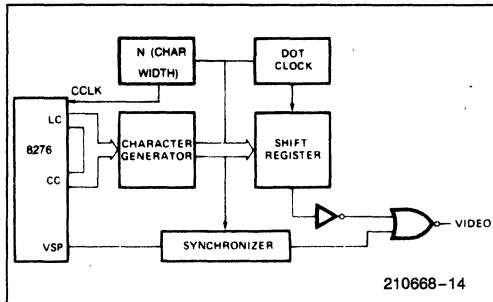


Figure 14. Typical Dot Level Block Diagram

Dot width is a function of dot clock frequency.

Character width is a function of the character generator width.

Horizontal character spacing is a function of the shift register length.

NOTE:

Video control and timing signals must be synchronized with the video signal due to the character generator access delay.

Raster Timing

The character counter is driven by the character clock input (CCLK). It counts out the characters being displayed (programmable from 1 to 80). It then causes the line counter to increment, and it starts counting out the horizontal retrace interval (programmable from 2 to 32). This process is constantly repeated.

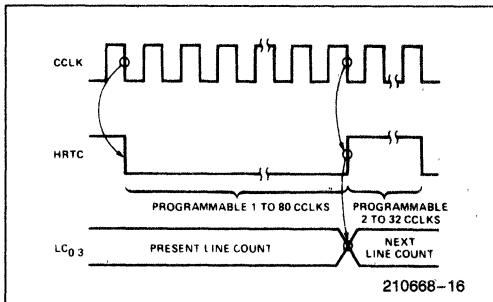


Figure 15. Line Timing

The line counter is driven by the character counter. It is used to generate the line address outputs (LC₀₋₃) for the character generator. After it counts all of the lines in a character row (programmable from 1 to 16), it increments the row counter, and starts over again. (See Character Format Section for detailed description of Line Counter functions.)

The row counter is an internal counter driven by the line counter. It controls the functions of the row buffers and counts the number of character rows displayed.

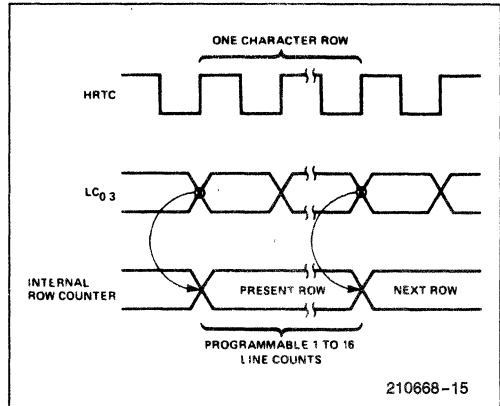


Figure 16. Row Timing

After the row counter counts all of the rows in a frame (programmable from 1 to 64), it starts counting out the vertical retrace interval (programmable from 1 to 4).

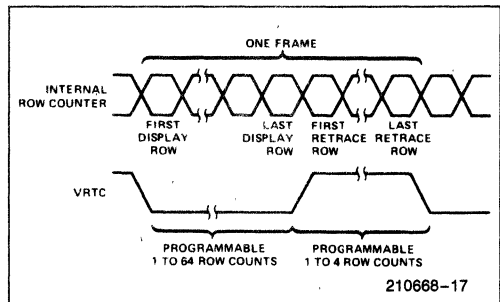


Figure 17. Frame Timing

The Video Suppression Output (VSP) is active during horizontal and vertical retrace intervals.

Dot level timing circuitry must synchronize these outputs with the video signal to the CRT Display.

Interrupt Timing

The 8276H can be programmed to generate an interrupt request at the end of each frame. If the 8276H interrupt enable flag is set, an interrupt request will occur at the *beginning of the last display row*.

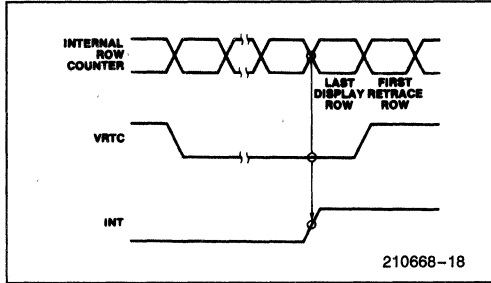


Figure 18. Beginning of Interrupt

INT will go inactive after the status register is read.

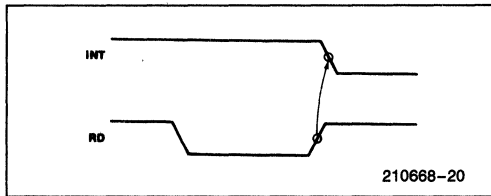


Figure 19. End of Interrupt

A reset command will also cause INT to go inactive, but this is not recommended during normal service.

NOTE:

Upon power-up, the 8276H Interrupt Enable Flag may be set. As a result, the user's cold start routine should write a reset command to the 8276H before system interrupts are enabled.

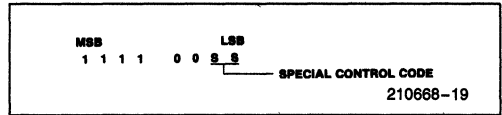
VISUAL ATTRIBUTES AND SPECIAL CODES

The characters processed by the 8276H are 8-bit quantities. The character code outputs provide the character generator with 7 bits of address. The Most Significant Bit is the extra bit and it is used to determine if it is a normal display character (MSB = 0), or if it is a Field Attribute or Special Code (MSB = 1).

Special Codes

Four special codes are available to help reduce bus usage.

SPECIAL CONTROL CHARACTER



S	S	Function
0	0	End of Row
0	1	End of Row-Stop Buffer Loading
1	0	End of Screen
1	1	End of Screen-Stop Buffer Loading

The End of Row Code (00) activates VSP and holds it to the end of the line.

The End of Row-Stop Buffer Loading (BRADY) Code (01) causes the Buffer Loading Control Logic to stop buffer loading for the rest of the row upon being written into the Row Buffer. It affects the display in the same way as the End of Row Code (00).

The End of Screen Code (10) activates VSP and holds it to the end of the frame.

The End of Screen-Stop Buffer Loading (BRDY) Code (11) causes the Row Buffer Control Logic to stop buffer loading for the rest of the frame upon being written. It affects the display in the same way as the End of Screen Code (10).

If the Stop Buffer Loading feature is not used, all characters after an End of Row character are ignored, except for the End of Screen character, which operates normally. All characters after an End of Screen character, are ignored.

NOTE:

If a Stop Buffer Loading is not the last character in a row, Buffer Loading is not stopped until after the next character is read. In this situation, a dummy character must be placed in memory after the Stop Buffer Loading character.

Field Attributes

The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the code up to, and including, the character which precedes the *next* field attribute code, or up to the end of the frame. The field attributes are reset during the vertical retrace interval.

The 8276H can be programmed to provide visible field attribute characters; all field attribute codes will occupy a position on the screen. These codes will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character.

There are six field attributes:

- 1) *Blink*—Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
- 2) *Highlight*—Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
- 3) *Reverse Video*—Characters following the code are caused to appear with reverse video by activating the Reverse Video output (RVV).
- 4) *Underline*—Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
- 5, 6) *General Purpose*—There are two additional 8276H outputs which act as general purpose, independently programmable field attributes. GPA₀₋₁ are active high outputs.

- H = 1 FOR HIGHLIGHTING
- B = 1 FOR BLINKING
- R = 1 FOR REVERSE VIDEO
- U = 1 FOR UNDERLINE
- GG = 1 FOR GPA₁, GPA₀

NOTE:

More than one attribute can be enabled at the same time. If the blinking and reverse video attributes are enabled simultaneously, only the reversed characters will blink.

Cursor Timing

The cursor location is determined by a cursor row register and a character position register which are loaded by command to the controller. The cursor can be programmed to appear on the display as:

1. a blinking underline
2. a blinking reverse video block
3. a non-blinking underline
4. a non-blinking reverse video block

The cursor blinking frequency is equal to the screen refresh frequency divided by 16.

If a non-blinking reverse video *cursor* appears in a non-blinking reverse video *field*, the cursor will appear as a normal video block.

If a non-blinking underline *cursor* appears in a non-blinking underline *field*, the cursor will not be visible.

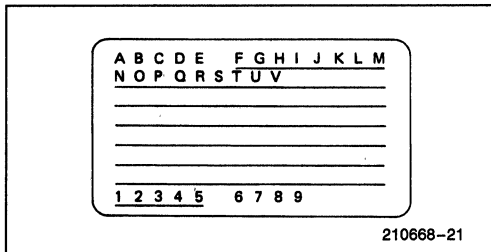
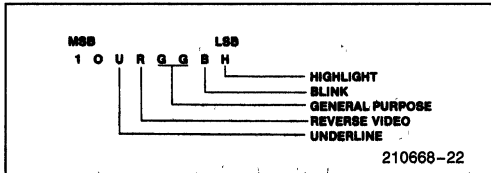


Figure 20. Example of a Visible Field Attribute (Underline Attribute)

FIELD ATTRIBUTE CODE



Device Programming

The 8276H has two programming registers, the Command Register and the Parameter Register. It also has a Status Register. The Command Register can only be written into and the Status Register can only be read from. They are addressed as follows:

C/P	Operation	Register
0	Read	Reserved
0	Write	Parameter
1	Read	Status
1	Write	Command

The 8276H expects to receive a command and a sequence of 0 to 4 parameters, depending on the command. If the proper number of parameter bytes are not received before another command is given, a status flag is set, indicating an improper command.

Instruction Set

The 8276H instruction set consists of 7 commands

Command	No. of Parameter Bytes
Reset	4
Start Display	0
Stop Display	0
Load Cursor	2
Enable Interrupt	0
Disable Interrupt	0
Preset Counters	0

In addition, the status of the 8276H can be read by the CPU at any time.

1. RESET COMMAND

Command	Operation	C/P	Description	Data Bus	
				MSB	LSB
Parameters	Write	1	Reset Command	0	0
	Write	0	Screen Comp Byte 1	S	H
	Write	0	Screen Comp Byte 2	V	R
	Write	0	Screen Comp Byte 3	U	U
	Write	0	Screen Comp Byte 4	M	1

Action—After the reset command is written, BRDY goes inactive, 8276H interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up.

As parameters are written, the screen composition is defined.

Parameter—S Spaced Rows

S	Functions
0	Normal Rows
1	Spaced Rows

Parameter—HHHHHHH Horizontal Characters/Row

H	H	H	H	H	H	H	No. Of Characters Per Row
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	2
0	0	0	0	0	1	0	3
			•				•
			•				•
			•				•
1	0	0	1	1	1	1	80
1	0	1	0	0	0	0	Undefined
			•				•
			•				•
			•				•
1	1	1	1	1	1	1	Undefined

Parameter—VV Vertical Retrace Row Count

V	V	No. Of Row Counts Per VRTC
0	0	1
0	1	2
1	0	3
1	1	4

Parameter—RRRRRR Vertical Rows/Frame

R	R	R	R	R	R	No. Of Rows/Frame
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
			•			•
			•			•
			•			•
1	1	1	1	1	1	64

Parameter—UUUU Underline Placement

U	U	U	U	Line Number Of Underline
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
		•		•
		•		•
		•		•
1	1	1	1	16

Parameter—LLL Number of Lines Per Character Row

L	L	L	L	No. Of Lines/Row
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
		•		•
		•		•
		•		•
1	1	1	1	16

Parameter—M Line Counter Mode

M	Line Counter Mode
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

Parameter—CC Cursor Format

C	C	Cursor Format
0	0	Blinking Reverse Video Block
0	1	Blinking Underline
1	0	Non-blinking Reverse Video Block
1	1	Non-blinking Underline

Parameter—ZZZZ Horizontal Retrace Count

Z	Z	Z	Z	No. Of Character Counts Per HRTC
0	0	0	0	2
0	0	0	1	4
0	0	1	0	6
.
.
1	1	1	1	32

NOTE:

uuuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

2. START DISPLAY COMMAND

	Operation	C/P	Description	Data Bus MSB	LSB
Command	Write	1	Start Display	0	0 1 0 0 0 0 0
No Parameters					

Action—8276H interrupts are enabled, BRDY goes active, video is enabled, Interrupt Enable and Video Enable status flags are set.

3. STOP DISPLAY COMMAND

	Operation	C/P	Description	Data Bus MSB	LSB
Command	Write	1	Stop Display	0	1 0 0 0 0 0 0
No Parameters					

Action—Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the “Start Display” command must be given to reenable the display.

4. LOAD CURSOR POSITION

	Operation	C/P	Description	Data Bus MSB	LSB
Command	Write	1	Load Cursor	1	0 0 0 0 0 0 0
Parameters	Write	0	Char. Number	(Char. Position in Row)	(Row Number)
	Write	0	Row Number	(Row Number)	(Row Number)

Action—The 8276H is conditioned to place the next two parameter bytes into the cursor position registers. Status flag not affected.

5. ENABLE INTERRUPT COMMAND

	Operation	C/P	Description	Data Bus MSB	LSB
Command	Write	1	Enable Interrupt	1	0 1 0 0 0 0 0
No Parameters					

Action—The interrupt enable flag is set and interrupts are enabled.

6. DISABLE INTERRUPT COMMAND

	Operation	C/P	Description	Data Bus MSB	LSB
Command	Write	1	Disable Interrupt	1	0 1 0 0 0 0 0
No Parameters					

Action—Interrupts are disabled and the interrupt enable status flag is reset.

7. PRESET COUNTERS COMMAND

	Operation	C/P	Description	Data Bus MSB	LSB
Command	Write	1	Preset Counters	1	1 1 0 0 0 0 0
No Parameters					

Action—The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU. After this command, two additional clock cycles are required before the first character of the first row is put out.

Status Flags

	Operation	C/P	Description	Data Bus MSB	LSB
Command	Read	1	Status Word	0	IE IR X IC VE BU X

- IE — (Interrupt Enable) Set or reset by command. It enables vertical retrace interrupt. It is automatically set by a “Start Display” command and reset with the “Reset” command.
- IR — (Interrupt Request) This flag is set at the beginning of display of the last row of the frame if the interrupt enable flag is set. It is reset after a status read operation.
- IC — (Improper Command) This flag is set when a command parameter string is too long or too short. The flag is automatically reset after a status read.
- VE — (Video Enable) This flag indicates that video operation of the CRT is enabled. This flag is set on a “Start Display” command, and reset on a “Stop Display” or “Reset” command.
- BU — (Buffer Underrun) This flag is set whenever a Row Buffer is not filled with character data in time for a buffer swap required by the display. Upon activation of this bit, buffer loading ceases, and the screen is blanked until after vertical retrace interval.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage On Any Pin
 With Respect to Ground -0.5V to +7V
 Power Dissipation 1 Watt

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

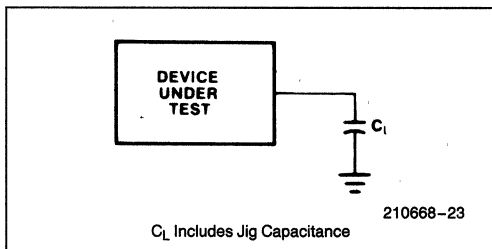
D.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to 70°C ; $V_{CC} = 5\text{V} \pm 5\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.5\text{V}$	V	
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.2\text{ mA}$
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -400\ \mu\text{A}$
I_{IL}	Input Load Current		± 10	μA	$V_{IN} = V_{CC}$ to 0V
I_{OFL}	Output Float Leakage		± 10	μA	$V_{OUT} = V_{CC}$ to 0.45V
I_{CC}	V_{CC} Supply Current		160	mA	

CAPACITANCE $T_A = 25^\circ\text{C}$; $V_{CC} = 25^\circ\text{C}$; $V_{CC} = \text{GND} = 0\text{V}$

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	$f_C = 1\text{ MHz}$ Unmeasured pins returned to V_{SS} .
$C_{I/O}$	I/O Capacitance		20	pF	

A.C. TESTING LOAD CIRCUIT



A.C. CHARACTERISTICS $T_A = 0^\circ\text{C to } 70^\circ\text{C}; V_{CC} = 5.0\text{V} \pm 5\%; \text{GND} = 0\text{V}$
Bus Parameters
READ CYCLE

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{AR}	Address Stable Before READ	0		ns	
t_{RA}	Address Hold Time for READ	0		ns	
t_{RR}	READ Pulse Width	250		ns	
t_{RD}	Data Delay from READ		200	ns	$C_L = 150 \text{ pF}$
t_{DF}	READ to Data Floating		100	ns	

WRITE CYCLE

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{AW}	Address Stable Before WRITE	0		ns	
t_{WA}	Address Hold Time for WRITE	0		ns	
t_{WW}	WRITE Pulse Width	250		ns	
t_{DW}	Data Setup Time for WRITE	150		ns	
t_{WD}	Data Hold Time for WRITE	0		ns	

CLOCK TIMING

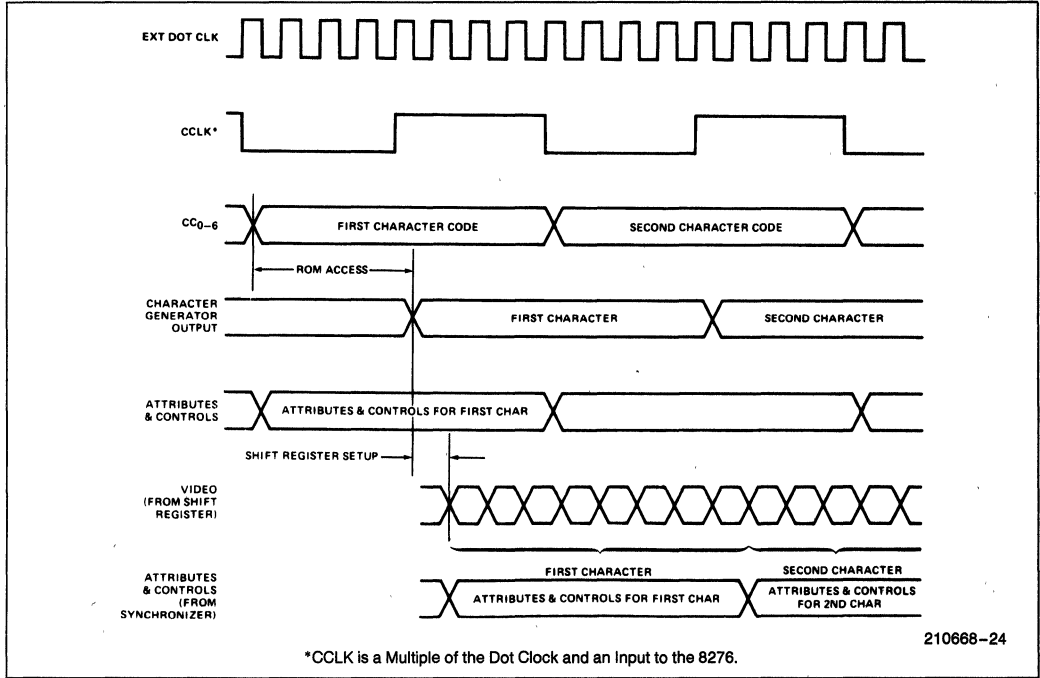
Symbol	Parameter	8276H		8276-2		Units	Test Conditions
		Min	Max	Min	Max		
t_{CLK}	Clock Period	480		320		ns	
t_{KH}	Clock High	240		120		ns	
t_{KL}	Clock Low	160		120		ns	
t_{KR}	Clock Rise	5	30	5	30	ns	
t_{KF}	Clock Fall	5	30	5	30	ns	

OTHER TIMING

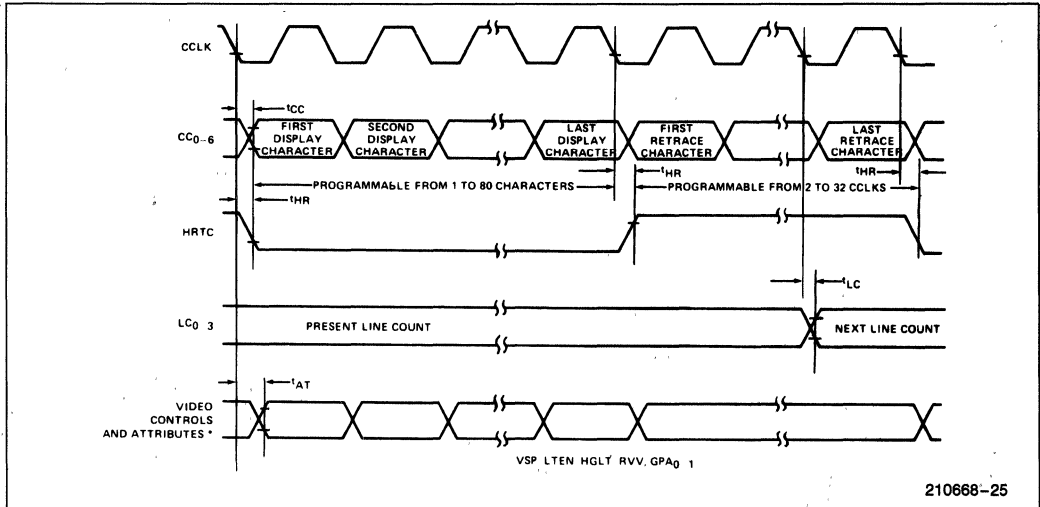
Symbol	Parameter	8276H		8276-2		Units	Test Conditions
		Min	Max	Min	Max		
t_{CC}	Character Code Output Delay		150		150	ns	$C_L = 50 \text{ pF}$
t_{HR}	Horizontal Retrace Output Delay		200		150	ns	$C_L = 50 \text{ pF}$
t_{LC}	Line Count Output Delay		400		250	ns	$C_L = 50 \text{ pF}$
t_{AT}	Control/Attribute Output Delay		275		250	ns	$C_L = 50 \text{ pF}$
t_{VR}	Vertical Retrace Output Delay		275		250	ns	$C_L = 50 \text{ pF}$
t_{RI}	INT \downarrow from RD \uparrow		250		250	ns	$C_L = 50 \text{ pF}$
t_{WQ}	BRDY \uparrow from WR \uparrow		250		250	ns	$C_L = 50 \text{ pF}$
t_{RQ}	BRDY \downarrow from WR \downarrow		200		200	ns	$C_L = 50 \text{ pF}$
t_{LR}	BS \downarrow to WR \downarrow	0		0		ns	
t_{RL}	WR \uparrow to BS \uparrow	0		0		ns	

WAVEFORMS

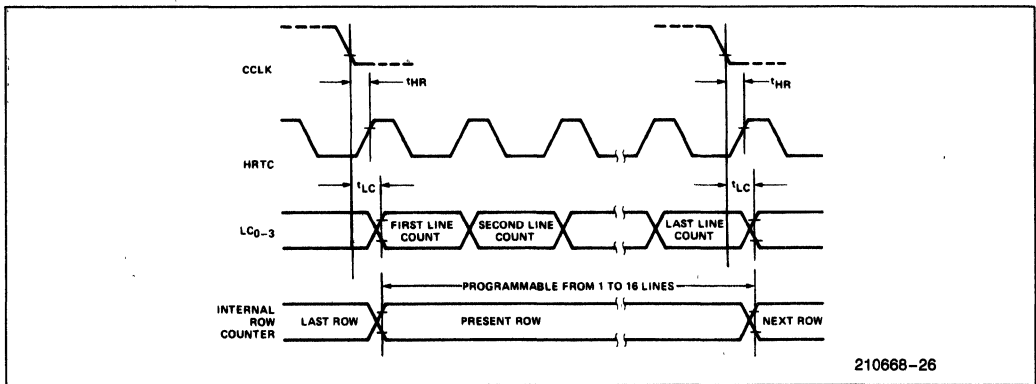
TYPICAL DOT LEVEL TIMING



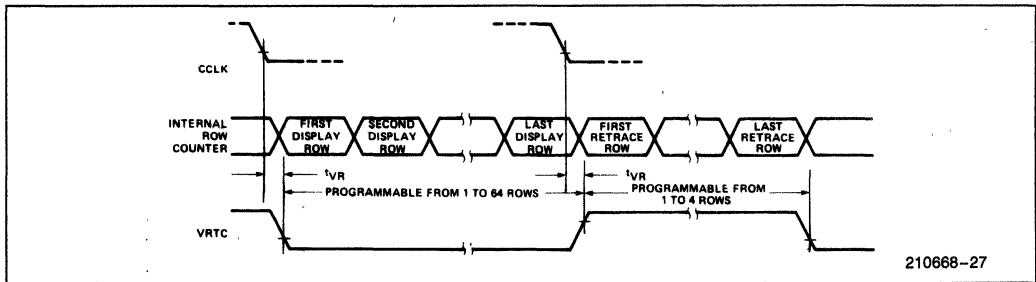
LINE TIMING



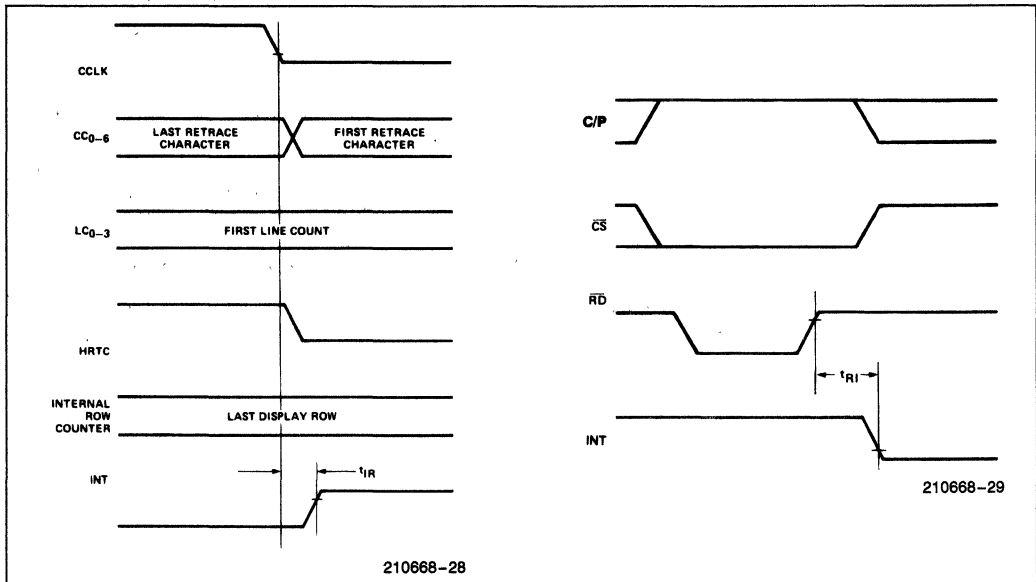
ROW TIMING



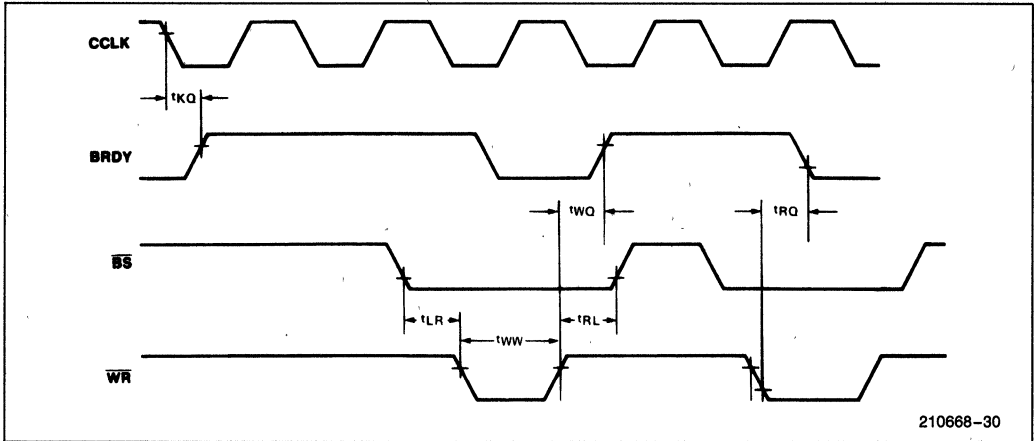
FRAME TIMING



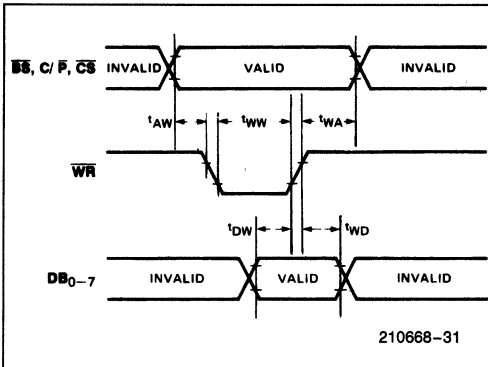
INTERRUPT TIMING



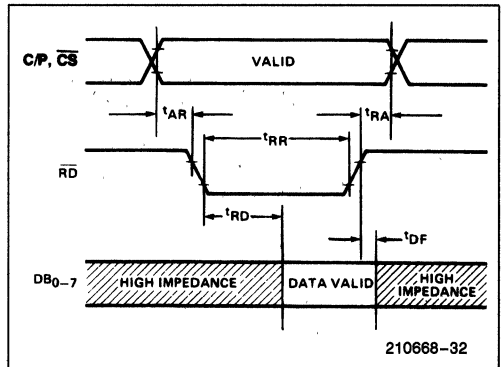
TIMING FOR BUFFER LOADING



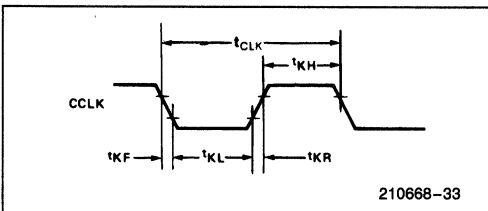
WRITE TIMING



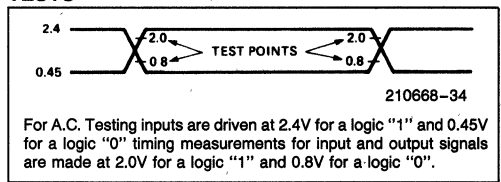
READ TIMING



CLOCK TIMING



INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS





APPLICATION NOTE

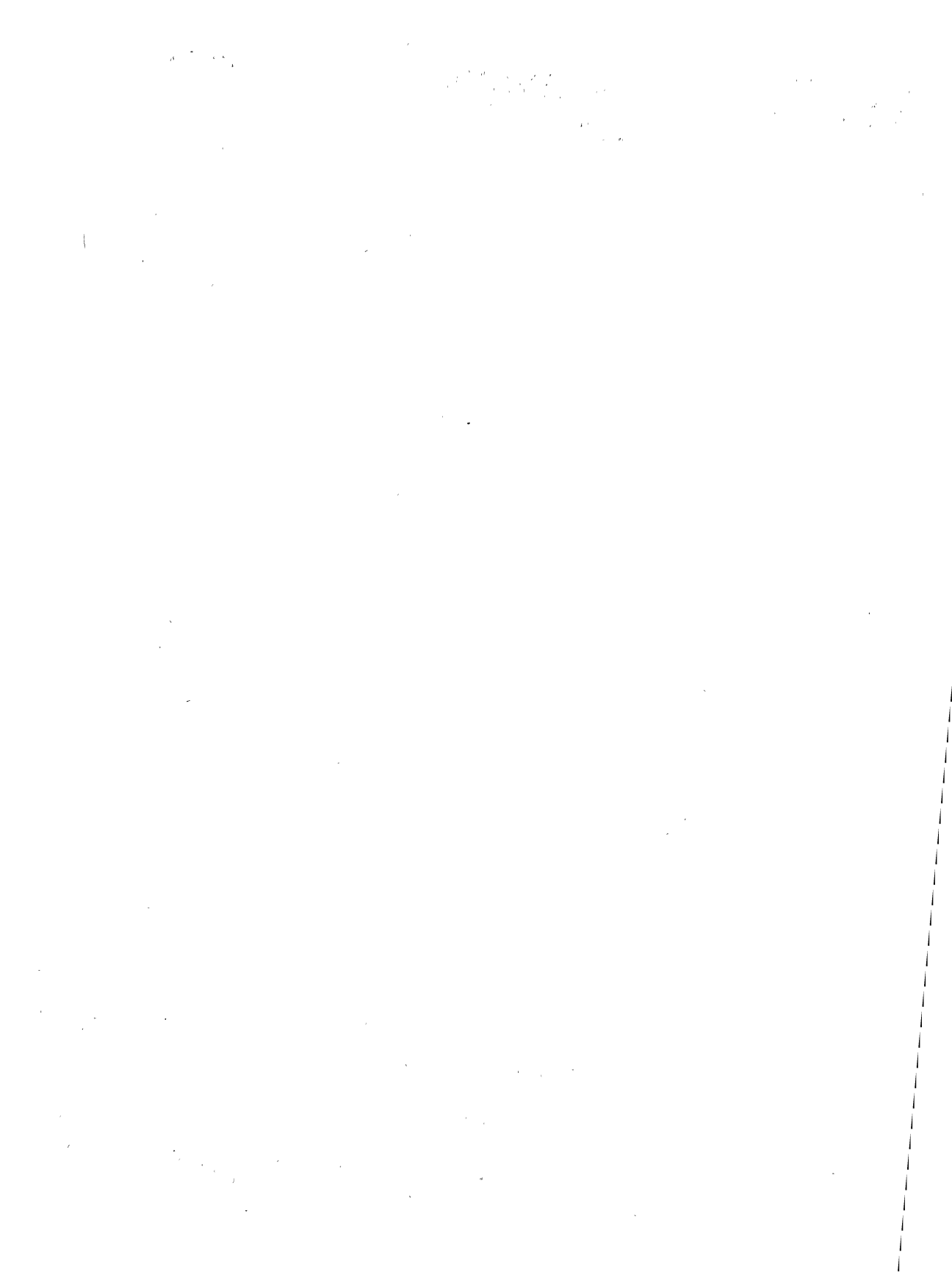
AP-62

November 1986

A Low Cost CRT Terminal Using the 8275

**JOHN KATAUSKY
PERIPHERALS APPLICATIONS**

Order Number: 207780-001



1.0 INTRODUCTION

The purpose of this application note is to provide the reader with the design concepts and factual tools needed to integrate Intel peripherals and microprocessors into a low cost raster scan CRT terminal. A previously published application note, AP-32, presented one possible solution to the CRT design question. This application note expands upon the theme established in AP-32 and demonstrates how to design a functional CRT terminal while keeping the parts count to a minimum.

For convenience, this application note is divided into seven general sections:

1. Introduction
2. CRT Basics
3. 8275 Description
4. Design Background
5. Circuit Description
6. Software Description
7. Appendix

There is no question that microprocessors and LSI peripherals have had a significant role in the evolution of CRT terminals. Microprocessors have allowed design engineers to incorporate an abundance of sophisticated features into terminals that were previously mere slaves to a larger processor. To complement microprocessors, LSI peripherals have reduced component count in many support areas. A typical LSI peripheral easily replaces between 30 and 70 SSI and MSI packages, and offers features and flexibility that are usually not available in most hardware designs. In addition to replacing a whole circuit board of random logic, LSI circuits also reduce the cost and increase the reliability of design. Fewer interconnects increases mechanical reliability and fewer parts decreases the power consumption and hence, the overall reliability of the design. The reduction of components also yields a circuit that is easier to debug during the actual manufacturing phase of a product.

Until the era of advanced LSI circuitry, a typical CRT terminal consisted of 80 to 200 or more SSI and MSI packages. The first microprocessors and peripherals dropped this component count to between 30 and 50 packages. This application note describes a CRT terminal that uses 20 packages.

2.0 CRT BASICS

The raster scan display gets its name from the fact that the image displayed on the CRT is built up by generating a series of lines (raster) across the face of the CRT. Usually, the beam starts in the upper left hand corner of the display and simultaneously moves left to right and top to bottom to put a series of zig-zag lines on

the screen (Figure 2.1). Two simultaneously operating independent circuits control the vertical and horizontal movement of the beam.

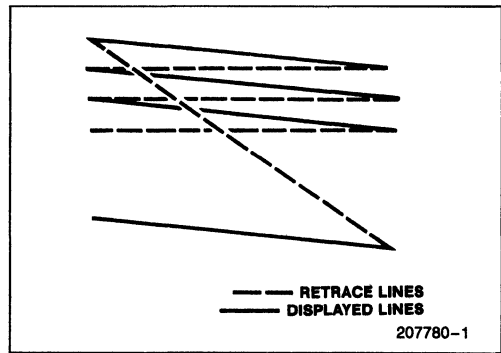


Figure 2-1. Raster Scan

As the electron beam moves across the face of the CRT, a third circuit controls the current flowing in the beam. By varying the current in the electron beam the image on the CRT can be made to be as bright or as dark as the user desires. This allows any desired pattern to be displayed.

When the beam reaches the end of a line, it is brought back to the beginning of the next line at a rate that is much faster than was used to generate the line. This action is referred to as "retrace". During the retrace period the electron beam is usually shut off so that it doesn't appear on the screen.

As the electron beam is moving across the screen horizontally, it is also moving downward. Because of this, each successive line starts slightly below the previous line. When the beam finally reaches the bottom right hand corner of the screen, it retraces vertically back to the top left hand corner. The time it takes for the beam to move from the top of the screen to the bottom and back again to the top is usually referred to as a "frame". In the United States, commercial television broadcast use 15,750 Hz as the horizontal sweep frequency (63.5 microseconds per horizontal line) and 60 Hz as the vertical sweep frequency or "frame" (16.67 milliseconds per vertical frame).

Although, the 60 Hz vertical frame and the 15,750 Hz horizontal line are the standards used by commercial broadcasts, they are by no means the only frequency at which CRTs can operate. In fact, many CRT displays use a horizontal scan that is around 18 KHz to 22 KHz and some even exceed 30 KHz. As the horizontal frequency increases, the number of horizontal lines per frame increases. Hence, the resolution on the vertical axis increases. This increased resolution is needed on high density graphic displays and on special text editing terminals that display many lines of text on the CRT.

Although many CRTs operate at non-standard horizontal frequencies, very few operate at vertical frequencies other than 60 Hz. If a vertical frequency other than 60 Hz is chosen, any external or internal magnetic or electrical variations at 60 Hz will modulate the electron beam and the image on the screen will be unstable. Since, in the United States, the power line frequency happens to be 60 Hz, there is a good chance for 60 Hz interference to exist. Transformers can cause 60 Hz magnetic fields and power supply ripple can cause 60 Hz electrical variations. To overcome this, special shielding and power supply regulation must be employed. In this design, we will assume a standard frame rate of 60 Hz and a standard line rate of 15,750 Hz.

By dividing the 63.5 microsecond horizontal line rate into the 16.67 millisecond vertical rate, it is found that there are 262.5 horizontal lines per vertical frame. At first, the half line may seem a bit odd, but actually it allows the resolution on the CRT to be effectively doubled. This is done by inserting a second set of horizontal lines between the first set (interlacing). In an interlaced system the line sets are not generated simultaneously. In a 60 Hz system, first all of the even-numbered lines are scanned: 0, 2, 4, . . . 524. Then all the odd-numbered lines: 1, 3, 5, . . . 525. Each set of lines usually contains different data (Figure 2.2).

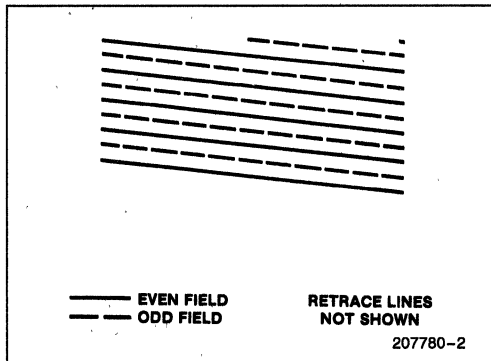


Figure 2-2. Interlaced Scan

Although interlacing provides greater resolution, it also has some distinct disadvantages. First of all, the circuitry needed to generate the extra half horizontal line per frame is quite complex when compared to a noninterlaced design, which requires an integer number of horizontal lines per frame. Next, the overall vertical refresh rate is half that of a noninterlaced display. As a result, flicker may result when the CRT uses high speed phosphors. To keep things as simple as possible, this design uses the noninterlaced approach.

The first thing any CRT controller must do is generate pulses that define the horizontal line timing and the vertical frame timing. This is usually done by dividing a crystal reference source by some appropriate numbers. On most raster scan CRTs the horizontal frequency is very forgiving and can vary by around 500 Hz or so and produce no ill effects. This means that the CRT itself can track a horizontal frequency between 15250 Hz and 16250 Hz, or in other words, there can be 256 to 270 horizontal lines per vertical frame. But, as mentioned earlier, the vertical frequency should be 60 Hz to insure stability.

The characters that are viewed on the screen are formed by a series of dots that are shifted out of the controller while the electron beam moves across the face of the CRT. The circuits that create this timing are referred to as the dot clock and character clock. The character clock is equal to the dot clock divided by the number of dots used to form a character along the horizontal axis and the dot clock is calculated by the following equation:

$$\text{DOT CLOCK (Hz)} = (N + R) \cdot D \cdot L \cdot F$$

where N is the number of displayed characters per row, R is the number of retrace character time increments, D is the number of dots per character, L is the number of horizontal lines per frame and F is the frame rate in Hz.

In this design N = 80, R = 20, D = 7, L = 270, and F = 60 Hz. If the numbers are plugged in, the dot clock is found to be 11.34 MHz.

The retrace number, R, may vary from system to system because it is used to establish the margins on the left and right hand sides of the CRT. In this particular design R = 20 was empirically found to be optimum. The number of dots per character may vary depending on the character generator used and the number of dot clocks the designer wants to place between characters. This design uses a 5 x 7 dot matrix and allows 2 dot clock periods between characters (see Figure 2.3); since 5 + 2 equals 7, we find that D = 7.

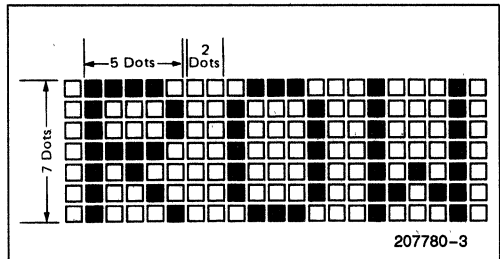


Figure 2-3. 5 x 7 Dot Matrix

The number of lines per frame can be determined by the following equation:

$$L = (H * Z) + V$$

where, H is the number of horizontal lines per character,

Z is the number of character lines per frame and

V is the number of horizontal lines during vertical retrace. In this design, a 5 × 7 dot matrix is to be placed on a 7 × 10 field, so H = 10. Also, 25 lines are to be displayed, so Z = 25. As mentioned before, V = 20. When the numbers are plugged into the equation, L is found to be equal to 270 lines per frame.

The designer should be cautioned that these numbers are interrelated and that to guarantee proper operation

on a standard raster scan CRT, L should be between 256 and 270. If L does not lie within these bounds the horizontal circuits of the CRT may not be able to lock onto the driving signal and the image will roll horizontally. The chosen L of 270 yields a horizontal frequency of 16,200 KHz on a 60 Hz frame and this number is within the 500 Hz tolerance mentioned earlier.

The V number is chosen to match the CRT in much the same manner as the R number mentioned earlier. When the electron beam reaches the bottom right corner of the screen it must retrace vertically to the top left corner. This retrace action requires time, usually between 900–1200 microseconds. To allow for this, enough horizontal sync times must be inserted during vertical retrace. Twenty horizontal sync times at

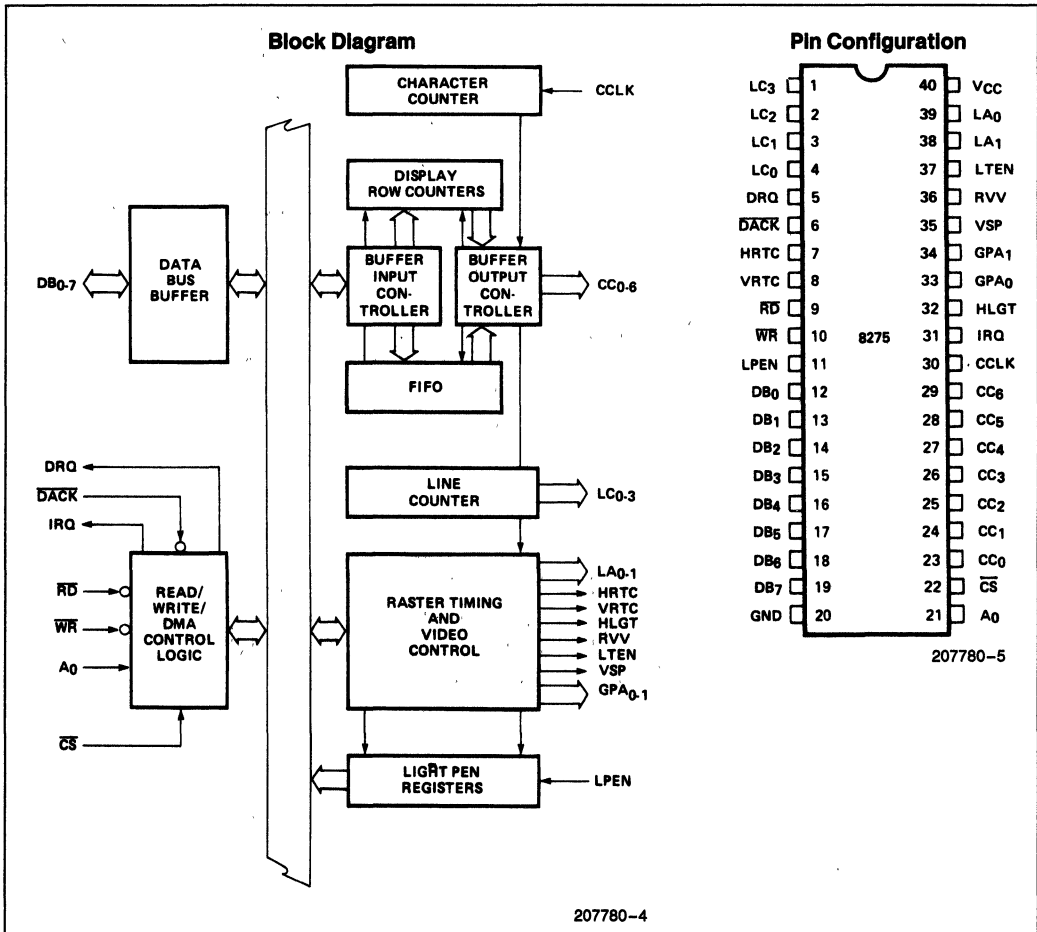


Figure 3-1. 8275 Block Diagram/Pin Configuration

61.5 microseconds yield a total of 1234.5 microseconds, which is enough time to allow the beam to return to the top of the screen.

The choices of H and Z largely relate to system design preference. As H increases, the character size along the vertical axis increases. Z is simply the number of lines of characters that are displayed and this, of course, is entirely a system design option.

3.0 8275 DESCRIPTION

A block diagram and pin configuration of the 8275 are shown in Figure 3.1. The following is a description of the general capabilities of the 8275.

3.1 CRT Display Refreshing

The 8275, having been programmed by the designer to a specific screen format, generates a series of DMA request signals, resulting in the transfer of a row of characters from display memory to the 8275's row buffers. The 8275 presents the character codes to an external character generator ROM by using outputs CC0-CC6. External dot timing logic is then used to transfer the parallel output data from the character generator ROM serially to the video input of the CRT. The character rows are displayed on the CRT one line at a time. Line count outputs LC0-LC3 are applied to the character generator ROM to perform the line selec-

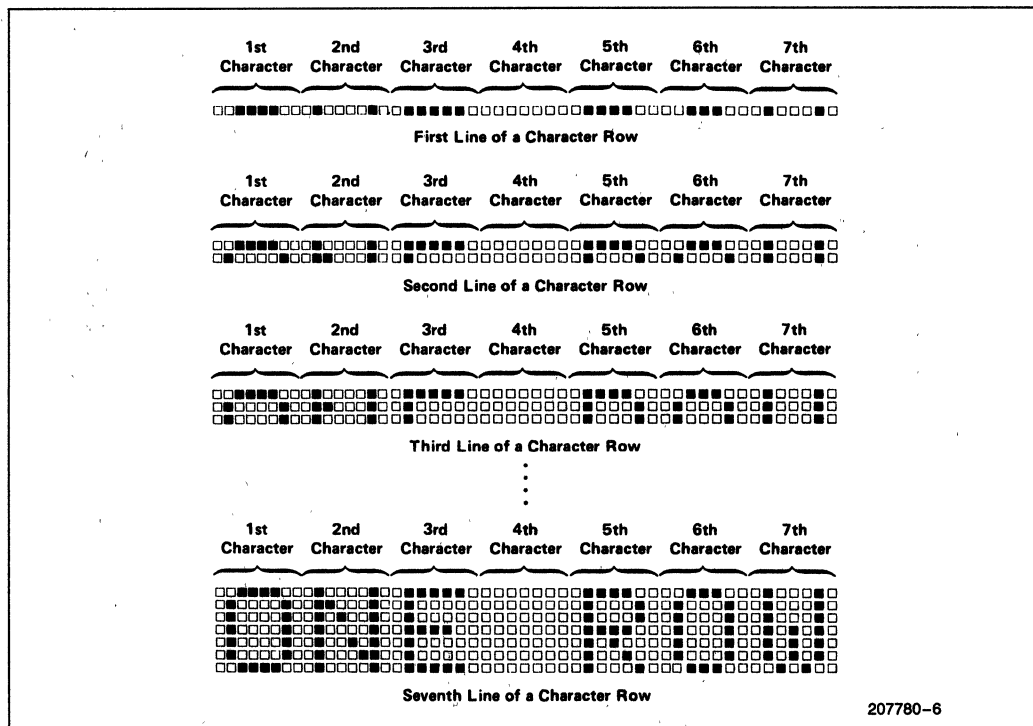


Figure 3-2. 8275 Row Display

tion function. The display process is illustrated in Figure 3.2. The entire process is repeated for each display row. At the beginning of the last displayed row, the 8275 issues an interrupt by setting the IRQ output line. The 8275 interrupt output will normally be connected to the interrupt input of the system central processor. The interrupt causes the CPU to execute an interrupt service subroutine. The service subroutine typically re-initializes DMA controller parameters for the next display refresh cycle, polls the system keyboard controller, and/or executes other appropriate functions. A block diagram of a CRT system implemented with the 8275 CRT Controller is provided in Figure 3.3. Proper CRT refreshing requires that certain 8275 parameters be programmed prior to the beginning of display operation. The 8275 has two types of programming registers, the Command Registers (CREG) and the Parameter Regis-

ters (PREG). It also has a Status Register (SREG). The Command Registers may only be written to and the Status Registers may only be read. The 8275 expects to receive a command followed by a sequence of from 0 to 4 parameters, depending on the command. The 8275 instruction set consist of the eight commands shown in Figure 3.4.

To establish the format of the display, the 8275 provides a number of user programmable display format parameters. Display formats having from 1 to 80 characters per row, 1 to 64 rows per screen, and 1 to 16 horizontal lines per row are available.

In addition to transferring characters from memory to the CRT screen, the 8275 features cursor position control. The cursor position may be programmed, via X

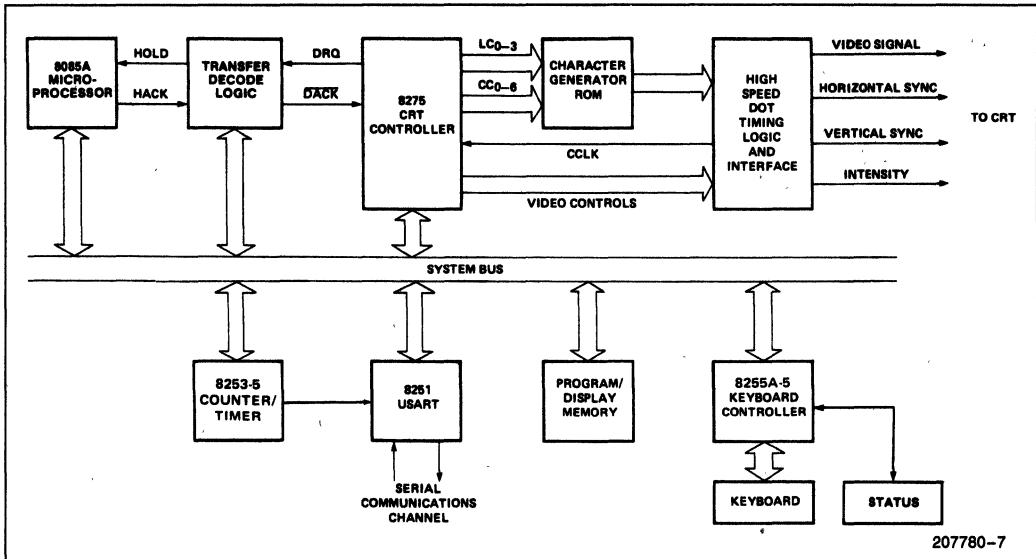


Figure 3-3. CRT System Block Diagram

and Y cursor position registers, to any character position on the display. The user may select from four cursor formats. Blinking or non-blinking underline and reverse video block cursors are available.

3.2 CRT Timing

The 8275 provides two timing outputs, HRTC and VRTC, which are utilized in synchronizing CRT horizontal and vertical oscillators to the 8275 refresh cycle. In addition, whenever HRTC or VRTC is active, a third timing output, VSP (Video Suppress) is true, providing a blinking signal to the dot timing logic. The dot timing logic will normally inhibit the video output to the CRT during the time when video suppress signal is true. An additional timing output, LTEN (Light Enable) is used to provide the ability to force the video output high regardless of the state of VSP. This feature is used by the 8275 to place a cursor on the screen and to control attribute functions. Attributes will be considered in the next section.

The HLGT (Highlight) output allows an attribute function to increase the CRT beam intensity to a level great-

er than normal. The fifth timing signal, RVV (Reverse Video) will, when enabled, cause the system video output to be inverted.

Command	No. of Parameter Bytes	Notes
RESET	4	Display format parameters required
START DISPLAY	0	DMA operation parameters included in command
STOP DISPLAY	0	—
READ LIGHT PEN	2	—
LOAD CURSOR	2	Cursor X,Y position parameters required
ENABLE INTERRUPT	0	—
DISABLE INTERRUPT	0	—
PRESET COUNTERS	0	Clears all internal counters

Figure 3-4. 8275's Instruction Set

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"	OUTPUTS				SYMBOL	DESCRIPTION	
	LA ₁	LA ₀	VSP	LTEN			
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

207780-8

NOTES:

*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

Blinking is active when B = 1.

Highlight is active when H = 1.

Figure 3-5. Character Attributes

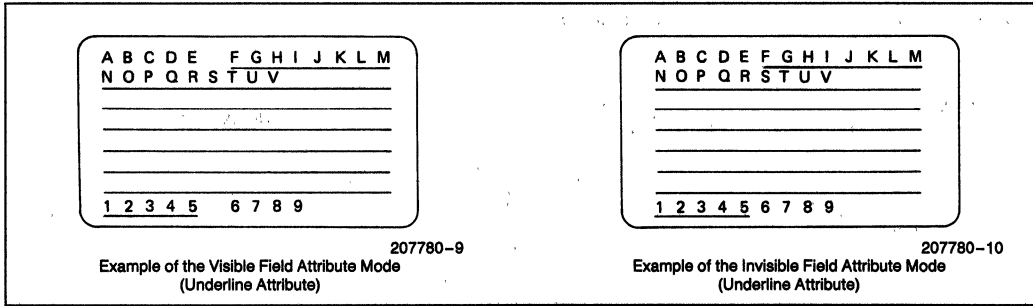


Figure 3-6. Field Attribute Examples

3.3 Special Functions

VISUAL ATTRIBUTES—Visual attributes are special codes which, when retrieved from display memory by the 8275, affect the visual characteristics of a character position or field of characters. Two types of visual attributes exist, character attributes and field attributes.

Character Attribute Codes: Character attribute codes can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA0-LA1), the Video Suppression output (VSP), and the Light Enable output (LTEN). The dot timing logic uses these signals to generate the proper symbols. Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT). Character attributes were designed to produce the graphic symbols shown in Figure 3.5.

Field Attribute Codes: The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the field attribute code up to, and including, the character which precedes the next field attribute code, or up to the end of the frame.

There are six field attributes:

1. *Blink*—Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. *Highlight*—Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. *Reverse Video*—Characters following the code are caused to appear in reverse video format by activating the Reverse Video output (RVV).

4. *Underline*—Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
5. *General Purpose*—There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. These attributes may be used to select colors or perform other desired control functions.

The 8275 can be programmed to provide visible or invisible field attribute characters as shown in Figure 3.6. If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character. If the 8275 is programmed in the invisible field attribute mode, the 8265 row buffer FIFOs are activated. The FIFOs effectively lengthen the row buffers by 16 characters, making room for up to 16 field attribute characters per display row. The FIFOs are 126 characters by 7 bits in size. When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the next character in the proper FIFO. When a field attribute is placed in the buffer output controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CCO-6). The chosen attributes are also activated.

LIGHT PEN DETECTION—A light pen consists fundamentally of a switch and light sensor. When the light pen is pressed against the CRT screen, the switch enables the light sensor. When the raster sweep coincides with the light sensor position on the display, the light pen output is input and the row and character position coordinates are stored in two 8275 internal registers. These registers can be read by the microprocessor.

SPECIAL CODES—Four special codes may be used to help reduce memory, software, or DMA overhead. These codes are placed in character positions in display memory.

1. *End of Row Code*—Activates VSP. VSP remains active until the end of the line is reached. While VSP is active, the screen is blanked.
2. *End of Row-Stop DMA Code*—Causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the row buffer. It affects the display in the same way as the End of Row Code.
3. *End of Screen Code*—Activates VSP. VSP remains active until the end of the frame is reached.
4. *End of Screen-Stop DMA Code*—Causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the row buffer. It affects the display in the same way as the End of Screen Code.

PROGRAMMABLE DMA BURST CONTROL—The 8275 can be programmed to request single-byte DMA transfers or DMA burst transfers of 2, 4, or 8 characters per burst. The interval between bursts is also programmable. This allows the user to tailor the DMA overhead to fit the system needs.

4.0 DESIGN BACKGROUND

4.1 Design Philosophy

Since the cost of any CRT system is somewhat proportional to parts count, arriving at a minimum part count solution without sacrificing performance has been the motivating force throughout this design effort. To successfully design a CRT terminal and keep the parts count to a minimum, a few things became immediately apparent.

1. An 8085 should be used.
2. Address and data buffering should be eliminated.
3. Multi-port memory should be eliminated.
4. DMA should be eliminated.

Decision 1 is obvious, the 8085's on-board clock generator, bus controller and vectored interrupts greatly reduce the overall part count considerably. Decision 2 is fairly obvious; if a circuit can be designed so that loading on the data and address lines is kept to a minimum, both the data and address buffers can be eliminated. This easily saves three to eight packages and reduces the power consumption of the design. Both decisions 3 and 4 require a basic understanding of current CRT design concepts.

In any CRT design, extreme time conflicts are created because all essential elements require access to the bus. The CPU needs to access the memory to control the system and to handle the incoming characters, but, at the same time, the CRT controller needs to access the memory to keep the raster scan display refreshed. To resolve this conflict two common techniques are employed, page buffering and line buffering.

In the page buffering approach the entire screen memory is isolated from the rest of the system. This isolation is usually accomplished with three-state buffers or two line to one line multiplexers. Of course, whenever a character needs to be manipulated the CPU must gain access to the buffered memory and, again, possible contention between the CPU and the CRT controller results. This contention is usually resolved in one of two ways: (1) the CPU is always given priority, or; (2) the CPU is allowed to access the buffered memory only during horizontal and vertical retrace times.

Approach 1 is the easiest to implement from a hardware point of view, but if the CPU always has priority the display may temporarily blink or "flicker" while the CPU accesses the display memory. This, of course, occurs because when the CPU accesses the display memory the CRT controller is not able to retrieve a character, so the display must be blanked during this time. Aesthetically, this "flickering" is not desirable, so approach 2 is often used.

The second approach eliminates the display flickering encountered in the previously mentioned technique, but additional hardware is required. Usually the vertical and horizontal blank signals are gated with the buffered memory select lines and this line is used to control the CPU's ready line. So, if the CPU wants to use the buffered memory, its ready line is asserted until horizontal or vertical retrace times. This, of course, will impact the CPU's overall throughput.

Both page buffered approaches require a significant amount of additional hardware and for the most part are not well suited for a minimum parts count type of terminal. This guides us to the line buffered approach. This approach eliminates the separate buffered memory for the display, but, at the same time, introduces a few new problems that must be solved.

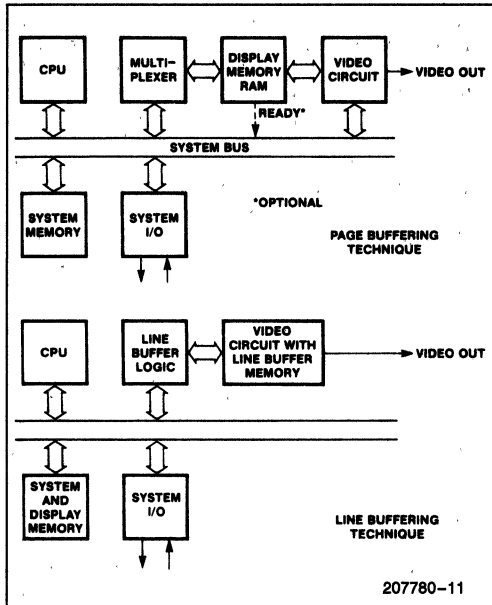


Figure 4-1. Line Buffering Technique

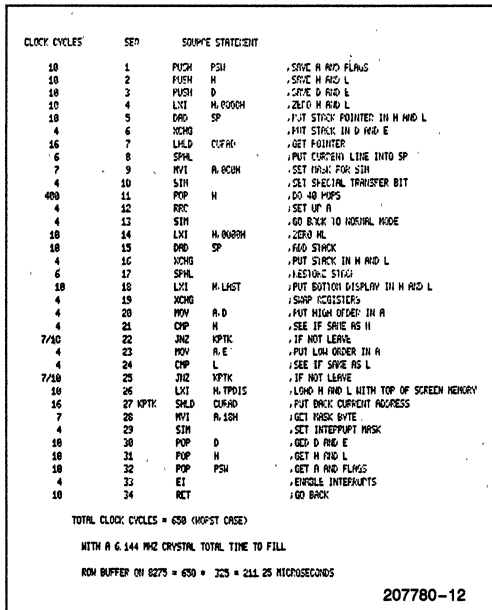


Figure 4-2. Routine To Load 8275's Row Buffers

In the line buffered approach both the CPU and the CRT controller share the same memory. Every time the CRT controller needs a new character or line of data, normal processing activity is halted and the CRT controller accesses memory and displays the data. Just how the CRT controller needs to acquire the display data greatly affects the performance of the overall system. Whether the CRT controller needs to gain access to the main memory to acquire a single character or a complete line of data depends on the presence or absence of a separate line or row buffer.

If no row buffer is present the CRT controller must go to the main memory to fetch every character. This of course, is not a very efficient approach because the processor will be forced to relinquish the bus 70% to 80% of the time. So much processor inactivity greatly affects the overall system performance. In fact terminals that use this approach are typically limited to around 1200 to 2400 baud on their serial communication channels. This low baud rate is in general not acceptable, hence this approach was not chosen.

If a separate row buffer is employed the CRT controller only has to access the memory once for each displayed character per line. This forces the processor to relinquish the bus only about 20% to 35% of the time and a full 4800 to 9600 baud can be achieved. Figure 4.1 illustrates these different techniques.

The 8275 CRT controller is ideal for implementing the row buffer approach because the row buffer is contained on the device itself. In fact, the 8275 contains two 80-byte row buffers. The presence of two row buffers allow one buffer to be filled while the other buffer is displaying the data. This dual row buffer approach enhances CPU performance even further.

4.2 Using the 8275 without DMA

Until now the process of filling the row buffer has only been alluded to. In reality, a DMA technique is usually used. This approach was demonstrated in AP-32 where an 8257 DMA controller was mated to an 8275 CRT controller. In order to minimize component count, this design eliminates the DMA controller and its associated circuitry while replacing them with a special interrupt-driven transfer.

The only real concern with using the 8275 in an interrupt-driven transfer mode is speed. Eighty characters must be loaded into the 8275 every 617 microseconds and the processor must also have time to perform all the other tasks that are required. To minimize the overhead associated with loading the characters into the 8275 a special technique was employed. This technique involves setting a special transfer bit and executing a string of POP instructions. The string of POP instruc-

tions is used to rapidly move the data from the memory into the 8275. Figure 4.2 shows the basic software structure.

In this design the 8085's SOD line was used as the special transfer bit. In order to perform the transfer properly this special bit must do two things: (1) turn processor reads into DACK plus WR for the 8275; and (2) mask processor fetch cycles from the 8275, so that a fetch cycle does not write into the 8275. Conventional logic could have been used to implement this special function, but in this design a small bipolar programmable read only memory was used. Figure 4.3 shows a basic version of the hardware.

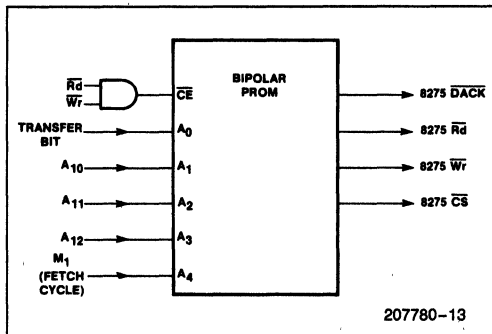


Figure 4-3. Simplified Version of Hardware Decoder

At first, it may seem strange that we are supplying a DACK when no DMA controller exists in the system. But the reader should be aware that all Intel peripheral devices that have DMA lines actually use DACK as a chip select for the data. So, when you want to write a command or read status you assert CS and WR or RD, but when you want to read or write data you assert DACK and RD or WR. The peripheral device doesn't "know" if a DMA controller is in the circuit or not. In passing, it should be mentioned that DACK and CS should not be asserted on the same device at the same time, since this combination yields an undefined result.

This POP technique actually compares quite favorably in terms of time to the DMA technique. One POP instruction transfers two bytes of data to the 8275 and takes 10 CPU clock cycles to execute, for a net transfer rate of one byte every five clock cycles. The DMA controller takes four clock cycles to transfer one byte but, some time is lost in synchronization. So the difference between the two techniques is one clock cycle per byte maximum. If we compare the overall speed of the 8085 to the speed of the 8080 used in AP-32, we find that at 3 MHz we can transfer one byte every 1.67 microseconds using the 8085 and POP technique vs. 2 microseconds per byte for the 2 MHz 8080 using DMA.

5.0 CIRCUIT DESCRIPTION

5.1 Scope of the Project

A fully functional, microprocessor-based CRT terminal was designed and constructed using the 8275 CRT controller and the 8085 as the controlling element. The terminal had many of the functions found in existing commercial low-cost terminals and more sophisticated features could easily be added with a modest amount of additional software. In order to minimize component count LSI devices were used whenever possible and software was used to replace hardware.

5.2 System Target Specifications

The design specifications for the CRT terminal were as follows:

Display Format

- 80 characters per display row
- 25 display rows

Character Format

- 5 × 7 dot matrix character contained within a 7 × 10 matrix
- First and seventh columns blanked
- Ninth line cursor position
- Blinking underline cursor

Special Characters Recognized

- Control characters
- Line feed
- Carriage Return
- Backspace
- Form feed

Escape Sequences Recognized

- ESC, A, Cursor up
- ESC, B, Cursor down
- ESC, C, Cursor right
- ESC, D, Cursor left
- ESC, E, Clear screen
- ESC, H, Home cursor
- ESC, J, Erase to the end of the screen
- ESC, K, Erase to the current line

Characters Displayed

- 96 ASCII alphanumeric characters
- Special control characters

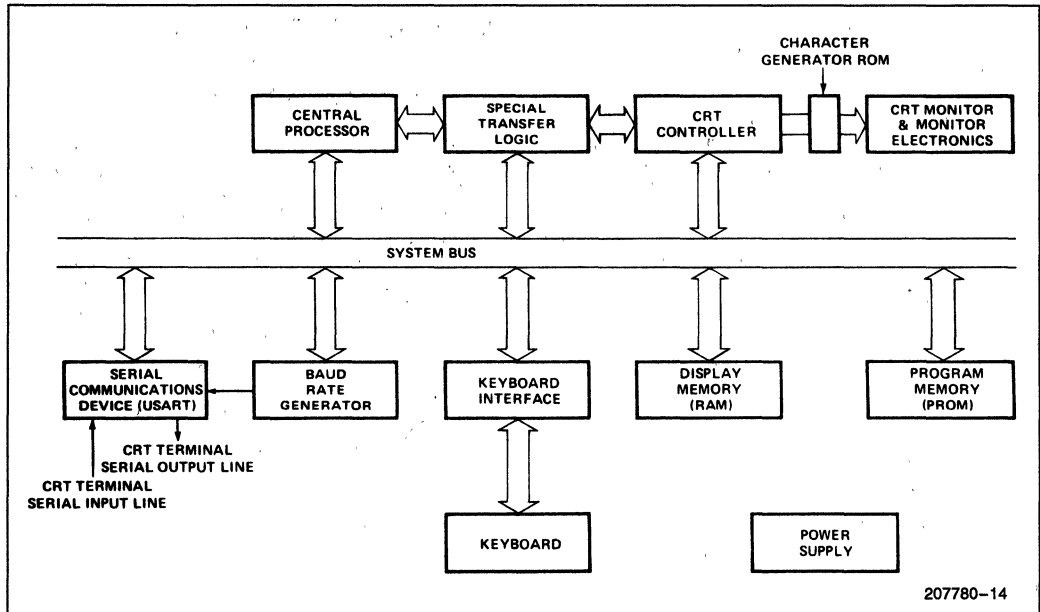


Figure 5-1. CRT Terminal Block Diagram

Characters Transmitted

- 96 ASCII alphanumeric characters
- ASCII control characters

Program Memory

- 2K bytes of 2716 EPROM

Display/Buffer/Stack Memory

- 2K bytes 2114 static memory (4 packages)

Data Rate

- 9600 BAUD using 3 MHz 8085

CRT Monitor

- Ball Bros TV-12, 12 MHz B.W.

Keyboard

- Any standard un-encoded ASCII keyboard

Screen Refresh Rate

- 60 Hz

5.3 Hardware Description

A block diagram of the CRT terminal is shown in Figure 5.1. The diagram shows only the essential system features. A detailed schematic of the CRT is contained in the Appendix. The terminal was constructed on a

simple 6" by 6" wire wrap board. Because of the minimum bus loading no buffering of any kind was needed (see Figure 5.2).

Worst case bus loading:		
Data Bus:	8275	20 pF
	8255A-5	20 pF
	8253-5	20 pF
	8253-5	20 pF
	8251A	20 pF
	2x2114	10 pF
	2716	12 pF
	8212	12 pF
		114 pF max
Only A ₈ -A ₁₅ are important since A ₀ -A ₇ are latched by the 8212		
Address Bus:	4x2114	20 pF
	2716	6 pF
		26 pF max
This loading assures that all components will be compatible with 3 MHz 8085 and that no wait states will be required		

Figure 5-2. Bus Loading

The "heart" of the CRT terminal is the 8085 micro-processor. The 8085 initializes all devices in the system, loads the CRT controller, scans the keyboard, assembles the characters to be transmitted, decodes the in-

coming characters and determines where the character is to be placed on the screen. Clearly, the processor is quite busy.

A standard list of LSI peripheral devices surround the 8085. The 8251A is used as the serial communication link, the 8255A-5 is used to scan the keyboard and read the system variables through a set of switches, and the 8253 is used as a baud rate generator and as a "horizontal pulse extender" for the 8275.

The 8275 is used as the CRT controller in the system, and a 2716 is used as the character generator. To handle the high speed portion of the terminal the 8275 is surrounded by a small handful of TTL. The program memory is contained in one 2716 EPROM and the data and screen memory use four 2114-type RAMs.

All devices in this system are memory mapped. A bipolar PROM is used to decode all of the addresses for the RAM, ROM, 8275, and the 8253. As mentioned earlier, the bipolar prom also turns READs into DACK's and WR's for the 8275. The 8255 and 8253 are decoded by a simple address line chip select method. The total package count for the system is 20, not including the serial line drivers. If this same terminal were designed using the MCS-85 family of integrated circuits, additional part savings could have been realized. The four 2114's could have been replaced by two 8185's and the 8255 and the 2716 program PROM could have been replaced by one 8755. Additionally, since both the 8185 and the 2716 have address latches no 8212 would be needed, so the total parts count could be reduced by three or four packages.

5.4 System Operation

The 8085 CPU initializes each peripheral to the appropriate mode of operation following system reset. After initialization, the 8085 continually polls the 8251A to see if a character has been sent to the terminal. When a character has been received, the 8085 decodes the character and takes appropriate action. While the 8085 is executing the above "foreground" programs, it is being interrupted once every 617 microseconds by the 8275. This "background" program is used to load the row buffers on the 8275. The 8085 is also interrupted once every frame time, or 16.67 ms, to read the keyboard and the status of the 8275.

As discussed earlier, a special POP technique was used to rapidly move the contents of the display RAM into the 8275's row buffers. The characters are then synchronously transferred to the character code outputs CC0-CC6, connected to the character generator address lines A3-A9 (Figure 5.3). Line count outputs LC0-LC2 from the 8275 are applied to the character generator address lines A0-A2. The 8275 displays character rows one line at a time. The line count outputs are used to determine which line of the character selected by A3-A8 will be displayed. Following the transfer of the first line to the dot timing logic, the line count is incremented and the second line of the character row is selected. This process continues until the last line of the row is transferred to the dot timing logic.

The dot timing logic latches the output of the character generator ROM into a parallel in, serial out synchronous shift register. This shift register is clocked at the dot clock rate (11.34 MHz) and its output constitutes the video input to the CRT.

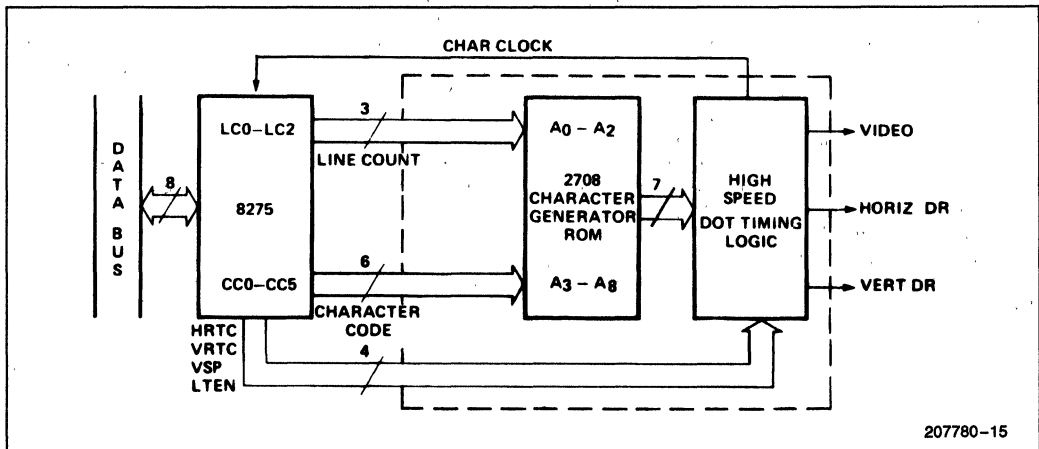


Figure 5-3. Character Generator/Dot Timing Logic Block Diagram

Table 5-1

Parameter	Range
Vertical Blanking Time (VRTC)	900 μ s nominal
Vertical Drive Pulsewidth	300 μ s \leq PW \leq 1.4 ms
Horizontal Blanking Time (HRTC)	11 μ s nominal
Horizontal Drive Pulsewidth	25 μ s \leq PW \leq 30 μ s
Horizontal Repetition Rate	15,750 \pm 500 pps

5.5 System Timing

Before any specific timing can be calculated it is necessary to determine what constraints the chosen CRT places on the overall timing. The requirements for the Ball Bros. TV-12 monitor are shown in Table 5.1. The data from Table 5.1, the 8275 specifications, and the system target specifications are all that is needed to calculate the system's timing.

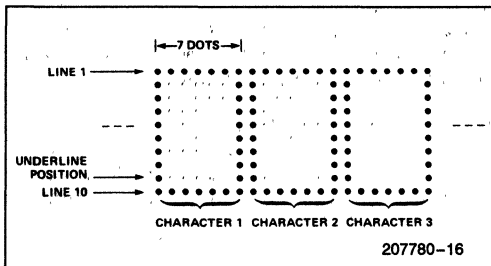


Figure 5-4. Row Format

First, let's select and "match" a few numbers. From our target specifications, we see that each character is displayed on a 7×10 field, and is formed by a 5×7 dot matrix (Figure 5.4). The 8275 allows the vertical retrace time to be only an integer multiple of the horizontal character line. This means that the total number of horizontal lines in a frame equals 10 times the number of character lines plus the vertical retrace time, which is programmed to be either 1, 2, 3, or 4 character lines. Twenty-five display lines require 250 horizontal

lines. So, if we wish to have a horizontal frequency in the neighborhood of 15,750 Hz we must choose either one or two character lines for vertical retrace. To allow for a little more margin at the top and bottom of the screen, two character lines were chosen for vertical retrace. This choice yields a net $250 + 20 = 270$ horizontal lines per frame. So, assuming a 60 Hz frame:

$$60 \text{ Hz} \times 270 = 16,200 \text{ Hz (horizontal frequency)}$$

This value falls within our target specification of 15,750 Hz with a 500 Hz variation and also assures timing compatibility with the Ball monitor since, 20 horizontal sync times yield a vertical retract time of:

$$61.7 \text{ ms} \times 20 \text{ horizontal sync times} = 1.2345 \text{ ms}$$

This number meets the nominal VRTC and vertical drive pulse width time for the Ball monitor. A horizontal frequency of 16,200 Hz implies a $1/16,200 = 61.73$ microsecond period.

It is now known that the terminal is using 250 horizontal lines to display data and 20 horizontal lines to allow for vertical retrace and that the horizontal frequency is 16,200 Hz. The next thing that needs to be determined is how much time must be allowed for horizontal retrace. Unfortunately, this number depends almost entirely on the monitor used. Usually, this number lies somewhere between 15 and 30 percent of the total horizontal line time, which in this case is $1/16,200 \text{ Hz}$ or 61.73 microseconds. Since in most designs a fixed number of characters can be displayed on a horizontal line, it is often useful to express retrace as a given number of character times. In this design, 80 characters can be displayed on a horizontal line and it was empirically found that allowing 20 horizontal character times for retrace gave the best results. So, in reality, there are 100 character times in every given horizontal line, 80 are used to display characters and 20 are used to allow for retrace. It should be noted that if too many character times are used for retrace, less time will be left to display the characters and the display will not "fill out" the screen. Conversely, if not enough character times are allowed for retrace, the display may "run off" the screen.

One hundred character times per complete horizontal line means that each character requires

$$61.73 \text{ ms}/100 \text{ character times} = 617.3 \text{ ns}$$

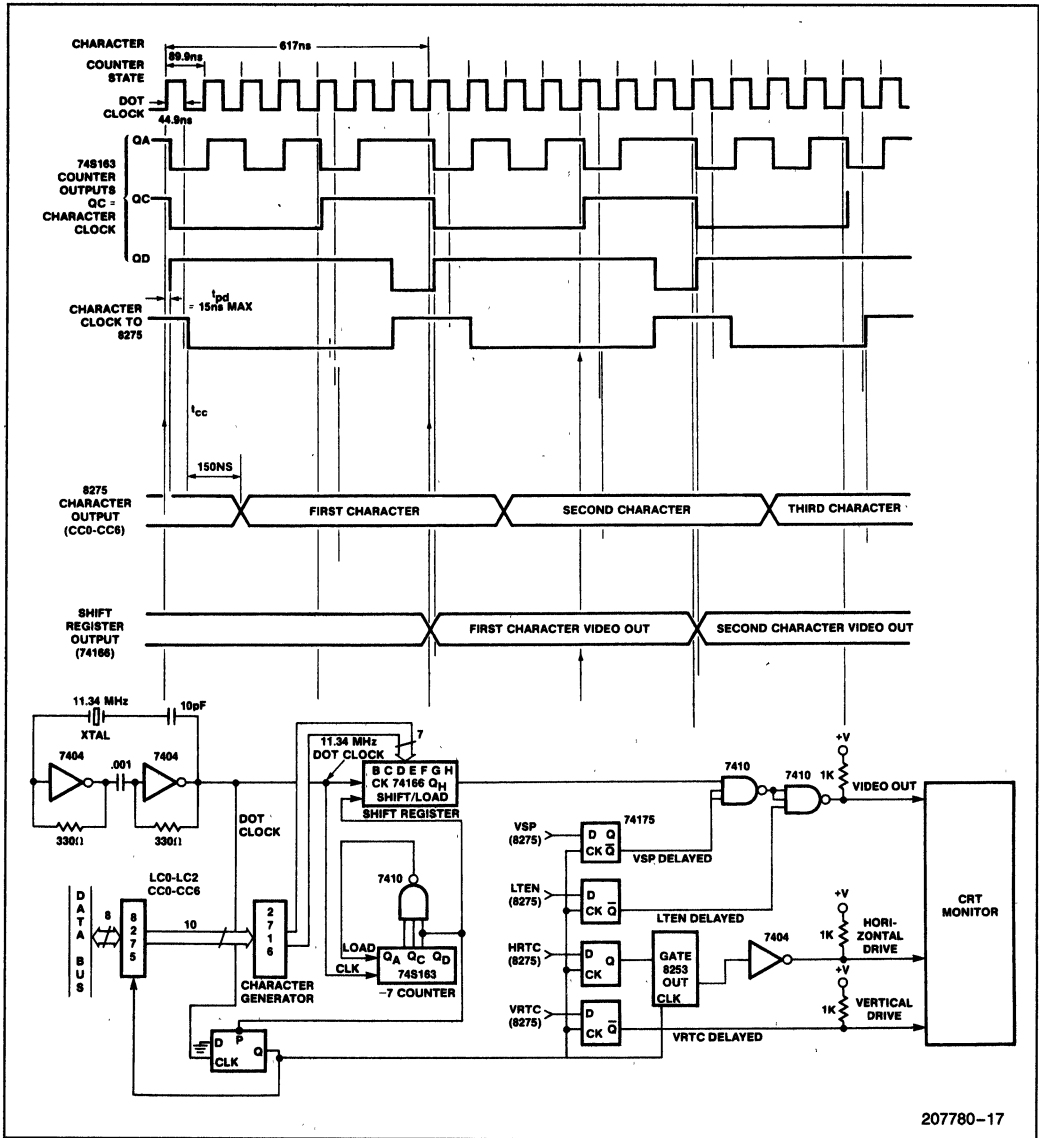


Figure 5-5. Dot Timing Logic

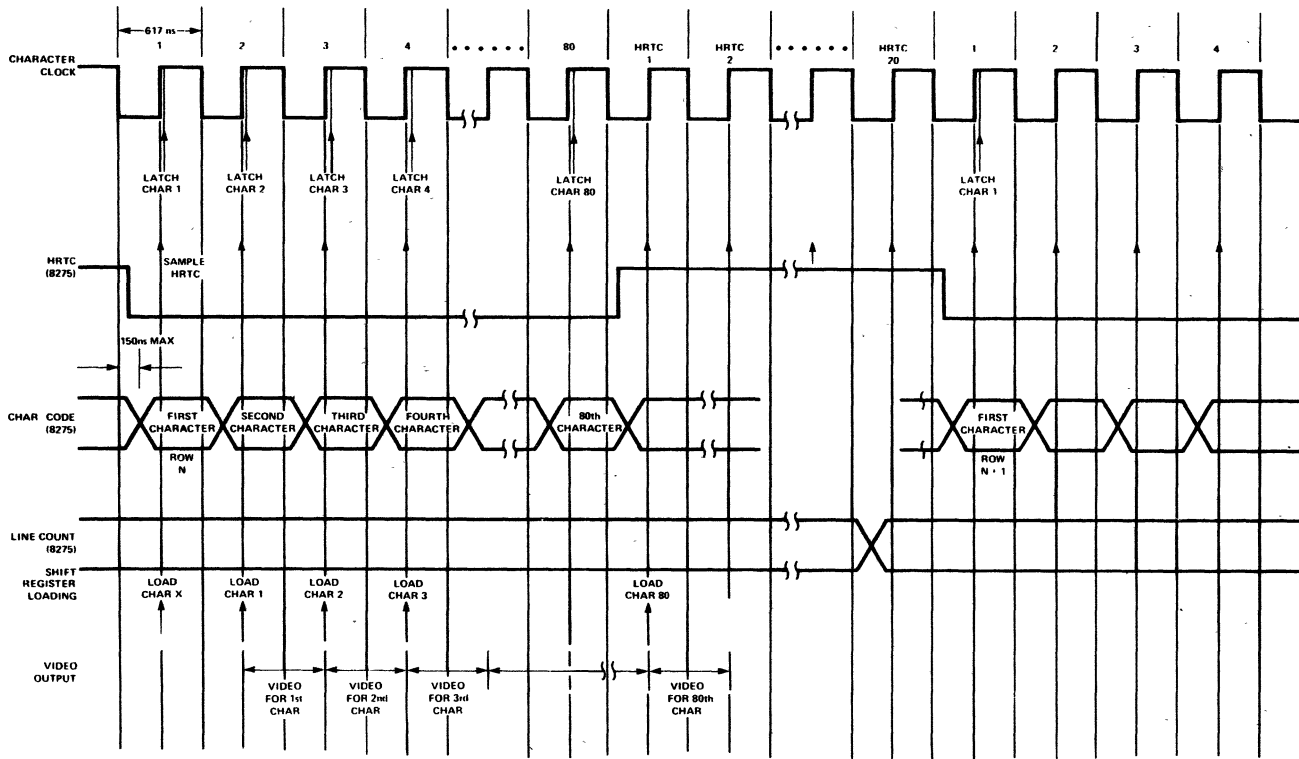


Figure 5-6. CRT System Timing

6-66

If we multiply the 20 horizontal retrace times by the 617.3 nanoseconds needed for each character, we find

$$617.3 \text{ ns} * 20 \text{ retrace times} = 12.345 \text{ ms}$$

This value falls short of the 25 to 30 microseconds required by the horizontal drive of the Ball monitor. To correct for this, an 8253 was programmed in the one-shot mode and was used to extend the horizontal drive pulsewidth.

Now that the 617.3 nanosecond character clock period is known, the dot clock is easy to calculate. Since each character is formed by placing 7 dots along the horizontal.

$$\text{DOT CLOCK PERIOD} = 617.3 \text{ ns} \\ (\text{CHARACTER CLK PERIOD}) / 7 \text{ DOTS}$$

$$\text{DOT CLOCK PERIOD} = 88.183 \text{ nanoseconds}$$

$$\text{DOT CLOCK FREQUENCY} = 1/\text{PERIOD} = 11.34 \text{ MHz}$$

Figures 5.5 and 5.6 illustrate the basic dot timing and the CRT system timing, respectively.

6.0 SYSTEM SOFTWARE

6.1 Software Overview

As mentioned earlier the software is structured on a "foreground-background" basis. Two interrupt-driven routines, FRAME and POPDAT (Figure 6.1) request service every 16.67 milliseconds and 617 microseconds respectively, frame is used to check the baud rate switches, update the system pointers and decode and assemble the keyboard characters. POPDAT is used to move data from the memory into the 8275's row buffer rapidly.

The foreground routine first examines the line-local switch to see whether to accept data from the USART or the keyboard. If the terminal is in the local mode, action will be taken on any data that is entered through the keyboard and the USART will be ignored on both output and input. If the terminal is in the line mode data entered through the keyboard will be transmitted by the USART and action will be taken on any data read out of the USART.

When data has been entered in the terminal the software first determines if the character received was an escape, line feed, form feed, carriage return, back space, or simply a printable character. If an escape was received the terminal assumes the next received character will be a recognizable escape sequence character. If it isn't no operation is performed.

After the character is decoded, the processor jumps to the routine to perform the required task. Figure 6.2 is a flow chart of the basic software operations; the program is listed in Appendix 6.8.

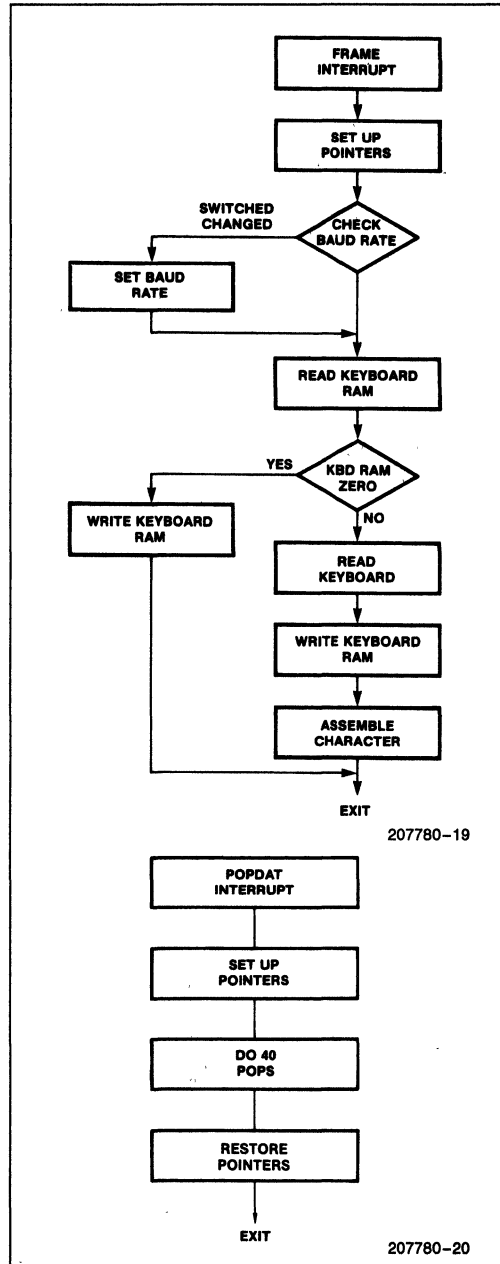


Figure 6-1. Frame and POPDAT Interrupt Routines

6.2 System Memory Organization

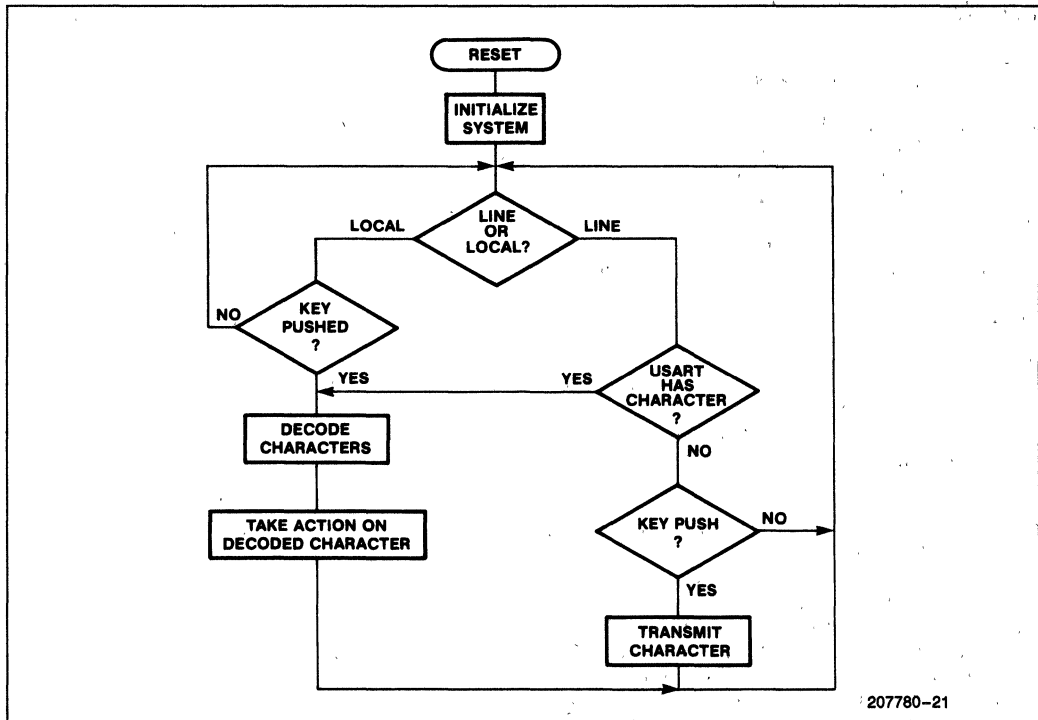
The display memory organization is shown in Figure 6.3. The display begins at location 0800H in memory and ends at location 0FCFH. The 48 bytes of RAM from location 0FDOH to 0FFFFH are used as system stack and temporary system storage. 2K bytes of PROM located at 0000H through 07FFH contain the systems program.

6.3 Memory Pointers and Scrolling

To calculate the location of a character on the screen, three variables must be defined. Two of these variables are the X and Y position of the cursor (CURSX, CURSY). In addition, the memory address defining the top line of the display must be known, since scrolling on the 8275 is accomplished simply by changing the pointer that loads the 8275's row buffers from memory. So, if it is desired to scroll the display up or down all that must be changed is one 16-bit memory pointer. This pointer is entered into the system by the variable TOPAD (TOP Address) and always defines the top line of the display. Figure 6.4 details screen operation during scrolling.

	1st Column	2nd Column	80th Column
ROW 1	0800H	0801H	084FH
ROW 2	0850H	0851H	089FH
ROW 3	08A0H	08A1H	08EFH
ROW 4	08F0H	08F1H	083FH
ROW 5	0940H	0941H	098FH
ROW 6	0990H	0991H	090FH
ROW 7	09E0H	09E1H	0A2FH
ROW 8	0A30H	0A31H	0A7FH
ROW 9	0A80H	0A81H	0ACFH
ROW 10	0AD0H	0AD1H	0B1FH
ROW 11	0B20H	0B21H	0B6FH
ROW 12	0B70H	0B71H	0BBFH
ROW 13	0BC0H	0BC1H	0C0FH
ROW 14	0C10H	0C11H	0C5FH
ROW 15	0C60H	0C61H	0CAFH
ROW 16	0CB0H	0CB1H	0CFFH
ROW 17	0D00H	0D01H	0D4FH
ROW 18	0D50H	0D51H	0D9FH
ROW 19	0DA0H	0DA1H	0DEFH
ROW 20	0DF0H	0DF1H	0E3FH
ROW 21	0E40H	0E41H	0E8FH
ROW 22	0E90H	0E91H	0EDFH
ROW 23	0EE0H	0EE1H	0F2FH
ROW 24	0F30H	0F31H	0F7FH
ROW 25	0F80H	0F81H	0FCFH

Figure 6-3. Screen Display After Initialization



207780-21

Figure 6-2. Basic Terminal Software

Subroutines `CALCU` (Calculate) and `ADX` (ADD X axis) use these three variables to calculate an absolute memory address. The subroutine `CALCU` is used whenever a location in the screen memory must be altered.

6.4 Software Timing

One important question that must be asked about the terminal software is, "How fast does it run". This is important because if the terminal is running at 9600 baud, it must be able to handle each received character in 1.04 milliseconds. Figure 6.5 is a flowchart of the subroutine execution times. It should be pointed out that all of the times listed are "worst case" execution times. This means that all routines assume they must do the maximum amount of data manipulation. For

instance, the `PUT` routine assumes that the character is being placed in the last column and that a line feed must follow the placing of the character on the screen.

How fast do the routines need to execute in order to assure operation at 9600 baud? Since `POPDAT` interrupts occur every 617 microseconds, it is possible to receive two complete interrupt requests in every character time (1042 microseconds) at 9600 baud. Each `POPDAT` interrupt executes in 211 microseconds maximum. This means that each routine must execute in:

$$1042 - 2 * 211 = 620 \text{ microseconds}$$

By adding up the times for any loop, it is clear that all routines meet this speed requirement, with the exception of `ESC J`. This means that if the terminal is operating at 9600 baud, at least one character time must be inserted after an `ESC J` sequence.

ROW 1	0800H	0801H084FH
ROW 2	0850H	0851H089FH
ROW 3	08A0H	08A1H08EFH
ROW 4	08F0H	08F1H093FH
ROW 5	0940H	0941H098FH
ROW 6	0990H	0991H090FH
ROW 7	09E0H	09E1H0A2FH
ROW 8	0A30H	0A31H0A7FH
ROW 9	0A80H	0A81H0ACFH
ROW 10	0AD0H	0AD1H0B1FH
ROW 11	0B20H	0B21H0B6FH
ROW 12	0B70H	0B71H0BBFH
ROW 13	0BC0H	0BC1H0C0FH
ROW 14	0C10H	0C11H0C5FH
ROW 15	0C60H	0C61H0CAFH
ROW 16	0CB0H	0CB1H0CFFH
ROW 17	0D00H	0D01H0D4FH
ROW 18	0D50H	0D51H0D9FH
ROW 19	0DA0H	0DA1H0DEFH
ROW 20	0DF0H	0DF1H0E3FH
ROW 21	0E40H	0E41H0E8FH
ROW 22	0E90H	0E91H0EDFH
ROW 23	0EE0H	0EE1H0F2FH
ROW 24	0F30H	0F31H0F7FH
ROW 25	0F80H	0F81H0FCFH

After Initialization

ROW 2	0850H	0851H089FH
ROW 3	08A0H	08A1H08EFH
ROW 4	08F0H	08F1H093FH
ROW 5	0940H	0941H098FH
ROW 6	0990H	0991H090FH
ROW 7	09E0H	09E1H0A2FH
ROW 8	0A30H	0A31H0A7FH
ROW 9	0A80H	0A81H0ACFH
ROW 10	0AD0H	0AD1H0B1FH
ROW 11	0B20H	0B21H0B6FH
ROW 12	0B70H	0B71H0BBFH
ROW 13	0BC0H	0BC1H0C0FH
ROW 14	0C10H	0C11H0C5FH
ROW 15	0C60H	0C61H0CAFH
ROW 16	0CB0H	0CB1H0CFFH
ROW 17	0D00H	0D01H0D4FH
ROW 18	0D50H	0D51H0D9FH
ROW 19	0DA0H	0DA1H0DEFH
ROW 20	0DF0H	0DF1H0E3FH
ROW 21	0E40H	0E41H0E8FH
ROW 22	0E90H	0E91H0EDFH
ROW 23	0EE0H	0EE1H0F2FH
ROW 24	0F30H	0F31H0F7FH
ROW 25	0F80H	0F81H0FCFH
ROW 1	0800H	0801H084FH

After 1 Scroll

ROW 3	08A0H	08A1H08EFH
ROW 4	08F0H	08F1H093FH
ROW 5	0940H	0941H098FH
ROW 6	0990H	0991H090FH
ROW 7	09E0H	09E1H0A2FH
ROW 8	0A30H	0A31H0A7FH
ROW 9	0A80H	0A81H0ACFH
ROW 10	0AD0H	0AD1H0B1FH
ROW 11	0B20H	0B21H0B6FH
ROW 12	0B70H	0B71H0BBFH
ROW 13	0BC0H	0BC1H0C0FH
ROW 14	0C10H	0C11H0C5FH
ROW 15	0C60H	0C61H0CAFH
ROW 16	0CB0H	0CB1H0CFFH
ROW 17	0D00H	0D01H0D4FH
ROW 18	0D50H	0D51H0D9FH
ROW 19	0DA0H	0DA1H0DEFH
ROW 20	0DF0H	0DF1H0E3FH
ROW 21	0E40H	0E41H0E8FH
ROW 22	0E90H	0E91H0EDFH
ROW 23	0EE0H	0EE1H0F2FH
ROW 24	0F30H	0F31H0F7FH
ROW 25	0F80H	0F81H0FCFH
ROW 1	0800H	0801H084FH
ROW 2	0850H	0851H089FH

After 2 Scrolls

ROW 4	08F0H	08F1H093FH
ROW 5	0940H	0941H098FH
ROW 6	0990H	0991H090FH
ROW 7	09E0H	09E1H0A2FH
ROW 8	0A30H	0A31H0A7FH
ROW 9	0A80H	0A81H0ACFH
ROW 10	0AD0H	0AD1H0B1FH
ROW 11	0B20H	0B21H0B6FH
ROW 12	0B70H	0B71H0BBFH
ROW 13	0BC0H	0BC1H0C0FH
ROW 14	0C10H	0C11H0C5FH
ROW 15	0C60H	0C61H0CAFH
ROW 16	0CB0H	0CB1H0CFFH
ROW 17	0D00H	0D01H0D4FH
ROW 18	0D50H	0D51H0D9FH
ROW 19	0DA0H	0DA1H0DEFH
ROW 20	0DF0H	0DF1H0E3FH
ROW 21	0E40H	0E41H0E8FH
ROW 22	0E90H	0E91H0EDFH
ROW 23	0EE0H	0EE1H0F2FH
ROW 24	0F30H	0F31H0F7FH
ROW 25	0F80H	0F81H0FCFH
ROW 1	0800H	0801H084FH
ROW 2	0850H	0851H089FH
ROW 3	08A0H	08A1H08EFH

After 3 Scrolls

Figure 6-4. Screen Memory During Scrolling

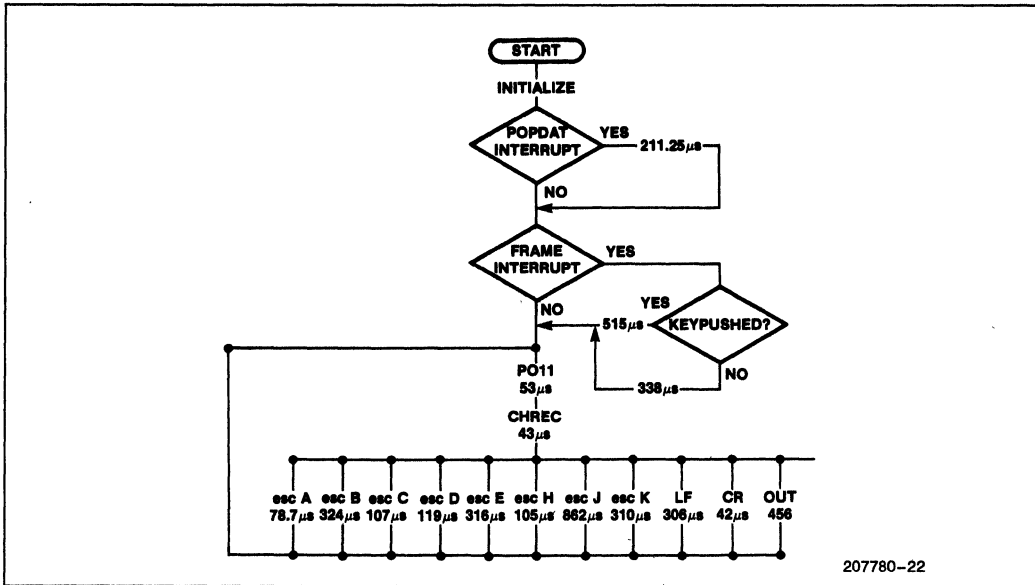
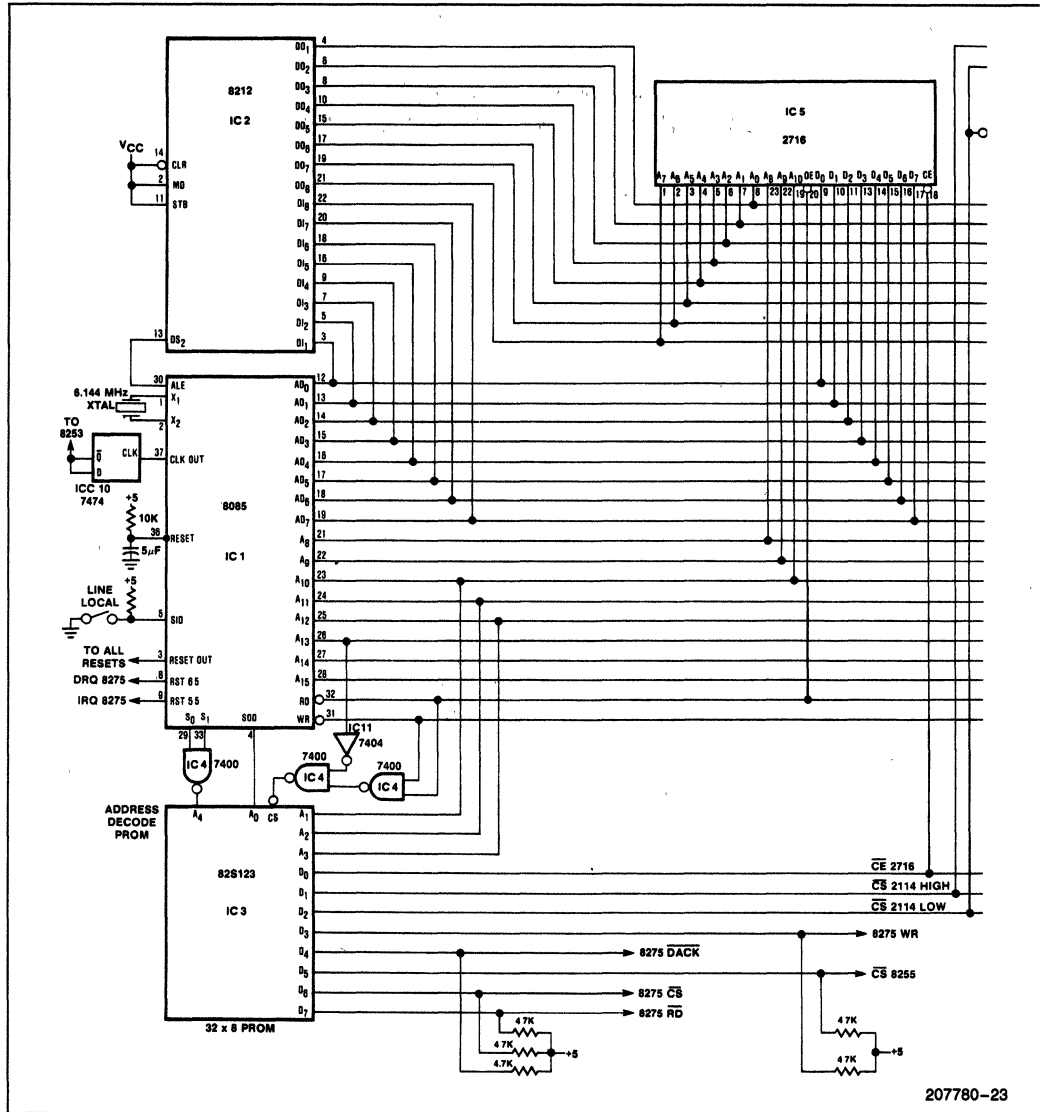
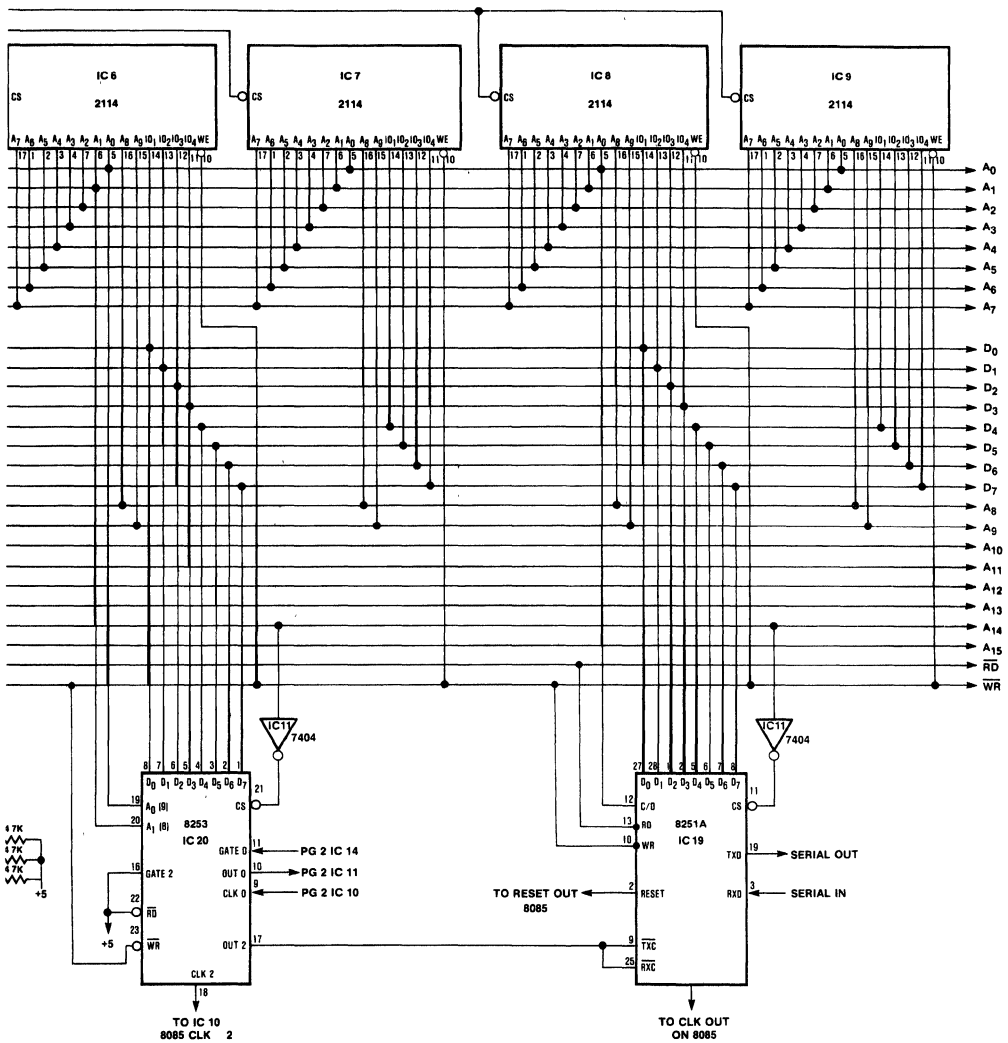


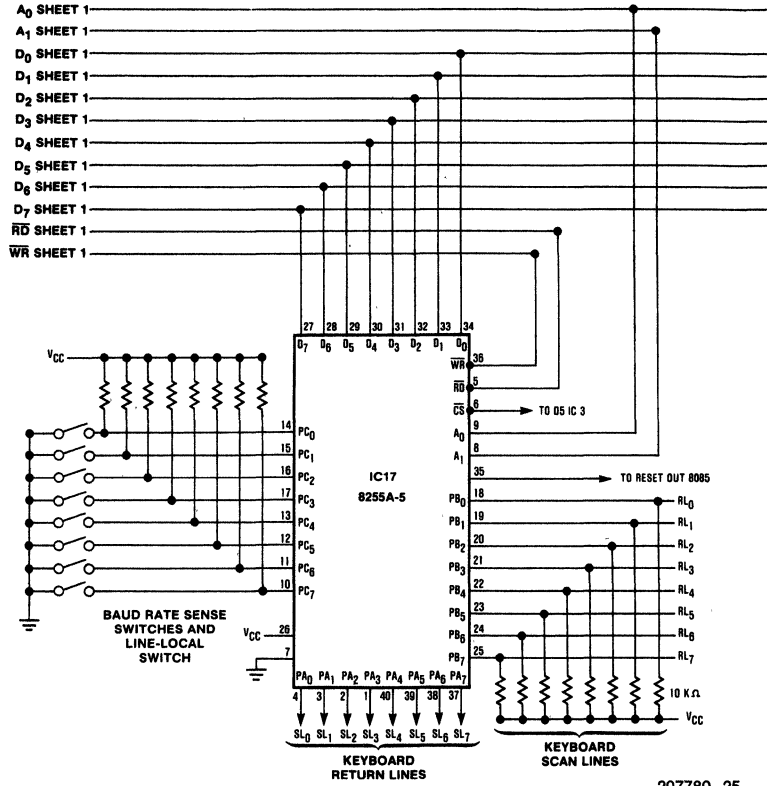
Figure 6-5. Timing Flowchart

APPENDIX 7.1 CRT TERMINAL SCHEMATICS





207780-24



207780-25

APPENDIX 7.2 KEYBOARD INTERFACE

The keyboard used in this design was a simple unencoded ASCII keyboard. In order to keep the cost to a minimum a simple scan matrix technique was implemented by using two ports of an 8255 parallel I/O device.

When the system is initialized the contents of the eight keyboard RAM locations are set to zero. Once every frame, which is 16.67 milliseconds the contents of the keyboard RAM is read and then rewritten with the contents of the current switch matrix. If a non-zero value of one of the keyboard RAM locations is found to be the same as the corresponding current switch matrix, a valid key push is registered and action is taken. By operating the keyboard scan in this manner an automatic debounce time of 16.67 milliseconds is provided.

Figure 7.2A shows the actual physical layout of the keyboard and Figure 7.2B shows how the individual keys were encoded. On Figure 7.2B the scan lines are the numbers on the bottom of each key position and the return lines are the numbers at the top of each key position. The shift, control, and caps lock key were brought in through separate lines of port C of the 8255. Figure 7.3 shows the basic keyboard matrix.

In order to guarantee that two scan lines could not be shorted together if two or more keys are pushed simultaneously, isolation diodes could be added as shown in Figure 7.4.

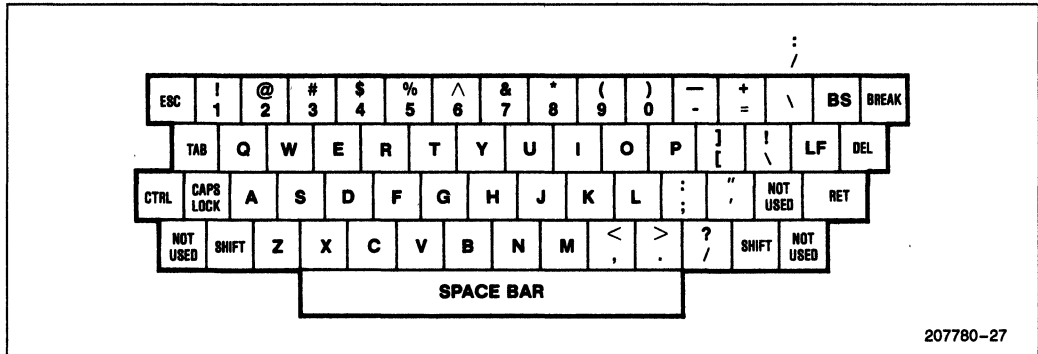


Figure 7-2A. Keyboard Layout

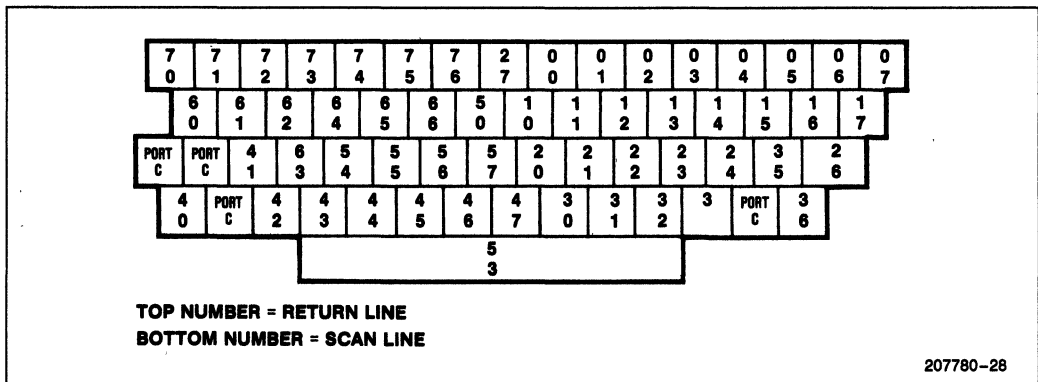


Figure 7-2B. Keyboard Encoding

APPENDIX 7.3 ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY

Bit	Control Characters		Displayable Character						Escape Sequence					
	000	001	010	011	100	101	110	111	010	011	100	101	110	111
0000	NUL @	DLE P	SP	φ	@	P		P						
0001	SOH A	DC1 Q		1	A	Q	A	Q			↑ A			
0010	STX B	DC2 R	"	2	B	R	B	R			↓ B			
0011	ETX C	DC3 S	#	3	C	S	C	S			→ C			
0100	EOT D	DC4 T	\$	4	D	T	D	T			← D			
0101	ENQ E	NAK U	%	5	E	U	E	U			CLR E			
0110	ACK F	SYN V	&	6	F	V	F	V						
0111	BEL G	ETB W	'	7	G	W	G	W						
1000	BS H	CAN X	(8	H	X	H	X			HOME H			
1001	HT I	EM Y)	9	I	Y	I	Y						
1010	LF J	SUB Z	*	:	J	Z	J	Z			EOS I			
1011	VT K	ESC I	+	;	K	[K				EL J			
1100	FF L	FS /	,	<	L	\	L							
1101	CR M	GS	-	=	M]	M							
1110	SO N	RS ^	.	>	N	^	N							
1111	S1 O	US -	/	?	O	-	O							

NOTE:

Shaded blocks = functions terminal will react to. Others can be generated but are ignored upon receipt.

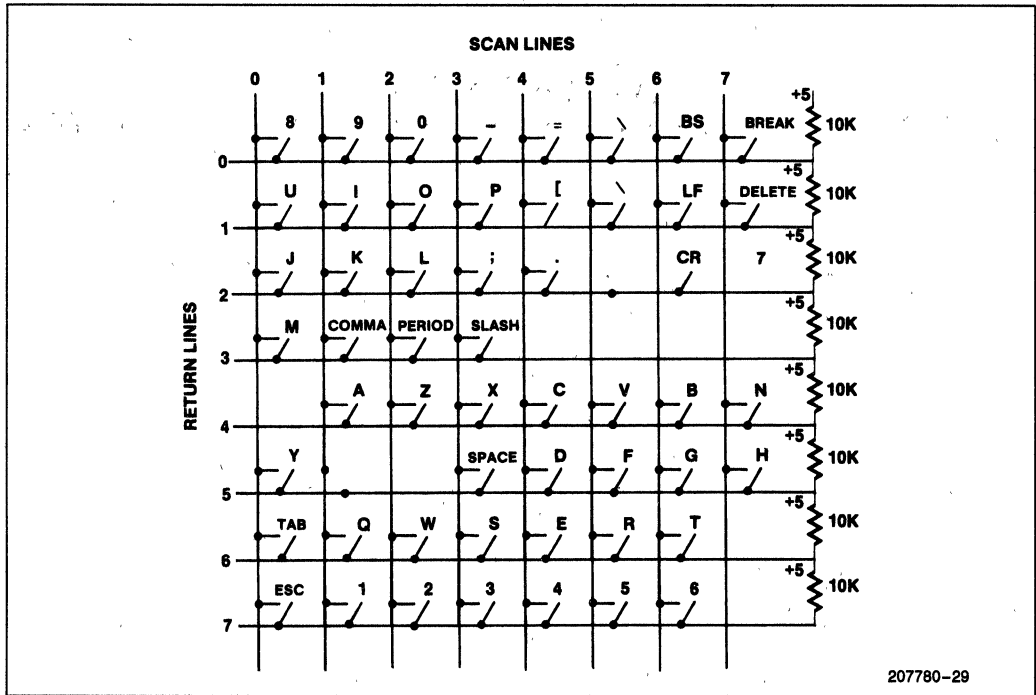


Figure 7-3. Keyboard Matrix

APPENDIX 7.4 PROM DECODING

As stated earlier, all of the logic necessary to convert the 8275 into a non-DMA type of device was performed by a single small bipolar prom. Besides turning certain processor READS into DACKS and WRITES for the 8275, this 32 by 8 prom decoded addresses for the system ram, rom, as well as for the 8255 parallel I/O port.

Any bipolar prom that has a by eight configuration could function in this application. This particular device was chosen simply because it is the only "by eight" prom available in a 16 pin package. The connection of the prom is shown in detail in Figure 7.5 and its truth table is shown in Figure 7.6. Note that when a fetch cycle (M1) is not being performed, the state of the SOD line is the only thing that determines if memory reads will be written into the 8275's row buffers. This is done by pulling both DACK and WRITE low on the 8275.

Also note that all of the outputs of the bipolar prom MUST BE PULLED HIGH by a resistor. This prevents any unwanted assertions when the prom is disabled.

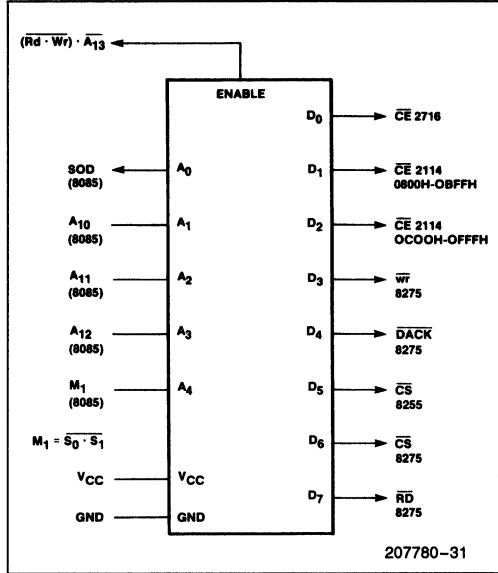


Figure 7-5. Bipolar Prom (825123) Connection

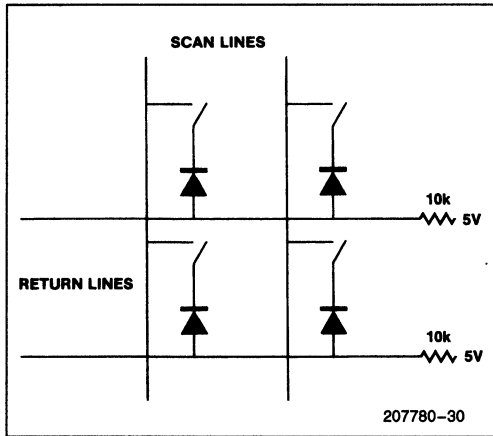


Figure 7-4. Isolating Scan Lines with Diodes

M1	A12	A11	A10	Sod	8275 Rd	8275 CS	8255 CS	8275 DACK	8275 WR	2114 H	2114 L	2716
A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	0
0	0	0	1	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0
0	0	1	0	1	1	1	1	1	1	1	1	0
0	0	1	1	0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	0
0	1	0	0	0	1	0	0	1	1	0	1	1
0	1	0	0	1	1	0	0	1	1	0	1	1
0	1	0	1	0	0	0	0	1	1	1	1	1
0	1	1	0	0	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	0	1	1	1	1	1
0	1	1	1	0	1	1	1	0	1	1	1	1
0	1	1	1	1	1	1	1	0	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	0	1	1	0
1	0	0	1	0	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	1	0	1	1	1	0
1	0	1	0	0	1	1	1	1	0	1	1	0
1	0	1	0	1	1	1	1	0	1	1	1	0
1	0	1	1	0	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	1	1	0	1	1
1	1	0	0	0	1	1	0	1	1	0	1	1
1	1	0	0	1	1	0	0	1	1	1	1	1
1	1	0	1	0	0	0	1	1	1	1	1	1
1	1	1	0	0	1	1	1	0	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	0	1	1	1	1	1

Figure 7-6. Truth Table Bipolar Prom

APPENDIX 7.5 CHARACTER GENERATOR

As previously mentioned, the character generator used in this terminal is a 2716 or 2758 EPROM. A 1K by 8 device is sufficient since a 128 character 5 by 7 dot matrix only required 8K of memory. Any "standard" or custom character generator could have been used.

The three low-order line count outputs (LC0-LC2) from the 8275 are connected to the three low-order address lines of the character generator and the seven character generator outputs (CC0-CC6) are connected to the A3-A9 of the character generator. The output from the character generator is loaded into a shift register and the serial output from the shift register is the video output of the terminal.

Now, let's assume that the letter "E" is to be displayed. The ASCII code for "E" is 45H. So, 45H is presented to address lines A2-A9 of the character generator. The scan lines will now count each line from zero to seven to "form" the character as shown in Figure 7.7. This same procedure is used to form all 128 possible characters.

It should be obvious that "custom" character fonts could be made just by changing the bit patterns in the character generator PROM. For reference, Appendix 7.6 contains a HEX dump of the character generator used in this terminal.

45H = 01000101
 Address to Prom = 01000101 SL2 SL1 SLO
 = 228H - 22FH
 Depending on state of Scan lines.

Character generator output									
ROM Address	Rom Hex Output	Bit Output*							
		0	1	2	3	4	5	6	7
228H	3E	0	1	2	3	4	5	6	7
229H	02	X	X	X	X	X			
22AH	02	X							
22BH	0E	X							
22CH	02	X	X	X					
22DH	02	X							
22EH	3E	X							
22FH	00	X	X	X	X	X			

Bits 0, 6, and 7 are not used.

*NOTE:

Bit output is backward from convention.

Figure 7.7. Character Generation

APPENDIX 7.7 COMPOSITE VIDEO

In this design, it was assumed that the monitor required a separate horizontal drive, vertical drive, and video input. However, many monitors require a composite video signal. The schematic shown in Figure 7.8 illustrates how to generate a composite video signal from the output of the 8275.

The dual one-shots are used to provide a small delay and the proper horizontal and vertical pulse to the composite video monitor. The delay introduced in the vertical and horizontal timing is used to "center" the display. VR1 and VR2 control the amount of delay. IC3 is used to mix the vertical and horizontal retrace and Q1 along with the R1, R2, and R3 mix the video and the retrace signal and provide the proper DC levels.

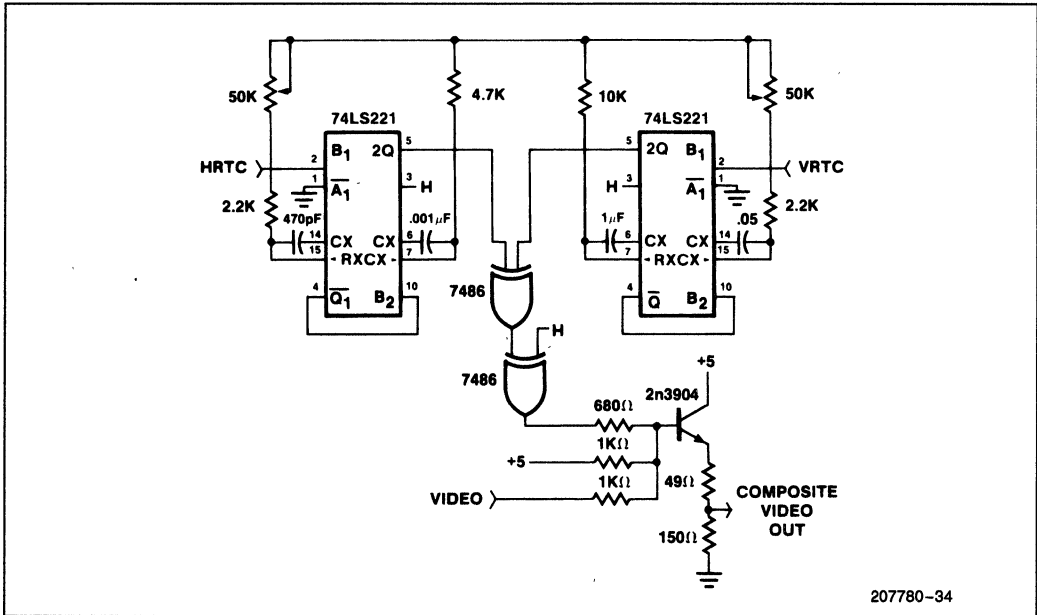


Figure 7-8. Composite Video

207780-34

APPENDIX 7.8 SOFTWARE LISTINGS

ISIS-II 8080/8085 MACRO ASSEMBLER, X108

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	SMOD85 MACROFILE
		2	;NO DMA 8275 SOFTWARE ALL I/O IS MEMORY MAPPED
		3	;SYSTEM ROM 0000H TO 07FFH
		4	;SYSTEM RAM 0800H TO 0FFFH
		5	;8275 WRITE 1000H TO 13FFH
		6	;8275 READ 1400H TO 17FFH
		7	;8255 READ/WRITE 1800H TO 1FFF
		8	;8251 ENABLED BY A14
		9	;8251 ENABLED BY A15
1800		10	PORTA EQU 1800H ;8255 PORT A ADDRESS
1801		11	PORTB EQU 1801H ;8255 PORT B ADDRESS
1802		12	PORTC EQU 1802H ;8255 PORT C ADDRESS
1803		13	CNWD55 EQU 1803H ;8255 CONTROL PORT ADDRESS
A001		14	USTF EQU 0A001H ;8251 FLAGS
A000		15	USTD EQU 0A000H ;8251 DATA
6000		16	CNT0 EQU 6000H ;8253 COUNTER 0
6001		17	CNT1 EQU 6001H ;8253 COUNTER 1
6002		18	CNT2 EQU 6002H ;8253 COUNTER 2
6003		19	CNTM EQU 6003H ;8253 MODE WORD
1001		20	CRTS EQU 1001H ;8275 CONTROL ADDRESS
1000		21	CRTM EQU 1000H ;8275 MODE ADDRESS
1401		22	INT75 EQU 1401H ;8275 INTERRUPT CLEAR
0000		23	TPDIS EQU 0000H ;TOP OF DISPLAY RAM
0FB0		24	BTDIS EQU 0FB0H ;BOTTOM OF DISPLAY RAM
0FD0		25	LAST EQU 0FD0H ;FIRST BYTE AFTER DISPLAY
0018		26	CURBOT EQU 18H ;BOTTOM Y CURSOR
0050		27	LNTH EQU 0050H ;LENGTH OF ONE LINE
0FE0		28	STPTR EQU 0FE0H ;LOCATION OF STACK POINTER
		29	;
		30	;START PROGRAM
		31	;ALL VARIABLES ARE INITIALIZED BEFORE ANYTHING ELSE
		32	;
0000	F3	33	DI ;DISABLE INTERRUPTS
0001	31E00F	34	LXI SP,STPTR ;LOAD STACK POINTER
0004	210000	35	LXI H,TPDIS ;LOAD H/L WITH TOP OF DISPLAY
0007	22E30F	36	SHLD TOPAD ;SET TOP = TOP OF DISPLAY
000A	22E80F	37	SHLD CURAD ;STORE THE CURRENT ADDRESS
000D	3E00	38	MVI A,00H ;ZERO A
000F	32E10F	39	STA CURSY ;ZERO CURSOR Y POINTER
0012	32E20F	40	STA CURSX ;ZERO CURSOR X POINTER
0015	32E80F	41	STA KBCHR ;ZERO KBD CHARACTER
0018	32E70F	42	STA USCHR ;ZERO USART CHAR BUFFER
001B	32EA0F	43	STA KEYDWN ;ZERO KEY DOWN

207780-35

```

001F 32ED0F 44 STA KEYOK ;ZERO KEYOK
0021 32E80F 45 STA ESCP ;ZERO ESCAPE
0024 C39800 46 JMP LFKGD ;JUMP AND SET EVERYTHING UP
47 ;
48 ;THIS JUMP VECTOR IS LOCATED AT THE RST 5.5 LOCATION
49 OF THE 8085. IT IS USED TO READ THE 8275 STATUS AND
50 ;READ THE KEYBOARD. THIS ROUTINE IS EXECUTED ONCE EVERY
51 ;16.667 MILLISECONDE.
52 ;
002C C35701 53 ORG 002CH
002C 54 JMP FRAME
55 ;
56 ;THIS ROUTINE IS LOCATED AT THE RST 6.5 LOCATION OF THE
57 ;8085 AND IS USED TO LOAD THE DATA TO BE DISPLAYED INTO
58 ;THE 8275. THIS ROUTINE IS EXECUTED ONCE EVERY 617 MICROSECONDS.
59 ;
0034 60 ORG 34H
0034 61 PUSH PSW ;SAVE A AND FLAGS
0035 62 PUSH H ;SAVE H AND L
0036 63 PUSH D ;SAVE D AND E
0037 210000 64 LXI H,0000H ;ZERO H AND L
0038 65 DAD SP ;PUT STACK POINTER IN H AND L
003B 66 XCHG ;PUT STACK IN D AND E
003C 2A800F 67 LHL CURAD ;GET POINTER
003E 68 SPHL ;PUT CURRENT LINE INTO SP
0040 69 MVI A,0C0H ;SET MASK FOR SIM
0042 70 SIM
71 REPT (LNTH/2)
72 POP H
73 ENDM
0043 74+ POP H
0044 75+ POP H
0045 76+ POP H
0046 77+ POP H
0047 78+ POP H
0048 79+ POP H
0049 80+ POP H
004A 81+ POP H
004B 82+ POP H
004C 83+ POP H
004D 84+ POP H
004E 85+ POP H
004F 86+ POP H
0050 87+ POP H
0051 88+ POP H
0052 89+ POP H
0053 90+ POP H
0054 91+ POP H
0055 92+ POP H
0056 93+ POP H
0057 94+ POP H
0058 95+ POP H
0059 96+ POP H
005A 97+ POP H
005B 98+ POP H
005C 99+ POP H
005D 100+ POP H
005E 101+ POP H
005F 102+ POP H
0060 103+ POP H
0061 104+ POP H
0062 105+ POP H
0063 106+ POP H
0064 107+ POP H
0065 108+ POP H
0066 109+ POP H
0067 110+ POP H
0068 111+ POP H
0069 112+ POP H
006A 113+ POP H
006B 114 RRC
006C 30 115 SIM ;SET UP A
006D 210000 116 LXI H,0000H ;GO BACK TO NORMAL MODE
0070 39 117 DAD SP ;ZERO HL
0071 6B 118 XCHG ;ADD STACK
0072 F9 119 SPHL ;PUT STACK IN H AND L
0073 21D00F 120 LXI H,LAST ;RESTORE STACK
0076 6B 121 XCHG ;PUT BOTTOM DISPLAY IN H AND L
0077 7A 122 MOV A,D ;SWAP REGISTERS
0078 4C 123 CMP H ;PUT HIGH ORDER IN A
0079 7B 124 JNZ KPTK ;SEE IF SAME AS H
007C 28400 125 MOV A,E ;IF NOT LEAVE
007D 8D 126 CMP L ;PUT LOW ORDER IN A
007E 28400 127 JNZ KPTK ;SEE IF SAME AS L
0081 210008 128 LXI H,TPDIS ;IF NOT LEAVE
0084 22580F 129 KPTK: SHLD CURAD ;LOAD H AND L WITH TOP OF SCREEN MEMORY
0087 3E18 130 MVI A,18H ;PUT BACK CURRENT ADDRESS
0089 30 131 SIM ;SET MASK
;OUTPUT MASK

```



```

219 ;8275 INITIALIZATION
220
00F9 210110 IN75: LXI H,CRTS
00FC 3500 MVI M,00H ;RESET AND STOP DISPLAY
00FE 28 DCX H ;HL=1000H
00FF 364F MVI M,4FH ;SCREEN PARAMETER BYTE 1
0101 3658 MVI M,5BH ;SCREEN PARAMETER BYTE 2
0103 3689 MVI M,89H ;SCREEN PARAMETER BYTE 3
0105 36DD MVI M,0DDH ;SCREEN PARAMETER BYTE 4
0107 3713 INH H ;HL=1001H
0108 CD8803 CALL LDCUR ;LOAD THE CURSOR
010B 36E0 MVI M,0E0H ;PRESET COUNTERS
010D 3623 MVI M,23H ;START DISPLAY
231
232 ;
233 ;THIS ROUTINE READS BOTH THE KEYBOARD AND THE USART
234 ;AND TAKES PROPER ACTION DEPENDING ON HOW THE LINE-LOCAL
235 ;SWITCH IS SET
236
010F 3E18 SETUP: MVI A,10H ;SET MASK
0111 30 SIM ;LOAD MASK
0112 FB EI ;ENABLE INTERRUPTS
240
241 ;
242 ;READ THE USART
243
0113 20 RXRDY: RIM ;GET LINE LOCAL
0114 E680 ANI 80H ;IS IT ON OR OFF?
0116 C22101 JNZ KEYINP ;LEAVE IF IT IS ON
0119 3A01A0 LDA ISTF ;READ 8251 FLAGS
011C B602 ANI 02H ;LOOK AT RXRDY
011E C2501 JNZ OK7 ;IF HAVE CHARACTER GO TO WORK
0121 3AE0F KEYINP: LDA KEYDWN ;GET KEYBOARD CHARACTER
0124 E680 ANI 80H ;IS IT THERE
0126 C23101 JNZ KEYS ;IF KEY IS PUSHED LEAVE
0129 3600 MVI A,00H ;ZERO A
012B 3ED0F STA KEYOK ;CLEAR KEYOK
012E C31301 JMP RXRDY ;LOOP AGAIN
0131 3AED0F LDA KEYOK ;WAS KEY DOWN
0134 4F MOV C,A ;SAVE A IN C
0135 3AB0F LDA KBCHR ;GET KEYBOARD CHARACTER
0138 B9 CMP C ;IS IT THE SAME AS KEYOK
0139 CA1301 JZ RXRDY ;IF SAME LOOP AGAIN
013C 3ED0F STA KEYOK ;IF NOT SAVE IT
013F 3E70F STA USCHR ;SAVE IT
0142 20 RIM ;GET LINE LOCAL
0143 E680 ANI 80H ;WHICH WAY
0145 C24801 JZ TRANS ;LEAVE IF LINE
0148 C34E02 JMP CHREC ;TIME TO DO SOME WORK
014B 3A01A0 LDA ISTF ;GET USART FLAGS
014E B601 ANI 01H ;READY TO TRANSMIT?
0150 CA4801 JZ TRANS ;LOOP IF NOT READY
0153 3AE70F LDA USCHR ;GET CHARACTER
0156 3200A0 STA USTD ;PUT IN USART
0159 C30F01 JMP SETUP ;LEAVE
015C 3A00A0 LDA USTD ;READ USART
015F B57E ANI 07FH ;STRIP MSB
0161 32E70F STA USCHR ;PUT IT IN MEMORY
0164 C34E02 JMP CHREC ;LEAVE
275
276 ;
277 ;THIS ROUTINE CHECKS THE BAUD RATE SWITCHES, RESETS THE
278 ;SCREEN POINTERS AND READS AND LOOKS UP THE KEYBOARD.
279
0167 F5 FRAME: PUSH PSW ;SAVE A AND FLAGS
0168 F5 PUSH H ;SAVE H AND L
0169 D5 PUSH D ;SAVE D AND E
016A C5 PUSH B ;SAVE B AND C
016B 3A0114 LDA INT75 ;READ 8275 TO CLEAR INTERRUPT
285
286 ;SET UP THE POINTERS
287
016E 2AE30F LHLD TOPAD ;LOAD TOP IN H AND L
0171 22E80F SHLD CURAD ;STORE TOP IN CURRENT ADDRESS
290
291 ;SET UP BAUD RATE
292
0174 3A0218 LDA PORTC ;READ BAUD RATE SWITCHES
0177 E60F ANI 0FH ;STRIP OFF 4 MSB'S
0179 47 MOV B,A ;SAVE IN B
017A 3AEC0F LDA BAUD ;GET BAUD RATE
017D 88 CMP B ;SEE IF SAME AS B
017E C4DC00 CNZ STBAUD ;IF NOT SAME DO SOMETHING
299
300 ;READ KEYBOARD
301
0181 3AEA0F LDA KEYDWN ;SEE IF A KEY IS DOWN
0184 E640 ANI 40H ;SET THE FLAGS
0186 C2C201 JNZ KYDWN ;IF KEY IS DOWN JUMP AROUND
0189 CD8E01 CALL RDKG ;GO READ THE KEYBOARD
018C C3BF00 JMP BYPASS ;LEAVE

```

```

018F 21EF0F 307 RDKB: LXI H,SHCON ;POINT HL AT KEYBOARD RAM
0192 3A0218 308 LDA PORTC ;GET CONTROL AND SHIFT
0195 5117 309 MOV M,A ;SAVE IN MEMORY
0196 3EFE 310 MVI A,0FFH ;SET UP A
0198 320018 311 LOOPK: STA PORTA ;OUTPUT A
0199 3A0118 312 MOV M,B ;SAVE A IN B
019C 3A0118 313 LDA PORTB ;READ KEYBOARD
019F 2F 314 CMA ;INVERT A
01A0 87 315 JNZ A ;SET THE FLAGS
01A1 2AF01 316 ORA SAVKEY ;LEAVE IF KEY IS DOWN
01A4 77 317 MOV M,A ;GET SCAN LINE BACK
01A5 07 318 RLC ;ROTATE IT OVER ONE
01A6 DA9001 319 JC LOOPK ;DO IT AGAIN
01A9 3E00 320 XRA A ;ZERO A
01AB 3EABF 321 STA KEYDOWN ;SAVE KEY DOWN
01AE C9 322 RET ;LEAVE
01AF 23 323 SAVKEY: INX H ;POINT AT RETURN LINE
01B0 2F 324 CMA ;PUT A BACK
01B1 77 325 MOV M,A ;SAVE RETURN LINE IN MEMORY
01B2 23 326 INX H ;POINT H AT SCAN LINE
01B3 70 327 MOV M,B ;SAVE SCAN LINE IN MEMORY
01B4 3E40 328 MVI A,40H ;SET A
01B6 3EABF 329 STA KEYDOWN ;SAVE KEY DOWN
01B9 C9 330 RET ;LEAVE
01BA 3E00 331 KYCHNG: MVI A,00H ;ZERO 0
01BC 32EA0F 332 STA KEYDOWN ;RESET KEY DOWN
01BF C9BF00 333 JMP BYPASS ;LEAVE
01C2 21F10F 334 KYDOWN: LXI H,SHCON ;GET SCAN LINE
01C5 7E 335 MOV M,A ;PUT SCAN LINE IN A
01C6 320018 336 STA PORTA ;OUTPUT SCAN LINE TO PORT A
01C9 2B 337 INX H ;POINT AT RETURN LINE
01CA 3A0118 338 LDA PORTB ;GET RETURN LINES
01CD B6 339 ORA M ;ARE THEY THE SAME?
01CE 2F 340 CMA ;INVERT A
01CF 87 341 ORA A ;SET FLAGS
01D0 CAB01 342 JZ KYCHNG ;IF DIFFERENT KEY HAS CHANGED
01D3 3AEA0F 343 LDA KEYDOWN ;GET KEY DOWN
01D6 E501 344 ANI 01H ;HAS THIS BEEN DONE BEFORE?
01D8 C9BF00 345 JMP BYPASS ;LEAVE IF IT HAS
01DB 3A0118 346 LDA PORTB ;GET RETURN LINE
01DE 06FF 347 MVI B,0FFH ;GET READY TO ZERO B
01E0 04 348 UP: INR B ;ZERO B
01E1 8F 349 RRC ;ROTATE A
01E2 DAE001 350 JC UP ;DO IT AGAIN
01E5 23 351 INX H ;POINT H AT SCAN LINES
01E6 7E 352 MOV M,A ;GET SCAN LINES
01E7 0E0F 353 MVI C,0FFH ;GET READY TO LOOP
01E9 0C 354 UP1: INR C ;START C COUNTING
01EA 0F 355 RRC ;ROTATE A
01EB DAE901 356 JC UP1 ;JUMP TO LOOP
01EE 78 357 MOV M,A ;GET RETURN LINES
01EF 07 358 RLC ;MOVE OVER ONCE
01F0 07 359 RLC ;MOVE OVER TWICE
01F1 07 360 RLC ;MOVE OVER THREE TIMES
01F2 81 361 ORA A ;OR SCAN AND RETURN LINES
01F5 47 362 MOV M,B ;SAVE A IN B
01F4 3A0218 363 LDA PORTC ;GET SHIFT CONTROL
01F7 E540 364 ANI 40H ;IS CONTROL SET
01F9 4F 365 MOV C,A ;SAVE A IN C
01FA 3AE00F 366 LDA SHCON ;GET SHIFT CONTROL
01FD 57 367 MOV D,A ;SAVE A IN D
01FE E540 368 ANI 40H ;STRIP CONTROL
0200 B1 369 ORA C ;SET BIT
0201 CA3E02 370 JZ ;IF SET LEAVE
0204 3A0218 371 LDA PORTC ;READ IT AGAIN
0207 E520 372 ANI 20H ;STRIP SHIFT
0209 4F 373 MOV C,A ;SAVE A
020B 7A 374 MOV C,D ;GET SHIFT CONTROL
020E E520 375 ANI 20H ;STRIP CONTROL
020D B1 376 ORA C ;ARE THEY THE SAME?
020F CA4702 377 JZ ;IF SET LEAVE
0211 58 378 SCR: MOV E,B ;PUT TARGET IN E
0212 1600 379 MVI D,00H ;ZERO D
0213 210705 380 LXI H,KYLKJP ;GET LOOKUP TABLE
0217 19 381 DAD D ;GET OFFSET
0218 7E 382 MOV M,A ;GET CHARACTER
0219 47 383 MOV M,B ;PUT CHARACTER IN B
021A 3A0218 384 LDA PORTC ;GET PORTC
021B E610 385 ANI 10H ;STRIP BIT
021D CA2E02 386 JC CAPLOC ;CAPS LOCK
0222 78 387 MOV A,B ;GET A BACK
0223 32E00F 388 STA KBCHR ;SAVE CHARACTER
0225 8C1 389 MVI A,0C1H ;SET A
0228 32EA0F 390 STA KEYDOWN ;SAVE KEY DOWN
022B C9BF00 391 JMP BYPASS ;LEAVE
022C 392 ;
022D 393 ;
022E 394 ; IF THE CAP LOCK BUTTON IS PUSHED THIS ROUTINE SEES IF
; THE CHARACTER IS BETWEEN 61H AND 7AH AND IF IT IS THIS

```

```

395 ;ROUTINE ASSUMES THAT THE CHARACTER IS LOWER CASE ASCII
396 ;AND SUBTRACTS 20H, WHICH CONVERTS THE CHARACTER TO
397 ;UPPER CASE ASCII
398
022E 78 399 CAPLOC: MOV A,B ;GET A BACK
022F FE60 400 CPI 60H ;HOW BIG IS IT?
0231 DA2302 401 JZ STKEY ;LEAVE IF IT'S TOO SMALL
0234 FE78 402 CPI 78H ;IS IT TOO BIG
0236 D22302 403 JNC STKEY ;LEAVE IF TOO BIG
0239 D520 404 SUI 20H ;ADJUST A
023B C32302 405 JMP STKEY ;STORE THE KEY
406
407 ;THE ROUTINES SHDN AND CNTDWN SET BIT 6 AND 7 RESPECTIVLY
408 ;IN THE ACC.
409
023E 3E80 410 CNTDWN: MVI A,00H ;SET BIT 7 IN A
0240 B6 411 ORA B ;OR WITH CHARACTER
0241 4B8F 412 JZ BRFH ;MAKE SURE SHIFT IS NOT SET
0243 47 413 MOV B,A ;PUT IT BACK IN B
0244 C31102 414 JMP SCR ;GO BACK
0247 3E40 415 SHDN: MVI A,40H ;SET BIT 6 IN A
0249 B6 416 ORA B ;OR WITH CHARACTER
024A 47 417 MOV B,A ;PUT IT BACK IN B
024B C31102 418 JMP SCR ;GO BACK
419
420 ;THIS ROUTINE CHECKS FOR ESCAPE CHARACTERS, LF, CR,
421 ;FF, AND BACK SPACE
422
024E 3AE80F 423 CHRBC: LDA ESCP ;ESCAPE SET?
0251 FE60 424 CPI 60H ;SEE IF IT IS
0253 CA7B02 425 JZ ESSO ;LEAVE IF IT IS
0256 3AE70F 426 LDA USCHR ;GET CHARACTER
0259 FE8A 427 CPI 0AH ;LINE FEED
025B CAE503 428 LAFO JZ TO LINE FEED
025E FE0C 429 CPI 0CH ;FORM FEED
0260 CACA03 430 JZ FMPD ;GO TO FORM FEED
0263 FE00 431 CPI 0DH ;CR
0265 CAAD03 432 JZ A CR ;DO A CR
0268 FE08 433 CPI 08H ;BACK SPACE
026A CA6E03 434 JZ LEFT ;DO A BACK SPACE
026D FE1B 435 CPI 1BH ;ESCAPE
026F CA8503 436 LAFO JZ AN ESCAPE
0272 B7 437 ORA A ;CLEAR CARRY
0273 C6E0 438 ADI 0E0H ;SEE IF CHARACTER IS PRINTABLE
0275 DA7704 439 JC CHRPUT ;IF PRINTABLE DO IT
0278 C30F01 440 JMP SETUP ;GO BACK AND READ USART AGAIN
441
442 ;THIS ROUTINE RESETS THE ESCAPE LOCATION AND DECODES
443 ;THE CHARACTERS FOLLOWING AN ESCAPE. THE COMMANDS ARE
444 ;COMPATIBLE WITH INTELS CREDIT TEXT EDITOR
445
027B 3E80 446 ESSO: MVI A,00H ;ZERO A
027D 32E80F 447 STA ESCP ;RESET ESCP
0280 3AE70F 448 LDA USCHR ;GET CHARACTER
0283 FE42 449 CPI 42H ;DOWN
0285 CAAB02 450 JZ DOWN ;MOVE CURSOR DOWN
0288 FE45 451 CPI 45H ;CLEAR SCREEN CHARACTER
028A CAE002 452 JZ CLEAR ;CLEAR THE SCREEN
028D FE4A 453 CPI 4AH ;CLEAR REST OF SCREEN
028F CAD502 454 JZ CLRST ;GO CLEAR THE REST OF THE SCREEN
0292 FE4B 455 CPI 4BH ;CLEAR LINE CHARACTER
0294 CA2703 456 JZ CLRLIN ;GO CLEAR A LINE
0297 FE41 457 CPI 41H ;CURSOR UP CHARACTER
0299 CA3303 458 JZ URCUR ;MOVE CURSOR UP
029C FE43 459 CPI 43H ;CURSOR RIGHT CHARACTER
029E CA4503 460 JZ RIGHT ;MOVE CURSOR TO THE RIGHT
02A1 FE44 461 CPI 44H ;CURSOR LEFT CHARACTER
02A3 CA6E03 462 JZ LEFT ;MOVE CURSOR TO THE LEFT
02A6 FE48 463 CPI 48H ;HOME CURSOR CHARACTER
02A8 CA9703 464 JZ HOME ;HOME THE CURSOR
02AB C30F01 465 JMP SETUP ;LEAVE
466
467 ;THIS ROUTINE MOVES THE CURSOR DOWN ONE CHARACTER LINE
468
02AE 3AE10F 469 DOWN: LDA CURSY ;PUT CURSOR Y IN A
02B1 FE18 470 CPI CURSB0 ;SEE IF ON BOTTOM OF SCREEN
02B3 CA0F01 471 JZ SETUP ;LEAVE IF ON BOTTOM
02B6 3C 472 INR A ;INCREMENT Y CURSOR
02B7 32E10F 473 STA CURSY ;SAVE NEW CURSOR
02BA CDB803 474 CALL LDCUR ;LOAD THE CURSOR
02BD CDA504 475 CALL CALCU ;CALCULATE ADDRESS
02C0 7E 476 MOV A,M ;GET FIRST LOCATION OF THE LINE
02C1 FE00 477 CPI 0E0H ;SEE IF CLEAR SCREEN CHARACTER
02C3 C20F01 478 JNZ SETUP ;LEAVE IF IT IS NOT
02C6 22E50F 479 SHLD LOCB0 ;SAVE BEGINNING OF THE LINE
02C9 CD1504 480 CALL CLLINE ;CLEAR THE LINE
02CC C30F01 481 JMP SETUP ;LEAVE
482

```

207780-40

```

483 ;THIS ROUTINE CLEARS THE SCREEN.
484
02CF CDE483 CLEAR: CALL CISCOR ;GO CLEAR THE SCREEN
02D2 C38F01 JMP SETUP ;GO BACK
;
; THIS ROUTINE CLEARS ALL LINES BENEATH THE LOCATION
; OF THE CURSOR.
02D5 CDA504 CLARST: CALL CALCU ;CALCULATE ADDRESS
02D8 CDCD84 CALL ADX ;ADD X POSITION
02DB 01204F LXI B,4F20H ;PUT SPACE AND LAST X IN B AND C
02DE 3AE20F LDA CURSX ;GET X CURSOR
02E1 B8 CMP B ;SEE IF AT END OF LINE
02E2 CABC02 OVR1: OVR1 ;LEAVE IF X IS AT END OF LINE
02E5 3C 497 LLP: INR A ;MOVE A OVER ONE X POSITION
02E6 23 498 INX H ;INCREMENT MEMORY POINTER
02E7 71 499 MOV M,C ;PUT A SPACE IN MEMORY
02E8 88 500 CMP B ;SEE IF A = 4FH
02E9 C2E502 JNZ LLP ;IF NOT LOOP AGAIN
02EC 01D00F OVR1: LXI B, LAST ;PUT LAST LINE IN BC
02EE 23 503 INX H ;POINT HL TO LAST LINE
02F0 78 504 MOV A,B ;GET B
02F1 8C 505 CMP A,B ;SAME AS H?
02F2 C2FD02 JNZ CONCL ;LEAVE IF NOT
02F3 88 506 MOV A,C ;GET C
02F4 8D 507 CMP A,C ;SAME AS L?
02F5 BD 508 JNZ CONCL ;LEAVE IF NOT
02F7 C2FD02 JNZ CONCL ;LEAVE IF NOT
02FA 210008 LXI H,TPDIS ;GET TOP OF DISPLAY
02FD 3AE10F CONCL: LDA CURSY ;GET Y CURSOR
0300 FE18 CPI CURBOT ;IS IT ON THE BOTTOM
0302 CA0F01 JZ SETUP ;LEAVE IF IT IS
0305 3C 514 INR A ;MOVE IT DOWN ONE LINE
0306 47 515 MOV B,A ;SAVE CURSOR IN B FOR LATER
0307 115000 LXI D,LENGTH ;PUT LENGTH OF ONE LINE IN D
030A 35F0 MVI M,0F0H ;PUT EOR IN MEMORY
030C 78 518 MOV A,B ;GET CURSOR Y
030D FE18 CPI CURBOT ;ARE WE ON THE BOTTOM
030E CA0F01 JZ SETUP ;LEAVE IF WE ARE
0312 3C 521 INR A ;MOVE CURSOR DOWN ONE
0313 19 522 DAD D ;GET NEXT LINE
0314 47 523 MOV B,A ;SAVE A
0315 7C 524 MOV A,H ;PUT H IN A
0316 FB0F CPI 0FH ;COMPARE TO HIGH LAST
0318 C28A03 JNZ CLOOP ;LEAVE IF IT IS NOT
031B 7D 527 MOV A,L ;PUT L IN A
031C FE00 CPI 00H ;COMPARE TO LOW LAST
031E C28A03 JNZ CLOOP ;LEAVE IF IT IS NOT
0321 210008 LXI H,TPDIS ;PUT TOP DISPLAY IN H AND L
0324 C38A03 JMP CLOOP ;LOOP AGAIN
;
; THIS ROUTINE CLEARS THE LINE THE CURSOR IS ON.
0327 CDA504 CLRALIN: CALL CALCU ;CALCULATE ADDRESS
032A 22E50F SHLD LOC00 ;STORE H AND L TO CLEAR LINE
032D CD1504 CALL CLLINE ;CLEAR THE LINE
0330 C38F01 JMP SETUP ;GO BACK
;
; THIS ROUTINE MOVES THE CURSOR UP ONE LINE.
0333 3AE10F UPCUR: LDA CURSY ;GET Y CURSOR
0336 FE00 CPI 00H ;IS IT ZERO
0338 CA0F01 JZ SETUP ;IF IT IS LEAVE
033B 3D 545 DCR A ;MOVE CURSOR UP
033C 32E10F STA CURSY ;SAVE NEW CURSOR
033F CB8003 CALL LDCUR ;LOAD THE CURSOR
0342 C38F01 JMP SETUP ;LEAVE
;
; THIS ROUTINE MOVES THE CURSOR ONE LOCATION TO THE RIGHT
0345 3AE20F RIGHT: LDA CURSX ;GET X CURSOR
0348 FE00 CPI 4FH ;IS IT ALL THE WAY OVER?
034A C26403 JNZ NTOVER ;IF NOT JUMP AROUND
034D 3AE10F LDA CURSY ;GET Y CURSOR
0348 FE18 CPI CURBOT ;SEE IF ON BOTTOM
0352 CA5903 JZ GD18 ;IF WE ARE JUMP
0355 3C 550 INR A ;INCREMENT Y CURSOR
0356 32E10F STA CURSY ;SAVE IT
0359 88 550 MOV A,00H ;ZERO A
035B 32E20F STA CURSX ;ZERO X CURSOR
035D CB8003 CALL LDCUR ;LOAD THE CURSOR
0358 C30F01 JMP SETUP ;LEAVE
0361 3C 554 INR A ;INCREMENT X CURSOR
0365 32E20F STA CURSX ;SAVE IT
0368 CB8003 CALL LDCUR ;LOAD THE CURSOR
036B C38F01 JMP SETUP ;LEAVE
;
; THIS ROUTINE MOVES THE CURSOR LEFT ONE CHARACTER POSITION

```

```

570
036E 3AE20F 571 LEFT: LDA CURSX ;GET X CURSOR
0371 FE00 572 CPI 00H ;IS IT ALL THE WAY OVER
0374 C30003 573 JNZ NOVER ;IF NOT JUMP AROUND
0376 3AE10F 574 LDA CURSY ;GET CURSOR Y
0379 FE00 575 CPI 00H ;IS IT ZERO?
037B C0F0F1 576 JZ SETUP ;IF IT IS JUMP
037E 0D 577 DCR A ;MOVE CURSOR Y UP
037F 32E10F 578 STA CURSY ;SAVE IT
0382 3E4F 579 MVI A,4FH ;GET LAST X LOCATION
0384 32E20F 580 STA CURSX ;SAVE IT
0387 CDB003 581 CALL LDCUR ;LOAD THE CURSOR
038A C30F01 582 JMP SETUP
038D 3D 583 NOVER: DCR A ;ADJUST X CURSOR
038E 32E20F 584 STA CURSX ;SAVE CURSOR X
0391 CDB003 585 CALL LDCUR ;LOAD THE CURSOR
0394 C30F01 586 JMP SETUP ;LEAVE
587
;THIS ROUTINE HOMES THE CURSOR.
588
0397 3E00 590 HOME: MVI A,00H ;ZERO A
0399 32E20F 591 STA CURSX ;ZERO X CURSOR
039C 32E10F 592 STA CURSY ;ZERO Y CURSOR
039F CDB003 593 CALL LDCUR ;LOAD THE CURSOR
03A2 C30F01 594 JMP SETUP ;LEAVE
595
;THIS ROUTINE SETS THE ESCAPE BIT
596
03A5 3E00 599 ESKAP: MVI A,00H ;LOAD A WITH ESCAPE BIT
03A7 32EE0F 599 STA ESCP ;SET ESCAPE LOCATION
03AA C30F01 600 JMP SETUP ;GO BACK AND READ USART
601
;THIS ROUTINE DOES A CR
602
03AD 3E00 603 CGRT: MVI A,00H ;ZERO A
03AF 32E20F 604 STA CURSX ;ZERO CURSOR X
03B2 CDB003 605 CALL LDCUR ;LOAD CURSOR INTO 8275
03B5 C30F01 607 JMP SETUP ;POLL USART AGAIN
608
;THIS ROUTINE LOADS THE CURSOR
609
03B8 3E00 611 LDCUR: MVI A,00H ;PUT 00H INTO A
03BA 320110 612 STA CRTS ;LOAD CURSOR INTO 8275
03BD 3A200F 613 LDA CURSX ;GET CURSOR X
03C0 320010 614 STA CRTM ;PUT IT IN 8275
03C3 3AE10F 615 LDA CURSY ;GET CURSOR Y
03C6 320010 616 STA CRTM ;PUT IT IN 8275
03C9 C9 617 RET
618
;THIS ROUTINE DOES A FORM FEED
619
03CA CDE403 620 FMFD: CALL CLSCR ;CALL CLEAR SCREEN
03CD 210000 621 LXI H,TPDIS ;PUT TOP DISPLAY IN HL
03D0 22E50F 622 SHLD LDC00 ;PUT IT IN LDC00
03D3 CD1504 624 CALL CLLINE ;CLEAR TOP LINE
03D6 3E00 625 MVI A,00H ;ZERO A
03D8 32E20F 626 STA CURSX ;ZERO CURSOR X
03DB 32E10F 627 STA CURSY ;ZERO CURSOR Y
03DE CDB003 628 CALL LDCUR ;LOAD THE CURSOR
03E1 C30F01 629 JMP SETUP ;BACK TO USART
630
;THIS ROUTINE CLEARS THE SCREEN BY WRITING END OF ROW
631 ;CHARACTERS INTO THE FIRST LOCATION OF ALL LINES ON
632 ;THE SCREEN.
633
03E4 3EF0 635 CLSCR: MVI A,0F0H ;PUT EOR CHARACTER IN A
03E6 0618 636 MVI B,CURBOT ;LOAD B WITH MAX Y
03E8 04 637 INR B ;GO TO MAX PLUS ONE
03E9 210000 638 LXI H,TPDIS ;LOAD H AND L WITH TOP OF RAM
03EC 115000 639 LXI D,LANGTH ;MOVE 50H = 80D INTO D AND E
03EF 10 640 LOADX: MOV M,A ;MOVE EOR INTO MEMORY
03F0 04 641 DAD D ;CHANGE POINTER BY 80D
03F1 05 642 DCR B ;COUNT THE LOOPS
03F2 C2EF03 643 JNZ LOADX ;CONTINUE IF NOT ZERO
03F5 C9 644 RET ;GO BACK
645
;THIS ROUTINE DOES A LINE FEED
646
03F6 CDF003 648 LNFD: CALL LNFDI ;CALL ROUTINE
03F9 C30F01 649 JMP SETUP ;POLL FLAGS
650
;LINE FEED
651
03FC 3AE10F 653 LNFDI: LDA CURSY ;GET Y LOCATION OF CURSOR
03FF FE18 654 CPI CURBOT ;SEE IF AT BOTTOM OF SCREEN
0401 CA5304 655 JZ ONBOT ;IF WE ARE, LEAVE
0404 3C 656 INR A ;INCREMENT A
0405 32E10F 657 STA CURSX ;SAVE NEW CURSOR

```

207780-42

```

0408 CDA504 658 CALL CALCU ;CALCULATE ADDRESS
0408 22E50F 659 SHLD LOC80 ;SAVE TO CLEAR LINE
0408 CDI504 660 CALL CLLINE ;CLEAR THE LINE
0411 CDB883 661 CALL LDCUR ;LOAD THE CURSOR
0414 C9 662 RET ;LEAVE
;
663
664 ;THIS ROUTINE CLEARS THE LINE WHOSE FIRST ADDRESS
665 ;IS IN LOC80. PUSH INSTRUCTIONS ARE USED TO RAPIDLY
666 ;CLEAR THE LINE
667
0415 F3 668 CALL: DI ;NO INTERRUPTS HERE
0416 2AE50F 669 LHLD LOC80 ;GET LOC80
0419 115000 670 LXI D,LENGTH ;GET OFFSET
041C 19 671 DAD D ;ADD OFFSET
041D EB 672 XCHG ;PUT START IN DE
0418 210000 673 LXI H,0000H ;ZERO HL
0421 39 674 DAD SP ;GET STACK
0422 EB 675 XCHG ;PUT STACK IN DE
0423 F9 676 SPHL ;PUT START IN SP
0424 212020 677 LXI H,2020H ;PUT SPACES IN HL
;
678 ;NOW DO 40 PUSH INSTRUCTIONS TO CLEAR THE LINE
679 ;
680
681 REPT (LENGTH/2)
682 H
683 ENDM
0427 EB 684+ PUSH H
0428 EB 685+ PUSH H
0429 EB 686+ PUSH H
042A EB 687+ PUSH H
042B EB 688+ PUSH H
042C EB 689+ PUSH H
042D EB 690+ PUSH H
042E EB 691+ PUSH H
042F EB 692+ PUSH H
0430 EB 693+ PUSH H
0431 EB 694+ PUSH H
0432 EB 695+ PUSH H
0433 EB 696+ PUSH H
0434 EB 697+ PUSH H
0435 EB 698+ PUSH H
0436 EB 699+ PUSH H
0437 EB 700+ PUSH H
0438 EB 701+ PUSH H
0439 EB 702+ PUSH H
043A EB 703+ PUSH H
043B EB 704+ PUSH H
043C EB 705+ PUSH H
043D EB 706+ PUSH H
043E EB 707+ PUSH H
043F EB 708+ PUSH H
0440 EB 709+ PUSH H
0441 EB 710+ PUSH H
0442 EB 711+ PUSH H
0443 EB 712+ PUSH H
0444 EB 713+ PUSH H
0445 EB 714+ PUSH H
0446 EB 715+ PUSH H
0447 EB 716+ PUSH H
0448 EB 717+ PUSH H
0449 EB 718+ PUSH H
044A EB 719+ PUSH H
044B EB 720+ PUSH H
044C EB 721+ PUSH H
044D EB 722+ PUSH H
044E EB 723+ PUSH H
044F EB 724+ XCHG ;PUT STACK IN HL
0450 F9 725 SPHL ;PUT IT BACK IN SP
0451 FB 726 EI ;ENABLE INTERRUPTS
0452 C9 727 RET ;GO BACK
;
728 ;IF CURSOR IS ON THE BOTTOM OF THE SCREEN THIS ROUTINE
729 ;IS USED TO IMPLEMENT THE LINE FEED
730
0453 2AE30F 731 ONBOT: LHLD TOPAD ;GET TOP ADDRESS
0456 22E50F 732 SHLD LOC80 ;SAVE IT IN LOC80
0459 115000 733 LXI D,LENGTH ;LINE LENGTH
045C 19 734 DAD D ;ADD HL + DE
045D 01D00F 735 LXI B,LAST ;GET BOTTOM LINE
0460 7C 736 MOV A,H ;GET H
0461 BB 737 CMP B ;SAME AS B
0462 C26D04 738 JNZ ARND ;LEAVE IF NOT SAME
0465 7D 739 MOV A,L ;GET L
0466 B9 740 CMP C ;SAME AS C
0467 C26D04 741 JNZ ARND ;LEAVE IF NOT SAME
0468 210000 742 LXI H,TPDIS ;LOAD HL WITH TOP OF DISPLAY
046D 22E30F 743 ARND: SHLD TOPAD ;SAVE NEW TOP ADDRESS

```

```

0470 CD1504      745      CALL  CLLINE      ;CLEAR LINE
0473 CDB803      746      CALL  LDCUR      ;LOAD THE CURSOR
0476 C9          747      RET
              748      ;
              749      ; THIS ROUTINE PUTS A CHARACTER ON THE SCREEN AND
              750      ; INCREMENTS THE X CURSOR POSITION. A LINE FEED IS
              751      ; INSERTED IF THE INCREMENTED CURSOR EQUALS 81D
0477 CD4504      752      CHRPUT: CALL  CALCU      ;CALCULATE SCREEN POSITION
047A 7E          754      MOV   A,M        ;GET FIRST CHARACTER
047B FE20        755      CPI   000H      ;IS IT A CLEAR LINE
047D 22E50F      756      SHLD LOC00     ;SAVE LINE TO CLEAR
0480 C11504      757      CZ    C,LINE    ;CLEAR LINE
0483 2AE50F      758      LHL  LOC00     ;GET LINE
0486 CDCD04      759      CALL  ADX       ;ADD CURSOR X
0489 3AE70F      760      LDA   USC R    ;GET CHARACTER
048C 77          761      MVI  A,SCREEN  ;PUT IT ON SCREEN
048D 3AE20F      762      LDA   CURSX   ;GET CURSOR X
0490 3C          763      INR  A        ;INCREMENT CURSOR X
0491 FE50        764      CFI  INOATH   ;HAS IT GONE TOO FAR?
0493 C29C04      765      JNZ  OK1      ;IF NOT GOOD
0496 CDE003      766      CALL  LNF01   ;DO A LINE FEED
0499 C3AD03      767      JMP  CGRT     ;DO A CR
049C 32E20F      768      OK1: STA  CURSX ;SAVE CURSOR
049F CDB803      769      CALL  LDCUR   ;LOAD THE CURSOR
04A2 C30F01      770      JMP  SETUP    ;LEAVE
              771      ;
              772      ; THIS ROUTINE TAKES THE TOP ADDRESS AND THE Y CURSOR
              773      ; LOCATION AND CALCULATES THE ADDRESS OF THE LINE
              774      ; THAT THE CURSOR IS ON. THE RESULT IS RETURNED IN H
              775      ; AND L AND ALL REGISTERS ARE USED.
04A5 21D504      777      CALCU: LXI  H,LINTAB ;GET LINE TABLE INTO H AND L
04A8 3AE10F      778      LDA   CURSY   ;GET CURSOR INTO A
04AB 07          779      RLC          ;SET UP A FOR LOOKUP TABLE
04AC 0F00        780      MVI  B,00H    ;ZERO B
04AE 4E          781      MOV  C,A     ;PUT CURSOR INTO A
04AF 09          782      DAD  B       ;ADD LINE TABLE TO Y CURSOR
04B0 7E          783      MOV  A,M     ;PUT LOW LINE TABLE INTO A
04B1 4E          784      MOV  C,A     ;PUT LOW LINE TABLE INTO C
04B2 23          785      INX  H       ;CHANGE MEMORY POINTER
04B3 7E          786      MOV  A,M     ;PUT HIGH LINE TABLE INTO A
04B4 47          787      MOV  B,A     ;PUT HIGH LINE TABLE INTO B
04B5 2100F8     788      LXI  H,0F00H ;TWO'S COMPLEMENT SCREEN LOCATION
04B8 09          789      DAD  B       ;SUBTRACT OFFSET
04B9 EA          790      XCHG        ;SAVE HL IN DE
04BA 2AE30F     791      LHL  TOPAD   ;GET TOP ADDRESS IN H AND L
04BD 10          792      DAD  D       ;GET DISPLACED ADDRESS
04BE 03          793      XCHG        ;SAVE IT IN D
04BF 2130F0     794      LXI  H,0F03H ;TWO'S COMPLEMENT SCREEN LOCATION
04C2 19          795      DAD  D       ;SEE IF WE ARE OFF THE SCREEN
04C3 DAC804     796      JC   FIX     ;IF WE ARE FIX IT
04C6 EA          797      XCHG        ;GET DISPLACED ADDRESS BACK
04C7 C9          798      RET         ;GO BACK
04C8 2130F8     799      FIX: LXI  H,0F83H ;SCREEN BOUNDARY
04CB 19          800      DAD  D       ;ADJUST SCREEN
04CC C9          801      RET         ;GO BACK
              802      ;
              803      ; THIS ROUTINE ADDS THE X CURSOR LOCATION TO THE ADDRESS
              804      ; THAT IS IN THE H AND L REGISTERS AND RETURNS THE RESULT
              805      ; IN H AND L
04CD 3AE20F     806      ADX:  LDA   CURSX ;GET CURSOR
04D0 0F00        807      MVI  B,00H    ;ZERO B
04D2 4E          808      MOV  C,A     ;PUT CURSOR X IN C
04D3 09          809      DAD  B       ;ADD CURSOR X TO H AND L
04D4 C9          810      RET         ;LEAVE
              811      ;
              812      ; THIS TABLE CONTAINS THE OFFSET ADDRESSES FOR EACH
              813      ; OF THE 25 DISPLAYED LINES.
0000           814      LINTAB: LNNUM SET 0
              815      REPT (CURBOT+1)
04D5 0000        821+     DW   TDIS+(LANGH*LNNUM)
0001           822+     DW   LNNUM SET (LNNUM+1)
04D7 5000        823+     DW   TDIS+(LANGH*LNNUM)
0002           824+     DW   LNNUM SET (LNNUM+1)
04D9 A000        825+     DW   TDIS+(LANGH*LNNUM)
0003           826+     DW   LNNUM SET (LNNUM+1)
04DB F000        827+     DW   TDIS+(LANGH*LNNUM)
0004           828+     DW   LNNUM SET (LNNUM+1)
04DD 4000        829+     DW   TDIS+(LANGH*LNNUM)
0005           830+     DW   LNNUM SET (LNNUM+1)
04DF 9000        831+     DW   TDIS+(LANGH*LNNUM)

```



```

0006      832+   LINNUM SET (LINNUM+1)
04E1 E009    833+   DW      TPOIS+(LENGTH*LINNUM)
0007      834+   LINNUM SET (LINNUM+1)
04E3 300A    835+   DW      TPOIS+(LENGTH*LINNUM)
0008      836+   LINNUM SET (LINNUM+1)
04E5 800A    837+   DW      TPOIS+(LENGTH*LINNUM)
0009      838+   LINNUM SET (LINNUM+1)
04E7 D00A    839+   DW      TPOIS+(LENGTH*LINNUM)
000A      840+   LINNUM SET (LINNUM+1)
04E9 200B    841+   DW      TPOIS+(LENGTH*LINNUM)
000B      842+   LINNUM SET (LINNUM+1)
04EB 700B    843+   DW      TPOIS+(LENGTH*LINNUM)
000C      844+   LINNUM SET (LINNUM+1)
04ED C00B    845+   DW      TPOIS+(LENGTH*LINNUM)
000D      846+   LINNUM SET (LINNUM+1)
04EF 100C    847+   DW      TPOIS+(LENGTH*LINNUM)
000E      848+   LINNUM SET (LINNUM+1)
04F1 600C    849+   DW      TPOIS+(LENGTH*LINNUM)
000F      850+   LINNUM SET (LINNUM+1)
04F3 800C    851+   DW      TPOIS+(LENGTH*LINNUM)
0010      852+   LINNUM SET (LINNUM+1)
04F5 000D    853+   DW      TPOIS+(LENGTH*LINNUM)
0011      854+   LINNUM SET (LINNUM+1)
04F7 500D    855+   DW      TPOIS+(LENGTH*LINNUM)
0012      856+   LINNUM SET (LINNUM+1)
04F9 A00D    857+   DW      TPOIS+(LENGTH*LINNUM)
0013      858+   LINNUM SET (LINNUM+1)
04FB F00D    859+   DW      TPOIS+(LENGTH*LINNUM)
0014      860+   LINNUM SET (LINNUM+1)
04FD 400E    861+   DW      TPOIS+(LENGTH*LINNUM)
0015      862+   LINNUM SET (LINNUM+1)
04FF 900E    863+   DW      TPOIS+(LENGTH*LINNUM)
0016      864+   LINNUM SET (LINNUM+1)
0501 E00E    865+   DW      TPOIS+(LENGTH*LINNUM)
0017      866+   LINNUM SET (LINNUM+1)
0503 300F    867+   DW      TPOIS+(LENGTH*LINNUM)
0018      868+   LINNUM SET (LINNUM+1)
0505 800F    869+   DW      TPOIS+(LENGTH*LINNUM)
0019      870+   LINNUM SET (LINNUM+1)
          871     ;
          872     ;KEYBOARD LOOKUP TABLE
          873     ;THIS TABLE CONTAINS ALL THE ASCII CHARACTERS
          874     ;THAT ARE TRANSMITTED BY THE TERMINAL
          875     ;THE CHARACTERS ARE ORGANIZED SO THAT BITS 0,1 AND 2
          876     ;ARE THE SCAN LINES, BITS 3,4 AND 5 ARE THE RETURN LINES
          877     ;BIT 6 IS SHIFT AND BIT 7 IS CONTROL
          878
0507 38      879 KYLKUP: DB    38H,39H      ;8 AND 9
0508 39
0509 36      880     DB    30H,20H      ;0 AND -
050A 2D
050B 3D      881     DB    3DH,5CH      ;= AND \
050C 5C
050D 08      882     DB    08H,00H      ;BS AND BREAK
050E 08
050F 75      883     DB    75H,69H      ;LOWER CASE U AND I
0510 69
0511 6F      884     DB    6FH,70H      ;LOWER CASE O AND P
0512 6E
0513 5B      885     DB    5BH,5CH      ;[ AND \
0514 5C
0515 0A      886     DB    0AH,7FH      ;LF AND DELETE
0516 7F
0517 6A      887     DB    6AH,6BH      ;LOWER CASE J AND K
0518 6B
0519 5C      888     DB    6CH,3BH      ;LOWER CASE L AND ;
051A 3B
051B 27      889     DB    27H,00H      ;' AND NOTHING
051C 00
051D 0D      890     DB    0DH,37H      ;CR AND 7
051E 37
051F 6D      891     DB    6DH,2CH      ;LOWER CASE M AND COMMA
0520 2C
0521 2E      892     DB    2EH,2FH      ;PERIOD AND SLASH
0522 2F
0523 00      893     DB    00H,00H      ;BLANK AND NOTHING
0524 00
0525 00      894     DB    00H,00H      ;NOTHING AND NOTHING
0526 00
0527 00      895     DB    00H,61H      ;NOTHING AND LOWER CASE A
0528 61
0529 7A      896     DB    7AH,78H      ;LOWER CASE Z AND X
052A 78
052B 63      897     DB    63H,76H      ;LOWER CASE C AND V
052C 76
052D 62      898     DB    62H,6EH      ;LOWER CASE B AND N
052E 6E

```

052F 79	899	DB	79H,00H	;LOWER CASE Y AND NOTHING
0530 00				
0531 00	900	DB	00H,20H	;NOTHING AND SPACE
0532 20				
0533 64	901	DB	64H,6FH	;LOWER CASE D AND F
0534 65				
0535 67	902	DB	67H,68H	;LOWER CASE G AND H
0536 68				
0537 00	903	DB	00H,71H	;TAB AND LOWER CASE Q
0538 71				
0539 77	904	DB	77H,73H	;LOWER CASE W AND S
053A 73				
053B 65	905	DB	65H,72H	;LOWER CASE E AND R
053C 72				
053D 74	906	DB	74H,00H	;LOWER CASE T AND NOTHING
053E 00				
053F 1B	907	DB	1BH,31H	;ESCAPE AND I
0540 31				
0541 32	908	DB	32H,33H	; 2 AND 3
0542 33				
0543 34	909	DB	34H,35H	; 4 AND 5
0544 35				
0545 36	910	DB	35H,00H	; 6 AND NOTHING
0546 00				
0547 2A	911	DB	2AH,2BH	; * AND
0548 2B				
0549 29	912	DB	29H,5FH	; (AND -
054A 5F				
054B 2B	913	DB	2BH,00H	;+ AND NOTHING
054C 00				
054D 08	914	DB	08H,00H	;BS AND BREAK
054E 00				
054F 55	915	DB	55H,49H	;U AND I
0550 49				
0551 4F	916	DB	4FH,50H	;O AND P
0552 50				
0553 5D	917	DB	5DH,00H	;] AND NO CHARACTER
0554 00				
0555 0A	918	DB	0AH,7FH	;LF AND DELETE
0556 7F				
0557 4A	919	DB	4AH,4BH	;J AND K
0558 4B				
0559 4C	920	DB	4CH,3AH	;L AND :
055A 3A				
055B 22	921	DB	22H,00H	; " AND NO CHARACTER
055C 00				
055D 0D	922	DB	0DH,26H	;CR AND &
055E 26				
055F 4D	923	DB	4DH,3CH	;M AND <
0560 3C				
0561 3E	924	DB	3EH,3FH	; > AND ?
0562 3F				
0563 00	925	DB	00H,00H	;BLANK AND NOTHING
0564 00				
0565 00	926	DB	00H,00H	;NOTHING AND NOTHING
0566 00				
0567 00	927	DB	00H,41H	;NOTHING AND A
0568 41				
0569 5A	928	DB	5AH,58H	;Z AND X
056A 58				
056B 43	929	DB	43H,5FH	;C AND V
056C 5F				
056D 42	930	DB	42H,4EH	;B AND N
056E 4E				
056F 59	931	DB	59H,00H	;Y AND NOTHING
0570 00				
0571 00	932	DB	00H,20H	;NO CHARACTER AND SPACE
0572 20				
0573 44	933	DB	44H,4FH	;D AND F
0574 46				
0575 47	934	DB	47H,48H	;G AND H
0576 48				
0577 00	935	DB	00H,51H	;TAB AND Q
0578 51				
0579 57	936	DB	57H,53H	;W AND S
057A 53				
057B 45	937	DB	45H,52H	;E AND R
057C 52				
057D 54	938	DB	54H,00H	;T AND NO CONNECTION
057E 00				
057F 1B	939	DB	1BH,21H	;ESCAPE AND !
0580 21				
0581 40	940	DB	40H,23H	;@ AND #
0582 23				
0583 24	941	DB	24H,25H	; \$ AND &
0584 24				
0585 5E	942	DB	5EH,00H	; ^ AND NO CONNECTION

```

0586 00
043
044 ;THIS IS WHERE THE CONTROL CHARACTERS ARE LOOKED UP
045
046 DB 00H,00H ;NOTHING
0587 00
0588 00
0589 00
058A 00 947 DB 00H,00H ;NOTHING
058B 00
058C 00 948 DB 00H,00H ;NOTHING
058D 00
058E 00
058F 00
0590 00
0591 15 950 DB 15H,09H ;CONTROL U AND I
0592 09
0593 0F 951 DB 0FH,10H ;CONTROL O AND P
0594 10
0595 18
0596 00 952 DB 0BH,0CH ;CONTROL [ AND \
0597 00 953 DB 0AH,7FH ;LF AND DELETE
0598 00 954 DB 0AH,0BH ;CONTROL J AND K
0599 00 955 DB 0CH,00H ;CONTROL L AND NOTHING
059A 00
059B 00 956 DB 00H,00H ;NOTHING
059C 00
059D 00 957 DB 0DH,00H ;CR AND NOTHING
059E 00
059F 00 958 DB 0DH,00H ;CONTROL M AND COMMA
05A0 00
05A1 00 959 DB 00H,00H ;NOTHING
05A2 00
05A3 00 960 DB 00H,00H ;NOTHING
05A4 00
05A5 00 961 DB 00H,00H ;NOTHING AND NOTHING
05A6 00
05A7 1A 962 DB 1AH,18H ;CONTROL Z AND X
05A8 18
05A9 03 963 DB 03H,16H ;CONTROL C AND V
05AA 16
05AB 02 964 DB 02H,0EH ;CONTROL B AND N
05AC 0E
05AD 19 965 DB 19H,00H ;CONTROL Y AND NOTHING
05AE 00
05AF 00 966 DB 00H,20H ;NOTHING AND SPACE
05B0 20
05B1 04 967 DB 04H,0FH ;CONTROL D AND F
05B2 06
05B3 07 968 DB 07H,00H ;CONTROL G AND H
05B4 00
05B5 00 969 DB 00H,11H ;NOTHING AND CONTROL Q
05B6 11
05B7 17 970 DB 17H,13H ;CONTROL W AND S
05B8 13
05B9 06 971 DB 06H,12H ;CONTROL E AND R
05BA 12
05BB 14 972 DB 14H,00H ;CONTROL W AND NOTHING
05BC 00
05BD 1B 973 DB 1BH,1DH ;ESCAPE AND HOME (CREDIT)
05BE 1D
05BF 1E 974 DB 1EH,1CH ;CURSOR UP AND DOWN (CREDIT)
05C0 1C
05C1 14 975 DB 14H,1FH ;CURSOR RIGHT AND LEFT (CREDIT)
05C2 1F
05C3 00 976 DB 00H,00H ;NOTHING
05C4 00
977 ;
978 ;LOOK UP TABLE FOR 8253 BAUD RATE GENERATOR
979 ;
980 BDLC: DB 00H,05H,69H,03H ;75 AND 110 BAUD
05C5 00
05C6 05
05C7 03
05C8 03
05C9 00 981 DB 80H,02H,40H,01H ;150 AND 300 BAUD
05CA 02
05CB 49
05CC 01
05CD A0 982 DB 0A0H,00H ;600 BAUD
05CE 00
05CF 56 983 DB 50H,00H ;1200 BAUD
05D0 00
05D1 28 984 DB 28H,00H ;2400 BAUD
05D2 00
05D3 14 985 DB 14H,00H ;4800 BAUD
05D4 00
05D5 0A 986 DB 0AH,00H ;9600 BAUD
05D6 00

```

```

987          ;
988          ; DATA AREA
989          ;
00E1        ORG      00E1H
0001        991 CURSY: DS      1
0001        992 CURSX: DS      1
0002        993 TOPAD: DS      2
0002        994 LOCS0: DS      2
0001        995 USCHR: DS      1
0002        996 CURAD: DS      1
0001        997 KEYDWN: DS     1
0001        998 KBCHR: DS      1
0001        999 BAUD:  DS      1
0001        1000 KEYOK: DS      1
0001        1001 ESCP:  DS      1
0001        1002 SHCON: DS      1
0001        1003 RETLIN: DS      1
0001        1004 SCNLIN: DS      1
0001        1005        END
    
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

AIX	A 04CD	ARND	A 046D	BAUD	A 00FC	BCLK	A 05C5	BTDIS	A 0080	BYPASS	A 000F
CAPLOC	A 022E	CGRT	A 03AD	CHREC	A 024E	CHRPUR	A 0477	CLRAR	A 02CF	CLLINE	A 0415
CLRLIN	A 0327	CLAST	A 02D5	CLSCR	A 0384	CNT0	A 5000	CNT1	A 5001	CNT2	A 5002
CNTM	A 5003	CNTD5	A 1003	CONCL	A 02FD	CRFM	A 1000	CRIS	A 1001	CURAD	A 00E8
CURSK	A 00E2	CURSO	A 00E1	DOWN	A 02AE	ESCP	A 00E0	ESKAP	A 03A5	ESSQ	A 027B
EMFD	A 03CA	FRAME	A 0157	GD18	A 0359	HOME	A 0397	IN75	A 00F9	INT75	A 1401
KEYDWN	A 00EA	KEYLIP	A 0121	KEYOK	A 00ED	KEYS	A 0131	RPTR	A 0004	KYCHNG	A 01BA
KYLUP	A 0507	LAST	A 00D0	LCLR	A 0389	LEFT	A 036E	LINNUM	A 0019	LINTAB	A 04D5
LNFD	A 03F6	LNFD1	A 03FC	LNTPH	A 0050	LOADX	A 03EE	LOC00	A 00E5	LOOP	A 00A7
LPKBD	A 0098	NOVER	A 038D	NTOVER	A 0364	OK1	A 049C	OK7	A 015C	ONBOT	A 0453
POPDAT	A 0034	PORTA	A 1002	PORTB	A 1001	PORTC	A 1002	RDKB	A 010F	RETLIN	A 00E0
RDDY	A 0113	SAWKEY	A 01AF	SCNLIN	A 00E1	SCR	A 0211	SETUP	A 010F	SHCON	A 00E0
STBAUD	A 00DC	STKEY	A 0223	STPR	A 00E0	TOPAD	A 00E3	TFDIS	A 0000	TRANS	A 0148
UPI	A 01E9	URCLR	A 0333	USCHR	A 00E7	USTD	A A000	USTE	A A001		

ASSEMBLY COMPLETE, NO ERRORS

THE UNIVERSITY OF CHICAGO LIBRARY

1215 EAST 58TH STREET, CHICAGO, ILL. 60637

82716/VSDD VIDEO STORAGE AND DISPLAY DEVICE

- Low Cost Graphics Capability
- Minimum Chip Count Display Controller
- Displays Up to 16 Bit Map and Character Objects of Any Size
- On-Chip 16/4096 Color Palette
- On-Chip DRAM Controller
- On-Chip D/A Converters
- Arbitration of Processor RAM Requests
- NAPLPS and CEPT Compatible
- Objects Allow Windowing or Animation
- Resolution Up to 640 x 512 Pixels
- Up to 512K Bytes of Display Memory
- Compatible with 8 and 16 Bit Processors/Micro Controllers
- Twin Mode Operation for Higher Throughput
- Powerful External Sync and Overlay Capabilities

82716/VSDD is a low cost, highly integrated video controller. It displays graphics and textual information using a minimum of chips. It allows the management of up to 16 display objects on the screen at any one time. These objects may be formatted as bit map or character arrays and can be used for windowing or animation.

An on-chip color palette allows the selection of up to 16 colors, from a range of 4096. The palette can be programmed to drive a set of on-chip D/A converters. The VSDD also provides DRAM controller functions.

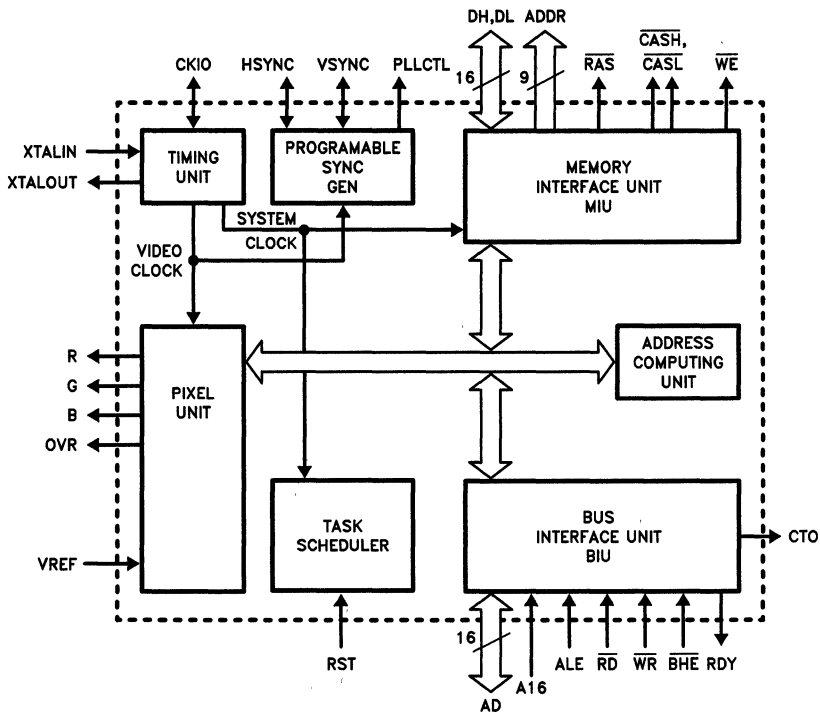


Figure 1: VSDD Block Diagram

231680-1

GENERAL DESCRIPTION

The 82716/VSDD is a low cost, highly integrated VLSI CRT controller offering advanced display capabilities for Videotex and color graphics displays. Its internal architecture allows it to be connected to any Intel compatible processor. The screen image is constructed from various user-specified objects residing in the VSDD memory (mapped into the processor's address space). Pixels are taken directly from the memory for display on the screen. Characters are constructed employing user-defined RAM-based character generators. The VSDD takes the object data from its memory, buffers it, and runs it through a color palette and D/A converters to produce a video signal. The VSDD also supports overlapped objects and transparent pixels.

In conjunction with appropriate software, the VSDD can be compatible with such video standards as NAPLPS, CEPT or custom configurations. Its multi-window features and resolution make the VSDD ideal for:

- Home Information Systems, TV's, VCR's, Games and Home Computers
- Alphanumeric Color/Monochrome Terminals
- Real-Time Process Control Monitoring Equipment
- Videotex Terminals of the Alphageometric, Alphanumeric and Alphaphotographic Type
- Automotive Displays
- Medical Electronics

Figure 1 shows the block diagram of the VSDD.

FUNCTIONAL DESCRIPTION

Bus Interface Unit (BIU): BIU is the interface between the CPU and the VSDD. CPU accesses the DRAM through the BIU.

Memory Interface Unit (MIU): It is the interface between the VSDD and the DRAM. MIU generates the control signals and the row and column addresses for DRAM.

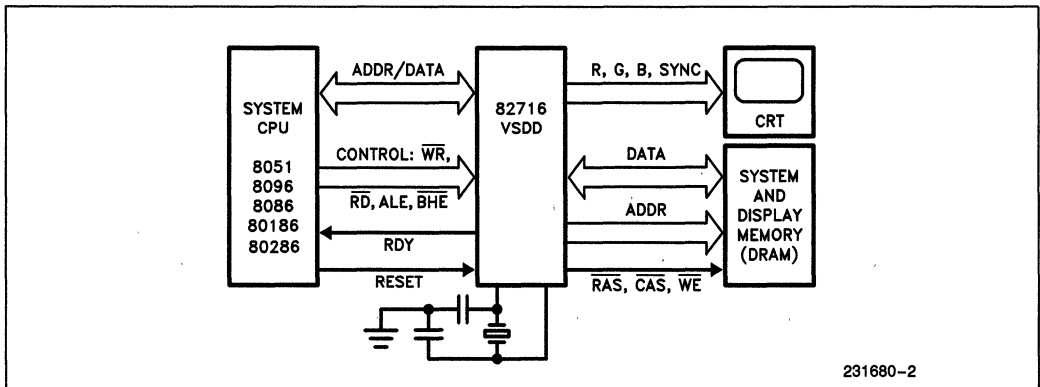
Timing Unit: It consists of oscillator and clock generators. The Video and internal clocks are generated by timing unit.

Sync Generator: The sync generator controls the horizontal and vertical timings for raster generation (HSYNC and VSYNC).

Pixel Unit: The pixel unit contains pixel formatting unit as well as scan line buffers in which display information is placed for each scan line. It also contains the color lookup table (color palette) and D/A converters (DACs). DACs convert the digital color specifications to analog RGB signals for the monitor.

Task Scheduler: This unit is the control circuit of the VSDD. It provides the control signals for internal logic.

Address Computing Unit: It computes the DRAM addresses.



231680-2

Figure 2. Simple System Configuration

SYSTEM OPERATION

The VSDD has 3 primary external interfaces: the CPU interface, the dynamic RAM display memory interface and the video pixel output.

The video subsystem looks like a memory to the CPU. All communication with the video subsystem occurs via that memory. The CPU develops display objects in memory and the VSDD constructs the actual video signal for the display from that memory. The CPU accesses the DRAM via the VSDD's BIU interface. The DRAM contains register segments and display information. CPU access of the dynamic RAM is controlled exclusively by the VSDD's DRAM controller.

The VSDD supports the simultaneous display of information from several sources. Each of these sources is an "object" and is assigned a display window within the VSDD screen. The VSDD can display up to 16 different objects. The size of each object can vary from a few pixels to larger than the full screen. The VSDD forms a scan line by gathering object information into one of the two internal line buffers. While one buffer is being updated with the next scan line, the other buffer is being read out to the color look-up table for display.

An object is defined as a list of pixels or string of characters within the VSDD DRAM memory. Each object is described by an entry, in the Object Descriptor Table (ODT) that contains positional information, color, size and various other attributes. The effective X-Y coordinates of an object can be changed at any time, without touching the object itself, thus allowing independent object animation as shown in Figure 3.

An object can be replaced by another object by changing the pointer in the ODT, allowing the possibility of many more objects in memory than on display at any one time.

Microprocessor Interface

The VSDD supports both 8 and 16 bit microprocessors and microcontrollers from all Intel compatible families. It uses a multiplexed data/address bus.

The VSDD accepts Read (\overline{RD}), Write (\overline{WR}), Address Latch Enable (ALE) and multiplexed Address and Data Bus (AD0-AD15) input signals as well as the Address 16 (A16) input. For 16 bit accesses the Byte High Enable (\overline{BHE}) input is also used. This allows the VSDD to distinguish between 16 and 8 bit ac-

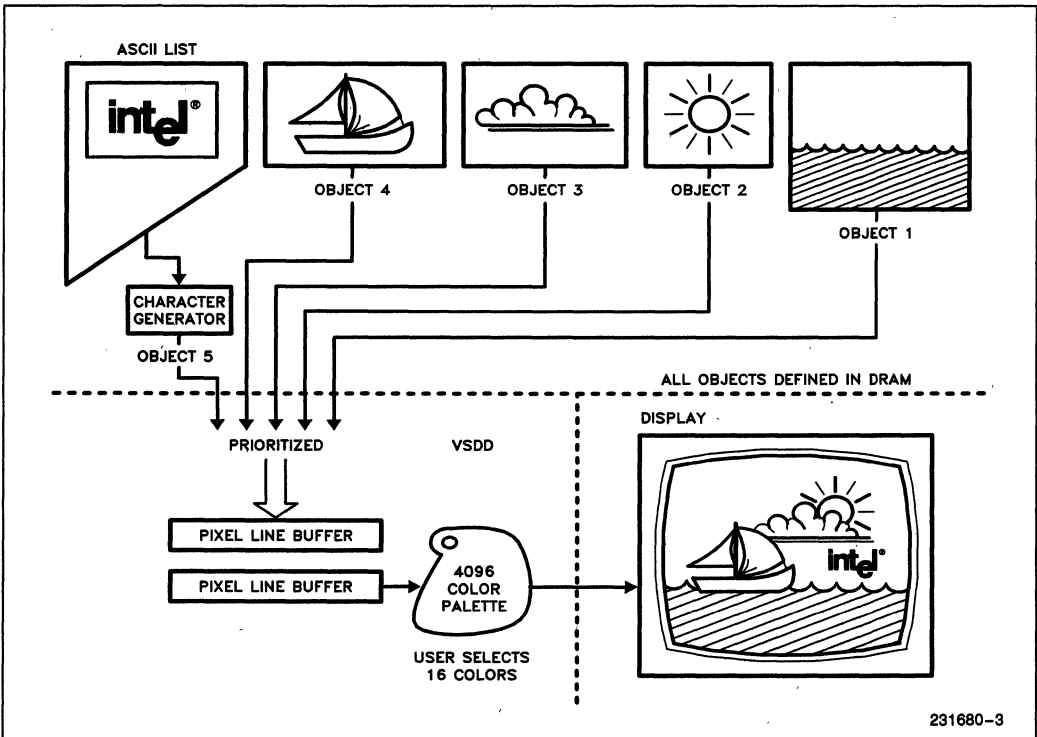


Figure 3. Building an Animation Scene

cesses. If the VSDD cannot service the processor request immediately, then it generates a ready signal (RDY) to extend the processor cycle. The VSDD allows the processor to access up to 512 Kbytes of display DRAM via memory mapping. CPU accesses DRAM with a 16 bit address plus a chip-select input A16 (maximum of 64 Kbytes of address space). A16 behaves like other address inputs. It should be active low.

Arbitration of display memory access is carried out internally by the VSDD. The processor normally has priority over the VSDD Display Logic. Accesses made by the CPU through the VSDD to the display memory can impact VSDD's scan line building process.

When construction of a scan line is complete, the VSDD enters an idling state to wait till the previously constructed line is displayed. If display of the previously constructed line ends before construction of the new line is complete, the remainder of the line building algorithm is aborted. The VSDD's Construction Time Overflow (CTO) signal is activated to indicate this condition to the CPU. This can happen when there are more objects on the line than the VSDD has time to process or when CPU generated accesses to DRAM take up too much of the VSDD's time. To avoid this the VSDD can be programmed to allow only a certain (programmable) number of CPU accesses to the DRAM during line construction.

After each frame the user is able to specify the number of high priority accesses ($n = 0$ to 15) that the system processor may have during each line building process. Thus, n accesses from the system processor will be serviced with minimum delay concurrently with line buffer building. The $(1 + n)$ th access will be delayed via wait-states (RDY) until completion of the line buffer. Whenever the VSDD isn't constructing a horizontal line, system processor accesses will be serviced with minimum delay.

For the MCS-51 family the interface is slightly different. This family has no RDY input and cannot be temporarily halted during a memory access. In this case the RDY output is programmed as a "Free Access" indicator. The 8051 can test this bit to see if the VSDD is using the memory and, if not, can gain access immediately. Because the 8051 has no RDY input, all read operations on the VSDD memory must be pipelined. In this mode a single read access to the DRAM requires two CPU read cycles. The first one is to address the desired DRAM location, but will not return data from that location. The second read cycle can be to any DRAM location, but will return the data that was addressed in the first cycle. Less overhead is required for a series of reads: The

first read cycle returns random data, but after that each read cycle returns the data that was addressed in the previous cycle.

Video Section

The VSDD receives raw data from its display memory and performs all necessary conversions and manipulations to convert the display data to RGB signals. Two line buffers are implemented in on-chip dynamic RAM to store data from two complete scan lines. While one scan line is being displayed, the other buffer is being filled with the data for the next line.

The line buffer has the capacity to hold, at the user's selection, up to 640 pixels at 4 bits/pixel or up to 320 pixels at 8 bits/pixel.

4 bits/pixel are chosen if the display requires more than 320 pixels/line. This is called the High Resolution mode. This mode is selected by setting the HRS (High Resolution Screen) bit to 1.

The on-chip color look-up table [CLUT] contains 16 color entries defined by 12 bits (4-green, 4-red, 4-blue) for a possible palette of 4096 colors. The RGB signals are generated by 3 internal DACs (Digital-to-Analog Converter) whose inputs are the 12 bits (4/color) from the color look-up table. The actual data for color palette is stored in VSDD DRAM. The color palette in external DRAM consists of 16 entries. Each entry is 16 bits long with the lowest 4 bits specifying the address of the entry in the CLUT and the upper 12 bits specifying the color as shown in Figure 4. Four bit pixel codes are used to address the CLUT. The pixel code is matched with the lowest 4 bits of the CLUT entry and the pixel is given the color specified by the upper 12 bits. The color corresponding to the address 0010B is reserved for the background. At the end of every frame, VSDD accesses this data to load the on-chip color look-up table. The loading possibility at every frame allows the user to make real time changes in the color palette.

In some applications it is necessary to overlay external video signals. To support this the VSDD has an Overlay output pin "OVR" which can be used as a fast switch signal to allow display of external video instead of the VSDD output. The OVR pin is controlled by the outputs of the color look-up table. Whenever the color being displayed is RGB = 111H (0001 0001 0001B), the DAC driving the OVR pin goes to 'white' level [OFH or 1111B]. Any other color will cause the OVR pin to go to "black" level (00H or 0000B).

In a system where VSDD generated video will overlay video from an external video source, the "background" palette location 0010B would typically be programmed with 111H. Then, whenever the background color is displayed, user-supplied logic will switch in the external video source.

The overlay function is not available when the on-chip D/A converters are bypassed.

A digital mode is also available. In this mode the RGB and "OVR" pins provide direct digital outputs from the pixel buffer bypassing the internal color table and DACs. Up to 256 colors can be obtained in this mode using 8 bits/pixel with external color table

and DACs. In 8 bits/pixel mode the data is available in two 4-bit nibbles. Low nibble always precedes the high nibble. VSDD provides CKIO signal to latch low and high nibbles using off-chip decoders as shown in Figure 4. Digital mode is also available with 4 bits/pixel.

Memory Mapping

The VSDD can support up to 512 Kbytes of DRAM. The DRAM is organized as 256K words of 16 bits by the VSDD for its own accesses. The VSDD allows CPU to access up to 512 Kbytes of DRAM via mem-

On Chip Color Look Up Table (CLUT)

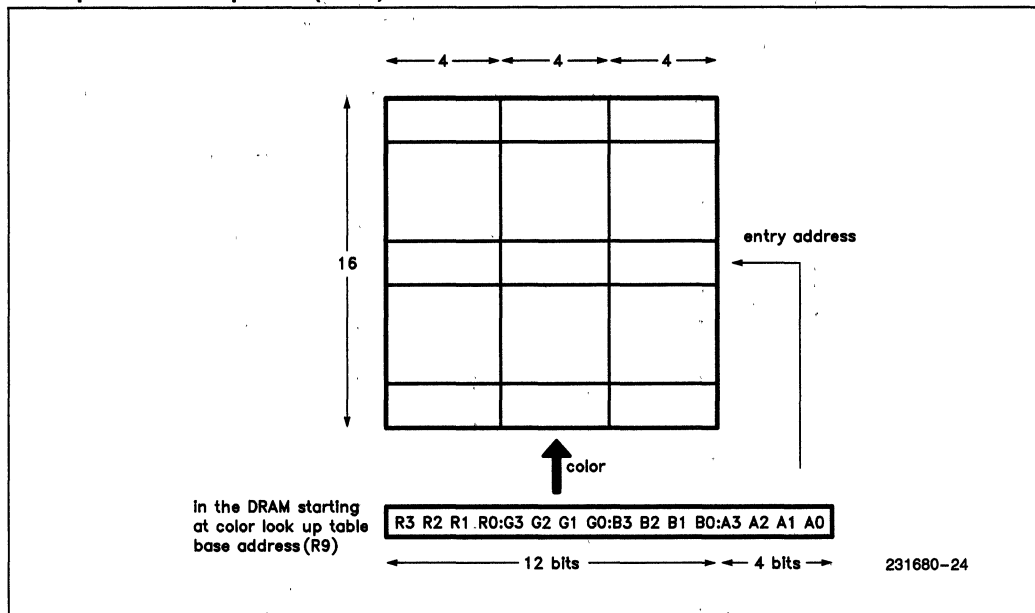


Figure 4. Filling the CLUT

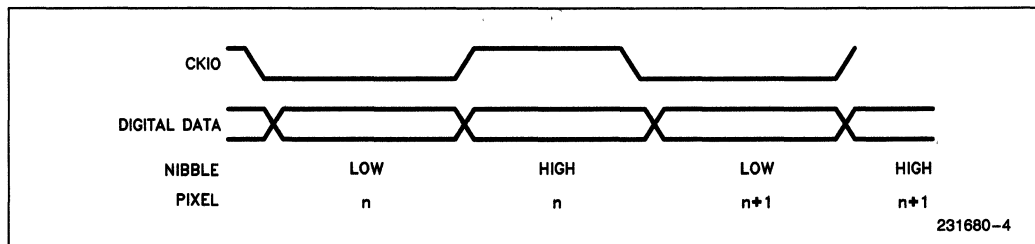


Figure 5. Digital Output Data, 8 Bits/Pixel (Hrs = 0)

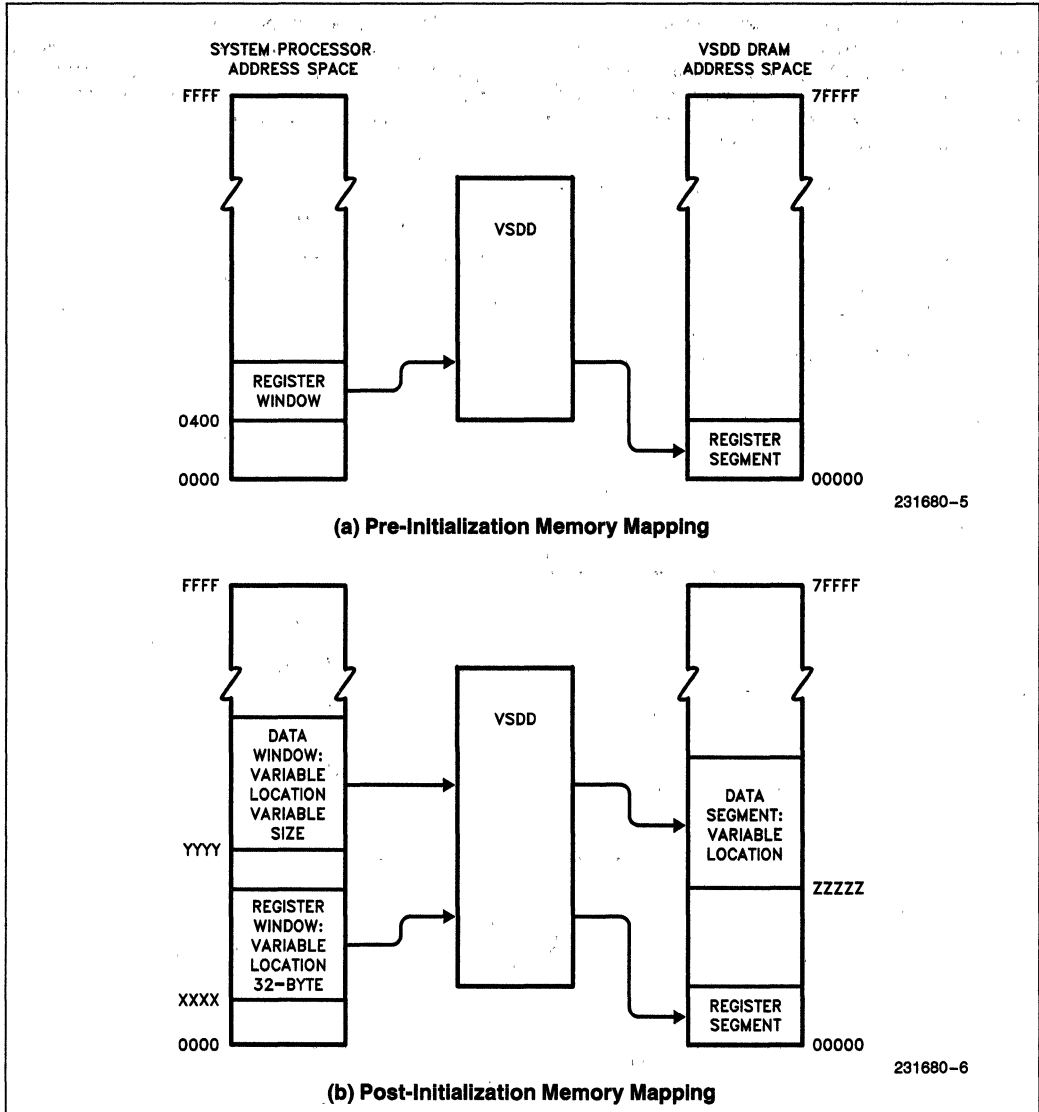


Figure 6. VSDD Memory Mapping

ory mapping. The DRAM is organized as 4 banks of 64K x 16 words. Even byte-addresses are in the lower half of a word and odd byte-addresses are in the upper half.

The VSDD provides two logical windows to map portions of the processor address space into portions of the VSDD-DRAM address space. In the CPU address space, these windows are referred to as the Data Window and the Register Window. In the VSDD DRAM address space, they are referred to as the Data Segment and Register Segment. Thus the Data Window maps onto the Data Segment and the Register Window onto the Register Segment. The Windows are relocatable anywhere in the processor address space. While the Data Segment is relocatable within the VSDD DRAM address space, the Register Segment (32 bytes long) is fixed at VSDD

DRAM starting location 00000H. The length of the Data Window/Segment can be specified from 4K to 64 Kbytes (Figure 6).

REGISTER SEGMENT

The register segment is the first 16 words (32 bytes) of VSDD DRAM. These registers contain the basic information for screen constants, DRAM organization, timing and base addresses. These registers have hardware counterparts on the VSDD. At the end of each frame, VSDD reads register R0 on to an internal register on the chip. If bit UCF (Update Control Flag) in R0 is set, the other registers will also be written on to the chip. These 16 registers are organized in DRAM as shown in Table 1.

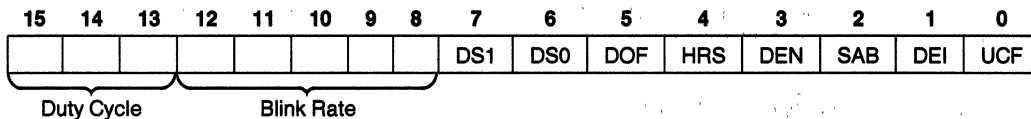
Table 1. Register Window Organization

			VSDD Byte Loc
R15	Horiz. Constant 3	Vert. Constant 3	1EH
R14	Horiz. Constant 2	Vert. Constant 2	1CH
R13	Horiz. Constant 1	Vert. Constant 1	1AH
R12	Horiz. Constant 0	Vert. Constant 0	18H
R11	Access Table Base Address Counter (ATBAC)		16H
R10	Char Base Address 0 and 1		14H
R9	Color Table Base Address (CTBA)		12H
R8	Access Table Base Address (ATBA)		10H
R7	Object Descriptor Table Base Address (ODTBA)		0EH
R6	Priority Access Quantity (PAQ)		0CH
R5	Data Segment Base Address (DSBA)		0AH
R4	Data Window/Segment Length Mask (DWSLM)		08H
R3	Data Window Base Address (DWBA)		06H
R2	Register Window Base Address (RWBA)		04H
R1	Video Configuration Register 1 (VCR1)		02H
R0	Video Configuration Register 0 (VCR0)		00H

NOTE:

Where zeroes are shown in register locations, 0 must be written to those bits in order to ensure proper operation and upward compatibility with any future versions of this device.

R0: Video Configuration Register 0



Bit(s)	Description																								
UCF	Update Control Flag— If set (1), all the registers will be used to update the VSDD at the end of each frame. If not (0), only ATBA and VCR0 will be updated.																								
DEI	Digitally Encoded Color Information— If set (1), RGB and OVR outputs are digital. If not (0), RGB and OVR are analog.																								
SAB	Slow Access Bit— If set (1), then slow DRAM (page cycle time = 210 ns) can be used. If not (0), fast DRAM (page cycle time = 140 ns) can be used.																								
DEN	Display Enable Flag— If set (1), the VSDD display is enabled. If not (0), the VSDD display is disabled.																								
HRS	High Resolution Screen— If set (1), the maximum horizontal resolution is 640 pixels. If not (0), the resolution is 320 pixels.																								
Blink Rate	Blink rate of selected objects is set from 8 frames to 256 frames in multiples of 8 frames. For 50 Hz/60 Hz, this translates into blink rate increments of 160 ms/133 ms starting from 6.2 Hz/7.5 Hz (code 00000) down to 0.20 Hz/0.23 Hz (code 11111).																								
Duty Cycle	The duty cycle of the blink rate can be selected as below: <table style="margin-left: 40px; border: none;"> <tr> <td>111</td> <td>Always On</td> <td></td> </tr> <tr> <td>110</td> <td>12.5% Off</td> <td>87.5% On</td> </tr> <tr> <td>101</td> <td>25.0% Off</td> <td>75.0% On</td> </tr> <tr> <td>100</td> <td>37.5% Off</td> <td>62.5% On</td> </tr> <tr> <td>011</td> <td>50.0% Off</td> <td>50.0% On</td> </tr> <tr> <td>010</td> <td>62.5% Off</td> <td>37.5% On</td> </tr> <tr> <td>001</td> <td>75.0% Off</td> <td>25.0% On</td> </tr> <tr> <td>000</td> <td>87.5% Off</td> <td>12.5% On</td> </tr> </table>	111	Always On		110	12.5% Off	87.5% On	101	25.0% Off	75.0% On	100	37.5% Off	62.5% On	011	50.0% Off	50.0% On	010	62.5% Off	37.5% On	001	75.0% Off	25.0% On	000	87.5% Off	12.5% On
111	Always On																								
110	12.5% Off	87.5% On																							
101	25.0% Off	75.0% On																							
100	37.5% Off	62.5% On																							
011	50.0% Off	50.0% On																							
010	62.5% Off	37.5% On																							
001	75.0% Off	25.0% On																							
000	87.5% Off	12.5% On																							

DS1 and DS0 indicate the array size (16K, 64K or 256K) of the DRAM used to implement the display memory. DOF (DRAM organization flag) is used to indicate if the DRAM is bit-wide (DOF = 0) or nibble-wide (DOF = 1). See Table 2.

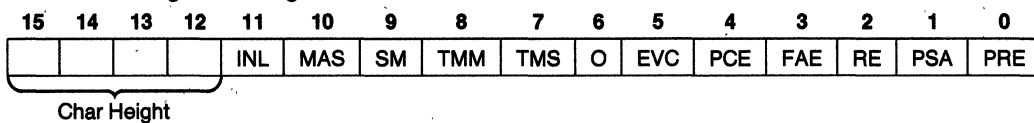
Table 2

DS1	DS0	DOF	Dram Configuration	Maximum Capacity	ADDR Pins Used		
					Row	Col	Bank Select
0	0	0	16K x 1	32 Kbytes	0-6	0-6	(None)
0	0	1	16K x 4	128 Kbytes	0-7	0-5	6, 7 at $\overline{\text{CAS}}$
0	1	0	64K x 1	128 Kbytes	0-7	0-7	(None)
0	1	1	64K x 4	512 Kbytes	0-7	0-7	8 at $\overline{\text{RAS}}$, $\overline{\text{CAS}}$
1	0	0	256K x 1	512 Kbytes	0-8	0-8	(None)

NOTE:

For 16K x 4, 2 bank select bits are emitted on address pin 6 and 7 when $\overline{\text{CAS}}$ goes low. By using external 2-to-4 decoders up to 4 banks can be selected. For 64K x 4, two bank select bits come out on address pin 8: one with the row address (MSB) and one with the column address (LSB).

R1: Video Configuration Register 1



Bit	Description
PRE	Pipeline Read Enable—If set (1), enables the pipeline read mode: CPU read cycles always return data from the previous read cycle. If not (0), then accesses are not pipelined.
PSA	Pre Scaler Active—This bit defines the relationship between the video clock frequency and the sync generator clock frequency. (Figure 7). GCLK is used for programming horizontal timings. If the video clock exceeds 16 MHz, the PSA bit should be set (1).
RE	This flag, when set (1), enables the CPU to read data from the display memory through 82716. If not (0), the output buffers of the VSDD are disabled, thus preventing CPU from reading the DRAM.
FAE	Free Access Enable—Enables the RDY pin to act as a free access indicator to the processor, if set (1).
PCE	Priority Counter Enable—If set (1), enables the VSDD to limit the number of CPU access to DRAM. Only valid with processors that have wait states.
EVC	External Video Clock—If set (1), it enables the CKIO pin to be used as input for a video clock up to 25 MHz. If not set, CKIO is a buffered clock output. (Figure 6)
TMS	Twin Mode Slave—Used for twin mode. If set (1), it specifies the VSDD as a slave, displaying only the even lines.
TMM	Twin Mode Master—In twin mode if set (1), it specifies the VSDD as a master, displaying only the odd lines and supplying sync to slave. The combination TMM = 0, TMS = 0 means twin mode operation is not in use. TMM = 1, TMS = 1 is illegal.
SM	Sync Mode—If set (1), enables the HSYNC pin to operate in composite sync mode. Otherwise HSYNC outputs horizontal sync.
MAS	Master—If set low, the VSDD accepts external synchronization signals and locks to it via an on-chip PLL circuit.*
INL	Interlace—If set (1), selects interlaced mode. If not (0), selects non-interlaced video.
Char Height	These 4 bits encode the number of scan lines per character. The number is encoded as a simple unsigned binary integer, except that 0000 means 16.

* If set high (1), HSYNC and VSYNC are outputs.

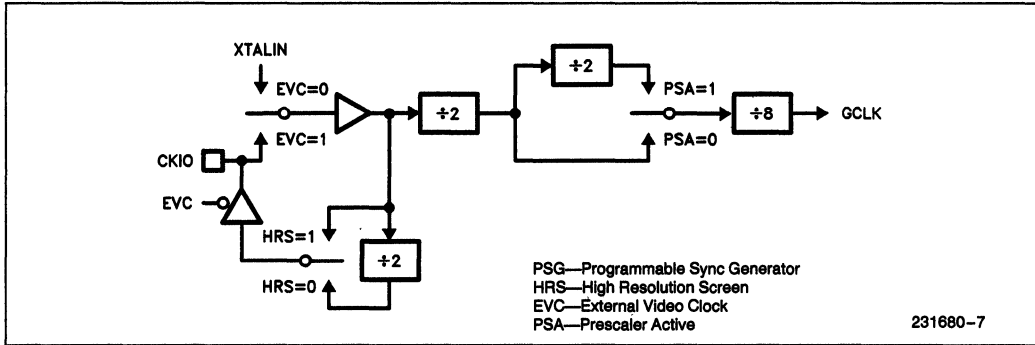


Figure 7. PSG Clock Generator

R2: Register Window Base Address (RWBA)

Register Window Base Address: R16-R5							0	TF2	TF1	ME
--------------------------------------	--	--	--	--	--	--	---	-----	-----	----

ME—Margin Enable: When set (1), a margin (in background color) can be added in a standard TV mode.

TF2,TF1 (Test Flags): If DEI = 1, then these flags determine what type of digital output is emitted. The output options are summarized in the table below:

DEI	TF2	TF1	Outputs	Signals
0	X	X	Analog	RGBO
1	0	0	Digital	Reds Only*
1	0	1	Digital	Greens Only*
1	1	0	Digital	Blues Only*
1	1	1	Digital	Pixel Code

NOTE:

*These three combinations can be used to test the on-chip color look-up table. The chosen combination selects one of the color components to be output via the DV3-DV0 outputs. The DEI bit has to be set to 1 to switch off the DACs.

R16-R5 specify the Register Window base address. This is the window (mapped into the CPU's address space) through which the CPU accesses the Register Segment of the DRAM. This register may be placed on 32 byte boundaries.

R3: Data Window Base Address (DWBA)

Data Window W16–W12	0	Screen Boundary SB9–SB3	0	0	0
---------------------	---	-------------------------	---	---	---

SB9–SB3 Screen Boundary bits specify the upper 7 bits of the 10-bit x coordinate of the right edge of the screen. VSDD would process pixels up to x = to this number with lower 3 bits taken to be 1s. No pixels will be processed which are to the right of the screen boundary.

W16–W12 bits specify the Data Window Base Address. This is the window through which the CPU accesses the Data Segment of the DRAM.

R4: Data Window/Segment Length Mask (DWSLM)

L16	L15	L14	L13	L12	0	0	0	0	0	0	0	0	0	0	0
-----	-----	-----	-----	-----	---	---	---	---	---	---	---	---	---	---	---

L16–L12 bits are the Data Window Length Mask, which specify the length of the Data Window in bytes, as follows:

L16	L15	L14	L13	L12	Data Window Length
1	1	1	1	1	4 Kbytes
1	1	1	1	0	8 Kbytes
1	1	1	0	0	16 Kbytes
1	1	0	0	0	32 Kbytes
1	0	0	0	0	64 Kbytes

R5: Data Segment Base Address (DSBA)

Data Segment S16–S12	0	0	BS0	BS1	0	0	0	0	0	0	0
----------------------	---	---	-----	-----	---	---	---	---	---	---	---

BS1 BS0 divide the 512 Kbyte-address space into four banks of 128 Kbyte each. Note however that only bitmapped object data can reside in banks 1 through 3. All other display data such as character generators, register segment, access table etc. must be written to bank 0.

S16–S12 bits specify the Data Segment base address in the VSDD’s address space. The display data is stored in the data segment. The data must be placed on the boundaries corresponding to the size specified in the data window length mask (see DWSLM).

R6: Priority Access Quantity (PAQ)

0	0	0	0	0	0	0	0	0	0	0	0	← PA Quantity →
---	---	---	---	---	---	---	---	---	---	---	---	-----------------

PAQ 4 bits indicate the maximum number of CPU accesses to the DRAM that are allowed during building of each scan line, if PCE bit (in R1) is 1.

R7: Object Descriptor Table Base Address (ODTBA)

Object Descriptor Table Base: A15–A6	0	0	0	0	0	0
--------------------------------------	---	---	---	---	---	---

This register contains the word base address of the object descriptor table in the VSDD’s address space. It is accessed by the VSDD at the end of each frame. This table must reside in bank 0.

R8: Access Table Base Address (ATBA)

Access Table Base Address: B15–B0

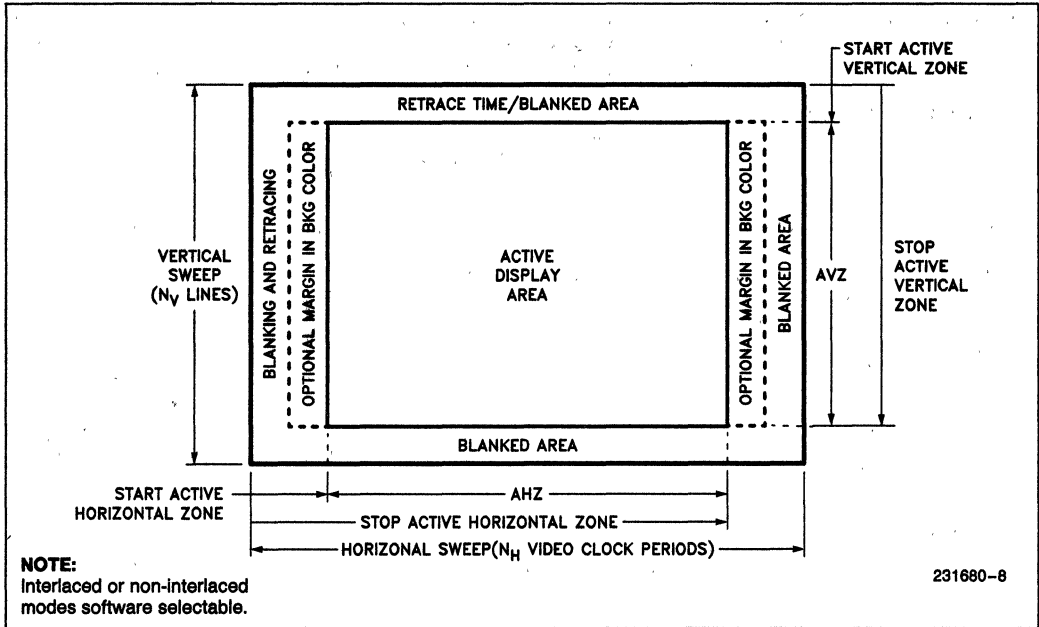


Figure 8. Programmable Raster Parameters

ACCESS TABLE

The Access Table contains the vertical positioning information for each object. The Table begins at the location designated by the Access Table Base Address register, R8 in the Register Bank. The Table contains one word in DRAM for each scan line in the Active Vertical Zone of the display.

The first line (at the top of the display) is associated with the word at the Table's base address. Within each word, bit number *i* is the Access Flag associated with object number *i* in the display and has the priority *i*. Object number 4 has a lower priority than object number 5 (see Figure 9).

b0 is the access flag for object 0, b1 for object 1, etc.

The Access Flags indicate to the VSDD which objects are to be present on which lines of the display. If an Access Flag is set (1), then there is to be no change in the object's display status; that is, if the object did not appear on the previous line, it will also not appear on this line. If the object's Access Flag is clear (0), the object's display status is reversed from what it was in the previous line. All objects are disabled at the end of the Active Vertical Zone.

An object is activated by putting a zero in the word corresponding to the scan line on which the object is first displayed. This turns on the object for all following scan lines. The object is toggled off by putting a zero in the scan line following the last line of the object.

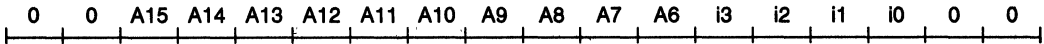
At the beginning of each frame, the VSDD writes the contents of Access Table base address, R8, into Access Table Counter, R11. At the end of each line this counter is read into the Access Table Entry Address Register (on the VSDD). This entry address is then used to read the access flags for the line that is to be constructed. Access Table Entry Address is then incremented and written back into R11 preparing it for the next line.

Then the Access Flag Register is examined bit by bit to determine if there is a change in any object's display status. If object number *i* is to be displayed, then its Object Descriptor field is read. The base address for the Object Descriptor field for object number *i* is constructed from the Object Descriptor Table Base Address register, R7, by concatenating bits A15 through A6 from R7 with 4 bits representing the number *i* (*i* = 0 to 15).

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----

Figure 9. Access Flag Register

Then the Object Descriptor field base Address is:



An Object Descriptor field is 4 words long, and all 4 words are read, so the last two bits in the above address are incremented to get all 4 words. Access Table and Object Descriptor Table must reside in bank 0.

Different Access Tables may be defined at the same time but only one is activated during any one frame. The Access Table allows easy vertical scrolling of an object. If the object scrolls down, truncating takes place by simply moving the window down and truncating the object when it moves off the screen. Moving up the screen is similar except when the object moves past the top of the screen, the object base address must be incremented to point to the start of the next displayed line.

OBJECT DESCRIPTOR TABLE

The Object Descriptor Table (ODT) contains a 4-word Object Descriptor field for each object in the display. This field describes the base address, attributes, and X-position of each object. This information is initialized as well as updated by the system processor. The 16 available object descriptors (128 bytes) are located contiguously in the ODT, starting at the location specified by the ODTBA register. The ODT is located in the VSDD DRAM.

There are two types of objects: bitmapped and character. Their descriptor fields are as shown below.

Bitmap Descriptor Field:

N: Current Object Entry Address is the address of the pixel data for the next scan line for the object. At the beginning of each frame, the VSDD copies Object Base Address into this field. This is maintained by the VSDD and should not be altered by the CPU.

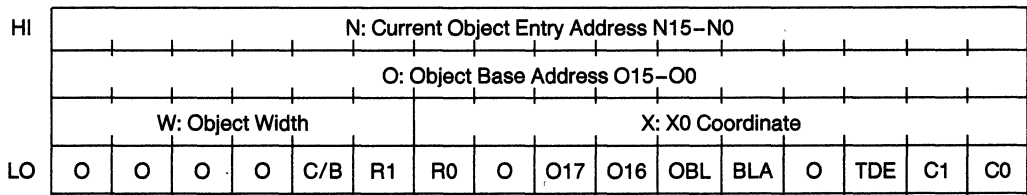
O: Object Base Address points to the beginning of the object's data base. This is a 18 bit address. Only low 16 bits are specified here. O17 and O16 are specified elsewhere in the descriptor field.

W: Object Width indicates how wide the object is in "64 bit words". The width of the object must be a multiple of four 16-bit words. 000001 specifies a width of 1 "64 bit word", 111111 a width of 63 "64 bit words".

X: X0 coordinate is a 10-bit signed number (2's complement) encoding the horizontal position of the left-most pixel in the object. X0 can be -512 to +511.

C/B: Character/Bitmap Object Specifier. C/B = 1 indicates a character object. C/B = 0 indicates a bitmap object.

Bitmap Descriptor Field:



R1, R0: Resolution: For a bitmap object, these 2 bits specify how many bits each pixel takes up in VSDD DRAM, as shown in the table below. HRS is a bit in R0.

Bitmapped Objects (C/B = 0)

HRS	R1	R0	Bits/Pixel
0	0	0	Do Not Use
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	Do Not Use
1	0	1	Do Not Use
1	1	0	2
1	1	1	4

O17, O16: Two highest bits of the object base address.

OBL: Object Blinker = 1 causes the object to blink between foreground and background color. The blink rate and duty cycle are specified in R0 in the Register Segment.

BLA: Blanker = 1 turns the object off i.e. the object is not displayed.

TDE: Transparency Detect Enable. If TDE = 1, then pixels that are encoded as all 0's are not written into the Line Buffers. The buffers will retain the previous pixel data. Thus a low priority object will be visible through the transparent pixels of a higher priority object. When TDE = 0, then pixels that are encoded as all 0's are written into the line buffer. "0000B" is then one of 16 color codes.

C1 C0: Default Color Specification. For bitmapped objects that are stored in external VSDD memory in the 2 bits/pixel mode, these two bits extend the pixel specification to 4 bits.

Character Descriptor Field:

Z: Slice Number contains a character object's slice number for the next scan line. It is reloaded by the VSDD once per frame with the slice number (YS3-YS0).

N: For character object, it's the beginning address of the current line of text. The entry address is formed of O15-O12 and N11-N0. Hence, a character object may not extend across a 4K word boundary. Two highest bits—O17, O16—are zero for character objects. Between each frame, lowest 12 bits of object base address are written into N.

Y: Start Slice Number is the first (topmost) character slice of this object. The CPU can modify this field to produce a scrolling effect in the display of the text. Y = 0 is the bottom of the character and Y = Character height (defined in R1) is the top.

R1 R0: Resolution: For character objects, R1 and R0 specify the width of the character, as shown in the table below.

Character Objects (C/B = 1)

HRS	R1	R0	Pixels/Char
0	0	0	6
0	0	1	8
0	1	0	12
0	1	1	16
1	0	0	16
1	0	1	6
1	1	0	8
1	1	1	12

Character Descriptor Field:

HI	Z: Slice No.		N: Current Object Entry Address N11-N0											
	O: Object Base Address O15-O0													
LO	W: Object Width						X: X0 Coordinate							
	Y: Slice No.	C/B	R1	R0	CRS	PSE	FAD	OBL	BLA	HCR	TDE	C1	C0	

FAD: Full Attribute Definition = 1 means character descriptions are 3 bytes long. Each character is encoded as an ASCII byte plus a 2-byte attribute word. FAD = 0 means character descriptions are 1 byte long.

CRS: Conceal/Reveal/Select. If FAD = 1, then CRS = 1 enables the MSK bit in the character's attribute word to cause the character to be concealed. If FAD = 0, then CRS selects one of two character generators. CRS = 0 selects CGBA0 as specified in the register segment. CRS = 1 selects CGBA1.

PSE: Proportional Spacing = 1 enables proportional spacing of characters.

HCR: High Color Resolution. If FAD = 1, then HCR = 1 means use 16-color palette for characters and their backgrounds. If FAD = 0, then HCR = 0 means use 8-color palette (see Attribute definition). If FAD = 0, then HCR should be 0.

C1 C0: For character objects that are stored in external memory in the 1 byte/character mode, these two bits become the MSBs of the foreground/background colors of the characters as shown below.

Foreground color = C1 C0 0 1
Background color = C1 C0 0 0

OBJECT DATA

Objects are rectangular windows on the screen. Object data begins at the Object Base Address specified in the "O" field of the Descriptor table. The length of the data file depends on the object's height, width and resolution. The width of the object

is specified in 4-word units by the 'W' field in the Object Descriptor. For example, if the 'W' field contains 001010 then the object is ten 4-word units wide.

The VSDD will read in $10 \times 4 = 40$ words of object data for each scan line in which the object appears. For bit-mapped objects, the beginning address of each block of 40 words is constructed from the 'N' field in the Object Descriptor. The 'N' field is updated for the next scan line after each block of data is read.

For character objects, 'N' field is used to construct the beginning address of a line of ASCII text. The object itself may consist of many lines of text. Each line of text consists of individual scan lines—each scan line presenting one "slice" of the text character. When the final slice of a line of text has been constructed the 'N' field is updated to the next line of text.

The Table 3 shows minimum and maximum width of character and bit-mapped objects.

Table 3

Object Type	Min. Width	Max. Width
Bitmap 2 Bit/Pixel	32 Pixels	2016 Pixels
Bitmap 4 Bit/Pixel	16 Pixels	1008 Pixels
Bitmap 8 Bit/Pixel	8 Pixels	504 Pixels
Character 1 Byte/Char	8 Chars	504 Chars
Character 3 Byte/Char	1 Char ⁽²⁾	168 Chars ⁽¹⁾

NOTES:

1. The last 16 bit-word of the object will not be used.
2. The minimum memory required is actually 4×16 -bit words. The second character can be eliminated by setting its transparent attribute bit.
3. For 3 bytes/character objects, full memory utilization can be obtained if the width of the object is a multiple of 12 words.

For character objects, two formats are defined. The first is a 1 byte/character mode. In this mode 2 ASCII character codes are stored in each DRAM word. The second format uses 3 bytes/character. They are formed as follows:

1 Byte/Character

X15	X0
2nd Character	1st Character
A7:A6:A5:A4:A3:A2:A1:A0	A7:A6:A5:A4:A3:A2:A1:A0

- A7–A0 = Character ASCII Code

3 Bytes/Character

X15	X0
HI	2nd Character Attribute Field
HI	1st Character Attribute Field
LO	A7:A6:A5:A4:A3:A2:A1:A0
LO	A7:A6:A5:A4:A3:A2:A1:A0
LO	2nd Character
LO	1st Character

The attribute field is formatted as follows:

CG	TFG	TBG	DW	MSK	INV	BLI	UND	FC3	FC2	FC1	FC0	BC3	BC2	BC1	BC0
----	-----	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

WHERE:

- A7–A0 8 bit ASCII code or any other 8 bit character code.
- BC2–BC0 Background color.
- BC3 (U/L) When HCR = 0, it specifies the upper or lower half of the character in double height mode. When set (1), it specifies the upper half. If not (0), lower half is specified. When HCR = 1, it is used as the MSB of the background color.
- FC2–FC0 Foreground color.
- FC3 (DH) When HCR = 0, it specifies the character to be double its normal height when set (1). When HCR = 1, it is the MSB of the foreground color. When HCR = 0, and DH = 0, then U/L must be set to 0.
- UND If set (1), the character is underlined.
- BLI Enables the character to alternate between foreground and background color when set (1).
- INV If set (1), the foreground and background colors are reversed.
- MSK If set (1), the character disappears from the screen (when CRS = 1) i.e. foreground color is same as background color. When CRS = 0, MSK attribute is ignored.
- DW If set (1), the character is expanded to double width.
- TBG Sets background transparent, when TBG = 1.
- TFG Sets background transparent, when TFG = 1.
- CG Selects one of two character generators.

Character Generators

The VSDD allows the simultaneous use of two independent character generators of 256 characters each. Bits 15–12 of their base addresses are specified in R10 in the Register Segment. Each character generator must begin on a 4K word-address boundary in memory bank 0.

The address for a character generator consists of four fields: See Figure 10

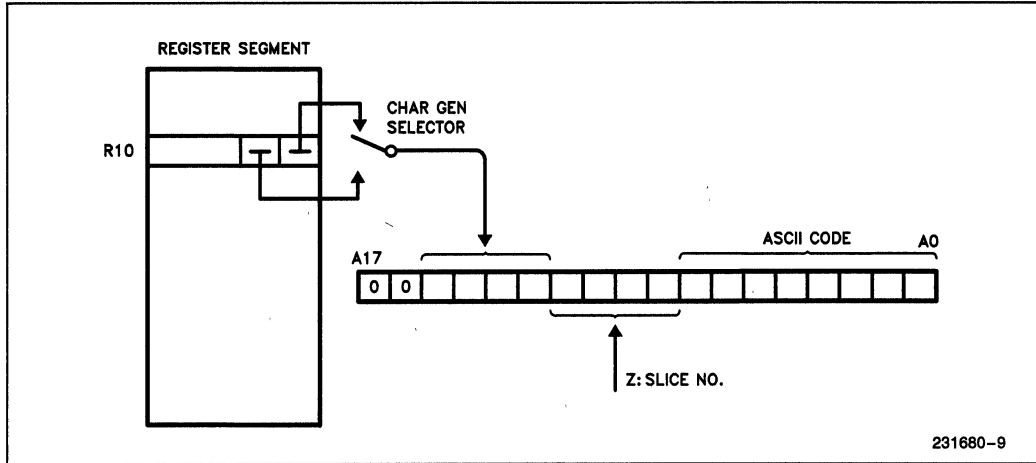


Figure 10

A character set consists of H blocks of 256 words, where H is the character height in scan lines. Each character is divided into H slices with slice zero defined as the bottom of the character and slice H-1 is the top scan line of the character. Character height, globally defined for the whole frame in register R1 can be up to 16.

Each slice occupies one word in DRAM. Within the word, the slice is encoded as a sequence of pixel bits, the leftmost pixel being the LSB in the word. If a pixel bit is 1, then the pixel is to be given foreground color. If a pixel bit is 0, the pixel is given background color.

If the characters are encoded in plain ASCII (FAD = 0), then the character generator is selected by the CRS bit in the Object Descriptor. If the characters are encoded with full attributes (FAD = 0), then the character generator is selected by attribute bit CG.

As the characters are defined in DRAM, a new version of the character generator can be obtained by either:

- modifying the character generator directly or
- updating one set while the other set is being displayed. The set can then be changed by updating the CGBA pointer in the register segment. This method results in an instantaneous change on the screen.

PICTURE CONSTRUCTION

VSD supports, 2, 4 or 8 bits pixels. Up to 640 x 512 pixels can be supported using 2 or 4 bits/pixel. In

the 8 bits/pixel mode, a picture size of 320 x 512 can be supported. In this mode only the lower 4 bits of the byte are used by the color look-up table, the upper four are ignored. In digital mode, using 8 bits/pixel 256 colors can be obtained with external color palette and DACs.

The VSD starts picture construction at the beginning of the frame using the logic flow shown in Figure 11.

At the beginning of each frame, the contents of Access Table Base Address, R8 is copied into R11, Access Table Counter. Simultaneously, the VSD also loads the color look-up table from the DRAM into the on-chip color look-up table. This feature enables the user to select a different set of 16 colors at every frame.

After each scan line, R11 is loaded into an on-chip register, Access Table Entry Address Register by the VSD. The on-chip register (Access Table Entry Address) points to an access table entry for the line that is to be constructed. The VSD reads this entry into an on-chip register called the Access Flag register. (R11 is then incremented by 1 to point to the access table entry for the next scan line.) (Simultaneous to this operation, the VSD fills the line buffer with the specified background color.) Each access table word contains 16 flag bits—one for each object. Access flags determine which objects are present on the line. Object priorities are fixed with object 15 being the highest and object 0 being the lowest.

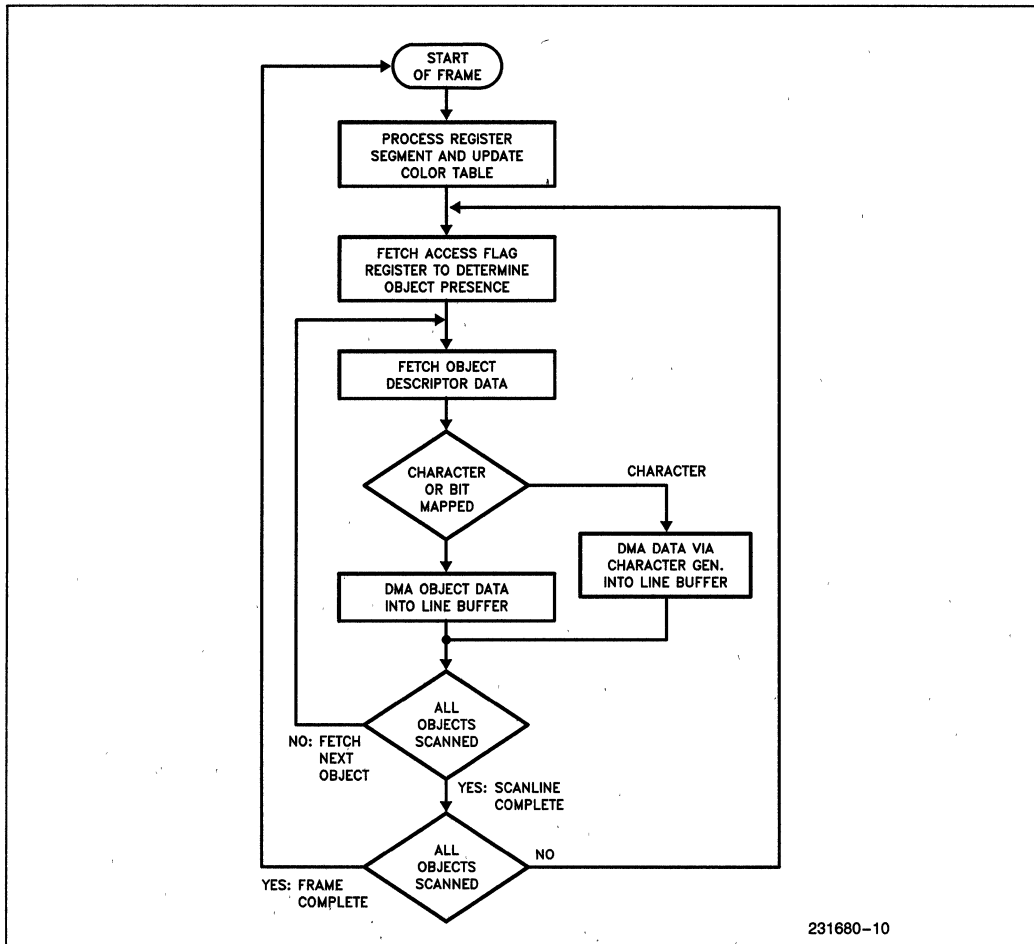


Figure 11. Scan Line Building Process

If an object is present, then its object descriptor data is read from the VSDD DRAM. This determines the object's width, horizontal position, type and where to find the display data for this line of the object. For bit-mapped objects the display data passes directly from the VSDD DRAM into the line buffer.

For character objects, the data passes via a character generator into the line buffer. The appropriate slice of character pixel information is written in the appropriate horizontal position in the line buffer. Both character and bit-mapped data overwrites the background pixels that were previously written into the buffer.

This procedure is repeated for each object that is present on the line. For overlapped objects, the high priority data overwrites the low priority data. Low priority object will be hidden behind the high priority

object. The priority of objects are determined by the order in which they are written into the line buffer. For example, the object number 5 has higher priority than the object number 4. Object number 1 is described in the first Object Descriptor Table entry, object number 2 in the second entry, etc. Transparent Pixels (0000B, when TDE bit is set) are not written into the line buffer. Previous pixel data is retained at the location where transparent pixels are present. Thus a lower priority object can still be visible behind the transparent parts of a higher priority object.

The construction process may result in more pixels being read from the DRAM than are actually displayed on the line. Since only a finite time exists for line construction, it is important that the number of objects and the amount of overlap between the objects be considered when examining display performance.

When construction of each line is complete the VSDD enters an idling state to wait till the previously constructed line finishes being displayed. If display of the previously constructed line ends before construction of the new line is complete, the remainder of the line construction algorithm is aborted, and the VSDD's Construction Time Overflow signal is activated to indicate this condition to the CPU.

Construction time overflow can result when there are more objects on the line than the VSDD has time to process or when CPU-generated accesses to DRAM takes up too much of the VSDD's time.

Figure 12 shows the VSDD and DRAM operation during line building process.

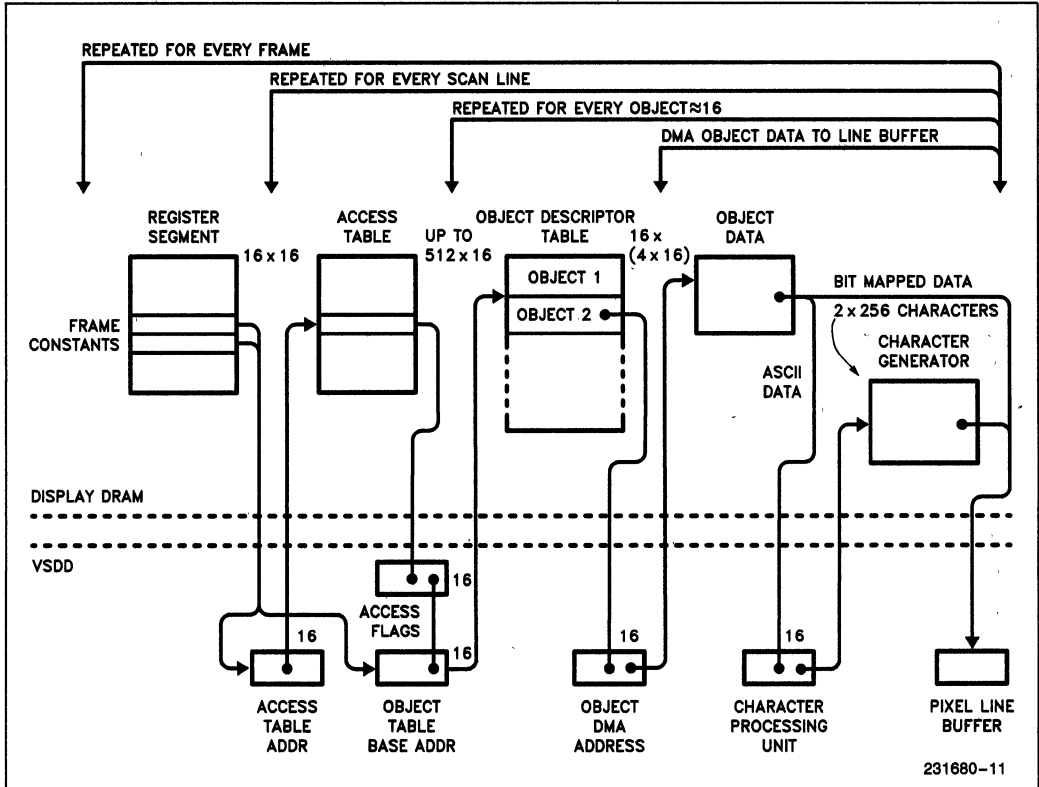


Figure 12. VSDD and DRAM Operation

Movement of objects is accomplished easily in the x direction by changing the value in the object descriptor. Movement in the y direction is accomplished by moving the bits which turn the object on and off within the access table. If the "off" bit falls below the bottom of the access table, the object will automatically be truncated at the bottom of the screen. Moving in an upwards direction requires that the object base address in the descriptor table be changed to truncate the top of the object.

TWIN-MODE OPERATION

For higher performance, it is possible to connect two VSDD chips in parallel. One of them is designated as the master and the other as the slave. The master generates information for the even lines of the display together with all the system timing. The slave accepts the synchronization pulses as inputs and displays the odd lines of the picture. Because each VSDD is essentially constructing half the picture, the scan line construction time is twice as great, allowing higher throughput in terms of information processed and objects displayed.

PERFORMANCE

The number of objects that a VSDD can support on a scan line is dependent upon the screen resolution, refresh rate, DRAM type, and resolution per object. In addition, the percent overlap of each object can affect the performance. Usually the amount of overlap can be kept to a minimum by keeping the object window only as large as necessary.

VIDEOTEX STANDARDS

The VSDD has been designed for these types of application. It can support several Videotex standards from Europe, North America and Japan. Although it has been optimized for alpha-geometric applications such as NAPLPS, GKS, and VDI, it is capable of supporting the existing alpha-mosaic standards and the higher resolution alpha-photographic standards. It supports most of the European CEPT standard that includes PRESTEL and TELETEL by using static character objects. In addition it offers bit-map objects and movement. Alpha-photographic standards such as Picture PRESTEL and Picture TELETEL can be supported in 8 bit pixel mode with the addition of external color translation and higher resolution hardware.

Pin Description

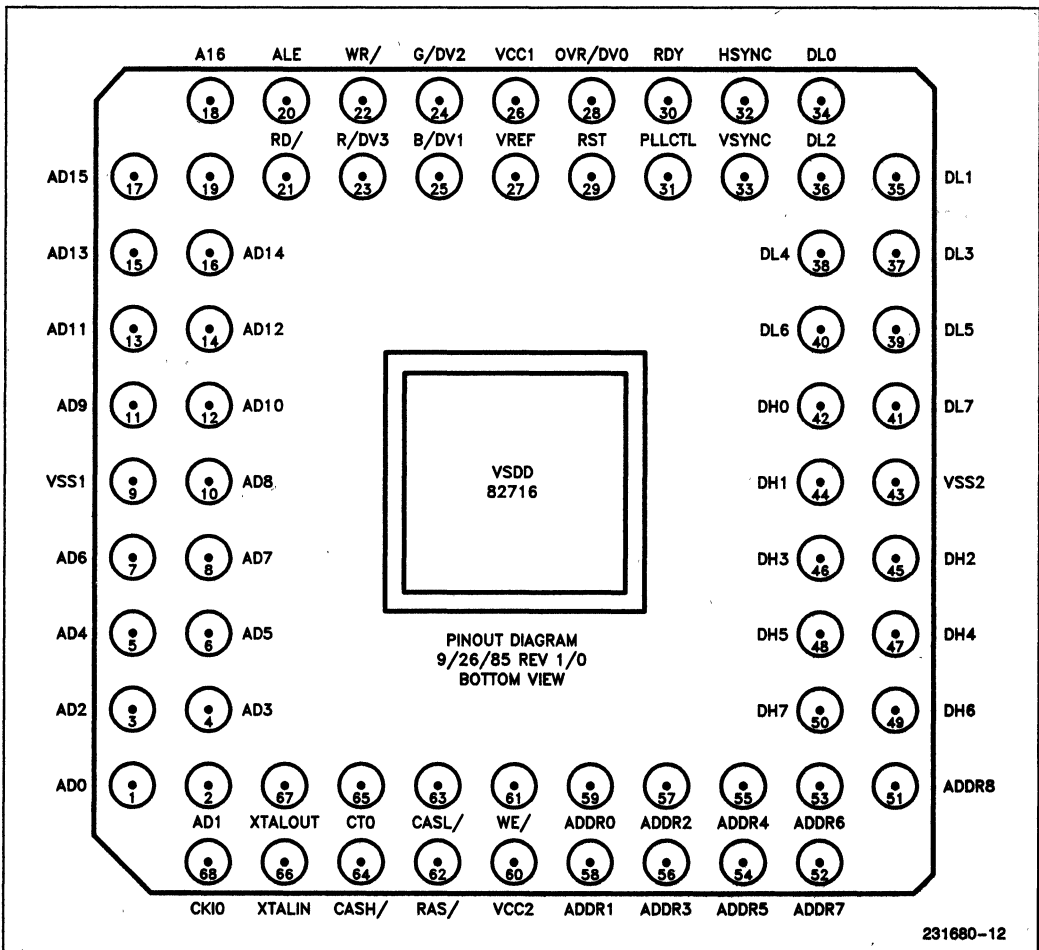
Symbol	Pin	Type	Function
AD0-AD7	1-8	I/O	Processor system bus multiplexed address/data.
AD8-AD15	10-17		
A16	18	I	Programmable chip select.
\overline{BHE}	19	I	Byte high enable.
ALE	20	I	Address latch enable.
\overline{RD}	21	I	Read strobe from processor.
\overline{WR}	22	I	Write strobe from processor.
RESET	29	I	Places the VSDD in initialization mode.
RDY	30	O	Ready-wait state for 86/88/96 and free access indicator for MCS-51.
R/DV3, G/DV2	23,24	O	Red, green, and blue analog outputs or 3 bits of digital output.
B/DV1	25		
OVR/DVO	28	O	Output signal or fourth digital output.
V_{REF}	27	I	Analog voltage reference.
HSYNC	32	I/O	As an output it supplied horizontal or composite sync. As an input it synchronizes the VSDD with an external video signal.
VSYNC	33	I/O	As an output it provides vertical sync. As an input it synchronizes the VSDD to external video.
DL0-DL7	34-41	I/O	Data input/output to DRAM low order byte.
DH0-DH7	42-50	I/O	Data input/output to DRAM high order byte.
ADDR0-ADDR8	59-51	O	DRAM row and column addresses.
\overline{RAS}	62	O	Row address strobe.
CTO	65	O	Construction time overflow.
\overline{CASL}	63	O	Column address strobe low.
CASH	64	O	Column address strobe high.

Pin Description (Continued)

Symbol	Pin	Type	Function
WE	61	O	Write enable.
XTALIN	66	I	Oscillator input or crystal terminal.
XTALOUT	67	O	Oscillator output or crystal terminal.
CKIO	68	O/I	As output it serves as a buffered dot clock. As an input it is used to receive external dot clock.
PLLCTL	31	O	PLL control used to fine tune oscillator
VCC	26, 60		5 volt main supply.
VSS	9, 43		Digital ground.

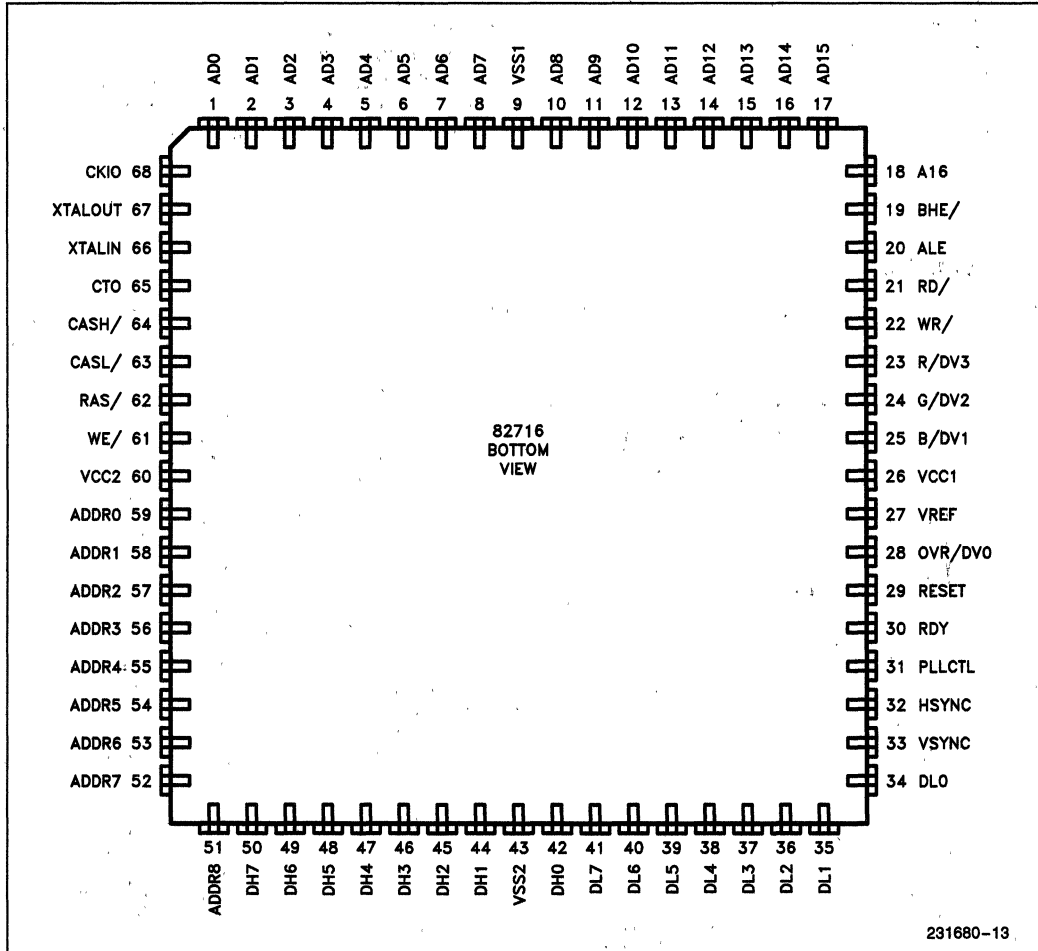
PACKAGING

Pinout for a 68-pin Pin Grid Array Package is shown below.



PLCC PACKAGE

Pinout for a 68-pin plastic, leaded chip carrier is as below:



ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0 to 70°C
 Storage Temperature -65°C to +150°C
 Voltage from any Pin
 with Respect to V_{SS} -1.0 to +7.0V
 Power Dissipation 3W

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

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS T_A = 0 to 70°C, V_{CC1}/V_{CC2} = +5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V _{REF}	Reference Voltage		2	V	Typ = 1.0V
R _{VREF}	Source Impedance of V _{REF}		200	Ω	
V _{OH}	Output High Voltage	2.4		V	I _{OH} = -400 μA
V _{OL}	Output Low Voltage		0.4	V	I _{OH} = 2.0 mA
V _{IH}	Input High Voltage	2.0	V _{CC} + 0.5V	V	
V _{IL}	Input Low Voltage	-0.5	0.8	V	
V _{IHC} (¹)	Input High Voltage Clock	3.5	V _{CC} + 0.5V	V	
V _{ILC} (¹)	Input Low Voltage Clock	-0.5	0.8	V	
I _{LI}	Input Leakage Current		±10	μA	0V < V _{IN} < V _{CC}
I _{LO}	Output Leakage Current		±10	μA	0.45V < V _{OUT} < V _{CC}
I _{CC}	Power Supply Current		300	mA	
C _{IN}	Capacitance of Inputs		10	pF	f _c = 1 MHz
C _{IO}	Capacitance of I/O's		15	pF	f _c = 1 MHz
C _{OUT}	Capacitance of Outputs RAS, CASL, CASH, WE, OE		15	pF	f _c = 1 MHz
C _{OUT}	Capacitance of Outputs		10	pF	f _c = 1 MHz
C _{OUT}	Capacitance of Outputs (R/DV3, G/DV2, B/DV1, I/DV)		7	pF	f _c = 1 MHz
C _{RAS}	RAS Load		200	pF	
C _{CAS}	CASn Load		100	pF	
C _{WE}	WE Load		200	pF	
C _{Dij}	DL0-DL7 DH0-DH7 Load		100	pF	
C _{ADD}	ADD0-ADD8 Load		150	pF	

NOTE:

1. For XTALIN, CKIO and RESET pins only.

A.C. CHARACTERISTICS $T_A = 0 \text{ to } 70^\circ\text{C}$, $V_{CC1}/V_{CC2} = +5\text{V} \pm 10\%$

TIMING REQUIREMENTS

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{CLCL}	CLOCK Cycle Period	70	200	ns	
DCCK	Duty Cycle	40	60	%	
t_{CLCH}	CLK Low Time	$0.4 t_{CLCL}$		ns	
t_{CHCL}	CLK High Time	$0.4 t_{CLCL}$		ns	
t_{VCLCL}	VIDEO CLOCK Cycle Period	40	200	ns	
DCVCK	Duty Cycle	40	60	%	
t_{VCILIH}	Video Clock Rise Time(1)		10	ns	From 1.0V to 3.5V
t_{VCIHIL}	Video Clock Fall Time(1)		10	ns	From 3.5V to 1.0V
t_{ILIH}	Input Rise Time		20	ns	From 0.8V to 2.0V
t_{IHIL}	Input Fall Time		20	ns	From 2.0V to 0.8V

NOTE:

1. Timings defined for CKIO in input mode.

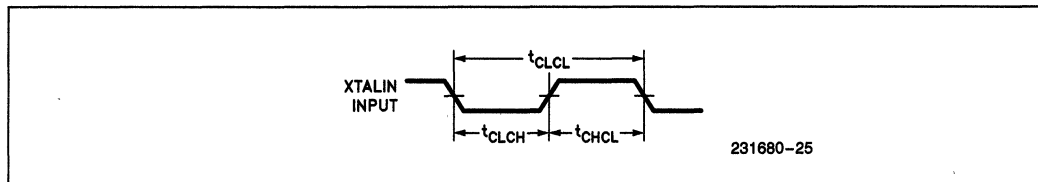


Diagram 1

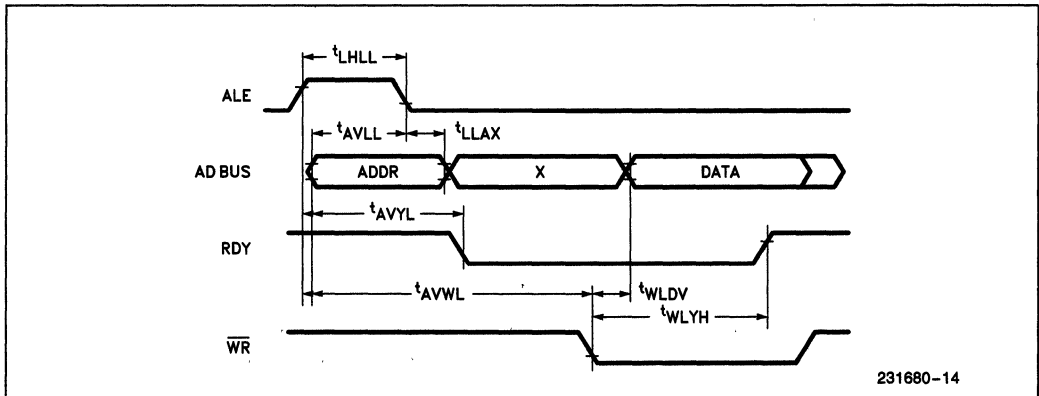
BUS INTERFACE UNIT

CPU WRITE CYCLE TIMINGS

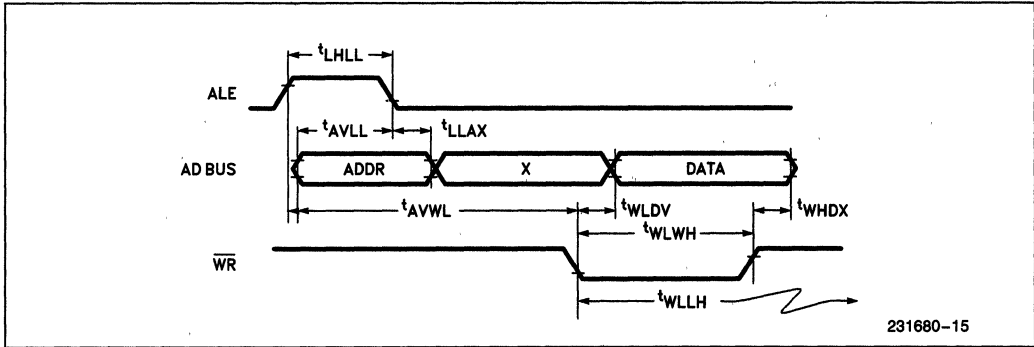
Symbol	Parameter	Min	Max	Units	Note
t_{LHLL}	ALE Pulse Width	35		ns	
$t_{AVLL}^{(1)}$	Address Set-Up Time	15		ns	
t_{LLAX}	Address Hold Time	25		ns	
t_{AVWL}	Address Valid or ALE HIGH (whichever is later) to \overline{WR} LOW	75		ns	
t_{AVYL}	Address Valid or ALE HIGH (whichever is later) to RDY LOW		100	ns	
t_{WLDV}	Data Valid after \overline{WR} LOW		$8t_{CLCL} - 100$	ns	
t_{WLYH}	\overline{WR} LOW to RDY High		$14t_{CLCL} + 100$	ns	
t_{WLWH}	\overline{WR} Pulse Width	$2t_{CLCL} + 20$		ns	
t_{WHDX}	Data Hold Time After \overline{WR} High	0		ns	
t_{WLLH}	\overline{WR} Low to ALE High of Next $\overline{RD}/\overline{WR}$ Cycle	$18t_{CLCL} + 100$		ns	

NOTE:

1. Chip select input, A16, has the same timing spec as the other address inputs.



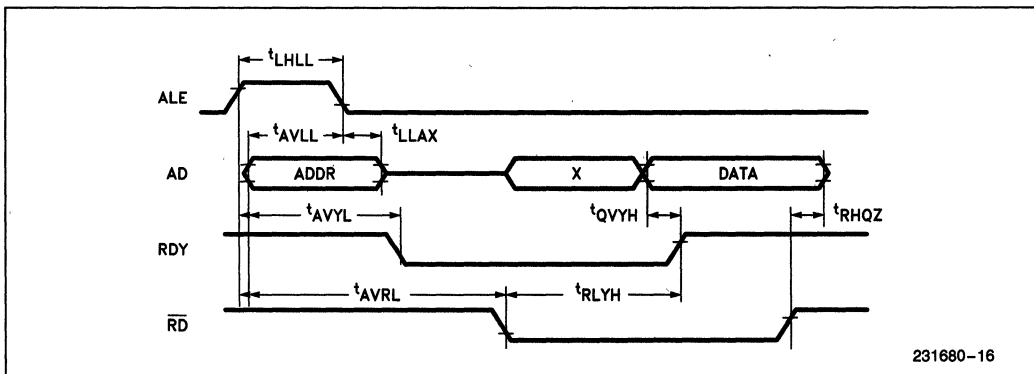
CPU Write Cycle Using Ready to Generate WAIT States



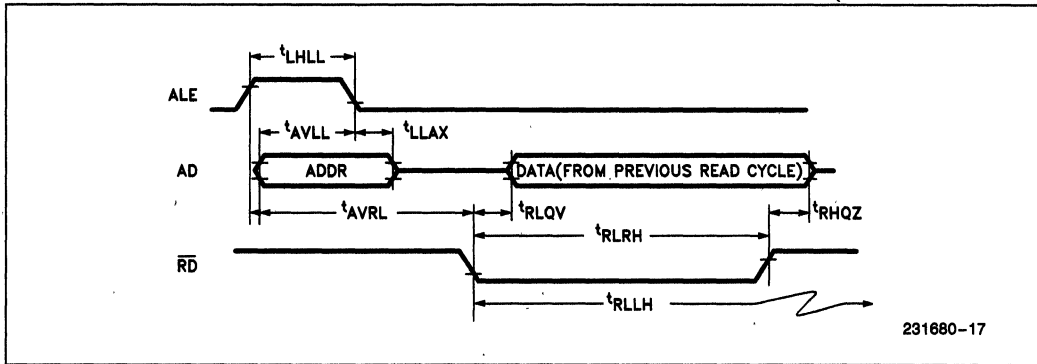
CPU Write Cycle Not Using Ready

CPU READ CYCLE TIMINGS

Symbol	Parameter	Min	Max	Units	Note
t_{RLYH}	\overline{RD} LOW to RDY High		$19t_{CLCL} + 100$	ns	
t_{QVYH}	Data Valid to RDY High	0		ns	
t_{RHQZ}	Data Float Time after \overline{RD}		40	ns	
t_{RLQV}	\overline{RD} LOW to Data Valid (PRE = 1)	10	40	ns	
t_{RLRH}	\overline{RD} Pulse Width (PRE = 1)	$2t_{CLCL} + 20$		ns	
t_{RLLH}	\overline{RD} LOW to ALE High of Next $\overline{RD}/\overline{WR}$ Cycle	$18t_{CLCL} + 100$		ns	
t_{AVRL}	Address Valid or ALE High (whichever is later) to \overline{RD} LOW	75		ns	



CPU Read Cycle Using Ready to Generate WAIT States (PRE = 0)



CPU Read Cycle Not Using Ready (PRE = 1)

231680-17

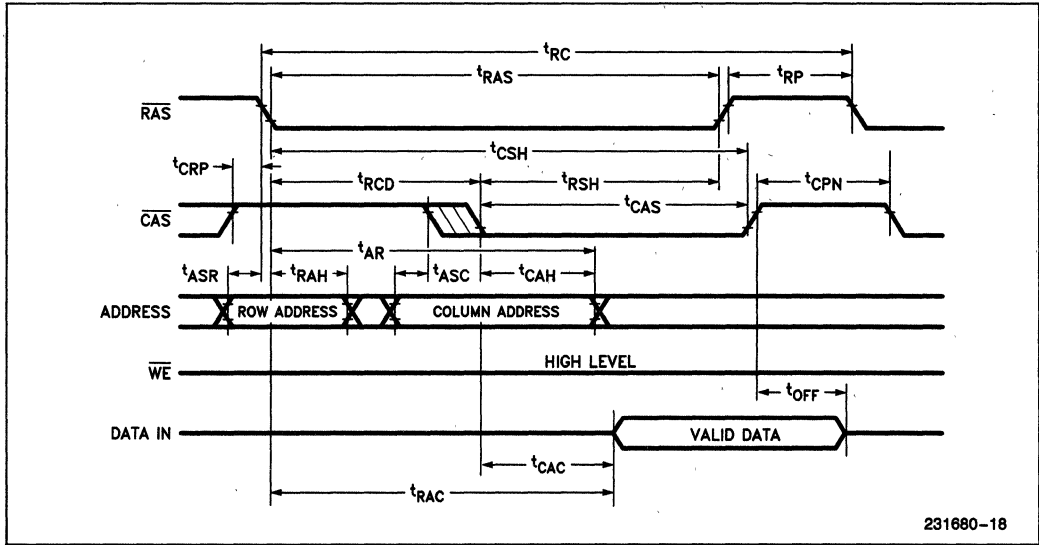
DRAM CONTROLLER

READ CYCLE

Symbol	Parameter	SAB = 1		SAB = 0		Units
		Min	Max	Min	Max	
t_{RC}	Random Read Cycle Time	$5 t_{c1} - 20$		$4 t_{c1} - 20$		ns
t_{REF}	Refresh Time 128 Cycles 256 Cycles	Note 1 Note 2		Note 1 Note 2		ms
t_{RP}	RAS Precharge Time	$2 t_{c1} - 20$		$2 t_{c1} - 20$		ns
t_{CPN}	CAS Precharge Time (Non-Page Mode)	$3 t_{c1} - 40$		$3 t_{c1} - 40$		ns
t_{RCD}	RAS to CAS Delay Time	$t_{c1} - 10$		$t_{c1} - 10$		ns
t_{RSH}	RAS Hold Time	$2 t_{c1} - 5$		$t_{c1} - 5$		ns
t_{CSH}	CAS Hold Time	$3 t_{c1} - 15$		$2 t_{c1} - 15$		ns
t_{ASR}	Row Address Set-Up Time	$t_{c1} - 15$		$t_{c1} - 15$		ns
t_{RAH}	Row Address Hold Time	$t_{c1} - 10$		$t_{c1} - 10$		ns
t_{ASC}	Column Address Set-Up Time	$t_{c1} - 10$		$t_{c1} - 10$		ns
t_{CAH}	Column Address Hold Time	$t_{c1} - 5$		$t_{c1} - 5$		ns
t_{AR}	Column Address Hold to RAS	$3 t_{c1} + t_{c1} - 20$		$2 t_{c1} + t_{c1} - 20$		ns
t_{RAS}	RAS Pulse Width	$3 t_{c1} - 15$		$2 t_{c1} - 15$		ns
t_{CAS}	CAS Pulse Width	$2 t_{c1} - 5$		$t_{c1} - 5$		ns
t_{CRP}	CAS to RAS Precharge Time	$2 t_{c1} - 50$		$2 t_{c1} - 50$		ns
t_{CAC}	Access Time from CAS		$2 t_{c1} - 10$		$t_{c1} - 10$	ns
t_{RAC}	Access Time from RAS		$3 t_{c1} - 30$		$2 t_{c1} - 30$	ns
t_{OFF}	Data-In Hold Time	0		0		ns

NOTES:

- $(128 / (12 * \text{scan line time})) + 10,000 t_{c1}$
- $(256 / (12 * \text{scan line time})) + 10,000 t_{c1}$

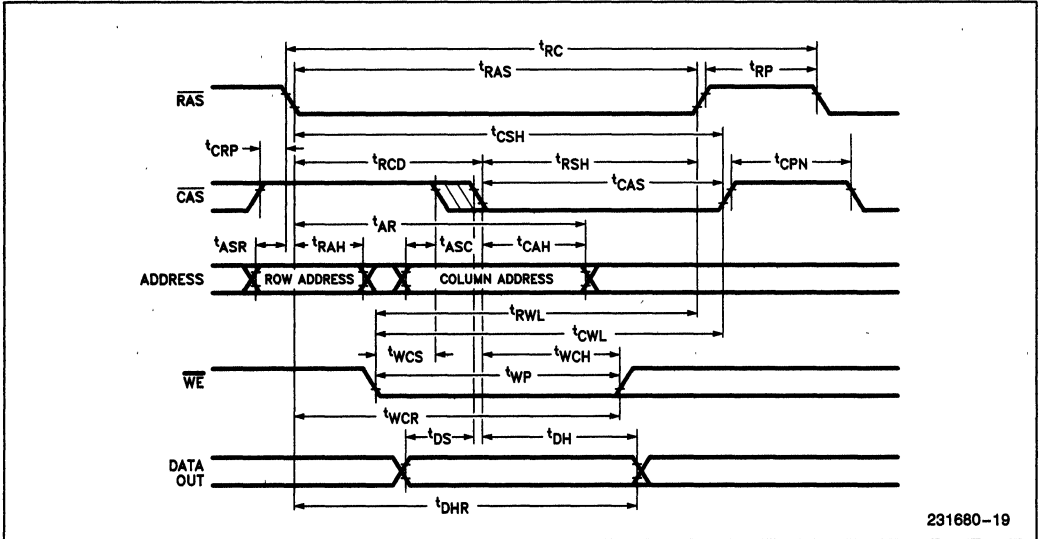


231680-18

Read Cycle

WRITE CYCLE

Symbol	Parameter	SAB = 1		SAB = 0		Units
		Min	Max	Min	Max	
t_{RC}	Random Write Cycle Time	5 tc1cl - 20		4 tc1cl - 20		ns
t_{RAS}	\overline{RAS} Pulse Width	3 tc1cl - 15		2 tc1cl - 15		ns
t_{CAS}	\overline{CAS} Pulse Width	2 tc1cl - 5		tc1cl - 5		ns
t_{WP}	Write Command Pulse Width	3 tc1cl - 20		2 tc1cl - 20		ns
t_{WCS}	Write Command Set-Up Time	tc1ch - 10		tc1ch - 10		ns
t_{WCH}	Write Command Hold Time to \overline{CAS}	2 tc1cl + tc1ch - 40		tc1cl + tc1ch - 40		ns
t_{WCR}	Write Command Hold Time to \overline{RAS}	3 tc1cl + tc1ch - 40		2 tc1cl + tc1ch - 40		ns
t_{RWL}	Write to \overline{RAS} Lead Time	2 tc1cl + tc1ch - 40		tc1cl + tc1ch - 40		ns
t_{CWL}	Write to \overline{CAS} Lead Time	2 tc1cl + tc1ch - 40		tc1cl + tc1ch - 40		ns
t_{DS}	Data-Out Set-Up Time	tc1cl + tc1ch - 50		tc1cl + tc1ch - 50		ns
t_{DH}	Data-Out Hold Time	2 tc1cl + tc1ch - 20		tc1cl + tc1ch - 20		ns
t_{DHR}	Data-Out Hold Time to \overline{RAS}	3 tc1cl + tc1ch - 20		2 tc1cl + tc1ch - 20		ns
t_R, t_F	Rise, Fall Time $\overline{RAS}, \overline{CAS}$	5	40	5	40	ns

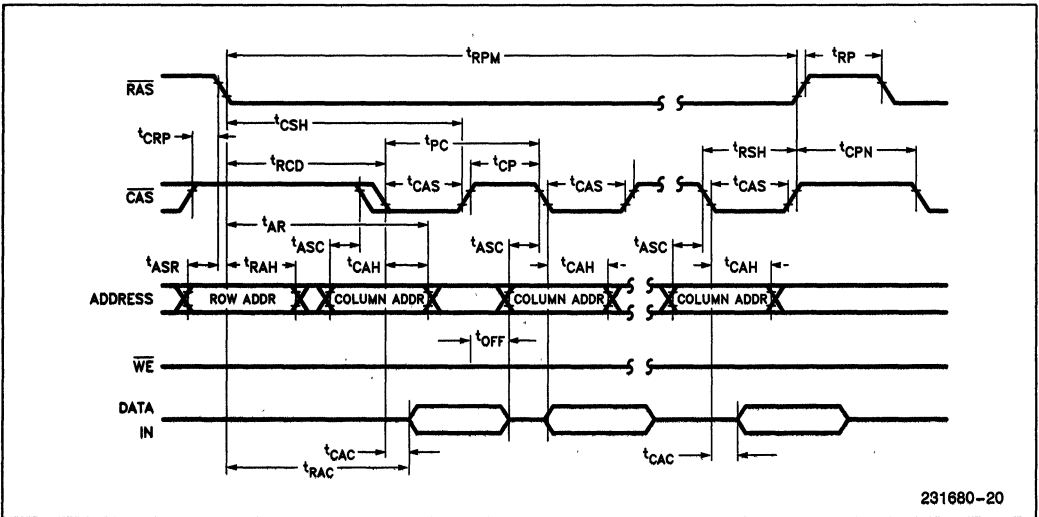


231680-19

Write Cycle

PAGE MODE

Symbol	Parameter	SAB = 1		SAB = 0		Units
		Min	Max	Min	Max	
t_{PC}	Page Mode Read Cycle	$3 t_{clcl} - 15$		$2 t_{clcl} - 15$		ns
t_{CP}	\overline{CAS} Precharge Time	$t_{clcl} - 15$		$t_{clcl} - 15$		ns
t_{CAS}	\overline{CAS} Pulse Width	$2 t_{clcl} - 5$		$t_{clcl} - 5$		ns
t_{RPM}	\overline{RAS} Pulse Width		$96 t_{clcl} - 5$		$65 t_{clcl} - 5$	ns

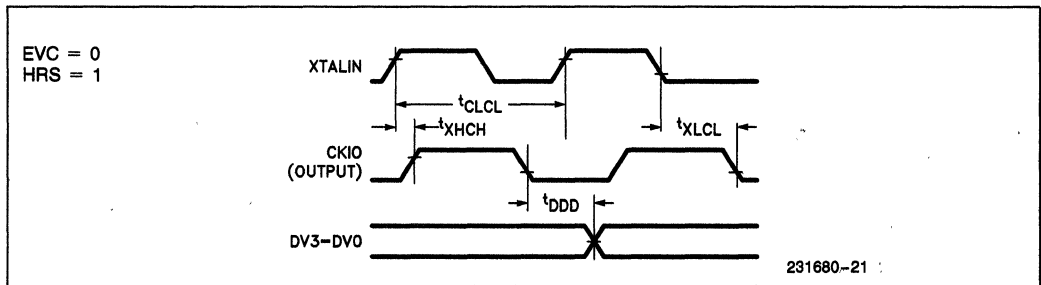


231680-20

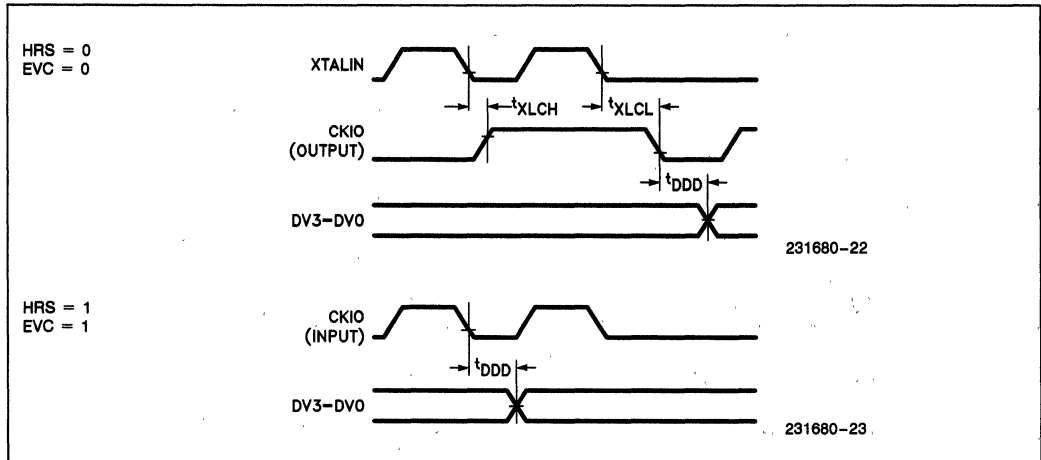
Page Mode Read Cycle

VIDEO OUTPUT TIMINGS

Symbol	Parameter	Min	Max	Units	Comments
t_{XHCH}	XTALIN High to CKIO High		60	ns	EVC = 0 HRS = 1
t_{XLCL}	XTALIN Low to CKIO Low		70	ns	EVC = 0 HRS = 1
			75	ns	EVC = 0 HRS = 0
t_{XLCH}	XTALIN Low to CKIO High		80	ns	EVC = 0 HRS = 0
t_{DDD}	Digital Data Delay		30	ns	EVC = 0 HRS = 1
			35	ns	EVC = 0 HRS = 0
			55	ns	EVC = 1 HRS = 1



Video Output Timings



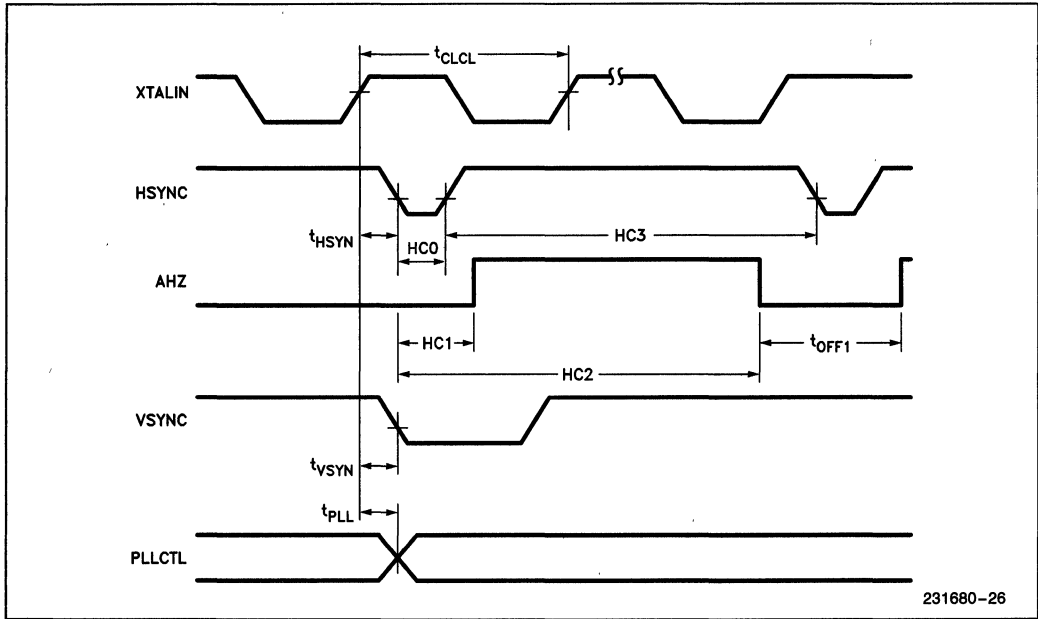
Video Output Timings

DAC SPEC

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{REF}	Reference Voltage		2.0	V	typ = 1.6V
	Linearity		1/2 LSB		$V_{REF} = 1.6V \pm 5\%$
t_s	Settling Time		20	ns	Max Load = 10 pF

SYNC SPEC

t_{HSYN}	HSYNC Delay from XTALIN		150	ns	
t_{VSYN}	VSYNC Delay from XTALIN		150	ns	
t_{OFF1}	Dead Zone between Two Active Horizontal Zones	120 tccl		ns	
t_{PLL}	PLLCTL Valid Delay from HSYNC		100	ns	



231680-26

Sync Specs

82730 TEXT COPROCESSOR

- High Quality Display for Text and Graphics Applications
- Provides Proportional Spacing, Simultaneous Superscript/Subscript, Soft Font Support and Bit Map Graphics
- High Performance Manipulation of Text/Graphics Strings
- Programmable Bus Interface Handles 8 or 16 Bit Data and 16 or 32 Bit Addressing; iAPX 86/88/186/188 Compatible
- On-Chip Processing Unit Simplifies Software Design by Executing High Level Commands and Supporting Linked List Data Structures
- Extremely Flexible; Programmable Features Include Screen and Row Formats, Two Cursors, Character and Field Attributes and Smooth Scrolling
- Supports Multiple Windows
- High Resolution Display; Up to 200 Characters/Graphics Cells per Row and 2048 Scan Lines per Frame
- Separate Bus and Video Clocks Allow Optimization of Overall System Performance
- Provides a Complete LSI Solution for Display Control when Used in Conjunction with the 82731 Video Interface Controller
- 68 Pin JEDEC Package: (See Intel packaging: Order Number: 210931-004)

The 82730 Text Coprocessor is a high performance VLSI solution for raster scan text and graphics displays. The 82730 works as a coprocessor and has processing capabilities specifically tailored to execute data manipulation and display tasks. It provides the designer the ability to functionally partition his system thereby offloading the system CPU and achieving maximum performance through concurrent processing. The 82730 supports the generation of high quality text displays through features like proportional spacing, simultaneous superscript/subscript, dynamically reloadable fonts and user programmable field and character attributes. It supports high quality graphics with fast manipulation and display of bit map strings. An intelligent system interface and efficient software capabilities makes 82730 based systems easy to design.

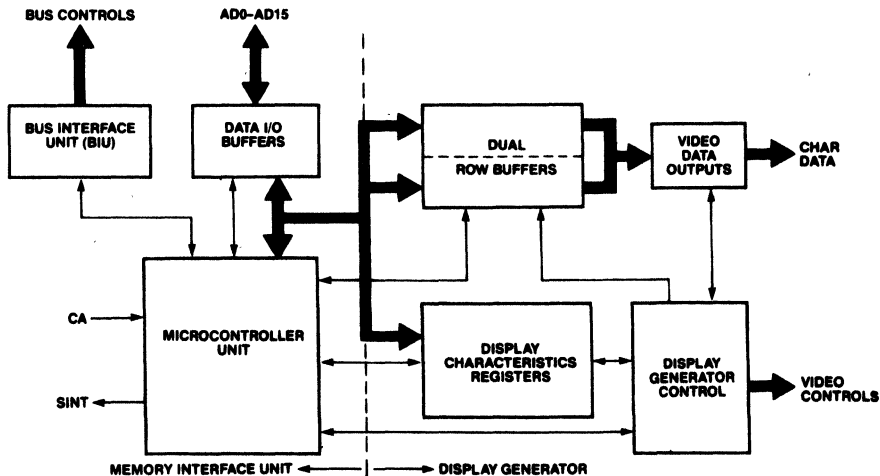


Figure 1. 82730 Block Diagram

210931-1

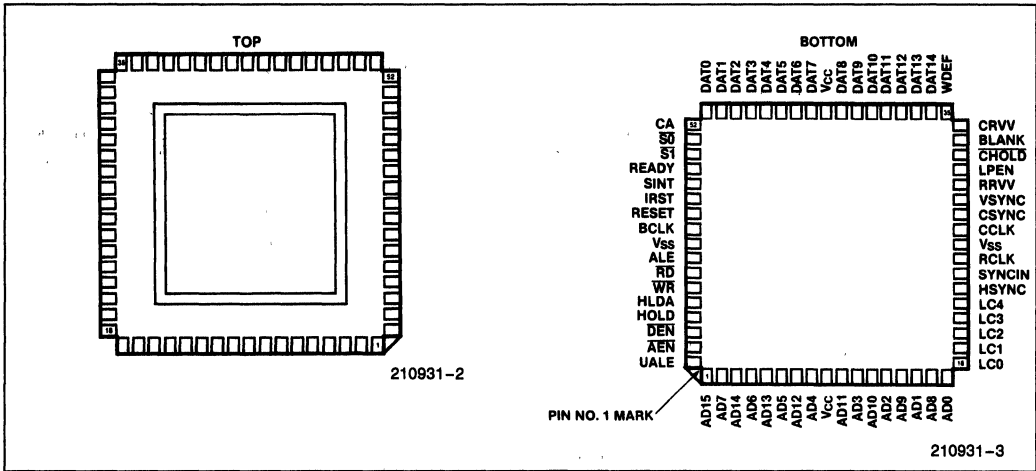


Figure 2. 82730 Pinout Diagram

Table 1. 82730 Pin Description

The 82730 is packaged in a 68 pin JEDEC Type A ceramic package.

Symbol	Pin Number	Type	Name and Function
AD15-AD0	1-8 10-17	I/O	ADDRESS DATA BUS: these lines output the time multiplexed address (TU, T1 states) and data (T2, T3, T4 and TW) bus. The bus is active HIGH and floats to 3-state OFF when the 82730 is not driving the bus (i.e., HOLD is not active or when HOLD is active but not acknowledged, or when RESET is active).
BCLK	59	I	BUS CLOCK: provides the basic timing for the Memory Interface Unit.
\overline{RD}	62	O	READ STROBE: indicates that the 82730 is performing a memory read cycle on the bus. \overline{RD} is active low for T2, T3 and TW of any read cycle and is guaranteed to remain high in T2 until the address is removed from the bus. \overline{RD} is active low and floats to 3-state OFF when 82730 is not driving the bus. \overline{RD} will return high before entering the float state and will not glitch low when entering or leaving float.
\overline{WR}	63	O	WRITE STROBE: indicates that the data on the bus is to be written in a memory device. \overline{WR} is active for T2, T3 and TW of any write cycle. It is active LOW and floats when 82730 is not driving the bus. \overline{WR} will return high before entering the float state and will not glitch low when entering or leaving float.
ALE	61	O	LOWER ADDRESS LATCH ENABLE: provided by the 82730 to latch the address into an external address latch such as 8282/8283 (active HIGH). Addresses are guaranteed to be valid on the trailing edge of ALE.
UALE	68	O	UPPER ADDRESS LATCH ENABLE: it is similar to ALE except that it occurs in upper address output cycle (TU).
\overline{AEN}	67	O	ADDRESS ENABLE: \overline{AEN} is active LOW during the entire period when 82730 is driving the bus. It can be used to unfloat the outputs of the Upper and Lower Address latches.

Table 1. 82730 Pin Description (Continued)

Symbol	Pin Number	Type	Name and Function															
DEN	66	O	DATA ENABLE: provided as a data bus transceiver output enable for transceivers like the 8286/8287. DEN is active LOW during each bus cycle and floats when 82730 is not driving the bus. DEN will not glitch when entering or leaving the float state.															
S ₀ , S ₁	53, 54	O	STATUS PINS: encoded to provide bus-transaction information:															
			<table border="1"> <thead> <tr> <th>S₁</th> <th>S₀</th> <th>Bus Cycle Initiated</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>--- (Reserved)</td> </tr> <tr> <td>0</td> <td>1</td> <td>Memory Read</td> </tr> <tr> <td>1</td> <td>0</td> <td>Memory Write</td> </tr> <tr> <td>1</td> <td>1</td> <td>Passive (No Bus Cycle)</td> </tr> </tbody> </table>	S ₁	S ₀	Bus Cycle Initiated	0	0	--- (Reserved)	0	1	Memory Read	1	0	Memory Write	1	1	Passive (No Bus Cycle)
			S ₁	S ₀	Bus Cycle Initiated													
			0	0	--- (Reserved)													
0	1	Memory Read																
1	0	Memory Write																
1	1	Passive (No Bus Cycle)																
These pins are directly compatible with iAPX 86, 186 status outputs S ₁ and S ₀ . The status pins are floated when 82730 is not driving the bus. They will not glitch when entering or leaving the 3-state condition.																		
READY	55	I	READY: signal to inform the 82730 that the data transfer can be completed. Immediately after RESET, READY is asynchronous (internally synchronized) but can be programmed during initialization to bus synchronous.															
HOLD	65	O	HOLD: indicates that the 82730 wants bus access. HOLD stays active HIGH during the entire period when 82730 is driving the bus.															
HLDA	64	I	HOLD ACKNOWLEDGE: indicates to 82730 that it is granted the bus access as requested. HLDA may be asynchronous to 82730 clock. If HLDA goes inactive (LOW) in the middle of an 82730 bus cycle, the 82730 will complete the current bus cycle first, then it will drop HOLD and float address and bus control outputs.															
CA	52	I	CHANNEL ATTENTION: used to notify 82730 that a command in the command block is waiting to be processed. CA is latched on its falling edge.															
SINT	56	O	STATUS INTERRUPT: used to inform the processor that an unmasked interrupt has been generated in the 82730 status register.															
IRST	57	I	INTERRUPT RESET: SINT is cleared by activating the IRST pin.															
RESET	58	I	RESET: causes 82730 to immediately terminate its present activity and enter a dormant state. The signal must be active HIGH for at least 4 BCLK cycles and is internally synchronized to the bus clock.															
CCLK	27	I	CHARACTER CLOCK: input used to clock row buffer data, attribute, cursor and line count out of 82730. When more than one 82730 is connected in cluster mode, CCLK is used to synchronize output from both master and slave chips. A character data word will be output at every rising edge of CCLK.															
RCLK	25	I	REFERENCE CLOCK: input used to generate timings for the screen layout and to define screen columns for data formatting. All raster output signals are specified relative to the rising edge of RCLK.															
DAT0-DAT14	36-42 44-51	O	VIDEO DATA BUS OUTPUT: the least significant 15 bits of the character data words are passed through the 82730 row buffer and made available on the pins DAT0-DAT14. The user has the flexibility to partition the data word into character and attribute bits per his requirements. The bits that are assigned for internally generated attributes may also be available at pin DAT0-DAT14. New character data will be shifted to these output pins at every rising edge of the CCLK. Together with LC0-LC4, they may be used to address the character generator or as attribute controls.															

Table 1: 82730 Pin Description (Continued)

Symbol	Pin Number	Type	Name and Function
WDEF	35	O	WIDTH DEFEAT: is used to indicate when the character is allowed to be a variable width or must be of fixed width. WDEF is LOW if the character being output is normal, but is HIGH if it is a superscript/ subscript character or visible attribute (TAB or GPA). Optionally, WDEF can be held high by user command.
LC0-LC4	18-22	O	LINE COUNT OUTPUTS: used to address the character generator for the line positions in a row. The line number output is a function of the display mode and character attributes programmed by the user.
CSYNC	28	O	CCLK SYNCHRONIZATION OUTPUT: used to synchronize external character clock generator to reference clock timing. This output is active (high) outside the display field.
CHOLD	32	O	CCLK INHIBIT OUTPUT: used by external logic to inhibit CCLK generation. This output is active (low) during the tab and end-of-row function.
SYNCIN	24	I	SYNCHRONIZATION INPUT: used to synchronize the vertical timing counters to an externally generated VSYNC signal. Used by slave mode 82730 to synchronize to a master mode 82730 and by the master 82730 to lock the frame to an external source such as the power line frequency.
HSYNC	23	O (MASTER) I (SLAVE)	HORIZONTAL SYNC: in master mode, it is used to generate the CRT monitor's horizontal sync signal. It is active HIGH during the programmed horizontal sync interval. In interlace slave mode it is used in conjunction with SYNCIN to indicate the start of the even field for timing counter reset. At RESET, pin is set as an output in the LOW state.
VSYNC	29	O	VERTICAL SYNC: active HIGH during the programmed vertical sync interval and used to generate the CRT monitor's vertical sync signal.
BLANK	33	O	BLANKING OUTPUT: used to suppress the video signal to the CRT. BLANK is clocked by CCLK.
CRVV	34	O	CHARACTER REVERSE VIDEO (CCLK OUTPUT): used to externally invert video data output. CRVV is clocked by CCLK.
RRVV	30	O	REFERENCE REVERSE VIDEO (RCLK OUTPUT): to externally invert video in the field and border area if so programmed by user. It is LOW outside the border area, RRVV is clocked by RCLK.
LPEN	31	I	LIGHT PEN INPUT: used to latch the position of a light pen. At the rising edge of this input, the column position and the row position of the 82730 will be loaded into the LPENROW and LPENCOL locations in the Command block.
V _{CC}	9, 43		POWER: + 5V nominal potential.
V _{SS}	26, 60		POWER: ground potential.

FUNCTIONAL DESCRIPTION

Figure 1 shows a basic block diagram of the 82730 Text Coprocessor. The chip is divided into two main sections, the Memory Interface Unit and the Display Generator. The Memory Interface Unit controls fetching of the data and commands and handles interrupts and status. The Display Generator takes the data fetched by the Memory Interface Unit and presents it to the Video Interface logic which in turn drives the CRT monitor.

Memory Interface Unit

The Memory Interface Unit is divided into two sections: The Bus Interface Unit and the Microcontroller Unit. The Bus Interface Unit does the actual interfacing to the memory bus. It fetches or writes data under the control of the Microcontroller Unit. The Microcontroller Unit is a microprogrammed controller which is designed to efficiently fetch data from memory (up to 4 Mbytes/sec), and decode and execute various control and data handling commands. The Bus Interface Unit may be configured for 8 or 16 bit bus operation. With 8 bit bus selection, the user may specify either 8 or 16 bit character data. It also handles address manipulation automatically after being loaded from the Microcontroller Unit.

Display Generator

The Display Generator takes the data fetched from memory plus the modes programmed into it at initialization and produces all the video timing and the data transfers to support the CRT monitor at the character level. The 82730 works with an external character generator and the 82731 Video Interface Controller. The data is passed to the Display Generator from the Memory Interface Unit through the dual row buffers (similar in operation to the one in the 8275 CRT controller). The row buffers allow the user to use cheaper and slower main memory for display needs, provide on-chip attribute and display function generation, and avoid the conflict of access to the display memory (that would otherwise take place) by using an ordinary DMA access mechanism.

SYSTEM BUS INTERFACE

The Memory Interface Unit provides communication with system processor as well as memory interactions. Communication between the processor and the 82730 is performed via messages placed in communication blocks in shared memory. The processor can issue commands by preparing message blocks and directing the 82730's attention to them by asserting a hardware channel attention. The

82730 can cause interrupts on certain conditions, if enabled by the processor by activating its System Interrupt output, with status and error reporting taking place through the communication block in memory.

BUS INTERFACE UNIT

The 82730 Bus Interface Unit provides an 8086 compatible bus interface which consists of:

- a 16/32 bit multiplexed Address/Data Bus: AD₀-AD₁₅
- A complete set of local bus control signals compatible with 8086 min mode: RD, WR, ALE, DEN, and READY
- Two status signals $\overline{S0}$ and $\overline{S1}$, compatible with 8086 max mode so that a bus controller (8288) can be shared for Multibus® access.
- Local bus arbitration through HOLD/HLDA
- Two upper Address Latch controls: UALE and AEN

The BUS INTERFACE UNIT (BIU) utilizes the same Bus structure as the 80186 or basically the same bus structure as the 8086 in both Min. and Max. mode, (with the exception of RQ/GT) and it performs a bus cycle only on demand (e.g., to fetch a command from the command block, or fetch a character from display data memory). The same set of T-states (T1, T2, T3, T4 and TW) of 8086 are used to handle the time multiplexed address/data bus. However, adaptations are made to handle 32 bit addresses as explained in the following sections where specific details of the BIU operation are described. Those details not mentioned can be assumed to be the same as those of the 80186.

ADDRESS BUS

The 82730 can be programmed during initialization to operate on either 16 bit or 32 bit (including any length between 17 and 32) physical addresses. Note that the 82730 does not use memory segmentation. The programmer must calculate physical addresses from segment and offset values to manipulate data structures.

To support 32 bit physical addresses with a 16 bit physical bus, multiplexing is again used. An upper address output cycle, TU, is inserted between T4 and T1 to output the upper 16 bits of address. The upper address latch enable, UALE, is used to latch the upper addresses during TU. Figure 3 shows the configuration of a 32 bit address bus.

TU occurs only when the 32 bit mode is specified and the upper address register of BIU is reloaded by MCU. This may result from:

- i) Initialization
- ii) Manipulation of display data or command pointers, for example, when a new string pointer is loaded during the execution of the END OF STRING command.
- iii) DMA address incrementing across a 64K byte segment boundary.
- iv) Regaining the bus after losing it to a higher priority master.

Timing of UALE is identical to that of ALE. \overline{AEN} is equivalent to the active period of 82730 driving the bus.

If 16 bit address mode is programmed, TU will never occur in any bus cycle since the MIU treats all display pointers as 16 bit quantities and loading of internal upper address register is bypassed during address calculation. UALE always stays inactive, but \overline{AEN} still goes active to indicate the 82730 has control of the bus.

DATA BUS

The 82730 is capable of operating on either an 8 bit or a 16 bit Data bus, as programmed during initialization on the SYSBUS byte.

When an 8 bit data bus is specified, the address present on AD15 to AD8 Address/Data lines is maintained for the complete bus cycle. Therefore, compatibility with 80188, 8088, 8089 and 8085 multiplexed address peripherals is maintained. Since the internal processing of the 82730 generally operates on 16 bit data quantities, two Bus fetch cycles are performed for each 16 bit data item. The first cycle fetches the low order byte, the second cycle the high order byte. These 2 fetch cycles are always executed back to back. If HLDA drops during the first cycles, the 82730 will not respond until the second cycle is completed. An 8 bit data mode can be selected in an 8 bit bus system that requires only 8 bit character data be fetched.

In 16 bit bus system, the 82730 requires all 16 bit quantities to start on even address boundary. Word transfer to or from odd boundary is not allowed since this type of transfer not only doubles the use of bus bandwidth but also can be easily avoided in application software. All that is required is to make sure all address pointers be an even number ($A0 = 0$).

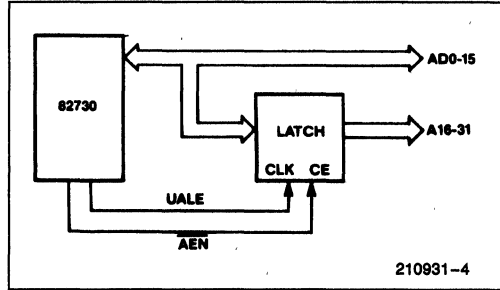


Figure 3. Address Extension up to 32 Bits

BUS CONTROLS

The 82730 BIU provides both the 8086 MIN. Mode (Local Bus Control) and MAX. mode bus control signals simultaneously in any bus cycle. By providing a complete set of Local Bus control signals, the component count of the Local processing module is minimized.

Because only two types of Bus operations, Memory Read and Memory Write, are executed in the 82730 BIU, the 8086's $\overline{S2}$ status signal is omitted from the Max. mode controls. $\overline{S2}$ could be set to "1" during any 82730 Bus cycle. \overline{AEN} can be used to produce $\overline{S2}$ since it stays active whenever 82730 is driving the bus. The status signals become valid at the middle of the cycle before T1 which could be either T4 or TU.

\overline{BHE} is not provided on the 82730 because, the 82730 only writes words to even address boundaries and bytes to the upper byte position. For these writes \overline{BHE} is always high. A pullup resistor or a three-state buffer controlled by \overline{AEN} , can provide this signal.

DT/R is also not provided on the 82730 because its function can be replaced with $\overline{S1}$, latched by ALE.

After RESET is applied, READY is set to be an asynchronous input. An on-board synchronization circuit provides reliable operation for any type of system. During initialization, READY may be programmed to be bus synchronous. For those systems that can meet the set-up time specifications, this mode provides more efficient bus utilization.

LOCAL BUS ARBITRATION

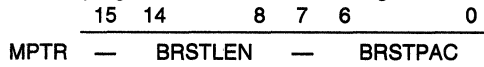
The 82730 BIU is designed to function as a bus master in a multimaster Local bus environment using the HOLD/HLDA protocol for Bus arbitration.

In the Self Contained Arbitration scheme, one processor and one 82730 share access to the local bus. The 82730 raises its HOLD request whenever it needs bus access. After HLDA is granted from the processor, the 82730 will not start driving the bus until 2 clock cycles later. This latency allows sufficient time for the 8086 or 80186 processor to get off the bus. When 82730 completes its bus accesses, it will first float its output drivers before dropping the hold request.

In a Local bus configuration with three or more bus masters, a higher priority DMA Peripheral device can preempt the HLDA from a 82730 which is the current bus master. The 82730 will complete its current bus cycle, then float its output drivers and drop the HOLD request. However, the 82730 may raise the HOLD request again 2 clock cycles later if it still needs the bus to complete the interrupted burst DMA activities.

DMA BURST AND SPACE

Some system configurations using the 82730 would be adversely affected by the long burst data transfers which the Memory Interface Unit (MIU) may occasionally desire. Since the 82730 will normally be configured as one of the higher priority bus masters, burst lengths must be limited for these systems. For this reason, the length of a burst transfer and the number of memory cycles between burst transfers are both programmable via the mode registers:



BRSTLEN—Burst Length. Determines the number of contiguous word-fetch cycles which may be requested. Programmable from 1 to 127. Note that in an 8 bit bus, 16 bit data system, the burst counter only increments once for the 2 bus cycles required to complete a word fetch. (Note: burst length = 0 is not defined and should not be programmed with a non-zero burst space.)

BRSTSPAC—Burst Space. Determines a minimum number of bus clocks to occur between burst accesses. Programmable from 0–511 in increments of four. Zero space selects an infinite burst length.

A DMA burst could be terminated before the programmed burst length is reached in the following circumstances:

- i) The MIU does not need any more bus accesses, for example, when the row buffer is filled.
- ii) A datastream command is encountered and the MIU must execute the command first before it resumes data accessing.

- iii) The bus is taken away by a higher priority device in multi-master bus configuration.

In these cases, the burst counter is cleared. The BIU must complete a full burst before it waits through the SPACE cycles. DMA Burst/Space will be set to zero space until the completion of the first MODESET command.

INITIALIZATION OF BIU

Upon activation of the RESET input, the 82730 BIU will stop all operations in progress and deactivate all outputs. It will stay in this quiescent state until memory access is requested by the MCU after MCU receives its first channel attention after RESET. The following table shows the state of all MIU outputs during and after reset.

Table 2. 82730 Bus During and After Reset

Signals	Condition
AD15–0	Three-state
RD, WR, DEN	Driven to '1' then three-state
S0, S1	Driven to '1' then three-state
ALE, UALE	Low
AEN	High
HOLD	Low
SINT	Low

82730 COMPATIBILITY ISSUES

82730 Bus Clock Compatibility

The 82730 uses the 50% duty cycle output of the iAPX-186 at 8 MHz or that generated by a clock generator such as the 82285. A different duty cycle clock may be used at lower frequencies, so the 82730 is also usable with the iAPX-86, 88 family.

82730 Bus Interface Compatibility

The bus interface compatibility between the 82730 and another bus master has four main issues: data bus width, addressability, control bus structure and local bus mastership arbitration.

Data Bus

Data Bus width compatibility with all 85/86 family processors (8085, 8086, 8088, 80188, 80186, and 80286) is being supported by the 8/16 data bit pro-

grammability already discussed. This allows interfacing to the above processors either directly or through a Multibus-like interface.

Address Bus

The 82730 uses real 32-bit addresses. The user's software must calculate real addresses; this general addressing scheme allows the 82730 to be used with any microprocessor.

Control Bus

The 82730 implements both 8086 minimum and maximum mode bus control structures. This was done to maximize compatibility with the 80186 which has the same structure. This allows the 82730 to be run locally (minimum mode) with a 8085, 8086, 8088, 80188, or 80186. The 80186/188 and 82730 can run together at 8 MHz because of clock duty cycle considerations. The 82730 can only communicate to an 80286 via a system bus (such as MULTIBUS), bus interface, or dual-port RAM.

INITIALIZATION SEQUENCE

The first CA (Channel Attention) after Reset causes an Initialization Sequence to be executed. The system processor must set up the appropriate initialization information in memory and set the BUSY flag in the Intermediate Block to a non-zero value prior to issuing this CA.

Initially, 32-bit addressing and 8-bit bus width are assumed until the corresponding information is fetched during the initialization. First the SYSBUS byte is fetched from memory location FFFF FFF6. (When the address bus is less than 32 bits wide, the higher order bits are unused.) The format for SYSBUS byte

is shown in Figure 4 and is the same as that used for 8089. The data bus width is specified by the least significant bit w , with $w = 0$ indicating an 8-bit bus and $w = 1$ signifying a 16-bit bus.

A 32-bit real address pointer is then fetched from memory locations FFFF FFFC through FFFF FFFF, with lower bytes of the pointer residing in lower addresses. This pointer is used as an Intermediate Block Pointer (IBP).

The Intermediate Block Pointer (IPB) is incremented by two and is used to locate the Command Block Pointer (CBP). Four bytes are fetched irrespective of whether a 16-bit or 32-bit addressing option is used. The System Configuration byte (SCB) is then fetched from location $(IBP + 6)$.

The least significant, (U of the SCB) specifies 16 or 32-bit addressing option, with $U = 0$ indicating 16 bit addressing and $U = 1$ specifying 32-bit addressing. The SCB also contains information about cluster operation. Since up to four 82730's can be connected in a cluster with their respective data interleaved in memory, cluster information is needed for the data access task. The SCB specifies Cluster Number (CL NO), which is the number of 82730's connected in a cluster and Cluster Position (CL POS) which is the position of this particular 82730 within the cluster. CL NO = 0, 1, 2 or 3 indicates a cluster containing 1, 2, 3 or 4 82730's respectively. Similarly, CL POS = 0, 1, 2 or 3 indicates 1st, 2nd, 3rd or 4th position respectively. Each 82730 adds an offset equal to $2 * CLPOS$ to the SPTR fetched from memory and increments the pointer by $2 * (CL NO + 1)$. The programming of CL NO and CL POS is independent. No checking is done for CL POS greater than CL NO on the 82730. Note that at least one 82730, in a cluster (even if it is a cluster of one), must be assigned as cluster position zero (CL POS = 0) for Virtual Display mode to work properly.

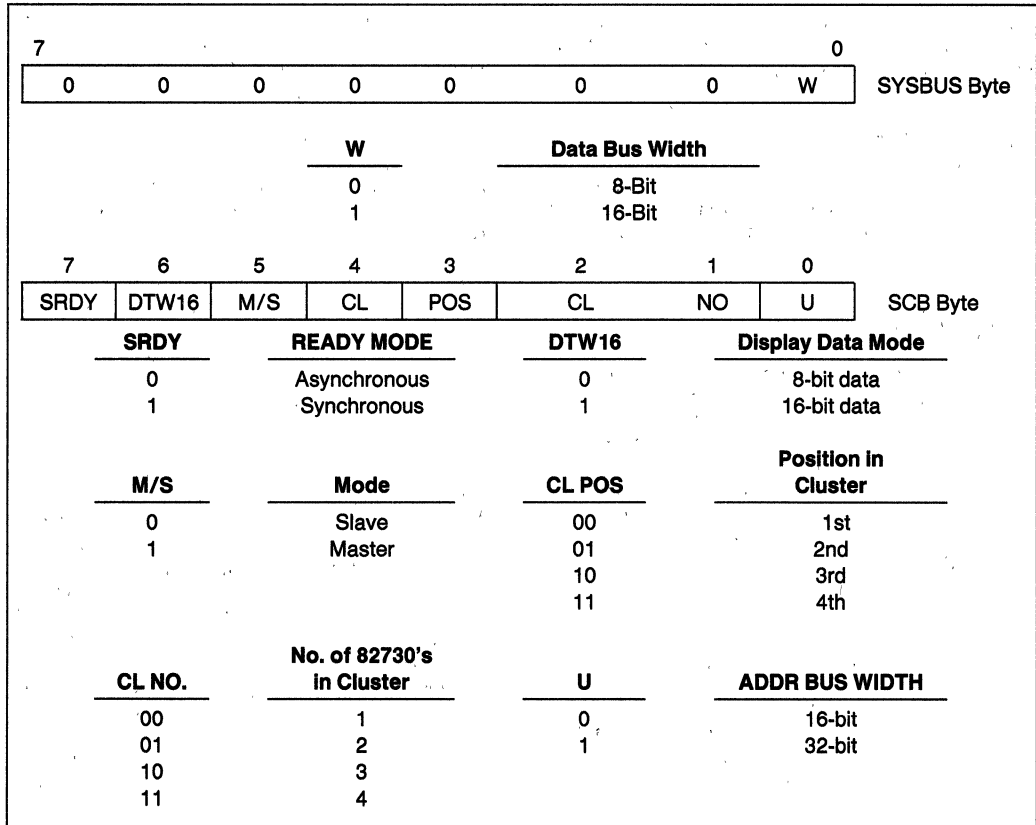


Figure 4. SYSBUS and SCB Encoding

The SCB also contains an M/\bar{S} bit which specifies a master or slave mode. The M/\bar{S} bit is stored internally for use by the Display Generator (DG). $M/\bar{S} = 1$ indicates a master mode and $M/\bar{S} = 0$ specifies a slave mode. The format for the System Configuration Byte (SCB) is shown in Figure 4. Following these actions, the BUSY flag in the Intermediate Block at address IBP is cleared and a normal Channel Attention sequence is then executed.

The last two bits in the SCB are DTW16 and SRDY. DTW16 specifies whether the display data in 8 bit bus mode ($W = 0$) is 8 or 16 bit. If a 16 bit system is specified ($W = 1$) then DTW16 is ignored and forced internally to a "one". SRDY specifies whether the clock synchronization circuit for the READY pin is internal (SRDY = 0) or external (SRDY = 1).

The Initialization Control Blocks in memory are illustrated in Figure 5a. How these fit into the control structure of the 82730 is shown in Figure 5b.

Channel Attention Sequence

When the processor activates CA, an internal latch in 82730 is set on the falling edge of CA input and this latch is sampled by the MCU. The first CA activation after reset causes the 82730 to execute an initialization sequence. Any subsequent activation will cause the MCU to start processing the command block by fetching a channel command.

If a display is in progress, the MCU will sample CA at each end of frame, otherwise it will sample CA every cycle until it is found active. When CA is found active, the MCU will fetch the command byte from "COMMAND" location in the command block, execute the command and clear the BUSY flag upon completion. The internal CA latch is also cleared by the MCU. An invalid command code has the effect of NOP and the BUSY flag is cleared. It will also cause the Reserved Channel Command (RCC) status bit to be set.

	15	8	7	0	
INTERMEDIATE BLOCK POINTER	IBP UPPER				FFFF FFFE
	IBP LOWER				FFFF FFFC
	(RESERVED SYSBUS)				FFFF FFF6
INTERMEDIATE BLOCK	(RESERVED) SCB)				IBP + 6
	CBP UPPER				IBP + 4
	CBP LOWER				IBP + 2
	(RESERVED) BUSY				IBP
COMMAND BLOCK	COMMAND BUSY				CBP
	LOW SYSTEM MEMORY				

Figure 5a. Initialization Control Blocks

210931-5

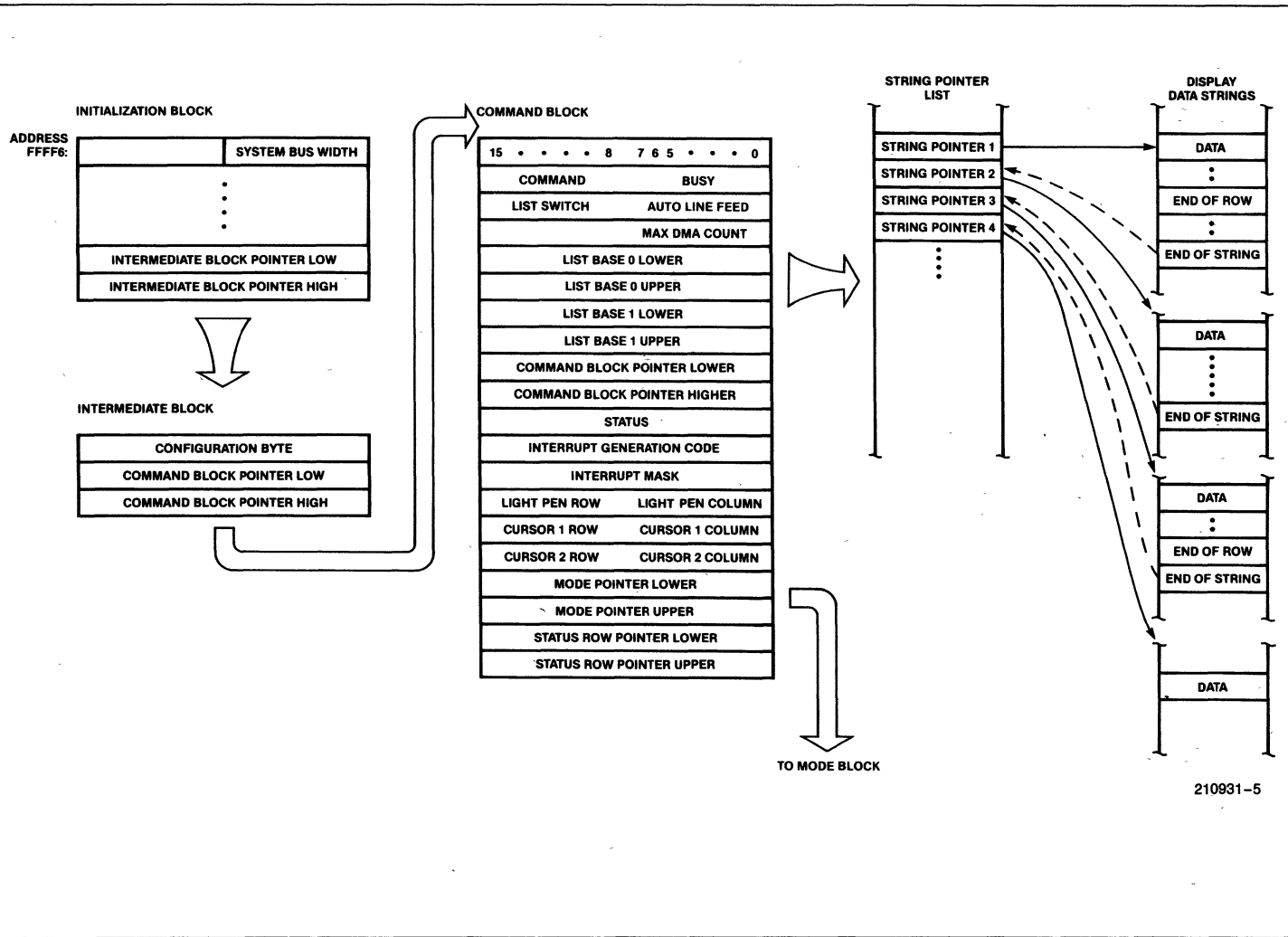


Figure 5b. Control Structure of the 82730

82730 TEST FEATURES

The 82730 has built in Self-Test features that provide testability at the component or at the board level. These features include the test commands and the output pin force capability and are described below.

Output Pin Force Capability

A capability to force logic state (high, low, high impedance) on all output pins is provided in the 82730 Text Co-Processor. This is accomplished by providing a stimulus on pins LC0-LC2 during chip reset. This feature is used for dc parametric tests on the output pins.

The state of pins LC0-LC2 is monitored during chip reset. The state of these pins is latched internally on the falling edge of chip reset. If no external inputs are applied during reset, the state observed will be all 1's and no action will be taken by the 82730. If any external inputs are applied to pins LC0-2 during reset, the resulting action will depend upon the state latched on the falling edge of reset. The 82730 maintains pins LC0-LC2 in high impedance state for the duration of chip reset to avoid contention with external inputs. Also internal pull-ups ensure that a state of all 1's will be detected if no external inputs are applied.

The actions corresponding to each of the observed states of pins LC0-LC2 are summarized in Figure 6a.

State of Pins LC0-LC2 During Chip Reset			Action
LC2	LC1	LC0	
0	X	X	Invoke Stand-Alone Self Test
1	0	0	Force all Outputs to High Impedance State
1	0	1	Force all Outputs to Logic High State
1	1	0	Force all Outputs to Logic Low State
1	1	1	NOP

Figure 6a. Output Pin Forcing and Stand-Alone Self Test Invocation

Stand-Alone "Self Test"

The built-in Self Test capability of the 82730 can be invoked in a stand-alone mode by applying an external stimulus through pins LC0-LC2 during chip reset. This is the same mechanism as the one used for forcing logic states on output pins. Figure 6a.

If pin LC2 is pulled low during chip reset, the 82730 executes a built-in self test. Upon completion of the self-test, a 16-bit signature, generated internally as the test result, is output via pins WDEF, DAT14-DAT0. The completion is signalled by providing a logic "O" output on pin LC3 as a completion flag. The signature will remain on the output pins until the next chip reset. The 82730 will enter an idle state awaiting chip reset and will not respond to any external inputs until a reset signal is applied. During the process of presenting the signature onto WDEF, DAT14-DAT0, the signature will also appear briefly on the AD bus in the form of a bus cycle with two 8-bit accesses to addresses, AAAAH, AAABH. However, this phenomenon is only incidental. Pins WDEF, DAT14-DAT0 should be used for observing the signature.

The stand-alone self test includes the testing of internal address pointer registers. These registers are not tested when the self test is invoked by issuing a "Self Test" command. (See under Channel Commands below). Therefore, the signature generated during stand-alone self test will be different from that generated by the "Self Test" command.

82730 CHANNEL COMMANDS

Table 3. Channel Commands

Command	OP CODE		
1 START DISPLAY	0000	0001	01 H
2 START VIRTUAL DISPLAY	0000	0010	02 H
3 STOP DISPLAY	0000	0011	03 H
4 MODE SET	0000	0100	04 H
5 LOAD CBP	0000	0101	05 H
6 LOAD INTMASK	0000	0110	06 H
7 LPEN ENABLE	0000	0111	07 H
8 READ STATUS	0000	1000	08 H
9 LD CUR POS	0000	1001	09 H
10 SELF TEST	0000	1010	0A H
11 TEST ROW BUFFER	0000	1011	0B H
12 NOP	0000	0000	00 H
13 (RESERVED)	From: 0000	1100	0C H
	To: 1111	1111	FF H

The system processor issues channel commands to 82730 via the Command Block. The processor first checks if the BUSY flag in the command block has been cleared. It should wait for the BUSY flag to be cleared before proceeding with the issuing of a command. When the BUSY flag is cleared, the processor places a command byte in the "COMMAND" location in command block, sets the BUSY flag to a non-zero value and asserts Channel Attention (CA), by activating the CA input to 82730. A Channel Attention should not be issued, if the BUSY flag has not been cleared.

START DISPLAY

0000	0001	CMD Byte
------	------	----------

LISTSWITCH, Auto Linefeed, Max DMA Count and Cursor Position values are fetched from the Command Block and stored internally after this command is received. The BUSY flag is cleared and the normal display process is activated.

The MCU fetches strings of data from the memory, using the parameters LISTSWITCH, LBASE0 and LBASE1. The data fetched is interpreted as datastream commands or character data to be displayed by the Display Generator. The MCU loads the data into one of the two Row Buffers in the CRT controller, while the Display Generator displays the data from the other buffer, the buffers being swapped at the end of the row. Any datastream commands encountered during data fetch are immediately executed.

The display process is continued until it is deactivated by a STOP DISPLAY command or a Reset. Other channel commands can be issued while a display is in progress and they will be executed when CA is found active at one of the periodic samplings at each end of frame.

The DIP (Display in Progress) status bit is set and the VDIP (Virtual Display in Progress) is cleared upon receiving a START DISPLAY command. Both bits are reset upon receiving a STOP DISPLAY command or a Reset.

It is necessary to load in proper mode information through a MODESET command before activating the display. Following Reset, START DISPLAY command will not be executed, i.e., will result in a NOP until a MODESET command has been issued.

START VIRTUAL DISPLAY

0000	0010	CMD Byte
------	------	----------

LISTSWITCH, Auto Linefeed, Max DMA Count and Cursor Positions are fetched from the Command Block and stored internally upon receiving this command. The BUSY flag is cleared and the Virtual Screen display process is activated.

The operation of Virtual Screen display process is similar to that of a regular display process, except for following a different data access mechanism. The parameters LISTSWITCH, LBASE0 and LBASE1 in the command block represent ACCESS SWITCH, ACCESS BASE0 and ACCESS BASE1 respectively, in virtual screen display.

The VDIP (Virtual Display in Progress) status bit is set and the DIP status bit is cleared upon receiving a START VIRTUAL DISP command: Both DIP and VDIP are reset upon receiving a STOP DISPLAY command or a Reset.

START VIRTUAL DISPLAY command will not activate a display and results in a NOP until a MODESET command is issued after a Reset.

STOP DISPLAY

0000	0011	CMD Byte
------	------	----------

The display process is deactivated upon receiving this command. The DIP and VDIP status bit are reset and the BUSY flag is cleared.

This command blanks the display. HSYNC and VSYNC are **not** affected.

MODESET

0000	0100	CMD Byte
------	------	----------

The Mode Pointer contained in command block location (CBP + 30) is used to access the Mode Block and the modes are fetched sequentially and loaded into the corresponding internal registers in 82730. LISTSWITCH, Auto Linefeed, Max DMA Count and Cursor Positions are fetched from the Command Block and stored internally upon completion and the BUSY flag is cleared.

The organization of mode words in the mode block and the parameters supplied by them are shown below (See Figure 10). Some of these parameters which are critical to the operation of a text coprocessor are required to remain unchanged over most of normal operation. No provision is made to prevent MODESET from changing these parameters and it is left to the designer to insure that they are not changed.

The modes provide horizontal and vertical mode display parameters, interlace information, DMA burst and spacing specifications, cursor characteristics as well as attribute enables and bit-selects. Typically, this would be the first command issued after initialization. The Mode Block provides all the parameters needed for a complete initialization of the 82730 for display. Thus a single Moderset command can fully initialize the chip. Note that until the first Moderset command is sent, certain functions such as VSYNC and HSYNC are not enabled.

It is necessary to set up proper mode information, before activating a display. Therefore, a display activating command should not be issued unless proper mode information has been loaded through a MODESET command. START DISPLAY and START VIRTUAL DISPLAY commands will result in a NOP if a MODESET command has not been issued since the most recent Reset.

LOAD CBP

0000	0101	CMD Byte
------	------	----------

The address pointer "NEW CBP" contained in the command block is fetched and stored in the CBP register in the text coprocessor, replacing the old CBP. This effectively moves the command block in the memory. The Command byte from the new Command Block is fetched and the specified channel command is executed. The BUSY flag in the new Command Block is cleared upon completion.

LOAD INTMASK

0000	0110	CMD Byte
------	------	----------

The interrupt mask contained in location "INT-MASK" in the command block is fetched and stored internally in the CRT controller. When a particular mask bit is set, the interrupt is disabled for a status bit in the corresponding bit position. An interrupt is generated by the text coprocessor by activating the SINT pin, if a status bit is 1 and the corresponding bit in the interrupt mask is 0. The BUSY flag is cleared upon completion.

Interrupts can be enabled for the following status bits.

7	6	5	4	3	2	1	0	BIT
—	RDC	RCC	FDE	EOF	DBOR	LPU	DUR	STATUS WORD

- RDC:** Reserved Datastream Command Encountered
- RCC:** Reserved Channel Command Executed
- FDE:** Frame Data Error (Fetching characters past physical End of Frame)
- EOF:** End of "n" frames (Logical end of nth frame)
- DBOR:** Data Buffer Overrun (Row Buffer filled completely without encountering END OF ROW command)
- LPU:** Light Pen Update
- DUR:** Data Underrun (Buffer swap initiated before finishing Row Buf loading)

READ STATUS

0000	1000	CMD Byte
------	------	----------

The internal status register is written to "STATUS" location in the command block. The status register is then cleared, however DIP and VDIP status bits are not cleared. LISTSWITCH, AUTO LINEFEED, Max DMA Count and Cursor Positions are fetched from the Command Block and stored internally. The BUSY flag is then cleared.

STATUS WORD

15-9	8	7	6	5	4	3	2	1	0
—	VDIP	DIP	RDC	RCC	FDE	EOF	DBOR	LPU	DUR

LPEN ENABLE

0000	0111	CMD Byte
------	------	----------

The Light Pen detection process is enabled to search for a rising edge on the LPEN pin. The BUSY flag is then cleared.

If the display process is active and a rising edge is detected on the LPEN input, the corresponding row and column position on the screen is stored internally. At the next end of frame, the LPEN position is written to locations "LPENROW" and "LPENCOL" in the command block and the LPU (Light Pen Update) status bit is set.

If the display process is not active, this command has no immediate effect. However, the LPEN detection process remains enabled and will take effect if a display is activated subsequently.

LD CUR POS

0000	1001	CMD Byte
------	------	----------

The display row and column positions of cursors 1 & 2 as set in locations "CUR1 ROW," "CUR1 COL," "CUR2 ROW" and "CUR2 COL" in the command block are loaded into internal registers in the CRT controller. Also LISTSWITCH Auto Linefeed and Max DMA Count are loaded from the Command Block and the BUSY flag is cleared. This command is used to change the cursors only. Note that the cursor positions are also updated with the execution of other channel commands.

The cursor characteristics for display are defined by the mode. During the display process, a cursor will be displayed accordingly at the position specified above.

TEST COMMANDS

The test commands for the 82730 are issued in the same manner as the normal channel commands. However, the parameters used by test commands are different from those used by the channel commands in normal operation. Therefore, a Test Block which is similar in format to the Command Block is defined. Switching between Command Block and Test Block is accomplished using the "Load CBP" command. The Test Block differs only in the parameter locations associated with the command. The locations for New CPB, command byte and busy flag are the same for both Command Block and Test Block. The "Test Result" location in Test Block corresponds to the "Status" location in Command Block.

The test commands can be executed, following chip reset, only until the first Modeset command is issued. Once a Modeset command has been execut-

ed following chip reset, any subsequent test commands will not be executed and will result in a NOP.

"Self Test" Command

0000	0010	CMD Byte
------	------	----------

A built-in Self test is performed using an internal test pattern. The signature generated during the test is written to the Test Result location (TBP + 18) in the Test Block. The Busy Flag in the Test Block is then cleared. The Self Test command must be immediately preceded by a chip reset in order to ensure a consistent signature.

The Test Block format for issuing the Self Test command is shown in Figure 6b.

"Row Buffer Test" Command

0000	1011	CMD Byte
------	------	----------

The Load Pointer in Test Block is fetched. It points to the system memory area storing the test pattern to be used for testing the on-chip RAM (i.e. - the Row Buffers). The Store Pointer, which points to memory area where the data read back from the RAM will be written, is also fetched from Test Block.

Successive words are fetched from memory and written to the Row Buffer, until it is completely filled. Note that three extra words beyond the maximum Row Buffer capacity will be fetched. If $N = \text{Max Row Buffer capacity}$, $(N + 3)$ words will be fetched from memory. The extra words fetched will be ignored. The Row Buffer contents are then read back and are written to successive locations in memory area pointed to by the Store Pointer. The test is then repeated on the second Row Buffer. Note that the $(N + 4)$ th word in the pattern stored in memory constitutes the first word written to the second Row Buffer. The data storage for the Row Buffer test patterns is illustrated in Figure 6c.

Internally, the Row buffers are 17-bits wide, while the data path is 16-bits wide. During the writing of data to Row Buffers, a complement of bit 15 is written to bit 16 of the Row Buffer in order to test all 17 bits. During the read back, two data words are stored in system memory for each location in the Row Buffer. The first word will consist of bits 0-15 read from the Row Buffer, while the second word will consist of bits 0-14 and bit 16 from the Row Buffer. Thus a total of $4*N$ words will be stored back in system memory as a result of the Row Buffer Test ($2*N$ for each Row Buffer).

	15	8	7	0	
				BIT	
	COMMAND			BUSY	TEST BLOCK POINTER (TBP)
PARAMETER					TBP + 2
LOCATIONS					TBP + 4
NOT USED					TBP + 6
FOR SELF TEST					TBP + 8
COMMAND					TBP + 10
					TBP + 12
	NEW CBP LOWER				TBP + 14
	NEW CBP UPPER				TBP + 16
	TEST RESULT				TBP + 18

**Figure 6b. Test Block Format for "Self Test" Command
(For both 16-bit and 32-bit addressing modes)**

A signature is generated during the test and is written to Test Result location in Test Block upon completion. The BUSY flag in the Test Block is then cleared.

The Test Block format for issuing the Row Buffer Test command is illustrated in Figures 6d.1 and 6d.2. Note that the locations for Load Pointer and Store Pointer parameters are different for 16-bit and 32-bit addressing modes.

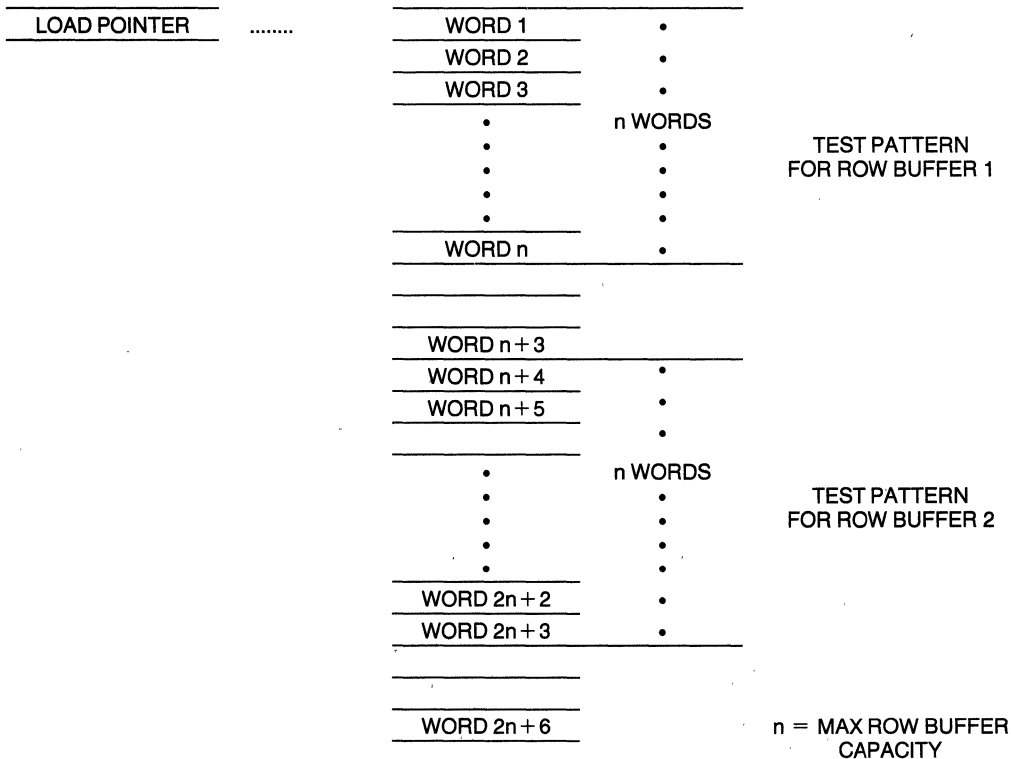


Figure 6c. Data Storage for Row Buffer Test Command

15	8	7	BIT 0	TEST BLOCK POINTER (TBP)
COMMAND		BUSY		
(RESERVED)				TBP + 2
LOAD POINTER LOWER				TBP + 4
LOAD POINTER UPPER				TBP + 6
STORE POINTER LOWER				TBP + 8
STORE POINTER UPPER				TBP + 10
(RESERVED)				TBP + 12
NEW CBP LOWER				TBP + 14
NEW CBP UPPER				TBP + 16
TEST RESULT				TBP + 18

Figure 6d.1 Test Block Format for "Row Buffer Test" Command (32-bit addressing mode)

15	8	7	BIT 0	TEST BLOCK POINTER (TBP)
COMMAND		BUSY		
(RESERVED)				TBP + 2
(RESERVED)				TBP + 4
LOAD POINTER				TBP + 6
(RESERVED)				TBP + 8
STORE POINTER				TBP + 10
(RESERVED)				TBP + 12
NEW CBP LOWER				TBP + 14
NEW CBP UPPER				TBP + 16
TEST RESULT				TBP + 18

Figure 6d.2 Test Block Format for "Row Buffer Test" Command (16-bit addressing mode)

NOP

0000	0000	CMD Byte
------	------	----------

LISTSWITCH, Auto Linefeed, Max DMA Count, and Cursor Positions are fetched from the command block and stored internally as in all other channel commands. The Busy flag is then cleared.

acter data by the command bit. The most significant bit (MSB) of each data word is designated as the command bit. If the command bit is "1", the lower 15 bits of the data word are interpreted as a datastream command, while if the command bit is "0" the lower 15 bits (or 7 bits if DTW16 = 0) are interpreted as character data.

82730 DATASTREAM COMMANDS

Datastream Commands

Datastream commands are commands embedded in the data fetched from memory by the data access task. These commands are differentiated from char-

Datastream Command Operation

During the data access task, the Micro Controller Unit (MCU) examines the command bit of each data word fetched. If the command bit is 1, it executes the datastream command specified in the data word. Otherwise, it stores the lower 15 bits of the data word in the Row Buffer as character data. This process is repeated for each data word fetched.

Datastream Command List

Table 4. 82730 Datastream Commands

	Command	Command Code				OP Code
		Op Code		Parameters		
1	ENDROW	1000	0000	XXXX	XXXX	80
2	EOF	1000	0001	XXXX	XXXX	81
3	END OF STRING & END OF ROW	1000	0010	XXXX	XXXX	82
4	FULROWDESCPT	1000	0011	"n"		83
5	SL SCROLL STRT	1000	0100	XXX SCR LINE		84
6	SL SCROLL END	1000	0101	XXX END LINE		85
7	TAB TO n	1000	0110	"n"		86
8	LD MAX DMA COUNT	1000	0111	COUNT		87
9	ENDSTRG	1000	1000	XXXX	XXXX	88
10	SKIP n	1000	1001	"n"		89
11	REPEAT n	1000	1010	"n"		8A
12	SUB SUP n	1000	1011	"n"		8B
13	RPT SUB SUP n	1000	1100	"n"		8C
14	SET GEN PUR ATTRIB	1000	1101	GPA OP		8D
15	SET FIELD ATTRIB	1000	1110	XXXX	XXXX	8E
16	INIT NEXT PROCESS (Command process command)	1000	1111	XXXX	XXXX	8F
17	(RESERVED)	10XX	XXXX	XXXX	XXXX	90-BF
18	NOP	11XX	XXXX	XXXX	XXXX	C0-FF

Datastream commands can be used for changing Row Characteristics on a row by row basis, for carrying out editing functions and for formatting data into rows and frames. These commands are executed by the MCU immediately after they are encountered. As a convenience for the user, the set of all possible command codes starting with "1" in the two most significant bits has been designated as NOP commands. The user can use these command codes for any desired purpose. All other command codes which are not presently defined, are reserved for future expansion and should not be used by the user. The currently undefined codes cause the RDC (Reserved Datastream Command) status bit to be set and also generate an interrupt, if enabled. Reserved command codes should not be used.

The preceding commands are recognized as valid datastream commands. The corresponding command codes are also indicated. It should be noted that the most significant bit of the command bit is always 1, in order for the word to be interpreted as command.

The "Init Next Process" command can be issued only through a command process in Virtual Screen Display. It is included in this list because its operation is analogous to a datastream command in a virtual screen access environment. Also, in virtual screen display certain datastream commands are interpreted differently, depending upon whether

they are encountered in a process datastream or as command process commands. When a command is ignored (becomes a NO-OP) in a virtual display, any parameters that are associated with it are also ignored. The command process command operation is discussed separately. The operation of all other datastream commands is described below.

ENDROW

15	14	8	7	0
1	000	0000	XXXX	XXXX

This command signifies that no more characters will be loaded in the Row Buffer for this row and an End of Row indicator is stored accordingly. When the row currently being loaded is displayed, the Display Generator (DG) will blank the screen from the end of row character position until the physical end of row.

The Micro Controller Unit (MCU) stops fetching data and waits for DG to swap the Row Buffers. The data access task is resumed following the buffer swap. If a physical end of frame is reached while the MCU is waiting for a buffer swap the MCU ceases to wait and executes an EOF (End of Frame) command.

In virtual display, this command is interpreted as a VEOR (Virtual End of Row) if encountered in a virtual process datastream.

VEOR

ENDROW command in a virtual process datastream is interpreted as VEOR (Virtual End of Row) and it terminates a virtual row. The current LPTR is stored in the process header addressed by the "Process Addr" register. The Max Count register is also stored in the Max DMA Count location in the process header. Similarly, the Field Attribute Mask is also stored in the header. In addition, in auto linefeed mode (ALF = 1) other parameters characterizing the process state are also saved in the header. The "Process Addr" register is loaded with the address of the header of the next process fetched from the Access table. The "Access Tab Addr" register is post-incremented by two if a 16-bit addressing option is used and by four if 32-bit addressing is used. The data access task is then resumed for the next process.

EOF

15	14	8	7	0
1	000	0001	XXXX	XXXX

This command (End of Frame) signifies that no more characters will be loaded in the Row Buffers for this frame. The Micro Controller Unit (MCU) stops fetching data words and waits for the physical end of frame. If a virtual display is in progress, this command is interpreted as VEOS (Virtual End of Frame), if encountered in a virtual process datastream.

The Display Generator (DG) swaps the row buffers at the end of the current display row and starts displaying the row containing the EOF command. When the character preceding the EOF command is displayed, the DG blanks the screen until the physical end of frame. The MCU fetches the Status Row data then waits until its display is completed. It then performs the actions described below.

If LPEN has been enabled and a rising edge on the LPEN input has been detected, the LPENROW and LPENCOL positions in the command block are updated and the LPU status bit is set. If a Channel Attention has occurred, i.e., if CA has been activated, the command byte is fetched from command block and the specified channel command is executed. If the command issued is a "Stop Display" command, the MCU will terminate the display process and wait for the next channel attention. Otherwise, the MCU resumes the data access task by reinitializing pointers for the new frame and continues to fill the Row Buffers.

VEOF

EOF command in a virtual process datastream is interpreted as VEOF (Virtual End of Frame). It provides for reinitialization of LPTR using LISTSWITCH, LBASE0 AND LBASE1 for each process, analogous to the automatic reinitialization of LPTR at each end of frame in a Normal Display.

LPTR for the current process is reinitialized using LISTSWITCH, LBASE0 and LBASE1 contained in the process header. The End of Display (EOD) bit in the header is set to 1. The current process is terminated as in a VEOR and the next process in Access Table is accessed.

EOL

15	14	8	7	0
1	000	0010	XXXX	XXXX

The EOL (End of Line) command has a combined effect of NXTRROW and NXTSTRG commands. All the actions performed in a END OF ROW command are carried out. In addition a END OF STRING command is executed before resuming the data access task. Thus, following the end of row, the data access is continued with the next data string. In virtual process datastream, this command has the combined effect of VEOR and END OF STRING.

FULROWDESCRPT

15	14	8	7	0
1	000	0011	N	

The next "n" words fetched from memory are loaded into the Row Characteristics holding registers. "n" is specified by the lower order byte of the command word and should be between 0 and 7.

The parameters loaded by this command will be used to define the row characteristics at the time the row currently being loaded is displayed. The data words defining these characteristics which follow the FULROWDESCRPT command must be ordered and organized in memory in a specific format. The format for FULROWDESCRPT parameters is shown below in Figure 6e starting with "Lines Per Row" as the first parameter loaded.

This command will be ignored if encountered in a virtual process datastream. The MSB of all the parameters must be zero for proper operation in virtual display.

	Upper Byte								Lower Byte							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						RVV	BLK	DBL	W							
Lines Per Row	—	—	—		ROW	ROW	HGT	DEF	—	—	—		LPR			
Normal Start/Stop	—	—	—			NRMSTRT			—	—	—		NRMSTOP			
Superscript Start/Stop	—	—	—			SUPSTRT			—	—	—		SUPSTOP			
Subscript Start/Stop	—	—	—			SUBSTRT			—	—	—		SUBSTOP			
Cursor 1 Start/Stop	—	—	—			CUR1	STRT		—	—	—		CUR1	STOP		
Cursor 2 Start/Stop	—	—	—			CUR2	STRT		—	—	—		CUR2	STOP		
Underline Line Selects	—	—	—			UL2	LINE	SEL	—	—	—		UL1	LINE	SEL	

RVV ROW, when this bit is set the CRVV pin will be inverted for the next full row.
 BLK ROW, when this bit is set the row will be blanked (BLANK high).
 DBLHGT, when the double height bit is set, all character are displayed with twice the scan lines per row.
 WDEF, when the width defeat bit is set, the WDEF pin is activated for the entire row.
 The following can be programmed from 0 to 31 yielding a range of 1 to 32 lines.
 LPR specifies number of lines per row.
 NRMSTRT, SUPSTRT, SUBSTRT specify line numbers in a display row which mark the start of normal, superscript and subscript characters respectively.
 NRMSTOP, SUBSTOP, SUBSTOP specify line numbers in a row where normal, super script and subscript characters end respectively.
 CUR1 STRT, CUR2 STRT specify the starting line numbers in a row for cursor 1 and cursor 2 respectively.
 ULINE1 SEL, ULINE2 SEL specify the line numbers in a row where underline 2 will appear respectively.
 All FULROWDESCRPT parameters affect the row in which they are programmed and stay in effect until changed by another FULROWDESCRPT command.

Figure 6e. Format for FULROWDESCRPT

SL SCROLL STRT

15	14	8	7	5	4	0
1	000	0100	XXX	SCR	LINE	

SL SCROLL END

15	14	8	7	5	4	0
1	000	0101	XXX	END	LINE	

The Slow Scan register in 82C3 is loaded with the scroll line specified by the five least significant bits of the command word. When the row currently being loaded is displayed, the line count for that row will start with the value specified by the Slow Scan register. A "Margin" (MGN) parameter, loaded by MODESET, specifies the number of blank lines plus one to be added at the top of the slow scroll field on the screen. This ensures the availability of sufficient DMA time for fetching the next row, when only a small number of scan lines are displayed in the top row of slow scroll window. This command is used for starting a slow scroll. (Note: MGN = 0 results in no margin buffer lines.)

This command will be ignored if encountered in a virtual process datastream or if a SL SCROLL END command is encountered later on the same row.

The scroll location in row characteristics holding registers is loaded with the number of lines specified by the five least significant bits of the command word. This number specifies the number of lines to be displayed when the row currently being loaded is displayed. This is used instead of the regular LPR (Lines Per Row) characteristics, for this particular row. This command is used in the last row of a slow scroll for terminating a slow scroll. The Margin (MGN) parameter, loaded by MODESET, is used in the same way as in slow scroll start except that the specified number of blank lines are inserted at the bottom of the slow scroll in this case. This command will be ignored if encountered in a virtual process datastream or if followed by a SL SCROLL STRT on the same row.

TAB TO n

15	14	8	7	0
1	000	0110	"n"	

The lower byte of the command word specifies the column (RCLK count) after SYNCSTRT at which a Tab should occur. At display time, after the character preceding the Tab command is displayed, the screen is blanked until the RCLK count specified by the command ("n") is reached. After reaching the specified count, display is resumed by displaying the character following the TAB command.

If the RCLK count specified by the Tab command has already occurred before beginning the blanking for Tab, the display will be blanked until the end of the row.

This command is ignored, if encountered in a virtual display process datastream.

LD MAX DMA COUNT

15	14	8	7	0
1	000	0111		MAX COUNT

The Max Count register in 82730 is loaded with the Max DMA Count specified by the lower byte of the command word. The DMA Counter is also reinitialized with the Max Count value in the Command Block after all channel commands.

MAX DMA Count is programmable in the range of 1 to 256 (MAX COUNT value 0 equals 256). However, counts greater than the row buffer capacity will cause row buffer overruns if the data strings depend on MAX DMA to terminate the fetching.

The DMA counter is decremented for each data word as the Row Buffer is being loaded. Datastream commands and words supplying parameters for datastream commands as in FULROWDESCRPT, are not counted. Superscript/Subscript characters are counted in pairs, i.e., a pair of characters causes only one count.

In virtual screen display, every time a new process is accessed, the DMA counter is initialized with the Max DMA Count contained in the process header. This value is also stored in a Max Counter register.

At virtual end of row (VEOR) the Max Count register is written to the process header. The "LD Max DMA Count" command is ignored if encountered in a virtual process datastream.

ENDSTRG

15	14	8	7	0
1	000	1000	XXXX	XXXX

The SPTR register in the 82730 is loaded with a new String Pointer (SPTR) value fetched from the memory location indexed by the List Pointer (LPTR), which is stored in the LPTR register. The LPTR register is incremented by two if a 16-bit addressing option is used and by four if 32-bit addressing is used. When more than one 82730 is connected in a cluster, each of them adds an offset, determined by its position in the cluster, to the pointer fetched from memory, before storing it in its SPTR register.

This command directs the data access to the next data string in the list of strings indexed by LPTR. The operation of this command is identical for a Virtual or Normal Display. In virtual display, the next data string within the current display process is accessed.

SKIP n

15	14	8	7	0
1	000	1001	n	

The next "n" data words fetched from memory are ignored. "n" is specified by the lower byte of the command word and is programmable from 0 to 255. If n equal to 0 is specified, no words are skipped. Any datastream commands encountered in the data fetch are not counted towards these n words. Also parameters following the datastream command as in FULROWDESCRPT are not counted. All embedded datastream commands are executed with the following exceptions.

If a Tab To N data stream command is encountered during the execution of a Skip N command, the Tab command will result in a NOP, i.e. a Tab embedded in the data to be skipped will be ignored.

If an EOL (End Of Line) data stream command is encountered during the execution of a Skip N command, it will be executed with the following effect. In non-auto line feed mode, (ALF = 0) the EOL command has the combined effect of End Of Row and End Of String commands. In auto line feed mode, (ALF = 1) the EOL command has the effect of an End Of String command only.

If the data words skipped include any superscript-subscript characters, they are skipped in pairs and a pair of characters is counted as only one count in "n". If another skip command is encountered its value of "n" is added to the present skip count and skipping continues.

REPEAT n

15	14	8	7	0
1	000	1010	n	

The next data word (byte, if DTW16 = 0) fetched from memory is stored in the Row Buffer “n” times, where “n” is specified by the lower byte of the command word, “n” is programmable from 0 to 255. If n equal to 0 is specified no repetitions will occur, and the word following the Repeat n command will be ignored. This character will eventually be displayed n times. The DMA counter is also made to count n times. In non-auto linefeed mode (ALF = 0), reaching Max DMA Count before the n repetitions are completed will result in a termination of the Repeat n command. This command will also be terminated if the Row Buffer gets filled completely before the n repetitions are completed.

It should be noted that the data word immediately following the Repeat n command is treated as character data, irrespective of the value of its command bit.

SUP/SUB n

15	14	8	7	0
1	000	1011	n	

The next “n” pairs of data words (bytes, if DTW16 = 0) fetched from memory are treated as superscripts or subscript characters. “n” is specified by the lower byte of the command word. These n pairs are assumed to be ordered with the superscript preceding the subscript.

No datastream commands are permitted in the 2n words following this command. All of these words are interpreted as superscript-subscript pairs. The DMA counter is made to count only once for each pair of characters. In non-auto linefeed mode (ALF = 0), reaching the Max DMA Count will result in a termination of this command. If n equal to zero is specified, no action will result.

RPT SUB/SUP n

15	14	8	7	0
1	000	1100	n	

The operation of this command is similar to that of the “Repeat n” command except that the pair of characters following the “RPT SUB/SUP n” com-

mand is repeated n times. “n” is specified by the lower byte of the command word and is programmable from 0 to 255. If n equal to zero is specified, no repetitions will occur, and the two data words following the “RPT Sub/Sup n” command will be ignored. The two data words (bytes, if DTW16 = 0) immediately following the command word are interpreted as a superscript-subscript pair and are repeated. The DMA counter is made to count only once for each repetition of the pair. In non-auto linefeed mode (ALF = 0), reaching Max DMA Count prior to completion of n repetitions will cause a termination of this command.

SET GEN PUR ATTRIB

15	14	8	7	0
1	000	1101	GPA	OPERAND

This command provides control over the output pins assigned to General Purpose Attributes, GPA 1 through GPA4.

The GPA in the Process Header is updated each time a SET GPA command is executed. Thus the GPA state in the header is updated to reflect any changes caused by the “Set Gen Pur Attrib” command. The GPA command occupies a character space on the screen. Consequently, a GPA command is counted as a character towards MAX DMA count. However, a GPA command nested in a Skip N or a TAB to N command is skipped, i.e., it has no effect.

The encoding of the operand, specifying GPA operation, is shown below.

SET FIELD ATTRIB

15	14	8	7	0
1	000	1110	XXXX	XXXX
0	FIELD ATTRIBUTE MASK			

The word following this command is fetched. This word is used as a Field Attribute Mask in storing all subsequent display data words in row buffer. The bits in the data words fetched from memory corresponding to the bit positions containing a “1” in Field Attribute Mask are all set to 1 before storing the data word in row buffer. The Field Attribute Mask is used on all display data words fetched from memory. The mask register will contain all 0’s upon reset and is cleared at the beginning of each frame.

NOP

15	14	8	7	0
1	1XX	XXXX	XXXX	XXXX

No action is taken. The data access task is resumed by fetching the next data word.

	7	6	5	4	3	2	1	0
GPA	GPA4	GPA4	GPA3	GPA3	GPA2	GPA2	GPA1	GPA1
OPERAND	DATA	EN	DATA	EN	DATA	EN	DATA	EN

ENCODING	GPAx		FUNCTION
	DATA	EN	
	0	0	ROW BUFFER DATA
	1	0	ROW BUFFER DATA
	0	1	GPA DATA = 0
	1	1	GPA DATA = 1

Datastream Command Conventions

The reaching of Max DMA Count, encountering of terminating commands such as ENDROW, EOF, etc. and occurrences of these while executing a "skip n" command give rise to various possible combinations of events. The behavior of 82730 under these circumstances is described below:

- i) When Max DMA Count is reached, it has the effect of a VEOR command if a Virtual Display is in progress or a ENDROW command if a Normal Display is in progress. It also causes an automatic end of string i.e., the effect of a NXTSTRG command in non-auto linefeed mode (ALF = 0).
- ii) In non-auto linefeed mode, "Repeat n", "Sub/Sup n" and "RPT Sub/Sup n" commands are terminated upon reaching a max DMA count, even if "n" is not reached.
- iii) "Skip n" command is terminated if EOF command is encountered. It is also terminated upon encountering a ENDROW command in non-auto linefeed mode (ALF = 0).
- iv) "Repeat n" "Sub/Sup n" and "RPT Sub/Sup n" commands can be nested within a "Skip n" command. If superscript-subscript characters are skipped, each pair of characters counts as one skipped character. If the above commands are encountered during a "Skip n" and if the specified count (n) in these commands is not reached by the end of execution of the "Skip n" command, the execution of the nested command is continued beyond the termination of "Skip n" command until the remaining portion of the count specified in the nested command is completed.

VIRTUAL SCREEN MODE

Command Process Commands

In Virtual Screen Display, 82730 accesses display processes and command processes through the Access table. The command processes enable the I/O Driver process to direct 82730 to execute certain datastream commands by inserting an appropriate command process address in the Access table. This capability enables the preservation of uniformity and consistency of operation between normal and virtual environments, by assigning different interpretations to the command according to the access environment. It is especially useful for termination and initialization commands. The operation of command process commands is analogous to that of datastream commands except for a different access environment.

Command Process of Command List

The commands allowed in command processes can be divided into two subsets. The first subset consists of commands that can be issued only through a command process, while the second one consists of normal datastream commands that can also be issued through a command process. The command code for a datastream command issued through a command process is the same as that for the normal datastream command embedded in the data. However, certain datastream commands are interpreted differently when they are issued through a command process as opposed to embedding in the datastream of a virtual display process. The most significant bit (MSB) of the command word must be a "1". In the datastream, this bit distinguishes a command word from character data. In the process environment, this bit distinguishes a command process from a display process. The commands permitted in command processes are listed below. No other commands will be recognized if encountered in a command process and will result in a NOP. All undefined command codes apart from those designated as NOP are reserved and should not be used. Encountering an illegal command code causes the RDC (Reserved Datastream Command) status bit to be set and will generate an interrupt, if enabled.

Table 5. Command Process Command List

Command	Interpretation In Virtual Process Datastream	Command Code				OP Code
		Op Code		Parameters		
Command Process Only Command:						
1 INIT NEXT PROCESS	NOP	1000	1111	XXXX	XXXX	8F
Command Process or Datastream Commands:						
2 ENDROW	VEOR	1000	1000	XXXX	XXXX	80
3 EOF	VEOR	1000	0001	XXXX	XXXX	81
4 EOL	VEOR + NXTSTRG	1000	0010	XXXX	XXXX	82
5 FULROWDESCRPT	NOP	1000	0011		"n"	83
6 SL SCROLL STRT	NOP	1000	0100	XXX	"SCR LINE"	84
7 SL SCROLL END	NOP	1000	0101	XXX	"END LINE"	85
8 TAB TO n	NOP	1000	0110		"n"	86
9 LD MAX DMA COUNT	NOP	1000	0111		"COUNT"	87
10 (RESERVED)	RESERVED	10XX	XXXX	XXXX	XXXX	90-BF
11 NOP	NOP	11XX	XXXX	XXXX	XXXX	C0-FF

INIT NEXT PROCESS

15	14	8	7	0
1	000	1111	XXXX	XXXX

This command can be used only in a command process to initiate a virtual display "window".

Upon receiving this command, the command process is terminated and the next process in Access Table is accessed by fetching the new process address. However, the LPTR register is not directly loaded from the LPTR location in the process head-

er. Instead, LISTSWITCH in the process header is examined and LPTR is initialized with the value LBASE 0 or LBASE 1 depending upon whether LISTSWITCH is 0 or 1 respectively. Both LBASE0 and LBASE1 are contained in the header.

The process header format is shown in Figure 7. Also the End of Display Bit (EOD) in the header is reset.

The data access task for a virtual display is then resumed, with this value of LPTR.

	15	14	13	8	7	6	0	LOCATION
LS: LISTSWITCH ALF: AUTO LINE FEED	0	—			EOD	—		PROCESS ADDR
			—		LS ALF			PROC ADDR + 2
			—		MAX DMA COUNT			PROC ADDR + 4
					LBASE0 LOWER			PROC ADDR + 6
					LBASE0 UPPER			PROC ADDR + 8
					LBASE1 LOWER			PROC ADDR + 10
					LBASE1 UPPER			PROC ADDR + 12
	1	—			GPA			PROC ADDR + 14
	1				FIELD ATTRIBUTE MASK			PROC ADDR + 16
					LPTR LOWER			PROC ADDR + 18
				LPTR UPPER			PROC ADDR + 20	
SAVE AREA					SPTR LOWER			PROC ADDR + 22
					SPTR UPPER			PROC ADDR + 24
	RPT S/S	S/S	RPT	—	—	REPT COUNT		PROC ADDR + 26
	1					REPT CHAR		PROC ADDR + 28
	1					REPT CHAR 2		PROC ADDR + 30

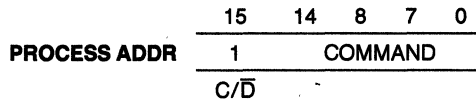


Figure 7. Process Header for Display and Command Process

ENDROW

15	14	8	7	0
1 000 0000 XXXX XXXX				

The actions performed by a ENDROW datastream command in a Normal Display are carried out. The next process in Access Table is accessed and the data access task is resumed, after the next Row Buffer swap.

EOL

15	14	8	7	0
1 000 0010 XXXX XXXX				

This command is identical to ENDROW command in Virtual Display in Command Process environment. ENDSTRG, which is strictly a data operation within a display process is meaningless in the command process environment.

EOF

15	14	8	7	0
1 000 0001 XXXX XXXX				

The actions performed by an EOF (End of Frame) datastream command in a Normal Display are carried out.

FULROWDESCRPT

15	14	8	7	0
1 000 0011 "n"				

The actions performed by the FULROWDESCRPT datastream command are carried out. The data access task is resumed by accessing the next process in the Access Table.

SL SCROLL STRT

15	14	8	7	5	4	0
1	000	0100	XXX	"SCR LINE"		

The same actions as the SL SCROLL STRT data-stream command. The data access is resumed with the next process in Access Table.

SL SCROLL END

15	14	8	7	5	4	0
1	000	0101	XXX	"END LINE"		

The actions performed by a SL SCROLL END data-stream command, in a Normal display, are carried out. The data access task is resumed with the next process in Access Table.

TAB TO n

15	14	8	7	0	
1	000	0110	"n"		

The effect of this command process command is identical to that of the TAB TO n datastream command. The TAB can be used to establish the left edge of a virtual display "window".

LD MAX DMA COUNT

15	14	8	7	0	
1	000	0111	MAX COUNT		

The Max Count register on 82730 is loaded with the value specified by the lower byte of the command word. The DMA counter is also initialized with this Max Count Value.

The next process in the Access Table is accessed. However, the Max DMA Count value in the process header is not used for initializing the DMA counter. Instead, the DMA counter as initialized by the LD Max DMA Count command is used for this process. The virtual display data access task is then resumed normally. When the process is terminated, the new Max Count value is written to the process header. Thus the Max Count value in the header is updated as a result of this command.

NOP

15	14	8	7	0
1	1XX	XXXX	XXXX	XXXX

No action is taken. Data access task is resumed by fetching the next process address from Access Table.

ERROR AND STATUS HANDLING

Error Conditions

Since the MCU and DG function asynchronously with respect to each other, different relative timings in MCU and DG operation are possible, some of which result in error conditions. The lack of appropriate termination commands for row or frame data in the datastream also gives rise to certain error conditions. These types of situations occurring in display process operation are described below.

In normal operation, DG initiates a buffer swap at the physical end of a display row. If the MCU has not finished loading its row buffer by that time, a "Data Underrun" occurs. This results in blanking of the screen until physical end of frame by DG and execution of an EOF (End of Frame) command by MCU. Data underrun also occurs when the first row of the frame has not finished loading by the start of the character field. The entire frame will be blanked in this case.

If a physical end of frame is reached prior to encountering an EOF datastream command, a "Frame Data Error" occurs, which results in the execution of an EOF command by MCU. (Note that this does not disrupt the visible display action, and may not constitute an error for certain data structures. The error indication is included as a flag where knowledge of this condition is desired.) Similarly, when the MCU fills up a row buffer completely, without encountering an ENDROW command, the "Data Buffer Overrun" flag is set.

All of the above conditions result in the setting of an appropriate status bit and generation of an interrupt if the corresponding interrupt has been enabled.

Status and Interrupt Handling

A status word is maintained in an internal register by 82730 and it is written to the "STATUS" location in command block when the "Read Status" channel command is executed. The processor can thus read status information by issuing this command. The processor can also enable interrupts for certain status bits by specifying an interrupt mask which is loaded in 82730 as a result of a "Load Int Mask" channel command. This establishes a communication mechanism between 82730 and the processor for error and status reporting.

Status Word

The format for the status word is shown below. The function of each of the status bits is described below.

The status bits get set under the conditions described above. Interrupts can be enabled for all status bits except DIP and VDIP bits. The interrupt status bits are cleared at the beginning of each new display field. DIP and VDIP bits are cleared only after receiving a "STOP DISPLAY" command or a Reset.

All status bits are cleared by a Reset.

15	9	8	7	6	5	4	3	2	1	0
(RESERVED)	VDIP	DIP	RDC	RCC	FDE	EOF	DBOR	LPU	DUR	

VDIP: Virtual Display In Progress

DIP: Display In Progress

RCC: Reserved Channel Command

RDC: Reserved Datastream Command

FDE: Frame Data Error

DUR: Data Under Run

This status bit is set by Display Generator if the Microcontroller Unit (MCU) has not finished loading its Row Buffer when the DG initiates a buffer swap at the physical end of a display row. This condition is defined as data underrun and causes the MCU to execute an EOF command and the DG to blank the screen until the physical end of frame.

LPU: Light Pen Update

This status bit is set by the MCU after updating the LPENROW and LPENCOL locations in command block. The detection of LPEN input is enabled by the LPEN ENABLE channel command. The detection of a rising edge on the LPEN input causes the current row and column position to be stored internally. The MCU updates the LPEN ROW and LPEN COL locations in command block at the next end of frame and sets the LPU status bit. Further updates of these command block locations are inhibited until another LPEN ENABLE command is issued.

EOF: End of Frame

DBOR: End of Row

LPU: Light Pen Update

DUR: Data Under Run

DBOR: Data Buffer Over Run

This status bit is set when the MCU tries to fill a row buffer beyond its capacity. The MCU will stop fetching characters after this point and the display is blanked following the completion of the row currently being displayed.

EOF: End of Frame

This bit is set by the DG at the physical end of the nth frame, where 'n' is specified by the MODESET parameter FRAME INTERRUPT COUNT. This provides the means for timing frame related events such as slow scrolls.

FDE: Frame Data Error

This status bit is set by the DG at the physical end of frame if no EOS datastream command has been encountered until then. This also results in the execution of the EOS command by the MCU.

RCC: Reserved Channel Command

This bit is set by the MCU upon encountering an illegal datastream or command process command. This can be used to trap software errors during program development.

RDC: Reserved Datastream Command

This bit is set by the MCU upon encountering an illegal datastream or command process command. This can be used to trap software errors during program development.

DIP: Display In Progress

This bit is set by the MCU immediately after receiving a "Start Display" channel command. It remains set as long as the display process is active and is reset upon receiving a "Start Virtual Display" or "Stop Display" command or a Reset. Interrupts cannot be enabled for this status bit.

VDIP: Virtual Display In Progress

This bit is set by the MCU immediately after receiving a "Start Virtual Display" channel command and is reset upon receiving a "Start Display" or "Stop Display" command or a Reset. This bit remains active as long as the virtual display process is active. Interrupts cannot be enabled for this status bit.

Interrupt Processing

The system processor can enable interrupts on any of the status bits, with the exception of DIP and VDIP bits, by specifying an interrupt mask. A "1" in a bit position in the interrupt mask disables (masks out) interrupts on the status bit located in the corresponding bit position in the status word. The format for Interrupt Mask is shown below. The Int Mask can be loaded into 82730 from the INTMASK location in command block by a "Load Int Mask" channel command.

15	7	6	5	4	3	2	1	0
(RESERVED)		RDC INT MASK	RCC INT MASK	FDE INT MASK	EOF INT MASK	DBOR INT MASK	LPU INT MASK	DUR INT MASK

INT MASK = 0 Enables the corresponding interrupt.

INT MASK = 1 Masks or disables the corresponding interrupt.

Figure 8. Interrupt Mask

If the interrupt is enabled for a particular status bit by programming a "0" in the corresponding bit position in INTMASK and if the status bit gets set during the course of the display, an interrupt will be generated by 82730 at the next end of frame. At the end of frame, the 82730 will first perform the tasks of updating LPEN position (if required) and servicing the Channel Attention (if CA was activated). Then the status word in the internal register will be written to the INT GENERATION CODE location in the Command Block and the SINT output will be activated. The SINT pin is not deactivated until an interrupt reset signal is received at the IRST pin.

82730 continues to perform its normal display task activating the SINT pin. If no interrupt reset is received until the next end of frame then any new interrupts that might have been generated at that end of frame will be lost. Therefore, it is essential for the system processor to issue an interrupt reset within a frame time after an interrupt is generated.

When the display is not activated, the only interrupt that can occur is the Reserved Channel Command interrupt. Upon receiving an invalid channel command, 82730 will write the status word to INT Generation Code location in the Command Block and activate SINT output, if that interrupt is enabled.

The processor can use the interrupt capability to get status information from 82730. A possible interrupt service routine for the system processor is shown in flow chart form in Figure 9.

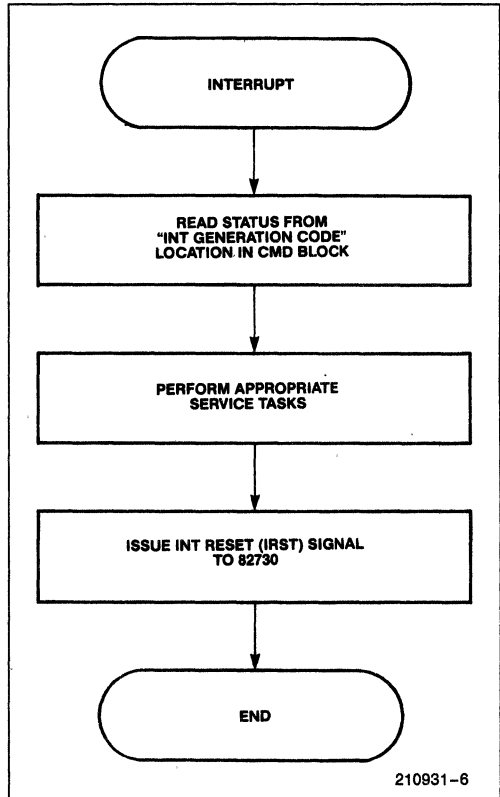


Figure 9. Interrupt Service Routine For System Processor

82730 VIDEO INTERFACE

The Mode Pointer in the Command Block points to a parameter block containing the Mode information required for the display. The organization of the mode words in the Mode Block is shown below.

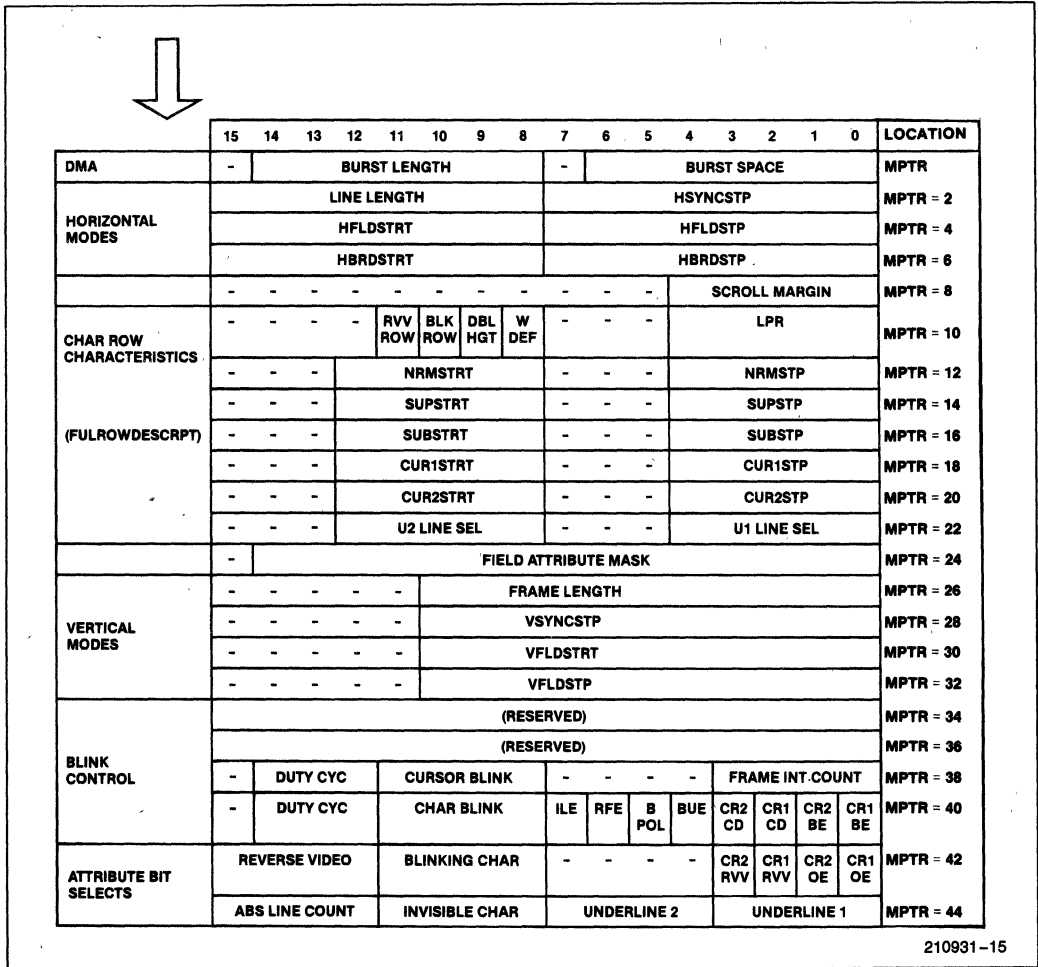


Figure 10. Mode Block Organization

CAM ARRAYS

Three Content Addressable Memory arrays are used for generating timing parameters to control the video display: the HORIZ MODE CAM, the VERT MODE CAM and the CHAR ROW CAM. The user has the flexibility to define his own timing parameters by loading them into the CAM arrays via the MIU. All of these parameters can be modified at the end of every frame. All the parameters in the CHAR ROW CAM, except MARGIN, are changeable on a row by row basis. Each of the three CAM arrays is described separately below:

Timing Sources

RCLK and CCLK inputs are provided by the external video logic to the 82730. The RCLK is used to increment the HORIZ COL CNTR and hence generates all horizontal timing parameters. CCLK is used to clock the character and attribute data output from the 82730 to the external display dot logic. Data changes on the positive going edge of RCLK or CCLK.

Initialization

Upon activation of the RESET input, the 82730 display generator will stop all operations in progress and deactivate all outputs. It will stay in this quiescent state until the MIU executes the MODESET command. The following table shows the states of all the Display Generator outputs during and after RESET.

Pin Name	Condition
DAT0-14	Low
WDEF	Low
LC0-4	High
BLANK	Low
CSYNC	High
$\overline{\text{CHOLD}}$	High
HSYNC	Low
VSYNC	Low
CRVV	Low
RRVV	Low

After reset of the 82730, the CAM arrays are in undetermined states. The CAM arrays are set upon the execution by the MIU of the MODESET command. The HORIZ and VERT MODE CAM contents are especially critical since they are used to generate timing control signals to the external video logic. Without the generation of the timing signals, no display process can take place. Hence, START DISPLAY command cannot be executed before the first MODESET command after the device reset. The START DISPLAY command will be ignored if it precedes the MODESET command.

The row buffers also contain unknown information after power up and reset. In executing the START DISPLAY command, the MIU would first load the two row buffers with the first two rows of character data to be displayed. Upon completion of loading of both buffers, it will signal the DG to begin the display process. In this way, only valid character data will be output to the external video logic.

Timing Parameters

The timing parameters read from the MODESET Block and stored in the VERT MODE CAM and HORIZ MODE CAM are used to control the video display and they can be best illustrated in the following Map of Timing Parameters. All of these timings have to be defined after power up and reset and can be changed on a frame by frame basis during display.

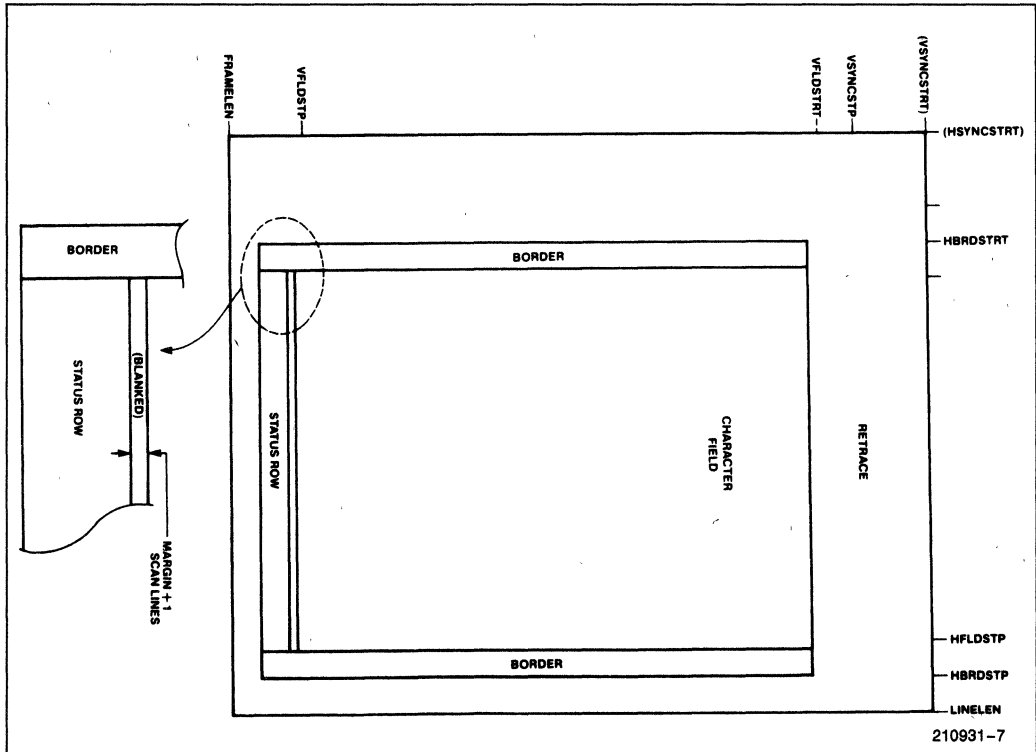


Figure 11. Timing Parameters

Row Timing Parameters

The row timing parameters are stored in **HORIZ MODE CAM** and are programmable from 0 to 255 RCLK times. These parameters are:

- a) **HSYNCSTRT**—Horizontal Sync Start. The RCLK count on each scan line where HSYNC pin is activated. This parameter is not programmable. The RCLK period that follows the rising HSYNC edge is defined as column zero. It is used as the reference for all other horizontal timing parameters.
- b) **HSYNCSTP**—Horizontal Sync Stop. The RCLK count on each scan line where the HSYNC pin is deactivated. The falling edge of HSYNC occurs at the leading edge of the programmed RCLK period.
- c) **LINELLEN**—Line Length. This parameter defines the total number of RCLK's in each scan line including display time, border and horizontal retrace time. There are $LINELIN + 1$ RCLK periods per horizontal line scan.
- d) **HBRDSTRT**—Horizontal border start. The RCLK count on a scan line where the border begins. The border begins at the leading edge of the programmed RCLK period.
- e) **HBRDSTP**—Horizontal Border Stop. The RCLK count on a scan line where the border ends. The border terminates at the leading edge of the programmed RCLK period.
- f) **HFLDSTRT**—Horizontal Field Start. The RCLK count on a scan line where the character display field begins. If the row buffer is ready to be displayed, the CSYN pin will be deactivated at this point. This field begins at the leading edge of the programmed RCLK period.
- g) **HFLDSTP**—Horizontal Field Stop. The RCLK count on a line where the character display field stops. When this timing point is reached, CSYN will be activated. This field ends at the leading edge of the programmed RCLK period.

There is also one pseudo parameter, **SYNCDLY**. It is fixed at one half **LINELLEN** and is used as the start and end timing for **VSYNC** in odd frames in interlaced displays. **VSYNC** starts at **HSYNCSTRT** in even frames for interlaced displays and all frames for non-interlaced displays.

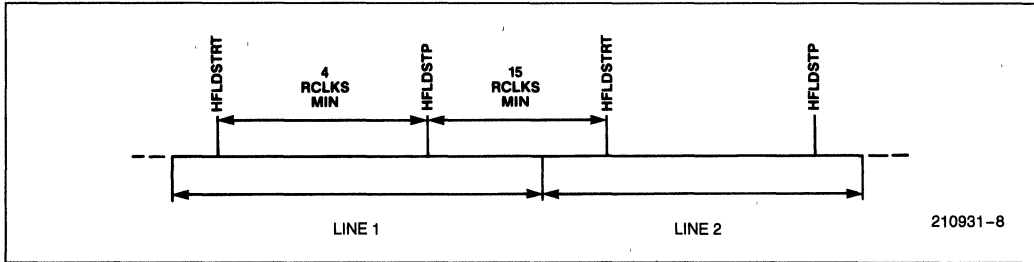


Figure 12. Horizontal Timing Restrictions

There are certain restrictions in the programming of HFLDSTRT and HFLDSTP and those restrictions are best illustrated below. There has to be at least 4 RCLKS in between HFLDSTRT and HFLDSTP of the same scan line and 15 RCLKS in between HFLDSTP of one line and HFLDSTRT of the next. The minimum delay of 15 RCLKS is for the charging of the pipeline from the row buffer to the character data output DAT0-DAT14 as well as the setting of the correct value for the scan line output LC0-LC4.

the display of the Status Row is completed. See * below.

***NOTE:**

(Character Field Boundary definition: The starting or ending event is defined to occur at HFLDSTP on the scan line following the programmed value. Thus the visible character field effectively begins two scan lines below the programmed start value and ends one scan line below the programmed stop value.)

Frame Timing Parameters

Frame timing parameters are stored in the VERT MODE CAM and are programmable from 0-2047 scan lines. These parameters are:

- a) VSYNCSTRT—Vertical Sync Start. The line count where the VSYNC is activated. This occurs at the end of a field automatically. This parameter is not programmable. The rising edge of VSYNC occurs with the rising edge of HSYNC for all non-interlace fields and for odd fields in the interlace mode.
- b) VSYNCSTP—Vertical Sync Stop. The line count at which the VSYNC pin is normally deactivated. VSYNC changes at the rising edge of HSYNC normally. However it occurs at SYNC DLY at the beginning of odd fields of an interlaced display.
- c) FRAMELEN—Frame Length. This parameter defines the total number of scan lines per frame. It is used to reset the FRAME LINE CNTR. In an interlaced display, FRAMELEN must be an even number. If an odd number is programmed, one additional line will occur automatically. There will be FRAMELEN + 1 scan lines per frame. (Note that interlace mode contains two fields per frame).
- d) VFLDSTRT—Vertical Field Start. Programs the scan line count where the character display field begins.
- e) VFLDSTP—Vertical Field Stop. Programs the scan line count where the regular character display field ends. VFLDSTP times the beginning of the Status Row. The channel attention sequences, interrupt handling, row buffer swap and initialization for the next frame are started after

Status Row

The Vertical Frame Timing Parameters have no border controls, unlike the Horizontal Row Timing Parameters. The top and bottom borders can be replaced with regular display rows that are video-reversed and contain no data. The top border is easily timed from VFLDSTRT. The bottom border is more difficult without help from the Vertical Timing generators. If there were no help, the user would have to keep track of the number of scan lines used in each row to know when to stop regular display and create the bottom border. This would also preclude his ending his regular display with an EOF command before the border.

The 82730 provides this help with the Status Row feature. The display of the Status Row is timed from VFLDSTP and allows the user to display a row in a fixed position at the bottom of the screen that is independent of the regular data and any display errors (display ended by an EOF command or the DURN, DBOR, or FDE errors). (There is one dependency on the regular display data: the row format. The last FULROWDESCPT (FRD) set in the regular data will be used on the Status Row unless a new command is issued for the row. It is recommended that the user include a new FRD command in the Status Row data to eliminate this dependency).

Status Row display starts SCROLL MARGIN plus one scan line after VFLDSTP. This margin is provided to insure enough DMA time if the regular display runs up to VFLDSTP. The user can create a bottom border or any end-of-display row that he chooses. A

display status or system status line, or special programmable key function definition line can be implemented with this feature.

CHARACTER ATTRIBUTES

The 15 bits of the character word can be partitioned into character address and attribute bits.

Some common attributes may be individually defined and enabled or disabled by fields in the attribute parameter registers. Each attribute has two means of being enabled. The enable bits defined below are set during the MODESET channel command and are used as a global enable. The user does not have to enable the provided attributes. He may free more data bits for his own use this way. The second enable bit is contained in each character loaded to the row buffer to enable the attribute on a character by character basis. They are individually described in detail in the following sections.

Reverse Video

When a character with the reverse video attribute is displayed, the CRVV pin will be inverted during the time the character is being displayed. The reverse video affects the entire height of the row for that character space. For superscript/subscript pairs, the reverse video effect is controlled by superscript until SUBSTRT when the subscript attribute bit takes control. The parameter for this attribute is:

RVBS—Reverse Video Bit Select. This parameter selects one of the 15 bits of a character data word. Values 0 through 14 select the corresponding bit. Value 15 disables the Reverse Video attribute.

Blinking Character

When a character with the blinking character attribute is displayed, the BLANK pin will be activated and deactivated during the character display time according to programmable rate and duty cycle. The parameters for this attribute are:

- a) **BCBS**—Blinking Character Bit Select. Selects one of the 15 bits of a character data word as the blinking character attribute control. As with Reverse Video above, the value of the select determines the controlling bit or disables the attribute.
- b) **CHAR BLNK FREQ**—Selects one of the 32 blinking frequencies available for the blinking character and blinking underline. The character blink rate is calculated as below:

$$\text{Blink Rate} = \frac{\text{Frame Refresh Rate}}{4 \times \text{CHAR BLNK FREQ}}$$

- c) **CHAR DUTY CYCLE**—A 2-bit register to select 4 duty cycles available for blinking character and blinking underline. The selection logic is defined to be as follows:

00 = 100% always on
 11 = 75% on
 10 = 50% on
 01 = 25% on

Underline # 1

When a character with underline is displayed, the BLANK Pin will be activated and the CRVV pin will be inverted during the time the scan line specified by the underline select register is displayed. The parameters used to define underline # 1 are:

- a) **ULS1**—Underline Line Select 1. It determines which scan line of a character row will be used for the underline #1. This parameter is modifiable on a row by row basis by the FULROWDESCRPT command.
- b) **ULBS1**—Underline Bit Select 1. This parameter can only be changed by MODESET. It selects one of the 15 bits of a character data word as the underline #1 attribute control. Again, a value of 15 in the select field disables this attribute.

Underline # 2 (Blinking)

Underline #2 can be made to blink. When its blinking feature is deactivated, its visual effect is exactly the same as underline #1. When it is enabled to blink, its blink rate and blinking duty cycle are the same as those defined for blinking character. The parameters used to define this attribute are:

- a) **UL2SEL**—Underline Line Select 2. This parameter determines which scan line of a character will be the 2nd underline. It is changeable on a row by row basis by the FULROWDESCRPT command.

The next two parameters can only be modified by the MODESET Command.

- b) **ULBS2**—Underline Bit Select 2. Selects one of the 15 bits of a character data word or GPA1 as the second underline attribute control. A bit select value of 15 disables the second underline.
- c) **BUE**—Blinking Underline Enable. Activation of this bit will cause the second underline attribute to start blinking.

Invisible

A character with this attribute will occupy its character position on the screen but will not be displayed (i.e. BLANK will be active). This attribute does not

affect the Reverse Video attribute if they are programmed together. The parameter that is used to implement this attribute is:

IBS—Invisible Bit Select. Selects one of the 15 bits of a character data word as the invisible attribute control. Value 15 disables the invisible attribute.

Absolute Line Cntr Attribute

This character attribute allows the display of special graphic characters, or may be used to upshift normal characters to implement displays with overlapping superscript and subscript fields. When a character with this character attribute enabled is being displayed, its LC0–LC4 pins will reflect the output from the CHAR ROW LNE CNTR which counts the absolute line count of a row. The activation of this attribute overrides the line count mode of both normal and subscript/superscript characters. The parameter used to select the attribute is:

ABS LINE BIT SEL. This four bit register selects one of the 15 bits of a character data word as the absolute line counter output attribute control. Select value 15 disables the ABS Line attribute.

Cursor Generation

The cursor characteristic parameters are changeable on a frame by frame basis by MODESET.

- a) **CUR FREQ—Cursor frequency.** Selects the blinking frequency for both cursors. The selection logic is similar to CHAR BLNK FREQ
- b) **CUR DUTY CYCLE—Cursor duty cycle.** Selects the blinking duty cycle for both cursors. Its selection logic is similar to CHAR DUTY CYCLE.
- c) **CR1RVV—Cursor 1 Reverse Video Enable** selects a reverse video type cursor as opposed to a solid (blanking) cursor.
- d) **CR1BE—Cursor 1 Blink Enable** changes the cursor 1 block or underline to a blinking block or underline. Enabling this bit also causes DAT 14 pin to “blink” as well, if the CR10E bit is set.
- e) **CR10E—Cursor 1 Output Enable** reconfigures the DAT 14 pin to indicate when cursor 1 is active. CR20E enabled directs the cursor 2 signal to DAT 13 pin in a similar fashion.
- f) **CR1CD—Cursor 1 Light Pen Cursor Detect** directs the CCLK cursor #1 position to be translated to its nearest equivalent RCLK position through the LPEN facility.

An identical set of parameters (c) through (f) is available for the generation of CURSOR 2. The two cursors share the same FREQ and DUTY CYCLE parameters.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias0°C to 80°C
 Storage Temperature -65°C to +150°C
 Voltage on Any Pin with
 Respect to Ground -1.0V to +7V
 Power Dissipation3 Watts

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

NOTICE: Specifications contained within the following tables are subject to change.

D.C. CHARACTERISTICS $T_A = 0^\circ\text{C TO } 70^\circ\text{C}, V_{CC} = 5\text{V} \pm 10\%$

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 _{OL}	Output Low Voltage		0.45	V	I _{OL} = 2 mA (1)
V _{OH}	Output High Voltage	2.4		V	I _{OH} = -400 μA
I _{CC}	Power Supply Current		400	mA	@ T _A = 0°C
I _{LI}	Input Leakage Current		10	μA	V _{IN} = 0 - V _{CC}
I _{LO}	Output Leakage Current		± 10	μA	V _{OUT} = 0.45 - V _{CC}
I _{LCL}	LC0, LC1, LC2 Input Low Current	-125	-450	μA	V _{IN} = 0V Reset = "1" (2)
V _{BLI}	Bus Clock Input Low Voltage	-0.5	0.8	V	
V _{BHI}	Bus Clock Input High Voltage	2.0	V _{CC} + 1.0	V	
V _{CLI}	Character Clock Input Low Voltage	-0.5	0.8	V	
V _{CHI}	Character Clock Input High Voltage	2.2	V _{CC} + 0.5	V	
V _{RLI}	Reference Clock Input Low Voltage	-0.5	0.8	V	
V _{RHI}	Reference Clock Input High Voltage	2.2	V _{CC} + 0.5	V	

NOTES:

- I_{OL} = 2.6 mA on the $\overline{S1}$ and $\overline{S0}$ pins.
- Measured after at least 5 BCLK cycles after RESET = High.

A.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 10\%$. All timings in nanoseconds. $CL = 50\text{ pF}$.

82730 Bus Interface Input Timing Requirements

Symbol	Parameter	Min	Max	Units	Test Conditions
TCLCL	BCLK Cycle Period	125	2500	ns	
TCLCH	BCLK Low Time	52		ns	
TCHCL	BCLK High Time	52		ns	
TCH1CH2	BCLK Rise Time		30	ns	0.45C \rightarrow 2.4V (1)
TCL1CL2	BCLK Fall Time		30	ns	2.4V \rightarrow 0.45V (1)
TDVCL	Data In Set-Up Time	20		ns	
TCLDX	Data In Hold Time	5		ns	
TARYHCH	Async. READY Active Set-Up Time	35		ns	
TSRYHCL	Sync. READY Active Set-Up Time	20		ns	
TRYLCL	READY Inactive Set-Up Time	10		ns	
TCLRYX	READY Hold Time	20		ns	
TCTVCL	HLDA, RESET Set-Up Time	35		ns	
TCLCTX	HLDA, RESET Hold Time	10		ns	
TCAVCAX	CA Pulse Width	100		ns	
TRIVRIX	IRST Width	100		ns	
TRLLCH	LC _x Input Hold Time	5TCLCL		ns	(2)

NOTE:

2. Applies only to test mode invocation.

82730 Bus Interface Output Timing Response

Symbol	Parameter	Min	Max	Units	Test Conditions
TCLAV	Address Valid Delay	0	55	ns	
TCLAX	Address Hold Time	0		ns	
TAVAL	Address Valid to ALE/UALE Inactive	TCLCH – 30		ns	
TLLAX	Address Hold to ALE Inactive	TCHCL – 10		ns	
TCLAZ	Address Float Delay	TCLAX	45	ns	
TAZRL	Address Float to \overline{RD} Active	0		ns	
TLHLL	ALE/UALE Width	TCLCH – 10		ns	
TCLLH	ALE/UALE Active Delay	0	45	ns	
TCHLL	ALE/UALE Inactive Delay	0	45	ns	
TCVCTV	Control Active Delay (\overline{DEN} , \overline{WR} , \overline{AEN})	0	70	ns	
TCVCTXW	Control Inactive Delay (\overline{WR} , \overline{AEN})	0	80	ns	
TCVCTXD	Control Inactive Delay (\overline{DEN})	5	80	ns	
TCLDOV	Data Out Valid Delay	0	50	ns	
TCLDOX	Data Out Hold Time	0		ns	
TWHDX	Data Out Hold Time After \overline{WR}	TCLCL – 60		ns	
TCLHV	Hold Output Delay	0	85	ns	
TRLRH	\overline{RD} Width	2TCLCL – 50		ns	
TCLRL	\overline{RD} Active Delay	0	95	ns	
TCLRH	\overline{RD} Inactive Delay	5	70	ns	
TRHAV	\overline{RD} Inactive to Next Address Active	TCLCL – 40		ns	
TCLSIN	SINT Valid Delay	0	70	ns	
TRHSIL	RINT Active to SINT Inactive		250	ns	
TCHSV	Status Active Delay	0	75	ns	
TCLSH	Status Inactive Delay	0	70	ns	
TWLWH	\overline{WR} Width	2TCLCL – 40		ns	
TFLHL	Bus Float to HOLD Inactive	0		ns	

82730 Display Generator Input Timing Requirements

Symbol	Parameter	Min	Max	Units	Test Conditions
TRCHRCH	RCLK Cycle Period	100	2500	ns	
TRCHRCL	RCLK High Time	40		ns	
TRCLRCH	RCLK Low Time	40		ns	
TRRCK	RCLK Rise Time		30	ns	0.45V → 2.4V (1)
TFRCK	RCLK Fall Time		30	ns	2.4V → 0.45V (1)
TCCHCCH	CCLK Cycle Period	100	None	ns	
TCCHCCL	CCLK High Time	30		ns	
TCCLCCH	CCLK Low Time	40		ns	
TRCCK	CCLK Rise Time		30	ns	0.45V → 2.4V (1)
TFCK	CCLK Fall Time		30	ns	2.4V → 0.45V (1)
TVCVCR	HSYNC, SYNCIN Set-Up Time	30		ns	
TCRVCX	HSYNC, SYNCIN Hold Time	10		ns	
TLPVCF	LPEN Set-Up Time	30		ns	
TCFLPX	LPEN Hold Time	10		ns	
TRCHCCH	CCLK/RCLK Skew During CSYNC	-10	10	ns	

82730 Display Generator Output Timing Response

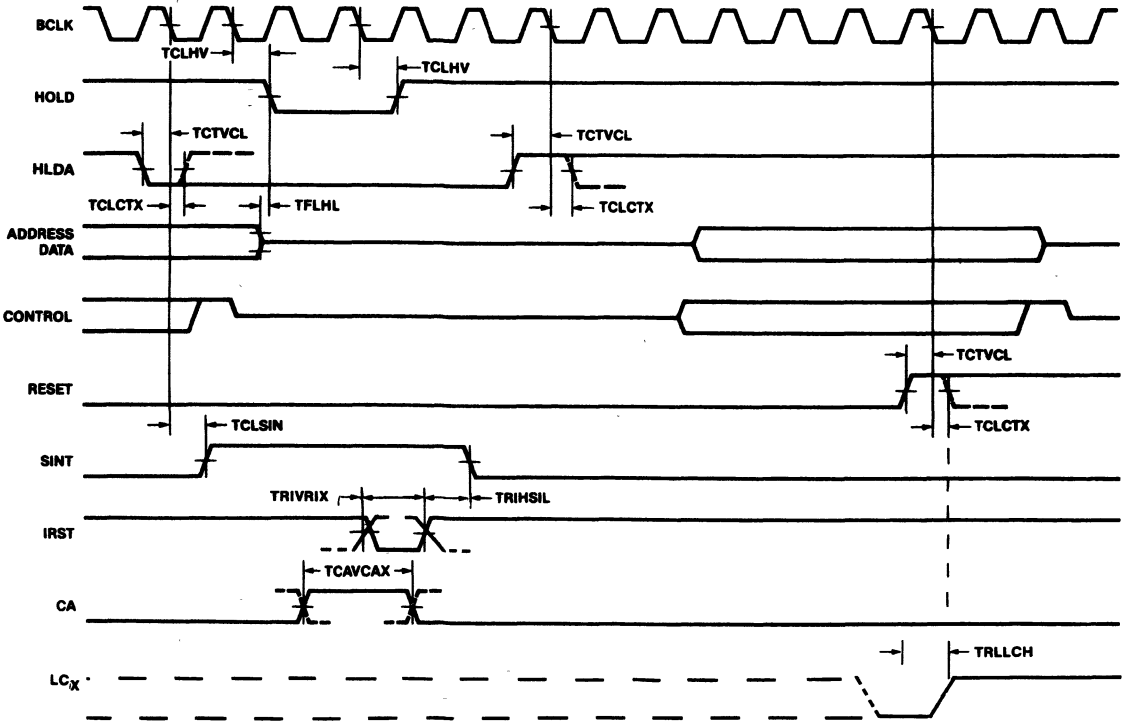
Symbol	Parameter	Min	Max	Units	Test Conditions
TCCHDV	Data, Line Count and Attribute and Output Valid Delay from the Rising Edge of CCLK		60	ns	$C_L = 100$ pF
TCCHDX	Data, Line Count and Attribute and Output Hold Time	5		ns	$C_L = 100$ pF
TRCHCV	Delay of Outputs CSYNC, VSYNC, HSYNC or RRVV from the Rising Edge of RCLK		70	ns	$C_L = 100$ pF
TCCHCL	CCLK Rising to \overline{CHOLD} Low		75	ns	$C_L = 50$ pF
TRCLCH	RCLK Falling to \overline{CHOLD} High		60	ns	$C_L = 50$ pF

NOTES:

1. Clock maximum rise and fall times are for functionality only. AC timings are not tested at this condition.
2. Applies only to test mode invocation.

WAVEFORMS (Continued)

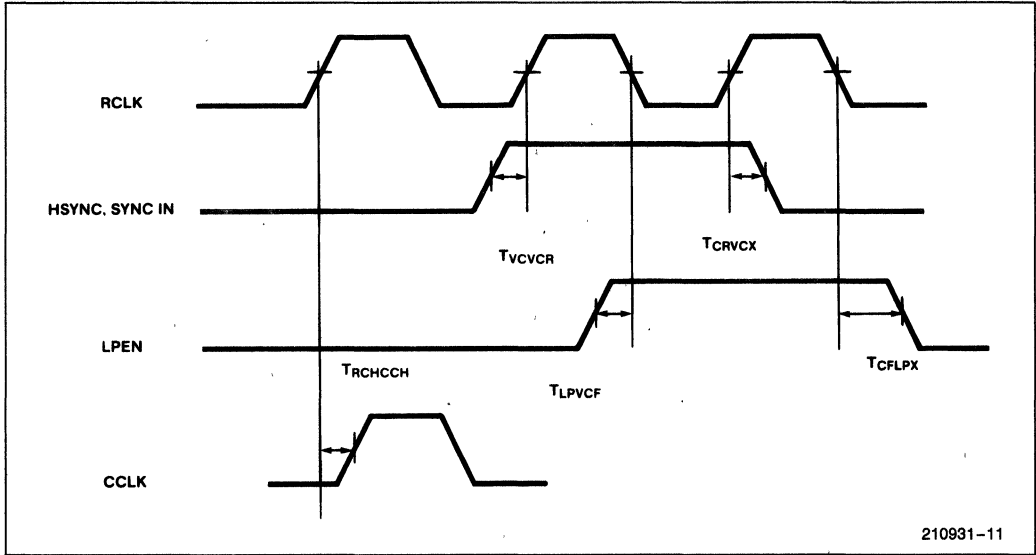
HOLD, RESET, SINT AND CA TIMING



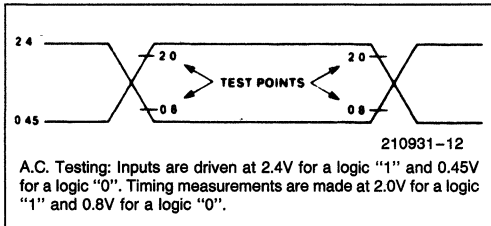
210931-10

WAVEFORMS (Continued)

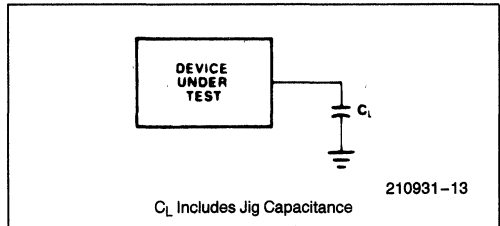
DISPLAY GENERATOR INTERFACE TIMING



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING LOAD CIRCUIT





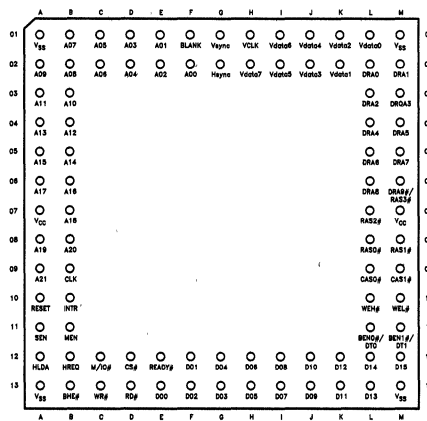
82786 CHMOS GRAPHICS COPROCESSOR

- High Performance Graphics
- Fast Polygon and Line Drawing
- High Speed Character Drawing
- Advanced DRAM/VRAM Controller for Graphics Memory up to 4 Mbytes
- Supports up to 200 MHz CRTs
 - up to 640 by 480 by 8 Bits (1K*1K*8)
 - or 1400 by 1400 by 1 Bit (2K*2K*2)
- Up to 1024 Simultaneous Colors
- Integral DRAM/VRAM Controller, Shift Registers and DMA Channel
- IBM Personal Computer Color Graphics Adapter-Compatible
- International Character Support
- Interface Designed for Device-Independent Standards
 - Virtual Device Interface
 - Graphics Kernal System
 - NAPLPS
- Hardware Windows
- Fast Bit-Block Copies Between System and Bit-Map Memories
- Integral Video DRAM Support
 - Up to 1900 by 1900 by 8
- Third-Party Software Support
- Multi-tasking Support
- Provides Support for Rapid Filling with Patterns
- Programmable Video Timing
- Advanced CHMOS Technology
- Supports Dual Port Video DRAMs & Sequential Access DRAMs
- 88 Pin Grid Array and Leadless Chip Carrier

(See Intel Packaging; Order Number: 231369-001)

The 82786 is a powerful, yet simple component designed for microcomputer graphics applications including personal computers, engineering workstations, terminals, and laser printers. Its advanced software interface makes applications and systems level programming efficient and straight-forward. Its performance and high-integration make it a cost-effective component while improving the performance of nearly any design. Hardware windows provide instantaneous changes of display contents and support multiple graphics applications from multiple graphics bit maps. Applications programs written for the IBM Personal Computer can be run within one or more windows of the display when used with Intel CPUs.

The 82786 works with all Intel microprocessors, and is a high-performance replacement for sub-systems and boards which have traditionally used discrete components and/or software for graphics functions. The 82786 requires minimal support circuitry for most system configurations, and thus reduces the cost and board space requirements of many applications. The 82786 is based on Intel's advanced CHMOS III process.



231676-27

Figure 1. 82786 Pinout—Bottom View

INTRODUCTION

The 82786 is an intelligent peripheral capable of both drawing and refreshing raster displays. It has an integrated drawing engine with a high level VDI like graphics commands. Multiple character sets (fonts) can be used simultaneously for text display applications. The 82786 provides hardware support for fast manipulation and display of multiple win-

dows on the screen. It supports high resolution displays with a 25 MHz pixel clock and can display up to 256 colors simultaneously. Using multiple 82786s and/or in conjunction with dual port video DRAMs (VRAMs), the 82786 is virtually unlimited in terms of color support and resolution.

Table 1. 82786 Pin Description

Symbol	Pin Number	Type	Description
A21:0	A09,B08,A08,B07, A06,B06,A05,B05, A04,B04,A03,B03, A02,B02,B01,C02, C01,D02,D01,E02, E01, F02	I/O	Address lines for the External Bus. Inputs for Slave Mode accesses of the 82786 supported Graphics memory array or 82786 internal memory or I/O mapped registers. Driven by the 82786 when it is the External Bus Master.
D15:0	N12,M12,M13,L12, L13,K12,K13,J12, J13,H12,H13,G12, G13,F13,F12,E13	I/O	Data Bus for the 82786 Graphics memory array and the External Bus.
$\overline{\text{BHE}}$	B13	I/O	Byte High Enable. An input of the 82786 Slave Interface; driven LOW by the 82786 when it is a Bus Master. Determines asynchronous vs. synchronous operation for $\overline{\text{RD}}$, $\overline{\text{WR}}$ and HLDA inputs at the falling (trailing) edge of RESET. A HIGH state selects synchronous operation.
$\overline{\text{RD}}$	D13	I/O	Read Strobe. An input of the 82786 Slave Interface; driven by the 82786 when it is a Bus Master. Selects normal/test mode at falling RESET.
$\overline{\text{WR}}$	C13	I/O	Write Strobe. An input of the 82786 Slave Interface; driven by the 82786 when it is a Bus Master. Selects normal/test mode at falling RESET.
$\text{M}/\overline{\text{IO}}$	C12	I/O	Memory or I/O indication. An input of the 82786 Slave Interface; driven HIGH by the 82786 when it is the Bus Master. Determines synchronous 80286 or 80186 interface at the falling edge of RESET. A LOW state selects a synchronous 80286 interface.
$\overline{\text{CS}}$	D12	I	Chip Select. Slave Interface input qualifying the access.
MEN	B11	O	Master Enable. Driven HIGH when the 82786 is in control of the External Bus. (i.e., HLDA received in response to a 82786 HREQ.) Used to steer the data path and select source of bus cycle status commands.
SEN	A11	O	Slave Enable. Driven HIGH when the 82786 is executing a Slave bus cycle for an External Master into the 82786 controlled memory or registers. Used to enable the data path and as a READY indication to the External Bus Master.
READY	E12	I	Synchronous input to the 82786 when executing External Bus cycles. Identical to 80286 READY.

Table 1. 82786 Pin Description (Continued)

Symbol	Pin Number	Type	Description
HREQ	B12	O	Hold Request. Driven HIGH by the 82786 when an access is being made to the External Bus by the Display or Graphics Processors. Remains HIGH until the 82786 no longer needs the External Bus.
HLDA	A12	I	Hold Acknowledge. Input in response to a HREQ output. Asynchronous vs. synchronous input determined by state of $\overline{\text{BHE}}$ pin at falling RESET.
INTR	B10	O	Interrupt. The logical OR of a Graphics Processor and Display Processor interrupt. Cleared with an access to the BIU Interrupt Register.
RESET	A10	I	Reset input, internally synchronized, halts all activity on the 82786 and brings it to a defined state. The leading edge of RESET synchronizes the 82786 clock to phase 2. The trailing edge latches the state of $\overline{\text{BHE}}$ to establish the type of Slave Interface. It also latches $\overline{\text{RD}}$, $\overline{\text{WR}}$ and MIO) to set certain test modes.
CLK	BO9	I	Double frequency clock input. Clock input to which pin timings are referenced. 50% duty cycle.
$\overline{\text{CAS0}}$	M09	O	Column Address Strobe 0. Drives the CAS inputs of the even word Graphics memory bank if interleaved; identical to $\overline{\text{CAS1}}$ if non interleaved Graphics memory. Capable of driving 16 DRAM/VRAM CAS inputs.
$\overline{\text{CAS1}}$	N09	O	Column Address Strobe 1. Drives the CAS inputs of the odd word Graphics memory bank if interleaved; identical to $\overline{\text{CAS0}}$ if non-interleaved Graphics memory. Capable of driving 16 DRAM/VRAM CAS inputs.
$\overline{\text{RAS2:0}}$	M07,N08,M08	O	Row Address Strobe. Drives the RAS input pins of up to 16 DRAMs/VRAMs. Drives the first three rows of both banks of Graphics memory.
$\overline{\text{DRA9/RAS3}}$	N06	O	Multiplexed most significant Graphics memory address line and $\overline{\text{RAS3}}$. DRA9 when using 1 Mb DRAMs; $\overline{\text{RAS3}}$ otherwise.
$\overline{\text{WEL}}$	N10	O	Write Enable Low Byte. Active LOW strobe to the lower order byte of Graphics memory.
$\overline{\text{WEH}}$	M10	O	Write Enable High Byte. Active LOW strobe to the higher order byte of Graphics memory.
$\overline{\text{DRA8:0}}$	M06,N05,M05, N04,M04,N03, M03,N02,M02	O	Multiplexed Graphics memory Address. Graphics memory row and column address are multiplexed on these lines. Capable of driving 32 DRAMs/VRAMs.
$\overline{\text{BEN1:0}}$ DT1:0	N11,M11	O	Multiplexed Bank Enable and Data Transfer Line. In normal memory cycle enables the output of the Graphics memory array on to the 82786 data bus, D15:0. In data transfer cycle, loads the serial register in dual port video DRAMs (VRAMs). $\overline{\text{BEN1/DT1}}$ and $\overline{\text{BEN0/DT0}}$ control Bank1 and Bank0 respectively.
BLANK	F01	I/O	Output used to blank the display at particular positions on the screen. May also be configured as input to allow the 82786 to be synchronized with external sources.

Table 1. 82786 Pin Description (Continued)

Symbol	Pin Number	Type	Description
V _{DATA7:0}	H02,J01,J02, K01,K02,L01, L02,M01	O	Video data output.
V _{CLK}	H01	I	Video Clock input used to drive the display section of the 82786. Maximum frequency of 25 MHz.
H _{SYNC/WS0}	G02	I/O	Horizontal Sync. Window status is multiplexed on this pin. May also be configured as input to allow the 82786 to be synchronized with external sources. May also be configured to output Window status.
V _{SYNC/WS1}	G01	I/O	Vertical Sync. Window status is multiplexed on this pin. May also be configured as input to allow the 82786 to be synchronized with external sources. May also be configured to output Window status.
V _{SS}	A01,M01,A13, N13		4 V _{SS} pins.
V _{CC}	N07,A07		2 V _{CC} pins.

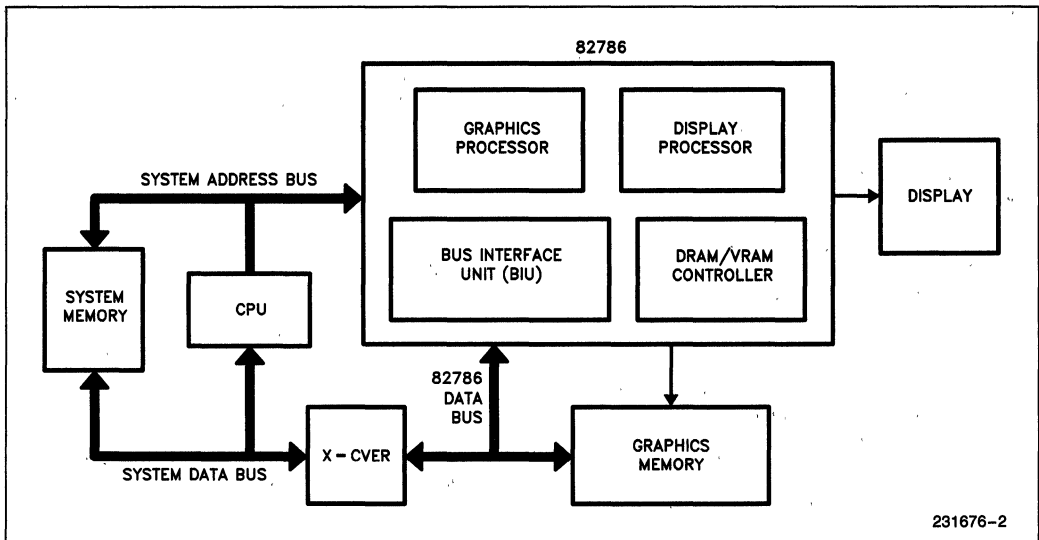


Figure 2

ARCHITECTURE

The 82786 is a high integration device which contains three basic modules (figure 2):

1. Display Processor (DP)
2. Graphics Processor (GP)
3. Bus Interface Unit (BIU) with DRAM/VRAM Controller.

Display Processor

The 82786 Display Processor controls the CRT timings and generates the serial video data stream for the display. It can assemble several windows on the screen from different bit-maps scattered across the memory accessible to the 82786.

Graphics Processor

The 82786 Graphics Processor executes commands from a Graphics Command Block (GCMB) (placed in memory by the host CPU) and updates the bit-map memory for the Display Processor. The Graphics Processor has high level VDI like commands and can draw graphical objects and text at high speeds.

Bus Interface Unit (BIU)

The BIU controls all communication between the 82786, external CPU and memory. The BIU includes an integrated DRAM/VRAM controller that can take advantage of the high speed burst access modes of page mode and fast page mode DRAMs to perform block transfers. The Display Processor and Graphics Processor use the BIU to access the bit-maps in memory.

Memory Structure and Internal Registers

The 82786 address range is 4 Mbytes. This is divided between the Graphics memory directly supported by the 82786 and External system memory. The 82786 distinguishes between Graphics memory and External system memory by assuming Graphics memory space starts at address 0H and goes up to whatever amount of Graphics memory is configured. External system memory occupies the rest of the address space. The amount of Graphics memory configured, and therefore the Graphics memory/External system memory boundary, is controlled by the "DRAM Control Register" in the BIU. The upper limit of configured Graphics memory is 4 Mbytes.

A 128 byte block (contiguous) of internal control registers is distributed throughout the three modules on the 82786. This block can be either memory or I/O mapped in the CPU address space. The base address and memory or I/O mapped for this register block is programmable through the "Internal Relocation Register" in the BIU.

External Memory Access (Master Mode)

The 82786 goes into "Master Mode" whenever it needs to access a memory address that is beyond the upper limit of configured graphics memory. This memory is typically external memory shared between the 82786 and the external CPU. The bus timings in this mode are similar to the 80286 style bus timings.

An 82786 request for the bus is indicated by a high level on the HREQ line. The 82786 drives the external bus (A21:0, D15:0, RD, WR, MIO and BHE) only after receiving a HLDA (acknowledge) from the external master. The HLDA line could be externally synchronized (82786 synchronous mode) or internally synchronized (82786 asynchronous mode). The 82786 will deactivate the HREQ line when it no longer needs to access external memory or when it senses an inactive HLDA. The 82786 indicates that it is in control of the external bus by a high level on the MEN output.

Slave Mode

The 82786 Slave Interface allows an external CPU access into the Graphics memory array or the 82786 Internal Registers. The external CPU directs a (read/write) slave access to the 82786 by asserting the 82786 \overline{CS} input. When the 82786 is not driving the external bus, the A21:0, RD, WR, MIO and BHE lines are inputs. The RD, WR, MIO and \overline{CS} lines are constantly monitored by the 82786 to detect a CPU cycle directed at the 82786. After beginning a slave access to the 82786, the external CPU must go into a wait state. The 82786 will not process new slave commands from the CPU before the previous command has been serviced. The 82786 indicates the termination of the slave access by a high level on the SEN output. The data bus transceivers can be enabled by SEN.

SEN as Slave Ready Indication

SEN is optimized for connection to the 82284 ARDY or SRDY input when the Slave Interface is set in synchronous 80286 mode. When operating in sync with the 80286, slave write cycles can always execute with a minimum of 2 wait states. The number of wait states for a read cycle is a function of the DRAM/VRAM speed. For 2 wait state reads, SEN is connected to SRDY. For 3 wait state reads, it is connected to ARDY. Write cycles in both cases execute with 2 wait states because the 82786 issues SEN with different timing during write cycles.

The 82786 supports byte accesses to Graphics memory. The combination of \overline{BHE} and A0 generate the proper WEL and WEH signals. BHE and A0 are ignored for read cycles. Since the Display and Graphics Processors always generate word addresses, the slave cycles directed to graphics memory are the only time WEL may not exactly follow WEH.

The 82786 will acknowledge a slave access from an external CPU while waiting for an acknowledge (HLDA) to its own request for the external bus. This prevents a potential deadlock situation.

Synchronous/Asynchronous Operation

The synchronous/asynchronous mode is selected by the state of the BHE input at the falling edge of reset. A high state selects synchronous operation. The synchronous interface requires that the 82786 and the 80286/80186 clock inputs are shared. For the 80286 case, a common RESET ensures that the 82786 and the 80286 initialize to the same state. With the 80186, external hardware must ensure that the 82786 phase1 is coincident with the 80186 CLKOUT LOW. In the Master Mode, the HLDA line is sampled synchronously or asynchronously. The 82786 slave interface provides for synchronous or asynchronous sampling of the command lines (\overline{RD} and \overline{WR}).

EIGHT AND SIXTEEN BIT HOST

On reset, the 82786 always assumes an 8 bit host interface. The first few accesses to the 82786 must be 8 bit accesses. The 82786 can be switched to a 16 bit interface by setting the "BCP" bit in the "BIU Control Register".

In 16 bit mode, the Internal Register Block is only word addressable. Odd word or odd byte accesses to the internal locations will not produce the desired result. Even byte access, however will work as desired. The least significant address bit, A0, is ignored in 16 bit mode.

In 8 bit mode, the internal registers must be accessed by two successive bus cycles. This is not necessary for reads, but is necessary for writes to the internal registers. The low byte (A0 = 0) must be written first, followed by the high byte (A0 = 1) of the register. A21:1 must be the same for both bus cycles. The register is not changed until the second byte (the high byte) is written to the 82786. There is no restriction on the time between the two bus cycles, but if successive low bytes are written before a high byte is written, the last low byte is the one written to the register. The BIU latches even bytes (A0 = 0) of write data in a temporary register. When an odd byte is subsequently written to location address + 1, this byte and the even byte in the temporary register are written to the desired location. A lock out mechanism prevents a high byte write to modify an internal register if there is no valid word in the temporary register.

There is no crossing done by the 82786 in 8 bit mode: low bytes are transferred on the low data lines D7:0 and the high bytes on D15:8. An external cross creates the 8 bit bus for the host. This is not additional hardware since a crossover is needed for an 8 bit host accessing of the memory array anyway.

MEMORY ACCESS ARBITRATION

The BIU receives requests to access the Graphics Memory from the Display Processor, the Graphics Processor and the External CPU. Additionally the internal DRAM/VRAM Controller also generates refresh requests. The DRAM/VRAM refresh requests are always highest priority. The other requests are arbitrated with programmable priorities. A higher priority request can interrupt lower priority memory cycles. Block transfers however can only be interrupted on doubleword boundaries.

There are two priority levels for requests from the Display and Graphics Processors: a First Priority (FPL) and a Subsequent Priority (SPL). The First Priority is the priority at which the first request of a bus cycle is arbitrated with. The Subsequent Priority is the priority associated with subsequent requests of a block transfer bus cycle. This allows for block transfers to execute with a different priority level. If a higher priority request occurs while a block transfer is executing, the BIU suspends the current block transfer and acknowledges the higher priority request. After completion of that higher priority memory access, the requests are arbitrated again. The suspended block transfer is arbitrated with its SPL priority since it is still executing a block transfer. The External Request has no Subsequent Priority level since it cannot execute block transfers. The default priorities from highest to lowest following RESET are:

External	FPL	7
Display	FPL	6
Graphics	FPL	5
Displays	SPL	3
Graphics	SPL	2

Three bit codes describe the priorities so 7 is the highest and 0 is the lowest. If two priority registers are programmed with the same value, a default priority chain is used. The default order is, from highest to lowest priority:

1. Display Processor
2. Graphics Processor
3. External

Graphics Memory Interface

The 82786 directly supports up to 32 DRAMs without additional external logic. This capability has the advantage of Simple utilization of cost effective memory devices and significant performance improvement through the use of either standard Page Mode or the newer Fast Page Mode/Static Column Decode sequential access RAMs. The Fast Page Mode/Static Column Decode parts enable the

82786 to cycle the DRAMs in 100 ns instead of the 200 ns used for Page Mode parts. The 82786 also allows the memory to consist of either standard single port memory devices or dual port Video RAM devices (VRAMs).

The 82786 supports a wide range of DRAM/VRAM configurations. The choices include interleaving or non-interleaving (1 or 2 banks - one CAS line/bank), number of rows per bank (1, 2, 3 or 4 - one RAS line/row), width (x1, x4 or x8), height (16k, 64k, 256k or 1M) and performance (Page Mode or Fast Page Mode/Static Column Mode). The only limitation is the address space limit of 4Mbytes. The 82786 DRAM/VRAM address lines (DRAx) can directly drive 32 memory devices while the RAS, CAS, WE and BEN lines can directly drive 16 devices. When the memory array consists of more than 32(16) devices then external drivers have to be used to drive the memory array.

There are some special DRAM configurations:

- i) When 1 Mb * 1 DRAMs are used, $\overline{RAS3}$ is used as DRA9.
- ii) When only one interleaved row is configured (32 devices), $\overline{RAS1}$ is identical to $\overline{RAS0}$. Additional buffering on $\overline{RAS0}$ is therefore not required.
- iii) When two non interleaved rows are configured (32 devices), $\overline{CAS1}$ is identical to $\overline{CAS0}$. Additional buffering on $\overline{CAS0}$ is therefore not required.

DRAM Cycle Types

The 82786 supports two fundamental memory cycle types: single and block. A single cycle involves a single 16 bit word, while a block transfer is a minimum of 2 16 bit words with no maximum length. The single cycle types supported and their cycle times are given below. The cycle times are counted in system clocks, $\frac{1}{2}$ the CLK input frequency.

- | | | |
|-----------------------|----------|-----------------|
| 1. Single Reads | 3 cycles | 300 ns @ 10 MHz |
| 2. Single Writes | 3 cycles | 300 ns @ 10 MHz |
| 3. Read-Modify-Writes | 4 cycles | 400 ns @ 10 MHz |

The block cycles use the high speed sequential access modes of page mode, fast page mode (ripple mode) and static column DRAMs. Typical performance numbers for this case are:

- | | | |
|------------------------------------|-----------|------------------|
| 1. Page Mode, Non-Interleaved | 2 cycles | 10 Mb/s @ 10 MHz |
| 2. Page Mode, Interleaved | 1 cycle | 20 Mb/s @ 10 MHz |
| 3. Fast Page Mode, Non-Interleaved | 1 cycle | 20 Mb/s @ 10 MHz |
| 4. Fast Page Mode, Interleaved | .5 cycles | 40 Mb/s @ 10 MHz |

All accesses into the graphics memory by the Display Processor use the high speed sequential access mode whenever possible. The Graphics Processor uses a single Read-Modify-Write cycles for all pixel drawing operations. Block copy operations by the Graphics Processor use the high speed sequential access modes. External CPU access into graphics memory is always a single read or write cycle. When configured to interface with dual port VRAMs, the 82786 generates Page Mode and Fast Page Mode style control signals for memory access through the normal random access port. It also executes a data transfer cycle when the video shift register in the VRAMs have to be loaded.

Graphics Memory Refresh

The BIU has an internal DRAM/VRAM refresh controller. The refresh period is programmable through the "DRAM/VRAM Refresh Control" Register in the BIU. All configured rows are refreshed simultaneously by activating the corresponding \overline{RAS} lines periodically (\overline{RAS} only refresh). The refresh row address (10 bits) is generated internally. On power up, the refresh row address is undefined. On normal reset, the refresh row address is not affected. It is initialized only if the 82786 is reset into the "BIU Test Mode". Not modifying the refresh address during RESET allows for a "warm RESET" implementation: contents of DRAM/VRAM can be insured to remain valid if RESET is short enough (less than three DRAM/VRAM refresh cycles). DRAM/VRAM refresh will continue at the proper row after RESET goes inactive again.

The graphics memory refresh cycles are always the highest priority cycles. There is some latency possible between the internal refresh request and the actual refresh cycle. This latency is critical only in one case: The 82786 is in a wait state while executing a bus cycle on the External Bus.

The worst case is a refresh request occurring just after the 82786 receives a HLDA from the host CPU to execute a block transfer on the External bus. Refresh requests can interrupt block transfers, but only on doubleword boundaries — the 82786 must execute 2 full bus cycles on the External Bus before the refresh cycle is run. The possibility of many wait states when executing these two bus cycles creates a need for a large refresh latency tolerance. The 82786 can queue up to 3 refresh requests internally. In the default mode (refresh request @ 15.2 micro seconds) and at 10 MHz operation, this implies that each bus cycle to external memory should not have more than 225 wait states.

There is no warm up logic on the 82786. The system must either wait for sufficient number of refresh cycles to execute or the boot software on the host can

quickly access the memory array for the required number of cycles.

The default value of the DRAM/VRAM Control Register configures the array as 4 rows of Non-Interleaved Page Mode 256k × 1 with refresh requests generated every 15.2 micro seconds.

Internal Register and Graphics Memory Slave Access

The external master can access either 82786 internal memory I/O mapped registers or the Graphics memory. The 82786 internal address space consists of a contiguous 128 byte block. It is mapped to memory or I/O space depending on the state of the M/I \bar{O} bit in the "Internal Relocation Register". The 82786 determines the access type (memory vs I/O) by the M/I \bar{O} pin. An address comparison is done between the Internal Relocation Register and the incoming address to determine if the CPU access is directed to internal memory/I/O mapped registers.

Intel reserves the right to add functions to future versions of the 82786. Users should not use reserved locations in order to ensure future compatibility.

PERFORMANCE

Slave performance is measured here by assuming a request is made to an idle 82786. A synchronous interface is assumed.

Minimum 80286 Wait States = 3
(10 MHz 80286 and 82786)

Minimum 80386 Wait States = 8
(16 MHz 80386, 8 MHz 82786)

Minimum 80186 Wait States = 3
(10 MHz 80186 and 82786, WT = 1)

Minimum 80186 Wait States = 2
(10 MHz 80186 and 82786, WT = 0)

The values mentioned above are for read cycles. In some cases, write cycles can operate with fewer wait states. For instance, the 80286 can execute 2 wait state synchronous write cycles. The 80186 can execute write cycles with one less wait state than mentioned above.

For asynchronous interfaces, if the CPU is operating at the same frequency as the 82786, the number of wait states are typically 1 more than those indicated above. For CPUs operating at a slower frequency than the 82786, the number of wait states are, on the average, less than 1 greater than those given above. In some cases, (eg. a 6 MHz 80286) an asyn-

chronous interface acutally has less wait states than those quoted above for the synchronous interface.

INTERNAL REGISTERS

The 82786 Internal Register block is relocatable by programming the address in the "Internal Relocation Register" in the BIU. The register block can be memory or I/O mapped. The Register Block is physically distributed between the three 82786 modules, BIU, Graphics Processor and Display Processor.

Accesses to reserved locations have no effect; they execute normally but may produce indeterminate read data. No register is altered when a write is executed to a reserved location.

Location of Internal Registers within 128 byte block:

Byte Address		
'00-0F H	BIU Registers	8 words
'10-1F H	reserved	
'20-2B H	GP Registers	6 words
'2C-3F H	reserved	
'40-49 H	DP Registers	5 words
'4A-7F H	reserved	

The BIU register map is as follows:

Byte Address	
BASE + 0H	Internal Relocation
BASE + 2H	Reserved
BASE + 4H	BIU Control
BASE + 6H	Refresh Control
BASE + 8H	DRAM Control
BASE + AH	Display Priority
BASE + CH	Graphics Priority
BASE + EH	External Priority

The field definitions for the BIU Registers are as follows:

Internal Relocation

	15	14	13	12	...	4	3	2	1	0
Addr =	Base					Address				MIO
BASE + 0H										

Reset values: xxx0

The Base Address determines the location of the 128 byte Internal Register Block. The MIO bit selects between memory or I/O mapping. If it is set (1), the Register Block is memory mapped. At RESET, the Base Address is set so that the Internal Relocation Register is located at every 128-byte address in the entire I/O space whenever CS is asserted. The Base Address must be written into this register before any other registers can be accessed.

BIU Control

6	5	4	3	2	1	0
VR	WT	BCP	GI	DI	WPI	WP2

Addr = BASE + 4H

RESET value: 0 1 0 0 0 0 0

- VR: If set (1) then the 82786 generates dual port video DRAM (VRAM) type memory cycles for display data fetch. If reset (0) then conventional page mode type memory cycles are performed to fetch display data.
- WT: Determines the minimum number of wait states possible in a synchronous 80186 interface. If set (1), there is a minimum of 2 (3) wait states during memory write (read) cycles.
- BCP: Determines whether the Internal Register block is accessed as bytes or words by the external CPU. If set (1), a 16 bit interface is selected.
- GI: Graphics Processor Interrupt. Set when the Graphics Processor issues an Interrupt. Cleared with RESET or a read of this register.
- DI: Display Processor Interrupt. Set when the Display Processor issues an Interrupt. Cleared with RESET or a read of this register.
- WP1: Write Protection One. When set (1), all BIU Register contents except for the WP1 and WP2 bits of this register are write protected.
- WP2: Write Protection Two. When set (1), all BIU Register contents are write protected, including WP1 and this bit, WP2. The only way to regain write access to the BIU registers after this bit is set, is to RESET the 82786.

Refresh Control

6	5	4	3	2	1	0
Refresh Scalar						

Addr = BASE + 6H

RESET value: 0 1 0 0 1 0

The Refresh Scalar is a 6 bit quantity that determines the frequency of refresh cycles to the Graphics memory.

$$\text{Refresh interval} = (\text{Scalar} + 1) * 16 * \text{Input clock period}$$

DRAM/VRAM Control

6	5	4	3	2	1	0
RW1	RW0	DC1	DC0	HT2	HT1	HT0

Addr = BASE + 8H

RESET value: 1 1 0 0 1 0 1

RW1:0: Number of Graphics memory Rows. One of the variables in defining the Graphics memory/External system boundary. Also disables RAS signals not driving any DRAMs/VRAMs.

RW1	RW0	
0	0	: 1 Rows
0	1	: 2 Rows
1	0	: 3 Rows
1	1	: 4 Rows

DC1:0: DRAM/VRAM Configuration. Controls the rate of block transfers and orientation of CAS1 and CAS0.

DC1 DC0

0	0	: Page Mode, Non-Interleaved
0	1	: Page Mode, Interleaved
1	0	: Fast Page Mode, Non-Interleaved
1	1	: Fast Page Mode, Interleaved

HT2:0: DRAM/VRAM Height. Defines the HEIGHT (not size) of each DRAM/VRAM chip in the system. All DRAMs/VRAMs must be the same size.

HT2 HT1 HT0

0	0	0	: 8K Devices
0	0	1	: 16K Devices
0	1	0	: 32K Devices
0	1	1	: 64K Devices
1	0	0	: 128K Devices
1	0	1	: 256K Devices
1	1	0	: 512K Devices
1	1	1	: 1M Devices

Display Processor Priority

6	5	4	3	2	1	0
FPL			SPL			

Addr = BASE + AH

RESET value: 1 1 0 0 1 1

Graphics Processor Priority

6	5	4	3	2	1	0
FPL			SPL			

Addr = BASE + CH

RESET value: 1 0 1 0 1 0

External CPU Priority

Addr = BASE + EH		FPL	
RESET value:	1	1	1

Specifies the priorities of the Display Processor, Graphics Processor and External CPU requests for the first request (FPL) and subsequent requests for block transfers (SPL). Code 111 is highest priority. Code 000 is lowest priority.

RESET AND INITIALIZATION

The state of $\overline{\text{BHE}}$ at trailing RESET determines synchronous vs. asynchronous operation. In Master mode, synchronous/asynchronous operation affects the sampling of the HLDA signal only. In Slave mode, synchronous/asynchronous operation affects the sampling of $\overline{\text{RD}}/\overline{\text{WR}}$ signals. Synchronous operation is set if $\overline{\text{BHE}}$ is sensed HIGH at trailing RESET. This enables direct connection of the 80286 $\overline{\text{BHE}}$ pin in synchronous systems since it is driven HIGH during RESET. The 80186 "tristates" its $\overline{\text{BHE}}$ during RESET so a small static load on this line can select asynchronous operation.

All internal registers are set to their default values on reset. The first slave I/O write access to the 82786 will always be directed at the Internal Registers (ignoring the upper fifteen address bits). The Internal Relocation Register must be programmed before any other Internal registers can be accessed. The DRAM/VRAM configuration registers must also be programmed to conform to any specific environment.

The 82786 assumes an 8 bit external CPU interface on reset. The graphics memory interface is always 16 bits wide. The BCP bit in the "BIU Control Register" must be set to 1 to enable a 16 bit external interface. Interrupts are cleared on reset.

GRAPHICS PROCESSOR

Introduction

The Graphics Processor (Graphics Processor) is an independent processor within the 82786. Its primary task is to draw bit-map graphics. It executes commands residing in the memory, accessing the memory through the Bus Interface Unit (BIU). The Graphics Processor addresses 4 MB of linear memory (22 bit addresses).

The Graphics Processor draws into a predefined area in the memory which is referred to as a "bit-map". A bit-map can be thought of as a rectangular drawing area composed of pixels. A coordinate system is defined for this bit-map with the origin at the

upper left corner, the x-coordinate increasing from left to right and the y-coordinate increasing from top to bottom. A bit-map can be up to 32K pixels wide and 32K pixels high.

The 82786 can draw several graphics primitives such as points, lines, arcs, circles, rectangles, polygons and characters. During the figure drawing process, the 82786 follows several programmable attributes.

The graphics attributes supported by the 82786 are:

- color
- depth (bits/pixel)
- texture
- logical operation
- color bit mask
- clipping rectangle

Each graphics primitive can be drawn in any one of 2, 4, 16 or 256 "Colors". The color details (bits/pixel and exact color) are programmable. The "Texture" controls the appearance of any line (or figure). The texture pattern can be up to 16 bits long thus enabling drawing of solid, dashed, dotted, dot-dash etc. types of lines. Each bit in the Texture corresponds to one pixel. The 82786 supports all sixteen binary "Logical Operation" between a figure being drawn in bit-map memory and the existing contents of memory. It is thus possible to overlay a figure on a background. The "Color Bit Mask" restricts the drawing operation to only some "color planes". The clipping rectangle restricts the drawing operation to a specific area in the bit-map.

The pixel information is stored in the bit-map memory in a packed pixel format. Different color bits for the same pixel are stored in adjacent bit positions within the same byte. Each byte represents 1, 2, 4 or 8 pixels (in one of 256, 16, 4 or 2 colors).

The Graphics Processor fetches its commands directly from a linked list Memory-resident Graphics Processor Command Block (GCMB). The GCMB is created and maintained by the CPU. The initial address for the GCMB is contained in a Graphics Processor Opcode Register in the 82786. Addresses for subsequent (next) GCMBs are contained in the previous GCMBs. The Graphics Processor can be forced to stop by appropriate commands.

When the Graphics Processor is idle, it is said to be in the "Poll State". This is the default mode after reset. While in the Poll State, the Graphics Processor continuously monitors its internal "Opcode Register". A valid command in this register starts the Graphics Processor. The first command placed in the internal Opcode Register must always be a "LINK" command directing the Graphics Processor to the main GCMB in memory.

Address Internal Opcode Register		Function	
BASE + 20h	GR0	OPCODE	GECL
BASE + 22h	GR1	Parameter 1	(Link Address Lower)
BASE + 24h	GR2	Parameter 2	(Link Address Upper)

Graphics Processor Internal Registers used in Poll State

Graphic Processor Command Format

The commands are placed (along with their parameters) sequentially in memory. Several GCMBs may be linked together through a LINK command. All commands have a standard format as described below:

15	8	7	1	0
OPCODE		0 0 0 0 0 0 0		GECL
Parameter 1				

Parameter i				
etc.				

Each command to the Graphics Processor consists of an opcode, a Graphics End of Command List

(GECL) bit and a list of parameters as required by the command. The opcode is 8-bits wide. The remaining 7-bits in the first word of the command must be all 0's to ensure future compatibility. Also, whenever a parameter for the command is an address, 32-bits have been set aside but the 82786 uses only 22-bit addresses. The user must ensure that the higher 10-bits in the address parameter are always all 0's. All commands must lie at even byte addresses.

After fetching each command, the Graphics Processor checks the GECL bit. If the GECL bit is not set, the command is executed and the next command is then fetched from the GCMB. If the GECL bit is found to be set, the Graphics Processor does not execute the command and enters a POLL state.

Graphics Processor Status Register

One of the 82786 internal registers contains the Graphics Processor Status Byte. The bits in the Status Byte are represented as:

Address BASE + 26H	GPOLL	GRCD	GINT	GPSC	GBCOV	GBMOV	GCTP	GIBMD
-----------------------	-------	------	------	------	-------	-------	------	-------

1. GPOLL - Poll State
Indicates if the Graphics Processor is in a POLL state.
2. GRCD - Reserved Command
This bit is set if the Graphics Processor encounters an illegal opcode.
3. GINT - This bit is set as a result of the INTR_GEN command.
4. GPSC - Pick Successful
This bit is set or cleared while the Graphics Processor is in the PICK mode. The bit is set if the pick operation resulted in success on any command.
5. GBCOV - bit-map Overflow for BitBit or CharBit
An attempt to execute a CHAR or a BitBit command with any portion of the destination rectangle lying outside the clip rectangle causes this bit to be set.
6. GBMOV - bit-map Overflow for Geometric Commands
An attempt to draw a pixel lying outside clip rectangle as a result of any geometric drawing commands (LINE, CIRCLE etc.) would cause this bit to be set. The reason for separating these two bits is the difference between the clipping operations for the two types of commands.
7. GCTP - Character Trap
This bit indicates that a character specified in the character string as a parameter for the CHAR command had its TRAP bit set.
8. GIBMD - Illegal Bit Map Definition
This bit is set if the DEF_BIT_MAP command is executed with illegal parameters. The illegal parameters are bits per pixel defined to be other than 1, 2, 4 or 8 or Xmax defined to be greater than 32k-1.

All the status bits except GPOLL are cleared upon reset. The GPOLL bit is set on reset.

Graphics Instruction Pointer

The Graphics Processor Instruction Pointer is a 22 bit quantity stored in two registers in the Graphics processor. It points to the current command in the GCMB.

		Address
GCIPL	Instruction Pointer Lower	BASE + 28h
GCIPIH	IP Upper	BASE + 2Ah

Clipping Rectangle

The 82786 can be instructed to restrict drawing to certain portion of the bit-map only. This portion is called the "Clipping Rectangle". The default clipping rectangle is the entire bit-map. The clipping rectangle must be redefined after a DEF_BIT_MAP command. For figures that are partially inside and partially outside the clipping rectangle, only the part inside the clipping rectangle is updated in the bit-map. For Block Transfer and Character drawing operations, if any part of the destination area lies outside the clipping rectangle, the command is not executed and the bit-map is left unchanged.

In order for the clipping to have predictable results, there are some restrictions on the x,y coordinates of each pixel. The rules to be observed are:

1. For lines, circles, polygons, polylines, BitBits and CharBits, each pixel lying on the figure (both the visible and the invisible parts) must not have its x or y coordinate outside $\pm 32K$ range.
2. For circular arcs, the above restriction applies to the circle of which the arc is a part.

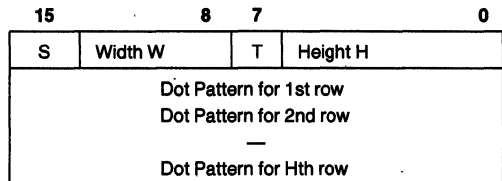
Pick Mode

The Graphics Processor can be put in the "PICK Mode" by executing the ENTER_PICK command. In the PICK Mode, the Graphics Processor performs all pixel computations for all drawing, BitBit and Character commands. However, the bit-map memory is not updated. Instead every computed pixel is compared against the clipping rectangle. If any computed pixel is found to lie within the clipping rectangle, the GPSC bit in the Graphics Processor Status Register is set.

Character Font Storage

The character fonts are stored in memory. Starting from an even address, the character information is

stored in consecutive words of memory forming a character block. Each block can be of different lengths for different characters. A character font is selected by programming its base address into the 82786 through the DEF_CHAR_SET command. The font could be established for 8 or 16 bit character codes. Each character block within a font has the following format:



S - Character Space bit

T - Trap Bit

Each character block must start at a word address and the dot patterns for each line of the font must reside in separate words. The height and width of each character cannot be more than 16 pixels. In case the width of a character is less than 16 pixels, the dot pattern for each line must be stored as right justified.

Note that width and height of the character refer to the difference between their limiting x and y coordinates respectively. Thus width = 0 specifies a character one pixel wide and a height = 0 specifies a character one pixel high.

Graphics Registers

All the registers in the Graphics Processor can be read from or written into, through the Graphics Processor commands - DUMP_REG and LOAD_REG. Each register is identified by a 9-bit Register Id.

These registers are different from the registers that are mapped into the 82786's On-Chip-Memory (I/O) space, i.e., they are accessible only through the DUMP_REG and the LOAD_REG command. There are four user accessible graphics registers. They are listed below.

REGISTER NAME	REGISTER-ID (# of bits)	REGISTER FUNCTION
GPOEM	0003 (6)	Poll Mask
GIMR	0004 (8)	Interrupt Mask
GSP	010C (21)	Stack Pointer
GCNT	0015 (16)	Character Count while drawing characters in bit-map

There are some other registers in the Graphics Processor. These registers are normally of no use to a user except in the event of saving and restoring them during a CPU context switch. Any other direct access to these registers must be avoided.

REGISTER NAME	REGISTER-ID (# of bits)
GP REG 5	0007 (2,2)
GP REG 6	010B (21)
GP REG 7	010D (21)
GP REG 8	010F (21)
GP REG 9	0010 (16)
GP REG 10	0011 (16)
GP REG 11	0012 (16)
GP REG 12	0013 (16)
GP REG 13	0016 (16)
GP REG 14	001C (4)
GP REG 16	0090 (16)
GP REG 17	0091 (16)
GP REG 18	0096 (16)
GP REG 19	0097 (16)
GP REG 20	0099 (16)
GP REG 21	009B (16)
GP REG 22	009C (16)

Graphics Processor Exception Handling

The status bits GPOLL, GRCD, GINT, GPSC, GBCOV, GBMOV, GCTP, and GIBMD are capable of generating an interrupt to the CPU depending upon the Interrupt Mask Register (GIMR). If the corresponding bit in the GIMR is a "0" an interrupt is generated. If another bit in the Graphics Processor Status Register is set before an acknowledgement for a previously generated interrupt, then another interrupt is not generated. Reading the Status Register serves the purpose of an Interrupt Acknowledge to the Graphics Processor. Reading the Graphics Processor Status Register clears the offending status bit(s) - bits not masked out in the Interrupt Mask. If the interrupt is generated due to the GPOLL bit, then this bit is not cleared on an interrupt acknowledgement. However this does not generate repeated interrupts.

The status bits GINT, GPSC, GBCOV, GBMOV, GTRP and GIBMD can also cause the Graphics Processor to stop its normal instruction fetch/execution and enter the POLL state. This is determined by the contents of the POLL On Exception Mask register (GPOEM). The GPOEM is 6 bits wide. If the corresponding bit in the GPOEM is a "0", the POLL state is entered. On entering the POLL state, the internal GECL bit in the GR0 register is automatically

set. When the processor is in the POLL state, it can be restarted by writing the appropriate opcode into the register GR0 (which writes a zero into the GECL bit). The act of clearing the GECL bit also causes the status bits that caused the POLL state to be cleared. Interrupt generation due to the GPOLL bit is enabled on exit from the POLL state.

The status bit GRCD when set, always causes the Graphics Processor to enter the Poll State. The Interrupt and the POLL mechanisms are two independent mechanisms. It is possible for the Graphics Processor to issue an interrupt and not POLL, or to issue an interrupt and POLL, or not to issue an interrupt and POLL or do none of them - all depending upon the GIMR and the GPOEM Register.

Initialization And Software Abort

There are two ways to force the Graphics Processor to enter the POLL state:

- i) An attempt to write into the Graphics Processor Status Register is considered a software ABORT signal.
- ii) An attempt to write into the Graphics Current Instruction Pointer also initiates a software ABORT.

The ABORT signal causes the Graphics Processor to enter POLL state after the execution of the currently executing command.

Upon RESET, the Graphics Processor immediately enters a well defined state. The following events take place:

1. Command execution is halted and the Graphics Processor enters the POLL state.
2. The GECL bit of register GR0 is set to indicate an End of Command List.
3. All status bits except GPOLL are cleared. GPOLL is set.
4. Interrupt Mask Register (GIMR) is set to all ones (disabled).
5. Poll on Exception Mask register (GPOEM) is set to all ones (disabled).
6. Graphics Processor exits the pick mode.

The Graphics Processor command set is divided into the following classes:

1. Non-Drawing Commands
2. Drawing Control Commands
3. Geometric Commands
4. Bit Block Transfer (BitBlit) Commands
5. Character Block Transfer (CharBlit) Commands

**List of Graphics Processor Commands
(Higher Byte - Hex)**

Command	Opcode	Command	Opcode
LINK	02	DEF_CHAR_ORIENT	4E
NOP	03	ABS_MOVE	4F
DEF_TEXTURE_OPAQUE	06	REL_MOVE	52
DEF_TEXTURE_TRANSPARENT	07	POINT	53
DEF_CHAR_SET_WORD	0A	LINE	54
DEF_CHAR_SET_BYTE	0B	RECT	58
INTR_GEN	0E	BIT_BLT	64
ENTER_MACRO	0F	ARC_EXCLUSION	68
EXIT_MACRO	17	ARC_INCLUSION	69
DEF_BIT_MAP	1A	POLYGON	73
DUMP_REG	29	POLYLINE	74
LOAD_REG	34	CIRCLE	8E
DEF_COLOR	3D	CHAR_OPAQUE	A6
DEF_LOGICAL_OP	41	CHAR_TRANSPARENT	A7
ENTER_PICK	44	BIT_BLT_M	AE
EXIT_PICK	45	INCR_POINT	B4
DEF_CLIP_RECT	46	HORIZ_LINES	BA
DEF_CHAR_SPACE	4D		

NON-DRAWING COMMANDS

NOP = No Operation	0300h			
LINK = Link to Next Command	0200h	Link Address Low	Link Address High	
INTR_GEN = Generate Interrupt	0E00h			
DUMP_REG = Dump Register	2900h	Dump Address Low	Dump Address High	Register ID
LOAD_REG = Load Register	3400h	Load Address Low	Load Address High	Register ID
ENTER_MACRO = Enter Macro	0F00h	Macro Addr Low	Macro Addr High	
EXIT_MACRO = Exit Macro	1700h			
HALT = Enter Poll State	xxx1h			

DRAWING CONTROL COMMANDS

DEF_BIT_MAP = Define bit-map	1A00h	Origin Addr Low	Origin Addr High	Xmax	Ymax	Bits/pixel
DEF_CLIP_RECT = Define Clip Rectangle	4600h	xmin	ymin	xmax	ymax	
DEF_COLORS = Define Colors	3D00h	Foreground Color	Background Color			
DEF_TEXTURE = Define Texture Opaque/Transparent	0600/0700h	Pattern				
DEF_LOGICAL_OP = Define Logic Operation	4100h	Color Bit Mask	Function Code			(see table below)

The functions performed and their codes are:

FCODE	FUNCTION	FCODE	FUNCTION
0000	0	1000	CMP (source) AND CMP (dest)
0001	source AND dest	1001	CMP (source) XOR dest
0010	CMP (source) AND dest	1010	CMP (source)
0011	dest	1011	CMP (source) OR dest
0100	source AND CMP(dest)	1100	CMP (dest)
0101	source	1101	source OR CMP (dest)
0110	source XOR dest	1110	CMP (source) OR CMP (dest)
0111	source OR dest	1111	1

DEF__CHAR__SET = Define Character Set (Word/Byte mode)	0A00/0B00h	Font Addr Low	Font Addr High
DEF__CHAR__ORIENT = Define Char Orientation	4000h	Path /Rotation	

There are four defined values for both the path and rotation. They are:

CODE	INCREMENT
00	0 degrees
01	90 degrees
10	180 degrees
11	270 degrees

DEF__CHAR__SPACE = Define Inter Char Space	4D00h	Inter Char Space	
ABS__MOV = Move	4F00h	x coordinate	y coordinate
REL__MOV = Relative Move	5200h	dx	dy
ENTER__PICK = Enter Pick Mode	4400h		
EXIT__PICK = Exit Pick Mode	4500h		

GEOMETRIC COMMANDS

POINT = Draw Point	5300h	dx	dy	
INCR__POINT = Draw Incremental Points	B400h	Array Addr Low	Array Addr High	N (# of pts)

INCREMENTAL POINTS ARRAY

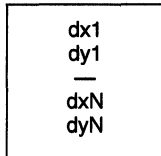
INC4	INC3	INC2	INC1
—	—	—	—
—	incN	incN-1	incN-2

The upper two bits of the “inc” field specify the increment for the x coordinate while the lower two bits specify the increment for the y coordinate. The encoding for the two bits is as follows:

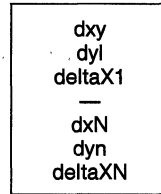
CODE	INCREMENT
00	0
01	+1
10	-1
11	Unused

LINE = Draw Line	5400h	dx	dy	
CIRCLE = Draw Circle	8E00h	radius		
RECT = Draw Rectangle	5800h	dx	dy	
POLYLINE = Draw Polyline	7400h	Array Addr Low	Array Addr High	N (# of lines)
POLYGON = Draw Polygon	7300h	Array Addr Low	Array Addr High	N (# of lines)

POLYLINE/POLYGON ARRAY



HORIZONTAL LINE ARRAY



ARC = Draw Arc (Exclusion/Inclusion)	6800/6900h	dxmin	dymin	dxmax	dymax	radius
HORIZ_LINES = Draw Series of Horizontal Lines	BA00/BA01h	Array Addr Low	Array Addr High	N (# of lines)		

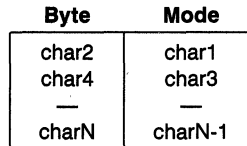
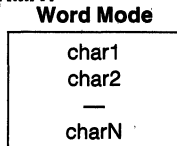
BITBLT COMMANDS

BIT__BLT = Bit Block Transfer within bit-map	6400h	Source x coord	Source y coord	dx	dy
BIT__BIT__M = Bit Block Transfer across bit-maps	AE00h	Source Addr Low	Source Addr High	Source Xmax	
	Source Ymax	Source x coord	Source y coord	dx	dy

CHARBLT COMMANDS

CHAR = Draw Character String (opaque/transparent)	A600/A700h	String Ptr Low	String Ptr High	N (# of char)
--	------------	----------------	-----------------	---------------

CHARACTER STRING FORMAT



DISPLAY PROCESSOR

Introduction

The Display Processor (Display Processor) is an independent processor responsible for controlling the display of video data on a CRT, laser printer and other display devices. Its functions include the generation of horizontal and vertical timing signals, blanking signal and the control of 8 Video Data output pins.

The 82786 can function in two distinct types of graphics memory environments – i) using single port DRAMs (normal display mode) and ii) using dual port video DRAMs (VRAM mode). When the 82786 is configured to interface with single port DRAMs, the Display Processor uses the BIU to fetch the screen parameters and display data from memory. The Display Processor then internally shifts the video data into the video stream for screen refresh. When configured to run with VRAMs, the Display Processor uses the BIU to load the shift registers in the VRAMs at the beginning of every scan line. The screen

refresh is then done by the second port of the VRAMs. The BIU and Graphics Processor have the rest of the scan line time to access the graphics memory.

Bit Map Organization

The Display Processor is optimized to display data in packed bit-map form. The Graphics Processor writes pixel data in the memory in this form. The Display Processor supports display of 1, 2, 4 or 8 bits/pixel data, stored in sequential bit-map form, with the first (left-hand) pixel to be displayed occupying the Most Significant Bit(s) of a word in memory, and subsequent pixels occupying sequentially lower bits in the word. Ascending word addresses represent subsequent pixels, moving left to right and top to bottom on the screen.

Windows and Normal Display Mode

In the normal display mode, Windows may be displayed on the screen in a flexible format. There can be up to 16 window segments or tiles appearing on any single display line. There is no limit on the number of windows vertically (limited by the number of scan lines in the active display area). At the basic video rate (25 MHz, 8 bpp), these windows may be placed at pixel resolution on the screen, and mapped at pixel resolution into the bit-map. Windows can be made to overlap, by breaking the windows into tiles and assembling the tiles on the screen.

Cursor (Normal Display Mode)

The Display Processor supports a single hardware cursor which may be 8 x 8 pixels or 16 x 16 pixels. This cursor may be positioned anywhere on the screen with a pixel resolution. The cursor may be defined to be transparent or opaque, and may be either a block cursor with its hot-spot at the top-left of the cursor pattern, or a cross-hair cursor one pixel across, stretching the width and height of the screen with its hot-spot at the center of the cross. The cursor color and pattern (shape) are programmable. The cursor may be programmed off if not required, or to implement a blinking cursor.

Video Rates (Normal Display Mode)

The Display Processor is clocked from an external Video Clock. In this mode, the 82786 fetches video data from memory into an internal FIFO. An internal shift register then generates the serial video data

stream to the display. The 82786 will support CRT screens of up to about 640 x 480 pixels at 8 bits/pixel and 60 Hz non-interlaced, or about 1024 x 640 x 8 at 60 Hz interlaced. The Display Processor supports Interlaced, non-interlaced and interlace-sync displays.

The Display Processor also has higher speed modes which enable the user to trade off bits/pixel for dot-rate. Thus it is possible to run at a maximum of 8 bpp with a 25 MHz dot-rate, 4 bpp at a 50 MHz dot-rate, 2 bpp at a 100 MHz dot-rate or 1 bpp at 200 MHz dot-rate; with corresponding increase in size and resolution. Note that in the high speed modes, horizontal window and cursor placement resolution is reduced to 2, 4 or 8 pixel resolution at 50 MHz, 100 MHz, or 200 MHz rates respectively.

VRAM Mode

In the VRAM mode, the first tile for every scan line is used to load the shift register in the VRAMs by executing a data transfer cycle. Subsequent tiles (if any) for all strips will still be available through the VDATA pins of the 82786. The window status bits can be used to internally multiplex the VRAM video stream and the 82786 generated video stream. The address for this data transfer cycle is determined from the Tile Descriptor. The 82786 BEN# pin is used as a DT pin for this case. If the graphics memory banks are interleaved, then both banks are loaded in the transfer cycle. During the Blank period, Default VData appears on the VDATA pins.

CRT Controller

CRT timing signals HSYNC, VSYNC, and BLANK are each programmable at a pixel resolution, giving a maximum display size of 4096 x 4096 pixels. If high-speed or very-high-speed display modes are selected, the horizontal resolution of the CRT timing signals becomes 2 pixels, 4 pixels or 8 pixels at 50 MHz, 100 MHz and 200 MHz respectively.

Window Status

The HSync and VSync CRT timing pins may be configured to serve as Window Status output pins, which can be programmed to present a predefined code while the Tile Processor is displaying a tile. This code is programmable as part of the Tile Descriptor, and may be used externally to multiplex in video data from another source, or select a palette range for a particular window, etc. External logic must be used to enable VSync and HSync as CRT timing signals when Blank is high, and as encoded Window Status signals when Blank is low. This is valid in both DRAM and VRAM modes.

Zoom Support

The Display Processor allows windows to be zoomed in the normal display mode. The zoom factor is an integer between 1 and 64. There are independent zoom factors for the x and y direction. The zoom function results in pixel replication.

All zoomed windows on a display are zoomed by the same amount. A window is therefore either zoomed or not zoomed. Zoom offset is not supported—a pixel must either be fully displayed or not displayed at all. This places a restriction on window placement—a window may not be placed such that a zoomed pixel is partially obscured. VRAM displays can be zoomed vertically by using this feature. Horizontal zooming of VRAM windows requires external hardware support.

Extended 82786 Systems

The CRT timing signal pins may be configured as output pins (for the normal stand-alone 82786 system), or as input pins for a system in which multiple 82786's are ganged in parallel to provide a greater number of bits/pixel, higher dot rates, larger display area, or more windows. In multiple 82786 systems, each of the Display Processors run in lock step, allowing the individual outputs to be combined on a single display. The HSync, VSync and Blank pins for the "Slave" 82786 are configured as inputs and are driven by the "Master" 82786.

When programmed as inputs, VSync and HSync still serve as outputs for Window Status while Blank is inactive.

External Video Source

The HSync and VSync pins on the 82786 can be configured as inputs to synchronize the 82786 to external video sources (VCR, broadcast TV etc.). In this case, the Blank pin is configured as output and the active 82786 display period is determined by the programmed 82786 parameters.

Memory Bandwidth Requirements

The memory bandwidth required by the Display Processor depends on the display size and mode of operation. The 82786 has a 40 Mbyte/sec maximum bandwidth during fast block accesses to Graphics Memory. In the normal display mode the Display Processor makes use of these fast block reads for screen refresh, thereby minimizing its use of the memory bus, which the other 82786 modules share. For worst-case displays, when the Display Pro-

cessor is running at its maximum speed of 25 MHz and 8 bits/pixel, about 50% of the memory bandwidth is used for display refresh. Correspondingly, at only 1 bit/pixel the Display Processor's bus requirements are reduced to about one-eighth of its requirement at 8 bpp. In the VRAM mode, the Display Processor does not fetch any of the display data. The display data is passed directly from the graphics memory to the pixel logic. In this case about 1% of the graphics memory bandwidth is required by the Display Processor to fetch the Strip Descriptors. The Display Processor Access to memory can be "tuned" through a programmable trip point for the Display Processor Data FIFO. This controls the length and frequency of block transfers to fill the FIFO of display data fetches.

Display Processor Registers

There are two different register sets for the Display Processor. Six of the 82786 Internal Registers are dedicated to the Display Processor. These registers are memory (or I/O) mapped in the external CPU address space. They can therefore be directly accessed by the external CPU. Another set of registers is totally local to the Display Processor. These are the display control registers and are used for display parameters.

82786 Registers For Display Processor

There are six of these Registers. They are listed below:

Address	Function
Base + 40	Display Processor Opcode
Base + 42	Param1
Base + 44	Param2
Base + 46	Param3
Base + 48	Display Processor Status
Base + 4A	Default Video

The Display Processor Opcode and the three parameter registers are used to send a command to the Display Processor. The Display Processor Status Register contains the status for the Display Processor. This is described in more detail later. The Default Video Register contains the data that appears on the Video Out pins during the blanking intervals. The CPU can use this register to address an external palette RAM while loading the palette, thereby eliminating a separate address path and external logic.

Display Control Registers

The display control registers can be loaded under control of the Display Processor during the Vertical Blanking interval. This synchronizes parameter updates with display refresh and ensures that the display remains clean, with no updates occurring during data display.

The Display Processor may also be programmed to provide a Frame Interrupt once per certain number of frames. This may be used to facilitate blinking, scrolling, panning, animation or other periodic functions.

Command Execution

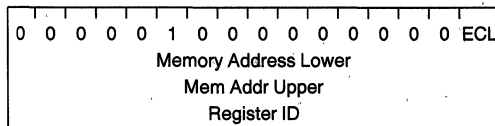
At the beginning of each Vertical Blanking time, the Display Processor checks the ECL bit in the Display Processor Opcode Register. If the ECL bit is 1, the Display Processor status remains unchanged. If the ECL bit is 0, the Display Processor executes the command. Only one command is executed per frame.

On completion of the command, the Display Processor sets its ECL bit back to 1, indicating to the CPU that a new command may be written into the Command Register. This handshake prevents the CPU from writing a new command before the old one has finished executing. The commands for the Display Processor are:

1. Load Register
2. Load All Registers
3. Dump Register
4. Dump All Registers

The command formats are:

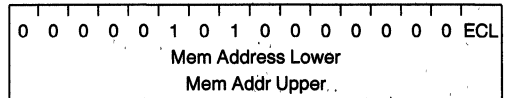
LOAD REGISTER (LD—REG):



This command loads a pair of display control registers with values stored in memory starting at the lo-

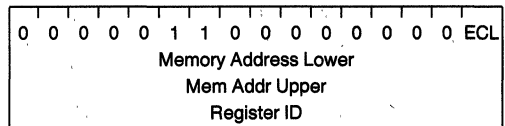
cation given by Memory Address. The Memory Address must be an even byte address. The Register ID for the register pair is given in the register block description below. This command may be used to update individual pairs of registers (such as the Cursor Position registers).

LOAD ALL (LD—ALL):



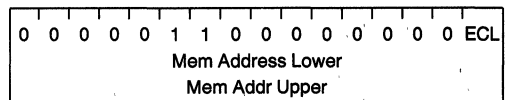
This command loads the entire block of display control registers in a block read starting from the Memory Address given in the command. The Memory Address must be an even byte address. This command must be the first command executed and has to be executed after reset to enable the display operation. The registers are listed below.

DUMP (DMP—REG):



This command causes the Display Processor to write the contents of the display control register pair specified by Register ID to the location in memory specified by Memory Address. The Memory Address must be an even byte address.

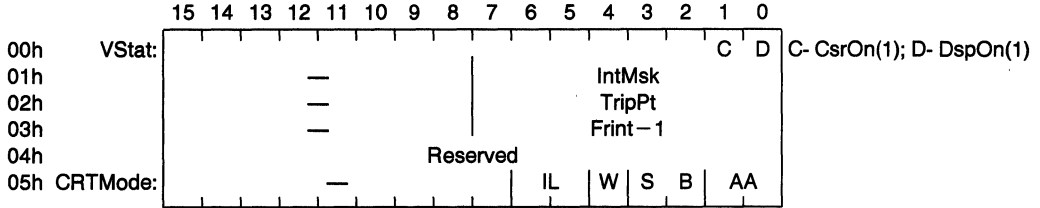
DUMP ALL (DMP—ALL):



This command causes the Display Processor to write its entire display control register block out to a block in memory, starting at the Memory Address specified. The Memory Address must be an even byte address. The write occurs as a series of single write cycles.

Display Control Register Block

The display control register block is shown below. Each register is 16-bits wide. The numbers in parentheses are the number of bits per parameter.



- IL - Interface(2): 00 → Non-Interface
- 01 → Reserved
- 10 → Interface
- 11 → Interface-Sync

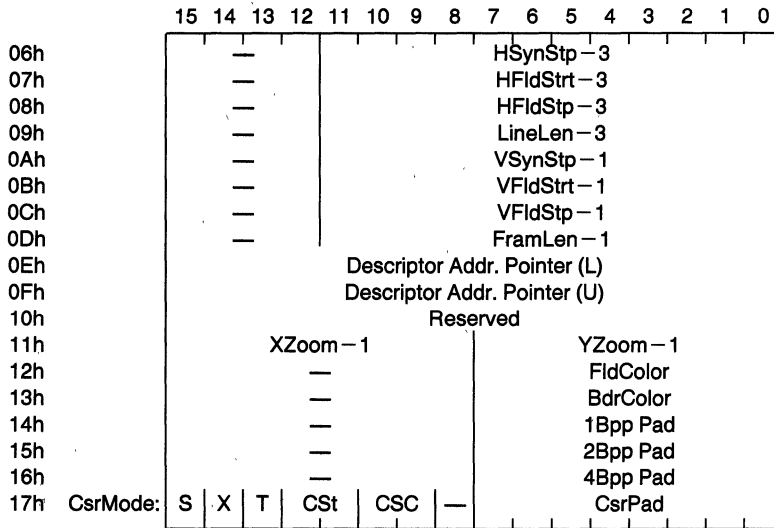
W - Window Status Enable(1)

S - HSYNC, VSYNC Slave Mode(1)

B - Blank Slave Enable(1)

AA - Accelerated Video (High Speed Video, etc.)(2)

- 00 → Normal (25 MHz)
- 01 → High Speed (50 MHz)
- 10 → Very High Speed (100 MHz)
- 11 → Super High Speed (200 MHz)



- CsrStyle: S - CsrSize(1): 0 → 8 x 8 Csr
- 1 → 16 x 16 Csr

X - CsrX-Hair(1)

T - CsrTransparent(1)

CSt - CursorStatus (to Window Status output)(2)

- CSC - CursorStatusControl(2): 00 → Current Window Status
- 01 → Foreground
- 10 → Background
- 11 → Block

8. **HfIdStp**
Enter the number of VCIks minus 3 between the rising edge of HSYNC and the rising edge of the next BLANK (the end of Video Data).
9. **LineLen**
Enter the number of VCIks minus 3 between the rising edge of HSYNC and the rising edge of the next HSYNC.
10. **VSynStp**
Enter VSYNC width minus OAP in number of HSYNC periods. In the non-interlaced mode, VSYNC rises and falls on the rising edge of HSYNC. In interlaced and interlace-sync mode, VSYNC has the same timing as in non-interlace mode at the start of each Even Field (lines 0, 2, 4, etc), but is delayed by half LineLen at the start of each Odd Field (lines 1, 3, 5, etc). (See Figure 3.)
11. **VfIdStrt**
Enter the number of HSYNCs minus one between the beginning of VSYNC and the end of vertical BLANK.
12. **VfIdStp**
Enter the number of HSYNCs minus one between the beginning of VSYNC and the beginning of the next vertical BLANK.
13. **FramLen**
Enter the number of HSYNCs minus one between the beginning of VSYNC and the beginning of the next VSYNC.
14. **Descriptor Address Pointer (L)**
The address of the first Strip Descriptor for the display. After fetching the first descriptor the Display Processor uses the Link Address in the descriptor to fetch the next descriptor. The Descriptor address must be an even byte address.
15. **Descriptor Address Pointer (U)**
The most significant end of the Descriptor Address Pointer.
16. This field should always be set to zero.
17. **ZoomX, ZoomY**
The x-zoom minus one and y-zoom minus one factors for the zoomed windows. Can be any integer number between 1 and 64. In the VRAM mode, ZoomX is not used.
18. **Field Color**
An 8-bit value indicating the color of the background field to be displayed in the absence of windows.
19. **Border Color**
An 8-bit value indicating the color of the border to be displayed inside selected windows.
20. **1Bpp Pad**
An 8-bit value where the upper 7 bits represent the upper 7 bits of video data concatenated to the 1 bit video data from a 1 bit/pixel bit-map.
21. **2Bpp Pad**
An 8-bit value where the upper 6 bits represent the upper 6 bits of video data concatenated to the 2 bit video data from a 2 bit/pixel bit-map.
22. **4Bpp Pad**
An 8-bit value where the upper 4 bits represent the upper 4 bits of video data concatenated to the 4 bit video data from a 4 bit/pixel bit-map.
23. **CsrStyle:S(1) X(1) T(1) CSt(2) CsrPad**
The Cursor Mode Register. The Cursor Pad is an 8-bit value where the upper 7 bits are the higher 7 bits for the cursor color.
Cursor Style: S is the size bit. If S is 0 an 8 x 8 pixel cursor will be displayed. If S is 1, a 16 x 16 pixel cursor will be displayed.
X is the Cross-Hair Mode bit. If X is 0, a block cursor will be displayed. The pattern for the cursor is specified in the Cursor Pattern registers. The cursor hot-spot is at the top-left of the cursor block. If X is 1, a cross-hair cursor will be displayed. Its hot-spot is at the center of the cross, and it will stretch the full height and width of the display.
T is the Transparent Mode bit. If T is 0, the cursor is opaque. Its foreground color is determined by the concatenation of the cursor padding bits (7 MSB's) with 1. The background color is determined by the concatenation of the cursor padding bits with 0. If T is 1, the cursor background reverts to whatever bit-map data is being displayed "behind" the cursor.
CSt is the Cursor Status. The code to be output onto the Window Status outputs while the Cursor is being displayed.
CSC is the Cursor Status Control (2 bits). The cursor status may be output whenever the cursor foreground color is being output, whenever the cursor background color is being output, or whenever the cursor block is active, whether it is displaying background color or foreground color or transparent pixels (useful for inverse video), or else the cursor status may default to the current Window Status. The code is shown above.
24. **CsrPos X**
This is the Cursor X-Position Register—the position of the cursor hot-spot relative to the beginning of the line (the rising edge of the previous HSYNC). Enter the value minus 2.

25. CsrPos Y

This is the Cursor Y-Position Register—the position of the cursor hot-spot relative to the beginning of the frame (the beginning of the previous VSYNC). Enter the value minus one.

26. CsrPat0:F

These 16 registers contain the pattern to be displayed as a cursor. CsrPat0 is the top row of the cursor, and the MSB is the left bit of the cursor. For an 8 x 8 cursor, the cursor pattern used is the higher byte of the first eight cursor registers.

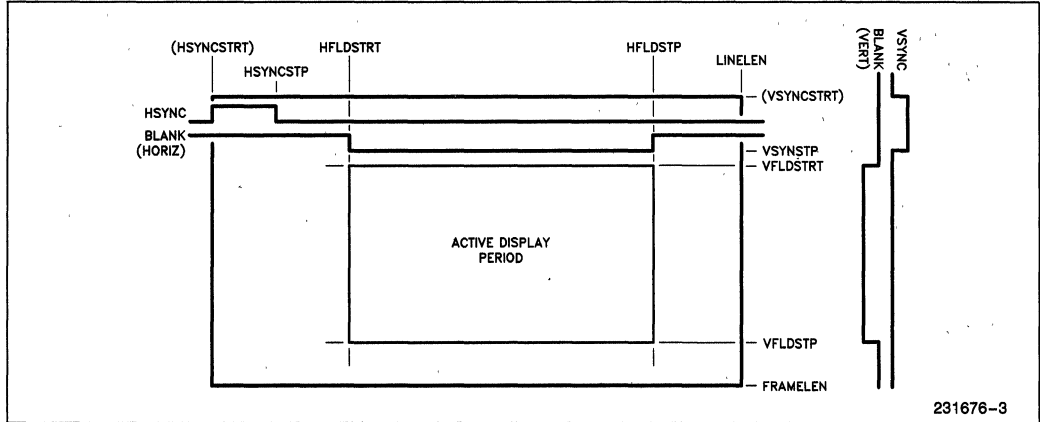


Figure 3. Timing Parameters

Windows

The CPU creates Strip Descriptors in memory that describe windows for the Display Processor. The Strip Descriptors are organized as one descriptor per strip of window segments (tiles) as shown in Figure 4. Each descriptor contains information for the tiles within that strip in the order they are displayed on the screen (left to right). The descriptor for a particular strip must be contiguous in memory. The Strip Descriptors for several strips are linked to each other in the order they are displayed (top to bottom).

The linking is done through a field in each descriptor that points to the following descriptor. The descriptor for the first strip is accessed during the VBlank interval, using an address specified by the Descriptor Address Pointer, one of the display control register pairs.

The strip descriptor consists of a header followed by one or more tile descriptors. The header and tile descriptors must occupy one contiguous block in memory.

The format of the Window Strip Descriptors is:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Header																Number of Lines in Strip - 1		
																Link to Next Strip Descr. (L)		
																Link to Next Strip Descr. (U)		
1st Tile Descr.	C															Number of Tiles in Strip - 1		
																bit-map Width		
																Mem Start Address (L)		
																Mem Start Address (U)		
																Bpp		
																StartBit		
																StopBit		
																Fetch Count (bytes - 2)		
2nd Tile Descr.	T	B	L	R	WSt											PC	Z	F
																bit-map Width		
																Mem Start Address (L)		
																Mem Start Address (U)		
																Bpp		
																StartBit		
																StopBit		
																Fetch Count (bytes - 2)		
	T	B	L	R	WSt											PC	Z	F

etc ...

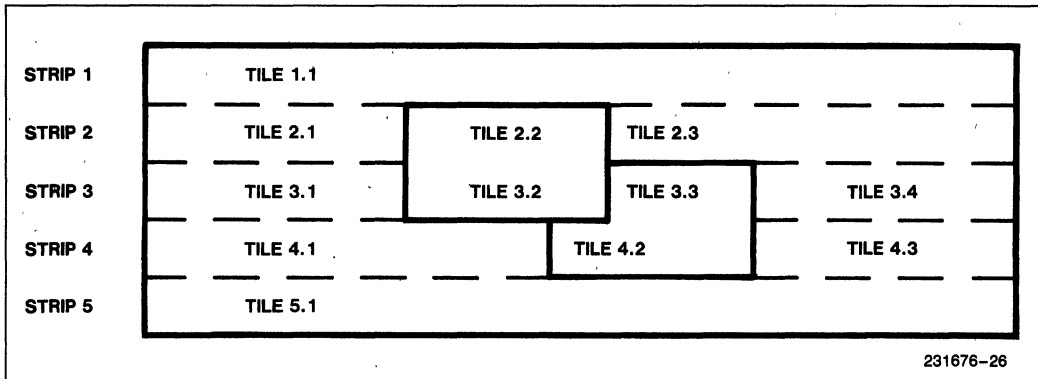


Figure 4. Display Shows Strips and Tiles with Two Overlapping Windows

The strip descriptor header is programmed with values for the number of display lines minus one and the number of tiles in the strip minus one. There may be any number of lines in a strip, up to the number of lines on the display (within their restrictions imposed by zoom, if used). In DRAM mode there may be up to 16 tiles within a single strip. In the VRAM mode the first tile is used to load the VRAM shift register, leaving 15 tiles to be used by the Display Processor. The header also contains a link to the next strip descriptor.

The C bit (the most significant bit) in the Number of Tiles in Strip parameter tells the DP to color the display area following the current strip with FldColor data or link to the next strip. If the C bit is set to one, the DP colors the remainder of the display with the background color defined in the FldColor Register. If the C bit is zero, the DP links to the next strip.

Each tile descriptor contains the following parameters:

1. **Bit-Map Width**—the width of the bit-map in bytes. This must be even and is added to the Memory Address for each scan line in the window, each HSync period within the strip to get the start address of the next display line (if y-zoom inactive or counted out). In case of interlaced displays, the Memory Address is incremented by twice the bit-map width. In the VRAM mode, the bit map width of the first tile must be a power of 2 and must be less than the maximum width of the VRAM shift register.
2. **Memory Start Address**—the memory address for the window. This is an even byte address, corresponding to the address of the first word of bit-map data for the window tile (top left corner). In the VRAM mode the start address for the first tile must guarantee that the entire scan line is contained in a single row of the VRAM.
3. **Bpp**—The number of bits/pixel in the current window—must be programmed to 1, 2, 4, or 8 in the normal mode. In the VRAM mode this field should be zero.

4. StrtBt—Start Bit—The bit position in the corresponding memory word for the first bit of the first pixel in the window. Gives bit resolution to the Memory Start Address (and pixel resolution to the start of the window). In the normal mode this must be programmed to be consistent with the Bpp defined for that window. In the VRAM mode, this must be programmed to zero for the first tile.
5. StopBit—The bit position in the corresponding memory word for the last bit of the last pixel in the widow. Gives bit resolution to the window width. In the normal mode this must be programmed to be consistent with the Bpp defined for that window. Illegal value will result in incorrect display. In the VRAM mode, this must be programmed to zero for the first tile.
6. Fetch Count—In the DRAM mode, this specifies the number of bytes minus two from the bit-map to be fetched for each scan line in the current window tile. This must be an even quantity. The value programmed in this field is 2 less than the number of bytes to be fetched rounded off to the next higher even number. In the VRAM mode, this must be programmed to zero for the first tile.
7. TBLR—Border Control Bits—Indicate border on Top, Bottom, Left or Right of window tile. This is a four bit field with one bit controlling each border. The most significant bit controls the top border and the least significant bit controls the right border.
8. WST—Window Status (2 bits)—The code to be presented on the Window Status pins while the window is being displayed.
9. PC—IBM PC Mode—Indicates that this window is being displayed from a bit-map created in IBM PC format. The Display Processor supports the IBM Color Graphics Adapter bit-map format in which the least significant byte of a word appears on the left of the most significant byte on the screen as opposed to the 82786 format in which the least

significant byte appears to the right of the most significant byte. Also, the 2-bank and 4-bank bank oriented bit-maps used in the PC and PCjr systems are supported. These modes enable bit-maps created by IBM PC or PCjr (or compatible) systems to be upward compatible with 82786 displays, with the PC format bit-maps being displayed either as the whole screen, or as windows on a screen together with 82786 created bit-maps. The PC mode bit-maps can be zoomed or used with interlaced or accelerated displays. In the VRAM mode, this field must be programmed to zero for the first tile.

Note that although the Display Processor can display bit-maps created in these formats, the Graphics Processor always draws bit-maps in 82786 format. The vertical mapping of IBM format bit-maps is restricted in that the Memory Start Address of an IBM format window must be in the first of the 2 or 4 banks.

The coding for IBM PC mode is given below:

- 00 → 82786 Mode
- 01 → Swapped Byte Mode
- 10 → Swapped Byte, 2 banks
- 11 → Swapped Byte, 4 banks

Bit-map formats in 82786 and PC Modes are shown below:

Pixel # (from left as displayed on screen):

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
82786 Mode Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC Mode Bit #	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8

- 10. Z—Zoom—This bit if set, indicates that in the normal display mode the window is to be zoomed using the zoom parameters programmed into the ZoomX and ZoomY registers.
- 11. F—Field—This bit if set, indicates that the window tile is background field. In the normal mode the field color is displayed for the window. The number of pixels of Field to be displayed should be programmed into what would normally be the Bpp, StartBit, StopBit fields. This bit must be set to zero for a first tile in the VRAM mode.

If the Strip Descriptor list causes a window to be displayed that extends beyond the active display area, then only the upper left hand portion of the window is displayed and the rest of it is truncated.

In interlace mode, in order to maintain a line resolution on vertical positioning of windows, a double-length descriptor table must be used. The first part contains window position information for the even lines, the second part for the odd lines. Also note that in interlace mode, one frame takes two fields to display. Command execution occurs at frame boundaries, not field boundaries, so the instruction execution frequency will typically be 25/30 Hz instead of the non-interlaced 50/60 Hz.

Initialization

The Display Processor is reset during the main 82786 reset process. Upon reset it enters a well-defined reset state described below:

- 1. Any command execution is immediately halted.
- 2. Parameter, Descriptor, or Display Data fetches are terminated.
- 3. Display Outputs VDATA7:0 are all reset to default video.
- 4. HSync, VSync, Blank are tristated (Display Processor defaults to Slave Operation). These stay tristated until the first LOAD_ALL instruction.
- 5. Display Processor Status Register is cleared.
- 6. Display Processor Interrupt Mask to set to all 1's (all interrupts disabled).
- 7. ECL bit is set to 1.

Display Processor Interrupts, Status Register and Exception Handling

The Display Processor Status Register is an 8-bit memory (or I/O) mapped register which indicates the current status of the Display Processor, and allows the generation of interrupts depending on

the state of individual bits in the status register. Interrupts may be masked off using the Display Processor Interrupt Mask register. The format of the Display Processor Status Register is:

ADDRESS 7 6 5 4 3 2 1 0

BASE + 48 h

FRI	RCD	DOV	FMT	BLK	EVN	ODD	ECL
-----	-----	-----	-----	-----	-----	-----	-----

Display Processor Status Register

The functions of each bit, and the action taken in the case of exceptions is described below:

- 1. FRI—Frame Interrupt. This bit is set every n frames, where n is a value between 1 and 256 loaded into the Frint Register. This may be used, for example, for timing in animation applications, or to time blink rates.
- 2. RCD—Reserved Command. This bit is set if the Display Processor does not recognize the Opcode it has been instructed to execute. The Display Processor will not execute the command.
- 3. DOV—Descriptor Overrun. This bit is set if the Display Processor has not completed its descriptor load when horizontal blanking ends.
- 4. FMT—FIFO Empty. This indicates that the Display FIFO has underrun. This forces an End of Line condition and the rest of the Display Line will display the Default VData color. At the beginning of HBlank, the Display Processor uses the current descriptor to start a new Display Data fetch. A FIFO underrun therefore does not necessarily mean that the whole field is lost—just the current display line is corrupted.
- 5. BLK—Blank. This indicates that the BLANK pin is currently active.
- 6. EVN—Even Field. In Interlace and Interlace-Sync modes, this bit is set during the even field (Field 1).
- 7. ODD—Odd Field. In Interlace and Interlace-Sync modes, this bit is set during the odd field (Field 2). The Even and Odd status bits assist in synchronizing the 82786 with other interlaced display systems.
- 8. ECL—End of Command List. This is set at the same time the ECL bit in the Opcode Register is set, and allows the Display Processor to inform the CPU as soon as it has completed execution of a command.

All active interrupts are OR'ed together to drive a single 82786 interrupt line. Once set, the interrupt line remains active until the Status Register is read. The active bits in the Status Register (bits with 0 in the corresponding bit in the Interrupt Mask) are reset to zeroes after the Status Register is read.

Test Modes

The 82786 implements several special modes of operation beyond normal use to aid in debug, characterization and production testing. When RESET goes inactive, the RD and WR pins are sampled. If either of these two pins is low, one of the special test modes is enabled according to the state of RD, WR and MIO pins.

The 82786 implements three global pin conditioning features. Specifically, the 82786 can drive all output and I/O pins high, or low, or can tristate all pins. The test modes are activated according to the following table:

RD#	WR#	MIO	Mode
0	0	0	Reserved
0	0	1	Reserved
0	1	0	Reserved
0	1	1	Drive Output Pins High
1	0	0	Drive Output Pins Low
1	0	1	Tristate Pins
1	1	X	Normal Operation

NOTE:

All timing numbers in the parametric section are preliminary and are subject to change.

V_{OL}/V_{OH} Pin Conditioning

The 82786 has the capability to bring all its output pins to a constant logic high (or low) state. This feature can be used for testing the output buffers on the 82786.

Tristate Feature

The 82786 has the ability to tristate all of its I/O and output pins to effectively isolate the 82786 from any connected circuitry. This allows testing a completely assembled PC board by isolating the 82786. Leakage on all I/O pins can also be tested in this mode.

82786 PARAMETRICS

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65°C to +150°C
Operating Temperature	0°C to 70°C
Voltage V _{CC} -V _{SS}	-0.5V to +6.5V
Voltage on Other Pins	-0.5V to V _{CC} + 0.5V

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

Symbol	Parameter	Min	Max	Units	Notes
V _{ILC}	Input Low Voltage	-0.5	+0.8	V	CLK Input
V _{IHC}	Input High Voltage	+2.0	V _{CC} + 0.5	V	CLK Input
V _{ILVC}	Input Low Voltage	-0.5	+0.8	V	V _{CLK} Input
V _{IHVC}	Input High Voltage	+2.0	V _{CC} + 0.5	V	V _{CLK} Input
V _{IL}	Input Low Voltage	-0.5	+0.8	V	All Other Pins
V _{IH}	Input High Voltage	+2.0	V _{CC} + 0.5	V	All Other Pins
V _{OL}	Output Low Voltage	—	+0.45	V	All Pins I _{OL} = 2.0 mA
V _{OH}	Output High Voltage	+2.8	—	V	All Pins I _{OH} = -400 μA
I _{LI}	Input Leakage Current	—	±1	μA	0 < V _{IN} < V _{CC}
I _{LO}	Output Leakage Current	—	±10	μA	0.45 < V _{IN} < V _{CC}
I _{CC}	Power Supply Current	—	200	mA	@ 0°C Temp CLK @ 20 MHz V _{CLK} @ 25 MHz

A.C. CHARACTERISTICS $T_A = 0^\circ \text{ to } 70^\circ\text{C}$, $V_{CC} = 5V \pm 10\%$
CLOCK and RESET Timings

AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units	Notes
T_C	CLK Period	50	500	ns	@ 1.5V
T_{CL}	CLK Low Time	15	—	ns	@ 1.5V
T_{CH}	CLK High Time	15	—	ns	@ 1.5V
T_{CR}	CLK Rise Time	—	10	ns	@ 0.8V–2.0V
T_{CF}	CLK Fall Time	—	10	ns	@ 0.8V–2.0V
T_{R1}	Test Input Setup Time	10	—	ns	
T_{R2}	Test Input Hold Time	5	—	ns	
T_{R3}	Reset Active Hold Time	25	$2 T_C$	ns	
T_{R5}	Reset Inactive Hold Time	10	—	ns	
T_{R6}	Reset Active Setup Time	10	—	ns	
T_{R7}	Forced Output Delay	30	—	ns	
T_{R8}	Reset Width	$10 T_C$	—	ns	

DRAM Interface Timings

AC timings are referenced to 0.8V/2.4V on all pins and are valid for total DRAM capacitance on each pin between 30 pF and 200 pF

SINGLE READ, WRITE, READ MODIFY WRITE AND PAGE MODE CYCLES

Symbol	Parameter	Min	Max	Units
T_{RC}	Single Cycle Time	$6 T_C - 5$	—	ns
$T_{RAC}^{(1)}$	Access Time from RAS#	—	$4 T_C - 20 - 0.050 C_R$	ns
$T_{CAC}^{(1)}$	Access Time from CAS#	—	$2 T_C + T_{CH} - 15 - 0.050 C_C$	ns
$T_{CAA}^{(1)}$	Acc Time from Col Addr	—	$3 T_C - 15 - 0.075 C_A$	ns
$T_{OAC}^{(1)}$	Access Time from BEN#	—	$2 T_C - 20 - 0.050 C_B$	ns
T_{RP}	RAS# Precharge Time	$2 T_C - 5$	—	ns
T_{RAS}	RAS# Width	$4 T_C - 5 - 0.025 C_R$	—	ns
T_{RCD}	RAS# to CAS# Delay	$T_C + T_{CL} - 5 + 0.050 C_C - 0.050 C_R$	—	ns
T_{RSH}	RAS# Hold Time	$2 T_C + T_{CH} + 0.025 C_R - 0.050 C_C$	—	ns
T_{CSH}	CAS# Hold Time	$4 T_C - 10 + 0.025 C_C - 0.050 C_R$	—	ns
T_{CAS}	CAS# Width	$2 T_C + T_{CH} - 5 - 0.025 C_C$	—	ns
T_{ASR}	Row Address Setup Time	$T_C + 0.075 C_R - 0.075 C_A$	—	ns
T_{RAH}	Row Address Hold Time	$T_C - 5 + 0.075 C_A - 0.050 C_R$	—	ns
T_{ASC}	Column Addr Setup Time	$T_{CL} - 5 + 0.075 C_C - 0.075 C_A$	—	ns
T_{CAR}	Col Addr Setup to RAS#	$3 T_C + 0.025 C_R - 0.075 C_A$	—	ns
T_{OFF}	Data in Hold Time	0	—	ns
T_{BOV}	BEN0# to BEN1# Overlap	0	—	ns

SINGLE WRITE CYCLE

Symbol	Parameter	Min	Max	Units
T_{RWL}	WE# to RAS# Lead Time	$T_C - 5 + 0.025 C_R - 0.050 C_W$	—	ns
T_{WCH}	WE# Hold Time	$3 T_C + T_{CH} + 0.025 C_W - 0.050 C_C$	—	ns
T_{WP}	WE# Width	$2 T_C - 5 - 0.025 C_W$	—	ns
T_{CWL}	WE# to CAS# Lead Time	$T_C - 10 + 0.025 C_C - 0.050 C_W$	—	ns
$T_{DS(W)}$	Data Out Setup Time	$T_C + 0.075 C_W - 0.075 C_D$	—	ns
T_{DH}	Data Out Hold Time	$T_C - 5 + 0.075 C_D - 0.050 C_W$	—	ns

READ MODIFY WRITE CYCLE

Symbol	Parameter	Min	Max	Units
T_{RWC}	RMW Cycle Time	$8 T_C - 5$	—	ns
$T_{DS(TW)}$	Data Out (RMW) Setup Time	$3 T_{CH} - 5 + 0.075 C_W - 0.075 C_D$	—	ns
T_{DH}	Data Out (RMW) Hold Time	$T_C - 5 + 0.075 C_D - 0.050 C_W$	—	ns
$T_{OFF(RW)}$	Data In Hold/Data Out (RMW) Drive Time	0	$T_{CL} + 5 + 0.075 C_D - 0.075 C_B$	ns

PAGE MODE READ AND WRITE CYCLES

Symbol	Parameter	Min	Max	Units
T_{PC}	Page Mode Cycle Time	$4 T_C - 5$	—	ns
T_{CP}	CAS# Precharge Time	$T_C + T_{CL} - 5$	—	ns
T_{CAS}	CAS# Width	$2 T_C + T_{CH} - 5 - 0.025 C_C$	—	ns
$T_{CAH(n)}$	Col Addr Hold (Non Interleaved)	$3 T_C + T_{CH} + 0.075 C_A - 0.050 C_C$	—	ns
$T_{DS(n)}$	Data Out Setup (Non Interleaved)	$T_C + T_{CL} - 10 + 0.075 C_C - 0.075 C_D$	—	ns
$T_{DH(n)}$	Data Out Hold (Non Interleaved)	$2 T_C + T_{CH} + 0.075 C_D - 0.050 C_C$	—	ns
$T_{CAH(i)}$	Col Addr Hold (Interleaved)	$T_C + T_{CH} + 0.075 C_A - 0.050 C_C$	—	ns
$T_{DS(i)}$	Data Out Setup (Interleaved)	$T_{CL} - 10 + 0.075 C_C - 0.075 C_D$	—	ns
$T_{DH(i)}$	Data Out Hold (Interleaved)	$T_C + T_{CH} + 0.075 C_D - 0.050 C_C$	—	ns

FAST PAGE MODE READ AND WRITE CYCLES

Symbol	Parameter	Min	Max	Units
T_{PC}	Fast Cycle Time	$2 T_C - 5$	—	ns
T_{CP}	CAS# Precharge Time	$T_{CL} - 5$	—	ns
T_{CAS}	CAS# Width	$T_C + T_{CH} - 5 - 0.025 C_C$	—	ns
$T_{CAA}^{(1)}$	Col Address Access Time		$2 T_C - 15 - 0.075 C_A$	ns
$T_{CAC}^{(1)}$	CAS# Access Time		$T_C + T_{CH} - 15 - 0.050 C_C$	ns
$T_{CAP}^{(1)}$	Access Time from Col Precharge		$2 T_C - 15 - 0.075 C_C$	ns

FAST PAGE MODE READ AND WRITE CYCLES (Continued)

Symbol	Parameter	Min	Max	Units
$T_{OAC(i)}$	Access Time from BEN # (Interleaved)		$T_C - 20 - 0.050 C_B$	ns
$T_{CAH(n)}$	Col Addr Hold (Non Interleaved)	$T_C + T_{CH} + 0.075 C_A - 0.050 C_C$	—	ns
$T_{DS(n)}$	Data Out Setup Non Interleaved)	$T_{CL} - 10 + 0.075 C_C - 0.075 C_D$	—	ns
$T_{DH(n)}$	Data Out Hold (Non Interleaved)	$T_C + T_{CH} + 0.075 C_D - 0.050 C_C$	—	ns
$T_{CAH(i)}$	Col Addr Hold (Interleaved)	$T_{CH} + 0.075 C_A - 0.050 C_C$	—	ns
$T_{DS(i)}$	Data Out Setup (Interleaved)	$T_{CL} - 10 + 0.075 C_C - 0.075 C_D$	—	ns
$T_{DH(i)}$	Data Out Hold (Interleaved)	$T_{CH} + 0.075 C_D - 0.050 C_C$	—	ns

DUAL PORT DRAM DATA TRANSFER CYCLE

Symbol	Parameter	Min	Max	Units
T_{DTR}	DT High to RAS# High Setup	$T_C - 10 + 0.025 C_R - 0.075 C_B$	—	ns
T_{DTH}	DT High from RAS# High Hold	$T_C - 10 + 0.075 C_B - 0.075 C_R$	—	ns
T_{RDH}	DT Low from RAS# Low Hold	$3 T_C - 10 + 0.025 C_B - 0.050 C_R$	—	ns
T_{DLS}	DT Low to RAS# Low Setup	$T_C - 10 + 0.075 C_R - 0.050 C_B$	—	ns
T_{CDH}	DT Low from CAS# Low Hold	$T_C + T_{CH} - 10 + 0.025 C_B - 0.050 C_C$	—	ns
T_{DTC}	DT High to CAS# High Setup	$T_C - 10 + 0.025 C_C - 0.075 C_B$	—	ns

MASTER MODE TIMINGS $C_L = 100$ pF on all output pins

AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units
$T_{M1A(2)}$	CLK to MEN Delay	—	40	ns
$T_{M1B(3)}$	HLDA to MEN Delay	—	45	ns
$T_{M2A(2)}$	CLK to A21:0, MIO, RD#, WR#, BHE# Drive	—	40	ns
$T_{M2B(3)}$	HLDA to A21:0, MIO, RD#, WR#, BHE# Drive	—	45	ns
T_{M3}	HREQ, MEN Inactive Delay	—	35	ns
T_{M4}	A21:0, D15:0 Float Delay	—	40	ns
T_{M5}	Async HLDA Setup	5	—	ns
T_{M8}	Read Data Setup Time	10	—	ns
T_{M9}	Read Data Hold Time	5	—	ns
T_{M10}	READY Setup Time	20	—	ns
T_{M11}	READY Hold Time	5	—	ns
T_{M12}	Command Valid Delay	—	35	ns
T_{M13}	Address Valid Delay	—	40	ns
T_{M14}	Write Data Valid Delay	—	40	ns
T_{M15}	Write Data Hold Time/Float Delay	—	40	ns
T_{M16}	Sync HLDA Setup : 01	5	—	ns
T_{M17}	Sync HLDA Setup : 02	20	—	ns
T_{M18}	CLK to HREQ Delay	—	35	ns

SLAVE INTERFACE TIMINGS $C_L = 100$ pF on all output pins
 AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units
T _{S1}	Active Input Setup	5	—	ns
T _{S2}	Active Input Hold Time	10	—	ns
T _{S3}	Inactive Input Setup	5	—	ns
T _{S4}	Inactive Hold Time	10	—	ns
T _{S5}	Active Status Hold Time (Sync 186)	T _C	—	ns
T _{S14}	Active Command Width	2 T _C + 30	—	ns
T _{S15}	Inactive Command Width	2 T _C + 30	—	ns
T _{S16}	A21:0, MIO, CS #, BHE # Hold Time	2 T _C + 30	—	ns
T _{S17}	A21:0, MIO, CS #, BHE # Delay	—	T _C - 20	ns
T _{S18}	SEN Active Delay	0	25	ns
T _{S19}	Write Data Delay	0	2 T _C - 25	ns
T _{S20} ⁽⁴⁾	Write Data Hold (Memory Write)	3 T _C + T _{DH} + 30	—	ns
T _{S20}	Write Data Hold (Int. Write)	4 T _C	—	ns
T _{S21}	SEN Inactive Delay	0	35	ns
T _{S22} ⁽⁵⁾	Read Data Delay (Memory Read)	0	(Note 5)	ns
T _{S22}	Read Data Delay (Int. Read)	0	T _C + 40	ns
T _{S23}	Read Data Hold	5 T _C - 15	—	ns
T _{S24} ⁽⁶⁾	RD/WR to SEN Delay (Mem Write)	4 T _C + 20	—	ns
T _{S24} ⁽⁶⁾	RD/WR to SEN Delay (Int. Write)	4 T _C + 20	—	ns
T _{S24} ⁽⁶⁾	RD/WR to SEN Delay (Mem Read)	5 T _C + 20	—	ns
T _{S24} ⁽⁶⁾	RD/WR to SEN Delay (Int. Read)	7 T _C + 20	—	ns
T _{S25}	SEN Width (Write Cycle)	4 T _C - 25	4 T _C + 35	ns
T _{S25}	SEN Width (Read Cycle)	5 T _C - 25	5 T _C + 35	ns

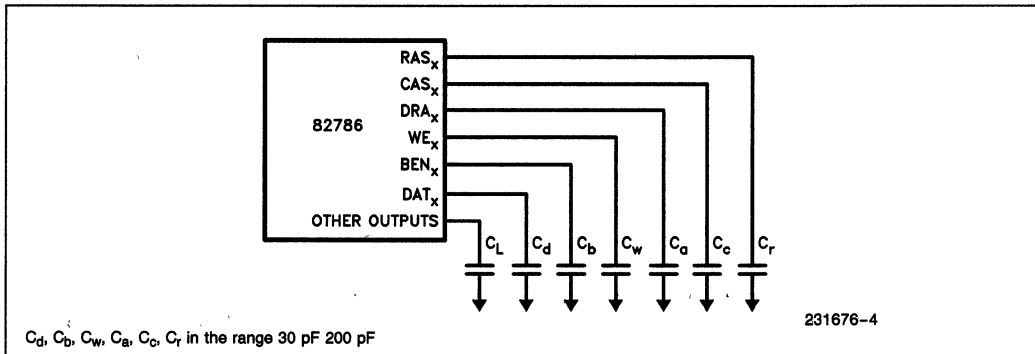
VIDEO INTERFACE $C_L = 50$ pF on all output pins
 AC timings are referenced to 1.5V on clock input and 0.8V/2.0V on other pins

Symbol	Parameter	Min	Max	Units	Notes
$T_{V_{CYC}}$	VCLK Cycle Time	40	—	ns	@ 1.5V
$T_{V_{CL}}$	VCLK Low Time	15	—	ns	@ 1.5V
$T_{V_{CH}}$	VCLK High Time	15	—	ns	@ 1.5V
$T_{V_{CR}}$	VCLK Rise Time	—	10	ns	@ 0.8V–2.0V
$T_{V_{CF}}$	VCLK Fall Time	—	10	ns	@ 0.8–2.0V
$T_{V_{DL}}$	Delay VCLK to Output Valid	0	25	ns	
$T_{V_{S}}$	Input Setup Time	0	—	ns	
$T_{V_{H}}$	Input Hold Time	8	—	ns	

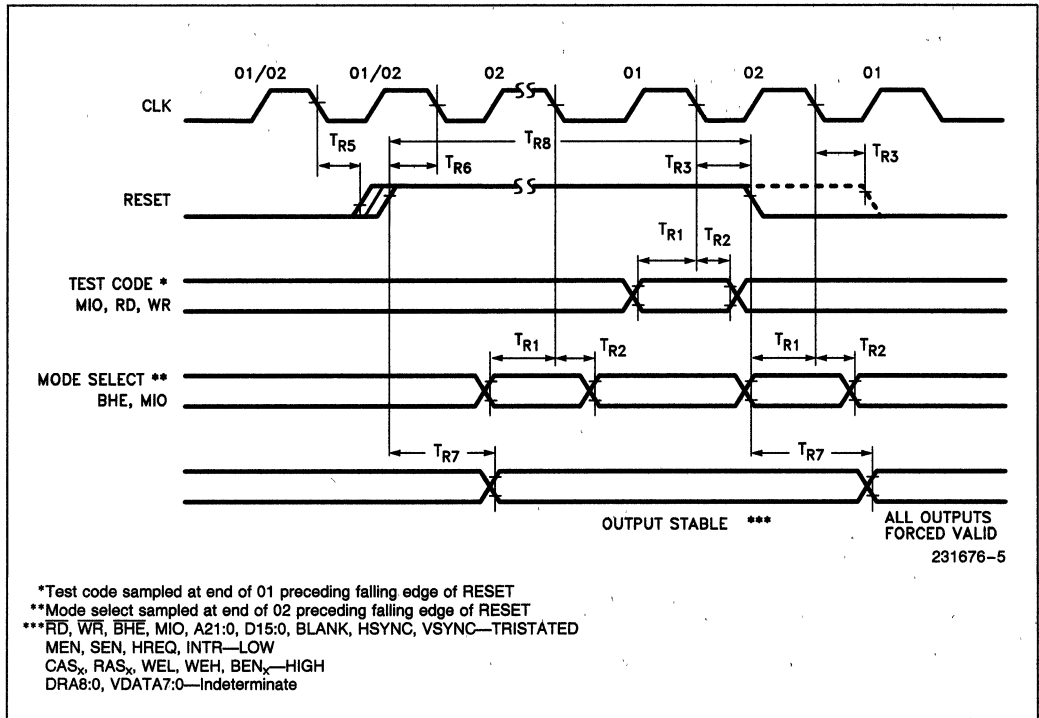
NOTES:

1. Subtract transceiver delay from these number for xl devices.
2. Valid for asynchronous interface or for synchronous interface when TM16 is satisfied. Synchronous interface requires same clock and reset input for 82786 and 80286.
3. Valid for synchronous interface when TM16 is not satisfied.
4. TS20 (memory write) is dependent on DRAM specs.
5. TS22 (memory read) is dependent on DRAM specs. This is the maximum of:
 - i) $T_{RAC} - T_C + 10$
 - ii) $T_C - T_{CH} + T_{CAC} + 10$
 - iii) $T_C - T_{OAC} + 10$
6. TS24 numbers mentioned above are the absolute minimum values for a synchronous 80286/82786 type interface (or synchronous 80186/82786 interface with $WT = 0$). Add T_C to get corresponding minimum number for an asynchronous interface. Add T_C for 80186 interface with $WT = 1$. Typical delay from Command to SEN active will be greater than the minimum value, depending on level of activity of the 82786 and priority for external access.

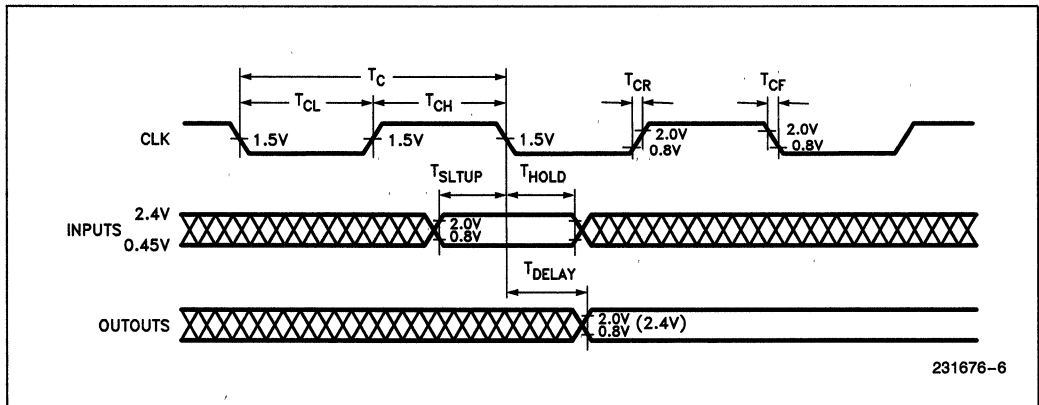
AC TEST LOADS (Use capacitance values in pico farads in the timing equations)



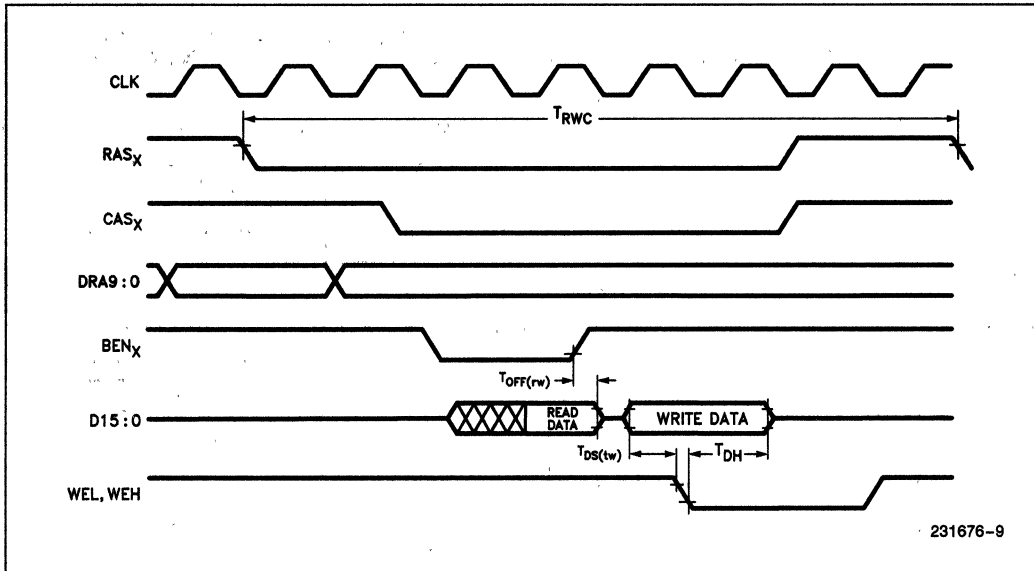
RESET TIMINGS



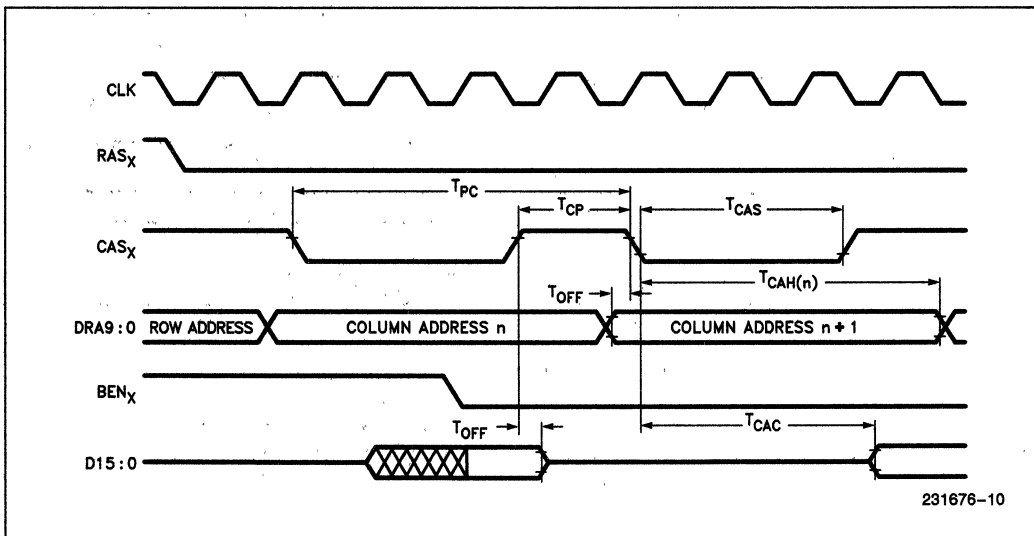
CLOCK AND AC TEST CONDITIONS



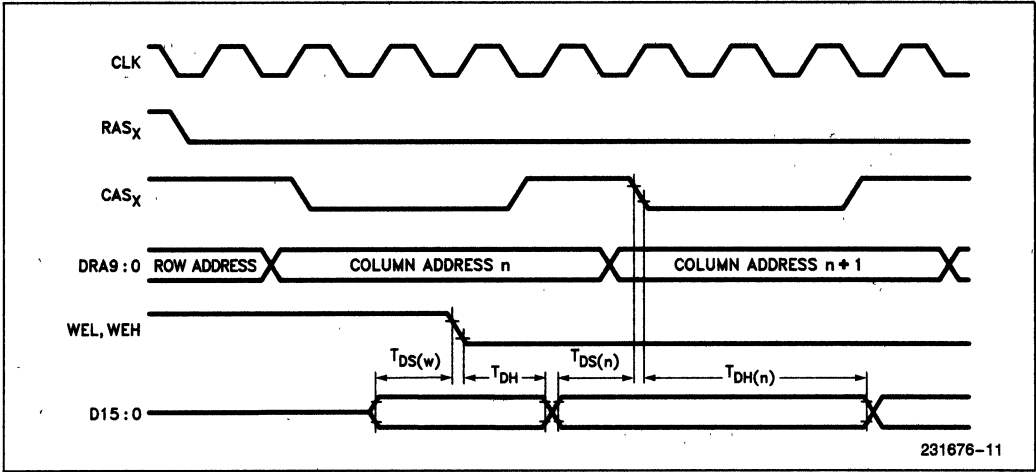
DRAM SIGNALS—READ MODIFY WRITE CYCLE



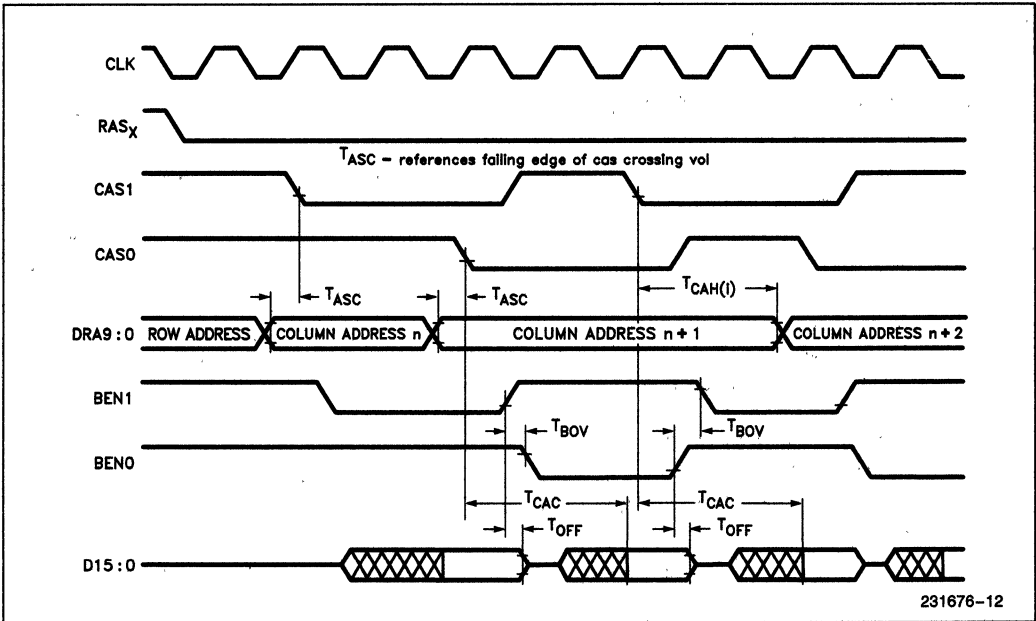
DRAM SIGNALS—NON INTERLEAVED PAGE MODE READ



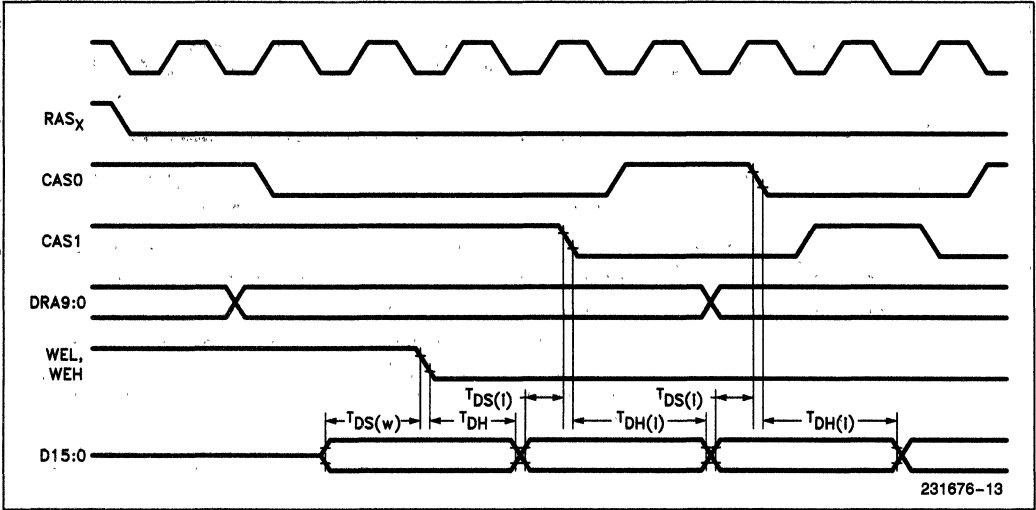
DRAM SIGNALS—NON INTERLEAVED PAGE MODE WRITE



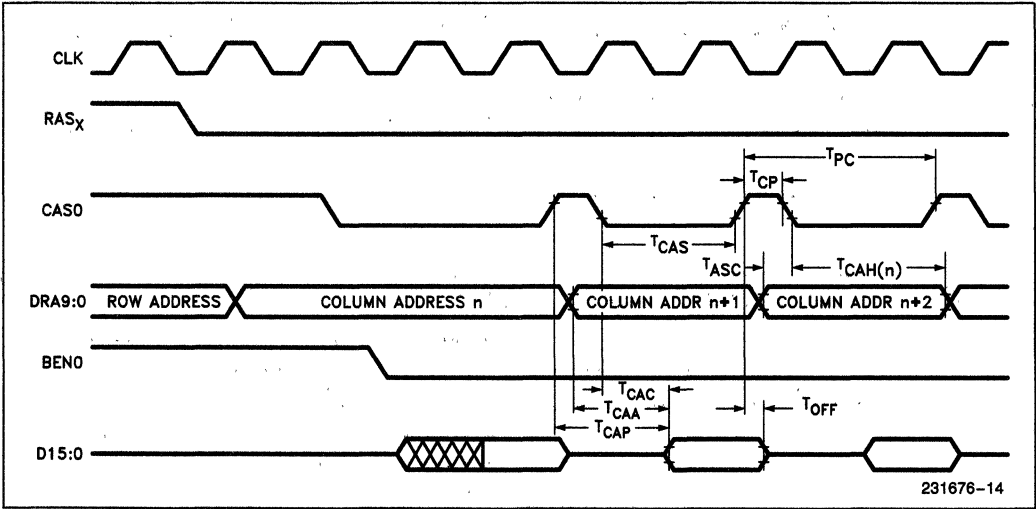
DRAM SIGNALS—INTERLEAVED PAGE MODE READ



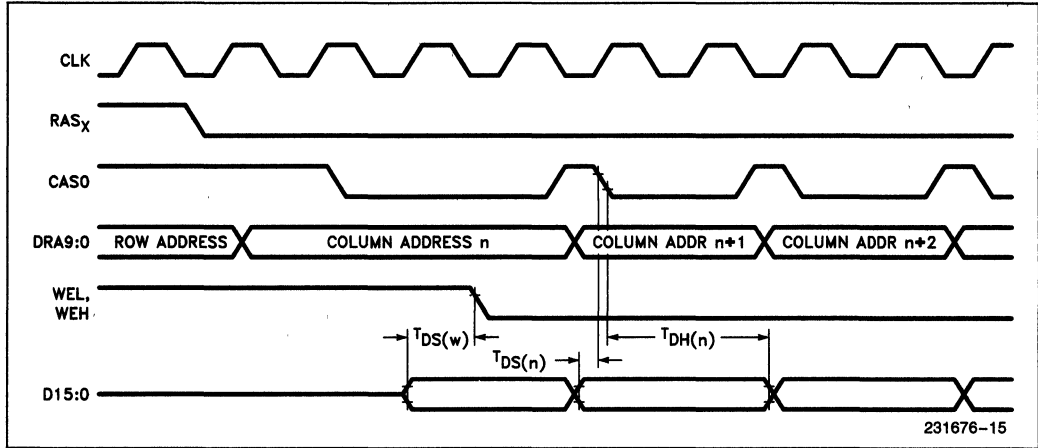
DRAM SIGNALS—INTERLEAVED PAGE MODE WRITE



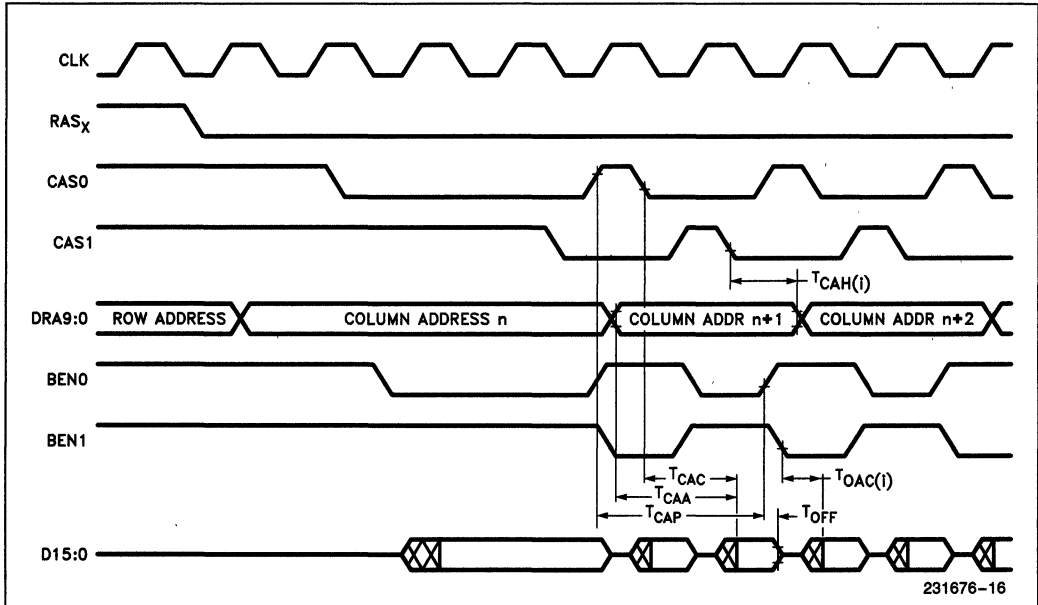
DRAM SIGNALS—NON INTERLEAVED FAST PAGE MODE READ



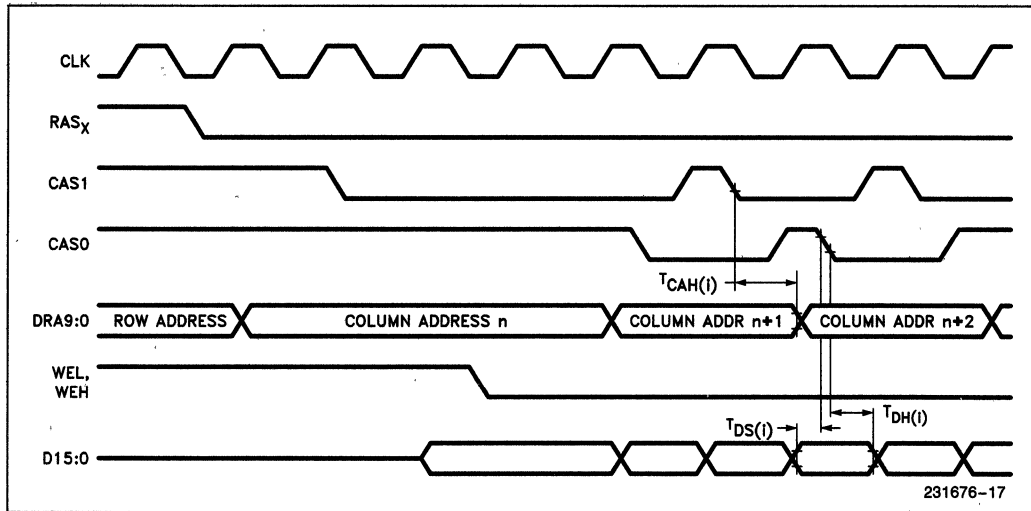
DRAM SIGNALS—NON INTERLEAVED FAST PAGE MODE WRITE



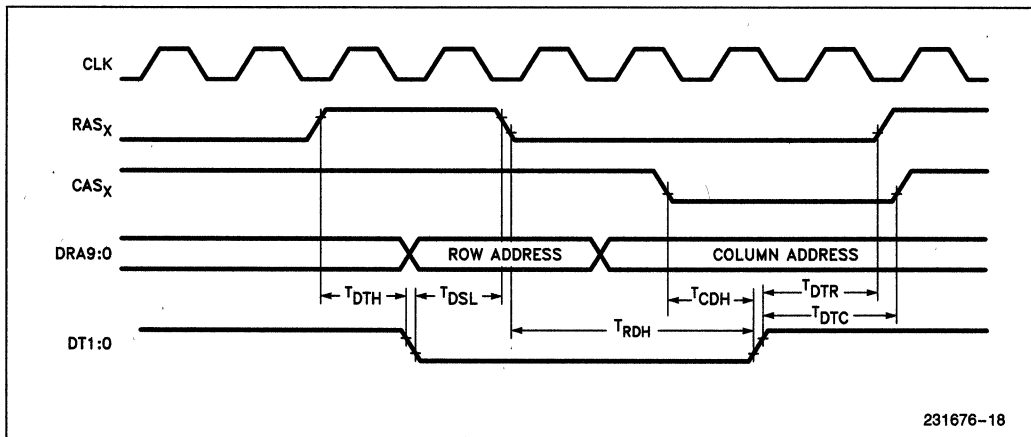
DRAM SIGNALS—INTERLEAVED FAST PAGE MODE READ



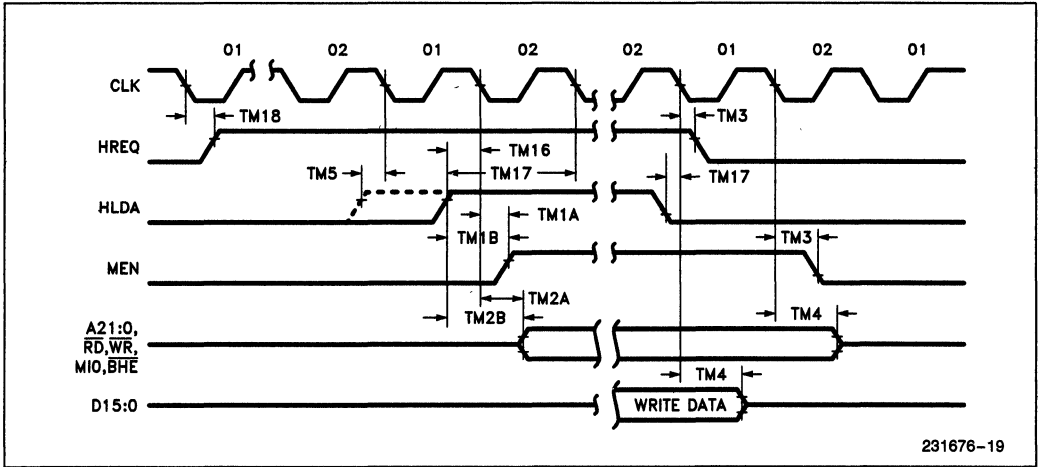
DRAM SIGNALS—INTERLEAVED FAST PAGE MODE WRITE



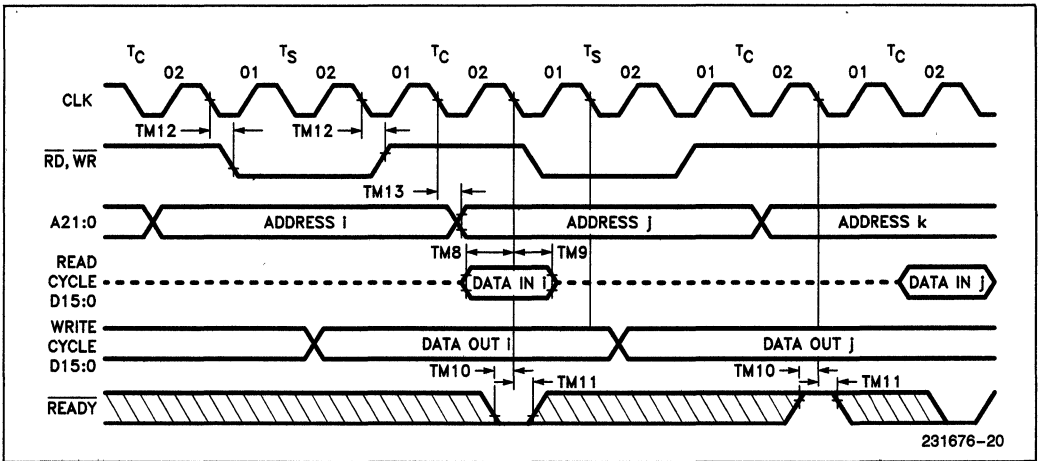
DRAM SIGNALS—DATA TRANSFER CYCLE



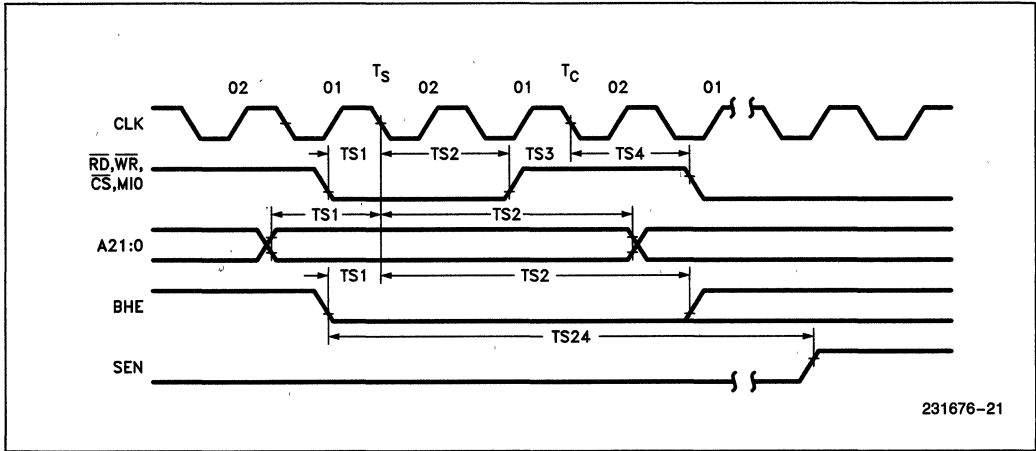
ENTERING AND LEAVING MASTER MODE



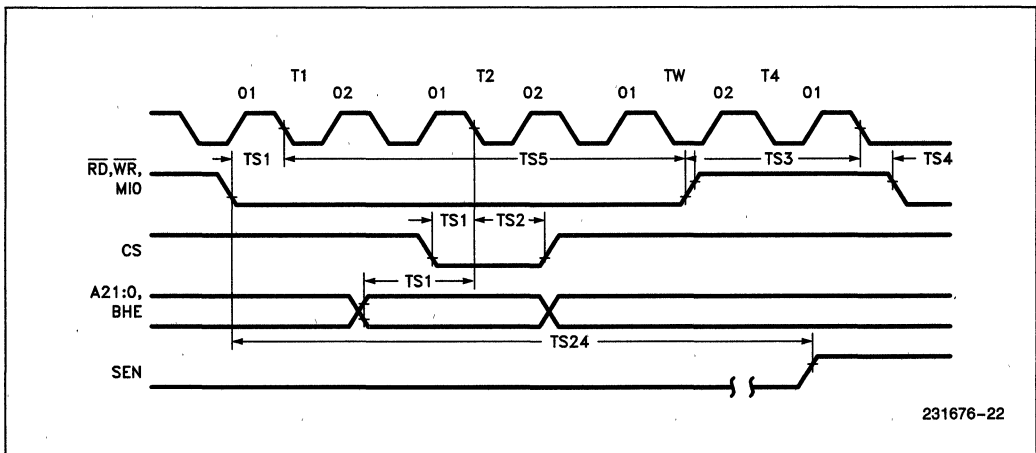
MASTER MODE TIMINGS



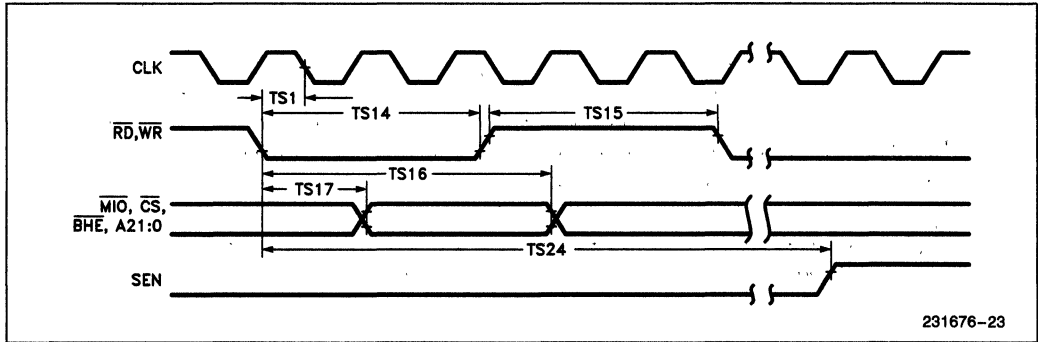
SYNCHRONOUS 80286 (STATUS) SLAVE INTERFACE



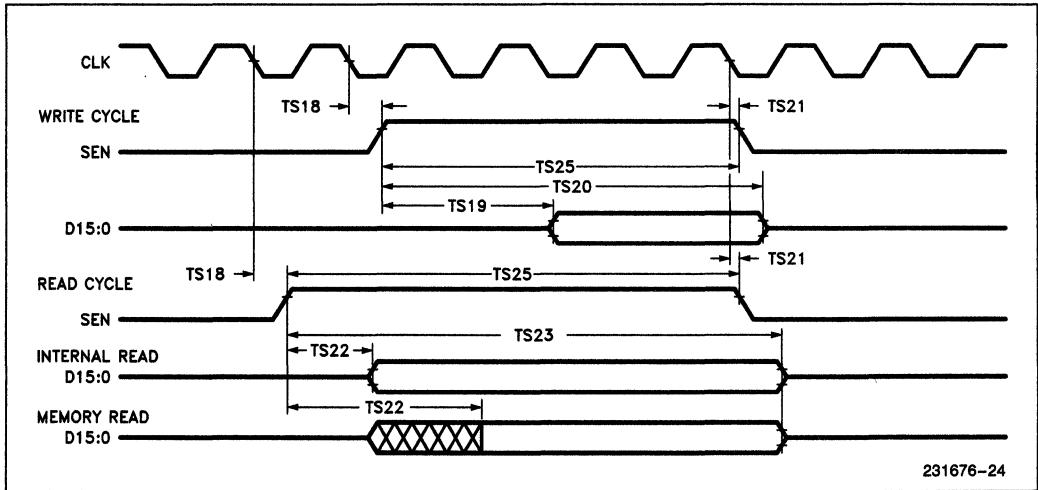
SYNCHRONOUS 80186 (STATUS) SLAVE INTERFACE



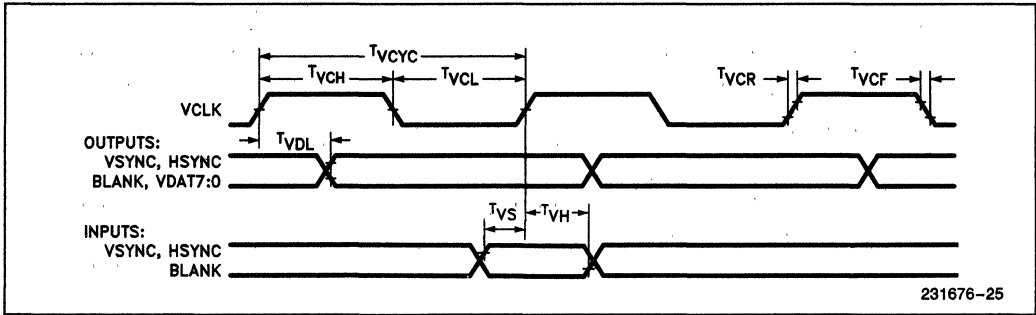
ASYNCHRONOUS SLAVE INTERFACE



SEN/DATA—SLAVE INTERFACE



VIDEO TIMINGS





**APPLICATION
NOTE**

AP-268

November 1986

**A Low Cost and High Integration
Graphics System Using 82716**

**VIREN SHAH
APPLICATIONS ENGINEER**

Order Number: 231679-001

1.0 INTRODUCTION

The role of graphics in personal computers and engineering workstations is becoming increasingly important. Lately, the graphics software which support separate windows for separate tasks have made multitasking an attractive feature on the PCs. To support this feature, the system must handle windows efficiently.

Windowing and graphics make very heavy demands on processing power. If a main processor in a PC or graphics workstation is used to move information around on the display, the response time becomes unacceptably slow. While keeping the system cost low, the VSDD solves this problem by supporting hardware windows and providing other additional features on chip.

The purpose of this application note is to familiarize the reader with the 82716 VSDD (video storage and display device) operation. This note will guide the reader in designing a text and graphics system. This document is intended as a supplement to the VSDD data sheet and user's manual.

The VSDD is aimed at applications needing a low cost and highly integrated color graphics controller for both bit-mapped and alphanumeric displays. The chip's high level of integration allows designers to build graphics systems with a very low chip count thus improving the

reliability of their equipment. The system designed in this note needs only 7 components besides VSDD. VSDD's high integration and low cost makes it ideal for compact, low cost video displays found in home computers, home information systems and industrial and commercial monitoring equipments. The chip also supports videotex standards such as NAPLPS, TELE-TEL, PRESTEL, and CAPTAIN.

82716 Description

The block diagram of the VSDD is shown in Figure 1.

The VSDD has the following features:

- Manages up to 16 bit-map and character objects.
- On chip 16/4096 color palette.
- On chip DRAM controller.
- Up to 640 x 512 pixel resolution.
- Extremely simple interface to Intel 8 bit and 16 bit CPUs.
- On chip D/A converters.
- Low chip count display controller.
- Analog or digital video outputs.
- Up to 512K bytes of display memory.
- 2, 4, or 8 bits/pixel.

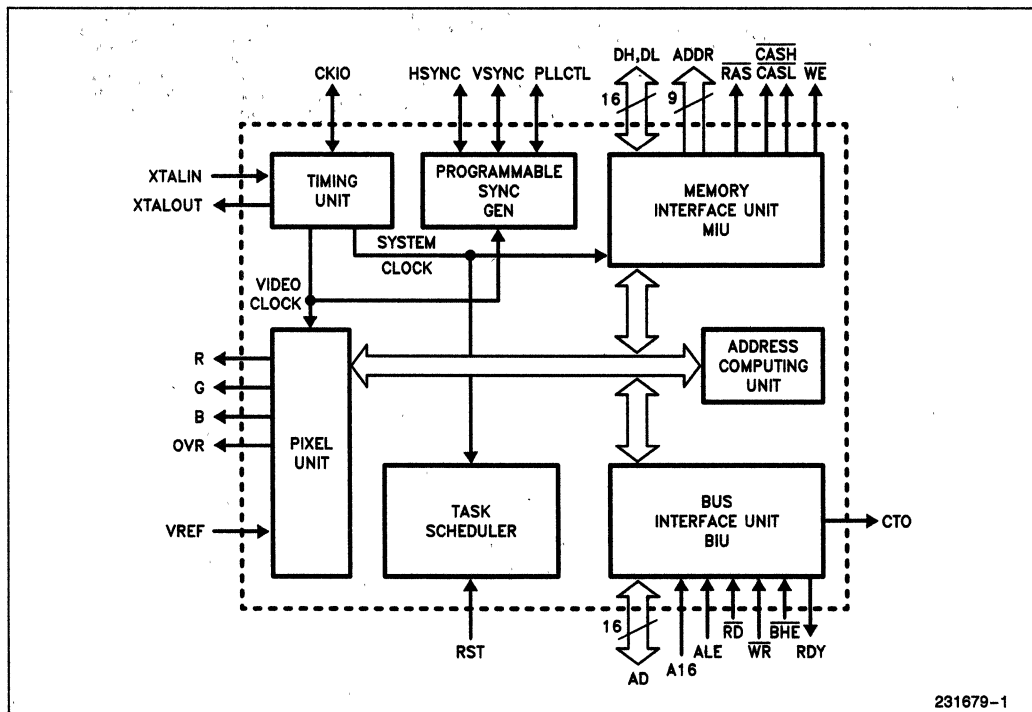


Figure 1. VSDD Block Diagram

The VSDD can control up to 16 simultaneous windows. It can change the position and contents of any window independently. This allows easy scrolling and animation. It interfaces easily to 8088 and 8086 family microprocessors without glue logic. The chip controls CRT monitors with up to 640 x 512 x 4 resolution. The on chip DRAM controller manages up to 512K bytes of display RAM. A pair of pixel buffers—each can hold 640 pixels at 4 bits/pixel—help speed up the operation by providing a continuous video data output stream.

The VSDD also integrates a color lookup table (storing 16 colors from a possible 4096), three 4 bit D/A converters and a programmable sync and timing generator. A microprocessor, its program ROM, DRAMs and a VSDD will complete a workstation—less than 10 chips in all. The VSDD also provides digital video outputs. 8 bits/pixel digital output combined with external color look up table and DACs can provide 256 colors. The VSDD supports overlapped objects. Transparent windows too are supported by the display controller.

The screen image is constructed from various user-specified objects residing in the VSDD's display memory (mapped into the processor's address space). Figure

2 shows how the CPU's address space is mapped onto the VSDD's space. The data window in the CPU space maps onto the data segment in the VSDD space and the register window maps into the register segment. The CPU uses these windows to access the display RAM. The register segment length is fixed at 32 bytes. But the data window length can vary from 4K bytes up to 64K bytes. The 512K bytes of display memory can be thought of as 8 banks of 64K bytes each. The CPU can access only one bank at a time. But all the eight banks are accessible via memory mapping, thus allowing it effectively to access all of 512K bytes.

Pixels are taken directly from the memory for display on the screen. Characters are constructed using user-defined RAM-based character generators. The VSDD takes the object data from its memory, buffers it and runs it through the color look-up table and D/A converters to produce video signals. These signals then drive the display.

There are two segments in the display memory—the data segment and the register segment. The data segment contains the actual object data, window attributes such as object's position on the screen, object's width, etc., access table, color look up table and two character

Memory Mapping

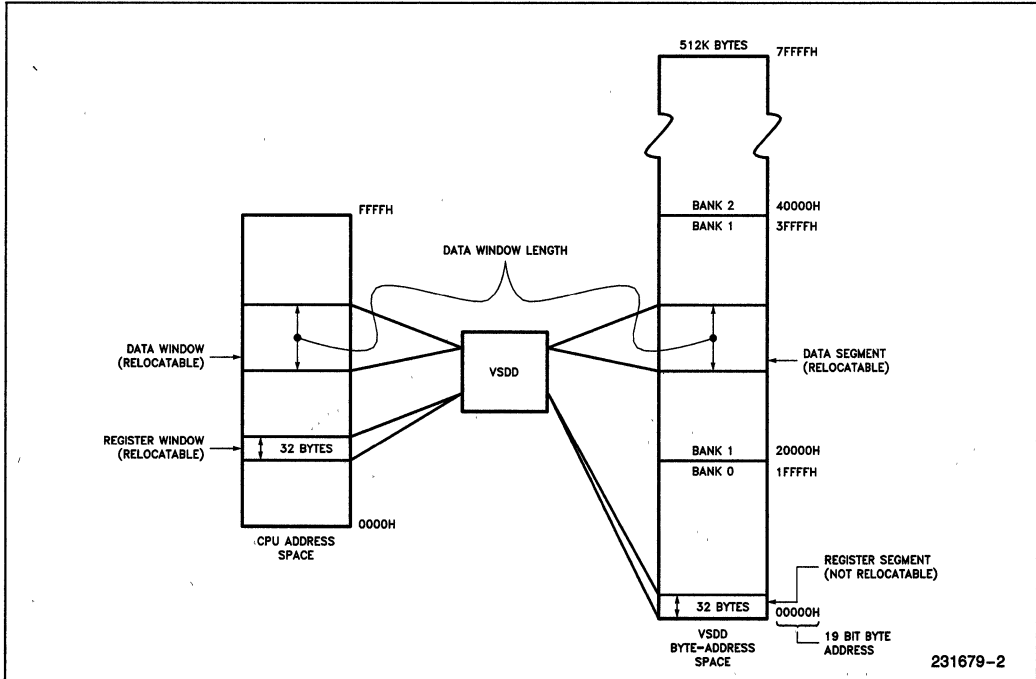


Figure 2

generators. The access table contains the vertical position and priority of each object. The data segment can be placed anywhere within the external 512K byte display RAM.

Information on the system's configuration is kept in a 32-byte register segment, which always begins at hexadecimal address 00000 in the display DRAM. The VSDD reads the register segment once per frame to update its on-chip registers. The register segment stores the size and speed of DRAM, the raster parameters and the base addresses of the other tables stored in the data segment. Figure 3 shows the register segment.

			DRAM Loc.
R15	Horiz. Constant 3	Vert. Constant 3	1EH
R14	Horiz. Constant 2	Vert. Constant 2	1CH
R13	Horiz. Constant 1	Vert. Constant 1	1AH
R12	Horiz. Constant 0	Vert. Constant 0	18H
R11	Access Table Base Address Counter		16H
R10	Character Base Address		14H
R9	Color Table Base Address		12H
R8	Access Table Base Address		10H
R7	Object Descriptor Table Base Address		0EH
R6	Priority Access Quantity		0CH
R5	Data Segment Base Address		0AH
R4	Data Length Mask		08H
R3	Data Window Base Address		06H
R2	Register Window Base Address		04H
R1	Video Configuration Register 1		02H
R0	Video Configuration Register 0		00H

Figure 3

The CPU programs the data segment and the register segment. After these segments are initialized the VSDD assumes control of the CRT and controls refresh of the DRAM. This relieves the CPU of maintaining the display thus considerably increasing the performance of the graphics system.

The chief purpose of this application note is to show the user how to program data and register segments by describing a specific design example.

2.0 DESIGN EXAMPLE

In this design, the VSDD is used to display 3 bit-mapped objects and one character object on the screen. Hardware is very simple and compact. Only seven

chips besides the VSDD are needed to build the system. 80186 (8 MHz) is used as the CPU. 4 DRAMs (64K x 4) give a total display memory of 128K bytes. This much memory can support a resolution of 640 x 400 at 4 bits/pixel. No glue logic is needed between the 80186 and the VSDD for the bus interface. The display used is an IBM color monitor. The digital video outputs are used to drive the monitor through the line drivers (74LS244). The VSDD generates active low VSYNC and HSYNC. Since the IBM color monitor needs active high VSYNC and HSYNC, the VSDD generated sync signals are inverted (74LS368). By using PAL for the random logic in this design the total chip count for this graphics system can be reduced to seven including the VSDD. Appendix E shows the circuit schematics.

2.1 Initialization

The VSDD and the register segment must be initialized before a display can be obtained. The RESET pin on the 82716 is active high. It is a high impedance input to a Schmitt Trigger. On power up, RESET should be held active long enough to allow the VSDD system oscillator (on XTALIN) to start, and then be held for a minimum of 20 clock periods with the oscillator running.

RESET STATUS

RESET puts the 82716 into a special initialization mode. When RESET goes low, the device commences generating refresh cycles to the external DRAM. The CPU should not try to access the DRAM for at least 10 μ s after RESET has been released. The status of the device at this time is as follows:

Control Bit	Reset Value	Effect
TMM	0	Single Mode
TMS	0	
DEN	0	No Display
DEI	0	Analog Mode
MAS	0	Sync pins are in high impedance
RE	0	CPU can access DRAM only to write to it
PCE	0	Priority Access Mode disabled
FAE	0	RDY pin emits Ready signal
EVC	0	Video clock is from XTALIN signal

DEN = 0 means no display is being generated. DEI = 0 means the device is in the analog mode, and, since DEN = 0, the R, G, B, and 0 pins are emitting retrace black.

MAS = 0 configures the sync pins in high impedance state. EVC = 0 would normally configure the CKIO pin as an output, but in the initialization mode CKIO is in a high impedance state.

The device comes out of reset with the following default values for the screen constants:

HCO = 1	HSYNC Width = 32 XTALIN periods	4.0 μ s
HC1 = 2	AHZ Start Time = 48 XTALIN periods	6.0 μ s
HC2 = 5	AHZ Stop Time = 96 XTALIN periods	12.0 μ s
HC3 = 8	HSYNC Period = 144 XTALIN periods	18.0 μ s
VC0 = 1	VSYNC Width = 2 lines	36.0 μ s
VC1 = 2	AVZ Start Time = 3 lines	54.0 μ s
VC2 = 3	AVZ Stop Time = 4 lines	72.0 μ s
VC3 = 7	VSYNC Period = 8 lines	144 μ s

This is called the "fast frame" mode. Its main purpose is to give the PSG something to work with that is not contradictory (such as HCO > HC3) until the CPU has written the correct values into DRAM. The timings listed assume a 125 nsec clock (8.0 MHz) at XTALIN.

Most important to the initialization procedure are the reset values of the Register and Data Windows:

Register Window Base Address: 00400H
Data Window Base Address: (undefined)

INITIALIZATION PROCEDURE

The first access must be a write cycle to Register R0 at 00400H. In the first write cycle the CPU should program the DRAM configuration bits DS1, DS0 and DOF for 64K x 4 DRAMS employed in this design.

This first write cycle should leave DEN and UCF at 0. UCF (Update Control Flag) should be left clear till the

entire Register Segment has been initialized, lest the device "update" its on-chip control bits with random data.

After the first write, the CPU can continue to initialize the Register Segment by writing to the Register Window addresses, 00400H through 0041FH. All this information is mapped by the VSDD into its register segment from 00000H to 0001FH. Except for the control bits written into R0, the values as written do not take effect as long as UCF = 0.

After the Register Segment has been completely initialized, one more write cycle is directed to R0 to set UCF to 1, while holding DEN = 0. The CPU now waits 1 frame time (144 μ s, in the "fast frame" mode). At the end of the frame time in progress, during the FRAM-ESTOP sequence, the VSDD will be flagged by UCF = 1 to update its internal registers from the Register Segment in DRAM. Further access to the Register Segment by the CPU must be through the newly defined Register Window.

Now, through the newly defined Data Window, the CPU can commence initializing the display data. The DEN flag in R0 should be set(1) after the display data is loaded in the DRAM. This bit will then enable the display.

At the beginning the CPU programs R0 at 00400H as follows:

```

                DEN   UCF
R0 011 00000 011 1 0 0 1 0 6072H
DEN and UCF are set to zeroes.
```

After the Register Segment is completely initialized, CPU sets the UCF bit by writing to R0.

```
R0 011 00000 011 1 0 0 1 1 6073H
```

The VSDD would now update its on-chip control registers with the data programmed by CPU in the Register Segment.

The CPU then programs the display data through the data window. After the data has been written into memory, the CPU enables the display by setting DEN bit.

```
R0 011 00000 011 1 1 0 1 1 607BH
```

The register segment is programmed as follows to obtain a display shown in Figure 4. The VSDD reads the register segment on every frame to update its on-chip registers. The reader should refer to the user's manual for description of the bits in the register segment. See Appendix A for horizontal and vertical sync constants. The VSDD DRAM is word addressable while the CPU address space is byte addressable.

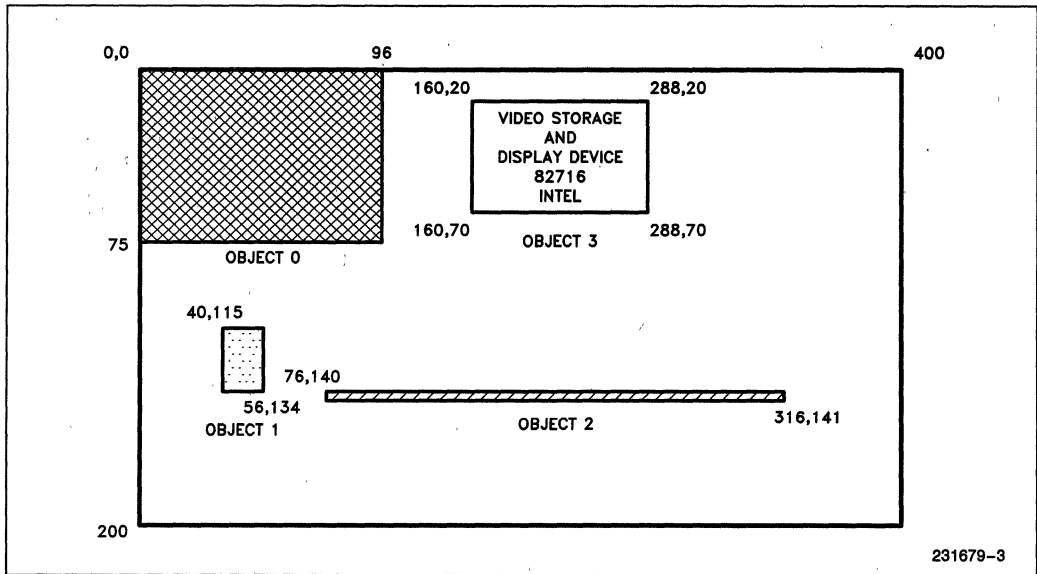


Figure 4. Display Screen

REGISTER SEGMENT

CPU ADDR	DRAM WORD ADDR	REGISTER	CONTENTS	COMMENTS
0000H	00000H	R0	011 0000 01111011	607BH
			Duty Cycle 011	50%
			Blink Rate 00000	7.5 Hz for 60 Hz frame rate
			DS1 DS0 DOF 011	64K x 4 DRAMs
			HRS 1	640 Pixels horizontally
			DEN 1	Display Enabled
			SAB 0	Fast DRAM
			DEI 1	Digital Outputs on RGB & OVR pins
			UCF 1	Update all the registers on every frame
0002H	00001H	R1	1010 010 00 00101 0 0	A414H
			Char height 1010	10 Scan lines
			INL 0	Non interlaced
			MAS 1	HSYNC, VSYNC are outputs
			SM 0	Non composite SYNC mode
			TMM, TMS 00	Twin mode is disabled
			EVC 0	CKIO is O/P. Video Clk derived from XTALIN.
			PCE 1	Priority counter enable
			FAE 0	Use RDY as ready signal
			RE 1	CPU can read the DRAM
			PSA 0	GCLK = 1/16 XTALIN
			PRE 0	Disable pipeline read
0004H	00002H	R2	0000 0000 0000 11 0	0006H
			RWBA 000H	RWBA = 0
			TF2 TF1 11	Digital pixel codes
			ME 0	No margin

REGISTER SEGMENT (Continued)

CPU ADDR	DRAM WORD ADDR	REGISTER	CONTENTS	COMMENTS
0006H	00003H	R3	0000 00 0101000 000 DWBA 00000 Screen 0101000 111 Boundary	0140H A16 Should be low = Rt edge of the screen is at X= 327
0008H	00004H	R4	1 0000 000000000000 Length Mask 10000	8000H 64K byte data window
000AH	00005H	R5	0000 0000 0000 0000 Data Segment Base S16-S12 00000 Bank Select Bits 0 0	0000H Bank 0
000CH	00006H	R6	0000 0000 0000 1010 PAQ 1010	000AH CPU is allowed 10 DRAM accesses during line building
000EH	00007H	R7	0000 0101 0000 0000 ODTBA 0500H	0500 H Object descriptor table starts at 0500H in bank 0
0010H	00008H	R8	0000 0000 0010 0000 ATBA 0010H	0010 H Access table starts at 0010H in bank 0
0012H	00009H	R9	0000 0001 1000 0000 CTBA 0180H	0180H The color table is not used in this design. The space is reserved for future use.
0014H	0000AH	R10	0000 0000 0010 0011 CGBA0 0010 CGBA2 0011	0023H Char gen 0 starts at 2000H in bank 0 Char gen 1 starts at 3000H in bank 0
0016H	0000BH	R11	0000 0000 0010 0000	Access table address counter = access table base address (initially) = 0010H
0018H	0000CH	R12	000001 0000000010B HC0 VC0	HSYNC Width = 4 μ s VSYNC Width = 198 μ s
001AH	0000DH	R13	000100 0000100100B HC1 VC1	AHZ Start = 10 μ s AVZ Start = 2.5 ms
001CH	0000EH	R14	011101 0011101100 HC2 VC2	AHZ Stop = 60 μ s AVZ Stop = 16.17 ms
001EH	0000FH	R15	100000 001110100	Hori. sweep rate = 66 μ s Vert. sweep rate = 16.67 ms

NOTE:

See Appendix I on how to program registers R12-R15.

3.0 THE DATA SEGMENT

The actual object data and the different tables are stored in the data segment by the 80186. There are 5 tables in the data segment: The access table, the object descriptor table, the color lookup table and two character generators. Since digital outputs are used to drive the monitor in this design, the color lookup table is left blank.

3.1 Access Table

The access table contains the vertical start and end locations of each object. The table begins at the locations designated by access table base address register, R8, in the register bank. Each entry in the table contains 16 bits—each bit representing one object. Bit number 0 has the lowest priority and bit number 15 the highest.

The first entry in the table corresponds to the topmost line on the screen and so on. Each entry indicates to the VSDD which objects are to be present on this line of the display. If a bit is set (1), then there is no change in the objects display status; that is, if the object did not appear on the previous line, it will also not appear on this line. If the object's access flag is set to zero, then the display status is reversed from what it was on the previous line. The VSDD assumes that at the beginning of a frame all objects are turned off (1's).

The access table for the screen in Figure 4 is shown in the following pages. The screen has 400 x 200 resolution. There are 200 entries in the table for 200 vertical lines on the screen. Object 0 starts on line 1 and ends at line 75. Bit 0 is set to zero at line 1 to turn the object 0 on and again set to 0 at line 76 to turn the object off. Note that the 80186's address space is byte addressable and the VSDD's space is word addressable.

b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0

Figure 5. Access Table Word

ACCESS TABLE

CPU ADDR	DRAM WORD ADDR	ACCESS FLAGS																
		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1		b0
0020H	00010H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	Line 1
0022H	00011H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0024H	00012H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0026H	00013H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0028H	00014H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
002AH	00015H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
002CH	00016H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
002EH	00017H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0030H	00018H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0032H	00019H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 10
0034H	0001AH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0036H	0001BH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0038H	0001CH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
003AH	0001DH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
003CH	0001EH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
003EH	0001FH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0040H	00020H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0042H	00021H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0044H	00022H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0046H	00023H	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	Line 20
0048H	00024H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Turn on the
004AH	00025H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	character
004CH	00026H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	object
004EH	00027H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0050H	00028H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0052H	00029H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0054H	0002AH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0056H	0002BH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0058H	0002CH	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

ACCESS TABLE (Continued)

CPU ADDR	DRAM WORD ADDR	ACCESS FLAGS																	
		b 15	b 14	b 13	b 12	b 11	b 10	b 9	b 8	b 7	b 6	b 5	b 4	b 3	b 2	b 1		b 0	
00100H	00080H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 112
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	Line 115 (Turn on Obj. 1)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 120
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Rectangle		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 120
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 125
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Object #1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 125
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 130
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Horizontal Line Object #2		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 134
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 135 (Turn off Obj. 1)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 140 (Turn on Obj. 2)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	Line 142 (Turn off Obj. 2)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Horizontal Line Object #2		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 140 (Turn on Obj. 2)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	Line 142 (Turn off Obj. 2)
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Line 151
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Up to 200 Lines

3.2 Object Descriptor Table:

This table contains a 4-word object descriptor field for each object in the display. The table starts at the location specified by the OBTBA register, R7. This field specifies the base address of the object in the RAM, horizontal position, its width and other attributes. The descriptor fields for bit-map and character objects are shown in Figure 6.

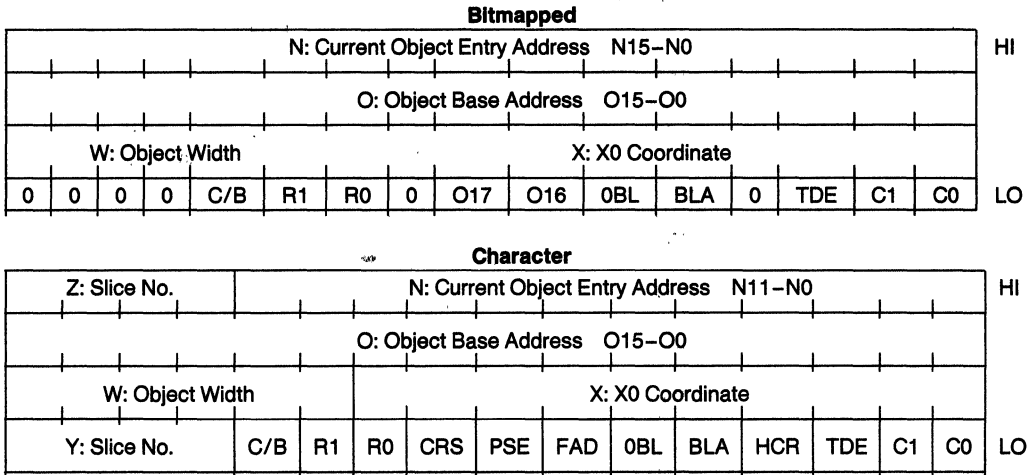


Figure 6

Objects are rectangular windows on the screen. The object data begins in the display RAM at the object base address specified in the object descriptor field. The length of the data file depends on the objects height, width and resolution. The width is specified in 4-word units by the "W" field. In this design a 4 bits/pixel specification is chosen. Hence, each 4-word unit represents 16 pixels. Objects 0 in Figure 4 is 6 x four word wide, that is 96 pixels wide. The object descriptor field and object data for each of the objects in Figure 4 are as follows:

OBJECT 0 DESCRIPTOR FIELD

Fill 75 Lines on the Screen with One Color

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A00H	00500H	0000011000000000 C/B = 0 R1R0 = 11 017, 016 = 00 OBL = 0 BLA = 0 TDE = 0 C1C0 = 00	0600H Bit mapped object 4 bits/pixel Object is in bank 0 No blinking Object is not turned off Non transparent pixels Don't care for 4 bits/pixel
0A02H	00501H	000110 0000000000 WIDTH: 000110 X 0000000000	6 four word wide = 96 pixels wide Object starts at the left edge of the screen
0A04H	00502H	0001 0000 0000 0000 Object base address	01000H
0A06H	00503H	0001 0000 0000 0000 Current object entry	01000H

OBJECT DATA
OBJECT 0

OBJECT BASE ADDR = 01000H

CPU ADDR	DRAM WORD ADDR	PIXEL DATA	
2000H	01000	8888H	Line 1
2002H	01001	8888H	
2004H	01002	8888H	
2006H	01003	8888H	
2008H	01004	8888H	
...	01005	8888H	24 Words = 96 Pixels wide @ 4 bits/pixel
	01006	8888H	
	01007	8888H	
	01008	8888H	
	01009	8888H	
	0100A	8888H	
	0100B	8888H	
	0100C	8888H	
	0100D	8888H	
	0100E	8888H	
	0100F	8888H	
	01010	8888H	
	01011	8888H	
	01012	8888H	
	01013	8888H	
	01014	8888H	
	01015	8888H	
	01016	8888H	
202EH	01017	8888H	End of Line 1
	
	

OBJECT 0 DATA CPU ADDR	DRAM WORD ADDR	PIXEL DATA	
2060H	01030H	8888H	Line 3
	
	
	016F0H	8888H	Line 75
	016F1H	8888H	
	016F2H	8888H	
	016F3H	8888H	
	016F4H	8888H	
	016F5H	8888H	
	016F6H	8888H	
	016F7H	8888H	
	016F8H	8888H	
	016F9H	8888H	
	016FA	8888H	
	016FB	8888H	
	016FC	8888H	
	016FD	8888H	
	016FE	8888H	
	016FG	8888H	
	01700	8888H	
	01701	8888H	
	01702	8888H	
	01703	8888H	
	01704	8888H	
	01705	8888H	
	01706	8888H	
2EOE	01707	8888H	End of line 75

**OBJECT 1 DESCRIPTOR FIELD
RECTANGLE**

20 SCAN LINES/
16 PIXELS WIDE

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A08H	00504H	0000 0 11 00 00 00 00 C/B = 0 R1R0 = 11 017, 016 = 00 OBL = 0 BLA = 0 TDE = 0 C1C0 = 00	0600H Bit Mapped Object 4 Bits/Pixel Object in bank 0 No Blinking Object is not turned off Non-transparent pixels Don't care
0A0AH	00505H	000001 0000010100 WIDTH = 000001	1 Four word wide = 16 Pixels wide Objects starts at X = 20
0A0CH	00506H	*X = 0000010100 0001 0111 0000 1010	Object base addr 0170AH
0A0EH	00507H	0001 0111 0000 1010	Current obj. entry 0170AH

***NOTE:**

When HRS = 1, unit displacement in X direction moves the object by 2 pixels. Thus the object 1 will start at pixel number 40.

OBJECT 1 DATA

CPU ADDR	DRAM WORD ADDR	PIXEL DATA	
2E14	01070H	7777H	Line 1
	...	7777H	
	...	7777H	
	...	7777H	
	0170EH	7777H	Line 2
	0170FH	7777H	Line 2
	01710H	7777H	Line 2
	01711H	7777H	Line 2
	
	
	
	
	
	
	
	01756H	7777H	Line 20
	01757H	7777H	...
	01758H	7777H	...
2EB2	01759H	7777H	...

OBJECT 2 DESCRIPTOR FIELD

HORIZONTAL 2 SCAN LINES/
240 PIXELS WIDE

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A10H	00508H	0000 011 00 00 00 00	Same as obj 0 and obj 1
0A12H	00509H	001111 00001 00 11 0 WIDTH = 001111	
		X = 38	15 four word wide = 250 pixels wide
0A14H	0050AH	0001 0111 0110 0000	Obj base addr. 01760H Current obj entry = Obj base address (Initially)
0A16H	0050BH	0001 0111 0110 0000	

OBJECT 2 DATA
CPU ADDR
 2EC0H

DRAM WORD ADDR

PIXEL DATA

Line 1

01760H 5555H
 ... 5555H
 ... 5555H
 ... 5555H
 ... 5555H
 ... 5555H
 ...
 ...
 ... 5555H
 ...
 ...

60 Words

0179C

Line 2

... 5555H
 ... 5555H
 ...
 ...
 ...

2FAEH

017D7

5555H

OBJECT 3 DESCRIPTOR FIELD

TEXT 50 SCAN LINES/
 16 CHARACTERS WIDE

CPU ADDR	DRAM WORD ADDR	CONTENTS	COMMENTS
0A18	0050CH	1010 1 10 0 0 0 0 0 0 100 Y: Slice No. = 1010 C/B = 1 R1R0 = 10 CRS = 0 PSE = 0 FAD = 0 OBL = 0 BLA = 0 HCR = 0 TDE = 1 CICO = 00	Start Slice Number Character Object 8 Pixels/Character Character Generator 0 Monospace Characters 1 Byte/Character No blinking Object is not turned off Don't care Transparent pixels Default color bits
0A1A	0050DH	000010 0001010000 WIDTH = 000010 X = 0001010000	2 four word wide = 16 characters wide Object starts at X = 80
0A1C	0050EH	0001 0111 1101 1010	Object Base Address 017DAH
0A1FH	0050FH	0001 0111 1101 1010	Current Object Entry 017DAH

OBJECT 3 DATA

CPU ADDR	DRAM ADDR	ASCII CODE	
2FB4	017DA	5620	Line 1
...	017DB	4449	
...	017DC	4F45	
...	017DD	5320	
...	017DE	4F54	
...	017DF	4152	
...	017EO	4547	
...	017E1	2020	
2FC4	017E2	2020	Line 2
...	017E3	2020	
...	...	2020	
...	...	4E41	
...	...	2044	
...	...	2020	
...	...	2020	
...	...	2020	
2FD4	017EA	4420	Line 3
...	...	5349	
...	...	4C50	
...	...	5941	
...	...	4420	
...	...	5645	
...	...	4359	
...	...	2045	
2FE4	017F2	2020	Line 4
...	...	2020	
...	...	2020	
...	...	2020	
...	...	3238	
...	...	3137	
...	...	2036	
...	...	2020	
2FF4	017FA	2020	Line 5
...	...	2020	
...	...	2020	
...	...	4E49	
...	...	4554	
...	...	204C	
...	...	2020	
...	...	2020	

CHARACTER GENERATOR 0

The pixel data for characters are stored in one of 2 character generators in the display RAM. Character generator 0 is used in this design. The base address of character generator 0 as obtained from R10, is 02000H in bank 0. Character height as defined in R1 is 10 scan lines. The character set 0 consists of 10 blocks of 256 words each. Block No. 1 contains the 1st slice of each of 256 characters, block 2 has the 2nd slice of all the 256 characters and so on. In this example, 26 alphabets and 10 numerals are defined. The VSDD addresses

character generator as shown in Figure 7. Individual slices are addressed by concatenating the four bits of the character generator base address with the slice number Z and the ASCII code itself. Slice number 0 is the bottom scan line for the character and slice number H-1 is the top line. Each slice is encoded as a sequence of pixel bits. If a pixel bit is 1, then the pixel is given the foreground color. If the bit is 0, the pixel is displayed in background color. This is shown in Figure 8.

Figure 9 shows how the pixels are encoded for character "F" (ASCII code = 46H).

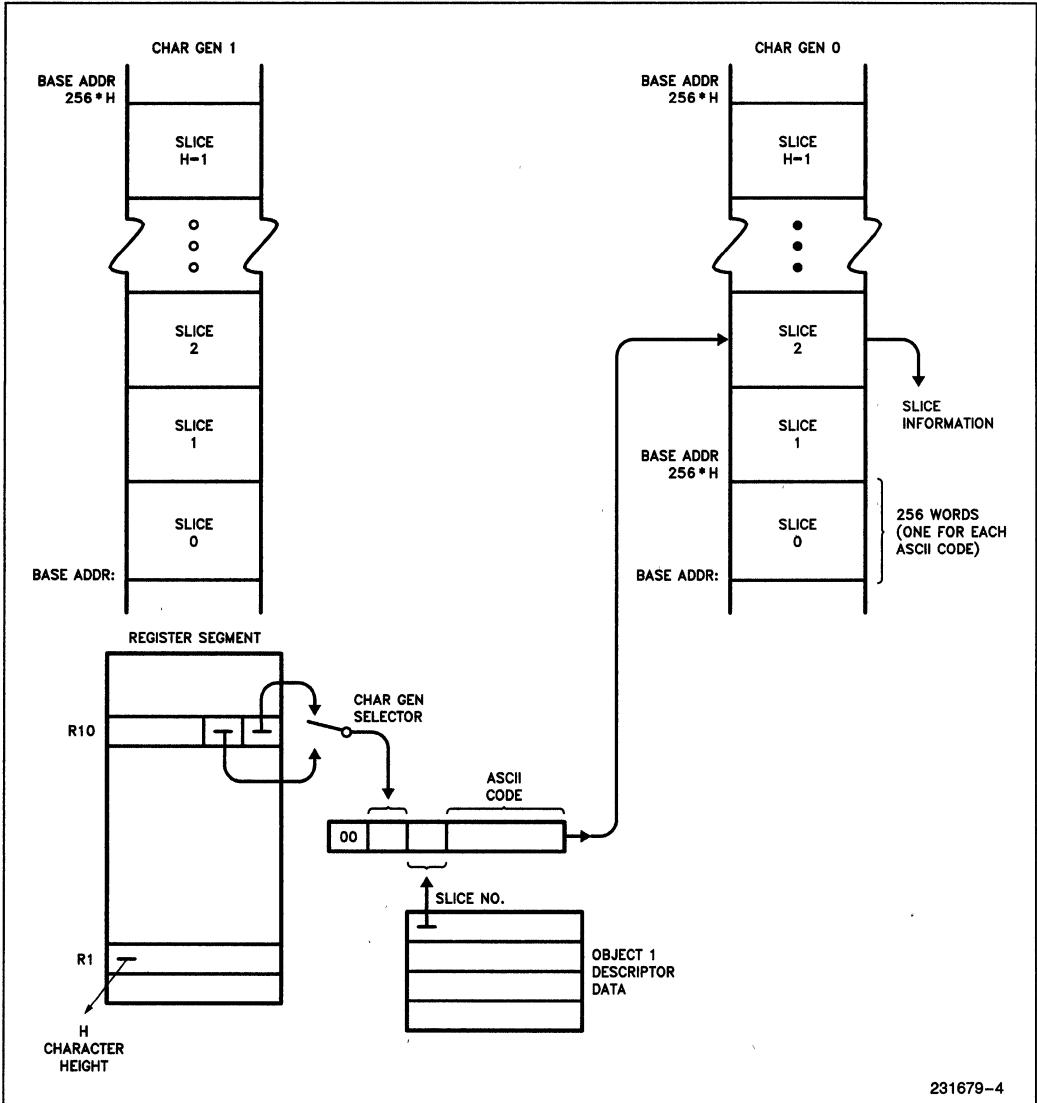


Figure 7. Addressing Character Slices

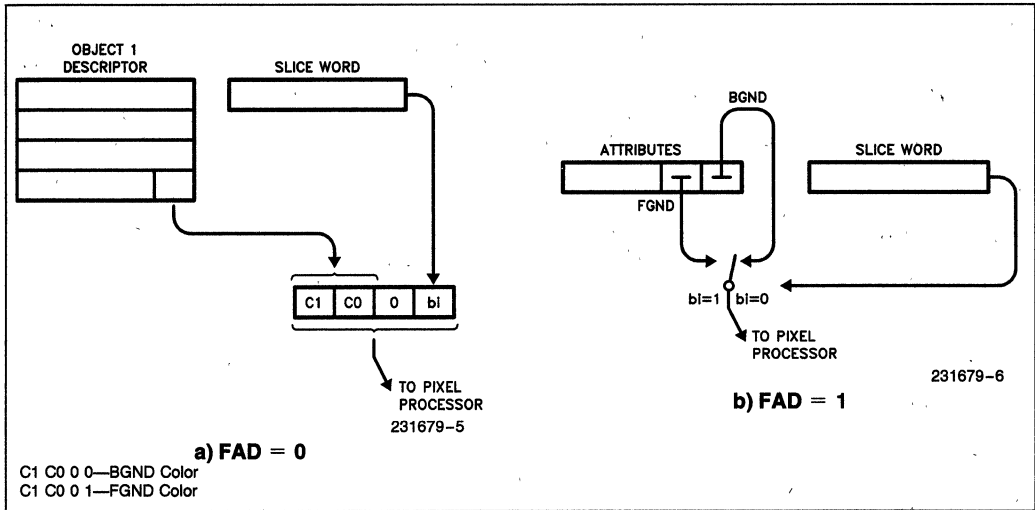


Figure 8. Character Generator Pixel Construction

	DRAM ADDRESS		PIXEL BITS								CPU ADDRESS	HEX DUMP
			P7	P6	P5	P4	P3	P2	P1	P0		
SLICE 9	02946H	00000000	0	0	0	0	0	0	0	0	0528CH	0000H
	02846H	00000000	0	1	1	1	1	1	1	0	0508CH	007EH
	02746H	00000000	0	0	0	0	0	0	1	1	04E8CH	0006H
	02646H	00000000	0	0	0	0	0	0	1	1	04C8CH	0006H
	02546H	00000000	0	0	0	1	1	1	1	0	04A8CH	001EH
	02446H	00000000	0	0	0	0	0	1	1	0	0488CH	0006H
	02346H	00000000	0	0	0	0	0	1	1	0	0468CH	0006H
	02246H	00000000	0	0	0	0	0	1	1	0	0448CH	0006H
	02146H	00000000	0	0	0	0	0	0	0	0	0428CH	0000H
SLICE 0	02046H	00000000	0	0	0	0	0	0	0	0	0408CH	0000H

Character Width = 8 Pixels
 Character Height = 10 Scan Lines
 Character Generator Base Address = 02000H

NOTE:

The leftmost pixel corresponds to the LSB in the slice word.

Figure 9. Bit Storage for Character "F"

The 80186 used in this design is resident on a single board computer known as SDV-186 board. This board is available from Red River Technology, Inc. in Addison, Texas. The VSDD board is connected to the 186 board via expansion connectors. The memory maps for

the SDV-186 board and the VSDD are shown on the following pages. The CPU address space from 60000H-6FFFFH is used for 64K data window. This data window maps onto the data segment in the VSDD's memory space.

CPU ADDR	VSDD MEMORY MAP	VSDD DRAM ADDR
60000	R0	00000
...		...
...	CONTROL REGISTERS	...
...		...
6001E	R15	0000FH
60020	SCAN LINE 0 FLAGS	...
...		...
...	ACCESS TABLE	...
...		...
601B0H	SCAN LINE 200 FLAGS	...
6022A	BLANK	...
...		...
602FE		...
60300	COLOR 0000	00180H
...	COLOR LOOKUP TABLE	...
...		...
6031E	COLOR 1111	...
60320	BLANK	...
...		...
603FE		00500H
60A00	OBJECT DESCRIPTOR TABLE	...
...		...
60A18	BLANK	...
...		...
61FFE		01000H
62000	OBJECT DATA	...
...		...
63FFE		02000H
64000	CHAR GEN 0	...
...		...
65FFE		03000H
66000	CHAR GEN 1	...
...		...
67FFE		...
68000	BLANK	...
...		...
...		...

Memory Map of the SDV-186 Board

00000H-003FFH	INTERRUPT VECTORS	
00400H-0076FH	PROGRAM DATA	
00770H-007FFH	STACK	LCS/
00800H-00FFFFH	MEMORY	
01000H-1FFFFH	VACANT	
20000H-3FFFFH	VACANT	
40000H-4FFFFH	64-K MEMORY	MCS0/
50000H-5FFFFH	64-K MEMORY	MCS1/
60000H-6FFFFH	64-K MEMORY (VACANT)	MCS2/
70000H-7FFFFH	64-K MEMORY (VACANT)	MCS3/
80000H-FBFFFH	VACANT	
FC000H-FFFFFFH	MONITOR CODE	UCS/

APPENDIX A PROGRAMMING HORIZONTAL AND VERTICAL CONSTANTS FOR IBM MONITOR

The constants will be programmed for a resolution of 400 x 200 at 60 Hz, non-interlaced mode.

60 Hz gives a frame period of 16.67 ms.

For IBM Color Monitor, vertical blanking = 3 ms.

Active Vertical zone = 16.67 - 3 = 13.67 ms.

There are 200 lines in one frame.

$$\text{Line Time} = \frac{13.67}{200} = 68.35 \mu\text{s}$$

Horizontal Blanking = 16 μs

$$\begin{aligned} \text{Active Horizontal zone} &= 68.35 - 16 \\ &= 52.35 \mu\text{s} \end{aligned}$$

$$\begin{aligned} \text{Horizontal Sync Frequency} &= \frac{1}{\text{Line Time}} \\ &= \frac{1}{68.35} \\ &= 14.6 \text{ kHz} \end{aligned}$$

$$\begin{aligned} \text{Pixel CLOCK PERIOD} &= \frac{\text{Active horizontal time}}{\text{no. of pixels/scan line}} \\ &= \frac{52.35 \mu\text{s}}{400} = 130.1 \text{ ns} \end{aligned}$$

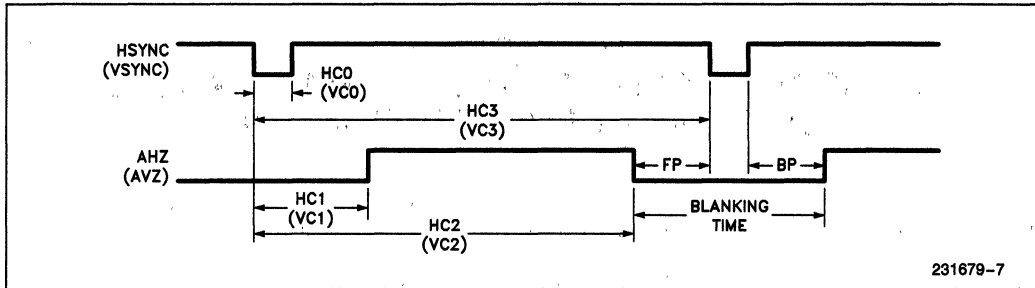
$$\text{PIXEL CLOCK} = \frac{1}{130.1} = 7.7 \text{ MHz}$$

We will use a pixel CLK of 8 MHz. As EVC = 0, System CLK is also 8 MHz.

Horizontal Blanking for the monitor = 16 μs .

This blanking time includes the front porch, sync width and the back porch.

RASTER TIMINGS



231679-7

HORIZONTAL CONSTANTS

For IBM color monitor, Hsync width = 4 μ s

Horizontal blanking = 16 μ s

Assume:

FP = 6 μ s

BP = 6 μ s

HCO = 4 μ s

HC1 = 4 + 6 = 10 μ s

HC2 = Active Horizontal Time + HC1
= 52.3 + 10 = 62.3 μ s

HC3 = Line Time = 68.3 μ s

For 8 MHz Video Clock, the period is 125 ns.

NOW, PSA = 0

GCLK PERIOD = 16 \times 125 = 2000 ns
= 2 μ s

HCO - HC3 are programmed in terms of GCLK

HCO = 2 GCLK PERIODS = 4 μ s

HCO = 000001B

NOTE:

HC0-HC3 and VC0-VC3 values are offset by 1, i.e. if HC0 is 2, then time programmed is 3 GCLK periods. HC2 and HC3 had to be tweaked to obtain a steady display on the screen. The following values give a flicker-free display.

HC1 = 10 μ s = 5 GCLK Periods

HC1 = 000100B

HC2 = 60 μ s = 30 GCLK Periods

HC2 = 011101B

HC3 = 66 μ s = 33 GCLK Periods

HC3 = 100000B

VERTICAL CONSTANTS

Vertical Blanking = 3 ms

Line Time = 33 GCLK Periods
= 66 μ s

For IBM Monitor

VC0 = VSYNC Width = 198 μ s
= 3 Line times

VC0 = 0000000010B

Assume, FP = 0.5 ms

BP = 2.3 ms

VC1 = VC0 + BP

= 0.198 + 2.3 = 2.5 ms = 37 Line times

VC1 = 0000100100B

VC2 = Active vertical time + VC1

= 13.67 + 2.5 = 16.17 ms = 237 Lines time

VC2 = 0011101100

VC3 = Vertical sweep rate = 16.67 ms = 245 Line times

VC3 = 0011110100

APPENDIX B CHARACTER GENERATOR 0

This character set is located in the VSDD's DRAM in bank O. It starts at 02000H. 26 alphabets, 10 numerals and a blank space are defined here. The ASCII code for the characters are as follows:

CHARACTER	ASCII	NUMERALS	ASCII CODE
A	41H	0	30H
B	42H	1	31H
C	43H	2	32H
D	44H	3	33H
E	45H	4	34H
F	46H	5	35H
G	47H	6	36H
H	48H	7	37H
I	49H	8	38H
J	4AH	9	39H
K	4BH		
L	4CH		
M	4DH		
N	4EH		
O	4FH		
P	50H		
Q	51H		
R	52H		
S	53H		
T	54H		
U	55H		
V	56H		
W	57H		
X	58H		
Y	59H		
Z	5AH		
BLANK SPACE	20H		

The character set has 10 blocks of 256 words each. All the locations except those corresponding to above 37 characters are blank in the display RAM. They are programmed with 0's. The hex dump for the characters is as follows:

SLICE 0		SLICE 1		SLICE 2		CHARACTER
DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	
2020H	00000H	2120H	00000H	2220H	00000H	BLANK SPACE
2030	0000	2130	0000	2230	003C	0
2031	0000	2131	0000	2231	007E	1
2032	0000	2132	0000	2232	007E	2
2033	0000	2133	0000	2233	003C	3
2034	0000	2134	0000	2234	0060	4
2035	0000	2135	0000	2235	003C	5
2036	0000	2136	0000	2236	003C	6
2037	0000	2137	0000	2237	0018	7
2038	0000	2138	0000	2238	003C	8
2039	0000	2139	0000	2239	003C	9
2041	0000	2141	0000	2241	0066	A
2042	0000	2142	0000	2242	003E	B
2043	0000	2143	0000	2243	0038	C
2044	0000	2144	0000	2244	001E	D
2045	0000	2145	0000	2245	007E	E
2046	0000	2146	0000	2246	0006	F
2047	0000	2147	0000	2247	0038	G
2048	0000	2148	0000	2248	0066	H
2049	0000	2149	0000	2249	003C	I
204A	0000	214A	0000	224A	003C	J
204B	0000	214B	0000	224B	0066	K
204C	0000	214C	0000	224C	007E	L
204D	0000	214D	0000	224D	0066	M
204E	0000	214E	0000	224E	0046	N
204F	0000	214F	0000	224F	003C	O
2050	0000	2150	0000	2250	000C	P
2051	0000	2151	0000	2251	007C	Q
2052	0000	2152	0000	2252	0066	R
2053	0000	2153	0000	2253	003C	S
2054	0000	2154	0000	2254	0018	T
2055	0000	2155	0000	2255	003C	U
2056	0000	2156	0000	2256	0018	V
2057	0000	2157	0000	2257	003C	W
2058	0000	2158	0000	2258	0066	X
2059	0000	2159	0000	2259	0018	Y
205A	0000	215A	0000	225A	007E	Z

SLICE 3		SLICE 4		SLICE 5		CHARACTER
DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	
2320H	0000H	2420H	0000H	2520H	0000H	SPACE
2330	0066	2430	0066	2530	006E	0
2331	0018	2431	0018	2531	0018	1
2332	0006	2432	000C	2532	0030	2
2333	0066	2433	0060	2533	0038	3
2334	0060	2434	007E	2534	0066	4
2335	0066	2435	0060	2535	0060	5
2336	0066	2436	0066	2536	003E	6
2337	0018	2437	0018	2537	0018	7
2338	0066	2438	0066	2538	003C	8
2339	0066	2439	0060	2539	007C	9
2341	0066	2441	007E	2541	0066	A
2342	0066	2442	0066	2542	003E	B
2343	006C	2443	0006	2543	0006	C
2344	0036	2444	0066	2544	0066	D
2345	0006	2445	0006	2545	001E	E
2346	0006	2446	0006	2546	001E	F
2347	006C	2447	0046	2547	0066	G
2348	0066	2448	0066	2548	007E	H
2349	0018	2449	0018	2549	0018	I
234A	0066	244A	0060	254A	0060	J
234B	0036	244B	000E	254B	0006	K
234C	007E	244C	0006	254C	0006	L
234D	0066	244D	0066	254D	0066	M
234E	0066	244E	0076	254E	007E	N
234F	0066	244F	0066	254F	0066	O
2350	0006	2450	0006	2550	003E	P
2351	0026	2451	0036	2551	0066	Q
2352	0036	2452	001E	2552	003E	R
2353	0066	2453	0060	2553	003C	S
2354	0018	2454	0018	2554	0018	T
2355	0066	2455	0066	2555	0066	U
2356	0018	2456	003E	2556	0024	V
2357	007E	2457	005A	2557	005A	W
2358	0066	2458	003C	2558	0018	X
2359	0018	2459	0018	2559	003C	Y
235A	0006	245A	000C	255A	0018	Z

SLICE 6		SLICE 7		SLICE 8		CHARACTER
DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	DRAM LOCATION	HEX DUMP	
2620H	0000H	2720H	0000H	2820H	0000H	SPACE
2630	0076	2730	0066	2830	003C	0
2631	001C	2731	0018	2831	0018	1
2632	0060	2732	0066	2832	003C	2
2633	0060	2733	0066	2833	003C	3
2634	0078	2734	0070	2834	0060	4
2635	003E	2735	0002	2835	007E	5
2636	0006	2736	0066	2836	003C	6
2637	0030	2737	0066	2837	007E	7
2638	0066	2738	0066	2838	003C	8
2639	0066	2739	0066	2839	003C	9
2641	003C	2741	003C	2841	0018	A
2642	0066	2742	0066	2842	003E	B
2643	0006	2743	006C	2843	0038	C
2644	0066	2744	0036	2844	001E	D
2645	0006	2745	0006	2845	007E	E
2646	0006	2746	0006	2846	007E	F
2647	0006	2747	006C	2847	0038	G
2648	0066	2748	0066	2848	0066	H
2649	0018	2749	0018	2849	003C	I
264A	0060	274A	0060	284A	0070	J
264B	000E	274B	0036	284B	0066	K
264C	0006	274C	0006	284C	0006	L
264D	007E	274D	0066	284D	0024	M
264E	006E	274E	0066	284E	0062	N
264F	0066	274F	0066	284F	003C	O
2650	0066	2750	0066	2850	003E	P
2651	0066	2751	0066	2851	003C	Q
2652	0066	2752	0066	2852	003E	R
2653	0006	2753	0066	2853	003C	S
2654	0018	2754	0018	2854	007E	T
2655	0066	2755	0066	2855	0066	U
2656	0066	2756	0042	2856	0042	V
2657	0042	2757	0042	2857	0042	W
2658	003C	2758	0066	2858	0066	X
2659	003C	2759	0066	2859	0066	Y
265A	0030	275A	0060	285A	007E	Z

SLICE 9		
DRAM LOCATION	HEX DUMP	CHARACTER
2920H	0000H	SPACE
2930	0000	0
2931	0000	1
2932	0000	2
2933	0000	3
2934	0000	4
2935	0000	5
2936	0000	6
2937	0000	7
2938	0000	8
2939	0000	9
2941	0000	A
2942	0000	B
2943	0000	C
2944	0000	D
2945	0000	E
2946	0000	F
2947	0000	G
2948	0000	H
2949	0000	I
294A	0000	J
294B	0000	K
294C	0000	L
294D	0000	M
294E	0000	N
294F	0000	O
2950	0000	P
2951	0000	Q
2952	0000	R
2953	0000	S
2954	0000	T
2955	0000	U
2956	0000	V
2957	0000	W
2958	0000	X
2959	0000	Y
295A	0000	Z

APPENDIX C ANALOG OPERATION

In the example described in the application note, digital video outputs were used.

For analog operation, DEI (digitally encoded color information) bit in register R0 is set to 0. The on-chip color look table (CLUT) is loaded with 16 entries from the external DRAM. These 16 words of data are stored in the memory starting at color look up table base address. (Register R9). Each entry is 16 bits long—with lowest 4 bits specifying the address of the entry in the CLUT and upper 12 bits specifying the color as shown in Figure 10. Four bit pixel code is used to address the CLUT. The pixel code is matched with the lowest four bits of the CLUT RAM and the pixel is given the color specified by the upper 12 bits. The address entries need not be sequential from 0–15 but they can be random. The color corresponding to address 0010 in the CLUT is reserved for the background color.

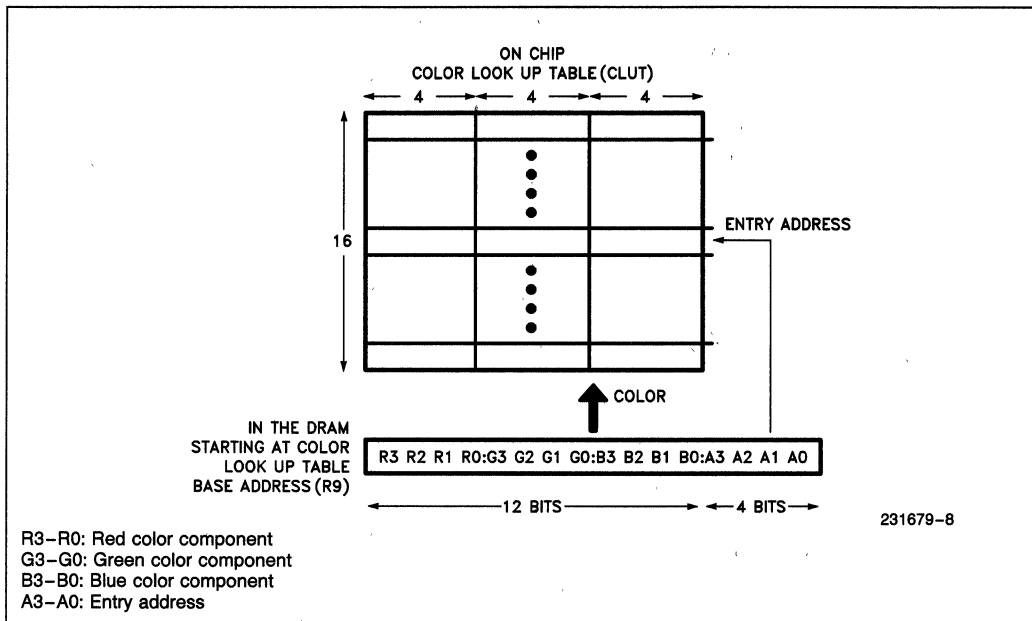


Figure 10. Filling the CLUT

The color look up table outputs—4 bits/color—drive 3 internal DACs (digital-to-analog converter). RGB signals are generated by the DACs. An externally supplied reference voltage (VREF) is used to drive the DACs. The value of VREF should be between 0 and 2V. As DACs have high output resistance, external analog buffers are used to interface them to low input resistance monitors as shown in Figure 11.

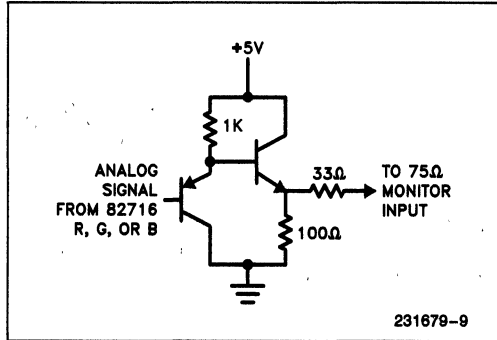


Figure 11. Buffering 82716 Analog Output to Low Input Resistance Monitor

APPENDIX D CLOCKING SCHEMES

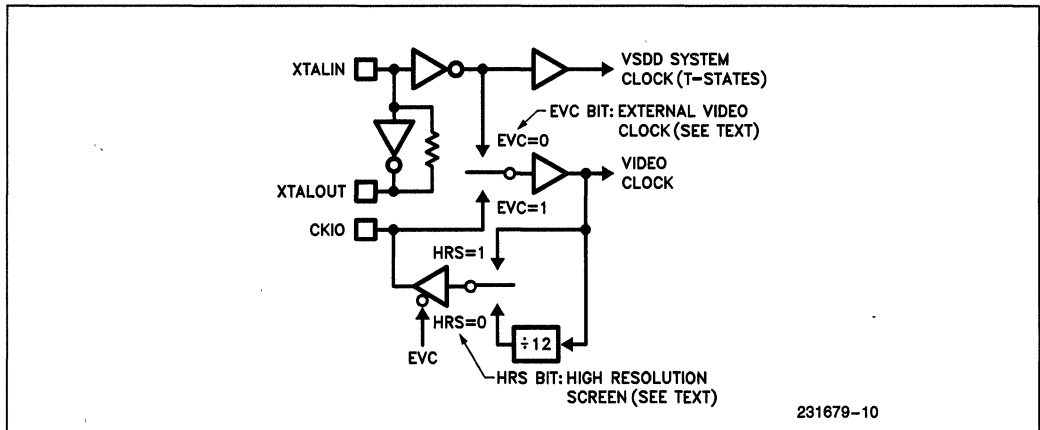
The VSDD uses two clock signals: system clock and video clock. The video clock can be derived from the system clock or may be external.

The system clock is generated by an internal oscillator using an external crystal at XTALIN and XTALOUT pins. The crystal frequency can be between 5 MHz and 15 MHz.

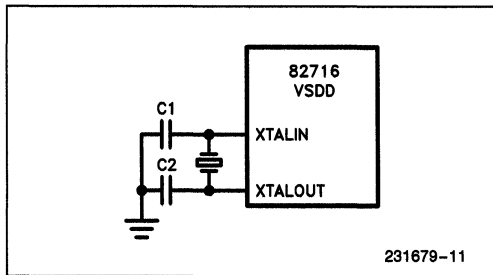
An external clock may also be fed to the XTALIN pin (instead of using a crystal). For example, CLKOUT pin of 80186 can be used to drive the VSDD system clock.

If an independent video (dot) clock is desired (EVC = 1) CKIO is used to input this clock. Maximum video clock frequency can be 25 MHz. EVC should be set to zero if video clock is derived from system clock.

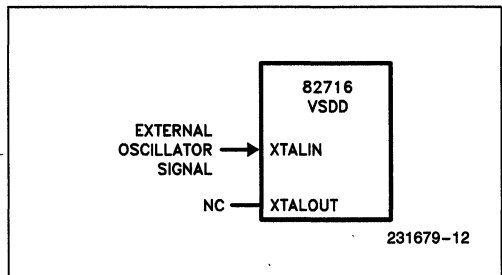
Figure 12 explains various clocking schemes.



(a) VSDD Timing Unit



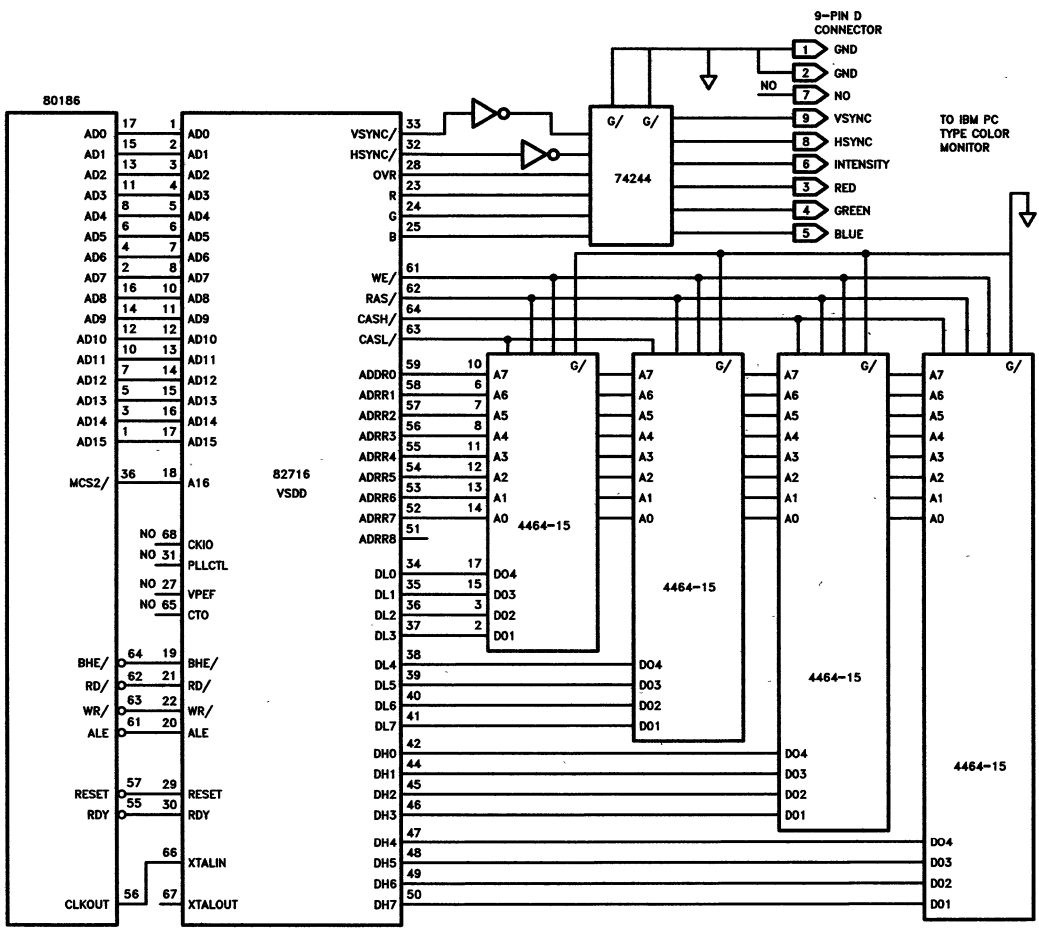
(b) With Crystal



(c) With External Clock

Figure 12. Clocking Schemes

APPENDIX E



231679-13

APPENDIX F

SERIES III 8086/87/88/186 MACRO ASSEMBLER V2.0 ASSEMBLY OF MODULE VSDD_PICTURE
 OBJECT MODULE PLACED IN : F3:APPSDF.OBJ
 ASSEMBLER INVOKED BY: ASMB6 86 : F3:APPSDF.ASM

```

LOC  OBJ          LINE   SOURCE
                                1 +1 $mod186
                                2 +1 $iref
                                3
                                4   name   vsdd_picture
                                5
                                6   ; A routine to display a simple picture on the display
                                7   ; 3 bit-map and 1 character objects are displayed.
                                8   ; this routine also shows how to move windows.
                                9
0001  10   UCF    equ      1h           ; update control flag
0006  11   DEN    equ      8h           ; display enable bit
                                12   ; initial value of R0 after reset
6072  13   r0_res equ      6072h        ; msb 0:11 duty cycle 50 percent
                                14   ; 00000 blink rate 7.5Hz
                                15   ; 011 64x4 DRAMS
                                16   ; 1 HRS 640 pixels/line
                                17   ; 0 DEN display is disabled
                                18   ; 0 SAB fast DRAMS
                                19   ; 1 DEI Digital video outputs
                                20   ; 0 UCF No register update
                                21
6073  22   r0_up  equ      (r0_res OR UCF) ; set the UCF bit
0006  23   r0_disp equ      (r0_res OR UCF OR DEN) ; set the DEN bit
                                24
A414  25   r1_d   equ      0A414h        ; 1010 character height 10 scan lines
                                26   ; 0 INL non interlaced mode
                                27   ; 1 MAS VSYNK & HSYNK are outputs
                                28   ; 0 EN Non composite sync mode
                                29   ; 00 THM TMS twin mode is disabled
                                30   ; 0 dont care bit
                                31   ; 0 EVC CKID is output.video clk is
                                32   ; derived from XTALIN
                                33   ; 1 PCE priority counter enabled
                                34   ; 0 FAE use RDY as READY signal
                                35   ; 1 RE CPU can read the DRAM
                                36   ; 0 PSA QCLK = 1/16 XTALIN
                                37   ; 0 PRE disable pipeline read
                                38
0006  39   r2_d   equ      0006h        ; register window base addr is 0000h
                                40   ; 11 TF2 TF1 digital pixel codes
                                41   ; 0 ME no margin
                                42
0140  43   r3_d   equ      0140h        ; data window base addr is 0000h
                                44   ; right edge of the screen is at
                                45   ; x = 327
                                46
8000  47   r4_d   equ      8000h        ; data window length 64k bytes
                                48
0000  49   r5_d   equ      0000h        ; data segment base addr is 0000h
                                50   ; note data segment and register

```

231679-14

```

LOC  OBJ          LINE  SOURCE
                                     ; segment overlap. In the overlapped
51                                     ; region register segment will
52                                     ; overwrite the data segment
53                                     ; 00 BSI BSO bank 0
54
55
56      U00A        56      r6_d   equ      000Ah      ; CPU is allowed 10 accesses during each
57                                     ; line building process.
58
59      0500        59      r7_d   equ      0500h      ; object descriptor table starts at
60                                     ; 0500h in bank 0
61
62      001C        62      r8_d   equ      0010h      ; access table at 0010h
63
64      0180        64      r9_d   equ      0180h      ; color look up table base addr
65
66      0023        66      r10_d  equ      0023h      ; char gen0 at 2000h
67                                     ; char gen1 at 3000h
68
69      0010        69      r11_d  equ      0010h
70
71      0402        71      r12_d  equ      0402h      ; HC0.VC0
72
73      1024        73      r13_d  equ      1024h      ; HC1.VC1
74
75      74EC        75      r14_d  equ      74ECh      ; HC2.VC2
76
77      80F4        77      r15_d  equ      80F4h      ; HC3.VC3
78
79      ; allocate memory for register and data segments
80
81
82      video_vsdd  segment      at 6000H
83
84      0000 (1     84      r0_v   dw      1      dup(?)
85      )
86
87      0002 (1     85      r1_v   dw      1      dup(?)
88      )
89
90      0004 (1     86      r2_v   dw      1      dup(?)
91      )
92
93      0006 (1     87      r3_v   dw      1      dup(?)
94      )
95
96      0008 (1     88      r4_v   dw      1      dup(?)
97      )
98
99      000A (1     89      r5_v   dw      1      dup(?)
100     )
101
102     000C (1     90      r6_v   dw      1      dup(?)
103     )
104
105     000E (1     91      r7_v   dw      1      dup(?)

```

LOC	OBJ	LINE	SOURCE				
0010	(1 ????)	92	r8_v	dw	1	dup(?)	
0012	(1 ????)	93	r9_v	dw	1	dup(?)	
0014	(1 ????)	94	r10_v	dw	1	dup(?)	
0016	(1 ????)	95	r11_v	dw	1	dup(?)	
0018	(1 ????)	96	r12_v	dw	1	dup(?)	
001A	(1 ????)	97	r13_v	dw	1	dup(?)	
001C	(1 ????)	98	r14_v	dw	1	dup(?)	
001E	(1 ????)	99	r15_v	dw	1	dup(?)	
0020	(512 ????)	100					
		101		org	020h		
		102	oat_v	dw	512	dup(?)	;Object Access Table(max length)
		103					
		104					
		105		org	400h		;power up locations of registers
0400	(1 ????)	106	ir0_v	dw	1	dup(?)	
0402	(1 ????)	107	ir1_v	dw	1	dup(?)	
0404	(1 ????)	108	ir2_v	dw	1	dup(?)	
0406	(1 ????)	109	ir3_v	dw	1	dup(?)	
0408	(1 ????)	110	ir4_v	dw	1	dup(?)	
040A	(1 ????)	111	ir5_v	dw	1	dup(?)	
040C	(1 ????)	112	ir6_v	dw	1	dup(?)	

LDC OBJ	LINE	SOURCE				
040E (1 ????)	113	ir7_v	dw	1	dup(?)	
0410 (1 ????)	114	ir8_v	dw	1	dup(?)	
0412 (1 ????)	115	ir9_v	dw	1	dup(?)	
0414 (1 ????)	116	ir10_v	dw	1	dup(?)	
0416 (1 ????)	117	ir11_v	dw	1	dup(?)	
0418 (1 ????)	118	ir12_v	dw	1	dup(?)	
041A (1 ????)	119	ir13_v	dw	1	dup(?)	
041C (1 ????)	120	ir14_v	dw	1	dup(?)	
041E (1 ????)	121	ir15_v	dw	1	dup(?)	
0300	122		org	300H		
0300 (16 ????)	123	clut_v	dw	16	dup(?)	; colour lookup table
0A00	124		org	0A00H		
0A00 (4 ????)	125	odt0_v	dw	4	dup(?)	; Object descriptor table
0A08 (4 ????)	126	odt1_v	dw	4	dup(?)	
0A10 (4 ????)	127	odt2_v	dw	4	dup(?)	
0A18 (4 ????)	128	odt3_v	dw	4	dup(?)	
	129					
2000	130		org	2000H		; 3.6K bytes
2000 (1800 ????)	131	object_0_v	dw	1800	dup(?)	; 4 bits/pixel 75*96 bit mapped
2E14	132		org	2E14H		
2E14 (80 ????)	133	object_1_v	dw	80	dup(?)	
2E00	134		org	2E00H		
2E00 (120	135	object_2_v	dw	120	dup(?)	

231679-17

LDC	OBJ	LINE	SOURCE
	2FB4	136	org 2FB4H
	2FB4 (100	137	object_3_v dw 100 dup(?)
	4000	138	org 04000H
	4000 (256	139	cg0_slice_0_v dw 256 ;Character Generator 0 dup(?)
	4200 (256	140	cg0_slice_1_v dw 256 dup(?)
	4400 (256	141	cg0_slice_2_v dw 256 dup(?)
	4600 (256	142	cg0_slice_3_v dw 256 dup(?)
	4800 (256	143	cg0_slice_4_v dw 256 dup(?)
	4A00 (256	144	cg0_slice_5_v dw 256 dup(?)
	4C00 (256	145	cg0_slice_6_v dw 256 dup(?)
	4E00 (256	146	cg0_slice_7_v dw 256 dup(?)
	5000 (256	147	cg0_slice_8_v dw 256 dup(?)
	5200 (256	148	cg0_slice_9_v dw 256 dup(?)
	5400 (256	149	cg0_slice_10_v dw 256 dup(?)
	5600 (256	150	cg0_slice_11_v dw 256 dup(?)
	5800 (256	151	cg0_slice_12_v dw 256 dup(?)
	5A00 (256	152	cg0_slice_13_v dw 256 dup(?)
	5C00 (256	153	cg0_slice_14_v dw 256 dup(?)
	5E00 (256	154	cg0_slice_15_v dw 256 dup(?)

```

LOC  OBJ          LINE  SOURCE
-----
                                155
                                156  video_vsdd      ends
                                157
                                158
                                159  video_data      segment
                                160
0000  20564944454F20  161          object_3_data      db      ' VIDEO STORAGE '
      53544F52414745
      2020
                                162
0010  2020202020414E  163          db      '      AND      '
      44202020202020
      2020
                                164
0020  20444953504C41  165          db      ' DISPLAY DEVICE '
      59204445564943
      4520
                                166
0030  20202020383237  167          db      '      82716      '
      31362020202020
      2020
                                168
0040  20202020494E54  169          db      '      INTEL      '
      454C0920202020
      20
                                170
                                171  ; Data for character generator 0
                                172  ; Characters are 10 scan lines high
                                173  ; Slice #0,1 and 9 are empty(0's)
                                174  ; 26 alphabets and 10 numbers are defined
                                175
                                176  ; slice information for 10 numbers
                                177
                                178
                                179
004F  3C              180          numbers_data      db      3Ch, 7Eh, 7Eh, 3Ch, 60h ;slice 2
0050  7E
0051  7E
0052  3C
0053  60
0054  3C              181          db      3Ch, 3Ch, 18h, 3Ch, 3Ch
0055  3C
0056  18
0057  3C
0058  3C
0059  66              182          db      66h, 18h, 06h, 66h, 60h ;slice 3
005A  18
005B  06
005C  66
005D  60
005E  66              183          db      66h, 66h, 18h, 66h, 66h
005F  66
0060  18
0061  66
0062  66

```

LOC	OBJ	LINE	SOURCE
0063	66	184	
0064	18		db 66h, 18h, 0Ch, 60h, 7Eh ; slice 4
0065	0C		
0066	60		
0067	7E		
0068	60	185	db 60h, 66h, 18h, 66h, 60h
0069	66		
006A	18		
006B	66		
006C	60		
006D	6E	186	db 6Eh, 18h, 30h, 38h, 66h ; slice 5
006E	18		
006F	30		
0070	38		
0071	66		
0072	60	187	db 60h, 3Eh, 18h, 3Ch, 7Ch
0073	3E		
0074	18		
0075	3C		
0076	7C		
0077	76	188	db 76h, 1Ch, 60h, 60h, 78h ; slice 6
0078	1C		
0079	60		
007A	60		
007B	78		
007C	3E	189	db 3Eh, 06h, 30h, 66h, 66h
007D	06		
007E	30		
007F	66		
0080	66		
0081	66	190	db 66h, 18h, 66h, 66h, 70h ; slice 7
0082	18		
0083	66		
0084	66		
0085	70		
0086	02	191	db 02h, 66h, 66h, 66h, 66h
0087	66		
0088	66		
0089	66		
008A	66		
008B	3C	192	db 3Ch, 18h, 3Ch, 3Ch, 60h ; slice 8
008C	18		
008D	3C		
008E	3C		
008F	60		
0090	7E	193	db 7Eh, 3Ch, 7Eh, 3Ch, 3Ch
0091	3C		
0092	7E		
0093	3C		
0094	3C		
		194	
		195	
		196	slice information for 26 alphabets
		197	
		198	;character_set_0

231679-20

LOC	OBJ	LINE	SOURCE		
		199			;slices 0, 1 and 9 are 0's (empty)
		200			
0095	66	201	slice_2_d	db	66H, 3EH, 3BH, 1EH, 7EH, 06H, 3BH, 66H
0096	3E				
0097	38				
0098	1E				
0099	7E				
009A	06				
009B	38				
009C	66				
009D	3C	202		db	3CH, 3CH, 66H, 7EH, 66H, 46H, 3CH, 0CH
009E	3C				
009F	66				
00A0	7E				
00A1	66				
00A2	46				
00A3	3C				
00A4	0C				
00A5	7C	203		db	7CH, 66H, 3CH, 1BH, 3CH, 1BH, 3CH, 66H
00A6	66				
00A7	3C				
00A8	1B				
00A9	3C				
00AA	1B				
00AB	3C				
00AC	66				
00AD	1B	204		db	1BH, 7EH
00AE	7E				
00AF	66	205	slice_3_d	db	66H, 66H, 6CH, 36H, 06H, 06H, 6CH, 66H
00B0	66				
00B1	6C				
00B2	36				
00B3	06				
00B4	06				
00B5	6C				
00B6	66				
00B7	1B	206		db	1BH, 66H, 36H, 7EH, 66H, 66H, 66H, 06H
00B8	66				
00B9	36				
00BA	7E				
00BB	66				
00BC	66				
00BD	66				
00BE	06				
00BF	26	207		db	26H, 36H, 66H, 1BH, 66H, 1BH, 7EH, 66H
00C0	36				
00C1	66				
00C2	1B				
00C3	66				
00C4	1B				
00C5	7E				
00C6	66				
00C7	1B	208		db	1BH, 06H
00C8	06				
00C9	7E	209	slice_4_d	db	7EH, 66H, 06H, 66H, 06H, 06H, 46H, 66H

LDC	OBJ	LINE	SOURCE		
00CA	66				
00CB	06				
00CC	66				
00CD	06				
00CE	06				
00CF	46				
00D0	66				
00D1	18	210		db	18H, 60H, 0EH, 06H, 66H, 76H, 66H, 06H
00D2	60				
00D3	0E				
00D4	06				
00D5	66				
00D6	76				
00D7	66				
00D8	06				
00D9	36	211		db	36H, 1EH, 60H, 18H, 66H, 3CH, 5AH, 3CH
00DA	1E				
00DB	60				
00DC	18				
00DD	66				
00DE	3C				
00DF	5A				
00E0	3C				
00E1	18	212		db	18H, 0CH
00E2	0C				
00E3	66	213	slice_5_d	db	66H, 3EH, 06H, 66H, 1EH, 1EH, 66H, 7EH
00E4	3E				
00E5	06				
00E6	66				
00E7	1E				
00E8	1E				
00E9	66				
00EA	7E				
00EB	18	214		db	18H, 60H, 06H, 06H, 66H, 7EH, 66H, 3EH
00EC	60				
00ED	06				
00EE	06				
00EF	66				
00F0	7E				
00F1	66				
00F2	3E	215		db	66H, 3EH, 3CH, 18H, 66H, 24H, 5AH, 18H
00F3	66				
00F4	3E				
00F5	3C				
00F6	18				
00F7	66				
00F8	24				
00F9	5A				
00FA	18				
00FB	3C	216		db	3CH, 18H
00FC	18				
00FD	3C	217	slice_6_d	db	3CH, 66H, 06H, 66H, 06H, 06H, 06H, 66H
00FE	66				
00FF	06				
0100	66				

LOC	OBJ	LINE	SOURCE		
0101	06				
0102	06				
0103	06				
0104	66				
0105	18				
0106	60	218		db	18H, 60H, 0EH, 06H, 7EH, 6EH, 66H, 66H
0107	0E				
0108	06				
0109	7E				
010A	6E				
010B	66				
010C	66				
010D	66	219		db	66H, 66H, 06H, 18H, 66H, 66H, 42H, 3CH
010E	66				
010F	06				
0110	18				
0111	66				
0112	66				
0113	42				
0114	3C				
0115	3C	220		db	3CH, 30H
0116	30				
0117	3C	221	slice_7_d	db	3CH, 66H, 6CH, 36H, 06H, 06H, 6CH, 66H
0118	66				
0119	6C				
011A	36				
011B	06				
011C	06				
011D	6C				
011E	66				
011F	18	222		db	18H, 60H, 36H, 06H, 66H, 66H, 66H, 66H
0120	60				
0121	36				
0122	06				
0123	66				
0124	66				
0125	66				
0126	66				
0127	66	223		db	66H, 66H, 66H, 18H, 66H, 42H, 42H, 66H
0128	66				
0129	66				
012A	18				
012B	66				
012C	42				
012D	42				
012E	66				
012F	66	224		db	66H, 60H
0130	60				
0131	18	225	slice_8_d	db	18H, 3EH, 38H, 1EH, 7EH, 7EH, 38H, 66H
0132	3E				
0133	38				
0134	1E				
0135	7E				
0136	7E				
0137	38				

231679-23

```

LOC OBJ          LINE  SOURCE
0138 66
0139 3C          226          db      3CH, 70H, 66H, 06H, 24H, 62H, 3CH, 3EH
013A 70
013B 66
013C 06
013D 24
013E 62
013F 3C
0140 3E
0141 3C          227          db      3CH, 3EH, 3CH, 7EH, 66H, 42H, 42H, 66H
0142 3E
0143 3C
0144 7E
0145 66
0146 42
0147 42
0148 66          228          db      66H, 7EH
0149 66
014A 7E
-----
229
230          video_data      ends
231
232
233
234          ; monitor segment. The GSP board monitor starts at OFFF0h
235          monitor      segment      at      OFFF0h
236
237          org 0
0000 (1          238          reset      dw      1 dup(?)      ; a label
0000 (1
      )
-----
239
240          monitor      ends
241
242          ; -----
243          ; Main routine. This routine loads the DRAM with access table ;
244          ; the object descriptor table, the character generator and the ;
245          ; actual object data. The VSDD is also initialized. ;
246          ; -----
247          prog_code      segment
248
249          assume cs:prog_code, ds:video_data, es:video_vsdd
250
0000          251          simple_display      proc      far
0000          252          start:
-----
253
0000 8B----- R          254          mov ax,video_data      ; initialize the data segment
0003 8ED8          255          mov ds,ax
-----
256
0005 8B0060          257          mov ax,video_vsdd      ; initialize the extra
0008 8ECC          258          mov es,ax              ; segment
-----
259
260          ; initialize the register segment. the register
261          ; segment is located at 0400h after reset.
262

```



```

LOC  OBJ                LINE  SOURCE
000A  26C70600047260      263          mov ir0_v,r0_res
0011  26C706020414A4      264          mov ir1_v,r1_d
0018  26C70604040600      265          mov ir2_v,r2_d
001F  26C70606044001      266          mov ir3_v,r3_d
0026  26C70608040080      267          mov ir4_v,r4_d
002D  26C7060A040000      268          mov ir5_v,r5_d
0034  26C7060C040A00      269          mov ir6_v,r6_d
003B  26C7060E040005      270          mov ir7_v,r7_d
0042  26C70610041000      271          mov ir8_v,r8_d
0049  26C70612048001      272          mov ir9_v,r9_d
0050  26C70614042300      273          mov ir10_v,r10_d
0057  26C70616041000      274          mov ir11_v,r11_d
005E  26C70618040204      275          mov ir12_v,r12_d
0065  26C7061A042410      276          mov ir13_v,r13_d
006C  26C7061C04EC74      277          mov ir14_v,r14_d
0073  26C7061E04F480      278          mov ir15_v,r15_d
279
280
281          ; all the registers are initialized in the DRAM. Enable the UCF
282          ; flag to allow the VSDD to update its on chip registers.
283
007A  26C70600047360      284          mov ir0_v,r0_up
285
286          ; wait 150us for the VSDD to update its on chip registers
287          ; the loop assumes that the B0186 runs at 6Mhz.
288
0081  B94700              289          mov cx,71
0084  E2FE                290          loop1: loop loop1
291          ; the register window is initialized to 60000h
292          ; the cpu programs the display data through newly
293          ; defined data window in R3.
294
295          ; load the object descriptor fields for four objects
296
297          ; object 0
298
0086  26C706000A0006      299          mov odt0_v,600h          ;4bits/pixel.non-transparent
008D  26C706020A0018      300          mov odt0_v[2],1800h     ;object starts at x = 0
301          ; the width is 96 pixels
0094  26C706040A0010      302          mov odt0_v[4],1000h    ;object base address
009B  26C706060A0010      303          mov odt0_v[6],1000h
304
305          ; object 1
306
00A2  26C706080A0006      307          mov odt1_v,600h
00A9  26C7060A0A1404      308          mov odt1_v[2],0414h    ; x = 20,width = 16 pixels
00B0  26C7060C0A0A17      309          mov odt1_v[4],170ah
00B7  26C7060E0A0A17      310          mov odt1_v[6],170ah
311
312          ; object 2
313
00BE  26C706100A0006      314          mov odt2_v,600h
00C5  26C706120A263C      315          mov odt2_v[2],3c26h    ; x = 38,width = 240 pixels
00CC  26C706140A6017      316          mov odt2_v[4],1760h
00D3  26C706160A6017      317          mov odt2_v[6],1760h

```

231679-25

```

LOC  OBJ          LINE  SOURCE
                                318
                                319      ; object 3
                                320
00DA 26C706180A04AC      321      mov odt3_v, 0AC04h      ; character object
                                322                                ; 8 pixels/character
                                323                                ; transparent pixel
00E1 26C7061A0A5008      324      mov odt3_v[2], 0B50h    ; x = 80, Width = 16
                                325                                ; characters
00E8 26C7061C0ADA17      326      mov odt3_v[4], 17DAh
00EF 26C7061E0ADA17      327      mov odt3_v[6], 17DAh
                                328
                                329
                                330      ; set up the object data
                                331
00F6 BA0200              332      mov dx, 2
00F9 BB0000              333      mov bx, 0
                                334
                                335      ; object 0
00FC B90807              336      mov cx, 24*75          ; number of data words -
                                337                                ; 75 lines, 24 words (96 pixels)
00FF B88888              338      mov ax, 8888h          ; pixel data
                                339
0102 2689870020          340      fill_obj_0:      mov object_0_v[bx], ax
0107 03DA                341                        add bx, dx
0109 E2F7                342      loop fill_obj_0
                                343
                                344      ; object 1
                                345
010B BB0000              346      mov bx, 0
010E B95000              347      mov cx, 4*20          ; number of data words
0111 B87777              348      mov ax, 7777h          ; pixel data
                                349
0114 268987142E          350      fill_obj_1:      mov object_1_v[bx], ax
0119 03DA                351                        add bx, dx
011B E2F7                352      loop fill_obj_1
                                353
                                354      ; object 2
                                355
011D BB0000              356      mov bx, 0
0120 B93C00              357      mov cx, 15*4
0123 B85555              358      mov ax, 5555h
                                359
0126 268987C02E          360      fill_obj_2:      mov object_2_v[bx], ax
0128 03DA                361                        add bx, dx
012D E2F7                362      loop fill_obj_2
                                363
                                364      ; object 3 - character object
                                365
012F BB0000              366      mov bx, 0
0132 B92800              367      mov cx, 40            ; total 80 characters
                                368                                ; in the object, 2/word
                                369
0135 B807                370      fill_obj_3:      mov ax, word ptr object_3_data[bx] ; read the ASCII code
                                371                                ; for 2 characters

```

LOC	OBJ	LINE	SOURCE
0137	268987B42F	372	mov object_3_v[bx],ax ; write the data
013C	03DA	373	add bx,dx
013E	E2F5	374	loop fill_obj_3
		375	
		376	; load the character generator
		377	
0140	B80000	378	mov ax,0
0143	B80000	379	mov bx,0
		380	; load the numbers
		381	
0146	BE6000	382	mov si,30h*2 ; store at proper ASCII
		383	; location. Note cpu space
		384	; byte addressable
0149	B90A00	385	mov cx,10 ; 10 numbers
014C	BA0700	386	mov dx,7 ; 7 slices
		387	
014F		388	write_a_number:
		389	
014F	8A474F	390	mov al,numbers_data[bx] ; read data byte
0152	2689840044	391	mov cg0_slice_2_v[si],ax ; write data word in
		392	; the DRAM
0157	43	393	inc bx ; next byte
0158	83C602	394	add si,2 ; next location in
		395	; the DRAM
0158	E2F2	396	loop write_a_number
		397	
015D	81C6EC01	398	add si,(256*2)-20 ; next slice
0161	B90A00	399	mov cx,10
0164	4A	400	dec dx
		401	
0165	75E8	402	jnz write_a_number
		403	
		404	; store the 26 alphabets
		405	
0167	B80000	406	mov ax,0
016A	B80000	407	mov bx,0
016D	B91A00	408	mov cx,26 ; 26 alphabets
0170	BE8200	409	mov si,41h*2 ; proper offset into
		410	; character generator
0173	BA0700	411	mov dx,7 ; 7 slices
		412	
0176		413	write_a_character:
		414	
0176	BAB79500	415	mov al,slice_2_d[bx] ; read data byte
017A	2689840044	416	mov cg0_slice_2_v[si],ax ; write a word
017F	43	417	inc bx ; next byte
0180	83C602	418	add si,2 ; next location
		419	
0183	E2F1	420	loop write_a_character
		421	
0185	B91A00	422	mov cx,26
0188	81C6CC01	423	add si,(256*2)-52 ; next slice
018C	4A	424	dec dx
018D	75E7	425	jnz write_a_character
		426	

231679-27

```

LOC  OBJ                LINE  SOURCE
                                427
                                428 ; load the access table
                                429
018F  BB0000            429          mov bx,0
0192  B90002            430          mov cx,length oat_v
0195  8BFFFF            431          mov ax,0ffffh
0198  BA0200            432          mov dx,2
                                433
                                434 ; fill the access table with all 1s
                                435
0198  26B94720          436  fill_oat:   mov oat_v[ebx],ax
019F  03DA                437          add bx,dx
                                438
01A1  E2F8                439          loop fill_oat
                                440
                                441 ; enable the objects
                                442
01A3  BBFEFF            443          mov ax,0ffffh
01A6  26A32000          444          mov oat_v,ax ; enable object 0 at line 0
01AA  26A3B600          445          mov oat_v[75*2],ax ; disable object 0 at line 75
                                446
01AE  BBFDFF            447          mov ax,0fffdh
01B1  26A30601          448          mov oat_v[115*2],ax ; enable object 1 at line 115
01B5  26A32C01          449          mov oat_v[134*2],ax ; disable object 1 at line 131
                                450
01B9  8BF8FF            451          mov ax,0fff8h
01BC  26A33B01          452          mov oat_v[140*2],ax ; enable object 2 at line 140
01C0  26A33A01          453          mov oat_v[141*2],ax ; disable object 2 at line 141
                                454
01C4  8BF7FF            455          mov ax,0fff7h
01C7  26A34B00          456          mov oat_v[20*2],ax ; enable object 3 at line 20
01CB  26A3AC00          457          mov oat_v[70*2],ax ; disable object 3 at line 70
                                458
                                459
                                460 ; the display data is initialized by the 80186. Now enable the
                                461 ; set the display enable bit (DEN) in the VSDD to enable the
                                462 ; display.
                                463
01CF  26C70600007B60    464          mov r0_v,r0_disp ; the register segment is now located
                                465 ; at 60000h
                                466
                                467
                                468 ; a simple routine to scroll objects horizontally and vertically
                                469 ; object 0 is moved horizontally while object 1 is moved vertic-
                                470 ; ally
                                471
01D6                                472          movexy:
01D6  26C706020A001B    473          mov odt0_v[2],1800h
01DD  BA4201            474          mov dx,322 ; maximum value of x for obj 0
01E0  BB0000            475          mov bx,0 ; start y value for obj 1
                                476
01E3  268306020A01    477          movex:   add odt0_v[2],1 ; mov obj 0 by 2 pixels in x direction
01E9  4A                478          dec dx
                                479
                                480
01EA  26816720FDFF    481          movex:   and oat_v[ebx],0fffdh ; object 1 start

```

231679-28

LOC	OBJ	LINE	SOURCE
01F0	26816748FDFF	482	and oat_v[bx+40],0fff0h ; object 1 stop
01F6	897017	483	mov cx,6000 ; delay counter
01F9	26C706080A0006	484	mov odt1_v,600h ; turn the object on
0200	E2FE	485	delay2: loop delay2
0202	26C706080A1006	486	mov odt1_v,610h ; turn the object off
		487	; before disabling the
		488	; access table
0209	26814F20F2FF	489	or oat_v[bx],0fff2h ; reset the access table
020F	26814F48F2FF	490	or oat_v[bx+40],0fff2h ; to original values
0215	83C302	491	add bx,2
0218	81FB3403	492	cmp bx,820 ; max value of y is 410
		493	
021C	748B	494	je movexy ; if y = max then start
		495	; at top of frame
021E	83FA00	496	cmp dx,0
0221	74C7	497	je movey ; if x = max then finish y move
0223	EBBE	498	jmp movex ; else continue with x move
		499	
0225	EA0000F0FF	500	jmp far ptr RESET ; go back to the board monitor
		501	
		502	simple_display endp
		503	
----		504	prog_code ends
		505	
		506	end start

231679-29

XREF SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
??SEG	SEGMENT		SIZE=0000H PARA PUBLIC
C00_SLICE_0_V	V WORD	4000H	(256) VIDED_VSDD 139#
C00_SLICE_1_V	V WORD	4200H	(256) VIDED_VSDD 140#
C00_SLICE_10_V	V WORD	5400H	(256) VIDED_VSDD 149#
C00_SLICE_11_V	V WORD	5600H	(256) VIDED_VSDD 150#
C00_SLICE_12_V	V WORD	5800H	(256) VIDED_VSDD 151#
C00_SLICE_13_V	V WORD	5A00H	(256) VIDED_VSDD 152#
C00_SLICE_14_V	V WORD	5C00H	(256) VIDED_VSDD 153#
C00_SLICE_15_V	V WORD	5E00H	(256) VIDED_VSDD 154#
C00_SLICE_2_V	V WORD	4400H	(256) VIDED_VSDD 141# 391 416
C00_SLICE_3_V	V WORD	4600H	(256) VIDED_VSDD 142#
C00_SLICE_4_V	V WORD	4800H	(256) VIDED_VSDD 143#
C00_SLICE_5_V	V WORD	4A00H	(256) VIDED_VSDD 144#
C00_SLICE_6_V	V WORD	4C00H	(256) VIDED_VSDD 145#
C00_SLICE_7_V	V WORD	4E00H	(256) VIDED_VSDD 146#
C00_SLICE_8_V	V WORD	5000H	(256) VIDED_VSDD 147#
C00_SLICE_9_V	V WORD	5200H	(256) VIDED_VSDD 148#
CLUT_V	V WORD	0300H	(16) VIDED_VSDD 123#
DELAY2	L NEAR	0200H	PR0G_CODE 485# 485
DEN	NUMBER	0008H	11# 23
FILL_DAT	L NEAR	019BH	PR0G_CODE 436# 439
FILL_OBJ_0	L NEAR	0102H	PR0G_CODE 340# 342
FILL_OBJ_1	L NEAR	0144H	PR0G_CODE 350# 352
FILL_OBJ_2	L NEAR	0126H	PR0G_CODE 360# 362
FILL_OBJ_3	L NEAR	0135H	PR0G_CODE 370# 374
IR0_V	V WORD	0400H	VIDED_VSDD 106# 263 284
IR1_V	V WORD	0402H	VIDED_VSDD 107# 264
IR10_V	V WORD	0414H	VIDED_VSDD 116# 273
IR11_V	V WORD	0416H	VIDED_VSDD 117# 274
IR12_V	V WORD	0418H	VIDED_VSDD 118# 275
IR13_V	V WORD	041AH	VIDED_VSDD 119# 276
IR14_V	V WORD	041CH	VIDED_VSDD 120# 277
IR15_V	V WORD	041EH	VIDED_VSDD 121# 278
IR2_V	V WORD	0404H	VIDED_VSDD 108# 265
IR3_V	V WORD	0406H	VIDED_VSDD 109# 266
IR4_V	V WORD	0408H	VIDED_VSDD 110# 267
IR5_V	V WORD	040AH	VIDED_VSDD 111# 268
IR6_V	V WORD	040CH	VIDED_VSDD 112# 269
IR7_V	V WORD	040EH	VIDED_VSDD 113# 270
IR8_V	V WORD	0410H	VIDED_VSDD 114# 271
IR9_V	V WORD	0412H	VIDED_VSDD 115# 272
LOOP1	L NEAR	0084H	PR0G_CODE 290# 290
MONITOR	SEGMENT		SIZE=0002H PARA ABS 235# 240
MOVEX	L NEAR	01E3H	PR0G_CODE 477# 498
MOVEXY	L NEAR	01D6H	PR0G_CODE 472# 494
MOVEY	L NEAR	01EAH	PR0G_CODE 481# 497
NUMBERS_DATA	V BYTE	004FH	(5) VIDED_DATA 180# 390
DAT_V	V WORD	0020H	(512) VIDED_VSDD 102# 430 436 444 445 448 449 452 453 456 457 481 482 489 490
OBJECT_0_V	V WORD	2000H	(1800) VIDED_VSDD 131# 340
OBJECT_1_V	V WORD	2E14H	(80) VIDED_VSDD 133# 350
OBJECT_2_V	V WORD	2EC0H	(120) VIDED_VSDD 135# 360

NAME	TYPE	VALUE	ATTRIBUTES, XREFS
OBJECT_3_DATA	V BYTE	0000H	(16) VIDEO_DATA 161# 370
OBJECT_3_V	V WORD	2FB4H	(100) VIDEO_VSDD 137# 372
ODT0_V	V WORD	0A00H	(4) VIDEO_VSDD 125# 299 300 302 303 473 477
ODT1_V	V WORD	0A08H	(4) VIDEO_VSDD 126# 307 308 309 310 484 486
ODT2_V	V WORD	0A10H	(4) VIDEO_VSDD 127# 314 315 316 317
ODT3_V	V WORD	0A18H	(4) VIDEO_VSDD 128# 321 324 326 327
PROG_CODE	SEGMENT		SIZE=022AH PARA 247# 249 504
RO_DISP	NUMBER	607BH	23# 464
RO_RES	NUMBER	6072H	13# 22 23 263
RO_UP	NUMBER	6073H	22# 284
RO_V	V WORD	0000H	VIDEO_VSDD 84# 464
R1_D	NUMBER	A414H	25# 264
R1_V	V WORD	0002H	VIDEO_VSDD 85#
R10_D	NUMBER	0023H	66# 273
R10_V	V WORD	0014H	VIDEO_VSDD 94#
R11_D	NUMBER	0010H	69# 274
R11_V	V WORD	0016H	VIDEO_VSDD 95#
R12_D	NUMBER	0402H	71# 275
R12_V	V WORD	0018H	VIDEO_VSDD 96#
R13_D	NUMBER	1024H	73# 276
R13_V	V WORD	001AH	VIDEO_VSDD 97#
R14_D	NUMBER	74ECH	75# 277
R14_V	V WORD	001CH	VIDEO_VSDD 98#
R15_D	NUMBER	80F4H	77# 278
R15_V	V WORD	001EH	VIDEO_VSDD 99#
R2_D	NUMBER	0006H	39# 265
R2_V	V WORD	0004H	VIDEO_VSDD 86#
R3_D	NUMBER	0140H	43# 266
R3_V	V WORD	0006H	VIDEO_VSDD 87#
R4_D	NUMBER	8000H	47# 267
R4_V	V WORD	0008H	VIDEO_VSDD 88#
R5_D	NUMBER	0000H	49# 268
R5_V	V WORD	000AH	VIDEO_VSDD 89#
R6_D	NUMBER	000AH	56# 269
R6_V	V WORD	000CH	VIDEO_VSDD 90#
R7_D	NUMBER	0500H	59# 270
R7_V	V WORD	000EH	VIDEO_VSDD 91#
R8_D	NUMBER	0010H	62# 271
R8_V	V WORD	0010H	VIDEO_VSDD 92#
R9_D	NUMBER	0180H	64# 272
R9_V	V WORD	0012H	VIDEO_VSDD 93#
RESET	V WORD	0000H	MONITOR 238# 500
SIMPLE_DISPLAY	P FAR	0060H	SIZE=022AH PROG_CODE 251# 502
SLICE_2_D	V BYTE	0095H	(8) VIDEO_DATA 201# 415
SLICE_3_D	V BYTE	00AFH	(8) VIDEO_DATA 205#
SLICE_4_D	V BYTE	00C9H	(8) VIDEO_DATA 209#
SLICE_5_D	V BYTE	00E3H	(8) VIDEO_DATA 213#
SLICE_6_D	V BYTE	00FDH	(8) VIDEO_DATA 217#
SLICE_7_D	V BYTE	0117H	(8) VIDEO_DATA 221#
SLICE_8_D	V BYTE	0131H	(8) VIDEO_DATA 225#
START	L NEAR	0000H	PROG_CODE 252# 506
UCF	NUMBER	0001H	10# 22 23
VIDEO_DATA	SEGMENT		SIZE=014BH PARA 159# 230 249 254
VIDEO_VSDD	SEGMENT		SIZE=4000H PARA ABS 82# 156 249 257
WRITE_A_CHARACTER	L NEAR	0176H	PROG_CODE 413# 420 425

231679-31

NAME	TYPE	VALUE	ATTRIBUTES	XREFS
WRITE_A_NUMBER	L NEAR	014FH	PROG_CODE	388# 396 402
END OF SYMBOL TABLE LISTING				
ASSEMBLY COMPLETE, NO ERRORS FOUND				

231679-32



**APPLICATION
NOTE**

AP-270

October 1986

82786 Hardware Configuration

Order Number: 292007-002

1.0 INTRODUCTION

The 82786 is an intelligent co-processor capable of creating and displaying high performance graphics. Both drawing and display functions are integrated into a single VLSI chip to provide an inexpensive solution for bit-mapped graphics subsystems.

This application note is intended to show, through examples, use of the 82786 and the hardware interfaces between the 82786 and the rest of the system. Because the 82786 integrates many functions onto one chip, the hardware design of a graphics system is greatly simplified.

2.0 OVERVIEW

Internally, the 82786 consists of two independent processors.

- Graphics Processor: executes high-level line drawing, character drawing and bit-block-transfer commands to create and modify bit-maps in memory
- Display Processor: displays portions of bit-maps in regions on the CRT called windows

Figure 1 illustrates these processors and their hardware interfaces.

- Graphics Memory Interface: connects dedicated graphics memory to the 82786
- System Bus Interface: connects CPU and system memory to the 82786
- Video Interface: connects the 82786 to CRT or other display

The video interface is controlled directly by the Display Processor. The other interfaces are controlled by the

82786 Bus Interface Unit (BIU). The BIU connects the internal Graphics and Display Processors to the CPU and system memory as well as to the graphics memory through the internal DRAM/VRAM controller.

2.1 Dedicated Graphics Memory

The dedicated graphics memory provides the 82786 with very fast access to memory without contention with the CPU and system memory. Typically, the bit-maps to be drawn and displayed, the character fonts, and the command lists for the 82786 processors are all stored in this memory. In some instances it is desirable to have the Graphics Processor command lists stored in system memory.

The 82786 contains a complete DRAM/VRAM controller on-chip which interfaces directly with a wide variety of DRAMs without external logic. This direct connection not only reduces chip count but also allows the 82786 to perform very fast burst accesses to the DRAMs. The DRAM/VRAM controller can take advantage of the quick burst-mode sequential accesses made possible by page-mode, fast-page-mode (sometimes called Ripplemode™), and Static Column DRAMs. In addition, interleaved DRAM/VRAM arrays are fully supported by the on-chip DRAM/VRAM controller allowing even faster burst access.

2.2 System Bus Interface

The system bus interface connects the CPU and its system memory to the 82786 and its graphics memory.

The most common 82786 configuration (shown in Figure 1) allows the CPU to access the system memory while the 82786 accesses its dedicated graphics memory simultaneously. It also allows the CPU to access the graphics memory and for the 82786 to access the system memory (but not simultaneously). The system bus

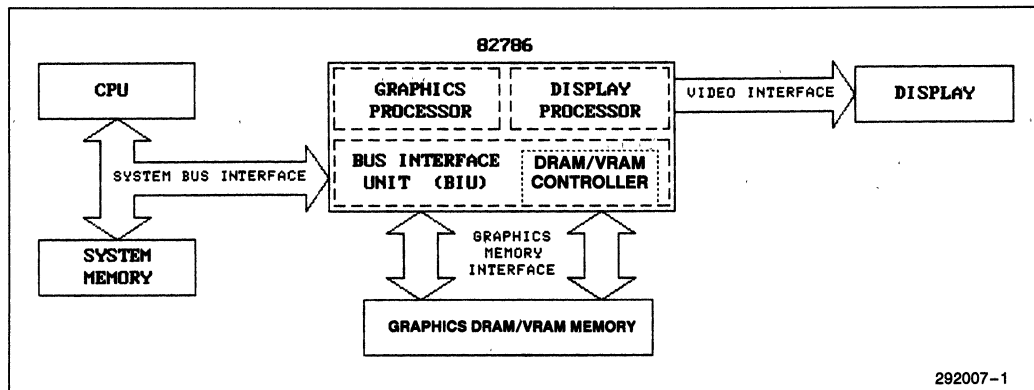


Figure 1. 82786 System Block Diagram

connects the 82786 graphics subsystem to the system CPU and memory. If DMA capability is also provided in the system, it interfaces to the 82786 exactly as the CPU does. The interface allows accesses in two directions.

- Slave Mode: CPU or DMA read or write access of the 82786 internal registers or dedicated graphics memory through the 82786
- Master Mode: 82786 read or write access to system memory

Therefore, any processor (CPU, DMA, Graphics and Display Processors) can access both the system memory and the graphics memory. The 82786 BIU arbitrates between both of the internal 82786 processors as well as the external processor (CPU and DMA) to decide which processor gets access of the bus.

The CPU software accesses both system and graphics memory in an identical manner (except that the specific memory addresses are different). Therefore the actual location of the memory (whether in system or graphics memory) is transparent to the software. However, the CPU can access the system memory faster than the graphics memory because there is less contention with the Graphics and Display Processors. When the CPU accesses the 82786, the 82786 BIU is said to be running in slave mode.

In slave mode, the 82786 looks like an intelligent DRAM/VRAM controller to the CPU (Figure 2). The CPU can chip-select the 82786 and the 82786 will acknowledge when the cycle is complete by generating a READY signal for the CPU.

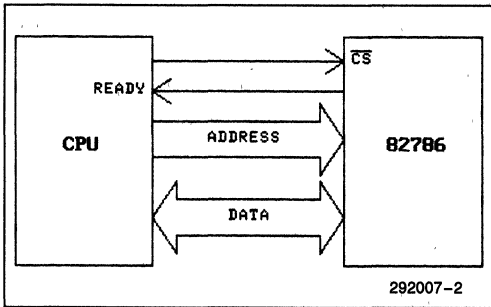


Figure 2. Slave Bus Cycle

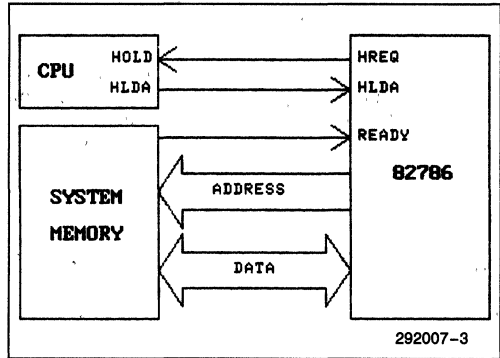


Figure 3. Master Bus Cycle

Conversely, the 82786 Graphics and Display Processors access both system memory and graphics memory in an identical manner. However, they can access the graphics memory faster than the system memory because there is less contention with the CPU. When the 82786 accesses the system memory, the 82786 BIU is said to be running in master mode.

In master mode, the 82786 looks like a second CPU controlling the local bus (Figure 3). The 82786 activates HOLD to request control of the system bus. When the CPU acknowledges the HLDA line, then the 82786 will take over the bus. When the 82786 is through with the bus, it will release HOLD and the CPU can remove HLDA to regain control of the bus.

The 82786 system bus interface is optimized to interface to an 80286 synchronously (using the same bus clock). As a synchronous slave it interprets the 80286 status lines directly and performs the requested bus accesses. As a master it generates 80286 style bus signals.

The 82786 system bus may alternatively be set up to interface asynchronously to virtually any processor. In this mode, read and write signals are used when slave accesses are performed.

2.3 Video Interface

The 82786 supports two different video interfaces in order to support both standard DRAMs and dual port video DRAMs/(VRAMs). When using standard

DRAMs the 82786 reads the video data from memory and internally serializes the video data to generate the serial video data stream up to 25 MHz. When using VRAMs the 82786 loads the VRAM shift register periodically and the internal system generates the serial video data stream.

With standard DRAMs displays up to 640 by 480 by 8 resolution can be generated at 60 Hz non-interlaced refresh. With VRAMs displays up to 2048 by 1936 by 8 can be generated at 60 Hz without interlacing.

In addition, horizontal and vertical sync signals and a blanking signal are provided and may be programmed to satisfy the requirements of nearly any CRT.

In the standard DRAM mode all of the logic to support the advanced capabilities of the Display Processor such as panning, zooming, windowing, and switching between various bits/pixel in various windows is contained internally in the 82786. Provision is also made for the addition of up to four external color look-up tables.

Higher resolution displays (dot clock rates greater than 25 MHz) can also be implemented by using external logic to trade-off bits/pixel for dot clock rates. Also, multiple 82786s can be used together for even greater performance.

2.4 82786 Internal Registers

A 64 word (128 byte) direct-mapped register block is contained internally in the 82786 (Figure 4). Software may locate this register block to the beginning of any 128 byte boundary anywhere in the 82786 I/O or memory address space. No matter where these registers are mapped, they are only accessible by the external CPU. The Graphics and Display Processors can not access these registers.

Registers, located at specified offsets within this block, allow programming of the BIU and Graphics and Display Processors. The Graphics and Display Processors also have other registers which are only accessible through commands to these processors. These commands are initiated by writing into the corresponding opcode and address registers within this 128 byte register block.

All of these registers are described in detail in the 82786 data sheet. Be careful when using "reserved" registers. When these reserved registers are read, the data returned is indeterminate. When these reserved registers are written, they should only be written as zeros to ensure compatibility with future products.

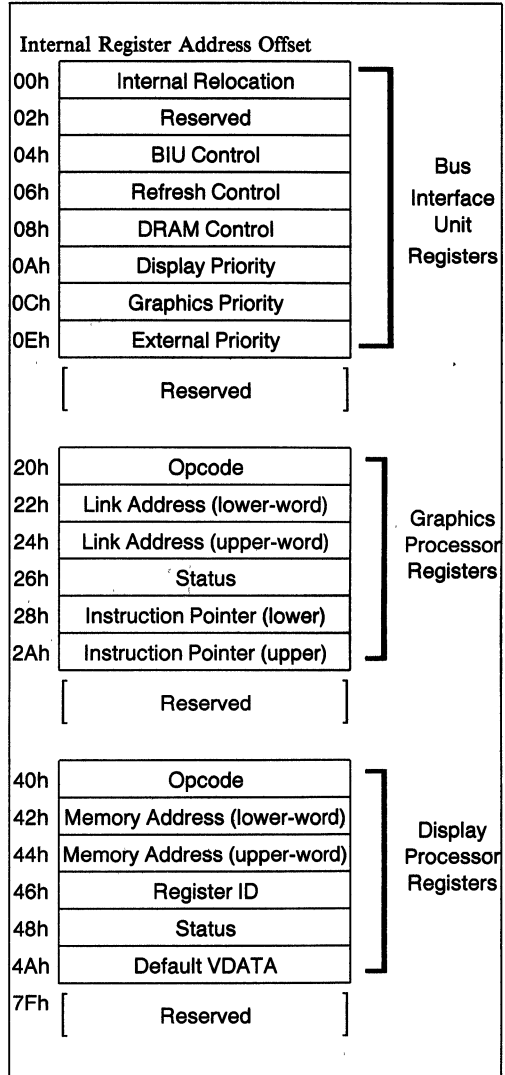


Figure 4. 82786 Internal Registers

3.0 DEDICATED GRAPHICS MEMORY INTERFACE

The 82786 contains a full DRAM/VRAM controller on-chip which allows it to be connected directly to arrays of DRAMs without external logic.

A wide range of DRAM configurations are possible for x 1, x 4 and x 8 bit wide DRAMs. Both Page mode and

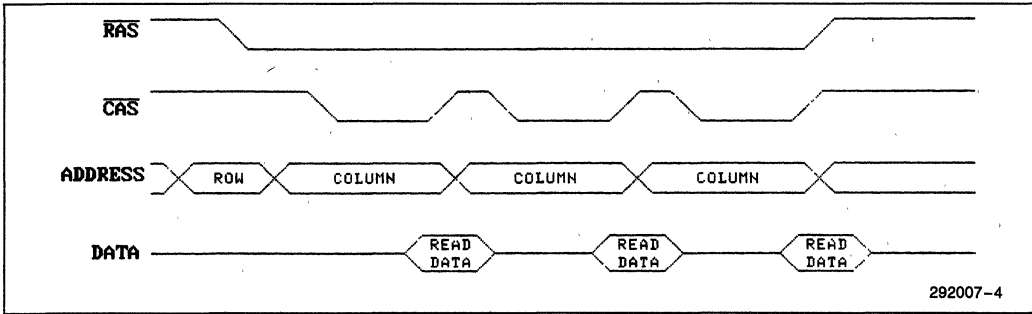


Figure 5. Fast-Page-Mode Burst-Access Read Cycle

Fast-page-mode burst accesses for block transfers are supported directly by the 82786 to take advantage of the fast sequential addressing capability of DRAMs (see Figure 5). Once the DRAM is set-up with the row address, the column addresses can be quickly scanned in for several burst-accesses to the same page. With the 82786, fast-page-mode bursts for block transfers run at twice the speed of page mode.

Interleaving of two banks of DRAMs is also supported directly by the 82786. For a sequential burst access, DRAM cycles for both banks can be initiated. Then, during the burst access, the 82786 can alternate accesses between the two banks, thus cutting the effective DRAM access time in half (see Figure 6).

Static Column DRAMs can also be used to get the same performance as fast-page-mode. The only difference between the two types is that Static Column DRAMs do not latch the column address, whereas, fast-page-mode DRAMs do latch the column address on the falling edge of CAS. In non-interleaved configurations, Static Column DRAMs can directly replace fast-page-mode. However, in an interleaved configuration, the column address must be latched externally for Static Column DRAMs.

The following table shows the burst-access rate of these various configurations for a 10 MHz 82786.

	Page Mode	Fast-page-mode and Static Column
Non-interleaved:	10 Mbyte/sec (2 cycles)	20 Mbyte/sec (1 cycle)
Interleaved:	20 Mbyte/sec (1 cycle)	30 Mbyte/sec (0.5 cycle)

The other cycle times, and speeds at 10 MHz, are the same for all DRAM configurations:

Single Reads	3 cycles	300 ns
Single Writes	3 cycles	300 ns
Read-Modify-Writes	4 cycles	400 ns
Burst-Access Set-Up	2 cycles	200 ns
Refresh	3 cycles	300 ns

All burst-accesses for block transfers perform an even number of 16-bit word accesses.

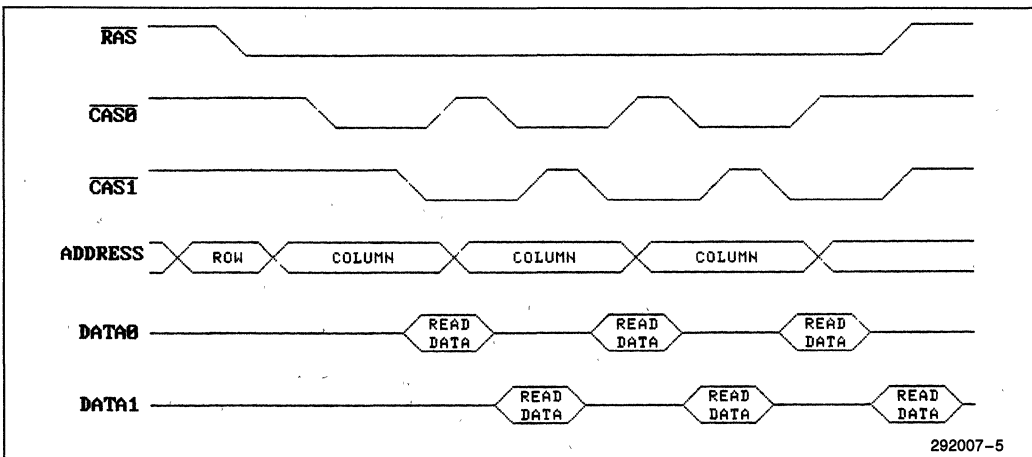


Figure 6. Interleaved Fast-Page-Mode Burst-Access Read Cycle

Burst-accesses for block transfers are used by all Display Processor memory accesses except the operand for LD_REG and DMP_REG operands. Block-read accesses are used by the Graphics Processor for command-block fetching and to fetch the character fonts. The Graphics Processor uses a block-read followed by a block-write for the read-modify-write operations of BitBlt, Scan_Line, and Character drawing. All other pixel drawing uses single read-modify-write cycles.

3.1 DRAM Configurations

Up to 4 rows per bank, and 1 non-interleaved or 2 interleaved banks are supported (see Figure 7). Each bank must always be 16 bits wide. If only one non-interleaved bank is used, it must be bank 0 (using CAS0 and BEN0). If interleaving is used, both banks must have the same number of rows. In either case, if only one row is used, it must be row 0 (using RAS0). For only two rows, row 0 and 1 are used (RAS0 and RAS1). Similarly, three rows use row 0, 1, and 2.

The 82786 can directly drive up to 32 DRAM/VRAM chips. One 82786 pin shares two DRAM functions DRA9/RAS3. These functions are never both used in the same configuration. DRA9 is only used by 1M x 1 DRAMs, which limit the number of rows to only two due to both addressing (4 Megabytes) and drive (32 chips) limitations.

Figure 8 shows a full connection diagram for thirty-two 64K x 4 DRAMs. Two interleaved banks of four rows each are used. Unlike most DRAM/VRAM controllers, no impedance-matching resistors are usually needed between the 82786 chip and the DRAM/VRAM chips. The impedance-matching for most configura-

tions is handled internally by the 82786. This is also the connections required for x4 VRAMs which use the BEN signal to control their DT/OE input which is used to determine when to load their internal shift register (Figure 9).

If Static Column DRAMs are used in an interleaved configuration, an external latch is required to latch the column address for the second bank (Figure 8a). The 82786 can directly drive up to thirty-two DRAM devices. For configurations requiring more than thirty-two devices, external buffering must be used.

DRAMs with separate data-in and data-out pins (such as the x 1 DRAMs) require a tristate buffer for the data-out lines of each bank. (All of the rows within each bank may share the same tristate buffer). Figure 10 shows a full connection diagram for thirty-two 256K x 1 DRAMs including the tristate buffers. Two interleaved banks of one row each are used. This is a special case for the RAS lines. Normally RAS0 would drive all of the DRAMs in both banks for the one row as in Figure 7. However, because the RAS lines have drive capability for only 16 DRAMs, both RAS0 and RAS1 are used. The 82786 recognizes this special case and automatically drives RAS1 identically to RAS0.

The other special DRAM case is using two rows of x 1 DRAMs in a non-interleaved configuration. This configuration has the advantage that only one bank of transceivers is required, but burst access time is reduced by half from the previous example. Normally, CAS0 would be used to drive all 32 DRAMs, but because of drive limitations, both CAS0 and CAS1 are used, (one for each bank). Again the 82786 recognizes this special case and automatically drives CAS1 identically to CAS0.

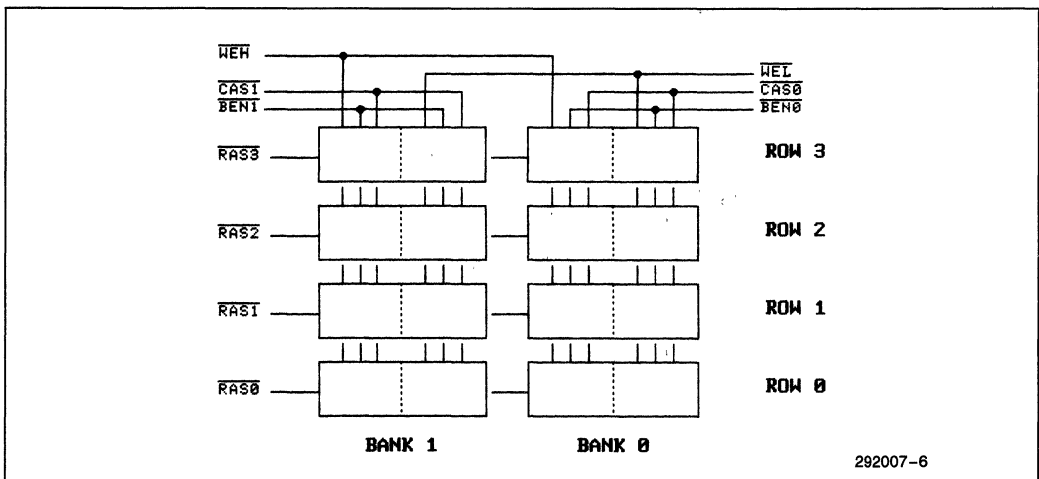
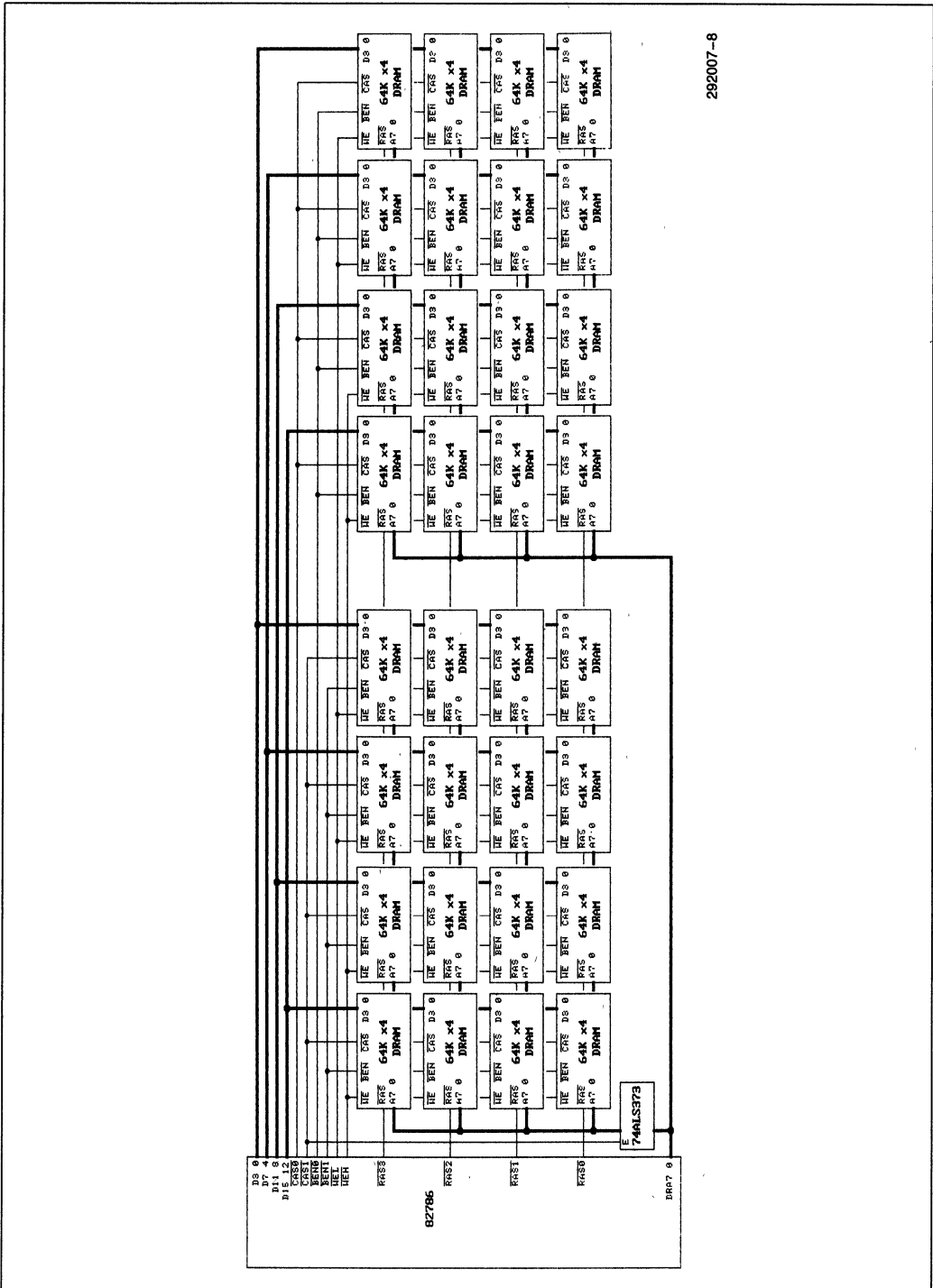
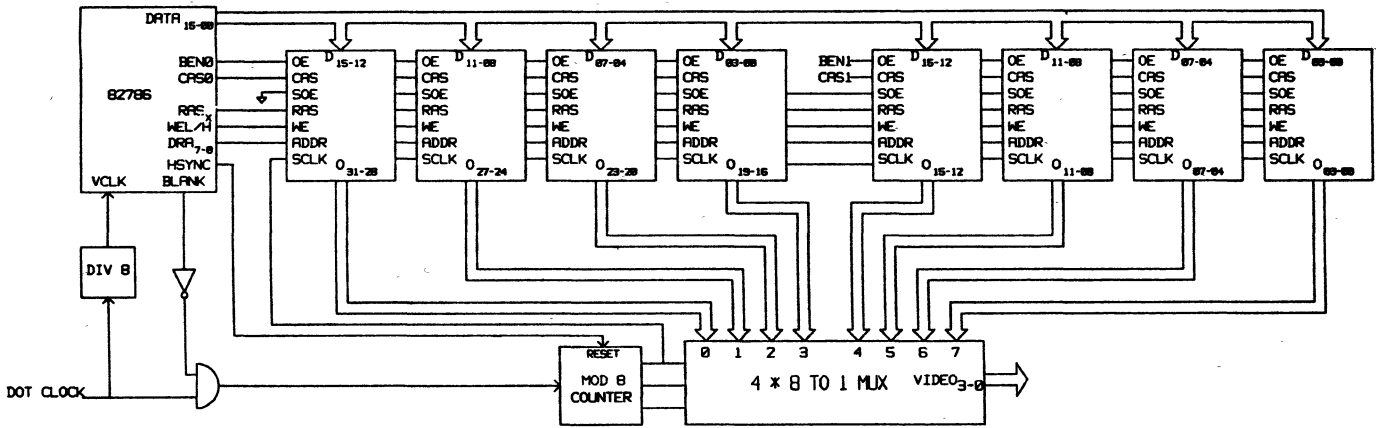


Figure 7. 82786 Supports up to 4 Rows of 2 Interleaved Banks of DRAMs
64K x 4 Video RAMs with 82786 1 Row, 2 Banks, 4 Bits/Pixel



282007-8

Figure 8a. 82786 Driving 4 Rows of Two Interleaved Banks of 64K x 4 Static Column DRAMs



292007-9

Figure 8b. 64K x 4 Video Rams with 82786 1 Row, 2 Banks, 4 Bits/Pixel

The table in Figure 11 shows all the possible configurations for 64K bit, 256K bit and 1 Mbit DRAMs.

3.2 DRAM Timing Parameters

Care should be taken to ensure that all of the timings of the DRAMs used, fit with those in the 82786 data sheet. To make the comparisons easier, the names of the parameter in the 82786 data sheet exactly correspond to the names in most DRAM data sheets. In addition, the parameters have been broken into the same four groups used by most DRAM data sheets.

The critical parameters for page mode DRAMs are generally:

Single rd/wrt/RMW	Single wrt	RMW	Page rd/wrt
Tcac Trp Trcd Trah Tasc Ton	Trwl Tcwl	Tds(rw) Toff	Tds(i)

Some of the 82786 parameters may not be found in all page-mode data sheets. If no corresponding DRAM parameters for Tcaa or Tcar is specified, then the 82786 spec may be ignored. The reason is that, if no such DRAM parameters exists, then the resulting minimum values for these parameters are at most:

$$Tcaa = Tasc + Tcac$$

$$Tcar = Tasc + Trsh$$

Then as long as the Tasc, Tcac, and Trsh specs fit, the 82786 timings guarantee Tcaa and Tcar to fit.

A third parameter that may not be found in all page-mode data sheets is Ton. If x 1 DRAMs are used, the external data transceiver is responsible for meeting this and the DRAM is not required to meet this spec. If, however, x 4 or x 8 DRAMs are used, without the data transceiver, care must be taken to ensure that this spec is met.

The critical parameters for Fast-page-mode and Static Column DRAMs are generally:

Single rd/wrt/RMW	Single wrt	RMW	Fast-page-mode rd/wrt
Trp Trah Tasc	Trwl Tcwl	Tds(rw) Toff	Tcp Tcaa Tcap Tds(n) Tcah(i) Tds(i) Tdh(i) Ton(ri)

For interleaved Static Column DRAMs, the address latch delay must be added to the DRAM parameters corresponding to the row and column addresses. These parameters are:

- Tasr
- Tasc
- Tcaa

For all types of x1 DRAMs, page-mode, Fast-page-mode and Static Column, the transceiver delay must be added to the DRAM parameters which correspond to read-data. These parameters are:

- Trac
- Tcac
- Tcaa

Notice that all of the 82786 DRAM timings are specified relative to the bus clock (CLK). This has two implications. First, a slower bus clock can be used to allow the 82786 to use slower DRAMs. Secondly, many of the parameters are determined by the duty cycle of the bus clock (as their specification is dependent on clock high or low time). A slightly non-symmetric clock, such as the clock for the 80286, can be used for the 82786 CLK, but care should be taken to examine the effects on the DRAM timing. In some circumstances, it may be advantageous to use a slightly non-symmetric clock.

Some of the specifications are relative to the 82786 clock period (Tc), while others are relative to a specific phase (THigh, TLow).

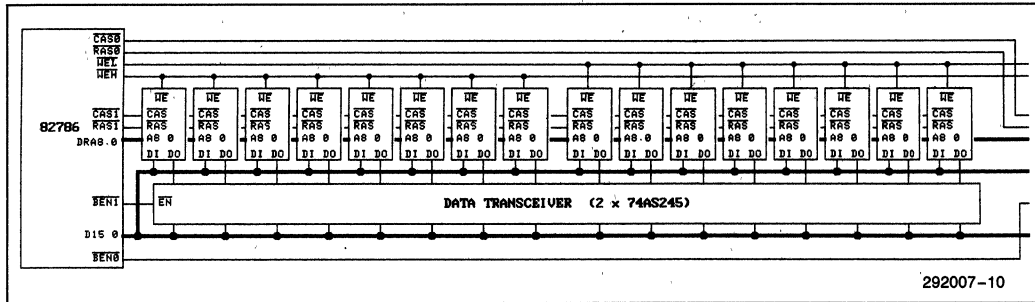


Figure 10. Two Interleaved Banks of 256K x 1 DRAMs

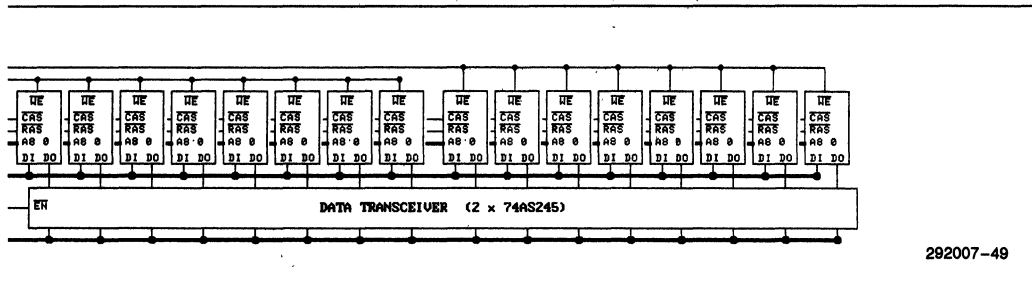
	Non-Interleaved				Interleaved			
	1-row	2-rows	3-rows	4-rows	1-row	2-rows	3-rows	4-rows
64K x1	128K 16	256K 32	384K 48*	512K 64*	256K 32	512K 64*	768K 96*	1024K 128*
16K x4	32K 4	64K 8	96K 12	128K 16	64K 8	128K 16	192K 24	256K 32
8K x8	16K 2	32K 4	48K 6	64K 8	32K 4	64K 8	96K 12	128K 16
256K x 1	512K 16	1024K 32	1536K 48*	2048K 64*	1024K 32	2048K 64*	3072K 96*	4096K 128*
64K x4	128K 4	256K 8	384K 12	512K 16	256K 8	512K 16	768K 24	1M 32
32K x8	64K 2	128K 4	192K 6	256K 8	128K 4	256K 8	384K 12	512K 16
1M x1	2M 16	4M 32	—	—	4M 32	—	—	—
256K x4	512K 4	1M 8	1.5M 12	2M 16	1M 8	2M 16	3M 24	4M 32
128K x8	256K 2	512K 4	768K 6	1M 8	512K 4	1M 8	1.5M 12	2M 16

*Requires external buffering

Figure 11. Possible DRAM configurations for 64K, 256K and 1 Mbit DRAMs. The top number in each box is total memory size in bytes, the bottom is the number of DRAM chips required.

Look at this example. Suppose you use 51C256H Fast-page-mode DRAMs with the 82786 as in Figure 10. First, look at the critical parameters shown above. Since it is not possible to create a precisely 50% duty cycle clock, you must consider clocks with a few percent tolerance. The table compares the 82786 using several clock frequencies and duty cycle tolerances with two versions of the 51C256H. The table is ordered with the tightest timings first.

From the table, you can see that the fast 120 ns access DRAMs can be used with the 82786 with a 10 MHz clock with as much as a 40%–60% duty cycle skew. The slower DRAMs can be used at 9 MHz with a tighter 45%–55% duty cycle skew or at 8 MHz with a 40%–60% skew.



292007-49

Figure 10. Two Interleaved Banks of 256K x 1 DRAMs (Continued)

Parameter		82786 Specifications				51C256H DRAM Specs		
		10 MHz 45-55%	10 MHz 40-60%	9 MHz 45-55%	8 MHz 40-60%	-12 120 ns	-15 150 ns	
Tdh	Min	Tph	22.5	20	25	25	20	25
Toff	Max	T1 + 3	25.5	23	28	28	20	25
Tcah	Min	Tch + 2	26.5	22	24.5	27	15	20
Tcp	Min	Tcl - 5	17.5	15	20	20	10	10
Tds	Min	Tcl - 8	14.5	12	17	17	0	0
Tcaa	Max	2Tc - 27	73	73	83	98	55	70
Tcap	Max	2Tc - 21	79	79	89	104	60	75
Tasc	Min	Tcl - 5	17.5	15	17.5	17.5	5	5
Trp	Min	2Tc - 5	95	95	105	120	70	85
Trwl	Min	Tc - 9	41	41	46	53.5	25	30
Tcwl	Min	Tc - 12	38	38	43	50.5	25	30
Trah	Min	Tc + 3	53	53	58	65.5	15	20
Ton	Max	Tc - 24	26	26	31	38.5	25	30

Because these x 1 DRAMs require transceivers between their data outputs and the 82786, the transceiver delays must also be considered. The two parameters in the table above, that are affected are Tcaa and Tcap. The transceiver delay must be added to the DRAM access time for these parameters. This implies that the data-in to data-out time of the transceivers must be 18 ns or less for the 10 MHz-120 ns case and the 8 MHz-150 ns case. The delay must be 28 ns or less for the 9 MHz-150 ns case and the 8 MHz-150 ns case.

3.3 Initializing the DRAM Controller

Two of the 82786 Internal Registers are used to configure the DRAM/VRAM Controller. Both of the regis-

ters are typically set once during initialization and then never changed. The DRAM/VRAM Control Register is set to indicate the configuration of the DRAMs/VRAMs used. The DRAM/VRAM Refresh Control Register is set to indicate the frequency of refresh cycles. Once programmed, the settings can be write-protected using the write-protect bits discussed in Section 4.2.

It is recommended that all fields of the DRAM/VRAM Control Register be written simultaneously to avoid illegal combinations. Also, no DRAM accesses should be attempted until the DRAM/VRAM Control Register has been set. For the configuration in Figure 10 using one row of 256K Fast-page-mode DRAMs in two interleaved banks:

$$\begin{aligned} \text{Request_time} &= \frac{16 \times (\text{RefreshCount} + 1)}{\text{CLK}} \\ &= \frac{16 \times (18 + 1)}{20 \text{ MHz}} = 15.2 \mu\text{s} \end{aligned}$$

The amount of latency that the DRAMs will tolerate for each row is:

$$\begin{aligned} \text{Allowed_Latency} &= \text{Tref} - (\text{RequestTime} \times \text{Refresh_Rows}) \\ &= 4 \text{ ms} - (15.2 \mu\text{s} \times 256) = 108.8 \mu\text{s} \end{aligned}$$

But the real latency limit is that the 82786 allows only three requests to be queued:

$$\begin{aligned} \text{Maximum_Latency} &= \text{Queue_Size} \times \text{Refresh_Time} \\ &= 3 \times 15.2 = 45.6 \mu\text{s} \end{aligned}$$

Therefore, the maximum number of wait-states allowed for a 82786 master mode transfer is:

$$\begin{aligned} \text{Wait_States} &= ((\text{Maximum_Latency} \times \text{PCLK}) - \text{overhead}) / \text{bus-cycles} \\ &= ((45.6 \mu\text{s} \times 10 \text{ MHz}) - 7 \text{ cycles}) / 2 = 224 \end{aligned}$$

Clearly, in this situation, refresh latency is not a problem. If the system memory caused the 82786 to delay over 224 wait-states for a master-mode access, not only would DRAM/VRAM refresh be missed, but the display refresh would also be lost.

The 82786 always issues three refresh cycles following a RESET. Besides these first three refresh cycles, the 82786 does not perform any other DRAM/VRAM warm-up after cold or warm-reset. If the DRAMs/VRAMs require other warm-up cycles, the CPU should either perform dummy cycles to the DRAM/VRAM or wait until the refresh counter has requested enough refresh cycles to occur.

If the DRAM/VRAM Refresh Control Register is set to all ones, refresh cycles are disabled.

4.0 SYSTEM BUS INTERFACE

The 82786 system bus structure allows the 82786 to be easily connected to a variety of CPUs. The 82786 can act as both a slave and a master to the CPU's bus. As a slave, the CPU or DMA can perform read and write cycles to the 82786 internal registers or to the 82786 DRAM/VRAM. As a master, the 82786 Graphics and Display Processors can perform read and write cycles to the CPU's system memory.

The 82786 bus can operate in three different modes to handle various CPU interfaces. The 82786 determines which mode to use by sampling the BHE and MIO pins during RESET:

	BHE	MIO
Synchronous 80286 bus	1	0
Synchronous 80186 bus	1	1
Asynchronous bus	0	X

For synchronous 80286 interfaces, the Reset and Clock inputs into the 80286 and 82786 must be common. For synchronous interfaces to 80186, the 80186 CLKIN must be the same as the 82786 CLK (so external clock source must be used). The RES input into the 80186 must meet a set up and hold time with respect to the CLKIN. The RESET for the 82786 should be generated from the RES (for 80186) by delaying RES by one CLKIN cycle and inverting it. This ensures that the 82786 ph1 is coincident with 80186 CLKOUT low.

These pin states are easy to achieve for the synchronous modes. During RESET, the 80286 always drives BHE high and MIO low.

CPUs with timings different from the 80286 must use asynchronous mode (however, CPUs such as the 80386 can easily generate 80286 style timings). Care should be taken in this case to ensure BHE is low during RESET.

In each of these three modes it is possible to configure the 82786 to allow both master and slave accesses or to simplify the logic to allow only slave access. In the master mode, the 82786 always generates 80286 style bus signals.

If the 82786 is used as a master, it will activate its HREQ line when it needs to become the system master to access system memory. It waits until HLDA is received and then begins driving the system bus. Once HLDA is received, a 10 MHz 82786 can perform system bus accesses at the following rate (assuming 0 wait-states).

single reads/writes	4 cycles	5 Mbyte/sec
read-modify-writes	6 cycles	3.3 Mbyte/sec
burst-access read/write	2 cycles	10 Mbyte/sec

The 82786 will begin the first master-mode bus access on the cycle after HLDA is activated. The only delay is the time between when the 82786 activates HREQ and the system can release the bus and return HLDA. Most synchronous CPUs require a minimum of three cycles between the time HOLD is activated until they can return HLDA. The 82786 will keep HREQ activated until it no longer has more accesses to perform to system memory. (Until either the next 82786 access is to the dedicated graphics DRAM/VRAM or until neither the Graphics or Display Processors require the bus.) Once the 82786 is done using the system bus, it will remove HREQ and is able to immediately access its Graphics DRAM/VRAM on the next cycle.

It is potentially possible for the 82786 to require the system bus for a lengthy period of time. For example, if the 82786 has been programmed to give the Graphics Processor high priority, and the Graphics Processor executes a command that requires a lot of access to system memory, then the system bus could potentially be held by the 82786 for several consecutive accesses. Drawing a long vector into a bit-map residing in system memory is such a command. In this case, the maximum time the 82786 can potentially keep the system bus is determined by the frequency of DRAM/VRAM refresh cycles programmed into DRAM/VRAM Refresh Control Register.

If the CPU needs to regain control of the bus before the 82786 is done, it may remove HLDA early. The 82786 will then complete the current access and remove HREQ to indicate to the CPU that it may now take-over control of the bus. If the 82786 still requires more access to the system bus, it will re-activate HREQ two cycles after it had removed it and wait until the next HLDA. Since the 82786 removes HREQ for only two cycles, it is important that the CPU recognize it immediately. Otherwise a lock-out condition will occur in which the CPU is waiting for the 82786 to remove HREQ and the 82786 is waiting for the CPU to issue HLDA. This is not a problem for the synchronous interfaces. Extra logic may be required to prevent this situation if the 82786 is used as a master in an asynchronous interface and HLDA is ever removed prematurely, especially if the CPU clock is significantly slower than the 82786 clock.

4.1 Memory Map

Figure 12 shows the memory map as it appears to both the 82786 Graphics and Display Processors. These processors both use a 22-bit address which provides for up to 4 Megabytes of address space. They are only allowed to make memory accesses so no I/O map is applicable.

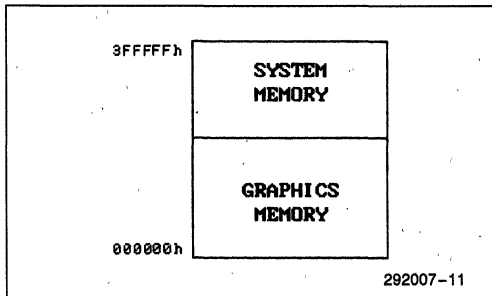


Figure 12. Memory Map for Graphics and Display Processors

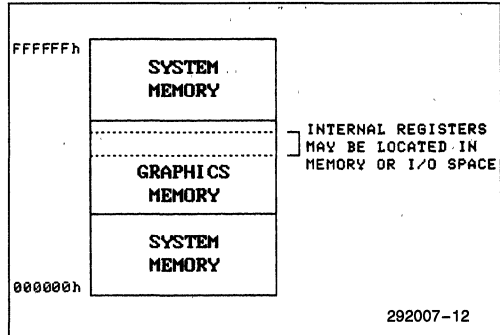


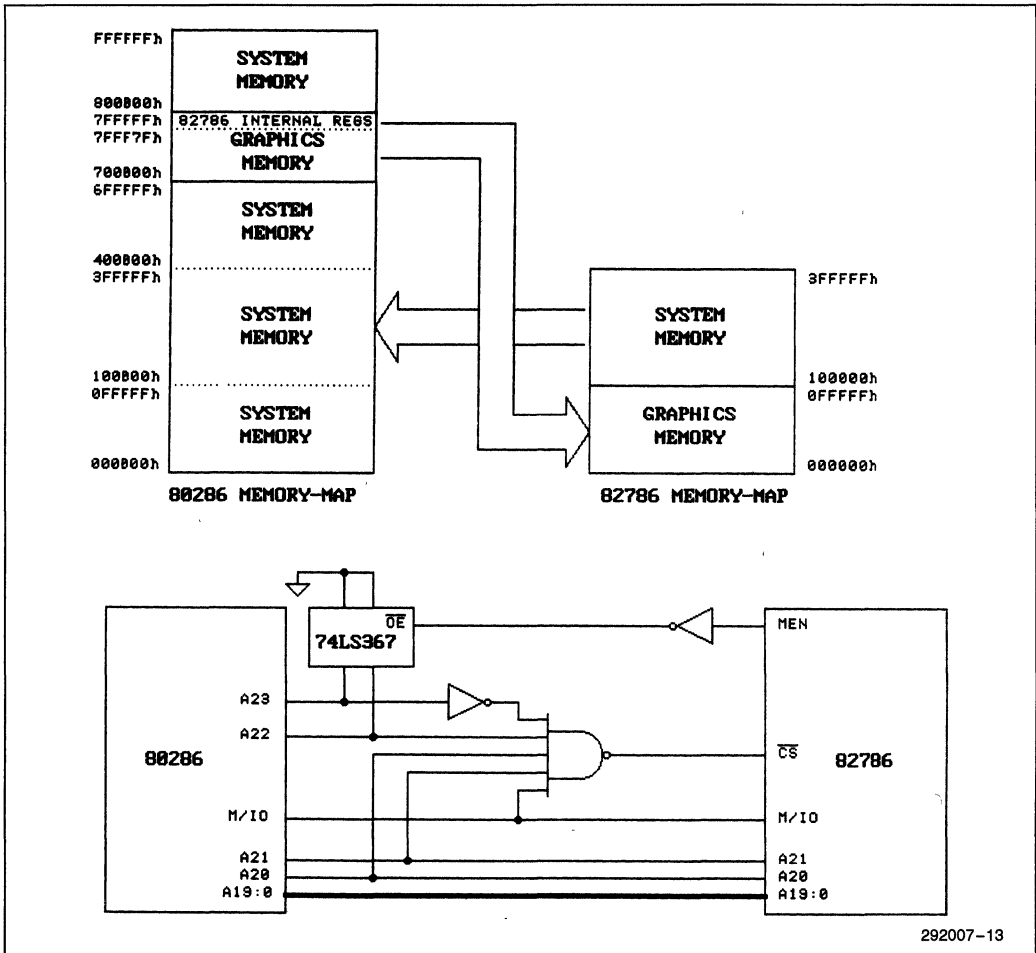
Figure 13. Memory Map for System CPU

The 82786 dedicated graphics DRAM/VRAM always starts at location 000000h and grows upwards. The upper address depends on the amount of DRAM/VRAM memory configured. The system bus memory begins where the DRAM/VRAM ends and continues to the highest addressable memory location 3FFFFFFh.

The memory map as it appears to the system CPU is shown in Figure 13. The area that the 82786 Graphics DRAM/VRAM is mapped into can be anywhere in the CPU address space and is completely defined by the address decode logic of the CPU system. Normally only the space for the configured graphics memory is mapped into CPU address space. If addresses above the configured graphics memory are mapped into the CPU address space, and the CPU writes to addresses above the configured 82786 memory, the write will be ignored. If it reads from these locations, the data returned is undefined.

The 82786 internal registers may be configured to reside in memory or I/O address space. If configured to reside in memory, then they will override a 128 byte area of the 82786 memory address space for external (CPU) accesses. The internal registers are only accessible by the external CPU and therefore are never found in the 82786 Graphics or Display Processor memory maps.

Suppose the 82786 is configured with 1 Megabyte of Graphics DRAM/VRAM and is used in an 80286 system. A possible memory map and connection diagram is shown in Figure 14. All of the 82786 memory is mapped into the 80286 address space. Also, a 3 Megabyte portion of the 80286 system memory is mapped into the 82786. Since the 80286 has two more address bits than the 82786, a tristate buffer is used to supply the top two address bits when 82786 enters master mode.



**Figure 14. Possible Memory Mapping for 80286/82786
82786 Internal Registers are Memory Mapped**

Notice that the same memory corresponds to one set of memory addresses for the CPU and a different set of memory address for the 82786 Graphics and Display Processors. Although it is possible to make these addresses match, it is not necessary as long as the controlling CPU software understands the relationship and makes the simple conversion. Often it is not desirable to make the addresses match. For example, most CPUs use the lowest memory addresses for special purposes, such as for interrupt vectors. If the lowest CPU memory were 82786 memory rather than the faster (for CPU access) system memory, then these operations would execute significantly slower.

Even though the real addresses don't match, the operating system for a CPU such as the 80286 could make the CPU's virtual addresses map easily to the 82786 real addresses.

The 82786 internal registers may either be memory or I/O mapped. If they are memory mapped over the Graphics DRAM/VRAM, the CPU will not be able to access the 128-bytes of DRAM/VRAM which they cover, (although the Graphics and Display Processors can). If they are memory mapped above the Graphics DRAM/VRAM (over non-configured memory), then they will not prevent the CPU from accessing any of the 82786 memory, but they must be included in the CPU memory space that the address decoder allocates for the 82786. The 82786 internal registers may be I/O mapped, so they do not overlap any memory, however the CPU chip select logic for the 82786 becomes slightly larger. Figure 15 shows a circuit similar to Figure 14, except the registers are I/O mapped. Memory mapping the internal registers allows the software slightly more flexibility in accessing the registers.

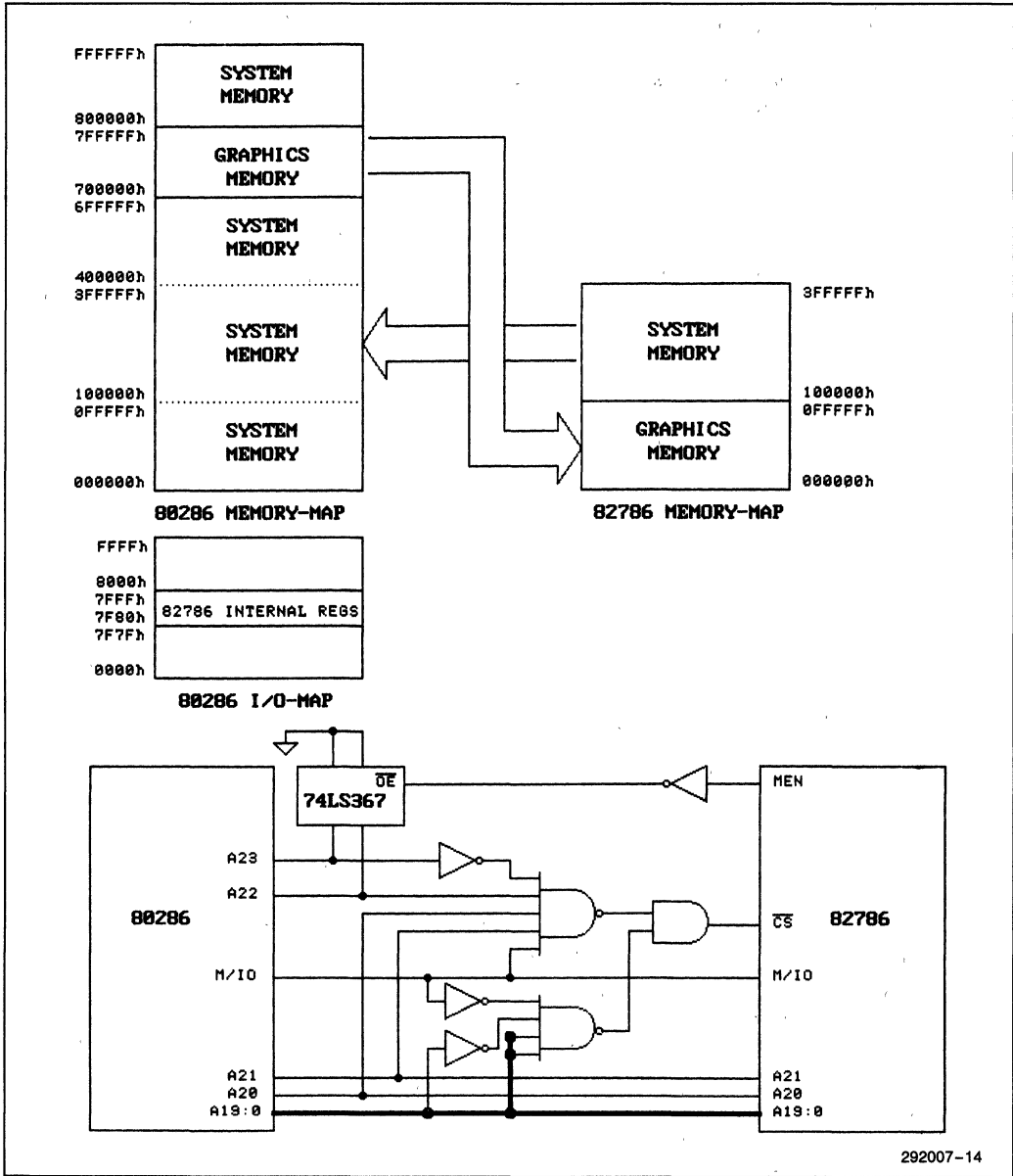


Figure 15. Possible Memory Mapping for 80286/82786
82786 Internal Registers are I/O Mapped

292007-14

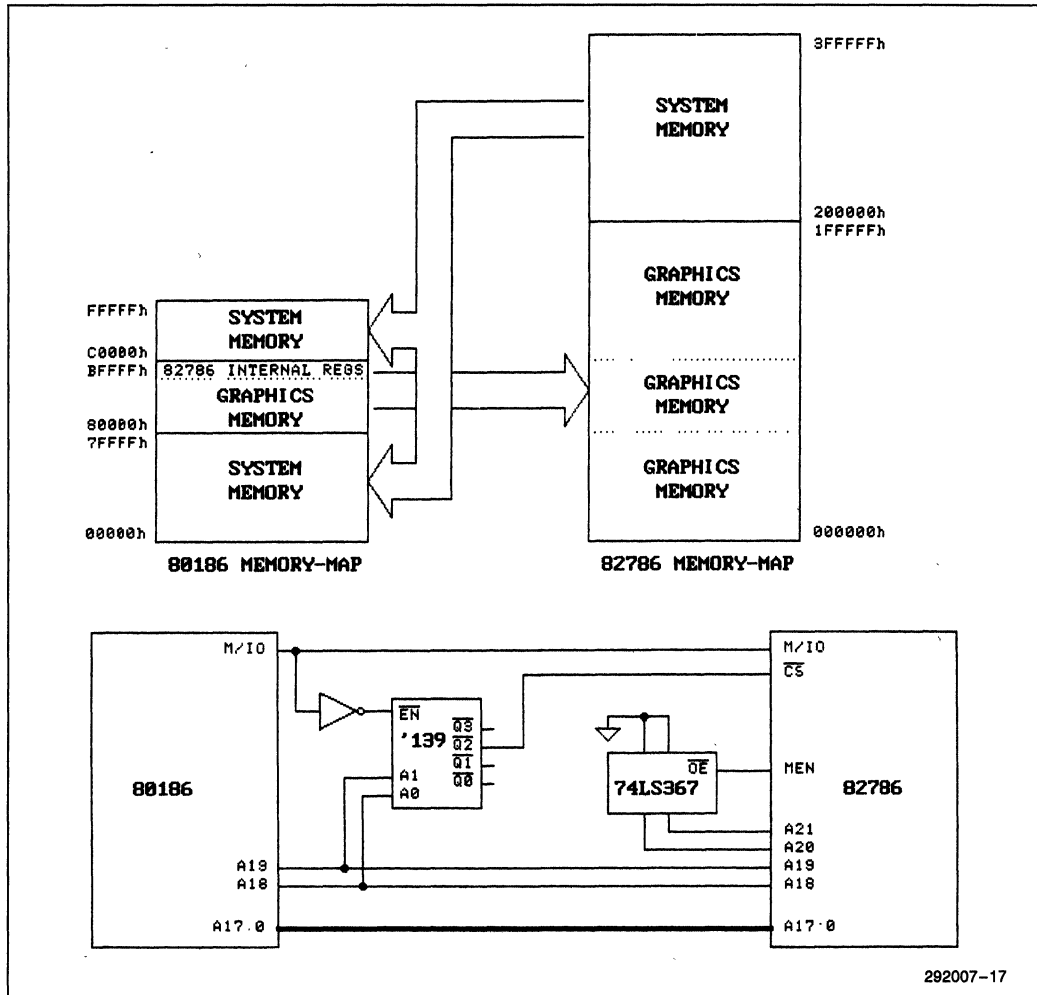


Figure 16. Possible Memory Mapping for 80186/82786

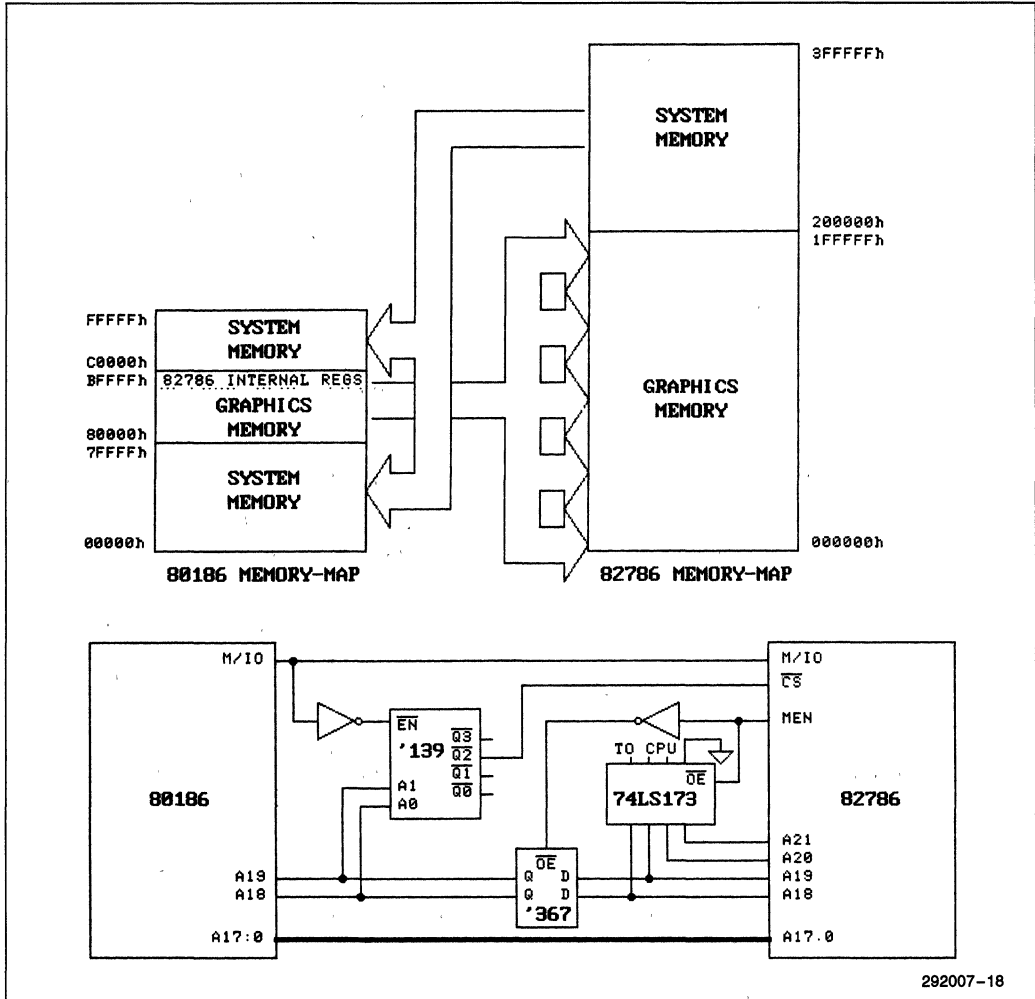
Because graphics memory can be quite large, some system designs might not allow all of the configured Graphics DRAM/VRAM to be directly mapped by the CPU. For example, if the 82786 has 2 Megabytes of Graphics DRAM/VRAM and is used with a 80186 processor, which can only address 1 Megabyte, then the 80186 can not directly access all of the 82786 memory. In this case the CPU can be permitted to only access a portion of the Graphics DRAM/VRAM. Figure 16 shows a memory map and connection diagram for such a system. Since the 82786 has two more address bits than the 80186, a tristate buffer is used to supply the two highest address bits when the 82786 is in slave mode.

In many cases the CPU does not require access to all of the graphics memory. For example, many situations

will not require the CPU to directly access the bit-maps. If the CPU must gain access to the Graphics memory which is not directly mapped to the CPU, the 82786 Graphics Processor can be instructed (using the BitBlt command) to move portions of the Graphics memory to and from the area accessible by the CPU.

Alternatively, the Graphics DRAM/VRAM areas can be bank switched to allow the CPU direct access at any portion of the graphics memory. Figure 17 shows the use of an I/O port (74LS173 latch) to which the CPU can write the highest 3 bits of the address for the 82786 slave accesses.

In both Figures 16 and 17, it is possible for the 82786 in master-mode to access the CPU memory addresses that



**Figure 17. Possible Memory Mapping for 80186/82786
Bank-Switching Allows 80186 to Access All 82786 Memory**

correspond to the 82786 slave addresses. In this case, the circuit will generate a 82786 chip-select, but the 82786 will not respond to this chip-select while it remains in master-mode. As long as the READY logic goes high (it may not since the 82786 will not perform the slave-access) then the 82786 will complete the master-mode cycle. By the time the 82786 returns to slave-mode, the chip-select will have gone away.

4.2 BIU Registers

Within the 82786 internal register block, the registers at offsets 00h-0Fh are used by the Bus interface Unit to

control the system configuration (Figure 4). These registers are normally set once during power-up initialization and never changed.

Two of these registers, DRAM/VRAM Refresh Control and DRAM/VRAM Control have already been discussed in Section 3.3. The rest of the registers are discussed in this section.

The Internal Relocation Register is used to locate the 82786 internal registers anywhere in the 82786 memory or I/O address space.

Internal Relocation - Internal Register Offset 00h

15	1	0
Base Address		MIO

RESET Default: XX XXXX XXXX XXXX X

Base Address: determines bits 21:7 of internal register address (bits 6:0 of address are used as offset)

0
0 = I/O mapped
1 = memory mapped

After RESET, any CPU slave I/O address to the 82786 (which activates the 82786 Chip-Select) will access the internal register block. During initialization, a write to the Internal Relocation Register should be performed to locate the register at the specific memory or I/O address desired. Once the write to the Internal Relocation Register occurs, the 82786 internal register block no longer occupies all of 82786 I/O space, rather it is restricted to just the 128 memory or I/O bytes specified. The internal registers can be located anywhere accessible by the CPU, however, if they are memory-mapped and located over configured graphics memory, they will take precedence over the memory for CPU accesses to those addresses. Graphics or Display Processor accesses to these addresses will still be directed to DRAM/VRAM. For example, writing the value of 03F8h locates the internal registers at I/O addresses FE00h – FE7Fh.

03F8h = 00 0000 1111 1110 0 0
| |
Base Address 00FE00h(offsets 0–7Fh) I/O mapped

Note that the address written to the Internal Relocation Register determines the memory or I/O address that is required to be placed on the 82786 address pins during a CPU access to the 82786 internal registers. The actual CPU address used may be different, and is dependent on the chip select and memory mapping logic described in Section 4.1.

- There are four sources of requests for the 82786 bus:
- DRAM/VRAM refresh
 - Display Processor
 - Graphics Processor
 - External Processor (CPU or DMA slave accesses)

The DRAM/VRAM refresh requests are always top priority. That is, once the DRAM/VRAM refresh request is made, the 82786 bus will complete the current bus access and then perform the DRAM/VRAM refresh. Three BIU registers are used to set the priorities of the other three bus requests. Two priority values are used:

FPL - First Priority Level - priority used when processor first requests bus.

SPL - Subsequent Priority Level - priority used for processor to maintain bus during a block transfer. If a block transfer is interrupted, this is also the priority used for regain bus to complete the burst access.

When a processor first requests the 82786 bus, its FPL value is used. The processor with the highest priority gets access to the bus. Once the bus is granted, the first access occurs. If a multiple-word block transfer is performed the SPL value is then used as the priority to maintain the bus for subsequent cycles. As long as no other processor of higher priority requests the bus, the burst-access is allowed to continue to completion. If a higher priority request is made, the block transfer will be suspended and the bus granted to the new request. The suspended block transfer will not get the bus back until its SPL value is again the highest priority request.

A separate register is used to program the priority for each of the three processors. Because the External Processor can not perform block transfers, no External SPL value is required for it.

Display Priority - Internal Register Offset 0Ah

15	6	5	4	3	2	1	0
Reserved		FPL			SPL		

RESET Default: 1 1 0 0 1 1

Graphics Priority - Internal Register Offset 0Ch

15	6	5	4	3	2	1	0
Reserved		FPL			SPL		

RESET Default: 1 0 1 0 1 0

External Priority - Internal Register Offset 0Eh

15	6	5	4	3	2	1	0
Reserved		FPL			Reserved		

RESET Default: 1 1 1

All of the priorities are programmable values from 0 to 7 with 7 being the highest priority. If two processors

that are programmed with the same priority both request the bus, the priority in which the bus will be granted for the two will be (from highest to lowest):

- Display Processor
- Graphics Processor
- External Processor

There are two exceptions to these programmable priorities. If the CPU makes a slave request while one of the 82786 processors makes a master request, the CPU's request will always be handled first by the 82786 regardless of priority settings. This is necessary to prevent the lock-out situation where the CPU will not grant HLDA until it completes the bus access to the 82786 and the 82786 will not complete the CPU bus cycle until the higher priority master cycle completes. Refresh cycles also always will be handled while the 82786 is in a HLDA loop.

The values programmed into these priority registers should be selected carefully. There is a performance penalty whenever a block transfer is interrupted. However, if block transfers are not interrupted, then it is possible that one processor must wait a long time to get the bus while another is finishing. A balance between overall bus performance and maximum tolerable latency must be made.

For example, if the Display Processor is not given high enough priority, it may not always be able to fetch the bit-mapped display data fast enough to keep up with the CRT. When this happens, the Display Processor will not be able to send the correct video data to the CRT and will instead place the value in the Default-VDATA register on the VDATA pins. To prevent this "snow" on the display, the Display Processor can be programmed for the highest priority (after DRAM/VRAM refresh).

The Display Processor internally contains a FIFO which is used to buffer the bit-map data to be displayed. The FIFO consists of 32-double-words of 32 bits each. Each FIFO double-word contains the results of a 32-bit fetch from the bit-map memory. A double-word can therefore contain as many as 32 pixels, or as few as 1 pixel (such as at window borders).

Display Processor Register 2 (TripPt) controls when this FIFO is loaded. If the trip point is set at 16, the Display Processor waits until the FIFO is half empty (only 16 double-words left) before it requests a new block transfer to refill the FIFO. The block transfer request will not end until the FIFO is again full (although the block transfer may be interrupted by a higher priority request). If the trip point is set at 28, the Display Processor will begin requesting a new block transfer after only 4 FIFO double-words are emptied (28 left remaining). A low trip point generates fewer but longer block transfers and therefore the overall Display Processor bus efficiency is increased. However, a low trip point also requires that the bus latency be smaller. A low trip point means that there are less double-words left in the FIFO when the bus request is

made. If the FIFO drains completely before the bus has been granted, then the DefaultVDATA will be used from the current pixel through the end of the current scan line. The trip point may be programmed to 16, 20, 24, or 28 using the Display Processor LD_REG or LD_ALL commands.

The Display Processor also keeps busy during blank times. During Vertical Blank time it performs any command loaded into its Opcode Register. During Horizontal Blank time it loads a new Strip Descriptor if necessary and begins fetching the first pixels on the line. (Actually, the descriptor fetch begins as soon as the last pixel of the last line has been placed in the FIFO). If the Display Processor priority is not high enough to allow these fetches during blank time, then again part of the display can not be generated correctly and Field Color will be used. Two bits in the Display Processor Status Register can be used to determine if the Display Processor ever gets behind:

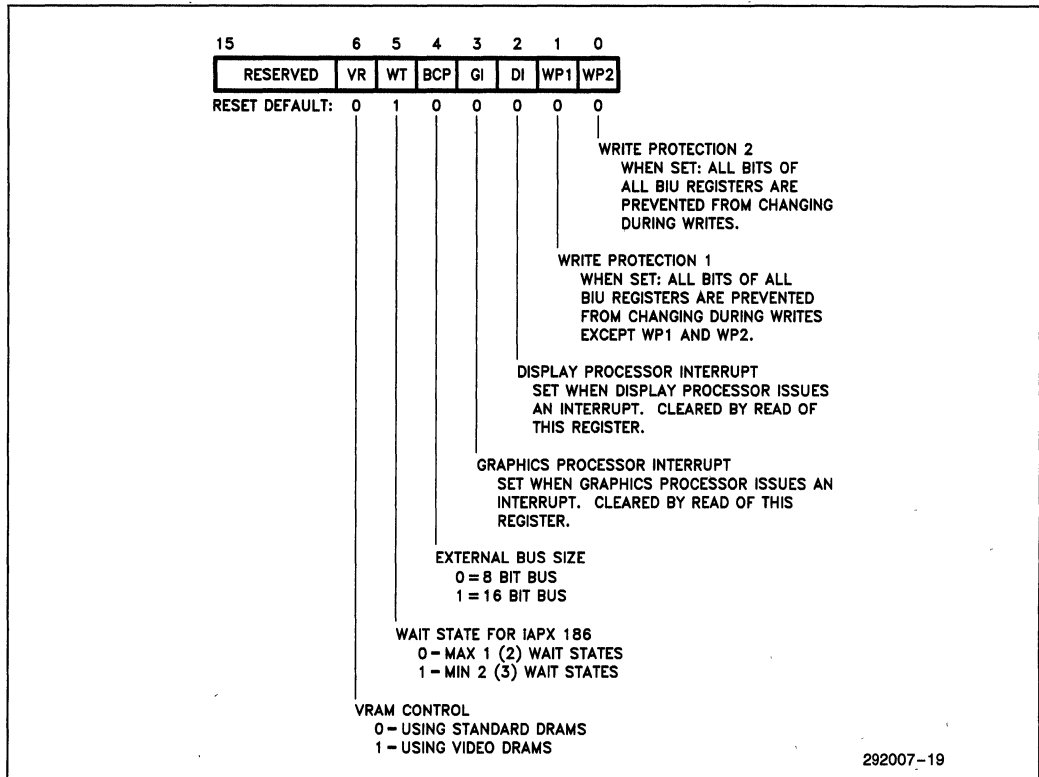
- bit-5 - DOV - Descriptor Overrun - set if strip descriptor fetch has ever not completed by the time horizontal blanking ends.
- bit-4 - FMT - FIFO Empty - set if the Display FIFO has ever completely drained.

Both bits are reset after reading the Status Register.

The setting of the External Priority register can greatly affect the performance of the external CPU when it performs an access to the 82786. Unless the External Priority is greater than the Graphics Processor, whenever the Graphics Processor is busy with a command stream that demands significant bus bandwidth, the CPU may have to wait a significant amount of time before it can complete an access to the 82786. The CPU waits for the 82786 in the middle of a bus access until the 82786 returns the READY signal. During this wait time, the CPU will not be able to process anything, including interrupts. Of course, if the application is very Graphics intensive and the CPU throughput is of lesser concern, then the Graphics Processor can be programmed with a higher priority.

Use the following priority values during your initial design. Once the system is working properly, you may wish to tweak the values for optimum performance. The optimum values are dependent on the CPU and video speeds as well as the CPU and graphics instruction mix and the window arrangement. In most cases, these registers will be initialized once and never changed. It may, however, be advantageous in some specialized applications to adjust these values when the application changes modes.

	FPL	SPL
Display Processor	6	6
Graphics Processor	2	2
External Processor	4	
Trip Point		24



One final BIU register contains a miscellany of bits.

After the BIU registers have been initialized, the WP1 and WP2 bits can be used to protect all of the BIU registers (82786 internal register offsets 00h - 0Fh) from being rewritten. This will prevent faulty software from going wild and placing the 82786 into an unwanted state. Once WP1 is set, the only way to change the BIU registers is to reset WP1 first. Once WP2 is set, there is no way for the software to modify the BIU registers until a 82786 hardware RESET is performed.

After the 82786 causes an interrupt, the GI and DI interrupt bits are used to allow the software to determine whether the Graphics or Display Processor caused the interrupt. It is possible that both of these bits may be set if both processors have caused an interrupt by the time the interrupt handler reads this register. In this case, both interrupts should be handled by the interrupt handler.

Although it is not absolutely necessary to allow the 82786 to interrupt the CPU, it is very desirable. Graphics Processor interrupts can inform the software when it has completed all the commands as well as to report

error conditions. Display Processor interrupts can inform the software when a new display field has begun. A new command can then be loaded into the Display Processor to be executed before the next display field. This facilitates operations such as smooth scrolling and blinking. The only hardware requirement to permit 82786 interrupts is that the 82786 INTR pin is tied to one of the interrupt controller inputs.

Although the 82786 always uses 16 bits, the 82786 can be used with both 8 and 16 bit processors. For an 8-bit CPU, separate transceivers are required for the low and high bytes to the 82786 (Figure 18). In both 8 and 16 bit modes, graphics memory may be accessed a byte at a time. Although the 82786 internal registers may be read a byte at a time, they all are considered to be 16 bits (even if some of the bits aren't used) and must always be written in 2-byte even-word pairs. In 16-bit mode, they must be written as a 16-bit word. In 8-bit mode, first the lower (even-address) byte is written and then the upper (odd-address) byte is written. With an 8-bit processor such as the 8088, both of the following assembly routines may be used to load the 16-bit BIUControl Register with AX.

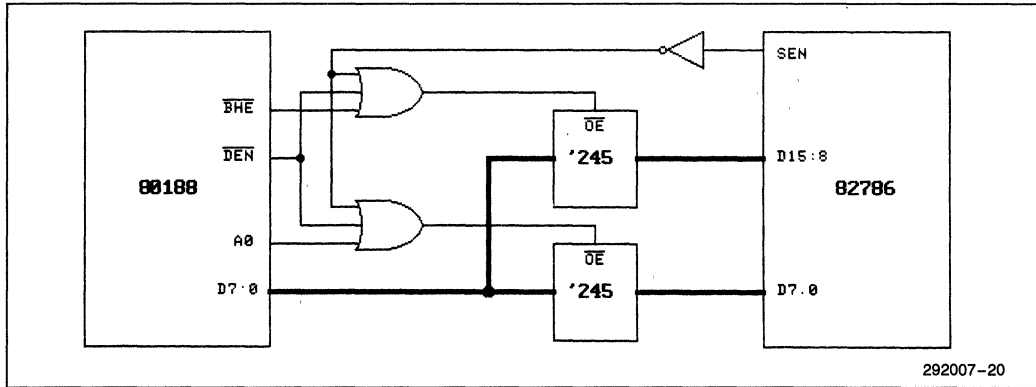


Figure 18. 8-Bit CPU Uses Two Data Transceivers to Connect to 82786

```

mov dx,BIUControl
out dx,al      ;write AL into low-byte of BIUControl
mov al,ah
inc dx
out dx,al      ;write AH into high-byte of BIUControl

or:

mov dx,BIUControl
out dx,ax      ;write AX into BIUControl word

```

In 8-bit mode, an even-byte write to a 82786 internal register does not change any of the 82786 internal registers, the data is simply saved until an odd-byte write to a 82786 internal register is performed. Then both the high and low bytes are written into that register. In effect, the even-byte address is ignored and an odd-byte write will write into the register both the odd-byte data and whatever even byte data was last written, into the register address specified by the odd-byte access. There is no limit to the amount of time allowed between the even-byte and corresponding odd-byte writes. An odd-byte write that is not preceded by an even-byte will be ignored.

The 82786 always comes up in 8-bit mode after RESET. This means that a 16-bit CPU should change the BCP bit to one. It must perform two byte-wide accesses to do this. The following initialization code can be used.

```

mov dx,BIUControl
mov al,30h
out dx,al      ;write 30h into low-byte of BIUControl

xor al,al
inc dx
out dx,al      ;write 00h into high-byte of BIUControl

mov dx,InternalRelocation
mov ax,03F8h
out dx,ax      ;write 03F8h into InternalRelocation word

```

The 82786 is first placed in 16-bit mode (using two 8-bit writes), then the 82786 internal registers are located at the desired address (which is done with a 16-bit write). Next, the DRAM/VRAM and priority registers should be initialized. Byte-wide writes into the 82786 internal registers can not be performed while BCP = 1.

All the 82786 master mode operations are 16 bits wide independent of the BCP bit. This means that system memory must be accessible 16-bits at a time if master mode is to be used. The WT bit is set to 1 on reset. The VR bit is reset to 0 at reset.

4.3 80286 Synchronous Interface

Since the 82786 has been optimized for the 80286, it is not surprising that the interface logic is very minimal. Figure 19 shows a 82786 connected synchronously to an 80286. Much of the logic, such as the 82288, chip-select, and ready, can be shared by the rest of the 80286 system.

This configuration allows both master and slave accesses. The data transceivers allow the 80286 to access the 82786 and graphics memory and the 82786 to access the 80286 system memory. They also provide the isolation required to allow the 80286 to access system memory while the 82786 accesses graphics memory simultaneously. The tristate buffer 74LS367 is used to pull the 80286 upper address lines, $\overline{\text{COD/INTA}}$, $\overline{\text{LOCK}}$ and $\overline{\text{PEACK}}$ to their proper states during master-mode. If any of these signals are not used by the rest of the system, they need not be driven by a tristate buffer.

If master mode is not required, $\overline{\text{MEN}}$ will stay low and three of the four gates driving the data transceivers can be eliminated. Also, the tristate buffer, which is only used in master-mode, may be eliminated. $\overline{\text{HREQ}}$ should be left open and the 82786 $\overline{\text{HLDA}}$ pin should be tied to ground so that the 82786 will never enter master mode.

Both the 80286 and the 82786 internally divide-by-two the CLK input and use both phases. For the 82786 to run correctly with the 80286, these phases must be correlated correctly. This can easily be done by observing the setup and hold times for rising RESET for both

chips (see 80286 data sheet specifications 6 and 7 and 82786 data sheet specifications C6 and C7). The 82284 chip will meet this requirement.

Depending on the CLK speed and the type of DRAM/VRAM used, the 82786 may have very stringent CLK duty cycle requirements (see Section 3.2). It may not be possible to use the internal oscillator of the 82284 chip but it may be possible to use an external oscillator to drive the 82284 external clock (EFI) pin.

Clock skew between the 80286 and the 82786 should be kept to a minimum so the chips should be placed as close together as possible.

When the 82786 bus is free, the circuit in Figure 19 permits CPU slave accesses using 2 wait-states for writes and 3 wait-states for reads. Using DRAMs/VRAM with slightly faster access times, the circuit in Figure 20 permits both read and write slave accesses using 2 wait-states. The 82284 SRDY input is used instead of ARDY. The 82786 SEN timing is such that a minimum of 2-wait states are always generated for writes but a minimum of 2 or 3 wait-states are used for reads depending on the use of SRDY or ARDY. Notice that with 2 wait-state reads, the SEN signal must be qualified with $\overline{\text{CS}}$ so that SEN does not extend into the cycle following the slave write. The most critical relationship to be satisfied in order for 2 wait-state writes is:

$$T_{\text{cac}} < T_{\text{c}} + T_{\text{ch}} - 15 - 45$$

For a 10 MHz 82786 the DRAM/VRAM column access time must be:

$$T_{\text{cac}} < 50 + 25 - 45 = 30 \text{ ns}$$

Note that x 1 DRAMs have two transceiver delays.

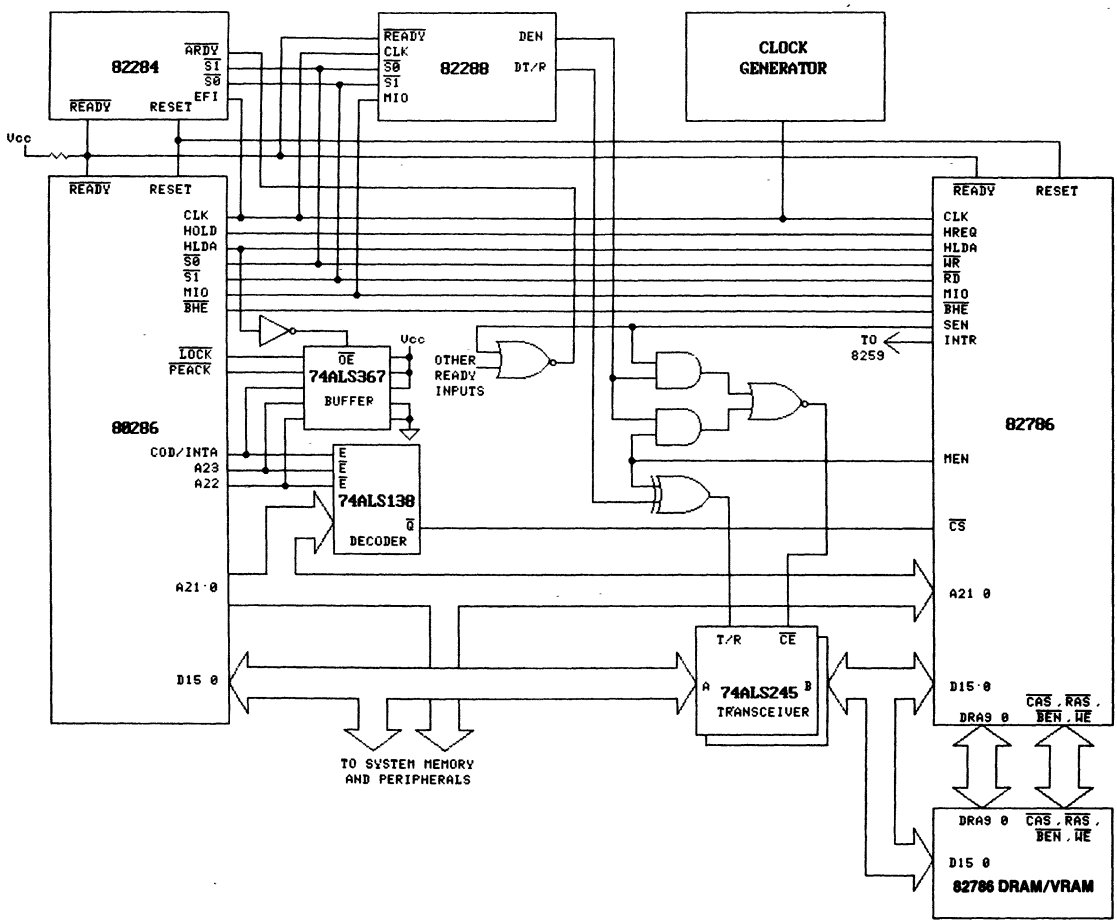
The critical timing calculations for slave mode are calculated as follows. The actual numbers calculated are for an 80286/82786 system running at 10 MHz.

chip-select-logic	=	path from 80286 address to 82786 \overline{CS} pin				
	<	$2 \times$ clock period	—	address valid	—	setup
	<	$2 \times 286.T1$	—	286.T13	—	82786.Ts1
	<	2×50 ns	—	60 ns	—	5 ns
	<	35 ns				
ready-logic	=	path from 82786 SEN to 82284 SRDY pin				
	<	clock period	—	SEN active	—	ARDY setup
(if ARDY is used as in Figure 19)	<	286.T1	—	82786.S18	—	82284.T13
	<	50 ns	—	25 ns	—	0 ns
	<	25 ns				
ready-logic	=	path from 82786 SEN to 82284 ARDY pin				
	<	clock period	—	SEN active	—	SRDY setup
(if SRDY is used as in Figure 20)	<	286.T1	—	82786.S18	—	82284.T11
	<	50 ns	—	25 ns	—	15 ns
	<	10 ns				
				from SEN active to read data valid		
read data valid \geq 82786.Ts22 + transceiver delay				from SEN active to write data valid		
write data valid \geq 82786.Ts20						

The master mode signals generated by the 82786 are all within the specification range guaranteed by the 80286. In other words, if the system memory is designed to function with the 80286, it will also be able to function with the 82786. The only signals that may not be within the range of the 80286 specifications are the data bus signals due to the transceiver delays. Care must be taken to ensure that the memory subsystems that the 82786 is to be able to access in master mode can meet these more stringent requirements:

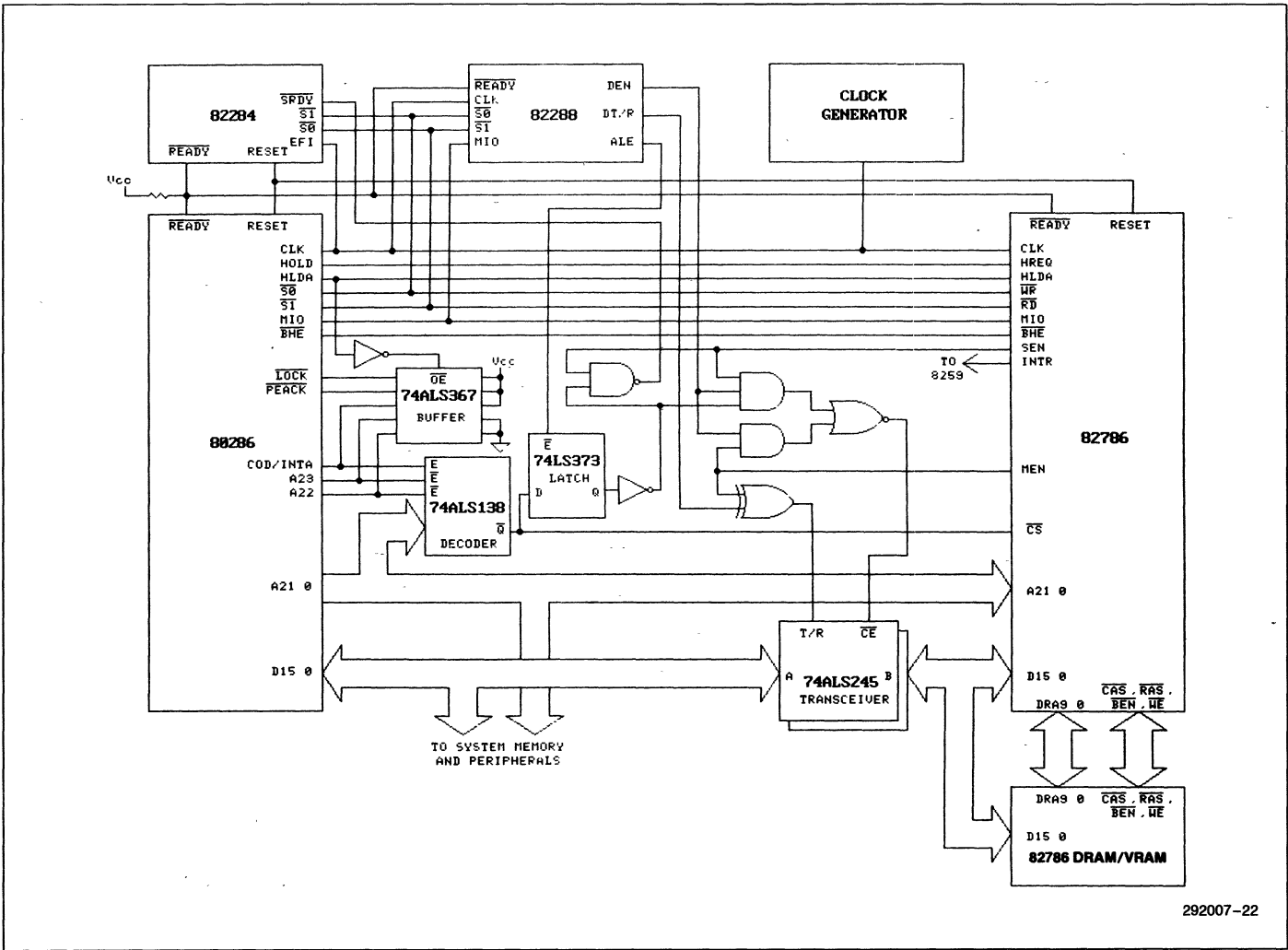
		data valid to falling clock after Tc phase 2		
read data setup	>	82786 read data setup	+	transceiver-delay
	>	82786.T8	+	data in to data out
	>	5 ns	+	Tprop
		data valid delay from falling clock after Ts phase 1		
write data valid	<	82786 write data valid	—	transceiver delay
	<	82786.T14	—	data in to data out
	<	40 ns	—	Tprop

The clock skew between the 80286 and the 82786 must be considered in all these calculations.



292007-21

Figure 19. 286/82786 Synchronous Master/Slave Interface Permits Minimum of 2 Wait-State Write, 3 Wait-State Read



292007-22

Figure 20. 286/82786 Synchronous Master/Slave Interface
Fast DRAMs Permit Minimum of 2 Wait-State Read/Write

4.4 80186 Synchronous Interface

The 82786 supports a synchronous status interface to the 80186. The 82786 and the 80186 must be driven with the same external clock (EFT). The $\overline{\text{Reset}}$ inputs to the 82786 must be generated from the $\overline{\text{RES}}$ of the 80186 by delaying it by one clock (input). This guarantees that the 82786 Clock phase 1 is coincident with 80186 CLKOUT low. A synchronous 80186 interface is selected if BHE is high and MIO is high prior to falling 82786 RESET.

Generally this configuration will be used with a minimum of 3 wait states for the 82786 slave read and write accesses. Therefore the WT bit in the 82786 BIU Control Register should be set. The 82786 slave accesses will then only be initiated when the 82786 CS is actually activated.

There is, however, a way to allow this interface to use a minimum of 2 wait states. (Set WT=0) Rather than wait for $\overline{\text{CS}}$ to go active the 82786 can be allowed to request a slave access as soon as the 80186 status lines go active. If the 82786 is not in the midst of another bus cycle and the CPU request is the highest priority, the bus will immediately be granted to the CPU and a bus cycle started. If the $\overline{\text{CS}}$ then goes active the 82786 can complete the access within 2 wait-states. If $\overline{\text{CS}}$ does not go active (because the 80186 is not accessing the 82786 but rather its own memory or I/O) then the 82786 will abort the bus cycle by running a dummy 82786 bus cycle.

If there is other RAM or ROM in the system besides the 82786 Graphics DRAM/VRAM that the 80186 often accesses, then this 2 wait-state will probably hinder rather than help performance. Every time the 80186 fetches from its own system memory (such as an opcode fetch or operand access), and the 82786 bus is idle, the 82786 will waste time running a dummy cycle. Fortunately, the busier the 82786 bus is, the less likely it will be free when the 80186 initiates a bus cycle, and therefore the less likely the 82786 will waste time running a dummy cycle.

4.5 Asynchronous Interface

An asynchronous interface can be used to interface the 82786 with nearly any CPU. The CPU clock and the 82786 clock are independent and may run at different speeds. If the 80286 is connected asynchronously with the 82786 and both processors are run at approximately the same clock frequency, then the minimum possible wait-states is one more than for the corresponding synchronous mode.

Figure 21 shows a slave-only 10 MHz 82786 interface to an 8 MHz 80186. At 10 MHz, the 82786 requires that the address becomes valid $S17 = 80$ ns after RD or $\overline{\text{WR}}$ falls and remains valid for $S16 = 130$ ns. Because the 80186 address disappears the same cycle RD and $\overline{\text{WR}}$ fall, the address must be latched. This latched address can be shared by the other components on the 80186 bus.

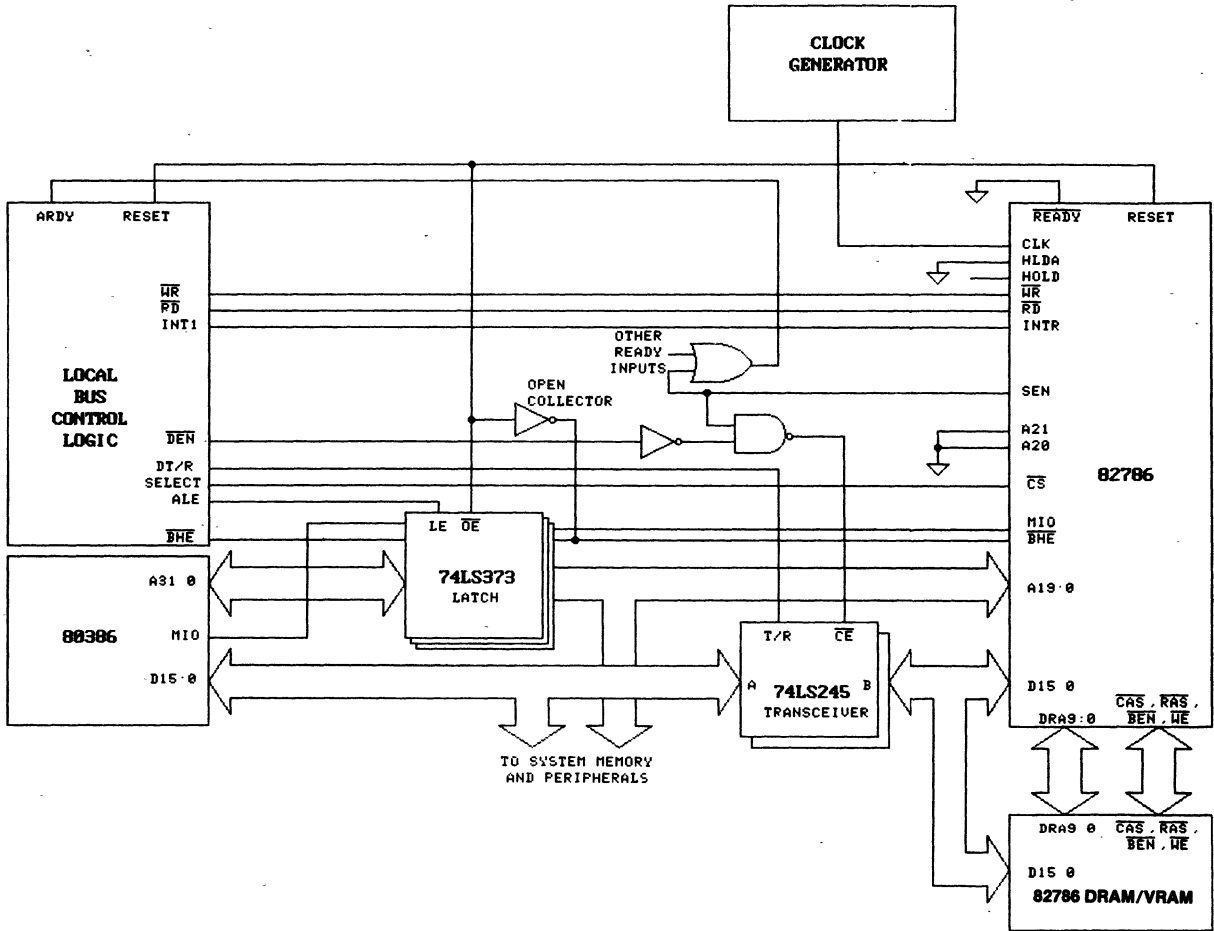
Due to the indeterminate phase relationship between the CPU and 82786 clocks, care must be taken to ensure the read/write data timings have enough slack. When the read data is sampled, and when the write data is removed is determined by the CPU's ARDY input. The 82786 SEN signal is used to generate the ready signal which is responsible to ensure that the data is indeed available. D-flip-flops can be used to delay the SEN signal to delay the CPU ready signal. For a 10 MHz 82786:

	from SEN active to read data valid
read data valid \geq	82786.Ts22 + Tprop
	from SEN active to write data valid
write data valid \geq	82786.Ts20

To initially place the 82786 into the asynchronous interface mode, the 82786 BHE pin must be low during the falling edge of RESET. To ensure this, the 74LS373 latch for BHE is tristated and an open-collector inverter pulls down $\overline{\text{BHE}}$ during RESET.

The 80386 processor can be interfaced to the 82786 either synchronously or asynchronously. For a synchronous interface, standard logic can be used around the 80386 to emulate a 80286 style bus for use with the interface described in Section 4.3. In this configuration the 82786 bus would run at half the clock rate of the 80386 (a 16 MHz 80386 would run with an 8 MHz 82786 bus). For an asynchronous interface, the standard local bus controller logic used by the 80386 to interface most peripherals can be used (Figure 22).

Although the actual bus transfers of a synchronous bus are faster than for an asynchronous bus, there are cases where an asynchronous interface provides the highest performance. For example, for a given display resolution, the Display Processor overhead of a 10 MHz 82786 is a lower percentage of the total bus throughput than for an 8 MHz 82786. If the 82786 is used with a 16 MHz 80386, then an asynchronous 10 MHz 82786 would have more bandwidth for the CPU and Graphics Processor than a synchronous 8 MHz 82786 and therefore CPU accesses, generally, will be completed faster with the asynchronous interface.



292007-B5

Figure 22. 80386/82786 Asynchronous Slave-Only Interface

4.6 Multiple 82786 Interface

For higher performance, it is possible to use several 82786 chips in the same system. Any of the above CPU/82786 interfaces can be used to attach multiple 82786s to one CPU in the system. Each 82786 will require its own separate DRAM/VRAM array.

The driving software for these multiple CPUs would most likely be sending nearly the same commands to all of the 82786s. Rather than forcing the software to write commands to each 82786 individually, it is possible to allow write commands to go to several or all the 82786s. One method of determining which 82786s should receive the write command would be to first write to an I/O port in which each bit corresponded to a different 82786. In Figure 23, the port bits set to 0 enable the corresponding 82786 for CPU writes. When a write to 82786 address-space occurs, all of the selected 82786s are chip-selected. The CPU will then wait for READY from all the selected 82786s before completing the bus cycle. In this manner, one, all, or any combination of 82786s can be written into at once.

Because it is impossible to read from several 82786s at once, a priority scheme is used on the I/O port to allow a read from only one of the selected 82786s. The circuit in Figure 23 only allows slave-accesses, the 82786s may not enter master-mode.

If master-mode operation of the multiple 82786s is desired, each 82786 must access the bus separately. A priority scheme is used to determine which 82786 is awarded the bus when the CPU issues HLDA. With only two possible 82786 masters, the random circuitry to hold one 82786 off the bus while the other is using it is straight-forward (Figure 24). With more 82786 masters, it is more feasible to use a state-machine (possibly implemented in PALs) to perform the arbiting.

5.0 VIDEO INTERFACE

The video interface connects the 82786 to the video display. The 82786 is optimized to drive CRT monitors but may also be used to drive other types of displays. Because CRTs provide an inexpensive method of generating moderate and high resolution, monochrome and color displays, this application note will concentrate on CRT interfaces. Section 5.10 briefly describes other display interfaces.

The video interface for a CRT is very dependent on the CRT requirements and the resolution and depth (bits/pixel) of the image desired. The 82786 can be programmed to directly generate all the CRT signals for up to 8 bits/pixel (256 color) displays at video rates up to 25 MHz. In addition, external hardware can be add-

ed to allow a color look-up table or to trade-off the number of bits/pixel for higher display resolutions, or to use VRAMs.

Some of the possible display configurations are shown below. The calculations assume a 60 Hz refresh rate. High resolution CRTs are often run at a slower rate, which permits the 82786 to generate significantly higher resolutions than those in the following table. All cases assume a CRT horizontal retrace time of 7 μ s, except the 512 \times 512 \times 8 (10 μ s) and 640 \times 400 \times 8 (13 μ s) cases.

With Standard DRAMs

	Non-Interlaced	Interlaced
8 Bits/Pixel (256 colors)	512 \times 512 640 \times 400 640 \times 480	900 \times 675
4 Bits/Pixel (16 colors)	870 \times 650	1290 \times 968
2 Bits/Pixel (4 colors)	1144 \times 860	1740 \times 1302
1 Bit/Pixel (monochrome)	11472 \times 1104	2288 \times 1716

Multiple 82786s can be used together to generate even higher resolutions with more colors. For example, two 82786s allow a non-interlaced 1144 \times 860 sixteen color display.

With Video DRAMs*

	Non-Interlaced
8 Bits/Pixel (256 colors)	1024 \times 1024
4 Bits/Pixel (16 colors)	2048 \times 1024
2 Bits/Pixel (4 colors)	2048 \times 2048
1 Bit/Pixel (monochrome)	4096 \times 2048

*For 64K by 4 - with 256K by 4 higher resolutions are supported

5.1 Various CRT Interfaces

CRT monitors use a wide variety of interfaces. Some use TTL-levels on all inputs, others require analog inputs. Some use separate color inputs (red, green and blue) and separate horizontal and vertical sync while others require that some or all of these signals be combined into composite signals. This application note will concentrate on the generation of separate color and horizontal and vertical sync signals. Standard techniques can be used to convert these separate signals into composite signals to meet the requirements of other displays.

The video clock (VCLK) required by the 82786 may be generated by a simple oscillator with TTL-outputs. Alternatively, the VCLK can be tied to the bus clock (CLK) (or any other available clock) if they are to run at the same speed.

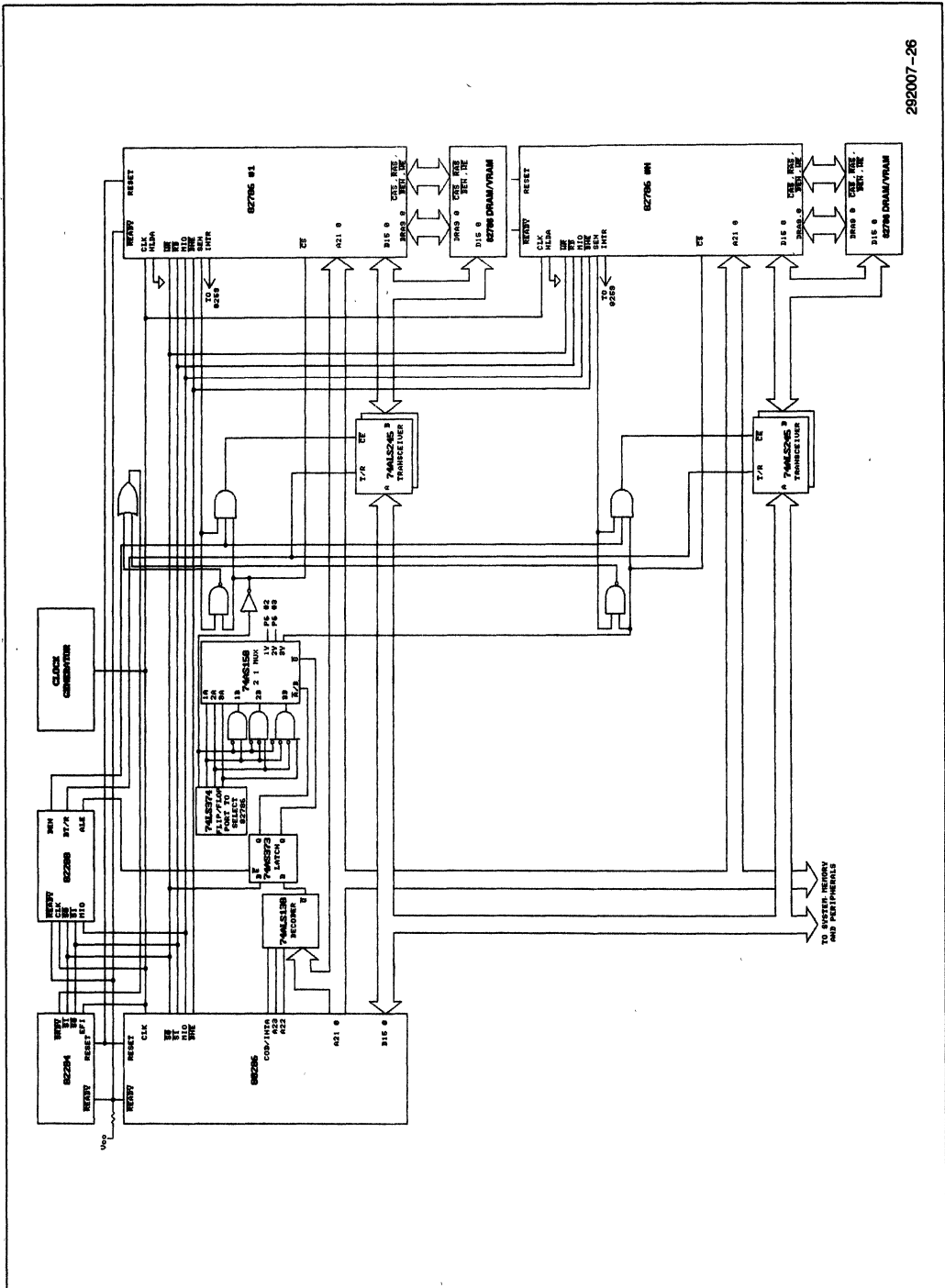


Figure 23. This Configuration Allows Several 82786s to be Written by 80286 Simultaneously—Only Slave Accesses are Supported

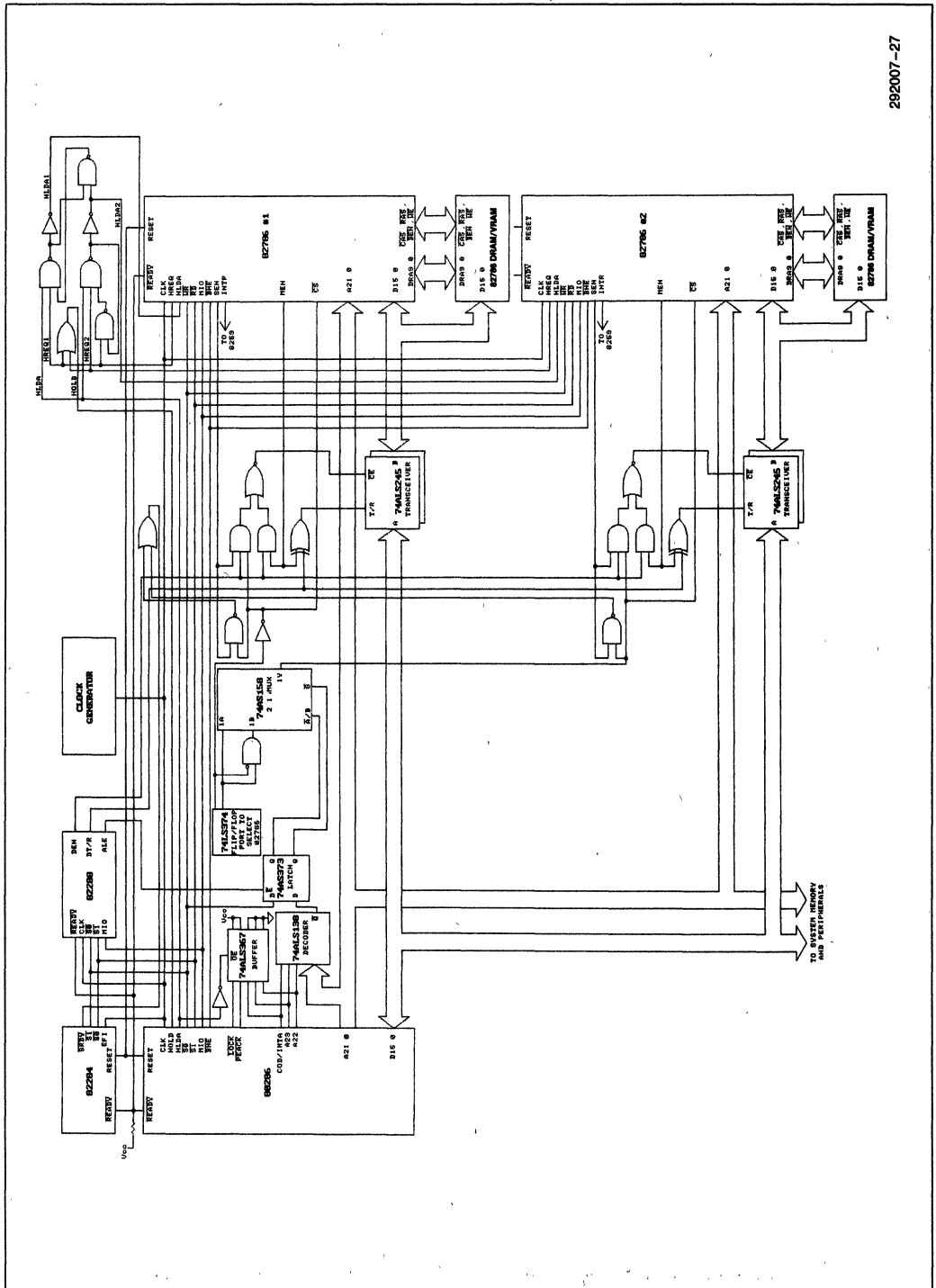


Figure 24. Two 82786s Connected to 80286, Permits Slave and Master Accesses

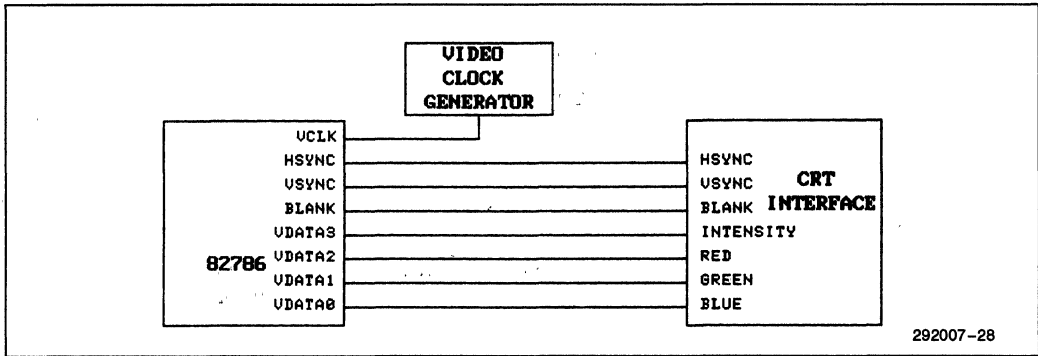


Figure 25. 82786 Can Directly Drive TTL-Input CRT Interface

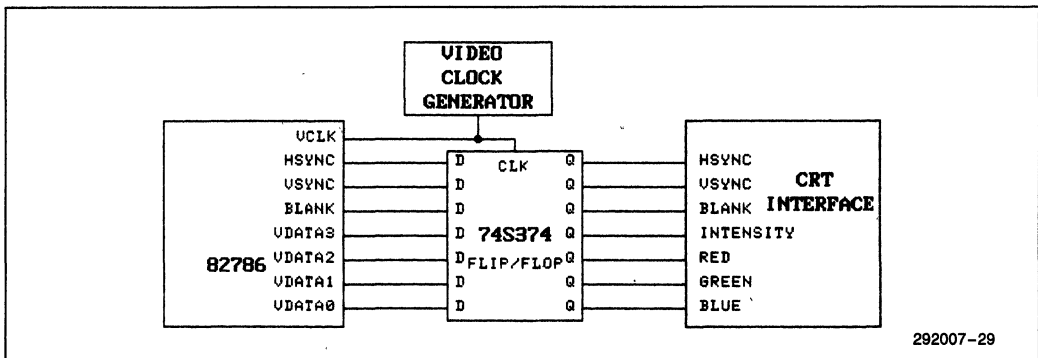


Figure 26. Buffer Used to Drive TTL-Input CRT Interface

5.2 CRTs with TTL-level Inputs

The simplest interface is to CRTs that use TTL-level inputs. The 82786 can generate these signals directly (Figure 25). However, the drive requirements of the CRT and cabling may make it necessary to buffer the signals (Figure 26). The example monitor in both of these cases happens to use a CRT that uses four-bits of color information per pixel. This means that 16 different colors are available and the CRT can use the 82786 1, 2, and 4 bits/pixel modes but can not take advantage of the 8 bit/pixel (256 color) mode. A monochrome monitor with only one TTL-level input could be connected directly to VDATA0 and use the 82786 1 bit/pixel mode but it then can not take advantage of any of the higher bit/pixel modes.

5.3 CRTs with Analog Inputs

Taking advantage of the 8 bit/pixel mode of the 82786 usually requires using a CRT with analog inputs. Signals for color CRTs with three separate analog video inputs, (red, green, and blue) can be generated using three digital-to-analog converters (Figure 27). Often these digital-to-analog converters can be constructed using simple resistor ladders (Figure 28). With 8 bits/pixel, usually three bits are used to select red, three for green and two for blue. This is because our eyes are much more sensitive to variations of red and green than of blue. These configurations can take advantage of all the 82786 modes; 1, 2, 4, and 8 bits/pixel.

The VDATA pins may be assigned to the three colors in any manner desired. In Figure 29 they are assigned so that a variety of colors are available for each mode (1, 2, 4, and 8 bits/pixel).

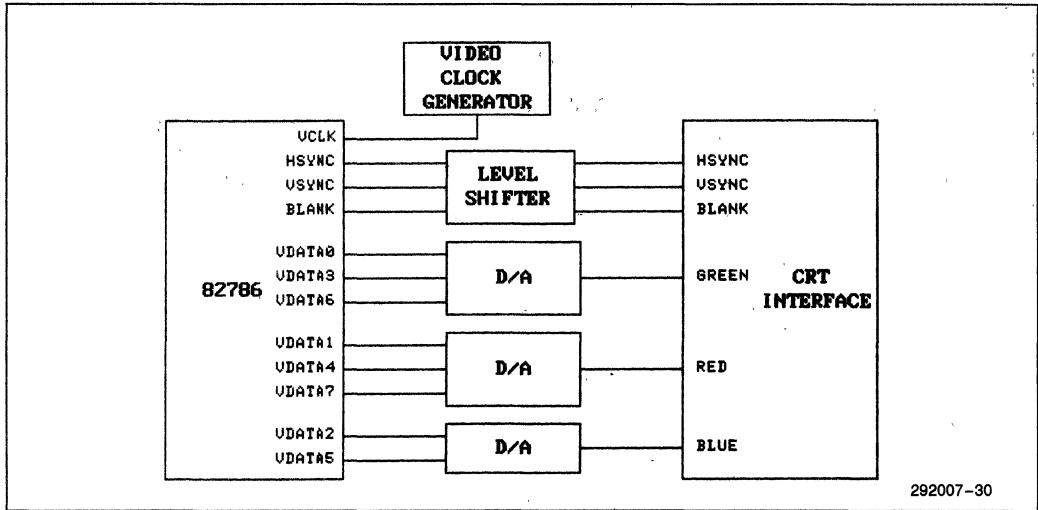


Figure 27. Analog CRT Interface Allows 256 Colors

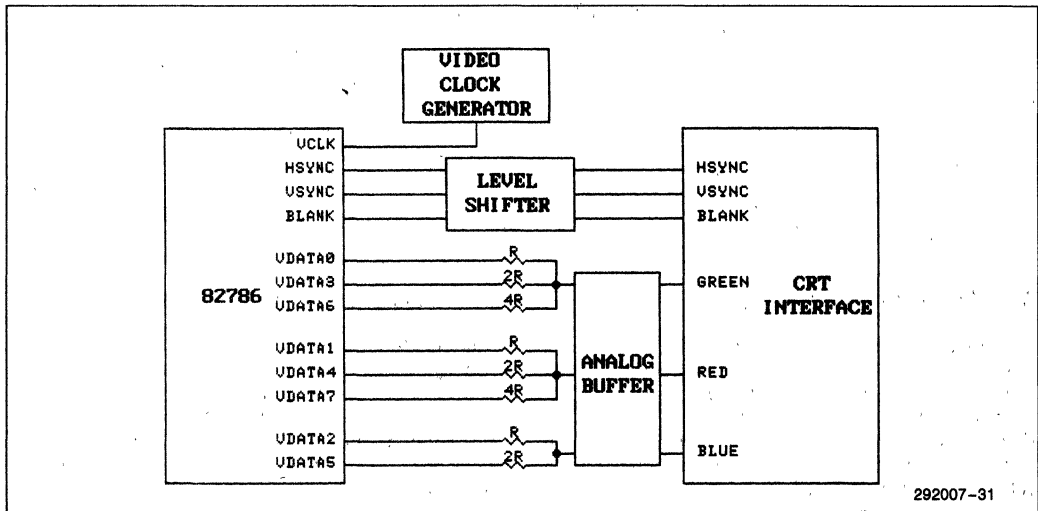


Figure 28. Resistor-Ladder Used for D/A

VDATA7	VDATA6	VDATA5	VDATA4	VDATA3	VDATA2	VDATA1	VDATA0
Red bit 0	Green bit 0	Blue bit 0	Red bit 1	Green bit 1	Blue bit 1	Red bit 2	Green bit 2

Figure 29. VDATA Pin Assignments

- The most-significant Green bit is connected to VDATA0 so that in the one bit/pixel mode this bit is controlled while the other bits are set to a constant level by the padding register internal to the Display Processor. If, for example, the padding bits are all set to zero, then a green and black image is shown in one bit/pixel windows.
- With two bits/pixel the most significant Green and Red bits are controlled while the rest are padded to a constant value. If, for example, the padding bits are set to zero then the colors black, green, red, and yellow are available in two bits/pixel windows.
- Four bit/pixel windows contain two Green bits and the most significant Red and Blue bits making 16 colors available.
- Eight bit/pixel windows allow control of all eight bits to make all 256 colors available.

colors. In this way an 8-bit/pixel bit-map can be used to control the 16-bit colors.

The host CPU is responsible for loading the 16-bit colors into the look-up table. To load a color into a specific location in the look-up table, the 82786 Display Processor can be programmed to output the 8-bit address on the 8 VDATA pins during the horizontal and vertical blank times or on RESET by setting the DefaultVDATA register. Then the CPU can load the color value into the 16-bit latch.

The circuitry in Figure 30 will then automatically write the 16-bit value into the look-up table during the next horizontal sync time. The CPU should generate the 74AS373 latch enable input so that the latch can be mapped into memory or I/O space and loaded by a CPU write. The register between the 82786 and the palette RAM is used to allow the use of a RAM with a slower access time. This register is not necessary if a faster RAM is used.

5.4 Using a Color Look-Up Table

Color Look-up Tables, also known as Video Palette RAMs, allow more colors to be available with a minimal of actual bits/pixel and thus a minimal amount of display memory is required for the bit-map. For example, in a system using 16 bits of color information, 65536 different colors are possible. In such a system it is rarely necessary to display all 65536 colors on the screen simultaneously. It may be feasible to support a maximum of 256 colors simultaneously, providing that these 256 selections can be any combination of the 65536 available colors. Color look-up tables permit such a cost-effective system.

The CPU program should wait until the color is loaded into the look-up table before loading the next color. One way to ensure this is to route the LookupLoading signal through a port which the CPU may poll. Sample assembly language code for this configuration follows this section. Another way is for the CPU program to delay a sufficient amount of time to ensure that HSYNC has occurred before writing the next value.

A block diagram of such a system is shown in Figure 30 and Figures 30a and 30b show actual circuits. The color look-up table can be loaded with up to 256 16-bit

Hybrid circuits can be used which combine the functions of the look-up table, analog-to-digital conversions, and voltage shift for composite sync signals into one package. Figure 30b shows such a configuration. This particular hybrid circuit internally contains a 16 × 12-bit look-up table, 4 bits for each red, green, and blue.

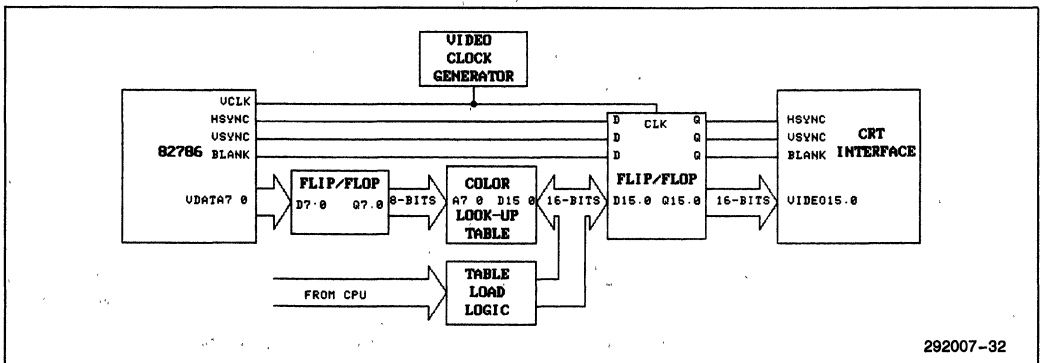


Figure 30. Block Diagram of Color Look-Up Table Used to Generate 16 Video Bits From 8

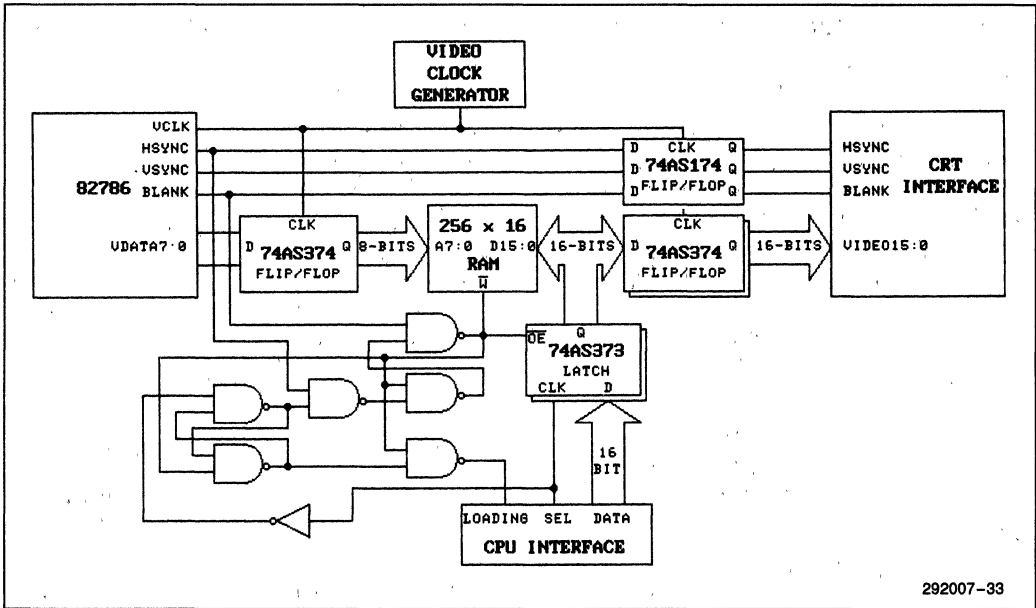


Figure 30a. Circuit for Color Look-Up Table Used to Generate 16 Video Bits From 8

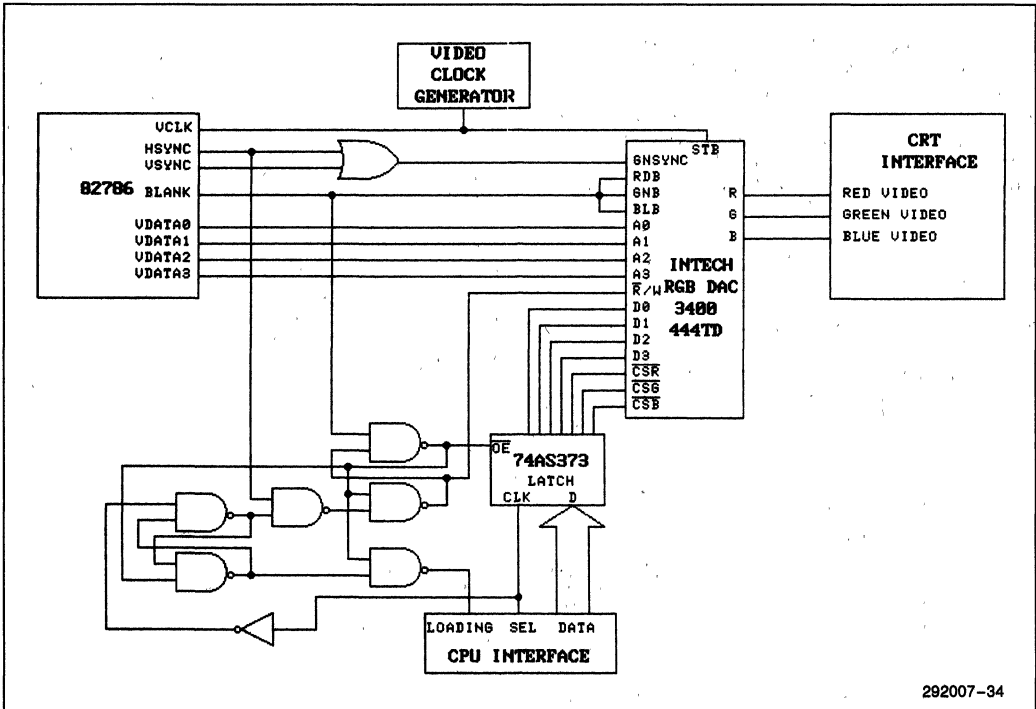


Figure 30b. Hybrid Color Look-Up Table and DAC Simplifies Interface

```

Wait:  in    al,StatusPort      ;read port
        test al,LookupLoadingBit ;test LookupLoading bit
        jnz  Wait              ;wait til last load completed
        mov  ax,EightBitAddr    ;get 8-bit value to load
        mov  DefaultVDATA,ax    ;make 82786 output during BLANK
        mov  ax,SixteenBitColor ;get 16-bit color
        out  LookupLatch,ax     ;write color into latch
    
```

The look-up table is loaded by first writing the location into the 82786 DefaultVDATA register. Then a 4-bit color value is loaded into the latch along with color-select information. Therefore, in one load it is possible to place this 4-bit color value into any combination of the red, green, and/or blue tables.

5.5 Using the Window Status Signals

A graphics system design may require that the video data bits for different windows be interpreted in different ways. For example, the attributes controlled by various video data bits may need to be changed between windows for different tasks or number of bits/pixel. For these reasons, two Window Status bits are available externally which reflect a value which may be individually programmed for each window. These two pins always reflect the window which the display is currently scanning. The software is responsible for placing the two bit values for each window in the Tile Descriptor list.

In addition, the cursor can be programmed with a value for the window status bits which can be programmed to override the status bits from the windows for the portion of the display where the cursor resides.

The Window Status bits are multiplexed onto the HSYNC and VSYNC pins. Since they are only applicable during the visible display time, and since HSYNC and VSYNC are only applicable during the non-visible display time, BLANK can be used to de-multiplex these pins (Figure 31).

A mode bit (bit 4 of CRTMode) in the Display Processor enables the Window Status bits so they become multiplexed onto the HSYNC and VSYNC signals. This bit must be set when the Window Status signals

are used. In systems where the Window Status bits are not needed, this bit can be reset so that the HSYNC and VSYNC pins remain low during the visible display. This allows simpler systems to use HSYNC and VSYNC directly eliminating the need to AND these pins with BLANK.

As an example, suppose the interpretation of the video data bits by a color look-up table was to be different for different windows. Possibly four different look-up tables are required for four different types of 8 bit/pixel windows. A large look-up table (1024 words) could be divided into four areas, one for each of the window interpretations. Then the Window Status bits could be used to select the area of the look-up table to be used for each specific window. Essentially four look-up tables would be available, one for each of four different types of windows. Figure 32 illustrates such a system.

The system also requires circuitry to load the look-up table such as that in the previous section. Note that the look-up table's Window Status inputs must be generated directly from the CPU when the RAM is to be loaded since they can not be programmed in specific states during the blank time as the VDATA pins can.

Another use of the Window Status bits is to allow 1, 2, 4, and 8 bit/pixel windows to each use a different look-up table along with a fifth look-up table for the cursor. A 1024 word look-up table above could be split up into four areas as above. Two of the areas can be used for two separate 8 bit/pixel look-up table and the other two shared by the 1, 2, and 4 bit/pixel windows for two separate look-up table for each of 1, 2, and 4 bits/pixel (Figure 33). The padding bits can be used to sub-divide this second area into separate tables for 1, 2 and 4 bit/pixel windows. Finally, this same table could also be used for twelve look-up tables, four each for 1, 2, and 4 bit/pixel windows.

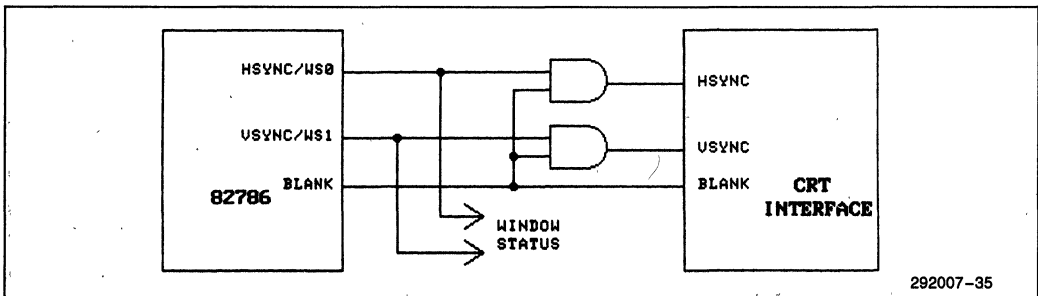


Figure 31. Using Blank to De-Multiplex Window Status

292007-35

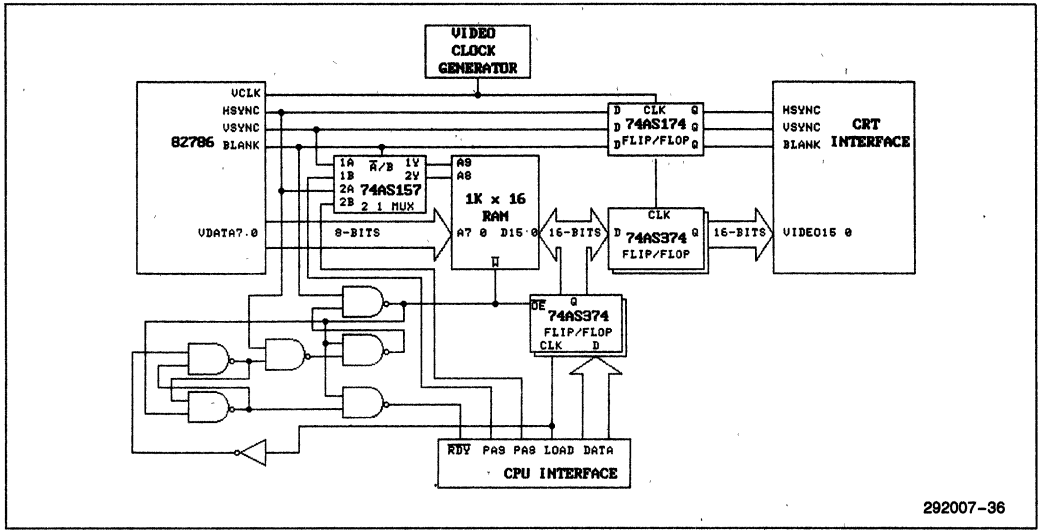


Figure 32. Four Color Look-Up Tables—Selectable by Window Status Outputs

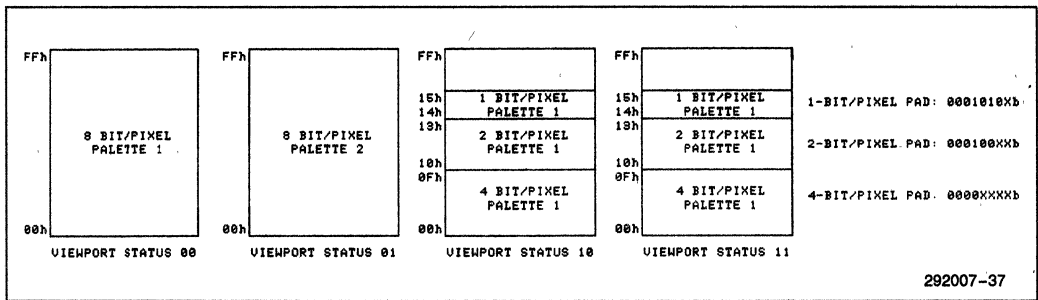


Figure 33. Window Status and Padding Bits Allow Two Separate Look-Up Tables for Each of 1, 2, 4, and 8 Bit/Pixels

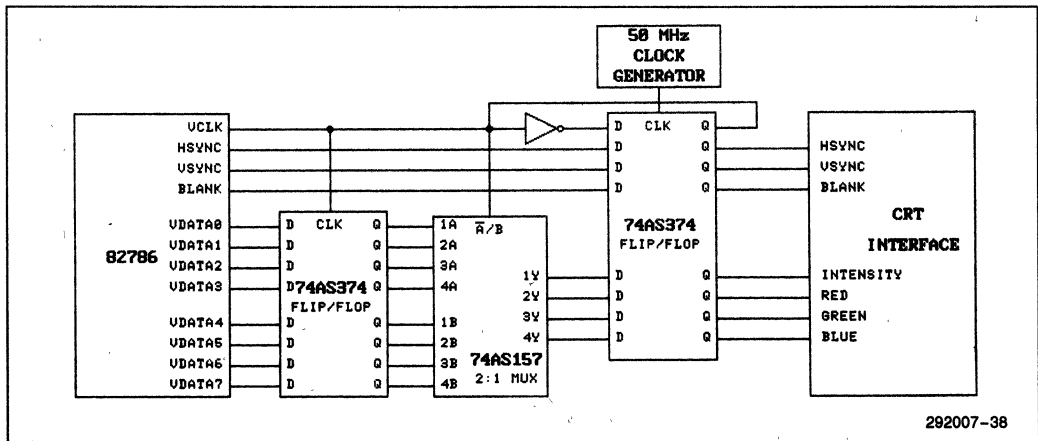


Figure 34. External Multiplexer Allows Up to 50 MHz Video with 4 Bits/Pixel

5.6 Higher Resolutions with Standard DRAMs

The Video Clock rate on the 82786 can be a maximum of 25 MHz. For a non-interlaced display refreshed 60 times per second this limits the resolution to 512×512 or 640×400 or equivalent displays. 640×480 can also be achieved using a CRT with fast horizontal retrace. Still, some graphics system designs may require more detailed displays and therefore more resolution. It may very well be cost-effective to trade-off the number of bits/pixel for higher resolution. This is especially true in the case of monochrome displays where 256 grey-shades are not required but high resolution is.

The 82786 allows this trade-off to be made very effectively. Figure 34 shows how a video data rate of up to 50 MHz may be obtained with 4 bits/pixel (16 colors). The 82786 is used to output 8 bits of video data at a 25 MHz rate. The external multiplexer switches between the low 4 bits and the high 4 bits at a rate of 50 MHz. The register before the multiplexer is used to ensure that enough set-up time is provided for the multiplexer. The register after the multiplexer ensures that the video data out has smooth transitions. The circuit uses an inverter and one register stage to divide the 50 MHz clock by 2 to create the 25 MHz video clock for the 82786. Instead of the 74S157 multiplexer, a 74AS298 chip could be used which contains the multiplexer and the register in the same package.

The software has a minimum number of changes. The Graphics Processor is programmed identically and manipulates the bit-maps in the conventional manner (although it does not make sense to use 8 bits/pixel bit-maps since they cannot be displayed). The display processor programming is slightly different. The Accelerated Video control bits (CRTMode bits 1,0) are set for High Speed video (01). The HSynStp, HFldStrt, HFldStp, and LineLen registers are programmed for half the number of dot clocks (because the 82786 VCLK is half the speed of the pixel dot clock).

The Strip and Tile Descriptors list also change only slightly. Windows are programmed for same number of bits/pixel and FetchCount as they would be for non-accelerated modes. However, windows may only be positioned horizontally to start on even pixel boundaries. That is, they may only start at every-other pixel, not at any pixel as permitted with non-accelerated modes. This is because both an even and odd pixels are output on the VData pins simultaneously and it is not possible to mix windows during a single VCLK. The only valid values for the start/stop bits are listed in the following table. Notice that the accelerated modes do not permit all possible bit-map depths because fewer than 8 bits/pixel are available to the display.

Vertically, the windows may still be positioned at any pixel. The programming of the one pixel horizontal and vertical borders also does not change.

High-Speed video mode also requires that the Field windows are programmed with half the number of actual pixels for the pixel count (BPP/Start/Stopbit) register which again restricts horizontal positioning to a two pixel resolution.

The horizontal cursor position is programmed as half the actual value so the positioning is also restricted to a two pixel resolution. Vertically, the cursor is programmed as normal. Since the cursor is only a 1 bit/pixel region, every other horizontal pixel reflects only the cursor padding value so although simple cursor patterns are possible, arbitrary shapes are not possible with the box cursor. For this reason, the programmer may wish to create the cursor in software when using these high-resolution modes rather than use the 82786 hardware cursor. The cross-hair style cursor works well in accelerated mode, although the horizontal and vertical lines become two pixels wide and horizontal positioning is also limited to two pixels.

Bit-Map Depth	Acceleration			
	None (25 MHz)	High-Speed (50 MHz)	Very-High-Speed (100 MHz)	Super-High-Speed (200 MHz)
1 bit/pixel:	any (0-15)	even numbers	0,4,8,12	0,8
2 bit/pixel:	even numbers	0,4,8,12	0,8	—
4 bit/pixel:	0,4,8,12	0,8	—	—
8 bit/pixel:	0,8	—	—	—

It is also possible to use external hardware to create the cursor. One method is to program the cursor as invisible (transparent and all background) and use the cursor's window status signals to activate the external hardware.

The horizontal zoom capability is also affected. Rather than replicating each individual pixel, pairs of pixels are replicated. Vertical zoom works as normal.

Figure 35 shows a configuration for video data rates of up to 100 MHz with 2 bits/pixel. A shift-register is used to multiplex the 8 video bits from the 82786 into 2-bit streams. A 74AS74 flip/flop is used to divide the 100 MHz clock by four. Every fourth clock the 82786 VCLK is raised and the shift registers are loaded with the previous 82786 VDATA. The video data is delayed two cycles by this circuit while the sync and blank are delayed only one. This should not be a problem if the 82786 is programmed to generate the correct sync. The 82786 is limited to positioning the sync transitions at multiples of four pixels. If more accurate positioning is required, extra flip/flops can be used to delay sync for more cycles.

The timing in Figure 35 is very tight and the circuit may not operate at 100 MHz over all operating temperatures. The limiting speed path is the 74F195 shift-reg-

ister parallel-load time (delay from clock to outputs valid) which must meet the set-up time of the 74AS374.

Figure 36 shows a configuration for video data rates of up to 200 MHz with 1 bit/pixel. Unfortunately, there is no TTL-logic available today which will run at the speeds required for 200 MHz. Therefore ECL or some other high-speed logic must be used to generate video at these high rates. Figure 36 converts the video data signals from the 82786 from TTL to ECL levels and then uses ECL shift-registers to generate the 200 MHz signal.

The software for the configurations in Figures 35 and 36 requires changes similar to the Figure 34 case. The window start/stop bits are programmed restricted as shown in the table above. The pixel count for Field regions is also one-fourth or one-eighth the actual size. Horizontal positioning is also restricted to four and eight pixels for the 100 MHz and 200 MHz rates respectively. The Accelerated Video control bits must also be programmed for these configurations.

After the video signals are accelerated to these higher speeds, color look-up tables and analog-to-digital converters may be used. The circuits in the previous sections must be adapted for these higher speeds.

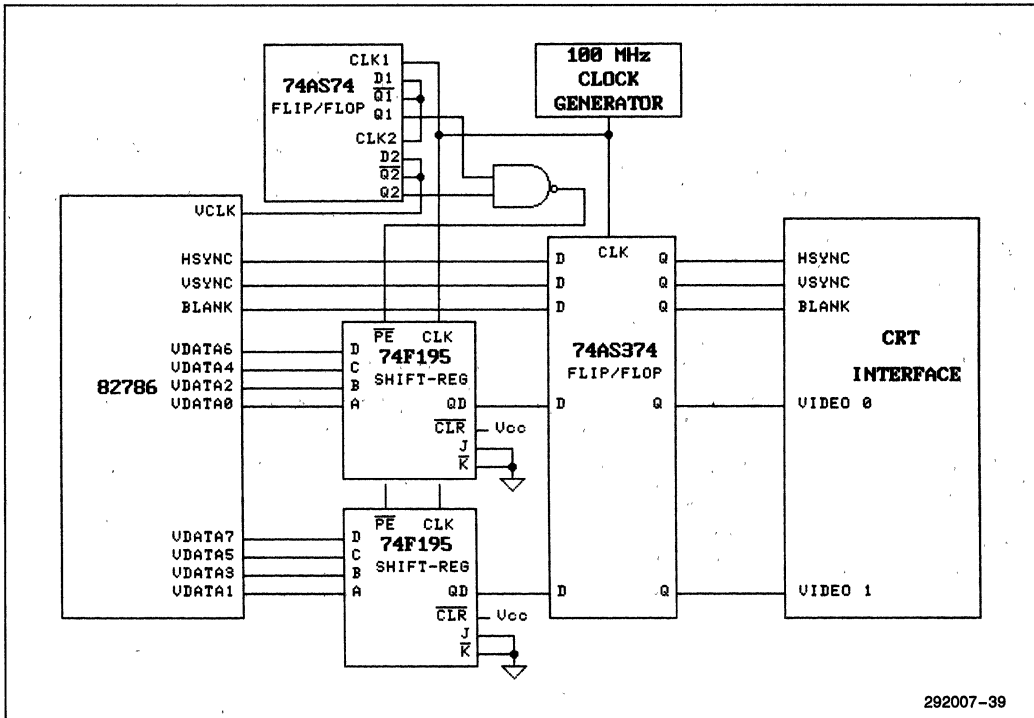
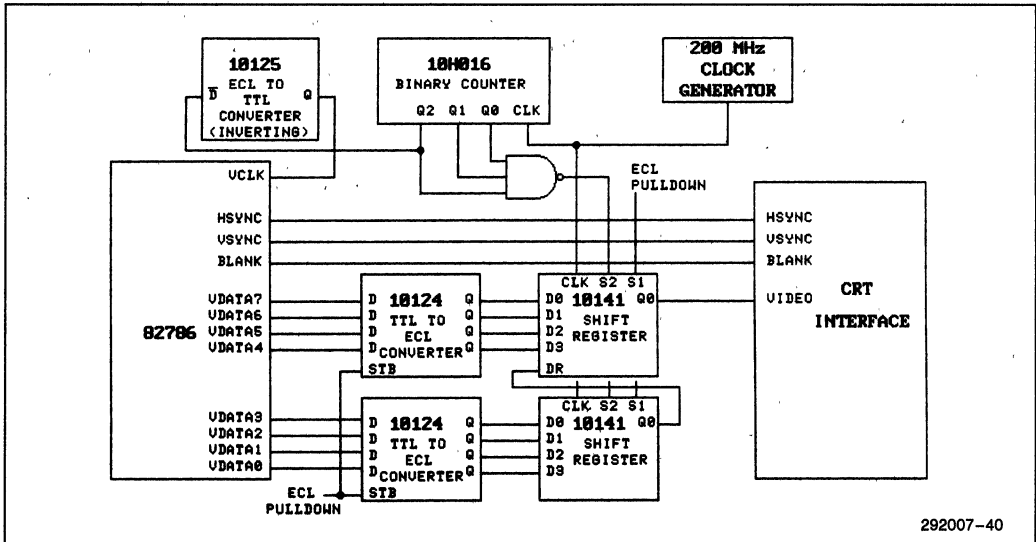
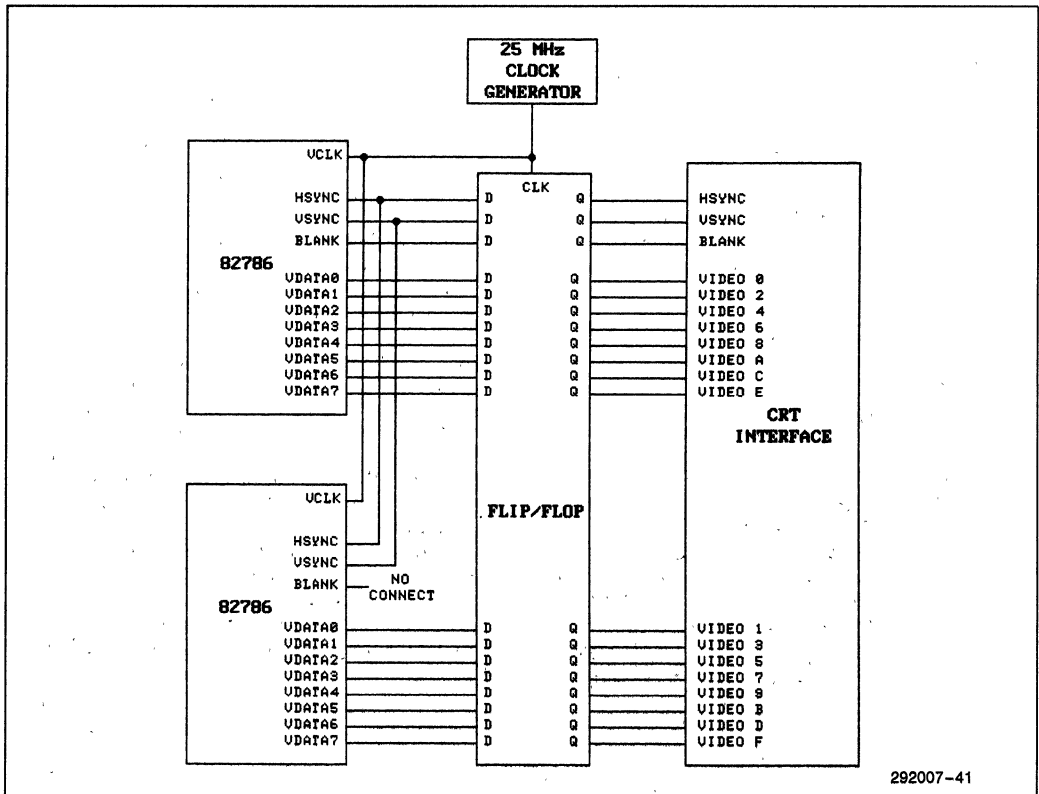


Figure 35. External Shift-Register Allows Up to 100 MHz Video with 2 Bits/Pixel



292007-40

Figure 36. External ECL Shift-Register Allows Up to 200 MHz Video with 1 Bit/Pixel



292007-41

Figure 37. Two 82786s Can Generate 25 MHz Video with 16 Bits/Pixel

5.7 Multiple 82786s

If more colors or resolution are required than possible with one 82786 at a given resolution, several 82786s can be used together to generate the necessary bits/pixel. Figure 37 shows two 82786s used together to generate 16 bits/pixel at a 25 MHz video rate. This configuration would allow a 640×480 display with 65536 colors.

Both 82786s' video must be kept in sync. To allow this, one 82786 is programmed as normal to generate the master video horizontal and vertical sync. The second 82786 is programmed for slave video sync with the Slave CRT control bit in the CRTMode Register (Display Processor register 5-bit 3 set). The HSYNC and VSYNC lines of the slave 82786 then become inputs and are driven by the HSYNC and VSYNC output lines of the master 82786. If the window status signals are used, the master's HSYNC and VSYNC signals should be qualified with the BLANK signal (similar to Figure 31) to correctly drive the slave 82786 HSYNC and VSYNC inputs. Window status signals are only available from the master 82786 since the slave uses these pins as inputs.

Both 82786s should have six of their eight video timing registers (HSynStp, HFldStprt, HFldStp, LineLen, VSynStp, VFldStp, VFldStprt, FramLen) programmed identically; HFldStprt and HFldStp should be programmed to be 2 greater than the master. (These parameters are calculated in Section 5.11.)

The slave 82786 will then automatically sync itself up to the master 82786 by waiting for its HSYNC input to fall before each scan line and waiting for its VSYNC input to fall before beginning a new display field. If a non-interlaced display is used, the two 82786s will always be in sync.

If an interlaced display is used, care must be taken to ensure both 82786s start on the same field. The easiest way to ensure they lock in sync correctly is to ensure they start scanning the display simultaneously. First set up the slave 82786 CRTMode and video timing registers with a LD_ALL command. The slave 82786 will then be ready to begin scanning the display but will wait until HSYNC and VSYNC fall. HSYNC and VSYNC will be floating high because they are tristated by all the 82786s. Then the master 82786 can be set up with a LD_ALL command to program its CRTMode and video timing registers. Once the master starts scanning, the HSYNC and VSYNC signals will be driven by the master and all 82786s will begin on the even interlaced field.

To create a 16 bit/pixel bit-map, both 82786 Graphics Processors should be programmed for 8-bit/pixel bit-maps of identical size. To draw in both bit-maps, a graphics command block (GCMB) can be created for both 82786s. These GCMBs are generally identical for both 82786s except for the color values for the Def_Color and the mask value for the Def_Logical_Op commands. To display 16 bit/pixel bit-maps, both 82786s should be given an identical strip descriptor list for each to display 8 bits of each 16 bit pixel.

Similarly, 8-bit/pixel bit-maps could be created by splitting the bit-maps between both 82786s having each 82786 responsible for 4 of the 8 bits/pixel. This would split the work between the two 82786s so that the BitBlit and Scan_Line fill graphic commands will execute twice as fast. Also, because the Display Processor bus overhead is split between the 82786s, there will be less bus contention so all other drawing commands will be faster.

Alternatively, 8-bit/pixel bit-maps could be generated by only one of the 82786s. This would minimize the overhead between the host CPU and the 82786 since the CPU must communicate with only one 82786.

The method in which the 16 video data bits are mapped into colors for the display interface will determine which of the two above methods will be used for bit-maps of 8 bits/pixel or less. If the mapping is flexible enough, it may be feasible to create any bit-map depth. For example, 9 bits/pixel bit-maps could potentially be created using one 82786 for 8 bits and the other for only 1 bit of each pixel.

The displays discussed in the previous section obtained high resolutions at the expense of bits/pixel. Several 82786s can be combined to provide more bits/pixel at these high resolutions.

Figure 38 shows a configuration that uses two 82786s to create a 4-bit/pixel display at a video rate of 100 MHz. This configuration would support a 1144×860 sixteen-color non-interlaced 60 Hz display. Each 82786 is required to generate 2 bits of each 4-bit/pixel. Therefore, both 82786s draw and maintain half of the bit-map in their own graphics memory, 4-bit/pixel windows are divided into two 2-bit/pixel bit-maps, one generated by each 82786. The Graphics Processors are programmed as normal for 2-bit/pixel bit-maps. The Display Processors are programmed the way mentioned in the previous section. Each window is programmed with one-fourth the horizontal positioning resolution.

5.8 Video RAM Interface

The 82786 can use dual-port video DRAMs (VRAMs) to generate the video data stream. The VR bit in the BIU Control Register must be set to 1 to enable the mode. In this mode the first tile in each strip generates VRAM cycles; subsequent tiles in the strip generate DRAM cycles. There is no limit on the number of strips. The pixel data for every scan line in the entire display must be contained in a single row in memory (256 words for non-interleaved memory and 512 words for interleaved memories). The strip descriptors for each tile are set up to indicate only 1 pixel. The address specified for this pixel corresponds to the first display pixel.

During the horizontal retrace period, the 82786 transfers the contents of the memory row containing the first pixel into the VRAM shift register. The VRAM shift clock is gated with a BLANK signal. During the active display time, the shift clock is active and periodically clocks out the video data. External multiplexers must be used to convert the 16 bit (32 interleaved) data stream into a serial stream depending upon the bits per pixel needed (Figure 9).

In this mode, pixel depth is fixed by external hardware and all Display Processor registers referring to video data fetch should be programmed to zero.

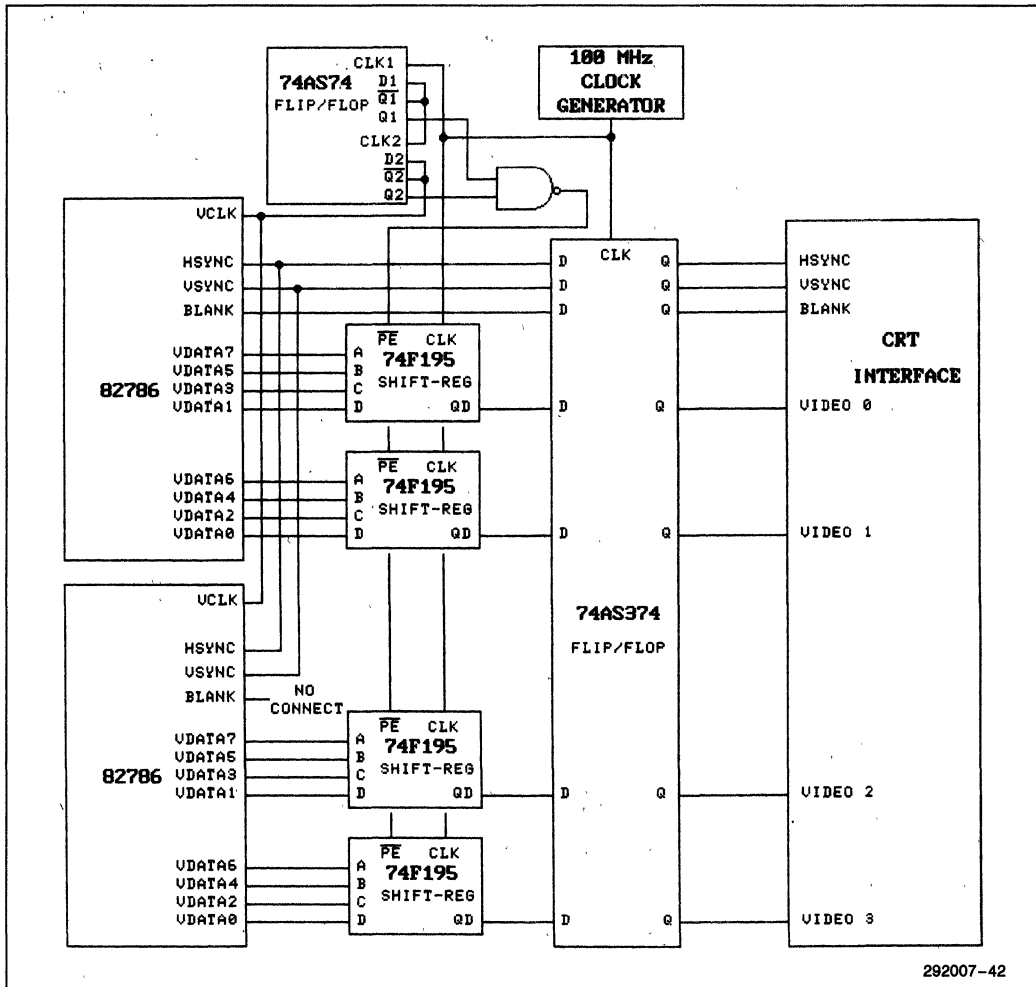


Figure 38. Two 82786s Can Generate 100 MHz Video with 4 Bits/Pixel

5.9 External Character ROM

Few 82786 applications will require, or even benefit from, the use of an external character ROM.

The 82786 Graphics Processor can very rapidly draw characters. It can fill an 80x25 character screen with highly detailed 16x16 characters in less than one tenth of a second.

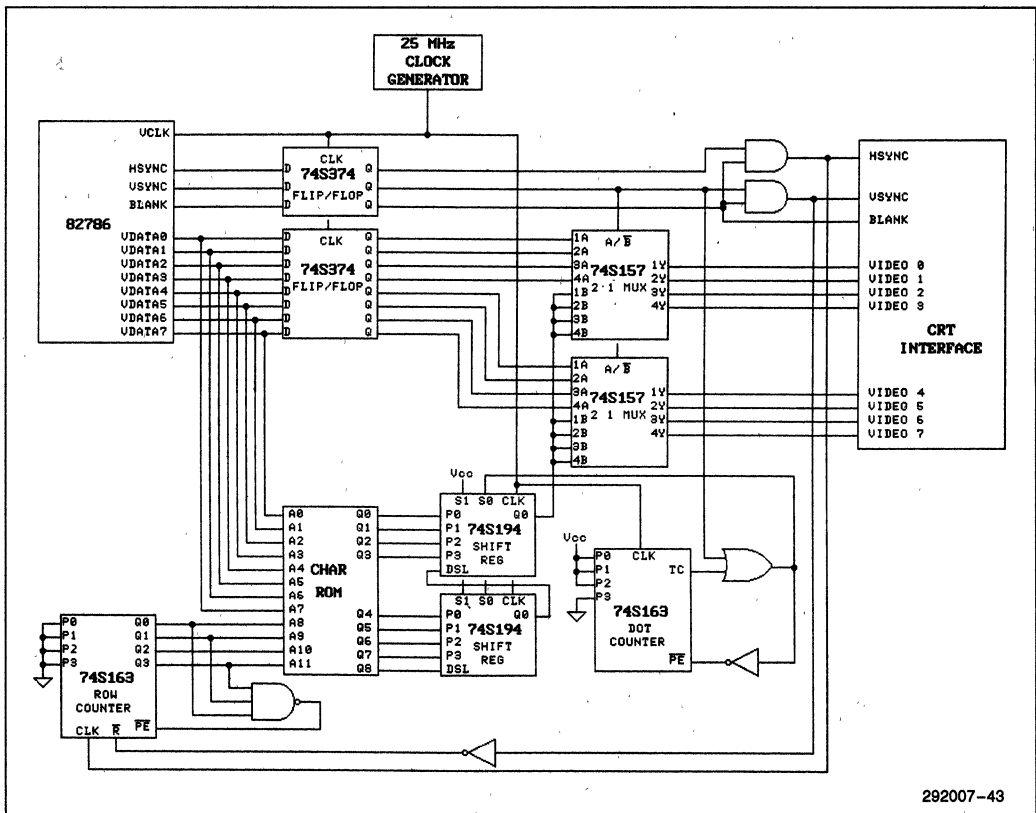
The Graphics Processor is also very flexible in the way it draws characters. Characters may be:

- formed from an unlimited number of character fonts
- placed at any pixel on the screen
- rotated in 4 directions with 4 paths
- combined with graphics
- drawn in any color
- have transparent or opaque background

A character ROM display forces characters from a pre-defined font to be restricted to character-cell positions on the screen with few, if any, of the above flexibilities.

For downward compatibility reasons, however, it may be necessary to provide the character ROM function in a 82786 system. Figure 39 illustrates a system capable of displaying both character ROM text and 82786 graphics. A multiplexer is used to switch between the character ROM output and the direct 82786 output. One of the window status bits is used to switch the multiplexer so both the character ROM and the 82786 graphics windows can be shown simultaneously on the same screen. It is important to delay the direct 82786 VDATA and window status signal the same number of clocks as the character-cell video so that all signals get to the multiplexer on the same clock. The extra D-flip/flops before the multiplexer are used to perform the needed delay.

The character ROM in Figure 39 is capable of displaying 256 characters using a 9x14 pixel character-cell. The characters are stored as an 8-bit pixels within a 82786 bit-map. To display the character, the window is programmed as an 8-bit/pixel bit-map with a horizontal zoom of 9 and a vertical zoom of 14. The 82786 will then place the 8-bit character code on its VDATA pins



292007-43

Figure 39. Support of Character-ROM and Bit-Mapped Graphics on Same Screen

during the scan lines when the character is to be displayed. The pixel counter is used to load the shift register every 9 pixels. This counter is synchronized to the beginning pixel of the window by starting when the window status pin falls. The row counter is used to supply row information to the character ROM. This counter is synchronized to the frame by starting from the end of the VSYNC pulse. Therefore, any character ROM window must begin at a multiple of 14 scan lines after VSYNC.

Another situation in which a character ROM display may be practical is if a very large character set is required. The Japanese Kanji characters are an example. The size of this character set is so large that it may be more practical to store the characters in a character ROM rather than load them from disk into the 82786 graphics memory. Figure 40 illustrates a configuration that can display up to 65536 characters from a very detailed (32x32) font. This circuit allows both text and graphics windows to be displayed on the screen simultaneously. One of the window status signals is used to select between text and graphics.

Such a character set requires a high resolution, generally monochrome display. The circuit in Figure 40 allows up to 200 MHz video (one bit/pixel) for very high resolution screens. The 82786 is programmed in super high-speed acceleration mode as described in Section 5.6.

The character-codes to be displayed should be placed in one bit/pixel bit-maps with 16 consecutive bits for each character. The hardware combines the 8-bit VDATA values from two consecutive pixels to generate the 16-bit character-code for the Character-ROM. If less than 65536 characters are required, not all of the 16-bit character code addresses need be used for the character-ROM. Some of these bits may be used for attributes such as blinking and reverse video. The ROM contains a 32x32 character font, each character is split up into 32-lines of four 8-bit bytes. The "pane" counter selects one of the four 8-bit bytes at a time. The "row" counter determines the current row of the character.

Character cell windows should be zoomed by 2 horizontally and by 32 vertically. The window must be placed at a multiple of 4-pixels from HSYNC and a multiple of 32-lines from VSYNC. It is possible to place windows at non-multiples from HSYNC and VSYNC if the "pane" and "row" counter parallel inputs are tied to other than ground.

5.10 Combining the 82786 With Other Video Sources

It is possible to combine graphics output from the 82786 with output from other video sources such as

broadcast TV, video recorders, and video laser disc players. The main requirement to perform such a feat is that both 82786 and the video source are locked in sync.

The 82786 has two independent Video Slave modes and HSYNC/VSYNC and BLANK can be independently configured as outputs or inputs. When HSYNC/VSYNC are programmed as inputs, then they are still outputs during the active display period if the window status is enabled. External HSYNC/VSYNC reset the 82786 horizontal and vertical counters respectively.

When BLANK is configured as output, the active display period is determined by the programmed values of VFLDSTRT, VFLDSTP, HFLDSTRT, and HFLDSTP. When BLANK is configured as an input, the external system determines the active display period. The internal video shift register generates video data only during the active display period.

HSYNC/VSYNC and BLANK would normally be programmed as input/output as follows:

HSYNC/ VSYNC	Blank	Application
Output	Output	Normal display generated by 82786
Input	Output	82786 generated display superimposed on externally-generated video
Input	Input	Multiple 82786 systems

The 82786 sync timing registers should be programmed to be as close to the frequency of the video source as possible. The 82786 should also be programmed for slave video-sync. The sync signals from the video source must be converted into separate TTL-level horizontal and vertical sync and fed to the 82786 HSYNC and VSYNC pins. The 82786 will then automatically sync itself up to the video source by waiting for its HSYNC input to fall before each scan line and waiting for its VSYNC input to fall before beginning a new display field.

For many applications, the 82786 video clock can be derived directly from a crystal oscillator. Since the 82786 syncs up to the nearest pixel on every scan line, even video sources with imperfect timings, such as video recorders where speed variations are common, will produce an acceptable picture. The frame-to-frame deviation of the 82786 graphics information on the screen relative to the video source will never be more than one pixel.

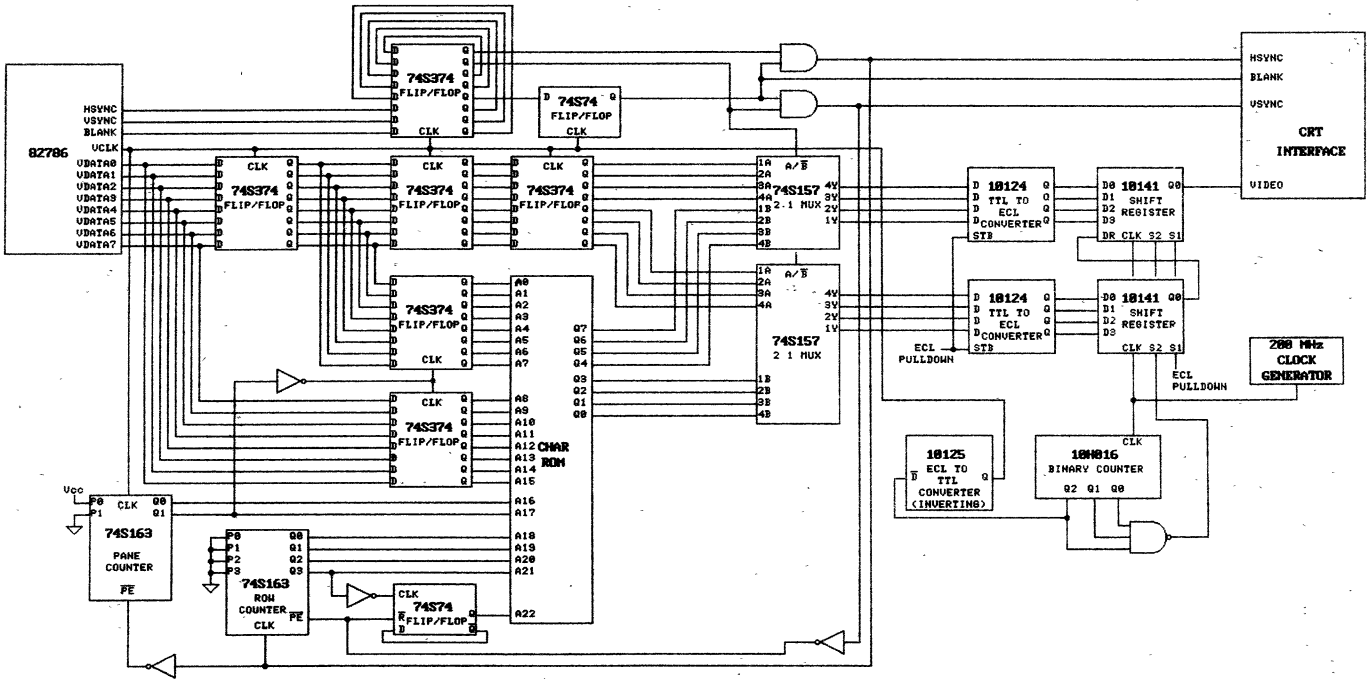


Figure 40. Support of Very Large Character-ROM and Bit-Map

7-218

For more demanding applications, the 82786 video clock can be synthesized directly from the video source timings using a phase-locked-loop circuit. The 82786 will still sync itself up every scan line, but now the relationship between HSYNC and the 82786 VCLK will remain constant. This implementation will create virtually no deviation between the 82786 graphics and the video source.

In the case of interlaced video, care must be taken to initially start the 82786 display just prior to the VSYNC before an even-field. The 82786 initialization software is responsible to guarantee that the first LD_ALL to start the 82786 display occurs sufficiently before the VSYNC during an odd-field so the first 82786 display field will match the video source even-field.

Once the 82786 is locked in sync with the video source, then the VDATA information from the 82786 can easily be combined with the video from the video source. Although the two video signals could be mixed on top of each other, probably the most common implementation is to switch between one or the other source. For example, the 82786 could create letters that are to be placed over the video picture. During the display scan, whenever a portion of a letter is to be displayed, the video from the 82786 can be switched in, otherwise the video source is switched in.

If the output of the video source is analog, the 82786 VDATA can be converted into an analog signal and an analog switch can be used. The state of the switch can be derived in a number of ways. If the switching is to be done on window boundaries, one window status pin can be used to control the switch. If the switching must be done within a window, a special graphics color code can be used to indicate that the 82786 video should be replaced with that from the video source. Possibly the color 11111111 could be placed on the VDATA pins and an 8-input NAND gate used to control the analog switch.

5.11 Other Types of Displays and Printers

The 82786 not only can be used with CRTs, but can also be used with other types of displays such as LCD, plasma, and intelligent printers. These devices have such a wide range of interface requirements that space does not permit each individual situation to be addressed in detail. Rather, some example requirements are discussed to illustrate how the 82786 can meet those needs.

PIXEL CLOCK RATE

The rate at which pixels are clocked into displays can vary immensely. The 82786 allows a very wide range of video clock frequencies from DC levels to 25 MHz to accommodate such devices. In addition, faster clock rates can be generated using the method described in Section 5.6

NO REFRESH

Printers and some displays are not required to be continuously refreshed. Needlessly running the 82786 Display Processor through refresh cycles steals bus bandwidth from the Graphics and other Processors. To eliminate this waste, the display can be turned off by resetting the DspOn bit (bit 0) in the Display Processor VStat Register (register number 0). When DspOn is reset, the Display Processor will continue to generate HSYNC, VSYNC, and BLANK and place Default-VDATA on the VDATA lines, but no bus bandwidth will be required.

When a change to the display is required, the DspOn bit can be set using the LD_REG or LD_ALL command. Once the refresh starts, another LD_REG command to turn the display back off can be placed in the Display Processor Opcode Register. The Display Processor will then automatically execute it when the refresh is completed.

PARTIAL DISPLAY UPDATES

Some displays that do not require continuous refresh, do have a long update time. It may take several seconds to update every pixel on the display. For small changes to the display, such as adding each character as it is typed by the user, it may be much more feasible to update only the portion of the display which is affected.

Using the very flexible windowing capability of the 82786, it is possible to only scan through a specific portion of the display.

PIXEL ADDRESS GENERATION

Some displays, especially those which allow only partial display updates, require that pixel location addresses be generated along with the pixel data. Although external circuitry could be used to generate these addresses, the 82786 can be used to generate them directly. If a single 8-bit address is all that is required, the DefaultVDATA register can be programmed to a value that the VDATA pins will reflect during blank time. With proper programming of the sync timing registers, this value can be clocked into the display before each scan line using HSYNC.

More complex pixel addresses can be generated by using the 82786 windowing capability. By creating a thin window at the beginning of each scan line, one or more bytes of address information can be sequentially clocked out over the VDATA pins before each line.

ULTRA HIGH RESOLUTION

Because some displays use either slow refresh times or don't require refresh at all, it is possible to have very high resolutions. All of the display counters in the 82786 are 12 bits allowing up to a 4096×4096 display size (some of this resolution may not be available depending on the number of sync clock cycles required). Trading off bits/pixel for resolution, the accelerated modes can provide 2, 4, or 8 times this resolution horizontally, up to 32768 pixels.

Still, some applications, such as printers, may require even greater resolutions. This is possible with the 82786 using external counters to generate the HSYNC, VSYNC, and BLANK inputs for the 82786. The 82786 should be programmed for slave video mode by setting the CRTMode register (Display Processor register 5 bits 2 and 3 should be set). Using all 16 horizontal windows, the horizontal resolution may be up to $4096 \times 16 = 65536$ pixels. Again, trading off bits/pixel for resolution, the accelerated modes can provide 2, 4, or 8 times this resolution, up to 524288 pixels. Vertically there is no limit to the resolution.

Such high resolutions require a lot of memory. For example, suppose a printer can generate 300 x 300 dots per square inch and is used on 8.5 x 11 inch paper. Assuming only one bit/pixel (no gray scale) the entire page would require:

$$\frac{300 \times 300 \times 8.5 \times 11}{8 \text{ bits}} = 1.05 \text{ Megabytes}$$

It may not be feasible to place this much memory into a printer. But it may be feasible to generate the display one strip at a time. Suppose that the first 300 lines are generated and printed. Once printed the next 300 lines can be generated and printed using the same memory. Now the memory required is only:

$$\frac{300 \times 300 \times 8.5}{8 \text{ bits}} = 96 \text{ Kbytes}$$

If the image to be printed can be described by a set of commands for the Graphics Processor, each strip can be very easily generated. The drawing bit-map and clipping rectangle are set for the first strip and the Graphics processor then runs through the command list. Once completed the strip may be printed. Then the bit-map and clipping rectangle are set for the second strip and the Graphics Processor again traverses the same command list to generate the second strip.

If there is enough memory for two strips, double buffering can be used to pipeline the operation. While the Display Processor is busy printing one strip, the Graphics Processor can be generating the next strip. The same approach can be extended to multiple pages, even using more than one 82786.

5.12 Calculating the Video Parameters

The 82786 video Display Processor is programmable to afford a wide variety of display formats. To determine the display format(s) that one would like to generate, several parameters must be considered.

Application parameters: these are dependent on the needs of the specific application and must be chosen by the designer.

Hres—horizontal resolution — number of pixels per horizontal line. When using accelerated video, Hres must be a multiple of Accel (following pages).

Vres—vertical resolution — number of vertical pixels (scan lines) per display

Vfreq—vertical frequency — rate at which CRT beam makes one pass from the top of the screen to the bottom. It is common to use 60 Hz but almost any other frequency can be generated by the 82786. US broadcast television standards use a 59.95 Hz rate. European video systems are based on a 50 Hz field rate. High resolution displays often use 40 Hz or lower. Slower rates reduce the speed requirements of the monitor and the 82786 video circuitry and also permit higher resolutions. However, slower rates also flicker more and may be intolerable to the operator. Generally, rates significantly under 60 Hz will tend to cause some perceptible flicker unless CRTs with long persistence phosphor are used.

ILC—interlacing — A non-interlaced display generates the entire display frame in one field scan. One method to double the resolution of a display is to use interlacing. Rather than use just one field to display all the information, two consecutive fields are used to create the entire display frame (Figure 41). Alternate scan lines are written during each alternate field. For TV-like pictures, where the image generally doesn't change drastically from one line to the next, interlacing allows a 30 Hz frame rate with a 60 Hz field rate without perceptible flicker. For detailed computer graphics, however, one line may change drastically from the next in color and/or intensity, in which case interlacing at such rates do cause perceptible flicker.

The 82786 supports both non-interlaced and interlaced displays. In addition, an interlaced-sync mode is available which generates sync signals in the manner used by interlaced displays, but generates the video signals in

the manner used by non-interlaced displays (both fields identical). This permits interlaced screens with consecutive pairs of lines identical.

Monitor parameters: these are dependent on the specific requirements of the display monitor used.

Hblank—horizontal blanking time — the time required for the CRT beam to jump from the right side of the display back to the left and stabilize. This is also called horizontal retrace time and is the sum of the horizontal sync and front and back porch times (Figure 42). Monitors typically range from 5–12 μ s.

Vblank—vertical blanking time — the time required for the CRT beam to jump from the bottom of the display back to the top and stabilize. This is also called vertical retrace time and is the sum of the vertical sync and front and back porch times (Figure 43). Monitors typically range from 600–1400 μ s.

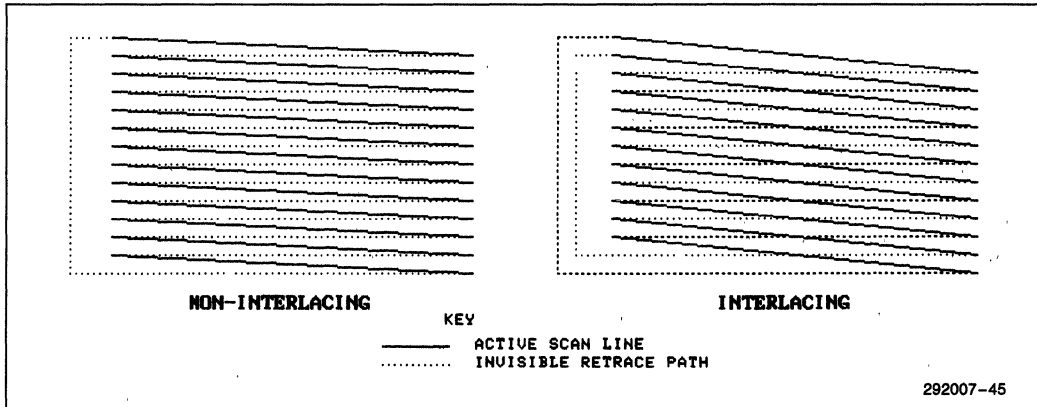


Figure 41. Non-Interlacing and Interlacing Displays

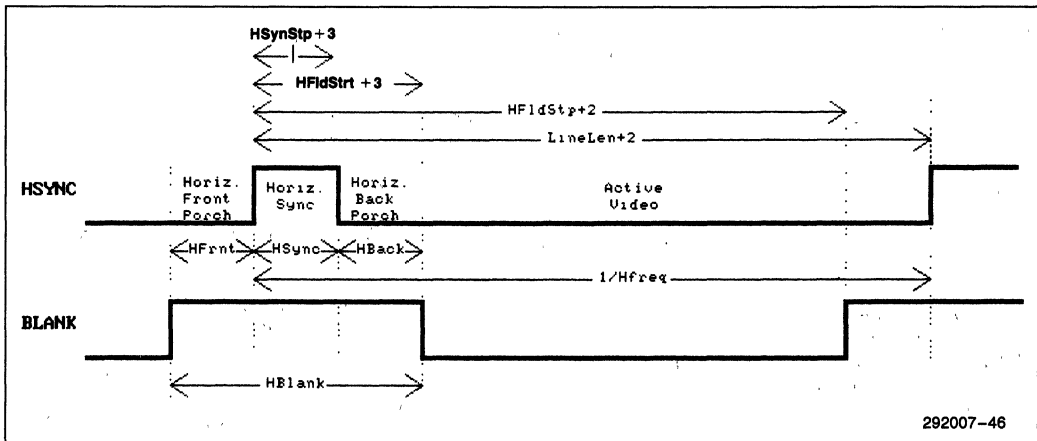


Figure 42. Horizontal Sync and Blank Timing Parameters

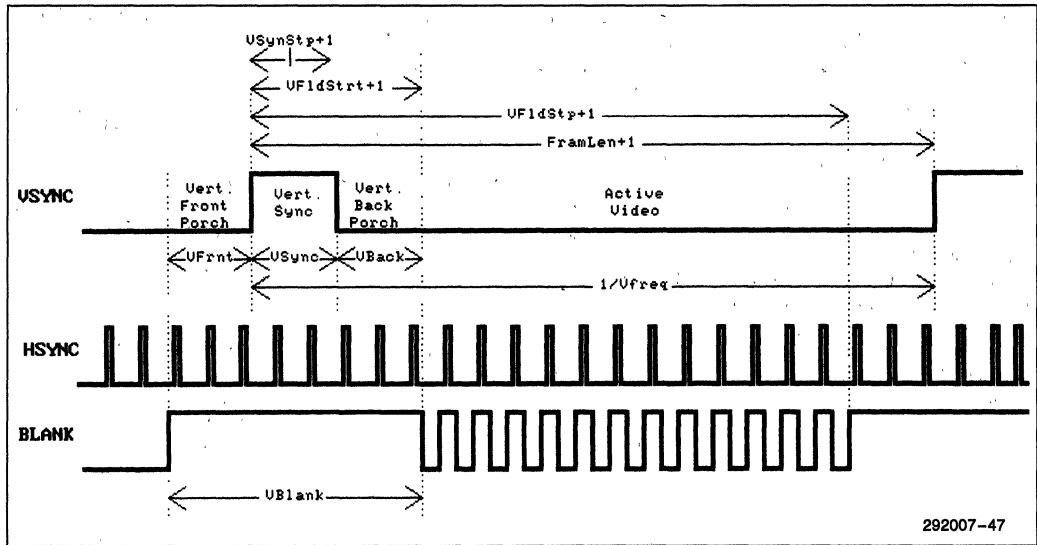


Figure 43. Vertical Sync and Blank Timing Parameters

Hfreq—horizontal frequency — the frequency at which horizontal lines are scanned. Monitors typically range from 15–36 kHz.

Vfreq—vertical frequency — the frequency at which the display field is scanned. Monitors typically range from 40–70 Hz.

BPP—bits per pixel — monitors with digital inputs restrict the number of usable bits/pixel. Monitors with analog inputs allow a virtually unlimited range of intensities with the use of Digital-to-Analog converters. This parameter is mainly dependent on the video interface hardware described in the previous sections.

Color monitors generally limit the perceivable horizontal and vertical resolution due to their shadow mask. See the specific monitor specifications for more details.

Video interface parameters: these are dependent on the 82786 component and the video interface logic.

VCLK—video clock frequency — the video input clock into the 82786. It has a maximum rate of 25 MHz and may be chosen so that the frequency evenly divides by both Hfreq and Vfreq.

Accel—82786 video acceleration — this parameter is determined by what mode the 82786 is used in. Normally Accel = 1. If the trade-offs mentioned in Section 5.6 are used to attain higher video rates at the expense of fewer bits/pixel, then the value for Accel should be 2, 4, or 8.

	Video Mode	Max DotClk	Programmed Accel Bits
Accel = 1	Normal	25 MHz	0 0
Accel = 2	High Speed	50 MHz	0 1
Accel = 4	Very High Speed	100 MHz	1 0
Accel = 8	Ultra High Speed	200 MHz	1 1

DotClk—pixel dot clock frequency — this is normally the same as VCLK. However, when accelerated video modes are used, this is either 2, 4, or 8 times VCLK.

$$\text{DotClk} = \text{VCLK} \times \text{Accel}$$

HSynStp, HFldStrt, HFldStp, LineLen—these are values programmed into the 82786 Display Processor to determine the horizontal scan timing (Figure 42). They may be set to any value from 0 to 4095. Their values should also fit the formula:

$$\text{HSynStp} < \text{HFldStrt} < \text{HFldStp} < \text{LineLen}$$

VSynStp, VFldStrt, VFldStp, FramLen—these are values programmed into the 82786 Display processor to determine the vertical scan timing (Figure 43). They may be set to any value from 0 to 4095. Their values should also fit the formula:

$$\text{VSynStp} < \text{VFldStrt} < \text{VFldStp} < \text{FramLen}$$

Once the above parameters are evaluated, the video parameters can actually be calculated. The parameters interact quite heavily so that, for example, if a specific

horizontal and vertical resolution at a specific field rate is required, the monitor frequencies and blank times may need to be altered. It may take several iterations to optimize all the parameters. The calculations can be performed by hand. However, a much easier way to manipulate these values is by using a spreadsheet program. A spreadsheet allows the parameters to be easily manipulated with their affects immediately displayed. A spreadsheet template for this purpose is given in Section 5.13.

The following formulas are used to determine the video parameters. Along with the formulas is an example calculation. For the example, let's generate a $640 \times 400 \times 8$ bit/pixel (256 color) screen at 60 Hz non-interlaced. We will assume:

Hres	=	640 pixels
Vres	=	400 pixels
Vfreq%	=	60 Hz
Hblank%	=	12 μ s
Vblank%	=	1300 μ s
Accel	=	1 (no external acceleration)

Variables with a percent (%) after them represent desired values, the actual value will be calculated below.

ROUND(X) will be used to denote rounding off X to the nearest integer.

First, calculate the vertical resolution per field. Since our display is non-interlaced, the value is the same as the vertical resolution.

If interlaced then: $VresFld = Vres/2$
 else: $VresFld = Vres$
 $VresFld = 400$ pixels

With interlaced screens, $VresFld$ is half the vertical resolution. For example, with 525 lines, use 262.5 for $VresFld$.

Now, calculate the horizontal frequency required. Subtract the vertical blank time from the vertical period and divide by the active vertical lines to obtain the horizontal period. Inverting all that gives the horizontal frequency.

$$\begin{aligned} Hfreq\% &= \frac{1}{Hperiod\%} = \frac{VresFld}{(1/Vfreq\%) - Vblank\%} \\ &= \frac{400}{(1/60) - 1300 \mu s} = 26.03 \text{ kHz} \end{aligned}$$

In a similar manner, calculate the pixel dot clock required.

$$\begin{aligned} \text{DotClk}\% &= \frac{1}{\text{DotPeriod}\%} = \frac{Hres}{(1/Hfreq\%) - Hblank\%} \\ &= \frac{640}{(1/26.03 \text{ kHz}) - 12 \mu s} = 24.23 \text{ MHz} \end{aligned}$$

And then calculate the actual 82786 VCLK. Since external acceleration circuits are not used in our example, it turns out to be the same as the DotClk.

$$\begin{aligned} \text{VCLK}\% &= \text{DotClk}\% / \text{Accel} = 24.23 \text{ MHz} / 1 \\ &= 24.23 \text{ MHz} \end{aligned}$$

Great, now all we need is a 24.23 MHz crystal is needed to generate VCLK. But since such a crystal is tough to find, try a 25 MHz crystal instead and see how it affects the rest of the parameters. First of all, the pixel dot clock changes.

$$\text{DotClk} = \text{VClk} \times \text{Accel} = 25.00 \text{ MHz} \times 1 = 25.00 \text{ MHz}$$

Now, see how many VCLK's are required for the horizontal blank time.

$$\begin{aligned} \text{HblankClks} &= \text{ROUND}(\text{VCLK} \times \text{Hblank}\%) = \text{ROUND} \\ &(25 \text{ MHz} \times 12 \mu s) = 300 \end{aligned}$$

Now we calculate the actual horizontal blank time.

$$\text{Hblank} = \frac{\text{HblankClks}}{\text{VCLK}} = \frac{300}{25 \text{ MHz}} = 12 \mu s$$

The actual horizontal period is then the time required to display one line of pixels plus the blanking time.

$$\begin{aligned} Hfreq &= \frac{1}{(Hres / \text{DotClk}) + \text{Hblank}} \\ &= \frac{1}{(640 / 25 \text{ MHz}) + 12 \mu s} = 26.60 \text{ kHz} \end{aligned}$$

The number of horizontal lines per field can now be calculated:

$$\begin{aligned} \text{VFieldLines} &= \text{ROUND}(Hfreq / Vfreq\%) \\ &= \text{ROUND}(26.60 \text{ kHz} / 60 \text{ Hz}) = 443 \end{aligned}$$

If an interlaced display is used, $VFieldLines$ should be rounded-off to the closest odd integer.

The number of scan lines determines the actual vertical frequency:

$$\begin{aligned} \text{Vfreq} &= Hfreq / \text{VFieldLines} = 26.60 \text{ kHz} / 443 \\ &= 60.05 \text{ Hz} \end{aligned}$$

Now that the major parameters are calculated and we are satisfied with them, we can break up the blanking times into sync, front and back porch times. Typical monitor values are:

$$\begin{aligned} \text{HSync} &= 2 \mu\text{s} & \text{VSync} &= 300 \mu\text{s} \\ \text{HBack} &= 6 \mu\text{s} & \text{VBack} &= 800 \mu\text{s} \end{aligned}$$

$$\begin{aligned} \text{HSyncClks} &= \text{ROUND}(\text{VCLK} \times \text{HSYNC}) \\ &= \text{ROUND}(25 \text{ MHz} \times 2 \mu\text{s}) = 50 \\ \text{HBackClks} &= \text{ROUND}(\text{VCLK} \times \text{HBack}) \\ &= \text{ROUND}(25 \text{ MHz} \times 6 \mu\text{s}) = 150 \end{aligned}$$

$$\begin{aligned} \text{VSyncClks} &= \text{ROUND}(\text{Hfreq} \times \text{VSYNC}) \\ &= \text{ROUND}(26.6 \text{ kHz} \times 300 \mu\text{s}) = 8 \\ \text{VBackClks} &= \text{ROUND}(\text{Hfreq} \times \text{VBack}) \\ &= \text{ROUND}(26.6 \text{ kHz} \times 800 \mu\text{s}) = 21 \end{aligned}$$

Now it's a simple matter to calculate the values for the eight 82786 Display Processor video timing registers.

$$\begin{aligned} \text{HSynStp} &= \text{HSyncClks} - 3 \\ &= 50 - 3 = 47 \end{aligned}$$

$$\begin{aligned} \text{HFldStrt} &= \text{HSynStp} + \text{HBackClks} \\ &= 47 + 150 = 197 \end{aligned}$$

$$\begin{aligned} \text{HFldStp} &= \text{HFldStrt} + (\text{Hres} / \text{Accel}) \\ &= 197 + (640 / 1) = 837 \end{aligned}$$

$$\begin{aligned} \text{LineLen} &= \text{HBlankClks} + (\text{Hres} / \text{Accel}) - 3 \\ &= 300 + (640 / 1) - 3 = 937 \end{aligned}$$

For non-interlaced displays:

$$\begin{aligned} \text{VSyncStp} &= \text{VSyncLines} - 1 \\ &= 8 - 1 = 7 \end{aligned}$$

$$\begin{aligned} \text{VFldStrt} &= \text{VSyncStp} + \text{VBackLines} \\ &= 7 + 21 = 28 \end{aligned}$$

$$\begin{aligned} \text{VFldStp} &= \text{VFldStrt} + \text{Vres} \\ &= 28 + 400 = 428 \end{aligned}$$

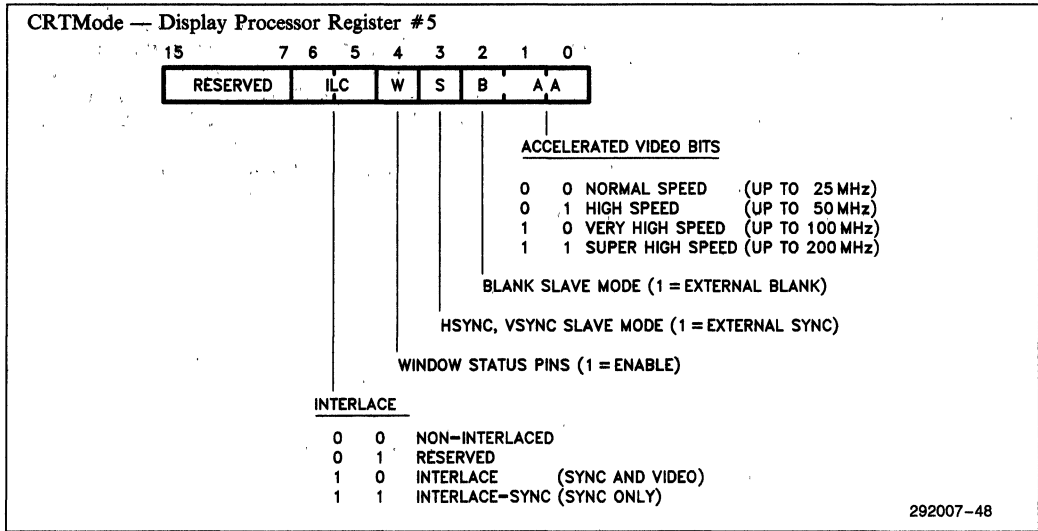
$$\begin{aligned} \text{FramLen} &= \text{VFieldLines} - 1 \\ &= 443 - 1 = 442 \end{aligned}$$

For interlaced displays:

$$\begin{aligned} \text{VSyncStp} &= (\text{VSyncLines} - 1) \times 2 \\ \text{VFldStrt} &= \text{VSyncStp} + (\text{VBackLines} \times 2) \\ \text{VFldStp} &= \text{VFieldsLines} - 2 \end{aligned}$$

Make sure $\text{LineLen} > \text{HFldStp}$ and that $\text{FramLen} > \text{VFieldLines}$. If not, your parameters are inconsistent and you should modify your requirements and re-calculate.

Finally, the bits for the CRTMode Register should be determined. For our example, non-interlaced mode is used and no accelerated video is required. Assuming the 82786 is used to generate the HSYNC, VSYNC and BLANK signals and assuming the window Status pins are not used, the CRTMode registers should be loaded with all zeros.



The host CPU software is required to load the values of the eight video timing registers and the CRTMode register. Generally, this is done during system initialization. The registers should all be loaded simultaneously using the LD_ALL command rather than using individual LD_REG commands. This ensures that the video sync signals are never invalid while registers are being loaded.

Some CRTs can be permanently damaged by supplying the wrong sync frequencies to them. To prevent invalid video sync signals, the HSYNC, VSYNC, and BLANK

pins are tristated after RESET until the CRTMode Register has been written to.

5.13 A Spreadsheet for Calculating Video Parameters

As seen in the previous section, quite a number of calculations are required to determine the 82786 video parameter constants. Often several iterations through the calculations are required to optimize the display format. This process can be greatly simplified by using a spreadsheet.

An example of the output from such a spreadsheet is shown below. This example illustrates a 1290 x 968 x 4-bit/pixel (16 color) interlaced 60 Hz display. The user has supplied all of the values under the "DESIRED" column and the spreadsheet program has calculated the rest. The "ACTUAL" column shows the closest timings and parameters that the 82786 can actually supply. The "82786 DP REGISTER VALUES" shows the values that should be programmed into the Display Processor registers to generate such a display.

The User can easily modify the "DESIRED" values until the "ACTUAL" values meet the application's needs. Care should be taken to ensure that all "ACTUAL" values are logically correct. If for example, any of the calculated parameters are negative, then the set of "DESIRED" parameters can not produce such a display, so some parameters must be adjusted.

8 2 7 8 6 V I D E O P A R A M E T E R S

Type under DESIRED column only: ACTUAL & REGISTER columns are calculated

PARAMETER	DESIRED	ACTUAL	82786 DP REGISTER VALUES
-----	-----	-----	-----
Video Clock VCLK (MHz):	25	25	
Acceleration (1,2,4 or 8):	2	2	
Interlacing (1 = no, 2 = yes):	2	2	
Horiz Resolution (Pixels):	1290	1290	
Vert. Resolution (Pixels):	968	968	
Horiz Line Rate (kHz):	---	30.487	LineLen: 818
Horiz Sync Width (µs):	2	2	HSynStp: 48
Horiz Back Porch (µs):	4	4	HFlldStrt: 148
Horiz Front Porch (µs):	1	1	HFlldStp: 793
Vert. Frame Rate (Hz):	60	59.956	FramLen: 1015
Vert. Sync Width (µs):	200	196.8	VSynStp: 10
Vert. Back Porch (µs):	400	393.6	VFlldStrt: 34
Vert. Front Porch (µs):	---	213.2	VFlldStp: 1002

The template follows. This template should be easily adaptable to nearly any spreadsheet program. This particular spreadsheet program uses @ROUND(X,0) to denote rounding to the nearest integer. If no rounding function is available in your spreadsheet program, you can substitute the integer function (which truncates the fractional portion to return the next lowest integer) for the round function:

substitute @INT(X+0.5) for @ROUND(X,0)

After entering the template into your favorite spreadsheet, you may wish to verify that it is working correctly by entering the "DESIRED" values of the above example and checking that the "ACTUAL" and "REGISTER" results match.

-----A-----	---B---	-----C-----	---D---	-----E-----
1:	8 2 7 8 6	V I D E O	P A R A M E T E R S	
2:	Type under DESIRED column only: ACTUAL & REGISTER columns are calculated			
3:	-----			
4:	PARAMETER	DESIRED	ACTUAL	82786 DP REGISTER VALUES
5:	-----	-----	-----	-----
6:	Video Clock VCLK (MHz):		+B6	
7:	Acceleration (1,2,4 or 8):		+B7	
8:	Interlacing (1=no, 2=yes):		+B8	
9:	Horiz Resolution (Pixels):		@ROUND(B9/C7,0)*C7	
10:	Vert. Resolution (Pixels):		@ROUND(B10,0)	
11:				
12:	Horiz Line Rate (kHz):	---	(C6*1000)/(E12+2)	LineLen: @ROUND(C6*B15,0)+E15
13:	Horiz Sync Width (μs):		(E13+2)/C6	HSynStp: @ROUND(C6*B13,0)-3
14:	Horiz Back Porch (μs):		(E14-E13)/C6	HFldStrt: @ROUND(C6*B14,0)+E13
15:	Horiz Front Porch (μs):		(E12-E15)/C6	HFldStp: +E14+(C9/C7)
16:				
17:	Vert. Frame Rate (Hz):		(C8*C12*1000)/(E17+C8)	FramLen: @ROUND((C12*1000)/B17-(C8-1)/2,0)*C8-1
18:	Vert. Sync Width (μs):		((E18+C8)*1000)/(C12*C8)	VSynStp: (@ROUND((C12*B18)/1000,0)-1)*C8
19:	Vert. Back Porch (μs):		((E19-E18)*1000)/(C12*C8)	VFldStrt: @ROUND((C12*B19)/1000,0)*C8+E18
20:	Vert. Front Porch (μs):	---	(E17-E20)*1000/(C12*C8)	VFldStp: +E19+C10

APPENDIX A SAMPLE INITIALIZATION CODE

Many registers within the 82786 must be initialized to configure the 82786 for the particular hardware environment it resides in. This appendix contains assembly language code to initialize the 82786 for one particular configuration:

- synchronous 10 MHz 80286 interface (Sections 4.2 and 4.3, Figure 18)
- one row of two interleaved banks of 51C256 Fast-page-mode DRAM (Section 3.3, Figure 9)
- 640 x 300 x 8-bit/pixel non-interlaced 60 Hz display, 25 MHz VCLK (Section 5.11, Figure 27)

All of the parameters to be initialized for this configuration are calculated under their corresponding sections in the body of this application note. To calculate the parameters for other configurations, refer to these sections.

This example of initialization code can be used to initially test many of the hardware functions. The code should create a stable display on the CRT. The display will consist of a black field which covers the entire screen (a 640 x 400 black rectangle). In the center of the rectangle is a 16 x 16 pixel arrow-shaped red and yellow cursor.

```
name Initialization82786
```

```
Memory82786 segment at 0C000h
```

```
    ;segment located at start of CPU-mapped 82786 memory
```

```
    ;define locations of 82786 internal registers
```

```
        org 0
```

```
Internalrelocation    dw    ?    ;BIU registers
Reserved              dw    ?
BIUControl            dw    ?
RefreshControl        dw    ?
DRAMControl           dw    ?
DisplayPriority        dw    ?
GraphicsPriority       dw    ?
ExternalPriority       dw    ?
```

```
        org 20h
```

```
GPOpcode              dw    ?    ;Graphics Processor registers
GPLinkAddressLower    dw    ?
GPLinkAddressUpper    dw    ?
GPStatus              dw    ?
GPInstructionPtrLower dw    ?
GPInstructionPtrUpper dw    ?
```

```
        org 40h
```

```
DPOpcode              dw    ?    ;Display Processor registers
DPParameter1          dw    ?
DPParameter2          dw    ?
DPParameter3          dw    ?
DPStatus              dw    ?
DefaultVDATA          dw    ?
```

;location of values for Display Processor LD_ALL instruction

org 80h

DPLdAllRegs label word

```

dw 3 ;VStat: turn on display and cursor
dw 0FFh ;IntMask: mask all interrupts
dw 24 ;TripPt: trip point = 24 FIFO dwords
dw 0 ;Frint: cause interrupt every frame (interrupt is masked)
dw 0 ;
dw 0 ;CRTMode: non-interlaced, non-accelerated, master sync&blank
dw 47 ;HSynStp: horizontal sync stop :
dw 197 ;HFldStrt: horizontal field start :
dw 837 ;HFldStp: horizontal field stop : 8 video timing registers
dw 937 ;LineLen: horizontal line length : are programmed for
dw 7 ;VSynStp: vertical sync stop : 640 x 400 at 60 Hz
dw 28 ;VFldStrt: vertical field start : with 25 MHz VCLK
dw 428 ;VFldStp: vertical field stop :
dw 442 ;FramLen: vertical frame length :
dw offset WinDescL ;DescAddrL:descriptor address pointer lower
dw 0 ;DescAddrU:descriptor address pointer upper
dw 0 ;(Reserved)
db 0 ;ZoomY: no vertical zoom
db 0 ;ZoomX: no horizontal zoom
dw 0 ;FldColor: black field color
dw 0FFh ;BdrColor: white border color
dw 0 ;Pad1BPP: pad with zeros for 1 bit/pixel
dw 0 ;Pad2BPP: pad with zeros for 2 bits/pixel
dw 0 ;Pad4BPP: pad with zeros for 4 bits/pixel
db 2 ;CursorPad:pad with red for cursor (yellow cursor in red box)
db 80h ;CsrStyle: opaque 16x16 block cursor, no window status
dw 510 ;CsrPosX: put cursor in middle of screen (horizontally)
dw 220 ;CsrPosY: put cursor in middle of screen (vertically)

dw 0000000110000000b ;CsrPat0: create arrow-shaped cursor pattern
dw 0000001111000000b ;CsrPat1:
dw 0000011111100000b ;CsrPat2:
dw 0000111111110000b ;CsrPat3:
dw 0001111111111000b ;CsrPat4:
dw 0011111111111100b ;CsrPat5:
dw 0111111111111110b ;CsrPat6:
dw 1111111111111111b ;CsrPat7:
dw 0000011111000000b ;CsrPat8:
dw 0000011111100000b ;CsrPat9:
dw 0000011111100000b ;CsrPatA:
dw 0000011111100000b ;CsrPatB:
dw 0000011111100000b ;CsrPatC:
dw 0000011111100000b ;CsrPatD:
dw 0000011111100000b ;CsrPatE:
dw 0000011111100000b ;CsrPatF:

```

;location of strip descriptor list

WinDescL label word ;strip descriptor list

```

;header of strip descriptor
dw 399 ;lines in strip (400 covers entire screen)
dw 0 ;lower link to next strip descr (there is none)
dw 0 ;upper link to next strip descr (there is none)
dw 0 ;number of tiles in strip (only one)

```

```

                                ;first (and only) tile descriptor
dw      0                       ;bit-map width (not applicable, this is field)
dw      0                       ;memory start lower addr (not applicable)
dw      0                       ;memory start upper addr (not applicable)
dw      639                     ;field width (640 covers entire screen)
dw      0                       ;fetch count (not applicable, this is field)
dw      00001h                 ;set field bit,use top,bottom,left,right borders

```

Memory82786 ends

Initialize82786 segment ;code to initialize 82786

```

mov     ax,seg BIUControl
mov     ds,ax                    ;put 82786 register segment in ds

assume cs:Initialize82786, ds:Memory82786

mov     byte ptr BIUControl, 30h ;convert 82786 to 16-bit bus...
mov     byte ptr BIUControl+1, 0 ;...must use two 8-bit transfers

mov     InternalRelocation, 01h ;locate reg's at 82786 mem addr 0h

mov     DRAMControl, 1Dh        ;1 row, interleaved 51C256 DRAM
mov     RefreshControl, 18      ;request refresh every 15.2 uS

mov     DisplayPriority, 110110b ;set Display FPL, SPL = 6

mov     GraphicsPriority, 010010b ;set Graphics FPL, SPL = 2
mov     ExternalPriority, 100000b ;set External FPL      = 4

mov     DPParameter1, offset DPLdAllRegs ;address for LD_All command
mov     DPParameter2, 0CH
mov     DPOpcode, 5             ;let DP perform LD_All command

ret                                ;end of initialization subrtn

```

Initialize82786 ends

If the constants in the CPU-mapped 82786 memory for the LD—ALL command and the strip descriptor list (in Memory82786 segment) cannot be loaded into 82786 memory by the system's program loader, they will have to be loaded by the initialization code. One method is to have the loader load them into CPU system memory and use a repeat-move-string command in the initialization code to move these constants into the 82786 graphics memory. Alternatively, it is possible to place these constants in the 82786-mapped CPU memory and allow the 82786 to fetch them using master-mode. This method, however, is not as efficient because the 82786 must re-fetch the strip descriptor list for every display frame.

The Graphics Processor is not used in this initialization code. To fully initialize the Graphics Processor, the following commands are required:

- Def_Bit_Map for all drawing and BitBlt commands
- Def_Logical_Op for all drawing and BitBlt commands
- Def_Colors if line/character drawing used
- Def_Texture if line drawing used
- Def_Char_Set if character drawing used
- Def_Char_Orient if character drawing used
- Def_Char_Space if character drawing used
- Load_Reg initialize stack pointer if macros used
- Load_Reg set poll-on-exception mask if used
- Load_Reg set interrupt mask if interrupts used



**APPLICATION
NOTE**

AP-259

November 1986

**The 82786
CHMOS Graphics Coprocessor
Architectural Overview**

Order Number: 122711-002

PREFACE

82786 FEATURES AND PERFORMANCE

The 82786 is a powerful, yet flexible component which will be a candidate as a standard for microcomputer graphics applications including personal computers, engineering workstations, terminals, and laser printers. Its advanced software interface contrasts sharply with existing products by making applications and systems level programming efficient and straight-forward. Its performance and high-integration make it a cost-effective component while improving the performance of nearly any design.

The following list is a summary of the 82786's capabilities (assuming 10 MHz system clock and 25 MHz video clock):

Windows:	Practically unlimited support
Colors:	Up to 1024 displayable simultaneously with support for 4 external color palettes
Lines, Polylines, Polygons:	2.5 Million pixels per second
Circles, Arcs:	2.0 Million pixels per second
Fills:	Supported via horizontal line command (30 Million bits per second)
Bit Block Transfer:	24 Million bits per second
Bit-map Memory:	Up to 4 MBytes of directly accessed DRAM
Resolution:	Up to 200 MHz monitors supported; this is equivalent to configurations such as 640 x 480 x 8 or 1024 x 1024 x 2 @ 60 Hz (non-in-

terlaced); up to 4096 x 4096 x 1 or 2048 x 1536 x 8 with video DRAMs.

Zoom:	1 to 64 times vertical and horizontal
Character Drawing:	25 thousand per second with colors, path, and rotation attributes
Character Fonts:	Unlimited number from bit-map or system memory
Character Size:	16 x 16 maximum hardware size; unlimited with bit-block transfer
Scroll, Pan:	Instantaneous in any direction with no external logic

The performance of the 82786 is of little value without applications and system-level software to use it. Customers can either write their own software or use the appropriate third-party vendors' packages.

The 82786 was designed to permit compatibility with de facto hardware standards. Use of the 82786 with appropriate Intel microprocessors permits the design of systems which can emulate the family of IBM™ personal computer products. The 82786's support of the IBM Color Graphics Adapter-compatible bit-map eases the task of running existing applications software on new video hardware.

For details please refer to the 82786 Data Sheet, the User Manual and Application Notes.

For all questions, clarifications, or requests for additional documentation please contact your local Intel sales office or authorized distributor.

CHAPTER 1 INTRODUCTION

1.1 OVERVIEW

This document provides the reader with an introduction to the architecture and key features of the Intel 82786 Graphics Coprocessor from Intel. The 82786 serves such applications as graphics terminals and work stations, personal computers, printers, and other products requiring the capability to create, store, and output bit-map graphics.

The 82786 works with all Intel microprocessors, and is a high-performance replacement for sub-systems and boards which have traditionally used discrete components and/or software for graphics functions. The 82786 requires minimal support circuitry for most system configurations, and thus reduces the cost and board space requirements of many applications. The 82786 is based on Intel's advanced CHMOS process.

The advanced performance and ease-of-use of the 82786 make it a candidate for an industry standard for applications in microcomputer graphics markets. Some of the leading features of the 82786 are:

- Fast polygon and line drawing
- Hardware windows
- High speed character drawing
- Interface designed for device independent software standards
 - Virtual Device Interface
 - Graphics Kernel System
 - NAPLPS
- Advanced DRAM/VRAM controller for graphics memory up to 4 Mbytes
- Fast bit-block copies between system and bit-map memories
- Supports up to 200 MHz CRTs
- Up to 1024 simultaneous colors per frame
- Programmable video timing
- High Integration
- Third-party software support
- 88 pin leaded chip carrier and pin grid array
- Provides support for rapid filling with patterns
- IBM Personal Computer Color Graphics Adapter-compatible bit-map
- International character support
- Advanced CHMOS technology
- Integral dual port video DRAM/VRAM support

1.2 ARCHITECTURAL MODEL

The 82786 architecture fits with traditional computer graphics models. A typical subdivision of the tasks is:

- Graphics task partitioned into:
 - Drawing (line, polygons, characters, block image copies)
 - Windowing (concurrent windows on the screen)
 - Refresh (CRT timing, video data output)
- Typical integrated solutions to these functions have been:
 - First generation IC: 6845, 8275 - refresh
 - Second generation LSI: 82720 - drawing + refresh
 - Third generation VLSI: 82786 - drawing + windowing + refresh

The 82786 is a coprocessor with two separate on-chip processing units, the Graphics Processor and Display Processor, which operate concurrently with the system CPU. Commands to the display and graphics processors are placed in memory by the CPU. Registers on the 82786 are dedicated to pointing to the starting addresses of the first memory blocks of commands controlling the on-chip processors, and each memory block points to subsequent blocks in a linked-list architecture. Access by the CPU to these registers may be I/O- or memory-mapped, and portions of memory may be shared between the 82786 and the CPU.

1.3 BIT MAPS AND WINDOWS

The 82786 concepts of "bit maps" and "windows" are based upon definitions from the ANSI work on windows.

The 82786 can create and maintain multiple sets of graphics images in memory. These sets of images in memory are called "bit maps". 82786 can combine subsets of these bit-maps into a viewable, multi-region display screen. Each of these separate areas on the screen is called a "window".

Most graphics systems today use software to generate a bit-map representation of the full contents of the display called a "frame buffer". The 82786 uses a high-level strip descriptor list and specialized hardware to generate the screen contents using portions from separate bit maps of memory (Figure 1-1). This permits the display to be instantaneously altered, eliminating the time required to update a similar frame buffer image using software alone.

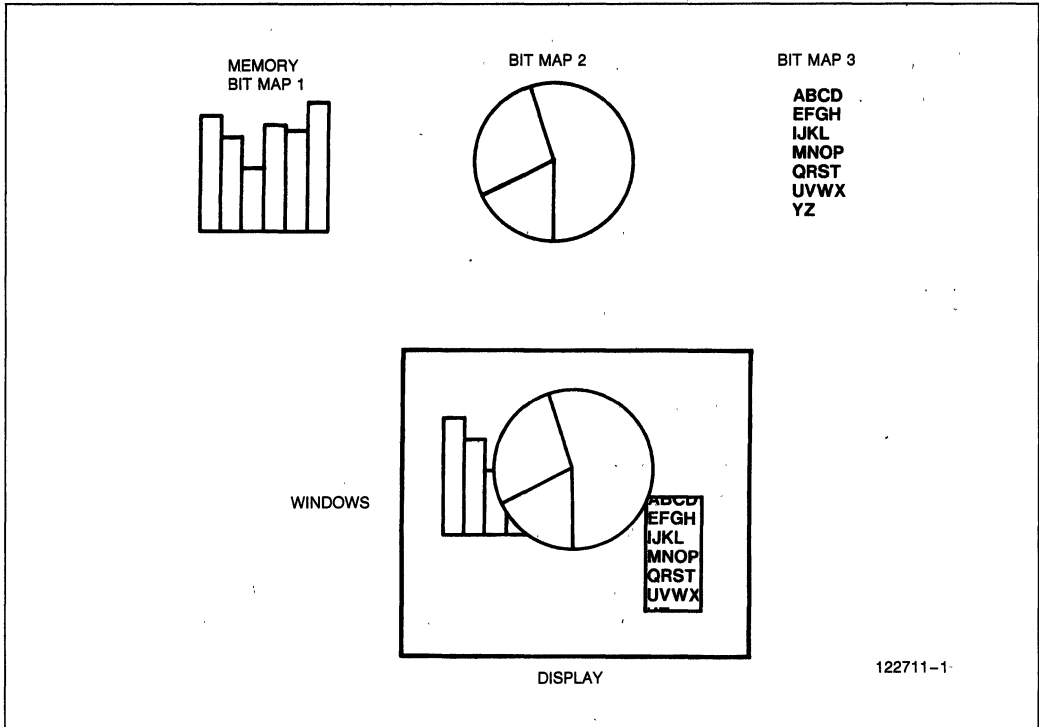


Figure 1-1. Bit Maps and Windows

1.4 FUNCTIONAL OVERVIEW

The 82786 performs many functions within a single integrated circuit. Figure 1-2 identifies a block diagram of the component and explanations of each function module.

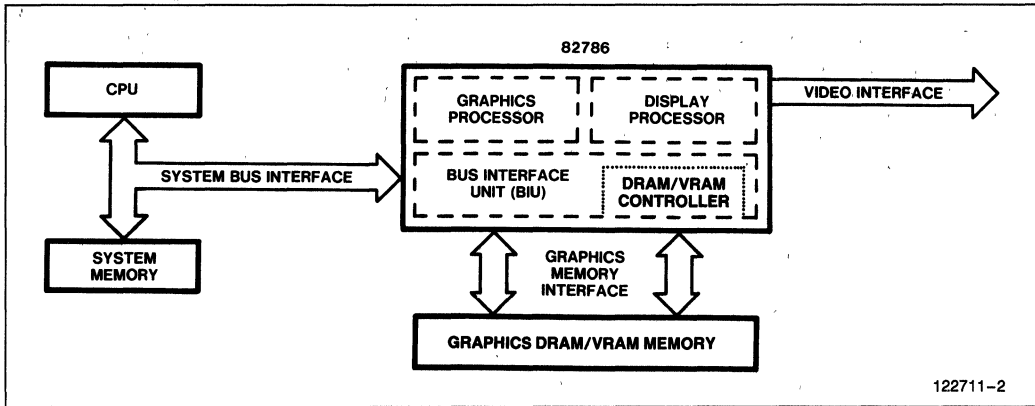


Figure 1-2. 82786 Block Diagram

The major functions of each block are:

- Graphics Processor (GP): — draws lines, circles, polygons, and other primitives
 - draws characters
 - executes block image manipulation instructions
- Display Processor (DP): — manages windows including zoom
 - provides cursor
 - refreshes screen (up to 200 MHz dot rate)
 - loads shift register of VRAMs
- DRAM/VRAM Controller: — controls up to 4 Mbytes of interleaved graphics memory including page-, static column-, and fast page-mode DRAMs (interleaved or non-interleaved banks)
- BIU: — allows the CPU to access the graphics memory and the 82786 to access the system memory

CHAPTER 2 GRAPHICS PROCESSOR

2.1 OVERVIEW

The Graphics Processor creates and updates all of the graphics and text in each of the bit maps within graphics memory. It is responsible for all of the geometric drawing, character drawing and image movement within and between the bit maps. Some features of the Graphics Processor are:

- permits bit maps to begin at any even word in system or graphics memory; only one bit map is active for GP drawing at one time although many bit maps may reside in memory simultaneously.
- permits bit maps to be any size (up to 32K x 32K pixels) and use 2, 4, 16, or 256 colors (i.e. 1, 2, 4, or 8 bits per pixel);
- draws geometric shapes with attributes such as texture and color, into bit maps;
- draws characters with attributes such as color, path, rotation, and proportional spacing using user-defined fonts into bit maps;
- combines one rectangular portion of an image with another area, within the same bit map or into another bit map. (Bit Block Transfer or BitBlit);
- all drawing allows logical operations between source and destination (for example Exclusive-Or of the Complement of Source with Destination);
- all drawing can be clipped to a rectangular region;
- supports picking, a mechanism for advanced user interfaces which allows the issuing commands via the selection of "graphic menus" (called icons) by manipulating pointing devices.

The Graphics Processor fetches its commands directly from a linked list memory-resident Graphic Command Block (GCMB), which is created and updated by the CPU. The initial address of the GCMB is contained in the Graphics Processor Instruction Pointer Register in the 82786 and the addresses of subsequent commands are pointed to by the contents of previous commands. Each command contains a bit which indicates to the Graphics Processor that it should stop (if set) and await new commands. More detail on the command format is given in section 2.8 "Graphics Processor Command List Format."

2.2 BIT MAPS

All graphics and text creation is written into bit maps. Bit maps are rectangular drawing areas composed of bits of pixel-oriented memory. The bit maps may be up

to 32,000 pixels in each direction and contain from one to eight bits of color or gray scale information. Bit maps may be started on any even address in the 4 Mbyte space and the number of bit maps in memory is unlimited (except by the amount of memory available). The variable bits per pixel feature permits the use of several bits per pixel for multicolor graphics while using only a single bit per pixel for efficient text memory.

2.3 GRAPHICS PROCESSOR COMMANDS

Graphics Processor commands are divided into five classes:

1. Non-Drawing Commands
2. Drawing Control Commands
3. Geometric Commands
4. Bit Block Transfer (BitBlit) Commands
5. Character Block Transfer (CharBlit) Commands

2.3.1 Non-Drawing Commands

The first class of commands are used to control the method in which the commands are fetched. Also included in this list are commands to load and dump 82786 internal registers. These commands are:

- NOP - No Operation
- LINK - Link To Next Command (Unconditional Jump)
- ENTER_MACRO - Enter Macro (Subroutine Call)
- EXIT_MACRO - Exit Macro (Subroutine Return)
- INTR_GEN - Generate Interrupt
- DUMP_REG - Dump Internal Register
- LOAD_REG - Load Internal Register

2.3.2 Drawing Control Commands

The Graphics Processor works in only one bit map and with one set of attributes at a time. The Graphics Processor maintains an imaginary cursor, GCPP (Graphics Current Position Pointer), which points to a particular position (x, y coordinates) within the bit map from which all relative coordinates are calculated. The GCPP is updated at the end of each drawing command.

The following commands are used to define the current bit map and attributes and set the Graphics Current Position Pointer:

- DEF_BIT_MAP - Define Bit Map
- DEF_CLIP_RECT - Define Clip Rectangle (see 2.4)
- DEF_COLORS - Define Colors
- DEF_TEXTURE - Define Texture
- DEF_LOGICAL_OP - Define Logical Operation (see 2.6)
- DEF_CHAR_SET - Define Character Set
- DEF_CHAR_ORIENT - Define Character Orientation
- DEF_CHAR_SPACE - Define Inter Character Spacing
- ABS_MOV - Absolute Move GCPP
- REL_MOV - Relative Move GCPP
- ENTER_PICK - Enter Pick Mode
- EXIT_PICK - Exit Pick Mode

2.3.3 Geometric Commands

These commands allow the 82786 to draw points, lines, and arcs in a variety of ways:

- POINT - Draw Point
- INCR_POINT - Draw Incremental Points
- CIRCLE - Draw Circle
- LINE - Draw Line
- RECT - Draw Rectangle
- POLYLINE - Draw Polyline
- POLYGON - Draw Polygon
- ARC - Draw Arc
- SCAN_LINES - Draw Series of Horizontal Lines

2.3.4 Bit Block Transfer (BitBlt) Commands

These commands allow rectangular image pieces to be combined from piece of bit-map memory to another. The Graphics Processor automatically inserts the new data in the correct order in the destination so that each line of pixels remains consecutive for both existing and new data.

- BIT_BLT - Bit Block Transfer within bit map
- BIT_BLT_M - Bit Block Transfer between bit maps

Each command specifies the origin of the source rectangle as well as the height and width. The destination origin is the GCPP coordinates. For BitBlts between bit maps, the destination is the active bit map and the

memory address of the source origin and source bit map size is specified. BitBlts between bit maps can only use bit maps with the same number of bits per pixel.

2.3.5 Character Command

This command allows character fonts stored in memory in pixel form to be drawn into the bit map by an application using character codes such as ASCII:

- CHAR - Draw Character String

The CHAR command defines transparency/opaque-ness for a character string, the pointer for the character string, and the number of character in the string. The pixel contents of the character to be drawn may be located anywhere in the memory space of the 82786 and accessed with either an 8- or 16-bit reference to the specific character. The string range specifies the 8- or 16-bit references for each character to be drawn. Section 2.7 discusses the use of character fonts.

Standard character fonts can be drawn flexibly because path and rotation are defined with a DEF_CHAR_ORIENT command and inter-character spacing is defined with a DEF_CHAR_SPACE command. This permits the variable spacing of text, direction of text, and rotation of characters to be specified by the application without altering the font. Simple one-bit per pixel character font definitions can be used in color applications because foreground and background colors are specified by the DEF_COLOR command and the necessary bits are written for each pixel during the drawing process.

2.4 DRAWING ATTRIBUTES

A drawing operation refers to the act of modifying pixels within a bit map during the execution of the GP commands. All drawing that the GP performs (including lines, arcs, characters and BitBlts) is subject (with exceptions noted) to six attributes which should be defined before any drawing commands are executed. The attributes are:

1. Pixel Plane Mask;
2. Logical Operation;
3. Clipping Rectangle;
4. Foreground and Background color (not applicable to BitBlt);
5. Transparent or Opaque mode (not applicable to BitBlt);
6. Pattern mask of 16 bits (not applicable to BitBlt or characters).

The pixel plane mask is helpful in restricting the graphics primitives to update a subset of the bits per pixel.

This permits one set of drawings to exist in one or more colors and allow other text or graphics information to reside in different color bits of the same bit map. Raster operations can be used to combine existing pixel information in the bit map with the new pixel information generated as a result of the new drawing operation, such as displaying only the overlapping regions of two shapes. The clipping rectangle limits the effects of drawing operations to a subset of the bit map.

Foreground and background colors set the two colors drawn by all drawing operations (if both are needed). The transparent mode draws only the foreground color into the bit map (for dotted lines or characters) and leaves the pixels between the dots or characters unchanged. The opaque mode draws the foreground color and fills in the background color between the dots or characters. The pattern defined in the mask cause a logical operation with drawing commands and permit dotted and dashed lines, arcs, and other shapes. DEF_PATTERN sets transparent/ opaque for drawing operations other than character, which is defined in CHAR.

2.5 CLIPPING

The clipping rectangle is used to prevent drawing outside a specified rectangular region. The clipping rectangle can be any rectangle within a bit map or the entire bit map. Pixels are not drawn beyond the limits of the clipping rectangle and characters which would be partially clipped are not drawn at all.

In "pick mode," the clipping rectangle is used to perform a different function. The clipping rectangle may be controlled by software to support the selection of objects on the display with a pointing device. When in pick mode the drawing commands are executed but pixels are not updated in memory. Instead, a flag is set in a register if any of the pixels generated by the command lie within the clipping rectangle. In this way it is easy to set the clipping rectangle to correspond to the location of a graphics pointing device (such as a mouse) and re-process the Graphics Processor Command Block (GCMB) to find which drawing command corresponds to the selected area.

2.6 LOGICAL OPERATION

The logical operation is an attribute that applies to all subsequent pixel update operations (line, arc, character, BitBlt etc.). It is an operation which can logically combine the contents of separate bit-map locations to produce new bit-map patterns. All sixteen binary functions are permitted between both the source and destination.

- AND
- OR
- EXCLUSIVE-OR

Six combinations provided are commonly used:

- REPLACE destination with source
- REPLACE destination with complement of source
- SET all destination bits to 0
- SET all destination bits to 1
- REPLACE destination with complement of destination
- REPLACE destination with destination (NOP)

2.7 CHARACTER FONTS

The Graphics Processor supports an unlimited number of character fonts, that can reside anywhere in the 4 Megabyte address space. The character string to be written can be defined either as a string of bytes or as a string of words depending upon the type of font used. The active font type and upper and lower memory addresses of the font to be used are set via the DEF_CHAR_SET command.

Each character in the character font has an independently programmable size of up to 16 by 16 pixels, allowing individual characters to have different sizes for proportional spacing. Each character resides in a block containing $n + 1$ words of memory where n is the pixel height of the character. The first word contains fourteen bits to define the height and width of the character. The remaining two bits specify if the following character should be an overstrike or if the character exceeds sixteen pixels in either dimension to cause a software trap. Overstriking is useful for efficient implementation of underline and accents, and prevents updating the GCCP after the character is drawn.

For larger characters than 16 by 16, the trap bit in the font can cause an interrupt to the CPU so that software can specially process that character such as a BitBlt. The perception of larger characters than 16 by 16 can also be created by dividing characters into subsets such as quadrants, and executing multiple character drawing commands. Software use of the DEF_CHAR_SPACE command supports negative inter-character spacing to permit kerning, such as for italic fonts.

The byte or word strings used as parameters for the CHAR command are used in conjunction with the 22-bit pointer defined in a register by the DEF_CHAR_SET command. Use of 16-bit, or word-mode, characters causes an add between the 22-bit pointer and the 16-bit reference value to access the starting address of the specific character. Because maximum character block size is seventeen words of data, approximately four thousand characters may be contained in one 16-bit font (worst case). Supplementary software in the form of a look-up table can be used to access as many as 65,000 characters in a single font. BitBlt can move characters of unlimited size.

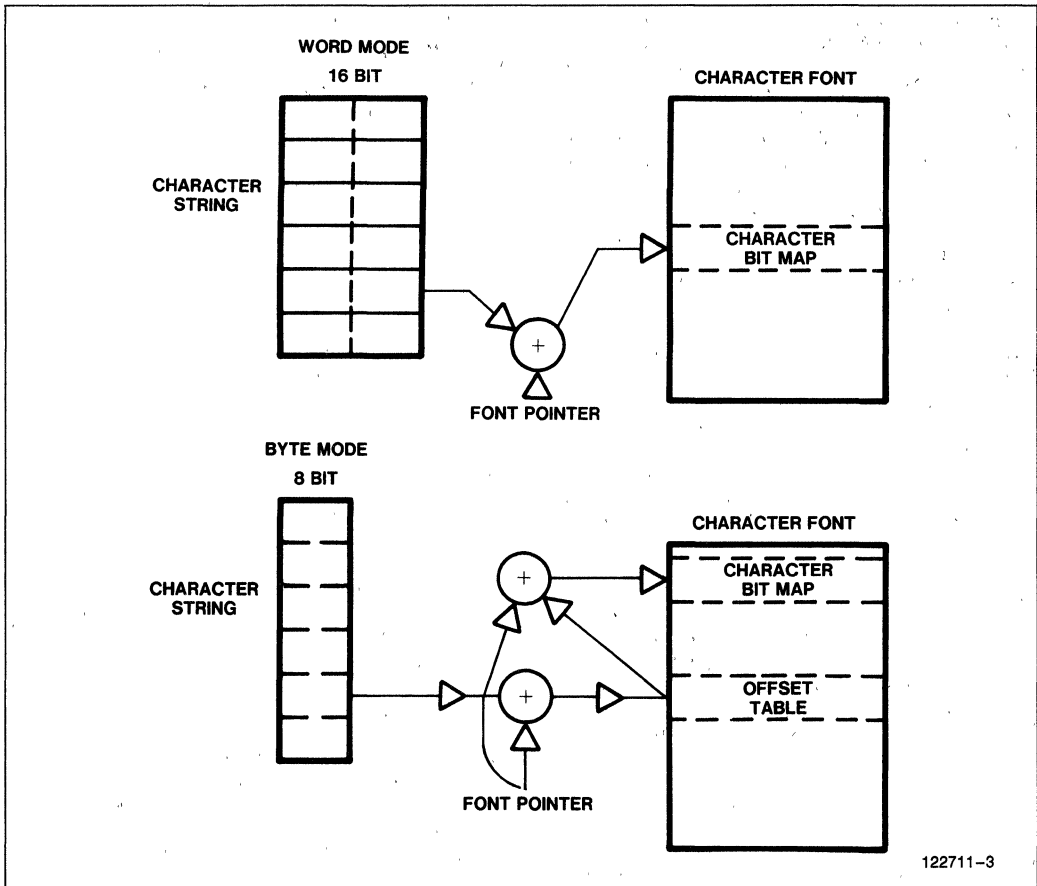


Figure 2-1. Word and Byte Mode

Use of byte-mode permits eight bit references to characters. This is important to permit existing software using ASCII and EBCDIC to be converted to 82786-based systems. 256 words of the font are reserved for a look-up table. Adding the 8-bit string parameter to the font pointer determines the word for the specific character within this table. The word is then added to the pointer to locate the character information in the font. Byte-mode permits only 256 characters in each 8-bit font. Figure 2-1 shows a description of word and byte mode.

application needs to change bit-map contents or support some special function such as picking. The general format of a command is shown in Figure 2-2.

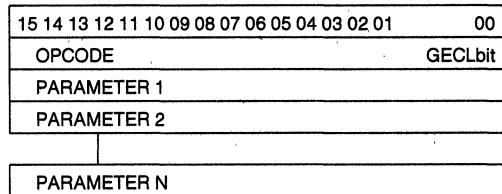


Figure 2-2. Command Format

2.8 GRAPHICS PROCESSOR COMMAND LIST FORMAT

The Graphics Processor executes a sequence of commands resident in memory and runs only when an ap-

Each opcode resides in the high byte of the word with a Graphics ECL (End of Command List) bit in the least significant bit of the low byte and followed by a varying number of parameters in consecutive words. The Graphics Processor tests the ECL of each command and sends the Graphics Processor into Poll Mode when set to "1" for any opcode. Poll mode halts the Graphics Processor until a LINK command and upper-

and lower-memory values for a link address are loaded into three reserved registers. The Graphics Processor then begins executing a new linked-list of commands starting at the specified address when the ECL bit with the LINK command in the register is reset to 0.

An example of a Graphics Command Block using linked-lists is shown in Figure 2-3.

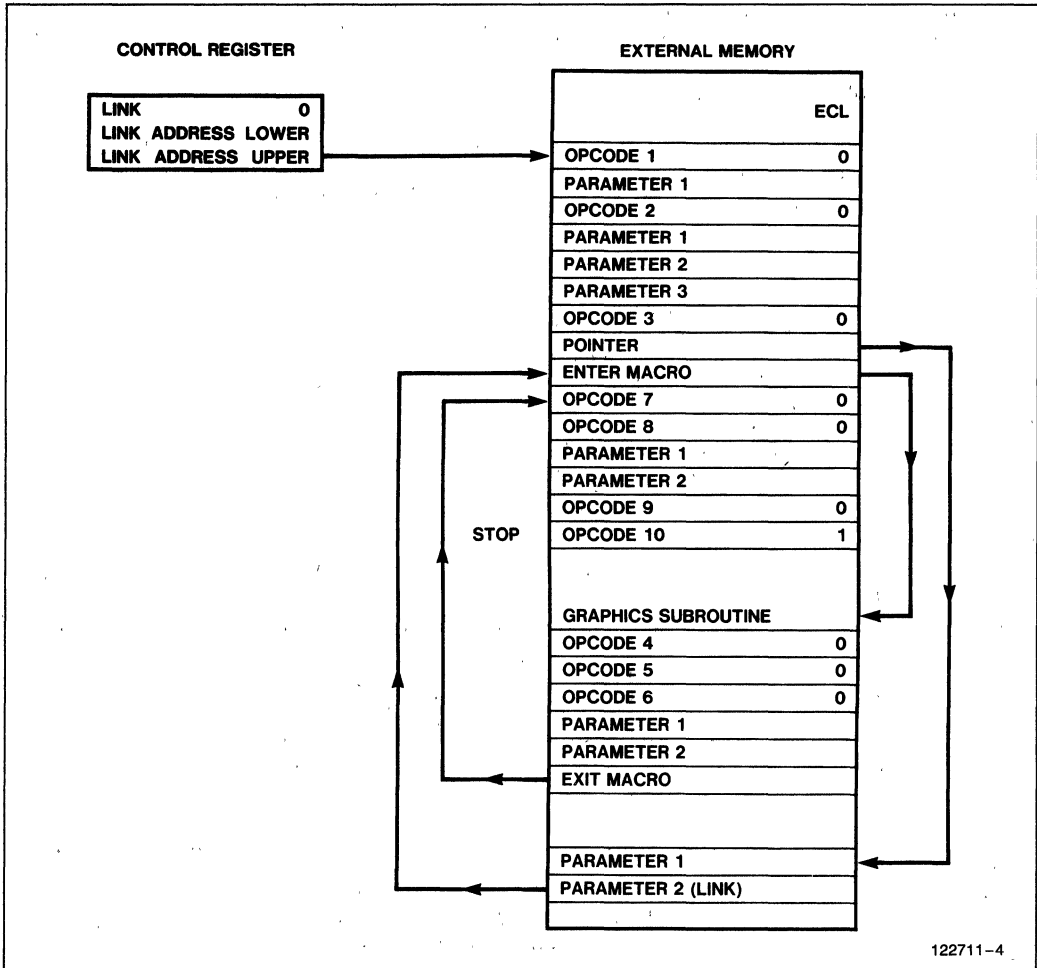


Figure 2-3. Graphics Processor Command Block

CHAPTER 3 DISPLAY PROCESSOR

3.1 OVERVIEW

The Display Processor has five main functions in generating the display contents for output:

1. To retrieve the memory contents of selected bit maps and output corresponding pixels into separate regions on the display screen (windows);
2. To permit selected portions of bit maps to be magnified on the display (zooming) horizontally and/or vertically via pixel replication;
3. To provide a "pointing symbol" (cursor);
4. To generate control and video data signals to the display hardware;
5. Load the shift registers of VRAMs.

Control of the Display Processor is programmed via on-chip registers. Content of the display is dynamically altered by the application (or system software) without causing unacceptable display blinking. Using memory-mapped CPU alteration of parameters, the DP will load the register set with the new parameters during vertical retrace. By altering the registers to point to a new display list, the change of display lists can occur between refresh cycles.

3.2 WINDOWS

Windows contain the portions of bit maps which are output by the Display Processor. Up to 16 window segments or tiles can be displayed on the same scan line of the CRT, while there may be as many windows vertically as the number of scan lines.

The 82786 treats the screen as divided into horizontal strips (Figure 3-1) of arbitrary width, where the horizontal format of window tiles across the strip remains constant for the whole strip. This divides the region into rectilinear areas, which are easy to manage. By combining strips, overlapping windows can easily be obtained.

Windows may essentially be arbitrarily shaped (circular, irregular, etc.) because a new strip may be defined every display line, similar to the format shown in Figure 3-2.

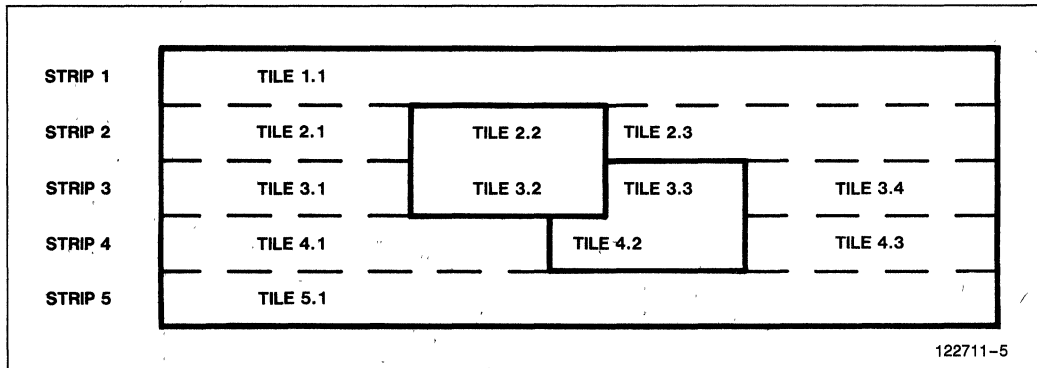


Figure 3-1. Sample Display Implementation of Two Overlapping Windows

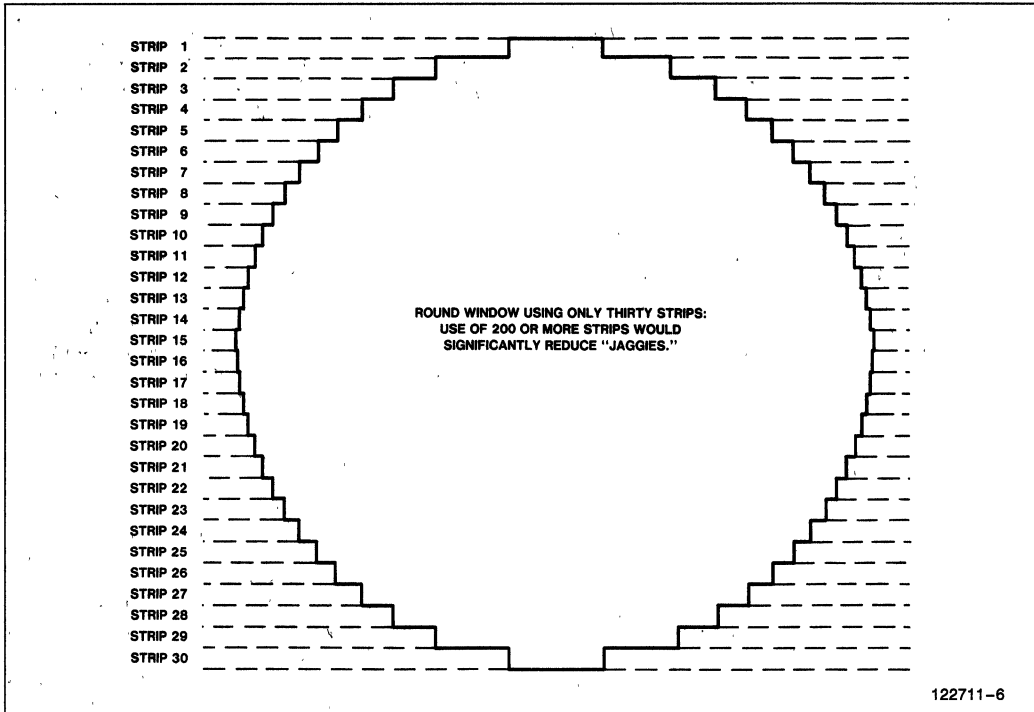


Figure 3-2. Sample Display of Irregular Window

The information needed for the Display Processor is contained in strip descriptor tables, each made up of a header and one or more tile descriptors. The header contains:

- the number of lines in the strip minus 1;
- the number of tiles in the strip minus 1;
- upper and lower addresses of the next strip descriptor

Each tile descriptor (which is consecutive in memory) contains:

1. the width of the bit map from which the window is being retrieved (in words);
2. the start address of the bit-map data to be displayed (word in memory and first bit location);
3. the number of words to fetch for the tile;
4. the first and last bit locations of the bit-map data to be displayed;

5. the number of bits per pixel;
6. four bits to indicate border presence for top, bottom, left, and right edges (1 indicates show border, 0 indicates show bit-map for those pixels);
7. window status information which can be used to select color palettes or other attributes (2 bits);
8. two bits to indicate bit-map configuration, which can be byte rather than word-oriented with byte order switched and if bit-map is non-linear (for PC compatibility);
9. bit to indicate if window is to be zoomed by pixel replication of the bit-map data;
10. bit to indicate if tile uses field background data.

A one-pixel border can be displayed on any or all sides of each window tile. This border color is defined in the Display Control FldColor Register, an 8-bit register which is the same user-definable color for all windows. Borders may be turned on or off for individual tiles.

In the absence of windows, the field background color is displayed. This single color is definable by the user in the `FldColor` Register. The use of background on the display minimizes system bandwidth because data is only fetched for windows and not for background, and thus saves bit-map memory.

The Display Processor provides padding bits when bit maps to be displayed have fewer bits/pixel than the hardware display, with no performance decrease. This allows windows of various bits/pixel to be shown simultaneously on the same display. The user programs the desired 8-bit color patterns into three registers, one serving to map each of 1-, 2-, and 4-bits per pixel information into full colors on the display.

All video output from the 82786 can be defined to begin and end at any pixel (except when in accelerated mode using external shift logic). This includes the positioning of every window and the cursor.

The Display Processor command list is controlled by the CPU. The double-word location of the first strip descriptor block is located in a register. The locations of subsequent strip descriptor tables are based upon a linked-list architecture and are provided in the preceding descriptor table. This descriptor linked-list needs only to be updated by the CPU when the window arrangement on the screen changes. New strips and tiles are easily inserted into the display list by simply modifying the linked-list pointers of the preceding strips or tiles.

The use of redundant lists is possible because the description of a typical display is memory-efficient and requires only about 1,000 bytes. This would permit the CPU to alter the contents of one list while the second is being used to control Display Processor. When the creation of the new list is complete, the registers pointing to the first strip descriptor table may be switched to the locations for the new list during vertical retrace. This permits the application to alter the display list without causing temporary swimming or blinking of the display.

3.3 CURSOR

The Display Processor supports a single hardware cursor which may be up to 16 x 16 pixels. This cursor may be positioned by the user anywhere on the screen. The cursor may be defined to be transparent or opaque, and may be either a block cursor or a cross-hair cursor one pixel across that stretches the width and height of the screen. The color of the cursor is user-definable, as is the block cursor's pattern. Eight bits of register memory define the color and sixteen 16-bit words of register

define the pattern, which is then padded with the cursor color register. Support for a blinking cursor is provided with a register for `CURSOR_ON` which can be toggled by the CPU as often as necessary to cause an appropriate blink rate. Multiple cursors can be simulated by drawing them in software, especially using `BitBlt`.

3.4 ZOOM

The Display Processor allows selected windows to be zoomed (using pixel replication) up to 64 times horizontally and vertically (independently, in steps of one). The setting of the zoom bit in the tile descriptor table causes replication of the pixels in memory according to horizontal and vertical scaling factors contained in `Zoom X` and `Zoom Y` Registers.

3.5 VIDEO INTERFACE

Eight parallel video data output lines provide video output which may be used as eight bits pixel on the CRT, or externally shifted to boost maximum display resolution. The dot rate output is controlled by an independent video clock which may be up to 25 MHz. Horizontal signals are programmable from 1 to 4096 cycles of the video clock and vertical sync signals from 1 to 4096 scan lines. Use of eight external video data pins allow up to 256 different colors to be directly displayed. Other CRT control lines provided by the Display Processor are `VSYNC`, `HSYNC`, `BLANK`.

Several 82786s can be used together for higher performance graphics. For multiple 82786 Systems, one 82786 acts as a master generating `VSYNC` and `HSYNC`, and the other 82786s act as slaves using the master sync signals for timing by using their own `VSYNC` and `HSYNC` as inputs. Each 82786 has its own bit-map memory with separate Graphics Processor Command Blocks (GCMBs) to form a bit-plane architecture, but use the same display list. The `BLANK` signal is not used by slave 82786s.

External color palettes are supported, and, by use of the two window status lines, the application may select one of four color combinations for any window. This supports a maximum of 1024 simultaneous colors per frame. The palette may be programmed by latching the default video data when the `BLANK` pin is high. The Display Processor can support non-interlaced, and interlaced-sync displays. Selection of the interlacing, control to support external shifting of the video data, default video data contents, and slave/master status for each 82786 are controlled via dedicated registers. The 82786 may be synchronized to an external source ("Gen-Locking").

CHAPTER 4 82786 SYSTEMS

4.1 TYPICAL SYSTEM CONFIGURATIONS

The 82786 can be used in many different configurations, each providing cost and performance appropriate for different applications and markets.

Three typical applications in which the 82786 could be used are:

1. Low-priced personal computer (Figure 4-1);
2. Multitasking office workstation (Figure 4-2);

3. High-performance workstation for processing-intensive, high-resolution engineering applications (Figure 4-3).

4.2 DRAM/VRAM CONTROL

The DRAM/VRAM controller on the 82786 supports an array of up to 32 memory chips without extra logic and up to a 4 megabyte address space. DRAMs supported have densities ranging from 8K to 1 megabit and

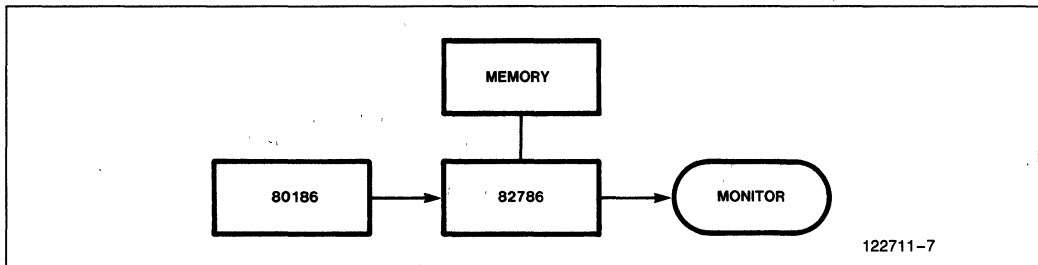


Figure 4-1. Low End Personal Computer

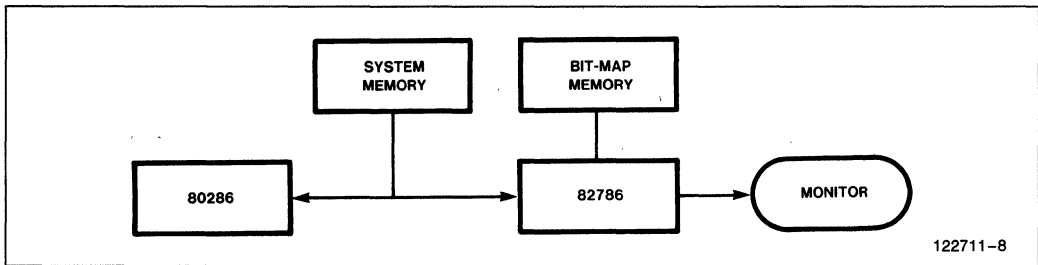


Figure 4-2. Desktop PC/Graphics Terminal

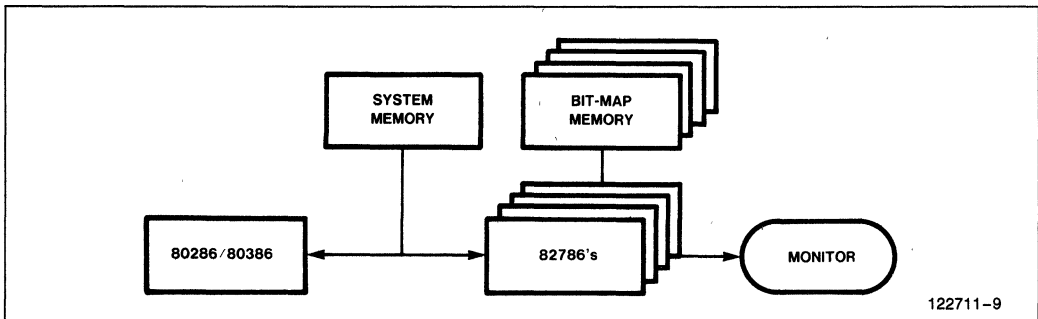


Figure 4-3. High End Workstation

organizations of x1, x4, or x8. The bandwidth of the memory system can be increased by interleaving memories and/or using the Ripplemode™ or static-column mode supported by Intel CHMOS DRAMs. Both interleaving and Ripplemode™ are completely handled on chip and require no extra external circuits. Use of static-column DRAMs requires one 74X373 latch per bank. Interleaving refers to the use of multiple DRAM banks with one set of memories receiving new CAS signals while the other outputs data. Table 4-1 shows memory burst-bandwidth for the different configurations at 10 MHz.

DRAM/VRAM refresh is done automatically by the DRAM/VRAM controller. The memory array can be accessed both by 82786 internal processors (GP, DP) and by external masters (CPUs) through the BIU. The 82786 DRAM/VRAM controller can be used to control system memory within its 4 megabyte address space, provided the target application can accept the decreased bandwidth of system memory. The portions of the address space dedicated to graphics and system memory are configured at initialization in the DRAM CONTROL REGISTER. Graphics memory is assumed to start at OH and continue up to the configuration limit. Memory addresses above this are used for system memory.

4.3 BUS INTERFACE

The Bus Interface Unit of the 82786 is designed to support all 8-, 16-, and 32-bit microprocessors from Intel, with optimization for the 80286. This permits the

82786 to run synchronously with the 80286, increasing throughput by eliminating wait states. A special 8-bit mode allows 82786 to also work with 8-bit data bus microprocessors. The 80386 makes interfacing to the 82786 possible. Interfacing to Intel CPUs is detailed in the Hardware Configurations Applications Note.

The bus interface allows slave access by the CPU to the graphics memory controlled through the 82786 DRAM/VRAM controller. This allows the CPU to update the Graphics Processor Command Block (GCMB) and the Display Processor descriptor lists in the graphics memory where maximum throughput can be supported. Low-end systems could use only a single memory shared by both the 82786 and CPU and use the 82786 DRAM/VRAM controller for this memory.

For performance reasons, many systems will have at least two sections of memory: the 82786 graphics memory (using the on-chip DRAM/VRAM controller) and the system memory. In this configuration, the 82786 can execute bus cycles on the system bus so the 82786 can access the CPU's own memory. This master mode is designed in accordance with the 80286 definitions. This configuration allows the best of both worlds, the system and graphics memories are split for performance reasons, but the split is transparent to the software for flexibility. Character fonts and graphic objects may be retrieved from disk and placed in system memory locations reserved for access by the 82786 using a virtual mode 80286 or 80386 configuration with appropriate system software.

Table 4-1. 82786 DRAM Bandwidths

	Pagemode DRAM	Ripplemode DRAM
Non Interleaving DRAM banks	10 Megabyte/sec (diagnostics or $640 \times 480 \times 2$)	20 Megabyte/sec ($640 \times 480 \times 4$ or $1K \times 1K \times 1$ noninterlaced)
Interleaving DRAM banks	20 Megabyte/sec ($640 \times 480 \times 4$ or $1K \times 1K \times 1$ noninterlaced)	40 Megabyte/sec ($2K \times 2K \times 1$ interlaced: $1K \times 2K \times 1$, $1K \times 1K \times 2$, $800 \times 600 \times 4$, $640 \times 480 \times 8$ noninterlaced)

CHAPTER 5 PACKAGE AND PIN DESCRIPTION

5.1 OVERVIEW

The 82786 is an eighty-eight pin component due to the large number of functions integrated within the device. It is available in both pin grid array and leaded chip carrier versions. The pinout of a pin grid array is shown in Figure 5-1 and a description of the pins is shown in Table 5-1.

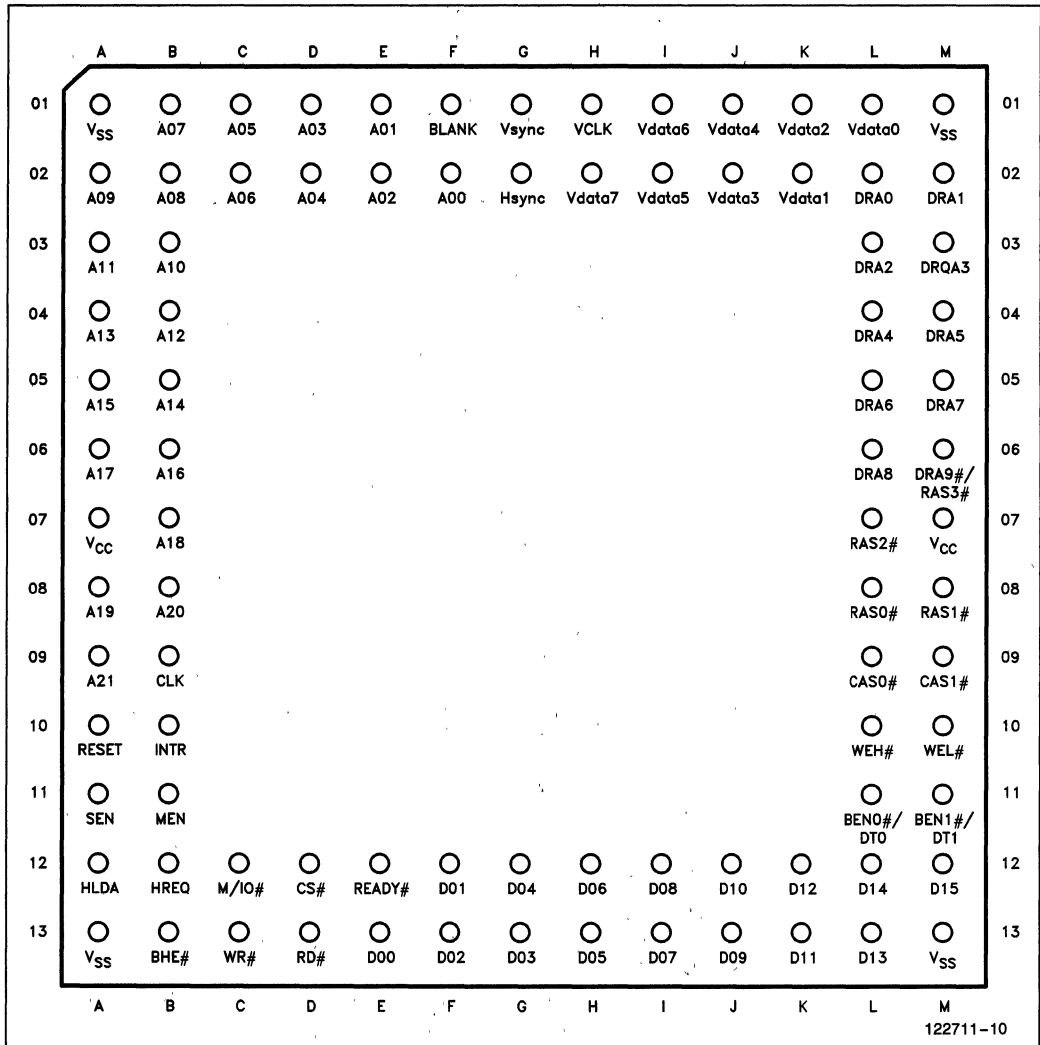


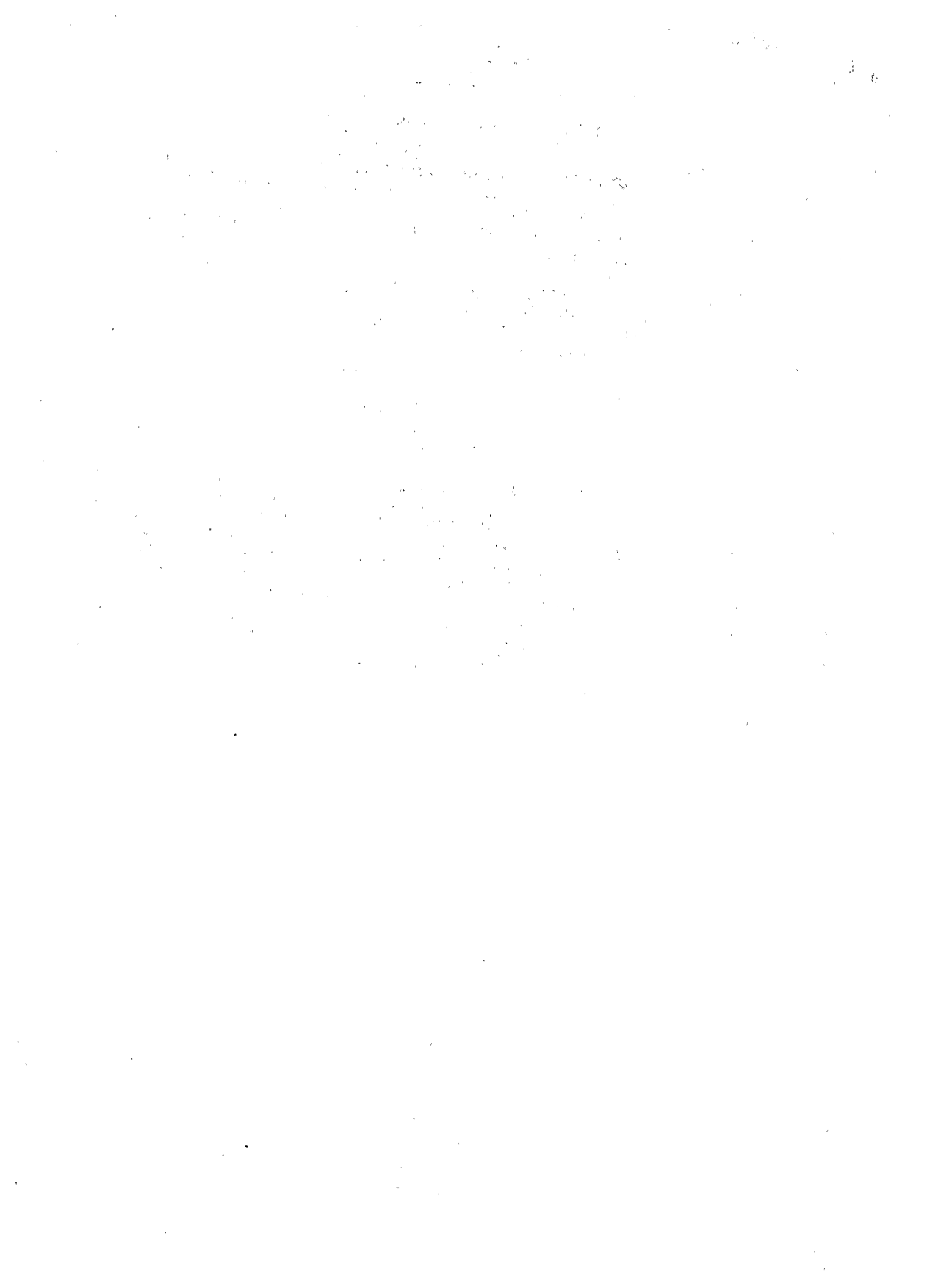
Figure 5-1. 82786 Pinout—Bottom View

Table 5-1. 82786 Pin Names and Descriptions

Symbol	Type	Description
A21-0	I/O	ADDRESS LINES FOR THE LOCAL BUS: Normally inputs for Slave Mode accesses of the 82786 supported DRAM array or internal memory or I/O mapped registers. Driven by the 82786, when it is the Local Bus Master.
D15-0	I/O	DATA BUS: For the 82786 DRAM/VRAM array and the Local Bus.
BHE #	I/O	BUS HIGH ENABLE: An input of the 82786 Slave Interface: driven LOW by the 82786 when it is Local Bus Master. Determines asynchronous vs. synchronous operation for RD #, WR # and HLDA inputs at the falling (trailing) edge of RESET. A HIGH state selects synchronous operation.
RD #	I/O	READ STROBE: An input of the 82786 Slave Interface: driven by the 82786 when it is Local Bus Master. Asynchronous vs. synchronous input determined by state of BHE # pin at falling RESET.
WR #	I/O	WRITE STROBE: An input of the 82786 Slave Interface: driven by the 82786 when it is Local Bus Master. Asynchronous vs. synchronous input determined by state of BHE # pin at falling RESET.
MIO	I/O	MEMORY / I/O INDICATION: An input of the 82786 Slave Interface: driven HIGH by the 82786 when it is the Local Bus Master. Selects 286 Status or Command Mode vs. 8086/186 Status Mode of the 82786 Slave Interface at the falling (trailing) edge of RESET. A LOW state selects the 286 Status or Command Mode.
CS #	I	CHIP SELECT: Slave Interface input qualifying the access.
MEN	O	MASTER ENABLE: Driven HIGH when the 82786 is in control of the Local Bus, (i.e. HLDA received in response to a 82786 HREQ). Used to steer the data path and select source of bus cycle status commands.
SEN	O	SLAVE ENABLE: Driven HIGH when 82786 is executing a Slave bus cycle for an external master on the Local Bus. Used to enable the data path and as a READY indication to the Local Bus Master.
READY #	I	SYNCHRONOUS INPUT: To the 82786 when executing Local Bus cycles. Identical to 80286 timing.
HREQ	O	HOLD REQUEST: Driven HIGH by the 82786 when an access is being made to the Local Bus by the Display or Graphics Processors. Remains HIGH until the 82786 no longer needs the Local Bus.
HLDA	I	HOLD ACKNOWLEDGE: Input in response to a HREQ output. Asynchronous vs. synchronous input determined by state of BHE # pin at falling RESET.
INTR	O	INTERRUPT: The logical OR of a Graphics Processor and Display Processor interrupt. Cleared with an access to the BIU Interrupt Register.
CAS0 #	O	COLUMN ADDRESS STROBE 0: Drives the CAS inputs of the even word DRAM/VRAM bank if interleaved; identical to CAS1 # if non-interleaved DRAM/VRAM. Capable of driving 16 DRAM CAS inputs.
CAS1 #	O	COLUMN ADDRESS STROBE 1: Drives the CAS inputs of the odd word DRAM bank if interleaved; identical to CAS0 # if non-interleaved DRAM. Capable of driving 16 DRAM CAS inputs.
RAS2-0 #	O	ROW ADDRESS STROBE: Drives the RAS input pins of up to 16 DRAMs. Drives the first three rows of both banks of DRAM/VRAM.
DRA9/ RAS3 #	O	MULTIPLEXED MOST SIGNIFICANT DRAM/VRAM ADDRESS LINE AND RAS3 #: Support of 1Mb DRAMs requires DRA9. When 1Mb DRAMs are used, four rows of DRAMs cannot be supported (RAS3 # unnecessary) due to 82786 addressing limit of 4 Mbytes being exceeded.
WEL #	O	WRITE ENABLE LOW BYTE: Active LOW strobe to the lower order byte of DRAM/VRAM.
WEH #	O	WRITE ENABLE HIGH BYTE: Active LOW strobe to the higher order byte of DRAM/VRAM.

Table 5-1. 82786 Pin Names and Descriptions (Continued)

Symbol	Type	Description
DRA8-0	O	MUTIPLEXED DRAM/VRAM ADDRESS: DRAM/VRAM row and column address are multiplexed on these lines. Capable of driving 32 DRAMs/VRAMs without buffers.
BEN1-0#	O	BANK ENABLE 1 AND 0: Enables the output of the DRAM array on to the 82786 data bus (D15-0). BEN1 # controls Bank 1. BEN0# controls Bank 0.
BLANK	I/O	OUTPUT USED TO BLANK THE DISPLAY AT PARTICULAR POSITIONS ON THE SCREEN: May also be configured as inputs to allow the 82786 to be synchronized with external sources.
VDATA7-0	O	VIDEO DATA OUTPUT.
VCLK	I	VIDEO CLOCK INPUT: used to drive the display section of the 82786. Its maximum frequency is 25 MHz.
HSYNC/ WST0	I/O	HORIZONTAL SYNC: Window status may be multiplexed on this pin. Can also be configured as input to allow the 82786 to be synchronized with external sources. Even as input, window status still is output when BLANK is low.
VSYNC/ WST1	I/O	VERTICAL SYNC: Window status can be multiplexed on this pin. Can also be configured as input to allow the 82786 to be synchronized with external sources. Even as input, window status still is output when BLANK is low.
RESET	I	RESET INPUT: internally synchronized. Halts all activity on the 82786 and brings it to defined state. The leading edge of RESET synchronizes the clock to PH1. The trailing edge latches the state of BHE# and MIO to establish the type of Slave Interface. It also latches RD# and WR# to set certain test modes.
CLK	I	DOUBLE FREQUENCY CLOCK OUTPUT: Clock input to which pin timings are referenced. 50% duty cycle.
V _{SS} , V _{CC}		4 V_{SS} AND 2 V_{CC} PINS.





UNITED STATES

**Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051**

JAPAN

**Intel Japan K.K.
5-6 Tokodai Toyosato-machi
Tsukuba-gun, Ibaraki-ken 300-26
Japan**

FRANCE

**Intel Paris
1 Rue Edison, BP 303
78054 Saint-Quentin en Yvelines
France**

UNITED KINGDOM

**Intel Corporation (U.K.) Ltd.
Piper's Way
Swindon
Wiltshire, England SN3 1RJ**

WEST GERMANY

**Intel Semiconductor GmbH
Seidlstrasse 27
D-8000 Munchen 2/
West Germany**

HONG KONG

**Intel Semiconductor Ltd.
1701 Connaught Centre
1 Connaught Road
Hong Kong**