TurbosPort 386 Portable Computer

Technical Reference Manual

Printed in U S A

595-4105-02
TM-3034

ZENITH | data systems

THE QUALITY GOES IN BEFORE THE NAME GOES ON

FEDERAL COMMUNICATIONS COMMISSION
RADIO FREQUENCY INTEREFERENCE STATEMENT

WARNING: As sold by the manufacturer, the equipment described in this manual has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference with radio and TV reception.

NOTE: In order to meet Class B emission limits, the I/O cables that interconnect between the computer and any peripheral (such as a printer, external modem, etc.) must be shielded.

USER MODIFICATIONS

The user is responsible for any interference with radio or TV reception caused by a user-modified computer. The manufacturer supplied this manual for the sole purpose of providing technical information concerning its product(s). The user is responsible for any user-modifications made to the software, firmware, or hardware that are the result of information supplied in this manual.

The manufacturer reserves the right to make changes at any time to the design of the product(s) without obligation to update existing product(s), including the information in this manual.

Zenith Data Systems Corporation
St. Joseph, Michigan 49085

Contents

Contents

Contents

Contents

Contents

Contents

Figures

Contents

Contents

Tables

Contents

Contents

Contents

Part I
Introduction

Chapter 1

Introduction and General Information

The High-Performance TurbosPort 386, illustrated in Figure 1-1, offers increased computing power while maintaining compatibility with earlier PC's. Some of the features of this computer are:

- An Intel 80386 microprocessor operating at 12 or 6 Mhz
- Optional support for the 80387 coprocessor
- A 79-key keyboard with an integrated numeric keypad that can emulate a 101-key advanced keyboard
- 2M of system memory
- Optional memory expansion up to 3M
- Internal hardware support for EMS memory
- CGA video support
- High-density 3.5-inch floppy disk drive and 40M hard disk drive
- 640 x 400 pixel black-on-white flat panel liquid crystal display
- Fluorescent backlighting.

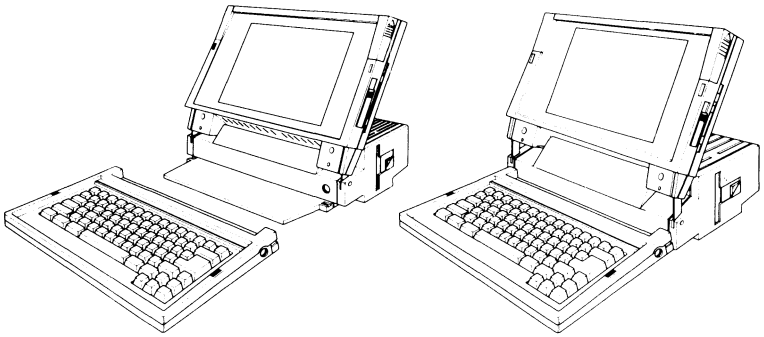


Figure 1-1. TurbosPort 386 Portable Computer

Manual Description

This manual contains information on computer operation and programming methods. It is a companion manual for the *MS-DOS Programmer's Utility Pack* (CB-3163-30). Refer to the Owner's Manual for this computer for explanations of commonly used acronyms and terms.

This technical manual contains the following parts, plus an appendix, and an index:

- Part I contains an introduction to the computer, including information on installation, system configuration, operation, and disassembly.

- Part II describes system programming. This includes the Monitor program and the software-based interrupts common to PC-compatible computers.

- Part III provides information on the base configuration and the hardware used to implement that configuration. This includes a discussion of each major programmable integrated circuit in the computer. Part III also includes a chapter on common problems and how to resolve them.

- The Appendix contains the instruction sets for the CPU and coprocessor.

Related Publications

Several available publications contain information related to the topics discussed in this manual. This material may be helpful in increasing your knowledge and understanding of this computer.

TurbosPort 386 Portable Computer Owner's Manual **(595-4063)** — This manual describes the installation and operation of the computer from a user's point of view. It also includes a brief discussion of the hardware, configuration, and maintenance.

Intel Introduction to the 80386 — This manual, published by the Intel Corporation, provides an overview and introduction to the 80386 microprocessor. The Intel order number for this publication is 231252.

***Intel Microsystems Component Handbook*, Vol. 1 & 2** — These manuals, published by Intel Corporation, provide information on the architecture and programming of the 80287 and 80387 coprocessors. They also contain information relating to system support circuitry and microprocessors. The Intel order number for this set is 230843-004.

Intel 80386 Programmer's Reference Manual — This manual, published by Intel Corporation, contains detailed information on the architecture, instruction set, and programming procedures required for the 80386 microprocessor. The Intel order number for this manual is 230985-001.

Intel 80386 Hardware Reference Manual — This manual, published by the Intel Corporation, contains detailed descriptions of the architecture and operation of the 80386 microprocessor. Device timing diagrams and pinouts are also included in this manual. The Intel order number for this publication is 231732-001.

Lotus/Intel Expanded Memory Device Interface Specification — This publication is available directly from Intel Corporation. This specification details the software and hardware interface for EMS memory systems. Intel Technical Support provides a toll free number, 1-800-538-3373, that you may call to obtain this specification.

Intel publications may be obtained by writing to the following address:

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

Intel also provides a toll-free number (1-800-548-4725) for literature requests.

MS-DOS Programmer's Utility Pack **(CB-3163-30)** — This software package, produced by Microsoft and Zenith Data Systems, contains information and software for writing and assembling MS-DOS and native 8086, 8088, 80286, and 80386 assembly language programs.

Recommended Environment

Most computers will operate over a wide range of environmental conditions; temperature and humidity most affect the way a computer operates. In general, always be aware of the published specifications for the equipment you are trying to use. Unusual applications may require you to contact the manufacturer for information. The TurbosPort 386 is designed to meet working conditions in the average business office, at home, or "on the road." Use these guidelines to maintain a proper operating environment for your computer:

- Keep the computer away from sources of heat, including direct sunlight.

- Keep the computer away from moisture sources that might cause liquid to come into contact with the computer. Do not allow drinks or other liquids in the work area.

- Place the computer in an area that allows free air flow around the computer. Restricted ventilation around the computer can result in overheating problems.

- Computers are extremely sensitive to dust and dirt particles. Consider using protective covers whenever the computer is not in operation.

- Computers and floppy disks are also sensitive to electromagnetic radiation. The work area should be free of potential sources of high electromagnetic emissions. Keep floppy disks at least two feet from telephones and associated wiring.

- Computers can be sensitive to mechanical shock and external vibrations. Place the computer on a secure work surface that is free from vibration. Always place dot-matrix and daisy wheel printers on separate printer stands. These devices are vibration sources and it is best to isolate them from the computer.

Memory Expansion

The basic computer comes with 2M of installed memory. The maximum internal memory for the computer is 3M. Additional expansion memory can be added by installing optional EMS memory cards in an external expansion chassis.

Coprocessors

Coprocessors are integrated circuits that increase performance in a specific area. An optional math coprocessor is available for this computer: the 80387. Intel refers to this device as a "numeric processor extension." This term, and the term "coprocessor" refer to the same type of device in this manual. This device provides increased performance on computation-intensive operations.

Video Options

The basic computer provides direct support for CGA video operation. This enhanced level video system provides screen resolution of 640 x 400 pixels double scan (640 x 200) with up to eight gray scales selectable from eight palettes. By attaching an external RGBI color monitor to the video connector, 16 colors from a palette of 64 are available. Other video options are available by installing optional video cards in the external expansion chassis.

Mass Storage

The term "mass storage" refers to any means of storing large amounts of data. The basic computer provides one 3.5-inch microfloppy disk drive with a formatted capacity of 1.4M on high-density disks or 720K on double-density disks and one 3.5-inch hard disk drive with a formatted capacity of 40M.

Specifications

CPU	
Processor:	80386, CMOS technology.
Type:	32-bit internal/external.
Clock speed:	12 MHz or 6 MHz, keyboard and program selectable.
Memory:	2M dynamic RAM, expandable to 3M. Hardware support for EMS memory.
Display:	Fluorescent backlit, black-on-white, liquid crystal display. Text mode: 80 x 25 characters; 640 x 400 pixels. Graphics mode: double scan 640 x 200 pixels. PC-compatible in normal text and graphics modes.
Sound:	Miniature transducer.
Input/output	
Serial port:	Asynchronous serial RS-232C port (AT- style DB-9 connector). One start bit; 7- or 8-bit word length; one or two stop bits; selectable baud rates of 110, 150, 300, 600, 1200, 2400, 4800, or 9600 baud; RD, CTS, DSR, CD signals recognized; TD, RTS, DTR control signals generated; half- or full-duplex operation.

Parallel port:	Centronics-type parallel output port (DB-25 connector).
Video:	RGB (color) video with intensity signals from a single 15-pin connector.
Disk drives:	3.5-inch double-sided, high-density microfloppy disk drive, 1.4M formatted capacity per drive, 135 tpi, nine sectors per track. Write-protection recognized.
	3.5-inch hard disk drive, 40M capacity.
Keyboard:	79 keys. 101-key keyboard compatibility maintained by using multiple keys to duplicate keypad and special function key operation. Duplicate CTRL, ALT, and SHIFT keys, 4-key cursor control pad. CAPS LOCK, SCROLL LOCK, PAD LOCK, and NUM LOCK keys use status LEDs.
Power requirements	
AC adapter/charger:	Input: 95 to 132 VAC, 48 to 64 Hz. Output: 25 to 29 VDC, 30 watts continuous, 500 mVpp maximum ripple.
Battery power:	24 V (2.2 AHr) removable battery pack with overcharge and short circuit protection. Battery life is 1000 charge/discharge cycles.

Environment

Operating:	60° - 85° F (16° - 30° C) at 20% - 80% relative humidity (non-condensing).
Storage:	–40° - +125° F (–40° - +51° C) at 20% - 80% relative humidity (non-condensing).
Dimensions:	13.25" wide x 14.81" deep x 4.75" high (33.66 cm x 37.62 cm x 12.07 cm) with display closed.
Weight:	14.98 lbs. (6.74 kg) without battery, 17.34 lbs (7.80 kg) with battery.

Chapter 2

Installation

This chapter discusses connecting the computer to peripherals and power in preparation for use. A flat-blade screwdriver is the only tool required.

Operating Environment

- The room temperature should be between 60° and 85° F (16° to 30° C).
- The relative humidity should be between 20% and 80% (non-condensing).
- Locate the computer away from heat sources such as radiators and heaters. Allow six inches of unobstructed space at the back and sides of the computer to prevent overheating.
- Choose a location where lights or windows will not cause reflections on the LCD.
- When using your computer with the AC adapter, be sure you use a properly rated and easily accessible power source.

Unpacking and Setting Up

1. Carefully unpack the computer and keyboard and place them on the work surface. The operating system documentation and disks are packed on top of the computer.
2. Allow six inches of open space around the back and sides of your computer.

3. Locate the computer power cord and AC adapter. Plug the appropriate end of the cord into the connector on the AC adapter and then connect the adapter to the computer, as shown in Figure 2-1.

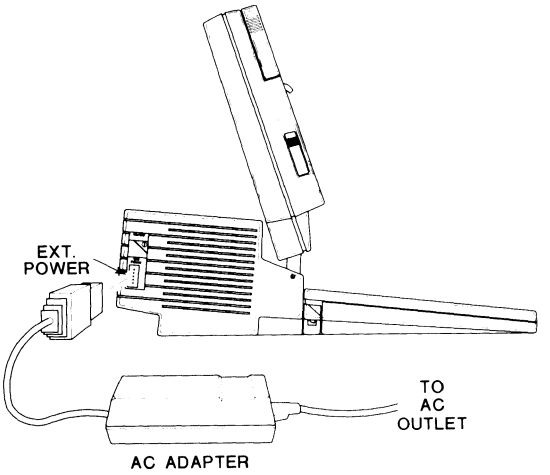


Figure 2-1. AC Adapter Connection

Connecting Peripherals

If you want to connect any peripherals (such as an external monitor, printer or modem) to your computer, connect them now. Figure 2-2 shows the back panel of your computer.

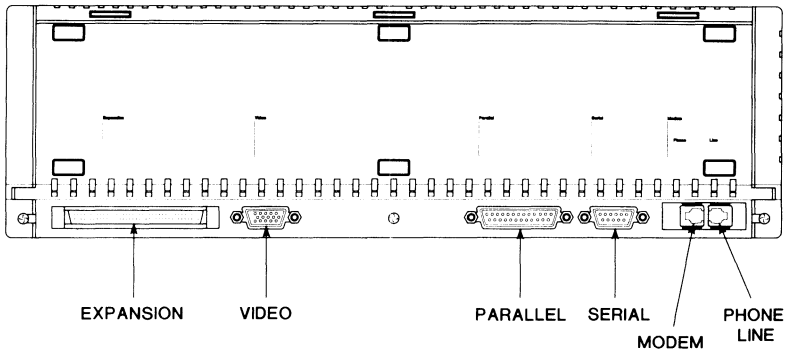
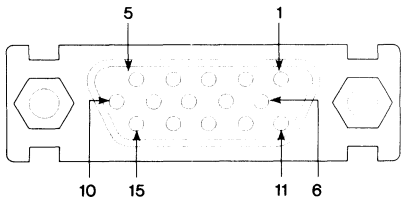


Figure 2-2. The Back Panel

Your computer can display information on the built-in LCD or on an RGBI color monitor. Video signals are available at the 15-pin D-type video connector. Table 2-1 provides the pin definitions for the video connector.

Table 2-1. Video Connector Pinout

PIN	SIGNAL	PIN	SIGNAL
1	Red	9	Not connected
2	Green	10	Ground
3	Blue	11	Reserved
4	Reserved	12	Intesity
5	Ground	13	Horizontal sync
6	Reserved	14	Vertical sync
7	Reserved	15	Reserved
8	Reserved		



Your computer is equipped with a serial port and a parallel port. The serial port connector is a 9-pin D-type configured as an RS-232C communications port. The parallel port connector is a 25-pin D-type configured as a Centronics-type printer port. Table 2-2 provides pinout information for the serial port and Table 2-3 provides pinout information for the parallel port.

Table 2-2. Serial Connector Pinout

PIN	SIGNAL	PIN	SIGNAL
1	Carrier detect	6	Data set ready
2	Receive data	7	Request to send
3	Transmit data	8	Clear to send
4	Data terminal ready	9	Ring indicator
5	Ground		

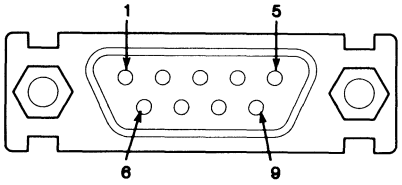
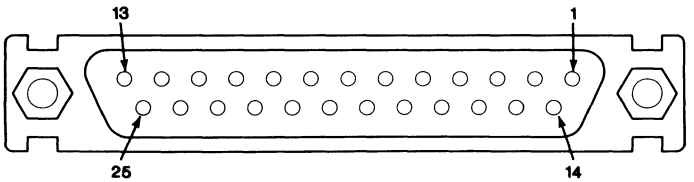


Table 2-3. Parallel Connector Pinout

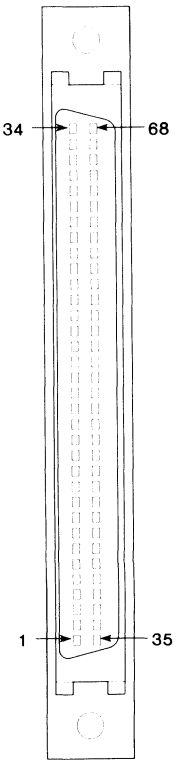
PIN	SIGNAL	PIN	SIGNAL
1	Strobe	10	Acknowledge
2	Data bit 0	11	Busy
3	Data bit 1	12	Page end
4	Data bit 2	13	Select
5	Data bit 3	14	Auto feed
6	Data bit 4	15	Error
7	Data bit 5	16	Initialize printer
8	Data bit 6	17	Select input
9	Data bit 7	18-25	Ground



Your computer is also equipped with an external expansion connector. This connector is a 68-pin male connector and provides access to an XT-compatible bus in the computer. Table 2-4 provides pinout information for the expansion connector.

Table 2-4. Expansion Connector Pinout

PIN	SIGNAL	PIN	SIGNAL
1	I/O channel check	35	Data bit 7
2	Reset	36	Data bit 6
3	Ground	37	Data bit 5
4	Interrupt request 2	38	Data bit 4
5	DMA request 2	39	Data bit 3
6	Interrupt request 7	40	Ground
7	Terminal count	41	Data bit 2
8	ZX bus enable	42	Data bit 1
9	I/O channel ready	43	Data bit 0
10	Ground	44	Address bit 19
11	Address enable	45	Address bit 18
12	Memory write	46	Ground
13	Memory read	47	Address bit 17
14	I/O write	48	Address bit 16
15	Ground	49	Address bit 15
16	I/O read	50	Address bit 14
17	DMA acknowledge 3	51	Address bit 13
18	DMA request 3	52	Ground
19	DMA acknowledge 1	53	Address bit 12
20	DMA request 1	54	Address bit 11
21	Ground	55	Address bit 10
22	DMA acknowledge 0/Refresh	56	Address bit 9
23	Write configuration	57	Ground
24	Clock	58	Ground
25	Data buffer enable	59	Address bit 8
26	Interrupt request 6	60	Address bit 7
27	Ground	61	Address bit 6
28	Interrupt request 5	62	Address bit 5
29	Interrupt request 4	63	Address bit 4
30	Interrupt request 3	64	Ground
31	DMA acknowledge 2	65	Address bit 3
32	Data buffer direction	66	Address bit 2
33	Ground	67	Address bit 1
34	Address latch enable	68	Address bit 0



Opening the Computer

Position the computer so that the front is facing you and you can easily reach the disk drive to insert and remove floppy disks. You should also be able to reach the power switch on the left side of the computer.

To open the computer when it is first removed from the packing material, simply lift up on the lid. When the keyboard is attached, follow the instructions listed below.

1. With the handle inserted fully into the computer, push the latch on each side toward the front, as shown in Figure 2-3.

2. Lift the top up from the front as illustrated.

3. The top contains the LCD display. Adjust the screen for the most comfortable viewing angle.

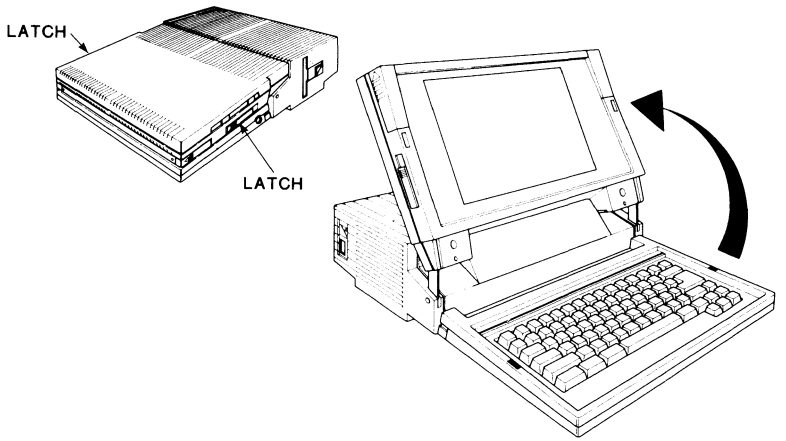


Figure 2-3. Opening the Computer

Keyboard

The keyboard contains 79 keys and emulates a 101-key advanced keyboard. (For more information about the keys and the modes of operation, refer to Chapter 5.)

1. Connect the keyboard cable to the keyboard connector, as shown in Figure 2-4.

2. Place the keyboard on the tongue of the computer and slide it back until it locks into place.

3. Place the cable in the recessed area between the keyboard and the computer.

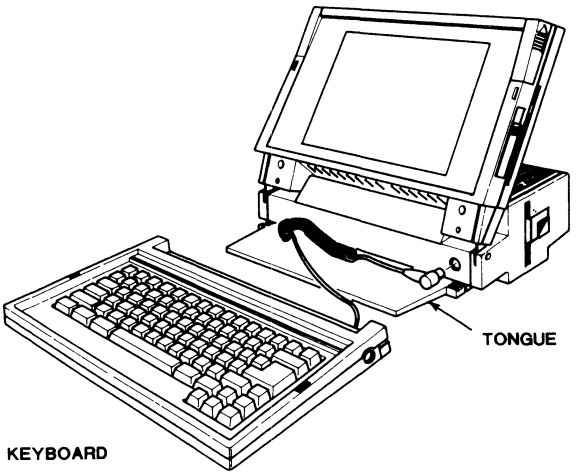


Figure 2-4. Attaching the Keyboard

The keyboard may also be used as a stand-alone unit. To remove the keyboard from the tongue of the computer:

1. Press up on the latches that hold the keyboard to the computer, as shown in Figure 2-5

2. Pull the keyboard toward you to separate it from the computer.

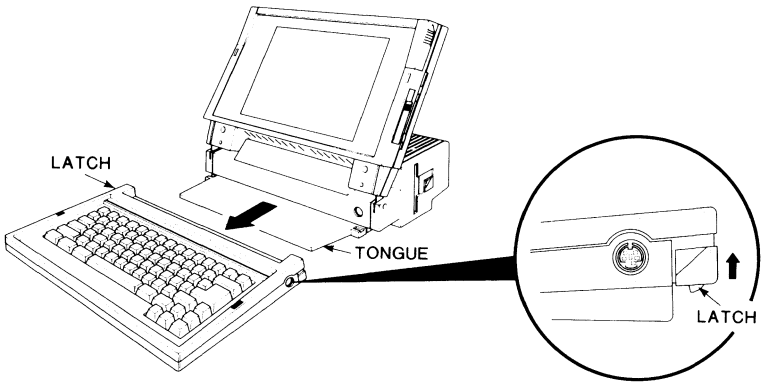


Figure 2-5. Removing the Keyboard

When using your keyboard as a stand-alone unit, you can adjust the tilt of the keyboard. Simply move the legs on the bottom of the keyboard, as shown in Figure 2-6.

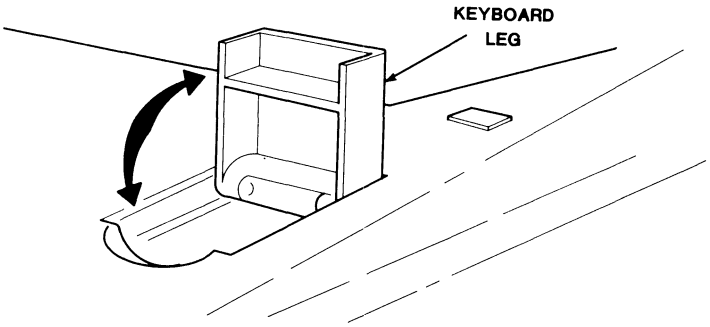


Figure 2-6. Keyboard Tilt Adjustment

Controls and Indicators

The brightness and contrast controls are on the right side of the lid, as shown in Figure 2-7. Slide the contrast control toward the bottom of the LCD display to increase the contrast. Slide the brightness control toward the top of the display to increase the brightness of the backlight. For now, set the brightness and contrast controls to the middle of their ranges.

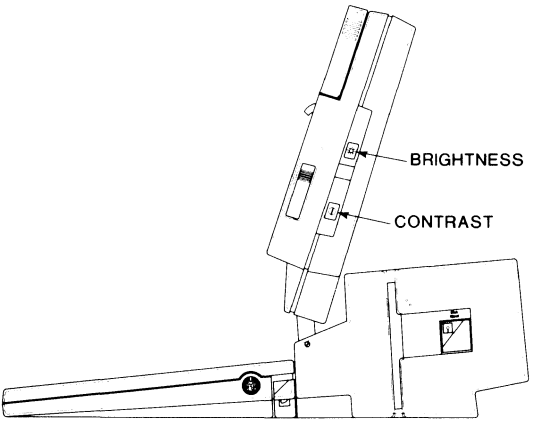


Figure 2-7. Brightness and Contrast Controls

Several LEDs are located on the keyboard, as shown in Figure 2-8. These LEDs indicate whether the power is on, when the disk drives are in use, and if certain keys are pressed on the keyboard. The power LED may light in any of three colors. Table 2-5 lists the colors and their meanings. For information about the remaining indicators, refer to Chapter 5.

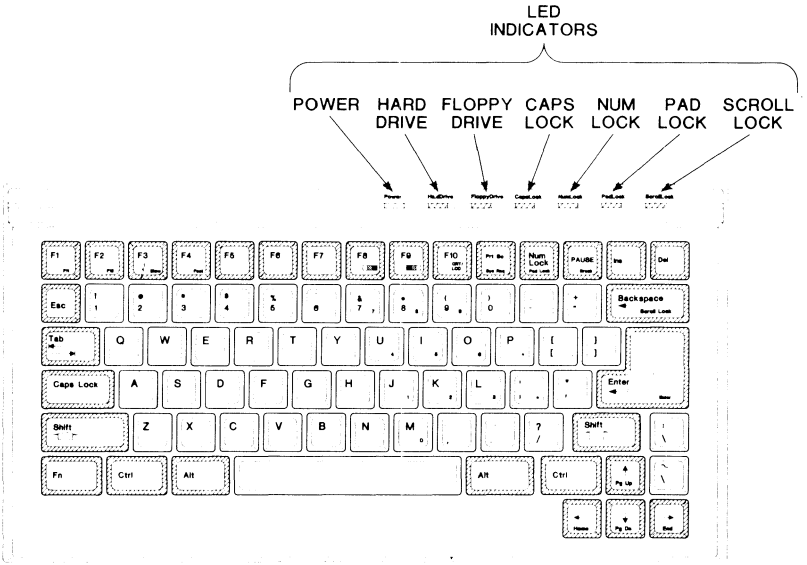


Figure 2-8. LED Indicators

Table 2-5. Power LED Indications

COLOR	INDICATION
Amber	AC adapter supplying power to computer
Green	Battery pack supplying power to computer
Red	Battery pack power is low (needs recharging)

Up and Running

After you have unpacked, set up, and connected all of your computer equipment, you are ready to get your computer up and running. This part of Chapter 2 describes:

- The power-up procedure
- Entering the Monitor program
- Setting the clock and calendar
- Loading the first floppy disk

Power-Up Procedure

To turn on the power to the computer, press the button on the left side of the computer, as shown in Figure 2-9. This button operates a momentary contact switch so be sure to press the button in for at least one second.

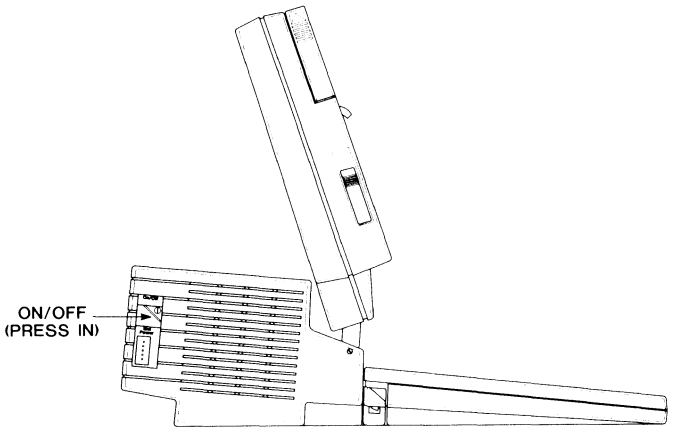


Figure 2-9. Turning On the Power

Refer to Figure 2-10 and press the ejection button on the 3.5-inch floppy disk drive. Remove any shipping insert installed in the drive. Use the insert when your computer is transported. Drives without shipping inserts installed have an auto-lock feature to protect the drive's read/write heads.

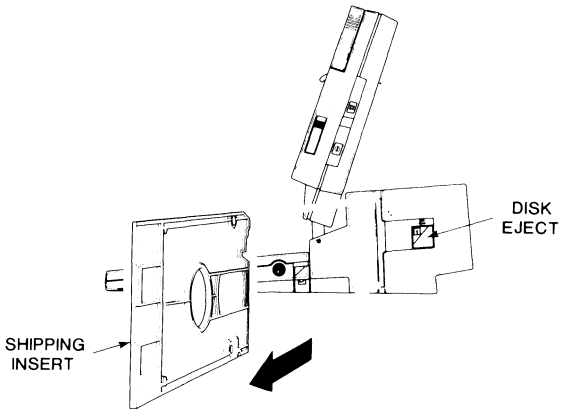


Figure 2-10. Removing the Shipping Insert

Each time your computer is turned on, it automatically performs a series of internal self-tests that check the major circuits and verify that various functions operate properly. If some part of the computer fails to operate correctly, the computer attempts to display an error message on the LCD. Chapter 19 contains more detailed information on these self-tests and error messages.

In addition to the self-tests, the following events occur when power is turned on:

- The power LED on the keyboard glows.
- The LEDs for the floppy disk drive and hard disk drive light alternately.

- The drives make some sounds as their read/write heads move back and forth.

- A blinking cursor appears on the LCD.

- After about 20 seconds, the computer starts the "autoboot" procedure. Autoboot is the automatic loading of an operating system from the floppy disk drive or the hard disk drive into the computer's memory.

Next, the floppy drive LED will light and remain on for a few seconds. Since there is no floppy disk in the drive, the hard disk LED will then light and remain on. Since the operating system has not yet been installed on the hard disk, the following error message will appear on the screen:

```
Not a bootable partition
```

This message is normal. It indicates that the computer is operating properly and has detected that the hard disk has no operating system installed.

Entering the Monitor Program

The Monitor program is a special program permanently built into your computer that helps the computer perform its tasks. Enter the Monitor program by pressing and holding the CTRL, ALT, and INS keys in sequence and then release them.

Chapter 6 discusses the Monitor program in detail. When you enter the Monitor program, the following message will be displayed on your LCD:

```
MFM-300 Monitor, Version x.xx
Memory Size: xxx
Enter "?" for help.
->_
```

Your computer will display numbers in place of the x's shown. This message is the Monitor program prompt. It lets you know that you have entered the Monitor program and that it is waiting for a command.

Setting the Clock and Calendar

Your computer has a built-in clock and calendar. It keeps track of the time and date even when the computer is turned off. To keep accurate time, you must set the clock and calendar when you turn on the computer for the first time. To set the clock and calendar:

1. Type the word SETUP and press the ENTER key. This command allows you to enter the Setup/Configuration program and set your computer's clock and calendar. Figure 2-11 shows the Setup/Configuration menu.

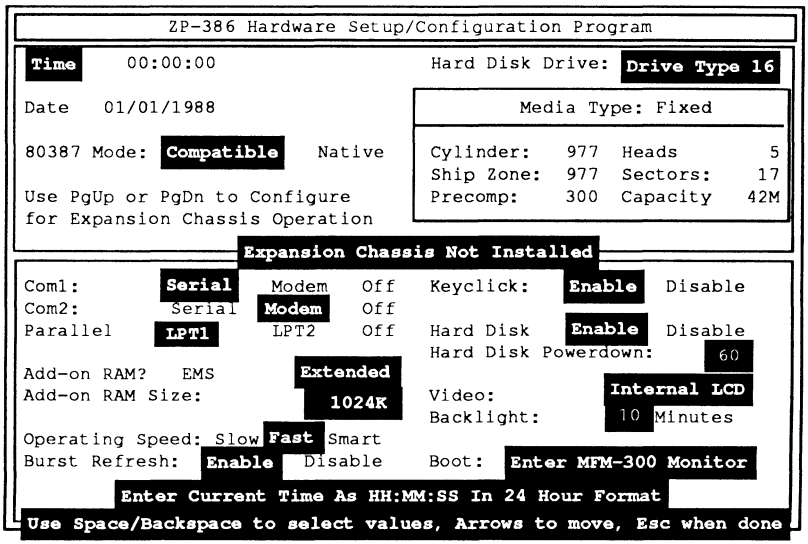


Figure 2-11. Setup/Configuration Menu

The display is divided into sections (called fields) that contain information about the hardware installed in your computer. Check to make sure that these fields match the hardware actually installed. If you have any questions or notice that something does not match, refer to Chapter 4 for instructions for updating the Setup/Configuration program. If you still have questions, contact your service representative.

2. The Time field is highlighted in the upper left corner of the display and the blinking cursor is located under the first number immediately to the right of the word Time.

   Your computer keeps track of time using a 24-hour clock, in hours (00 - 23), minutes (00 - 59), and seconds (00 - 59). Type in the time of day using this format, and then press the ENTER key. For example, to enter 8:35 a.m., type 083500; to enter 8:35 p.m., type 203500. The computer automatically adds the colons (:) to separate the hours, minutes, and seconds.

   If you make a mistake while typing, use the BACKSPACE key to erase the incorrect entry, type it again, and then press the ENTER key.

3. The highlight has now moved to the Date field and the blinking cursor is now located under the first number to the right of the word Date.

   Your computer keeps track of the date using months (01 - 12), days (01 - 31), and years. Type in the date using this format, and then press the ENTER key. For example, to set the calendar for January 5, 1988, type 01051988. The computer automatically adds the slashes (/) to separate the month, day, and year.

   Again, if you make a mistake while typing, use the BACKSPACE key to erase the incorrect entry, type it again, and then press the ENTER key.

4. When you are finished setting the calendar and clock, press the ESC key. The following message will appear at the bottom of the screen:

```
Are You Done Making Changes (Y/N)?
```

5. Press the Y key to let the computer know that you are finished setting the clock and calendar. The message at the bottom of the screen will change to read:

```
Press Return to save changes or ESC to ignore changes.
```

6. Press the ENTER key to let the computer know that you want it to use the clock and calendar information you entered.

7. Your computer will attempt to autoboot. Use the CTRL-ALT-INS key combination to return to the Monitor program.

Loading a Disk

1. Locate the operating system material shipped with your computer. The package contains one 3.5-inch floppy disk with the operating system and a setup program on it. Since the disk contains the operating system, it is "bootable," that is, it can be loaded into computer memory.

2. Insert the floppy disk into the floppy disk drive (drive A), as shown in Figure 2-12. Be sure the disk is oriented properly, with the label toward you. Slide the floppy disk all the way into the drive until you hear it click into place. You are now ready to boot your computer.

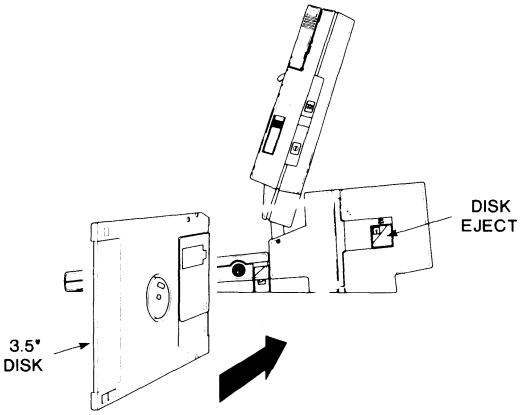


Figure 2-12. Inserting a Disk

3. Press the B key, the F key, and then the ENTER key. This command means "boot from the floppy drive". Pressing the ENTER key tells the computer to carry out the command.

   During the boot process, the floppy drive's LED on the keyboard lights, indicating that the disk is being read by the computer. You will also hear a clicking sound from your floppy drive. This is the normal sound of the disk drive motors. After a few seconds, the following opening message appears on the LCD:

```
Current date is xxx x-xx-xxxx
Enter new date (mm-dd-yy):
```

(The messages displayed on your LCD after you boot up the operating system may vary, depending on the operating system you have.) The booting process is now complete. You have just loaded the operating system from the floppy disk into your computer's memory. The booting process is also called "bootup".

4. Reply to the requests for information displayed on the LCD until the operating system prompt (A>) appears. If you make a mistake when typing a response, simply backspace over the mistake and retype your entry.

The operating system's setup program guides you through the beginning steps of making copies of your operating system disks, setting up your computer, and preparing your floppy and hard disks so that you can use applications software packages. If you encounter any problems in loading the correct floppy disk or using the operating system's setup program, refer to your operating system documentation. It describes the setup program in detail and includes procedures for correcting problems.

Resetting Your Computer

After you have completed the operating system's setup program and installed the operating system on the hard disk drive, you are ready to use the operating system. This section describes how to reset your computer and cause it to autoboot, that is, automatically load the operating system into the computer's memory.

1. Make sure there is no floppy disk installed in your floppy disk drive.

2. Press and hold the CTRL, ALT, and DEL keys in sequence, and then release them. This key combination resets your computer and causes it to autoboot.

3. After 20 or 30 seconds you should see the operating system's opening messages and prompt appear on the screen. If this does not happen, check your operating system documentation and make sure that you completed all of the procedures in the operating system setup program.

Booting Manually and Autobooting

As you become more familiar with your computer, you will want to take advantage of the autoboot feature. At other times, you may want to disable autoboot and boot a disk manually.

There are three ways to disable the autoboot feature:

- With no floppy disk in the drive and no operating system installed on the hard disk drive, turn the computer on (or reset it using the CTRL-ALT-DEL key combination), and wait until the autoboot process is completed. When an error message appears on the video monitor, press the ESC key to enter the Monitor program.

- Enter the Monitor program directly by pressing the CTRL-ALT-INS key combination.

- Change the boot drive information stored in the computer's memory by the Setup/Configuration program. Refer to Chapter 4 to do this.

There are three ways to enable your computer to autoboot:

- Insert a bootable disk in the floppy disk drive within 20 seconds of turning on power to the computer. Remember that a bootable disk has the operating system on it.

- With a bootable disk installed in the floppy disk drive, reset the computer using the CTRL-ALT-DEL key combination.

- If necessary, change the boot drive information stored in the computer's memory by the Setup/Configuration program. Refer to Chapter 4 to do this.

Your computer is factory-set to autoboot from the floppy disk drive and, if no bootable disk is in the drive, to boot from the hard disk. If you have loaded the operating system onto the hard disk and have not changed the boot drive information, your computer will autoboot each time you turn on the power or reset the computer. You do not have to put a bootable disk in the floppy drive.

Transporting the Computer

To prepare your portable computer for transportation, complete the following steps:

1. Turn off the computer by pressing in the power switch for at least one second.

2. Remove any floppy disk ejected from the drive.

3. If the keyboard has been used as a remote unit, attach it to the computer as described earlier in this chapter.

4. Disconnect the AC adapter.

5. Close the lid and make sure it latches.

6. Pull the handle out from the lid as shown in Figure 2-13. The lid must be closed before you can pull out the built-in handle.

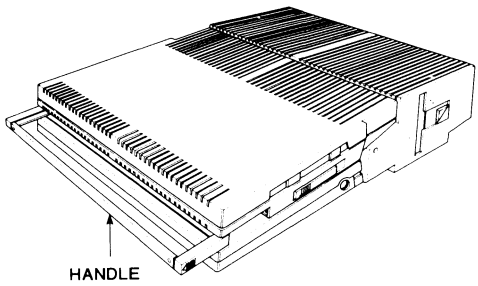


Figure 2-13. Built-In Handle

Chapter 3

Disassembly

There are no user-servicable components inside this computer. If service is required, contact your nearest authorized Zenith Service Center.

The battery pack is the only user-installable option. To have the expansion memory, numeric coprocessor, and modem options installed in your computer, contact your nearest authorized Zenith Service Center.

Battery Pack

Your computer can use a rechargeable nickel-cadmium battery for portable operation. The length of time a computer will operate with the battery depends upon the number of options installed and the battery's capabilities.

Use the following procedure for installing the optional battery pack in your computer. A flat-head screwdriver or coin is the only tool required.

1. Loosen the three screws and remove the battery pack cover, as shown in Figure 3-1. Set it to one side.

2. Place the battery pack inside the computer and attach the connector.

3. Replace the battery pack cover and tighten the three screws.

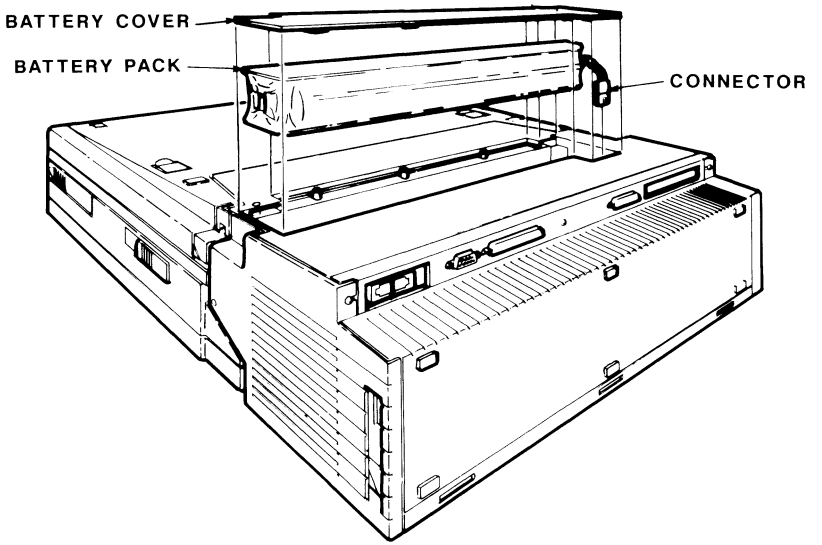


Figure 3-1. Battery Pack Installation

Use the following procedure for removing the optional battery pack from the computer.

1. Remove the battery pack cover.

2. Lift the battery pack from the compartment and detach the connector.

3. Replace the battery pack cover and screws.

Chapter 4

Configuration

There are no switches or jumpers to set on this computer. All the configuration is done through the Setup/Configuration program that is part of the Monitor program.

Setup/Configuration Program

The Setup/Configuration program serves as a reference for your computer by storing information about the hardware installed. It is also used when you set your computer's built-in clock and calendar. To enter the program, type the word SETUP after the Monitor prompt (->) and then press the ENTER key.

The Setup/Configuration program display consists of two menu screens. One displays the hardware information when the computer is used as a portable, that is, when the optional expansion chassis is not attached to the computer. This screen displays the message "Expansion Chassis Not Installed." The other screen displays hardware information when the optional expansion chassis is attached. This screen displays the message "Expansion Chassis Installed." You can change from one menu to the other with the PgUp and PgDn keys. Each menu may contain different information.

When the Setup/Configuration menu appears, as shown in Figure 4-1, some of the information will be highlighted in rectangular boxes, called "fields." The fields contain information about the hardware currently in your computer. Check to make sure that each field matches the hardware installed in your computer. If it does not, change it by highlighting the correct information.

Figure 4-1 shows the default entries for the "Expansion Chassis Not Installed" menu. The default entries for the "Expansion Chassis Installed" menu are the same. However, each menu can contain different entries.

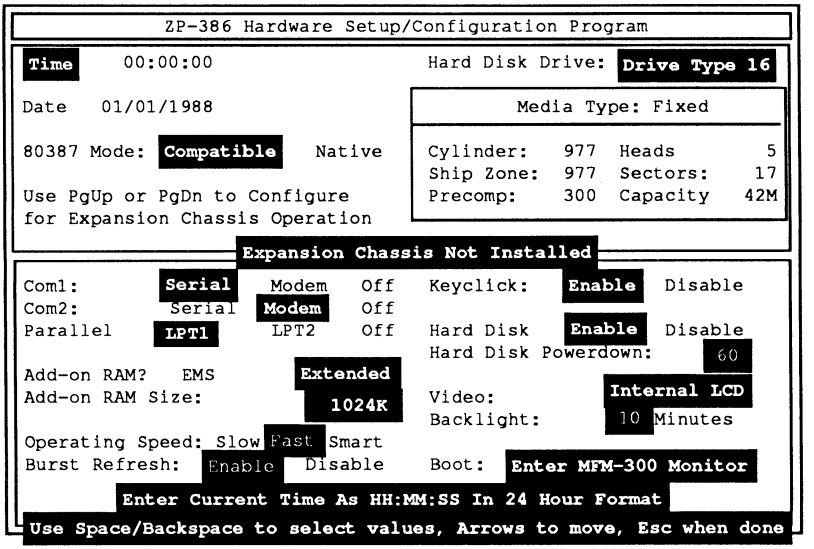


Figure 4-1. Setup/Configuration Menu

You can move between the fields of information using the arrow keys. Within each field, except the Time, Date and Backlight fields, use the space bar and BACKSPACE key to highlight the entry that matches your computer's configuration. The Time, Date, and Backlight entries must be typed in.

Error Messages

The backup battery provides power for the computer memory that stores the configuration information. If your backup battery loses power, the information stored in the Setup/Configuration program is lost and one of the following messages will be displayed:

```
+++ ERROR: Please replace the backup battery! +++
+++ ERROR: Bad configuration information found in CMOS! +++
```

When this happens, press the ESC key and update the Setup/ Configuration program. The default entry for each field will be highlighted.

If the error message was caused by a faulty backup battery, the information you enter now will be lost again when you turn your computer off. If it is, the backup battery needs replacing. Refer to Chapter 19 for more information on error messages and the backup battery.

NOTE: The backup battery can only be replaced by a qualified service technician. Take your computer to your nearest Zenith Service Center to have the backup battery replaced.

Time and Date

The computer's clock and calendar are battery-operated and will run continuously, even when the computer is turned off. To keep accurate time, you must set the time and date when you turn on the computer for the first time. After that, you only need to enter a new time and date if you replace the backup battery. Refer to "Setting the Clock and Calendar" in Chapter 2.

80387 Mode

The 80387 Mode field selects the method by which the 80386 CPU and the coprocessor are connected. In the compatible mode, the CPU and the coprocessor are connected through a logic interface. The logic interface maintains compatibility with previous IBM-compatible software. In the native mode, the CPU and the coprocessor are directly connected. Future software may support this mode. For now, leave this field set to "Compatible".

Hard Disk Drive

When the computer is shipped from the factory, the hard disk drive type is set to match the drive that was installed in the computer. You should not change this selection unless the original drive is replaced by a drive of a different type. If the backup battery is replaced, the drive type may have to be re-entered. Press the space bar to step through the drive types.

COM1 and COM2

Com1 and Com2 are each used to address either the serial port or the internal modem, if it is installed. They cannot both be set to the same value except Off. Setting Com1 and/or Com2 to Off conserves power during battery operation.

Parallel

The Parallel field determines whether the parallel port is addressed as LPT1 or LPT2. Turning the parallel port off conserves battery power.

Add-On RAM

The Add-On RAM field determines whether the expansion memory will be used as EMS memory or extended memory.

Add-On RAM Size

Your computer comes equipped with 2 megabytes of memory. A 1-megabyte expansion memory board is available. If your computer has the standard 2 megabytes, set this field to 1024K. If the 1-megabyte expansion board is installed, set this field to 2048K. Use the space bar to increase the number and the BACKSPACE key to choose a lower number.

Operating Speed

You have a choice of speed modes: Slow, Fast, and Smart. The fast speed is the normal (and default) mode of operation. However, some programs may require a slower speed in order to run. The Smart mode slows the computer down for floppy disk drive operations but allows the fast speed for other operations. If the program still does not run properly, it may require the Slow mode. This mode slows down all operations of the computer.

Burst Refresh

The Burst Refresh field selects the way memory is refreshed. Enabling burst refresh increases overall computer performance by as much as 75%. However, some hardware expansion boards, and some application programs, may not work properly with burst refresh enabled. Disable burst refresh if you have trouble using an expansion board (in a Zenith Data Systems expansion box) or any application program.

Keyclick

The Keyclick field allows you to disable the audible feedback of the keyboard.

Hard Disk

The Hard Disk field allows you to disable the hard disk drive. You may want to do this during battery operation to conserve power.

Hard Disk Powerdown

This field sets the number of seconds the hard disk drive runs after the last disk access. Set this field to a high number if your application requires frequent disk access. If your disk access is infrequent, set this field to a low number to conserve battery power.

Video

The Video field allows you to choose between the built-in LCD or an external monitor. An RGB monitor may be attached to the video connector on the back panel by using an optional video adapter cable. This cable is available from your Zenith Data Systems dealer. If the optional expansion chassis is attached to the computer, there are four additional selections available:

Color Card: 40x25 — This settings displays text at 40 characters per line, 25 lines per screen.

Color Card: 80x25 — This setting displays text at 80 characters per line, 25 lines per screen.

Mono Card: 80x25 — This setting selects a high-resolution monochrome display.

Enhanced Graphics — This setting selects an EGA or a high-resolution analog display.

Backlight

Use the Backlight field to set the time the backlight remains lit when there is no activity on the keyboard. Setting this value to 0 (zero) turns the backlight off. Setting the value to 99 (maximum) causes the backlight to remain on continuously. Turning the backlight off conserves power during battery operation.

Boot

The Boot field allows you to select the drive your computer will attempt to boot from when it autoboots. You can select one of the following options:

Floppy Drive 0 — The computer will attempt to load the operating system from the floppy disk drive.

Hard Disk Drive 0 — The computer will attempt to load the operating system from the hard disk drive.

Floppy then Hard Disk — The computer will attempt to load the operating system from the floppy disk drive. If that drive is not ready, the computer will attempt to load the operating system from the hard disk drive.

Enter MFM-300 Monitor — This selection defeats autoboot. The computer will not attempt to load the operating system from either drive but will call up the Monitor program instead.

Chapter 5
Operation

This chapter describes the keyboard and battery.

Keyboard

The keyboard's layout makes it easy to use for all types of entries. The 79 keys are arranged in the following groups, as shown in Figure 5-1:

- Keyboard indicator LEDs
- Alphanumeric keys
- Control and special purpose keys
- Cursor control keys
- Numeric keypad

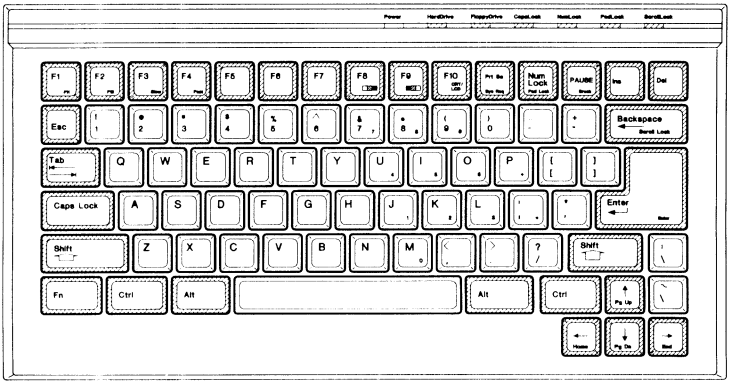


Figure 5-1. Keyboard Arrangement

Some of the key groups contain familiar keys, like those found on a typewriter keyboard. Others contain unique keys. Some familiar keys also have unique functions; your computer and software can program them with special features. The unique keys and special features of familiar keys are described in the following pages.

Almost every key on the keyboard auto-repeats. This means that the key repeats itself as long as you hold it down. The longer you hold an auto-repeat key down, the faster it repeats itself. This feature is useful with keys that move the cursor, like the DEL (delete) key, the ENTER key, the space bar, and the arrow keys. The only keys that do not have the auto-repeat feature are SHIFT, CTRL (control), ALT (alternate), FN (function), CAPS LOCK, SCROLL LOCK, NUM LOCK (number lock), and PAUSE.

Also, all keys, except SHIFT, CTRL, ALT, and FN, have audible feedback. Audible feedback is the clicking sound made each time a key is pressed. You can turn it off permanently by an entry in the Setup/Configuration program (refer to Chapter 4). It can be turned off (or back on) temporarily by pressing and holding the ALT key, pressing the tilde (~) key, and then releasing both keys.

Keyboard Indicator LEDs

Figure 5-2 illustrates the LEDs located at the upper right of the keyboard. These indicators light when the CAPS LOCK, NUM LOCK, PAD LOCK, or SCROLL LOCK keys are active. These keys are discussed later in this chapter.

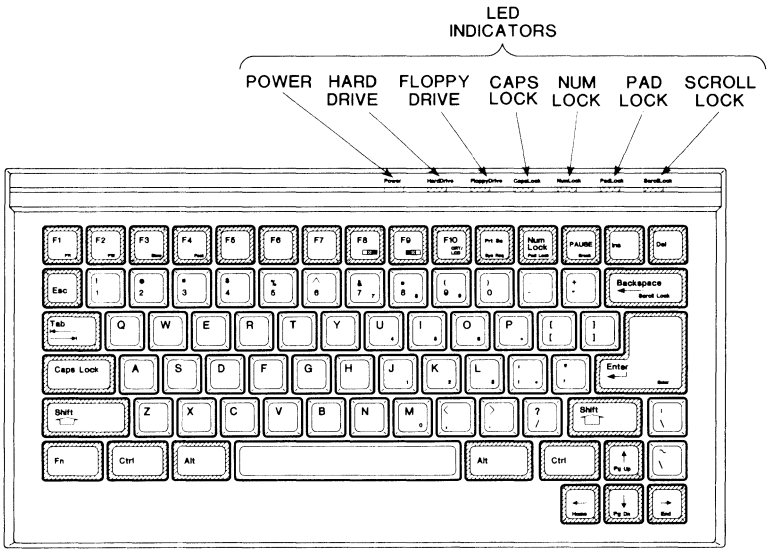


Figure 5-2. Keyboard Indicator LEDs

Alphanumeric Keys

Many of the alphanumeric keys are the same as on a typewriter keyboard. These keys are shown in Figure 5-3. To help touch typists keep their fingers on the proper keys, the F and J keys have a raised dot.

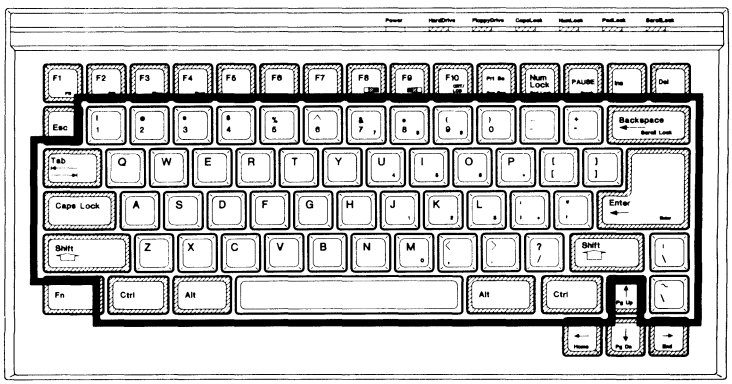


Figure 5-3. Alphanumeric Keys

Most of these keys also perform like typewriter keys. When a number, letter, punctuation mark, or symbol key is pressed, that character appears on the screen. The CAPS LOCK key acts like a typewriter shift lock key, and the ENTER key acts like a typewriter carriage return key.

The space bar, SHIFT, BACKSPACE, and TAB, when programmed by software, perform quite differently from typewriter keys. For example, the SHIFT keys affect some keys, but not others.

There are also some alphanumeric keys that are unique to computers. The two modifier keys, CTRL (control) and ALT (alternate), are always used with other keys. These keys perform no function when used alone.

This section describes how the alphanumeric keys work.

Shift — There are two SHIFT keys on the keyboard; one is located on the right side of the keyboard and one on the left. Usually, when a SHIFT key is pressed, capital letters, symbols, and alternate punctuation marks are generated. If CAPS LOCK is engaged, pressing a SHIFT key causes the letter keys to generate lowercase letters. In some software programs, the SHIFT keys are also used in combination with other keys to enter commands.

Caps Lock — This is almost the same as a typewriter shift lock key: press it once to begin typing capital letters and press it again to stop typing capital letters. The difference is that CAPS LOCK only affects the letter keys. You must press either the right or left SHIFT key to change the top row of numbers into special symbols (!, @, #, etc.) and to engage the alternate punctuation marks and symbols ({, :, ?, <, etc.). Also, when CAPS LOCK is engaged, the right and left SHIFT keys allow you to type lowercase letters. The CAPS LOCK indicator lights when CAPS LOCK is engaged.

Enter — This key returns the cursor to the left side of the display. Software usually adds a line feed instruction as well. Also, pressing the ENTER key after data or instructions have been entered tells the computer to process them.

Tab — This key moves the cursor to the next tab setting. Many software programs allow you to change the tab settings. Your software documentation will give you specific instructions on setting tabs.

Space bar — When you press the space bar, it usually enters a blank character (space). The space bar is also used with some software to move the cursor around the screen or to change the text on the screen. For example, in the Setup/Configuration program, you press the space bar to view information about different types of hard disk drives.

Backspace — This key moves the cursor one space to the left and usually erases any character in the cursor's path. The BACKSPACE key is also used with some software to move the cursor around the screen without erasing characters. Other programs use the BACKSPACE key to view previous text.

Ctrl — There are two CTRL keys on the keyboard, one on the right and one on the left. This key is one of the main keys for entering commands. Typically, you press and hold the CTRL key and then press another key. The CTRL key is often symbolized by a caret (^). For example, to enter ^K, you would press and hold the CTRL key, press the K key, and then release both keys.

Alt — There are two ALT keys on the keyboard, one on the right and one on the left. The ALT key is similar in operation to the CTRL key. It is used with other keys to enter commands. Usually, the function of the ALT key is programmed by the software.

Control and Special Purpose Keys

The control and special purpose keys, illustrated in Figure 5-4, provide control over the computer and keyboard. The following paragraphs describe the normal function of each key. However, software can direct almost any key to perform a different function.

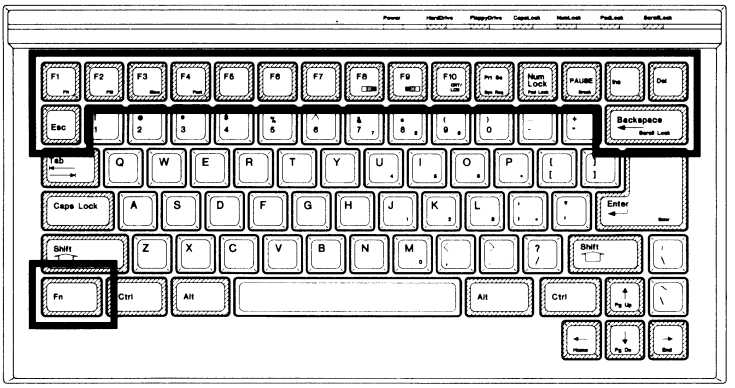


Figure 5-4. Control and Special Purpose Keys

Esc — The ESC key, located just above the TAB key, performs special functions when you press it and another key in sequence. It is commonly used to stop the execution of a program or exit a function.

Fn — The function key, located in the lower-left corner of the keyboard, acts similarly to the CTRL key. It provides additional codes when you hold it down and press another key. The FN key causes the keys with blue labels to generate those characters and functions.

F1-F12 — These function keys are used for special purposes by software packages. For example, in some word processing programs, the function keys perform indenting, setting right and left margins, underlining, and boldfacing. The function keys have special capabilities in the MS-DOS operating system. To use function keys F11 and F12, first press and hold FN and then press either F11 or F12.

Prt Sc/Sys Req — When the PRT SC key is pressed, whatever is on the screen will be sent to the printer to be printed. This key is particularly useful when you run operating system commands like DIR and TYPE and want to have a paper copy of the information. If your printer is turned off, not on-line, or out of paper, the computer pauses for about 10 seconds. If the printer is still not ready at the end of this time, the computer ignores the command.

NOTE: The PRT SC key must be held down for at least one second. This prevents the print screen feature from being activated accidentally.

When used with the ALT key, SYS REQ performs a function similar to the BREAK key. Various software programs use it to return to the operating system.

Num Lock/Pad Lock — This key provides two functions. When used with the FN key (FN-PAD LOCK), it toggles the keypad lock. The PAD LOCK indicator lights when the keypad lock is active. When the keypad lock is active, the NUM LOCK key toggles the keypad between the numeric mode and the cursor control mode.

Pause/Break — Pressing the PAUSE key freezes the display of text on the screen. For example, if you enter an operating system TYPE command, the text scrolls up the screen faster than you can read it. Pressing the PAUSE key will stop the display temporarily. Pause can be canceled by pressing any key other than SHIFT, CTRL, ALT and CAPS LOCK.

BREAK is generally used to halt commands or programs as they are running. For example, if you begin an operating system DISKCOPY routine, then realize that you cannot find the disk you want to copy, you can return to the operating system prompt by pressing the CTRL key and the BREAK key in combination.

Ins — When pressed in combination with the CTRL and ALT keys, the INS key calls up the Monitor program. Since the computer is not actually reset, you can boot an alternate drive using Monitor commands. In many software programs, this key allows text or commands to be inserted.

Del — When pressed in combination with the CTRL and ALT keys, the DEL key causes the computer to reset. Many software packages also use this key to erase characters, words, or whole documents.

Scroll Lock — When you press the FN key and the BACKSPACE key, an indicator on the keyboard lights. This indicates that SCROLL LOCK is engaged. Press the keys again to turn SCROLL LOCK off. Some programs use SCROLL LOCK to keep the cursor on the same screen line while moving the text.

Cursor Control Keys

The cursor control keys are illustrated in Figure 5-5. The arrows on the keys indicate the direction in which they move the cursor. The Setup/ Configuration program uses these keys to move the highlight from one field to the next. They can also be used with word processing software to move the cursor for typing and positioning text. With many spreadsheet programs, these keys move the active cell indicator.

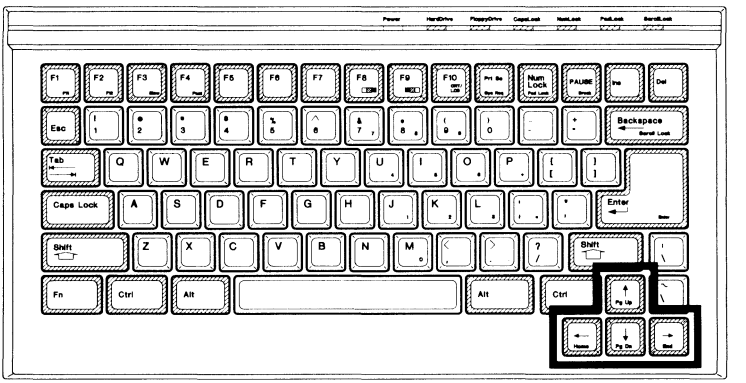


Figure 5-5. Cursor Control Keypad

NOTE: A few older software programs may not allow some functions to be performed by the separate cursor control keys. If you are running one of these programs, it may be necessary to use the cursor control keys in the numeric keypad section of the keyboard.

Home — In many software programs, pressing the FN-HOME key combination moves the cursor to the upper-left corner of the screen. Some word processing programs use the HOME key to move the cursor to the beginning of the current line. With spreadsheet programs, pressing the HOME key usually moves the active cell indicator to the upper-left corner of the spreadsheet.

End — Pressing the FN-END keys typically moves the cursor to the lower-left corner of the screen. With some word processing programs, the END key moves the cursor to the end of the current line. Many spreadsheets use the END key to moves the active cell indicator to the most remote cell.

Page Up (PgUp) and Page Down (PgDn) — The FN-PAGE UP and FN-PAGE DOWN keys move the cursor a set number of lines upward or downward.

Numeric Keypad

The numeric keypad, shown in Figure 5-6, has many of the same keys as a calculator keypad. For example, the numbers keys are arranged as on a calculator to allow numeric data to be entered rapidly. The FN-PAD LOCK key combination toggles the keypad lock on and off. Pressing the FN key while PAD LOCK is engaged causes the keypad keys to generate alphanumeric characters. These keys are also dependent on the state of CAPS LOCK. An indicator on the keyboard lights when PAD LOCK is engaged.

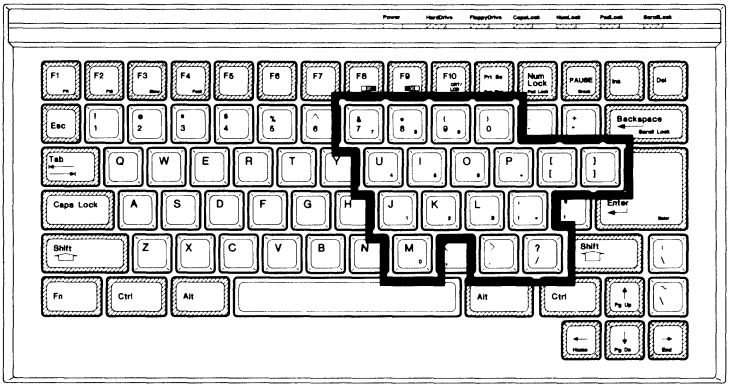


Figure 5-6. Numeric Keypad

When PAD LOCK is engaged, the NUM LOCK key toggles the numeric and cursor control modes of the keypad. The numeric mode causes the keys with light blue labels to generate numbers and the cursor control mode causes the keys with light blue labels to move the cursor. When the NUM LOCK key is active, an indicator on the keyboard lights. When only the PAD LOCK indicator is lit, the keypad is in the cursor control mode. When both the PAD LOCK and the NUM LOCK indicators are lit, the keypad is in the numeric mode. The SHIFT keys reverse the action of the NUM LOCK key. If the keypad is in the numeric mode, press a SHIFT key to generate cursor movement codes. If the keypad is in the cursor control mode, press a SHIFT key to generate numeric codes.

Table 5-1 lists the results of pressing a keypad key in the unshifted mode, with the SHIFT (or CAPS LOCK) key, with the FN key, and with the SHIFT and FN keys for each of the possible combinations of the PAD LOCK and NUM LOCK modes.

Table 5-1. Keypad Results

KEY PRESSED	RESULT NORMAL KEYBOARD[1]	NORMAL KEYBOARD[2]	NUMERIC MODE[3]	CURSOR MODE[4]
Keypad key	Lowercase letter or number printed in black.	Lowercase letter or number printed in black.	Number or symbol printed in blue.	Cursor function printed in blue.
SHIFT-keypad key	Uppercase letter or symbol printed in black.	Uppercase letter or symbol printed in black.	Cursor function printed in blue.	Number or symbol printed in blue.
FN-keypad key	Number or symbol printed in blue.	Cursor function printed in blue.	Lowercase letter or number printed in black.	Lowercase letter or number printed in black.
SHIFT-FN-keypad key	Cursor function printed in blue.	Number or symbol printed in blue.	Uppercase letter or symbol printed in black.	Uppercase letter or symbol printed in black.

NOTES

1. NUM LOCK on, PAD LOCK off (default)
2. NUM LOCK off, PAD LOCK off.
3. NUM LOCK on, PAD LOCK on.
4. NUM LOCK off, PAD LOCK on.

Keyboard-Selectable Modes

In addition to the keypad modes mentioned in the preceding section, there are several operating modes that can be selected from the keyboard. These modes are described in the following paragraphs and summarized in Table 5-2. Some software programs also create special keyboard modes not described in this manual.

Slow/Fast Mode — The Setup/Configuration program determines the speed at which the computer operates. There are also two function key combinations that affect the computer's speed. FN-F3 slows the machine to 6MHz for software that may not run properly at the normal fast speed. All other operations remain the same. FN-F4 returns the computer to the Fast mode of operation.

Video Palettes — Three function key combinations affect video operation. FN-F10 toggles the video output between the CRT connector and the LCD. FN-F8 and FN-F9 toggle the LCD through ten display palettes. These two key combinations do not affect the normal color video palettes.

The video system in this computer is similar to the color graphics adapter in PC-compatible computers for software that checks for color and automatically goes into a color display mode. However, although the computer can produce a gray scale on the LCD, not all colors are dark enough to be readable. Ten LCD display palettes allow you to select a gray scale that makes these colors more visible. Since color combinations vary from one program to another, a gray scale that works for one program may not work for another. To see these gray scales, press CTRL-ALT-INS to enter the Monitor program, press C to call up the color bar, and use the FN-F8 and FN-F9 key combinations to change the gray scales.

Table 5-2. Keyboard-Selectable Modes

KEY COMBINATION	DESCRIPTION
FN-(blue labels)	Generates the function printed in blue. If the keypad lock is active, this combination generates the alphanumeric character for the keypad key pressed.
FN-PAD LOCK	Toggles the keypad lock on and off. The keypad lock establishes the keys in Figure 5-5 as a numeric keypad. An LED lights when the keypad lock is on.
FN-F3	Places the computer in the Slow mode of operation.
FN-F4	Places the computer in the Fast mode of operation.
FN-F8 and FN-F9	These two combinations toggle the display palette. You can move the display palette selection forward (FN-F8) or backward (FN-F9) through ten different gray scales.
FN-F10	Toggles the video output between the LCD and the CRT connector.

Key Combinations

Table 5-3 describes the most common key combinations used by this computer. The words "software dependent" in the description indicate those key combinations that depend on software to produce an action.

Table 5-3. Key Combinations

KEY SEQUENCE	DESCRIPTION
ALT-SYS REQ	System request (software dependent).
ALT-~	Toggles key click on and off.
CTRL-ALT-DEL	Resets computer to autoboot sequence.
CTRL-ALT-INS	Resets computer to Monitor program.
CTRL-BREAK	Stops operation (software dependent).
CTRL-S	The output on the screen pauses until you press another key.
CTRL-C	Stops operation (software dependent).
FN-(down arrow)	Page down (software dependent).
FN-(keypad keys)	Numeric keypad characters.
FN-(left arrow)	HOME (software dependent).
FN-(right arrow)	END (software dependent).
FN-(up arrow)	PAGE UP (software dependent)
FN-F1	F11 (software dependent).
FN-F2	F12 (software dependent).
FN-F8	Changes LCD gray scale palette.
FN-F9	Changes LCD gray scale palette.
FN-FAST	Activates fast mode of operation.
FN-CRT/LCD	Toggles CRT/LCD output.
FN-PAD LOCK	Toggles numeric keypad lock.
FN-SLOW	Activates Slow mode of operation.
FN-SCROLL LOCK	Scroll lock (software dependent).
FN-SHIFT-(keypad keys)	Activates cursor control keys on keypad.

Battery Operation

Nickel-cadmium batteries have three unique operating characteristics:

- They are rechargeable.
- They maintain a constant voltage level over their operating period. (Carbon or alkaline batteries do not maintain this constant voltage level. Instead, their output gradually drops.)
- They can develop a severely reduced operating period if you do not care for them properly.

To get the most use from the battery:

- Run the Setup/Configuration program and turn off or disable all the features that will not be used. For example, if you do not have a serial device attached to the computer, turn off the serial port.
- Keep disk drive use to a minimum. Some programs use a lot of disk input/output. For these programs, set up a virtual disk (sometimes called a RAM disk or memory disk). Refer to your operating system documentation for more information on virtual disks.
- Keep the display backlight set to the lowest level possible for comfortable operation. In some situations, natural light will be enough for normal operation without the backlight. This is particularly true in bright sunlight.
- Operate the battery as long as possible before recharging it. When the battery power is low, the power LED turns red. You then have about ten minutes to complete and save your work.
- Allow the computer to operate until the battery is completely dead (the backlight ceases to function and the screen data becomes garbled).
- Charge the battery for eight to twelve hours. If you are using the quick-charge battery pack, you will only have to charge it for one to two hours.

All ni-cad batteries have an operating ledge, which is the point at which the ni-cad will no longer supply an operating current. This ledge is normally located at the point where the ni-cad reaches its lowest level of charge before it must be recharged. Under certain conditions this ledge will rise to the point that the battery can be operated only a short time (sometimes, as short as ten minutes) before it fails to deliver adequate voltage and current and must be recharged.

Once the operating ledge has moved to a high level, the battery pack must be fully discharged, sometimes several times. To bring the operating ledge back to its normal position:

1. Fully discharge the battery pack. Operate the computer until it shuts down. To provide the highest amount of battery drain, use the read boot track test from the ROM-based tests (refer to Chapter 19).

2. Continue to operate the computer until the low-power indicator stops glowing.

3. Recharge the battery for a minimum of 16 hours.

At this point, you can attempt to use the battery pack. However, if the battery pack still fails to deliver a normal operating life on a single charge, repeat the full discharge-recharge process three times.

If the battery delivers an operating voltage for a very short time, typically less than ten minutes on a full charge, one of the cells of the battery pack is probably shorted and the battery pack must be replaced.

Part II

System Programming

Chapter 6

Software Interface

This chapter provides an overview of the Monitor program and its operation. Each time the computer is turned on, a number of system interrupts become available for use. An overview of these interrupts is included at the end of this chapter. The chapters that follow explain each interrupt.

The Monitor Program

The MFM-300 Monitor program performs self-tests each time the computer is turned on. Additional routines provided in the Monitor program are user-executed tests, video commands, disk boot routines, and a machine language debugger. The program also controls the Setup/Configuration program. Chapter 4 contains more information on the Setup/Configuration program.

In addition to performing the self-tests, the Monitor program initializes all circuits and synchronizes the disk drive heads with the rest of the system. If it detects a malfunction, one or more messages will appear on the system's display to alert the operator of a problem. Chapter 19 contains descriptions of these messages.

When all power-up tests are complete, the computer will either attempt to boot the operating system or enter the Monitor program. This action is determined by the settings in the Setup/Configuration program. In its default setting, the computer will attempt to boot the floppy disk drive and then the hard disk drive.

The computer will load the first sector of track 0 from the disk in the A drive into memory. If no disk is in the A drive, the computer will attempt to load the first sector of track 0 from the hard disk drive into memory. If the autoboot operation is unsuccessful, an error message will be displayed.

There are two jump vectors set up by the Monitor program that can be used by applications programs. They are located at F000:FFF0, which is the power-up reset vector, and F000:FFED, which is an unconditional jump to the Monitor program (the Monitor program prompt will appear on the screen).

ZBIOS

Many programs require direct access to hardware to implement specific features. In this computer, many of these hardware-dependent features are implemented differently. Therefore, program utilities are required to allow compatibility.

ZBIOS routines are special BIOS instructions written for the MS-OS/2 and MS-DOS operating systems. These instructions provide both operating systems with a consistent means of accessing particular machine features (provided the programs are written for ZBIOS).

By incorporating this flexibility into firmware, hardware differences (that may require special operating system handlers) between two similar computers become less obvious. As a result, many "off the shelf" versions of the operating system will work correctly without need for revision.

A program written to exploit ZBIOS routines must check F000:FFE3 through F000:FFE7 for a 5-byte ZBIOS signature. If the signature is valid, the program must provide an indirect far call using the offset stored at F000:FFE8. All calls to ZBIOS must have the following entry parameters, unless otherwise noted:

AH = Major function code
AL = Minor function code
DX = Data selector for F000H
CS = Code selector for F000H
ES = Data selector for segment 40H.

The major and minor function codes are always validated. If a call is returned with the carry flag set, the function was not implemented and all registers are preserved. Each function call should load register AL with FFH to return the current state of the function. This will also allow programs to see if a major function is implemented without actually executing the function.

In this computer, ZBIOS functions are written for the real-address mode and the protected mode. Therefore, ZBIOS functions have limited segment usage. However, ES is always a valid data selector for 40H and DX is always valid for F000H.

Table 6-1 lists the major functions that are currently supported in ZBIOS. These functions are subject to change as new firmware releases occur.

Table 6-1. ZBIOS Functions

FUNCTION CODE (AH)	DESCRIPTION
00H	GATEA20 — Enables/disables address line A20
01H	REAL MODE — Places CPU in real address mode
02H	REBOOT — Reboots the computer
03H	BACKLIGHT — Sets timeout for backlit display
04H	MODEM — Enables/disables modem
05H	SPEED — Sets CPU speed
06H	CACHE — Enables/disables cache memory (not used)
07H	PALETTE — Sets LCD palette values
08H	BATTERY — Returns battery life information
09H	MODEL — Returns address of model name
0AH	VERSION — Returns address of ROM version
0BH	MOTOR — Sets hard disk motor timeout
0CH	MONITOR — Monitors keystrokes

Function Code 00H: Gate A20 — When the value in AH is 00H and the value in AL is 00H, this function will disable address line A20. The purpose of disabling A20 is to provide compatibility with programs written for the 8088/8086 CPU. These programs rely on a memory wrapping technique beyond the one-megabyte address boundary.

NOTE: On entry to this function, the selector values for Monitor RAM (F000H) will not be in DX and the selector values for ROM data (40H) will not be in ES.

When the value in AL is 01H, this function will enable address line A20. When the value in AL is FFH, this function will return a value in AL which represents the method used to gate A20. Refer to Table 6-2.

Table 6-2. Address Line A20 Gating Methods

REGISTER AL VALUE	GATING METHOD	SOURCE
00H	AT-Compatible	System control processor
01H	Zenith	Port EEH
02H	PS/2	Port 92H

Function Code 01H: Real Mode — When the value in AH is 01H and the value in AL is 00H, this function places the CPU in its real address mode. This call can only be made from the protected mode.

NOTE: On entry to this function, the selector values for Monitor RAM (F000H) will not be in DX.

When this function is entered, the selector value to the real mode interrupt descriptor table (IDT) must be placed in BX and the real mode stack segment must be placed in CX. The selector to the real mode IDT is provided so 80386 implementations that must reset the CPU to switch to real mode can save the three bytes at 0:300H (normally destroyed by the ROM stack).

When this function is exited, the value in SS points to the location of the real mode stack segment (CX). When the system sees that CX holds the stack segment, it determines that the CPU is in real mode.

When the value in AL is FFH, this function will return a value in AL which represents the method used to place the CPU in real mode. Refer to Table 6-3.

Table 6-3. Real Mode Implementation Methods

AL VALUE	METHOD	SOURCE
00H	AT-Compatible	Triple fault
01H	Zenith	Port EFH
02H	PS/2	Port 92H

Function Code 02H: Reboot — When the value in AH is 02H and the value in AL is 00H, this function will perform a warm boot. When the value in AL is 01H, this function will perform a cold boot. When the value in AL is 02H, this function will enter the Monitor program. If AL is loaded with FFH, this function returns with no operation.

Function Code 03H: Backlight — When the value in AH is 03H and the value in AL is 00H, this function sets the timeout value for a backlit display. BX must contain the timeout value (in minutes) where 00H indicates no backlighting and FFFFH indicates constant backlighting.

If the zero flag sets, an error condition has occurred. If register AX returns 00H, an unsupported timeout value was entered.

When the value in AL is FFH, this function will report the current timeout (in minutes) for the backlit display. If the zero flag is set, the timeout value could not be determined.

Function Code 04H: Modem — When the value in AH is 04H and the value in AL is 00H, this function will disable the internal modem (if installed). When the value in AL is 01H, this function will enable the internal modem.

When the value in AL is FFH, this function will return the current state of the internal modem (in AL). If AL contains 00H, the modem is disabled and if AL contains 01H, the modem is enabled. If the zero flag sets and AX returns 00H, the modem state could not be determined.

Function Code 05H: Speed — When the value in AH is 05H and the value in AL is 00H, this function changes the CPU speed to Fast. When the value in AL is 01H, this function changes the CPU speed to Slow. When the value in AL is 02H, this function changes the CPU speed to Smart. When the value in AL is 03H, this function changes the CPU speed to Crawl (a power saving feature).

When the value in AL is FFH, this function returns a value in AL that indicates the current processing speed. Values of 00H, 01H, 02H, and 03H indicate the Fast, Slow, Smart, and Crawl speed modes, respectively.

Function Code 06H: Cache — When the value in AH is 06H and the value in AL is 00H, this function disables the memory caching feature. Register CX must contain a selector value that points to the first 64K of base memory.

When the value in AL is 01H, this function enables the memory caching feature (if the hardware supports it). Register CX must contain a selector value that points to the first 64K of base memory (memory cache).

NOTE: The selector to the 64K base memory segment is provided so that a memory read can be used to check for cache hardware. This selector may be a code selector.

If AL is loaded with FFH, this function returns the current state of the memory cache feature. Register CX holds the selector value that points to the first 64K of base memory. If the value in AL is 00H, it indicates that memory caching is disabled and if the value in AL is 01H, it indicates that memory caching is enabled. If the zero flag is set and AX contains 00H, the state of the memory cache could not be determined.

Function Code 07H: Palette — When the value in AH is 07H and the value in AL is 00H, this function will set the active LCD palette. Register BX must be loaded with a value representing the palette to be used.

NOTE: To determine how many palettes exist in this computer and which specific values select each palette, load AL with FFH and observe the values returned in AX and BX.

When the value in AL is FFH, this function returns the maximum number of palettes in AX and the current palette value in BX. If the zero flag sets and AX contains 00H, the palette value could not be determined.

Function Code 08H: Battery —When the value in AH is 08H and the value in AL is FFH, this function returns a value in AX that represents the percent of battery life remaining. If the percent is fractional, AH will contain a value representing the nearest whole number of the percent and AL will contain a value representing the fractional portion.

Function Code 09H: Model — When the value in AH is 09H and the value in AL is FFH, this function returns the linear address of the model name string in AXBX and the length of the string in CX.

Function Code 0AH: Version — When the value in AH is 0AH and the value in AL is FFH, this function returns the linear address of the ROM version string in AXBX and the length of the string in CX.

Function Code 0BH: Motor — When the value in AH is 0BH and the value in AL is 00H, this function will set the timeout value (in 5-second units) for the hard disk drive motor. Register BX must contain a value ranging from 00H (motor off) to FFFFH (motor on) that represents the amount of time the motor will run after the last access is made to the drive.

NOTE: A suitable minimum timeout value should be selected to prevent repeated start-up and spin-down of the drive.

When the value in AL is FFH, this function returns the current timeout value in BX. The value ranges from 00H (motor off) to FFFFH (motor on). If the zero flag sets and the value in AX is 00H, the motor timeout value could not be determined.

Function Code 0CH: Monitor — When the value in AH is 0CH and the value in AL is 00H, this function will monitor a specified keystroke or process keys. The value in BX contains the raw scan code of the key to be monitored and CX contains current shift states, as defined in Table 6-4. When the value in AL is FFH, this function returns with no operation.

Table 6-4. Register CX Current Shift States

CL REGISTER BIT	CH REGISTER BIT
0 — Right SHIFT key	0 — Left CTRL key
1 — Left SHIFT key	1 — Left ALT key
2 — Left CTRL key	2 — Right CTRL key
3 — Left ALT key	3 — Right ALT key
4 — SCROLL LOCK state	4 — SCROLL LOCK key
5 — NUM LOCK state	5 — NUM LOCK key
6 — CAPS LOCK state	6 — CAPS LOCK key
7 — INSERT LOCK state	7 —SYS REQ key

User Commands

To reach the Monitor program after the operating system boots, press CTRL-ALT-INS. The Monitor program opening message, similar to the following, will be displayed.

```
MFM-300 Monitor, version x.xx
Memory Size: 640K
Press "?" for help.
->
```

When the Monitor prompt (->) appears on the display, you may enter commands for the Monitor program. To display the Monitor command summary, enter a question mark and press the ENTER key. A display similar to the one shown in Figure 6-1 will appear. The paragraphs following Figure 6-1 explain each command.

```
                    - MFM-300 Command Summary -

CMD:  Explanation                  Syntax

?   Help                       ?
B   Boot from disk             B [{F|W}][{0|1|2|3}][:<partition>]
C   Color Bar                  C
D   Display memory             D [<range>]
E   Examine memory             E <addr>
F   Fill memory                 F <range>,{<byte>"<string>"}...
G   Execute (Go)                G [=<addr>][,<breakpoint>]...
H   Hex math                    H <number1>,<number2>
I   Input from port            I <port>
M   Move memory block          M <range>,<dest>
O   Output to port             O <port>,<value>
R   Examine Registers          R [<register>]
S   Search memory              S <range>,[<byte>|"<string>"}...
T   Trace program              T [<count>]
U   Unassemble program         U [<range>]
V   Set Video/Scroll           V [M<mode>][S<scroll>][100][200]
    Where <range> is:          <addr>{<addr>|L<length>}

TEST  Extended diagnostics TEST
SETUP Define Hardware Setup SETUP

    Copyright (C) 1988, by Zenith Data Systems
```

Figure 6-1. Monitor Command Summary Menu

The Monitor program contains three video commands and one boot command. These are summarized in Table 6-5 and described in the following paragraphs.

Table 6-5. Video and Disk Commands

COMMAND	DESCRIPTION	SYNTAX
?	Help	?
C	Display color bar	C
V	Set video/scroll mode	V [M<mode>] [S<scroll>]
B	Boot disk	B [FW] [0123]

Help

Syntax: ?

Example: ? ENTER

Use the help command to display a summary of the Monitor program commands. The summary contains a list of each Monitor program command, followed by its title and the syntax for entering the command. Enter the above example to display the command summary illustrated in Figure 6-1.

Color Bar

Syntax: C

Example: C ENTER

Use the color bar command to display a set of 16 color bars. Use the color bars to adjust a color video monitor and to adjust the contrast and brightness controls of the built-in LCD. Use the FN-F8 and FN-F9 key combinations to adjust the LCD palette.

Set Video/Scroll Mode

Syntax: V [M<mode>] [S<scroll>]

Example 1: `V M3 ENTER`
Example 2: `V S0 ENTER`
Example 3: `V M6 S2 ENTER`

Use the set video/scroll mode command to set one of twelve video modes and one of three scroll modes, as described in Table 6-6. The first example sets the display to video mode 3 (color, 80 characters by 25 rows). The second example sets the scroll mode to 0 (software-controlled scrolling). The third example sets the video mode to 6 (monochrome graphics, 640 by 200 resolution) and scroll mode 2 (hardware-controlled smooth scrolling).

Table 6-6. Video and Scroll Modes

MODE	DESCRIPTION
M0	40 characters by 25 rows text, monochrome display on the LCD screen or RGB output (16 displayable colors).
M1	40 characters by 25 rows text, color display at the RGB output or as a gray scale display on the LCD.
M2	80 characters by 25 rows text, monochrome display on the LCD screen or RGB output. Individual video pages may be scrolled without affecting other video pages.
M3	80 characters by 25 rows text, color display at the RGB output or as a gray scale on the LCD. Individual text pages may be scrolled.
M4	320 x 200 pixel resolution graphics, color display at the RGB output (four displayable colors) or as a gray scale on the LCD. The text display is 40 characters by 25 rows.
M5	320 x 200 pixel resolution graphics, RGB output (four displayable colors) or monochrome display on the LCD. The text display is 40 characters by 25 rows.
M6	640 x 200 pixel resolution graphics, monochrome display on the RGB output and on the LCD. All three scrolling modes are available. The text display is 80 characters by 25 rows.

Table 6-6 (continued). Video and Scroll Modes

MODE	DESCRIPTION
M7	720 × 350 pixel resolution graphics, monochrome text display. This mode requires a Zenith Data Systems expansion box with a TTL-compatible monochrome graphics adapter card. The text display is 80 characters by 25 rows.
MD	320 × 200 pixel resolution graphics, RGB output (16 displayable colors). This mode requires a Zenith Data Systems expansion box with an enhanced graphics adapter card. Text display is 40 characters by 25 rows.
ME	640 × 200 pixel resolution graphics, RGB output (16 of 64 colors are displayable). All three scrolling modes are available. This mode requires a Zenith Data Systems expansion box with an enhanced graphics adapter card. The text display is 80 characters by 25 rows.
MF	640 × 350 pixel resolution graphics, monochrome text display. This mode requires a Zenith Data Systems expansion box with a TTL-compatible monochrome graphics adapter card. The text display is 80 characters by 25 rows.
M10	640 × 350 pixel resolution graphics, monochrome text display. Four of 16 colors are displayable with 64K of video RAM. Sixteen of 64 colors are displayable with more than 64K of video RAM. This mode requires a Zenith Data Systems expansion box with an enhanced graphics adapter card. The text display is 80 characters by 25 rows.
S0	Software scrolling. This mode operates in all video modes and is the most common mode used in PC-compatible software. When material is scrolled, the display moves a line at a time.
S1	Hardware jump scrolling. This mode is available for video modes 3 through 6 and is similar in action to software scrolling. In this mode the hardware controls the scrolling action, rather than the software.
S2	Hardware smooth scrolling. This mode works only in video mode 6. When the material is scrolled, the display moves a partial line at a time providing a much smoother appearance to the scrolling action.

Boot Disk

Syntax: B [{F|W}][{0|1|2|3}][:<partition>]

Example 1: `B` ENTER
Example 2: `BF` ENTER
Example 3: `BW:1` ENTER

This command boots the operating system from a bootable disk in any drive. You may specify floppy or hard drives and the specific drive you wish to boot from.

In the first example, the computer will boot from the default drive. When the computer first powers up, the default drive will either be the hard disk drive or floppy disk drive A, depending on the settings in the Setup/ Configuration program. If you manually boot from another disk drive, that drive will become the new default drive. The default setting will remain the same until you boot from a different drive or turn the computer off. If the computer is turned off, the default drive once again becomes the boot drive.

In the second example, the computer will boot from the floppy disk drive. The F in the syntax specifies the internal 3.5-inch floppy disk drive.

The drives are numbered 0, 1, 2, and 3, with 0 assigned to the first drive. The partitions are numbered 1, 2, 3, and 4, with 1 assigned to the first partition. The letter W indicates a hard disk drive. In the third example, the computer will boot from the first partition on the hard disk drive.

Machine Language Debugger

The Monitor program's machine language debugging commands are listed in Table 6-7. The syntax in this table is clearer than the one shown in the command summaries illustrated in Figure 6-1. Examples and descriptions follow the table.

The message `^ Invalid Command` is the Monitor program's syntax error message. The caret (^) points to where the first syntax error occurred. The Monitor program does not recognize additional syntax errors after the first error is detected. Also, the first error does not lock up the computer. Therefore, the command can be re-entered or another command entered instead.

Table 6-7. Machine Language Debugger Commands

COMMAND	SYNTAX	DESCRIPTION
Display memory	D<address>	Display contents of 128 bytes of memory, beginning at specified addresses.
	D<address>L<bytes>	Displays contents of specified number of bytes memory beginning at specified address.
	D<range>	Displays contents of specified block of memory.
Examine memory	E <address>	Displays and allows user to alter contents of specified memory location.
Fill memory	F<range>,<data byte>	Enters specified data byte into each memory location in specified memory block.
	F<range>,"ASCII string"	Enters specified ASCII string into specified memory block.
Go (Execute)	G=<address>	Begins execution of program at specified address.

Table 6-7 (continued). Machine Language Debugger Commands

COMMAND	SYNTAX	DESCRIPTION
	G=<address>, <breakpoint>	Begins execution of program at specified address and halts at breakpoint
Hex math	H<number1>,<number2>	Displays the sum and difference of the specified hexadecimal numbers.
Input from port	I<port address>	Displays contents of specified port.
Move memory block	M<range>,<destination>	Copies contents of specified memory block to another specified memory block.
Output to port	O <port address>,<data>	Writes specified data to specified port address.
Examine Registers	R	Displays contents of all CPU registers.
	R<register name>	Displays contents of specified CPU register and allows modification of contents.
Search memory	S<address>,L<bytes>, <data>	Searches specified memory block for specified data byte and displays address data found.
	S<range>L<bytes>, "ASCII"	Searches specified memory block for specified ASCII character and displays address character found.
Trace program	T<count>	Executes specified number of lines of an assembled program in single-step mode.
Unassemble	U<range>	Displays assembler mnemonics and hex coding for specified memory block.

Display Memory

Syntax: D <address>[L<length> <offset>]

Example 1: `D 1000:0 F` ENTER
Example 2: `D DS:7000 L200` ENTER

Use this command to display the contents of the specified portion of memory. Memory contents cannot be altered by this command. If the length is not specified in the command line, the debugger will display 128 bytes of data starting at the specified address. The resulting display contains three sections. The left column is the base address and offset of the first byte in the second part of the display. The second part uses hexadecimal format to display 16 bytes of data. The right part of the display contains the ASCII characters represented by the data in the second part of the display. If any byte is outside the ASCII character range, a period will be printed in its place.

If the first example is entered, a display of 16 bytes of memory will result, starting at address 1000:0. If that portion of memory is empty (containing only nulls or 0's), the resulting display will be:

```
1000:0000 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
```

If the second example is entered, the resulting display will contain 200H bytes of memory. This display will start at the address offset located 7000H bytes from the base address of the data segment. The base address is stored in CPU segment register DS.

If the memory range is greater than can be displayed on the screen, the data will scroll. Use CTRL-S to stop and CTRL-Q to continue the screen action.

Examine Memory

Syntax: E <address>

Example: `E 1000:100` ENTER

Use the examine command to examine or change memory one byte at a time. When this command executes, the display will show the byte at the specified memory address. The command will wait for an entry from the keyboard. Enter a hexadecimal value from 0 to FF to modify the current memory location. Press the space bar to examine the next byte of memory or the minus (hyphen) key to display the previous byte of memory. Press the ENTER key to complete this command and return to the Monitor program prompt.

You can examine and modify any addressable memory location, including those reserved for interrupts and interrupt vectors. Therefore, be careful that you do not accidentally modify routines that the Monitor program uses or you could lock up the computer. If this occurs, turn the computer off to reset it.

Table 6-8 provides an overall map of the system's addressable memory. System memory at 00000H - 9FFFFH is divided between the Monitor program and operating system, as shown in Table 6-8. The information presented in this table is subject to change as new products are introduced for these computers.

Table 6-8. System Memory Map

ADDRESS RANGE	DESCRIPTION
000000 - 09FFFF	System RAM
0A0000 - 0BFFFF	Video RAM
0C0000 - 0CFFFF	Expansion ROM/RAM
0D0000 - 0DFFFF	EMS or expansion ROM/RAM
0E0000 - 0EFFFF	OEM ROM (slushware)
0F0000 - 0F0FFF	Scratchpad RAM
0F1000 - 0FEFFF	Monitor ROM (slushware)
100000 - FDFFFF	Extended RAM
FE0000 - FEFFFF	OEM ROM (non-slushed)
FF0000 - FF0FFF	Monitor ROM (non-slushed)

Fill Memory

Syntax: F <start address>,<range>,<data list>

Example: `F 1800:0,1FF,"TEST",20,54,45,53,54,20` ENTER

Use the fill command to enter one or more bytes of data directly into memory. You can use ASCII characters, delineated by quotation marks, or hexadecimal format. Each byte and ASCII text word requires a comma for separation. Data in the list will be reused as often as necessary to fill the specified memory range. You must specify the starting address, range, and data before the command will function (refer to Table 6-8). In the example, 1FFH bytes, starting at memory address 1800:0, will fill with repeating strings of the data. The entire string is used, which includes the material in quotes plus the material expressed as hexadecimal values. The result of this example is the string TEST TEST.

You can use the fill command to modify any addressable memory range, including those reserved for interrupts and interrupt vectors. Therefore, be careful that you do not accidentally modify routines that the Monitor program uses or you could lock up the computer. If this occurs, turn the computer off to reset it.

Go (Execute)

Syntax: G[=<address>] [,<breakpoint>]...

Example 1: `G 1000:0` ENTER
Example 2: `G 100` ENTER
Example 3: `G=1000 100` ENTER

You can use the go command to transfer control from the Monitor program to a machine language or user program. The transfer will occur at the address specified in the command line or in the instruction pointer and code segment of the CPU. Also, as a debugging aid, you may specify breakpoints (points in the program where you want the program to halt) in the command line.

In the first example, control will transfer to the address specified in the CPU registers and a breakpoint will be set at memory address location 1000:0. In the second example, the Monitor program will transfer control to the address specified in the CPU registers and set a breakpoint at the memory address offset. This address is located 100H bytes from the location of the first byte of code to be executed. In the third example, the Monitor program sets the code segment register to 1000. Control is then transferred to this address and the Monitor program sets a breakpoint at memory offset 100H.

When the go routine encounters a breakpoint, the Monitor program will save the status and register information. This information is displayed as a register dump (refer to the examine/modify registers command). The screen will display the status of the CPU's registers and flags as well as the current instruction executed just prior to the breakpoint.

You may set up to eight breakpoints and the debugger will halt when it encounters one of them. Once a breakpoint is found, the debugger will not remember the other breakpoints, since they are stored as parameters to the command line. You must set breakpoints to addresses that contain valid instruction codes, otherwise the computer will lock up.

Hex Math

Syntax: H <number1>,<number2>

Example: `H 2A,28 ENTER`

Use the hex math command to compute the sum and difference of two hexadecimal numbers. The debugger always subtracts the second number from the first. In the example, the results will display as `SUM: 0052 Diff: 0002.`

Input from Port

Syntax: I <port address>

Example: `I 3F8 ENTER`

Use the input command to obtain a byte of data from a port address. In the example, a single byte from port 3F8H will be displayed on the screen. You can address most programmable devices used in this computer through port addresses. Table 6-9 is a general map of the system's ports. The information presented in this table is subject to change as new products are introduced for these computers. The values represented by question marks vary, according the version of MS-DOS and the user-installed .COM and .EXE files.

Table 6-9. System Port Map

PORT ADDRESSES	DESCRIPTION
000H - 00FH	8-bit DMA controller (slave)
020H - 03FH	Master interrupt controller
040H - 05FH	System interval timer
060H - 06FH	System control processor (even addresses) and system control port (odd addresses)
070H - 07FH	NMI enable/RTC address port (even addresses) and RTC data port (odd addresses)
080H - 08FH	DMA page register
0A0H - 0BFH	Slave interrupt controller
0C0H - 0DFH	16-bit DMA controller (master)
0E0H - 0FFH	Reserved
170H - 177H	Secondary hard disk drive controller
1F0H - 1F7H	Primary hard disk drive controller
278H - 27FH	Parallel port 2 (LPT2)
2F8H - 2FFH	Serial port 2 (COM2)
370H - 377H	Secondary floppy disk drive controller
378H - 37FH	Parallel port 1 (LPT1)
3B0H - 3BFH	Monochrome video monitor
3D0H - 3DFH	Color graphics controller
3F0H - 3F7H	Primary floppy disk drive controller
3F8H - 3FFH	Serial port 1 (COM1)

NOTE: These computers do not support a light pen port. Some computer games use this feature and may not operate correctly.

Move Memory Block

Syntax: M <start address>,<range>,<destination address>

Example: `M 0:1000,107F,0:2000 ENTER`

Use the move command to copy a specified block of memory to another specified location in memory (refer to Table 6-8). In the example, the Monitor program will copy a block of memory, 107FH bytes long starting at memory address 0:1000, to a memory location that starts at 0:2000.

NOTE: You can use the move command to move a block of memory from any addressable memory location to any other addressable memory location, including one reserved for interrupts and interrupt vectors. Therefore, be careful that you do not accidentally modify memory that contains routines used by the Monitor program or you could lock up the computer. If this occurs, turn the computer off to reset it.

Output to Port

Syntax: O <port address>,<data>

Example: `O 2F8,00 ENTER`

Use the output command to send a byte of data to a port address. In the example, the Monitor program will send the value of 0H to port address 2F8H. The computer can address most programmable devices through port addresses.

Examine Registers

Syntax: R [<register>]

Example 1: `R ENTER`
Example 2: `R CS ENTER`
Example 3: `R FL ENTER`

Use the examine registers command to examine and alter the contents of the CPU's registers and flags. In the first example, the computer performs a register dump. The dump will display the current instruction along with the contents of all registers and flags.

In the second example, the display will contain the contents of the CS register and the computer will wait for an entry from the keyboard. Enter a hexadecimal value to modify the contents of the current register. If the value you enter is not valid, the register's contents will not change. Press the ENTER key alone to leave the contents of the register unchanged.

In the third example, the display will contain the status flags. Table 6-10 contains the two-letter abbreviation that describes the status of each flag. The computer will wait for an entry from the keyboard. Enter the two-letter abbreviation to set the status of a flag, followed by a space and another two-letter abbreviation for another flag or by the ENTER key to terminate the command. If you make more than one entry for the same flag, the last value entered will be the one that remains. Press only the ENTER key to leave the flags unmodified.

Table 6-10. Processor Status Flag Codes

FLAG	ON	OFF
Auxiliary Carry	AC (auxiliary carry)	NA (no auxiliary carry)
Carry	CY (carry)	NC (no carry)
Direction	DN (down)	UP (up)
Interrupts	EI (enabled)	DI (disabled)
Overflow	OV (overflow)	NV (no overflow)
Parity	PE (even)	PO (odd)
Sign	NG (negative)	PL (plus)
Zero	ZR (zero)	NZ (not zero)

NOTE: This above table does not represent all flags available within the 80386 microprocessor.

Search Memory

Syntax: S <start address>,<range>,<data list>

Example: `S 0:2000,1000,"TEST",20,"TEST"` ENTER

Use the search command to search the specified area of memory for the occurrence of specific data strings or bytes of information. In the example, the search routine will examine 1000H bytes of memory starting at memory location 0:2000 for the occurrence of TEST TEST. Notice that quotation marks enclose the ASCII text; otherwise, you must enter data in hexadecimal format. You must separate each item in the data list by a comma from the other items in the list. If you do not provide any data in the list, the Monitor program will return the `^ Invalid Command!` error message.

Trace Program

Syntax: T [=<address>] [,<value>]

Example 1: `T=400:0,200` ENTER
Example 2: `T 5` ENTER

Use the trace command to single-step the execution of a program. Each time you execute the command, it executes the next instruction and performs a register dump. The dump will display the current instruction and the contents of the registers and flags. Enter T by itself to execute the instruction pointed to by the current code segment and IP register. Use the examine/modify registers command to change these values and to specify a new memory location for trace operation.

In the first example, the routine places the value following the equal sign (=) in the CS and IP registers. Trace will use this value as the starting address and will execute 200H times. Since this is a large number, the register dumps will fill and scroll the screen. Use CTRL-S to stop and CTRL-Q to start the scrolling action.

In the second example, trace will start at the current address stored in the CS and IP registers of the CPU. The command will trace the next five instructions.

CAUTION

The trace command does not contain extensive error trapping routines to prevent accidental operation of the computer. It is possible that timing-dependent operations, such as reading or writing to a disk could cause problems and even damage the media. Therefore, do not use trace to step through the Monitor program or interrupt service routines.

Unassemble Program

Syntax: U [<address>[<length>|,<offset>]

Example 1: `U` ENTER
Example 2: `U 1000` ENTER
Example 3: `U 0:1000` ENTER

Use the unassemble command to disassemble a section of memory into assembly language mnemonic format. The resulting display resembles an assembly language source code listing but contains no comments. The Monitor program will treat data areas that contain ASCII text as code. Therefore, it is possible to create meaningless assembly language code from raw text or other data.

In the first example, the routine will begin disassembly in the current code segment (pointed to by the CS register) at the address pointed to by the IP register. The number of lines displayed on the screen will vary depending upon the number of bytes generated by each instruction. In the second example, disassembly starts at the memory location offset 1000H bytes from the value of the CS register. In the third example, disassembly starts at memory location 0:1000. In all three cases, the routine will disassemble and display 32 (20H) bytes of data. You can also use the length parameter to disassemble more or less than 32 bytes of memory.

Since this may result in a display that scrolls the screen, use CTRL-S to stop and CTRL-Q to start the scrolling action.

Although unassemble modifies the values in the CS and IP registers, the routine stores the original values in a stack. If you use the examine/modify registers command, the routine will restore the original values and subsequent use of the unassemble command will use these values.

Interrupts

An interrupt directs the microprocessor to stop execution of the current program and execute an interrupt service routine or an interrupt handler. In most cases, upon completion of the interrupt service routine, the microprocessor is directed back to the program that was interrupted and continues execution of the interrupted program.

At the end of each instruction cycle, the microprocessor checks to see if it has received an interrupt request. The microprocessor recognizes interrupt requests initiated by any of the following sources:

- Asynchronous external interrupts (hardware interrupts) initiated by particular hardware devices and processed through the interrupt controller circuitry.
- Program interrupts initiated by an INT xxH instruction.
- Internal interrupts generated by the CPU.

If it has been interrupted, the microprocessor pushes the contents of the instruction pointer, code segment and flag registers onto the stack. No other register values are automatically saved. (If additional register contents are required to be saved, the service routine must save them.) Next, the interrupt flag and the trap flag are reset, preventing any other interrupt requests from being recognized until the current interrupt request has been serviced. The CPU then reads the interrupt vector provided with the interrupt request. The

interrupt controller circuits provide the vector for asynchronous external interrupt requests. These interrupt vectors are programmed during system initialization. The CPU generates its own interrupt vector for internal CPU interrupt requests. The xxH value in the INT instruction provides the vector for program interrupt requests.

The starting addresses for all interrupt service routines are stored in an interrupt vector table located in the first 1K of system memory (memory addresses 000000H-0003FFH). When the processor is operating in real mode, the addresses are loaded directly into the instruction pointer and the code segment registers from the table. These addresses are obtained from a descriptor table if the processor is in protected mode. To obtain the specific address pointer location, the interrupt vector is multiplied by four if the processor is operating in real mode or by eight if the processor is in protected mode. The result provides the location of the first byte of the starting address for the service routine.

Table 6-11 lists the interrupt vectors used by the TurbosPort computer. The table also describes the function of the service routine and indicates the chapter where the particular interrupt service routine is discussed.

Table 6-11. Interrupt Vector Summary

INTERRUPT VECTOR	CHAPTER	FUNCTION
00H	7	Divide by zero
01H	7	Single step
02H	7	Nonmaskable interrupt (NMI)
03H	7	Software breakpoint
04H	7	Arithmetic overflow
05H	9	Print screen
08H	7	Timer (time-of-day)
09H	8	Key pressed
0AH	7	Programmable interrupt controller
0BH	9	Communications (COM2)
0CH	9	Communications (COM1)
0DH	9	Alternate parallel printer (LPT2)
0EH	10	Floppy disk hardware return
0FH	9	Parallel printer (LPT1)
10H	11	Video input/output
11H	7	Equipment configuration
12H	7	Memory size
13H	10	Floppy and hard disk input/output call
14H	9	Serial input/output
15H	7	Device control
16H	8	Keyboard input/output
17H	9	Printer input/output
18H	9	Parallel/serial configuration
19H	10	Operating system boot routine call
1AH	7	Set/read the time of day
1BH	8	Keyboard break
1CH	7	Tick timer
1DH	11	Video initialization
1EH	10	Floppy disk parameters
1FH	11	Defining characters

Following execution of the interrupt service routine the instruction pointer, code segment, and flag register contents are restored.

Hardware Interrupt Requests

Programmable interrupt controller circuitry manages 16 external asynchronous interrupt requests. When two or more requests for an interrupt occur at the same time, the requests are handled on a priority basis. The highest priority is assigned to IRQ0 and the lowest to IRQ15. Table 6-12 describes the hardware interrupt requests and, where defined by the system, the interrupt vectors for their service routines.

Table 6-12. Hardware Generated Interrupt Requests

HARDWARE INTERRUPT REQUEST	INTERRUPT VECTOR	USUAL SOURCE OF INTERRUPT
IRQ0	08H	Time-of-day timer
IRQ1	09H	System control processor
IRQ2	0AH	Slave controller interrupt
IRQ3	0BH	Communications (COM2) or PC bus
IRQ4	0CH	Communications (COM1) or PC bus
IRQ5	0DH	Parallel printer (LPT2) or PC bus
IRQ6	0EH	PC bus
IRQ7	0FH	Parallel printer (LPT1) or PC bus
IRQ8	70H	Real-time clock
IRQ9	71H	IRQ2 on PC bus
IRQ10	—	AT bus
IRQ11	—	AT bus
IRQ12	—	AT bus
IRQ13	—	Coprocessor interrupt, 80387
IRQ14	—	AT bus
IRQ15	—	AT bus

When one or more of the IRQ inputs to the interrupt controllers goes high, the master interrupt controller sends an interrupt request to the CPU by sending the active high INTR signal to the INTR input of the microprocessor. This sets the IF bit in the flag register. At the end of the current instruction cycle, the CPU checks the flag register and responds by sending the interrupt acknowledge signal INTA to

the controller. The interrupt controller then places the interrupt vector for the highest priority interrupt request received onto the data bus for the CPU to read.

Masking Hardware Interrupt Requests

For some applications, you may want to disable or mask particular hardware interrupt requests. Hardware interrupt requests may be masked by outputting a mask interrupt byte to the interrupt controller circuitry. IRQ0 through IRQ7 are handled by the interrupt controller and are masked using port 21H. IRQ8 through IRQ15 are handled by the interrupt controller and are masked using port A1H.

Each bit in a mask interrupt byte corresponds directly to an interrupt request, with the least-significant bit representing the highest priority interrupt handled by the controller. A 1 in a bit position masks the corresponding interrupt request. A 0 in a bit position unmasks the corresponding interrupt request. For example, the mask interrupt byte 01H (with a 1 as the least-significant bit) masks IRQ0 when output to port 21H and masks IRQ8 when output to port A1H.

Internal CPU Interrupts

Internal CPU interrupt requests are used to process unusual conditions that prevent further instruction processing during execution of a program. These include invalid division errors and a low to high transition on the NMI (nonmaskable interrupt) pin. An overflow error may also cause the CPU to interrupt itself if the INTO (interrupt if overflow) instruction is used for error handling in the program instead of JO (jump if overflow) instruction.

When these conditions occur, the CPU automatically:

1. Interrupts itself

2. Saves the flag, instruction pointer, and code segment register contents

3. Generates and processes its own interrupt vectors

4. Loads the service routine address from the interrupt vector table

5. Executes the service routine.

The interrupt vectors for these routines are listed in Table 6-11. Application programs may replace the service routines initialized by DOS or firmware to handle these interrupts with custom error handling routines.

Program Interrupts

The INT instruction allows any program to interrupt the CPU and cause it to execute a service routine. This instruction can be used for single-stepping through a program, inserting user breakpoints, testing interrupt service routines, and calling specific procedures (such as reading data from a disk) from application programs. The remaining chapters in this part of the manual provide more information about specific INT instructions.

Programming Interrupt Service Routines

This section provides some general guidelines for using INT instructions and writing interrupt service routines.

In many interrupt service routines, parameters are passed to and from the routine through the CPU registers. Before calling the INT instruction, be sure to save the contents of the registers that will be modified by the interrupt routine if the original values are needed to

complete execution of the program. Remember that only the IP, CS, and flag register contents are automatically saved when the CPU is interrupted. The original values must also be restored upon return from the interrupt service routine, since the IRET instruction only restores the registers that were automatically saved when the CPU was interrupted.

If your interrupt service routine is replacing or being used in addition to an existing interrupt service routine, you must patch the interrupt vector with the address of your routine. The MS-DOS function requests Get Interrupt Vector (35H) and Set Interrupt Vector (25H) may be used to patch the vector. For more information about these function requests, refer to the Programmer's Utility Pack for your version of MS-DOS.

If your interrupt service routine is replacing an existing service routine, execute an IRET instruction at the end of your routine. If your interrupt service routine is being used in addition to an existing service routine, do not execute an IRET instruction at the end of your routine. Instead, execute a JMP DWORD PTR to the existing routine's starting address.

Chapter 7

CPU Interrupts

This chapter describes the CPU interrupts defined in Table 7-1. For information about the use and programming of interrupts, see Chapter 6, "Software Interface."

Table 7-1. CPU Interrupts

INTERRUPT	FUNCTION
00H	Divide by Zero
01H	Single Step
02H	Non-Maskable Interrupt (NMI)
03H	Software Breakpoint
04H	Arithmetic Overflow
08H	Time-of-Day Timer
0AH	Real-time Clock
11H	Equipment Configuration
12H	Memory Size
15H	Device Control
1AH	Set/Read Time-of-day
1CH	Tick Timer
4AH	User Alarm
70H	RTC Alarm

Divide by Zero (INT 00H)

INT 00H (divide by zero) will be executed if a divide instruction produces a quotient too large to fit in the result register (such as dividing a value by 0). The error routine will print `Divide Overflow` and return control to the operating system. This routine is initialized by the DOS. If you intend to use this routine other than with the operating system, you must set up the vector to intercept DIV and IDIV instructions that can result in exceptions to the rule (for instance, BASIC does this to retain control inside BASIC).

Single Step (INT 01H)

INT 01H (single step) is used for executing a single machine instruction at a time. It is called by the 80386 when an instruction is executed with the trace flag (TF) set. It is most commonly used by routines such as the MS-DOS DEBUG command and the MFM-300 trace command. Since this routine is initialized by the command (MS-DOS or MFM-300) calling it, you must define the routines to be executed when you establish the conditions and routines that will call this interrupt.

Non-Maskable Interrupt (INT 02H)

INT 02H is the non-maskable interrupt (NMI). It is initiated by hardware external to the 80386, but in these computers, it usually indicates a power-down condition has started. Although it is off at powerup, it usually is not disabled except by design or by sending 00H to the input/output port A0H. To enable the NMI, send 80H to the input/output port A0H..

NOTE: The 80387 numeric processor extension uses this interrupt in its normal operation.

Software Breakpoint (INT 03H)

You may run a program until the processor encounters a breakpoint (an INT 03H instruction). Usually, the MFM-300 debugging routines or MS-DOS DEBUG will allow the user to set up breakpoints and subsequently will handle the condition accordingly (by usually returning control to the last command level).

Arithmetic Overflow (INT 04H)

The INT 04H instruction is executed from an INTO instruction when the overflow flag (OF) is set. The flag is set (usually) by a previous arithmetic or logic operation.

Time-of-Day Timer (INT 08H)

The INT 08H instruction is used to keep track of the time-of-day for the system. It also calls the INT 1CH instruction (Tick Timer).

The programmable interval timer circuitry initiates this interrupt 18.2159 times per second, or every .054897095 seconds. This timer circuitry is not affected by the CPU clock speed and is used for such functions as keeping track of the time-of-day, timing out the disk motors, and calling the tick timer interrupt (1CH).

The timer keeps track of the time-of-day in a 32-bit word (sometimes called a double-word). When the count reaches approximately 1803D8H (1,573,848 decimal), a lag is set to 1 to indicate that the timer has rolled past midnight to a new day. Interrupt 1AH is used to set and/or read the value of this word. This interrupt is established by the hardware and is IRQ0.

Real-Time Clock (INT 0AH)

INT 0AH is the real-time clock alarm interrupt. The real-time clock starts when a hardware interrupt request (IRQ3) or when the alarm from the real-time clock IC takes place. Function 83H (event wait) and function 86H (wait) of INT 15 set and handle the alarm. The hardware interrupt request takes place approximately 1,024 times a second.

Most systems do not use this interrupt. The hardware within the computer essentially controls this interrupt. This makes the interrupt unique to individual machines such as this computer. Because of this fact, it should not be called by user programs.

Equipment Configuration (INT 11H)

The INT 11H instruction can be used to report back the configuration of the equipment as established by the Setup/Configuration program and stored in CMOS RAM (see Chapter 4). The report is returned in a 16-bit word (register AX).

Since this interrupt is common to PC AT-compatible equipment, you need to be aware of all possible responses and what they mean. Refer to Table 7-2 for the definitions of each bit of the word returned in register AX.

Table 7-2. Equipment Configuration (Register AX Report from INT 11H)

BIT NUMBER	DESCRIPTION
0	Floppy disk drives are installed in the system if set to 1.
1	The 80387 numeric processor extension is installed in the system if set to 1.
2-3	These two bits indicate the device size of RAM chips installed in the computer. Bits 2 and 3 are always set to 1 in Zenith Data Systems computers.
4 and 5	Initial video mode at powerup.
0 0	80 × 25 text mode on an EGA card (requires expansion chassis).
0 1	40 × 25 text mode.
1 0	80 × 25 text mode.
1 1	80 × 25 text mode on a monochrome card (requires expansion chassis).
6 and 7	Reports the number of floppy disk drives available if bit 0 is set.
0 0	1 drive.
0 1	2 drives.
1 0	3 drives.
1 1	4 drives.
8	Unused in Zenith Data Systems computers.
9, 10, and11	Reports the number of RS-232 ports in the system in binary format. The standard input/output on this system emulates the PC AT input/output, so the minimum will be one port (0 0 1).

Table 7-2 (continued). Equipment Configuration (Register AX Report from INT 11H)

BIT NUMBER	DESCRIPTION
12	If set (1), a game card is present (requires expansion box).
13	Unused by Zenith Data Systems computers.
14 and 15	Reports the number of printers installed in binary format.

Memory Size (INT 12H)

The INT 12H instruction returns in register AX the number of contiguous 1K blocks of user memory in the system, as established by data found in CMOS RAM. For instance if the value in AX is 256 following this interrupt, there is 256K of base memory installed in the system.

Device Control Interrupt (INT 15H)

The device control interrupt (15H) allows the programmer various functions such as placing the 80386 CPU into protected mode, accessing memory above the 1 megabyte real mode limit, and so on. The particular function chosen depends on the value of the AH register when the interrupt is executed. Table 7-3 lists the functions. Those that are marked as "user trappable" indicate that application programs do not normally make these function requests and, therefore, user programs may intercept interrupt 15H and trap these functions as needed. Regardless of the use of these functions made by other software, MFM-300 will execute these functions at the times described in the text following the table.

Note that "devices" and "processes" are defined by software, most notably an operating system environment, such as that provided by MS-DOS or XENIX.

Table 7-3. Interrupt 15H Functions

FUNCTION (AH) CODE	DESCRIPTION
00H - 7FH	Reserved — unused by MFM-300.
80H	Device open — user trappable.
81H	Device close — user trappable.
82H	Program terminate — user trappable.
83H	Event wait.
84H	Joysticks support.
85H	System request key — user trappable.
86H	Wait.
87H	Block move.
88H	Determine extended memory size.
89H	Place processor in virtual mode.
8AH - 8FH	Reserved — unused by MFM-300.
90H	Device busy — user trappable.
91H	Set interrupt complete flag — user trappable.
92H - FFH	Reserved — unused by MFM-300.

Function Code 80H: Device Open — MFM-300 will generate this interrupt whenever a device is opened. The device ID (identification) is placed in register BX and the process ID is placed in register CX. Both identification codes must be defined in the operating system environment or the program. Normally, this handler provides only an RET operation. However, this interrupt can be used to detect open devices.

Function Code 81H: Device Close — MFM-300 will generate this interrupt whenever a device is closed. The device ID is placed in register BX and the process ID is placed in register CX. Both identification codes must be defined in the operating system environment or the program. Normally, this handler provides only an RET operation. However, this interrupt can be used to provide the process ID. This interrupt can be used to detect closed devices.

NOTE: Both function code 80H and 81H require that the same information be placed in register BX and register CX. The most common use for these two functions would be in a multiuser or multifunction environment where detection of open and closed devices is critical to dependable operation of the multiple tasks being handled by the computer.

Function Code 82H: Program Termination — This interrupt is used to identify devices that are in use by a program when that program terminates. The device ID must be in register BX. The intended use for this interrupt is to automatically deallocate devices that are no longer in use, similar to the closing of open files at the end of a BASIC program.

Function Code 83H: Event Wait — This function enables a timer-based delay. After the specified number of microseconds has elapsed (this number is loaded in registers CX:DX), the high order bit of the flag byte pointed to by the address in ES: BX will be set to a 1.

NOTE: The firmware does not wait for the time to elapse before returning control to the calling routine. Instead, control is returned immediately. It is up to the program to monitor the flag byte to determine when it has been set by the function.

Function Code 84H: Joystick Support — This function provides joystick support through port 201H. When the value in register DX is 0 (zero), the current joystick switch settings are returned in register AL (bits 7 - 4). When the value in DX is 1, the values representing the x-axis and y-axis position of the joystick shown in Table 7-4, are returned.

Table 7-4. Joystick Values

REGISTER	DESCRIPTION
AX	Joystick #1 x-axis value
BX	Joystick #1 y-axis value
CX	Joystick #2 x-axis value
DX	Joystick #2 y-axis value

Function Code 85H: System Request Key — This interrupt takes place each time the SYS REQ key is pressed or released. The value returned in AL reports the action of the key (pressed or released). If the value in AL is 0, the key was pressed. If the value in AL is 1, the key was released.

Function Code 86H: Wait — This interrupt will cause the program to wait the number of microseconds stored in CX:DX. Control will be returned to the caller only after the indicated time has elapsed.

Function Code 87H: Block Move — The interrupt will cause a specified block of memory to be moved from one location to another. The amount of memory (measured in 16-bit words) is stored in register CX. The six 8-byte global descriptor tables (GDT) are used to describe to source and destination addresses, along with other information. The function of each GDT is described in Table 7-5. The value in the register combination ES:SI points to the beginning of the descriptor block. During the operation, the 1 megabyte real mode limit can be addressed. The structure of each GDT is described in Table 7-6.

Table 7-5. Block Move Global Descriptor Tables

BYTE OFFSET ADDRESS RANGE	DESCRIPTION
00H - 08H	Required dummy descriptor table (set to zero).
09H - 0FH	Pointer to the beginning of these tables as a data segment (set to zero).
10H - 18H	Source to be moved descriptor table.
19H - 1FH	Destination descriptor table.
20H - 28H	Descriptor table (to be used by function to create a protected code segment — set to zero).
29H - 2FH	Descriptor table (to be used by function to create a protected stack segment — set to zero).

Table 7-6. Global Descriptor Table Structure

BYTES	DESCRIPTION
00H and 01H	Number of bytes to be moved.
02H, 03H, and 04H	The 24-bit physical address of the first byte in LSB form.
05H	Access rights byte.
06H and 07H	Reserved, must be set to zero.

The maximum number of bytes that can be moved is 8000H (32K).

Function Code 88H: Determine Extended Memory Size — MFM-300 will return a value in register AX that represents the amount of contiguous memory installed above the 1 megabyte real mode memory limit. The amount of memory is indicated in 1024 byte increments.

Function Code 89H: Processor to Virtual Mode — This function will place the 80386 CPU into its protected mode. ES:SI contains the base address of the global descriptor table to be used; BH and BL provide the base address of the interrupt vector table for the interrupt controller circuitry.

This function expects the address in the registers ES:SI to point to the global descriptor table that will be used by the 80386 while in protected mode. In addition, the function will reprogram the interrupt controller circuitry, setting the base interrupt vector values to those stored in BH and BL. The contents of global descriptor table for this function is described in Table 7-7.

NOTE: Once the 80386 is in protected mode, the MFM-300 interrupt routines are no longer addressable. Therefore, the code segment in protected memory must contain all interrupts and routines needed by the program.

Table 7-7. Global Descriptor Tables for Function Code 89H

BYTE OFFSET ADDRESS RANGE	DESCRIPTION
00H - 08H	Required dummy descriptor table (set to zero).
09H - 0FH	Pointer to the beginning of these tables as a data segment.
10H - 18H	Pointer to the interrupt descriptor table (IDT).
19H - 1FH	Pointer to the DS (data segment) in protected memory.
20H - 28H	Pointer to the ES (extra segment) in protected memory.
29H - 2FH	Pointer to the SS (stack segment) in protected memory.
30H - 38H	Pointer to the CS (code segment) to which the function will return. Therefore, it should be initialized to the CS of the original program. Upon completion of the function, the control will return to the code segment specified using the IP (instruction pointer) that was placed on the stack when the INT 15, function code 89, was called.
39H - 3FH	Pointer to the CS (code segment) that the function will use while in protected mode.

The format of each descriptor table described in this table is the same as the one described in Table 7-6, which is located in the discussion for function code 87H. The access rights bytes will be different for the CS and IDT entries.

Function Code 90H: Device Busy — Before MFM-300 enters a timing loop waiting for a given device, it will generate this interrupt function for that device. Register AL must provide the device type code. User programs that wish to perform some function during this "busy loop" can take advantage of this interrupt function. Table 7-8 contains the description of type codes.

Table 7-8. Device Type Codes

TYPE CODE OR RANGE	DESCRIPTION
00H - 7FH	These are serially reusable devices. The operating system must provide serial access for these devices which include the following.
00H	Hard disk access, timeout supported.
01H	Floppy disk access, timeout supported.
02H	Keyboard, no timeout support.
80H - BFH	These are reentrant devices; typically, a network. The registers ES:BX is used to distinguish different I/O calls, which are allowed to be placed simultaneously.
80H	Network, no timeout support. ES:BX must point to the network control block (NCB).
C0H - FFH	These are "wait-only" codes that have no source for the "wait." The timeout length of the wait is the function number. These codes include the following.
FDH	Floppy disk motor startup time.
FEH	Printer response time.

Function Code 91H: Set Interrupt Complete Flag — This function advises the user when the ROM has completed some operation or function. The AL register will contain the appropriate code from those listed in Table 7-8.

Functions 80 - 91 support operations that are available on systems based on either the 80286 or 80386 processor. The extended functions, listed in Table 7-9, are only available on 80386-based computer systems.

Table 7-9. INT 15 Extended Functions

FUNCTION	DESCRIPTION
E2H	Ram protect enable/disable
E3H	Speed control
E4H	Current speed setting
E5H	Enable/disable cache memory
E6H	Read cache state

Function Code E2H: Ram Protect Enable/Disable — This function is used to enable and disable protection for the Monitor ROM work area. The Monitor ROM maintains a 1K memory block for various pointers and variables used during normal operations. If this area is accidentally written too by the system, unpredictable results or even a system halt could result.

When entering this function, the AL register contains information on the level of protection. If the AL register is zero, the ROM will increment the protection level. If the value in AL is 10, the ROM will decrement the protection level.

Protection for this area of RAM is implemented in a stack area, similar to a last-in, first-out (LIFO) stack. The protection level is incremented or decremented depending on the value in the AL register. A byte in memory is used by the ROM to maintain this count. Whenever the value changes from 1 to 0, the RAM will be protected. If the byte changes from 0 to 1, protection is disabled. The protection state of the RAM will only be affected by transitions between the values of 0 and 1. The stack will support a maximum of 255 levels. If this value is exceeded, the byte will reset to 0 (wrap around), causing unpredictable results.

Function Code E3H: Set Processor Speed — This function sets the operating speed of the system processor. The AL register contains the required setting information according to the following list:

- AL = 0 : Set processor to run fast at all times
- AL = -1 : Set processor to run slow at all times
- AL = 1 : Normally fast, but run slow for floppy disk accesses

The preferred operational mode for the 80386-based computer is fast at all times. It is possible that some applications software may have problems at the normal operating speed of this computer. In those cases you may try setting the processor to run fast except when attempting a floppy disk access. If there is a known problem with a particular package, you may have to set the processor for slow operation.

Function Code E4H: Read Processor Speed — This function returns the current operating speed of the system processor. The AL register contains the returned value. This value will correspond exactly with the speed codes used by function E3H.

Function Code E5H: Enable/Disable Cache Memory — This function permits the user to selectively turn cache memory on or off. It is not used in this computer.

Function Code E6H: Read Cache State — This function returns the operational status of the cache memory. It is not used in this computer.

NOTE: When using the extended INT 15 functions, it is a good idea to set the carry flag bit before making the function call. If you attempt a call on a computer that does not support it, the carry bit will remain set. Making an extended call on a computer that supports the function, such as the 80386-based systems, will result in a cleared carry bit.

Set/Read Time-of-Day (INT 1AH)

The INT 1AH instructions allows the programmer to set and/or read the CMOS clock time and date, and to set the CMOS alarm on or off. Register AH is used to establish the operation and registers CX and DX contain the values to be placed in the 32-bit counter or read from it. Register AL is used as the flag if the value has rolled over to a new day since the last time it was read.

There are eight functions available from within the 1AH interrupt, as shown in Table 7-10. The function value is placed in the AH register prior to calling the 1AH interrupt. A description of each of these functions follows the table.

Table 7-10. INT 1AH Functions

FUNCTION	DESCRIPTION
00H	Read time of day
01H	Set time of day
02H	Read real time clock
03H	Set real time clock
04H	Read RTC date
05H	Set RTC date
06H	Set RTC alarm
07H	Disable RTC alarm

Function 00H: Read Time of Day — This function allows the programmer to access the current time of day from the time and date structure. The AL register will contain one of two values. If the AL register contains a 0, then the timer count has not passed midnight since the last system reset. Otherwise, the value in the AL register will indicate the number of times the timer has passed midnight since the last system reset. The CX register contains the most-significant word in this count and the DX register contains the least-significant.

Function 01H: Set Time of Day — This function compliments function 00H. It allows the programmer to set the time of day. The CX and DX registers contain the same values as in function 00H.

Function 02H: Read Real Time Clock — This function allows the programmer to read directly from the timer. All returned values are in BCD format. Three registers are used to return the current time:

CH:Hours
CL:Minutes
DH:Seconds

If the real time clock is not operating properly, the CY bit will be set on return from this function.

Function 03H: Set Real Time Clock — This function complements function 02H, allowing the programmer to set the time directly in the real time clock. Registers CH, CL, and DH contain the same information, in the same format, as in function 02H. Register DL contains daylight saving time information. A value of 1 means that the DST option is in use; 0 means that it is not.

Function 04H: Read Real Time Clock Date — This function allows the programmer to obtain the date directly from the real time clock. All returned values are in BCD format. This function uses four registers:

CH:Century
CL:Year
DH:Month
DL:Day

If the real time clock is not operating properly, the CY bit will be set on return from this function.

Function 05H: Set Real Time Clock Date — This function complements the operation of function 04H. Using this function the programmer can set the date directly in the real time clock. The same registers are used here as in function 04H with the same values and formats.

Function 06H: Set Real Time Clock Alarm — This function allows the programmer to set the alarm feature of the real time clock. All values returned by this function are in BCD format. Three registers are used by this function:

CH:Hours
CL:Minutes
DH:Seconds

If a value in the range of C0H - FFH is present in any of the registers, it indicates a "don't care" condition for that time value. This allows the programmer the option of setting a periodic alarm condition. Whenever the alarm conditions are satisfied, the system generates interrupt 4AH.

If the real time clock is not operating properly, the CY bit will be set on return from this function.

Function 07H: Disable Real Time Clock Alarm — This function, when called, will turn off the alarm feature of the real time clock.

User Alarm (INT 4AH)

The user alarm interrupt is used within the computer whenever a real time clock alarm is active. This interrupt is generated by the system in response to the alarm conditions specified for the real time clock. (Refer to interrupt 1AH, function 06H).

If an application program uses the real time clock alarm feature, it must provide a handler for this interrupt. If no handler is provided in the application software, the system will crash when it attempts to execute this interrupt.

To set the time-of-day value, place the high count in the CX register and the low count in the DX register. Set register AH to 1 and execute the interrupt.

The value in the 32-bit time-of-day word is updated 18.2159 times a second. Therefore, 1 a.m. would be represented by a count of 65,577 and 12 noon would be 786.926. The values returned by INT 1AH would be as follows: for 1 a.m., register CX would contain 1 and DX would contain 41 (29H); for 12 noon, register CX would contain 12 (0CH) and DX would contain 494 (1EEH). The contents of register AL would depend upon whether the count had rolled over since the last time-of-day value was read. If AL is zero, then the time-of-day has not advanced past 24 hours. If AL is other than zero, it has.

Tick Timer (INT 1CH)

The INT 1CH interrupt provides a means for you to provide timing loops in your programs. The speed of the CPU in PC-compatible computers is not always consistent from one machine to another. Furthermore,some models offer switchable-speed CPU clock frequencies. This interrupt provides a convenient means of controlling timing loops regardless of the speed of the CPU.

This interrupt is called automatically every time INT 08H (the time-of-day interrupt) is executed (18.2159 times a second). Therefore, if you use this interrupt in your program, you must save the registers at the beginning of the routine and then later restore them before returning by way of an IRET instruction.

At powerup, this interrupt points to an IRET instruction. If you find any other routine, it will have been initialized by an application program.

Real-Time Clock Alarm Interrupt (INT 70H)

The INT 70H interrupt is initiated by a hardware interrupt request (IRQ8) or when an alarm from the real-time clock takes place. The alarm is set and handled through INT 15H, function 83H (event wait) and function 86H (wait). The hardware interrupt request takes place approximately 1,024 times a second. Since this interrupt is essentially controlled by hardware, it should not be called by user programs.

Programming Sound

This computer does not contain a dedicated sound chip. However, tones may be generated and played through the speaker via the programmable interval timer in three different ways, which may be implemented simultaneously:

- Generate a pulse train by toggling a program control register bit.
- Program the output of channel 2 of the timer/counter to deliver a sound waveform to the speaker.
- Modulate the clock input to the timer/counter through a program-controlled input/output register bit.

Programming sound, particularly music, is a specialized application in itself and, therefore, this capability is usually reserved for specific application programs and high-level languages, such as BASIC.

Chapter 8

Keyboard Interrupts

This chapter describes the interrupts used for keyboard communications and lists the codes generated by the keyboard.

Programming Keyboard Interrupts

The interrupts used exclusively by the keyboard are similar to those used in previous designs. However, the 101-key keyboard support provides some additional function calls that user programs can exploit. Programs designed to run on PC or PC-AT computers will function as normal. Table 8-1 indicates the keyboard interrupts for this computer.

Table 8-1. Keyboard Interrupts

INTERRUPT	FUNCTION
09H	Key pressed
16H	Keyboard input/output
1BH	Keyboard break

Key Pressed (INT 09H)

INT 09H is the key pressed interrupt which is executed each time you press or release a key. The interrupt routine reads the key from the keyboard register and encodes it or notes its action if it is a shift or control key. The following describes the action resulting from pressing or releasing the keys.

Keyboard Interrupts

When you press or release an ASCII character key, the code is read from I/O port 60H. If the code is a make code, the character and scan code is placed in the 32-byte keyboard buffer. The base address of the buffer is located at 40:1E. Address 40:1C points to the address in the buffer where the character and scan code is to be placed. Since the character and scan code require two bytes (one for each type of code), the keyboard buffer can hold up to 16 keys. The pointer at 40:1C increments by 2 until it goes past the end of the buffer. At that point, the service routine resets the pointer to the beginning of the table. Address 40:1A points at the current character to be read. This pointer increments in the same manner as the pointer at 40:1C. When the two pointers contain the same address, the buffer is empty.

CTRL, ALT, and SHIFT key make and break codes cause the interrupt service routine to update bytes at 40:17, 40:18, and 40:96. These bytes contain status codes, described later.

CTRL-ALT-DEL and CTRL-ALT-INS combinations cause the interrupt service routine to place 1234H at 40:72. The Monitor program executes the computer initialization routines. Before executing the initial memory tests, the routines check memory location 40:72 and if 1234H is found, the initialization routines bypass the memory test. After the routines complete the tests, they will start autoboot if CTRL-ALT-DEL had been pressed, or enter the Monitor program if CTRL-ALT-INS had been pressed. The Monitor program displays the version number and memory size of MS-DOS-usable base (up to 640K) and protected memory.

The PAUSE key puts the service routine in a loop until another ASCII key is pressed.

The PRINT SCREEN key calls INT 05H.

The CTRL-BREAK combination calls INT 1BH.

Whenever a make or break code is read from I/O port 60H, the service routine calls INT 15H, function 4FH. This allows programs to intercept and modify the keyboard code.

Since INT 09H directly affects the results of pressing or releasing any key, you should not modify the service routine unless you are an experienced system programmer. Build service routines for INT 15H, fucntion 4FH instead.

Keyboard Input/Output (INT 16H)

INT 16H is the keyboard input/output interrupt. This interrupt performs a keyboard operation specified by one of the function calls described in this section. The function code is placed in the AH register and then the interrupt is executed. The function codes implement:

- Key code retrieval (receive a character from the keyboard)
- Check keyboard buffer (to see if any codes are in it)
- Report keyboard status (SHIFT and CTRL keys)
- Set repetition rate (for applicable keys)
- Place character in keyboard buffer (as if it were entered from the keyboard).

NOTE: Seventeen keys provide unique keycodes. These keys duplicate some of the functions of other standard keys. Some software exploits the unique codes to provide enhanced functions for 101-key keyboard compatibility. To allow compatibility with programs written for standard codes, function codes 00H - 02H can be used to map the unique codes (of the duplicate keys) into standard codes. Refer to the "Keyboard Codes" section of this chapter for specific code information.

Table 8-2 lists the INT 16H functions. A detailed discussion of each function follows the table.

Table 8-2. INT 16H Functions

FUNCTION	DESCRIPTION
00H	Get and remove the character code from the keyboard buffer
01H	Read the character code from the keyboard buffer
02H	Get keyboard status
03H	Set key repetition rate
04H	Key click
05H	Place character in buffer
06H - 0FH	Reserved
10H	Get and remove extended character code from the keyboard buffer
11H	Read the extended character code from the keyboard buffer
12H	Get extended keyboard status
13H - FFH	Reserved

Function Code 00H: Get Character Code — This function code causes the interrupt to read a character from the keyboard buffer. When you press a key, the corresponding character code is placed in the keyboard buffer. This function will retrieve that code (ASCII codes 00H - 7FH or non-ASCII codes 80H - FFH) and place it in the AL register. The code is then removed from the keyboard buffer. The AH register will contain the scan code (01H - 84H) for the pressed key. If the keyboard buffer is empty, the routine waits until the next key press, generating another character code. Note that the SHIFT and CTRL keys do not generate codes, but affect the codes generated by other keys.

Function Code 01H: Check Keyboard Buffer — This function code causes the interrupt to check the status of the keyboard buffer. The zero flag (ZF) will be set if the buffer is empty. If the buffer contains key codes awaiting processing, the zero flag will be cleared (not set). The code (ASCII codes 00H - 7FH or non-ASCII codes 80H - FFH) assigned to the first character in the buffer is placed in the AL register. The scan code (01H - 84H) for that same key will appear in the AH register.

NOTE: This operation does not remove any codes from the keyboard buffer. Therefore, if you execute function code 01H followed by function code 00H, the same key codes will be returned. Only function code 00H removes the key codes from the buffer.

Function Code 02H: Get Keyboard Status — Two bytes of memory, 0040:0017 and 0040:0018, hold the keyboard status. This interrupt function code reports the status of the keyboard stored at location 0040:0017. Table 8-3 describes the value of this byte as stored in the AL register. If the report states that a key is pressed, that key was being held down at the time this function was executed.

Table 8-3. Keyboard Status Report

BIT	DESCRIPTION
0	If set (1), the right SHIFT key is pressed.
1	If set (1), the left SHIFT key is pressed.
2	If set (1), the CTRL key is pressed.
3	If set (1), the ALT key is pressed.
4	If set (1), the SCROLL LOCK mode is active.
5	If set (1), the NUM LOCK mode is active.
6	If set (1), the CAPS LOCK mode is active.
7	If set (1), the INSERT mode is active.

The keyboard status stored at location 0040:0018 can be read directly and interpreted by your own routines. Table 8-4 describes the possible values of this byte.

Table 8-4. Keyboard Status (0040:0018)

BIT	DESCRIPTION
0	Not used.
1	Not used.
2	Not used.
3	If set (1), the CTRL-NUM LOCK (pause) mode is active.
4	If set (1), the SCROLL LOCK key is pressed.
5	If set (1), the NUM LOCK key is pressed.
6	If set (1), the CAPS LOCK key is pressed.
7	If set (1), the INSERT key is pressed.

Function Code 03H: Set Key Repetition Rate — This function call will be executed when its code (03H) is placed in register AH and interrupt 16H is executed. The keyboard will respond to this interrupt and wait for a rate/delay determinant byte to be sent from the system. The rate is determined by bits 0-4 and the delay is determined by bits 5 and 6. Bit 7 is always 0. Table 8-5 indicates the repetition rates.

Table 8-5. Repetition Rates

BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	REPETITION RATE
0	0	0	0	0	30.0
0	0	0	0	1	26.7
0	0	0	1	0	24.0
0	0	0	1	1	21.8
0	0	1	0	0	20.0
0	0	1	0	1	18.5
0	0	1	1	0	17.1
0	0	1	1	1	16.0
0	1	0	0	0	15.0
0	1	0	0	1	13.3

Table 8-5 (continued). Repetition Rates

BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	REPETITION RATE
0	1	0	1	0	12.0
0	1	0	1	1	10.9
0	1	1	0	0	10.0
0	1	1	0	1	9.2
0	1	1	1	0	8.0
0	1	1	1	1	8.0
1	0	0	0	0	7.5
1	0	0	0	1	6.7
1	0	0	1	0	6.0
1	0	0	1	1	5.5
1	0	1	0	0	5.0
1	0	1	0	1	4.6
1	0	1	1	0	4.3
1	0	1	1	1	4.0
1	1	0	0	0	3.7
1	1	0	0	1	3.3
1	1	0	1	0	3.0
1	1	0	1	1	2.7
1	1	1	0	0	2.5
1	1	1	0	1	2.3
1	1	1	1	0	2.1
1	1	1	1	1	2.0

NOTE: The key repetition rate is calibrated in characters per second.

Function Code 05H: Place Character in Buffer — This function call is executed by placing a value of 05H in register AH and then executing interrupt 16H. This call's purpose is to allow placement of a character and associated scan code in the keyboard buffer as if it were entered from the keyboard. The character code is determined in register CL and the scan code is determined in register CH. When the INT 16H instruction occurs, these values will be placed in the keyboard buffer.

NOTE: The keyboard contains 17 keys that duplicate some of the functions of other keys. These keys generate unique extended codes. You can use function codes 10H - 12H to exploit these key codes in programs written specifically for them. However, if you do, you will loose compatibility with machines that use standard codes.

Function Code 10H: Get Extended Character Code — This function code performs the same operation as function code 00H. However, there are 17 keys that have unique codes in the 101-key keyboard. Many of these keys duplicate the function of other keys (for example, there are two sets of cursor control keys but each set generates unique code). This function code causes the interrupt to read the new key's code from the keyboard buffer.

Function Code 11H: Check Extended Keyboard Buffer — This function code places the value of any of the 17 new keys into register AL. The code (ASCII codes 00H - 7FH or non-ASCII codes 80H - FFH) assigned to the first character in the keyboard buffer is placed in the AL register. The scan code (01H - 84H) for that same key will appear in the AH register.

Function Code 12H: Get Extended Keyboard Status — This function code reports the status of the keyboard the same way that function code 02H does. However, function code 12H places a 2-byte status report in registers AL and AH, respectively. Table 8-6 describes the values of the byte in register AL and Table 8-7 describes the value of the byte in register AH.

Table 8-6. Keyboard Status Report (Register AL)

BIT	DESCRIPTION
0	If set (1), the right SHIFT key is pressed.
1	If set (1), the left SHIFT key is pressed.
2	If set (1), the CTRL key is active.
3	If set (1), the ALT key is active.
4	If set (1), the SCROLL LOCK mode is active.
5	If set (1), the NUM LOCK mode is active.
6	If set (1), the CAPS LOCK mode is active.
7	If set (1), the INSERT mode is active.

Table 8-7. Keyboard Status Report (Register AH)

BIT	DESCRIPTION
0	If set (1), the left CTRL key is pressed.
1	If set (1), the left ALT key is pressed.
2	If set (1), the right CTRL key is pressed.
3	If set (1), the right ALT key is pressed.
4	If set (1), the SCROLL LOCK key is pressed.
5	If set (1), the NUM LOCK key is pressed.
6	If set (1), the CAPS LOCK key is pressed.
7	System flag.

Keyboard Break (INT 1BH)

The INT 1BH instruction executes when the key pressed interrupt (09H) detects that the CTRL and BREAK keys (CTRL-BREAK) are pressed at the same time (you must press and hold the CTRL key before pressing the BREAK key).

The interrupt for this routine must exit with an IRET instruction to properly terminate the 09H interrupt. When you turn the computer on, this routine provides only the IRET instruction. That way, if a key press occurs during the self-tests, nothing will happen (unless the ESC key is pressed, which will terminate the self-tests and drive program execution to the Monitor).

If you allow this interrupt to retain control, you may have to accommodate one or more of the following conditions.

- The break may occur during interrupt processing. You must then send one or more end-of-interrupt commands to the interrupt controller.

- If an operation was in process when you caused the interrupt to take place, all input/output devices must be reset.

- Programs that use this interrupt must not chain to the previous owner of this interrupt and you must restore the interrupt when you finish processing.

Remember, when servicing an INT 1BH instruction, your routine must perform an IRET to make sure that the interrupt restores properly. Do not link your service routine to the next one.

GW-BASIC is an example of software that uses this routine. It uses this interrupt to halt execution of a BASIC program whenever you press the CTRL-BREAK key sequence.

Keyboard Codes

The keyboard supports three different keycode sets. Set 2 is the default set. Sets 1 and 3 are optional and can be selected by sending F0H to the keyboard. The keyboard will clear its buffer and expect the system to send a byte of data representing the desired set. Byte value 01H represents code set 1 and byte value 03H represents code set 3.

In this computer, the set 2 keycodes are translated into codes used in keycode set 1 by the 8742 system control processor. The system control processor does not currently translate keycode set 3. If keycode set 3 is used, a program designed to handle this translation may have to be written. The make and break key codes for the three different code sets are provided in Tables 8-8 through 8-12.

Table 8-8. Alphabetic Keycode Sets

	SCAN SET 1		SCAN SET 2		SCAN SET 3	
KEY	MAKE	BREAK	MAKE	BREAK	MAKE	BREAK
A	1E	9E	1C	F0 1C	1C	F0 1C
B	30	B0	32	F0 32	32	F0 32
C	2E	AE	21	F0 21	21	F0 21
D	20	A0	23	F0 23	23	F0 23
E	12	92	24	F0 24	24	F0 24
F	21	A1	2B	F0 2B	2B	F0 2B
G	22	A2	34	F0 34	34	F0 34
H	23	A3	33	F0 33	33	F0 33
I	17	97	43	F0 43	43	F0 43
	24	A4	3B	F0 3B	3B	F0 3B

Table 8-8 (continued). Alphabetic Keycode Sets

KEY	SCAN SET 1 MAKE	SCAN SET 1 BREAK	SCAN SET 2 MAKE	SCAN SET 2 BREAK	SCAN SET 3 MAKE	SCAN SET 3 BREAK
K	25	A5	42	F0 42	42	F0 42
L	26	A6	4B	F0 4B	4B	F0 4B
M	32	B2	3A	F0 3A	3A	F0 3A
N	31	B1	31	F0 31	31	F0 31
O	18	98	44	F0 44	44	F0 44
P	19	99	4D	F0 4D	4D	F0 4D
Q	10	90	15	F0 15	15	F0 15
R	13	93	2D	F0 2D	2D	F0 2D
S	1F	9F	1B	F0 1B	1B	F0 1B
T	14	94	2C	F0 2C	2C	F0 2C
U	16	96	3C	F0 3C	3C	F0 3C
V	2F	AF	2A	F0 2A	2A	F0 2A
W	11	91	1D	F0 1D	1D	F0 1D
X	2D	AD	22	F0 22	22	F0 22
Y	15	95	35	F0 35	35	F0 35
Z	2C	AC	1A	F0 1A	1A	F0 1A

NOTE: All values are expressed in hexadecimal.

Table 8-9. Numeric and Punctuation Keycode Sets

KEY	SCAN SET 1 MAKE	SCAN SET 1 BREAK	SCAN SET 2 MAKE	SCAN SET 2 BREAK	SCAN SET 3 MAKE	SCAN SET 3 BREAK
1/!	02	82	16	F0 16	16	F0 16
2/@	03	83	1E	F0 1E	1E	F0 1E
3/#	04	84	26	F0 26	26	F0 26
4/$	05	85	25	F0 25	25	F0 25
5/%	06	86	2E	F0 2E	2E	F0 2E
6/^	07	87	36	F0 36	36	F0 36
7/&	08	88	3D	F0 3D	3D	F0 3D
8/*	09	89	3E	F0 3E	3E	F0 3E
9/(	0A	8A	46	F0 46	46	F0 46
0/)	0B	8B	45	F0 45	45	F0 45
-/_	0C	8C	4E	F0 4E	4E	F0 4E
=/+	0D	8D	55	F0 55	55	F0 55
'/~	29	A9	0E	F0 0E	0E	F0 0E

Table 8-9 (continued). Numeric and Punctuation Keycode Sets

KEY	SCAN SET 1 MAKE	SCAN SET 1 BREAK	SCAN SET 2 MAKE	SCAN SET 2 BREAK	SCAN SET 3 MAKE	SCAN SET 3 BREAK
[/{	1A	9A	54	F0 54	54	F0 54
]/}	1B	9B	5B	F0 5B	5B	F0 5B
;/:	27	A7	4C	F0 4C	4C	F0 4C
'/"	28	A8	52	F0 52	52	F0 52
,/<	33	B3	41	F0 41	41	F0 41
./>	34	B4	49	F0 49	49	F0 49
//?	35	B5	4A	F0 4A	4A	F0 4A
\/|	2B	AB	5D	F0 5D	5C	F0 5C

NOTE: All values are expressed in hexadecimal.

Table 8-10. Function and Control Key Keycode Sets

KEY	SCAN SET 1 MAKE	SCAN SET 1 BREAK	SCAN SET 2 MAKE	SCAN SET 2 BREAK	SCAN SET 3 MAKE	SCAN SET 3 BREAK
F1	3B	BB	05	F0 05	07	F0 07
F2	3C	BC	06	F0 06	0F	F0 0F
F3	3D	BD	04	F0 04	17	F0 17
F4	3E	BE	0C	F0 0C	1F	F0 1F
F5	3F	BF	03	F0 03	27	F0 27
F6	40	C0	0B	F0 0B	2F	F0 2F
F7	41	C1	83	F0 83	37	F0 37
F8	42	C2	0A	F0 0A	3F	F0 3F
F9	43	C3	01	FO 01	47	F0 47
F10	44	C4	09	F0 09	4F	F0 4F
F11 (FN-F1)	57	D7	78	F0 78	56	F0 56
F12 (FN-F2)	58	D8	07	F0 07	5E	F0 5E
Slow (FN-F3)	5A	DA	17	F0 17	—	—
Fast (FN-F4)	5B	DB	1F	F0 1F	—	—
Palette up[1]	5C	DC	27	F0 27	—	—
Palette dn[2]	5D	DD	2F	F0 2F	—	—
CRT/LCD[3]	5E	DE	37	F0 37	—	—
BACKSPACE	0E	8E	66	F0 66	66	F0 66
TAB	0F	8F	0D	F0 0D	0D	F0 0D

Table 8-10 (continued). Function and Control Key Keycode Sets

KEY	SCAN SET 1 MAKE	SCAN SET 1 BREAK	SCAN SET 2 MAKE	SCAN SET 2 BREAK	SCAN SET 3 MAKE	SCAN SET 3 BREAK
CAPS LOCK	3A	BA	58	F0 58	14	F0 14
SCROLL LOCK[4]	46	C6	7E	F0 7E	5F	F0 5F
ENTER[5]	1C	9C	5A	F0 5A	5A	F0 5A
Left SHIFT	2A	AA	12	F0 12	12	F0 12
Right SHIFT	36	B6	59	F0 59	59	F0 59
Space bar	39	B9	29	F0 29	29	F0 29
ESC	01	81	76	F0 76	08	F0 08
Left CTRL	1D	9D	14	F0 14	11	F0 11
Right CTRL	E0 1D	E0 9D	E0 14	E0 F0	14 58	F0 58
Left ALT	38	B8	11	F0 11	19	F0 19
Right ALT	E0 38	E0 B8	E0 11	E0 F0	11 39	F0 39

NOTES

1. The FN-F8 key combination produces the palette up code sets.
2. The FN-F9 key combination produces the palette dn code sets.
3. The FN-F10 key combination produces the CRT/LCD code sets.
4. The FN-BACKSPACE key combination produces the SCROLL LOCK code sets.
5. These codes are also produced when the FN-ENTER key combination is pressed.
6. All values are expressed in hexadecimal.

Table 8-11. 101-Key Keyboard-Compatible Screen Control Keycode Sets

KEY	MAKE BREAK	SHIFT ACTIVE MAKE BREAK	NUM LOCK ACTIVE MAKE BREAK
Scan Code Set 1			
INSERT	E0 52 E0 D2	E0 AA E0 52 E0 D2 E0 2A	E0 2A E0 52 E0 D2 E0 AA
DELETE	E0 53 E0 D3	E0 AA E0 53 E0 D3 E0 2A	E0 2A E0 53 E0 D3 E0 AA
←	E0 4B E0 CB	E0 AA E0 4B E0 CB E0 2A	E0 2A E0 4B E0 CB E0 AA

Table 8-11 (continued). 101-Key Keyboard-Compatible Screen Control Keycode Sets

KEY	MAKE BREAK	SHIFT ACTIVE MAKE BREAK	NUM LOCK ACTIVE MAKE BREAK
HOME	E0 47 E0 C7	E0 AA E0 47 E0 C7 E0 2A	E0 2A E0 47 E0 C7 E0 AA
END	E0 4F E0 CF	E0 AA E0 4F E0 CF E0 2A	E0 2A E0 4F E0 CF E0 AA
↑	E0 48 E0 C8	E0 AA E0 48 E0 C8 E0 2A	E0 2A E0 48 E0 C8 E0 AA
↓	E0 50 E0 D0	E0 AA E0 50 E0 D0 E0 2A	E0 2A E0 50 E0 D0 E0 AA
PAGE UP	E0 49 E0 C9	E0 AA E0 49 E0 C9 E0 2A	E0 2A E0 49 E0 C9 E0 AA
PAGE DOWN	E0 51 E0 D1	E0 AA E0 51 E0 D1 E0 2A	E0 2A E0 51 E0 D1 E0 AA
→	E0 4D E0 CD	E0 AA E0 4D E0 CD E0 2A	E0 2A E0 4D E0 CD E0 AA
PRINT SCREEN	E0 2A E0 37 E0 B7 E0 AA	E0 37 E0 B7[note 2]	
PAUSE/BREAK	[note 3]		

Scan Code Set 2

INSERT	E0 70 E0 F0 70	E0 F0 12 E0 70 E0 F0 70 E0 12	E0 12 E0 72 E0 F0 70 E0 F0 12
DELETE	E0 71 E0 F0 71	E0 F0 12 E0 71 E0 F0 71 E0 12	E0 12 E0 71 E0 F0 71 E0 F0 12
←	E0 6B E0 F0 6B	E0 F0 12 E0 6B E0 F0 6B E0 12	E0 12 E0 6B E0 F0 6B E0 F0 12
HOME	E0 6C E0 F0 6C	E0 F0 12 E0 6C E0 F0 6C E0 12	E0 12 E0 6C E0 F0 6C E0 F0 12
END	E0 69 E0 F0 69	E0 F0 12 E0 69 E0 F0 69 E0 12	E0 12 E0 69 E0 F0 69 E0 F0 12
↑	E0 75 E0 F0 75	E0 F0 12 E0 75 E0 F0 75 E0 12	E0 12 E0 75 E0 F0 75 E0 F0 12
↓	E0 72 E0 F0 72	E0 F0 12 E0 72 E0 F0 72 E0 12	E0 12 E0 72 E0 F0 72 E0 F0 12

Table 8-11 (continued). 101-Key Keyboard-Compatible Screen Control Keycode Sets

KEY	MAKE BREAK	SHIFT ACTIVE MAKE BREAK	NUM LOCK ACTIVE MAKE BREAK
PAGE UP	E0 7D E0 F0 7D	E0 F0 12 E0 7D E0 F0 7D E0 12	E0 12 E0 7D E0 F0 7D E0 F0 12
PAGE DOWN	E0 7A E0 F0 7A	E0 F0 12 E0 7A E0 F0 7A E0 12	E0 12 E0 7A E0 F0 7A E0 F0 12
→	E0 74 E0 F0 74	E0 F0 12 E0 74 E0 F0 74 E0 12	E0 12 E0 74 E0 F0 74 E0 F0 12
PRINT SCREEN	note 4		
PAUSE/BREAK	note 5		
Scan Code Set 3			
INSERT	67 F0 67	note 6	note 6
DELETE	64 F0 64		
←	61 F0 61		
HOME	6E F0 6E		
END	65 F0 65		
↑	63 F0 63		
↓	60 F0 60		
PAGE UP	6F F0 6F		
PAGE DOWN	6D F0 6D		
→	6A F0 6A		

Table 8-11 (continued). 101-Key Keyboard-Compatible Screen Control Keycode Sets

KEY	MAKE BREAK	SHIFT ACTIVE MAKE BREAK	NUM LOCK ACTIVE MAKE BREAK
PRINT SCREEN	57 F0 57		
PAUSE/BREAK	62 F0 62		

NOTES:

1. All values are expressed in hexadecimal.
2. In scan code set 1 the PRTSC key generates E0 37 (make) and E0 B7 (break) when a SHIFT key or the CTRL key is active (pressed down). It also generates codes 54 (make) and D4 (break) when the ALT key is active.
3. This key generates E1 1D 45 E1 9D C5 when pressed. It does not generate any code when released. When the CTRL key is active (pressed down), E0 46 E0 C6 is generated.
4. In scan code set 2 the PRTSC key generates E0 12 E0 7C (make) and E0 F0 7C E0 F0 12 (break). When a SHIFT key or the CTRL key is active the key generates E0 7C (make) and E0 F0 7C (break). When the ALT key is active the key generates 84 (make) and F0 84 (break).
5. This key generates E1 14 77 E1 F0 14 F0 77 when pressed. It does not generate any code when released. When the CTRL key is active (pressed down), E0 7E E0 F0 7E is generated.
6. The same codes are generated when a SHIFT or the NUM LOCK key is active.

Table 8-12. Numeric Key Keycode Sets

KEY	CODE SET 1 MAKE	 BREAK	CODE SET 2 MAKE	 BREAK	CODE SET 3 MAKE	 BREAK
NUM LOCK	45	C5	77	F0 77	76	F0 76
1 / END	4F	CF	69	F0 69	69	F0 69
2 / ↓	50	D0	72	F0 72	72	F0 72
3 / PGDN	51	D1	7A	F0 7A	7A	F0 7A
4 / ←	4B	CB	6B	F0 6B	6B	F0 6B
5	4C	CC	73	F0 73	73	F0 73

Table 8-12 (continued). Numeric Key Keycode Set

KEY	CODE SET 1 MAKE	CODE SET 1 BREAK	CODE SET 2 MAKE	CODE SET 2 BREAK	CODE SET 3 MAKE	CODE SET 3 BREAK
6 / →	4D	CD	74	F0 74	74	F0 74
7 / HOME	47	C7	6C	F0 6C	6C	F0 6C
8 / ↑	48	C8	75	F0 75	75	F0 75
9 / PGUP	49	C9	7D	F0 7D	7D	F0 7D
0 / INS	52	D2	70	F0 70	70	F0 70
. / DEL	53	D3	71	F0 71	71	F0 71
+	4E	CE	79	F0 79	7C	F0 7C
-	4A	CA	7B	F0 7B	84	F0 84
*	37	B7	7C	F0 7C	7E	F0 7E
ENTER	E0 1C	E0 9C	E0 5A	E0 F0 5A	79	F0 79

NOTES:

1. In code set 1, the / (divide) key generates E0 35 (make) and E0 B5 (break). When a SHIFT key is also pressed the divide key generates E0 AA E0 35 (make) and E0 B5 E0 2A (break). In scan code set 2, the divide key generates E0 4A (make) and E0 F0 4A (break). When a SHIFT key is also pressed, the divide key generates E0 F0 12 4A (make) and E0 12 F0 4A (break). In scan code set 3, the divide key generates 77 (make) and FO 77 (break).
2. All values are expressed in hexadecimal.
3. The same codes are generated for each key. The state of the NUM LOCK key differentiates the function.

System Scan Codes

Table 8-13 through Table 8-17 provide the hexadecimal codes generated by the computer when a key is pressed. These scan codes consist of the key's make/break code plus an ASCII code. The scan codes that are generated when the SHIFT, CTRL, and ALT keys are active are also provided. The make/break codes for each code set (1, 2, and 3) are provided earlier in this chapter in Table 8-7 through Table 8-12. Figure 8-1 illustrates the keyboard layout.

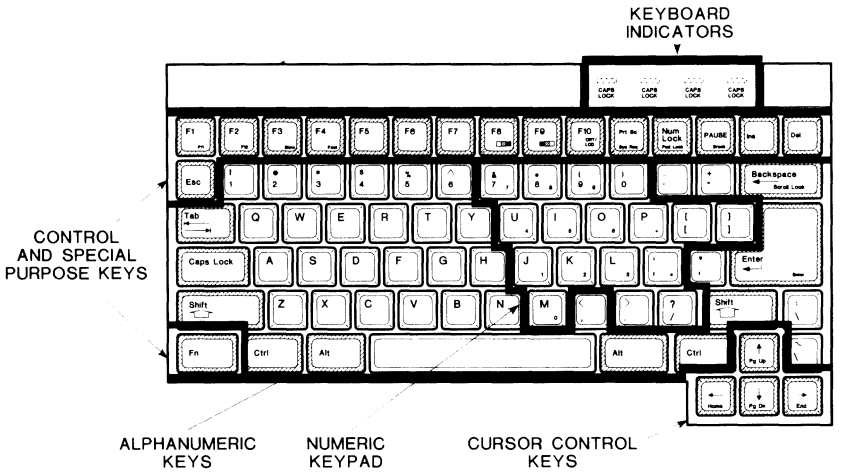


Figure 8-1. Keyboard

When you press a key, the keyboard processor generates a code that represents the pressed key. The BIOS (initialized by the Monitor program, operating system, or application program) interprets the code and generates a scan code through one or more of interrupt routines called by the keyboard interrupts. The SHIFT CTRL, and ALT keys can alter this code and affect the final scan code generated.

When you release a key, the keyboard processor sends a key "break" code. Normally, break codes cause no further action to take place. However, application programs can modify the interrupts and interpret these break codes for their own purposes.

Alphabetic Keys

The 26 alphabetic keys use the standard American typewriter layout. Figure 8-2 illustrates their position on the keyboard. The CAPS LOCK key toggles the alphabetic keys between lower and upper case. The SHIFT keys reverse the action of the CAPS LOCK key on the alphabetic keys.

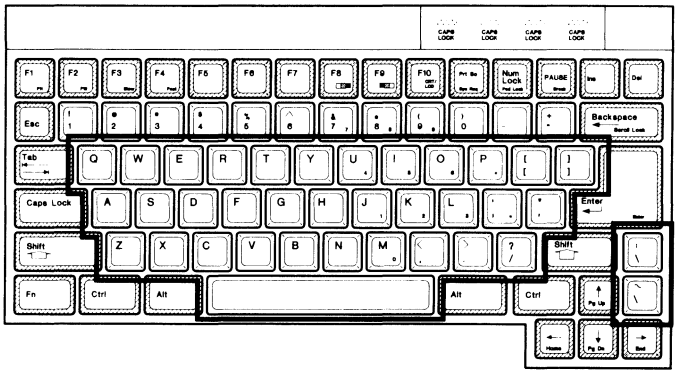


Figure 8-2. Alphabetic Keys

Table 8-13 provides the make, ASCII, and scan codes generated for the alphabetic keys. The table also provides the scan codes produced when the SHIFT, CTRL, and ALT keys are also pressed.

Table 8-13. Alphabetic Key Scan Codes (AT Mode)

KEY	MAKE CODE	ASCII CODE	SCAN CODE	KEY PRESSED		
				SHIFT	CTRL	ALT
A	1E	61	1E 61	1E 41	1E 01	1E 00
B	30	62	30 62	30 42	30 02	30 00
C	2E	63	2E 63	3E 43	2E 03	2E 00
D	20	64	20 64	20 44	20 04	20 00
E	12	65	12 65	12 45	12 05	12 00
F	21	66	21 66	21 46	21 06	21 00
G	22	67	22 67	22 47	22 07	22 00
H	23	68	23 68	23 48	23 08	23 00
I	17	69	17 69	17 49	17 09	17 00
J	24	6A	24 6A	24 4A	24 0A	24 00
K	25	6B	25 6B	25 4B	25 0B	25 00
L	26	6C	26 6C	26 4C	26 0C	26 00
M	32	6D	32 6D	32 4D	32 0D	32 00
N	31	6E	31 6E	31 4E	31 0E	31 00
O	18	6F	18 6F	18 4F	18 0F	18 00

Table 8-13 (continued). Alphabetic Key Scan Codes (AT Mode)

KEY	MAKE CODE	ASCII CODE	SCAN CODE	KEY PRESSED SHIFT	CTRL	ALT
P	19	70	19 70	19 50	19 10	19 00
Q	10	71	10 71	10 51	10 11	10 00
R	13	72	13 72	13 52	13 12	13 00
S	1F	73	1F 73	1F 53	1F 13	1F 00
T	14	74	14 74	14 54	14 14	14 00
U	16	75	16 75	16 55	16 15	16 00
V	2F	76	2F 76	2F 56	2F 16	2F 00
W	11	77	11 77	11 57	11 17	11 00
X	2D	78	2D 78	2D 58	2D 18	2D 00
Y	15	79	15 79	15 59	15 19	15 00
Z	2C	7A	2C 7A	2C 5A	2C 1A	2C 00

NOTE: All values are expressed in hexadecimal.

Numeric/Punctuation Keys

The numeric/punctuation keys, shown in Figure 8-3 include the numbers 0 through 9, the common punctuation marks, and the special programming characters that make up the remainder of the printable ASCII character set. Each of these keys can generate two characters. The uppercase character is generated when you press either SHIFT key in combination with another key. Table 8-14 details the make codes, ASCII codes, and scan codes generated for the numeric/punctuation keys.

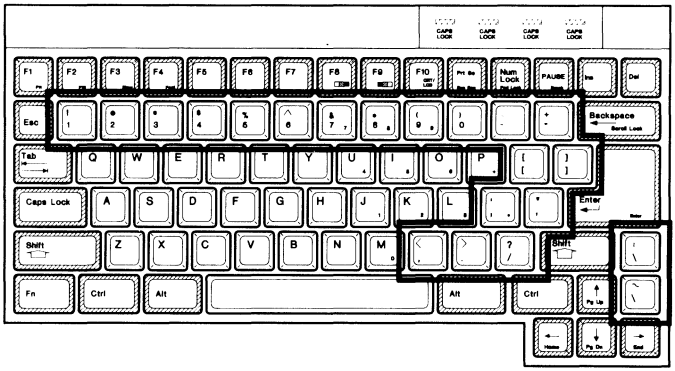


Figure 8-3. Numeric/Punctuation Keys

Table 8-14. Numeric/Punctuation Key Scan Codes

KEY	MAKE CODE	ASCII CODE	SCAN CODE	KEY PRESSED		
				SHIFT	CTRL	ALT
! / 1	02	31	02 31	02 21	—	78 00
@ / 2	03	32	03 32	03 40	03 00	79 00
# / 3	04	33	04 33	04 23	—	7A 00
$ / 4	05	34	05 34	05 24	—	7B 00
% / 5	06	35	06 35	06 25	—	7C 00
^ / 6	07	36	07 36	07 5E	07 1E	7D 00
& / 7	08	37	08 37	08 26	—	7E 00
* / 8	09	38	09 38	09 2A	—	7F 00
(/ 9	0A	39	0A 39	0A 28	—	80 00
) / 0	0B	30	0B 30	0B 29	—	81 00
_ / -	0C	2D	0C 2D	0C 5F	0C 1F	82 00
+ / =	0D	3D	0D 3D	0D 2B	83 00	—
~ / '	29	60	29 60	29 7E	—	—
{ / [	1A	5B	1A 5B	1A 7B	1A 1B	—
} /]	1B	5D	1B 5D	1B 7D	1B 1D	—
: / ;	27	3B	27 3B	27 3A	—	—
" / '	28	27	28 27	28 22	—	—
< / ,	33	2C	33 2C	33 3C	—	—
> / .	34	2E	34 2E	34 3E	—	—
? / /	35	2F	35 2F	35 3F	—	—
| / \	2B	5C	2B 5C	2B 7C	2B 1C	—

NOTE: All values are expressed in hexadecimal.

Function and Control Keys

The function and control keys, shown in Figure 8-4, do not generate their own codes but modify the codes produced by other keys. They are described in Table 8-15 and their scan codes are listed in Table 8-16.

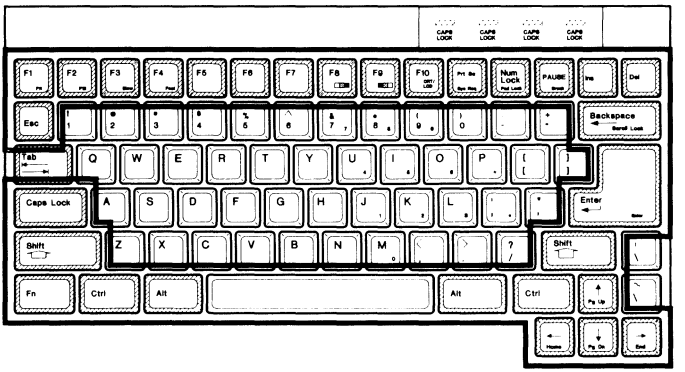


Figure 8-4. Function and Control Keys

Table 8-15. Function and Control Keys

KEY	DESCRIPTION
Left ALT	Alternate. In addition to modifying the codes produced by other keys, the ALT key can be used to generate any hexadecimal code from 0 to FFH (0 to 255 decimal). The values are displayed as an equivalent ASCII character. Lock the keypad (FN-PAD LOCK), press and hold either ALT key, and then enter the decimal value through the keypad. When you release the ALT key, the keyboard controller generates the hexadecimal value and the system displays it on the video monitor.
Right ALT	Alternate. This key generates unique make and break codes (and therefore unique scan codes). Programs can be written to interpret the unique codes and execute a specific function. To allow compatibility with existing programs, function calls 00H-02H remove the unique code characteristics from the right ALT key. It then functions like the left ALT key.

Table 8-15 (continued). Function and Control Keys

KEY	DESCRIPTION
BACK SPACE	Back space. Normally generates ASCII code 08H.
BREAK	Break. Normally used to break program execution. Press and hold the SHIFT key and then press the BREAK key.
CAPS LOCK	Caps Lock. Toggles the CAPS LOCK mode. The SHIFT keys reverse the action of the CAPS LOCK mode.
Left CTRL	Control. Modifies the codes produced by other keys by adding a special hexadecimal prefix code to the keys individual code. Applications recognize the additional code and respond as required.
Right CTRL	Control. This key generates unique make and break codes (and therefore unique scan codes). Programs can be written to interpret the unique codes and execute a specific function. To allow compatibility with existing programs, function calls 00H-02H remove the unique code characteristics from the right CTRL key. It then functions like the left CTRL key.
Cursor keys	Cursor/screen control. Software uses the codes produced by these keys to move the cursor up, down, left, right, and control screen movement. The secondary cursor control keys generate unique make and break codes. Software can be written to exploit these keys or a function call (00H-02H) can be used to provide the same functions as the primary cursor control keys (activated through the numeric keypad).
DEL	Delete. The code produced by the DEL key is used by most text editing and word processing software to delete material. It is also used with the CTRL and ALT keys to reset the computer. The secondary DEL key generates unique make and break codes. Software can be written to exploit these codes or a function call (00H-02H) can be used to provide the same functions as the primary DEL key (activated through the numeric keypad).

Table 8-15 (continued). Function and Control Keys

KEY	DESCRIPTION
END	End. The code produced by the END key is used by most text editing and word processing software for screen, page, or document manipulation. The secondary END key generates unique make and break codes. Software can be written to exploit these codes or a function call (00H-02H) can be used to provide the same functions as the primary END key (activated through the numeric keypad).
ESC	Escape. Normally generates ASCII code 1BH.
F1 - F12	Function keys. The codes produced by these keys work with different software products that generate user-defined functions. Note that F11 and F12 are generated through the FN-F1 and FN-F2 key combinations, respectively.
HOME	Home. The code produced by the HOME key is used by most text editing and word processing software for screen, page, or document manipulation. The secondary HOME key generates unique make and break codes. Software can be written to exploit these codes or a function call (00H-02H) can be used to provide the same functions as the primary HOME key (activated through the numeric keypad).
NUM LOCK	Numbers Lock. When the numeric keypad is active (FN-PAD LOCK) this key toggles the keypad between numeric and secondary cursor/screen control modes. The SHIFT keys reverse the numeric and secondary cursor/screen control modes.
PGDN	Page Down. The code produced by the PGDN key is used by most text editing and word processing software for screen, page, or document manipulation. The secondary PAGE DOWN key generates unique make and break codes. Software can be written to exploit these codes or a function call (00H-02H) can be used to provide the same functions as the primary PGDN key (activated through the numeric keypad).

Table 8-15 (continued). Function and Control Keys

KEY	DESCRIPTION
PGUP	Page Up. The code produced by the PGUP key is used by most text editing and word processing software for screen, page, or document manipulation. The secondary PAGE UP key generates unique make and break codes. Software can be written to exploit these codes or a function call (00H-02H) can be used to provide the same functions as the primary PGUP key (activated through the numeric keypad).
PRTSC	Print Screen. Sends the contents of the screen to the LPT1 device. If the operating system is configured to map the parallel port to the serial port (COM1), the screen contents is routed through COM1.
Slow	This key function is activated through the FN-F3 key combination. The slow mode sets the CPU clock speed to an AT-compatible rate.
Fast	This key function is activated through the FN-F4 key combination. The fast mode increases the CPU clock speed to run applications faster.
Palette up	This key function toggles the LCD through eight different gray scales with increasing intensity. It does not affect the display of a color video monitor if it is attached to the computer as the primary display.
Palette dn	This key function toggles the LCD through eight different gray scales with decreasing intensity. It does not affect the display of a color monitor if it is attached to the computer as the primary display.
CRT/LCD	This key function toggles the active video output between raster-scan and liquid-crystal formats. In this computer, the LCD can be set as primary display or a color video monitor can be attached to the computer and used as the primary display.
ENTER/RETURN	Enter/Return. Normally generates ASCII code 0DH. Software usually adds ASCII code 0AH.

Table 8-15 (continued). Function and Control Keys

KEY	DESCRIPTION
SCROLL LOCK	Scroll Lock. The code produced by this key is used by software to control screen scrolling.
SHIFT	Shift. Modifies the codes produced by other keys. Reverses the action of CAPS LOCK and NUM LOCK.
Space bar	Space. Normally generates ASCII code 20H.
TAB	Tab. Normally generates ASCII code 09H.

Table 8-16. Function and Control Key Scan Codes

				KEY PRESSED		
KEY	MAKE CODE	ASCII CODE	SCAN CODE	SHIFT	CTRL	ALT
F1	3B	00	3B 00	54 00	5E 00	68 00
F2	3C	00	3C 00	55 00	5F 00	69 00
F3	3D	00	3D 00	56 00	60 00	6A 00
F4	3E	00	3E 00	57 00	61 00	6B 00
F5	3F	00	3F 00	58 00	62 00	6C 00
F6	40	00	40 00	59 00	63 00	6D 00
F7	41	00	41 00	5A 00	64 00	6E 00
F8	42	00	42 00	5B 00	65 00	6F 00
F9	43	00	43 00	5C 00	66 00	70 00
F10	44	00	44 00	5D 00	67 00	71 00
F11 (FN-F1)	57	00	85 00	87 00	89 00	8B 00
F12 (FN-F2)	58	00	86 00	88 00	8A 00	8C 00
BACK SPACE	0E	08	0E 08	0E 08	0E 7F	0E 00
ESC	01	1B	01 1B	01 1B	01 1B	01 00
RETURN	1C	0D	1C 0D	1C 0D	1C 0A	1C 00
Space bar	39	20	39 20	39 20	39 20	39 20
7/HOME	47	2D	47 37	47 00	77 00	00 07
HOME	note 2	E0	47 E0	47 E0	77 E0	97 E0
9/PGUP	49	2D	49 39	49 00	84 00	99 00
PAGE UP	note 2	E0	49 E0	49 E0	84 E0	00 09
1/END	4F	00	4F 31	4F 00	75 00	9F 00
END	note 2	E0	4F E0	4F E0	75 E0	00 01

Table 8-16 (continued). Function and Control Key Scan Codes

KEY	MAKE CODE	ASCII CODE	SCAN CODE	KEY PRESSED SHIFT	CTRL	ALT
3/PGUP	51	00	51 33	51 00	76 00	—
PAGE DOWN	note 2	E0	51 E0	51 E0	76 E0	A1 E0
. / DEL	53	00	53 2E	53 00	93 00	—
DELETE	note 2	E0	53 E0	53 E0	93 E0	A3 00
TAB	0F	09	0F 09	—	94 00	A5 00
PRINT SCREEN	—	—	—	—	72 00	—

NOTES

1. The scan code for the majority of keys consist of that key's make code plus its ASCII code. Some keys generate different scan codes with respect to their make codes, however.

2. These codes are described in Table 8-16.

3. The following keys usually modify the values produced by other keys:
   ALT
   CAPS LOCK
   CTRL
   NUM LOCK
   PAUSE
   SHIFT
   SCROLL LOCK
   BREAK

Make/Break Key Codes

Each time you press a key, it produces two distinct codes: the make code when you press the key and the break code when you release the key. One or more bytes are produced by the keyboard processor and interpreted by INT 09H. These codes are not the same codes that result from INT 16H. To access the make/break codes, you have to modify INT 09H. Do not alter this interrupt unless you are an experienced programmer. Table 8-17 lists all the make/break codes for scan set 2.

Table 8-17. Make/Break Key Codes

KEY	MAKE	BREAK
A	1E	9E
B	30	B0
C	2E	AE
D	20	A0
E	12	92
F	21	A1
G	22	A2
H	23	A3
I	17	97
J	24	A4
K	25	A5
L	26	A6
M	32	B2
N	31	B1
O	18	98
P	19	99
Q	10	90
R	13	93
S	1F	9F
T	14	94
U	16	96
V	2F	AF
W	11	91
X	2D	AD
Y	15	95
Z	2C	AC
1 / !	02	82
2 / @	03	83
3 / #	04	84
4 / $	05	85
5 / %	06	86
6 / ^	07	87
7 / &	08	88
8 / *	09	89
9 / (	0A	8A
0 /)	0B	8B
- / _	0C	8C
= / +	0D	8D
‘ / ~	29	A9

Table 8-17 (continued). Make/Break Key Codes

KEY	MAKE	BREAK
[/ {	1A	9A
] / }	1B	9B
; / :	27	A7
' / "	28	A8
, / <	33	B3
. / >	34	B4
/ / ?	35	B5
\ / |	2B	AB
F1	3B	BB
F2	3C	BC
F3	3D	BD
F4	3E	BE
F5	3F	BF
F6	40	C0
F7	41	C1
F8	42	C2
F9	43	C3
F10	44	C4
F11	57	D7
F12	58	D8
BACK SPACE	0E	8E
TAB	0F	8F
CAPS LOCK	3A	BA
SCROLL LOCK	46	C6
ENTER/RETURN	1C	9C
Left SHIFT	2A	AA
Right SHIFT	36	B6
Space bar	39	B9
ESC	01	81
Left CTRL	1D	9D
Left ALT	38	B8
Right CTRL	E0 1D	E0 9D
Right ALT	E0 38	E0 B8
PAUSE	E1 1D 45	E1 9D C5
PRINT SCREEN	E0 2A E0 37	E0 B7 E0 AA

Table 8-17 (continued). Make/Break Key Codes

KEY	MAKE	BREAK
Separate Keypad		
INSERT	E0 2A E0 52	E0 D2 E0 AA
HOME	E0 2A E0 47	E0 C7 E0 AA
PAGE UP	E0 2A E0 49	E0 C9 E0 AA
DELETE	E0 2A E0 53	E0 D3 E0 AA
END	E0 2A E0 4F	E0 CF E0 AA
PAGE DOWN	E0 2A E0 51	E0 D1 E0 AA
↑	E0 2A E0 48	E0 C8 E0 AA
←	E0 2A E0 4B	E0 CB E0 AA
↓	E0 2A E0 50	E0 D0 E0 AA
→	E0 2A E0 4D	E0 CD E0 AA
Numeric Keypad		
1	4F	CF
2	50	DF
3	51	D1
4	4B	CB
5	4C	CC
6	4D	CD
7	47	C7
8	48	C8
9	49	C9
NUM LOCK	45	C5
*	37	B7
–	4A	CA
+	4E	CE
INS	52	D2
DEL	53	D3
ENTER	E0 1C	E0 9C
/	E0 35	E0 B5

NOTE: All values are expressed in hexadecimal.

Chapter 9

I/O Interrupts

Those interrupts that are generally considered to be input/output interrupts are defined in Table 9-1. For information pertaining to the use and programming of interrupts, see Chapter 6, "Software Interface."

Table 9-1. I/O Interrupts

INTERRUPT	FUNCTION
05H	Print Screen
0BH	Communications (COM2)
0CH	Communications (COM1)
0DH	Alternate Parallel Printer (LPT2)
0FH	Parallel Printer (LPT1)
14H	Serial Input/Output
17H	Printer Input/Output
18H	Parallel/Serial Configuration

Print Screen (INT 05H)

This interrupt sends the contents of the screen to the LPT device (usually a printer). It will only print out valid characters. Graphics that do not match a valid character shape will be ignored. INT 05H performs exactly the same function as the SHIFT-PRTSC entry.

To aid in using the Print Screen function, a byte of global RAM at location 0050:0000 has been reserved as a status byte. While the print screen routine is executing, this byte is set to 01H (this is actually a flag used by the print screen routine to prevent additional print screen requests from being honored while it is executing). Upon completion of the screen dump, the status byte will be 0H if no errors occurred or 0FFH if an error was encountered (usually this is caused by a printer time-out).

Custom Communications (INT 0BH, INT 0CH, INT 0DH, and INT 0FH)

Some application programs initialize these interrupt vectors and supply custom service routines for serial or parallel communications. They are not initialized by either DOS or firmware, and no service routines are provided for these instructions. If you initialize the vectors and provide code for the service routine, you may use them to provide custom communications interrupt handlers. The vectors are commonly assigned as follows:

INT 0BH — COM2
INT 0CH — COM1
INT 0DH — LPT2
INT 0FH — LPT1

Serial Input/Output (INT 14H)

The INT 14H instruction will allow you to perform the serial functions described in Table 9-2. Each function code is described in the following paragraphs. Also refer to the "Parallel/Serial Configuration" section in this chapter.

Before the interrupt can be successfully executed, register AH must be loaded with the specified function code. Register DX must be loaded with the serial device designator, 0 or 1 (for COM1 or COM2). Note that the specified serial device must be addressable by the system.

Table 9-2. Serial Device Function Codes

CODE	DESCRIPTION
00H	Initialize the serial input/output port.
01H	Send character in AL to the serial port.
02H	Receive character in AL from serial port.
03H	Read communications status.

Function Code 00H: Initialize the Serial Input/Output Port — This function code allows the interrupt to determine the mode of the selected port by the bit-mapped value contained in AL and described in Table 9-3.

Table 9-3. Mode-Select Byte Breakdown

BIT	DESCRIPTION
0	With bit 1, sets word length (see Table 9-4).
1	With bit 0 sets word length (see Table 9-4).
2	Number of stop bits. Set to 0 for 1 stop bit, or 1 for 2 stop bits.
3	With bit 4, sets parity selection (see Table 9-5).
4	With bit 3, sets parity selection (see Table 9-5).
5	With bits 6 and 7, sets baud rate (see Table 9-6).
6	With bits 5 and 7, sets baud rate (see Table 9-6).
7	With bits 5 and 6, sets baud rate (see Table 9-6).

Table 9-4. Word Length Selection

BIT 1	BIT 0	WORD LENGTH
0	0	5 bits (not supported in PC-compatible computers)
0	1	6 bits (not supported in PC-compatible computers)
1	0	7 bits
1	1	8 bits

NOTE: Transmitted 8-bit words will be converted to the specified length. Received words will be converted from the received length to a full 8-bits. Extra bits will be ignored or set to 0.

Table 9-5. Parity Selection

BIT 4	BIT 3	SELECTION
0	0	No parity.
0	1	Odd parity.
1	0	Don't care.
1	1	Even parity.

Table 9-6. Baud Rate Selection

BIT 7	BIT 6	BIT 5	BAUD RATE
0	0	0	110
0	0	1	150
0	1	0	300
0	1	1	600
1	0	0	1200
1	0	1	2400
1	1	0	4800
1	1	1	9600

Upon return from the initialization function, the status of the serial port will be in AX; AH will contain the line control status and AL the modem control status. See the Function Code 03H: Read Communications Status description for details.

Function Code 01H: Send Character to the Serial Port — This function code will cause the interrupt to transmit a byte out the serial port. Upon return from this call, the most-significant bit of register AH will contain the transmit status. If the routine could not transmit the character placed in AL, the transmit status will be set to 1. The remaining AX bits will reflect the status as defined in the discussion of Function Code 03H: Read Communications Status.

Function Code 02H: Receive Character from the Serial Port — This function code will cause the interrupt to receive a byte from the serial port. Upon return from this call bits 7, 4, 3, 2, and 1 of AH will contain the data transfer status. Function Code 03H: Read Communication Status describes the transfer status information. If register AH is set to 0, the routine has read the byte properly into AL. If AH is not 0, some type of error occurred. In these cases, timeout errors refer to either the absence of the data set ready (DSR) or clear to send (CTS) signals.

Function Code 03H: Read Communications Status — This function code will cause the interrupt to check the communications interface status. The status is returned as a bit-mapped value in register AX. Register AH contains the line control status and register AL contains the modem control status, as shown in Tables 9-7 and 9-8.

Table 9-7. Line Control Status (Register AH)

BIT	DESCRIPTION
0	Received data ready — when set (1), the received data is ready.
1	Overrun error — when set (1), an overrun error has occurred. Another character arrived before the system read the previously received character.
2	Parity error — when set (1), a parity error has occurred. The parity of the incoming character did not match the programmed parity selection.
3	Framing error — when set (1), a framing error has occurred. The first bit of the word and the stop bit handle framing. Either the system incorrectly received the transmitted character or the total number of characters in the received word did not match the programmed selection.
4	Break detect — when set (1) remote BREAK key operation was detected. Some terminals have a BREAK key that, when pressed, will hold the data line low.
5	Transmitter holding register empty — when set (1), the transmitter holding register is empty. The serial input/output channel is ready to receive another character for transmission.
6	Transmitter shift register empty — when set (1), the transmitter shift register is empty. The specified serial input/output channel is not currently transmitting.
7	Timeout error — when set (1), a time-out error occurred. The receiving device did not respond within a reasonable period of time. A printer, when off-line, will typically return this type of error.

Table 9-8. Modem Control Status (Register AL)

BIT	DESCRIPTION
0	Clear to send — when set (1), the clear to send (CTS) line has changed state.
1	Data set ready — when set (1), the data set ready (DSR) status has changed.
2	End of ring pulse detector — when set (1), the tailing edge of the ring signal has been detected.
3	Carrier detect — when set (1), the carrier detect (CD) signal has changed state.
4	CTS status — this bit reflects the current state (high or low) of the clear to send line.
5	DSR status — this bit reflects the current state (high or low) of the data set ready line.
6	Ring indicator — this bit reflects the current state (high or low) of the ring indicator line.
7	Carrier detect status — this bit reflects the current state (high or low) of the carrier detect line.

Printer Input/Output (INT 17H)

The INT 17H instruction is used to perform input/output functions with the parallel port on the computer. The configuration table (as described in the Parallel/Serial Configuration section of this chapter) is referenced by this interrupt for proper operation. Table 9-9 describes the function codes of this interrupt.

Table 9-9. Parallel Device Function Codes

FUNCTION CODE	DESCRIPTION
00H	Print the next character in AL. If the parallel device does not return a ready status within a reasonable amount of time, bit 0 of register AH will be set to 1, otherwise the register will return the status, as described in function code 2. Register DX must contain the port to be used (00H - 03H).
01H	Initialize the parallel port. Register AH will contain the device status following execution of the interrupt; Refer to function code 2.

Table 9-9 (continued). Parallel Device Function Codes

FUNCTION CODE	DESCRIPTION
02H	This function will cause the interrupt to return the status of the device in register AH as follows. Bit 0 — Time out error, device not ready. Bit 1 — Not used. Bit 2 — Not used. Bit 3 — Input/output error. Bit 4 — Device on-line. Bit 5 — Out of paper signal. Bit 6 — Character acknowledge. Bit 7 — Device is busy or in an error state.

Parallel/Serial Configuration (INT 18H)

INT 18H serves as a pointer to the parallel and serial device tables. These tables are used by MS-DOS and many application packages to configure the computer's serial and parallel ports. In other PC-compatible computers it may be used as a pointer to a resident BASIC.

In order to maintain compatibility, the default configuration table values are loaded from MFM-300. Application programs can install another pointer to new tables, if needed, to implement specific features. The MS-DOS CONFIGUR utility modifies the parameters in these tables as described in the documentation for the operating system. Do not execute an INT 18H instruction. Figure 9-1 illustrates the format of the serial and parallel device parameter mapping table.

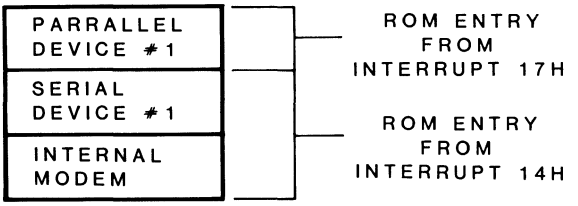


Figure 9-1. Serial and Parallel Device Layout

Parallel Format

The format for each of the three parallel maps is shown in Table 9-10.

Table 9-10. Parallel Map Format

BYTE	PARAMETER DESCRIPTION
1	Performs remapping, parity, and upper-/lowercase handling as follows. Bits 0 - 3: 00 — No remapping. 01 — Remap parallel output to COM1. 10 — Remap parallel output to COM2. 11 — Not used. Bit 4: Strip parity on output. Bit 5: Not used. Bit 6: Map lowercase output to uppercase. Bit 7: Not used.
2	The pad character to be used after a CR.
3	Number of pad characters to be sent.
4	ROM data segment time out value for a parallel device.

Serial Format

The format for the two serial maps is shown in Table 9-11.

Table 9-11. Serial Map Format

BYTE	PARAMETER DESCRIPTION
1	Type of handshake used by device (see Table 9-12).
2	Attributes of transmission (see Table 9-13).
3	Pad character after carriage return.
4	Number of pad characters to issue.
5	Flag for burst count for ETX/ACK.
6	Counter for ETX/ACK.
7	Device initialization (see Table 9-14).
8	Reserved.

The serial byte #1, serial byte #2, and serial byte #7 breakdowns are detailed in Tables 9-12, 9-13, and 9-14 respectively.

Table 9-12. Serial Byte #1 Breakdown

BIT	PARAMETER DESCRIPTION
0	Compatibility bit. If logic 1, IBM compatible DTR and RTS high. If logic 0, handshake determined by bit 1.
1	Protocol bit. If logic 0, hardware handshake. If logic 1, software handshake.
2	If bit 1 is logic 0, bit 2 is logic 1 if DTR handshake active, logic 0 if not. If bit 1 is logic 1, bit 2 is logic 1 if DC1/DC3 handshake, logic 0 if ETX/ACK handshake.
3	If bit 1 is logic 0, bit 3 is polarity of DTR if DTR active. If bit 1 is logic 1, bit 3 is logic 1 if waiting for handshake character, logic 0, if not.
4	If bit 1 is 0, bit 4 is logic 1 if RTS handshake active, 0 if not.
5	If bit 1 if logic 0, bit 5 is polarity of RTS if RTS active, 0 if low, 1 if high.

Table 9-13. Serial Byte #2 Breakdown

BIT	PARAMETER DESCRIPTION
0	If logic 0, do not strip parity on input; if logic 1, strip parity.
1	If logic 0, do not strip parity on output; if logic 1, strip parity.
2	If logic 1, map lowercase to uppercase on input; if logic 0, do not.
3	If logic 1, map lowercase to uppercase on output; if logic 0, do not.

Table 9-14. Serial Byte #7 Breakdown

BIT	PARAMETER DESCRIPTION
0,1	Word size.
2	Number of stop bits.
3,4	Define parity; yes/no odd/even.
5, 6, and 7	Baud rate (see Table 9-6).

Chapter 10

Disk Drive Interrupts

This chapter provides specific information about the nine interrupts used for disk input, output, and setup. Table 10-1 lists the disk drive interrupts described in this chapter.

Table 10-1. Disk Drive Interrupts

INTERRUPT	FUNCTION
0EH	Floppy disk hardware interrupt return
13H	Floppy and hard disk software input/output call
19H	Operating system boot routine call
1EH	Floppy disk parameters
40H	Floppy disk software input/output call
41H	Hard disk 0 parameters
46H	Hard disk 1 parameters
4BH	Hard disk 2 parameters
76H	Hard disk hardware interrupt return

Floppy and Hard Disk Drive Hardware Interrupt Return (INT 0EH and INT 76H)

INT 0EH services command-complete interrupts from the floppy disk controller (FDC). Upon completion of a command, the FDC returns an interrupt causing a jump to this vector. INT 76H performs the same function for the hard disk controller. These interrupt routines cannot be altered without affecting the performance of the computer. Do not modify these interrupts with user programs.

Disk Input/Output (INT 13H and INT 40H)

These interrupts provide access to the system's disk drives. They allow you to call a particular BIOS routine and carry out a disk operation such as reading or writing a sector. INT 13H can handle either floppy disk or hard disk operations. INT 40H can handle only floppy disk operations. When INT 13H is called, the routine checks the contents of the DL register. If the most-significant bit of the register is set (the drive identification code is 8X, indicating that a hard disk operation was requested), the request is handled. If the most-significant bit is clear (the drive identification code is 0X), the floppy disk operation is passed to the floppy disk driver routine and handled by INT 40H. The drive identification codes are listed in Table 10-3. Floppy operations may be passed directly to INT 40H without going through INT 13H.

Before calling a particular routine, the CPU registers must be initialized. Any interrupt that is called without first initializing the appropriate CPU registers will cause a jump to a return (RET) function (no operation). The discussion of each function provides specific information about initializing the registers. To identify a particular function, its code is loaded into the AH register. Table 10-2 describes the function codes used with INT 13H and INT 40H.

Table 10-2. Disk Drive Function Codes

DE	FUNCTION	DESCRIPTION
00H	Reset disk system	Resets each disk drive and drive controller and reprograms the controllers for the appropriate drive characteristics.
01H	Read disk status	Returns the error status information for the drive last used in a disk input/output operation.
02H	Read sectors from a disk	Reads specified sector(s) into a buffer and returns error information.

Table 10-2 (continued). Disk Drive Function Codes

CODE	FUNCTION	DESCRIPTION
03H	Write sectors to a disk	Writes buffer contents to specified sector(s) and returns error information.
04H	Verify sector(s)	Tests specified sector(s) for readability and returns error information.
05H	Format a track	Formats specified tracks and returns error information.
08H	Return drive parameters	Returns drive parameter block information for the drive specified.
09H	Set hard drive parameters	Writes hard drive parameter block information from the tables pointed to by INT 41H and INT 46H to the RDC.
0AH	Read sectors with ECC bytes	Reads data and ECC bytes from specified sectors to specified memory block.
0BH	Write sectors with ECC bytes	Writes data and ECC bytes from specified memory block to specified number of sectors.
0CH	Seek	Moves the heads of the specified drive to the specified track or cylinder.
0DH	Reset hard disk drives	Resets the specified hard disk drive.
10H	Test	For hard disk drive operations, this function tests the status of the drive ready line. For floppy disk drive operations, it tests the disk changed line status.

Table 10-2 (continued). Disk Drive Function Codes

CODE	FUNCTION	DESCRIPTION
11H	Set drive type/recalibrate	For floppy disk drive operations, this function allows the drive type to be specified prior to formatting. For hard disk drive operations, this function causes the specified drive to seek to track 00.
14H	RDC self-test	This function causes the hard disk controller (RDC) to execute its internal diagnostics and report any errors.
15H	Return DASD type	If a hard disk drive ("direct access storage device") is installed, this function returns the number of 512-byte sectors available.
18H	Acknowledge cartridge change	This function call acknowledges the cartridge changed signal for the drive specified.
19H	Reinitialize DMA 360 drive	This function call causes the specified DMA 360 (removable cartridge) drive to reset. If the specified drive specified is not a DMA 360 drive, an error code is generated.
1BH	Assert recovery mode	This function call is used locally by the hard disk drive interrupt handler in the event of errors during read operations on removable cartridge drives.
1CH	Deassert recovery mode	This function call is used locally by the hard disk drive interrupt handler in the event of errors during read operations on removable cartridge drives.

Drive Selection

The routines called by INT 13H or INT 40H control a maximum of two floppy disk drives and two hard disk drives. All function calls except reset drive system (00H) and read disk status (01H) require that you select a particular drive to perform the operation. To select a particular drive, load one of the drive codes listed in Table 10-3 into the DL register.

Table 10-3. Drive Identification Codes

CODE	DRIVE
00H	Floppy disk drive 1
01H	Floppy disk drive 2
80H	Hard disk drive 1
81H	Hard disk drive 2

Error Status Codes

Many of the function calls return an error status code in either the AH or the AL register. The description of the individual functions indicates which register to read for that particular call. Table 10-4 lists each of the codes and describes the disk status indicated.

Hard disk drive operations retry four times before setting the CF flag and sending the error status code to either the AH or the AL register. If either register or the carry flag indicates an error, reset the drive using function code 00H and then retry the operation.

Floppy disk drive operations do not retry. The carry flag is set and an error status code is sent to the appropriate register as soon as an error is encountered. If the CF flag and AH or AL register indicate an error occurred, reset the drive using function code 00H and retry the operation at least three times.

Table 10-4. INT 13H and INT 40H Error Status Codes

CODE	DISK STATUS OR ERROR
00H	No error. The last operation was successfully completed.
01H	Illegal function. An illegal function number was called.
02H	No address mark. The address mark or sector header was not found.
03H	Write protected. A write operation was attempted to a write-protected disk.
04H	Sector not found. The selected sector could not be located.
05H	Reset failed. The reset operation failed. Track 00 was not detected or invalid drive information was stored in CMOS.
06H	Disk change. The disk has been changed since the last disk I/O operation.
07H	Parameter failure. The drive parameters could not be sent to the controller.
08H	DMA overrun. A DMA overrun occurred and data was lost.
09H	Transfer incomplete. The data to be transferred would not fit in the specified segment.
0BH	Bad track. The controller detected a bad track flag.
10H	CRC error. The CRC read from the disk did not match the calculated CRC value. (Refer to Chapter 20 for the CRC polynomial.)
11H	Soft error. The controller detected a correctable error and the data read was corrected using ECC.
20H	Invalid information. The FDC or hard disk controller responded to the function call with invalid information.
40H	Seek failure. The drive could not seek to the specified track.

Table 10-4 (continued). INT 13H and INT 40H Error Status Codes

CODE	DISK STATUS OR ERROR
80H	No response. The FDC or hard disk controller did not respond to the function call.
BBH	Undefined error. The controller responded with an undefined error code.
FFH	No drive status. The sense drive status operation failed.

Function Call Codes

Function Code 00H: Reset Disk System — Calls the interrupt routine that resets each disk drive in the computer. The reset sequence moves the read/write head(s) to track 0. This interrupt also resets both drive controllers and programs them with the drive characteristics stored in the drive parameter tables pointed to by INT 1EH, INT 41H, INT 46H, and INT 4BH.

Function Code 01H: Read Disk Status — Calls the interrupt routine that returns the disk status code in register AL. Use this call when an applications program needs to see the disk status generated by the last disk operation. Table 10-4 describes the error status codes returned by this operation.

Function Code 02H: Read Sector(s) — Calls the interrupt routine that reads the specified number of sectors into the disk buffer. Table 10-5 lists the information that must be loaded into each CPU register prior to calling this function. This information is also used to initialize the CPU for write sectors, verify data, read sectors and ECC bytes, and write sectors and ECC bytes function calls.

Table 10-5. CPU Register Initialization — Function Codes 02H - 04H and 0AH - 0BH

REGISTER	CONTENTS
AH	Initialize this register with the function code.
AL	Initialize this register with the number of sectors to be transferred.
CH	For floppy drive operations, initialize this register with the starting track. For hard disk operations, initialize this register with the eight least-significant bits of the starting cylinder number.
CL	For floppy drive operations, initialize this register with the logical starting sector number. For hard drive operations, initialize the six least-significant bits with the logical starting sector number and the two most-significant bits with the two most-significant bits of the starting cylinder number.
DH	Initialize this register with the head or disk side number. Use either 00H or 01H if register DL specifies a floppy disk drive. Use a value between 00H - 0FH if DL specifies a hard disk drive.
DL	Refer to Table 10-3 and initialize this register with the drive identification code.
ES:BX	For function call 02H, initialize this register pair with the address of the disk buffer, the segment number in register ES, and the offset address in register BX. For function call 03H, these registers will contain a pointer to the memory address for the data to be transferred. The ES:BX register pair need not be initialized for function call 04H since the call verifies data without actual transferring it to the CPU.

Upon completion of the operation, the controller returns drive status information through register AH and the full carry (CF) flag. If the operation was successful, the CF flag will be clear. Register AH will contain one of the error status codes listed in Table 10-4.

Function Code 03H: Write Sector(s) — Calls the interrupt routine that writes the contents of the disk buffer into the specified number of sectors on the disk. Refer to Table 10-5 for the initialization information required for this operation. As with the read sector call, upon completion of the operation, the drive controller returns drive status information through the AH register and CF flag. The AH register will contain the error status code (refer to Table 10-4) and the CF flag will be cleared if the operation was successfully completed.

Function Code 04H: Verify Sector(s) — Calls the interrupt routine that tests the specified sector(s) to verify that the data is readible. If errors that cannot be corrected using either ECC or CRC methods are detected, the CF flag is set. If correctable errors are encountered, the operation is considered successfully completed and the CF flag is cleared. In either case, an error code (refer to Table 10-4) is returned in the AH register. The data read is not transferred during this procedure. Table 10-5 lists the initialization information required for this operation.

Function Code 05H: Format a Track — Calls the interrupt routine that formats the specified track. You must place the sector header information for each sector on the track in a block in memory. Table 10-6 lists the initialization information that must be loaded into the CPU registers prior to calling this function. The CF flag will be set upon completion of the operation if an error occurred. The error status code (see Table 10-4) will be returned in register AH.

Table 10-6. CPU Register Initialization — Function Code 05H

REGISTER	CONTENTS
AH	Initialize this register with the function code.
AL	For floppy drive operations, initialize this register with the number of sectors per track. For hard drive operations, initialize this register with the value (01H to 10H) of the interleave. For example, for an interleave of 2 to 1, enter 02H.
CH	For hard drive operations, initialize this register with the eight least-significant bits of the logical starting cylinder. This register is not used for floppy drive operations.
CL	For hard drive operations, initialize two most-significant bits of this register with the two most-significant bits of the logical starting cylinder number. This register is not used for floppy drive operations.
DH	For floppy drive operations, initialize this register with the side select number (00H or 01H). For hard drive operations, initialize this register with the head number (00H - 0FH).
DL	Refer to Table 10-3 and initialize this register with the drive identification code.
ES:BX	For floppy drive operations, this register pair is initialized as a pointer to a memory block containing the sector header information for each of the sectors to be formatted. The register pair is not used for hard drive operations.

Function Code 08H: Return Drive Parameters — Calls the interrupt routine that reads and reports the drive parameters block for the drive specified in the DL register. To execute the function call, load the function code into register AH and the drive identification code from Table 10-3 into the DL register. The controller then reads the parameters block and returns the drive parameters information to one of the CPU registers. Table 10-7 lists the CPU registers and their contents. For floppy drive operations, if no drives are installed, each of the registers listed will contain 00H. If the drive number is invalid, the drive type is unknown, or CMOS drive information is invalid, bad, or not present, the value returned in the DL register will be valid and all other registers will contain 00H. For hard drive operations, if an error occurs an error status code (see Table 10-4) will be returned in register AH. In either case, if an error occurs the CF flag will be set.

Table 10-7. CPU Register Results — Function Code 08H

REGISTER	CONTENTS
AH	This register returns the value 00H for all floppy operations. For hard drive operations, it returns an error status code.
BL	For floppy drive operations, this register returns the drive type information stored in CMOS. It is not used for hard drive operations.
CH	For floppy drive operations, this register returns the last addressable track number. For hard drive operations, the last addressable cylinder number is returned.
CL	This register returns the last addressable sector number.
DL	This register returns the number of drives installed.
DH	This register returns the last addressable head number for the drive selected.
ES:DI	For floppy drive operations, this register pair returns an address pointer for the drive parameters block for the selected drive. The register pair is not used for hard drive operations.

Function Code 09H: Set Hard Drive Parameters — Calls the interrupt routine that causes the drive parameters block pointed to by INT 41H to be sent to the controller as drive characteristics for hard disk drive 0 and the drive parameters block pointed to by INT 46H to be sent as drive characteristics for hard disk drive 1. The format for the drive parameters blocks is described Table 10-16. To execute the function call, load the function code into the AH register and the starting drive code (see Table 10-3) into the DL register. If the drive characteristics cannot be transferred to the controller, the CF flag will be set and the AH register will return an error status code.

Function Code 0AH: Read Sectors with ECC Bytes — Calls the interrupt routine that reads the specified long sectors into a specified memory block. This function is similar to the read sectors function (function code 02H) but is valid for hard drive operations only. Typically, it causes 516-byte sectors (512 data bytes and 4 ECC bytes) to be read from hard disks. Table 10-5 lists the CPU register initialization information required for this function call. Upon completion of the operation, an error status code (see Table 10-4) is returned to register AH. If the operation was successfully completed, the CF flag will be cleared.

Function Code 0BH: Write Sectors with ECC Bytes — Calls the interrupt routine that writes long sectors from a specified memory block into specified sectors. This function is similar to the write sectors function (function code 03H) but is valid only for hard drive operations. Typically, it causes 516-byte sectors (512 data bytes and 4 ECC bytes) to be written. Table 10-5 lists the CPU register initialization information required for this function call. Upon completion of the operation, an error status code (see Table 10-4) is returned to register AH. If the operation was successfully completed, the CF flag will be cleared.

Function Code 0CH: Seek — Calls the interrupt routine that moves the heads of the specified hard disk drive to the specified cylinder. Table 10-8 lists the CPU initialization information required for this function call. Upon successful completion of the operation the CF flag is cleared. The error status code (see Table 10-4) is returned in register AH.

Table 10-8. CPU Register Initialization — Function Code 0CH

REGISTER	CONTENTS
AH	Initialize this register with the function code.
CH	Initialize this register with the eight least-significant bits of the cylinder number.
CL	Initialize the two least-significant bits of this register with the two most-significant bits of the cylinder number.
DL	Initialize this register with a hard disk drive identification code from Table 10-3.

Function Code 0DH: Reset Hard Disk Drive — Calls the interrupt routine that resets the specified hard drive. This function call is similar to the reset disk system function (function call 00H) but resets only the specified hard drive. To execute this call, initialize the DL register with a hard disk drive identification code from Table 10-3. If the operation is successfully completed the CF flag will be cleared. An error status code (see Table 10-4) is returned in register AH. This command is valid for hard drive operations only.

Function Code 10H: Test — Calls the interrupt routine to check the status of the drive ready line for the specified hard disk drive or the status of the media changed line for floppy disk drives. To execute this call, initialize the AH register with the function code and the DL register with a disk drive identification code from Table 10-3.

Upon completion of this operation on floppy drives, the CF flag will be set if a media change has been detected or clear if no media change is detected. Table 10-9 describes the media state code that will be returned in the AH register. Upon completion of the operation on a hard drive, the CF flag will be set if the drive is not ready or cleared if the drive is ready. An error status code (see Table 10-4) will be returned to the AH register.

Table 10-9. CPU Register Results — Function Code 10H

MEDIA STATE CODE	DESCRIPTION
00H	This code indicates that the media changed line was inactive.
06H	This code indicates that the media changed line was active.
80H	This code indicates that the operation was attempted on a drive that was not present in the system.

Function Code 11H: Set Drive Type/Recalibrate — For hard drive operations, this function calls the interrupt routine that recalibrates the specified drive (move the read/write heads to cylinder 0). To execute this call, register AH is initialized with the function code and register DL is initialized with the drive identification code from Table 10-3. Upon completion of the operation, the CF flag is set if an error was encountered. An error status code (see Table 10-4) is returned in the AH register.

For floppy drive operations this function calls the interrupt routine that sets the device type for the specified floppy drive. Either this function call or function call 12H must be successfully executed prior to formatting a floppy disk. To execute this function call, initialize the AH register with the function code, the DL register with a drive identification code from Table 10-3, and the AL register with a device type code from Table 10-10. Upon completion of the operation, an error status code (see Table 10-4) will be returned in register AH.

Table 10-10. CPU Register Initialization — Function Code 11H

DEVICE TYPE CODE	DESCRIPTION
01H	This code indicates a 360K disk in a 360K drive.
02H	This code indicates a 360K disk in a 1.2M drive.
03H	This code indicates a 1.2M disk in a 1.2M drive.
04H	This code indicates a 720K disk in a 1.2M drive.

Function Code 14H: Hard Disk Controller Self-Test — Calls the interrupt routine that runs the controller's self-tests. These tests include a check of the controller's processor, data buffer, ECC circuits, and ROM checksum. To execute these tests, initialize the AH register with the function code. Upon successful completion of the tests, the CF flag will be cleared. An error status code (see Table 10-4) is returned in the AH register. This function is valid for hard disk drive operations only.

Function Code 15H: Return DASD (Direct Access Storage Device) Type — Calls the interrupt routine that returns the number of 512-byte data sectors available on the selected hard disk drive. This function code is valid for hard drive operations only. To execute this call, initialize the AH register with the function code and the DL register with a hard drive identification code. The routine returns OOH in the AH register if the operation was attempted on a drive not present in the system and 03H if the drive specified was a valid hard disk drive. The CX and DX registers return the number of available data sectors. This number is valid only if the AH register contains the 03H valid drive code.

Function Code 18H: Acknowledge Cartridge Changed — Calls the interrupt routine that acknowledges and deasserts the cartridge changed signal. To execute this call, initialize the AH register with the function code and the DL register with a hard drive identification code from Table 10-3. Upon successful completion of the operation, the CF flag is cleared. An error status code (see Table 10-4) is returned in the AH register. This function is valid for removable cartridge hard drives only.

Function Code 19H: Reinitialize DMA 360 (Removable Cartridge) Drive — Calls the interrupt routine that reinitializes the specified removable cartridge hard disk drive. To execute this function call, load the function code into register AH and the hard disk drive identification code from Table 10-3 into the DL register. If the operation is attempted on a drive that is not a DMA 360 drive or if an error occurs, the CF flag is set. An error status code (see Table 10-4) is returned in the AH register. This function is valid for removable cartridge hard disk drives only.

Function Code 1BH: Assert Recovery Mode — Calls the interrupt routine that asserts the recovery mode line on removable cartridge drives. This routine is used locally by the hard disk interrupt handler when an error is detected during a read operation. This function is valid for removable cartridge hard drives only.

Function Code 1CH: Deassert Recovery Mode — Calls the interrupt routine that deasserts the recovery mode line on removable cartridge drives. This routine is used locally by the hard disk interrupt handler when an error is detected during a read operation. This function is valid for removable cartridge hard drives only.

Booting an Operating System (INT 19H)

INT 19H attempts to read from sector 1 of track 0 floppy drive 0. The code located at this position is loaded into memory and executed. If the operation is unsuccessful, the routine attempts to boot hard drive 0. If the boot routine fails again, an error message will appear and control passes to the Monitor program.

Floppy Disk Parameters (INT 1EH)

INT 1EH contains an address pointer. The pointer allows access to an area in ROM that contains the disk parameter tables for the supported floppy disk drive and media combinations. When you turn the computer on, the pointer is loaded with the starting address for the appropriate disk parameter table. Table 10-11 describes the structure of the tables.

Table 10-11. ROM Disk Parameters

BYTE	FUNCTION	DESCRIPTION
01	Specify 1	Bits 0 - 3 contain the head unload time (the range is 16 - 240 milliseconds in 16-millisecond increments). Bits 4 - 7 contain the head step rate for the drive (the range is 1 - 16 milliseconds in 1-millisecond increments, in reverse order — FH = 1 millisecond, EH = 2 milliseconds, and so on). Typically, this byte will contain 0FH.
02	Specify 2	Bits 1 - 7 contain the head load time (the range is 2 - 254 milliseconds in 2-millisecond increments). Bit 0 contains the DMA flag. If bit 0 is clear (0), the DMA mode in use. This byte will contain 02H.
03	Motor delay	This is the amount of time the motor will remain on after the last disk operation, measured in timer ticks. The timer tick rate is 18.2 timer ticks per second, or one every 55 ms. A value of 25H is used for all supported media and drive combinations.
04	Bytes per sector	This byte contains a hex code that sets the number of data bytes per sector. A value of 02H, indicating 512 bytes per sector, is used for all supported drive and media combinations.

Table 10-11 (continued). ROM Disk Parameters

BYTE	FUNCTION	DESCRIPTION
05	Sectors per track	This byte contains the number of the last addressable sector on the track. 09H indicates a double-density disk. 0FH indicates a high-density disk.
06	Gap length	This byte contains the hex code that establishes the length of gap 3 for read, write, and verify operations. Refer to Table 10-14. The information contained in this byte overrides any user-programmed gap length during INT 13H and INT 40H operations.
07	Data length	This byte allows partial sectors. If the bytes per sector code equals 00H, then any number between 00H and FFH may be entered, allowing a sector length from 1 byte to 255 bytes. Enter FFH if the bytes per sector code (byte 4) is a number other than 00H.
08	Reformat gap length	This byte contains a hex code that sets the length of gap 3 for format operations. This code is overridden by the format gap length contained in any user-programmed disk parameter tables. See Table 10-14.
09	Reformat data	This byte contains the bit pattern used by the format command to fill and test the sector. Typically, the bit pattern 1110 0101 (E5H) is used.
0A	Head settle	This byte contains the head settle time, measured in 1-millisecond increments. A value of 0FH is used for all supported drive and media combinations.[1]

Table 10-11 (continued). ROM Disk Parameters

BYTE	FUNCTION	DESCRIPTION
0B	Motor start	This byte contains the motor start time, measured in 125-millisecond increments. A value of 08H is used for all supported drive and media combinations.[2]
0C	Cylinders per disk	This byte contains the number of the last addressable track on the disk. 4FH indicates a high- density (80 track) disk. 27H indicates a double-density (40 track) disk.
0DH	Data transfer rate	This byte contains a hex code that indicates the data transfer rate. 00H indicates a transfer rate of 500kbs, 40H indicates 300kbs, and 80H indicates 250kbs.

NOTES

1. A minimum head settle time of 15 ms for high density drives and 20 ms for double-density drives is enforced for write operations.

2. A minimum motor start time of 1 second is enforced for write and format operations. A minimum motor start time of 625 ms is enforced for read or verify operations.

The computer supports the following three media/drive combinations with ROM disk parameter tables:

- 720K media in a 720K drive (720/720)
- 720K media in a 1.44M drive (720/1.44)
- 1.44M media in a 1.44M drive (1.44/1.44).

Table 10-12 lists the bytes per sector, gap length, and format gap length parameters used for each combination.

Table 10-12. Drive Parameters — INT 1EH

MEDIA/ DRIVE	BYTES PER SECTOR (DECIMAL)	BYTES PER SECTOR CODE	SECTORS PER TRACK	GAP LENGTH	FORMAT GAP LENGTH
360/360	512	02H	09H	2AH	50H
360/1.2	512	02H	09H	2AH	50H
1.2/1.2	512	02H	0FH	1BH	54H
720/720	512	02H	09H	2AH	50H
720/1.44	512	02H	09H	2AH	50H
1.44/1.44	512	02H	12H	1BH	6CH

User-programmed disk parameter tables may be created for this computer. These tables must be created if you intend your software to be used on older PC-compatible systems. Many older systems do not have built-in ROM disk parameters and rely entirely on user-programmed disk parameter tables. Tables 10-13 and 10-14 contain the information for building your own disk parameter tables. In systems like this computer, the ROM disk parameter table overrides some of the information contained in user-programmed disk parameter tables as noted in the tables. Also, the user-programmed disk parameter table does not permit a variable last addressable track parameter or a variable data transfer rate. Use the appropriate ROM disk table where these parameters are required.

Table 10-13. Floppy Disk Parameters

BYTE	FUNCTION	DESCRIPTION
01	Specify 1	Bits 0 - 3 contain the head unload time (the range is 16 - 240 milliseconds in 16-millisecond increments). Bits 4 - 7 contain the head step rate for the drive (the range is 1 - 16 milliseconds in 1-millisecond increments, in reverse order — 0FH = 1 millisecond, 0EH = 2 milliseconds, and so on).
02	Specify 2	Bits 1 - 7 contain the head load time (the range is 2 - 254 milliseconds in 2-millisecond increments). Bit 0 contains the DMA flag. If bit 0 is clear (0), the DMA mode is enabled.
03	Motor delay	This is the amount of time the motor will remain on after the last disk operation, measured in timer ticks. The timer tick rate is 18.2 timer ticks per second, or one every 55 ms.
04	Bytes per sector	This byte contains the number of data bytes per sector. Refer to Table 10-13.
05	Sectors per track	This byte contains the number of sectors per track.
06	Gap length	This byte contains the gap length of gap 3 for read and write operations. Refer to Table 10-14. This information is overridden by the gap length specified in the most-similar ROM disk parameters table during INT 13H and INT 40H operations.

Table 10-13 (continued). Floppy Disk Parameters

BYTE	FUNCTION	DESCRIPTION
07	Data length	If the bytes per sector parameter equals 00H, this byte specifies bytes per sector. Any value between 00H and FFH may be entered. This allows partial sector read, write, and verify operations. If the bytes per sector parameter (byte 04) is not 00H, enter FFH.
08	Formats gap length	This byte contains the gap length of gap 3 for format operations. Refer to Table 10-13. "Reformat gap length" does not override this parameter.
09	Format data	This byte contains the bit pattern used by the format command to fill and test the sector. Typically, the bit pattern 1110 0101 (E5H) is used.
0A	Head settle time	This byte contains the head settle time, measured in 1-millisecond increments.[1]
0B	Motor start time	This byte contains the motor start time, measured in 125-millisecond increments.[2]

NOTES

1. A minimum head settle time of 15 ms for high density drives and 20 ms for double-density drives is enforced for write operations.

2. A minimum motor start time of 1 second is enforced for write and format operations. A minimum motor start time of 625 ms is enforced for read or verify operations.

Hard Disk Parameters (INT 41H, INT 46H, and INT 4BH)

These vectors contain address pointers to memory blocks that contain the hard disk drive parameter tables. INT 41H contains the vector for hard drive 0, INT 46H for hard drive 1, and INT 4BH for hard drive 2. Table 10-14 describes the hard disk parameter table structure.

Table 10-14. Hard Disk Parameters

ENTRY	FUNCTION	DESCRIPTION
01	Max cylinders	This 16-bit data word contains the last addressable cylinder number.
02	Max heads	This 8-bit data byte contains the last addressable head number.
03	Not used	Fill this 16-bit data word with 0000H.
04	Precomp	This 16-bit data word contains the number of the first precompensated cylinder. 0000H indicates that precompensation begins on the first cylinder. FFFFH indicates that no precompensation is required.
05	ECC length	This 8-bit data word contains the maximum number of error bytes that can be corrected with ECC.
06	Options 200	Enter 08H if the drive has more than eight addressable heads.
07-09	Not used	Fill these 8-bit data bytes with 00H.
0A	Landing zone	This 16-bit data word contains the number of the cylinder to be used as a head landing zone.

Table 10-14 (continued). Hard Disk Parameters

ENTRY	FUNCTION	DESCRIPTION
0B	Sectors per track	This 8-bit data byte contains the number of the last addressable sector on the track. This computer accepts only 17 sectors per track formats for hard disks. Enter 11H.
0C	Reserve 200	This 8-bit data byte contains a hex code that indicates whether the drive is a removable cartridge type, servo-stepper type, or removable cartridge and servo-stepper type. Enter 01H for cartridge types, 02H for servo-stepper types, and 03 for drives that are removable cartridge and servo-stepper types.

Chapter 11

Video Interrupts

This chapter describes the video interrupts and how these interrupts are used in programming the video system in this computer. This computer is capable of operating in color graphics mode (CGA), enhanced graphics mode (EGA), Hercules mode, and monochrome video mode (MDA). Operating characteristics and functional abilities are different for each of these video modes. Table 11-1 lists the video interrupts discussed in this chapter.

Table 11-1. Video Interrupts

INTERRUPT	FUNCTION
10H	Video input/output
1DH	Video initialization
1FH	Defining characters

Video Input/Output (INT 10H)

INT 10H provides communications with the video logic section of the computer. To use this interrupt, place one of the function codes described in Table 11-2 in register AH. The following pages contain descriptions of each of the function codes and indicate any other CPU registers that need to be initialized.

Table 11-2. Video Input/Output Function Codes

CODE	FUNCTION
00H	Set video mode
01H	Set cursor type
02H	Set cursor position
03H	Read cursor position

Table 11-2 (continued). Video Input/Output Function Codes

CODE	FUNCTION
04H	Read light pen position (not used)
05H	Select active display page
06H	Scroll an area of screen up
07H	Scroll an area of screen down
08H	Read cursor position
09H	Send character and attribute to screen
0AH	Send character to screen
0BH	Set graphics foreground color
0CH	Write graphics pixel
0DH	Read graphics pixel
0EH	Dumb terminal display
0FH	Return video state
10H	Set palette registers
11H	Character generator routine
12H	Alternate select
13H	Write character string
64H	Set scroll mode

Function Code 00H: Set Video Mode — Causes the interrupt to set the video mode according to the values placed in register AL and described in Table 11-3.

Table 11-3. Video Modes

MODE	DESCRIPTION
0	40 characters by 25 rows, monochrome display at the RGB output.
1	40 characters by 25 rows, color text display at the RGB output.
2	80 characters by 25 rows, monochrome display at the RGB output. Individual video pages may be scrolled without affecting other video pages.
3	80 characters by 25 rows, color text display at the RGB output. Individual text pages may be scrolled.
4	320 × 200 pixel resolution, color graphics display at the RGB output. Text displays appear as 40 characters by 25 rows.

Table 11-3 (continued). Video Modes

MODE	DESCRIPTION
5	320 × 200 pixel resolution, monochrome graphics display at the RGB output. Text displays appear as 40 characters by 25 rows.
6	640 × 200 pixel resolution, monochrome graphics display at the RGB output. All three scrolling modes are available. Text displays appear as 80 characters by 25 rows.
7	80 characters by 25 rows, monochrome text display. This mode requires a Zenith Data Systems expansion box with a monochrome display adapter.

Function Code 01H: Set Cursor Type — Causes the interrupt to set the cursor size. Load bits 0 - 4 of the CH register with the starting scan line number of the cursor and bits 0 - 4 of the CL register with the ending scan line number. For a full block cursor, the starting scan line bits must equal 0 and the ending scan line bits must equal 7. While it is possible to load any value between 0 and 31, use only values 0 - 7 for the starting and ending scan line. Also, the starting scan value must not be greater than the ending scan value, or no cursor will appear.

Function Code 02H: Set Cursor Position — Causes the interrupt to place the cursor at the specified row and column location on the screen. Load register DH with the cursor row number and register DL with the column number. The row numbers extend from 0 to 24. The column numbers extend from 0 to 79 (80 characters per line) or 39 (40 characters per line). Therefore, when you load a value of 0 in the DH and DL registers, the cursor appears in the upper left-hand corner of the screen. Load register BH with the video page number; this must be consistent with the video mode selected (only page 0 is valid when you are in the graphics mode).

Function Code 03H: Read Cursor Position — Causes the interrupt to return the current cursor information. The row number will be in register DH and the column number will be in register DL. The cursor's starting scan line number will be in register CH and its ending scan line number will be in register CL. Before executing the interrupt, register BH must contain the page number.

Function Code 04H: Read Light Pen Position — A light pen option is not currently supported with this computer, even though this is a valid function code for PC-compatible computers.

The function code causes the interrupt to attempt to obtain the light pen's position. After the interrupt has been executed, register AH will contain the light pen trigger/switch status (0 or 1). A value of 1 in the AH register indicates an active switch closure (the light pen is on). If the register contains a 0 instead, the switch is not activated (the light pen is off). If the AH register contains a 1, then a number of other registers contain related position information. The DH register will contain the row number and register DL will contain the column number. The CH register will contain the scan line row number (0 - 199), and register BX will contain the pixel column number. The pixel column number can vary between 0 - 319 or 0 - 639, depending upon the graphics mode.

Function Code 05H: Select Active Display Page — Causes the interrupt to display the page specified in register AL. In the text modes, unused portions of video memory can serve as additional video pages. In video modes 0 and 1, the valid page numbers are 0 to 7; in video modes 2 and 3, they are 0 - 3. In the graphics modes, only page 0 is valid.

Function Code 06H: Scroll an Area of the Screen Up — Causes the interrupt to scroll the specified area of the screen up a specified number of lines. Prior to executing this interrupt, you must load certain registers with specific data. The CX register must contain the upper left-hand coordinates (place the row number in register CH and the column number in register CL) of the scroll area. The DX register must contain the lower right-hand coordinates (place the row number in register DH and the column number in register DL) of the same area. The BH register must contain the attribute byte for the blank lines and the AL register must contain the number of lines to scroll. If the AL register contains a 0, the entire window will be cleared.

NOTE: If a hardware or smooth scrolling mode is active when this interrupt executes, it will function only if the entire screen is scrolled. Hardware and software scrolling operations affect the entire screen and not just a portion of the screen.

Function Code 07H: Scroll an Area of the Screen Down — Causes the interrupt to scroll the specified area of the screen down the specified number of lines. Before executing this interrupt, you must load certain registers with specific data. The CX register must contain the upper left-hand coordinates (place the row number in register CH and the column number in register CL) of the scroll area. The DX register must contain the lower right-hand coordinates (place the row number in register DH and the column number in register DL) for the same area. The BH register must contain the attribute byte for the blank lines and the AL register must contain the number of lines to scroll. If the AL register contains a 0, the entire window will be cleared.

Function Code 08H: Read Cursor Position — Causes the interrupt to return the character and attribute codes for the character that resides at the current cursor position. In text modes, register BH must contain the video page number. Upon completion of the routine, the character code will be in register AL and the attribute code will be in register AH.

Function Code 09H: Send Character and Attribute to Screen — Causes the interrupt to write the specified character and attribute codes to the cursor location. Place the character code in register AL and the attribute code in register BL. Register CX contains the number of times the character is to repeat and, in text modes, the page number is in register BX.

Function Code 0AH: Send Character to Screen — Causes the interrupt to write the specified character, but not the attribute byte, to the screen. Place the character code in register AL, the number of times to repeat the character in register CX, and in text modes, the page number in register BH.

Function Code 0BH: Set Graphics Foreground Color — This function code is only available in mode 4, the 320 × 200 graphics mode. Place a value from 0 to 127 in register BH (see the following text) and a value from 0 to 4 in register BL.

If the value placed in register BH is even, the current background color will become the foreground color (normally, 0 - 31). Values above 15 will select the intensified level of the 16 colors.

If the value placed in register BH is odd, the value in the BL register will determine which one of two available palettes is chosen. The palette and pixel color number will determine the foreground color. A value of 0 will select palette 0 and a value of 1 will select palette 1. Refer to Table 11-4 for the palette number and color number matrix.

Table 11-4. Palette and Pixel Colors

COLOR NUMBER	PALETTE 0	PALETTE 1
1	Green	Cyan
2	Red	Magenta
3	Yellow	White

Function Code 0CH: Write Graphics Pixel — Causes the interrupt to light a single pixel at the specified location on the screen. Place the pixel row number in register DX, the pixel column number in register CX, and the color in the AL register. Row numbering extends from 0 - 199 and column numbering from 0 - 319 or 0 - 639. Color values range from 0 - 3 in medium-resolution (320 × 200) mode or 0 - 1 in high-resolution (640 × 200). In all cases, 0 is the background color (usually black). In the high-resolution mode, 1 is the foreground color. In the medium-resolution mode, the palette and the color number determine the color, as described in Table 11-4.

If the most-significant bit (bit 7) of AL is set, the color is XORed with the current color, permitting simple animation. For more information on animation techniques, see the discussion on graphics in the GW-BASIC manual.

Function Code 0DH: Read Graphics Pixel — Returns the color of the pixel at the specified location. Place the pixel row number in register DX and the pixel column number in register CX. The returned color value will appear in the AL register.

Function Code 0EH: Dumb Terminal Display — Causes the interrupt to treat the character as if it were sent to a dumb terminal. The routine will treat the back space (08H), carriage return (0DH), line feed (0AH), and bell (07H) as console commands rather than screen formatting characters. If a character should print at the end of a screen line, the cursor will position at the start of the next line. If you perform a line feed on the last display line of the screen or if a character prints at the last position of the last line, the screen will scroll up one line. When scrolling, the attribute for the new row (when in text mode) will be the same as the attribute of the character at the cursor position on the line when the scrolling takes place.

Place the character in register AL, the foreground color in register BL (for graphics modes), and the display page number in register BH.

Function Code 0FH: Return Video State — Returns the current video state. Register AL will contain the current video mode (refer to function code 00H), register AH will contain the screen width in columns, and register BH will contain the active video page number.

Function Code 10H: Set Palette Registers — Causes the interrupt to set the palette registers. This function may be used to set individual palette registers, set the overscan register, set all palette registers and overscan, or set the intensify/blinking bit as follows:

- To set individual palette registers, set register AL to 00H, register BH to the desired palette register, and register BL to the required value.

- To set the overscan register, set register AL to 01H and register BH to the desired value.

- To set all palette registers and overscan, address ES:DX contains a 17-byte table where bytes 0-15 are the palette values and byte 16 is the overscan value.

- To set the intensify/blinking bit, set register AL to 03H and register BL to 00H to intensify or to 01H to enable blinking.

Function Code 11H: Character Generator Routine — Causes the interrupt to reset the video environment and enter a character generator routine. The functions of this routine are determined by the value you set in register AL as described in the following paragraphs. These functions will not clear the video buffer.

- Set register AL to 00H to load user-specified files. ES:BP is the address of the user table. Set register CX to the number of patterns to store, set DX for the character offset into the table, set BL for the block to load, and set BH to the number of bytes for each character pattern.

- Set register AL to 01H to load ROM monochrome patterns (8 × 14). Set register BL for the block to load.

- Set register AL to 02H to load ROM double-dot patterns (8 × 8). Set register BL for the block to load.

- Set register AL to 03H to set the block specifier. To set register BL for the character generator block specifier, set bits 3 and 2 to the block for attribute bit 3 = 1 and bits 1 and 0 to the block for attribute bit 3 = 0. Call INT 10H with AX = 1000H and BX = 0712H to set the color plane enable register to ignore bit 3 in addressing color palette registers.

The following functions (AL = 1×H) should be used only after a mode set. They are similar to AL = 0×H with the following exceptions:

Page 0 must be active.
The bytes per character is recalculated.
The maximum character rows is recalculated.
The CRT buffer length is recalculated.
The CRTC registers are reprogrammed.

- Set register AL to 10H to load user-specified files. ES:BP is the address of the user table. Set register CX to the number of patterns to store, set DX for the character offset into the table, set BL for the block to load, and set BH to the number of bytes for each character pattern.

- Set register AL to 11H to load ROM monochrome patterns (8 × 14). Set register BL for the block to load.

- Set register AL to 12H to load ROM double-dot patterns (8 × 8). Set register BL for the block to load.

The following functions (AL = 2×H) should be called only immediately after a mode set.

- Set register AL to 20H for user 8 × 8 graphics characters (INT 1FH). ES:BP is the address of the user table.

- Set register AL to 21H for user graphics characters. ES:BP is the address of the user table. Set register CX for the bytes per character and register BL to the desired number of rows. Refer to Table 11-5 for row specifier values.

- Set register AL to 22H for the ROM 8 × 14 set. Set register BL to the desired number of rows. Refer to Table 11-5 for row specifier values.

- Set register AL to 23H for ROM 8 × 8 double dot. Set register BL to the desired number of rows. Refer to Table 11-5 for row specifier values.

Table 11-5 Row Specifier Options

REGISTER BL	NUMBER OF ROWS
00H	User set (DL = number of rows)
01H	14
02H	25
03H	43

- Set register AL to 30H to return information. Register BH must be set to select the information you want returned. Refer to

Table 11-6 for available register values and what information they will select. When exiting the return information routine ES:BP is the specified pointer value, CX is the bytes per character, and DL is the character rows on screen.

Table 11-6. Register Values to Return Information for Function Code 11H

REGISTER BH	INFORMATION RETURNED
00H	INT 1F pointer
01H	INT 44 pointer
02H	ROM 8 × 14 character pointer
03H	ROM 8 × 8 double dot pointer
04H	ROM 8 × 8 double dot (top half) pointer
05H	ROM alternate (9 × 14) pointer

NOTE: When exiting the return information routine (AL = 30H), ES:BP is the specified pointer value, CX is the bytes per character, and DL is the character rows on screen.

Function Code 12H: Alternate Select — Causes the interrupt to return current EGA information or select a routine to set an alternate print screen. Setting register BL to 10H returns current EGA information. Refer to Table 11-7 for interpretation of EGA data. Setting register BL to 20H selects a routine to set up an alternate print screen.

Table 11-7. EGA Information Returned by Function Code 12H

REGISTER	VALUE	EXPLANATION
BH	00H	Currently in color mode.
	01H	Currently in monochrome mode (MDA).
BL	00H	64K video memory
	01H	128K video memory
	02H	192K video memory
	03H	256K video memory
CH	—	Feature bits
CL	—	Switch setting

Function Code 13H: Write Character String — Causes the interrupt to write a specified string where ES:BP contains the address of the string to be written. Set register CX for the character count, set register DX for the position to begin the string (in cursor terms), and set register BH for the page number. Refer to Table 11-8 for further register, string, and cursor information.

NOTE: Carriage return, line feed, back space, and bell are interpreted as commands, not as printable characters.

Table 11-8. Register, String, and Cursor Data for Function Code 13H

EGISTER AL	REGISTER BL	STRING	CURSOR ACTION
00H	Attribute	"Char ... Char"	No movement
01H	Attribute	"Char ... Char"	Moves
02H	Not used	"Char,Attr...Char,Attr"	No movement
03H	Not used	"Char,Attr...Char,Attr"	Moves

Function Code 64H: Set Scroll Mode — Causes the interrupt to select one of three scrolling modes. Place the scrolling mode value in register AL: mode 0 is software scrolling, mode 1 is hardware jump scrolling, and mode 2 is hardware smooth scrolling. Keep in mind the following limitations:

- Hardware scrolling will not work in the 40 × 25 text modes (0 and 1).
- Hardware smooth scrolling works only in the high-resolution graphics mode (6).
- Hardware jump scrolling works only in the graphics modes (4, 5, and 6) and 80 × 25 text mode (3).
- Software scrolling will work in all modes. If you write software that bypasses the Monitor program, use software scrolling.

Video Initialization (INT 1DH)

Unless otherwise programmed by an application program, INT 1DH initializes the video section parameters according to the information stored in the Monitor program ROM. This is the same data used to initialize the video section of the computer when you turn the system on. Table 11-9 defines the values for each register in the 6845-compatible register set. The table shows the relation between each register and the various display modes.

Table 11-9. Video Initialization Default Values

STER NUMBER	TEXT 40 × 25	TEXT 80 × 25	GRAPHICS	MONOCHROME TEXT (TTL)
R0	38	71	38	61
R1	28	50	28	50
R2	2D	5A	2D	52
R3	0A	0A	0A	0F
R4	1F	1F	7F	19
R5	06	06	06	06
R6	19	19	64	19
R7	1C	1C	70	19
R8	02	02	02	02
R9	07	07	01	0D
R10	06	06	06	0B
R11	07	07	07	0C
R12	00	00	00	xx
R13	00	00	00	xx
R14	xx	xx	xx	xx
R15	xx	xx	xx	xx
R16	xx	xx	xx	xx
R17	xx	xx	xx	xx

NOTE: All values are expressed in hexadecimal; "xx" represents any value between 00H and FFH.

Defining Characters (INT 1FH)

INT 1FH allows access to an extended character set for use with the medium- and high-resolution color graphics modes. Normally, the character generator ROM supplies the first 128 characters (00H - 7FH) used in the graphics modes. In addition to these characters, you can create a custom character set of 128 additional characters (80H - FFH) using the following procedure:

1. Allocate a 1K section of memory (not video memory) to hold the character set. You will need eight bytes for each character you create.

2. Define each character in an 8 × 8 matrix as shown by the 7 in Figure 11-1. Since the top line in the matrix butts up against the bottom line of the character above it, you will need to allow for ascenders and descenders. Also, do not forget to allow space between characters.

3. For each line of the matrix, identify which of the eight pixels will be on.

4. With the first pixel as the most-significant bit, add the binary weights of the lit pixels in each row to produce a decimal value for the byte representing that row.

5. Load all eight bytes that form the defined character into the first eight bytes of the memory allocated for the character set.

6. Repeat this procedure for each of the 128 characters in the set.

7. Once the characters are defined and placed in memory, set the pointer at interrupt 1FH (memory location 0000:007C) to the start of memory allocated for the defined character set.

8. Execute the INT 1FH instruction. Then, whenever a character code between 80H and FFH is required for printing, the character from your set will be used.

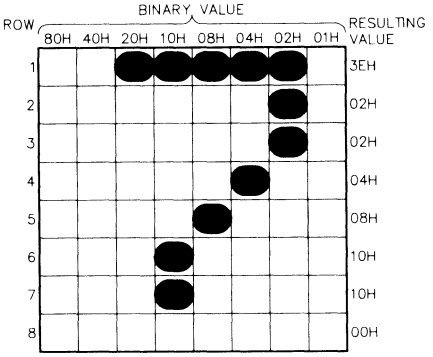


Figure 11-1. Character Design Matrix

EGA Video Considerations

The EGA mode supports a wide range of video modes, including a number of modes also supported by the Z-100 PC standard CGA color video mode. However, video modes 0, 1, 2, and 3 offer expanded capabilities in the EGA mode over the standard CGA system. The discussion in this section is divided into three areas: monochrome video modes, normal color video modes, and enhanced color video modes. Each of these modes requires that a Zenith Data Systems expansion box and an EGA video card be installed.

Monochrome Video Modes

Table 11-10 describes the attributes of the two possible modes that may be used to generate a monochrome display.

Table 11-10. Monochrome Video Modes

VIDEO MODE	DESCRIPTION
7	A monochrome alphanumeric text mode of eighty 9 × 14 characters per line on 25 lines in a screen that has an overall resolution of 720 × 350 pixels. Each character may be displayed in one of four attribute modes. The memory is organized into eight pages and has a base address at B0000H.
F	A monochrome, bit-mapped mode that generates eighty 8 × 14 characters per line on 25 lines in a screen that has an overall resolution of 640 × 350 pixels. Each character may be displayed in one of four attribute modes. The memory is organized into two pages and has a base address at A0000H. This video mode is unique to the EGA format and will not create a display if a Z-100 PC compatible color video mode is controlling the video output.

Normal Color Video Modes

Table 11-11 describes the attributes of the nine possible normal color modes supported by the EGA card.

Table 11-11. Normal Color Video Modes

VIDEO MODE	DESCRIPTION
0	A 16-color alphanumeric mode that generates forty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 320 × 200 pixels.
1	A 16-color alphanumeric mode that generates forty 8 × 8 characters on 25 lines in a screen that has an overall resolution of 320 × 200 pixels.

Table 11-11. Normal Color Video Modes

VIDEO MODE	DESCRIPTION
2	A 16-color alphanumeric mode that generates eighty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 640 × 200 pixels.
3	A 16-color alphanumeric mode that generates eighty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 640 × 200 pixels.
4	A 4-color, bit-mapped mode that generates forty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 320 × 200 pixels.
5	A 4-color, bit-mapped mode that generates forty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 320 × 200 pixels.
6	A 2-color, bit-mapped mode that generates eighty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 640 × 200 pixels.
D	A 16-color, bit-mapped mode that generates forty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 320 × 200 pixels. The memory is organized into eight pages with a base address of A0000H. This mode is unique to the EGA format and will not create a display if a Z-100 PC color video mode is actively controlling the video output.
E	A 16-color, bit-mapped mode that generates eighty 8 × 8 characters per line on 25 lines in a screen that has an overall resolution of 640 × 200 pixels. The memory is organized into four pages with a base address of A0000H. This mode is unique to the EGA format and will not create a display if a Z-100 PC color video mode is actively controlling the video output.

NOTE: In video modes 0, 1, 2, and 3 the memory is organized into eight pages with a base address of B8000H. In video modes 4, 5, and 6 the memory is organized into one page with a base address of B8000H. Unless specified, all modes are compatible with the display produced by a Z-100 PC color video card.

Enhanced Color Video Modes

Table 11-12 describes the attributes of the five possible enhanced color video modes supported by the EGA card.

Table 11-12. Enhanced Color Video Modes

VIDEO MODE	DESCRIPTION
0	A 16-color, alphanumeric mode that generates forty 8 × 14 characters per line on 25 lines in a screen that has an overall resolution of 320 × 350 pixels.
1	A 16-color, alphanumeric mode that generates forty 8 × 14 characters per line on 25 lines in a screen that has an overall resolution of 320 × 350 pixels.
2	A 16-color, alphanumeric mode that generates eighty 8 × 14 characters per line on 25 lines in a screen that has an overall resolution of 640 × by 350 pixels.
3	A 16-color, alphanumeric mode that generates eighty 8 × 14 characters per line on 25 lines in a screen that has an overall resolution of 640 × 350 pixels.
10	A 16-color, bit-mapped mode that generates eighty 8 × 14 characters per line on 25 lines in a screen that has an overall resolution of 640 × 350 pixels. The memory is organized into two pages with a base address of A0000H. The 16 colors can be selected from a palette of 64 colors.

NOTE: In video modes 0, 1, 2, and 3 the memory is organized into eight pages with a base address of B8000H. The 16 colors can be selected from a palette of 64 colors.

Basic Modes of Operation

The 256K of video memory is divided into four separately addressable, but parallel, 64K memory planes. Depending upon the video mode selected, this memory may be combined in different ways. The following discussions describe the basic modes of operation and how they work with the video memory.

Text (Alphanumeric) Modes

In the text modes, a two-byte character/attribute format is used to define each character position on the screen. These two bytes are mapped into assigned locations in video memory with the character byte in memory plane 0 and the attribute byte in memory plane 1.

Normally, bit 3 of the attribute byte determines the intensity characteristic of the character being displayed. However, the EGA format allows bit 3 to be redefined by the character map select register and used as a switch between character sets. This gives the programmer access to two character sets, for a total 512 characters. To accommodate the additional 256 characters, the patterns of both character generators are transferred into memory plane 2 of the video memory. In addition, with 256K of video memory, it is possible to support two additional user-defined, 256-member character sets for a total of 1024 different characters. The patterns of these additional character sets must also be loaded into memory plane 2 of the video memory

To display a screen of data, the CRT controller generates a series of sequential addresses that represent the sequential locations of characters on the screen. These addresses correspond to video memory addresses where memory planes 0 and 1 are accessed. The byte in memory plane 0 combined with the scan row count determines the address of the bit pattern stored in memory plane 2. If bit 3 of the attribute byte has been selected to act as a switch between character sets, the video memory address is modified accordingly. The bit pattern found in memory plane 2 is then assigned the attributes found in the attribute byte and sent as a serial stream of data out the appropriate pins of the video connector.

Monochrome and color attribute data is formatted differently. Table 11-13 describes the attribute byte characteristics for monochrome modes.

Table 11-13. Monochrome Attribute Byte Characteristics

BIT 7	6	5	4	3	2	1	0	ATTRIBUTE DESCRIPTION
B	0	0	0	1	1	1	1	Normal video
B	1	1	1	1	0	0	0	Reverse video
B	0	0	0	1	0	0	0	No display (black output)
B	1	1	1	I	1	1	1	No display (white output)

NOTE: Bit 7 defines whether the character blinks. If bit 7 is a 1, the character will blink. If bit 7 is a 0, the character will not blink. Bit 3 defines either the intensity attribute or the character set. Character set definition is determined by the character map select register. If bit 3 controls the intensity attribute, then a 1 will cause the character to be displayed at high intensity and a 0 will cause the character to be displayed at normal intensity.

Table 11-14 describes the attribute byte characteristics for color modes. The bit pattern for the foreground color and background color are identical. The foreground color attribute occupies bits 0 - 2 and background color attribute occupies bits 4 - 6.

Table 11-14. Color Attribute Byte Characteristics

BIT 6	5	4	2	1	0	NORMAL COLOR	INTENSIFIED COLOR
0	0	0	0	0	0	Black	Dark gray
0	0	1	0	0	1	Blue	Light blue
0	1	0	0	1	0	Green	Light green
0	1	1	0	1	1	Cyan	Light cyan
1	0	0	1	0	0	Red	Light red (pink)

Table 11-14 (continued). Color Attribute Byte Characteristics

BIT 6	5	4	2	1	0	NORMAL COLOR	INTENSIFIED COLOR
1	0	1	1	0	1	Magenta	Light magenta
1	1	0	1	1	0	Brown	Yellow
1	1	1	1	1	1	White	Intensified white

NOTE: Bit 7 defines whether the character blinks. If bit 7 is a 1, the character will blink. If bit 7 is a 0, the character will not blink. Bit 3 defines either the intensity attribute or the character set. Character set definition is determined by the character map select register. If bit 3 controls the intensity attribute, then a 1 will cause the character to be displayed at high intensity and a 0 will cause the character to be displayed at normal intensity.

Graphics Modes

With the exception of video mode F (high-resolution monochrome) and video mode 10 (enhanced color graphics), addressing, mapping, and data formatting with the EGA format are the same as those used with Z-100 PC color video formats and PC-equivalent color/graphics adapters.

In the color video graphics modes, two degrees of resolution are available: 320 × 200 and 640 × 200. 640 × 200 is obtainable only as a monochrome, or two-color, display (one of 16 border colors may be used for the displayed pixels). In the 320 × 200 modes, each displayed pixel may be one of four colors selected from two palettes and the border color.

Medium-Resolution Color

Pixel information for the 320 × 200 display is stored in two memory planes of 8000 bytes each, starting at address B8000H. The even-numbered (0, 2, 4, ... 198) scan row information is stored from B8000H to B9F3FH. The odd-numbered (1, 3, 5, ... 199) scan row information is stored from BA000H to BBF3FH.

Each byte of video memory contains four two-bit pairs that define the color displayed for that pixel. Note that there is no unlit pixel capability; if you want to unlight any given pixel, you will have to sacrifice one of the possible colors (the background color) to black. Table 11-15 defines the color selected for the defined pixel. Since only one palette can be active, Table 11-16 defines the colors in the two possible palettes for these video modes. The 16 available border colors are the same that can be used to display text characters: black, blue, green, cyan, red, magenta, brown, white, dark gray, light blue, light green, light cyan, light red (pink), light magenta, yellow, and intensified white.

Table 11-15. Color Selection

MSB	LSB	COLOR DEFINITION
0	0	The pixel will display the current background color.
0	1	The pixel will display color 1 of the current palette.
1	0	The pixel will display color 2 of the current palette.
1	1	The pixel will display color 3 of the current palette.

Table 11-16. Palette Colors

COLOR NUMBER	PALETTE 1	PALETTE 2
1	Cyan	Green
2	Magenta	Red
3	White	Brown

High-Resolution Color

The high-resolution color mode is actually considered a monochrome display mode. An entire 16K range of video memory is needed to define the on or off states of a full screen of pixels. Each pixel location on the 640 × 200 screen is represented by a location in video memory, starting at B8000H. Because each location can be set only to a 1 or a 0, each pixel may be the border color or not lit. The 16 border colors are: black, blue, green, cyan, red, magenta,

brown, white, dark gray, light blue, light green, light cyan, light red (pink), light magenta, yellow, and intensified white.

Mode F

Mode F is a high-resolution 640 × 350 mode that supports monochrome black and lit-pixel graphics with the following attributes: black, video, blinking video, and intensified video. Two memory planes, which are required to support the four attributes, are both addressed at A0000H by the CPU. The first memory plane is identified as the video memory plane and the second is identified as the intensity memory plane. Actually, a combination of pixel pairs between the two memory planes is required to define the attributes, as shown in Table 11-17.

Table 11-17. Mode F Attributes

ATTRIBUTE	VIDEO MEMORY PLANE	INTENSITY MEMORY PLANE
Black (not lit)	0	0
Video (lit)	0	1
Blinking video	1	0
Intensified video	1	1

Some IBM-equivalent enhanced graphics adapters are supplied with only 64K of video memory. When this is the case, memory planes 0 and 1 and memory planes 2 and 3 are mapped together to form the two 32K memory planes needed to support this mode. Although the four memory planes all reside at an A0000H starting address, memory planes 0 and 2 are addressed as even memory planes and memory planes 1 and 3 are addressed as odd memory planes. When data is brought out of video memory for display, the first byte comes out of the even memory planes, the second out of the odd memory planes, the third out of the even, and so on.

Other EGA formats have 256K of memory, so the memory planes are not mapped together. The first (memory plane 0) and third (memory plane 2) are used for mode F. The second (memory plane 1) and fourth (memory plane 3) are ignored.

Two bits, one from each plane, define each pixel on the screen. Since memory organization is sequential, the first eight pixels are contained in the first byte, starting at video memory address A0000H and with the most-significant bit. The second byte is located at video memory address A0001H, and so on.

Mode 10

Mode 10 supports graphics in 16 colors selected from 64 possibilities. All four available memory planes in video memory are used. Each memory plane normally represents a color or intensity, as shown in Table 11-18. The base colors that can be produced without programming additional registers are described in Table 11-19. The other possible shades are obtained by programming the palette registers in the attribute controller.

Table 11-18. Mode 10 Memory Plane Assignments

MEMORY PLANE	COLOR OR FUNCTION
0	Blue
1	Green
2	Red
3	Intensified

Table 11-19. Mode 10 Base Colors

BP3	BP2	BP1	BP0	RESULTING COLOR
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta

Table 11-19 (continued). Mode 10 Base Colors

BP3	BP2	BP1	BP0	RESULTING COLOR
0	1	1	0	Brown
0	1	1	1	White
1	0	0	0	Dark gray
1	0	0	1	Light blue
1	0	1	0	Light green
1	0	1	1	Light cyan
1	1	0	0	Light red
1	1	0	1	Light magenta
1	1	1	0	Yellow
1	1	1	1	Intensified white

There are 16 palette registers that correspond to the 16 possible base colors. The registers allow you to remap the color represented by the bit patterns shown in the preceding table to any one of 64 possible colors that can result from combining the six bits in each register. Refer to Table 11-20 for the bit-to-color relationship. Setting a bit to 1 activates that color. Setting the bit to 0 deactivates that color.

Table 11-20. Palette Register Color Attributes

BIT	COLOR
0	Blue
1	Green
2	Red
3	Secondary blue
4	Secondary green
5	Secondary red

Part III

Hardware

Chapter 12
The CPU

This chapter explains the programmer accessible features of the Intel 80386 microprocessor. This device is one of the most advanced microprocessor products in use today.

Introduction

The CPU/memory board contains the majority of the computer's control circuitry. The board contains the 80386 CPU, a socket for the 80387 coprocessor, the system firmware, and up to 3M of computer memory. The board also contains address and data bus buffers, system decoding and control logic, 32-bit memory control logic, and the system clock circuits. It also contains the interfaces to the video and I/O boards. Figure 12-1 is a block diagram of the board showing the CPU portion.

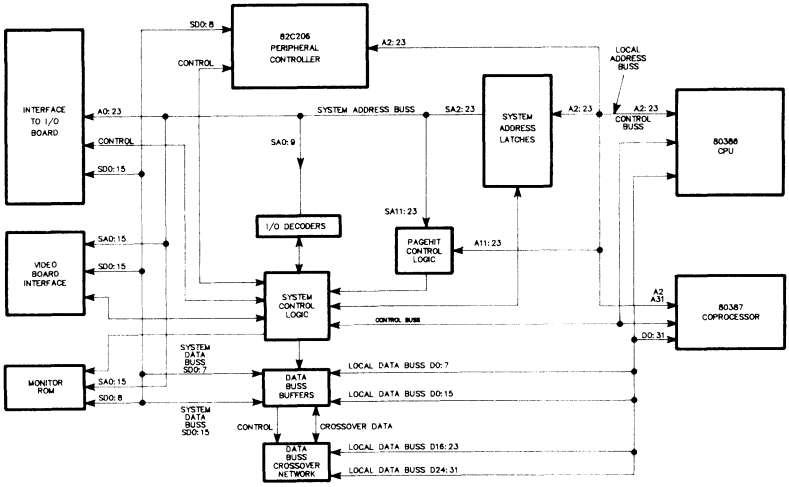


Figure 12-1. CPU Block Diagram

The 80386 is a full 32-bit microprocessor employing a 32-bit data path internally as well as externally. The design of the 80386 is an extension of the 8086 and the 80286 microprocessors previously developed by Intel. The 80386 instruction set (refer to Appendix A) is a superset of the instruction sets of these earlier microprocessors. This provides a great deal of flexibility by permitting programs developed for the earlier devices to run on the 80386. The 80386 differs from the earlier 8086 and the 80286 only in the extent of its capabilities. The processor maintains the instruction sets and capabilities of these earlier devices. It extends those capabilities by adding more instructions, features, and raw speed to the computing environment.

80386 Base Architecture

This section describes the base architecture of the 80386. For the purposes of this discussion, the base architecture refers to the general internal architecture of the processor.

NOTE: The processor architecture alters somewhat when the device switches between protected mode or virtual 8086 mode. Later sections of this chapter contain discussions of these modes.

Internally, the 80386 consists of three main units: the central processing unit, the memory management unit, and the bus interface. The central processing unit consists of the execution unit and the instruction unit, which are very similar to the earlier 8086 and 80286 microprocessors. However, the 80386 also includes a memory management unit with virtual page capability. The memory management unit consists of a segmentation unit and a paging unit. Figure 12-2 is a block diagram of the 80386 architecture.

Bus Interface Unit — The bus interface unit (BIU) provides the interface between the 80386 microprocessor and the rest of the computer. This unit accepts requests for code fetches and data transfers and provides priorities for these requests. This unit also provides the control signals required for execution of the current bus cycle and controls the interface to the system coprocessor.

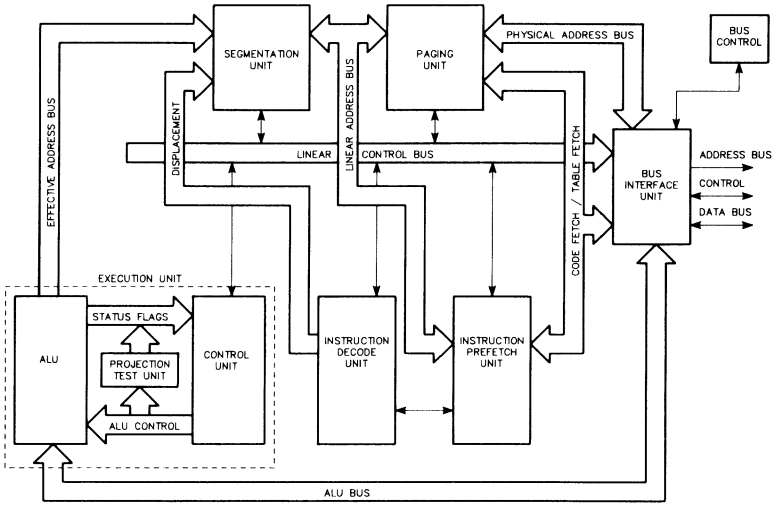


Figure 12-2. 80386 Processor Architecture

Instruction Prefetch Unit — The prefetch unit essentially performs a program look ahead function for the 80386. This unit monitors the bus interface unit. When the BIU is not performing data transfers, this unit uses the BIU to sequentially prefetch instructions. The BIU uses a 16-byte queue to store these instructions. Since prefetch operations have a lower processor priority than data transfers, the activity of the prefetch unit does not degrade processor performance.

Instruction Decode Unit — This unit accepts the instructions from the prefetch unit and translates them into microcode for the processor. This unit has a FIFO buffer capable of storing three decoded instructions. This unit also generates immediate data and opcode offsets for logical addresses.

Execution Unit — The execution unit consists of the ALU, the control unit, the protection test unit, and eight 32-bit general purpose registers. This unit accepts the decoded instructions and executes them. In doing this, it also communicates with other portions of the microprocessor to obtain required data.

Segmentation Unit — The segmentation unit converts the logical addresses into linear addresses under the control of the execution unit. The segmentation unit also checks for segmentaton violations. The processor uses an on-chip cache to help speed up these translations.

Paging Unit — If the paging mode is active in the 80386, this unit will translate linear addresses into physical addresses. If the paging mode is not active, no translation takes place. The paging unit also uses an on-chip cache to speed up translations. The processor sends the physical addresses to the BIU for memory and I/O accesses.

Register Resources

The 80386 microprocessor contains 32 programmer-accessible registers. These registers can be grouped according to their functions, as follows:

- General purpose registers
- Instruction pointer and flags
- Segment registers
- Control registers
- Address registers
- Debug registers
- Test registers.

The design of the registers within the 80386 maintains compatibility with previous Intel processors. The registers in the 80386 are a superset of the registers contained in the 8086 and 80286. Table

12-1 provides an overall graphic view of the various registers and their uses in the 80386 while Figure 12-3 illustrates the internal register organization.

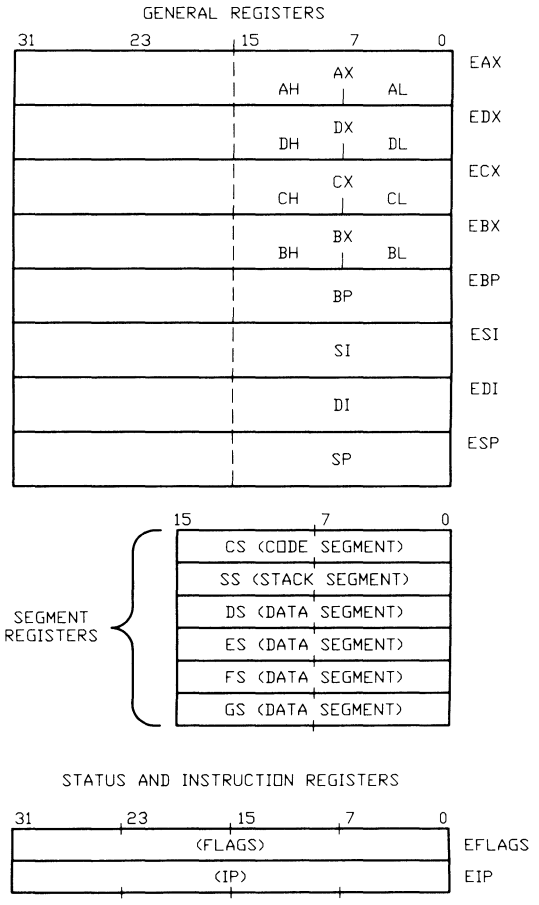


Figure 12-3. 80386 Internal Register Structure

Table 12-1. 80386 Register Usage

	USAGE MODE					
	REAL		PROTECTED		VIRTUAL	
REGISTER	LOAD	STORE	LOAD	STORE	LOAD	STORE
General	Y	Y	Y	Y	Y	Y
Segment	Y	Y	Y	Y	Y	Y
Flag	Y	Y	Y	Y	I/O	I/O
Control	Y	Y	P	P	N	Y
GDTR	Y	Y	P	Y	N	Y
IDTR	Y	Y	P	Y	N	Y
LDTR	N	N	P	Y	N	N
TR	N	N	P	Y	N	N
Debug	Y	Y	P	P	N	N
Test	Y	Y	P	P	N	N

NOTES: I/O This indicates that the PUSHF and POPF instructions are I/O privilege level sensitive in this mode.

P This indicates that the registers can be accessed only when the current privilege level is equal to zero.

The registers illustrated in Figure 12-3 are task specific. The processor automatically loads the registers with new contents whenever the processor executes a task switch operation. Table 12-2 describes the functions of these registers.

Table 12-2. 80386 Internal Registers

REGISTER	DESCRIPTION
EAX	32-bit general purpose register containing the AX, AH and AL registers. Normally operates as the 32-bit accumulator.
AX	16-bit accumulator register (least-significant 16 bits of register EAX). This register consists of the 8-bit AH (high) and AL (low) registers. Specifically used by word multiply, word divide, and word input/output instructions.
AH	The high eight bits of register AX. Specifically used by 16-bit byte multiply and byte divide instructions.

Table 12-2 (continued). 80386 Internal Registers

REGISTER	DESCRIPTION
AL	The low eight bits of register AX. Specifically used by 16-bit byte multiply, byte divide, byte input/output, translate, and decimal arithmetic instructions.
EBX	32-bit general purpose register containing the BX, BH, and BL registers. Functions as the 32-bit base register.
BX	16-bit base register (least-significant 16 bits of register EBX). This register consists of the 8-bit BH (high) and BL (low) registers. Specifically used by 16-bit translate instructions.
BH	The high eight bits of register BX.
BL	The low eight bits of register BX.
ECX	32-bit general purpose register containing the CX, CH, and CL registers.
CX	16-bit count register (least-significant 16 bit of register ECX). This register consists of the 8-bit CH (high) and CL (low) registers. Specifically used by 16-bit string operations and loops.
CH	The high eight bits of register CX.
CL	The low eight bits of register CX. Specifically used by variable shift and rotate instructions.
EDX	32-bit general purpose register containing the DX, DH, and DL registers.
DX	16-bit data register (least-significant 16 bits of register EDX). This register consists of the 8-bit DH (high) and DL (low) registers. Specifically used by 16-bit word multiply, word divide, and indirect input/output instructions.
DH	The high eight bits of register DX.
DL	The low eight bits of register DX.
ESI	32-bit general purpose register containing the SI register.
SI	16-bit source index register (least-significant 16 bits of register ESI). The processor uses this 16-bit register for string operations.
EDI	32-bit general purpose register containing the DI register.
DI	16-bit destination index register (least-significant 16 bits of register EDI). The processor also uses this 16-bit register for string operations.
EBP	32-bit general purpose register containing the BP register.

Table 12-2 (continued). 80386 Internal Registers

REGISTER	DESCRIPTION
BP	16-bit base pointer register (least-significant 16 bits of register EBP). Although the processor does not use this 16-bit register for any specific operation, it does come into use with certain addressing modes.
ESP	32-bit general purpose register containing the SP register.
SP	16-bit stack pointer register (least-significant 16 bits of register ESP). The processor uses this 16-bit register for stack operations. The register contains the current stack address.
CS	Code segment register. This 16-bit register points to the base address of the current code segment. The CPU fetches instructions from this segment.
SS	Stack segment register. This 16-bit register points to the base address of the current stack segment. This segment holds the stack for the CPU.
DS	Data segment register. This 16-bit register points to the base address of the current data segment. This register normally contains variables for the program being executed.
ES	16-bit data segment register.
FS	16-bit data segment register.
GS	16-bit data segment register.
EIP	32-bit register containing the offset, relative to the base of the current code segment, of the next sequential instruction to be executed. This register is not directly visible to the programmer. The low-order 16 bits include the 16-bit IP register.
IP	16-bit instruction pointer (least-significant 16 bits of register EIP). This 16-bit register is similar in purpose to the program counter in 8-bit CPU designs. The contents of this register point to the next instruction location. The processor updates this information and fetches the next instruction through the bus interface unit. Saving the instruction pointer on the stack will insure that the information will remain current.
EFLAGS	32-bit flag register containing the 16-bit register FLAGS.
FLAGS	16-bit flag register (least-significant 16 bits of register EFLAGS) for 16-bit operations.

NOTE: All eight 16-bit general purpose registers fit the definition of accumulator as defined for 8-bit CPUs.

General Purpose Registers

The general purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP) are 32-bit registers. As shown in Figure 12-3 and Table 12-1, they can also be divided into 16-bit and 8-bit sub-units. The registers can hold either data or address information. They support data operands of 1, 8, 16, 32, or 64 bits and bit fields from 1 to 32 bits long. The registers also support address operands of 16 or 32 bits.

Instruction Pointer and Flags

The instruction pointer (EIP) contains the offset of the next instruction requiring execution. The offset always remains relative to the base of the code segment (CS).

The flags register in the 80386 is a 32-bit register containing the 16-bit FLAGS register. Two new registers exist in the 80386 in addition to those used in the 80286. Figure 12-4 illustrates these additions. Table 12-3 describes the 80386 flag set. You should not use the reserved flag bits in the 80386 for systems or applications programs.

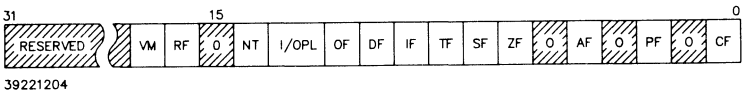


Figure 12-4. 80386 Flags Register

Table 12-3. 80386 Flag Definitions

FLAG BIT	PURPOSE
VM (bit 17)	Virtual 8086 mode. If set while the processor is in protected mode, the 80386 will switch to virtual 8086 operation.
RF (bit 16)	Resume flag. Use this flag in conjunction with the debug registers. If this bit is active, the processor will ignore any debug fault occurring on the next instruction.

Table 12-3 (continued). 80386 Flag Definitions

FLAG BIT	PURPOSE
NT (bit 14)	Nested task. This flag deals with protected mode operations. In the 80386 a task may be nested within another task. A set condition for this flag indicates that the current task has a valid back link to the previous task.
IOPL (bit 12-13)	Input/output privilege level. This flag also pertains to protected mode operations. This flag indicates the maximum current privilege level (CPL) permitted for I/O instruction execution.
OF (bit 11)	Overflow flag. This flag is set if an operation results in a signed overflow. A signed overflow condition exists if an operation causes a carry/borrow into the sign bit without a corresponding carry/borrow out of the high-order bit, or vice versa.
DF (bit 10)	Direction flag. If set, the ESI and/or EDI registers will postdecrement during string operations. If not, the registers will postincrement.
IF (bit 9)	INTR enable flag. If set, the processor will recognize external interrupts on the INTR pin.
TF (bit 8)	Trap enable flag. If set, the 80386 will generate an exception 1 trap after the next instruction executes.
SF (bit 7)	Sign flag. This flag sets if the high-order bit of the operation results in a high logic state (1).
ZF (bit 6)	Zero flag. This flag is set if all bits of the result are zero.
AF (bit 4)	Auxiliary carry flag. This flag is set if the addition of a packed BCD value resulted in a carry out of bit 3. It is also set if the subtraction of a packed BCD value resulted in a borrow from bit 3. This flag is only affected by the operation on bit 3, regardless of the overall operand length.
PF (bit 2)	Parity flag. This flag is set if the low-order eight bits of an operation contains an even number of 1s. Only the value of the low-order eight bits affects the parity flag.
CF (bit 0)	Carry flag. This flag is set for addition if the operation resulted in a carry out of the high order bit. A subtraction operation resulting in a borrow out of the high-order bit will also set the flag.

Segment Registers

There are six 16-bit registers within the 80386. These registers are available for holding segment selector values to identify the currently addressable memory segments. The available segment size varies with the mode of operation for the 80386. In protected mode the segment size can vary from one byte up to the maximum memory size of the computer (16M). Real address mode restricts the segment size to a maximum of 64K. The CS register maintains the current code segment information. The SS register maintains the current stack segment information. The remaining four registers are available for storing information related to data storage. There is an additional register associated with each segment register in the 80386. These registers are the segment descriptor registers.

The segment descriptor registers are not totally visible to the programmer; however, their function is closely tied to the programming task. Each descriptor register contains a 32-bit segment base address, a 20-bit segment limit, and segment attributes. When the processor loads a value into a segment register, it automatically updates the descriptor register. In real address mode the base address is the only portion of the descriptor that the processor updates. This is because the segment limits and attributes remain fixed in real mode. In protected mode the processor updates the entire descriptor register.

Whenever a memory reference occurs, the processor automatically involves the segment descriptor register with the reference. The descriptor register's 32-bit segment base address becomes a part of the linear address calculation, the limit serves in limit check operations. The system checks the attributes against the type of reference requested.

Control Registers

The 80386 employs four 32-bit control registers to maintain the current global machine state. Three of these registers are accessible to the programmer. Figure 12-5 illustrates the 80386 control registers.

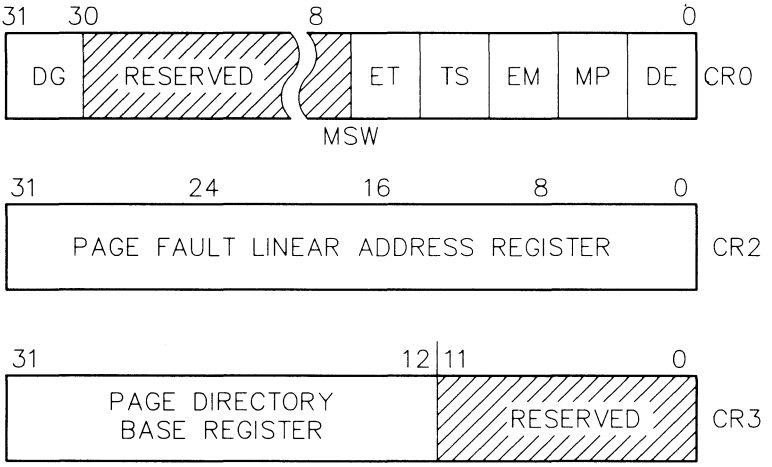


Figure 12-5. 80386 Control Registers

The CR0 register is the machine control register. It contains six bits that define machine status and control modes. In order to maintain compatibility with 80286 protected mode, the low-order 16 bits of CR0 are known as the machine status word. The special LMSW and SMSW instructions (refer to Appendix A) handle these 16 bits the same in the 80386 as they do in the 80286. Intel recommends the use of the MOV CR0, REG class instruction to change bits that the LMSW instruction does not affect. You should not use the reserved bits in this or any other register for programming operations. Table 12-4 details the defined bit functions in CR0.

Table 12-4. Register CR0 Bit Definitions

BIT	FUNCTION
PG (bit 31)	Paging enable. Setting this bit enables the on-chip paging unit.
ET (bit 4)	Processor extension type. If set, this bit indicates that a 80387 coprocessor is present and the processor will use 32-bit protocol. If clear, the processor will expect a 80287 coprocessor and will use 16-bit protocol. For strict 80286 compatibility the LMSW instruction does not affect this bit.
TS (bit 3)	Task switch. Whenever a task switch operation occurs, the processor sets this bit. A coprocessor escape opcode will cause a "coprocessor not available" trap. The trap handler will save the previous task's 80287/80387 context, load the 80287/80387 context for the current task, and clear the TS bit before returning to the faulting opcode.
EM (bit 2)	Emulate coprocessor. If set, this bit will cause all coprocessor opcodes to generate a "coprocessor not available" fault. The setting of this bit does not affect the coprocessor WAIT opcode.
MP (bit 1)	Monitor coprocessor. Use this bit with the TS bit to determine if the WAIT opcode will generate a "coprocessor not available" fault when TS = 1. If the TS and MP bits are both set, the opcode will generate a trap.
PE (bit 0)	Protection enable. Setting this bit enables protected mode in the processor. If the bit is clear, the processor will operate in real mode. Loading the machine status word or CR0 will set this bit. The only way to clear this bit is to load CR0. This allows the 80386 to maintain compatibility with the 80286 .

The CR2 register is the page fault linear address register. This register contains the 32-bit linear address that caused the last detected page fault. You can find additional information about the error in the error code pushed onto the page fault handler's stack.

The CR3 register is the page directory base address register. This register contains the physical base address of the page directory table. In the 80386 this table is always page-aligned on 4K boundaries. This means that the processor ignores the lowest 12 bits of this register. These are reserved bits and should not be used.

If a task switch operation changes the value of CR3, or CR3 is explicitly loaded with any value, all cached page table entries in the paging unit cache are no longer valid.

Address Registers

The 80386 supports four special registers to refer to tables or segments in protected mode. These registers support similar operations in the earlier 80286 processor. The tables or segments referred to are:

- TSS (Task State Segment)
- LDT (Local Descriptor Table)
- GDT (Global Descriptor Table)
- IDT (Interrupt Descriptor Table).

The CPU uses special registers to store these values. These are the system address and system segment registers referred to as TR, LDTR, GDTR and IDTR, respectively. Figure 12-6 illustrates these registers.

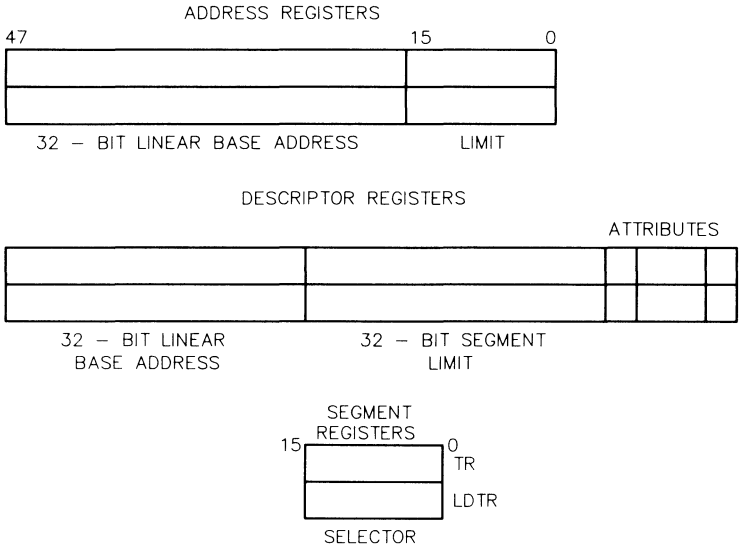


Figure 12-6. 80386 System Segment Registers

The GDTR and IDTR registers hold the 32-bit linear base address and the 16-bit limit of the global and interrupt descriptor tables, respectively. The LDTR and TR registers hold the 16-bit selector for the LDT and TSS segments.

Debug and Test Registers

The 80386 includes a powerful debug feature that is built into the processor hardware. The processor provides eight registers for on-chip support of debugging operations. Four of the registers hold the linear address of the breakpoints. The processor uses one register to control the breakpoint registers while another maintains the status on each of the breakpoints. Do not use the remaining two reserved registers. Figure 12-7 illustrates the debug and test registers.

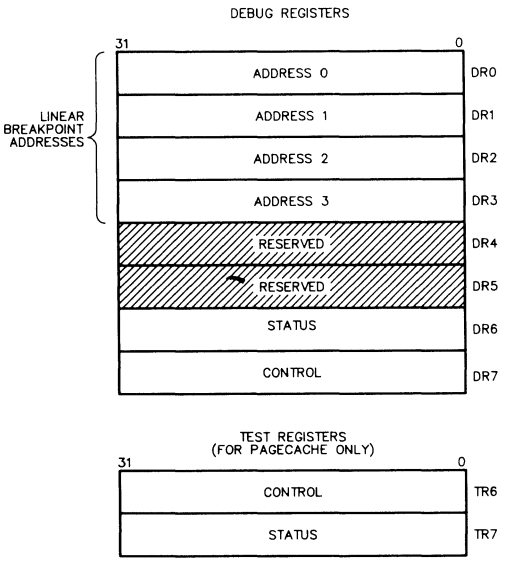


Figure 12-7. 80386 Debug and Test Registers

The 80386 also contains two registers used for testing purposes. These registers, also illustrated in Figure 12-7, control the testing of the content addressable memories in the translation lookaside buffer within the 80386. Register TR6 is the command test register and register TR7 holds the resulting data from the test.

Processor Addressing

The 80386 supports eleven different addressing modes to specify operands used in processing. Two of these modes provide for instructions that operate on register or immediate operands.

Register Operand Mode — In this mode the operand resides in one of the 8-, 16-, or 32-bit general purpose registers.

Immediate Operand Mode — In this mode the instruction itself contains the operand as part of the opcode.

The remaining nine addressing modes are the memory addressing modes. These modes allow the user to specify the effective address of the operand. The Intel documentation referred to in Chapter 4 includes a full description of these modes with examples.

The 80386 uses an advanced method of address calculation. The physical address of a given operand is based on a number of different factors. Some of the more critical factors are the effective address, the linear address, and which mode of operation the processor is executing. You can find details on the address calculation mechanism in Chapter 14.

The linear address used in these calculations consists of two components, the effective address and the segment base address. The processor calculates the effective address by summing any combination of the following four elements:

Displacement — This is an 8- or 32-bit immediate value which follows the instruction.

Base — The contents of any general purpose register makes up this value.

Index — This value may be the contents of any general purpose register except the stack register (ESP).

Scale — You can multiply the value of the index register by a specified scale factor. The permitted values for this scale factor are 1, 2, 4, or 8.

The 80386 is capable of executing 16-bit instructions in both the real and protected modes. This feature enhances processor compatibility with the 8086 and 80286 processors. The processor accomplishes this by examining the D bit in the segment descriptor. If the value of this bit is 1, then the default operand length is 32 bits. Otherwise, the default operand length is 16 bits. Two override prefixes are available to override the value of the D bit. One prefix is the operand size prefix, the other is the address length prefix. Both of these prefixes override the value of the D bit on an individual instruction basis.

The 80386 has two distinct physical address spaces, memory address space and I/O address space. The following sections discuss these address spaces.

Memory

The 80386 views memory space as a collection of 8, 16, or 32-bit quantities. These collections have specific names associated with them. A group of 8-bits is a byte, 16-bits is a word, and 32-bits is a doubleword (dword). The processor stores this information in memory from the lowest memory address to the highest. Thus, a word is two consecutive bytes, with the low-order byte at the lowest address. In a similar manner a dword consists of four consecutive bytes with the low-order byte also occupying the lowest address. This means that a reference to the address of a word or dword in memory is the address of the low-order byte.

The 80386 also supports two larger memory organizations: pages and segments. Segments can be of different variable lengths, shared between programs, or swapped to mass storage devices. Pages of memory consist of 4K blocks of memory.

The 80386 operates with three distinct address spaces: logical, linear, and physical. A logical address consists of a selector and an offset. (This value is also referred to as a virtual address.) An offset is essentially the same as the effective address mentioned in the previous section. It consists of a summation of the various address components (base, index, and displacement) These are combined with the scale factor to create the effective address.

The processor architecture limits each task in the 80386 to a maximum of 16K selectors. Each offset may consist of a memory block up to 4 gigabytes in size. The combination of these two factors means that a total logical address space of 64 terabytes is available per task.

Input/Output

The memory space assigned to I/O in the 80386 has a maximum size of 64K. You may arrange this area in any combination of 8-bit and 16-bit ports as long as the total space does not exceed the 64K maximum. The 64K used for I/O processing resides in physical memory. I/O operations in the 80386 do not involve segmentation or paging hardware.

Any program can directly access the I/O space using the processor IN and OUT instructions. Place the desired port address in the DL, DX, or EDX register. The 80386 will zero extend all 8- and 16-bit port addresses on the upper address lines.

Interrupts

The 80386 uses two methods to alter the normal program flow: interrupts and exceptions. Exceptions handle processor instruction faults while interrupts handle asynchronous external events. This section includes a discussion of both methods.

Exceptions can be classed in one of three groups depending on the reporting method used and whether or not the processor supports instruction restart:

Faults — A class of exceptions detected and serviced before the execution of the faulting instruction. The processor supports instruction restart for this class. The return address from an exception fault routine will always point to the instruction causing the fault.

Traps — A class of exception reported immediately after instruction execution. The processor does not support instruction restart for this class. (User-defined interrupts are examples of traps.)

Aborts — A special class of exceptions which occur when the processor cannot locate the faulting instruction. The processor does not support instruction restart for this class.

Interrupts are restricted to one of two classifications:

Maskable — A hardware interrupt occurring when the interrupt flag bit (IF) is enabled. (Disabling the IF bit turns off interrupt processing.)

Nonmaskable (NMI) — This class of hardware interrupt is reported regardless of the setting of the IF bit in the flags register. The processor provides an internal vector value of 2 for this interrupt but does not provide the usual interrupt acknowledge sequence.

Basically, all interrupts are processed in the same sequence of steps:

1. The interrupt occurs.

2. The processor saves the current program address and flag values on the stack.

3. The processor receives an 8-bit vector identifying the source of the interrupt.

4. The processor selects and executes the appropriate interrupt service routine.

5. At the completion of the service routine (IRET instruction) the processor restores the program address and flags from the stack.

6. Program execution resumes with the instruction immediately after the interrupted instruction.

The 80386 can handle a total of 256 different interrupts/exceptions. It is also possible for more than one interrupt to be in process at the same time. When a maskable interrupt occurs, the processor clears the IF flag in the flags register, disabling interrupts. If the executing program expects additional interrupt requests, the interrupt service

routine must set this bit. Otherwise the processor will not recognize any further interrupt requests from the system. In the case of a nonmaskable interrupt, the 80386 will not recognize additional interrupts until it processes an IRET instruction or it is reset. If a second NMI should occur while one is in process, the 80386 will store it until the processor executes the IRET instruction.

To handle multiple interrupts, the 80386 implements a priority structure. Six different priority levels exist for interrupt processing. Table 12-5 summarizes the priority levels. If multiple interrupts occur, they will be processed according to their priority level (highest to lowest). Table 12-6 summarizes all interrupt vector assignments available in the 80386.

Table 12-5. Interrupt Priorities

PRIORITY	INTERRUPT TYPE
1	Exception faults
2	TRAP Instructions
3	Debug traps
4	Debug faults
5	NMI Interrupt
6	INTR Interrupt

NOTE: Priority level 1 is the highest and level 6 is the lowest.

Table 12-6. 80386 Interrupt Vector Assignments

INTERRUPT VECTOR	FUNCTION
0	Divide error
1	Debug exception
2	NMI interrupt
3	One-byte interrupt
4	Interrupt on overflow
5	Array bounds check

Table 12-6 (continued). 80386 Interrupt Vector Assignments

INTERRUPT VECTOR	FUNCTION
6	Invalid opcode
7	Device not available
8	Double fault
9	Coprocessor segment overrun
10	Invalid TSS
11	Segment not present
12	Stack fault
13	General protection fault
14	Page fault
16	Coprocessor error
17 - 32	Reserved (do not use)
0 - 255	User available two-byte interrupt

When the 80386 receives the vector identifying the interrupt source, it must determine the location of the interrupt service routine. A standard MS-DOS system can accomplish this is in one of two ways.

If the processor is operating in real mode, the processor multiplies the interrupt vector by a factor of four. This multiplication provides the address of the service routine. If the processor is in protected mode, it multiplies the interrupt by a factor of 8. The resulting value is added to the base value in the interrupt descriptor table. This final value is the address of the appropriate interrupt service routine. Naturally, as with other reserved functions in the processor, reserved interrupts should not be used.

Debug Registers

The 80386 provides support, in hardware, for software debugging. These features allow the user to set breakpoints as well as single step through code. Using the debug registers, the programmer can set data access as well as code execution breakpoints. This can be especially useful for debugging code that is not accessible through more normal debug facilities.

The 80386 contains eight registers dedicated to debugging operations. Two reserved registers reduce the programmer usable total to six. Of the remaining registers, four contain breakpoint addresses and the remaining two contain control and status information. Figure 12-8 illustrates the debug registers in greater detail.

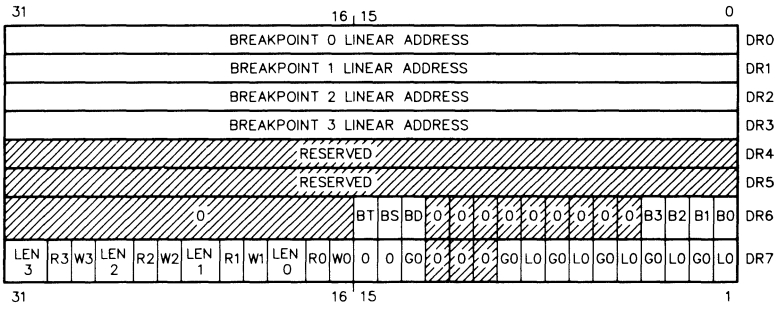


Figure 12-8. 80386 Debug Registers

Seven bits in the DR6 register are usable by the programmer. Table 12-7 defines the function of these bits. The processor controls the setting of the flags in the DR6 register but the software control routines must clear these bits. The DR6 register is also referred to as the debug status register.

Table 12-7. DR6 Register Definitions

BIT	DEFINITION
BT	The processor sets this bit when a task switch occurs to a task where the TSS has the T bit set.
BS	The processor sets this bit if the exception handler is invoked when the TF bit in the flags register is set. This allows the debug handlers to distinguish between single step traps and other debug conditions.
BD	The processor will set this bit if the next instruction attempts to read or write the debug registers when the GD bit in DR7 is set.
B0 - B3	These bits define which of the qualified breakpoints occurred. These bits correspond to the four breakpoint registers. For example, the processor will set bit B1 if it detects breakpoint 1.

Register DR7 is referred to as the debug control register. This register provides control over several debug features including setting breakpoints and breakpoint control options. Table 12-8 provides descriptions for the defined bit fields in register DR7.

Table 12-8. Register DR7 Bit Definitions

BIT	NAME	DEFINITION
0	L0	Local breakpoint enable 0 — This bit enables breakpoint detection for the associated breakpoint register (DR0). If this bit or the associated global bit (G0) is set, and the 80386 detects a breakpoint condition, then the exception handler is invoked.
1	G0	Global breakpoint enable 0 — This bit also enables breakpoint detection for register DR0. If set when the 80386 detects a breakpoint condition, the exception handler will be invoked.
2	L1	Local breakpoint enable 1 — Same as L0 for breakpoint register DR1.
3	G1	Global breakpoint enable 1 — Same as G0 for breakpoint register DR1.
4	L2	Local breakpoint enable 2 — Same as L0 for breakpoint register DR2
5	G2	Global breakpoint enable 2 — Same as G0 for breakpoint register DR2.
6	L3	Local breakpoint enable 3 — Same as L0 for breakpoint register DR3.
7	G3	Global breakpoint enable 3 — Same as G0 for breakpoint register DR3.
8	LE	Local breakpoint match — If this bit or the associated GE bit is set, the 80386 will report any exact data breakpoint match immediately after the completion of the instruction that transferred the operand. This bit is automatically cleared during a task switch.

Table 12-8 (continued). Register DR7 Bit Definitions

BIT	NAME	DEFINITION
9	GE	Global breakpoint match — This bit serves the same function as the previous bit, indicating an exact data breakpoint match. This bit is unaffected by a task switch and remains in the same state.
10 - 12		Reserved — Do not use.
13	GD	Global debug register access bit — If this bit is set when an instruction attempts to read or write any debug register, the 80386 will generate an exception fault. When the exception handler is invoked, it will automatically clear the exception fault. Since access to the debug registers is only permitted in real mode or in protected mode at privilege level 0, this provides extra protection against a debug register access.
14,15		Reserved — Do not use.
16	W0	Write memory access qualifier bit — This bit works in conjunction with the read qualifier bit to determine what type of breakpoint usage is being requested. One pair of read write qualifiers exist for each breakpoint.
17	R0	Read memory access qualifier bit — This bit works with the write qualifier to determine breakpoint usage according to the following pattern:

R	W	USAGE
0	0	Instruction execution only
0	1	Data writes only
1	0	Reserved
1	1	Data reads and writes

Table 12-8 (continued). Register DR7 Bit Definitions

BIT	NAME	DEFINITION
18,19	LEN0	Breakpoint length bit — The 2-bit length field specifies the length of the associated breakpoint field according to the following definition: LEN — FIELD WIDTH 00 — 1 byte 01 — 2 bytes 10 — Reserved 11 — 4 bytes Each breakpoint field is aligned: 2-byte fields begin on word boundaries while 4-byte fields begin on dword boundaries.

NOTE: The read/write qualifiers and the length fields repeat for each of the remaining breakpoints. Each group requires four bits per breakpoint.

NOTE: When using the exact data match bits (LE and GE), make sure that the handler routine requests an exact data breakpoint match. If the program does not request an exact match, a match may occur without being reported. To make sure this happens, when enabling the breakpoint bit, also enable the exact match bit.

Registers DR5 and DR4 are reserved and should not be used. Registers DR0 - DR3 are the linear breakpoint address registers. These registers contain the actual memory location of the selected breakpoints.

Test Registers

The 80386 hardware also incorporates features to allow internal testing. The 80386 can perform two major types of tests: an internal self-test and testing of the translation lookaside buffer.

The internal self-test checks about half of the internal device circuitry, including the control ROM and most of the internal non-random logic. If the test completes successfully, the 80386 will place zeroes in the EAX register. If any other value appears in the register, the device has failed the self-test. After the 80386 completes the self-test it re-initializes and begins normal operation.

Testing of the translation lookaside buffer (TLB) is more complex than the self-test. The two test registers, TR6 and TR7, store information related to this test. Figure 12-9 shows the test registers in greater detail.

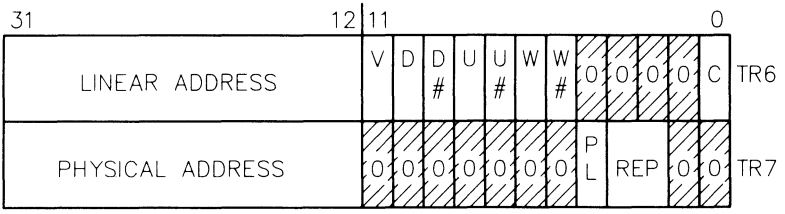


Figure 12-9. 80386 Test Registers

Table 12-9 provides descriptions of the bit positions used in these registers for testing purposes. TR6 is referred to as a test command register while TR7 is the test data register.

Table 12-9. Test Bit Definitions

BIT	NAME	DEFINITION
REGISTER TR6:		
0	C	Command bit — This bit controls writes and lookups in the TLB. When this bit is clear (0), a write into register TR6 will cause an immediate write into the TLB. If this bit is set (1), a write to the TR6 register will cause an immediate TLB lookup.
1 - 4	—	Reserved bits — Do not use.
5	$\overline{W}$	Write bit — If set, this bit allows a write from the TLB.

Table 12-9 (continued). Test Bit Definitions

BIT	NAME	DEFINITION
6	W	Write bit — If set, this bit allows writes to the TLB.
7	$\overline{U}$	User bit — If set, this bit allows a user to read from the TLB.
8	U	User bit — If set, this bit will allow a user to write to the TLB.
9	$\overline{D}$	Dirty bit — If set, this bit signifies that a TLB read has taken place.
10	D	Dirty bit — If set, this bit indicates a TLB write has taken place.
11	V	Valid bit — This bit is set if the current entry in the TLB was valid.
12 - 31		Linear address — These bits in register TR6 form the tag field for the TLB. Whenever a TLB write occurs, a TLB entry is placed in this address space. The rest of that entry is set according to the values in TR6 and TR7. When a TLB lookup occurs, the value in this memory space is compared with the TLB entry. If they match, the rest of the fields in TR6 and TR7 are set from the matching entry.
REGISTER TR7:		
0,1	—	Reserved bits — Do not use.
2,3	REP	If this bit is set during a TLB write, the bits in the REP field determine which of the four TLB blocks will be written to. If the bit is clear, then the pointer in the internal paging unit makes this determination.
4	HT	Hit — This bit is used only during a TLB lookup. During a read if the bit is set (1), it indicates the lookup was a hit. A clear bit indicates a miss. During TLB writes this bit must always be set (1).

Table 12-9 (continued). Test Bit Definitions

BIT	NAME	DEFINITION
5 - 11		Reserved bits — Do not use.
12 - 31		Physical address — These bits form the data field for the TLB. When a write occurs to the TLB, the entry in the linear address space of TR6 gets set to this value. If a TLB lookup occurs, the physical address of the entry is read from this area.

Real Mode

The processor uses real mode to initialize and start operation on power-up. Generally, this mode is most often used to move the 80386 into protected mode. While in real mode, applications programs view the processor as an extremely fast 8086 with a 32-bit register set.

Real mode maintains compatibility with the 80286 by retaining the same memory size limitations, memory addressing, and interrupt handling procedures as real mode on the 80286. All 80386 program instructions are available in this mode except for instructions used in the protected mode (refer to Appendix A). While in this mode the default operand size is 16-bits. Access to the 32-bit register set requires the use of override prefixes. Each memory segment has a size limit of 64K. Each 64K segment must begin on a 16-byte boundary. This implies that in this mode a 32-bit effective address cannot exceed a value of FFFFH.

The limit of available system memory in real mode is 1M. Since segments cannot exceed 64K boundaries, the 80386 will generate an exception if a data operand or data fetch operation exceeds this boundary. The processor supports segment overlapping in real mode to minimize program space requirements. Linear addresses are the same as physical addresses in this mode. The formation of the physical address is accomplished in the same manner as with the 8088 or 8086.

The processor reserves two regions of memory when operating in real mode. The interrupt table is located in the region of 00000H through 003FFH. A 4-byte jump vector is reserved for each of the 256 possible interrupts. System initialization takes place in the reserved area from FFFFFFF0H through FFFFFFFFH.

Protected Architecture

In order to adequately understand the protection mechanism employed by the 80386 CPU, you must first understand some basic concepts as they apply to the 80386. The following sections introduce these concepts and describe their implementation in the hardware of the 80386.

General Protection Concepts

One of the most powerful features of the 80386 is its ability to run in protected mode. In this mode, the CPU is capable of supporting a number of different users, each operating independently of the others. Each user interfaces with a limited portion of the computer and appears to be the only user on the system. The idea of protection is important to ensure that one user's programs and files do not interfere with any other user.

Descriptors

One of the primary responsibilities of the 80386 in protected mode is efficient memory management. The processor has to maintain discrete areas where separate user programs may be operating. To allow these areas to overlap one another would tempt disaster in the system.

One method of memory management employed by Intel processors is segmentation. Segments serve to isolate one region of memory from other regions. In the 80386 an 8-byte data structure, called a descriptor, contains information about each segment. The 80386 uses several different types of descriptors: segment, code, data, system, LDT, TSS, and gate. System descriptors are organized into tables recognized by the system hardware.

The 80386 uses three different types of tables to store descriptor information: global, local, and interrupt. Each type of descriptor is maintained in a table for the system. A descriptor table can vary in size up to a maximum of 64K of memory. This provides enough space for 8192 8-byte descriptors per table. Each table has an associated register which holds a 32-bit linear base address and a 16-bit limit for each table.

The global descriptor table (GDT) contains the descriptors which are potentially available to all tasks in the system. The GDT can contain any type of a descriptor with the exception of interrupt descriptors. The GDT generally contains code and data segments, which are used by the operating system, and the task state segments. The first position of the GDT is not used. This position corresponds to a null selector which defines a null pointer value.

The local descriptor table (LDT) contains the descriptors associated with a specific task in the system. An LDT can only contain code, data, stack, task gate, and call gate descriptors. Normally, an operating system will provide an LDT for each task in the system. The LDT provides a means of separating the code and data segment for an individual task from the rest of the operating system. A task cannot access a segment unless the segment descriptor exists in the current LDT.

The interrupt descriptor table (IDT) contains the descriptors that point to the system interrupt service routines. The IDT can only contain task gates, interrupt gates, and trap gates. To properly service the 80386, the IDT should contain a minimum of 256 bytes of space. Each interrupt in the system must have an entry in the IDT.

Each of the above tables is associated with its own register. The register names refer to the related table; they are GTDR, LDTR, and IDTR. These registers contain the base and limit addresses for their respective tables. The information is moved to and from these registers by the appropriate 80386 instructions.

Segment Access

The 80386 permits two methods of access to the system memory: code accesses and data accesses. In order to determine if a task has legal access rights, the 80386 must be able to determine the type of segment, the instruction used, the type of descriptor, and the CPL and RPL.

Whenever an instruction loads the DS, ES, FS, or GS registers, the 80386 will perform protection validation checks. Selectors loaded into these registers must refer to data or readable code segments only. The privilege validation check consists of comparing the CPL with the EPL. If the EPL is more privileged than the CPL, a general protection fault will result. Instructions that load a selector into the stack segment must refer to the data segment descriptors to find writeable data segments.

Privilege

The 80386 employs a multiple-level protection mechanism. The four-level hierarchal privilege system employed by the 80386 is similar in many respects to the systems used on larger computer systems, such as minicomputers. In fact, the common user/supervisor mode found on many minicomputers is fully supported by the 80386 when in paging mode. The privilege mechanism is the basis for the 80386's protection scheme.

The privilege system in the 80386 provides four separate levels, 0 through 3. Level 0 is the most privileged level in this system; level 3 is the least privileged. In order to access certain system resources, the user must have an authorized privilege level equal to or greater than the level he wishes to access. This idea can be condensed into two general privilege rules that the 80386 follows:

- If data stored in a segment has a privilege level of P, then that data may only be accessed by a program with a privilege level A such that A >= P.

- If a program segment or procedure has a privilege level of P, then that program/procedure can only be called by a task with privilege level B such that B <= P.

In protected mode, each task operating on the 80386 has a privilege level associated with it. The privilege level determines what resources are available to the task within the hardware and the operating system environments.

Privilege Types

A number of different privilege types are available within the 80386 system. The different types allow the 80386 to maintain the integrity of the operating system environment. The following brief discussion identifies these types.

Task Privilege — The 80386 assigns a current privilege level (CPL) to each task in the environment. The CPL can only change when the program transfers control to a code segment with a different privilege level. This transfer must occur via a gate descriptor within the system. This means that if you have an application program executing at a privilege level of 3, it can call an operating system procedure with a privilege level of 1. When this happens, the current privilege level of the application program will change to 1 until the operating system routine finishes execution.

Selector Privilege — The processor assigns a privilege level to each selector in the system. A special field, referred to as the RPL field, contains the selector's privilege level. The RPL field consists of the two least-significant bits of the selector. The processor uses the RPL field value and the CPL of the task to establish an effective privilege level (EPL). The definition of the EPL is the least-privileged level of the compared CPL and RPL fields. Changes to the RPL assignment field are made using the adjust RPL instruction. The main purpose of the RPL is to provide a means to verify that a pointer passed to an operating system procedure does not access data of a higher privilege level than the originating pointer.

I/O Privilege — The I/O privilege level (IOPL) is a special 2-bit field contained within the EFLAG register. This field defines the least-privileged level which may perform unconditional I/O instructions. In order to have unconditional access to I/O instructions, the CPL must be less than or equal to the current IOPL. The IOPL has some other tasks in the system as well.

The IOPL determines whether the CLI or STI instructions can be executed by the calling procedure. These instructions are sometimes called IOPL-sensitive. The IOPL level also determines whether or not the IF bit can be changed when loading a value in the EFLAGS register. If the level of the CPL is greater than the level of the IOPL, then the IF bit will not change when a new value loads in the EFLAGS register.

In order to speed pointer testing and verification of selector values, several instructions are provided in the CPU instruction set. The ARPL, VERR, VERW, LSL, and LAR instructions (refer to Appendix A) help to maintain system integrity by verifying that a given selector refers to the appropriate segment.

Inter-Segment Privilege Transfers

An inter-segment control transfer occurs whenever a selector is loaded into the CS register. Changing a privilege level can only happen if the operation which loaded the selector references the proper descriptor type. There are five different types of control transfers in the 80386 system:

- Intersegment within the same privilege level
- Intersegment to the same or higher privilege level
- Intersegment to a lower privilege level
- Call
- Task switch.

A change of privilege level can only occur via a control transfer, a gate, a task switch, an interrupt, or a trap gate. If the CPL of a task changes as a result of a control transfer, the 80386 will change the system stacks. The original SS:ESP values are retained in a special register called the task state segment (TSS). When a jump or call transfer executes, the new stack pointer loads into the SS and ESP registers. The previous pointer is then stored on the new stack. During a return to the original privilege level, the IRET or RET instructions restore the original stack values. If the transfer involves a subroutine, a limited number of words of data are copied from the previous stack to the current one. The number of words copied are in the gate's word count field.

Call Gates

A call gate provides the system with the ability to make protected indirect calls. The major purpose of a gate is to provide a method of making secure privilege transfers within a task. The operating system defines all gates within the system. In this way, the operating system can limit access to procedures through the gates. Gates can only be accessed by more privileged descriptors. In the same way, a gate can only transfer control to a more privileged level.

Call gates are accessed by a CALL instruction. The syntax used for this procedure is the same as subroutine call. When a call gate is activated, the following procedures occur:

1. Load the CS:EIP from the gate and check validity.

2. Push SS value, zero-extended to 32-bits, onto stack.

3. Push ESP value onto stack.

4. Copy the 32-bit word count parameters from the old stack to the new stack.

5. Push the return address onto the stack.

NOTE: The above procedures are the same as on a 80286 based system except for the use of 32-bit values instead of 16-bit values.

Task Switching

Task switching is an essential capability in a multi-tasking/multi-user operating system. This capability allows the operating system to switch between different procedures. The faster the operating system can accomplish this, the better.

The 80386 enhances this level of performance by providing a special task switch instruction within the hardware. When a task switch executes, the 80386 will save the entire machine state, load a new execution state, and perform protection checks. This entire procedure is accomplished in approximately 17 microseconds.

A program can invoke the task switch operation by using an inter-segment JMP or CALL instruction. The instruction must reference a task state segment (TSS) or a task gate descriptor within the GDT or the LDT. The task switch operation can also be invoked by an exception, trap, or external interrupt, provided there is a task gate descriptor in the IDT descriptor slot. Figure 12-10 illustrates the 80386 task state segment format.

A TSS descriptor will point to an entire segment of memory containing the current 80386 execution state. A task gate descriptor, on the other hand, only contains a TSS selector. The TSS for the 80386 must contain a minimum of 64K of space. Any free space in the TSS is available for the operating system to use for the storage of additional information. For the sake of compatibility, the 80386 will also operate with an 80286-style TSS.

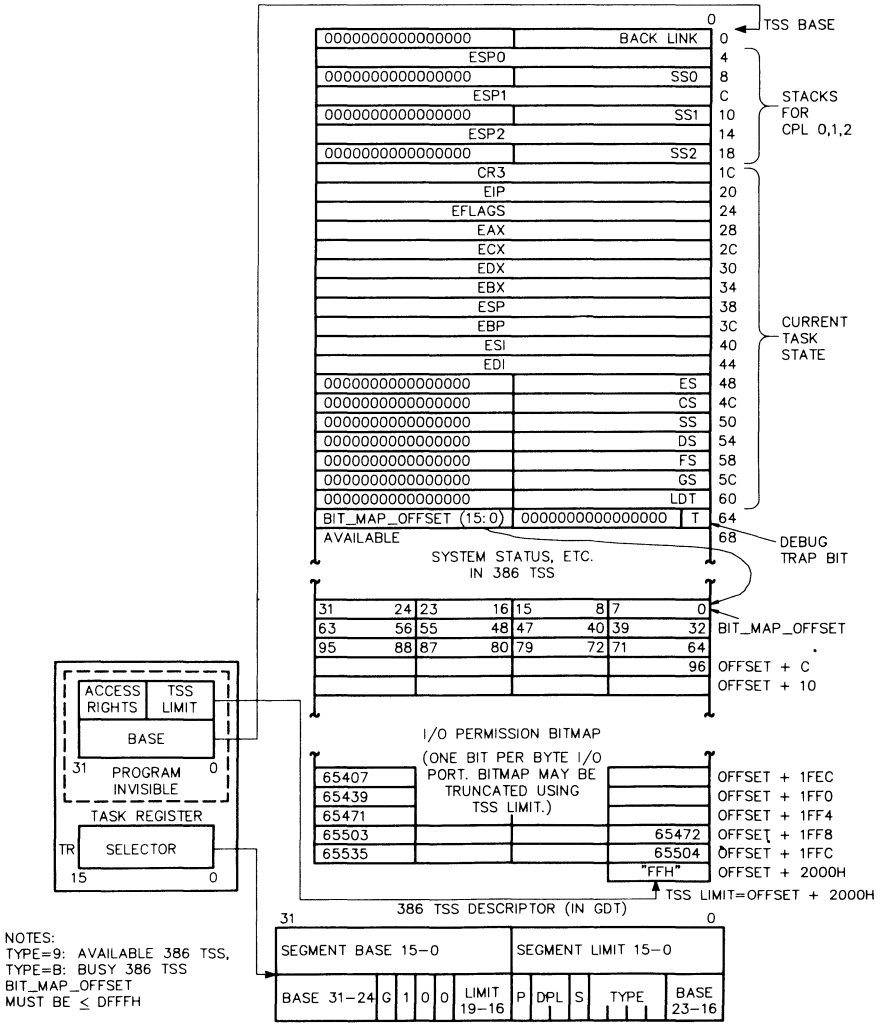


Figure 12-10. 80386 TSS and TSS Registers

The 80386 requires that each task within the system be associated with a TSS. The task state segment register (TR) is a special register in the 80386 that identifies the current TSS. The TR contains a selector which refers to a descriptor that defines the current TSS. Whenever the TR loads a new selector, a hidden associated base and limit register, illustrated in Figure 12-11, is also loaded. The IRET instruction, if properly set, will return control to the interrupted task.

Part of the information the operating system requires about the task state resides in the flag register and the machine status word. Bit 14 in the EFLAGS register (the NT bit) controls the operation of the IRET instruction. If this bit is set to a 1, the IRET instruction will perform a task switch return, restoring control to the proper task. If this bit is set to a 0, IRET will perform a normal return. The machine status register (CR0) contains information dealing with the state of the coprocessor during a task switch.

Setting or resetting the NT bit requires a specific procedure. A CALL or INT instruction initiates the task switch operation. The 80386 will then mark the new TSS as busy and establish a back link field in the new TSS to the old TSS selector. The TSS changes to busy by changing the descriptor field type from 9H to BH. The NT bit in the new TSS is set by CALL- or INT-initiated task switches. If an interrupt does not cause a task switch, it will clear the NT bit. The 80386 restores the NT bit on completion of the interrupt handler execution.

The 80386 does not automatically save the current state of the coprocessor during a task switch. The reason for this is that the requirements of the incoming task are unknown. The incoming task may not even use the coprocessor. The 80386 handles this situation by setting the TS bit in CR0 each time a task switch occurs. The 80386 will then detect the first use of a coprocessor instruction after the task switch. At that point the 80386 will issue a processor extension exception. It is then the responsibility of the exception handler to determine whether or not to save the state of the coprocessor.

Paging

Paging is one additional method of system memory management employed by the 80386. Like the other methods already mentioned, paging has applications in the multi-tasking operating system environment. While segmentation methods tend to modularize programs or data, paging divides a program into multiple, uniformly sized pages. Since most programs exhibit a locality of reference, this is advantageous to the programmer. Because of this, only a small number of active pages from each task need to be active in memory at any given time. Each page frame in the 80386 environment consists of 4096 bytes of memory.

Page Descriptor Base Register — Register CR3 is the page directory physical base address register. This register contains the physical starting address of the page directory. The lower 12 bits all contain zeros to ensure that the page directory remains page aligned. If a program loads this register by using the MOV CR3, reg instruction, the page table entry cache will be flushed.

Page Directory — Each page directory entry is four bytes long. The directory itself contains 4096 bytes, providing enough room for 1024 entries. Each entry contains the address of the next level of tables, the page tables, and related information. Figure 12-11 illustrates the format of a typical page directory entry. Table 12-10 describes the table entries themselves.

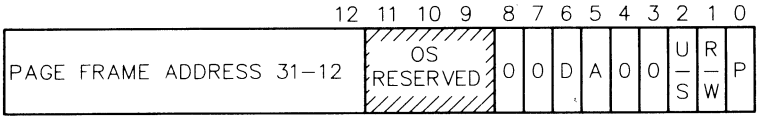


Figure 12-11. Page Directory Entry Format

Table 12-10. Page Directory/Page Table Entries

BIT	NAME	DESCRIPTION
0	P	Present. This bit indicates whether or not an entry can be used for address translation. If the bit is set (1), it can be used, otherwise it cannot. If the entry is not used for address translation, the remaining bits of the entry space become available to the system.
1	R/W	Read/write. This bit, in conjunction with the U/S bit (bit 2), provides a protection attribute for each page entry reference. The setting of the R/W and U/S bits determines the access level permitted for the page. The following table summarizes the permitted page access levels.

U/S	R/W	LEVEL 3	LEVELS 0, 1, OR 2
0	0	None	Read/Write
0	1	None	Read/Write
1	0	Read-Only	Read/Write
1	1	Read/Write	Read/Write

2	U/S	User/supervisor. Refer to the description of bit 1.
3,4	—	Reserved. Do not use these bits.
5	A	Accessed. The 80386 sets this bit prior to read or write accesses to the referenced entry. If a software program modifies this bit, it should use the LOCK prefix to make sure the page retains its integrity.
6	D	Dirty. The 80386 sets this bit prior to a write to the referenced page table entry. This bit is undefined for page directory entries. If a software program modifies this bit, it should use the LOCK prefix to make sure the page retains its integrity.

Table 12-10 (continued). Page Directory/Page Table Entries

BIT	NAME	DESCRIPTION
7,8	—	Reserved. Do not use these bits.
9 - 11	—	OS reserved. These bits are reserved for use by the operating system. One example would be to use this area to record page aging. By knowing how long a page has resided in memory, a least recently used replacement algorithm could be implemented.
12 - 31	—	These bits are where address information for the entry is stored. For a page directory, this area is the page table address. For a page table, the page frame address is available here.

Page Table — The page table is the next level of information below the page directory. The page table also contains 4096 bytes of data space. This provides sufficient room for 1024 table entries since each entry consist of four bytes. The format for the page table entry is identical to the page directory entry format. The page table entry contains the starting address of the page frame and information concerning the page.

Translation Lookaside Buffer

The translation lookaside buffer (TLB) is a hardware implemented, four-way, set associative, 32-entry, page table cache. This feature was added to the 80386 as an aid to support demand paged virtual memory systems. The cache helps to prevent performance degradation by removing the requirement that the 80386 access two levels of tables for each memory reference. Instead, the TLB

maintains information on the most recently accessed page table entries. Since the TLB can hold 32 entries, and each page contains 4K, the TLB can cover 128K of memory addresses. This will normally translate into a 98% hit rate in the average multi-tasking system. The TLB operates in conjunction with the 80386's paging mechanism.

Whenever the paging hardware is active, the segmentation unit passes a 32-bit linear address to the paging unit. The paging unit compares the upper 20 bits with all entries in the TLB. If one entry matches, referred to as a TLB hit, then the 32-bit physical address is calculated and placed on the address bus.

If, on the other hand, the entry is not in the TLB, then the 80386 reads the appropriate page directory entry. If the P bit in the page directory entry is set (1), it indicates that the appropriate page table is in memory. The 80386 then reads the entry and sets the access bit. If the P bit is set (1) in the page table entry, indicating that it is also in memory, then the 80386 will update the access and dirty bits and fetch the operand. The 80386 then stores the upper 20 bits of the linear address in the TLB for future access.

If the P bit is clear (0) in the page table entry or the page directory entry, then the processor generates a page fault. A page fault could also be generated if a memory reference violated the page protection attributes. If a page fault occurs, the CS:EIP register will point to the instruction causing the error and a 16-bit error code will be placed on the stack by the page fault handler. The operating system uses the error code to try to determine how to handle the page fault error.

Only three bits in the error code are used to provide information concerning the error. Table 12-11 describes these bits and the functions that they perform.

Table 12-11. Page Fault Error Code Bits

BIT	NAME	FUNCTION
0	P	Page present. If this bit is set (1), the error was caused by a page protection violation. If the bit is clear, the error resulted from attempted access to a not present page.
1	W/R	Write/read. If the bit is set, the access that caused the error was a write. Otherwise the error resulted from a read access. These bits work in conjunction with the U/S bit.
2	U/S	User/supervisor. If the bit is set, the processor was in user mode when the access fault occurred. If the bit is clear, then it was in supervisor mode. The following table correlates the bit pattern with the function.

U/S	W/R	TYPE OF ACCESS
0	0	Supervisor read
0	1	Supervisor write
1	0	User read
1	1	User write

NOTE: The remaining bits in the 16-bit error code are undefined and should not be used.

Descriptor Attributes

Each descriptor in the 80386 consists of eight bytes. The descriptor contains the attributes which describe the segment in memory. The attributes contained in the descriptor are:

- The 32-bit base linear address of the segment,
- The 20-bit segment length,
- A 1-bit granularity field,
- The segment protection level,
- The segment read, write, and execute privileges,
- The segment default operand size,
- The segment type.

Figure 12-12 illustrates the general layout of the segment descriptor.

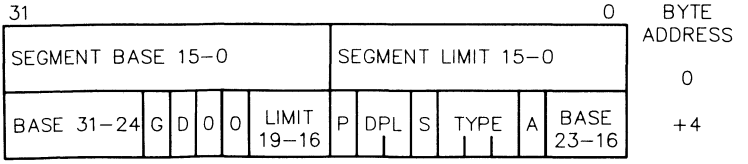


Figure 12-12. Segment Descriptor Format

The first two bytes of the descriptor are the first 16 bits (0 - 15) of the 20-bit segment length. The remaining four bits (16 - 19) are the first four bits of the seventh byte of the descriptor. In a similar fashion the 32-bit segment address information resides within several bytes in the descriptor. The second and third bytes of the descriptor contain the first 16 bits (0 - 15) of the segment address. Bits 16 through 23 of the segment address are in the fifth byte of the descriptor. The remaining eight bits (24 - 31) are in the last byte of the descriptor. Each of the descriptor types within the 80386 vary this general format to meet the requirements of their specific descriptor class.

Whenever a descriptor is accessed by the 80386, the A (access) bit is set. The G bit is the granularity bit. This bit specifies whether a segment is byte-granular or page-granular. If the granularity bit is set (1), the segment is page-granular. The maximum size of a page-granular segment is four gigabytes with each page consisting of 4K. If the bit is clear (0), the segment is byte-granular. The maximum size of this type of segment is one megabyte.

The D bit determines the default length for operands and effective addresses. If this bit is set (1), the default length is 32 bits. If the bit is clear (0), the default length is 16 bits.

The DPL field is the individual descriptor privilege level field. These two bytes describe the descriptor privilege level from 0 - 3. The P bit is the present bit. If this bit is set (1), the segment exists in physical memory. If a program attempts to access a segment that is not present, the processor will generate a "not present" exception.

The 80386 recognizes two categories of segments: system and non-system. A system segment contains information pertaining to the operating system. A non-system segment contains code and data only. The system uses the S bit to determine what type of segment it is dealing with. If the S bit is set (1), then the segment is either a code or a data segment. A cleared bit indicates a system segment.

Code and Data Descriptors

Code and data descriptors follow the same general format as illustrated in Figure 12-12. The primary difference between the two formats is the use of an access rights byte in the code and data descriptors. The access rights byte contains information describing

the segment privilege level, access mode, read and write privileges, and whether or not the segment is code or data. Figure 12-13 is an illustration of the descriptor format for the code and data descriptors. Table 12-12 describes the bit definitions used in the access rights byte of the descriptor.

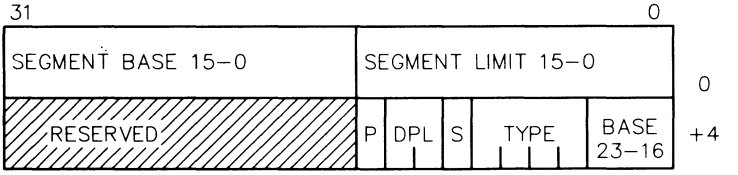


Figure 12-13. Code and Data Descriptor Format

Table 12-12. Access Rights Byte Bit Definitions

BIT	NAME	DESCRIPTION
0	A	Accessed. The system uses this bit to determine whether or not a segment is present in memory. If the bit is set (1), then the segment is present in memory. If the bit is clear (0), the segment has not been accessed.
CODE SEGMENT (see note)		
1	R	Readable. If this bit is set, the code segment may be read by the program. If the bit is clear, it can not.
2	C	Conforming. When this bit is set, the code segment may only be executed when the CPL is greater than or equal to the DPL. The CPL must remain unchanged. Segments that conform may be executed by programs with different privilege levels.
3	E	Executable. The system uses this bit to determine if the segment is a code segment or a data segment. If the bit is clear, it is a code segment.

Table 12-12 (continued). Access Rights Byte Bit Definitions

BIT	NAME	DESCRIPTION
DATA SEGMENT (see note)		
1	W	Writable. If this bit is set, the data segment may be written to. If the bit is clear, it can not.
2	ED	Expansion direction. This bit determines in which direction the segment will expand in memory. If the bit is clear the segment will expand upward in memory and offsets must be less than or equal to the limit. (Upward expansion indicates a data segment.) If the bit is set, the segment will expand downward in memory. (Downward expansion indicates a stack segment) In this case the offsets must be greater than the limit.
3	E	Executable. The system uses this bit to determine if the segment is a code segment or a data segment. If the bit is clear, it is a data segment.
4	S	Segment descriptor. The system uses this bit to determine if the descriptor is code, data, or system. If this bit is set, the descriptor is a code or a data descriptor. If the bit is clear, the descriptor is a system or a gate descriptor.
5 - 6	DPL	Descriptor privilege level. These two bits determine the privilege level of the descriptor.
7	P	Present. This bit determines if the segment is mapped into the physical memory. If the bit is set, the segment exists in memory.

NOTE: These three bits have different functions in the code descriptor and the data descriptor. The remaining bits function the same with either type.

System Segment Descriptors

The 80386 recognizes 12 different types of system segment descriptors. These descriptors provide information about operating system tables, tasks, and gates. Figure 12-14 illustrates the general form of the system segment descriptor. Table 12-13 lists the types of system segments that may appear in the descriptor type block.

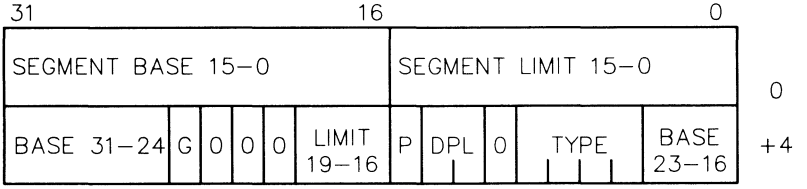


Figure 12-14. System Segment Descriptor Format

Table 12-13. System Segment Type

TYPE	SEGMENT DEFINED AS
0000	Invalid
0001	Available 286 TSS
0010	LDT
0011	Busy 286 TSS
0100	286 call gate
0101	Task gate (286 or 386)
0110	286 interrupt gate
0111	286 trap gate
1000	Invalid
1001	Available 386 TSS
1010	Undefined
1011	Busy 386 TSS
1100	386 call gate
1101	Undefined
1110	386 interrupt gate
1111	386 trap gate

An 80386 system descriptor contains a 32-bit base linear address and a 20-bit segment limit. This differs from the 80286 system descriptor which contains a 24-bit base address and a 16-bit limit. This means that an 80386 descriptor can be easily identified. The upper 16 bits of the descriptor will all be zeros.

The following material describes the different classes of system segment descriptors.

LDT — The LDT descriptor contains information about the local descriptor tables. These descriptors are only allowed in the global descriptor table. The DPL field is ignored due to the fact that the load instruction is only available in privilege level 0.

TSS — This descriptor contains information about the location, size, and privilege level of a task state segment. The TSS itself is a special segment that contains all state information for a task. It also contains a linkage field to permit the nesting of tasks.

Gate — Gates control access to entry points within the target code segment. There are four different types of gates: call, task, trap, and interrupt. This mixture allows the programmer to control the entry points to the operating system. Call gates are used to change privilege levels, task gates to perform a task switch, and interrupt and trap gates to specify interrupt service routines. Gate descriptors use a slightly modified format, as illustrated in Figure 12-15.

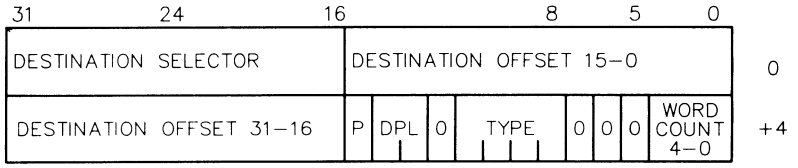


Figure 12-15. Gate Descriptor Format

A program generally uses a call gate to transfer control to a higher privilege level. The descriptor used in this process consists of three distinct fields. The three fields are an access byte, a long pointer (consisting of a destination selector and offset), and a word count. The pointer points to the start of a routine, and the word count indicates how many parameters to copy from the calling program stack to the called routine's stack. The word count field is only used if there is a change of privilege level; otherwise it is ignored.

A task gate only uses the selector portion of the descriptor. Since task gates are only used to switch tasks, they may only refer to the TSS. That is why only the selector is used.

Trap gates handle the selector and offset in a different manner. In a trap gate the selector and offset form a pointer. The pointer indicates the beginning of the interrupt or trap service routines. The main difference between a trap gate and an interrupt gate is that the interrupt gate disables interrupts. The trap gate does not disable interrupts.

To improve processing throughput, a special system descriptor cache is provided within the 80386. Each segment register has an associated segment descriptor cache register. Whenever a segment register's contents change, the 8-byte descriptor associated with the selector is cached. Thereafter, each reference to that segment comes from the cache instead of the descriptor. The contents of the descriptor cache is not accessible to the programmer.

NOTE: The contents of the descriptor cache only changes when a segment register is changed. If a program modifies a descriptor table, it must also reload the appropriate registers after changing the descriptor's value.

Memory Addressing

In protected mode, the 80386 uses a method similar to that used in real mode to determine physical addresses. The system uses a 16-bit selector to determine the linear base address of the segment. The base address is then added to a 32-bit effective address to form a 32-bit linear address. If the paging mode is enabled, the paging unit maps the 32-bit

linear address into the memory space. Otherwise, the 32-bit linear address becomes the 32-bit physical address. The difference between real mode and protected mode is in how the base address is determined. Figure 12-16 illustrates the addressing mechanism used in protected mode.

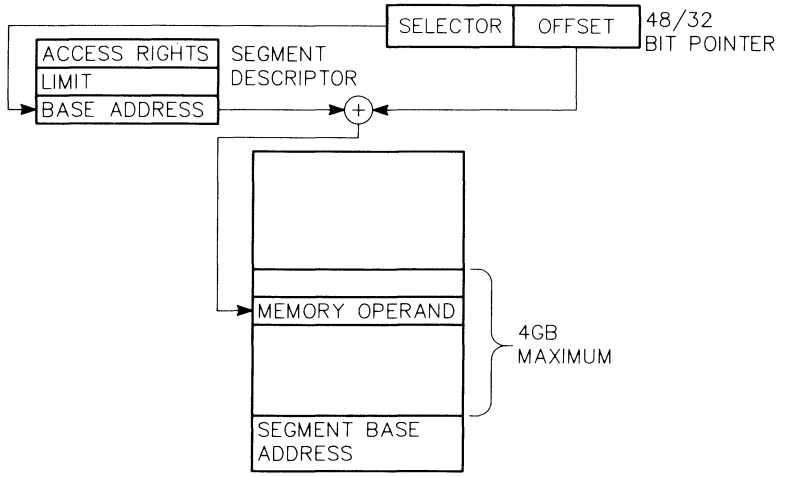


Figure 12-16. Protected Mode Addressing

While in protected mode, the selector specifies an index into a table defined by the operating system. The table contains the 32-bit base address of the desired segment. The physical address is then formed by adding the base address contained in the table to the offset.

Protected Mode Initialization

The 80386 always begins operation immediately after a reset in real mode. Because of this, certain registers must be initialized in order to change into protected mode. Figure 12-17 illustrates a simple protected system while Figure 12-18 shows the descriptors for this system.

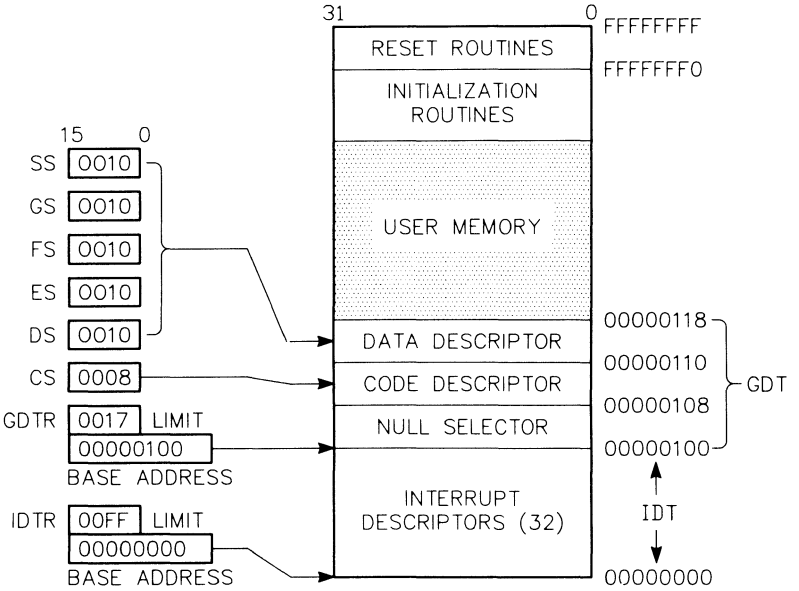


Figure 12-17. Protected System Model

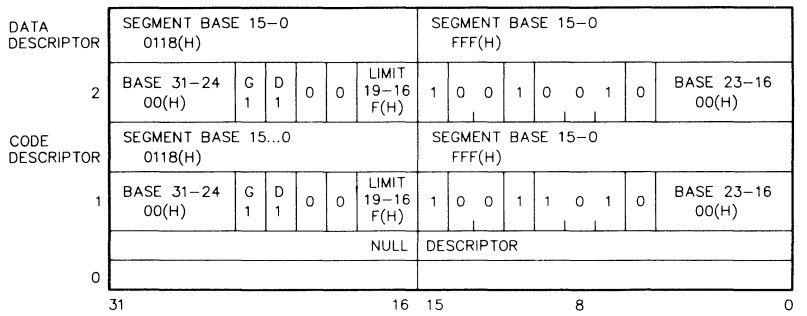


Figure 12-18. GDT Descriptors

The GDT and IDT registers must be initialized prior to transition to protected mode. Both registers must refer to a valid table of the appropriate type. In addition, the IDT must be at least 256 bytes long and the GDT must contain the descriptors for the initial code and data segments. Transition to protected mode occurs using one of two methods.

The first method requires three steps:

1. Load CR0 with the PE bit set, via the MOV CR0, R/M instruction. (This places the processor in protected mode.)

2. Perform an intersegment jump to load the CS register and flush the decode queue.

3. Load all data segment registers with initial selector values.

The second method is more appropriate for multi-tasking operating systems. This method uses the built-in task switch to load all registers. The procedure is as follows:

1. Load CR0 with the PE bit set, via the MOV CR0, R/M instruction. (This places the processor in protected mode.)

2. Perform a jump to the TSS. (This will cause a task switch and load all registers with values stored in the TSS.)

Using this method the GDT should contain two TSS descriptors in addition to the code and data descriptors required for the first task. The TSS register must be initialized to point to a valid TSS descriptor.

Virtual 8086 Mode

Virtual 8086 mode is an extension of the 80386's protected mode environment. In this mode the 80386 permits the execution of 8086-based programs as in real mode, but with more flexibility. While in this mode it is possible to run an 8086 application and operating system, an 80386 operating system, and an 80386 application with its operating system simultaneously. In a multi-user environment, this can provide a great deal of responsiveness to user needs.

Virtual Mode Addressing

In virtual mode, memory address calculations are performed in the same manner as in real mode. The contents of the segment register is added to the offset after first being shifted left four bits. The resulting value is the base linear address.

The 80386 also allows the operating system to determine, on a per task basis, how a program will access memory. Any program can access memory by using the 8086 address mechanism or the protected mode addressing mechanism.

In virtual mode each program is allowed access to a memory area of 1 megabyte. This memory area can be mapped anywhere within the allowable address space of the 80386. In virtual mode, as in real mode, if a program's effective address exceeds 64K, the 80386 will generate an exception.

Paging

It is not required to activate the paging hardware while operating in the virtual mode. However, it is recommended, since paging does provide some protection and operating system isolation. Paging should be used whenever there are multiple virtual mode tasks executing. Paging must be used to relocate the address space of a virtual mode task above the 1 megabyte boundary.

The paging hardware within the 80386 permits a 20-bit linear address, produced by a virtual mode program, to be divided into 256 distinct pages. Each page may be located anywhere within the allowable address space of the 80386. Since the page directory base register (CR3) is loaded by a task switch, a virtual mode task can use different mapping schemes. The paging hardware also allows multiple applications to share 8086 operating system code.

Protection and I/O Permission

Each program running in virtual mode is assigned a privilege level of 3. If a program attempts to execute a privileged instruction while in virtual mode, the processor will generate an exception. This differs from real mode since all programs in real mode execute at a privilege level of zero. The following instructions are considered privileged and may only be executed at privilege level 0:

```
LIDT;       MOV DRn,reg;      MOV reg,DRn
LGDT;       MOV TRn,reg;      MOV reg,TRn
LMSW;       MOV CRn,reg;      MOV reg,CRn
CLTS;
HLT;
```

The following instructions are only available in protected mode:

```
LTR;    LLDT;    LAR;    LSL;
ARPL;   STR;     SLDT;   VERR;
VERW;
```

The following instructions are IOPL-sensitive in protected mode:

```
IN;         OUT;         INS;    OUTS;
REP INS:    REP OUTS;    STI;    CLI;
```

In virtual 8086 mode, the processor uses a slightly different set of IOPL sensitive instructions. These instructions are:

```
INT n;    PUSHF;    POPF;    STI;
CLI;      IRET;
```

Note that several I/O instructions (IN, OUT, INS, OUTS, REP INS, and REP OUTS) are not IOPL-sensitive while in virtual 8086 mode. While in this mode the I/O instructions become sensitive to a structure within the TSS called an I/O permission bit map.

The I/O permission bit map can be thought of as a bit string, ranging in size from 0 - 64 kilobits. The string is located in the current TSS at an offset referred to as bit map offset. The location

of this offset must be such that the bit string does not exceed a value of DFFFH. This way the entire bit map, and the byte which follows it, can be located in a space that is offset less than FFFFH from the TSS base. (Refer to the illustration of the system TSS in Figure 12-10.) Figure 12-19 illustrates an example of a typical I/O permission bit map found in the TSS.

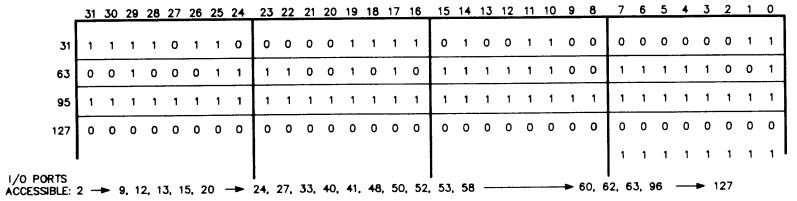


Figure 12-19. I/O Permission Bit Map

Each bit in the I/O permission bit map corresponds with an 8-bit system I/O port. If the bit for the respective port is set to 0, the system will permit I/O with that port. Since each I/O port must be protectable, all bits corresponding to a word or dword port must be set to zero before I/O will be permitted with that port. If they are not, the processor will generate an exception error.

The bit map can generally be located anywhere within the TSS provided the guideline values stated earlier are observed. If the bit map is placed beyond the limit of the TSS segment, in other words ignored, then an additional 8K of memory normally used for the bit map would be freed for the system.

NOTE: The last byte of information in the I/O permission bit map must contain all 1s. This byte must exist within the limit of the defined TSS segment.

Interrupts

Interrupt handling in virtual 8086 mode is more complicated than in the real or protected modes discussed earlier. In virtual 8086 mode all generated interrupts and exceptions require a privilege change back to the 80386 operating system level. The 80386 operating system then handles the interrupt or exception and returns control to the 8086 program.

The 80386 operating system can determine whether the interrupt came from a protected mode or a virtual mode program by examining the VM bit in the EFLAGS register. If a virtual mode program is interrupted, this bit will be cleared (0). However, in either case, the VM bit in the EFLAG image on the stack will remain unchanged.

Using this interrupt method, the 80386 can provide a totally transparent 8086 environment for an 8086 application. It does this by intercepting, and then emulating, 8086 operating system calls and IN and OUT instructions.

Virtual Mode Transitions

There are basically two available methods used by the 80386 to enter virtual 8086 mode. The first method requires that the 80386 already be in protected mode. To enter virtual mode by this method, switch to a task with a 386 TSS that has the VM bit in the EFLAGS image set to 1. To enter using the second method, execute a 32-bit IRET instruction at privilege level 0 with the VM bit set to 1 in the EFLAGS image.

The instruction set of the 80386 makes it appear that other methods of entry are possible, but that is not the case. For example, the POPF instruction does not affect the VM bit even if the processor is already in protected mode or privilege level 0. PUSHF, on the other hand, always pushes a 0 in the VM bit, even if the processor is already in virtual 8086 mode. Once it does that, it is impossible for the program to tell if it is executing in real or virtual mode.

The transition out of virtual 8086 mode returns to the 80386 protected mode. This can occur only when the processor receives an interrupt or an exception while executing in virtual 8086 mode. In virtual 8086 mode all interrupts and exceptions are vectored through the protected mode IDT and end up in an interrupt handler in the protected mode. This means that performing a task switch or an interrupt will exit virtual 8086 mode. During this process the VM bit is cleared.

If an interrupt or a trap gate is used to handle an interrupt or exception from virtual 8086 mode, a matching IRET must be executed at privilege level 0. The gate must perform an inter-level interrupt only to level 0.

The following material provides a general sequence of steps to illustrate the method used to exit virtual 8086 mode via an interrupt and to return to the interrupted program. The first sequence of steps covers the exit procedure to the interrupt handler while the second covers the return to the program.

To exit from virtual 8086 mode via interrupt gate:

1. Save the FLAGS register and turn off the VM and TF bits. If the interrupt is serviced by an interrupt gate, turn off the IF bit.

2. Switch from the virtual stack to the TSS level 0 stack. Save the virtual 8086 mode SS and ESP registers.

3. Push the 8086 register values onto the new stack in the following order: GS, FS, DS, ES. Push these values as 32-bit quantities with the upper 16 bits undefined. Load all four registers with null selectors (0).

4. Push the old 8086 stack pointer onto the new stack (push it as 32-bits with the high 16 bits undefined), and then push the 32-bit EIP register saved earlier.

5. Push the 32-bit flag register saved in step 1.

6. Push the old 8086 instruction pointer onto the new stack. Do this by pushing the CS register (32 bits, high 16 bits undefined), followed by the 32-bit EIP register.

7. Load the new CS:EIP value from the interrupt gate and begin execution of the interrupt routine in protected mode.

To return to the interrupted program:

1. First, check the value of the NT bit in the FLAGS register. If this bit is on, an interrupt task is performed. The current state is stored in the current TSS and the link field in the current TSS is used to locate the TSS for the interrupted task you wish to resume. Otherwise, continue as follows:
2. Read the value of the FLAGS image from SS:8[ESP] into the FLAGS register. (This restores the VM bit.)
3. Pop the instruction pointer CS:EIP. Pop EIP first, and then the 32-bit word containing CS. CS is contained in the lower 16 bits of this word. (If VM=0, this is a protected mode segment load; if VM=1, it is an 8086 segment load.)
4. Increment the ESP register by 4. (This bypasses the FLAGS image from step 1.)
5. If VM=1, load registers ES, DS, FS, and GS respectively from memory locations SS:[ESP+8], SS:[ESP+12], SS:[ESP+16], and SS:[ESP+20]. (These are 8086 segment loads.) Otherwise, check that the ES, DS, FS, and GS selectors are valid in the interrupted routine. Null out any invalid selectors.
6. If RPL(CS) > CPL, pop the stack pointer SS:ESP from the stack. Pop the ESP register first, then the 32-bits containing SS. The lower 16-bits contain the value of SS. (IF VM=1, load as a protected mode segment register load, if VM=0, load as an 8086 segment register.)
7. Resume execution of the interrupted program. (The value of the VM bit determines whether the processor resumes in protected mode, or virtual 8086 mode.)

80386 Pinouts

Figure 12-20 illustrates the locations of the various signal pins on the 80386 processor package. The figure includes both a top and a bottom view of the processor. This pinout is referred to as a pin grid array (PGA) package and is used because of the high signal density of this device.

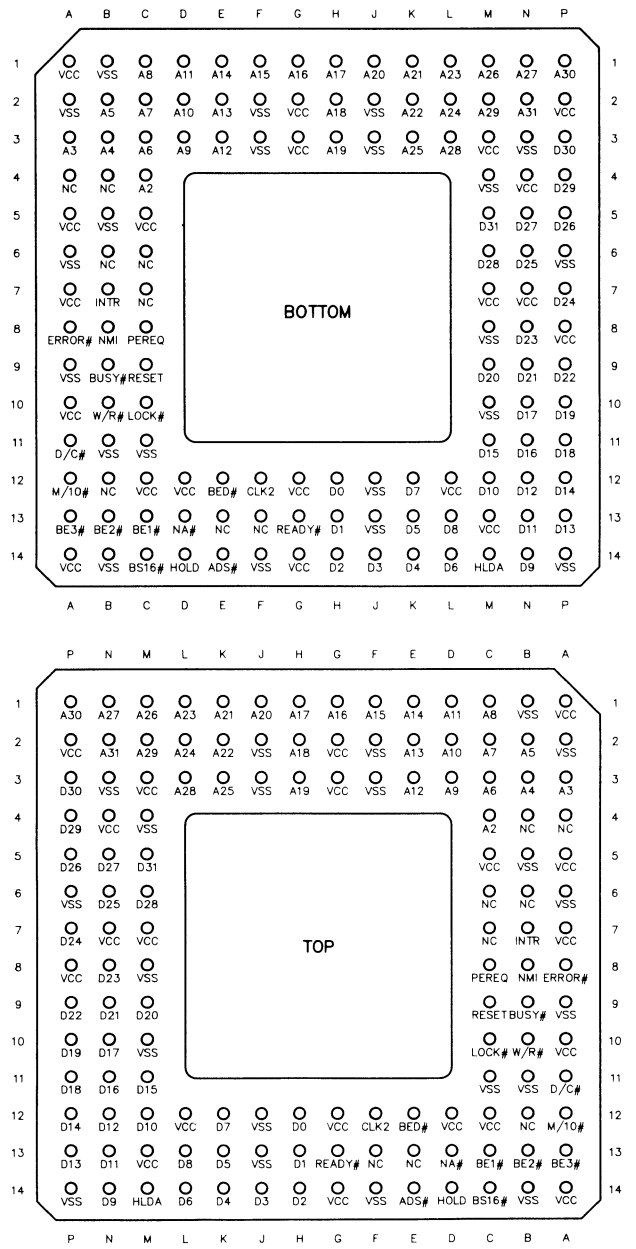


Figure 12-20. 80386 Pin Assignments

Tables 12-14 and 12-15 list the signals available within the 80386. Table 12-14 lists the signals in alphabetical order. Table 12-15, on the other hand, lists the signals by the pin numbers in Figure 12-20. You can locate a specific pin by intersecting the row and the column guide numbers in column and row order.

Table 12-14. Device Pinout Grouped by Function

SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN
A31	N2	D31	M5	RESET	C9	VCC	A1
A30	P1	D30	P3	HOLD	D14	VCC	A5
A29	M2	D29	P4	HLDA	M14	VCC	A7
A28	L3	D28	M6	CLK2	F12	VCC	A10
A27	N1	D27	N5	ADS	E14	VCC	A14
A26	M1	D26	P5	W/R	B10	VCC	C5
A25	K3	D25	N6	D/C	A11	VCC	C12
A24	L2	D24	P7	M/IO	A12	VCC	D12
A23	L1	D23	N8	LOCK	C10	VCC	G2
A22	K2	D22	P9	NA	D13	VCC	G3
A21	K1	D21	N9	BS16	C14	VCC	G12
A20	J1	D20	M9	READY	G13	VCC	G14
A19	H3	D19	P10	INTR	B7	VCC	L12
A18	H2	D18	P11	PEREQ	C8	VCC	M3
A17	H1	D17	N10	BUSY	B9	VCC	M7
A16	G1	D16	N11	ERROR	A8	VCC	M13
A15	F1	D15	M11	NMI	B8	VCC	N4
A14	E1	D14	P12	N.C.	A4	VCC	N7
A13	E2	D13	P13	N.C.	B6	VCC	P2
A12	E3	D12	N12	N.C.	B12	VCC	P8
A11	D1	D11	N13	N.C.	C6	GND	A2
A10	D2	D10	M12	N.C.	C7	GND	A6
A9	D3	D9	N14	N.C.	E13	GND	A9
A8	C1	D8	L13	N.C.	F13	GND	B1
A7	C2	D7	K12	GND	N3	GND	B5
A6	C3	D6	L14	GND	B11	GND	B14
A5	B2	D5	K13	GND	C11	GND	F2
A4	B3	D4	K14	GND	F3	GND	F3
A3	A3	D3	J14	GND	F14	GND	J2
A2	C4	D2	H14	GND	J3	GND	J12
BE3	A13	D1	H13	GND	J13	GND	M4
BE2	B13	D0	H12	GND	M8	GND	M10
BE1	C13	BE0	E12	GND	P6	GND	P14

Table 12-15. Device Pinout Grouped by Connection

PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
A1	VCC	C6	N.C.	H1	A17	M10	GND
A2	GND	C7	N.C.	H2	A18	M11	D15
A3	A3	C8	PEREQ	H3	A19	M12	D10
A4	N.C.	C9	RESET	H12	D0	M13	VCC
A5	VCC	C10	LOCK	H13	D1	M14	HLDA
A6	GND	C11	GND	H14	D2	N1	A27
A7	VCC	C12	VCC	J1	A20	N2	A31
A8	ERROR	C13	BE1	J2	GND	N3	GND
A9	GND	C14	BS16	J3	GND	N4	VCC
A10	VCC	D1	A11	J12	GND	N5	D27
A11	D/C	D2	A10	J13	GND	N6	D25
A12	M/IO	D3	A9	J14	D3	N7	VCC
A13	BE3	D12	VCC	K1	A21	N8	D23
A14	VCC	D13	NA	K2	A22	N9	D21
B1	GND	D14	HOLD	K3	A25	N10	D17
B2	A5	E1	A14	K12	D7	N11	D16
B3	A4	E2	A13	K13	D5	N12	D12
B4	N.C.	E3	A12	K14	D4	N13	D11
B5	GND	E12	BE0	L1	A23	N14	D9
B6	N.C.	E13	N.C.	L2	A24	P1	A30
B7	INTR	E14	ADS	L3	A28	P2	VCC
B8	NMI	F1	A15	L12	VCC	P3	D30
B9	BUSY	F2	GND	L13	D8	P4	D29
B10	W/R	F3	GND	L14	D6	P5	D26
B11	GND	F12	CLK2	M1	A26	P6	GND
B12	N.C.	F13	N.C.	M2	A29	P7	D24
B13	BE2	F14	GND	M3	VCC	P8	VCC
B14	GND	G1	A16	M4	GND	P9	D22
C1	A8	G2	VCC	M5	D31	P10	D19
C2	A7	G3	VCC	M6	D28	P11	D18
C3	A6	G12	VCC	M7	VCC	P12	D14
C4	A2	G13	READY	M8	GND	P13	D13
C5	VCC	G14	VCC	M9	D20	P14	GND

Signal Descriptions

The following material describes the 80386 signal lines referred to in the previous two tables. The signals have been grouped according to function for easier identification.

Bus Signals

Data Bus (D0 - D31) — These 32 signal lines are the three-state bidirectional data bus. This bus serves as the general purpose data path between the 80836 and other system components. Data bus inputs and outputs maintain a set (1) condition when in the high logic state.

Address Bus (A2 - A31, $\overline{BE0}$ - $\overline{BE3}$) — The 30 three-state address lines, along with the four byte enable lines, are the address bus for the 80386. This system is slightly different from the more familiar address organization. The byte enable lines are used to directly indicate which bytes of the 32-bit data bus are involved in the current transfer. This scheme can best be seen in the following list:

- If BE0 is asserted, then D0 - D7 are transferred
- If BE1 is asserted, then D8 - D15 are transferred
- If BE2 is asserted, then D16 - D23 are transferred
- If BE3 is asserted, then D24 - D31 are transferred

The byte enable lines can also be combined to generate the more familiar A0 and A1 signals in this computer. Table 12-16 illustrates this signal generation scheme.

Table 12-16. A0, A1 Address Line Generation

A1	A0	BE3	BE2	BE1	BE0
0	0	x	x	x	0
0	1	x	x	0	1
1	0	x	0	1	1
1	1	0	1	1	1

NOTE: x represents a "don't care" condition

If the current memory write cycle or I/O write cycle transfers an operand that occupies only the upper 16 bits of the data bus, duplicate data is simultaneously presented on the corresponding lower 16 bits of the bus. The data duplication pattern depends on the byte enable lines. Table 12-17 illustrates this process.

Table 12-17. Data Duplication as a Function of the Byte Enable Lines

WRITE DATA DUPLICATION				BYTE ENABLE LINE STATUS			
D0 - D7	D8 - D15	D16 - D23	D24 - D31	BE0	BE1	BE2	BE3
A	—	—	—	0	1	1	1
—	B	—	—	1	0	1	1
C	—	C	—	1	1	0	1
—	D	—	D	1	1	1	0
A	B	—	—	0	0	1	1
—	B	C	—	1	0	0	1
C	D	C	D	1	1	0	0
A	B	C	—	0	0	0	1
—	B	C	D	1	0	0	0
A	B	C	D	0	0	0	0

NOTES:

A = logical write data D0 - D7
B = logical write data D8 - D15
C = logical write data D16 - D23
D = logical write data D24 - D31
1 = high logic level
0 = low logic level
— = undefined condition

Bus Arbitration

Bus Hold Request (HOLD) — When this signal is active, one of the DMA controllers is requesting bus control. As long as the DMA controller retains bus control this signal will remain high. If RESET is asserted while HOLD is asserted, RESET has priority and will place the bus in an idle state.

Bus Hold Acknowledge ($\overline{\text{HLDA}}$) — When this signal is active, the 80386 has acknowledged the HOLD request and released control of the bus. When $\overline{\text{HLDA}}$ is asserted, the 80386 places all output or bidirectional signals (D0 - D31, A2 - A31, BE0 - BE3, W/R, D/C, M/IO, LOCK, and ADS) in a high-impedance state.

Bus Cycle Definition

The three primary bus cycle definition signals (W/R, D/C, and M/IO) are valid when the ADS signal is asserted. The relationship of these signals to the bus cycle definition is shown in Table 12-18.

Write/Read (W/$\overline{\text{R}}$) — This signal defines the read or write cycle.

Data/Control (D/$\overline{\text{C}}$) — This signal defines a data or control cycle.

Memory/IO (M/$\overline{\text{IO}}$) — This signal defines memory and I/O cycles.

Table 12-18. Bus Cycle Definition

M/IO	D/C	W/R	BUS CYCLE TYPE
0	0	0	Interrupt acknowledge
0	0	1	Does not occur (see note)
0	1	0	I/O data read
0	1	1	I/O data write
1	0	0	Memory code read

Table 12-18 (continued). Bus Cycle Definition

M/IO	D/C	W/R	BUS CYCLE TYPE
1	0	1	Shutdown: Halt:
1	1	0	Memory data read
1	1	1	Memory data write

NOTE: This condition will occur during idle bus states when ADS is not asserted.

Bus Control

Address Status ($\overline{ADS}$) — An active condition on this signal indicates that a valid bus cycle definition and address is present on the 80386 pins.

Transfer Acknowledge ($\overline{READY}$) — This signal indicates that the current bus cycle is complete and active bytes indicated by BE0 - BE3 and BS16 are accepted or provided. This signal is ignored on the first bus state of all bus cycles. Thereafter the signal is sampled each bus state until asserted.

Next Address Request ($\overline{NA}$) — When this signal is asserted, the 80386 is ready to accept new data from the address and control lines even if the current cycle is not complete. This signal requests address pipelining.

Bus Size ($\overline{BS16}$) — When this signal is active the 80386 will only use the low-order half (D0 - D15) of the data bus. If the operand is larger than 16 bits, the 80386 will automatically perform another 16-bit bus cycle. If BE2 or BE3 are asserted during a bus cycle and BS16 is also asserted, then the 80386 will adjust to correctly transfer data to the full 32-bit bus.

Coprocessor Interface

Coprocessor Request (PEREQ) — When this signal is asserted, the coprocessor is requesting that the 80386 transfer a data operand to or from memory.

Coprocessor Busy ($\overline{\text{BUSY}}$) — If this signal is active, the coprocessor is still executing an instruction. FNINIT and FNCLEX can execute even if BUSY is active. These instructions are used by the coprocessor for initialization and exception clearing. If BUSY is sampled low at the falling edge of reset, then the 80386 will perform an internal self-test.

Coprocessor Error ($\overline{\text{ERROR}}$) — When active, this signal indicates that the previous coprocessor instruction generated a coprocessor error that is not masked by the coprocessor's control register. The 80386 will generate an exception 7 to access error handling software when it encounters this condition. This signal may also be used to determine if an 80387 is present in the system. If ERROR is low no later than 20 CLK2 periods after the falling edge of RESET and remains low until the 80386 begins its first bus cycle, then an 80387 is presumed to be present.

Interrupt Signals

Maskable Interrupt Request (INTR) — If active, this signal indicates a request for interrupt service which can be masked by the 80386 IF bit. When the 80386 responds to this request, it will perform two interrupt acknowledge bus cycles. At the end of the second cycle the 80386 will place an 8-bit vector on the D0 - D7 data lines to indicate the source of the interrupt.

Nonmaskable Interrupt Request (NMI) — This signal indicates an interrupt service request which cannot be masked by software. The 80386 does not use interrupt acknowledge signals when processing an NMI. Once processing on an NMI has started, no further NMIs will be processed until the next IRET instruction.

Miscellaneous

Clock (CLK2) — This signal provides the basic timing for the 80386. The 80386 divides this signal internally to generate the internal processor clock used for instruction execution. Figure 12-21 illustrates the timing of these signals.

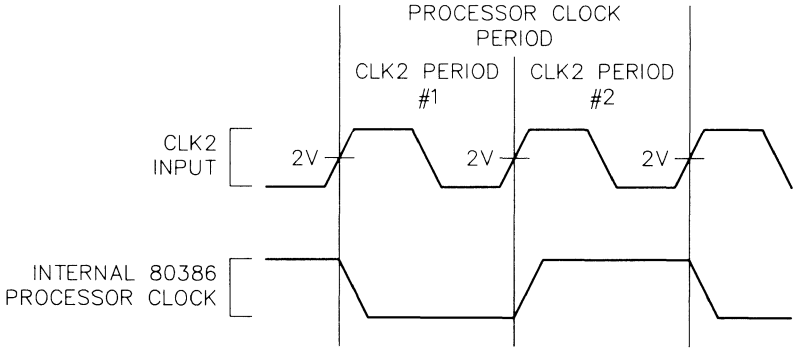


Figure 12-21. Processor Timing Relationships

Reset (RESET) — This signal suspends all processor operations and places the 80386 in a known reset state. When this signal is active, all other input pins are ignored and all other bus pins are driven to an idle state.

Chapter 13
Coprocessors

This chapter provides information on programmer accessible features of the Intel 80387 numeric processor extension (coprocessor). If you require more detailed information concerning this device, refer to the manufacturer's data sheets. (Refer to Chapter 1 for a list of related publications.)

The 80387 is often referred to as a coprocessor. This device extends and enhances the performance of the system CPU. This is accomplished by allowing the coprocessor to essentially operate in parallel with the CPU. The coprocessor provides additional instructions and performs dedicated mathematical processing tasks.

80387

The installation of the 80387 coprocessor (model Z-516) may require alteration of the system configuration. Refer to Chapter 4 for details on system configuration.

The 80387 serves as a complimentary device for 80386-based computer systems. It operates as an extension of the 80386 CPU, providing it with additional instructions and features. Some of the features of this device include:

- 80-bit internal architecture
- Object code compatibility with 8087 and 80287
- Support for multiple data formats (32, 64, and 80-bit floating point; 32, and 64-bit integers; and 18-digit BCD data)
- Support for the ANSI/IEEE 754-1985 Binary Floating-Point Standard

Architecture

Figure 13-1 is a block diagram of the 80387 coprocessor. Refer to this illustration while reading the following material.

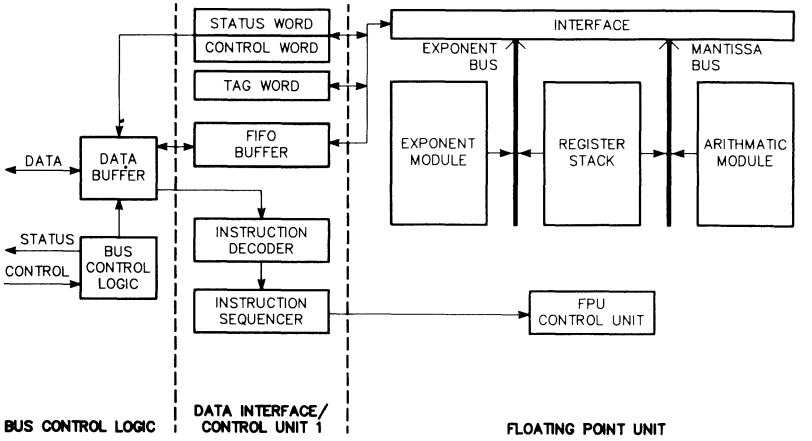


Figure 13-1. 80387 Block Diagram

Internally, the 80387 consists of three major circuit groups. The three major groups are the bus control logic unit, the data interface/control unit and the floating point unit. These groups combined with some functions within the 80386, perform the dedicated math processing functions. In order to enhance performance, the three groups operate in parallel.

Bus Control Logic Unit

The bus control logic unit interfaces the 80387 with the 32-bit data bus and controls the operation of that interface. As far as the CPU is concerned, the 80387 appears as a special peripheral device. Whenever the CPU encounters a coprocessor command, it automatically initiates I/O with the 80387 using reserved I/O addresses.

The bus control logic unit communicates only with the CPU. The CPU, in turn, handles all memory accesses required by the 80387, transferring data to and from the coprocessor.

Data Interface/Control Unit

The data interface/control unit controls the flow of data to and from the floating point unit and control registers. It also decodes instructions, sequences internal microinstructions, and performs some internal administrative functions.

The data interface/control unit reads data from the bus control logic area. Under the control of the bus control logic unit, this data is sent to the instruction decoder or an internal FIFO buffer. The instruction decoder controls the flow of data in the FIFO buffer as well as the microinstructions that execute each instruction. Some 80387 instructions can execute independently of the floating point unit and the microinstruction sequencer. The instructions in this group are FINIT, FCLEX, FSTSW, FSTSW AX and FSTCW (refer to Appendix C for information on the 80387 instruction set). This unit also generates the $\overline{\text{BUSY}}$, PEREQ, and $\overline{\text{ERROR}}$ signals to synchronize operations with the CPU.

The data interface/control unit uses the status word register to report the status of the coprocessor after each operation. The status word is a 16-bit word, as illustrated in Figure 13-2. Table 13-1 describes the bit definitions for this word.

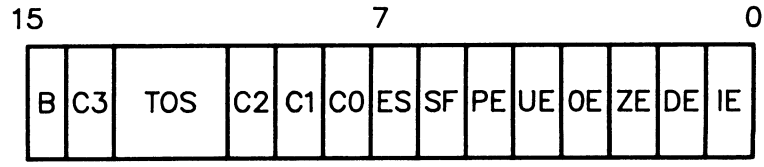


Figure 13-2. 80387 Status Word Format

Table 13-1. 80387 Status Word Bit Definitions

BIT	NAME	DESCRIPTION
0	IE	Invalid operation — stack overflow or underflow, indeterminate form, or use of a non-number as an operand.
1	DE	Denormalized operand — an operand with the smallest exponent, but a non-zero significand.
2	ZE	Zero divide — the dividend is a non-infinite, non-zero number while the divisor is zero.
3	OE	Overflow — the magnitude of the operation result is too large for the specified format.
4	UE	Underflow — the operation result is non-zero but too small to fit the specified format.
5	PE	Precision —it is not possible to correctly represent the result in the specified format. The result is rounded.
6	SF	Stack flag — This bit helps to distinguish between invalid operations involving the stack. When this bit is set, an invalid operation has occurred. If bit 9 is set, the operation was a stack overflow. If bit 9 is clear, the operation was a stack underflow.
7	ES	Error summary status — this bit is set if any unmasked exception bit is set. If this bit is set, the ERROR signal is active.
8 - 10, 14	C0 - C2 C3	Condition codes — report the condition of the operation. Refer to Table 13-2 for the interpretation of these codes.

Table 13-1 (continued). 80387 Status Word Bit Definitions

BIT	NAME	DESCRIPTION
11 - 13	TOS	Top of stack — these three bits define the current top of stack according to the following arrangement: 000 = Register 0 current TOS 001 = Register 1 current TOS 010 = Register 2 current TOS 011 = Register 3 current TOS 100 = Register 4 current TOS 101 = Register 5 current TOS 110 = Register 6 current TOS 111 = Register 7 current TOS
15	B	Busy bit — this bit is provided only for compatibility with the 8087. The condition of this bit reflects the contents of the ES bit.

NOTE: If bits 0 - 5 are set (1), then the condition is true.

Table 13-2. 80387 Condition Codes

INSTRUCTION TYPE	C3	C2	C1	C0	Interpretation
Compare	0	0	x	0	TOS > operand
	0	0	x	1	TOS < operand
	1	0	x	0	TOS = operand
	1	1	x	1	Unordered
Remainder	Q1	0	Q0	Q2	Complete reduction (See Table 13-7)
	U	1	U	U	Incomplete reduction
Operand Class Definitions	0	0	0	0	+Unsupported
	0	0	0	1	+NAN
	0	0	1	0	-Unsupported
	0	0	1	1	-NAN
	0	1	0	0	+Normal
	0	1	0	1	+Infinity
	0	1	1	0	-Normal
	0	1	1	1	-Infinity
	1	0	0	0	+0
	1	0	0	1	+Empty
	1	0	1	0	-0
	1	0	1	1	-Empty
	1	1	0	0	+Denormal
	1	1	1	0	-Denormal

NOTES:

x = unaffected value
U = undefined value
TOS = top of stack
Qx = quotient bit x
NAN = not a number

The NAN listed in Table 13-2 is a special condition that occurs in the 80387 coprocessor. Any value that contains a string of ones in the exponent, except infinity, is considered a NAN. This situation tends to propagate through arithmetic operations. The advantage to NAN values is that they can be handled as special situations by the software. This allows the programmer to test for special conditions or results with the coprocessor.

Complete reductions, as listed in Table 13-2, relate to a specific MOD value. Table 13-3 provides the relation between the condition codes, quotient bits, and the MOD value.

Table 13-3. 80387 Quotient Results

CONDITION CODE QUOTIENT BIT	C3	C2 Q1	C1 Q0	C0 Q2	VALUE Q MOD 8
	0	0	0	0	0
	0	0	1	0	1
	0	1	0	0	2
	0	1	1	0	3
	0	0	0	1	4
	0	0	1	1	5
	0	1	0	1	6
	0	1	1	1	7

The control word determines which of several different processing options the coprocessor will use. The 16-bit control word is normally loaded from memory to set specific coprocessor options. The five low-order bits (0 - 5) of the word contains individual masks for each of the six exceptions that the 80387 recognizes. If an exception is encountered during processing and the mask is in place, processing will continue as normal. If the exception is not masked, then the 80387 will generate an error condition. The eight high-order bits of the word contain operating parameter controls such as precision,

such as precision and rounding. In the 80387, only affine closure is supported in infinity arithmetic. Bit 12 initializes to a zero after a RESET or an FINIT instruction and can only be changed by loading the control word. Since this bit is defined as reserved, software should ignore this bit. Table 13-4 describes the bit definitions for the control word and Figure 13-3 represents the format of the control word.

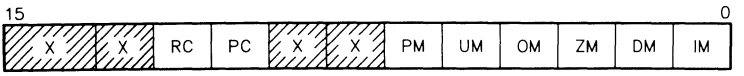


Figure 13-3. 80287 Control Word Format.

Table 13-4. 80287 Control Word Bit Definitions

BIT	NAME	DESCRIPTION
0	IM	Invalid operation exception mask. When the bit is set, the exception is masked.
1	DM	Denormalized operand exception mask. When the bit is set, the exception is masked
2	ZM	Zero divide exception mask. When this bit is set, the exception is masked.
3	OM	Overflow exception mask. When the bit is set, the exception is masked.
4	UM	Underflow exception mask. When the bit is set, the exception is masked.
5	PM	Precision exception mask. When the bit is set, the exception is masked.
6, 7	—	Reserved. Do not use.
8, 9	PC	Precision control. These bits determine the precision of processor calculations according to the following settings: 00 = 24 bits (short real) 01 = reserved 10 = 53 bits (long real) 11 = 64 bits (temp real)

Table 13-4 (continued). 80287 Control Word Bit Definitions

BIT	NAME	DESCRIPTION
10, 11	RC	Rounding control. These bits determine the mode used for rounding operations during calculations according to the following settings: 00 = round to nearest or even 01 = round down 10 = round up 11 = chop (truncate toward zero)
12 - 15	—	Reserved. Do not use.

Floating Point Unit

The floating point unit executes all numeric instructions. Numeric instructions are those instructions that involve the processor stack. They include arithmetic, logical, transcendental, constant, and data transfer instructions. The internal data path in the floating point unit is 84 bits wide. The data path is divided into three sections: 68 bits are the significant bits, 15 bits are exponential, and one bit in the path is the sign bit.

Register Resources

The 80387 employs a stack consisting of eight data registers. Each register is 80 bits wide and contains three fields. The lower 64 bits of each register comprise the significant data, the next 14 bits are the exponent data, and the highest bit is the sign bit. The instruction

pointer and the data pointer registers are located within the 80386 CPU and are 48 bits wide. Figure 13-4 illustrates the register set used by the 80387.

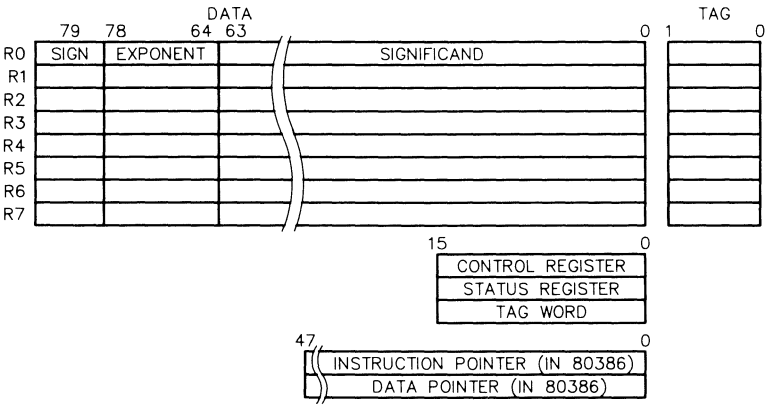


Figure 13-4. 80387 Registers

Each register has a two-bit tag associated with it. All eight tags are contained in the tag word register. The two lowest-order bits are the tag for register 0 and the two highest-order bits are the tag for register 7. There are four possible tag values, as follows:

- 00 — (valid tag)
- 01 — (zero tag)
- 10 — (invalid tag or infinity)
- 11 — (empty tag)

These tag values help to determine the contents of the register.

The 80387 supports both real and protected mode operations in 16- and 32-bit formats. Because of this, the interpretation of the various registers may change based on the current operating mode. Figure 13-5 illustrates the differences in 32-bit format while Figure 13-6 illustrates the differences in 16-bit format.

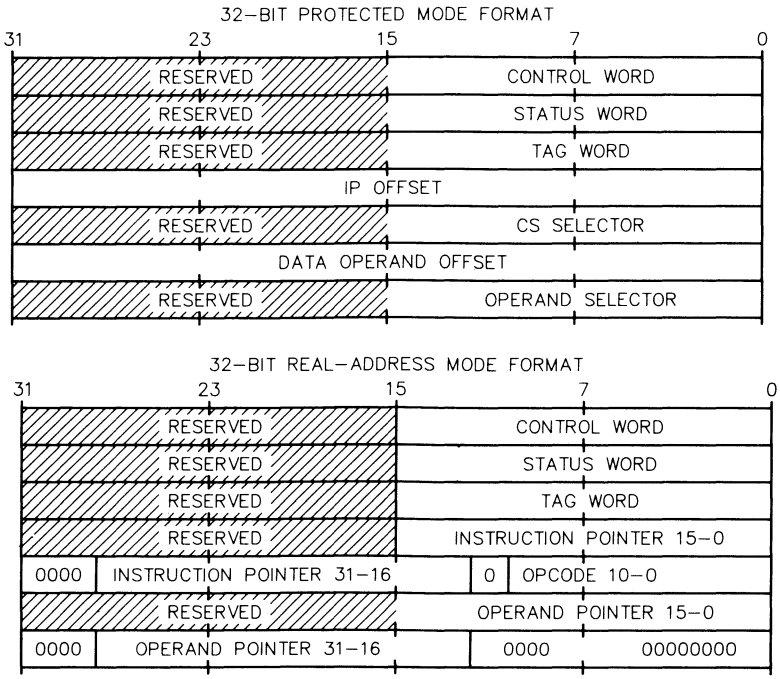


Figure 13-5. 32-Bit Format Register Differences

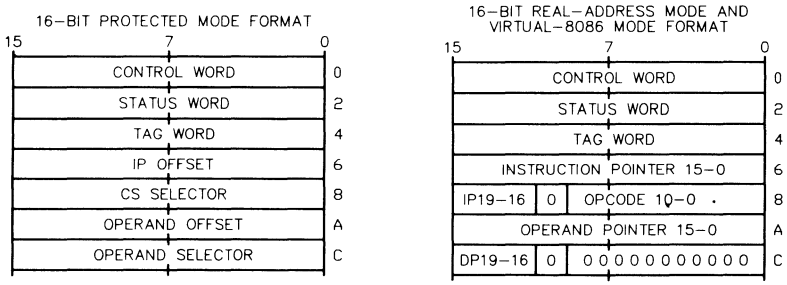


Figure 13-6. 16-Bit Format Register Differences

Operation

In operation, the 80387 operates in parallel with the 80386. Essentially, the 80386 looks at the coprocessor as an extension of itself. In both real and virtual 8086 mode, the 80387 is upward compatible with the 8087 and the 80287. In the protected mode the 80387 is upward compatible with the 80287.

Whenever the 80386 CPU encounters a coprocessor command, it communicates with the bus control logic of the 80387. The 80387 then latches the instruction into the data buffer so that the data interface/control unit can access it. The data interface/control unit passes the data to the FIFO buffer or to the instruction decoder, depending on the instructions received from the bus control unit. When the microinstruction sequencer starts to execute the instruction, the 80387 $\overline{\text{BUSY}}$ line goes active. If the 80387 encounters a processing exception during execution, the $\overline{\text{ERROR}}$ line will go active.

All computations, with the exception of those handled by the data control/interface unit, are performed by the floating point unit. The results of these calculations are routed back to the FIFO buffer so that they can be placed back on the system data bus.

Programming

Programming the 80387 is a very simple process. All device and control functions are accessible through the instruction set. Since the 80386 controls the 80387, all communications between the two devices will be transparent to applications software. The device can be utilized by high-level and assembly language routines.

All operands are stored in memory with the least-significant digit at the lowest memory address. To increase system performance, all operands should start at doubleword boundaries. A doubleword boundary is a memory address that is evenly divisible by four. This allows the processor to utilize the full 32-bit data path while moving data.

Device Pinout

Figure 13-7 illustrates the pinout of the 80387 coprocessor. Table 13-5 is a cross-reference of the device pinout by signal name, and Table 13-6 provides the same cross-reference by pin number.

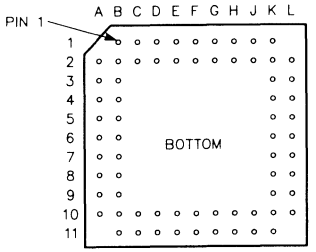


Figure 13-7. 80387 Device Pinout

Table 13-5. 80387 Pinout by Function

FUNCTION	PIN NAME	FUNCTION	PIN NAME	FUNCTION	PIN NAME
D0	H2	D23	C10	387CLK2	K11
D1	H1	D24	D10	Vcc	A6
D2	G2	D25	D11	Vcc	A9
D3	G1	D26	E10	Vcc	B4
D4	D2	D27	E11	Vcc	E1
D5	D1	D28	G10	Vcc	F1
D6	C2	D29	G11	Vcc	F10
D7	C1	D30	H10	Vcc	J2
D8	B1	D31	H11	Vcc	K5
D9	A2	$\overline{\text{ADS}}$	K7	Vcc	L7
D10	B3	$\overline{\text{BUSY}}$	K2	GND	B2
D11	A3	$\overline{\text{CKM}}$	J11	GND	B7
D12	A4	$\overline{\text{CMD0}}$	L8	GND	C11
D13	B5	$\overline{\text{ERROR}}$	L2	GND	E2
D14	A5	$\overline{\text{NPS1}}$	L6	GND	F2
D15	B6	$\overline{\text{NPS2}}$	K6	GND	F11
D16	A7	$\overline{\text{PEREQ}}$	K1	GND	J1
D17	B8	$\overline{\text{READY}}$	K8	GND	J10

Table 13-5 (continued). 80387 Pinout by Function

FUNCTION	PIN NAME	FUNCTION	PIN NAME	FUNCTION	PIN NAME
D18	A8	$\overline{\text{READY0}}$	L3	GND	L5
D19	B9	RESETIN	L10	N.C.	K9
D20	B10	STEN	L4	Tie High	K3
D21	A10	$\overline{\text{W/R}}$	K5	Tie High	L9
D22	B11	386CLK2	K10		

Table 13-9. 80387 Pinout by Pin Name

PIN NAME	FUNCTION	PIN NAME	FUNCTION	PIN NAME	FUNCTION
A2	D9	C11	GND	J10	GND
A3	D11	D1	D5	J11	CKM
A4	D12	D2	D4	K1	PEREQ
A5	D14	D10	D24	K2	$\overline{\text{BUSY}}$
A6	Vcc	D11	D25	K3	Tie High
A7	D16	E1	Vcc	K4	W/R
A8	D18	E2	GND	K5	Vcc
A9	Vcc	E10	D26	K6	NPS2
A10	D21	E11	D27	K7	$\overline{\text{ADS}}$
B1	D8	F1	Vcc	K8	$\overline{\text{READY}}$
B2	GND	F2	GND	K9	N.C.
B3	D10	F10	Vcc	K10	386CLK2
B4	Vcc	F11	GND	K11	387CLK2
B5	D13	G1	D3	L2	$\overline{\text{ERROR}}$
B6	D15	G2	D2	L3	$\overline{\text{READY0}}$
B7	GND	G10	D28	L4	STEN
B8	D17	G11	D29	L5	GND
B9	D19	H1	D1	L6	$\overline{\text{NSP1}}$
B10	D20	H2	D0	L7	Vcc
B11	D22	H10	D30	L8	$\overline{\text{CMD0}}$
C1	D7	H11	D31	L9	Tie High
C2	D6	J1	GND	L10	RESETIN
C10	D23	J2	Vcc		

80387 Pin Descriptions

This section is organized according to the signal groups used in the 80387. These groups are control signals, handshake signals, bus interface signals, chip select signals, and power connections.

Control Signals

The control signals are responsible for all execution control in the 80387.

386CLK2 — 80386 clock 2.. This signal clocks the floating point unit and the data interface/control unit. It is divided by two internally to form the internal clock.

CKM — Clocking mode. This pin determines which clocking mode the 80387 will use. It is set to run in synchronous mode in this computer.

RESETIN — System reset. When the signal level on this pin changes from low to high, the 80387 will terminate all present operations and revert to a dormant state.

Handshake Signals

The handshake signals are responsible for reporting the status of the 80387.

PEREQ — Processor extension request. Whenever the 80387 is ready to transfer data to or from the FIFO buffer, this line will go active. This line is normally connected directly to the 80386 CPU.

$\overline{\text{BUSY}}$ — Busy status. Whenever the 80387 is executing an instruction, this line will be active.

$\overline{\text{ERROR}}$ — Error status. If the 80387 encounters an unmasked exception during execution, this line will go active. This line reflects the state of the ES bit in the status register. Immediately following a system reset, the ES bit is read to determine if an 80387 is present in the system.

Bus Interface Signals

The bus interface signals control reading and writing information to and from the system bus.

D0 - D31 — Bidirectional data bus. All data going to and from the 80387 uses this bus. These lines are normally connected directly to the data bus for the 80386.

W/$\overline{R}$ — Write/read cycle. This line lets the 80387 know what type of bus cycle is currently in progress. If STEN, NSP1, or NSP2 are inactive, this signal will be ignored by the 80387.

$\overline{ADS}$ — Address strobe. This signal is used with the READY signal by the bus control logic to determine when the 80387 can sample the W/R and chip select signals.

$\overline{READY}$ — Bus ready. Whenever the 80386 is about to terminate a bus cycle, this signal will become active. The internal bus control logic uses this signal follow activity on the bus.

$\overline{READY0}$ — Ready output. When this signal is active, it indicates that the current 80387 bus cycle may be terminated if no additional wait states are required.

Chip Select Signals

The chip select signals allow the system to select the 80387 for operations.

STEN — Status enable. This line is essentially the chip select line for the 80387. This signal is permanently enabled in this computer.

$\overline{\text{NSP1}}$ — 80387 select #1. When this line is active along with STEN and NSP2, it indicates that the current bus cycle is communicating with the 80387. This line is selected when the 80386 performs I/O cycles. It is tied to the M/$\overline{\text{IO}}$ status line from the 80386.

$\overline{\text{NSP2}}$ — 80387 select #2. When this line is active along with STEN and NSP1, it indicates that the current bus cycle is communicating with the 80387 and distinguishes 80387 I/O cycles from other I/O cycles. It is connected to address bit 31 from the 80386.

$\overline{\text{CMD0}}$ — Command. This signal indicates whether the 80387 has received an opcode or data. If CMD0 is active, the command was an opcode for the 80387. If CMD0 is inactive, the information is data for the 80387. It is tied to address bit 2 from the 80386.

Power Signals

The final group of signals are those that provide operating power for the 80387.

Vcc — +5 VDC supply. This is the operating DC voltage for the 80387.

GND — Ground. All signals with this designation connect to system ground.

Chapter 14

Support Circuits

This chapter describes the major user-programmable support circuits in this computer. Support circuits remove some of the workload from the system CPU by maintaining critical system operations without requiring direct CPU attention.

The 82C206 Integrated Peripherals Controller

The 82C206 integrated peripherals controller (IPC) incorporates two 8237 DMA controllers, two 8259 interrupt controllers, one 8254 interval timer, one MC146818 real time clock, and several other interface logic chips all on a single integrated circuit. Figure 14-1 is a block diagram of the 82C206 IPC.

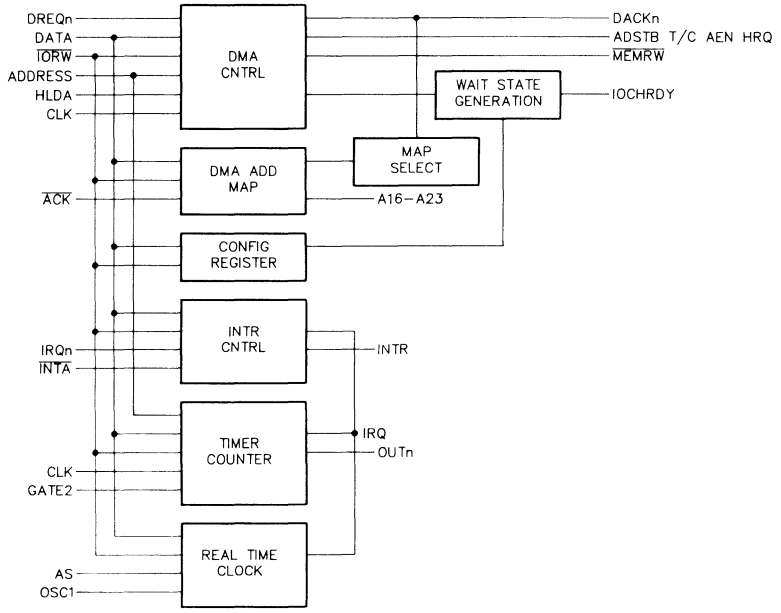


Figure 14-1. 82C206 Integrated Peripherals Controller Block Diagram

The Interrupt Controllers

The interrupt controllers in the 82C206 accept requests from peripherals, resolve priority on pending interrupts, and issue an interrupt request to the CPU. These include interrupts from the timer, keyboard, real-time clock, coprocessor, and input/output ports.

The interrupt controllers in the 82C206 IPC are compatible with the 8259 interrupt controller. Each controller can manage up to eight separate request lines. Since the interrupt controllers are designed in cascade mode, the slave controller uses one line as an input to the master controller. Two of the remaining interrupt request lines are connected internally to various devices, providing a total of 13 separate user definable interrupt channels.

The interrupt controller is treated as an input/output device for programming purposes. Interrupt controller #1 (master) has system address 020H - 021H. Interrupt controller #2 (slave) has system address 0A0H - 0A1H.

Interrupt Controller Operation

Refer to Figure 14-2 for a block diagram of the major circuit groups in the interrupt controller.

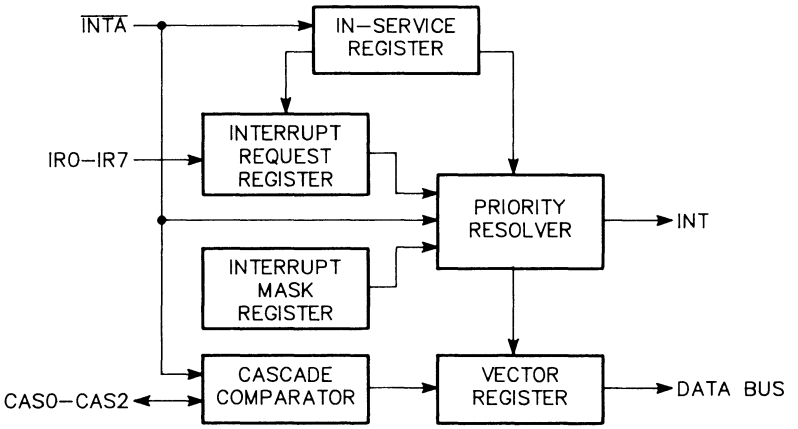


Figure 14-2. Interrupt Controller Block Diagram

The interrupt request register receives all the incoming interrupt requests and stores them until the CPU can process them. The interrupt request line must go inactive before the register will recognize another interrupt request. If the in-service register already holds the request, the interrupt request register will clear (which also clears the in-service register) and will recognize a subsequent interrupt request on the same line. This action prevents an interrupt request from going through the system a second time.

The in-service register is one of four registers in the interrupt controller. This register stores all interrupt requests acknowledged by the CPU that are currently being serviced.

The interrupt mask register operates in conjunction with the priority resolver to mask out incoming interrupt requests. Masking one priority level will not affect the operation of higher or lower priority interrupts. This computer uses two types of interrupts: maskable and non-maskable. The interrupt controller handles the maskable interrupts while non-maskable interrupts go directly to the CPU. The non-maskable interrupt is reserved for serious error conditions and for circuits that have critical timing requirements.

The priority resolver evaluates inputs from the interrupt request register, the in-service register, and the interrupt mask register, issues an interrupt request, and latches the corresponding bit into the in-service register. The priority resolver first examines the data in the in-service register to see if an incoming interrupt request is already being serviced by the CPU.

The cascade buffer/comparator compares the code that is transmitted from the master controller to the slave during the interrupt acknowledge cycle and the code that is already in the slave controller. If a match occurs in the slave controller, it will generate an interrupt vector.

The contents of the vector register provides the CPU with an interrupt vector during interrupt acknowledge cycles.

Interrupt Sequence

Once the interrupt controller has been programmed, the CPU must be instructed to set a bit in its flag register to enable interrupts. If one or more circuits generate an interrupt, the system will respond as follows:

1. One or more of the interrupt request lines, IRQ0-IRQ7, will go high.

2. The controller will check the priorities of the incoming interrupts and compare them to others that may also be waiting to be serviced.

3. The controller will assert the INTRQ line to the CPU.

4. The CPU recognizes the interrupt request, completes its current instruction, and places the interrupt acknowledge code on the status bus.

5. The interrupt controller will resolve the priority of the interrupts that are requesting service and prepare the vector address value of the selected interrupt line for placement on the data bus during the next interrupt acknowledge cycle.

6. The CPU generates the second interrupt acknowledge signal.

7. Upon receipt of the second interrupt acknowledge signal from the CPU, the controller places the 8-bit interrupt vector address value on the address/data bus.

8. The CPU multiplies the 8-bit value by four to determine the actual address in the interrupt pointer (vector) table for the interrupt service routine.

9. The CPU also disables interrupts by clearing the flag bit in its flag register. It then pushes the flag register, IP (instruction pointer), and CS (code segment register) values onto the CPU stack.

10. The new IP and CS values are received from the interrupt pointer table and control jumps to the routine at that address.

11. The interrupt-handling subroutine should push any other CPU registers to be used during the service routine onto the stack.

12. At the end of the second interrupt acknowledge cycle, the in-service register bit will be cleared if the automatic end of interrupt mode is selected. Otherwise, the in-service register bit must be cleared by an end of interrupt command from the CPU at the end of the interrupt service routine.

13. Before the routine is exited, the appropriate CPU registers are returned from the stack, interrupts are enabled, and control returns to the instruction following the one that the CPU completed when the interrupt service routine was called.

End of Interrupt

Three different end-of-interrupt formats are available through programming. They are the non-specific end-of-interrupt command, the specific end-of-interrupt command, and the automatic end-of-interrupt command.

The non-specific end-of-interrupt command does not inform the controller which level of interrupt was involved. The controller is in a mode where it can determine service routine levels. Because of this, it can determine that the interrupt level that applies to the routine just completed corresponds to the highest interrupt level set in the in-service register. The non-specific end-of-interrupt command will reset this bit.

There are two conditions that may cause the non-specific end-of-interrupt routine to fail. One is when the service routine resets the interrupt priorities. The other is when the special mask mode is in use. In both cases, the controller may not be able to determine the routine's interrupt level.

The specific end-of-interrupt command must include the in-service bit to be reset. This allows the programmer the latitude to change interrupt priorities with the servicing routine or perform other functions that might make it vague to the controller which interrupt routine was being serviced, particularly if other service routines were being executed at the same time.

The automatic end-of-interrupt mode eliminates the need for the CPU to issue an end-of-interrupt command to notify the controller that an interrupt service routine has been completed. While in this mode, the controller will perform a non-specific end-of-interrupt at the trailing edge of the second interrupt acknowledge signal from the CPU.

Using the automatic end-of-interrupt mode may cause problems in some instances, because the in-service register bit is reset right after it was acknowledged, leaving no sign that the service routine is being executed. Therefore, any interrupt request (when interrupts are enabled) will get serviced, regardless of its priority, making it possible for an interrupt request to interrupt its own service routine.

It is considered good programming practice not to use the automatic end-of-interrupt mode unless the CPU's interrupt input will be kept disabled while interrupt routines are being serviced.

Priority Assignment

The priority of interrupt channels is based on their relationship to the other channels on the controller. After the initialization sequence, the highest priority level is assigned to IRQ0, the lowest to IRQ15, and the priority assignment is fixed (Fixed Priority Mode). The priority assignment can be rotated manually (Specific Rotation Mode) or automatically (Automatic Rotation Mode).

The user can mask individual interrupt lines without affecting those with different priority. Table 14-1 shows the interrupt level assignments available in this computer.

Table 14-1. Interrupt Line Assignments

INTERRUPT	ASSIGNMENT
IRQ0	Timer channel 0
IRQ1	System control processor (SCP) port 2, bit 4
IRQ2	Slave controller input
IRQ3	PC bus or COM2
IRQ4	PC bus or COM1
IRQ5	PC bus or LPT2
IRQ6	Floppy disk controller
IRQ7	PC bus or LPT1
IRQ8	Real-time clock
IRQ9	IRQ2 on PC BUS
IRQ10	AT bus
IRQ11	AT bus
IRQ12	AT bus
IRQ13	Coprocessor
IRQ14	Hard disk controller
IRQ15	AT bus

Timer channel 0, which has the highest priority of the maskable interrupts, is a clock signal generated by the interval timer in the 82C206. It is used for disk drive timing and to generate certain clock signals used by the operating system. The keyboard interrupt, controlled by the SCP, has the next highest priority. The keyboard interrupt occurs whenever the user presses a key on the keyboard.

Fixed Priority Mode

In fixed priority mode, the interrupts are fully nested with the highest priority assigned to IRQ0 and the lowest priority assigned to IRQ15. Nesting allows interrupts of a higher priority to generate interrupt requests prior to the completion of the interrupt in service. After an interrupt is acknowledged, the vector for the highest priority interrupt is placed on the bus and the in-service register bit for that channel is set. The bit remains set until an end of interrupt is issued for that channel. While the in-service register bit is set, all interrupts of equal or lower priority are inhibited. Interrupts of higher priority will be acknowledged if the CPU has internally re-enabled interrupts. Fixed priority is the default mode.

Automatic Rotation Mode

Automatic rotation may be used when the peripheral interrupts are of equal priority. In this mode, the peripheral is assigned the lowest priority after being serviced. If IRQ4 has the highest priority request being served, the in-service priority before rotation will be:

(Lowest) IRQ8 IRQ7 IRQ6 IRQ5 IRQ4 IRQ3 IRQ2 IRQ1 IRQ0 (Highest).

Once IRQ4 has been serviced, the in-service priority after rotation will be:

(Lowest) IRQ4 IRQ3 IRQ2 IRQ1 IRQ0 IRQ8 IRQ7 IRQ6 IRQ5 (Highest).

If the automatic rotation mode is enabled, it will occur when the end of interrupt (automatic or CPU generated) signal is asserted.

Specific Rotation Mode

Specific rotation allows the system software to reassign the priority levels. Two commands allow this: the set priority command and the rotate on specific end-of-interrupt command.

The set priority command assigns an interrupt request line to the lowest priority and the priorities of the other lines are rotated to conform to the fully nested format.

If the set priority command is used during a service routine, then you must use either a specific end-of-interrupt command or the automatic end-of-interrupt mode to end the routine. The non-specific end-of-interrupt resets the highest in-service register bit, which may not represent the service routine that issued the set priority command. If the automatic end-of-interrupt mode is used, it performs the non-specific end-of-interrupt before the set priority command can be issued; however, the best practice is to use the specific end-of-interrupt command to eliminate any possible confusion.

The rotate on specific end-of-interrupt command is a combination of the set priority command and the specific end-of-interrupt command. With this command, you specify which interrupt request line is assigned to the lowest priority and issue the end-of-interrupt command at the same time. The other lines' priorities are rotated to conform to the fully nested format, based on the bit that has been assigned the lowest priority.

Masking interrupts allows the programmer to enable interrupt request lines that are at a lower priority than the one being serviced. As an example, suppose interrupt request line 4 has triggered its interrupt service routine, but you want to allow lower-level interrupt request lines to interrupt the service routine. Inside the service routine for interrupt request line 4, you would first mask interrupt request line 4 and then issue a special mask mode command. This disables normal nested mode priority operation and enables all interrupt request lines except those being masked. To leave this mode, execute the sequence in reverse order.

There is one problem with using the mask mode. You cannot use a non-specific end-of-interrupt command because all masked interrupt request line bits are hidden and are not clearable from the in-service register. You must exit the mask mode to use the non-specific end-of-interrupt command to clear masked bits from the in-service register. Therefore, it is best to issue a specific end-of-interrupt command when using this mode.

Interrupt trigger — This computer uses two standard methods to sense an interrupt: level-sensitive and edge-sensitive.

In the level-sensitive interrupt mode, the interrupt controller will recognize an active-high on any of its interrupt request lines. If the interrupt request line remains active after the end-of-interrupt command is issued, another interrupt will be generated if the CPU has been told to recognize interrupts. In this mode, the interrupt must remain active until the first interrupt acknowledge from the CPU has been received. Otherwise, the controller will act as if interrupt request line 7 had been active.

The edge-sensitive mode is the default in this computer. In this mode, the interrupt controller recognizes the interrupt on the rising edge as an interrupt request line goes active. If the interrupt request line remains active after the service routine has been completed and the processor is set to recognize interrupts, it will not trigger a subsequent interrupt. In this mode, the interrupt must also remain active until the first interrupt acknowledge from the CPU has been received. Otherwise, the controller will act as if interrupt request line 7 had been active.

The three internal registers of the interrupt controller, the in-service register, the interrupt request register, and the interrupt mask register, can be read by software. The interrupt request register specifies all interrupt request lines that are currently requesting service. The in-service register specifies all interrupt request lines that are currently being serviced by routines. The interrupt mask register specifies all interrupt request lines that are currently masked.

Programming The Interrupt Controllers

Since the system BIOS routines handle both initialization and operation of the 8259, we do not recommend programming the chip directly.

Two types of commands are used to program the interrupt controllers, initialization command words and operational command words. Normal operation cannot begin until the initialization commands have been issued. Operation commands allow the interrupt controller to operate in one of four interrupt modes: fully nested mode, rotating priority mode, special mask mode, and polled mode.

Initialization Commands

During initialization, the CPU must initialize certain internal registers and program the interrupt controller by sending it a series of commands.

The system will first initialize the SS (stack segment) register of the CPU. This register points to the start of the stack, which is located near the top of scratchpad memory between addresses F0000H and F3FFFH. The exact address depends on the version of the Monitor program installed in your computer. Note that only the Monitor program uses this stack location. Under MS-DOS operation, the stack segment register may point elsewhere.

Next, the interrupt vector table in low memory is initialized. This table contains the addresses of the service routines used by each interrupt. If the CPU is operating in real mode, each interrupt vector address is four bytes long. The CS (code segment) register uses two bytes for its base address, while the IP (instruction pointer) uses the other two bytes for the offset address. If the processor is operating in protected mode, each vector consists of 8 bytes of information. This is combined with the address in the interrupt descriptor table to obtain the vector address.

Finally, the CPU will initialize the interrupt controller with a series of initialization commands. These commands set up the controller for normal operation. Tables 14-2 through 14-5 describe the four initialization command words that may be sent to the controller. The first number in the table heading is address of the master interrupt controller and the second number is the address of the slave. Note that either three or four commands may be sent in sequence. The A0 line, when low, identifies the first command word. The A0 line is then set high and the following command words are sent to the controller in sequence.

Table 14-2. Initialization Command Word 1 (020H, 0A0H)

BIT	DESCRIPTION
0	Not used.
1	Informs the controller if it is the only controller in the system. If this bit is set (1), only one controller is in the system and initialization command word 3 will not be issued. This mode not recommended for this device.
2	Not used.
3	Tells the controller to recognize interrupt requests on the leading edge of the interrupt line transition or when the level is high. If this bit is set (1), the interrupt request is recognized on the edge of the transition.
4	Always set (1). This bit indicates to the controller that an initialization sequence is beginning.
5 - 7	Not used.

Initialization command word two helps to determine the interrupt vector for the CPU. Table 14-3 gives the bit descriptions.

Table 14-3. Initialization Command Word 2 (021H, 0A1H)

BIT	DESCRIPTION
0 - 2	Not used.
3 - 7	These are the upper 5 bits of the interrupt vector and are programmable by the CPU. The lower 3 bits of the interrupt vector are generated by the priority resolver during the interrupt acknowledge cycle.

Initialization command word three defines the format for the interrupt controllers. Refer to Table 14-4 for the bit descriptions.

Table 14-4. Initialization Command Word 3 (021H, 0A1H)

BIT	DESCRIPTION
0 - 7	When the master controller is active, these bits indicate which interrupt request line is attached to the slave controller. Each bit corresponds to an interrupt request line.
0 - 7	When the slave controller is active, the lowest three bits indicate which interrupt request line is attached to the master controller. For instance, if this controller is programmed with a 6 (bits 1 and 2 set), then the slave controller would be connected to interrupt request line 6 of the master controller. In addition, bit 6 in command word 3 for the master interrupt controller would be set.

Initialization command word four is explained in Table 14-5.

Table 14-5. Initialization Command Word 4 (021H, 0A1H)

BIT	DESCRIPTION
0	Not used.
1	Automatic end-of-interrupt enabled.
2-3	Not used.
4	Enable multiple interrupts on the same channel (fixed priority mode only).
5 - 7	Not used.

Operational Command Words

Once initialized, the interrupt controller is ready to accept interrupt requests through its input lines. Three different command options allow the controller to operate in different modes. These commands, unlike the initialization command words, do not have to be in sequential order, but may be sent to the controller as needed.

Operational command word 1 is located at 021H and 0A1H. Use command word 1 to place a mask in the controller's interrupt mask register. Each of the data bits sent to the controller during the write operation corresponds to one of the interrupt request lines. Bit 0 corresponds to interrupt request line 0, bit 1 to line 1, and so on. If a bit is set (1), the line is masked. If a bit is clear (0), the line is not masked. Address line 0 must be set prior to executing this write operation to the controller.

Operational command word 2 is located at 020H and 0A0H. Use command word 2 for the end-of-interrupt and rotation commands. It is summarized in Table 14-6. Address line 0 must be clear (0) for the write operation of this command word.

Table 14-6. Operation Command Word 2 (020H, 0A0H)

BIT	DESCRIPTION
0 - 2	Determine the interrupt channel that is selected.
3 - 4	These two bits are clear (0), signifying that this is command word 2.
5 - 7	Select operational function. Refer to Table 14-7.

Table 14-7. Operational Function Definition Bits 5-7

Bit 7	Bit 6	Bit 5	Function
0	0	0	Rotate on auto end-of-interrupt (EOI) disable
0	0	1	Non-specific EOI
0	1	0	No operation
0	1	1	Specific EOI
1	0	0	Rotate on auto EOI enable
1	0	1	Rotate on non-specific EOI
1	1	0	Specific rotate
1	1	1	Rotate on specific EOI

The controller uses command word 3 for a number of purposes. The functions performed by command word 3 are summarized in Table 14-8.

Table 14-8. Operation Command Word 3 (020H, 0A0H)

BIT	DESCRIPTION
0	In-service/interrupt request register select. A 0 indicates the interrupt request register will be read. A 1 indicates the in-service register will be read
1	Read register command enable. A 0 indicates the read register command will not be issued by the controller. A 1 indicates the read register command will be issued by the controller.
2	Polled mode enable/disable. In polled mode, the 82C206 performs the equivalent of an INTA cycle during the next I/O read operation to the controller. A 0 disables the polled mode and a 1 enables the polled mode.
3 - 4	Bit 3 is set (1) and bit 4 is clear (0), signifying that this is command word 3.
5 - 6	Special mask mode enable/disable. During special mask mode, writing a 0 to any bit position enables interrupts on the associated channel by causing the priority resolver to ignore the condition of the in-service register. Writing a 1 to any bit position inhibits the interrupts. When both bits are written with a 1, the special mask mode is enabled. When bit 6 is written with a 1 and bit 5 is written with a 0, the special mask mode is disabled.
7	Not used.

The Programmable Counter/Timer

The programmable counter/timer chip (CTC) on the 82C206 generates time delays and eliminates the need for software timing loops that are subject to such variables as CPU clock speed and interrupt service routines. The CTC contains three 16-bit counters which operate independantly of each other. Each counter can be programmed to operate as a timer or a counter. Refer to figure 14-3 for a block diagram of the counter/timer.

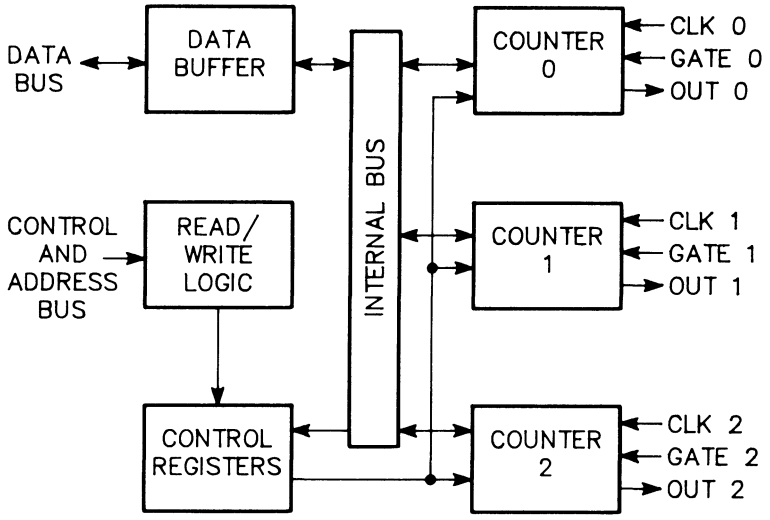


Figure 14-3. Programmable Counter/Timer Block Diagram

Operation

Each counter in the CTC contains a control register, a status register, a 16-bit counting element, two 8-bit counter input latches, and two 8-bit counter output latches. The clock input on each counter loads and decrements the counting element, while a mode defined GATE input controls the counter and an OUT signal. The state and function of the OUT signal are controlled by the counter mode and condition of the counting element.

The control register receives the instructions that program the counters. These instructions are the control words that contain the mode, the type of command and the count format information.

The status register allows the software to monitor the condition of the counter and read the contents of the control register.

The counting element is a 16-bit loadable synchronous down counter. It can be read or written to through the counter output or input latches. These latches are transparent and can be read while latched.

Control information written to the CTC is decoded by the control logic. This provides the necessary controls to load, read, and configure each counter. The output from counter 0 is used as a system interrupt, and is connected internally to the interrupt controller portion of this chip. Counter 1 may be programmed to generate pulses or square waves for external use. Counter 2 may be operated in any of six modes (Mode 0 - Mode 5) for use as an interval timer, a counter, or as a gated pulse/rate generator.

The Timer Control Word

At power-up, the contents of the registers and counters are undefined, so each counter must be programmed before it can be used. The control word register is located at address 043H. The bits that make up the control word are described in Table 14-9. Since all three registers (one for each counter) are identical, only one register is described.

Table 14-9. Control Word Bit Definitions (043H)

BIT	DESCRIPTION
0	Counter format: 0 16-bit binary format 1 4-digit binary-coded decimal
1 - 3	Counter mode select. Refer to Table 14-10.
4 - 7	Command selection. Refer to Table 14-11.

Table 14-10. Counter Mode Selection

BIT 3	BIT 2	BIT 1	MODE
0	0	0	Mode 0
0	0	1	Mode 1
0	1	0	Mode 2
0	1	1	Mode 3
1	0	0	Not valid
1	0	1	Not valid
1	1	0	Mode 4
1	1	1	Mode 5

Table 14-11. Command Selection

BIT 7	BIT 6	BIT 5	BIT 4	DESCRIPTION
0	0	0	0	Latch counter 0.
0	0	0	1	Read/Write counter 0 LSB only.
0	0	1	0	Read/Write counter 0 MSB only.
0	0	1	1	Read/Write counter 0 LSB then MSB.
0	1	0	0	Latch counter 1.
0	1	0	1	Read/Write counter 1 LSB only.
0	1	1	0	Read/Write counter 1 MSB only.
0	1	1	1	Read/Write counter 1 LSB then MSB.
1	0	0	0	Latch counter 2.
1	0	0	1	Read/Write counter 2 LSB only.
1	0	1	0	Read/Write counter 2 MSB only.
1	0	1	1	Read/Write counter 2 LSB the MSB.
1	1	X	X	Read-back command.

X = Don't care.

Each of the three counters contain two bytes and may be configured as either a 16-bit binary counter or a 4-digit binary-coded decimal counter. The input, gate, and output lines are configured by the modes programmed through the control register. In addition, each counter may be read selectively without first inhibiting its clock input.

Mode Definitions

While the programming is fully explained in this section, many of the functions are not available, since the timer is dedicated to specific tasks. The counter can operate in one of the following modes:

Mode 0	Interrupt on terminal count
Mode 1	Hardware retriggerable one-shot
Mode 2	Rate generator
Mode 3	Square-wave rate generator
Mode 4	Software-triggered strobe
Mode 5	Hardware triggered strobe.

Interrupt on Terminal Count (Mode 0) — This mode decrements the counter until it reaches 0 and then asserts its output line high. The counting element must be loaded with an initial count.

Hardware Retriggerable One-Shot (Mode 1) — In this mode, the output will go low in the clock cycle after the gate is asserted high (acting as a trigger). The output will remain low until the counter has decremented to zero. If the gate input line goes low and then high again, the counter will be reset and will be held low until the counter decrements to zero.

The counter, when acting as a one-shot, is retriggerable. Therefore, the output will remain low until the counter has decremented from the original value to zero following the rising edge of the gate. The rising edge of the gate reloads and reinitiates counting.

Rate Generator (Mode 2) —In this mode, the counter functions as a divide by n counter. It will produce a low for one clock cycle on its output pin every n clock cycles. As long as the gate remains high, the counter will repeatedly decrement the value to zero and start over. If the counter's value is changed, the new value will take effect for the next counting cycle and not affect the current cycle.

Square Wave Rate Generator (Mode 3) — This is similar to Mode 2. In this mode, the counter is decremented by two every clock cycle rather than by one, effectively cutting in half the time it takes to reach zero.

If the programmed value is even, the counter is always decremented by two. When the counter reaches zero, the state of the output will change. The counter is then reloaded and the decrementing starts over. When the counter reaches zero, the output state changes again, the counter is reloaded, and the process starts over again. This produces an even square wave.

If the value is odd and the output is high, the first clock cycle will decrement the counter by one and then by twos until it reaches zero. The clock will then change state to low, the counter will be reloaded with the programmed value, and the first clock pulse will decrement it by three, then by twos until it reaches zero. This produces a signal where the output is high for (n+1)/2 counts and low for (n-1)/2 counts.

When the gate is low, counting is disabled and the output is high. The rising edge of the gate initiates counting. When the gate is high, counting is enabled.

Software Triggered Strobe (Mode 4) — In this mode, when the counter reaches zero, it will place a single pulse on its output that lasts for one clock cycle. Counting will begin when the counter is programmed with a value.

Hardware Triggered Strobe (Mode 5) — In this mode, when the counter reaches zero, it will place a single pulse on its output that lasts for one clock cycle. The counter will not start decrementing its value until it senses the rising edge of the gate input. The counter is retriggerable and will reload after it reaches zero.

Programming Considerations

The Monitor ROM is used to program the mode and initial value in each of the three counters in the timer. In each case a control word is placed in the selected counter's register before the counter is started.

There is no special sequence for programming the timer. You can write a command word to any of the three counters without affecting the operation or programming of the other two. The only sequence that has to be followed is for loading two bytes into a counter. The least-significant byte is followed by the most-significant byte before the counter can start operating.

All counters decrement only; if you load a counter with a value of zero, the actual count will equal the maximum value (10,000 in BCD and 65,536 in binary).

The address lines determine which counter is read. Normally you would read the contents of the counter with normal read operations. However, you may wish to read the value in the counter while it is operating. This might present a problem since you may not get a valid result while the counter is continuing to decrement. The timer offers a method of latching the output of the counter so that it is stable while it is being read. The counter continues to decrement during the read operations but the data remains as it was when it was latched.

The timer can read or write values to the counters through IN (read) and OUT (write) programming instructions. Table 14-12 describes each read and write operation that may be performed on the timer.

Table 14-12. Timer Read and Write Operations

PORT	OPERATION	DESCRIPTION
040H	OUT	Write (load) counter 0.
040H	IN	Read the contents of counter 0.
041H	OUT	Write (load) counter 1.
041H	IN	Read the contents of counter 1.
042H	OUT	Write (load) counter 2.
042H	IN	Read the contents of counter 2.
043H	OUT	If either bit 4 or bit 5 are set (1), write a control word to the specified counter's control register. Refer to Table 14-8 for a description of the contents of the control word. If bits 4 and 5 are both clear (0), latch the specified counter's value for reading.
043H	IN	Place the data buffer in the high-impedance state; do not perform any operation.

The DMA Controllers

Direct memory access (DMA) speeds the transfer of data from one memory location, device, or peripheral to another. The primary advantage of DMA is in moving blocks of data. The DMA controller has hardware instructions that operate much faster than the software-based CPU instructions. The 82C206 has the equivalent of two 8237 DMA controllers. Refer to Figure 14-4 for a block diagram of the DMA controller.

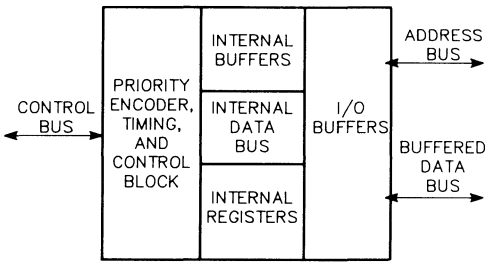


Figure 14-4. DMA Controller Block Diagram

The DMA controller may be dynamically programmed by the CPU whenever the HLDA line is held inactive. Even if a request for the system buss is being made, the CPU can still provide instructions to the controller. The CPU must not attempt to program the DMA controller whenever HLDA is active.

NOTE: If a request for DMA service occurs over an unmasked request line while the CPU is attempting to specify a particular address, that address may be corrupted. Then, when the operation is carried out, the DMA controller will attempt to address the wrong memory location, and therefore transfer the wrong byte of data. Therefore, it is good practice to mask each of the request lines before you attempt to dynamically program the DMA controller.

Operation

The DMA controller operates in either an idle cycle or an active cycle. When the controller is idle there is no valid software- or hardware-generated DMA request on any of the request lines. The controller continually samples each request line until one of them asserts an active request for service. If no requests occur and the CPU is free, the CPU can dynamically program the DMA controller's registers. The DMA chip select signal and the hold acknowledge signal from the CPU must both be low to place the controller in program mode.

When an active request for DMA service occurs on a channel that has not been previously masked, the DMA controller enters its active cycle. The request will be asserted until the processor recognizes it, completes its current operations according to priority, and then issues a hold acknowledge (HLDA) back to the DMA controller. This acknowledge informs the controller that the system buss is at its disposal.

The CPU can change values in one of the registers or merely read the status of the registers. A 4-bit address applied across lines A0 - A3 acts as a pointer to the desired register. Values can be read or written in a particular register in accordance with the IOR (input/ output read) and IOW (input/output write) control signals.

When a transfer of data is necessary, the CPU tells the controller where to get the data and where to put it in memory. The controller then handles the data transfer. It generates up to 16 address lines through address outputs A0-A7. The high-order address is first placed on the address bus and latched. The controller only latches the high-order address when it changes, speeding the transfer process. The address latches and buffers are enabled by a high on the DMA line. A logic array generates all of the memory and input/output read and write signals (MEMR, MEMW, IOR, and IOW) from the S0, S1, and S2 lines.

When the DMA controls the bus, a hold request signal looks for the CPU idle cycle, as defined by the S0, S1, and S2 lines. When the idle cycle is recognized, the controller locks the CPU in a hold state until the transfer is completed.

A peripheral can request DMA service from the controller on any of the DRQ lines. The controller responds by requesting a CPU hold and then sends a DMA acknowledge to the peripheral when the CPU enters a hold state.

Internal Registers

The DMA controllers are initialized at powerup and zeros are written to the internal register sets. There are eighteen 16-bit registers, three 8-bit registers, four 6-bit registers, and two 4-bit registers inside each DMA controller that store instructions written

to the device by the CPU. Refer to Table 14-13 for a description of each register. The following paragraphs describe each of the registers within the DMA controller. Table 14-14 shows the address locations of each register.

Table 14-13. DMA Controller Registers

REGISTER	SIZE	NUMBER
Base address registers	16-bit	4
Base word count registers	16-bit	4
Current address registers	16-bit	4
Current word count registers	16-bit	4
Temporary address register	16-bit	1
Temporary word count register	16-bit	1
Status register	8-bit	1
Command register	8-bit	1
Temporary register	8-bit	1
Mode registers	6-bit	4
Mask register	4-bit	1
Request register	4-bit	

Port addresses and their associated registers are listed in Table 14-14. Each DMA chip is addressed separately, and the exact register function selected is dependent on the state of the $\overline{\text{XIOR}}$ and $\overline{\text{XIOW}}$ lines. Selection is also dependent on the state of an internal flip-flop. At powerup, this flip-flop is set to zero, and each time a DMA register is selected it changes state. For example, assume the chip has been reset, clearing the internal register flip-flop. To write to Channel 0 base address an DMA chip two, you must first assert the $\overline{\text{XIOR}}$ line, set $\overline{\text{XIOW}}$ to 0, and write the data to address location 0C0H. This would enter the base address low byte. At this point the internal flip-flop changes state, and the next word that is written (assuming the state of $\overline{\text{XIOR}}$ and $\overline{\text{XIOW}}$ is unchanged) is routed to the base address high byte.

Table 14-14. DMA I/O Register Addresses

DMA1	DMA2	$\overline{XIOR}$	$\overline{XIOW}$	FLIP-FLOP	REGISTER FUNCTION
000H	0C0H	0	1	0	Read channel 0 current address low byte.
		0	1	1	Read channel 0 current address high byte.
		1	0	0	Write channel 0 base and current address low byte.
		1	0	1	Write channel 0 base and current address high byte.
001H	0C2H	0	1	0	Read channel 0 word count low byte.
		0	1	1	Read channel 0 word count high byte.
		1	0	0	Write channel 0 base and current word count low byte.
		1	0	1	Write channel 0 base and current word count high byte.
002H	0C4H	0	1	0	Read channel 1 current address low byte.
		0	1	1	Read channel 1current address high byte.
		1	0	0	Write channel 1 base and current address low byte.
		1	0	1	Write channel 1 base and current address high byte.
003H	0C6H	0	1	0	Read channel 1 word count low byte.
		0	1	1	Read channel 1word count high byte.
		1	0	0	Write channel 1 base and current word count low byte.
		1	0	1	Write channel 1 base and current word count high byte.

Table 14-14 (continued). DMA I/O Register Addresses

DMA1	DMA2	$\overline{XIOR}$	$\overline{XIOW}$	FLIP-FLOP	REGISTER FUNCTION
004H	0C8H	0	1	0	Read channel 2 current address low byte.
		0	1	1	Read channel 2 current address high byte.
		1	0	0	Write channel 2 base and current address low byte.
		1	0	1	Write channel 2 base and current address high byte.
005H	0CAH	0	1	0	Read channel 2 word count low byte.
		0	1	1	Read channel 2 word count high byte.
		1	0	0	Write channel 2 base and current word count low byte.
		1	0	1	Write channel 2 base and current word count high byte.
006H	0CCH	0	1	0	Read channel 3 current address low byte.
		0	1	1	Read channel 3 current address high byte.
		1	0	0	Write channel 3 base and current address low byte.
		1	0	1	Write channel 3 base and current address high byte.
007H	0CEH	0	1	0	Read channel 3 word count low byte.
		0	1	1	Read channel 3 word count high byte.
		1	0	0	Write channel 3 base and current word count low byte.
		1	0	1	Write channel 3 base .and current word count high byte.
008H	0D0H	0	1	X	Read status register.
		1	0	X	Write command register.
009H	0D2H	0	1	X	Read DMA request register.
		1	0	X	Write DMA request register.
00AH	0D4H	0	1	X	Read command register.
		1	0	X	Write single bit DMA request mask register.
00BH	0D6H	0	1	X	Read mode register.
		1	0	X	Write mode register.

Table 14-14 (continued). DMA I/O Register Addresses

DMA1	DMA2	$\overline{XIOR}$	$\overline{XIOW}$	FLIP-FLOP	REGISTER FUNCTION
00CH	0D8H	0	1	X	Set byte pointer flip-flop.
		1	0	X	Clear byte pointer flip-flop.
00DH	0DAH	0	1	X	Read temporary register.
		1	0	X	Master clear.
00EH	0DCH	0	1	X	Clear mode register counter.
		1	0	X	Clear all DMA request mask register bits.
00FH	ODEH	0	1	X	Read all DMA request mask register bits.
		1	0	X	Write all DMA request mask register bits.

Base Registers — These registers include the base address registers and the base word count registers. There is one of each register for every DMA channel. The values contained in the base registers represent the original value placed in the current address and current word registers (discussed later) by the host processor. The purpose of the base registers is to load the current registers with their base values at initialization.

The base registers cannot be read by the host processor but values can be placed in the registers (simultaneously) when the host processor is in the program mode.

Current Registers — There is a current address register and a current word count register for each DMA channel. The current address register reflects the source and destination addresses for data transfers. During a data transfer any intermediate address values are stored in the temporary address register. Each time a data transfer is completed, the current address register is incremented or decremented. The current address register can be reloaded with its original value through execution of an initialization routine. The original value is supplied by the base address register when the EOP (end of process) signal is active.

The current word count register is loaded with a value that determines how many data transfers to execute. The device will execute one additional transfer beyond the number that it is programmed to execute. For example, if ten transfers are required by the system, the current word count register must be programmed to execute nine operations. After each transfer occurs the register is decremented. Intermediate values during a given transfer are stored in the temporary word count register. The current word count register can be loaded with its original value through execution of an initialization routine. The original value is supplied by the base word count register when the EOP (end of process) signal is active.

The current registers can be read by the host processor and values can be placed in the registers (simultaneously) when the host processor is in program mode.

Status Register — This 8-bit register provides the host processor with current status of the DMA devices in its environment. When the processor reads this register, it can determine which channels are programmed to execute an operation and which ones have reached a terminal count (completed an operation). Bits 4 - 7 of the status register indicate that a corresponding channel is requesting service.

Command Register — The command register contains eight bits that control the overall operation of the DMA subsystem. The host processor can dynamically place values in this register when the CPU is in program mode. The contents of the register can be cleared through a system reset or a clear instruction implemented through software. Table 14-15 details the functions of each bit in the command register.

Table 14-15. Command Register

BIT	DESCRIPTION
0	0 = Memory-to-memory transfers disabled. 1 = Memory-to-memory transfers enabled.
1	0 = Channel 0 address hold disabled. 1 = Channel 0 address hold enabled .
NOTE: If bit 0 is set to zero, bit 1 is meaningless.	
2	0 = DMA controllers enabled. 1 = DMA controllers disabled.
3	0 = Normal timing mode. 1 = Compressed timing mode.
4	0 = Fixed priority mode. 1 = Rotating priority mode.
5	0 = Late write selection mode. 1 = Extended write selection mode.
6	0 = DMA request line is an active high signal. 1 = DMA request line is an active low signal.
7	0 = DMA acknowledge line is an active low signal. 1 = DMA acknowledge line is an active high signal.

Temporary Register — When a memory-to-memory transfer is taking place, this register temporarily stores data. The byte is moved into the register, remains there while addressing and other bus operations are occurring, and then is moved to its specified memory location. After the transfer is completed, the last byte transferred remains in the register until cleared. The last byte can also be read by the host processor when in program mode.

Mode Registers — The mode register is used to set the operating mode of the DMA controller, provide operation status, and select an active DMA channel. Each DMA channel has its own 6-bit mode register that can be written to by the CPU when it is in program mode. Table 14-16 details the function of each bit in the mode register.

Table 14-16. Mode Register

BIT	DESCRIPTION
0 - 1	Determine which DMA channel is active: BIT 0 BIT 1 0 0 Channel 0 0 1 Channel 1 1 0 Channel 2 1 1 Channel 3
2 - 3	Determine read, write, or verify transfer status: BIT 2 BIT 3 0 0 Verify 0 1 Write 1 0 Read 1 1 Invalid
	NOTE: If bits 6 and 7 are both set to one, then the values of bits 2 and 3 are meaningless.
4	Self-initialization routine enable/disable: 0 Self-initialization disabled 1 Self-initialization enabled
5	Decrement/increment the value in the current address register: 0 Increment value in in the current address register 1 Decrement value in in the current address register
6 - 7	Determine the operating mode in which the controller will execute data transfers: BIT 6 BIT 7 0 0 Demand mode 0 1 Single mode 1 0 Block mode 1 1 Cascade mode

Mask Register — The mask register can be used to disable DMA requests on each DMA channel. The values placed in the mask register determine which channel is masked or not masked. Table 14-17 details the function of the bits responsible for masking operations.

Table 14-17. Mask Register

BIT	DESCRIPTION
0 - 1	DMA channel mask: BIT 0 BIT 1 0 0 Channel 0 masked 0 1 Channel 1 masked 1 0 Channel 2 masked 1 1 Channel 3 masked
2	Clear/enable the conditions set by the DMA channel mask bits: 0 All mask bits cleared 1 Channel mask bits enabled to mask the chosen register.

When a master reset occurs (during powerup, for example), each bit for each channel is set to one. DMA requests on each channel will be disabled until a clear-mask instruction changes the state of the appropriate bits. Software can be used exclusively to control the operation of the mask registers. Table 14-18 details how a single command can be used to provide the same masking.

Table 14-18. Single Mask Command

BIT		DESCRIPTION
0	0	DMA channel 0 not masked
	1	DMA channel 0 masked
1	0	DMA channel 1 not masked
	1	DMA channel 1 masked

Table 14-18 (continued). Single Mask Command

BIT	DESCRIPTION	
2.	0	DMA channel 2 not masked
	1	DMA channel 2 masked
3	0	DMA channel 3 not masked
	1	DMA channel 3 masked

After completion of a given data transfer, the end of process (EOP) signal is asserted by the controller. Unless the device has been programmed to execute its self-initialization routine, each mask register bit will be placed in a set condition. To prevent masking of DMA requests on a given channel after each transfer operation, program the DMA controller to initialize itself.

Request Register — The request register is used to prioritize DMA requests. It can respond to requests initiated through software or hardware (DREQ signal). The priority of the requests is established by an internal encoder network. One request bit that cannot be masked is present for each DMA channel. Each of these bits can be set under software control. Table 14-19 details the function of each bit in the request register.

Table 14-19. Request Register

BIT	DESCRIPTION		
0 - 1	Set the DMA channel request bit:		
	BIT 0	BIT 1	
	0	0	Set channel 0
	0	1	Set channel 1
	1	0	Set channel 2
	1	1	Set channel 3
2	Clear/enable the DMA channel request bit:		
	0	All request bits cleared	
	1	Request bits are enabled to set the chosen channel.	

Operating Modes

The DMA controller can be programmed to operate in one of the four following modes:

Single Transfer Mode — In this mode of operation, the DMA controller will execute one data transfer at a time. The value in the current word count register will be decremented upon completion of the transfer and the value in the current address register may be incremented or decremented (depending on system requirements) upon completion of a transfer. If the current word count register increments beyond the value of FFFFH, a terminal count indication will occur and the controller will initialize itself (if programmed to). Self-initialization is discussed later in this section.

During the cycle, the channel will assert a request signal until the CPU issues a hold acknowledge to the controller, which then issues a DMA cycle acknowledgement (DACK) to the requesting device. When the transfer is completed, the system busses are relinquished even if the request for DMA service is still active. The cycle will then repeat.

Block Transfer Mode — In this mode of operation, the requesting device must wait for acknowledgement from the controller, which must wait for acknowledgement from the CPU, before the system busses will be available. Once the transfer starts, service will continue until a terminal count indication is generated. A terminal count occurs when the value in the current-word-count register exceeds FFFFH. Service may also be stopped if the end-of-process signal is asserted. When the transfer stops, the controller will self-initialize, if programmed to.

Demand Transfer Mode — In this mode, transfers will continue unless a terminal count (caused by the value in the current word count register exceeding FFFH) or end-of-process signal is generated or the requesting channel is masked or goes inactive. An advantage of this mode is when a transfer is made from memory to an I/O device that contains some type of data buffer, the transfer continues until that buffer is full and the request for service goes inactive. The system processor then executes other operations until the I/O device's buffer empties and it issues another request for service. Any values that are pertinent to the operation are

preserved in the current address and current word count registers until the operation resumes. In this mode the controller will not self-initialize unless it receives an end of process signal.

Cascade Mode — The cascade mode allows more than one slave DMA controller to be connected to a master controller. The slaves have to issue their hold request signals to the master controller. The hold-request is issued over the master's DMA request channel and the master's hold acknowledge signal is distributed to the slaves over the DMA acknowledge channel.

The master maintains direct communication with the system CPU to allow the CPU to establish programming values and to allow itself to ascertain when the system busses are free. One advantage of cascading controllers is that the slave controllers provide a type of prioritization over incoming requests for service. If multiple devices are requesting service, the latter of those devices must wait for acknowledgement in a prioritized manner.

Transfer Modes

The DMA controller in the 82C206 provides four types of transfers. The following transfer modes are available:

Memory-To-Memory Transfers — The DMA controller can perform transfers from one area of memory to a different area of memory. If bit 0 in the command register is set to a one, DMA channels 0 and 1 are programmed to operate as an interfacing channel from one segment of memory to another.

The service request for memory-to-memory transfers must occur through software because the normal DREQ (DMA request) lines are not tied directly to memory. The request and acknowledge logic is managed as with a typical hardware request.

During the actual transfer, the controller operates in its block transfer mode. The current address register for channel 0 provides the addresses of the memory string to be transferred. As the operation takes place, this register is incremented or decremented as required.

The bytes of data that are fetched during the operation are held in the controller's temporary register. The current-address register for channel 1 specifies the address at which the data byte is to be placed. This value is incremented or decremented as required. The data byte is then passed through channel 1 to the target memory location.

The channel 1 current word count register is decremented until it counts beyond 0000H and rolls under to FFFFH. It then generates a terminal count which causes an end of process signal to be issued to the system. External end of process signals will also terminate the operation.

Read Transfer — During this mode, data is moved to the I/O device from memory by generating the memory address and asserting DMAMEMR and XIOW during the same cycle.

Write Transfer — This mode moves data to memory from the I/O device by generating the memory address and asserting DMAMEMW and XIOR during the same cycle.

Verify Transfer — This is a pseudo-transfer used for diagnostics. In verify transfer mode, the DMA will operate as if it were performing a Read or Write transfer by generating HRQ, addresses and DACK but will do so without generating a command signal. Sinch no transfer takes place, the condition of IOCHRDY is ignored during this operation.

Self-Initialization

When bit 4 of the mode register is set to a one, it allows the controller to be placed in the self-initialization mode. During normal transfers, values in the current address register and current word count register may be changed. When an operation is completed, these registers contain their last hexidecimal value. Self-initialization clears these registers and then loads them with default values.

The base address register and base word count register contain the default values that are loaded into the current-address register and current word count register. This process occurs after detection of an end of process signal. After self-initialization, DMA operation can occur without intervention whenever a valid request is made.

Priority Management

The DMA controller can prioritize DMA service on a fixed priority basis or a rotating priority basis. These options can be selected through software.

Fixed priority means that channel 0 has highest priority, progressing downward to channel 3, which has lowest priority. Whenever a request is serviced and an operation is occurring through a given channel, that channel has priority over any other channels. When the operation is completed, channel 0 will be checked for a service request, then channel 1, and so on.

Rotating priority means that the channel currently performing an operation will have the lowest priority after the operation is completed. The advantage to rotating priority is that each channel receives service at least once every fourth service request. This provides a well-balanced environment for DMA operations and prevents prolonged service through one specific channel.

Compressed Timing

The compressed timing mode allows greater throughput in those systems that can support it. Basically, the controller executes the same operation in only two clock cycles. This mode should not be used if system timing requirements are such that bus contention will occur or data set-up times will not be met.

The Real-Time Clock

The 82C206 includes an internal real-time clock to maintain the system time and date. It also automatically tracks the end-of-month, leap years, and daylight saving time. The internal 100-year calendar ensures accurate date recording for the life of your computer. Figure 14-5 is a block diagram of the real-time clock.

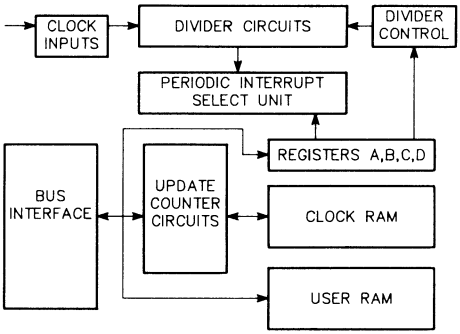


Figure 14-5. Real-Time Clock Block Diagram

Real Time Clock Operation

The output of a 22 stage binary-divider is the 1 Hz signal used to update the time keeping information. Once each second all ten bytes are switched to the update logic. When this happens a set sequence of events starts. First, the seconds byte is incremented and then checked for overflow, then the minutes byte is incremented, if necessary, and tested for overflow. This process of incrementing a byte where required and testing for an overflow condition continues through the last timekeeping register. If a program attempts to read this information during an update cycle, it will receive undefined data. Real time clock data is accessible at address locations 071H through 07FH.

Whenever an update cycle is in progress, the real time clock pulls the timekeeping registers off the system bus to prevent the system from reading transitional data. At the end of this process each alarm byte is compared with its corresponding time byte. If all three match, the clock issues an alarm.

Internal Registers

The 128 addressable locations in the real time clock consist of 14 bytes used as registers to record timekeeping information and 114 bytes of RAM. In addition to the memory storage areas, the device incorporates a binary divider to control interrupt selection and a bus interface to move data to and from the system bus lines. Figure 14-6 illustrates the internal memory map of this device.

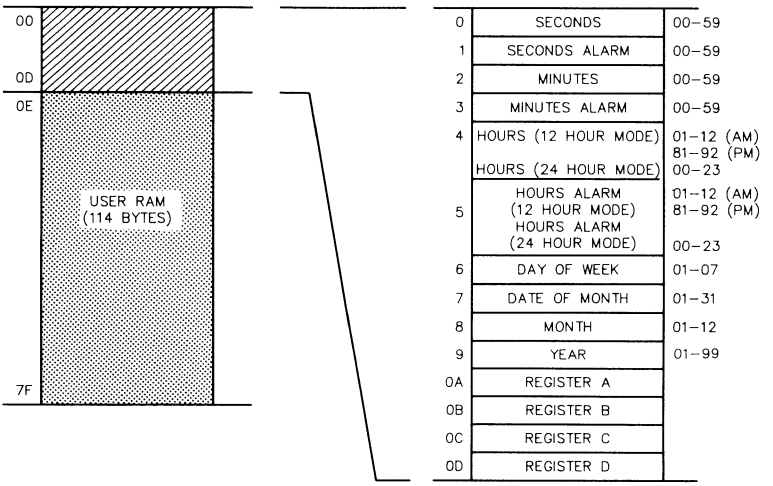


Figure 14-6. Internal Address Map

Of the 14 bytes used for timekeeping information, the first 10 bytes contain specific time-related information. The remaining four registers are used to record status information for the device. The status bytes are not totally accessible to programmers. Registers C and D are read-only registers. Bit 7 in register A is also read-only. Also, bit 7 of the seconds byte is a read-only bit. Figure 14-7 details

the organization of the four internal status registers. Refer to this figure as you read the following material.

	MSB 7	6	5	4	3	2	1	LSB 0	BIT #
REGISTER A	UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0	READ/WRITE
REGISTER B	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE	READ/WRITE
REGISTER C	IRQF	PF	AF	UF	· ·	· ·	· ·	· ·	READ ONLY
REGISTER D	VRT	· ·	· ·	· ·	· ·	· ·	· ·	· ·	READ ONLY

Figure 14-7. Real Time Clock Status Registers

Register A (0AH)

Each bit in register A can be read or written with the exception of bit 7 which is a read-only bit. Table 14-20 describes the functions of the bits in this register.

Table 14-20. Register A Bit Operations

BIT	NAME	DESCRIPTION
0 - 3	RSO - RS3	Rate selection bits 0 through 3. These bits allow the programmer to select one of 15 separate taps on the internal 22-stage divider. This establishes the rate for the periodic interrupt cycle. (Table 14-21 details these selection options.)
4 - 6	DV0 - DV2	Divider selection bits 0 through 2. The programmer uses these bits to reset the internal divider chain in the real time clock.

Table 14-20 (continued). Register A Bit Operations

BIT	NAME	DESCRIPTION
7	UIP	Update in progress bit. When this status flag is set (1), an update cycle is in progress or due to begin. When this bit is clear (0), the time, calendar, and alarm information is available to the programmer. If the bit is clear, another update cycle will not start for at least 244 µs.

Table 14-21. Periodic Interrupt Time Interval Options

RS3	RS2	RS1	RS0	RATE
0	0	0	0	None
0	0	0	1	3.90625 ms
0	0	1	0	7.8125 ms
0	0	1	1	122.070 µs
0	1	0	0	244.141 µs
0	1	0	1	488.281 µs
0	1	1	0	976.562 µs
0	1	1	1	1.953125 ms
1	0	0	0	3.90625 ms
1	0	0	1	7.8125 ms
1	0	1	0	15.625 ms
1	0	1	1	31.25 ms
1	1	0	0	62.5 ms
1	1	0	1	125 ms
1	1	1	0	250 ms
1	1	1	1	500 ms

Register B (0BH)

All of the bits in register B can be read or written by the programmer. Table 14-22 describes the functions of the bits in this register.

Table 14-22. Register B Bit Operations

BIT	NAME	DESCRIPTION
0	DSE	Daylight saving time bit. 0 Daylight saving time disable 1 Daylight saving time enable
1	24/12	24 hour/12 hour mode select. 0 12-hour mode enable 1 24-hour mode enable
2	DM	Data mode bit. This bit determines whether updates are to occur in binary or BCD format. 0 BCD mode enable 1 Binary data mode enable
3	SQWE	Square-wave enable bit. 0 No square wave output 1 Square wave output at the frequency selected by the rate selection bits (RS3 - RS0).
4	UIE	Update-ended interrupt enable bit. 0 Disables the update-end flag bit to assert IRQ. 1 Allows the update-end flag bit in register C to assert IRQ.
5	AIE	Alarm enable interrupt bit. 0 Alarm flag bit will not assert an IRQ signal. 1 Allows the alarm flag bit in register C to assert the IRQ line.
6	PIE	Periodic interrupt enable bit. 0 Disables the generation of periodic interrupts. 1 Allows the periodic interrupt flag in register C to assert the IRQ line. The rate of this interrupt is determined by RS3 - RS0.
7	SET	Set. 0 Enables the update cycle and allows the real time clock to function normally. 1 The update cycle is inhibited and any cycle in progress is aborted.

Register C (0CH)

Register C is one of two read-only registers in this set. The information in register C is returned by the real time clock so that a program can obtain current status. Table 14-23 describes the bit functions in this register.

Table 14-23. Register C Bit Operations

BIT	NAME	DESCRIPTION
0 - 3	—	Not used.
4	UF	Update-ended interrupt flag. This flag is set at the end of each update cycle. If the UIE bit in register B is also set, the IRQF bit will set, asserting the IRQ line.
5	AF	Alarm interrupt flag. If this bit is set at the end of a cycle, it means that the current time matches the stored alarm time. If the AIE bit in register B is also set, IRQF will set, asserting the IRQ line.
6	PF	Periodic interrupt flag. When this bit is set (1), the IRQF bit will also be set if PIE is set.
7	IRQF	Interrupt request flag. This bit is set (1) if one or more of the following conditions are true: PF and PIE are set, AF and AIE are set, or UF and UIE are set.

Register D (0DH)

Register D is the second read-only register in the real time clock. Only bit 7, the valid ram and time (VRT), bit is used. This bit reports the status of the 114 bytes of RAM within the real time clock. As long as this bit is set, the contents of the onboard RAM and clock registers is valid. If this bit is clear, it indicates that power has dropped below a nominal level. If this happens, assume that the contents of internal clock registers and RAM are no longer valid. Whenever register D is read, bit 7 will be set.

Time, Calendar, and Alarm Registers

Ten registers in the real time clock store information about the current time, date, and alarm status. These registers may be initialized or read by a program except when an update cycle is in progress. Table 14-24 details the available data modes for these registers.

Table 14-24. Time, Calendar, and Alarm Data Modes

REGISTER ADDRESS	FUNCTION	DECIMAL RANGE	RANGES BINARY DATA	BCD DATA
00	Seconds	0 - 59	00 - 3B	00 - 59
01	Seconds alarm	0 - 59	00 - 3B	00 - 59
02	Minutes	0 - 59	00 - 3B	00 - 59
03	Minutes alarm	0 - 59	00 - 3B	00 - 59
04	Hours:			
	12 hour format	1 - 12	01 - 0C	01 - 12 (A.M.)
			81 - 8C	81 - 92 (P.M.)
	24 hour format	0 - 23	00 - 17	00 - 23
05	Hours alarm:			
	12 hour format	1 - 12	01 - 0C	01 - 12 (A.M.)
			81 - 8C	81 - 92 (P.M.)
	24 hour format	0 - 23	00 - 17	00 - 23
06	Week day	1 - 7	01 - 07	01-07(Sunday=1)
07	Month date	1 - 31	01 - 1F	01 - 31
08	Month	1 - 12	01 - 0C	01 - 12
09	Year	0 - 99	00 - 63	00 - 99

Programming

All the information necessary for programming the real time clock is contained in the device data sheets from the manufacturer. The device can be accessed at memory address locations 070H and 071H. Address 070H is accessed first to select the desired register in the real-time clock. The most-significant bit of this address is the non-maskable interrupt bit. Be careful that you do not inadvertently disable this interrupt. Once the register has been selected, access the location at 071H to perform the read or write operation. The information in the following paragraphs highlights some of the more important points.

Initialization

When first attempting to program the device, check the status of the VRT bit in register D. If this bit is set, the data in the clock registers is valid and may be used immediately. If the bit is clear, you will have to initialize the device.

Start initialization by setting the SET bit in register B. This will disable updates while you are entering initial values into the clock registers. After setting your time options, formats, and initial values, all using the same data format, be sure to read register D prior to exiting your routine. This will set the VRT bit in that register, indicating that the current information is valid. Your final step should be to clear the SET bit in register B. Updates will begin one-half second after this bit is clear.

Once the device is initialized, it will perform updates according to the selected data mode. It is not possible to change the data mode without re-initializing the 10 data bytes of clock information.

The clock alarm bytes may be set several different ways. The following is a summary of these modes:

- Setting the alarm bytes to a specific time will cause a recurring alarm at that time each day.

- If a "don't care" condition (the two most-significant bits of the byte are set) exists in the hours byte, an alarm interrupt results each hour.

- If both the hour and minutes bytes contain a "don't care" condition, an alarm interrupt will occur each minute.

- If all three time bytes (hour, minute, and second) are set to "don't care", the alarm interrupt repeats each second.

Interrupts

The real time clock has three separate interrupt sources available to the system. It can generate an alarm interrupt, a periodic interrupt, or an update-ended interrupt. The control for these interrupts resides in three bits in register B.

The alarm interrupt enable bit (AIE) controls the alarm associated with the normal clock functions. When the three time bytes match the three alarm bytes and the AIE bit is set, an alarm interrupt will occur each second the condition is true. Setting the AIE bit allows the AF flag in register C to toggle when this condition is true, asserting the device IRQ line. An active condition on the device RESET line will clear the AIE bit.

The periodic interrupt enable bit (PIE) works with the PF bit in register C and the rate selection bits (RS0 - RS3) in register A. When enabled, this bit activates a recurring interrupt. The repetition rate is determined by the value selected by the rate selection bits. When the bit is set, the PF flag will toggle, asserting the IRQ line at the selected rate. Clearing the PIE bit prevents the IRQ line from becoming active but the PF flag will still toggle at the selected rate. Like the AIE bit, the PIE bit will be cleared when the RESET line is active.

The update-enabled interrupt bit (UIE) is the last of the three interrupts. If this bit is set, it enables the UF flag in register C, asserting the device IRQ line. This bit is set by the real time clock at the end of each update cycle. This bit may be cleared by activating the RESET line or by setting the SET bit in register B.

Each of the interrupts above operates independently of the other interrupt bits in the B register, but they are associated with their corresponding flag bits in register C. Each interrupt may operate in the flag mode or the fully enabled mode. The difference in their operation relates to the activation of the IRQ line.

One precaution is necessary when working with the interrupts. Whenever register C is read, all flag bits are cleared. If more than one interrupt is pending, it could be lost. Any program that has more than one interrupt enabled should check all the flag bits. If a program needs to know if the real-time clock generated an interrupt, it should check the IRQF bit in register C. This bit is set whenever the IRQ line goes active. Like the other flag bits in this register, a read of register C will clear this bit.

Update Cycle

The update cycle for the real time clock occurs once each second and lasts for 1984 µs. During this time the data in the timekeeping registers is not available to the system. It is possible for a program which randomly reads the time and date information to find that this data is not available. Three separate methods are recommended to handle this situation.

One method uses the update-ended interrupt. This interrupt goes active at the end of each update cycle, if it is enabled. When this bit is set, approximately 999 ms are available in which to read the time and date. A procedure that uses this method should clear the IRQF bit in register C before leaving the interrupt service routine.

Another method takes advantage of the update-in-progress bit (UIP) in register A. The UIP bit will go active whenever an update cycle is in progress. After this bit is set, the next update cycle will begin in 244 µs. If a program finds that this bit is clear, then a minimum of 244 µs are available to read data. If the bit is set, the time and date information is not valid.

The final method uses the periodic interrupt to determine if an update cycle is in progress. If a periodic interrupt occurs at a rate greater than once each 2228 µs, sufficient time is available to read time and date information at each periodic interrupt. The read must be completed within a set time factor. This time factor is the periodic interrupt time interval (refer to Table 14-17) plus 244 µs. If this factor is exceeded, the data will not be valid.

Memory

The real time clock contains 114 bytes of static RAM that are available to the system. In this computer, that memory is reserved for use by the Monitor ROM Setup/Configuration program.

The Setup/Configuration program configures the computer to your requirements. (Refer to Chapter 5 for specific details.) The information is retained in the static RAM so that it can be recalled each time the computer is powered-up.

CAUTION

If you attempt to access this memory, you could unintentionally erase memory information or cause alterations that could damage the computer. At the very minimum, you may change the configuration so that you will need to re-run the SETUP program before you can use the computer again.

Table 14-25 lists the pin descriptions of the 82C206 Integrated Peripheral Controller and Figure 14-8 is an illustration of the pinout.

Table 14-25. 82C206 Integrated Peripherals Controller Pin Descriptions

PIN NUMBER	NAME	FUNCTION
1	IRQ5	Interrupt request line 5, see Note 1.
2	IRQ4	Interrupt request line 4, see Note 1.
3	IRQ3	Interrupt request line 3, see Note 1.
4	IRQ1	Interrupt request line 1, see Note 1.
5-11	A23-A17	DMA page register address lines. These are the upper 7 bits of the page register.
12	Vss	Power ground.
13	A16	DMA page register address line. This line is used only for 8-bit transfers (channels 0-3)
14	PWRGD	Power good. This pin must be high during the bus cycles that the CPU accesses the real time clock . When this pin is low, all address, data, strobe, and read/write signals are disconnected from the CPU.

Table 14-25 (continued). 82C206 Integrated Peripherals Controller Pin Descriptions

PIN NUMBER	NAME	FUNCTION
15	$\overline{\text{PSRSTB}}$	This input is used to establish the condition of the control registers at power-up. In this computer, it is tied to the battery back-up circuit.
16	$\overline{\text{INTA}}$	Interrupt Acknowledge. This pin enables the interrupt controllers to put their data on to the data bus after two interrupt acknowledge pulses from the CPU.
17	TEST	This pin is tied low.
18	RESET	Reset. This pin clears registers in the DMA and interrupt controllers, assigns IRQ0 the highest priority, sets the slave interrupt controller address to 7, and disables special mask mode in the interrupt controllers.
19	OUT2	Timer 2 output. This pin drives the speaker.
20	OUT1	Timer 1 output. This timer output is a programmed a 15 micro second period signal for the interrupt request refresh cycles.
21	SCLK	Clock input. This clock generates the timing signals for DMA operations.
22	GATE2	Counter 2 gate input. This counter is driven by bit 0 of I/O port 61H to generate the speaker tone.
23	TMRCLK	Timer clock. Clock input for timers 0 - 2.
24-31	XD7-XD0	Tri state bi-directional data lines connected to the system data bus.
32	VCC	+5 VDC.
33	$\overline{\text{ACK}}$	Acknowledge. When this pin is high, it enables the chip select function for one of the following: DMS controller, DMA page register, interrupt controller, timer, real time clock, or the configuration register. When this pin is low, the 82C206 is essentially disconnected from the system bus.

Table 14-25 (continued). 82C206 Integrated Peripherals Controller Pin Descriptions

PIN NUMBER	NAME	FUNCTION
34-43	XA9-XA0	Address bus, during program condition, these lines are used to address the configuration register and the various internal registers of the chip.
44-47	DREQ0-DREQ3	DMA request lines. See Note 2.
48-51	DACK0-DACK3	DMA acknowledge lines. See Note 3.
52	$\overline{\text{XIOW}}$	I/O Write. This is a control signal used by the CPU to write information into the internal registers of the 82C206.
53	Vss	Power supply ground.
54	$\overline{\text{XIOR}}$	I/O read. This is a control signal used by the CPU to read the internal registers of the 82C206.
55-57	DACK7-DACK5	DMA acknowledge lines. See Note 3.
58-60	DREQ7-DREQ5	DMA request lines. See Note 2.
61	$\overline{\text{DMAMEMR}}$	DMA memory read. During DMA read or memory-to-memory transfers, this signal is used to access data from the selected memory location.
62	$\overline{\text{DMAMEMW}}$	DMA memory write. During DMA write or memory-to-memory transfers, this signal is used to write data to a selected memory location.
63	$\overline{\text{AEN8}}$	Address enable for 8-bit DMA transfers. Enables address bits A8 - A15 onto the address bus.
64	$\overline{\text{AEN16}}$	Address enable for 16-bit DMA transfers. Enables address bits A9 - A16 onto the system address bus.
65	ADSTB16	Address strobe for 16-bit DMA transfers (channels 5 - 7). This signal controls the latching of address byte A9 - A16.

Table 14-25 (continued). 82C206 Integrated Peripherals Controller Pin Descriptions

PIN NUMBER	NAME	FUNCTION
66	ADSTB8	Address strobe for 8-bit DMA transfers (channels 0 - 3). Controls the latching of address byte A8 - A15 for 8-bit peripherals.
67	TC	Terminal count. When the terminal count for any channel is reached, except channel 0 in memory-to-memory transfers, the DMA controller will generate a pulse. When the TC pulse occurs, the DMA controller will terminate service.
68	IOCHRDY	I/O channel ready. This signal extends the memory read and write pulses for the DMA controllers to accomodate slow memories or peripherals.
69	HRQ	Hold request. Requests control of the system bus when a DMA request is asserted.
70	INTR	Interrupt. Whenever an interrupt request is asserted, this signal goes active and interrupts the CPU.
71	AS	Address strobe. This is a positive pulse whose falling edge latches the address from the XD bus.
72	OSCI	Oscillator input. 32.768 KHz time base.
73	HLDA	Hold acknowledge. This signal orginates in the CPU and indicates it has turned over control of the system busses.
74	Vss	Power supply ground.
75	Vcc	+5 power.
76-82	IRQ15-IRQ9	Interrupt request lines. See Note 1.
83	IRQ7	Interrupt request line. See Note 1.
84	IRQ6	Interrupt request line. See Note 1.

Note 1: Interrupt request lines 1 - 15. These asynchronous input lines request an interrupt when they are raised to a high state and held there until the CPU sends an interrupt acknowledge.

Table 14-25 (continued). 82C206 Integrated Peripherals Controller Pin Descriptions

Note 2: DMA request lines are individual asynchronous inputs used by peripheral circuits to obtain DMA service. DACK will acknowledge the presence of DREQ signals. Polarity of DREQ is programmable. Reset initializes these lines to active high. DREQ will not be recognized while the clock is stopped, and must be maintained until the corresponding DACK line goes active. DREQ0-DREQ3 support 8-bit transfers between 8-bit I/O and 8 or 16-bit system memory. DREQ5-DREQ7 support 16-bit transfers between 16-bit peripheral and 16-bit system memory. DREQ4 is not available since it is used to cascade DREQ0-DREQ3.

Note 3: DMA acknowledge. Acknowledge is used to notify the individual peripherals when one has been granted a DMA cycle. The active polarity of these lines is programmable, but in this computer they must be programmed as active low. Reset initializes them as active low.

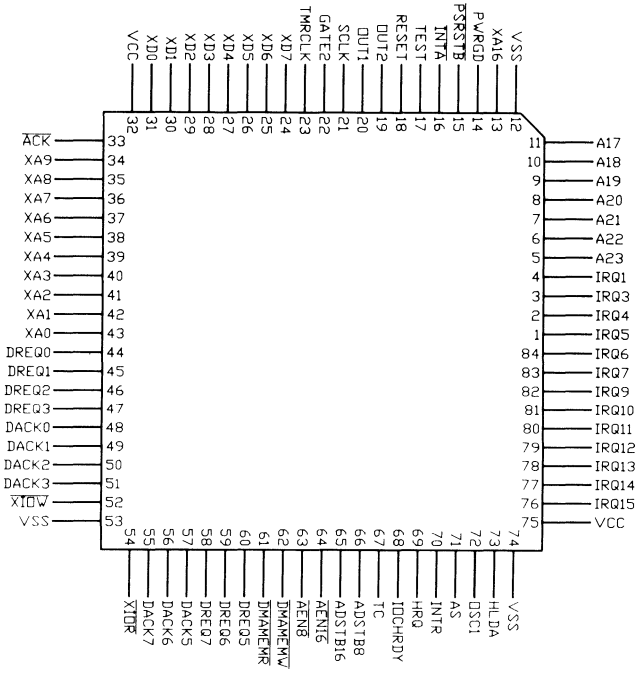


Figure 14-8. 82C206 Integrated Peripheral Controller Pinout

System Control Processor

The system control processor (SCP) is a single dedicated processor that performs a number of functions in the computer, including:

- Keyboard interface
- Reset the CPU
- Control access to slushware
- Control keyboard LED's
- Determine battery level and control charging
- Control hard drive power
- Monitor unit temperature
- Control backlight operation.

The SCP can also perform special enhanced functions that are not available in other PC-compatible systems. The programmer should be aware that programs that use these special Zenith functions may not operate properly on other PC-compatible computers.

The SCP used in this computer is an 83C451 with discrete logic between it and the main bus. The logic allows the 83C451 to emulate an IBM AT.

Status Register (64H) — The status register is an 8-bit read only register located at I/O port 64H. It can be read at any time, and the bits are defined in Table 14-26.

Table 14-26. SCP Status Register Bit Definitions

BIT	DEFINITION
0	Output buffer full. When this bit is set, the output buffer is full. The bit will be reset to 0 when the buffer is read. The output buffer should not be read until this bit is set.
1	Input buffer full. When this bit is set, the buffer contains data that the SCP has not read yet. When the SCP reads the buffer the bit will be reset to 0.
2	System flag. The system monitors this bit during reset. When the bit is clear (0), it indicates that the reset was caused by a normal powerup condition. After successful completion of the internal self test, the SCP sets this bit to 1.
3	Command/data. The SCP uses this bit to determine whether the information in the input buffer is a command or data. If the bit is set, the SCP interprets the byte as a command. If the bit is clear, the SCP considers the byte as data.
4	Inhibit switch. If this bit is set, the keyboard is enabled. This bit is updated whenever data is available in the output buffer.
5	Transmit time-out. The SCP sets this bit if it starts a transmission that it cannot complete properly.
6	Receive time-out. The SCP sets this bit when the keyboard controller fails to complete a transmission within the programmed delay period.
7	Parity error. This bit indicates the parity of the last byte received from the keyboard. A 1 indicates even parity; a 0 indicates odd parity. The keyboard transmits with odd parity.

Data Output Buffer (60H) — This is an 8-bit read-only register located at 60H and is used by the SCP to return scan codes received from the keyboard, and data bytes requested by commands, to the system. The output buffer should be read only when the output-buffer-full bit in the status register is set to 1.

Command Input Buffer (64H) — This is an 8-bit write-only register located at 64H and can transmit a command to the SCP. When a write to this register occurs, the Command/data bit in the Status register is set to 1. This register should only be written when the Input buffer full bit in the status register is set to 0.

Data Input Buffer (60H) — This is an 8-bit write-only port located at 60H. Normally, it is used to communicate with the keyboard, unless the SCP is expecting a data byte following a controller command. Data should be written to the controller's input buffer only if the input buffer full bit in the status register is 0.

SCP Commands

Whenever a write to I/O port address 64H occurs, the SCP expects to receive a valid command instruction. There is a defined set of instruction codes the SCP recognizes. Table 14-27 describes these commands.

Table 14-27. SCP Command Codes

COMMAND	DESCRIPTION
20H	Read keyboard controller's command byte. The keyboard controller will place the current command byte in the output buffer.
60H	Write keyboard controller's command byte. This command places the next byte written to the data input buffer in the keyboard controller's command byte. Refer to Table 14-24 for the bit definition of the command byte.
AAH	SCP self-test. On receiving this command, the SCP will perform internal diagnostics. If no errors occur, the SCP places 55H in the output buffer.

Table 14-27 (continued). SCP Command Codes

COMMAND	DESCRIPTION
ABH	SCP interface test. On receiving this command, the SCP will perform tests on the keyboard clock and data lines. One of the following results will appear in the output buffer: 00 — No errors detected. 01 — Keyboard clock low. 02 — Keyboard clock high. 03 — Keyboard data low. 04 — Keyboard data high.
ACH	SCP dump. The SCP will return 16 bytes of data, in scan code format, to the system. The data consists of SCP memory, the input and output port status, and the SCP status word.
ADH	Disable keyboard. The SCP will set bit 4 in the command byte, forcing the clock line low. Data cannot be sent or received.
AEH	Enable keyboard. This command clears bit 4 of the command byte. This enables the keyboard interface.
B7H	Enable slushware write. This command allows the programmer to access the slushware memory area. This command must be followed with a three-byte instruction sequence to enable writes. The required sequence is 76H, 92H, and 04H. If this sequence is altered or not supplied unpredictable results may occur.
B8H	Disable slushware writes. This command returns the slushware to its normal write protected state. Like the previous command, a three-byte command sequence must follow the command. The required sequence is F8H, 4BH, and 69H.
B9H	Return SCP version number. This command returns a one-byte BCD value to check the device version number. For example, if the device returns the value 22H, then the version number is 2.2.

Table 14-27 (continued). SCP Command Codes

COMMAND	DESCRIPTION
BAH	Process Zenith extended command. This signals the SCP that there is a Zenith extended command being written to the data input buffer. The currently supported Zenith extended commands are listed in Table 14-28.
C0H	Read input port. This commands the SCP to read its input port and place the data in its output buffer. Should only be used if the output buffer is empty.
D0H	Read output port. This commands the SCP to read its output port and place the data in its output buffer. Should only be used if the output buffer is empty.
D1H	Write output port. Must be followed by a byte written to the data input buffer which is then placed in the controller's output port. Do not write bit 0 low in this operation since it is directly connected to the fast CPU reset signal and it would reset the CPU.
E0H	Read test inputs. This command causes the controller to read its T0 and T1 inputs. The data is placed in the output buffer as bit 0 representing T0 and bit 1 representing T1.
F0H-FFH	Pulse output port. The SCP is capable of pulsing bits 0 - 3 low for approximately 6 microseconds. Bits 0 - 3 of this command correspond to the bits of the output port. A bit is pulsed by placing a zero in that bit location. Since bit 0 of the controller's output port is connected to the system reset, pulsing this bit will reset the microprocessor.

Table 14-28. SCP Command Byte Description

BIT	DESCRIPTION
0	Enable output-buffer-full interrupt. If set, the SCP will generate an interrrupt when it places data in the output buffer.
1	Reserved, should be written as a zero.
2	System flag. The SCP places the value of this bit in the system flag bit of the status register.
3	Inhibit override. Setting this bit enables normal keyboard operations.
4	Disable keyboard.
5	Personal computer mode. Setting this bit causes the SCP to change modes. It will not check parity or convert scan codes in this mode.
6	Compatibility mode. When this bit is set, the SCP will convert scan codes from the keyboard controller into codes used by PC-compatible computers.
7	Reserved, should be written as a zero.

Table 14-29. Zenith Extended Commands

COMMAND	DESCRIPTION
00H	Turn on power to hard disk.
01H	Turn off power to hard disk.
02H	Write word to auxiliary CMOS. This is followed by writing 3 bytes to the data input buffer, the first is the address for the word, the next two are the LSB and then MSB of the data word.
03H	Read word from auxiliary CMOS. Should be followed by a byte written to the data input buffer which is the address in CMOS to read. The word read will be available at the data output buffer.
05H	Set display backlight timeout. A byte containing the value (in minutes) of delay before the backlight turns off should be sent to the data input buffer after this command. If this value is set to 0FFH, the backlight will be permanently on, if 000H, the backlight is disabled. The backlight can also be controlled by commands 0EH and 0FH.

Table 14-29 (continued). Zenith Extended Commands

COMMAND	DESCRIPTION
06H	Enable modem.
07H	Disable modem.
08H	Set 80387 native mode. The bypass line to the 80387 interface control is set.
09H	Set 80387 compatible mode. The bypass line to the 80387 interface control is cleared.
0AH	Enable video. Enable internal video memory and I/O accesses.
0BH	Disable video. Disable internal video memory and I/O accesses.
0CH	Select font RAM. Directs video memory accesses to/from font RAM.
0DH	Select video RAM. Directs video memory accesses to/from display RAM.
0EH	Turn backlight on.
0FH	Turn backlight off.
10H	Select internal (LCD) display.
11H	Select external (RGB) display.
12H	Enable 1M expansion memory.
13H	Disable 1M expansion memory.
14H	EMS memory select. Designates 2nd megabyte and expansion memory as EMS base memory.
15H	Extended memory select. Designates 2nd megabyte and expansion memory as extended memory.
16H	Enable 80387.

Table 14-29 (continued). Zenith Extended Commands

COMMAND	DESCRIPTION
17H	Disable 80387.
18H	Select burst mode memory refresh.
19H	Select single mode memory refresh.
1AH	Return CMOS pass/fail status. Read the data output buffer for status of CMOS: 0 — CMOS pass. 1 — CMOS checksum failure. 2 — CMOS device failure.
1BH	Update CMOS checksum. Recalculates the CMOS checksum and updates the status. No parameters are returned.
1CH	Enable 101-key keyboard emulation. Only applicable if a 101-key keyboard is attached. Blocks access to LED's on unit keyboard.
1DH	Enable ZX-BUSS configuration.
1EH	Disable ZX-BUSS configuration.
1FH	Return ZX-BUSS status. The data ouput buffer will contain ZX-BUSS information: 0 — Buss not present. 1 — Buss present.
20H	Read byte from auxiliary CMOS. The address of the byte to read in auxiliary CMOS must be sent to the data input buffer after this command. The byte read is then available at the output buffer.
21H	Write byte to auxiliary CMOS. This command must be followed by two bytes written to the data input buffer, the first is the address in CMOS to write the byte, the second is the value to write.

Table 14-29 (continued). Zenith Extended Commands

COMMAND	DESCRIPTION
22H	Return analog system parameters. Returns three bytes in the following order; current average battery voltage, current average CPU temperature, and a powre source flag (1 - AC power, 0 - DC power).
23H	Load SCP slushware. After this command is issued, all bytes written to the data input buffer are stored in the SCP RAM. The bytes are stored sequentially starting at address 2000H. This process will continue until the End slushware load command is received. At that point, the SCP will begin executing the slushed code.
24H	End slushware load.
29H	Re-initialize Zenith extended control ports. This command causes the SCP to read all system parameters from auxiliary CMOS and update the control ports accordingly.

Keyboard Control

The SCP receives serial data from the keyboard, checks parity, translates the scan code and makes this data available to the output buffer. Each time a byte is available in the output buffer, the SCP will transmit an interrupt request for system service. The SCP can also send data to the keyboard. This is done for control of repeat and delay rates, error re-transmission, and the LED indicators on the keyboard bezel.

Receiving Keyboard Data

Data received from the keyboard consists of 11 bits in the format illustrated in Figure 14-9. When transmission is complete, the keyboard controller disables the interface until the system accepts the transmitted byte. If the received byte contains parity errors, the SCP will have the keyboard controller retransmit the invalid byte. The SCP indicates a received parity error to the system by placing the value FFH in the output buffer and setting the parity bit in the status register. The SCP can also time-out a data transmission from the keyboard controller. If a data transmission is

not completed within two milliseconds, the SCP will place FFH in the output buffer and set the receive time-out bit in the status register.

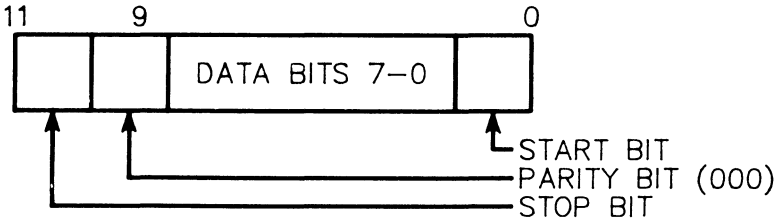


Figure 14-9. Keyboard Data Format

Sending Keyboard Data

Data sent to the keyboard by the SCP uses the same format as that illustrated in Figure 14-9. The data is sent in serial mode with an odd parity bit inserted. The keyboard controller must start clocking the data out of the SCP within 15 milliseconds and complete transmission within 2 milliseconds. If not, the SCP will place an FEH in the output buffer and set the transmit time-out error bit in the status register. The keyboard controller must acknowledge each transmission before another byte can be sent. If a parity error exists, the SCP will place a FEH in the output buffer and set both the transmit time-out and parity error bits in the status register. If the keyboard controller does not respond within 25 milliseconds to the SCP transmission, the SCP will place FEH in the output buffer and set the transmit and receive time-out error bits in the status register. By transmitting data to the keyboard controller the SCP controls error re-transmission, repeat and delay rates, and the keyboard LED indicators.

Slushware

Slushware refers to a 128K area of dynamic memory that stores the system ROM code. Instructions stored dynamically can be read much faster by the system CPU.

When the system initializes, some types of firmware are copied to this area of memory. Zenith products recognize and make use of the slushware area. This can result in significant performance improvements during operation.

The SCP provides a method for the programmer to access the slushware to change or alter the contents of this memory region. Two SCP commands, B7 and B8, referred to in Table 14-23, control access.

The slushware area is normally write-protected to prevent accidental alteration of the information stored there. SCP command B7 allows the programmer to perform writes into the slushware area. These memory writes operate in the same manner as writes to other portions of system memory. SCP command B8 will return the slushware to its normal write-protected state. The commands must be issued in the format described in Table 14-23.

CPU Protected Mode

The SCP assists the programmer in switching the 80386 to protected mode. It manages this by controlling the GATEA20 line in the system hardware.

When the 80386 initializes, it begins operation in real mode. In this mode the processor acts as if it were a very fast 8088. In order to switch to protected mode, the GATEA20 line must be active. This is accomplished by issuing the write output port command (D1) to the SCP. Follow this command with a byte of data having bit 1 set to enable the GATEA20 line. In order for this procedure to work correctly, the program must wait to maker sure the line is active. If a program does not wait a sufficient amount of time before attempting to enter protected mode, the results will be unpredictable.

SCP Pinout

Table 14-30 provides a detailed description of the signals associated with the SCP in this computer. Figure 14-10 shows the device pinout.

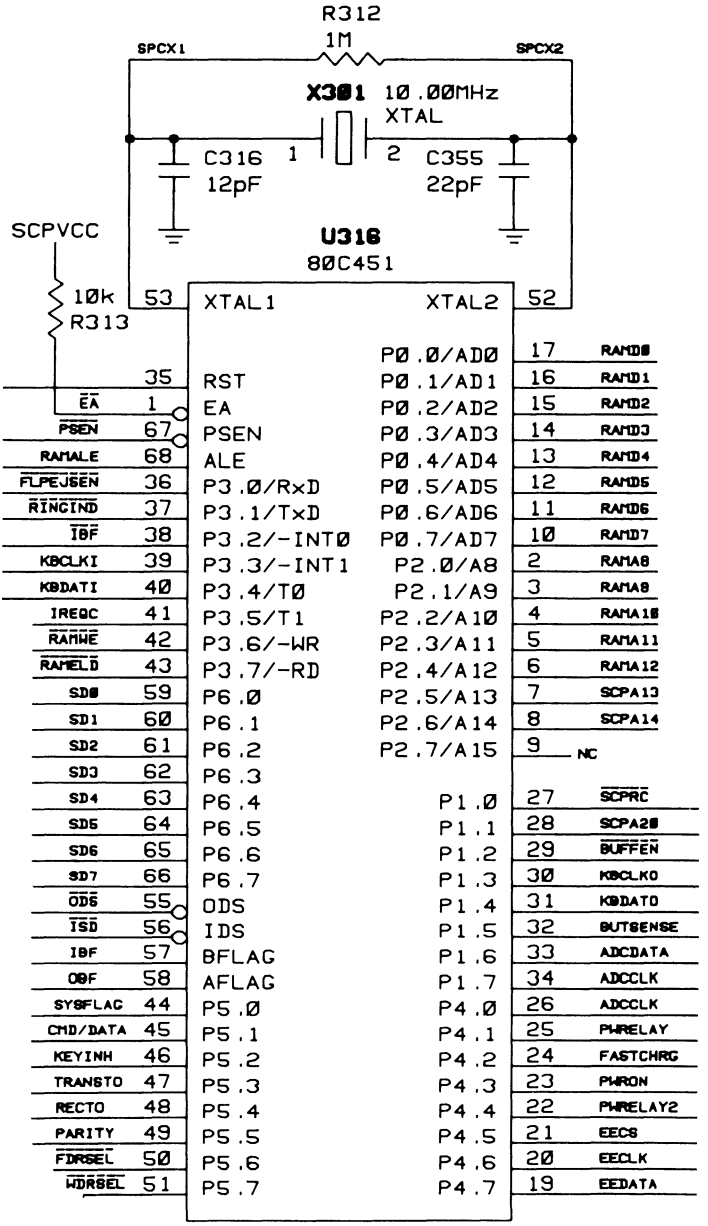


Figure 14-10. SCP Pinout

Table 14-30. 80C451 SCP Pinout

PIN NUMBER	SIGNAL NAME	DESCRIPTION
1	$\overline{EA}$	Instruction execution control. When held high, the CPU executes out of internal program memory, unless the program counter exceeds 0FFFH. When held low, the CPU executes out of external program memory.
2-9	P2.0 - P2.7	Port 2. An 8-bit bidirectional I/O port with internal pull-ups. Port 2 can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups.
10-17	P0.0 - P0.7	Port 0. An 8-bit, open drain, bidirectional I/O port. Port 0 is also the multiplexed low-order address and data bus during accesses to external memory, and outputs instruction bytes during program verification. External pull-ups are required during program verification. Port 0 can sink/source eight LS TTL inputs.
18	Vcc	Power supply, +5V.
19-26	P4.0 - P4.7	Port 4. A 4- or 8-bit bidirectional port with internal pull-ups. Port 4 can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups.
27-34	P1.0 - P1.7	Port 1. An 8-bit bidirectional I/O port with internal pull-ups. Port 1 reveives the low-order address bytes during program verification.
35	RESET	A high level on this pin for 2 machine cycles while the oscillator is running resets the device.
36	RXD (P3.0)	Serial input port.
37	TXD (P3.1)	Serial output port.

Table 14-30 (continued). 80C451 SCP Pinout

PIN NUMBER	SIGNAL NAME	DESCRIPTION
38	$\overline{\text{INT0}}$ (P3.2)	External interrupt 0.
39	$\overline{\text{INT1}}$ (P3.3)	External interrupt 1.
40	T0 (P3.4)	Timer 0 extrnal input.
41	T1 (P3.5)	Timer 1 external input.
42	$\overline{\text{WR}}$ (P3.6)	External data memory write strobe.
43	$\overline{\text{RD}}$ (P3.7)	External data memory read strobe.
44-51	P5.0 - P5.7	Port 5. An 8-bit I/O port with internal pull-ups. Port 5 can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups.
52	XTAL2	An output of the inverting amplifier that forms the oscillator. This pin should be floated when an external oscillator is used.
53	XTAL1	An input of the inverting amplifier that forms the oscillator. This input receives the external oscillator when an external oscillator is used, such as in this implementation.
54	Vss	Ground
55	$\overline{\text{ODS}}$	Port 6 control line: Output data strobe.
56	$\overline{\text{IDS}}$	Port 6 control line: Input data strobe.
57	BFLAG	Port 6 control line: A bidirectional I/O pin with internal pull-ups.
58	AFLAG	Port 6 control line: A bidirectional I/O pin with internal pull-ups.

Table 14-30 (continued). 80C451 SCP Pinout

PIN NUMBER	SIGNAL NAME	DESCRIPTION
59-66	P6.0 - P6.7	Port 6. A specialized 8-bit bidirectional I/O port with internal pull-ups. This port can sink/source three LS TTL inputs and drive CMOS inputs without external pull-ups. Port 6 can be used in a strobed or non-strobed mode of operation, and in conjunction with the four control lines.
67	$\overline{\text{PSEN}}$	Program store enable. This output is the read strobe to external program memory. $\overline{\text{PSEN}}$ is activated twice each machine cycle curing fetches from external program memory; however, when executing out of external program memory, two activations of $\overline{\text{PSEN}}$ are skipped during each access to external data memory. $\overline{\text{PSEN}}$ is not activated during fetcches from internal program memory. $\overline{\text{PSEN}}$ can sink/source eight LS TTL inputs and drive CMOS inputs without an external pull-up.
68	ALE	Address latch enable. An output for latching the low byte of the address during accesses to external memory. ALE is activated at a constant rate of one-sixth the oscillator frrequency except during an external data memory access, at which dtime one ALE is skipped. ALE can sink/source eight LS TTL inputs and drive CMOS inputs without an external pull-up.

Special Programming Features

In addition to the programmable functions described throughout this chapter, there are a number of program accessible features that are unique to this system. These features allow the programmer to access special modes or control ports that can enhance the performance of the system. Because these features may not be

available on other compatible systems, programs that use them may not perform in the same manner on other systems.

Scratchpad RAM Enable

Access to the scratchpad RAM is available to the programmer through a special hardware I/O port. If a program performs an I/O write to port 0FBH, writes to the scratchpad RAM will be enabled. An I/O write to port 0F9H will disable writes to the scratchpad RAM.

CAUTION

Be very careful when writing to the scratchpad RAM. Several important system variables are maintained and updated in this area. If a program writes to this area without taking precautions, it is possible to damage the system or halt the computer.

NMI Enable

A special port address exists to enable non-maskable interrupts within the system. If bit 7 at port address 070H is set (1), this feature will be enabled, and will report non-maskable interrupts during system parity errors or if the bus I/O channel check line is active. Clearing the bit at this address will disable this feature.

Chapter 15
Memory

This computer contains two megabytes of RAM contained in sixteen 256K × 4 dynamic RAM chips for data and four 1M × 1 dynamic RAM chips for parity. Expansion memory of one megabyte is available with an optional module. The expansion module can be set as extended memory (addressable through the 80386 protected mode) or expanded memory (EMS). Refer to the memory map in Figure 15-1.

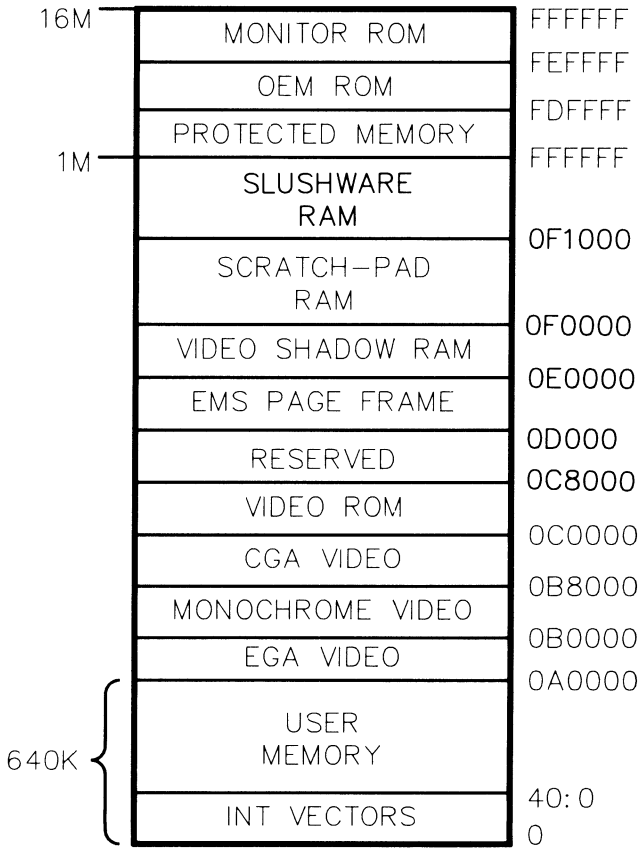


Figure 15-1. Memory Map

Access to extended memory can only occur through the 80386's protected address mode. Hardware support for EMS memory is provided through a fixed window in address range segment D (D0000H-DFFFFH). The CPU addresses segment D and software causes the contents of specific EMS memory locations to appear. EMS memory access requires a special software driver.

Slushware memory (F000H:1000H) is dynamic memory that stores the system ROM code, and 60K bytes are allocated for this purpose. Instructions stored in dynamic memory execute much faster. Note that access to slushware memory (addressed at F000H:1000H) is not slowed.

User Memory

User memory includes base memory up to 640K, extended memory beyond 1 megabyte, and EMS memory. Slushware memory is automatically loaded and write-protected at power-up. Most user memory programming can be performed directly from assembly or machine language programs and includes the following:

- Reading and writing data to specific memory locations.
- Allocating or reserving specific locations in memory for specific purposes, such as for software interrupt routines or storing user-defined character fonts.
- Storing character strings and numeric values that are widely used by a number of applications.
- Rerouting interrupts to user-defined routines.
- CPU stack operations
- Moving memory contents from one area to another.
- Using the contents of RAM to control video graphics.

Address Format

The 80386 uses a much more involved method of address computation than the earlier 8088 microprocessor. Because of the computation methods used, programming situations may arise where it is extremely difficult to identify the actual memory being addressed. Since special translation units within the 80386 control this address calculation, this should not be a critical programming issue.

The earlier 8088 used a 2-part number to designate specific memory locations. The hexadecimal number consists of a 4-digit number to identify the segment address and a 4-digit number to identify the offset address within that segment. The format for this value is XXXX:YYYY.

The first four digits actually represent a 5-digit hexadecimal memory address; the system performs an imaginary shift left (multiply by 16) on the value to arrive at the RAM bank and row to select. The second value selects the RAM column from the selected bank. For example, the value 3F3F:5B11 represents memory address 44F01H. 3F3FH shifted left equals 3F3F0H. 5B11H is the offset added to the segment address. The result of adding 3F3F0H and 5B11H is 44F01H.

In the 80386 the concept of the segment and offset have remained, but actual computation methods have changed. The 80386 refers to three distinct address spaces, logical, linear, and physical. Figure 15-2 illustrates the address translation process.

The logical address (referred to as the virtual address) consists of a selector and an offset. The selector is a value contained within a segment register. The offset (referred to as the effective address) is composed of a combination of three different values, the base, the index, and the displacement. How these values are combined depends on which one of 11 different addressing modes is in use. The resulting virtual space is the area the programmer will most often encounter and use.

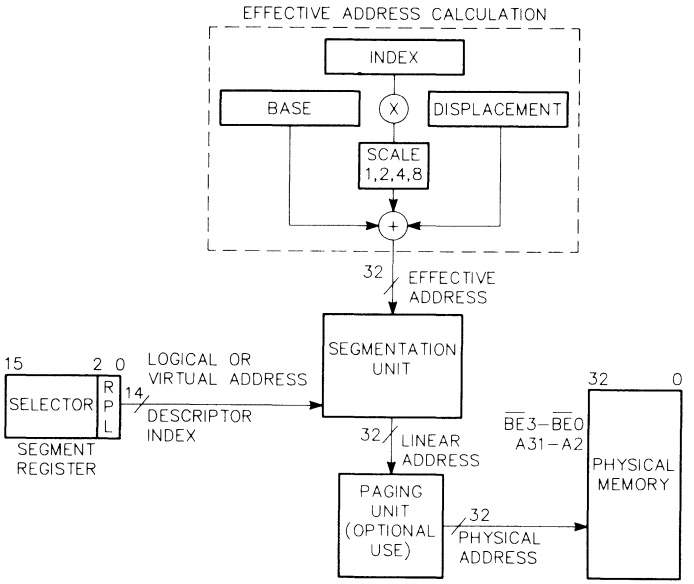


Figure 15-2. 80386 Address Translation Process

A special segmentation unit within the 80386 processes the logical address information to obtain the linear address. The control for this process is the processor mode. The 80386 can operate in real mode or protected mode, and each mode has a different process to determine the linear address.

In real mode, the segmentation unit shifts the selector value left four bits and adds the result to the offset to form the linear address. If the internal paging unit is not enabled the linear address becomes the physical address in memory. (This method duplicates exactly the method used in the 8088 processor.) If the internal paging unit is enabled, it performs an additional translation on the linear address to determine the physical address.

In protected mode, each selector value is associated with its own linear base address. These values are stored in special descriptor tables within the 80386. The selector's linear base address is added to the offset to form the final linear address. The internal paging unit will operate the same way as in real mode on this linear address.

Address Decoding

The memory address decoding in this computer is handled by special Programmed Array Logic (PAL) devices. The row address select (RAS), column address select (CAS), and MA0-MA9 address lines are used to select a particular location within the system memory.

A specific memory location is accessed through two distinct phases: the row-access phase and the column-access phase. The row-address and column-address signals are independently transferred to the memory device during each phase.

After the row-address is valid on the bus, the row-access phase occurs. When the particular row of memory devices has been selected, the column-access phase is initiated. During the column-access phase any of the memory columns within the enabled row can be read.

There are three types of memory cycles: memory read, memory write, or refresh. Any cycle can be requested by more than one device. Contention, timing, and row and column address selection is governed by the address control logic.

Refresh for a particular address is automatic whenever that location is read. During refresh cycles, memory rows in all pages are read simultaneously through the RAS and CAS signals, automatically refreshing the memory. By multiplexing the address, nine lines are used to address the first 640K of memory.

Memory Data Bus

Data interface between memory and the CPU is handled through a set of programmable logic arrays on the main board. During a given memory access (read or write), one 8-, 16-, or 32-bit segment, or one 8-bit segment of data can be transferred.

Each memory data line (D0-D31) from the CPU is tied to the bidirectional data input/output pin of one memory device. Data is either read from or written to the same memory location in each device during a memory cycle. A valid address on the memory address bus and the state of the WE (write enable) signal define the operation (high to read, low to write).

EMS Memory

This computer directly addresses the first megabyte of contiguous memory in its environment. Memory from 640K to the 1-megabyte address boundary has portions reserved for system ROM code, video controller ROM code, hard disk controller ROM code, and other functions. Memory from 1 megabyte to the 16-megabyte address boundary is termed extended memory and is directly accessible only through the protected mode of the 80386. In contrast, expanded memory (EMS) is page-mapped memory accessed through a complicated bank switching scheme in the 80386 real address mode.

Expanded memory is only useful to programs that support it. In other words, the program must instruct the hardware to implement the specific bank switching. The Lotus/Intel/Microsoft Expanded Memory Specification (LIM EMS) attempts to standardize the bank switching scheme. The LIM EMS contains numerous examples that an experienced assembly language programmer can interpret. In this way, different programs can be written to make use of expanded memory in a consistent manner. The specification is available free of charge from Intel Corporation and may be obtained by calling 1-800-538-3373.

This section briefly outlines the approach used in the LIM EMS to access expanded memory. As mentioned earlier, expanded memory is page-mapped. This means that, a small area of memory acts as a window through which a larger area of memory resides. This window is 64K wide and is composed of four 16K blocks of memory referred to as page frames. In this computer these page frames exist in segment D (address range D0000H - DFFFFH) of the reserved system address space (between 640K and 1 megabyte). The base-address of the 64K page frame must fall on a 16K boundary within the system address space as illustrated in Figure 15-3.

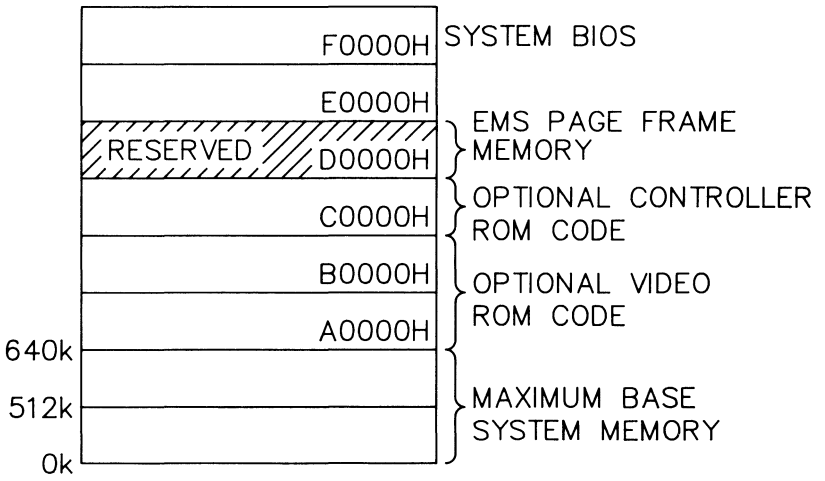


Figure 15-3. Conventional Memory

Although the EMS specification supports up to 8 megabytes of EMS base memory, this computer supports a maximum of two megabytes. EMS base memory is organized into 128 individual pages for each two megabytes, and each page frame is 16K wide. Because the CPU does not have direct addressing capabilities beyond 1 megabyte, it uses the page frames to access expanded memory (recall that the page frames reside within the addressable 640K to 1-megabyte memory range). Whenever the CPU addresses the memory range within system address space that is associated with a page frame, it accesses expanded memory. Since there are four contiguous page frames, 64K of expanded memory can be accessed without a page-swap occurring. Refer to Figure 15-4.

NOTE: Board 1 is always present in the computer, but only 256K of base memory is available. Board 2 is optional and may be used as EMS or extended memory. Boards 3 and 4 are available only with the use of the expansion box.

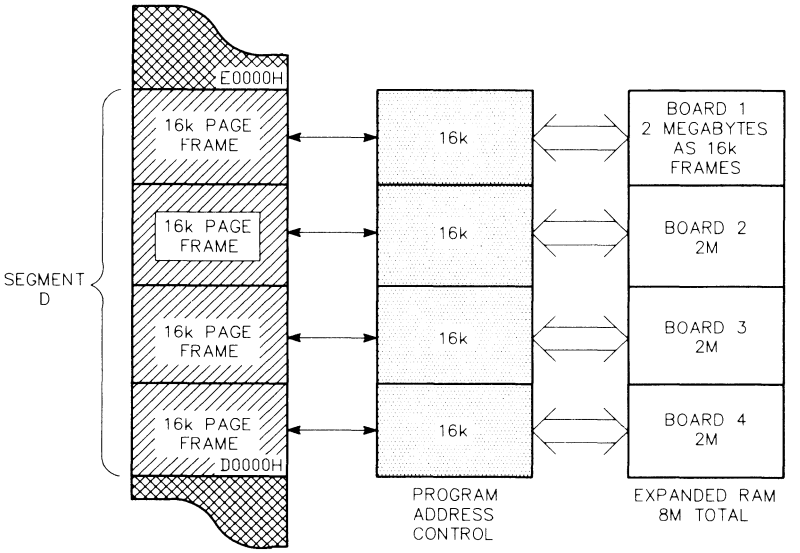


Figure 15-4. Expanded Memory

NOTE: The page pointing arrangement illustrated in Figure 15-4 is not the only pointing configuration possible with the EMS memory. Many different arrangements are possible, depending on the needs of the software being used.

As the system CPU addresses the memory range of one of the page frames, the EMS memory page pointer will remain set to that frame. If the page frame changes or the page is turned on or off, then an 8-bit value is written to an I/O port to select the correct frame. The value placed in the port moves a specified 16K expanded memory page to the page frame. The information is thereby made local to the CPU.

Bits 0-6 of the page pointer byte define the particular page, while bit 7 is the page enable/disable bit. When bit 7 is set (logic high), the page is allowed to appear in the memory space; when bit 7 is clear (logic low), the page does not appear in the memory space. Refer to Table 15-1 for the EMS I/O addresses for this computer.

Table 15-1. EMS I/O Addresses

BOARD NUMBER	PAGE 0	PAGE 1	PAGE 2	PAGE 3
Board 1	0258H	4258H	8258H	C258H
Board 2	0268H	4268H	8268H	C268H
Board 3	0208H	4208H	8208H	C208H
Board 4	0218H	4218H	8218H	C218H

EMS address decoding up to the 1-megabyte boundary occurs as normal. Access to specific memory pages is a function of the program that is executing at the time. Memory refresh cycles occur for expanded memory in the same way as for base memory. Memory refresh is handled during the appropriate processor cycle and addressing is manipulated so the refresh cycle includes expanded memory. Data is transferred between the CPU and EMS memory in the same way that data is interfaced to base memory, but at 8 MHz 1 wait state speeds.

Slushware Memory

The term "slushware memory" refers to a technique that combines the advantages of software and firmware for increased flexibility and speed. At system initialization, the Monitor ROM BIOS routines are downloaded into a reserved area of dynamic memory. Since the BIOS resides in RAM during normal operations, the routines execute much faster. To prevent the BIOS instructions from being corrupted by a user program, the slushware memory is write-protected.

This computer also supports additional user-specified system ROM, which can also take advantage of the slushware memory concept. A socket on the main board accepts an optional ROM device (frequently used by OEM groups) that gives an additional 64K for unique code applications. The code can also be downloaded to dynamic RAM from diskette.

Chapter 16

Mass Storage

The term "mass storage" applies to all forms of off-line storage including magnetic disk, magnetic tape, punched tape, and punched cards. This computer supports several types of mass storage devices including:

- 3.5-inch, 720K floppy disk drives
- 3.5-inch, 1.4M floppy disk drives
- 3.5-inch, 40M hard disk drives
- 3.5-inch, 100M hard disk drives

This chapter discusses the programmable elements of the mass storage system in this computer.

Supported Drives

The operating system and devices used in this computer support high-density 3.5-inch floppy disk drives and 3.5-inch hard disk drives. One high-density 3.5-inch drive is installed in the computer. One 3.5-inch, half-height, 40M or 100M hard disk drive is also installed.

Error Detection and Invalid Commands

Four error correction code (ECC) bytes are used for error detection and correction in the data fields. This permits correction of five bytes in the data fields. A cyclical redundancy check (CRC) byte is used for the ID field. Before calculating the checksums, all shift registers are preset to FH. All checksums begin with the respective address marks. The ECC polynomial is $X^{32} + X^{28} + X^{26} + X^{19} + X^{17} + X^{10} + X^6 + X^2 + 1$. The CRC polynomial is $X^{16} + X^{12} + X^5 + 1$.

If the floppy disk controller receives an invalid command, it is interpreted as a NOP (no operation) command. The controller goes into a standby or no operation state until a new command is issued. If the hard disk controller receives an invalid command, it issues an aborted command error. The error register must be read before the next command will be accepted by the controller.

Floppy Disk Control

The WD37C65A floppy disk controller (FDC) supports up to four disk drives. The controller makes record data transfers through the I/O bus and DMA circuitry and supports four data transfer rates. Write protect features are recognized and interrupts are used to indicate completion of an operation. The FDC supports FM and MFM disk encoding formats and can be operated in DMA and non-DMA modes.

DMA and Non-DMA Modes

This computer is factory-set to operate in DMA mode. In DMA mode, the processor loads a command into the FDC and the data transfer is carried out by the DMA and FDC controllers. In most cases, a processor interrupt is generated upon the completion of the data transfer.

In non-DMA mode, the FDC generates a processor interrupt for each data byte transferred. The processor must then load a

command into the FDC, allowing the single byte to be transferred. To restrict operation to non-DMA mode only, the $\overline{\text{DACK}}$ line (pin 5 of the WD37C65A) can be tied to Vcc. The specify command, described in the "Programming Floppy Disk Operations" section of this chapter, can also be used to operate in non-DMA mode.

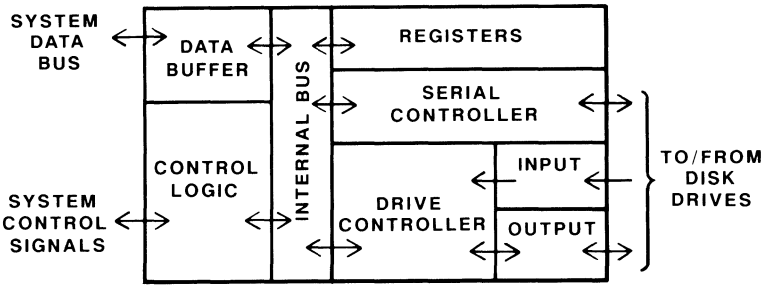


Figure 16-1. 765 Disk Controller Block Diagram

Master Control Logic

The master control logic oversees all operations within the FDC. The control signals it generates handle the data buffer, internal registers, serial controller, and the drive controller through the internal bus. It also handles interrupts, DMA transfers, request and acknowledge signals, and read and write operations.

Serial Controller

An internal data buffer transfers data between the internal bus and the system data bus. The internal bus passes information between the data buffer and the serial controller. The serial controller then reads or writes data from the disk and the registers. The serial controller sends data to the disk drives, converting the 8-bit parallel-format data into a serial form. The controller also converts serial-format data from the drive into an 8-bit parallel form. Using

timing control provided by the write clock signal, the controller makes sure that the data is correctly timed during all transfer operations. The serial controller also generates the synchronizing signals used by the data separator for timing.

Drive Controller

The drive controller monitors the activities of the input and output channels of the floppy disk controller. The input channel monitors and receives drive ready, write-protect, index, and track 0 signals. These signals go to the drive controller, which responds by sending signals to either the output channel or the master control logic of the floppy disk controller. The output channel generates seek, head load, head select, direction, and step signals for the drive. It also controls the recording mode (FM or MFM) used by the drive.

FDC Control Registers

Five registers control and provide status reports for floppy disk drive operations. The digital output register and floppy control register are write-only registers that provide command and operating parameter information to the FDC. The main status register is a read-only register that provides drive, FDC, and command status information. The floppy data register is a read/write register used to transfer data, command, and detailed status information between the FDC and the CPU. The digital input register is a read-only register primarily used for hard disk operations. Bit 7 of the digital input register reflects the state of the floppy disk changed signal. Table 16-1 lists the primary and secondary port addresses for these registers.

Table 16-1. Floppy Drive Control Register Port Addresses

PRIMARY PORT ADDRESS	SECONDARY PORT ADDRESS	READ OPERATION REGISTER	WRITE OPERATION REGISTER
3F2H	372H	Not used	Digital output
3F4H	374H	Main status	Main status
3F5H	375H	Floppy data	Floppy data
3F7H	377H	Digital input	Floppy control

Digital Output Register (Port 3F2H)

The digital output register controls the drive motors and drive selection, enables the disk interrupt (IRQ6) and DMA data request line (DRQ2), and resets the drives. Port address 3F2H is the PC-compatible address used for writing to this register. Table 16-2 describes the function of each bit in this register.

Table 16-2. Digital Output Register

BIT	FUNCTION	DESCRIPTION
0-1	Drive select bits 0 - 1	These bits select which drive is to be enabled. Refer to Table 16-3.
2	$\overline{765RESET}$	This bit, when cleared, resets the floppy disk control circuits and clears all internal registers.
3	Enable IRQ6 and DRQ2	This bit, when high, enables the IRQ6 (hardware floppy disk interrupt) and the DRQ2 (DMA request) lines.
4	Motor on 0	This bit asserts the $\overline{MTON0}$ and FDD ON signals to turn on the motor in drive A.
5	Motor on 1	This bit asserts the $\overline{MTON1}$ and FDD ON signals to turn on the motor in drive B.

Table 16-2 (continued). Digital Output Register

BIT	FUNCTION	DESCRIPTION
6	Motor on 2	This bit asserts the $\overline{\text{MTON2}}$ and FDD ON signals to turn on the motor in a third floppy drive.
7	Motor on 3	This bit asserts the $\overline{\text{MTON3}}$ and FDD ON signals to turn on the motor in a fourth floppy drive.

Table 16-3 defines the drive select bits (bit 0 and bit 1) of the digital output register. The appropriate motor on bit (bits 4 through 7) must also be set in order to select a drive. Use the information from Table 16-2 to set the appropriate motor on bit.

Table 16-3. Drive Select

BIT 0	BIT 1	SELECTED DRIVE
0	0	$\overline{\text{DRVSEL0}}$ — Selects disk drive A, the floppy disk drive mounted in the computer.
1	0	$\overline{\text{DRVSEL1}}$ — Selects a second floppy disk drive.
0	1	$\overline{\text{DRVSEL2}}$ — Selects a third floppy disk drive.
1	1	$\overline{\text{DRVSEL3}}$ — Selects a fourth floppy disk drive.

All eight bits of the digital output register are cleared (set to 0) when the computer is first turned on and each time an I/O reset occurs. Each bit is set to the corresponding input when the clock signal makes a transition from low to high. The clock signal for this register is developed by the chip select signal, address line 2, and the $\overline{\text{IOW}}$ signal.

Main Status Register (Port 3F4H)

The main status register is an 8-bit read-only register located at port address 3F4H. This register contains the status information for the floppy disk controller and may be read at any time. However, it is recommended that the CPU wait 12 µs each time it reads the main status register. The wait allows the data direction and request for master bits to change and settle. Each bit in the main status register is described in Table 16-4.

Table 16-4. Main Status Register

BIT	FUNCTION	DESCRIPTION
0	Floppy drive 0 busy	This bit is set (1) when disk drive A is busy (in the seek mode). The controller will not accept read or write commands for any disk drive.
1	Floppy drive 1 busy	This bit is set (1) when disk drive B is busy (in the seek mode). The controller will not accept read or write commands for any disk drive.
2	Floppy drive 2 busy	This bit is set (1) when disk drive C is busy (in the seek mode). The controller will not accept read or write commands for any disk drive.
3	Floppy drive 3 busy	This bit is set (1) when disk drive D is busy (in the seek mode). The controller will not accept read or write commands 1for any disk drive.
4	Controller busy	This bit is set (1) when a read or write command is in process. The controller will not accept any other command.
5	Execution mode	This bit is set (1) during the execution phase of a non-DMA operation. The bit operates only in non-DMA mode.

Table 16-4 (continued). Main Status Register

BIT	FUNCTION	DESCRIPTION
6	Data direction	This bit is set (1) when data is transferred from the data register stack to the system data bus. It is clear (0) when data is transferred from the system data bus to the data register stack.
7	Request for master	This bit is set (1) when the data register is ready for a transfer operation. It is cleared (0) when the data register is not ready.

Floppy Control Register (Port 3F7H)

Bits 0 and 1 of the floppy control register set the data transfer rate in kilobits per second. Bits 2 through 7 are reserved. Table 16-5 defines the encoding for the data transfer rates.

Table 16-5. Floppy Control Register

BIT 0	BIT 1	DATA TRANSFER RATE
0	0	500 kbps
1	0	300 kbps[1]
0	1	250 kbps
1	1	125 kbps[2]

NOTES

1. The controller must be programmed for MFM encoding.
2. The controller must be programmed for FM encoding.

Data Register (Port 3F5H)

The data register port provides sequential access to a stack of 8-bit registers used to program the functions of the controller, temporarily hold the data being read from or written to the disk, and provide additional status reports. The sequence and data used with the data register stack depends on the programmed instruction. Only one register is presented to the data bus at a time. Four of these registers are status registers (ST0-ST3). These status registers are only available during the result phase of a command and may only be read after a command has been executed. All of the command bytes (usually 9) must be written to the data register port before the execution phase can begin. All of the result bytes (usually 7) must be read from this register before another command can be executed. Table 16-6 provides bit descriptions for each of the status registers.

Table 16-6. Floppy Data Registers ST0-ST3

REGISTER	BIT	FUNCTION	DESCRIPTION
ST0	0-1	Drive select	These bits indicate which drive interrupted the CPU. See Table 16-3.
	2	Head address	This bit indicates the state of the read/write head at an interrupt.
	3	Not ready	This bit is set when the drive is not ready and a read or write command has been issued. It is also set if a read or write command has been issued for side 1 of a single-sided disk.
	4	Equipment check	This bit is set when a fault signal has been received from the controller or if the track 0 signal is not received after 77 step pulses during the execution of the recalibrate command.

Table 16-6 (continued). Floppy Data Registers ST0-ST3

REGISTER	BIT	FUNCTION	DESCRIPTION
	5	Seek end	This bit is set upon completion of a seek command.
	6-7	Interrupt code	These bits indicate whether a command was completed and executed properly, started but not executed properly, or not started.
ST1	0	Missing address	This bit is set if the controller cannot find an address mark or a deleted address mark. The missing address mark bit in register ST2 is also set when this occurs.
	1	Write protect	This bit is set when a write protect signal is detected during a write data, write deleted data, or format a cylinder command.
	2	No data	This bit is set if the controller cannot find the sector specified by the internal data register during a read, write, or scan command. It is also set if the ID field cannot be read without an error during a read ID command. It can also be set during a read cylinder command if the starting sector cannot be found.
	3	Not used	This bit is always cleared.
	4	Overrun	This bit is set if the controller is not serviced by the processor or DMA within a specified length of time.
	5	Data error	This bit is set when a CRC error is detected in either the ID or data field.

Table 16-6 (continued). Floppy Data Registers ST0-ST3

REGISTER	BIT	FUNCTION	DESCRIPTION
	6	Not used	This bit is always cleared.
	7	End of cylinder	This bit is set when the controller attempts to access a sector beyond the last sector in a cylinder.
ST2	0	Missing address	This bit is set in conjunction with the missing address bit of ST0 when the controller cannot find a data address mark or a deleted data address mark during a read command.
	1	Bad cylinder	When the contents of a cylinder is FFH and does not match the data stored in the internal data register, this bit is set.
	2	Scan not satisfied	This bit is set when the controller cannot find a sector that meets the condition specified in a scan command.
	3	Scan equal hit	This bit is set during the execution of a scan command if the equal condition is met.
	4	Wrong cylinder	This bit is set when the contents of a cylinder do not match the contents of the internal data register.
	5	Data error	When the controller detects a CRC error in the data field, this bit is set.
	6	Control mark	This bit is set when the controller detects a deleted address mark during the execution of a read or scan command.

Table 16-6 (continued). Floppy Data Registers ST0-ST3

REGISTER	BIT	FUNCTION	DESCRIPTION
	7	Not used	This bit is always cleared.
ST3	0-1	Drive select	These bits reflect the drive select signals sent to the drive. See Table 16-3.
	2	Head address	This bit reflects the status of the side select signal sent to the drive.
	3	Two-side	This bit reflects the status of the two-sided signal from the drive.
	4	Track 0	This bit reflects the status of the track 0 signal from the drive.
	5	Ready	This bit reflects the status of the ready signal from the drive.
	6	Write protect	This bit reflects the status of the write protect signal from the drive.
	7	Fault	This bit reflects the status of the fault signal from the drive.

FDC Pinout

The 3765 FDC requires a +5 VDC voltage for normal operation. The clock signal used for this device is a single phase 8 MHz square wave. The pinout for this device is illustrated in Figure 16-2. Table 16-7 describes the signals used by the 3765 FDC.

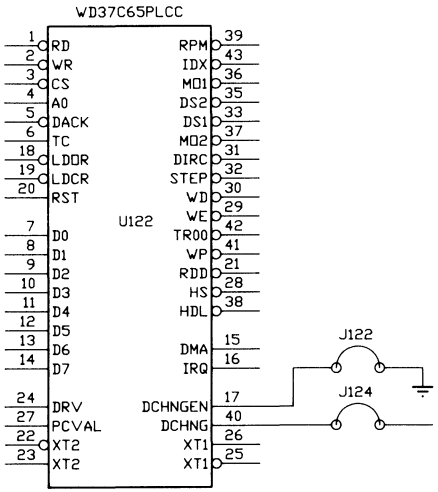


Figure 16-2. 3765 FDC Pinout

Table 16-7. 3765 FDC Signal Descriptions

PIN NUMBER	SIGNAL NAME	DESCRIPTION
1	$\overline{RD}$	Read. A low on this input allows data transfers from the FDC to the bus. A high (1) on the $\overline{CS}$ line disables this input.
2	$\overline{WR}$	Write. A low on this input allows data transfers from the bus to the FDC. A high (1) on the $\overline{CS}$ line disables this input.
3	$\overline{CS}$	Chip select. This active-low input selects the FDC and enables $\overline{RD}$, $\overline{WR}$, and A0.

Table 16-7 (continued). 3765 FDC Signal Descriptions

PIN NUMBER	SIGNAL NAME	DESCRIPTION
4	A0	Data/status register selection. A high on this input (A0 = 1) selects the data register contents and a low (A0 = 0) selects the status register contents for transfer to the data bus.
5	$\overline{\text{DACK}}$	DMA acknowledge. This input line is held low during an active DMA cycle, while the FDC is perfoming a DMA transfer.
6	TC	Terminal count. This output line indicates DMA cycle completion when high. It also terminates data transfers during read, write, and scan commands in either DMA or non-DMA mode.
7 - 14	D0 - D7	8-bit bidirectional data bus. These lines are disabled when $\overline{\text{CS}}$ is held high.
15	DMA	Data DMA request. The FDC requests a DMA transfer by asserting this output (DRQ = 1).
16	IRQ	Interrupt. This output is the FDC interrupt request line.
17	$\overline{\text{DCHNG}}$	Connected to ground.
18	$\overline{\text{LDOR}}$	Load operations register. Address decode which enables the loading of the operations register. Internally gated with $\overline{\text{WR}}$ to create the strobe that latches the data bus into the operations register.
19	$\overline{\text{LDCR}}$	Load control register. Address control which enables the loading of the control register. Internally gated with $\overline{\text{WR}}$ to create the strobe that latches the two LSB's from the data bus into the control register.

Table 16-7 (continued). 3765 FDC Signal Descriptions

PIN NUMBER	SIGNAL NAME	DESCRIPTION
20	RESET	Reset. This input resets the output lines and places the FDC in an idle state. If the RDY input is held high during RESET, an interrupt will be generated within 1.024 ms. To clear the interrupt, use the sense interrupt status command.
21	RDD	Read data. This is the serial data and clock input line used for reading data from the drive.
22	$\overline{XT2}$	Crystal oscillator drive output.
23	XT2	Crystal oscillator drive input used for non-standard data rates.
24	DRV	Drive type. Set at logic 1 for use with single speed drive motors.
25	$\overline{XT1}$	Crystal oscillator drive output.
26	XT1	Crystal oscillator input. This requires a 16 MHz crystal and is used for all standard data rates.
27	PCVAL	Precompensation value. These outputs reflect the write precompensation status for MFM mode. They indicate early, late, or normal timing; set for 125 ns.
28	$\overline{HS}$	Head select. This output line reflects the head selected: logic 1 = side 0, logic 0 = side 1.
29	$\overline{WE}$	Write enable. This output line enables writing data to disk.
30	$\overline{WD}$	Write data. This serial clock and data output line transfers data to disk.

Table 16-7 (continued). 3765 FDC Signal Descriptions

PIN NUMBER	SIGNAL NAME	DESCRIPTION
31	$\overline{\text{DIRC}}$	Direction. This output determines the direction of the head stepper motor: logic 1 = outward motion, logic 0 = inward motion.
32	$\overline{\text{STEP}}$	Step pulse. This line issues a pulse for every track-to-track movement.
33	$\overline{\text{DS1}}$	Drive select 1. This signal originates in the operations register and enables the interface in disk drive number 1.
34	VSS	Ground. DC power return.
35	$\overline{\text{DS2}}$	Drive select 2. This signal originates in the operations register and enables the interface in disk drive number 2.
36	$\overline{\text{MO1}}$,$\overline{\text{DS3}}$	Motor on 1, drive select 3. In PC/AT mode, this signal enables the MOTOR ON signal for disk drive number 1. In base or special mode, this signal enables disk drive number 3.
37	$\overline{\text{MO2}}$,$\overline{\text{DS4}}$	Motor on 2, drive select 4. In PC/AT mode, this signal enables the MOTOR ON signal for disk drive number 2. In base or special mode, this signal enables disk drive number 4.
38	$\overline{\text{HDL}}$	Head load. This output line causes the read/write heads to contact the disk.
39	$\overline{\text{RWC}}$,$\overline{\text{RPM}}$	Reduced write current, revolutions per minute. This output causes a reduced write current when bit density is increased toward the inner tracks, becoming active with numbers 29 and larger. In the PC/AT mode, this signal will be active when the 250 MFM or 125 FM data rate is selected.

Table 16-7 (continued). 3765 FDC Signal Descriptions

PIN NUMBER	SIGNAL NAME	DESCRIPTION
40	DCHNG	Disk change. This signal is active when a high density drive is attempting to format a low density diskette.
41	$\overline{WP}$	Write protected. This input senses write protection on a disk.
42	$\overline{TR00}$	Track 00. This input senses when the head is positioned over the outermost track.
43	$\overline{IDX}$	Index. A high on this input indicates the beginning of a disk track.
44	Vcc	+5 VDC power.

Programming Floppy Disk Operations

Before the FDC can carry out a disk operation, the digital output register and the floppy control register must be initialized. For example, a data byte consisting of the bit pattern 00000000 (00H), which selects a data transfer rate of 500 kbps, must be output to port 3F7H. Then, a data byte consisting of the bit pattern 00011100 (1CH) which, selects drive A, enables the DMA and interrupt circuits, and turns on disk drive A's motor, must be output to port 3F2H.

NOTE: Bit 2 of the digital output register should only be cleared if a serious error occurs. When the controller is reset, the disk drives must be recalibrated, moving each drive's head to track 0.

The following sequence represents a typical series of steps that could be used in programming a disk operation:

1. Specify head settling time and amount of data to be transferred.

2. Specify the data transfer rate and the recording mode (FM or MFM).

3. Select a drive and turn on the motor.

4. Wait for the motor to come to speed.

5. Send the command instruction set to the controller.

6. Wait for the interrupt that indicates the completion of the operation.

7. Initialize the DMA to move data to or from memory.

8. Send the proper instruction to the controller to read or write data.

9. Wait for the interrupt that indicates the completion of the operation.

10. Read the status report from the controller and analyze it to make sure the operation was performed correctly.

11. Turn off the motor.

These steps represent a simplified version of a basic disk access operation, such as reading or writing a sector. Due to the structure of the disk, reading or writing a sector is not quite this simple. In order to read or write a specific sector, first read the disk directory to locate the file. Next, use the file allocation tables to locate the desired sector on the disk. Third, move the heads to the track and locate the sector, and then perform the read or write operation. The software interrupts contain all the necessary routines to perform these complex operations. Refer to Chapter 10 for complete programming information on the disk drive interrupts. In addition,

refer to the Programmer's Utility Package for a description of the organization of the MS-DOS disk.

Floppy Disk Controller Commands

The FDC can execute 15 different commands. Each command is initiated by outputting a multibyte instruction to the floppy data register at port 3F5H. The controller then executes the commands and returns a multibyte result report to the floppy data register. These result bytes must be read by the microprocessor before another command can be initiated.

For more information on programming this controller, refer to the Western Digital WD37C65A data sheet. The available controller commands include the following:

- Reading and writing one or more sectors
- Reading an entire track
- Locating the position of the head on a track
- Formatting a track
- Scanning and comparing written data against memory
- Seeking to a track
- Calibrating (synchronizing) head-to-track information
- Checking interrupt and drive status
- Seeking to a specified track.

Table 16-8 describes the 15 commands available in the FDC.

Table 16-8. Floppy Disk Controller Commands

COMMAND	DESCRIPTION
Read data	Reads data from one or more sectors of a two-track cylinder.
Read deleted data	Allows a sector marked with a deleted data mark to be read.
Write data	Writes data to one or more sectors of a one- or two-track cylinder.
Write deleted data	Same as a write data command except that deleted data marks are written instead of normal data marks.
Read a track	Reads the contents of an entire track.
Read sector ID	Reads the values of the first ID field it is able to read.
Format a track	Allows an entire track to be formatted.
Scan equal	Reads the data on the disk and compares it to the data supplied by the processor. If the two sets of data are equal, the operation terminates.
Scan low or equal	Reads the data on the disk and compares it to the data supplied by the processor. If the disk data is of equal or lesser value, the operation terminates.
Scan high or equal	Reads the data on the disk and compares it to the data supplied by the processor. If the disk data is of equal or greater value, the operation terminates.
Recalibrate	Causes the head of the selected drive to be moved to track 0. The internal track counter is cleared for this drive.

Table 16-8 (continued). Floppy Disk Controller Commands

COMMAND	DESCRIPTION
Sense interrupt status	Allows the user to sense interrupts that result during seek and recalibrate commands, which have no result phase.
Specify	Sets the initial values of each of three internal timers. These are the head unload time, the step rate time, and the head load time.
Drive status	Reports the status of the disk drives.
Seek	Causes the disk drive to seek to the specified cylinder. The specified drive's track counter is incremented or decremented according to the number of step pulses sent to the drive.

Most of the commands begin with a byte that initializes the controller for the operation that follows. This byte allows deleted address marks to be skipped, selects FM or MFM encoding, and allows use of multi-track mode. This byte is usually followed by a head and drive select byte. Table 16-9 lists each of the bits used for these command bytes and describes their function. The bit position for the command bytes is described with each instruction.

Table 16-9. Controller Command Bits

BIT	DESCRIPTION
Skip bit	If set (1), this bit causes the operation to skip the deleted data address mark.
FM/MFM bit	If clear (0), the controller is placed in the FM read/write mode. If set (1), the controller is placed in the MFM read/write mode.

Table 16-9 (continued). Controller Command Bits

BIT	DESCRIPTION
Multitrack bit	If set (1), the multitrack mode is in effect. At the end of side 0, side 1 will be used automatically.
Head number bit	This is the same as the head address and will be set (1) for side 1 or clear (0) for side 0.
Drive unit number bits	These two bits specify the drive unit number: unit 0 = drive A, unit 1 = drive B, unit 2 = drive C, and unit 3 = drive D.

Read Data — This command uses nine command bytes and returns seven result bytes. It reads data from one or more sectors. Each of the command and status bytes are described in Tables 16-9 and 16-10.

Table 16-10. Read Data Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 06H; bit 5 = skip bit; bit 6 = FM/MFM bit; bit 7 = multitrack bit. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-6.
	5	Write the sector number to be read.

Table 16-10 (continued). Read Data Command

PHASE	BYTE	DESCRIPTION
	6	Write the number of bytes per sector.
	7	Write the end of track, which is the final sector number to be read on a cylinder.
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the data length. If the bytes per sector value (byte 6) is 0, this value is the data length to be read from or written to each sector.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read (cylinder) number.
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Read deleted data — This command uses nine command bytes and returns seven result bytes. It reads data from a sector that has been marked with a deleted data mark, as described in Table 16-11. With the exception of the first set of command codes, this command is identical to the read data command.

Table 16-11. Read Deleted Data Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 0CH; bit 5 = skip bit; bit 6 = FM/MFM bit; bit 7 = multitrack bit. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-9.
	5	Write the sector number to be read.
	6	Write the number of bytes per sector.
	7	Write the end of track, the final sector number to be read on a cylinder.
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the data length. If the bytes per sector value (byte 6) is 0, this value is the data length to be read from or written to each sector.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.

Table 16-11 (continued). Read Deleted Data Command

PHASE	BYTE	DESCRIPTION
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Write data — This command uses nine command bytes and returns seven result bytes. It writes data to one or more sectors of a two-track cylinder, as described in Table 16-12.

Table 16-12. Write Data Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 05H; bit 5 = 0; bit 6 = FM/MFM bit; bit 7 = multitrack bit. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-9.
	5	Write the sector number to be read.
	6	Write the number of bytes per sector.
	7	Write the end of track, the final sector number to be read on a cylinder.

Table 16-12 (continued). Write Data Command

PHASE	BYTE	DESCRIPTION
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the data length. If the bytes per sector value (byte 6) is 0, this value is the data length to be read from or written to each sector.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Write deleted data — This command uses nine command bytes and returns seven result bytes. It writes data with a deleted data mark as described in Table 16-13. With the exception of the first set of command codes, this command is identical to the write data command.

Table 16-13. Write Deleted Data Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 09H; bit 5 = 0; bit 6 = FM/MFM bit; bit 7 = multitrack bit. Refer to Table 16-9.

Table 16-13 (continued). Write Deleted Data Command

PHASE	BYTE	DESCRIPTION
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-9.
	5	Write the sector number to be read.
	6	Write the number of bytes per sector.
	7	Write the end of track, the final sector number to be read on a cylinder.
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the data length. If the bytes per sector value (byte 6) is 0, this value is the data length to be read from or written to each sector.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Read a track — This command uses nine command bytes and returns seven result bytes. It reads the contents of an entire track from the index (timing) hole to the end of the track, as described in Table 16-14.

Table 16-14. Read a Track Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 02H; bit 5 = skip bit; bit 6 = FM/MFM bit; bit 7 = 0. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-9.
	5	Write the sector number to be read.
	6	Write the number of bytes per sector.
	7	Write the end of track, the final sector number to be read on a cylinder.
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the data length. If the bytes per sector value (byte 6) is 0, this value is the data length to be read from or written to each sector.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.

Table 16-14 (continued). Read a Track Command

PHASE	BYTE	DESCRIPTION
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Read sector ID — This command uses two command bytes and returns seven result bytes. It reads the values of the first ID field it is able to read, as described in Table 16-15.

Table 16-15. Read Sector ID Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 0AH; bit 5 = 0; bit 6 = FM/MFM bit; bit 7 = 0. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.

Table 16-15 (continued). Read Sector ID Command

PHASE	BYTE	DESCRIPTION
	6	Read sector number.
	7	Read bytes per sector.

Format a track — This command uses six command bytes and returns seven result bytes. It formats an entire track, as described in Table 16-16.

Table 16-16. Format a Track Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 0DH; bit 5 = 0; bit 6 = FM/MFM bit; bit 7 = 0. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the number of bytes per sector.
	4	Write the number of sectors per track.
	5	Write the number of bytes in gap 3. During read/write commands, this value determines the time that the sync signal will be active.
	6	Write the pattern to be used in the data area of each sector.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.

Table 16-16. Format a Track Command

PHASE	BYTE	DESCRIPTION
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Scan equal — This command uses nine command bytes and returns seven result bytes. It reads the data on the disk and compares it with the data supplied by the processor as described in Table 16-17. If the two sets of data are equal, the operation terminates.

Table 16-17. Scan Equal Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 11H; bit 5 = skip bit; bit 6 = FM/MFM bit; bit 7 = multitrack bit. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-9.
	5	Write the sector number to be read.

Table 16-17 (continued). Scan Equal Command

PHASE	BYTE	DESCRIPTION
	6	Write the number of bytes per sector.
	7	Write the end of track, the final sector number to be read on a cylinder.
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the sector timing pattern. If this byte = 1, the sectors are contiguous. If this byte = 2, the sectors alternate.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Scan low or equal — This command uses nine command bytes and returns seven result bytes. It reads the data on the disk and compares it with the data supplied by the processor, as described in Table 16-18. If the disk data is of equal or of lesser value, the operation terminates.

Table 16-18. Scan Low or Equal Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 19H; bit 5 = skip bit; bit 6 = FM/MFM bit; bit 7 = multitrack bit. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-9.
	5	Write the sector number to be read.
	6	Write the number of bytes per sector.
	7	Write the end of track, the final sector number to be read on a cylinder.
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the sector timing pattern. If this byte = 1, the sectors are contiguous. If this byte = 2, the sectors alternate.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.

Table 16-18 (continued). Scan Low or Equal Command

PHASE	BYTE	DESCRIPTION
	6	Read sector number.
	7	Read bytes per sector.

Scan high or equal — This command uses nine command bytes and returns seven result bytes. It reads the data on the disk and compares it with the data supplied by the processor, as described in Table 16-19. If the disk data is of equal or greater value, the operation terminates.

Table 16-19. Scan High or Equal Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first set of command codes. Bits 0 - 4 = 1DH; bit 5 = skip bit; bit 6 = FM/MFM bit; bit 7 = multitrack bit. Refer to Table 16-9.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the cylinder (track) number.
	4	Write the head address. This value is equal to the value of the head number (0 or 1), as defined in Table 16-9.
	5	Write the sector number to be read.
	6	Write the number of bytes per sector.
	7	Write the end of track, the final sector number to be read on a cylinder.

Table 16-19 (continued). Scan High or Equal Command

PHASE	BYTE	DESCRIPTION
	8	Write the gap length, which is used for synchronization in read operations.
	9	Write the sector timing pattern. If this byte = 1, the sectors are contiguous. If this byte = 2, the sectors alternate.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read ST1. Refer to Table 16-6.
	3	Read ST2. Refer to Table 16-6.
	4	Read cylinder (track) number.
	5	Read head number.
	6	Read sector number.
	7	Read bytes per sector.

Recalibrate — This command uses two command bytes and does not return any result bytes. It causes the head of the selected drive to be moved to track 0. The internal track counter for the drive is cleared. Refer to Table 16-20 for the description of the two command bytes. A sense interrupt command must be sent following this command or the FDC will consider the next command to be invalid.

Table 16-20. Recalibrate Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first command code. Bits 0 - 7 = 07H.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bits 2 - 7 = not used. Refer to Table 16-9.

Sense interrupt status — This command uses one command byte and returns two result bytes. It allows the user to sense interrupts that result during seek and recalibrate commands, as described in Table 16-21. It is used in conjunction with seek and recalibrate commands, since they have no result phase.

Table 16-21. Sense Interrupt Status Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the command code. Bits 0 - 7 = 08H.
Result	1	Read ST0. Refer to Table 16-6.
	2	Read cylinder (track) number.

Specify — This command uses three command bytes. It sets the initial values of each of three internal timers, as described in Table 16-22. These are the head unload time, the step rate time, and the head load time.

Table 16-22. Specify Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the command code. Bits 0 - 7 = 03H.
	2	Write the head unload time and step rate time. Bits 0 - 3 = head unload time in 16 ms increments. Bits 4 - 7 = step rate time in 1 ms increment.
	3	Write the DMA mode flag and the head load time. Bit 0 = DMA mode flag: clear (0) = DMA mode, set (1) = non-DMA mode. Bits 1 - 7 = head load time in 2 ms increments.

Drive Status — This command uses two command bytes and returns one result byte. It reports the status of the specified disk drive, as described in Table 16-23.

Table16-23. Drive Status Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first command code. Bits 0 - 7 = 04H.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
Result	1	Read ST3. Refer to Table 16-6.

Seek — This command uses three command bytes. It causes the specified disk drive to seek to the specified cylinder, as described in Table 16-24. The specified drive's track counter is incremented or decremented according to the number of step pulses sent to the drive. A sense interrupt status command must be sent after this command or the FDC will consider the next command invalid.

Table 16-24. Seek Command

PHASE	BYTE	DESCRIPTION
Command	1	Write the first command code. Bits 0 - 7 = 0FH.
	2	Write the second set of command codes. Bits 0 - 1 = drive unit number (0 - 3); bit 2 = head number; bits 3 - 7 = not used. Refer to Table 16-9.
	3	Write the new cylinder number to which the heads are to be stepped.

NOTE: Any other command codes that are sent to the data register will result in a NOP operation and one status byte with a value of 80H will be returned.

Hard Disk Control

This computer contains all circuitry necessary to support the internal hard disk drive. The hard disk controller contains firmware that responds to control signals provided by the BIOS firmware. Chapter 10 discusses using the BIOS interrupts. Custom circuits provide PC-compatible hard disk control, including timing, automatic head and cylinder switching, automatic seek and verify of each data transfer, data error sensing and correction, sector interleave, enhanced step rates, and removable cartridge drive support.

Data is transferred to the hard disk at a rate of 5 Mbps, in 16-bit words. The internal controller can also select an 8-bit mode, allowing four additional bytes to be transferred for long-word read and write commands. A 512-byte sector buffer is provided to prevent data overruns during transfer operations.

Hard Disk Drive Control Registers

Eight task file registers and two auxiliary registers control and provide status reports for the hard disk drive controller. Table 16-25 lists the primary and secondary port addresses for these registers. The table also illustrates the read and write operation for each register.

Table 16-25. Hard Disk Drive Control Register Port Addresses

PRIMARY PORT ADDRESS	SECONDARY PORT ADDRESS	READ OPERATION REGISTER	WRITE OPERATION REGISTER
1F0H	170H	Data	Data
1F1H	171H	Error	Write precomp
1F2H	172H	Sector count	Sector count
1F3H	173H	Sector number	Sector number
1F4H	174H	Cylinder low	Cylinder low
1F5H	175H	Cylinder high	Cylinder high
1F6H	176H	Drive & head	Drive & head
1F7H	177H	Status	Command

Table 16-25 (continued). Hard Disk Drive Control Register Port Addresses

PRIMARY PORT ADDRESS	SECONDARY PORT ADDRESS	READ OPERATION REGISTER	WRITE OPERATION REGISTER
3F6H	376H	Not used	Fixed disk
3F7H	377H	Digital input	Digital output

NOTE: The digital output register and bit 7 of the digital input register are used in floppy disk drive operations.

Data Register (Port 1F0H)

The data register is used in parallel read and write operations. It provides a 16-bit path to the sector buffer. Data bytes pass through this buffer to prevent overruns during read and write operations. An internal controller allows the register to select 8-bit mode, allowing the additional 4 bytes of ECC code to be transferred in a read or write long command.

Error Register (Port 1F1H)

The error register is a read-only register which returns a 1-byte code giving error information about the last command. Two sets of codes are used. The first set, listed in Table 16-26, appears following the execution of a diagnostic command. The second set, listed in Table 16-27, appears following the execution of any other command.

Following the execution of a diagnostic command, the controller is in diagnostic mode. In this mode the hex value of the register contents determines the error condition. In normal operation mode, each bit of the register indicates a particular error condition. The contents of this register are valid only when the error bit of the status register is set.

Table 16-26. Diagnostic Mode Error Register Codes

CODE	ERROR
01H	No error
02H	Controller error
03H	Sector buffer error
04H	ECC device error
05H	Control processor error

Table 16-27. Operation Mode Error Register Codes

BIT	FUNCTION	DESCRIPTION
0	Data address mark not found	This bit is set if the data address mark is not found in the 16 bytes following the ID field. If the controller is programmed for retry, 16 passes are made before the bit is set. If the controller is not programmed for retry, only one pass is made.
1	Track 0	This bit is set during a restore command if the track 0 signal is not set within 2047 step pulses.
2	Aborted command	This bit is set if an invalid command is received by the controller. It is also set if a drive not ready, no seek complete, or write fault signal are received from the drive.
3	Cartridge changed	This bit reflects the status of the cartridge changed line of the drive. It must be cleared using the acknowledge cartridge changed command before any other command can be executed.

Table 16-27 (continued). Operation Mode Error Register Codes

BIT	FUNCTION	DESCRIPTION
4	ID not found	This bit is set if the desired ID field is not found within 2 revolutions of search. When the controller has been programmed for retry, 16 search attempts are made. Also, an automatic seek to the correct cylinder is issued if the heads are located on the wrong cylinder. If the controller is not programmed for retry, no automatic seek is issued if the heads are located on the wrong cylinder. The error bit is set following 2 revolutions of search.
5	Write protected	This bit is only used with removable cartridge drives. If the cartridge has been write protected, the bit is set.
6	Data ECC	This bit is set if an noncorrectable data ECC error is detected.
7	Bad block detect	This bit is set when a bad block mark is read from the ID field. When this bit is set, any read or write operation is aborted.

Write-Precompensation Register (Port 1F1H)

The write-precompensation register is an 8-bit write only register. The hex value written to this register is multiplied by 4 by the controller and the resultant value is used as the first precompensated cylinder number.

Sector Count Register (Port 1F2H)

The sector count register is an 8-bit register used in conjunction with the sector number register. It specifies the number of sectors to be transferred during a read or write operation. Multiple sector operations are supported and may cross tracks and cylinders. The appropriate drive characteristics must be loaded using the set drive parameter command before a multiple sector transfer can be executed. During multiple sector operations, this register is decremented and the sector number register is incremented after each successful sector transfer. When a value of zero is placed in the sector count register, a transfer of 256 sectors is specified.

Sector Number Register (Port 1F3H)

This 8-bit register is used during read, write, and verify operations. It must be loaded with the logical starting sector number prior to execution of the command. Upon completion of the command, the register is updated with the last sector read or written. If an error is detected, upon termination of the command, the register is loaded with the sector in error.

Cylinder Low and Cylinder High Registers (Ports 1F4H and 1F5H)

This pair of 8-bit registers are used during read, write, seek, and format operations. The cylinder low register (port 1F44H) is loaded with the low-byte hex value of the starting cylinder for the operation. The cylinder high register (port 1F5H) contains the starting cylinder high-byte hex information. Upon completion of the operation, the registers are updated and contain current cylinder information.

Drive and Head Register (Port 1F6H)

This 8-bit write-only register is used to program drive and head select for most operations. It is also used with the set parameters command to indicate the maximum number of heads for each drive.

Table 16-28 provides coding information for this register. The controller supports a maximum of two hard disk drives with a maximum of 16 heads each.

Table 16-28. Drive and Head Register Codes

BIT	FUNCTION	DEFINITION
0-3	Head select	Enter a hex value ranging from 0-F. The value indicates the head selected for most operations. The maximum number of heads for the drive indicated by bits 4-7 for the set parameters command.
4-7	Drive select	Enter the bit pattern 1010 (AH) to select or set the parameters for drive 0. Enter the bit pattern 1011 (BH) to select or set parameters for drive 1.

Status Register (Port 1F7H)

This 8-bit read-only register contains a 1- byte code that indicates the status of the drive and the controller. Reading this register clears the hard disk interrupt. Table 16-29 provides the decoding information for the register.

Table 16-29. Status Register Codes

BIT	FUNCTION	DEFINITION
0	Error	This bit is set when an error is encountered while executing a command. It indicates that the command was terminated. When this bit is set, the microprocessor should read the error register to determine the nature of the error.

Table 16-29 (continued). Status Register Codes

BIT	FUNCTION	DEFINITION
1	Index	This bit reflects the state of the index signal from the drive.
2	Corrected data	This bit is set during a read command when the data in the buffer awaiting transfer has been corrected using ECC. This bit does not cause a multiple sector read command to abort if a correctable data error is encountered, but does remain set until a new command is issued.
3	Data request	This bit is a signal to the microprocessor that the controller is ready for data to be read from or written to the data register. When the appropriate amount of data has been transferred, the bit is cleared.
4	Seek complete	This bit reflects the state of the seek complete signal from the drive.
5	Write fault	This bit reflects the state of the write fault signal from the drive.
6	Drive ready	This bit reflects the state of the drive ready signal from the drive.
7	Busy	This bit is set while the controller is executing a command. While this bit is set, the contents of bits 1, 4, 5, and 6 are valid. However, all other bits, and all other registers, are considered invalid. Check the status of this bit prior to reading any other registers.

Command Register (Port 1F7H)

The 8-bit code written to this register provides the controller with the commands to be executed. The controller is capable of executing thirteen commands. Prior to writing a command to this register, the other registers containing information to be used during execution must be loaded and the busy bit (bit 7) of the status register must be cleared. Tables 16-30 and 16-31 provide coding information for this register. The commands are described in greater detail later in this chapter.

Table 16-30. Command Register Coding

COMMAND	BINARY CODE	HEX CODE
Test drive	0000 0000	00H
Cartridge change	0000 0000	00H
Restore	0001 XXXX	1XH[1]
Read sector		
Data only with retry	0010 0000	20H
Data only, no retry	0010 0001	21H
Data & ECC with retry	0010 0010	22H
Data & ECC, no retry	0010 0011	23H
Write sector		
Data only with retry	0011 0000	30H
Data only, no retry	0011 0001	31H
Data & ECC with retry	0011 0010	32H
Data & ECC, no retry	0011 0011	33H
Verify with retry	0100 0000	40H
Verify, no retry	0100 0001	41H
Format	0101 0000	50H
Seek	0111 XXXX	7XH[1]

Table 16-30 (continued). Command Register Coding

COMMAND	BINARY CODE	HEX CODE
Diagnose	1001 0000	90H
Set parameters	1001 0001	91H
Acknowledge cartridge change	1011 0000	B0H
Set drive type	1111 0000	F0H
Recovery mode on	1111 0001	F1H
Recovery mode off	1111 0010	F2H
Start/stop	1110 00ES[2]	EXH[2]

NOTES:

1. The low-order bits of this command byte set the step rate for the seek or restore operation. See Table 16-31 for specific programming information.

2. E = Enable Timer
   0 = Timer select disable
   1 = Timer select enable

   S = Start/Stop Moter
   0 = Motor stop
   1 = Motor start

Both the seek and restore commands may be executed at step rates between 5 microseconds and 7.5 milliseconds. The step rate is programmed with the lower 4 bits of the command byte. Table 16-31 provides coding information for the variable step rates.

Table 16-31. Seek and Restore Step Rate Codes

STEP RATE	BINARY CODE	HEX CODE
5 us	1110	EH
10 us	0111	7H
20 us	1000	8H
30 us	1001	9H
35 us	0000	0H
40 us	1010	AH
50 us	1011	BH
60 us	1100	CH
70 us	1101	DH
0.5 ms	0001	1H
1.0 ms	0010	2H
1.5 ms	0011	3H
2.0 ms	0100	4H
2.5 ms	0101	5H
3.0 ms (see note)	0110	6H
7.5 ms	1111	FH

NOTE: Following execution of a reset or diagnose command, the controller sets the step rate to 3.0 milliseconds.

Fixed Disk Register (Port 3F6H)

This 8-bit write-only register enables the fixed disk interrupt and resets the fixed disk functions. It does not support reduced write current functions. Table 16-32 provides coding information for this register.

Table 16-32. Fixed Disk Register Codes

BIT	FUNCTION	DESCRIPTION
0	Not used	
1	Interrupt	When this bit is cleared, the fixed disk interrupt is enabled. Setting this bit disables the fixed disk interrupt.

Table 16-32 (continued). Fixed Disk Register Codes

BIT	FUNCTION	DESCRIPTION
2	Reset	When this bit is set, the fixed disk functions are reset.
3	Reserved	This controller does not support the reduced write current function.
4-7	Not used	

Digital Input Register (Port 3F7H)

This 8-bit read-only register reflects the control signals being sent to the drive during a read operation. Bits 0 through 6 reflect the status of the hard disk drive. Bit 7 indicates the status of the floppy disk changed signal. Table 16-33 provides decoding information for the digital input register.

Table 16-33. Digital Input Register Codes

BIT	FUNCTION	DESCRIPTION
0	Drive 0	This bit reflects the state of the drive 0 select signal sent from the controller.
1	Drive 1	This bit reflects the state of the drive 1 select signal sent from the controller.
2	Head 0	This bit reflects the state of the head 0 select signal sent from the controller.
3	Head 1	This bit reflects the state of the head 1 select signal sent from the controller.
4	Head 2	This bit reflects the state of the head 2 select signal sent from the controller.

Table 16-33 (continued). Digital Input Register Codes

BIT	FUNCTION	DESCRIPTION
5	Head 3	This bit reflects the state of the head 3 select signal sent from the controller.
6	Write gate	This bit reflects the state of the write gate signal sent from the controller.
7	Floppy	This bit reflects the state of the floppy disk changed signal sent from the floppy disk drive.

Programming Hard Disk Operations

Programming hard disk operations in this computer is not simple since a variety of hard disk drives are accommodated. Be sure you are thoroughly familiar with the parameters for the drive you intend to program. As with floppy disk operations, an error in programming results in lost data at best. The software interrupts discussed in Chapter 10 contain all the routines necessary to carry out hard disk operations. Use these whenever possible to prevent lost data and drive damage. Also, refer to Programmer's Utility Package for additional information.

Hard Disk Controller Commands

The hard disk controller (HDC) accepts thirteen commands. In general, commands are executed in the following manner:

1. Output the drive, head, disk, and operation information to the appropriate hard disk control register ports.

2. Read the controller status register to verify that the controller is not busy, the write fault signal is inactive, and the drive ready and seek complete signals are active.

3 Output the command byte to the command register.

4. Read the controller status register and error register ports to verify that the command was properly executed.

The remainder of this chapter provides information about each of these commands.

Set parameters, set drive type — A maximum of two hard disk drives may be installed in this computer. They need not be the same drive type or have the same drive parameters. The HDC supports several different hard disk drives. The set parameters and set drive type commands notify the controller which drives have been installed in a particular system.

These commands must be executed prior to attempting any multi-sector operations and after each power-up and reset. The drives are defaulted to fixed disk types at power-up and reset. The set parameters command is issued first, followed by the set drive type command. This command set must be issued for each hard disk drive installed in the system.

To execute the set parameters command, refer to Table 16-28 and output the drive number and the maximum number of heads in the drive to the head and drive register (port 1F6H) and the bit pattern 0001 0001 (11H) to the sector count register (port 1F2H). Then output the bit pattern 1001 0001 (91H) to the command register (port 1F7H).

To execute the set drive type command, output the drive number to the head and drive register (port 1F6H). If the drive is a fixed disk type, output the bit pattern 0000 0000 (00H) to the cylinder high register (port 1F5H). If the drive is a removable cartridge type, output the bit pattern 0000 0001 (01H) to the cylinder high register (port 1F5H). Then output the bit pattern 1111 0000 (F0H) to the command register (port 1F7H).

Test drive/cartridge change — This command tests the state of the drive ready and cartridge changed signals from the drive. It is executed by outputting the drive and head select bit pattern to the drive and head register (port 1F6H) and then the bit pattern

00000000 (00H) to the command register port (port 1F7). Refer to Table 16-28 for the drive and head register codes.

Acknowledge cartridge changed — This command resets the cartridge changed signal from a removable cartridge drive by deselecting and then reselecting the drive. If the controller is in hardware drive select mode, the drive to be reset must be deselected and reselected through the head and drive register before this command can be executed. Refer to Table 16-28 for drive and head register codes. This command must be issued for each drive indicating a cartridge changed error.

To execute this command, deselect and reselect the drive indicating an error by outputting the head and drive code bytes to the head and drive register (port 1F6H), and then output the bit pattern 1011 0000 (B0H) to the command register (port 1F7H).

Restore — For fixed disk drives, this command sends a series of 2047 step pulses to the selected drive. After each pulse, the controller monitors the seek complete and track 00 signals from the drive. If track 00 has not been found after all 2047 steps have been executed, the controller sets the error bit in the status register. Upon completion of the command, an interrupt is generated.

For removable cartridge drives, this command sends a burst of 2047 pulses to the selected drive. After receiving a seek complete signal, the controller checks the state of the track 00 signal. Again, an interrupt is generated upon completion of the command.

This command is executed by outputting the appropriate drive and head select code to the drive and head register (port 1F6H) and then outputting the bit pattern 0001 and the 4-bit step rate code to the command register (port 1F7H). Refer to Table 16-31 for the step rate code and Table 16-28 for the drive and head register code.

Format track — This command formats the track specified by the drive and head, cylinder high, and cylinder low registers. Sector interleave is programmable, allowing up to 17-way interleave. A sector interleave table for 17 sectors followed by a pad of 478 bytes of 00H must be output to the data register for each track to be formatted. The controller ignores the pad and issues an interrupt

upon completion of the command. The format track command may be repeated without issuing a restore command, provided the tracks are all located on the same disk. A restore command must be issued when switching disks.

The controller allows a 17 sector per track format. It will not operate properly if another format is attempted. The sector layout for this format, described in Table 16-34, is repeated 17 times on each track.

Table 16-34. Format Track Command Track Layout

FUNCTION	BYTES	VALUE	DESCRIPTION
SPEED GAP	30	4EH	These bytes provide the speed tolerance gap. Their presence assures a 4.19% speed tolerance.
SYNC1	14	00H	These bytes provide a synchronization signal for the phase locked loop (PLL).
IDAM	1	A1H	This is the first byte of the address ID mark.
IDEN	1	XXH	This is the last byte of the address ID mark.
CYL	1	XXH	This byte contains the current cylinder number.
HEAD	1	2XH	This byte contains the current head number.
SECTOR	1	XXH	This byte contains the logical sector number.
CRC	2	XXH	These bytes contain the cyclical redundancy code for the ID field.

Table 16-34 (continued). Format Track Command Track Layout

FUNCTION	BYTES	VALUE	DESCRIPTION
SYNC 2	15	00H	The first three bytes provide write splice. The write gate should open immediately after the third byte of this field. The remaining 12 bytes provide PLL sync.
DAM	1	A1H	This is the first byte of the data address mark.
DIN	1	F8H	This is the last byte of the data address mark.
DATA	512	XXH	These bytes are the data storage area. A maximum of 256 16-bit words are stored high byte first.
ECC	4	XXH	These are the error correction code bytes.
SYNC 3	3	00H	These bytes provide the write turn off area.

The sector interleave table is comprised of two bytes for each sector. The first byte indicates whether the sector is a good or a bad block. The second byte indicates the logical sector located in the particular physical sector. Table 16-35 gives an example of a 2 to 1 sector interleave table. The example indicates that all the sectors in the track are good blocks. Entering an 80 in place of the 00 byte indicates a bad block. A physical sector number is provided in the table to help you understand the sector layout on the track. Only the logical sector number, the sector indicator, and the pad of 00H bytes are entered into the data register.

Table 16-35. 2 to 1 Sector Interleave

PHYSICAL SECTOR NUMBER	SECTOR INDICATOR	LOGICAL SECTOR NUMBER
01H	00H	01H
02H	00H	0AH
03H	00H	02H
04H	00H	0BH
05H	00H	03H
06H	00H	0CH
07H	00H	04H
08H	00H	0DH
09H	00H	05H
0AH	00H	0EH
0BH	00H	06H
0CH	00H	0FH
0DH	00H	07H
0EH	00H	10H
0FH	00H	08H
10H	00H	11H
11H	00H	09H

Seek — This command causes the selected drive to seek to the cylinder indicated by the cylinder high and cylinder low registers. The controller does not wait for a seek complete signal before issuing a task complete interrupt. The head position is not verified.

To execute this command, output the drive and head code to the head and drive register (port 1F6H), output the selected cylinder number to the cylinder high (port 1F5H) and cylinder low (port 1F4H) registers, and output the bit pattern 0111 and the step rate code to the command register. Refer to Table 16-31 for the step rate code and Table 16-28 for the drive and head register code.

Read sector — This command causes the controller to automatically seek to the head, cylinder, and sector indicated by the drive and head register, cylinder high and cylinder low registers, and sector number register and read from 1 to 256 sectors as indicated in the sector count register. An interrupt is sent to the microproces-

sor upon completion of each sector, notifying it to read the data register. No interrupt is generated at the end of the command.

The command can be executed in either data and ECC mode or in data only mode. If the command is executed in data and ECC mode, sectors of 516 bytes are transferred. In this mode, a maximum of five data byte errors may be corrected without terminating the command. If the command is executed in data only mode, sectors of 512 bytes are transferred.

If the command is executed in retry mode, up to 16 attempts will be made before the command is terminated. If retry mode is disabled, two attempts are made before the command is terminated. The command is executed at the same step rate as specified in the last seek or restore command. The command is only executed following a successful match of the sector ID with the targeted ID.

To execute this command, output the drive and head information to the drive and head register (port 1F6H) using the information from Table 16-28, and output the logical starting cylinder number in hex to the cylinder high (port 1F5H) and cylinder low (port 1F4H). Next, output the number of sectors to be read in hex to the sector count register (port

1F2H), output the logical starting sector number in hex to the sector number register (port 1F3), and output the read command bit pattern to the command register (port 1F7H) using the information from Table 16-30. Finally, input the data from the data register (port 1F0H).

Verify — This command is used to verify the integrity of a hard disk. It is similar to the read command but sends no data to the microprocessor. The controller seeks to the drive, head, cylinder, and sector indicated by the head and drive register, the cylinder high and cylinder low registers, and sector number register, reads the data and ECC bytes for the number of sectors specified in the sector count register, checks the ECC bytes, and generates an interrupt when an error is encountered or when the command is completed.

Write sector — This command causes the controller to automatically seek to the drive, head, cylinder, and sector indicated by the drive and head register, cylinder high and cylinder low registers, and sector number register and write the number of sectors indicated by the sector count register. After the first sector has been transferred, an interrupt is sent to the microprocessor before each subsequent sector is transferred. Another interrupt is generated upon completion of the command.

The write sector command can be executed in either data and ECC mode or in data only mode. In data only mode, 512 bytes per sector are transferred. In data and ECC mode, 516 bytes per sector are transferred. In this mode, a maximum of five data byte errors may be corrected automatically without terminating the command.

If this command is executed in retry mode, up to 16 attempts will be made before the command is terminated. If retry mode is disabled, two attempts are made before the command is terminated. The command is executed at the step rate specified in the last seek or restore command. The command is only executed following a successful match of the sector ID with the targeted ID.

To execute this command, output the drive and head information to the drive and head register (port 1F6H) using the information from Table 16-28. Next, output the number of cylinders to be written to the sector count register (port 1F2H), output the logical starting sector number to the sector number register (port 1F3H), the starting cylinder number to the cylinder high (port 1F5H) and cylinder low (port 1F4H) registers. Finally, output the write sector command bit pattern to the command register (port 1F7H) using the information from Table 16-30 and the data to the data register (port 1F0H).

Diagnose — This command initiates the controller's self-tests. The results of the tests are reported in the error register. Upon completion of the tests, an interrupt is generated. To execute this command, output the bit pattern 1001 0000 (90H) to the command register (port 1F7H) and input the result from the error register (port 1F1H).

Set recovery mode on, set recovery mode off — These commands assert and deassert the recovery line to a removable cartridge drive. To operate in recovery mode, the recovery mode on command must be issued at power-up and after each reset, since the controller deasserts the recovery line at power-up and on reset. The commands must be issued for each removable cartridge drive. They are used only with removable cartridge drives and result in an aborted command error if used with a fixed drive.

To execute the set recovery mode on command, refer to Table 16-28 and output the drive select code to the drive and head register (port 1F6H). Then output the bit pattern 1111 0001 (F1H) to the command register (port 1F7H).

To execute the set recovery mode off command, refer to Table 16-28 and output the drive select code to the drive and head register (1F6H). Then output the bit pattern 1111 0010 (F2H) to the command register (port 17FH).

Start/stop drive motor — This command starts and stops the hard disk drive and sets the auto timer. The functions are specified by the E and S bits of the command.

If the drive is on and the OFF command is issued, the controller will turn the drive off immediately. If the drive is off and the ON command is issued, the $\overline{\text{POWER ON}}$ signal goes active and waits until the drive is turned on and $\overline{\text{READY}}$ goes true.

If the AUTO OFF timer function is enabled and the drive is not accessed within the time limit specified by the timer value, the $\overline{\text{POWER OFF}}$ signal goes inactive. The timer values range from 1 - 255 with each value equal to five seconds. Every command that accesses the drive checks the $\overline{\text{POWER OFF}}$ timing and the value of the internal timer. When these values become equal, the drive is turned off.

When the $\overline{\text{POWER OFF}}$ signal is active, the drive access command clears the internal timer value. The controller then turns on the drive and when the $\overline{\text{READY}}$ signal is active, the drive is accessed and the command is executed.

Chapter 17
Video Circuits

A simplified block diagram of the video circuits is illustrated in Figure 17-1. The video circuit is fed by the address bus, the data bus, and control signals generated by the processor and support circuits. The address bus selects locations in video memory, while the controller supports both static and dynamic memory devices. No external refresh circuits are needed.

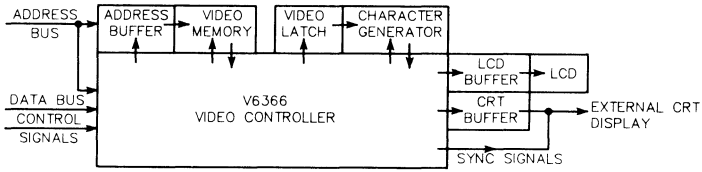


Figure 17-1. Video Block Diagram

Address Buffer — A pair of buffers isolate the computer's s-address bus and video memory address bus. This allows the CPU or video controller to execute independent read/write operations to video memory.

Video Controller — The video controller used in the computer is a Yamaha V6366 panel display and CRT display controller. It is an 84-pin surface mount device with standard 6845 CRTC functions built in, and many additional features needed for LCD operation.

Video Memory — The video memory in this computer is a 32K x 8 static RAM and has a base memory address of B8000H, standard for CGA video memory. The font memory in this computer is an 8K x 8 static RAM. The font memory base address is also B8000H and is selectable through the system control processor.

Video Latch — The video latch holds video memory addresses long enough for the character generator to output a particular character pattern.

Character Generator — The character generator produces character patterns in parallel. They are latched onto the data bus and processed by the video controller.

LCD Buffer — Control signals to the LCD panel are buffered through a 74AC244 device and data is buffered through a 74AC340 device.

Liquid Crystal Display (LCD) — The LCD is a 640 × 400 pixel display. It displays an EGA font in text mode and double scans in graphics mode.

CRT Buffer — The CRT buffer provides adequate drive for the RGBI signals required for CRT displays. The buffered signals are filtered to suppress noise and minimize ringing effects.

External CRT — The external CRT interface provides RGBI video signals, horizontal sync, and vertical sync signals.

The V6366 Video Controller

The V6366 controls LCD and CRT displays. Figure 17-2 is a block diagram of the controller. The control section handles decoding of the control signals and communication between the internal registers and the system data bus. The CRT/LCD support circuits process the video memory data with information from the internal registers to produce the CRT and LCD video signals (CGA mode). All video software that runs on CGA systems will run without need for modification on this computer. For more detailed information on this chip, consult the Yamaha PCDC V6366 Applications Manual, Catalog No. LSI-2463660.

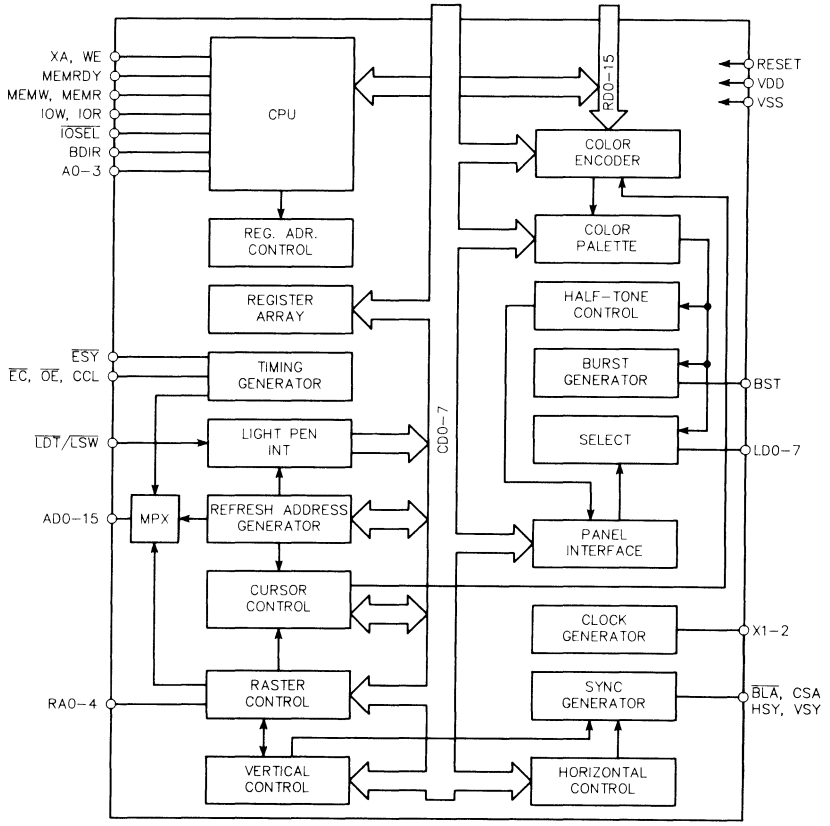


Figure 17-2. V6366 Controller Block Diagram.

Input/Output Ports

The V6366 video controller port addresses for this computer are listed in Table 17-1.

Table 17-1. Video Output/Input Port Addresses

ADDRESS	DESCRIPTION
3D0H	6845 address register (Register 0)
3D1H	6845 data port (Register 1)
3D2H	6845 address register (Register 2)
3D3H	6845 data port (Register 3)

Table 17-1 (continued). Video Output/Input Port Addresses

ADDRESS	DESCRIPTION
3D4H	6845 address register (Register 4)
3D5H	6845 data port (Register 5)
3D6H	6845 address register (Register 6)
3D7H	6845 data port (Register 7)
3D8H	Video mode register (Register 8)
3D9H	Color register (Register 9)
3DAH	Status register/Extended port (Register A)
3DBH	Light pen enable port (not supported)
3DCH	Light pen clear port (not supported)
3DDH	Register bank pointer address port (Register D)
3DEH	Register bank data port (Register E)
3DFH	Control/ID port (Register F)

NOTE: Ports 3D0H, 3D2H, 3D4H and 3D6H are the same. Ports 3D1H, 3D3H, 3D5H and 3D7H are also the same.

6845 Pointer Address and Data Ports

The address port is a write-only, 5-bit register that serves as a pointer to the 6845 data registers. To address any one of the eighteen data registers, you must first write the register number (00H-11H) into this register by executing an OUT command to one of the four 6845 controller address registers (3D0H, 3D2H, 3D4H, or 3D6H). Then either read (by executing an IN command) or write (by executing an OUT command) to one of the 6845 data ports (3D1H, 3D3H, 3D5H or 3D7H). The data port serves as the read/write port to the 6845-compatible registers of the V6366. These registers are listed in Table 17-2.

Table 17-2. 6845-Compatible Registers

REGISTER	NUMBER	DESCRIPTION
R0	00H	Horizontal total register.
R1	01H	Horizontal display register.
R2	02H	Horizontal sync position register.
R3	03H	Sync pulse width register.
R4	04H	Vertical total register.
R5	05H	Vertical total adjust register.

Table 17-2 (continued). 6845-Compatible Registers

REGISTER	NUMBER	DESCRIPTION
R6	06H	Vertical display register.
R7	07H	Vertical sync position register.
R8	08H	Interlace register.
R9	09H	Maximum raster address register.
R10	0AH	Cursor start raster register.
R11	0BH	Cursor end raster register.
R12	0CH	High-order byte start address register.
R13	0DH	Low-order byte start address register.
R14	0EH	High-order byte cursor address register.
R15	0FH	Low-order byte cursor address register.
R16	10H	High-order byte light-pen register or mouse X counter register (not supported).
R17	11H	Low-order byte light-pen register or mouse Y counter register (not supported).

Horizontal total register (R0) — This 8-bit, write-only register is used to set the horizontal scan time. This time, defined in character periods, includes all characters in a row minus one. If the total number of characters in a row is expressed as X, then the value written to this register should be equal to X-1. In the interlace mode (see Interlace register (R8) in this section), the value written to this register must be an even number. This should also be done for high resolution text mode.

Horizontal display register (R1) — This 8-bit, write-only register is used to select the number of characters in each row that will be displayed on the screen. In normal operation, this register will contain a lower value than the horizontal total register (R0). As with register R0, in high resolution text mode this should be set to an even number.

Horizontal sync position register (R2) — This 8-bit, write-only register is used to determine the position of the sync pulse in each scan line. Any value that is smaller than the total number of horizontal characters can be set. When the horizontal sync position desired is equal to X, then the value X-1 should be written to the register. When the value of R2 is increased, the display will shift to the left, decreasing the value shifts the display to the right.

Sync pulse width register (R3) — This 8-bit, write-only register is used to set the width of the horizontal and vertical sync pulses. The width of the horizontal sync is controlled by the lower four bits of the pulse width register and the width of the vertical sync is controlled by the upper four bits of the pulse width register. Increasing the value lengthens the width. When bit 7 of additional V6366 register R59 is set, the upper four bits can be written to. When bit 7 of R59 is equal to zero (cleared), the upper four bits are automatically set (equal to hex F).

NOTE: The sum of the values in register R1 and R3 must be equal to or less than the value in register R0.

Vertical total register (R4) — This 8-bit, write-only register is used to determine total time period for the vertical scan. Its value is expressed in character rows; each is normally eight scan lines. if the total number of character rows is equal to X, then the value written to this register should be equal to X-1. When bit 7 (6845 expansion register enable) of additional V6366 register R59 is set, bit 7 of this register can be written to by software. If bit of or R59 is cleared, 0 will be automatically entered for bit 7 of this register. Values for various modes can be found in Chapter 12.

Vertical total adjust register (R5) — This 5-bit, write-only register is used to adjust the total number of scan lines in the field. This value is also expressed in scan lines and can be found in Chapter 12. A zero is written to this register when a two screen LCD panel is used, as in this computer. Bits 5, 6 and 7 are not used.

Vertical display register (R6) — This 8-bit, write only register is used to determine the time period for the vertical display. This value must be less than or equal to the vertical total register value (R4). When bit 7 (6845 expansion register enable) of additional V6366 register R59 is set, bit 7 of this register can be written to by software. If bit of or R59 is cleared, 0 will be automatically entered for bit 7 of this register.

Vertical sync position register (R7) — This 8-bit, write-only register is used to control the position of the vertical sync. The value is expressed in character rows, and must be less than or equal to the vertical total register value (R4). When the value for the vertical sync position is equal to X, then the value X-1 should be written to

this register. When bit 7 (6845 expansion register enable) of additional V6366 register R59 is set, bit 7 of this register can be written to by software. If bit of or R59 is cleared, 0 will be automatically entered for bit 7 of this register.

Interlace register (R8) — This register only uses bit 0; writing anything in bits 1 through 7 will be ignored. Bit 0 specifies either interlaced or non-interlaced raster scan mode. Writing a 1 to R8 will select interlace mode, a 0 selects non-interlace mode. Combined interlaced and video mode is not supported, and CUDISP/ DISPTMG Skew functions are not supported by this chip. The LCD panel can only use non-interlace mode.

Maximum raster address register (R9) — This 5-bit, write-only register is used to control the number of scan lines in each character row. This first scan line is zero so this value will be stated as one less than the number of scan lines. Normally, eight scan lines are used for a character row, so the value would be seven. Bits 5, 6, and 7 are not used.

Cursor start raster register (R10) — This 7-bit, write-only register specifies the raster address for starting the cursor display as well as specifying the cursor display mode. The raster address for starting cursor display is set at the lower five bits of this register, and the cursor display mode is specified by its higher two bits. Refer to Table 17-3 for setting the cursor mode.

Table 17-3. Cursor Display Modes

BIT 6	BIT 5	CURSOR MODE
X	0	Cursor blinks (ON:OFF ratio = 1:1).
0	1	Cursor not displayed.
1	1	Cursor blinks (ON:OFF ratio = 1:3).

Cursor end raster register (R11) — This 5-bit, write-only register is used to define the last scan line of the cursor within the character field. Normally this value is seven. Bits 5, 6, and 7 of this register are not used.

Start address registers (R12 and R13) — These two read/write registers contain the first displayable address in video memory and

are used for scrolling the display. Register R12 contains the 6-bit high-order address and register R15 contains the 8-bit low-order address. Paging and scrolling can be performed by dynamically rewriting the contents of these registers.

Cursor address registers (R14 and R15) — These two read/write registers hold the address of the cursor in video memory. This allows the original cursor location to be retained even though the hardware may scroll or page the display. Register R14 contains the 6-bit high-order address and R15 contains the 8-bit low-order address.

Light pen or mouse registers (R16 and R17) — These two read only registers are not supported in this computer. However, mouse functions are handled by other circuits.

Video Mode Register

The video mode register can be addressed at 3D8H. It controls resolution, text/graphics mode, and other functions. It is an 8-bit, read/write port, of which data bit 6 is not used, so only 7 bits are meaningful. The data bits are somewhat interactive, and are described Table 17-4.

Table 17-4. Video Mode Register Bit Descriptions

BIT	DESCRIPTION
0	Selects text resolution for text mode. If set (Bit 0 = 1) it is in high resolution text mode. If Bit 0 = 0, low resolution text mode.
1	Selects text or graphics mode. If set (Bit 1 = 1), graphics mode selected. If Bit 1 = 1, text mode.
2	Selects color or monochrome mode. If set, monochrome mode is assumed. If equal to zero, color mode.
3	Enable video. When set, video is displayed. If equal to zero, only cursor and border are displayed. Video is usually disabled when switching video modes to prevent the display of disorderly information on the screen.

Table 17-4 (continued). Video Mode Register Bit Descriptions

BIT	DESCRIPTION
4	Selects resolution of graphics mode. If set, high resolution graphics mode selected (640 by 200). If zero, 320 by 200 low resolution mode selected.
5	Valid only in text mode, controls the blinking of text.
6	Not used.
7	Page set, valid only in Hercules mode. When set, second page is displayed.

Color Palette Register

The color pallet port is a 6-bit read/write register at address 3D9H. This register performs color specification of the border/background color in text mode or color selection in graphics mode. Table 17-5 describes the bit functions of this register.

Table 17-5. Color Palette Register Bit Descriptions

BIT	DESCRIPTION
0-3	When set (1), each bit specifies the border/background color (in text or graphics mode) according to the following scheme: BIT — COLOR 0 — Blue 1 — Green 2 — Red 3 — Intense
4	Specifies the intensity of B, G, and R in 2-bit graphics mode.
5	Selects one of two color settings (SET1 and SET2) in 2-bit graphics mode. If Bit 2 of the video mode register is set (specifying monochrome mode), both SET1 and SET2 become identical.

Status Register/Extended Port

This is an 8-bit read only port, of which bits 4, 5 and 6 are meaningless. This register is read to determine the LSI internal status. Table 17-6 describes the bit functions of this register.

Table 17-6. Status Register/Extended Port Bit Descriptions

BIT	DESCRIPTION
0	Indicates the timing of horizontal sync signals in Hercules mode or indicates the displayed timing in CGA mode.
1	Indicates contents of the light pen flip-flop. The light pen function is not supported by this computer.
2	Indicates status of the light pen switch signal. The light pen function is not supported by this computer.
3	Indicates that video dot data is being output in Hercules mode. It indicates the vertical sync signal in CGA mode.
4	Not used.
5	Not used.
6	Not used.
7	Used in Hercules mode, a one indicates vertical sync signal. A zero is output if in CGA mode.

Light Pen Registers

Light pen functions of the V6366 are not supported by this computer.

Register Bank Pointer Address and Data Ports

The address port acts as a pointer to the additional registers listed in Table 17-7. The value placed in the pointer selects one of the internal registers. The selected register can be read or written to through the register bank data port. These registers provide expanded functions that are unique to the V6366 controller.

Table 17-7. Additional V6366 Registers

REGISTER	ADDRESS	DESCRIPTION
R0-R1	00H-01H	Color palette 0 registers
R2-R3	02H-03H	Color palette 1 registers
R4-R5	04H-05H	Color palette 2 registers
R6-R7	06H-07H	Color palette 3 registers
R8-R9	08H-09H	Color palette 4 registers
R10-R11	0AH-0BH	Color palette 5 registers
R12-R13	0CH-0DH	Color palette 6 registers
R14-R15	0EH-0FH	Color palette 7 registers
R16-R17	10H-11H	Color palette 8 registers
R18-R19	12H-13H	Color palette 9 registers
R20-R21	14H-15H	Color palette 10 registers
R22-R23	16H-17H	Color palette 11 registers
R24-R25	18H-19H	Color palette 12 registers
R26-R27	1AH-1BH	Color palette 13 registers
R28-R29	1CH-1DH	Color palette 14 registers
R30-R31	1EH-1FH	Color palette 15 registers
R32	20H	Duty register
R33	21H	Horizontal size register
R34	22H	Vertical adjust register
R35	23H	Horizontal adjust register
R36	24H	Display modes
R37	25H	Register A.
R38	26H	Register B
R39	27H	Register C (see note 1)
R40	28H	Register D
R41	29H	Register E
R42	2AH	Register F
R43	2BH	Control Data
R44,R52	30H,38H	Horizontal total
R45,R53	31H,39H	Horizontal sync offset

Table 17-7 (continued). Additional V6366 Registers

REGISTER	ADDRESS	DESCRIPTION
R46,R54	32H,3AH	Vertical sync offset
R47,R55	33H,3BH	Sync width (see note 2)
R48,R56	34H,3CH	Vertical total
R49,R57	35H,3DH	Vertical adjust
R50,R58	36H,3EH	Maximum scan
R51	37H	Cursor offset/double scan
R59	3FH	Underline/double scan

NOTES

1. Registers R37 - R42 (A through F) are multiple function registers.

2. Registers R44 - R50 and R52 - R58 are 6845-compatible preset data registers.

Color Palettes (Addresses 00H-1FH) — 16 color palette selections are available with this controller. The palettes can be used as a color look-up table, or set up as a conversion table. Enabling the color palette is accomplished using bit 1 of Register C (27H). In the case of a monochrome LCD panel (such as the one used in this computer), the controller can be manipulated to provide a grey scale, hatching patterns, or combinations of grey scaling and hatching patterns to provide a pleasing display. Palette selection can be done through software or by using the FN-F9/F10 key combinations. 10 display palette formats are available by toggling through these key combinations. In addition, standard video modes 0 through 6 are available through software selection; refer to Chapter 6 for more information.

Each palette has two register locations, the low order register and the high order register. The low order register sets bits 0, 1, and 2 (red), the higher order register sets 6 bits that control blue and green (bits 0, 1, and 2 and bits 4, 5 and 6, respectively). The following figure illustrates this scheme, assume that the register bank pointer (3DDH) has selected address location 00H as a starting point for entering data. The controller has an auto-increment feature that allows consecutive data to be written to consecutive registers without needing to set the register bank pointer each time.

Table 17-8. Color Palette Control

ATA BIT	BIT NAME	LCD DISPLAY	CGA DISPLAY	EGA DISPLAY	LINEAR RGB
D^0	R^2	I	—	—	Red 2^2
D^1	R^1	$I2^3$	—	—	Red 2^1
D^2	R^0	$I2^2$	—	Sec. B	Red 2^0
D^3	G^2	$I2^1$	—	Sec. R	Green 2^2
D^4	G^1	$I2^0$	—	—	Green 2^1
D^5	G^0	—	I	Sec. G	Green 2^0
D^6	B^2	—	R	Pri. R	Blue 2^2
D^7	B^1	—	G	Pri. G	Blue 2^1
D^8	B^0	—	B	Pri. B	Blue 2^0

NOTE: As the table shows, bits 3, 4, 5, 6 and 7 of the lower order register are not used. Also, bits 3 and 7 of the higher order register are not used.

Duty Register (Address 20H) — This register sets the duty (number of lines) of a two-screen LCD panel. The number of lines equals the reciprocal of the duty value. For example, a value of C7H provides a duty of 1/200. Therefore, 200 lines will be displayed. The internal LCD of the TurbosPort 386 has 400 lines and simultaneously drives 2 lines at a time, so programming this register with the value C7H is appropriate.

Horizontal Size Register (Address 21H) — This register determines the number of pixels per line. For dual-drive two-screen displays (such as the one used in this computer), this number is set in 4-pixel units. To properly control the LCD in this computer, the value written to this register must be 9FH.

Vertical Adjust Register (Address 22H) — This register serves to correct the display positions in the vertical axis. The correction value can be specified in a range from 1 to 256 lines, but normal operation can be achieved only if the setting satisfies the conditions of the following equation:

$$C + 1 + 2D \leq RV$$

Where:
- C = The vertical adjust register (22H) correction value.
- D = The duty register (20H) value +1.
- R = The maximum raster address value (6845 register R9) +1.
- V = The vertical total register value (6845 register R4) +1.

Normally, video can be displayed on the internal LCD with a value of 0FH at this register.

Horizontal Adjust Register (Address 23H) — This register corrects the display position in the horizontal axis, selects timing for video RAM access, and sets the cursor blink cycle. Bits 0 thru 5 of this 8 bit port are used to correct the display position. The correction value can be specified in a range from 8 to 512 pixels, but normal operation can be achieved only if the setting satisfies the conditions of the following equation:

$$2T - P/8 - X \geq H + C$$

Where:
- T = The horizontal total register value (6845 register R0) +1.
- P = The horizontal size register (21H) value.
- X = Integer of 12 for dual-drive two-screen LCD (such as the one used in this computer). In low-resolution text or graphics mode, use 11.
- H = The horizontal sync position register value. (6845 register R2) +1. In low-resolution text or graphics mode, double this value.
- C = The horizontal adjust register (23H) correction value.

Bit 6 selects fast video RAM timing when it is set (1). This allows the CPU to access video RAM with minimal wait time. If the status of this bit is clear (0), wait states may have to be added to VRAM memory access cycles.

Bit 7 selects the cursor blink cycle. When bit 7 is set (1), it selects a 1:3 (ON:OFF) cursor blink cycle. This allows the slow response time of some LCDs to display the cursor properly. By setting bit 7 of this register to 1, even if data bit 5 of the 6845 cursor start raster register (R10) is 0, the cursor will blink at a rate of 1:3 (ON:OFF). If bit 7 of the horizontal adjust register is cleared, the cursor conforms to the cursor start raster register (R10) description.

Normally, a value of 01H is written to this register.

Display Mode Register (Address 24H) — This register performs multiple functions regarding monochrome displays. For instance, it can provide a conversion function to allow CGA- or Hercules-compatible software (that makes use of attribute settings) to be displayed on the LCD. This register also enables characters to be underlined. Its primary function in this computer is to set the horizontal font size (expressed in pixels). The default value at reset for this register is 02H, setting a horizontal font size of 8 pixels. Table 17-9 describes the function of all the bits in this register.

Table 17-9. Display Mode Register Bit Descriptions

BIT	DESCRIPTION
0-2	These three least-significant bits select font size according to the following logic:

BIT0	BIT1	BIT2	FONT SIZE
0	0	0	6-dot
1	0	0	7-dot
0	1	0	8-dot
1	1	0	9-dot
0	0	1	10-dot
1	0	1	8 x N Kanji.
0	1	1	16 x N Kanji

NOTE: The Kanji display is a Chinese character where N is an integer.

3-5	These bits are used to select special graphics functions. These bits expand graphics functions over those normally provided by the 6845. These functions are not implemented in the present configuration of this computer.
6	Under line enable. When this bit is set (1), it enables the underline attribute for the monochrome display.
7	Monochrome conversion enable. When this bit is set (1), it enables the computer to run CGA-compatible software. The display will occur in a monochrome format.

Register A (Address 25H) — This multiple function register allows video controller clock input selection and performs other functions, detailed in Table 17-10. At reset, 21H is written to this register for proper LCD operation.

Table 17-10. Register A Bit Functions

BIT	DESCRIPTION
0 - 1	Clock select. The status of these bits determine the source for the clock that runs the V6366 controller. The clock is selected according to the following logic:

BIT0	BIT1	CLOCK SOURCE
0	0	14.318 MHz (X1)
0	1	20.0 MHz (X2)
1	X	This mode allows a crystal to be connected to pins 6 (XTAL IN) and 7 (XTAL OUT) to form a regenerative oscillator.

X = Don't care

2	Inhibit video border. When this bit is set (1), black will always be displayed as the border. If this bit is clear (0), a CGA monitor will display a color border. However, a scan line will appear in the border if you are using an EGA monitor.
3	CPU bus size select. When this bit is clear (0), it enables an 8-bit data path between the CPU and video controller. This computer uses the 8-bit data path to read and write to font RAM. When this bit is set (1), it enables a 16-bit data path. This 16-bit data path is used to read and write to video RAM.
4	VRAM bus size select. When this bit is set (1), it enables an 8-bit data path between the V6366 video controller and VRAM. When this bit is clear (0), it enables a 16-bit data path.
5	Video RAM type. When this bit is set (1), static RAM can be used for video memory. When this bit is clear (0), dynamic RAM can be used. In this computer, the $\overline{\text{LDT}}$ line is tied high, since it uses static memory.

Table 17-10 (continued). Register A Bit Functions

BIT	DESCRIPTION
6	Software select. When this bit is set (1), Hercules software can be used (the correct attributes conversion for LCD display takes place). When this bit is clear (0), CGA software can be used. The main differences between Hercules and CGA modes are: • A different I/O register is used for the light pen register (a light pen is not supported in this computer). • Bits 0, 3, and 7 of the status register provide different functions. • The color palette register is disabled when Hercules is enabled. • Bits 0, 2, and 4 of the video-mode register are set (1) when Hercules is enabled. • Bit 7 of the video-mode register and bits 0 and 1 of the control/ID register are clear (0) when CGA is enabled. • Additional addresses are used for video memory in Hercules mode.
7	Horizontal/Vertical pulse width and vertical polarity select. When this bit is set (1), it selects the pulse width specified in the 6845-compatible sync pulse-width register (R3) and specifies negative vertical sync polarity. When this bit is clear (0), a slightly narrowed pulse width is provided and positive vertical sync polarity is selected. This computer must have a 1 written to this bit to provide proper operation of the LCD.

Register B (Address 26H) — This register selects input/output status for pin 68 (RA4), sets the function of pin 72 (AD15/GPE), enables an expanded address mode, and specifies a video page. Table 17-11 describes the function of all the bits in this register. For normal LCD operation, 80H is written to this register.

Table 17-11. Register B Bit Descriptions

BIT	DESCRIPTION
0	RA4 I/O select. In this computer, this pin is tied to ground through a 10K resistor to select CGA mode at RESET. This bit is set to (1) to output the M signal to the LCD.
1	AD15/GPE select. When this bit is set (1), address line AD15 becomes a valid address line (if static memory is used) . When this bit is clear (0), address line AD15 outputs the GPE signal (the OR'ed result of bits 0 and 1 of the control ID register). GPE is used in Hercules mode to specify a 4K segment of VRAM for text display.
2	Address expansion mode enable. When this bit is set (1), it enables 14 address lines to the 6845-compatible control circuits. When this bit is clear (0), the PC-compatible addressing scheme is selected (CGA mode requires 13 address lines and MDA requires 11). The remaining (high order) address lines are inactive (output with 0).
3	Video page select. The page mode can only be used if the upper high-order address bits (AD13 - AD15) are inactive. The status of these 3 bits select a specific page from VRAM.
4-7	Not used.

Register C (Address 27H) — This register enables the preset data registers, activates the color palette, selects consecutive addressing for graphics displays, inhibits the OFF status of the Video Enable register (high-resolution text mode), specifies the external sync mode, enables the intensity signal, and sets the bit-transmission format for the LCD display. Table 17-12 describes the function of all the bits in this register. The value written to this register normally is 8BH.

Table 17-12. Register C Bit Descriptions

BIT	DESCRIPTION
0	Preset data register enable. When this bit is set (1), the preset mode is enabled. The 6845-compatible preset data registers will allow the software and video display to operate properly even if they do not conform to PC-standards. When this bit is clear (0), values in the 6845-compatible registers will be used.
1	Color palette enable. When this bit is set (1), the color palette is used as a conversion table to change color video signals into gray scale (monochrome) signals. When this bit is clear (0), CGA-compatible video signals (RGBI) are output. With this computer, set this bit to (1) for normal operaton of the LCD.
2	Graphics consecutive addressing mode select. When this bit is set (1), screen data is stored in consecutive VRAM memory locations. In this mode, there is no need for bank switching schemes. When this bit is clear (0), consecutive addressing is disabled.
3	Normally, enabling/disabling video output is done by using the Video mode register bit 3. When this bit is set (1), it inhibits the Video mode register's control of video.
4	External sync. This bit controls synchronization of two video chips (if two are present). This computer has only one. A zero should be written in this position for normal operation.
5	Intensity enable. When this bit is set (1), R2 of the color palette becomes an intensity signal. This signal cannot be enabled with the dual-screen display used in this computer.

Table 17-12 (continued). Register C Bit Descriptions

BIT	DESCRIPTION
6-7	Bit transmission format. The status of these bits select the bit transmission format for the LCD according to the following logic:

BIT6	BIT7	TRANSMISSION TYPE
0	0	1-bit serial
1	0	2-bit even/odd
0	1	4-bit parallel
1	1	8-bit parallel

NOTE: For proper operation of the LCD, set this to 4-bit parallel data transmission.

Register D (Address 28H) — This register sets the data transmission scheme for dual-screen displays, selects the appropriate horizontal sync for CRT or LCD displays, sets the output time of the video shift clock, sets the horizontal and vertical sync polarity for the LCD, selects vertical sync output timing, and sets the horizontal sync pulse width. Table 17-13 describes the function of all the bits in this register. For control of the dual-drive dual-screen display used in this computer, a value of 6EH is written to this register.

Table 17-13. Register D Bit Descriptions

BIT	DESCRIPTION
0-1	These bits set the data transmission scheme for dual-screen displays according to the following logic:

BIT0	BIT1	TRANSMISSION SCHEME
0	0	No gray scaling or hatching.
1	0	Continuous transmission to a 1-screen display.
0	1	Simultaneous transmission to both screens of a dual-screen display.
1	1	Alternate transmission to each screen of a dual-screen display.

NOTE: For proper operation of the LCD, bit 0 and bit 1 of this register should be set to 0 and 1, respectively.

Figure 17-13 (continued). Register D Bit Descriptions

BIT	DESCRIPTION
2	RA4 I/O select. In this computer, this pin is tied to ground through a 10K resistor. This signal is also used to output the M signal to the LCD.
3	CRT/LCD horizontal sync select. When this bit is set (1), the enable clock signal for the LCD is output at pin 24 (HSY). When this bit is clear (0), horizontal sync for a CRT is output at pin 24. For proper operation of an external CRT, this kit should be clear.
4	Video shift clock output time select. When this bit is set (1), continuous data transmission to the display occurs. When this bit is clear (0), data is transmitted only during the active horizontal trace period. For normal operation of the LCD, set this bit to 0.
5	Horizontal and vertical sync polarity select. When this bit is set (1), positive polarity is enabled and the data latch (LC) and scan start (FLM) signals for the LCD are output. When this bit is clear (0), negative polarity is enabled and the horizontal and vertical sync signals are output.
6	Vertical sync output timing select. When this bit is clear (0), the vertical sync signal is synchronized with the falling edge of the horizontal sync signal. When this bit is set (1), vertical sync occurs approximately 1 microsecond after the rising edge of the horizontal sync signal. For normal operation of the LCD, this bit should be set (1).
7	Horizontal sync pulse width control. When this bit is set (1), it selects a horizontal sync pulse width that is narrower than the valid time for horizontal display data. This bit is usually cleared for use with the LCD.

Register E (Address 29H) — This register sets the enable clock synchronization, enables read operations to I/O registers 8, 9, and F, and enables the standby mode. Table 17-14 describes the function of all the bits in this register.

Table 17-14. Register E Bit Descriptions

BIT	DESCRIPTION
0-4	These bits select the cycle of the enable clock signal (ECK) for the LCD. The first ECK pulse is output during the first half of the data latch (LC) signal. Each consecutive ECK signal is output when the number of video shift clock (SCK) pulses exceeds the value specified by these bits. For example, to output ECK every 20 SCK pulses, set the value of bits 4-0 to 10011.
5	Pin 26 (ESY) I/O status. In this computer, pin 26 is the lower screen data bit 1 for the LCD.
6	This bit enables read operations to I/O registers 8, 9, and F. When this bit is set (1) and I/O register F is read, the contents of the control register will be output. When this bit is clear (0) and I/O register F is read, the V6366 identification code will be output. When this bit is clear and I/O registers 8 and 9 are read, the CPU data bus will assume high-impedance status.
7	Standby mode. When this bit is set (1), the V6366 controller is placed in the standby mode to conserve power. To cancel the standby mode: • Set bit 3 of the mode register to 0 • Set bit 7 of multi-function register E to 0 • Set bit 3 of the mode register to 1.

Register F (Address 2AH) — This register specifies the offset value for smooth scrolling. Table 17-15 describes the function of all the bits in this register. At power-up, a value of 00H is usually written to this register.

Table 17-15. Register F Bit Descriptions

BIT	DESCRIPTION
0-4	These bits specify an offset value to determine the number of lines shifted upward during smooth scrolling. Any lines beyond the upper-most limit will not be displayed. When scrolling reaches the maximum raster address, line scrolling will begin (line scrolling is controlled by the 6845-compatible start address register).
5-7	Not used. However, to achieve normal operation, these three bits must be set to 0. These bits are used for testing the IC.

Control Data Register (Address 2BH) — The contents of this register is output to the RD0-RD7 data bus on the rising edge of the VSY pin (vertical sync), in non-interlaced mode only. This data can be used for external control. For this data to be placed on the bus, the chip must be operating in non-interlace mode and the data will appear when the rising edge of vertical sync (VSY pin 25) does not occur during the vertical display time. At this point the data should be latched into memory.

NOTE: Registers 44 through 58 are 6845-compatible preset data registers. These registers hold preset data for the 6845-compatible register set. Duplicate registers are supplied to provide preset data for text or graphics modes. The preset data can be used for special functions such as executing CGA software with non-standard monitors.

For standard monitors, the 6845-compatible register set provides proper raster control. For non-standard applications, like running CGA software on a monochrome monitor, the preset data registers must provide the proper data format for the monitor in use. Initializing only the 6845-compatible register set will result in improper operation.

Since some software may switch between text and graphics modes, there are two sets of preset data registers. The first one can be set to handle text applications and the second one can be set to handle graphics applications. The registers relating to the horizontal and vertical cycles, which are fixed values determined by the monitor, must be set to the proper preset values. This preset data is valid only when bit 0 of address 27H (multi-function register C) is set to 1.

The following paragraphs describe the duplicate preset data registers, and provide sample data for displaying CGA video on the internal LCD.

Horizontal Total (Addresses 30H,38H) — This is the preset value for the 6845-compatible horizontal total register R0. In preset mode, this value is used in place of the R0 value. Its setting procedure is identical but the value must be correct for the type of display used. For text mode, write 55H. For graphics mode, write 2AH.

Horizontal Sync Offset (Addresses 31H,39H) — This preset value provides an offset value which is added to the 6845-compatible horizontal sync position register R2. If no offset is desired, write 00H to the offset value. For text mode, write F8H. For graphics mode, write FCH.

Vertical Sync Offset (Addresses 32H,3AH) — This preset value provides an offset value which is added to the 6845-compatible vertical sync position register R7. If no offset is desired, write 00H to the offset value. For text mode, write FDH. For graphics mode, write F4H.

Sync Width (Addresses 33H, 3BH) — This is the preset value for the 6845-compatible sync pulse width register R3. In preset mode, this value is used in place of the R3 value. Its setting procedure is identical but the value must be correct for the type of display used. For text and graphics, write 0AH.

Vertical Total (Addresses 34H, 3CH) — This is the preset value for the 6845-compatible vertical total register R4. In preset mode, this value is used in place of the R4 value. Its setting procedure is identical but the value must be correct for the type of display used. For text mode, write 19H. For graphics, write 67H.

Vertical Adjust (Addresses 35H, 3DH) — This is the preset value for the 6845-compatible vertical total adjust register R5. In preset mode, this value is used in place of the R5 value. Its setting procedure is identical but the value must be correct for the type of display used. For text and graphics mode, write 00H.

Maximum Scan (Addresses 36H, 3EH) — This is the preset value for the 6845-compatible maximum raster address register R9. In preset mode, this value is used in place of the R9 value. Its setting procedure is identical but the value must be correct for the type of display used. For text mode, write 0FH. For graphics mode, write 01H.

NOTE: The following paragraphs describe the preset data registers that are not duplicates.

Cursor Offset/Double Scan (Address 37H) — This register provides control over the cursor position and raster addressing in text mode. Table 17-16 describes the function of all the bits in this register. For normal operation, write 19H to this register.

Table 17-16. Cursor Offset/Double Scan Register Bit Descriptions

BIT	DESCRIPTION
0-4	Cursor offset. These bits allow an offset to be specified to provide cursor position correction. For example, if CGA software is run on a monochrome monitor, the software will attempt to display the cursor at raster addresses 6H and 7H. However, the MDA monitor requires cursor placement at raster addresses BH and CH. To correct the addresses, specify an offset to shift the raster addresses by +5.
5-6	Double-scan control. The status of these bits select the double-scan ratio (for text mode) according to the following logic:

BIT5	BIT6	DOUBLE-SCAN RATIO
0	0	Disabled
1	0	1.5 (200 line → 300 line)
0	1	1.75 (200 line → 350 line)
1	1	2.0 (200 line → 400 line)

Table 17-16 (continued). Cursor Offset/Double Scan Register Bit Descriptions

BIT	DESCRIPTION
7	Fixed cursor select. When this bit is clear (0), it enables cursor offset bits (0-4). The cursor will be shifted to a value suitable for the monitor. When this bit is set (1), the raster addresses that have been offset for the cursor are displayed at the 2H and 3H positions. This mode does not rely on cursor start/end register values.

Underline/Double-Scan (Address 3FH) — This register provides underlining and raster addressing control in graphics mode. The most significant bit of this register enables the 6845-compatible expanded register functions. Table 17-17 describes the function of all the bits in this register. For normal operation, write 67H to this register.

Table 17-17. Underline/Double-Scan Register Bit Descriptions

BIT	DESCRIPTION
0-4	Underline position. The status of these bits select the raster address position to be underlined. For a monochrome monitor the default underline position is CH. The underline position for a color monitor is 7H.
5-6	Double-scan control. The status of these bits select the double-scan ratio (for graphics mode) according to the following logic:

BIT5	BIT6	DOUBLE-SCAN RATIO
0	0	Disabled
1	0	1.5 (200 → 300 line)
0	1	1.75 (200 line → 350 line)
1	1	2.0 (200 line → 400 line)

7	6845-Compatible expanded register functions. When this bit is set (1), data can be written to the 6845-compatible registers and the preset data registers. When this bit is clear (0), the preset data registers are not valid and PC-compatible software can be run.

Control/ID Register (Register F, 3DFH)

When bit 6 of Additional V6366 Register E is set, and this port is read, the control status will be output to this port. If this port is read when bit 6 of Additional V6366 Register E is clear, the ID of the IC (C1H) will be output. Table 17-18 describes these bits in detail.

Table 17-18. Control/ID Register Bit Descriptions

BIT	DESCRIPTION
0	Graphic set enable. Writing to bit 1 of the video mode register is possible only when this bit is set to 1.
1	Page set enable. Writing to bit 7 of the video mode register is possible only when this bit is set to 1.
2-6	Not used.
7	$\overline{\text{Protect}}$. When this bit is 1, and the $\overline{\text{IOSEL}}$ pin is low, the register bank pointer address port (3DDH) and register bank data port (3DEH) can be written to.

Operating Modes

Two basic types of operating modes are possible; text and graphics. The V6366 is also capable of expanded graphics functions. However, these expanded functions may be inherently limited in the application of the controller in this computer. The mode is specified at the mode register, address 3D8H.

Text Mode

Each character is defined by two bytes. Even bytes are used for specifying the character code (the higher order addresses of an external font ROM are used). Odd bytes are used (raster address RA0-RA4) for specifying the attribute codes.

The font ROM data is input by the $\overline{EC}$ pin via the RD0-RD7 bus. When data is fetched from the font ROM, the color specified by the lower four bits of the attribute code are displayed. When no data from the font ROM is fetched, only the background color is displayed.

Graphics Modes

This chip supports several graphics modes, standard IBM PC graphics (1-Byte Mode) and expanded modes.

1-Byte Mode — There are two types of IBM PC standard graphics modes: the high-resolution 640 x 1-bit mode (200 rows of 640 pixels), where each bit of a byte has a 1:1 correspondence with screen pixels, and the medium resolution 320 x 2-bit mode (200 rows of 320 pixels), where the width of the displayed character is twice that of the high-resolution mode.

2-Byte Mode — This expanded graphics mode provides additional colors that can be simultaneously displayed. Memory capacity is double that of 1-byte mode. Refer to V6366 data sheets for specifics.

4-Byte Mode — This expanded graphics mode allows up to 256 colors that can be simultaneously displayed. Complex color images can therefore be created. Memory capacity is double that of the 2-byte mode (4 times the amount of memory required for 1-byte graphics mode). Refer to V6366 data sheets for specifics.

Table 17-19 lists the pin descriptions of the V6366 video controller Figure 17-3 is an illustration of the pinout.

Table 17-19. V6366 Video Controller Pin Functions

PIN NUMBER	SIGNAL NAME	DEFINITION
1	VCC	+5VDC
2-4	AD2-AD0	Video memory address bit 2 - video memory address bit 0. These signals can be generated either by the V6366 or through the address buffers.

Table 17-19 (continued). V6366 Video Controller Pin Functions

PIN NUMBER	SIGNAL NAME	DEFINITION
5	GND	Ground.
6	X1	Clock 1. This input pin receives the OSC clock signal from the 82C212 memory controller.
7	X2	Clock 2. This input pin receives the clock signal from the 18MHz crystal oscillator.
8	$\overline{EC}$	This output signal is used to control transmission of character font data on the RD bus.
9	CCL	Video memory latch clock. This signal gates the signals from the video memory and latches them for the character generator.
10	$\overline{XA}$	This output signal is used to enable the video address buffers.
11	WE	Write enable. This output pin enables writing to the video RAM.
12	$\overline{OE}$	Output enable. This is the output control for the video RAM.
13-20	LD7-LD0	LCD data bit 7 - LCD data bit 0. During LCD operation, these are the data bits sent to the LCD drivers. During CRT operation, LD7 - LD0 supply the red, green, blue, and intensity signals for the external monitor.
21	CSY	Composite sync.
22	BST	Color burst.
23	$\overline{BLA}$	Blank.
24	HSY	Horizontal sync. This is the horizontal sync signal used for CRT operation.

Table 17-19 (continued). V6366 Video Controller Pin Functions

PIN NUMBER	SIGNAL NAME	DEFINITION
25	VSY	Vertical sync. This is the vertical sync signal used for CRT operation.
26	$\overline{\text{ESY}}$	Not connected.
27-34	RD8-RD12	Video memory data line 8 - Video memory data line 15. These are the video memory data bus lines 8 through 15.
35	$\overline{\text{LSW}}$	Light pen switch. This pin is tied high. It is not used in this computer.
36	$\overline{\text{LDT}}$	Light pen detection. This pin is tied high. It is not used in this computer.
37	RESET	Reset. This input signal clears all internal registers of the V6366. It is generated by the 82C211 bus controller as RESET4.
38-42	RD0-RD4	Video memory data line 0 - video memory data line 4. These are the video memory data bus lines 0 through 4.
43	VCC	+5 VDC.
44-46	RD5-RD7	Video memory data line 5 - video memory data line 7. These are the video memory data bus lines 5 through 7.
47	GND	Ground.
48-51	A0-A3	Address line 0 - address line 3. These are the X address bus lines 0 through 3.
52	IOR	Input/output read. This is the active-low read signal for reading the contents of the I/O registers of the V6366.

Table 17-19 (continued). V6366 Video Controller Pin Functions

PIN NUMBER	SIGNAL NAME	DEFINITION
53	IOW	Input/output write. This is the active-low write signal for writing to the I/O registers in the V6366.
54	$\overline{\text{IOSEL}}$	Input/output select. This input pin is used as the chip select line for the V6366.
55-62	CD7-CD0	CPU data bit 7 - CPU data bit 0. These are the buffered CPU data bits from the SD data bus.
63	BDIR	Not connected.
64-68	RA0-RA4	Row address 0 - row address 4. These signals are used to address the Y matrix of the character generator.
69	MEMRDY	Memory ready. This is the video memory ready signal, indicating that video memory can be read from or written to. When low, video memory is busy.
70	MEMR	Memory read. This input pin controls reading from video memory.
71	MEMW	Memory write. This input pin controls writing to video memory.
72-84	AD15-AD3	Video memory address bit 15 - video memory address bit 3. These signals can be generated either by the V6366 or through the address buffers.

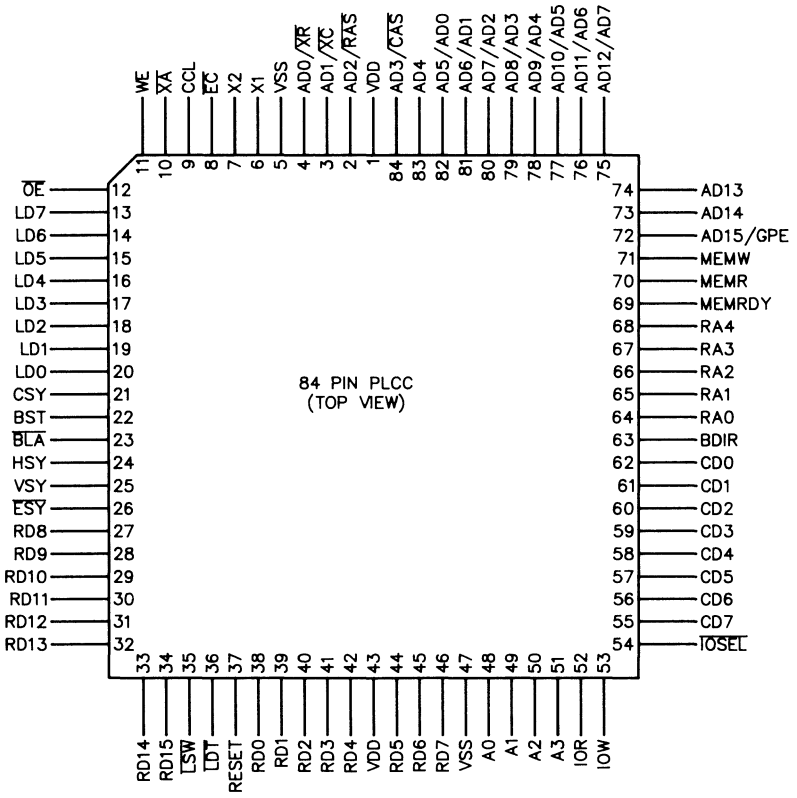


Figure 17-3. V6366 Pinout

Chapter 18

Serial and Parallel Communications

This chapter describes the hardware used for parallel and serial communications. This computer is supplied with two serial connectors and one parallel connector. (Optional cards can provide additional serial and parallel port capabilities.)

The circuits that control both the serial and parallel ports reside inside an 82C605 multifunction controller. Various buffers and drivers act as receivers and transmitters. Data transceivers handle the data and address bus interface.

Figure 18-1 illustrates the general design of the 82C605. Note that each port is handled by its own circuits. Only the configuration and decoding logic is shared between the three channels. The game enable signals are not used in this computer.

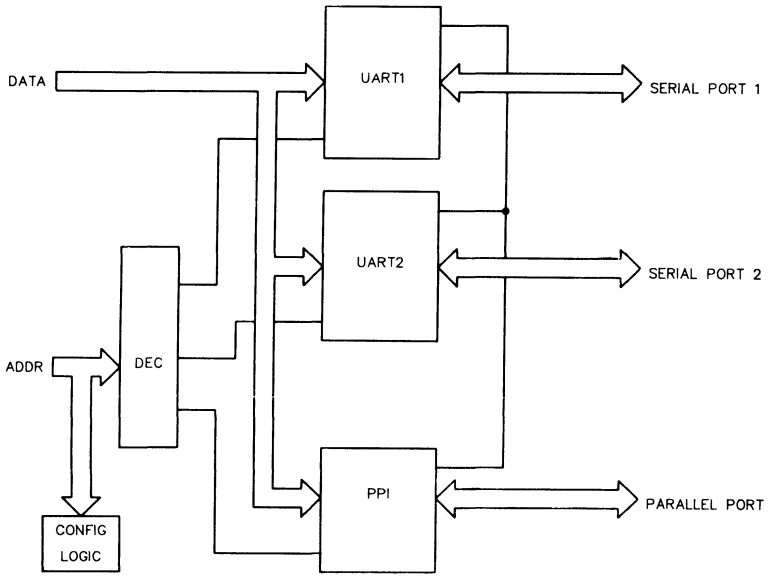


Figure 18-1. General Design of the 82C605

Serial Port

Two complete serial ports reside inside the 82C605. The circuits operate similarly to 16450 and 8250 asynchronous communications elements and are upward compatible from those devices. Each port is bidirectional, permitting full I/O communication with peripherals requiring a serial communications channel. Parallel-to-serial (data bus to device) and serial- to-parallel (device to data bus) data conversion, parity, and handshaking are all handled by each channel in the 82C605.

Universal asynchronous receiver transmitters (UARTs) in the 82C605 handle each serial communications channel. A block diagram of this device is shown in Figure 18-2.

The UART is the interface between the CPU and the serial input/output connector on the back of the computer. The controller receives information from the system data bus through an internal data buffer. You can use this data to program the UART or use the UART to transmit the data out the port. Addressing is handled by the chip select lines and address lines 0 - 2. Control logic inside the controller determines if a register is being programmed or if there is data ready to transmit or if the UART is to receive data from the port.

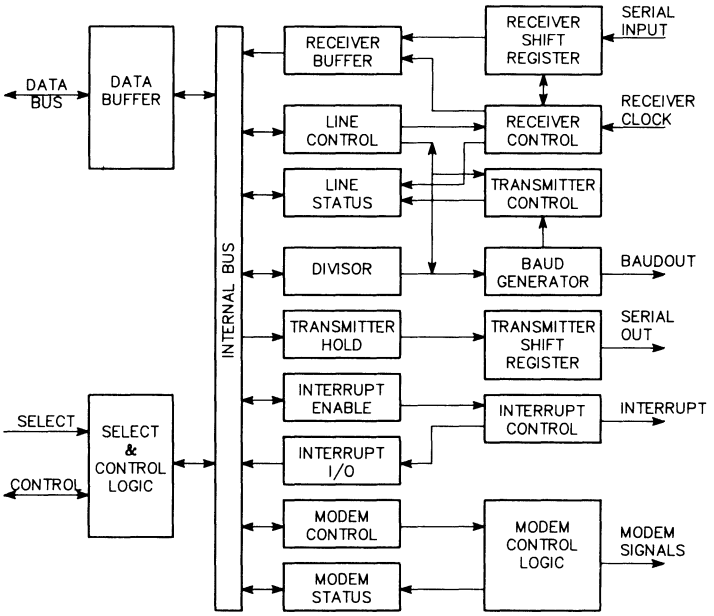


Figure 18-2. UART Block Diagram

The receiver circuits accept serial data from the connector and converts it into a parallel form before making it available for the system data bus. The transmitter circuits convert parallel data from the system data bus into serial form and then send it to the connector.

The UARTs have modem support circuits that provide control signals for serial communications across standard telephone lines. Interrupts signal the processor when the UART requires service to continue transmitting or receiving data. In this computer, the line is only used to request service while receiving data.

The UART provides a number of programmable options: manipulating interrupts, modem controls, word length, parity, number of stop bits, and baud rate. Most of these functions can be handled at the system level through the interrupts described in Chapter 9.

Handshaking

Handshaking allows two devices to communicate asynchronously. Two different methods — hardware and software — provide this function. This part of the manual is concerned only with hardware handshaking. Refer to Chapter 9 for information on software handshaking.

The four input hardware handshaking lines are $\overline{\text{DCD}}$, $\overline{\text{DSR}}$, $\overline{\text{CTS}}$ and $\overline{\text{RI}}$. The $\overline{\text{DCD}}$ (data carrier detect) line is connected to the carrier detect line at the connector. It is asserted whenever an externally connected modem detects an incoming carrier. The $\overline{\text{DSR}}$ (data set ready) line is asserted when a modem or other device is ready to establish communications with the computer. The $\overline{\text{CTS}}$ (clear to send) line is asserted when the external device is ready to accept data. The $\overline{\text{RI}}$ (ring indicator) line is asserted when an attached modem detects a ring signal on the telephone line. All signals are inverted between the connector and the UART by an inverter/receiver.

The two output hardware handshake lines are $\overline{DTR}$ and $\overline{RTS}$. The $\overline{DTR}$ (data terminal ready) line is asserted when the computer is ready to communicate with an external device. The $\overline{RTS}$ (request to send) line is asserted when the computer is ready to transmit data. These two lines are inverted and buffered by drivers between the UART and the connector.

Serial Port Programming

There are nine registers available to the programmer in the UART. Refer to Table 18-1 for the port addresses of these registers. COM1 refers to serial port 1. COM2 refers to serial port 2.

NOTE: The 82C605 must be placed in the programming mode before you can program the internal registers. Refer to "Programming the 82C605" later in this chapter.

Table 18-1. Serial Channel Register Ports

COM1 ADDRESS	COM2 ADDRESS	REGISTER
3F8H	2F8H	Receiver buffer register
3F8H	2F8H	Transmitter holding register
3F8H	2F8H	Divisor latch (least-significant byte)
3F9H	2F9H	Divisor latch (most-significant byte)
3F9H	2F9H	Interrupt enable register
3FAH	2FAH	Interrupt identification register
3FBH	2FBH	Line control register
3FCH	2FCH	Modem control register
3FDH	2FDH	Line status register
3FEH	2FEH	Modem status register

Receiver Buffer and Transmitter Holding Registers

The receiver buffer register and the transmitter holding register share the same port address. The receiver buffer register is a read-only register and the transmitter holding register is a write-only register.

These registers contain the data received and the data to be transmitted. The first data bit to be transmitted or received is the least-significant bit.

Divisor Latch Registers

The divisor latches establish the baud rate based on the input frequency of the oscillator clock to the UART. This computer uses an oscillator with a frequency of 1.8432 MHz. You can program the latches with any number to produce a value that is sixteen times the actual baud rate. The values that produce standard baud rates, along with the error percentage, are provided in Table 18-2.

NOTE: To write to the divisor latches, you must first set bit 7 in the line control register. Otherwise, the receiver buffer, the transmitter holding register, or the interrupt enable register will be written to.

Table 18-2. Baud Rate and Divisor Latch Values

DESIRED BAUD RATE	DIVISOR VALUE	ERROR PERCENTAGE
50	2304	0
75	1536	0
110	1047	0.026
134.5	857	0.058
150	768	0
300	384	0
600	192	0
1200	96	0

Table 18-2 (continued). Baud Rate and Divisor Latch Values

DESIRED BAUD RATE	DIVISOR VALUE	ERROR PERCENTAGE
1800	64	0
2000	58	0.69
2400	48	0
3600	32	0
4800	24	0
7200	16	0
9600	12	0
19200	6	0
38400	3	0
56000	2	2.86

NOTE: Although any baud rate can be programmed into the device, PC-compatible hardware and software supports only the following baud rates: 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19,200.

Interrupt Registers

The two interrupt registers work together but have different port addresses. The interrupt enable register uses four bits to enable the interrupt capabilities of the UART. Four interrupts are recognized: received data available interrupt, transmitter holding register empty interrupt, receiver line status interrupt, and modem status interrupt. These interrupts are identified by the first three bits of the interrupter identification register. Their priority level, source, and reset control are described in Table 18-3.

Table 18-3. Serial Port Interrupts

INTERRUPT REGISTER VALUE	ID PRIORITY LEVEL	SOURCE	RESET CONTROL
0	4	Clear to send data set ready, ring indicator, or received line signal detect.	Read modem status register.
1	—	None.	None.
2	3	Empty interrupt ID register or transmitter holding register.	Read interrupt ID register or write transmitter holding register, whichever caused the interrupt.
3	—	None.	None.
4	2	Received character available in the receiver buffer register.	Read the receiver buffer register.
5	—	None.	None.
6	1	Overrun, parity, or framing error, or break interrupt.	Read the line status register.
7	—	None.	None.

Line Control Register

The line control register establishes many of the parameters for serial communications. Each bit is described in Table 18-4.

Table 18-4. Line Control Register

BIT	DESCRIPTION
0 - 1	Word length: 0 = 5 bits, 1 = 6 bits, 2 = 7 bits, and 3 = 8 bits. Bit 0 is the least-significant bit.
2	Number of stop bits dependent upon the word length. Word length = 6, 7, or 8: clear (0) = 1 stop bit, set (1) = 2 stop bits. Word length = 5: clear (0) = 1 stop bit, set (1) = 1.5 stop bits.
3	Parity enabled or not: clear (0) = parity disabled, set (1) = parity enabled.
4	Even or odd parity: clear (0) = odd parity, set (1) = even parity.
5	Stick parity. When parity (bit 3) and stick parity are both enabled, parity will be opposite that indicated by bit 4 (odd/even parity). Clear (0) = stick parity disabled, set (1) = stick parity enabled.
6	Set break control. When this bit is set (1), the serial output goes low (space) and remains there until this bit is cleared (0). This feature allows the computer to alert a remote station, or vice versa, in a data communications network.
7	Allows divisor latches to be accessed: clear (0) = latch access disabled, set (1) = the latches can be read from or written to.

Modem Control Register

The modem control register handles several parameters and some of the handshaking protocol. Each bit is described in Table 18-5.

Table 18-5. Modem Control Register

BIT	DESCRIPTION
0	This bit controls the data terminal ready output: clear (0) = $\overline{\text{DTR}}$ output is high, set (1) = $\overline{\text{DTR}}$ output is low. (See the note at the bottom of this table.)
1	This bit controls the request to send output: clear (0) = $\overline{\text{RTS}}$ output is high, set (1) = $\overline{\text{RTS}}$ output is low. (See the note at the bottom of this table.)
2	This bit controls the output 1 signal: clear (0) = $\overline{\text{OUT1}}$ output is high, set (1) = $\overline{\text{OUT1}}$ output is low.
3	This bit controls the output 2 signal in same manner that the output 1 signal is controlled. However, $\overline{\text{OUT2}}$ is pulled high in this computer and not used.
4	This bit provides a local loopback condition to test the UART. It is used by the disk-based diagnostics to check the data transmit and receive circuits of the UART.
5 - 7	Not used, always clear (0).

NOTE: The outputs described in this table are at the chip and do not necessarily reflect output signals at the connector.

Line Status Register

The line status register provides a status report concerning data transfer. Each bit is described in Table 18-6.

Table 18-6. Line Status Register Report

BIT	DESCRIPTION
0	Indicates that a complete character has been received and transferred into the receiver buffer register: set (1) = character received and ready; clear (0) = receiver buffer register is empty (either no character has been received or the receiver buffer register has been read).
1	Indicates that the receiver buffer register was not read before the next character was received: set (1) = the next character has been received before the receiver buffer register was read; clear (0) = no overrun has taken place.
2	Indicates that a parity error has been detected: set (1) = parity error detected; clear (0) = no parity error.
3	Indicates that a framing error has taken place, no valid stop bit was received: set (1) = framing error detected; clear (0) = no framing error.
4	Indicates that a break interrupt has been detected. A break interrupt occurs when the received data input is held at a logical 0 state longer than a full word time (start plus data plus parity plus stop bits).
5	Indicates that the transmitter holding register is empty: set (1) = transmitter holding register empty; clear (0) = transmitter holding register receiving or has a character. This bit will produce an interrupt to the CPU if the transmitter holding register empty interrupt is enabled.
6	Indicates when the transmitter shift register is idle: set (1) = shift register is idle; clear (0) = shift register has received a character or is active (shifting data out).
7	Not used, always clear (0).

NOTE: Bits 1 - 4 produce a receiver line status interrupt when one of the conditions is detected.

Modem Status Register

The modem status register provides a status report concerning the handshaking and control lines. Each bit is described in Table 18-7.

Table 18-7. Modem Status Register Report

BIT	DESCRIPTION
0	Indicates that the $\overline{CTS}$ signal has changed state since the last time this register was read: set (1) = changed state, clear (0) = no change.
1	Indicates that the $\overline{DSR}$ signal has changed state since the last time this register was read: set (1) = changed state, clear (0) = no change.
2	Indicates that the $\overline{RI}$ signal has changed from a mark (clear or logical 0) condition to a space (set or logical 1) condition: set (1) = changed state, clear (0) = no change.
3	Indicates that the $\overline{DCD}$ signal has changed state since the last time this register was read: set (1) = changed state, clear (0) = no change.
4	Reflects the complement of the $\overline{CTS}$ input: set (1) = $\overline{CTS}$ clear (0), clear (0) = $\overline{CTS}$ set (1).
5	Reflects the complement of the $\overline{DSR}$ input: set (1) = $\overline{DSR}$ clear (0), clear (0) = $\overline{DSR}$ set (1).
6	Reflects the complement of the RI input: set (1) = $\overline{RI}$ clear (0), clear (0) = $\overline{RI}$ set (1).
7	Reflects the complement of the $\overline{DCD}$ input: set (1) = RLSD clear (0), clear (0) = $\overline{DCD}$ set (1).

NOTE: Bits 0 - 3 produce a modem status interrupt when one of the conditions is detected and the corresponding bit is set.

Serial Port Connector

The signals for each serial port are transmitted and received through a 9-pin connector. Table 18-8 lists the pin numbers and their corresponding signals. Figure 18-3 illustrates the pin positions on the connector.

Table 18-8. Serial Connector Signals

PIN NUMBER	SIGNAL NAME	DESCRIPTION
1	CD	Carrier detect
2	RXD	Receive data
3	TXD	Transmit data
4	DTR	Data terminal ready
5	GND	Signal ground
6	DSR	Data set ready
7	RTS	Request to send
8	CTS	Clear to send
9	RI	Ring indicate
Case	GND	Chassis ground

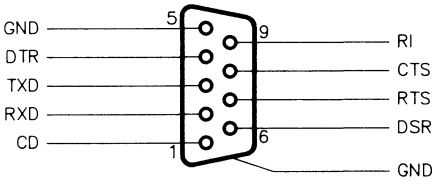


Figure 18-3. Serial Connector Pin Locations

Parallel Port

The parallel port consists of three individual I/O ports (A, B, and C). These devices interface system data to 8-bit parallel equipment. With the configuration jumpers (refer to Chapter 4) set for printer operation, I/O port A is a read/write data port that transmits data to the peripheral during a write operation. During a read operation, port A is addressed at 378 - 37F (LPT1) or 278 - 27F (LPT2). The same data previously transmitted is read. In other words, port A will not read data from the equipment. This procedure is done to make a check of the transmitted data. If transmitted data does not agree with received data, an error condition exists.

I/O port B is a read-only port addressed at 379H (LPT1) or 279H (LPT2). It allows polling of the following signals:

- Error status (parallel data bit 3)
- Select acknowledge (parallel data bit 4)
- Out of paper (parallel data bit 5)
- Data acknowledge (parallel data bit 6)
- Device busy (parallel data bit 7).

I/O port C is a read/write port that controls transfer of information to the peripheral equipment. It is addressed at 37AH (LPT1) or 27AH (LPT2). I/O port C is connected to the following parallel port control signals:

- Data strobe to peripheral equipment (parallel data bit 0)
- Automatic line feed (parallel data bit 1)
- Device initialization (parallel data bit 2)
- Device selection (parallel data bit 3)
- Parallel port interrupt enable (parallel data bit 4)

NOTE: The 82C605 must be placed in programming mode before you can program the internal registers. Refer to "Programming the 82C605" later in this chapter.

Figure 18-4 illustrates the 25-pin, D-type connector used in this computer system. This connector provides Centronics type signals for the parallel printer. Table 18-9 describes the types of signals used by this connector and their associated pin numbers.

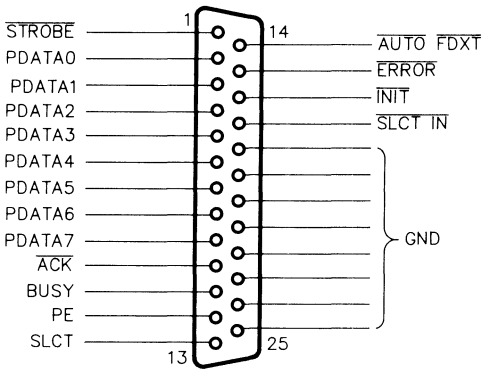


Figure 18-4. Parallel Connector Pin Locations

Table 18-9. Parallel Connector Signals

PIN NUMBER	SIGNAL NAME	DESCRIPTION
1	$\overline{\text{STROBE}}$	Strobe bit. This active low pulse indicates that the computer is transmitting parallel data. It is used to time the data being sent to the peripheral device.
2 - 9	PDATA0 - PDATA7	Data bits 0 through 7. These signals are the system buffered and latched data bits.
10	$\overline{\text{ACK}}$	Acknowledge. This active low signal indicates that the peripheral device has received the data. It can be used for hardware handshaking.
11	BUSY	Busy. This signal indicates that the peripheral device is busy and not ready to receive data. It can be used for hardware handshaking.

Table 18-9 (continued). Parallel Connector Signals

PIN NUMBER	SIGNAL NAME	DESCRIPTION
12	PE	Paper end (out). This signal indicates a peripheral fault. It is used by a printer to indicate that is is out of paper.
13	SLCT	Select. This signal informs the peripheral that it has been selected.
14	$\overline{\text{AUTO FDXT}}$	Automatic feed. This active low signal requests a paper feed by the peripheral.
15	$\overline{\text{ERROR}}$	Printer fault. This active low signal indicates that an error condition exists in the peripheral.
16	$\overline{\text{INIT}}$	Initialize. This active low signal is used to initialize the peripheral.
17	$\overline{\text{SLCT IN}}$	Select in. This active low signal is used by the peripheral to select the computer.
18 - 25	GND	Ground.

The 82C605 Multifunction Controller

The 82C605 multifunction controller incorporates two UARTs, one parallel port, and one game port decoder. The UARTs are upward compatible from the 8250 and NS16450 asynchronous communications elements used in earlier PC-compatible computers. The parallel port contains the necessary control signals for use as a Centronics-compatible (output only) printer port. The game port is not implemented.

82C605 Pinout

Figure 18-5 illustrates the locations of the pins on the 82C605. Table 18-10 lists the signal names by pin number and Table 18-11 lists the signals names alphabetically. The signal descriptions follow the tables.

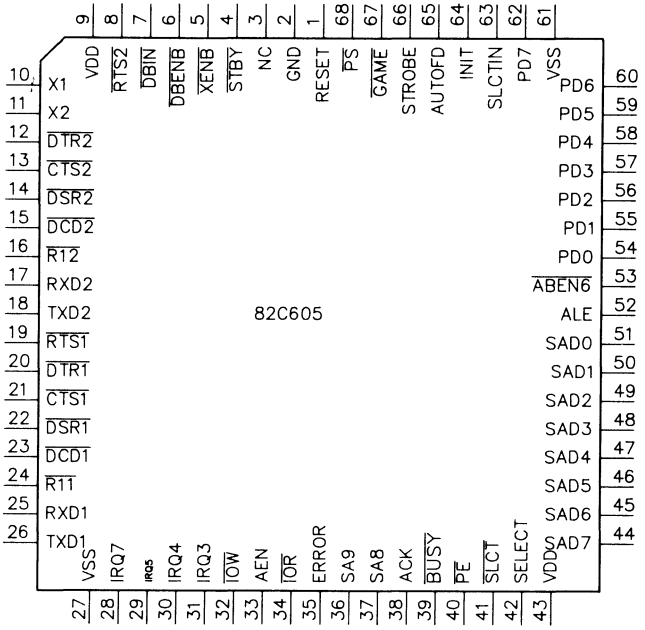


Figure 18-5. 82C605 Pin Locations

Table 18-10. 82C605 Signal Names by Pin Number

PIN	SIGNAL NAME	TYPE	PIN	SIGNAL NAME	TYPE
1	RESET	I	35	ERROR	I/O
2	GND	—	36	SA_9	I
3	—	—	37	SA_8	I
4	$\overline{STBY}$	I	38	$\overline{ACK}$	I
5	$\overline{XENB}$	I	39	$\overline{BUSY}$	I
6	$\overline{DBENB}$	O	40	$\overline{PE}$	I
7	$\overline{DBIN}$	O	41	$\overline{SLCT}$	I
8	$\overline{RTS2}$	O	42	SELECT	I
9	V_{DD}	I	43	V_{DD}	I
10	X1	I	44	SAD_7	I/O
11	$\overline{X2}$	O	45	SAD_6	I/O
12	$\overline{DTR2}$	O	46	SAD_5	I/O
13	$\overline{CTS2}$	I	47	SAD_4	I/O
14	$\overline{DSR2}$	I	48	SAD_3	I/O
15	$\overline{DCD2}$	I	49	SAD_2	I/O
16	RI2	I	50	SAD_1	I/O
17	RXD2	I	51	SAD_0	I/O
18	TXD2	O	52	$\overline{ALE}$	I
19	$\overline{RTS1}$	O	53	ABENB	O
20	$\overline{DTR1}$	O	54	PD_0	I/O
21	$\overline{CTS1}$	I	55	PD_1	I/O
22	$\overline{DSR1}$	I	56	PD_2	I/O
23	$\overline{DCD1}$	I	57	PD_3	I/O
24	RI1	I	58	PD_4	I/O
25	RXD1	I	59	PD_5	I/O
26	TXD1	O	60	PD_6	I/O
27	V_{SS}	—	61	V_{SS}	—
28	IRQ7	O	62	PD_7	I/O
29	IRQ5	O	63	SLCTIN	O
30	IRQ4	O	64	INIT	O
31	$\overline{IRQ3}$	O	65	AUTOFD	O
32	$\overline{IOW}$	I	66	$\overline{STROBE}$	O
33	$\overline{AEN}$	I	67	$\overline{GAME}$	O
34	$\overline{IOR}$	I	68	PS	I

Table 18-11. 82C605 Pins by Signal Name

PIN	SIGNAL NAME	TYPE	PIN	SIGNAL NAME	TYPE
3	—	—	62	$\overline{PD_7}$	I/O
38	$\overline{ACK}$	I	40	$\overline{PE}$	I
53	ABENB	O	68	PS	I
33	AEN	I	1	$\overline{RESET}$	I
52	ALE	I	24	$\overline{RI1}$	I
65	$\overline{AUTOFD}$	O	16	$\overline{RI2}$	I
39	$\overline{BUSY}$	I	19	$\overline{RTS1}$	O
21	$\overline{CTS1}$	I	8	$\overline{RTS2}$	O
13	$\overline{CTS2}$	I	25	RXD1	I
6	$\overline{DBENB}$	O	17	RXD2	I
7	$\overline{DBIN}$	O	37	SA_8	I
23	$\overline{DCD1}$	I	36	SA_9	I
15	$\overline{DCD2}$	I	51	SAD_0	I/O
22	$\overline{DSR1}$	I	50	SAD_1	I/O
14	$\overline{DSR2}$	I	49	SAD_2	I/O
20	$\overline{DTR1}$	O	48	SAD_3	I/O
12	$\overline{DTR2}$	O	47	SAD_4	I/O
35	$\overline{ERROR}$	I/O	46	SAD_5	I/O
67	$\overline{GAME}$	O	45	SAD_6	I/O
2	GND	—	44	SAD_7	I/O
64	$\overline{INIT}$	O	42	$\overline{SELECT}$	I
34	$\overline{IOR}$	I	41	$\overline{SLCT}$	I
32	$\overline{IOW}$	I	63	$\overline{SLCTIN}$	O
31	IRQ3	O	4	$\overline{STBY}$	I
30	IRQ4	O	66	STROBE	O
29	IRQ5	O	26	TXD1	O
28	IRQ7	O	18	TXD2	O
54	PD_0	I/O	9	V_{DD}	I
55	PD_1	I/O	43	V_{DD}	I
56	PD_2	I/O	27	V_{SS}	—
57	PD_3	I/O	61	V_{SS}	—
58	PD_4	I/O	10	X_1	I
59	PD_5	I/O	11	$\overline{X_2}$	O
60	PD_6	I/O	5	XENB	I

Clock and Control Signals

Standby Mode ($\overline{STBY}$) — Active when the main power supply is removed. It indicates that the bus interface is invalid and that no response should be made to any bus signals. This signal immediately terminates any bus signals in progress.

Power Sense ($\overline{PS}$) — This active low signal forces configuration registers into their default values and disables the decode logic. It is tied to the system RESET line in this computer.

Unit Select (SELECT) — This signal serves as a chip select signal for the 82C605 through a jumper. The position of the jumper determines whether the device is connected to the multiplexed chip address-data line 4, 5, 6, or 7, and allows more than one 82C605 in the computer at one time.

PC Bus Interface Signals

Address Latch Enable (ALE) — This signal typically is tied to the system address latches and is used to synchronize the 82C605 with the address bus.

Address Enable (AEN) — This signal, attached to the computer's AEN line, is used to gate $\overline{IOR}$ and $\overline{IOW}$ to the 82C605.

I/O Read ($\overline{IOR}$) — This signal controls all I/O port read operations.

I/O Write ($\overline{IOW}$) — This signal controls all I/O port write operations.

Address Buffer Multiplex Enable ($\overline{ABENB}$) — This signal controls the address buffer between the system address lines SA_0 - SA_7 and the 82C605 multiplexed address/data lines SAD_0 - SAD_7.

Data Bus Driver Enable ($\overline{DBENB}$) — This signal is combined with $\overline{IOW}$ and optionally with $\overline{PPID}$ (via a jumper) to produce $\overline{EN}$, which enables the bidirectional data buffer between the system data lines SD_0 - SD_7 and the 82C605 multiplexed address/data lines SAD_0 - SAD_7.

Data Bus Driver Direction ($\overline{DBIN}$) — This signal controls the direction of the bidirectional data buffer between the system data lines SD_0 - SD_7 and the 82C605 multiplexed address/data lines SAD_0 - SAD_7.

External Data Bus Enable ($\overline{XENB}$) — Internally ORed with the signal used to generate the $\overline{DBENB}$ output.

System Address 8 - System Address 9 (SA_8 - SA_9) — These two high-order address lines are used for port addressing in the 82C605.

Multiplexed Address/Data Line 0 - Address/Data Line 7 (SAD_0 - SAD_7) — These multiplexed address data lines go between the 82C605 and the signal buffers to the system data and address bus lines.

Interrupt Request Line 3 (IRQ3) — This signal typically is used by UART 2 to signal an interrupt from COM2. It can be used with UART 1.

Interrupt Request Line 4 (IRQ4) — This signal typically is used by UART 1 to signal an interrupt from COM1. It can be used with UART 2.

Interrupt Request Line 5 (IRQ5) — This signal is used to signal an interrupt from LPT2. Available if this device is programmed as the secondary I/O controller.

Interrupt Request Line 7 (IRQ7) — This signal is used to signal an interrupt from LPT1. Available if this device is programmed as the primary I/O controller.

System Reset (RESET) — This signal resets the internal registers and configuration to their powerup state.

Serial Interface Signals

Crystal Input (X1) — This input line comes from the 1.8432 MHz crystal that is used as the time base for the baud rate generators in the 82C605.

Crystal Output (X2) — This output line is used in some systems to stabilize the crystal oscillator. Not used in this computer.

Receive Data (RXD1 - RXD2) — These two signals carry the input data from the serial communications connector. RXD1 feeds UART1 and RXD2 feeds UART2.

Transmit Data (TXD1 - TXD2) — These two lines carry the output data from the UARTs to the serial communications connectors. TXD1 comes from UART1 and TXD2 comes from UART2.

Request To Send ($\overline{\text{RTS1}}$ - $\overline{\text{RTS2}}$) — This hardware handshake signal notifies the remote device that the UART is ready to transmit data. $\overline{\text{RTS1}}$ comes from UART1 and $\overline{\text{RTS2}}$ comes from UART2.

Clear To Send ($\overline{\text{CTS1}}$ - $\overline{\text{CTS2}}$) — This hardware handshake signal notifies the UART that the remote device is ready to receive data. $\overline{\text{CTS1}}$ feeds UART1 and $\overline{\text{CTS2}}$ feeds UART2.

Data Set Ready ($\overline{\text{DSR1}}$ - $\overline{\text{DSR2}}$) — This hardware handshake signal informs the UART that the remote device is ready to transmit data. $\overline{\text{DSR1}}$ feeds UART1 and $\overline{\text{DSR2}}$ feeds UART2.

Data Terminal Ready ($\overline{\text{DTR1}}$ - $\overline{\text{DTR2}}$) — This hardware handshake signal notifies the remote host device that the UART, acting as a terminal to a host, is ready to transmit data. $\overline{\text{DTR1}}$ comes from UART1 and $\overline{\text{DTR2}}$ comes from UART2.

Data Carrier Detect ($\overline{\text{DCD1}}$ - $\overline{\text{DCD2}}$) — This input line, sometimes identified as CD (carrier detect), notifies the UART that the remote device, usually a modem, is receiving a carrier signal. $\overline{\text{DCD1}}$ feeds UART1 and $\overline{\text{DCD2}}$ feeds UART2.

Ring Indicator ($\overline{RI1}$ - $\overline{RI2}$) — This input line notifies the UART that the remote device is receiving a RING signal from the telephone line. $\overline{RI1}$ feeds UART1 and $\overline{RI2}$ feeds UART2.

Parallel Interface Signals

Error (ERROR) — This input signal indicates a fault in the connected peripheral.

Acknowledge (ACK) — This input hardware handshake signal indicates the connected peripheral has received the transmitted data.

Busy ($\overline{BUSY}$) — This input hardware handshake signal indicates the connector peripheral is not ready to receive data.

Paper End ($\overline{PE}$) — This input signal indicates the connected peripheral has a fault, usually a paper out condition in a printer.

Select ($\overline{SLCT}$) — This input signal indicates the peripheral is selected, usually an on-line indication.

Data Line 0 - Data Line 7 (PD_0 - PD_7) — These lines carry the bit data between the connected peripheral and the 82C605.

Select In (SLCTIN) — This output signal selects the connected peripheral. The peripheral will respond with $\overline{SLCT}$.

Initialize (INIT) — This signal usually initializes the connected peripheral.

Auto Feed (AUTOFD) — This signal line usually causes the connected peripheral to generate a line feed (equivalent to 0AH) after receiving a carriage return (0DH). Some parallel devices do not recognize this signal.

Strobe (STROBE) — This signal line tells the connected peripheral that the bit data lines have stabilized data. When asserted, the connected peripheral will receive the data and acknowledge it with the ACK signal.

Game Port Signal

Game Port Enable ($\overline{\text{GAME}}$) — This signal is not used in this computer. It can be used to enable a game peripheral (such as a joystick) or as a programmable decoder for any peripheral device.

Power and Ground Signals

+5 VDC (VDD) — +5 VDC power supply voltage input.

Ground (VSS and GND) — Circuit ground.

Programming the 82C605

To program any of the features in the 82C605, you must first place it in the programming mode. This section describes how to put the 82C605 in its programming mode, how to modify a register, and how to take the 82C605 out of its programming mode. Normally, the firmware of the computer handles these functions.

To place the 82C605 in program mode:

1. Send OH to port 2FAH.
2. Send OFFH to port 3FAH.
3. Send 36H to port 3FAH.
4. Send 40H to port 3FAH.
5. Send OBFH to port 2FAH.

To modify a register:

1. Send the appropriate index (internal address) register number (80H, 81H, 84H, 85H, 86H, or 88H) to port 100H. Do not modify any other register.
2. Read the byte contained in the register from port 101H.

3. Modify the appropriate bit in the byte, leaving the other bits intact.

4. Send the byte back to port 101H.

Table 18-12 provides a summary of the functions in the 82C605.

To take the 82C605 out of programming mode:

1. Send 8FH to port 100H.

2. Send FFH to port 101H.

Table 18-12. 82C605 Register Functions

INDEX REGISTER	BIT(S)	VALUE	FUNCTION
80H	3	0	Disable parallel port
80H	3	1	Enable parallel port
80H	2	0	Disable modem
80H	2	1	Enable modem (not used in this computer)
80H	1	0	Disable serial port
80H	1	1	Enable serial port
81H	6	0	Only read data written to parallel port
81H	6	1	Make parallel port bidirectional
81H	5	1	Enable CTS on serial port (serial port enabled)
81H	4	1	Enable DSR on serial port (serial port enabled)
81H	4	1	Enable CTS on modem (modem enabled)
81H	3	1	Enable CD on serial port (serial port enabled)
81H	3	1	Enable DSR on modem (modem enabled)
81H	3	0	Enable CD on modem (modem enabled)

Table 18-12 (continued). 82C605 Register Functions

INDEX REGISTER	BIT(S)	VALUE	FUNCTION
84H	0 - 7	FEH	Set serial port to COM1
84H	0 - 7	BEH	Set serial port to COM2
85H	0 - 7	FEH	Set modem to COM1
85H	0 - 7	BEH	Set modem to COM2
86H	0 - 7	DEH	Set parallel port to LPT1
86H	0 - 7	9EH	Set parallel port LPT2
88H	6 - 7	OH	Disable IRQ3
88H	6 - 7	2H	Tie IRQ3 to serial port
88H	6 - 7	3H	Tie IRQ3 to modem
88H	4 - 5	OH	Disable IRQ4
88H	4 - 5	2H	Tie IRQ4 to serial port
88H	4 - 5	3H	Tie IRQ4 to modem
88H	2 - 3	OH	Disable IRQ5
88H	2 - 3	2H	Tie IRQ5 to serial port
88H	2 - 3	3H	Tie IRQ5 to modem
88H	0 - 1	OH	Disable IRQ7
88H	0 - 1	2H	Tie IRQ7 to modem
88H	0 - 1	3H	Tie IRQ7 to parallel port

Chapter 19

ROM-Based Tests and Error Messages

This chapter describes the ROM-based self-tests, selectable tests, and the error messages you may encounter. Self-test routines initialize the major devices in the computer when it is turned on. The selectable tests exercise specific circuits more intensely than the self-tests. In most cases, the selectable tests can be used to troubleshoot major circuits in the computer. An error message is a short line of text that is displayed if a test detects an error condition. The nature of the message is conclusive to the device or circuits that failed.

Self-Tests

On powerup the computer automatically executes a series of tests to verify that all of the circuits are in a starting configuration. These tests also check basic operational functions of the computer. The following systems are exercised during the self test sequence:

- Setup menu configuration
- Crystal frequencies
- Interrupt controllers
- DMA (direct memory access) controllers
- Disk drive controllers
- Disk drives
- CPU
- ROM
- RAM.

Various other controllers may be exercised during the self-tests. For example, video, keyboard, refresh, and any other controller will be initialized during the self-test sequence. When the tests are finished, the computer will display an opening message or start the automatic boot procedure (autoboot). If autoboot to the floppy drive is started, a disk must be placed in the drive within about 20 seconds.

Selectable Tests

The Monitor program provides a great deal of flexibility and power to the user. Among the commands available (most of which are described in Chapter 6) is the TEST command. Some of the tests that can be performed from this command are more comprehensive than the powerup self-tests and can help to quickly evaluate a number of apparent problems in memory, the disk drives, and the keyboard.

Test Menu

The ROM-based tests should be run after the computer is first turned on and before any application or other programs are run. The reason for this is that some programs modify the interrupts that are used during the tests. This can produce unpredictable and deceptive results.

The tests are identified in the Monitor program's command summary as "Extended diagnostics." To display a menu from which the tests can be executed, type TEST at the Monitor program prompt and press the ENTER key. The menu is illustrated in Figure 19-1.

```
CHOOSE ONE OF THE FOLLOWING

1. DISK READ TEST
2. KEYBOARD TEST
3. BASE MEMORY TEST
4. EXPANSION MEMORY TEST
5. POWER-UP TEST
6. EXIT

ENTER YOUR CHOICE:
```

Figure 19-1. Test Menu

To execute a test, press the number that corresponds to the desired test. The program will immediately start executing that test.

The Disk Read Test

The disk read test continuously reads the first sector of track 0 on the default drive. This sector is the first sector of the boot track and must be read successfully for the computer to be able to boot a disk. By continuously reading the first boot sector, you can determine if you have a reliable disk/drive combination. Errors that occur during this test can be attributed to the disk, the drive, or both. If the test immediately displays an error message, use a known formatted disk before assuming you have a hardware-oriented error.

NOTE: Even though the boot track is being read, the disk does not need an operating system on it. Any formatted disk, whether it has the operating system installed on it or not, can be used for this test. On formatted data disks, sector 1 of track 0 contains a short routine that, when booted and executed, will display No system and halt the computer. This message indicates that the disk is formatted but does not contain an operating system.

The disk read test is non-destructive. This means that the operating system or data on the disk you are testing will not be erased or destroyed by the test. During operation, the disk read test will display the test count, indicating the number of times that the first track has been successfully read.

The Keyboard Test

The keyboard test allows you to check most of the keys on the keyboard and generate the key scan codes described in Chapter 8. Some of the keys may cause the computer to exit the test rather than produce a display.

This test displays several items of information. Each time a key that produces a printable ASCII character is pressed, the entire display will be filled with that character. The character scan code will be displayed in the upper right corner of the screen if the key produces one. If the key does not produce a code, the last code displayed will remain on the screen.

This test is also provides a quick way to fill the screen with data to check an external video monitor.

The Base Memory Test

During the powerup self-tests, only the first and last bank of base memory is tested. This test checks all base system memory and video memory for errors. While video memory is being tested, various patterns will appear on the screen.

The test produces a short click sound and displays a number indicating the current memory test count. This count is updated each time the test is successfully completed. The upper-right corner of the screen displays the current 64K bank of memory being tested in hexadecimal format. If an error is detected, the test stops and information about the error is displayed. The message reports the address where the error was detected, along with the bit number and integrated circuit (chip) number.

The Expansion Memory Test

The expansion memory test checks memory above the 1-megabyte boundary. The test produces a short click sound and displays a number indicating the current memory test count. This count is updated each time the test is successfully completed. The upper-right corner of the screen displays the current 64K bank of memory being tested in hexadecimal format. If an error is detected, the test stops and information about the error is displayed. The message reports the address where the error was detected, along with the bit number. An integrated circuit (chip) number is not displayed for expansion memory.

The Powerup Test

The powerup test repeatedly runs the self-tests described at the beginning of this chapter. These tests check all major circuits. While the powerup self-tests are not all-inclusive, some random fault conditions can be detected using this test. The disk-based diagnostics are more useful in exhaustively and repeatedly testing key elements of the computer. For more information on the disk-based diagnostics, refer to the end of this chapter.

Exiting the Test Menu

All ROM-based selectable tests will display a status screen while running. To exit any test, press the ESC key. With the exception of the keyboard test, the ESC key first halts the test. Pressing it a second time returns the program to the test menu. When you have exited to the menu, press the 6 to return to the Monitor program. The Monitor program prompt will be displayed on the screen.

NOTE: Do not use the CTRL-ALT-INS or CTRL-ALT-DEL key sequence to exit any of the tests. Some of the tests modify the interrupt vector table or the interrupts themselves. Exiting directly to the Monitor program can leave the computer in an unknown state, which often produces unpredictable results.

Error Messages

An error message may be generated whenever a test algorithm (self-test or selectable test) detects that a specific circuit or component in the computer is not functioning as expected. The displayed message indicates the type of failure. This section presents the error messages in three categories:

- Error messages related to disk drives
- Error messages related to the Setup/Configuration program
- General error messages.

The error summary lines that may be displayed with certain error messages are explained at the end of this section.

Error Messages Related to Disk Drives

```
+++ DISK ERROR: Drive not ready! +++
```

This error message is displayed when the system attempts to boot an operating system but no disk has been placed in the floppy disk drive or the hard disk (if attempting to boot) has not been PREPed or formatted. A faulty floppy disk or controller can also cause this error message to be generated.

When booting a floppy disk drive, make sure that a disk is correctly placed in the drive. Try to boot the system and notice if the error occurs again. If it does, contact your Zenith Data Systems service representative.

If the error occurs with a hard disk drive, try to determine if the problem exists in a piece of equipment other than the drive itself. Perform diagnostic tests (ROM-based or disk-based if necessary) on the hard disk controller. If no conclusive results are obtained from these checks, you may need to PREP and format the hard disk drive (assuming it has been done before).

```
+++ DISK ERROR: Bad disk controller! +++
+++ DISK ERROR: DMA Overrun! +++
```

These error messages usually indicate a fault in the disk controller. Contact you Zenith Data Systems service representative.

```
+++ DISK ERROR: CRC error! +++
+++ DISK ERROR: Invalid address mark! +++
+++ DISK ERROR: Sector not found! +++
```

These error messages can happen when booting an operating system from a disk. They can also result from using an unformatted or defective disk, or from a faulty disk drive. Most often, this condition can be corrected by using another disk.

```
+++ DISK ERROR: Seek failure! +++
```

This error message indicates that the computer cannot read the disk or cannot read track 0. This error can be caused by a faulty disk, or by the failure of the disk controller, or by a fault in the disk drive.

```
+++ DISK ERROR: Disk not bootable! +++
```

This error message occurs when the computer attempts to boot a disk that does not contain an operating system. The portion of the disk that normally contains the operating system may have been damaged.

```
+++ DISK ERROR: Invalid data read! +++
```

This error message indicates that the computer has read data from a disk but cannot interpret the data. The data on the disk may have lost its integrity due to exposure to an external magnetic field or due to physical damage of the disk itself. Try reading data from a known good disk and observe if the error is repeated.

```
+++ DISK ERROR: Data corrected! +++
```

This error message indicates that the drive has read data from a disk but the data was not all defined as either logic 0 or logic 1. However, the data was adequately defined to allow the controller to determine each data bit's state.

```
+++ DISK ERROR: Cannot reset drive! +++
```

This error message indicates a probable fault in the disk drive. Contact your Zenith Data Systems service representative.

```
+++ DISK ERROR: Must run SETUP to boot from Winchester! +++
```

This error message indicates that a valid drive type must be specified in by the Setup/Configuration program. Refer to the configuration information in Chapter 5 for the correct procedure.

Error Messages Related to The Setup/Configuration Program

```
+++ ERROR: Please replace the backup battery! +++
```

This error message indicates that the real-time clock has lost power. If this message occurs, press the ESC key to allow the system to boot.

```
+++ ERROR: Bad configuration information found in CMOS! +++
```

This message indicates that the CPU has found invalid data present in the special CMOS device. The Setup/Configuration program will be accessed automatically by the computer after this error is detected. At this time, system configuration data may be determined and stored in the CMOS device.

```
+++ERROR: Base memory size error! SETUP:512K ACTUAL:640K+++
```

The CPU always checks the size of memory indicated in the Setup/Configuration program and compares it with the amount of actual memory installed in the computer. If the two figures are different, an error message similar to the one shown is generated. Be sure to update the Setup/Configuration program if additional memory is installed in your computer.

```
+++ ERROR: Expansion memory size error! SETUP: 15296K ACTUAL: 0K +++
```

An error message similar to the one shown is generated if the CPU detects a difference between the amount of expansion memory indicated by the Setup/Configuration program and the actual expansion memory installed. In the example, the actual memory is zero. This could mean that a major failure of memory devices has occurred or that the memory has been removed from the system.

General Error Messages

```
+++ ERROR: Memory parity failure! +++
+++ ERROR: System control processor failure! +++
```

These messages indicate that the system control processor is not responding to system requests. The keyboard and system will not function when this occurs.

```
+++ ERROR: CMOS memory failure! +++
```

This error message indicates that the CMOS RAM chip cannot be properly read or written to.

```
+++ ERROR: CPU failure! +++
+++ ERROR: ROM checksum failure! +++
```

When any of these error messages occur, it indicates a fatal condition. Either the CPU is defective or the BIOS code in the ROM has been corrupted. In either case the device must be replaced. Contact your Zenith Data Systems service representative.

```
+++ ERROR: Parity hardware failure! Address 00000:123D,
           Chip Uxxx +++
```

An error message similar to the one shown indicates that the CPU is unable to read or write to the system RAM. The routine will attempt to display a number to indicate which chip appears to have failed.

```
+++ ERROR: Keyboard not responding or not connected! +++
```

Normally this error message is caused by the keyboard being disconnected. If this is not the case, the problem may be in the keyboard controller IC or the keyboard cable.

```
+++ ERROR: Timer interrupt failure! +++
```

This error message indicates possible failure of interrupt control or timer logic. Contact your Zenith Data Systems service representative.

```
+++ ERROR: RAM failure! Address:00000:123D, Bit n, Chip Uxxx +++
```

This error message will be generated if the system detects a faulty memory device in its environment. The address of the memory location, the specific bit number, and the chip number will be displayed.

```
+++ Divide by zero! +++
```

This error message indicates an invalid mathematical operation and returns program execution to the Monitor program. The system may be booted using the boot command.

```
+++ Non-maskable interrupt! +++
```

The NMI error message indicates a possible program execution error or a power interruption. In either case, the system must be reset before it will continue to operate. Press the Ctrl-Alt-Del key sequence or if necessary turn the computer off, wait a few seconds, and then turn the computer back on.

```
+++ Overflow! +++
```

This error message occurs when the computer attempts to execute a mathematical calculation that exceeds the capabilities of the CPU. The system must be reset before it will continue to operate. Press the Ctrl-Alt-Del key sequence or if necessary turn the computer off, wait a few seconds, and then turn the computer back on.

```
+++ Wild interrupt! +++
```

This error message indicates that an interrupt generated by an unknown source was received. Program execution returns to the Monitor program. The system may be booted using the boot command.

```
+++ Wild hardware interrupt! +++
```

This error message occurs when a specific device generates an interrupt request that is out of sequence or unexpected. Program execution returns to the Monitor program. The system may be booted using the boot command.

Error Summary Lines

```
--- Errors found! Please press <ESC> to continue ---
```

When an error has been detected during the ROM-based tests, this error summary message will be displayed on the last line of the screen. Press the ESC key to clear the screen and continue the test.

```
--- Fatal Error: Cannot Continue! ---
```

This message indicates that a major failure has occurred during a test. The test can not continue when a fatal error is encountered. This error is most commonly caused by defective RAM at address 0000H.

Disk-Based Diagnostics

The optional disk-based diagnostic package (Model CB-31-3) is a supplemental testing package available for this computer. These tests contain comprehensive routines to check both the system core circuits and the interfaces required to operate peripheral devices. These tests can be very useful if you should encounter any difficulties with the computer.

The ROM-based tests check all functions of the hardware required to load and run the disk-based diagnostics. For a complete discussion on how to configure and execute disk-based diagnostics, refer to the disk-based diagnostic manual.

Appendix A
80386 Instruction

This appendix is an overview of the instruction set available for the 80386 microprocessor. This appendix does not contain technical details concerning the specific syntax or execution details of these instructions. Additional material may be obtained by referring to the related publications listed in Chapter 1 of this manual.

80386 Instruction Set

Data Transfer Instructions

General Purpose

MOV	Move operand
PUSH	Push operand onto stack
POP	Pop operand off stack
PUSHA	Push all registers onto stack
POPA	Pop all registers off stack
XCHG	Exchange Operand, Register
XLAT	Translate

Conversion

MOVZX	Move byte or Word, DWord, with zero extension
MOVSX	Move byte or Word, DWord, sign extended
CBW	Convert byte to Word, or Word to DWord
CDW	Convert word to DWord
CDWE	Convert word to DWord extended
CDQ	Convert DWord

Input/Output

IN	Input operand from I/O space
OUT	Output operand to I/O space

Address Object

LEA	Load effective address
LDS	Load pointer into D segment register
LES	Load pointer into E segment register

LFS	Load pointer into F segment register
LGS	Load pointer into G segment register
LSS	Load pointer into S(stack) segment register

Flag Manipulation

LAHF	Load A register from flags
SAHF	Store A register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack
PUSHFD	Push Eflags onto stack
POPFD	Pop Eflags off stack
CLC	Clear carry flag
CLD	Clear direction flag
CMC	Compliment carry flag
STC	Set carry flag
STD	Set direction flag

Arithmetic Instructions

Addition

ADD	Add operand
ADC	Add with carry
INC	Increment operand by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition

Subtraction

SUB	Subtract operand
SBB	Subtract with borrow
DEC	Decrement operand by 1
NEG	Negate operand
CMP	Compare operands
AAS	ASCII adjust for subtraction

Multiplication

MUL	Multiply double/single precision
IMUL	Integer multiply
AAM	ASCII adjust after multiply

Division

DIV	Divide unsigned
IDIV	Integer divide
AAD	ASCII adjust after division

String Instructions

MOVS	Move byte or Word, DWord string
INS	Input string from I/O space
OUTS	Output string to I/O space
CMPS	Compare byte or Word, DWord string
SCAS	Scan byte or Word, DWord string
LODS	Load byte or Word, DWord string
STOS	Store byte or Word, DWord string
REP	Repeat
REPE/ REPZ	Repeat while equal/zero
RENE/ REPNZ	Repeat while not equal/not zero

Logical Instructions

Logicals

NOT	"NOT" operand
AND	"AND" operand
OR	"Inclusive OR" operand
XOR	"Exclusive OR" operand
TEST	"Test" operand

Shifts

SHL/SHR	Shift logical left/right
SAL/SAR	Shift arithmetic left/right
SHLD/SHRD	Double shift left/right

Rotates

ROL/ROR	Rotate left/right
RCL/RCR	Rotate through carry left/right

Bit Manipulation Instructions

Single Bit Instructions

BT	Bit test
BTS	Bit test and set
BTR	Bit test and reset
BTC	Bit test and compliment
BSF	Bit scan forward
BSR	Bit scan reverse

Bit String Instructions

IBTS	Insert bit string
XBTS	Exact bit string

Program Control Instructions

Conditional Transfers

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

Unconditional Transfers

CALL	Call procedure/task
RET	Return from procedure/task
JMP	Jump

Iteration Controls

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

Interrupts

INT	Interrupt
INTO	Interrupt if overflow
IRET	Return from interrupt
CLI	Clear interrupt enable
SLI	Set interrupt enable

High Level Language Instructions

BOUND	Check array bounds
ENTER	Setup parameter block for entering procedure
LEAVE	Leave procedure

Protection Model

SGDT	Store global descriptor table
SIDT	Store interrupt descriptor table
STR	Store task register
SLDT	Store local descriptor table
LGDT	Load global descriptor table
LIDT	Load interrupt descriptor table
LTR	Load task register
LLDT	Load local descriptor table
ARPL	Adjust requested privilege level
LAR	Load access rights
LSL	Load segment limit
VERR/VERW	Verify segment for reading or writing
LMSW	Load machine status word (lower 16 bits of CR0)
SMSW	Store machine status word

Processor Control Instructions

HLT	Halt
WAIT	Wait until busy# negated
ESC	Escape
LOCK	Lock bus

Index

Index

Index

Index

Index

M

Index

Index

W

X

Z