IBM®

SA12-7009-00

Printed in U.S.A.

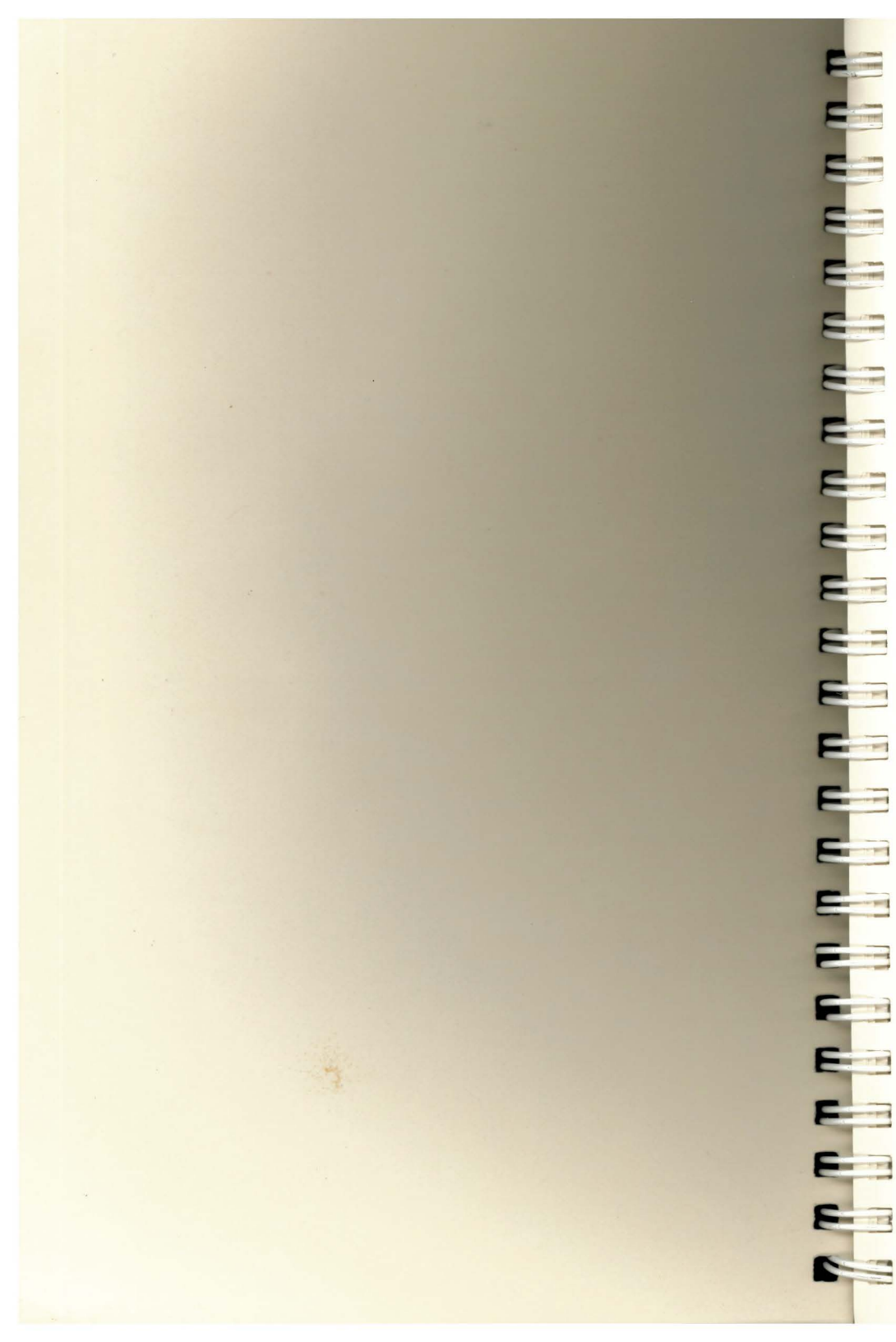
IBM®

Technical Reference

IBM 7690 Clinical Workstation

SA12-7009-00

49F6182



IBM®

Technical Reference

IBM 7690 Clinical Workstation

ATTENTION

THE IBM TYPE 7690 CLINICAL WORKSTATION IS DESIGNED FOR END USER DATA COLLECTION AND RETRIEVAL APPLICATIONS. SINCE IT IS NOT INTENDED FOR DIRECT ATTACHMENT TO PATIENTS, OR FOR DIRECT MONITORING OF ANY PATIENT FUNCTIONS, IBM HAS NOT SOUGHT U.S. FOOD AND DRUG ADMINISTRATION APPROVAL REQUIRED FOR SUCH DEVICES.

First Edition, April 1990

Federal Communications Commission (FCC) Statement

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Instructions to User: If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the computer with respect to the receiver.
- Move the computer away from the receiver.
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

Properly shielded and grounded cables and connectors must be used for connection to peripherals in order to meet FCC emission limits. Proper cables are available from IBM authorized dealers. IBM is not responsible for any radio or television interference caused by using other than recommended cables or by unauthorized modifications to this equipment. It is the responsibility of the user to correct such interference.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:

How to Identify and Resolve Radio-TV Interference Problems

This booklet is available from the following:

Consumer Assistance and
Small Business Division
Room 254
1919 M St. NW
Washington, DC 20554
Tele (202) 632-7000

FOB Public Contact Branch
Room 725
1919 M St. NW
Washington, DC 20554
Tele (202) 634-1940

Preface

This publication describes the components of the IBM 7690 Clinical Workstation (Type 7690) and their interactions.

The information in this publication is for reference and is intended to be used by hardware and software designers, programmers, engineers, and others who need to understand the design and operation of the Type 7690.

This manual is organized as follows:

- **Section 1, "System Board/Interface Adapter"** describes the functions provided by the system board and interface adapter.
- **Section 2, "Power Supply"** provides electrical input/output specifications as well as operational information about the power supply.
- **Section 3, "Keyboard"** describes the hardware, function, encoding, and layout of the keyboard.
- **Section 4, "Bar Code Feature"** provides a description of the bar code feature's hardware and details its programmable options.
- **Section 5, "Touch Panel"** describes the programming interface to the touch panel's device driver (LEDTOUCH.SYS).
- **Section 6, "System BIOS"** describes the basic input/output system (BIOS) and the interrupt interfaces. This section also contains a BIOS memory map, descriptions of vectors with special meanings, and a set of low memory maps.
- **Section 7, "Programming Tips and Techniques"** contains pointers that should be considered when developing software for the Type 7690.

An Index is also provided.

Prerequisite Publications

- *Guide to Operations* for the IBM 7690 Clinical Workstation

Suggested Related Publications

- *BASIC for the IBM Personal Computer*
- *Disk Operating System (DOS)*
- *Hardware Maintenance and Service* for the IBM 7690 Clinical Workstation
- *Macro Assembler for the IBM Personal Computer.*

Contents

Figures

Section 1. System Board/Interface Adapter

General Description

The IBM 7690 Clinical Workstation (Type 7690) is a highly integrated system based on the IBM Personal System/2® Model 25 architecture. The system uses five gate arrays on the system board: two for microprocessor support, two for video support, and one for diskette controller support.

The Type 7690 uses an integral adapter to interface its liquid crystal display (LCD), touch panel, keyboard, and bar code feature to the system board. This adapter is called the IBM 7690 Interface Adapter, and is connected to the system board through the Input/Output (I/O) channel.

Because the system board and interface adapter work together to provide system functions, they are described together in this section.

The major functional areas of the system are:

- 8086-2 microprocessor and its support logic
- 640K read/write memory standard
- 64K read-only memory (ROM) on the system board
- 16K ROM on the interface adapter
- I/O channel
- Integrated I/O functions
  - Diskette drive interface
  - Serial port
  - Parallel port
  - Keyboard port
  - Monochrome/graphics subsystem
  - Touch panel interface
  - Bar code feature interface

System dc power and a 'power good' signal from the power supply enter the system board through a 12-pin connector. Video signals are supplied to a 10-pin connector on the system board, and are then carried to the interface adapter for further processing. The LCD and

Personal System/2 and PS/2 are registered trademarks of the International Business Machines Corporation.

touch panel are connected to the interface adapter through a single, 50-pin connector.

The keyboard is internally connected to the interface adapter where keyboard and bar code input is combined and sent to the system board's keyboard port through the external keyboard cable. Other connectors on the system board are for attaching serial and parallel devices, and the diskette drive.

Four 62-pin card-edge sockets are attached to an expansion bus (the bus adapter) that is connected to the system board. The I/O channel is extended to these four I/O slots. One of these I/O slots is dedicated to the interface adapter, leaving three expansion slots.

System Functional Diagram

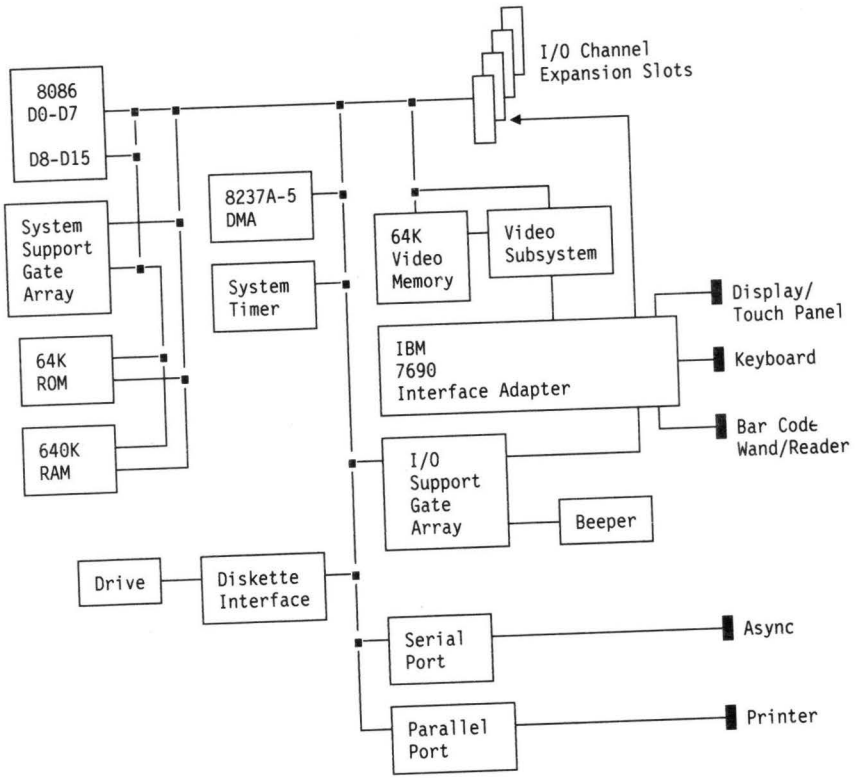


Figure 1-1. System Functional Diagram

Microprocessor

The Intel 8086-2 is a 16-bit microprocessor with a 16-bit external data bus, operating at 8 MHz. The microprocessor supports the same 20-bit addressing as IBM Personal Computers that use the 8088 microprocessor. The Type 7690 uses a 16-bit data bus with the system's read-only and read/write memory, and an 8-bit data bus for all I/O and direct memory access (DMA) operations.

The memory read and write are 16-bit operations and take four clock cycles of 125 ns, with no wait states, for a cycle time of 500 ns. Normal I/O operations are 8-bit operations and take eight clock cycles, including four wait states, for a cycle time of 1 µs. A signal on the I/O channel, I/O channel ready (I/O CH RDY), allows slower devices to add more wait states to I/O and DMA operations (see "I/O Channel" later in this section).

Logic has been added to the system board to support options for the IBM Personal Computer family. This includes converting 16-bit operations to sequential 8-bit operations, inserting wait states into all I/O and DMA operations, and delaying microprocessor cycles to ensure address setup times greater than or equal to the 8088-based systems.

The 8086-2 supports 16-bit operations, including multiply and divide, and 20-bit addressing to access 1M (M = 1,048,576) of address space. Memory is mapped as follows:

Hex Address	Function
00000 - 9FFFF	640K Read/Write Memory
A0000 - BFFFF	Video Buffer
C0000 - C3FFF	Interface Adapter POST and BIOS
F0000 - FFFFF	System ROM

Figure 1-2. Memory Map

The microprocessor is supported by two high-function support devices: a system support gate array and an I/O support gate array.

System Support Gate Array

The system support gate array contains the bus controller, the memory controller and parity checker, the wait-state generator and bus conversion logic, the system clock generator, and the DMA page register and support logic.

Bus Controller

The Type 7690 has two bus masters on the local bus: the microprocessor, and the system support gate array. The gate array seizes the bus to generate memory refresh and DMA bus cycles. It controls the request/grant line (CPU RQ/GT), which is connected to the microprocessor.

The gate array generates a request pulse to the microprocessor to get control of the bus. The microprocessor then gives control of the bus and pulses the same line to indicate a grant.

Memory Controller and Parity Checker

The memory controller functions of the gate array control memory and generate the memory refresh. Memory must be refreshed once every 4 ms. Memory refresh takes nine clock cycles of 125 ns through a dedicated refresh channel within the gate array.

The parity checker function generates the parity bits for system memory and activates the '-parity check' signal when a parity error is detected. Only the read/write memory on the system board is checked.

Bus Conversion Logic and Wait-State Generator

The bus conversion logic converts word transfers to I/O devices into 2-byte transfers. Sixteen-bit transfers are only supported for the system's read-only and read/write memory.

Additional logic generates the needed wait states for the microprocessor bus cycles to I/O devices. This logic monitors the 'I/O CH RDY' line to determine the wait states required.

System Clock Generator

The system clock generator uses a 48 MHz input that is internally divided to give the output clock signal of 8 MHz with a 33% duty cycle. It also generates a 1.84 MHz signal for the serial port.

The clock generator generates the 'reset' signal after sensing the 'power good' signal from the power supply.

I/O Support Gate Array

The I/O support gate array contains the chip select (CS) logic, keyboard, and I/O ports. It also contains the interrupt controller.

Chip Select Logic

The gate array controls the following CS signals on the system board:

- Serial port
- Diskette controller
- Video controller
- Parallel port.

Each select line can be disabled by programming the System Board Control register, address hex 65. When the bit is set to 1, that function of the system board is enabled. Bit 7, parallel port output enable, enables the parallel port's output drivers.

When the signal is enabled, the CS signal is generated to start a read or write operation, and the read and write signals to the I/O channel are blocked. When the signal is disabled, the CS signal is not generated, and all read and write operations are directed to the I/O channel. This register is read/write.

Bit	Function
7	Parallel Port Output Enable
6	Reserved = 0
5	Reserved = 0
4	Serial CS
3	Diskette CS
2	Video CS
1	Parallel Port CS
0	Reserved = 0

Figure 1-3. System Board Control Register, Hex 65

Keyboard Controllers

The keyboard controller (on the system board) receives serial data from the keyboard micro-controller (in the keyboard) and checks the parity. The data is then presented to the system at the interface's output buffer, I/O port hex 60.

System Timer

The system timer is an 8253 programmable interval timer/counter, or equivalent, that functions as an arrangement of four external I/O ports (hex 0040 through 0043). It receives its 1.19 MHz clock from the I/O support gate array. Three ports are treated as counters; the fourth, address hex 0043, is a control register for mode programming.

The content of a selected counter may be latched without disturbing the counting operation by writing to the control register. If the content of the counter is critical to a program's operation, a second read is recommended for verification.

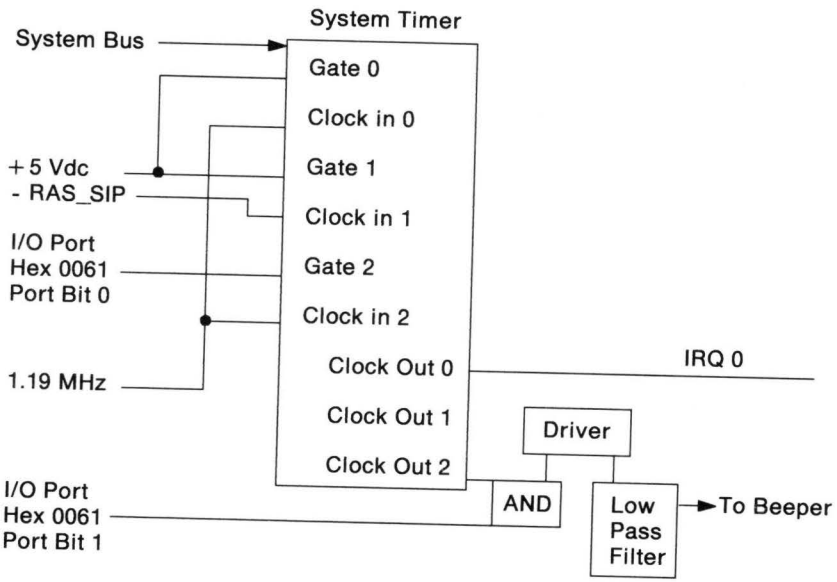


Figure 1-4. System Timer Block Diagram

The three timers are programmable and are used by the system as follows:

Channel 0 is a general-purpose timer providing a constant time base.

Channel 0	**System Timer**
GATE 0	Tied High
CLK IN 0	1.19 MHz OSC
CLK OUT 0	**IRQ 0**

Channel 1 is for internal diagnostic tests.

Channel 1	**Diagnostic**
GATE 1	Tied High
CLK IN 1	-RAS_SIP from system support gate array
CLK OUT 1	Not connected

Channel 2 supports the tone generation for the audio.

Channel 2	**Tone Generation**
GATE 2	Controlled by bit 0 at port hex 61
CLK IN 2	1.19 MHz OSC
CLK OUT 2	To the beeper data of the I/O support gate array

I/O Ports

The output port hex 60 is used by BIOS to store keystrokes. The output port hex 61 is used as beeper control, enables I/O channel check and parity checks, and is read/write. The input port hex 62 contains the status of certain signals on the system board and is read-only. The bit descriptions of ports hex 61 and hex 62 follow.

Bit	Function
7	Reserved
6	Reserved
5	-ENA I/O CH CK
4	-ENA RAM Parity CK
3	Reserved
2	Reserved
1	Beeper Data
0	Timer 2 Gate (to beeper)

Figure 1-5. Output Port, Hex 61

Port 61

Bit	**Connected to**	**Description**
7 - 6	Not connected	Reserved as 0.
5	-ENA I/O CH CK	When set to 1, this bit stops -I/O CH CK from generating a non-maskable interrupt (NMI). When cleared to 0, an NMI is generated when -I/O CH CK goes active.
4	-ENA RAM PARITY CK	When set to 1, this bit stops a memory parity error from generating an NMI. When cleared, an NMI is generated when a memory parity error is sensed.
3 - 2	Not connected	Reserved as 0.
1	Beeper Data	This bit gates the output of timer 2. It is used to disable the timer's sound source or modify its output. When set to 1, this bit enables the output; when cleared, it forces the output to zero.

Port 61 Bit	Connected to	Description
0	Timer 2 Gate	This line is routed to the timer input at GATE 2. When this bit is cleared to 0, the timer operation is halted. This bit and bit 1 (beeper data) control the operation of the timer's sound source.

Bit	Function
7	Parity
6	I/O CH CK
5	Timer 2 Output
4	Reserved
3	Reserved
2	Reserved
1	Coprocessor installed
0	Reserved

Figure 1-6. Input Port, Hex 62

Port 62 Bit	Connected to	Description
7	Parity	When set to 1, this bit indicates that a memory parity error has occurred.
6	I/O CH CK	When set to 1, this bit indicates that -I/O CH CK is active.
5	Timer 2 Output	This bit shows the status of the Timer 2 Output.
4 - 2	Not connected	Reserved.
1	Coprocessor installed	When set to 1, this bit indicates that the Math Coprocessor is installed.
0	Not connected	Reserved.

DMA Controller

The 8237 DMA controller and its support logic in the gate array support four channels of 20 address bit DMA. It operates at 4 MHz and handles only 8-bit data transfers. The DMA channel assignments and page register addresses are:

Level	Assignment
DRQ0	Not Available
DRQ1	Not Used
DRQ2	Diskette
DRQ3	Not Used

Figure 1-7. DMA Channel Assignments

Hex Address	DMA Page Register
080	Channel 2
081	Channel 3
082	Channel 1
087	Channel 0

Figure 1-8. DMA Page Register Addresses

Three of the DMA channels (1, 2, and 3) are available on the I/O bus and support high-speed data transfers between I/O devices and memory without microprocessor intervention.

DMA data transfers take six clock cycles of 250 ns, or 1.5 µs. I/O CH RDY can be pulled inactive to add wait states to allow more time for slower devices.

Interrupts

The interrupt controller has eight levels of interrupt that are handled according to priority in the I/O support gate array. Two levels are used only on the system board. Level 0, the highest priority, is attached to Channel 0 of the timer/counter and provides a periodic interrupt for the timer tick. Level 1 is shared by the keyboard and the pointing device. It is handled by a BIOS routine pointed to by interrupt hex 71. Level 2 is available to the video subsystem, and level 7

is available to the parallel port; however, the BIOS routines do not use interrupts 2 and 7. Level 4 is used by the serial port.

This controller also has inputs from the coprocessor's '-interrupt', the memory controller's '-parity', and the '-I/O channel check' signals. These three inputs are used to generate the NMI to the 8086-2.

The following table shows the hardware interrupts and their availability to the I/O channel.

Level	System Board	I/O Channel
NMI	Parity Check and Coprocessor	I/O Channel Check
IRQ0	Timer Channel 0	Not Available
IRQ1	Keyboard Pointing Device	Not Available
IRQ2	Video	Available
IRQ3	Not Used	Available
IRQ4	Serial Port	Available
IRQ5	Not Used	Available
IRQ6	Diskette Drive	Available
IRQ7	Parallel Port	Available

Note: Interrupts are available to the I/O channel if they are not enabled by the system board function normally assigned to that interrupt.

Figure 1-9. Hardware Interrupt Listing

Interrupt Sharing

A standardized hardware design concept has been established to enable multiple adapters to share an interrupt level. The integrated adapters do not use interrupt sharing. The following describes this design concept and discusses the programming support required.

Design Overview

Most interrupt supporting adapters hold the IRQ line inactive and then drive the line active to cause an interrupt. In contrast, the shared interrupt hardware design allows the IRQ line to float high. Each adapter on the line may cause an interrupt by pulsing the line low. The leading edge of the pulse arms the interrupt controller; the trailing edge of the pulse causes the interrupt.

Each adapter sharing an interrupt level must monitor the IRQ line. When any adapter pulses the line, all other adapters on that interrupt must not issue an interrupt request until they are rearmed.

If an adapter's interrupt is active when it is rearmed, the adapter must reissue the interrupt. This prevents lost interrupts in case two adapters issue an interrupt at exactly the same time and an interrupt handler issues a Global Rearm after servicing one of them.

The following diagram shows the shared interrupt hardware logic.

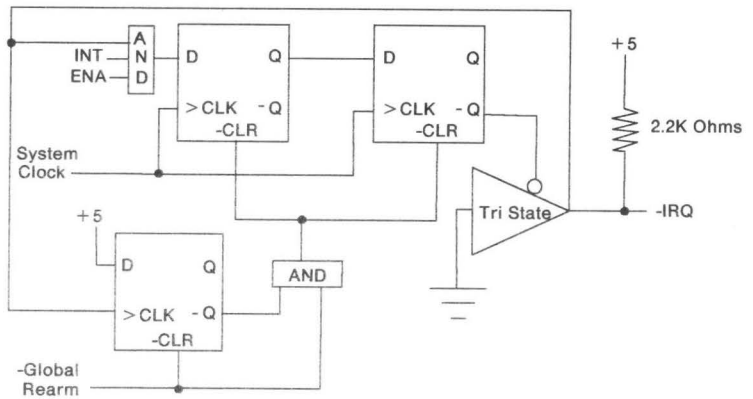


Figure 1-10. Shared Interrupt Hardware Logic

Program Support

The interrupt-sharing program support described in the following provides for an orderly means to:

- Link a task's interrupt handler to a chain of interrupt handlers
- Share the interrupt level while the task is active
- Unlink the interrupt handler from the chain when the task is deactivated.

Linking onto the Chain: Each newly activated task replaces the interrupt vector in low memory with a pointer to its own interrupt handler. The old interrupt vector is used as a forward pointer and is stored away at a fixed offset from the new task's interrupt handler. This method of linking means the last handler to link is the first one in the chain.

Sharing the Interrupt Level: When the new task's handler gains control as a result of an interrupt, the handler reads the contents of the adapter's Interrupt Status register to determine whether its adapter caused the interrupt. If its adapter did cause the interrupt, the handler services the interrupt, disables (clears) the interrupts (CLI), and writes to address hex 02F*X*, where *X* corresponds to interrupt levels 2 through 7. Each adapter in the chain decodes the address, which results in a Global Rearm. The handler then issues a nonspecific End of Interrupt (EOI) and finally issues a Return from Interrupt (IRET). If its adapter did not cause the interrupt, the handler passes control to the next interrupt handler in the chain.

Unlinking from the Chain: To unlink from the chain, a task must first locate its handler's position within the chain. By starting at the interrupt vector in low memory and using the offset of each handler's forward pointer to find the entry point of each handler, the chain can be methodically searched until the task finds its own handler. The forward pointer of the previous handler in the chain is replaced by the task's pointer, removing the handler from the chain.

Note: If the handler cannot locate its position in the chain or, if the signature of any prior handler is not hex 424B, it must not unlink.

Error Recovery: If the unlinking routine discovers that the interrupt chain has been corrupted, an unlinking error recovery procedure must be in place. Each application can incorporate its own unlinking error procedure into the unlinking routine. One application may choose to display an error message requiring the operator to either correct the situation or reset the system. The application, however, must not unlink.

Precautions

The following precautions must be taken when designing hardware or programs that use shared interrupts.

- Hardware designers should ensure that the adapters:
  - Do not power up with an interrupt pending or enabled.
  - Do not generate interrupts that are not serviced by a handler. Generating interrupts when a handler is not active to service them causes that interrupt level to lock up. The design concept relies on the handler to clear its adapter's interrupt and issue the Global Rearm.
  - Can be disarmed so that they do not remain active after their application has terminated.
- Programmers should:
  - Ensure that their programs contain a short routine that can be executed with the AUTOEXEC.BAT to disable their adapter's interrupts. This precaution ensures that the adapters are deactivated for a system reboot that does not clear memory.
  - Treat words as words, not as bytes.

    Note: Remember that data is stored in memory using the Intel format (word hex 424B is stored as hex 4B42).

Interrupt Chaining Structure

```
ENTRY:  JMP         SHORT PAST        ; Jump around structure
        FPTR        DD      0         ; Forward Pointer
        SIGNATURE   DW      424BH     ; Used when unlinking to identify
                                      ;  compatible interrupt handlers
        FLAGS       DB      0         ; Flags
        FIRST       EQU     80H       ; Flags for being first in chain
        JMP         SHORT   RESET
        RES_BYTES   DB      DUP 7(0)  ; Future Expansion
PAST:   ...                           ; Actual start of code
```

The interrupt chaining structure is a 16-byte format containing FPTR, SIGNATURE, RES_BYTES, and a Jump instruction to a reset routine. It begins at the third byte from the interrupt handler's entry point. The first instruction of every handler is a short jump around the structure to the start of the routine.

Except for those residing in adapter ROM, handlers designed for interrupt sharing must use hex 424B as the signature to avoid cor-

rupting the chain due to misidentification of an interrupt handler. Because each handler's chaining structure is known, the forward pointers can be updated when unlinking.

The flag indicates that the handler is first in the chain and is used only with interrupt 7. The Reset routine disables the adapter's interrupt and then does a Far Return to the operating system.

ROM Considerations

Adapters with interrupt handlers residing in ROM must store the forward pointer in latches or ports on the adapter. If the adapter is sharing interrupt 7, it must also store a First. Storing this flag is necessary because its position in the chain may not always be first.

Because the forward pointer is not stored in the third byte, these handlers must contain a signature of hex 00.

Examples

In the following examples, note that interrupts are disabled before passing control to the next handler on the chain. The next handler receives control as if a hardware interrupt had caused it to receive control. Note also that the interrupts are disabled before the nonspecific EOI is issued, and are not re-enabled in the interrupt handler. This ensures that the IRET is executed (at which point the flags are restored and the interrupts re-enabled) before another interrupt is serviced. This protects the stack from excessive buildup.

Interrupt Handler Example

```
OUR_CARD  EQU     xxxx              ; Location of our card's interrupt
ISB       EQU     xx                ; Interrupt bit in our cards interrupt
                                    ;  control/status register
REARM     EQU     2F7H              ; Global Rearm location for interrupt 7
SPC_EOI   EQU     67H               ; Specific EOI for interrupt 7
EOI       EQU     20H               ; Nonspecific EOI
OCR       EQU     03CH              ; Location of interrupt controller
                                    ;   operational control register
IMR       EQU     21H               ; Location of interrupt mask register

MYSEG     SEGMENT PARA
          ASSUME  CS:MYSEG,DS:DSEG
ENTRY     PROC    FAR
          JMP     SHORT PAST        ; Entry point of handler
FPTR      DD      0                 ; Forward Pointer
SIGNATURE DW      424BH             ; Used when unlinking to identify
                                    ;  compatible interrupt handlers
          FLAGS   DB      0         ; Flags
          FIRST   EQU     80H
          JMP     SHORT   RESET
RES_BYTES DB      DUP 7(0)          ; Expansion
PAST:     STI                       ; Actual start of handler code
          PUSH    ...               ; Save needed registers
          MOV     DX,OUR_CARD       ; Select our status register
          IN      AL,DX             ; Read the status register
          TEST    AL,ISB            ; Our card caused the interrupt?
          JNE     SERVICE           ; Yes, branch to service logic
          TEST    CS:FLAGS,FIRST    ; Are we the first ones in?
          JNZ     EXIT              ; If yes, branch for EOI and Rearm
          POP     ...               ; Restore registers
          CLI                       ; Disable interrupts
          JMP     DWORD PTR CS:FPTR ; Pass control to next handler on chain

SERVICE:  ...                       ; Service the interrupt
EXIT:
          CLI                       ; Disable the interrupts
          MOV     AL,EOI
          OUT     OCR,AL            ; Issue nonspecific EOI
          MOV     DX,REARM          ; Rearm our card
          OUT     DX,AL
          POP     ...               ; Restore registers
          IRET
RESET:    ...                       ; Disable our card
          RET                       ; Return Far to operating system
ENTRY:    ENDP
          MYCSEG  ENDS
          END     ENTRY
```

Linking Code Example

```
          PUSH     ES
          CLI                          ; Disable interrupts
; Set forward pointer to the value of the interrupt vector in low memory
          ASSUME   CS:CODESEG,DS:CODESEG
          PUSH     ES
          MOV      AX,350FH            ; DOS get interrupt vector
          INT      21H                 ;
          MOV      SI,OFFSET CS:FPTR   ; Set offset of our forward pointer
                                       ;  in an indexable register
          MOV      CS:[SI],BX          ; Store the old interrupt vector
          MOV      CS:[SI+2],ES        ;  in our forward pointer
          CMP      ES:BYTE PTR[BX],CFH ; Test for IRET
          JNZ      SERVECTR
          MOV      CS:FLAGS,FIRST      ; Set up first in chain flag
SERVECTR: POP      ES
          PUSH     DS
; Make interrupt vector in low memory point to our handler
          MOV      DX,OFFSET ENTRY     ; Make interrupt vector point to our
                                       ;  interrupt handler
          MOV      AX,SEG ENTRY        ; If DS not = CS, get it and
          MOV      DS,AX               ;  put it in DS
          MOV      AX,250FH            ; DOS set interrupt vector
          INT      21H                 ;
          POP      DS
; Unmask (enable) interrupts for our level
SET7:     IN       AL,IMR              ; Read interrupt mask register
          AND      AL,07FH             ; Unmask interrupt level 7
          OUT      IMR,AL              ; Write new interrupt mask
          MOV      AL,SPC_EOI          ; Issue specific EOI for level 7
          OUT      OCR,AL              ;  to allow pending level 7 interrupts
                                       ;   (if any) to be serviced
          STI                          ; Enable interrupts
          POP      ES
```


Unlinking Code Example

```
          PUSH     DS
          PUSH     ES
          CLI                          ; Disable interrupts
          MOV      AX,350FH            ; DOS get interrupt vector
          INT      21H                 ; ES:BX points to the first in the chain
          MOV      CX,ES               ; Pickup segment part of interrupt vector
; Are we the first handler in the chain?
          MOV      AX,CS               ; Get code seg into comparable register
          CMP      BX,OFFSET ENTRY     ; Interrupt vector in low memory
                                       ;  pointing to our handlers offset?
          JNE      UNCHAIN_A           ; No, branch
          CMP      AX,CX               ; Vector pointing to our handler's segment?
          JNE      UNCHAIN_A           ; No, branch
; Set interrupt vector in low memory to point to the handler
;  pointed to by our pointer
          PUSH     DS
          MOV      AX,CS:FPTR
          MOV      DX,WORD PTR CS:FPTR ; Set offset of interrupt vector
          MOV      DS,WORD PTR CS:FPTR[2] ; Set segment of interrupt vector
          MOV      AX,250FH            ; DOS set interrupt vector
          INT      21H
          POP      DS
          JMP      UNCHAIN_X
UNCHAIN_A: ; CX = FPTR segment, BX = FPTR offset
          CMP      ES:[BX+6],4B42H     ; Is handler using the appropriate
                                       ;  conventions (is SIGNATURE = 424BH?)
          JNE      exception           ; No, invoke error exception handler
          LDS      SI,ES:[BX+2]        ; Get FPTR's segment and offset
          CMP      SI,OFFSET ENTRY     ; Is this forward pointer pointing to
                                       ;  our handler's offset?
          JNE      UNCHAIN_B           ; No, branch
          MOV      CX,DS               ; Is this forward pointer pointing to
          CMP      AX,CX               ;  our handler's segment?
          JNE      UNCHAIN_B           ; No, branch
; Located our handler in the chain
          MOV      AX,WORD PTR CS:FPTR ; Get our FPTR's offset
          MOV      ES:[BX+2],AX        ; Replace FPTR offset pointing to us
          MOV      AX,WORD PTR CS:FPTR[2] ; Get our FPTR's segment
          MOV      ES:[BX+4],AX        ; Replace FPTR segment pointing to us
          MOV      AL,CS:FLAGS
          AND      AL,FIRST
          OR       ES:[BX+6],AX        ; Replace offset of FPTR of handler
          JMP      UNCHAIN_X
UNCHAIN_B: MOV     BX,SI               ; Move new offset to BX
          PUSH     DS
          PUSH     ES
          JMP      UNCHAIN_A           ; Examine the next handler in the chain
UNCHAIN_X: STI                         ; Enable interrupts
          POP      ES
          POP      DS
```

Read/Write Memory

The system board supports 640K bytes of read/write memory. The first 128K consists of four 64K by 4-bit and two 64K by 1-bit chips that are installed in sockets provided on the system board.

The next 512K (from 128K to 640K) is arranged as two banks of 256K by 9-bit single-inline packages (SIPs). All read/write memory is parity-checked.

The System Board RAM Control/Status register, hex 6B, is part of the system gate array and may be used to remap memory. Remapping occurs when the power-on self-test (POST) senses memory on the I/O channel that is in contention with system memory. Also, if a failure in the first 128K is sensed, POST remaps the remainder of memory to allow the system to operate.

Bit	Function
7	Parity Check Pointer 1 = Lower 128K failed 0 = Upper 512K failed
6	-Enable RAM, 90000-9FFFF
5	-Enable RAM, 80000-8FFFF
4	-Enable RAM, 70000-7FFFF
3	-Enable RAM, 60000-6FFFF
2	-Enable RAM, 50000-5FFFF
1	-Enable RAM, 40000-4FFFF
0	Remap Low Memory

Figure 1-11. System Board RAM Control/Status Register

ROM

The system board has 64K by 8-bits of ROM or erasable programmable read-only memory (EPROM). Two module sockets are provided; both sockets have 32K by 8-bits of ROM. This ROM contains POST, BIOS, dot patterns for 128 characters in graphics mode, and a diskette bootstrap loader. The ROM is packaged in 28-pin modules.

The interface adapter contains 16K by 8-bits of ROM, which contains the POST and BIOS routines for the LCD, touch panel, and bar code feature.

I/O Channel

The I/O channel is an extension of the 8086-2 microprocessor bus that is demultiplexed, repowered, and enhanced by the addition of interrupts and DMA functions.

The I/O channel contains:

- An 8-bit, bidirectional data bus
- Twenty address lines
- Six levels of interrupt
- Control lines for memory and I/O read and write
- Clock and timing lines
- Three channels of DMA control lines
- Memory-refresh control lines
- A channel check line
- Power and ground for the adapters.

Three voltage levels are provided for I/O cards:

- +5 Vdc ±5%
- +12 Vdc ±5%
- -12 Vdc ±10%.

The 'I/O CH RDY' line is available on the I/O channel to allow operation with slow I/O or memory devices. I/O CH RDY is made inactive by an addressed device to lengthen the operation. For each clock cycle that the line is held low, one wait state is added to the I/O and DMA operations.

I/O devices are addressed using mapped I/O address space. The channel is designed so that over 64,000 device addresses are available to the adapters on the I/O channel.

The following is the I/O address map for the Type 7690. Hex 0100 to FFFF are available for use by adapters on the I/O channel, except for those addresses noted.

Hex Range	Device
0000-001F	DMA Controller, 8237A-5
0020-003F	Interrupt Controller
0040-005F	Timer
0060-0062	I/O Ports
0063-006F	System Board/Control and Status
0080-008F	DMA Page Registers
00A0-00AF*	Interrupt Controller Extension
0378-037F	Parallel Port
03C0-03DF	Video Subsystem
03F0-03F7	Diskette
03F8-03FF	Serial Port
F300-F303	Touch Panel Registers
F304-F305	LCD Controller Registers

Note: I/O Addresses, hex 000 to 0FF, are reserved for the system board I/O.

* The NMI mask can be set and reset through system software as follows:

Write hex 80 to I/O address hex A0 (enable NMI)
Write hex 00 to I/O address hex A0 (disable NMI)

Figure 1-12. I/O Address Map

The '-I/O channel check' signal (-I/O CH CK) causes an NMI to the microprocessor.

Connectors

The I/O channel is repowered to provide sufficient power for both 62-pin connectors, assuming two low-power Schottky (LS) loads per slot. IBM adapters typically use only one load per adapter.

The following figure shows the pin numbering and signal assignments for the I/O channel connectors.

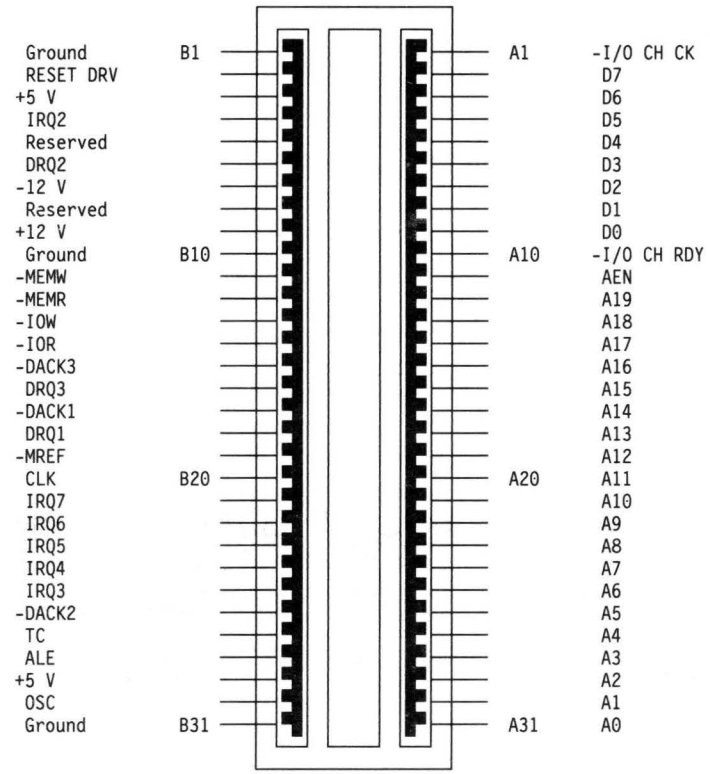


Figure 1-13. I/O Channel

Signal Description

The following is a description of the I/O channel signal lines. All lines are TTL-compatible. The (O), (I), or (I/O) notation refers to output, input, or input and output.

A0 – A19 (O): Address bits 0 to 19: These lines are used to address memory and I/O devices within the system. The 20 address lines allow access to 1M of address space. Only the lower 16 lines are used in I/O addressing, and all 16 should be decoded by I/O devices. A0 is the least significant and A19 is the most significant. These lines are generated by either the microprocessor or the DMA controller.

AEN (O): Address Enable: This line is used to de-gate the microprocessor and other devices from the I/O channel to allow DMA transfers to take place. When this line is active, the DMA controller has control of the address bus, data bus, and Read and Write command lines. When this line is inactive, the microprocessor has control. This line should be part of the adapter-select decode to prevent incorrect adapter selects during DMA operations.

ALE (O): Address Latch Enable: This line is provided by the bus controller and is used on the system board to latch valid addresses from the microprocessor. Addresses are valid at the falling edge of ALE and are latched onto the bus while ALE is inactive. This signal is forced active during DMA cycles.

CLK (O): System clock: This is the system clock signal with a frequency of 8 MHz and a 33% duty cycle.

D0 – D7 (I/O): Data bits 0 to 7: These lines provide data bus bits 0 to 7 for the microprocessor, memory, and I/O devices.

-DACK1 – -DACK3 (O): -DMA Acknowledge 1 to 3: These lines are used by the controller to acknowledge DMA requests. DACK0 is not available on the Type 7690's I/O channel.

DRQ1 – DRQ3 (I): DMA Request 1 to 3: These lines are asynchronous channel requests used by peripheral devices to gain DMA services. They are prioritized with DRQ1 being the highest and DRQ3 being the lowest. A request is generated by bringing a request line to an active level. A request line is held active until the corresponding acknowledge line goes active.

-I/O CH CK (I): -I/O Channel Check: This line generates an NMI. It is driven active to indicate an uncorrectable error and held active for at least two clock cycles.

I/O CH RDY (I): I/O Channel Ready: This line, normally active (ready), is pulled inactive (not ready) by a memory or I/O device to lengthen I/O or memory cycles. It allows slower devices to attach to the I/O channel with a minimum of difficulty. Any slow device using this line should drive it inactive immediately after detecting a valid address and a Read or Write command. For every clock cycle this line is inactive, one wait state is added. This line should not be held inactive longer than 17 clock cycles.

-IOR (O): -I/O Read: This command line instructs an I/O device to drive its data onto the data bus. This signal is driven by the microprocessor or the DMA controller.

-IOW (O): -I/O Write: This command line instructs an I/O device to read the data on the data bus. This signal is driven by the microprocessor or the DMA controller.

IRQ2 – IRQ7 (I): Interrupt requests 2 through 7: These lines are used to signal the microprocessor that an I/O device requires attention. They are prioritized with IRQ2 as the highest priority and IRQ7 as the lowest. When an interrupt is generated, the request line is held active until it is acknowledged by the microprocessor.

-MEMR (O): -Memory Read: This command line instructs memory to drive its data onto the data bus. This signal is driven by the microprocessor or the DMA controller.

-MEMW (O): -Memory Write: This command line instructs memory to store the data present on the data bus. This signal is driven by the microprocessor or the DMA controller.

-MREF (I/O): -Memory Refresh: This line indicates a refresh cycle.

OSC (O): Oscillator: This is a high-speed clock with a 70-ns period (14.31818 MHz). It has a 50% duty cycle.

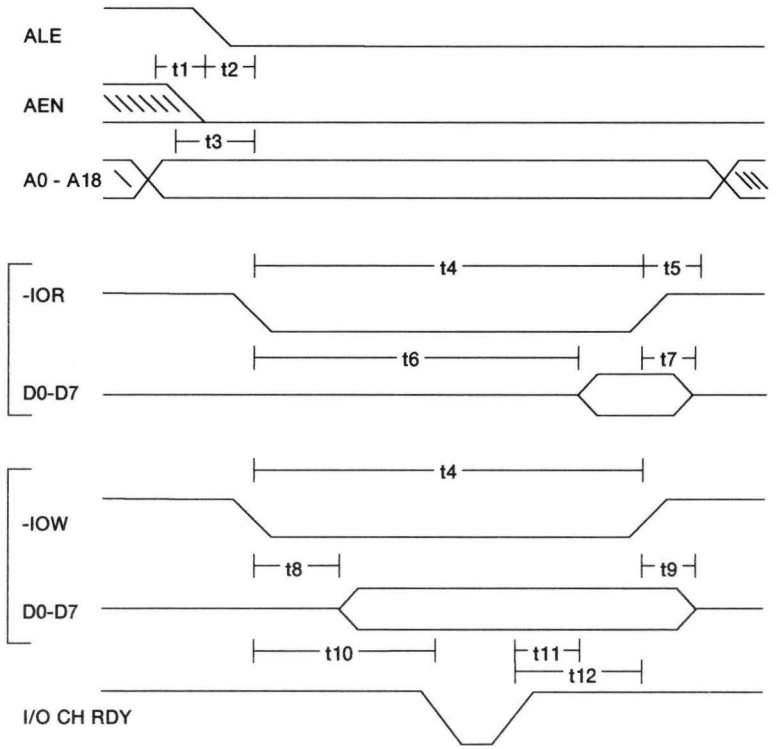
RESET DRV (O): Reset Drive: This line is used to reset or initialize system logic upon power-up or during a low line-voltage. This signal is synchronized to the falling edge of CLK.

TC (O): Terminal Count: This line provides a pulse when the terminal count for any DMA channel is reached.

Signal Timings

The following diagrams show the I/O signal timings for I/O and memory operations.

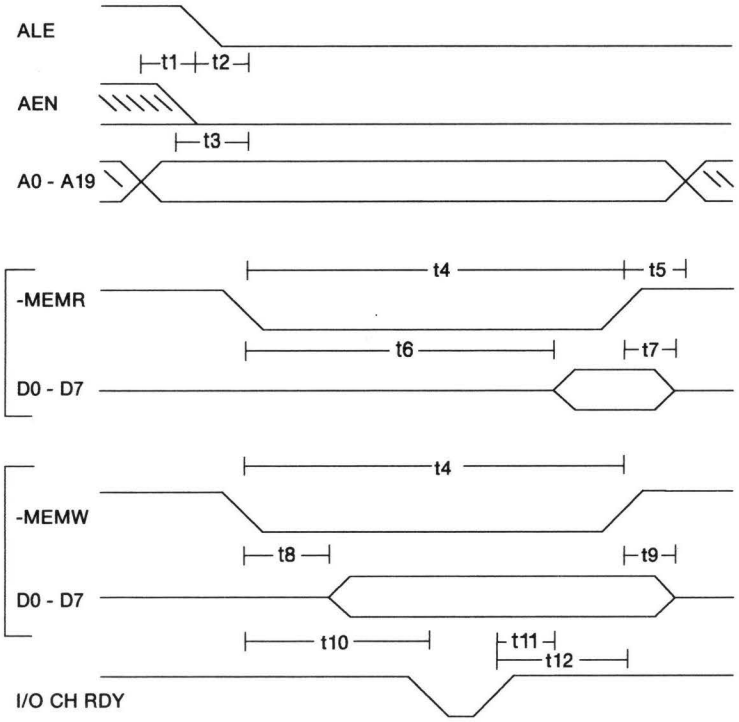
8-Bit I/O Bus Cycles



Symbol	Description	Min (ns)	Max (ns)
t1	Address valid to ALE inactive	20	
t2	ALE inactive to Command active	60	
t3	Command active from AEN inactive	95	
t4	Command pulse width	605	
t5	Address hold from Command inactive	45	
t6	Data valid from Read active		540
t7	Data hold from Read inactive	0	
t8	Data valid from Write active		120
t9	Data hold from Write inactive	25	
t10	I/O CH RDY inactive from Command active		325
t11	Read Data valid from I/O CH RDY active		0
t12	Command inactive from I/O CH RDY active	160	

Figure 1-14. 8-Bit I/O Timing

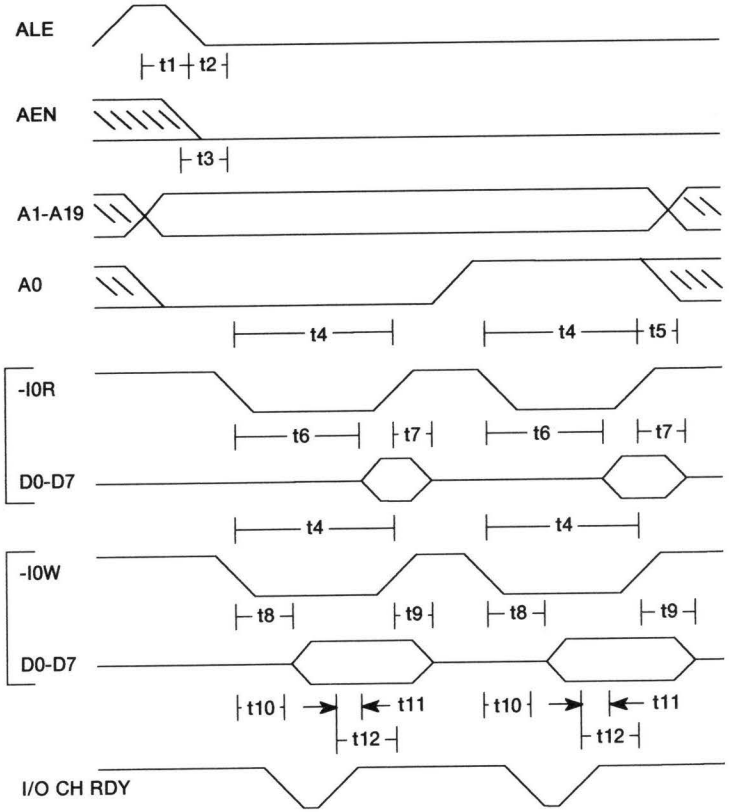
8-Bit Memory Bus Cycles



Symbol	Description	Min (ns)	Max (ns)
t1	Address valid to ALE inactive	20	
t2	ALE inactive to Command active	60	
t3	Command active from AEN inactive	95	
t4	Command pulse width	395	
t5	Address hold from Command inactive	45	
t6	Data valid from Read active		315
t7	Data hold from Read inactive	0	
t8	Data valid from Write active		120
t9	Data hold from Write inactive	25	
t10	I/O CH RDY inactive from Command active		115
t11	Read Data valid from I/O CH RDY active		0
t12	Command inactive from I/O CH RDY active	160	

Figure 1-15. 8-Bit Memory Timing

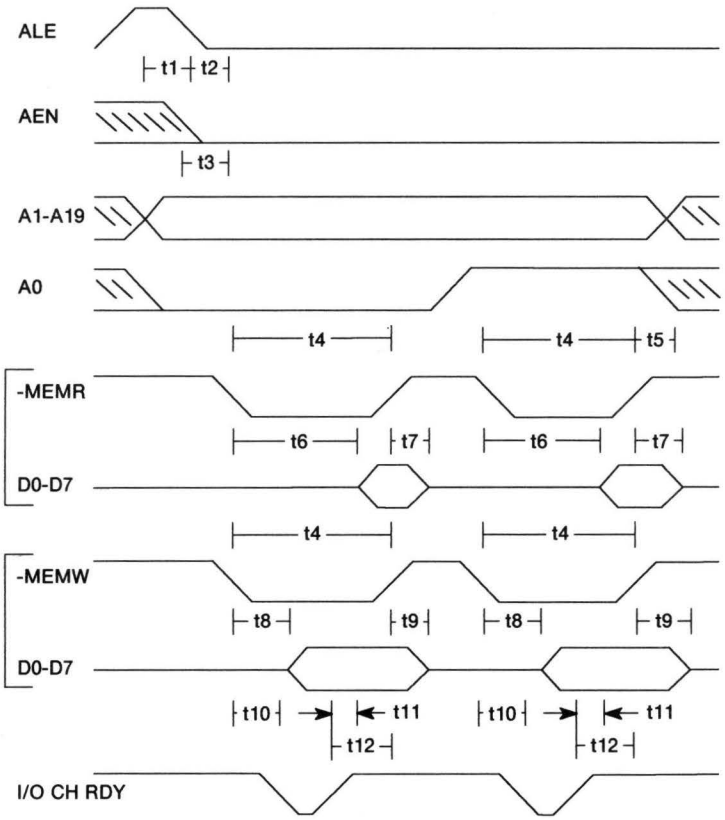
16-Bit I/O Bus Cycles



Symbol	Description	Min (ns)	Max (ns)
t1	Address valid to ALE inactive	20	
t2	ALE inactive to Command active	60	
t3	Command active from AEN inactive	95	
t4	Command pulse width	605	
t5	Address hold from Command inactive	45	
t6	Data valid from Read active		540
t7	Data hold from Read inactive	0	
t8	Data valid from Write active		120
t9	Data hold from Write inactive	25	
t10	I/O CH RDY inactive from Command active		325
t11	Read Data valid from I/O CH RDY active		0
t12	Command inactive from I/O CH RDY active	160	

Figure 1-16. 16-Bit I/O Timing

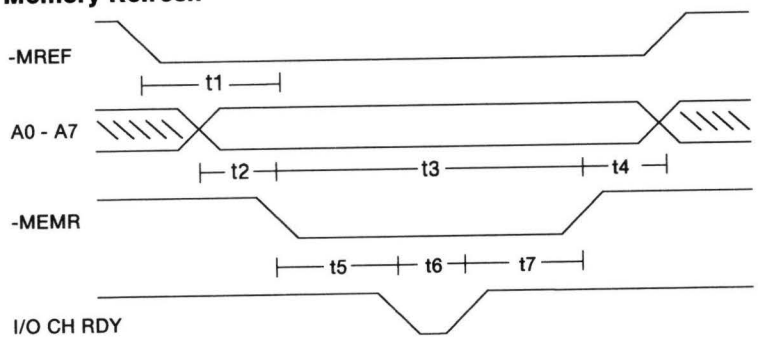
16-Bit Memory Bus Cycles



Symbol	Description	Min (ns)	Max (ns)
t1	Address valid to ALE inactive	20	
t2	ALE inactive to Command active	60	
t3	Command active from AEN inactive	95	
t4	Command pulse width	395	
t5	Address hold from Command inactive	45	
t6	Data valid from Read active		315
t7	Data hold from Read inactive	0	
t8	Data valid from Write active		120
t9	Data hold from Write inactive	25	
t10	I/O CH RDY inactive from Command active		115
t11	Read Data valid from I/O CH RDY active		0
t12	Command inactive from I/O CH RDY active	160	

Figure 1-17. 16-Bit Memory Timing

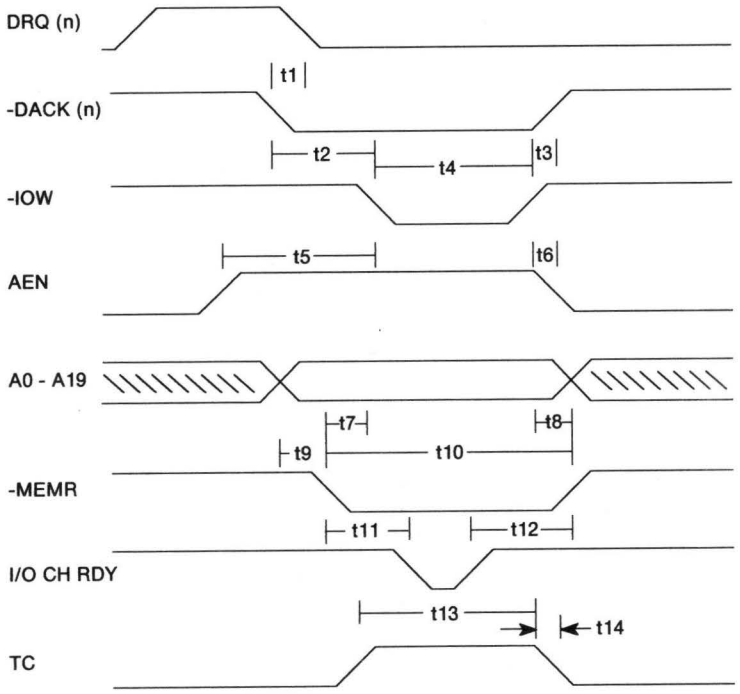
Memory Refresh



Symbol	Description	Min (ns)	Max (ns)
t1	-MREF active to -MEMR active	155	
t2	Address valid to -MEMR active	75	
t3	-MEMR pulse width	230	
t4	-MEMR inactive to -MREF inactive	10	
t5	-MEMR active to I/O CH RDY inactive		60
t6	I/O CH RDY pulse width		600
t7	-MEMR inactive from I/O CH RDY active	0	

Figure 1-18. Memory Refresh Timing

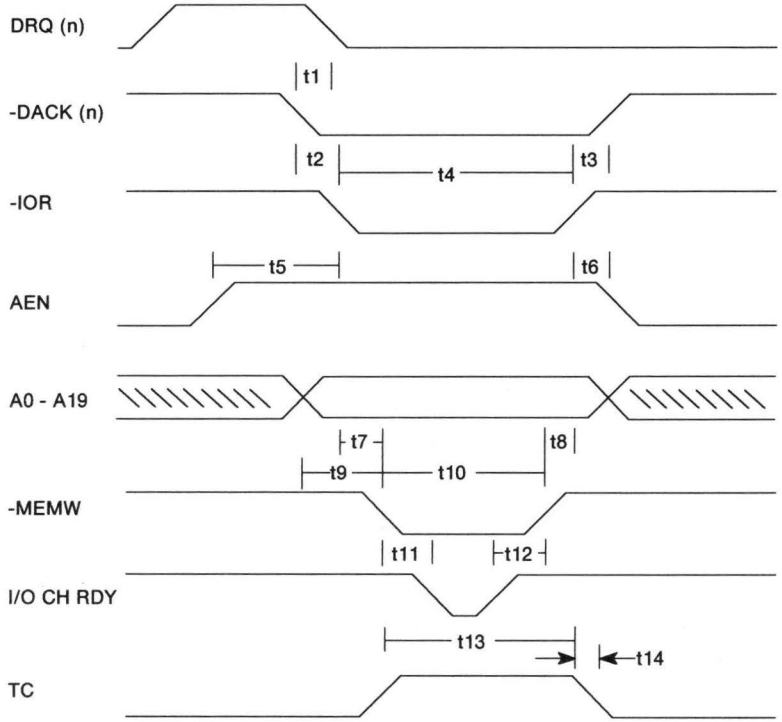
DMA Read



Symbol	Description	Min (ns)	Max (ns)
t1	-DACK active to DRQ inactive	0	
t2	-DACK active to -IOW active	200	
t3	-IOW inactive to -DACK inactive	0	
t4	-IOW pulse width	250	
t5	AEN active to -IOW active	500	
t6	-IOW inactive to AEN inactive	25	
t7	-IOW active from -MEMR active		360
t8	-IOW inactive to -MEMR inactive	0	
t9	Address valid to -MEMR active	0	
t10	-MEMR pulse width	470	
t11	-MEMR active to I/O CH RDY inactive		200
t12	-MEMR inactive from I/O CH RDY active	200	
t13	TC active setup to -IOW inactive	290	
t14	TC inactive from -IOW inactive	0	

Figure 1-19. DMA Read Timing

DMA Write



Symbol	Description	Min (ns)	Max (ns)
t1	-DACK active to DRQ inactive	0	
t2	-DACK active to -IOR active	0	
t3	-IOR inactive to -DACK inactive	0	
t4	-IOR pulse width	470	
t5	AEN active to -IOR active	300	
t6	-IOR inactive to AEN inactive	0	
t7	-MEMW active from -IOR active	55	
t8	-MEMW inactive to -IOR inactive	0	
t9	Address valid to -MEMW active	140	
t10	-MEMW pulse width	250	
t11	-MEMW active to I/O CH RDY inactive		30
t12	-MEMW inactive from I/O CH RDY active	200	
t13	TC active setup to -IOR inactive	290	
t14	TC inactive from -IOR inactive	0	

Figure 1-20. DMA Write Timing

Video Subsystem

The video subsystem resides on the system board and 7690 interface adapter. The output of the video circuitry on the system board (MCGA) is carried to the interface adapter (through the video interface cable) where it is processed by the LCD controller and displayed on the LCD.

The major components of the video subsystem are:

- System Board
  - Video memory controller gate array
  - Video formatter gate array
  - 64K bytes of multiport dynamic memory
  - 8K bytes static RAM character generator
- Interface Adapter
  - Custom LCD controller chip
  - 16K ROM
  - Two 32K-by-8-bit static RAM LCD image buffers

At the BIOS level (interrupt hex 10), the Type 7690 maintains compatibility with the IBM Color Graphics Adapter (CGA). The video modes are compatible with those modes supported by CGA with one addition: 640-by-480 graphics (mode 11). However, color is not supported by the LCD.

Throughout this section, you will see references to video functions provided by the MCGA circuitry on the system board. When appropriate, special notes are provided to discuss their implementation on the Type 7690.

Video Block Diagram

The following figure depicts the components of the video subsystem.

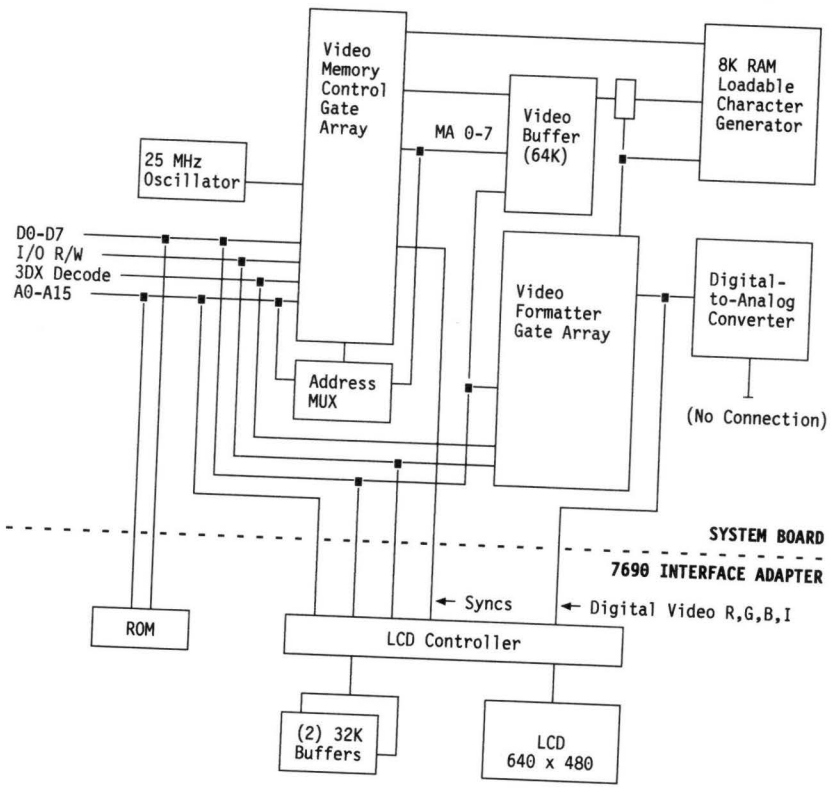


Figure 1-21. Video Subsystem Block Diagram

Display Support

The video subsystem supports a liquid crystal display (LCD) capable of 640-by-480 resolution. The LCD uses the digital video signals, bypassing the digital-to-analog converter. Since the Type 7690 uses a modified IBM Type 8525 system board, the video subsystem is similar in function to that of the PS/2 Model 25; however, color monitors cannot be connected.

At power on, the video POST on the system board senses *no monitor* even though the LCD is connected. The power-on self test contained in ROM on the interface adapter is responsible for proper initializa-

tion of the video subsystem, the LCD support circuitry, and the Display Combination Code (DCC).

MCGA Video Modes

The Type 7690 supports the standard MCGA video modes. However, their implementation has been modified for the LCD's lack of color capability.

Text Modes

In the text modes, the character box size is 8-by-16. The character font table is loaded into the character generator. All 16 scan lines are programmed into the character generator.

Graphics Modes

In the graphics modes, the character font table is used to create the character PELs. For most graphics modes, the character box is an 8-by-8 character box that is double-scanned to create an 8-by-16 character; however, all 16 scan lines of the 8-by-16 box are not programmable.

The 640-by-480 graphics mode is the exception. It uses an 8-by-16 character box and a separate font table. In this mode, 30 character rows are displayed.

Remember:

The following chart summarizes the MCGA video modes. Keep in mind that these modes are implemented differently in the Type 7690. Refer to "LCD Video Modes" on page 1-40 for these differences.

Video Modes	MCGA Characteristics
Mode 0,1 40 Column Alphanumeric	40 column by 25 rows 8-by-16 character box 320 by 400 16 of 256K colors or 16 of 64 shades of gray (monochrome) Display buffer B8000 2000 byte video buffer
Mode 2,3 80 Column Alphanumeric	80 column by 25 rows 8-by-16 character box 640 by 400 16 of 256K colors or 16 of 64 shades of gray (monochrome) Display buffer B8000 4000 byte video buffer
Mode 4,5 320 by 200 Graphics	8-by-8 character box Double-scanned 320 by 200 4 of 256K colors or 4 of 64 shades of gray (monochrome) Alternate palette select Display buffer B8000 16000 byte video buffer Two row scan address partitions
Mode 6 640 by 200 Graphics	8-by-8 character box Double-scanned 640 by 200 2 of 256K colors Display buffer B8000 16000 byte video buffer Two row scan address partitions
Mode 11 640 by 480 Graphics	8-by-16 character box 640 by 480 2 of 256K colors Display buffer A0000 38400 byte video buffer Linear addressing
Mode 13 320 by 200 Graphics	8-by-8 character box Double-scanned 320 by 200 256 of 256K colors Display buffer A0000 64000 byte video buffer Linear addressing

Figure 1-22. MCGA Video Mode Summary

LCD Video Modes

Mode	Type	Colors (MCGA)	Colors on LCD	Box Size on LCD	Resolution (MCGA)	Resolution on LCD
0,1	A/N	16/256K	2	8x16	320x400	640x400
2,3	A/N	16/256K	2	8x16	640x400	640x400
4,5	APA	4/256K	2	8x8	320x200	640x400
6	APA	2/256K	2	8x8	640x200	640x400 *
11	APA	2/256K	2	8x16	640x480	640x480
13	APA	256/256K	2	8x8	320x200	320x400 *

* Double scanned to use the full screen height.

Figure 1-23. LCD Video Modes

In the figure above, some modes do not fill the entire 480 pel height of the LCD. For those modes, the active video area will be centered on the screen, leaving a blank area at the top and bottom.

Video Modes Remapped for LCD

Because the LCD can only display two colors (black and white), the ROM on the interface adapter remaps the BIOS video modes to their black and white equivalents. This allows many existing color applications to run on the LCD in black and white mode.

Note: Refer to Section 7, "Programming Tips and Techniques" for programming considerations when developing color application programs.

When referring to sections in this manual pertaining to video modes, keep in mind that the modes are remapped as follows:

Mode Requested	Mode Enabled
0	0
1	0
2	2
3	2
4	4
5	5
6	6
11	11
13	13

Figure 1-24. Video Mode Map

Note: Even though video mode 13 is implemented, its use is not recommended due to the color limitations of the LCD.

Display Formats

In alphanumeric (text) modes 0 through 3, two bytes define each character on the display screen. The even byte accesses the character generator to create the PEL data. The odd byte defines the color of the PELs. Sixteen colors are available for foreground, and eight colors are available for background when blink is enabled (default). Blink is controlled in the CGA Mode Control register, hex 3D8.

The format of the two bytes is shown in the following:

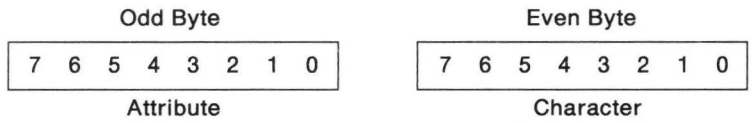


Figure 1-25. Alphanumeric Format

Remember:

On the Type 7690's LCD, odd-numbered colors appear white, while even-numbered colors appear black.

The following are the bit definitions of the attribute (odd) byte. Bit 7 selects a blinking character, or if blinking is disabled, selects palette addresses above hex 07 for the background color.

Bits	Function
7 to 4	Background Color Palette Address
3 to 0	Foreground Color Palette Address

Figure 1-26. Attribute Byte

In modes 4 and 5, the bit pair C1 and C0 selects one of four colors for each PEL.

Bit	PEL Definition	
7,6	C1,C0	First PEL
5,4	C1,C0	
3,2	C1,C0	
1,0	C1,C0	Last PEL

Figure 1-27. Modes 4 and 5

There are two color sets: color set 0 and color set 1. For information about the colors selected when using a color monitor, see "CGA Border Control Register, 3D9," later in this section under "Video Formatter Registers."

Remember:

For the Type 7690, bit C1 is the only significant bit; when set (1), the corresponding pel will be white. When cleared (0), the corresponding pel will be black.

In modes 6 and 11, one bit defines each PEL, with the most significant bit defining the first PEL. The foreground color maps to the color in the CGA Border Control register if the B&W bit in the CGA Mode Control Register is 0. If the B&W bit is 1, the foreground color maps to palette address hex 07. The background color always maps to address hex 00.

Remember:

The palette is not used in the Type 7690.

Bit	PEL Definition	
7	C0	First PEL
6	C0	
5	C0	
4	C0	
3	C0	
2	C0	
1	C0	
0	C0	Last PEL

Figure 1-28. Modes 6 and 11

Video Storage Organization

The following is the memory mapping for text modes 0 through 3.

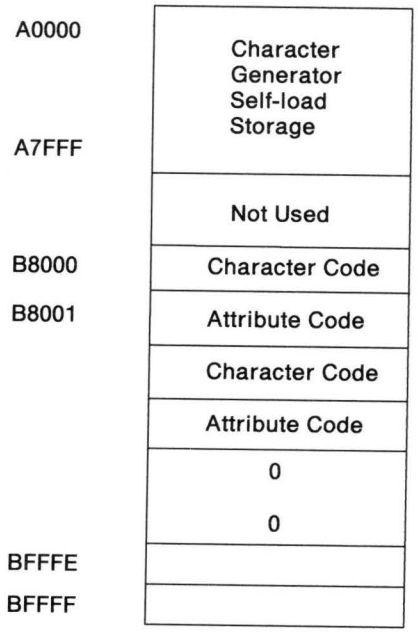


Figure 1-29. Text Modes 0 through 3

The following is the memory mapping for graphics modes 4 through 6. In modes 4 and 5, each byte defines four PELs. In mode 6, each byte defines eight PELs.

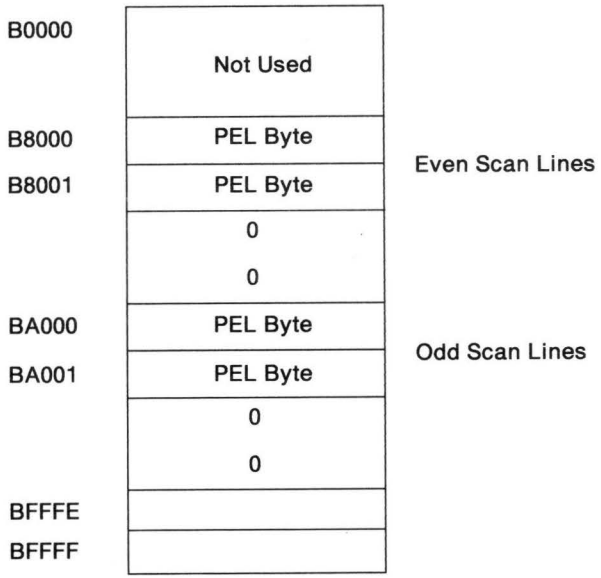


Figure 1-30. Graphics Modes 4 through 6

The following is the memory mapping for graphics modes 11 and 13. In mode 11, each byte defines eight PELs; in mode 13, each byte defines one PEL.

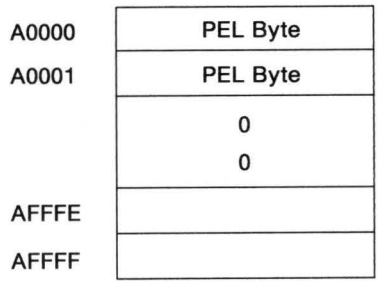


Figure 1-31. Graphics Modes 11 and 13

MCGA Video Registers

The video memory controller gate array responds to I/O addresses 3D4 and 3D5. The video formatter gate array responds to I/O addresses 3D8 through 3DF.

The color palette is programmed through the video formatter at addresses 3C6 through 3C9. All registers are readable.

The following pages describe the memory controller registers, the video formatter registers, the color palette registers, and the character generator. Sample programs of a font load and palette load are also included.

Video Memory Controller Registers

The video memory controller contains an index register and 22 data registers. Two I/O commands are required to write to one data register: writing the desired index value to address hex 3D4, and then writing the data to address hex 3D5.

Memory Controller Index Register, Hex 3D4: This register is read/write, and points to the specific data register addressed through hex 3D5.

Bit	Function
7	Reserved
6	Reserved
5	Index5
4	Index4
3	Index3
2	Index2
1	Index1
0	Index0

Figure 1-32. Video Memory Controller Index Register

The following is a list of the 22 data registers and their functions.

Index (Hex)	Register Description
00	Horizontal Total
01	Horizontal Characters Displayed
02	Start Horizontal Sync
03	Sync Pulse Width
04	Vertical Total
05	Vertical Total Adjust
06	Vertical Characters Displayed
07	Start Vertical Sync
08	Reserved
09	Scan Lines per Character
0A	Cursor Start
0B	Cursor End
0C	Start of Screen High
0D	Start of Screen Low
0E	Cursor Position High
0F	Cursor Position Low
10	Mode Control
11	Interrupt Control
12	Character Generator Interface and Sync Polarity, or Display Sense
13	Character Font Pointer
14	Number of Characters to Load
20	Reserved

Horizontal Total Register, Index 00: This register contains the total number of characters in the horizontal scan interval. The number consists of both displayed and nondisplayed characters. This register determines the frequency of the 'horizontal sync' signal.

Horizontal Characters Displayed Register, Index 01: This register determines the total number of characters to be displayed during the horizontal video scan interval. This register is loaded with a value of hex 27. The hardware calculates the correct value based on the mode selected.

Start Horizontal Sync Register, Index 02: This register specifies the character position count at which the 'horizontal sync' signal becomes active.

Sync Pulse Width Register, Index 03: This register specifies the pulse widths of the horizontal and vertical synchronization signals. The horizontal pulse width is programmed in units of character clocks. The vertical pulse width is programmed in units of the horizontal synchronization period. This register is programmed to match the display specifications.

Bit	Function
7	Width VSync3
6	VSync2
5	VSync1
4	VSync0
3	Width HSync3
2	HSync2
1	HSync1
0	HSync0

Figure 1-33. Sync Pulse Width Register

Vertical Total Register, Index 04: This register contains the 8 least significant bits for the total number of scan lines in the vertical scan interval. The most significant bit is the inversion of bit 6 of the Mode Control register. The total number consists of both the displayed and nondisplayed scan lines. This register and the Vertical Total Adjust register determine the frequency of the 'vertical sync' signal.

Vertical Total Adjust Register, Index 05: This register is used to adjust the total number of horizontal scan lines in the vertical scanning interval. It allows for an odd number of horizontal lines (525 for 60 Hz). The minimum value for this register is hex 02.

Bit	Function
7	Reserved
6	Reserved
5	VAdjust5
4	VAdjust4
3	VAdjust3
2	VAdjust2
1	VAdjust1
0	VAdjust0

Figure 1-34. Vertical Total Adjust Register

Vertical Characters Displayed Register, Index 06: This register contains the 8 least significant bits for the number of scan lines displayed in the vertical scan interval. The most significant bit is the inversion of bit 6 of the Mode Control register.

Start Vertical Sync Register, Index 07: This register contains the 8 least significant bits for the vertical scan line count. It determines when the 'vertical sync' signal becomes active. The most significant bit is the inversion of bit 6 of the Mode Control register.

Scan Lines per Character Register, Index 09: This register determines the number of horizontal scan lines in a character row. In text modes, the value is hex 07. In graphics modes 4 through 6, the value is hex 01, and in modes 11 and 13, the value is hex 00. The hardware calculates the proper value based on the mode selected.

Bit	Function
7	Reserved
6	Reserved
5	Reserved
4	Reserved
3	Row Size3
2	Row Size2
1	Row Size1

Figure 1-35 (Part 1 of 2). Scan Lines per Character Register

Bit	Function
0	Row Size0

Figure 1-35 (Part 2 of 2). Scan Lines per Character Register

Cursor Start Register, Index 0A: Bits 3 through 0 in this register determine the horizontal scan line count at which the cursor output becomes active. The value in this register should be lower than the value in the Cursor End register. The minimum is 0. The hardware will double-scan the cursor to produce the proper cursor display for a 16-scan-line character box.

When bit 5 is 1, the cursor is not displayed.

Bit	Function
7	Reserved
6	Reserved
5	Blank Cursor
4	Reserved
3	Cursor Start3
2	Cursor Start2
1	Cursor Start1
0	Cursor Start0

Figure 1-36. Cursor Start Register

Cursor End Register, Index 0B: This register determines the horizontal scan line count when the cursor output becomes inactive. The value should be greater than the value in the Cursor Start register. The maximum is 7.

Bit	Function
7	Reserved
6	Reserved
5	Reserved
4	Reserved
3	Cursor End3
2	Cursor End2
1	Cursor End1
0	Cursor End0

Figure 1-37. Cursor End Register

Start of Screen High Register, Index 0C: This register contains the 8 most significant bits for the starting memory address of the video display buffer. Sixteen address bits determine the starting address. This register is initialized to a value of hex 00.

Start of Screen Low Register, Index 0D: This register, together with the Start of Screen High register, gives the starting address of the display buffer. For all modes, this register is initialized to a value of hex 00.

Cursor Position High Register, Index 0E: This register contains the four most significant bits for the cursor location.

Bit	Function
7	Reserved
6	Reserved
5	Reserved
4	Reserved
3	Cursor PositionB
2	Cursor PositionA
1	Cursor Position9
0	Cursor Position8

Figure 1-38. Cursor Position High Register

Cursor Position Low Register, Index 0F: This register contains the eight least significant bits for the location of the cursor. A value of hex 00 in both of these registers will locate the cursor in the upper left corner. The cursor is not supported in any graphics mode.

Mode Control Register, Index 10: Writing to this register selects the type of display and clock times, and selects some of the graphics modes.

Bit	Function
7	Inhibit Write
6	Reserved = 0
5	Reserved
4	Clock = 1
3	Compatibility
2	Reserved
1	Mode 11
0	256 Color

Figure 1-39. Mode Control, Write

Write

Bit 7 When set to 1, the Inhibit Write bit prevents any writes to the horizontal and vertical registers. After a mode set, BIOS sets this bit to 1 to prevent applications designed for other color graphics adapters from altering those registers.

Bit 6 The inverse of this bit is used as the ninth bit of the vertical compare circuits and must be set to 0.

Bit 5 Reserved.

Bit 4 This bit selects the dot clock and must be set to 1.

Bit 3 When set to 1, this bit allows the circuitry to calculate the correct horizontal register values for the 80-by-25 text modes. This bit should be set to 1 for all modes.

Bit 2 Reserved.

Bit 1 When set to 1, this bit selects mode 11.

Bit 0 When set to 1, this bit selects mode 13. Bit 2 in the Extended Mode Control register must also be set.

During certain operations, the circuitry calculates some of the internal signals and returns the values to the Mode Control register.

Bit	Function
7	80x25
6	Reserved
5	Clock Select
4	Clock
3	Alpha Mode
2	Double-Scan
1	Mode 11
0	Mode 13

Figure 1-40. Mode Control, Read

Read

Bit 7 This bit indicates the state of bit 0 in the CGA Mode Control register. When set to 1, this bit indicates that 80-by-25 mode is selected.

Bit 6 Reserved.

Bit 5 When this bit is 1, it indicates that the clock is not divided by 2, and the resolution is 640 PELs wide. When it is 0, the resolution is 320.

Bit 4 When this bit is 1, it indicates that the dot clock is 25.175 MHz.

Bit 3 When set to 1, this bit indicates that the mode is a text mode.

Bit 2 When set to 1, this bit indicates that the scan lines are double-scanned.

Bit 1 When set to 1, this bit indicates that mode 11 is selected.

Bit 0 When set to 1, this bit indicates that mode 13 is selected.

Interrupt Control Register, Index 11: This register controls IRQ2 output to the interrupt controller. It also shows the status of the interrupt. The output drivers are tri-stated (bit 7) to allow a Read of the Display Sense register.

Bit	Function
7	Tri-State Output
6	IRQ2 Status
5	-Enable IRQ2
4	-Clear IRQ2 Latch
3	Reserved
2	Reserved
1	Reserved
0	Reserved

Figure 1-41. Interrupt Control Register

Bit 7 When set to 1, this bit disables (tri-states) the output drivers and selects the Display Sense register to be read at index 12 instead of the Character Generator Interface and Sync Polarity register.

Bit 6 When set to 1, this bit indicates that the memory controller is causing an interrupt. This bit is read-only.

Bit 5 When cleared to 0, this bit enables the interrupt.

Bit 4 When cleared to 0, this bit holds the interrupt latch clear.

Bits 3-0 These bits are reserved and should be 0.

Character Generator Interface and Sync Polarity Register, Index 12: This register controls the character font tables and the horizontal and vertical synchronization signals, HSYNC and VSYNC. To read this register, bit 7 of the Interrupt Control register must be 0.

Bit	Function
7	Load Character Generator
6	Load Full Character Set
5	Swap Active Font
4	Enable 512 Characters
3	Reserved = 0
2	Enable Sync Outputs
1	VSYNC Polarity
0	HSYNC Polarity

Figure 1-42. Character Generator Interface and Sync Polarity Register

Bit 7 When written as a 1, this bit loads the character generator. When read as a 0, the bit indicates that the load has finished. To start the load, this bit is first cleared and then set to 1.

Bit 6 When set to 1, this bit causes the character generator to load the display memory during normal display time. When clear, the display memory is loaded only during the vertical blanking interval.

Bit 5 This bit selects the font page that is used as font table or that the character generator loads. When set to 1, font page 1 is selected; when cleared to 0, font page 0 is selected.

Bit 4 When this bit is set to 1, 512 character codes are displayable in the text modes. Bit 3 of the attribute byte then determines the font page when displaying the character. When this bit is set to 1, only eight foreground colors are supported. When this bit is cleared to 0, only 256 character codes are displayed, and bit 5 of this register determines the active font.

Bit 3 Reserved = 0.

Bit 2 When set to 1, this bit enables HSYNC and VSYNC outputs to the display.

Bit 1 When set to 1, this bit causes VSYNC to be positive polarity.

Bit 0 When set to 1, this bit causes HSYNC to be positive polarity.

Display Sense Register, Index 12: This register contains the sensed levels of the monitor sense 1 and 0 signals at pins 13 and 14 of the display connector. This information is used by BIOS to properly initialize all video registers to match the display. To read this register, bit 7 of the Interrupt Control register is set to 1.

These levels are used to determine the type of display attached as shown in the following. The bit is set when the polarity is positive.

Sense 1 Bit 1	Sense 0 Bit 0	Type of Display Attached
0	0	Reserved
0	1	Analog Monochrome Display
1	0	Analog Color Display
1	1	LCD Attached

Figure 1-43. Monitor Sense Bits

Character Font Pointer Register, Index 13: This register contains a pointer to the character font table. The only valid pointer values are hex 00, 10, 20, or 30. The pointer value doubled and the hex value A0000 make up the segment for the font table. The character value doubled is the offset into the table. See "RAM-Loadable Fonts," later in this section.

Number of Characters to Load Register, Index 14: This register determines the number of characters to load into the RAM-loadable character generator during one vertical retrace interval. This register is used only in the text modes.

Video Formatter Registers

The video formatter registers at I/O addresses hex 3D8 and 3D9 duplicate the functions of the 6845 registers in the color graphics adapter. Registers are added at addresses hex 3DD through 3DF for Type 7690 initialization requirements. The video formatter registers at addresses hex 3C6 through 3C9 control the color palette.

Register	Description
3D8	CGA Mode Control
3D9	CGA Border Control
3DA	CGA Status
3DB	Reserved
3DC	Reserved
3DD	Extended Mode Control
3DE	Reserved
3DF	Reserved
3C6	PEL Mask
3C7	Palette Read Address
3C8	Color Palette Address
3C9	Color Palette Data

CGA Mode Control Register, 3D8: This register contains the mode control information for color/graphics compatible functions.

Bit	Function
7	Reserved
6	Reserved
5	Enable Blink
4	640-by-200 Mono
3	Enable Video
2	B&W
1	Graphics
0	80-by-25 Alpha

Figure 1-44. CGA Mode Register

Bits 7,6 Reserved.
Bit 5 When set to 1, this bit selects the blink option for text modes. When cleared to 0, 16 background colors are available in the text modes.
Bit 4 When set to 1, this bit selects mode 6, 640-by-200 double-scanned graphics.
Bit 3 When set to 1, this bit enables display image.
Bit 2 When this bit is 1, palette addresses hex 00 and 07 are the two colors used in modes 6 and 11. When the bit is 0, address hex 00 and the address specified in the CGA Border Control register are the two colors used.
Bit 1 When set to 1, this bit selects modes 4 and 5, 320-by-200 double-scanned graphics.
Bit 0 When set to 1, this bit selects the 80-by-25 text mode.

CGA Border Control Register, 3D9: This register contains the border color information and selects the alternate color palette for modes 4 and 5. Although analog displays do not have borders, the border color information selects the alternate foreground color for modes 6 and 11, and the background color for modes 4 and 5.

Bit	Function
7	Reserved
6	Reserved
5	320-by-200 Palette Select

Figure 1-45 (Part 1 of 2). CGA Border Control Register

Bit	Function
4	Alternate Intensity
3 to 0	Border Color

Figure 1-45 (Part 2 of 2). CGA Border Control Register

Bits 7,6 Reserved
Bit 5 When set to 1, this bit selects color set 1 for modes 4 and 5.
Bit 4 When set to 1 (default), this bit selects an intensified color set for modes 4 and 5.
Bits 3-0 These bits select the palette address for the border color information used by modes 4, 5, 6, and 11.

The following figure shows the effects of this register and the bit pair C1,C0 and how the two color sets map into the color palette.

BCR Bit 4	C1	C0	BCR Bit 5	Palette Address
X	0	0	X	**Background Color**
0	0	1	0	02 Color Set 0
0	1	0	0	04 Color Set 0
0	1	1	0	06 Color Set 0
0	0	1	1	03 Color Set 1
0	1	0	1	05 Color Set 1
0	1	1	1	07 Color Set 1
				Intensified Colors
1	0	1	0	0A Color Set 0
1	1	0	0	0C Color Set 0
1	1	1	0	0E Color Set 0
1	0	1	1	0B Color Set 1
1	1	0	1	0D Color Set 1
1	1	1	1	0F Color Set 1

Figure 1-46. Modes 4 and 5 Color Selection

CGA Status Register, 3DA: This register is read-only and contains the status information for the color graphics adapter.

Bit	Function
7	Reserved
6	Reserved
5	Reserved
4	Reserved
3	Vertical Sync
2	Reserved
1	Reserved
0	-Display Enable

Figure 1-47. Status Register

Extended Mode Control Register, 3DD: This register controls the selection of the type of display and the advanced color support. When cleared to 0, bit 7 indicates that a readable DAC is installed; when set, it indicates that the DAC is not a readable type. Bit 2 must be set to 1 to select mode 13.

Note: The LCD uses digital video signals before the DAC.

Bit	Function
7	-Readable DAC Installed
6	Reserved
5	Reserved
4	Reserved
3	Reserved
2	256 Colors
1	Reserved
0	Reserved = 0

Figure 1-48. Extended Mode Control Register

Color Palette Registers

Three registers are used to access the color palette: a mask register, a read address register, and a write address register.

The color palette has 256 18-bit data registers and an 8-bit address register. Each data register is divided into three 6-bit data areas, one for each color. To load each data register takes three outputs in the sequence of red, green, blue.

When accessing the palette, the interrupts should be disabled to prevent the sequence from being interrupted. The palette supports both a single-register write operation and a burst load operation.

To maintain software compatibility, programmers should use the BIOS interface when loading the color palette. BIOS supports two calls for setting and two calls for reading the color registers. The calls are through interrupt 10H with (AH) = hex 10. The value in the AL register determines the specific operation:

10 - Set individual color register
12 - Set block of color registers
15 - Read individual color register
17 - Read block of color registers

Remember:

Refer to Section 7, "Programming Tips and Techniques" when developing color applications that will run on the Type 7690.

Single Register Load: The address for the specific color register (0 - 255) is loaded into the BX register. The DH, CH, and CL registers contain the red, green, and blue values, respectively. In the following example using the BIOS interface, the yellow color value is loaded into the palette address normally assigned to white. If the Set Mode call has been initialized to restore the color palette to its default state, the mode must be set before changing the color palette.

```
;-----Set up the video mode

MOV     AX,0004H      ; Set mode to mode 4
INT     10H           ; Video BIOS interrupt

;-----Read color 14 to get the red, green, and blue values for yellow

MOV     AX,1015H      ; Read individual color register
MOV     BX,0EH        ; Read color register 0EH
INT     10H           ; Video BIOS interrupt
                      ; Return with DH = red value
                      ;             CH = green value
                      ;             CL = blue value

;-----Set color 15 to the red, green, and blue values of yellow

MOV     AX,1010H      ; Set individual color register
MOV     BX,0FH        ; Set color register 0FH
INT     10H           ; Video BIOS interrupt
```

Burst Load: This second call supports setting a block of color registers. Using this call, 1 to 256 color values can be set or read with a single BIOS call. The BX register contains the address for the first register to be set, and CX contains the number of registers. ES:DX point to a table of color values, where each table entry contains the red, green, and blue values for a color. The following example sets the first 16 colors in the color palette.


```
;-----Set colors 0 thru 15 with a set block of color registers call

CODE    SEGMENT 'CODE'
ASSUME  CS:CODE, ES:NOTHING, DS:NOTHING

SET_BLK_EX      PROC    FAR

PUSH    DS
XOR     AX,AX
PUSH    AX                      ; Return address for DOS

PUSH    CS
POP     ES                      ; Establish ES addressing for table
MOV     AX,1012H                ; Set block of color register call
MOV     BX,0                    ; Start with color 0
MOV     CX,16                   ; Set 16 color registers
MOV     DX,OFFSET CLR_TABLE     ; ES:DX point to color table
INT     10H                     ; Make the video BIOS interrupt
RET

SET_BLK_EX  ENDP

CLR_TABLE       LABEL   BYTE

DB      00H,00H,00H         ; Black         00
DB      00H,00H,2AH         ; Blue          01
DB      00H,2AH,00H         ; Green         02
DB      00H,2AH,2AH         ; Cyan          03
DB      2AH,00H,00H         ; Red           04
DB      2AH,00H,2AH         ; Magenta       05
DB      2AH,15H,00H         ; Brown         06
DB      2AH,2AH,2AH         ; White         07
DB      15H,15H,15H         ; Gray          08
DB      15H,15H,3FH         ; Lt blue       09
DB      15H,3FH,15H         ; Lt green      0A
DB      15H,3FH,3FH         ; Lt cyan       0B
DB      3FH,15H,15H         ; Lt red        0C
DB      3FH,15H,3FH         ; Lt magenta    0D
DB      3FH,3FH,15H         ; Lt yellow     0E
DB      3FH,3FH,3FH         ; Bright White  0F

CODE    ENDS
        END
```

PEL Mask Register, 3C6: This register is initialized to a value that does not affect the color selection, hex FF.

Remember:

This value should not be changed because mask operations are not supported on the Type 7690.

Palette Read Address Register, 3C7: This register contains the pointer to one of 256 palette data registers and is used when reading the color palette.

Reading this port returns the last command cycle to the palette. The description of bits 1 and 0 is in the following table. All other bits during a read of this port are reserved.

Bit 1	Bit 0	Last Palette Command
0	0	Write Palette Cycle
0	1	Reserved
1	0	Reserved
1	1	Read Palette Cycle

Figure 1-49. Last Palette Command

Color Palette Address Register, 3C8: This register contains the pointer to one of 256 palette data registers and is used during a palette load.

Color Palette Data Register, 3C9: This register contains a 6-bit value that yields one of 64 color levels. To write a color, the address is loaded into the Color Palette Address register. Three writes to this register are needed for each palette address: the first is the red color information, the second is the green, and the third is the blue.

To read a color, the address value is written to the Palette Read Address register, followed by three reads of this register. The first returns the red color information, the second returns the green, and the third returns the blue.

Bit	Function
7	Not Used
6	Not Used
5	PD 5
4	PD 4
3	PD 3
2	PD 2
1	PD 1
0	PD 0

Figure 1-50. Color Palette Data Register

MCGA to LCD Conversion

There are several special requirements in order to convert the MCGA data into LCD data. The following is a summary of some of the considerations.

The LCD panel requires that two halves of the display be drawn at the same time. Because the MCGA data is serial (the two halves come out at separate times), half of the data must be stored and later retrieved when the remainder of the image is available. This split frame control requires two 32K x 8 RAMs to completely store the image during this process.

LCD Controller Registers

The LCD controller is accessible through two I/O addresses. The registers are treated very much like the index and data registers of the 6845 controller. Specifically, one I/O address is used to write an index value, the other I/O address is used to read/write data to the LCD controller register specified by the Index Register value. A summary of the addresses follows:

I/O Address	Function
F304h	Specifies the LCD Controller Index Register value. This is a read/write port.
F305h	Specifies the LCD Controller Data Register value. This is a read/write port.

Figure 1-51. LCD Controller I/O Addresses

LCD Controller Address Space

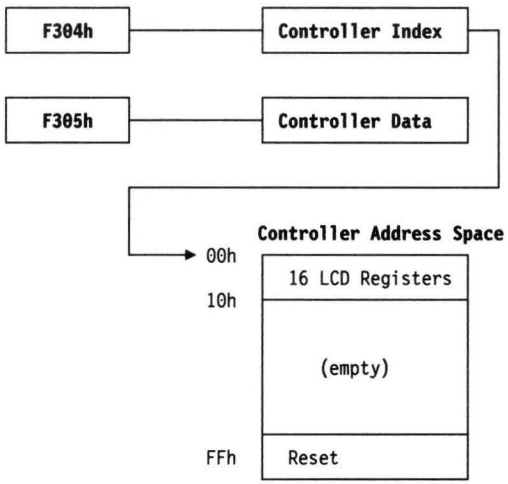


Figure 1-52. LCD Controller Address Space

LCD Index Register Select (F304)

The following chart details bit assignments used to select LCD registers.

Bit	I/O	Signal Name
7	I/O	Register Select
6	I/O	Register Select
5	I/O	Register Select
4	I/O	Register Select
3	I/O	Register Select
2	I/O	Register Select
1	I/O	Register Select
0	I/O	Register Select

Reg	Name
00	Horizontal Start Address
01	Vertical Start Address
02	Display Modes
03-04	Reserved
05	Ram Row Address
06	Upper RAM Data
07	Lower RAM Data
08	Reserved
09	Reserved
0A	Reserved
0B	Reserved
0C	Reserved
0D	Power Controls
0E	Mode Controls
0F	LP Period
10-FE	Reserved
FF	Master Reset

Figure 1-53. LCD Index Register Select (F304)

LCD Controller Register Assignments (F305)

The following charts detail the registers in the LCD Controller.

Normal Mode

Index	Bit	Function	R/W
00h	4-0	Start Address Pointer H	R/W
	7-5	(reserved)	
01h	7-0	Start Address Pointer V	R/W
02h	0	+Reverse video	R/W
	1	(reserved)	R/W
	2	(reserved)	R/W
	3	(reserved)	R/W
	4	(reserved)	R/W
	5	(reserved)	R/W
	6	(reserved)	R/W
	7	320(H) +Mode45 / +Mode13	R/W
03h	3-0	(reserved)	R/W
	7-4	(reserved)	

Figure 1-54. LCD Controller Register Assignment (Normal Mode)

Base Mode

Index	Bit	Function	R/W
0Dh	0	-YDIS (+LCD enable)	R/W
	1	+BLOFF (-LCD backlight enable)	R/W
	7-2	(reserved)	
0Eh	0	+Normal / -Test Mode	R/W
	1	+Rd nml / -Rd tst Clock mode	R/W
	2	(reserved)	
	3	+Color(P2) emulation reverse	R/W
	4	+320 / -640 (H) mode	R/W
	5	(reserved)	R/W
	6	(reserved)	R/W
	7	+Border Reverse	R/W
0Fh	7-0	LP period	R/W

Figure 1-55 (Part 1 of 2). LCD Controller Register Assignment (Base Mode)

Index	Bit	Function	R/W
FFh	-	Master LCDA Reset	W

Figure 1-55 (Part 2 of 2). LCD Controller Register Assignment (Base Mode)

Note: All registers (except 00, 01, 0C, and FF) are set to zero by a System Reset or Master Reset (Index FFh).

LCD Image Enhancement (Normal Mode)

Due to differences between LCD and CRT technology, certain functions have been added to improve LCD compatibility.

- Start Address Pointers (H & V)
- Reverse Video

Start Address Pointers (H & V)

- (Index=00, Bit=5-0) Start Address Pointer H
- (Index=01, Bit=7-0) Start Address Pointer V

Start Address Pointer H defines the top border of the LCD while Start Address Pointer V defines the left border on the LCD.

The LCD controller must match these pointers so that proper image alignment is obtained. Horizontally, the border delay is counted in pels (dot clocks) and the maximum value is +/- 8 pels (1 character). Vertically, the border delay is counted in HSYNCs (rows) and the maximum value is +/- 128 lines.

Reverse Video

- (Index=02, Bit=0) +Reverse Video

When the Reverse Video is enabled, it reverses video on LCD. The reverse area is only the screen size that each mode uses.

- (Index=0E, Bit=7) +Border Reverse (Base Mode).

When the Border Reverse is enabled, top porch and bottom porch are reversed. By combining +Reverse Video & +Border Reverse function, either white porch or black porch can be obtained.

LCD Special Controls (Base Mode)

Two controls for LCD power are available. The first controls the -22.3 Volt power (YDIS) to the LCD driver electronics (Bit 0, Index 0Dh). When this bit is one, the LCD display voltage is enabled. Setting this bit to zero conserves power when the LCD is not being used.

The second power control operates the back light (BLOFF) for the LCD (Bit 1, Index 0Dh). Setting this bit to zero enables the back light. Setting this bit to one conserves power and saves the limited life of the back light when the LCD is not being used. In order to improve the life of the back light, it is recommended that it be turned off after several minutes of not using the computer (no typing activity).

In order to achieve the lowest power consumption during non-use, both the LCD power and back light power should be turned off. In addition, the LCD controller and associated RAM can draw much less power when not operating. The easiest way to achieve this is to also turn on the LCD controller's Test mode (Bit 0 and Bit 1, Index 0Eh).

LCD power controls should be enabled and disabled through BIOS function calls. See *AL = BAH* under "Interrupt 10H - Video" on page 6-12.

Video Initialization Tables

The following figures show the video register values used by BIOS for the various display modes.

Index Pointer	Data Register Description	Modes 0,1	2,3	4,5	6	11	13
00	Horz. Total	30	30	30	30	30	30
01	Horz. Displayed	27	27	27	27	27	27
02	Start Horz. Sync	2A	2A	2A	2A	2A	2A
03	Sync Pulse Width	26	26	26	26	26	26
04	Vert. Total	B0	B0	B0	B0	FF	B0
05	Vert. Adjust	0D	0D	0D	0D	0A	0D
06	Vert. Displayed	8F	8F	8F	8F	DF	8F
07	Start Vert. Sync	9B	9B	9B	9B	E9	9B
08	Reserved	XX	XX	XX	XX	XX	XX
09	Char. Scan Lines	07	07	01	01	00	00
0A	Cursor Scan Start	06	06	XX	XX	XX	XX
0B	Cursor Scan End	07	07	XX	XX	XX	XX
0C	Start of Screen (High)	00	00	00	00	00	00
0D	Start of Screen (Low)	00	00	00	00	00	00
0E	Cursor Position (High)	00	00	XX	XX	XX	XX
0F	Cursor Position (Low)	00	00	XX	XX	XX	XX
10	Mode Control	18	18	18	18	1A	19
11	Interrupt Control	30	30	30	30	30	30
12	Char. Gen/Sync Pol.	46	46	46	46	04	46
13	Char. Font Pointer	00	00	XX	XX	XX	XX
14	Char. to Load	FF	FF	XX	XX	XX	XX

Figure 1-56. Memory Controller Initialization

Address	Data Register Description	Modes 0,1	2,3	4,5	6	11	13
3C6	PEL Mask	FF	FF	FF	FF	FF	FF
3D8	CGA Mode Control	28	29	0A	18	18	08
3D9	CGA Border Control	30	30	30	3F	3F	30
3DA	Status	XX	XX	XX	XX	XX	XX
3DB	Reserved	XX	XX	XX	XX	XX	XX
3DC	Reserved	XX	XX	XX	XX	XX	XX
3DD	Extended Mode Control	00	00	00	00	00	04
3DE	Reserved						
3DF	Reserved						

Figure 1-57. Video Formatter Initialization Table

3C8 Index	R	3C9 G	B	Display Color
00	00	00	00	Black
01	00	00	2A	Blue
02	00	2A	00	Green
03	00	2A	2A	Cyan
04	2A	00	00	Red
05	2A	00	2A	Magenta
06	2A	15	00	Brown
07	2A	2A	2A	White
08	15	15	15	Gray
09	15	15	3F	Light Blue
0A	15	3F	15	Light Green
0B	15	3F	3F	Light Cyan
0C	3F	15	15	Light Red
0D	3F	15	3F	Light Magenta
0E	3F	3F	15	Yellow
0F	3F	3F	3F	Bright White

Figure 1-58. 16-Color Compatibility Initialization

Note: The information in the preceding table is not applicable to the LCD.

RAM-Loadable Fonts

In the text modes, the video buffer is divided into two data areas: the text area at address B8000 and the character font tables at address A0000. The text area consists of the character and attribute code for each position on the display. The font table consists of the character code and PEL data for each character in the set.

Restrictions are placed on where the character font can be loaded into the video buffer. Four fonts are supported in text modes. The memory map below shows the areas (blocks) in the video buffer where the fonts are loaded. The font tables can be swapped in synchronization with the 'vertical retrace' signal with several output commands. A maximum of four fonts can be loaded into the font area, but only two can be loaded into and displayed from the character generator at any one time. Two fonts are provided in ROM, an 8-by-8 font and an 8-by-16 font. The font loaded depends on the mode that is active at the time.

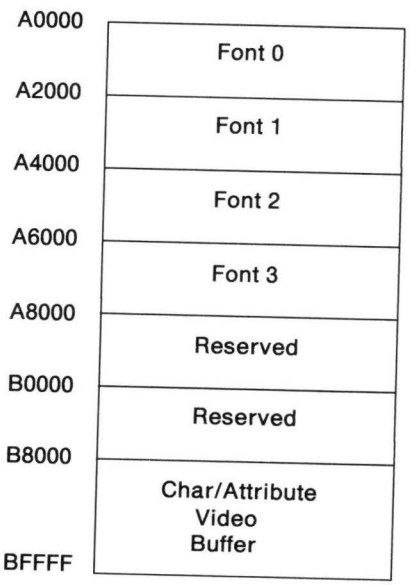


Figure 1-59. Font Memory Map

The following is an example of how the character "E" is defined in an 8-by-16 character box.

Scan Lines	Data in Hex	Data in Binary
0	00	00000000
1	00	00000000
2	7E	01111110
3	7E	01111110
4	60	01100000
5	60	01100000
6	7E	01111110
7	7E	01111110
8	60	01100000
9	60	01100000
10	7E	01111110
11	7E	01111110
12	00	00000000
13	00	00000000
14	00	00000000
15	00	00000000

Figure 1-60. Sample Character

The following programming example uses the BIOS routine to load a font table into block 0. Because of differences in the hardware, the character generator is not loaded the same for all display adapters; however, the BIOS routines are the same for all video subsystems with RAM-loadable fonts. The Type 7690, for instance, supports only 8-by-8 and 8-by-16 character fonts, depending on the mode selected.

```
TITLE   Load block 0 with character definitions from "SET_A"

CODE    SEGMENT PARA 'CODE'
        ASSUME  CS:CODE,ES:CODE

EX1     PROC  NEAR
        MOV   AX,0001H          ; Mode set BIOS call for mode 1
        INT   10H
        MOV   CX,100H           ; Load 256 characters into the block
        MOV   DX,0000H          ; Begin loading at offset zero
        MOV   BL,00H            ; Load the characters into block zero
        MOV   BH,10H            ; 16 bytes per character definition
        MOV   AX,SEG SET_A      ; Get the segment of the characters
        MOV   ES,AX             ; ES = segment of character definitions
        MOV   AH,11H            ; Character generator routines
        MOV   AL,00H            ; User alpha load BIOS call
        MOV   BP,OFFSET SET_A   ; BP = offset of character definitions
        INT   10H
        RET
EX1     ENDP

;----8-by-16 definitions for "SET_A"

SET_A       LABEL   BYTE
            INCLUDE SET_A_CHARS
SET_A_END   EQU   $
CODE        ENDS
            END
```

Block 0 now contains the 256 character definitions from file SET_A. To load block 1, change the block number, the character file pointer, and the pointer for the block to be loaded, as indicated below.

```
EX2     PROC  NEAR
        MOV   CX,100H           ; Load 256 characters into the block
        MOV   DX,0000H          ; Begin loading at offset zero
        MOV   BL,01H            ; Load the characters into block one
        MOV   BH,10H            ; 16 bytes per character definition
        MOV   AX,SEG SET_B      ; Get the segment of the characters
        MOV   ES,AX             ; ES = segment of character definitions
        MOV   AH,11H            ; Character generator routines
        MOV   AL,00H            ; User alpha load BIOS call
        MOV   BP,OFFSET SET_B   ; BP = offset of character definitions
        INT   10H
        RET
EX2     ENDP

;----8-by-16 definitions for "SET_B"

SET_B       LABEL   BYTE
            INCLUDE SET_B_CHARS

SET_B_END   EQU   $
```

Blocks 2 and 3 can be loaded in the same manner, until all four blocks contain character font information. The characters that were loaded into the blocks are not available for display until they are transferred to the character generator.

The character generator is broken into two parts, or font pages. Each font page contains 256 character definitions. The character generator is loaded from the four blocks of 256 character definitions.

A character set of 256 characters is loaded into the character generator by selecting one of the four blocks to be transferred. Two of the four blocks are selected for a character set of 512 characters. The Set Block Specifier call is used to transfer the blocks of character definitions to the character generator.

The Set Block Specifier call uses the input parameter in BL to specify which blocks are loaded into the character generator. Only the low nibble (4 bits) of BL is used. Bits 1 and 0 specify which block to load into the first 256 positions of the character generator, or font page 0. The first 256 positions are the character definitions for characters 0 - 255. Bits 3 and 2 indicate which block to load into the second 256 positions of the character generator, or font page 1. The second 256 positions of the character generator define characters 256 - 511. If the two bit pairs are equal (bit 0 is the same as bit 2 and bit 1 is the same as bit 3), only font page 0 is loaded, which limits the character set to 256 characters. The following figure summarizes the bit patterns that indicate with which blocks the character generator is loaded.

Bit Number					
3	**2**	**1**	**0**	**Font Page 1**	**Font Page 0**
0	0	0	0	Not Used	Block 0
0	0	0	1	Block 0	Block 1
0	0	1	0	Block 0	Block 2
0	0	1	1	Block 0	Block 3
0	1	0	0	Block 1	Block 0
0	1	0	1	Not Used	Block 1
0	1	1	0	Block 1	Block 2
0	1	1	1	Block 1	Block 3
1	0	0	0	Block 2	Block 0
1	0	0	1	Block 2	Block 1
1	0	1	0	Not Used	Block 2
1	0	1	1	Block 2	Block 3
1	1	0	0	Block 3	Block 0
1	1	0	1	Block 3	Block 1
1	1	1	0	Block 3	Block 2
1	1	1	1	Not Used	Block 3

Figure 1-61. Block Specifier

To load block 0 into font page 0 and block 3 into font page 1, the following BIOS call is used.

```
MOV   AH,11H            ; Character generator routines
MOV   AL,03H            ; Set block specifier BIOS call
MOV   BL,0CH            ; Character generator block specifier
INT   10H
```

Font page 0 now contains the character definitions from block 0, and font page 1 the character definitions from block 3. Because font page 0 specifies characters 0 through 255, and font page 1 specifies the characters 256 through 511, 512 characters are now available for display. The BIOS write character routines, however, accept the AL register as the character to be displayed. That allows a range of characters starting at 0 and stopping at 255, and appears to limit the number of characters to 256. The solution is to use a bit in the attribute byte to specify the font page (see "Programming Considerations" later in this section). Whenever a 512 character set

is available, bit 3 of the attribute byte selects font page 0 (characters 0 - 255) or font page 1 (characters 256 - 511). If bit 3 is 1, font page 1 is used; if the bit is 0, font page 0 is used.

To display character hex 30, the following BIOS call can be used.

```
MOV   AH,09H        ; Write attribute/character at cursor pos.
MOV   AL,30H        ; AL = character to write
MOV   BH,00H        ; Display page 0
MOV   CX,1          ; Display 1 character

MOV   BL,07H        ; White character on black background
INT   10H           ; Attribute bit off selects font page 0
```

To display character hex 130 (304), the following BIOS call can be used. Attribute bit 3 is still used as the intensity bit in alpha modes.

```
MOV   AH,09H        ; Write attribute/character at cursor pos.
MOV   AL,30H        ; AL = character to write
MOV   BH,00H        ; Display page 0
MOV   CX,1          ; Display 1 character
MOV   BL,07H        ; Intense white character on black background
OR    BL,08H        ; Turn on attribute bit 3 to select font page 1
INT   10H
```


Alternate Parameter Table

A table in BIOS, SAVE_TBL, is used to maintain various tables and save areas. Each entry in this table is a doubleword. The format for this table is:

Entry	Description
1	Video Parameter Table Pointer This must point to the video parameter table in BIOS.
2	Reserved = 0
3	Alpha Mode Auxiliary Font Pointer This is a pointer to a descriptor table used during a mode set to select a user font in A/N mode. The table has the following format: **Size** — **Description** Byte — Bytes per character Byte — Block to load, should be 00 for normal operation Word — Count to store, should be hex 100 for normal operation Word — Character offset, should be 00 for normal operation DWord — Pointer to a font table Byte — Displayable rows, if the value is FF, the maximum calculated value will be used; otherwise, this value is used. Byte — Consecutive bytes of mode values for which this font description is to be used. The end of this stream is indicated by a byte code of FF.
4	Graphics Mode Auxiliary Pointer This is a pointer to a descriptor table used during a mode set to select a user font in graphics mode. The table has the following format: **Size** — **Description** Byte — Displayable rows Word — Bytes per character DWord — Pointer to a font table Byte — Consecutive bytes of mode values for which this font description is to be used. The end of this stream is indicated by a byte code of FF.
5 - 7	Reserved as all 0's.

Figure 1-62. Alternate Parameter Table

Normally, the auxiliary pointers, the third and fourth entries, are set to all zeros. The Mode Set looks at these values and, if they are zero, goes to the BIOS font table. If they are not zero, the Mode Set loads the user font pointed to by the auxiliary pointer.

The pointer for SAVE_TBL exists at 40:A8. To use your own table, create two tables, SAVE_TBL and, optionally, the font descriptor table. Then set the pointer to point to the new SAVE_TBL.

Programming Considerations

Interrupt Usage: The Type 7690 video subsystem can be programmed to create an interrupt at the end of each vertical display refresh time. An interrupt handler must be written by the application to take advantage of this feature. The vertical retrace interrupt is on IRQ2. (This interrupt does not support interrupt sharing).

The programmer can poll the Interrupt Control register, port 3D5 index 11, to determine whether the video caused the interrupt. The IRQ2 status bit indicates that a vertical retrace interrupt did occur; it does not indicate that the video is still in retrace. To find the status of the 'vertical retrace' signal, check the CGA Status register, port 3DA.

The Interrupt Control register also has 2 bits that control the interrupt circuitry and 1 bit that controls the output of the video formatter. To enable the interrupt:

1. Clear bit 4 to clear the interrupt latch.
2. Clear bit 5 to enable the interrupt.
3. Set bit 4 to enable the latch.

512 Character Set: When using a 512 character set on the Type 7690, the following procedures are recommended to maintain consistent colors.

1. Set the block specifier, (AX) = 1103H.
2. Set the colors for 512, (AX) = 1000H (BX) = 0712H.
3. Reload the first eight colors into the palette.

Note: The character hex 20 (normally a space) is used to fill the blank area of the screen. Therefore, it is recommended that character hex 20 be a blank space.

Diskette Drive Interface

The diskette gate array contains the decode logic for the internal registers, the write logic, and the read logic. The gate array:

- Controls the clock signals needed for read and write
- Controls write precompensation
- Selects the data rate of transfer
- Provides a mask for the interrupt and DMA request lines
- Provides phase error detection for input to the phase-lock loop.

The phase detector/amplifier and the voltage controlled oscillator (VCO) make up the phase-lock loop (PLL). They adjust the clock used during data read to keep it in phase with the data signal.

The drives connect to the system board through a single 40-pin connector, which supplies all signals necessary to operate two diskette drives. The diskette drives are attached to the connector through an internal, flat cable.

Gate Array Registers

The diskette gate array has five registers: three registers that show the status of signals used in diskette operations, and two registers that control certain interface signals.

RAS Port A Register: The RAS Port A register, hex 3F0, is a read-only register that shows the status of the corresponding signals.

Bit	Function
7	IRQ6
6	DRQ2
5	Step (latched)
4	Track 0
3	-Head 1 Select
2	Index
1	Write Protect
0	-Direction

Figure 1-63. RAS Port A, Hex 3F0

RAS Port B Register: The RAS Port B register, hex 3F1, is a read-only register that shows the status of signals between the diskette drive and the controller.

Bit	Function
7	Reserved
6	-Drive Select 1
5	-Drive Select 0
4	Write Data (latched)
3	Read Data (latched)
2	Write Enable (latched)
1	-Drive Select 3
0	-Drive Select 2

Figure 1-64. RAS Port B, Hex 3F1

Digital Output Register: The Digital Output register (DOR), hex 3F2, is a write-only register that controls drive motors, drive selection, and feature enables. All bits are cleared by a reset.

Bit	Function
7	Motor Enable 3
6	Motor Enable 2
5	Motor Enable 1
4	Motor Enable 0
3	DMA and Interrupt Enable
2	-Controller Reset
1,0	Drive Select 0 through 3
	00 selects drive 0
	01 selects drive 1
	10 selects drive 2
	11 selects drive 3

Figure 1-65. Digital Output, Hex 3F2

Digital Input Register: The Digital Input register, hex 3F7, is a read-only register used to sense the state of the 'diskette change' signal. It is also used for diagnostic purposes.

Bit	Function
7	-Diskette Change
6 to 4	Reserved
3	DMA Enable
2	No Write Precomp
1	250K bps Rate Select
0	Reserved

Figure 1-66. Digital Input, Hex 3F7

Configuration Control Register: The Configuration Control register, hex 3F7, is a write-only register used to set the transfer rate and select write precompensation.

Bit	Function
7	Reserved = 0
6	Reserved = 0
5	Reserved = 0
4	Reserved = 0
3	Reserved = 0
2	No Write Precomp
1	250K bps Rate Select
0	Reserved = 0

Figure 1-67. Configuration Control, Hex 3F7

Controller Registers

The diskette controller has two registers that are accessed by the microprocessor: the Main Status register and the data register. The Main Status register, hex 3F4, has the status information about the controller and may be read at any time.

Data Registers, Hex 3F5: This address, hex 3F5, consists of several registers in a stack, with only one register presented to the data bus at a time. It stores data, commands, and parameters, and provides diskette-drive status information. Data bytes are passed through the data register to program or obtain results after a command.

Main Status Register, Hex 3F4: This register is read-only and is used to facilitate the transfer of data between the microprocessor and the controller.

Bit	Function
7	Request for Master
6	Data Input/Output
5	Non-DMA Mode
4	Diskette Controller Busy
3, 2	Reserved
1	Drive 1 Busy
0	Drive 0 Busy

Figure 1-68. Main Status Register

The bits are defined as follows:

Bit 7 The data register is ready for transfer with the microprocessor.

Bit 6 This bit indicates the direction of data transfer between the diskette controller and the microprocessor. If this bit is set to 1, the transfer is from the controller to the microprocessor; if it is clear, the transfer is from the microprocessor.

Bit 5 When this bit is set to 1, the controller is in the non-DMA mode.

Bit 4 When this bit is set to 1, a Read or Write command is being executed.

Bits 3, 2 Reserved

Bit 1 Drive 1 Busy—When set to 1, diskette drive 1 is in the seek mode.

Bit 0 Drive 0 Busy—When set to 1, diskette drive 0 is in the seek mode.

Commands

The diskette controller performs the commands listed below. Each command is initiated by a multibyte transfer from the microprocessor, and the result can also be a multibyte transfer back to the microprocessor. Because of this multibyte interchange of information between the controller and the microprocessor, each command is considered to consist of three phases:

Command Phase: The microprocessor issues a series of Writes to the controller that direct it to perform a specific operation.

Execution Phase: The controller performs the specified operation.

Result Phase: After completion of the operation, status and other housekeeping information are made available to the microprocessor through a sequence of Read commands from the microprocessor.

The following is a list of controller commands:

- Read Data
- Read Deleted Data
- Read a Track
- Read ID
- Write Data
- Write Deleted Data
- Format a Track
- Scan Equal
- Scan Low or Equal
- Scan High or Equal
- Recalibrate
- Sense Interrupt Status
- Specify
- Sense Drive Status
- Seek.

Symbol Descriptions: Following are descriptions of symbols used in the following section, "Command Format."

A0 Address Line 0—When clear, A0 selects the Main Status register; when set to 1, it selects the Data register.

DTL Data Length—When N is 00, DTL is the data length to be read from or written to a sector.

EOT End of Track—The final sector number on a cylinder.

GPL Gap Length—The length of gap 3 (spacing between sectors excluding the VCO synchronous field).

H Head Address—The head number, either 0 or 1, as specified in the ID field.

HD Head—The selected head number, 0 or 1. (H = HD in all command words.)

HLT Head Load Time—The head load time in the selected drive (2 to 256 milliseconds in 2-millisecond increments).

HUT Head Unload Time—The head unload time after a read or write operation (0 to 240 milliseconds in 16-millisecond increments).

MF FM or MFM Mode—A 0 selects FM mode and a 1 selects MFM (MFM is selected only if it is implemented).

MT Multitrack—A 1 selects multitrack operation. (Both HD0 and HD1 will be read or written.)

N Number—The number of data bytes written in a sector.

NCN New Cylinder—The new cylinder number for a seek operation.

ND Nondata Mode—This indicates an operation in the nondata mode.

PCN Present Cylinder Number—The cylinder number at the completion of a Sense Interrupt Status command (present position of the head).

R Record—The sector number to be read or written.

SC Sector—The number of sectors per cylinder.

SK Skip—The skip deleted-data address mark.

SRT Stepping Rate—These four bits indicate the stepping rate for the diskette drive as follows:

1111 1 ms
1110 2 ms
1101 3 ms

ST0 - 3 Status 0 through Status 3—The four registers that store status information after a command is executed.

STP Scan Test—If STP is 01, the data in adjacent sectors is compared with the data sent by the microprocessor during a scan operation. If STP is 02, alternate sectors are read and compared.

US0 - 1 Unit Select—The selected driver number, encoded the same as bits 0 and 1 of the Digital Output register.

Command Format

The following are commands that may be issued to the controller. An X is used to indicate a don't-care condition.

Read Data

Command Phase

MT = Multitrack
MF = MFM Mode
SK = Skip Deleted-Data Address Mark
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	0	0	1	1	0
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 1-69. Read Data Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-70. Read Data Result

Read Deleted Data

Command Phase

MT = Multitrack
MF = MFM Mode
SK = Skip Deleted-Data Address Mark
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	0	1	1	0	0
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 1-71. Read Deleted Data Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-72. Read Deleted Data Result

Read a Track

Command Phase

MF = MFM Mode
SK = Skip Deleted-Data Address Mark
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	0	MF	SK	0	0	0	1	0
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 1-73. Read a Track Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-74. Read a Track Result

Read ID

Command Phase

MF = MFM Mode
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	0	MF	0	0	0	1	1	0
Byte 1	X	X	X	X	X	HD	US1	US0

Figure 1-75. Read ID Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-76. Read ID Result

Write Data

Command Phase

MT = Multitrack
MF = MFM Mode
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	0	0	0	1	1	0
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 1-77. Write Data Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-78. Write Data Result

Write Deleted Data

Command Phase

MT = Multitrack
MF = MFM Mode
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	0	0	0	1	1	0
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 1-79. Write Deleted Data Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-80. Write Deleted Data Result

Format a Track

Command Phase

MF = MFM Mode
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	0	MF	0	0	1	1	0	0
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Number of Data Bytes in Sector							
Byte 3	Sectors per Cylinder							
Byte 4	Gap Length							
Byte 5	Data							

Figure 1-81. Format a Track Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-82. Format a Track Result

Scan Equal

Command Phase

MT = Multitrack
MF = MFM Mode
SK = Skip Deleted-Data Address Mark
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	1	0	0	0	1
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Scan Test							

Figure 1-83. Scan Equal Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-84. Scan Equal Result

Scan Low or Equal

Command Phase

MT = Multitrack
MF = MFM Mode
SK = Skip Deleted-Data Address Mark
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	1	1	0	0	1
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Scan Test							

Figure 1-85. Scan Low or Equal Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-86. Scan Low or Equal Result

Scan High or Equal

Command Phase

MT = Multitrack
MF = MFM Mode
SK = Skip Deleted-Data Address Mark
HD = Head Number
USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	1	1	1	0	1
Byte 1	X	X	X	X	X	HD	US1	US0
Byte 2	Cylinder Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End of Track							
Byte 7	Gap Length							
Byte 8	Scan Test							

Figure 1-87. Scan High or Equal Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Status Register 1							
Byte 2	Status Register 2							
Byte 3	Cylinder Number							
Byte 4	Head Address							
Byte 5	Sector Number							
Byte 6	Number of Data Bytes in Sector							

Figure 1-88. Scan High or Equal Result

Recalibrate

Command Phase

USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	1	1
Byte 1	X	X	X	X	X	0	US1	US0

Figure 1-89. Recalibrate Command

Result Phase: This command has no result phase.

Sense Interrupt Status

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	0	0	0

Figure 1-90. Sense Interrupt Status Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status Register 0							
Byte 1	Present Cylinder Number							

Figure 1-91. Sense Interrupt Status Result

Specify

Command Phase

SRT = Diskette Stepping Rate
HUT = Head Unload Time
HLT = Head Load Time
ND = NonData Mode

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	1	1
Byte 1	SRT					HUT		
Byte 2	HLT						ND	

Figure 1-92. Specify Command

Result Phase: This command has no result phase.

Sense Drive Status

Command Phase

USx = Unit Select
HD = Head Number

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	1	0
Byte 1	X	X	X	X	X	HD	US1	US0

Figure 1-93. Sense Drive Status Command

Result Phase

	7	6	5	4	3	2	1	0
Byte 0	Status 3 Register							

Figure 1-94. Sense Drive Status Result

Seek

Command Phase

USx = Unit Select

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	1	1
Byte 1	X	X	X	X	X	0	US1	US0
Byte 2	New Cylinder Number for Seek							

Figure 1-95. Seek Command

Result Phase: This command has no result phase.

Invalid Commands

Result Phase: The following status byte is returned to the microprocessor when an invalid command has been received.

	7	6	5	4	3	2	1	0
Byte 0	Status 0 Register							

Figure 1-96. Invalid Command Result

Command Status Registers

The following are definitions of the status registers ST0 through ST3.

Status 0 Register (ST0)

The following are bit definitions for the Status 0 register:

Bit 7, 6 Interrupt Code (IC)

- **00** Normal Termination of Command—The command was completed and properly executed.
- **01** Abrupt Termination of Command—The execution of the command was started but not successfully completed.
- **10** Invalid Command Issue—The issued command was never started.
- **11** Abnormal Termination—During the execution of a command, the 'ready' signal from the diskette drive changed state.

Bit 5 Seek End—Set to 1 when the controller completes the Seek command.

Bit 4 Equipment Check—Set if a 'fault' signal is received from the diskette drive, or if the 'track 0' signal fails to occur after 77 step pulses (Recalibrate command).

Bit 3 Not Ready—This flag is set when the diskette drive is in the not-ready state and a Read or Write command is issued.

Bit 2 Head Address—Indicates the state of the head at interrupt.

Bit 1, 0 Unit select 1 and 0 (US 1 and 0)—Indicate a drive's unit number at interrupt.

Status 1 Register (ST1)

The following are bit definitions for the Status 1 register:

Bit 7 End of Cylinder—Set when the controller tries to gain access to a sector beyond the final sector of a cylinder.

Bit 6 Reserved.

Bit 5 Data Error—Set when the controller detects a CRC error in either the ID field or the data field.

Bit 4 Overrun—Set if the controller is not serviced by the main system within a certain time limit during data transfers.

Bit 3 Reserved.

Bit 2 No Data—Set if the controller cannot find the sector specified in the ID register during the execution of a Read Data, Write Deleted Data, or Scan command. This flag is also set if the controller cannot read the ID field without an error during the execution of a Read ID command, or if the starting sector cannot be found during the execution of a Read Cylinder command.

Bit 1 Not Writable—Set if the controller detects a 'write-protect' signal from the diskette drive during execution of a Write Data, Write Deleted Data, or Format a Track command.

Bit 0 Missing Address Mark—Set if the controller cannot detect the ID address mark. At the same time, bit 0 of the Status 2 register is set.

Status 2 Register (ST2)

The following are bit definitions for the Status 2 register:

Bit 7 Reserved = 0.

Bit 6 Control Mark—This flag is set if the controller encounters a sector that has a deleted-data address mark during execution of a Read Data or Scan command.

Bit 5 Data Error in Data Field—Set if the controller detects an error in the data.

Bit 4 Wrong Cylinder—This flag is related to ND and is set when the content of C is different from that stored in the ID register.

Bit 3 Scan Equal Hit (SH)—Set if the adjacent sector data equals the microprocessor data during the execution of a Scan command.

Bit 2 Scan Not Satisfied (SN)—Set if the controller cannot find a sector on the cylinder that meets the condition during a Scan command.

Bit 1 Bad Cylinder—Related to ND and is set when the contents of C on the medium are different from that stored in the ID register or when the content of C is hex FF.

Bit 0 Missing Address Mark in Data Field— Set if the controller cannot find a data address mark or a deleted-data address mark when data is read.

Status 3 Register (ST3)

The following are bit definitions for the Status 3 register:

Bit 7 Fault—Status of the 'fault' signal from the diskette drive.

Bit 6 Write Protect—Status of the '-write protect' signal from the diskette drive.

Bit 5 Ready—Status of the 'ready' signal from the diskette drive.

Bit 4 Track 0—Status of the '-track 0' signal from the diskette drive.

Bit 3 Two Side—Status of the 'two side' signal from the diskette drive.

Bit 2 Head Address—Status of the '-head 1 select' signal from the diskette drive.

Bit 1 Unit Select 1—Status of the '-drive select 1' signal from the diskette drive.

Bit 0 Unit Select 0—Status of the '-drive select 0' signal from the diskette drive.

Signal Description

All signals are 74HCT series-compatible in both rise and fall times and in interface levels. The following are the input signals to the diskette drive. These signal thresholds are +2.0 Vdc high and +0.8 Vdc low.

-Drive Select 0—1: The select lines provide the means to enable or disable the drive interface lines. When the signal is active, the drive is enabled. When the signal is inactive, all control inputs are ignored, and the drive outputs are disabled. The maximum drive-select delay time is 500 ns.

-Motor Enable 0—1: When this signal is made active, the spindle starts to turn. When it is made inactive, the spindle slows to a stop.

-Step: An active pulse on this line causes the head to move one track. The minimum pulse width is 1 µs. The direction of the head motion is determined by the state of the '-direction' signal at the trailing edge of the '-step' pulse.

-Direction: When this signal is active, the head moves to the next higher track (toward the spindle) for each '-step' pulse. When the signal is inactive, the head moves toward track 0. This signal must be stable for 1 µs before and after the trailing edge of the '-step' pulse.

-Head 1 Select: When this signal is active, the upper head (head 1) is selected. When it is inactive, the lower head (head 0) is selected.

-Write Enable: When this signal is active, the write-current circuits are enabled and data can be written under the control of the '-write data' signal. This signal must be active 8 µs before data can be written.

-Write Data: An active pulse on this line writes a 1. These pulses have a 4-, 6-, or 8-µs spacing with a width of 250 ns for the 250,000-bps transfer rate. Write precompensation of 125 ns is done by the diskette gate array.

The following are the output signals from the diskette drive. These signal thresholds are +3.7 Vdc high and +0.4 Vdc low.

-Index: An active pulse of 1 ms indicates the diskette index.

-Track 0: When this signal is active, the head is on track 0. This signal is used to determine whether the drive is present. If, after commanding the drive to seek track 0, the '-track 0' signal does not go active, the drive is not present.

-Write Protect: This signal is active when the write-protect window is uncovered. When this happens, the write current circuits are disabled.

-Read Data: An active pulse on this line writes a logical 1. The pulse width for the 250,000-bps rate is 250 ns.

-Diskette Change: This signal is active at power-on and whenever the diskette is removed. It remains active until a diskette is present and a '-step' pulse is received.

Connector

The following shows the signals and pin assignments for the connector.

Pin	I/O	Signal	Pin	I/O	Signal
1	N/A	Signal Ground	2	N/A	Reserved
3	N/A	Signal Ground	4	N/A	Reserved
5	N/A	Signal Ground	6	N/A	Not Connected
7	N/A	Signal Ground	8	I	-Index
9	N/A	Signal Ground	10	O	-Motor Enable 1
11	N/A	Signal Ground	12	O	-Drive Select 0
13	N/A	Signal Ground	14	O	-Drive Select 1
15	N/A	Signal Ground	16	O	-Motor Enable 0
17	N/A	Signal Ground	18	O	-Direction
19	N/A	Signal Ground	20	O	-Step
21	N/A	Signal Ground	22	O	-Write Data
23	N/A	Signal Ground	24	O	-Write Enable
25	N/A	Signal Ground	26	I	-Track 0
27	N/A	Signal Ground	28	I	-Write Protect
29	N/A	Signal Ground	30	I	-Read Data
31	N/A	Signal Ground	32	O	-Head 1 Select
33	N/A	Signal Ground	34	I	-Diskette Change
35	N/A	Ground	36	N/A	Ground
37	N/A	Ground	38	O	+5 Vdc
39	N/A	Ground	40	O	+12 Vdc

Figure 1-97. Diskette Drive Connector

Serial Port

The serial port is fully programmable and supports asynchronous communications. It will add and remove start, stop, and parity bits. A programmable baud-rate generator allows operation from 50 baud to 9600 baud. The port supports 5-, 6-, 7-, and 8-bit characters with 1, 1.5, or 2 stop bits. A prioritized interrupt system controls transmit, receive, error, and line status as well as data set interrupts.

The rear of the system unit has a 25-pin D-shell connector that contains standard Electronic Industries Association (EIA) RS-232C interface signals.

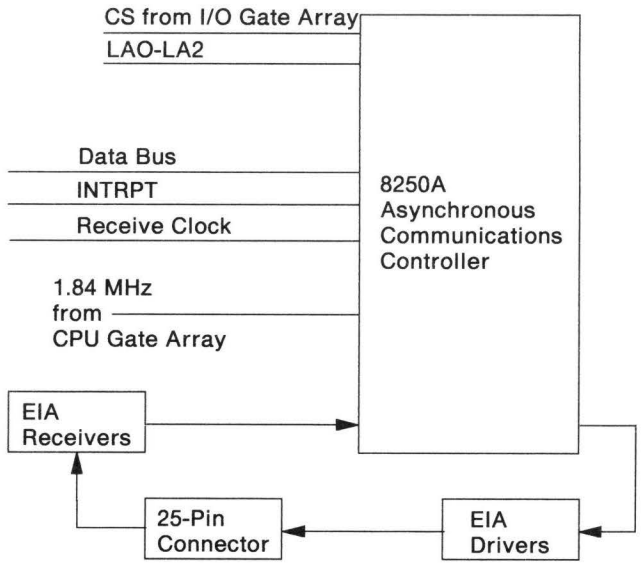


Figure 1-98. Serial Port Block Diagram

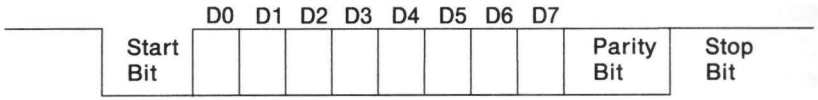
The serial port has a controller that provides the following functions:

- Adds or deletes standard, asynchronous communications bits to or from a serial data stream.
- Provides full double buffering, which eliminates the need for precise synchronization.
- Provides a programmable baud-rate generator.
- Provides modem controls (CTS, RTS, DSR, DTR, RI, and CD).

Application

The serial port is addressed as communications port 1, addresses hex 3F8 through 3FF; the port uses interrupt 4.

The data format is:



Data bit 0 is the first bit to be sent or received. The controller automatically inserts the start bit, the correct parity bit (if programmed to do so), and the stop bit (1, 1.5, or 2, depending on the command in the Line Control register).

Controller Registers

The register addresses are hex 3F8 through 3FF. These registers control the controller's operations and are used to transmit and receive data. The divisor latch access bit (DLAB), which is the most significant bit of the Line Control register, affects the selection of the divisor latches for the baud rate generator.

Specific registers are selected according to the following figure:

DLAB State	Port 1 Address	Read/Write	Register
0	03F8	W	Transmitter Holding
0	03F8	R	Receiver Buffer
1	03F8	R/W	Divisor Latch, Low Byte
1	03F9	R/W	Divisor Latch, High Byte
0	03F9	R/W	Interrupt Enable Register
X	03FA	R	Interrupt Identification Register
X	03FB	R/W	Line Control Register
X	03FC	R/W	Modem Control Register
X	03FD	R	Line Status Register
X	03FE	R	Modem Status Register
X	03FF	R/W	Scratch Register

Figure 1-99. Serial Port Addresses

Transmitter Holding Register, Hex 3F8: This register contains the character to be sent. Bit 0 is the least significant bit and the first bit sent serially.

Bit	Function
7	Bit 7
6	Bit 6
5	Bit 5
4	Bit 4
3	Bit 3
2	Bit 2
1	Bit 1
0	Bit 0

Figure 1-100. Transmitter Holding Register

Receiver Buffer Register, Hex 3F8: This register contains the received character. Bit 0 is the least significant bit and the first bit received serially.

Bit	Function
7	Bit 7
6	Bit 6
5	Bit 5
4	Bit 4
3	Bit 3
2	Bit 2
1	Bit 1
0	Bit 0

Figure 1-101. Receiver Buffer Register

Divisor Latch, 3F9 and 3F8: These two registers access the high byte (3F9) and low byte (3F8) of the divisor latch. More information about the divisor latch may be found under "Programmable Baud-Rate Generator" later in this section.

Bit	High Byte	Low Byte
7	Bit 15	Bit 7
6	Bit 14	Bit 6
5	Bit 13	Bit 5
4	Bit 12	Bit 4
3	Bit 11	Bit 3
2	Bit 10	Bit 2
1	Bit 9	Bit 1
0	Bit 8	Bit 0

Figure 1-102. Divisor Latch

Interrupt Enable Register, Hex 3F9: This register allows the four types of controller interrupts to separately activate the 'chip-interrupt' (INTRPT) output signal. The interrupt system can be totally disabled by resetting bits 3 through 0 of the Interrupt Enable register to 0. Similarly, by setting the appropriate bits of this register to 1, selected interrupts are enabled. Disabling the interrupt system inhibits the Interrupt Enable register and the active INTRPT output from the chip. All other system functions operate normally, including the setting of the Line Status and Modem Status registers.

Bit	Function
7	Reserved = 0
6	Reserved = 0
5	Reserved = 0
4	Reserved = 0
3	Enable Modem Status
2	Enable Rx Line Status
1	Enable Tx Buffer Empty
0	Enable Data Available

Figure 1-103. Interrupt Enable Register

Bits 7-4 Reserved = 0.

Bit 3 When set to 1, this bit enables the modem status interrupt.

Bit 2 When set to 1, this bit enables the receiver-line-status interrupt.

Bit 1 When set to 1, this bit enables the transmitter-holding-register-empty interrupt.

Bit 0 When set to 1, this bit enables the received-data-available interrupt.

Interrupt Identification Register (IIR), Hex 3FA: The controller has an internal interrupt capability that makes communications possible with reduced microprocessor intervention. To minimize programming overhead during data character transfers, the controller prioritizes interrupts into four levels: receiver line status (priority 1), received data ready (priority 2), transmitter holding register empty (priority 3), and modem status (priority 4).

Information about a pending interrupt is stored in the Interrupt Identification register. When addressed during chip-select time, this register stops the pending interrupt with the highest priority, and no other interrupts are acknowledged until the microprocessor services that particular interrupt.

Bit	Function
7	Reserved = 0
6	Reserved = 0
5	Reserved = 0
4	Reserved = 0
3	Reserved = 0
2	Interrupt ID Bit 1
1	Interrupt ID Bit 0
0	Interrupt Not Pending

Figure 1-104. Interrupt Identification Register

Bits 7-3 Reserved = 0.

Bits 2,1 These two bits, Interrupt ID 1 and 0, identify the pending interrupts as shown.

IIR Bits 2	1	Priority	Type	Interrupt Control Cause	To Reset
1	1	Highest	Receiver Line Status	Overrun, Parity, or Framing Error, or Break Interrupt	Read the Line Status Register
1	0	Second	Received Data Available	Data in Receiver Buffer	Read the Receiver Buffer Register
0	1	Third	Transmitter Holding Register Empty	THR is Empty	Read IIR or Write to THR
0	0	Fourth	Modem Status	Change in a Signal's Status from the Modem	Read the Modem Status Register

Bit 0 This bit can be used in either hard-wired, prioritized, or polled conditions to indicate whether an interrupt is pending. When bit 0 is 0, an interrupt is pending, and the Interrupt Identification register contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a 1, no interrupt is pending, and polling (if used) continues.

Line Control Register, Hex 3FB: This register specifies the format of the asynchronous data communications exchange. The register can also be read at any time, eliminating the need to store line characteristics separately in memory.

Bit	Function
7	DLAB
6	Set Break
5	Stick Parity
4	Even Parity Select
3	Parity Enable
2	Number of Stop Bits
1	Word Length Select 1
0	Word Length Select 0

Figure 1-105. Line Control Register

Bit 7 This is the divisor-latch access bit. It is set to 1 to gain access to the divisor latches of the baud-rate generator during a read or write operation. It is cleared to gain access to the Receiver Buffer, the Transmitter Holding, or the Interrupt Enable registers.

Bit 6 This bit is the set-break control bit. When bit 6 is set to 1, the serial output is forced to an inactive level and remains there regardless of other transmitter activity. The set-break is disabled by clearing bit 6 to 0.

Bit 5 This bit is the stick-parity bit. When this bit is set to 1 and parity is enabled, the parity bit is sent as a 0 if parity is even, or as a 1 if parity is odd.

Bit 4 This bit is the even-parity-select bit. When set to 1 and parity is enabled, an even number of logical 1's are sent or checked. When cleared to 0, an odd number of bits are sent or checked.

Bit 3 This bit is the parity-enable bit. When this bit is set to 1, a parity bit is sent or checked. The parity bit is used to produce an even or odd number of 1's when the bits in the data word and the parity bit are summed.

Bit 2 This bit specifies the number of stop bits in each serial character that is sent or received. When set to 1 and a word length greater than 5 is specified, 2 stop bits are generated or checked. If the word length is 5 and this bit is set to 1, then 1.5 stop bits are generated or checked. When this bit is cleared to 0, then 1 stop bit is specified.

Bits 1,0 These 2 bits specify the number of bits in each serial character that is sent or received. The encoding of these bits is as follows:

Bit 1	Bit 0	Word Length in Bits
0	0	5
0	1	6
1	0	7
1	1	8

Modem Control Register, Hex 3FC: The Modem Control register (MCR) controls the output signals to the modem or data set (an external device acting as a modem).

Bit	Function
7 to 5	Reserved = 0
4	Loop
3	Out 2
2	Out 1
1	Request to Send
0	Data Terminal Ready

Figure 1-106. Modem Control Register

Bits 7-5 Reserved = 0.

Bit 4 This bit provides a loopback feature for diagnostic testing of the controller. When this bit is set to 1, the following events occur:

- SOUT is set to the active state.
- SIN is disconnected.
- The output of the Transmitter Shift register is "looped back" to the Receiver Shift register input.
- The four modem-control inputs (-DSR, -CTS, -RLSD, and -RI) are disconnected.
- The four modem-control outputs (-DTR, -RTS, -OUT 1, and -OUT 2) are internally connected to the four modem control inputs.

In the diagnostic mode, data sent is immediately received. This feature allows the microprocessor to verify the transmit- and receive-data paths of the controller.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational, as are the modem-control interrupts; however, the interrupts' sources are now the lower 4 bits of the Modem Control register instead of the four modem-control inputs. The interrupts are still controlled by the Interrupt Enable register.

The controller's interrupt system can be tested by writing to the lower 6 bits of the Line Status register and the lower 4 bits of the Modem Status register. Setting any of these bits to 1 generates the appropriate interrupt (if enabled). Resetting these interrupts is the same as for normal controller operation. To return to normal operation, the registers must be reprogrammed for normal operation, and then bit 4 of the MCR is cleared to 0.

Bit 3 This bit controls -OUT 2; when set to 1, it forces -OUT 2 active.

Bit 2 This bit controls -OUT 1; when set to 1, it forces -OUT 1 active.

Bit 1 This bit controls -RTS; when set to 1, it forces -RTS active.

Bit 0 This bit controls -DTR; when set to 1, it forces -DTR active.

Line Status Register, Hex 3FD: This register provides the microprocessor with status information about the data transfer.

Note: Bits 1 through 4 are error conditions that produce a receiver line-status interrupt whenever any of the corresponding conditions are detected.

Bit	Function
7	Reserved = 0
6	Tx Register Empty
5	Transmitter Holding Register Empty
4	Break Interrupt
3	Framing Error
2	Parity Error
1	Overrun Error

Figure 1-107 (Part 1 of 2). Line Status Register

Bit	Function
0	Data Ready

Figure 1-107 (Part 2 of 2). Line Status Register

Bit 7 Reserved = 0.

Bit 6 This bit is the transmitter empty indicator. It is set to 1 whenever the Transmitter Holding register and the Transmitter Shift register are both empty.

Bit 5 This bit is the Transmitter Holding register empty (THRE) indicator. It indicates the controller is ready to accept a new character for transmission. In addition, this bit causes the controller to issue an interrupt to the microprocessor when the THRE interrupt is enabled. The THRE bit is set to 1 when a character is transferred from the Transmitter Holding register into the Transmitter Shift register. It is cleared when the microprocessor loads the Transmitter Holding register.

Bit 4 This bit is the break interrupt indicator. It is set to 1 whenever the received data input is held in the spacing state (0) for longer than a full-word transmission time (that is, the total time of start bit + data bits + parity + stop bits).

Bit 3 This bit is the framing error indicator. It indicates the received character did not have a valid stop bit. Bit 3 is set to 1 whenever the stop bit is detected as a 0 (spacing level).

Bit 2 This bit is the parity error indicator and indicates the received data character does not have the correct even or odd parity. The parity error bit is set to 1 on a parity error and is cleared when the Line Status register is read.

Bit 1 This bit is the overrun error indicator. It indicates that data in the Receiver Buffer register was not read by the microprocessor before the next character was transferred into the register, thereby destroying the previous character. This indicator is cleared when the microprocessor reads the contents of the Line Status register.

Bit 0 This bit is the receiver data ready indicator. It is set to 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer register.

Modem Status Register, Hex 3FE: This register provides the current state of the control lines from the modem (or external device) to the microprocessor. In addition, 4 bits provide change information. They are set whenever a control input from the modem changes state; they are cleared when the microprocessor reads this register.

Note: Whenever bit 0, 1, 2, or 3 is set, a modem status interrupt is generated.

Bit	Function
7	Data Carrier Detect
6	Ring Indicate
5	Data Set Ready
4	Clear to Send
3	Delta Data Carrier Detect
2	Trailing Edge Ring Indicate
1	Delta Data Set Ready
0	Delta Clear to Send

Figure 1-108. Modem Status Register

Bit 7 When set to 1, this bit indicates -DCD is active. In the diagnostic mode, this bit is equivalent to OUT 2 of the Modem Control register.

Bit 6 When set to 1, this bit indicates -RI is active. In the diagnostic mode, this bit is equivalent to OUT 1 of the Mode Control register.

Bit 5 When set to 1, this bit indicates -DSR is active. In the diagnostic mode, this bit is equivalent to DTR of the Mode Control register.

Bit 4 When set to 1, this bit indicates -CTS is active. In the diagnostic mode, this bit is equivalent to RTS of the Mode Control register.

Bit 3 This bit is the delta data-carrier-detect indicator. It indicates -DCD to the chip has changed state.

Bit 2 This bit is the trailing-edge ring-indicate indicator. It indicates that -RI to the chip has changed from active to inactive.

Bit 1 This bit is the delta data-set-ready indicator. It indicates that -DSR to the chip has changed state.

Bit 0 This bit is the delta clear-to-send indicator. It indicates that -CTS to the chip has changed state.

Programmable Baud-Rate Generator

The controller has a programmable baud-rate generator that can divide the clock input, which is 1.84 MHz, by any divisor from 1 to 65,535 (2^{16} - 1). The output frequency of the baud-rate generator is the bps rate multiplied by 16. Two 8-bit latches store the divisor in a 16-bit binary format. These divisor latches are loaded during setup to ensure desired operation of the baud-rate generator. When either of the divisor latches is loaded, a 16-bit baud counter is immediately loaded. This prevents long counts on the first load.

The following is a sample program that sets the baud rate at 1200 with an 8-bit data word, 1 stop bit, and odd parity.

```
BEGIN   PROC    NEAR
        MOV     AL,8BH          ; Set port parameters
        MOV     AH,00H          ; Initialize COM1 port
        INT     14H             ; Serial port BIOS interrupt

        ENDP
```

Signal Descriptions

The following describes the function of controller I/O signals.

Input Signals

-Clear to Send (-CTS): This signal is an input from the modem. The status of this signal is reflected in bit 4 of the Modem Status register. Bit 0 of the same register indicates whether -CTS has changed state since the last reading.

-Data Set Ready (-DSR): When active, this signal indicates the modem or data set is ready to establish the communications link and transfer data with the controller. This signal is an input from the modem. Its status is reflected in bit 5 of the Modem Status register. Bit 1 of the same register indicates whether this signal has changed state since the last reading.

Note: Whenever bit 5 of the Modem Status register changes state, an interrupt is generated if the modem status interrupt is enabled.

-Data Carrier Detect (-DCD): When active, this signal indicates the modem or data set detected a data carrier. This signal is an input from the modem. Its status is reflected in bit 7 of the Modem Status register. Bit 3 of the same register indicates whether the signal has changed state since the last reading.

-Ring Indicate (-RI): When active, this signal indicates the modem or data set detected a telephone ringing signal. This signal is an input from the modem. Its status is reflected in bit 6 of the Modem Status register. Bit 2 of the same register indicates whether the signal has changed from active to inactive.

Note: Whenever bit 6 of the Modem Status register changes from 0 to 1, an interrupt is generated if the modem status interrupt is enabled.

Output Signals

-Data Terminal Ready (-DTR): When active, this signal informs the modem or data set that the controller is ready to communicate. This signal can be made active by setting bit 0 of the Modem Control register. It is inactive after a master reset operation.

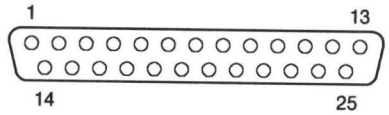
-Request to Send (-RTS): When active, this signal informs the modem or data set that the controller is ready to send data. It can be made active by setting bit 1 of the Modem Control register. This signal is inactive after a master reset operation.

-Output 1 (-OUT 1): User-designated output that can be made active by setting bit 2 of the Modem Control register. -OUT 1 is inactive after a master reset operation.

-Output 2 (-OUT 2): User-designated output that can be made active by setting bit 3 of the Modem Control register. -OUT 2 is inactive after a master reset operation. This signal controls interrupts to the system.

Connector

The following figure shows the pin assignments for the serial port in a communications environment when viewed from the rear of the system unit.



Pin	I/O	Signal Name	Pin	I/O	Signal Name
1	N/A	Not Connected	14	N/A	Not Connected
2	O	Transmit Data	15	N/A	Not Connected
3	I	Receive Data	16	N/A	Not Connected
4	O	-Request to Send	17	N/A	Not Connected
5	I	-Clear to Send	18	N/A	Not Connected
6	I	-Data Set Ready	19	N/A	Not Connected
7	N/A	Signal Ground	20	O	-Data Terminal Ready
8	I	-RLSD	21	N/A	Not Connected
9	N/A	Not Connected	22	I	-Ring Indicate
10	N/A	Not Connected	23	N/A	Not Connected
11	N/A	Tied to line 20	24	N/A	Not Connected
12	N/A	Not Connected	25	N/A	Not Connected
13	N/A	Not Connected			

Figure 1-109. Serial Port Connector

The following are the specifications for the serial interface.

Function	Condition
On	Spacing condition (binary 0, positive voltage).
Off	Marking condition (binary 1, negative voltage).

Voltage	Function
Above +15 Vdc	Invalid
+3 to +15 Vdc	On
-3 to +3 Vdc	Invalid
-3 to -15 Vdc	Off
Below -15 Vdc	Invalid

Figure 1-110. Serial Interface Specifications

Parallel Port

The parallel port allows the attachment of various devices that accept 8 bits of parallel data at standard TTL levels. The rear of the system unit has a 25-pin, D-shell connector. This port is addressed as parallel port 1.

To allow the parallel port to receive data from external devices, disable the output buffer by writing a 0 to bit 7 of the System Board Control register.

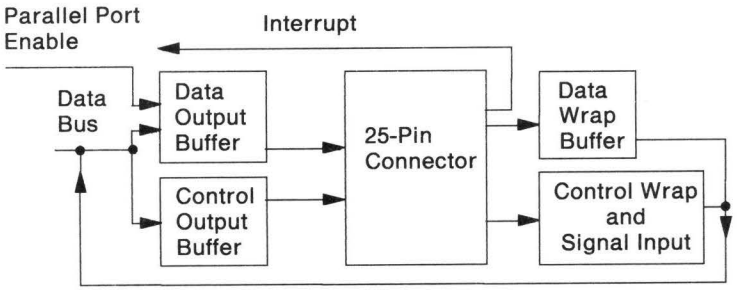


Figure 1-111. Parallel Port Block Diagram

Port Registers

The following describe the registers used for this port in a parallel printer application.

Data Latch, Hex 378: Writing to this address causes data to be stored in the device data buffer. Reading this address returns the contents of the buffer.

The output drivers for this data port will source 2.6 mA at a V_{OH} of 2.4 Vdc and sink 24 mA at a V_{OL} of .5 Vdc. Resistors (39 ohm) are in series with the output drivers.

Printer Controls, Hex 37A: Parallel port control signals are controlled through this address and can be read by the microprocessor. These signals are driven by open collector devices pulled up to +5 Vdc through 4.7K Ohm resistors. The output drivers can sink 16 mA at a V_{OL} of .4 Vdc.

Bit	Function
7	Reserved
6	Reserved
5	Reserved
4	IRQ Enable
3	Select Input (Slct In)
2	-Initialize (-Init)
1	Auto FD XT
0	Strobe

Figure 1-112. Printer Control Register

The following are bit definitions.

Bit 7-5 Reserved.

Bit 4 IRQ Enable—When set to 1, this bit allows an interrupt to occur when -ACK changes from active to inactive.

Bit 3 Slct In—When set to 1, this bit selects the device.

Bit 2 -Init—When cleared to 0, this bit resets the device (50-microsecond pulse, minimum).

Bit 1 Auto FD XT—When set to 1, this bit causes the device to line feed after a line is printed.

Bit 0 Strobe—An active pulse, minimum of 0.5 µs, clocks data into the device. Valid data must be present for a minimum of 0.5µs before and after the strobe pulse.

Printer Status - Address 379: Parallel port status is stored at this address to be read by the microprocessor. The following are bit definitions for this byte.

Bit	Function
7	-Busy
6	-Acknowledge (-ACK)
5	Page End (PE)
4	Selected (Slct)
3	-Error
2	Reserved
1	Reserved
0	Reserved

Figure 1-113. Printer Status Register

Bit 7 -Busy—This bit indicates the status of the device's 'busy' signal. When the signal is active, this bit is a 0, and the device cannot accept data. It is active during data entry, while the device is offline, or while in an error state.

Bit 6 -ACK—This bit represents the current state of the device's '-acknowledge' signal. A 0 means the device has received the character and is ready to accept another. Normally, this signal is active for approximately 5 µs before -BUSY goes active.

Bit 5 PE—When set to 1, this bit indicates a printer has detected the end of the paper.

Bit 4 Slct—When set to 1, this bit indicates the device is selected.

Bit 3 -Error—When cleared to 0, this bit indicates the device has encountered an error condition.

Bits 2-0 Reserved.

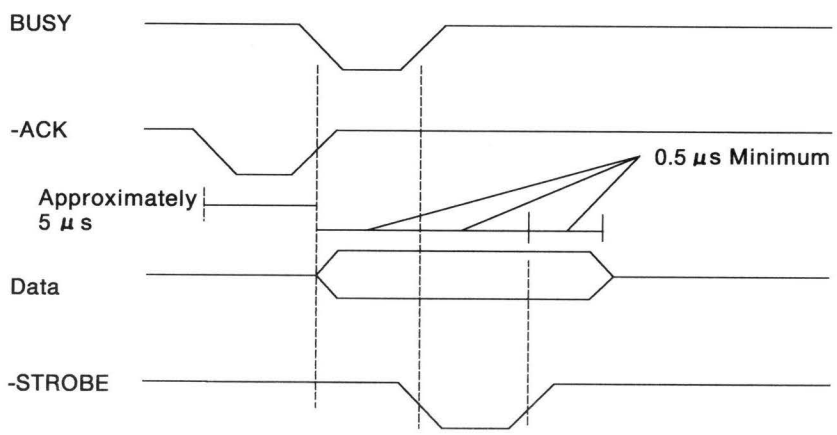
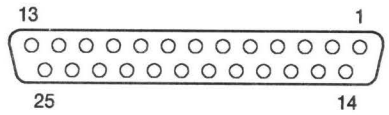


Figure 1-114. Parallel Port Signal Timing

Connector

The port has a 25-pin, D-shell connector at the rear of the system unit. The following figure shows the signals and their pin assignments. Typical printer input signals also are shown.



Pin	I/O	Signal Name	Pin	I/O	Signal Name
1	O	-Strobe	14	O	-Auto FD XT
2	I/O	D0	15	I	-Error
3	I/O	D1	16	O	-Init
4	I/O	D2	17	O	-Slct In
5	I/O	D3	18	N/A	Ground
6	I/O	D4	19	N/A	Ground
7	I/O	D5	20	N/A	Ground
8	I/O	D6	21	N/A	Ground
9	I/O	D7	22	N/A	Ground
10	I	-ACK	23	N/A	Ground
11	I	Busy	24	N/A	Ground
12	I	PE	25	N/A	Ground
13	I	Slct			

Figure 1-115. Parallel Port Connector

Beeper

The beeper and its control circuits and driver are on the system board. The beeper drive circuit is capable of approximately 1/2 watt of power. The control circuits allow the beeper to be driven three different ways:

1. A Direct Program Control register bit may be toggled to generate a pulse train.
2. The clock input to the timer can be modulated with a program-controlled I/O port bit.
3. The output from channel 2 of the timer may be programmed to generate a waveform to the beeper.

Channel 2	**(Tone generation for beeper)**
Gate 2	Controlled by I/O Port Bit 1
Clock In 2	1.9318 MHz OSC
Clock Out 2	Used to drive beeper

Figure 1-116. Beeper Tone Generation

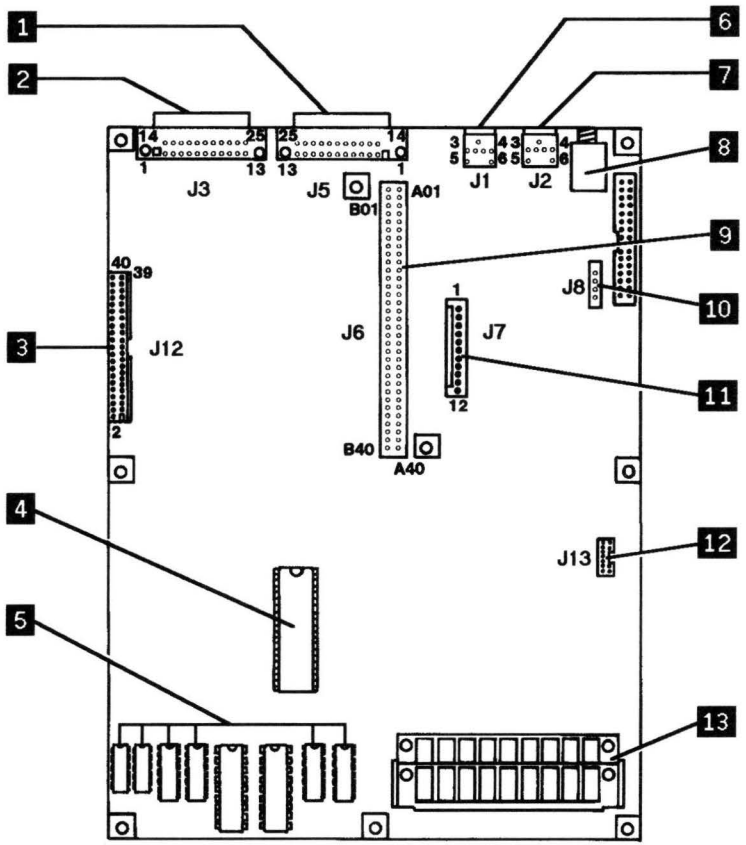
All three methods may be performed simultaneously. For more information, see "System Timer" and "I/O Ports" earlier in this section.

Earphone Connector

An earphone connector is provided at the rear of the system unit to allow the user to disable the internal beeper (speaker) and listen to the sound output through a set of earphones. The internal beeper is disabled whenever a plug is inserted into this connector.

The connector will accept any 1/4-inch diameter audio plug. A monophonic earphone with an impedance level of 15 to 35 ohms is recommended. Some earphones with an impedance level as high as 100 ohms may also be acceptable. The drive level is fixed so that the sound level will be directly related to the sensitivity of the earphones. Earphones with 1/8-inch plugs may be used with adapter plugs.

Connectors



Ref. #	**Description**	**Ref. #**	**Description**
1	Serial Connector	8	Earphone Connector
2	Parallel Connector	9	80-pin I/O Connector
3	Diskette Drive Connector	10	Fan Connector
4	Microprocessor	11	Power Supply Connector
5	Additional Memory (128K)	12	Display Connector
6	(Reserved)	13	Memory SIPs (512K)
7	Keyboard Connector		

Figure 1-117. System Board Connector Location

The pin assignments for the power supply connector, J7, are as follows. The pins are numbered 1 through 12 from the rear of the system.

Connector	Pin	Assignments
J7	1	Power Good
	2	Ground
	3	+12 Vdc
	4	-12 Vdc
	5	Ground
	6	Ground
	7	Ground
	8	Ground
	9	Open
	10	+5 Vdc
	11	+5 Vdc
	12	+5 Vdc

Figure 1-118. Power Supply Connector

The keyboard connector, J1 (on the system board), is a six-pin, 90-degree printed circuit board (PCB) mounting, miniature DIN connector. For pin numbering, see the "Keyboard" section. The pin assignments are as follows:

Pin	Assignments
1	Keyboard Data
2	Not Connected
3	Ground
4	+5 Vdc
5	Keyboard Clock
6	Not Connected

Figure 1-119. Keyboard Connector and Pointing Device

Specifications

Size

- Depth (storage): 223 millimeters (8.75 inches)
- Depth (operating): 410 millimeters (16.2 inches)
- Height: 493 millimeters (19.4 inches)
- Width: 417 millimeters (16.4 inches)

Weight

- 14.1 kilograms (31 pounds)

Power Cable

- Length: 1.8 meters (6 feet)

Environment

- Air Temperature
  - System On: 15°C to 32°C (60°F to 90°F)
  - System Off: 0.6°C to 60°C (33°F to 140°F)
- Humidity
  - System On: 8% to 80%
  - System Off: 5% to 80%
- Altitude
  - Maximum altitude: 2133.6 meters (7000 feet)

Noise Emission Values

- 5.3 bels — declared (upper limit) A-weighted, sound power level.
- 44 dB — mean value of A-weighted sound pressure levels at the operator's position for a random sample of machines.
- 40 dB — mean value of A-weighted sound pressure levels at the one-meter (bystander) positions for a random sample of machines.

Electrical

- Power: 120 VA
- Input
  - Nominal: 115 Vac
  - Minimum: 90 Vac
  - Maximum: 135 Vac

Section 2. Power Supply

Description

The system dc power supply is a single-phase, 60-watt, five-voltage level supply. It is internal to the system unit and supplies power for the system board, the LCD, the option adapters, the keyboard, and the battery pack. The supply provides 7.0 A of +5 Vdc, 1.6 A of +12 Vdc, 300 mA of -12 Vdc, and 70 mA of -22.3 Vdc to the system board. All power levels to the system board are monitored with undervoltage and overcurrent protection.

The power supply input is protected by an external fuse. If dc overcurrent or undervoltage conditions exist, the supply automatically shuts down until the condition is corrected.

The system board and storage devices take approximately 4 A of +5 Vdc, thus allowing approximately 3 A of +5 Vdc for the adapters in the system expansion slots. The +12 Vdc power level is designed to power the internal diskette drive. The +12 Vdc and -12 Vdc are used for powering the EIA drivers for the serial port. All three power levels are bussed across the four system expansion slots.

The power supply provides a constant current of 100 mA to the battery pack, which consists of seven Ni-Cad, 2.2 Ampere-hour cells. When properly charged, the battery pack provides 3 minutes of backup power for the keyboard and memory. The LCD, touch panel, and diskette drive are inoperative while the system is being powered by the battery pack. The battery pack becomes the active power source whenever primary power is lost while the power switch is in the ON position.

Input and Output Power

The nominal power requirements and voltages are listed in the following tables.

Nominal (Vac)	Minimum (Vac)	Maximum (Vac)	Maximum Current
115	90	137	2.0 A

Figure 2-1. Vac Input Requirements

Nominal (Vdc)	Load Current (A) Min	Load Current (A) Max	Regulation Tolerance	Ripple (mV p-p)
+5 Vdc	1.5	7.0	+5% to -3%	100
+12 Vdc	0.12	1.7	+5% to -3%	120
-12 Vdc	0.015	0.25	+10% to -8%	120

Figure 2-2. Vdc Output

Output Protection

The sense levels of the dc outputs are:

Output (Vdc)	Undervoltage Minimum (Vdc)
+5	+4.5
+12	+10.8
-12	-10.2
-22.3	-21.6

Figure 2-3. Output Protection

Power-Good Signal

The power supply provides a 'power good' signal to reset the system logic, to indicate proper operation of the power supply, and to give advance warning when the power is turned off.

The signal has a TTL-compatible active level of 2.4 to 5.25 Vdc during normal operation, or an inactive level of 0.0 to 0.4 Vdc. The signal is inactive if an undervoltage condition occurs, or during the power-on and power-off sequence. The 'power good' signal has a turn-on delay that is at least 100 ms but no greater than 500 ms. This line can sink 2 mA or source 100 μA.

-22.3 Vdc Enable

The power supply monitors a TTL-enable signal from the system board. When this signal is high (1) the -22.3 Vdc output is enabled; when this signal is low (0) the -22.3 Vdc output goes to +5 Vdc.

Connectors

The power supply attaches to the system board through a 12-pin connector, to the interface adapter through a 4-pin connector, and to the battery pack through a 2-pin connector. The pin numbering and signal assignments are shown below.

Connector	Pin	Assignments	Connects to:
P1	1	Power Good	J7 on system board
	2	Ground	
	3	+12 Vdc	
	4	-12 Vdc	
	5	Ground	
	6	Ground	
	7	Ground	
	8	Ground	
	9	Open	
	10	+5 Vdc	
	11	+5 Vdc	
	12	+5 Vdc	
P2	1	-22.3 Vdc	J5 on interface adapter
	2	-22.3 Vdc Enable	
	3	Polarizing	
	4	Ground	
P3	1	Battery +	Battery Pack
	2	Ground	

Figure 2-4. Power Supply Connectors

Section 3. Keyboard

Description

The Type 7690's integral keyboard is similar in layout and function to the keyboard provided with the IBM PC Convertible. It is a 78-key keyboard that uses Fn key combinations to generate the same keystrokes that are generated by the IBM enhanced 101-key keyboard.

When the keyboard is in Num Lock state, the numeric keypad is overlaid on the alphabetic keys. The Num Lock key (Fn + Num) toggles the overlay on or off. The overlay characters are printed in blue on the keytop of the keys that are affected. F11 and F12 are generated by pressing Fn + F1 and Fn + F2 respectively.

Keyboard Interface

The keyboard is internally connected to the interface adapter, where its output data is passed to the Type 7690's rear connector panel. The external keyboard cable then carries the keyboard signals to the system board.

The keyboard contains its own micro-controller, which is responsible for communications with the system board. This micro-controller decodes the keyboard signals and sends the appropriate scan codes to emulate the IBM enhanced keyboard (type "P" scan code set 1 and 2). The keyboard micro-controller also communicates with the bar code micro-controller to convert bar code signals into corresponding scan codes and merge them into the keyboard data stream.

At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol. The serial communications protocol between the keyboard micro-controller and the system board uses "mode 2" operation—the default mode for the IBM PS/2 Model 25 and Model 30. The serial protocol includes parity, and supports bidirectional communications.

Sequence Key-Code Scanning

The keyboard detects each key pressed, and sends each scan code in the sequence pressed. When not serviced by the system, the keyboard stores the scan codes in its buffer.

Keyboard Buffer

A 17-byte first-in-first-out (FIFO) buffer in the keyboard stores the scan codes until the system is ready to receive them.

A buffer-overrun condition occurs when more than 16 bytes are placed in the keyboard buffer. An overrun code replaces the 17th byte. If more keys are pressed while the buffer is full, the additional keystrokes are lost.

When the keyboard is allowed to send data, the bytes in the buffer are sent as in normal operation, and any additional keystrokes are sent. Response codes do not occupy a buffer position.

When keystrokes generate a multiple-byte sequence, the entire sequence must fit into the available buffer space; otherwise, a buffer-overrun condition occurs.

Make/Break Keys

With the exception of the Ctrl + Pause and Ctrl + Break key combinations, all keys are *make/break*. The make scan code of a key is sent to the keyboard controller when the key is pressed. When the key is released, its break scan code is sent.

Typematic Keys

With the exception of the Ctrl + Pause, Ctrl + Break, Num Lock, Scroll Lock, and Caps Lock keys; all keys are *typematic*. When a key is pressed and held down, the keyboard sends the make code for that key, delays 500 ms ±20%, and begins sending a make code for that key at a rate of 10.9 characters per second ±20%.

If two or more keys are held down, only the last key pressed repeats at the typematic rate. Typematic operation stops when the last key pressed is released, even if other keys are still held down. If a key is pressed and held down while keyboard transmission is inhibited, only

the first make code is stored in the buffer. This prevents buffer overflow as a result of typematic action.

Num Lock State Bit

The keyboard keeps track of the Num Lock state by means of a bit in memory, which is toggled every time the user types a Num Lock function. The Num Lock function is defined as Fn + Num Lock (Key 59 and key 78). When the system sends a 'Set/Reset Mode Indicators' command, the keyboard updates the state of the Num Lock bit to match the state of the LED as sent by the system. This synchronization occurs only at the time of the command. The Num Lock State bit is used for all subsequent make/break processing of the following keys:

- Cursor/Page Ctrl Keys
- Numeric Overlay Keys

Note: The make sequence associated with the current typematic key is unaffected.

Shift States

The keyboard generates two types of shift states: real and artificial. A *real* shift state is generated when the user presses a shift key. An *artificial* shift state is generated when the user presses any of the cursor or page control keys. The system manages four shift status bits (real/artificial and right/left shift) to maintain compatibility with the IBM enhanced keyboard. The real, right, and left shift status bits are updated only when a real shift code is placed in the keyboard buffer.

Artificial Shift Codes

The cursor keys and keys on the numeric keypad send the same base scan codes. The system differentiates the two by monitoring the Num Lock and shift status bits. To ensure that cursor keys are interpreted as cursor keys, artificial shift make/break scan codes are added to the base scan code. The keyboard restores the original shift states when all shift-dependent keys are released. Artificial shift codes are preceded by make code E0 hex.

During typematic operation, the artificial shift codes can overload the interface. Therefore, only the unique base scan code is repeated at the typematic rate; the artificial shift make/break codes are not repeated.

Key Roll Over

While a cursor or page control key is being held down, the system is in an artificial shift state. If another key is pressed while the system is in an artificial shift state, its function could be misinterpreted. When this condition occurs, the shift state of the keyboard and the system is checked, and appropriate shift make or break codes are sent so that no misinterpretation results. Similarly, if a shift key is released while a shift-dependent key is in typematic mode, the actual shift break code is sent, and a subsequent artificial shift make code is sent to place the system in the proper shift state for the typematic key.

Numeric Keypad Overlay

Certain alpha keys are used to create the numeric keypad scan codes when Num Lock is on, or when Fn key is pressed while Num Lock is off. The specific scan code of each key is based on the state of Num Lock or the Fn key at the time of the make/break. Also if the Num Lock flag bit is updated by a system command, the current state of the keypad is not updated until a subsequent break or make is sent. This may affect the operation of typematic keys.

Power-on Routine

Power-on Reset (POR) and the Basic Assurance Test (BAT) take place when power is first applied to the keyboard.

Power-on Reset

The keyboard logic generates a 'power-on reset' signal each time power is applied to the keyboard. POR occurs during 150 ms to 2.0 seconds from the time power is first applied to the keyboard.

Basic Assurance Test

The BAT consists of a keyboard processor test, a checksum of ROM, and a RAM test. During the BAT, activity on the 'clock' and 'data' lines is ignored. The BAT takes from 300 to 500 ms. This is in addition to the time required by the POR.

Upon satisfactory completion of the BAT, a completion code (Hex AA) is sent to the system, and keyboard scanning begins. If a BAT failure occurs, the keyboard sends an error code to the system. The keyboard is then disabled pending command input. Completion codes are sent 600 ms to 2.5 seconds after POR, and between 300 and 500 ms after a Reset command is acknowledged.

Following a successful POR, the system sets the line protocol to Scan Set 1.

Clock and Data Signals

The keyboard and system communicate over the 'clock' and 'data' lines. The source of each of these lines is an open-collector device on the keyboard that allows either the keyboard or the system to force a signal inactive. When no communication is occurring, the 'clock' line is active. The state of the 'data' line is held inactive by the keyboard.

An inactive signal is between 0.0 and +0.7 volts. A signal at the inactive level is a logical 0. An active signal is between +2.4 and +5.5 volts. A signal at the active level is a logical 1. Voltages are measured between a signal source and the dc ground.

The keyboard 'clock' line provides the clocking signals used to clock serial data from the keyboard. If the system forces the 'clock' line inactive, keyboard transmission is inhibited.

When the keyboard sends data to the system, it generates the 'clock' signal to time the data. The system can prevent the keyboard from sending data by forcing the 'clock' line inactive, or by holding the 'data' line inactive.

During the BAT, the keyboard allows the 'clock' and 'data' lines to go active.

Data Stream

Data transmissions from the keyboard consist of an 11-bit data stream sent serially over the 'data' line. The following table shows the data stream.

Bit	Function
1	Start Bit (Always 0)
2	Data Bit 0 (Least Significant)
3	Data Bit 1
4	Data Bit 2
5	Data Bit 3
6	Data Bit 4
7	Data Bit 5
8	Data Bit 6
9	Data Bit 7 (Most Significant)
10	Parity Bit (Odd Parity)
11	Stop Bit (Always 1)

Figure 3-1. Keyboard Data Stream

Data Output

When the keyboard is ready to send data, it first checks the status of the 'clock' and 'data' lines. When the 'clock' line is inactive (keyboard inhibit), the keyboard stores the data in its buffer. When the 'data' line is inactive and the 'clock' line is active (system request to send), the keyboard stores the data in its buffer and accepts the system input.

If both lines are active, the keyboard sends the data stream. During transmission, the keyboard monitors the 'clock' line. If the line goes inactive before the parity bit is sent, the keyboard stores the data in its buffer and returns the 'clock' and 'data' lines to active and then waits on the system. If the parity bit has been sent, the keyboard completes the transmission.

Commands

Commands can be sent between the keyboard and system board to perform various tasks. These commands are described as follows:

Inbound Commands sent from the keyboard controller on the system board to the keyboard micro-controller.

Outbound Commands sent from the keyboard micro-controller to the keyboard controller on the system board.

Inbound Commands

Reset (Hex FF): The system issues a RESET command to initiate a keyboard reset and an internal self-test. The keyboard acknowledges (ACK) receiving the command, but before executing the reset, the keyboard waits for the system to accept the ACK. If the system accepts the ACK, it pulses the clock and data lines with a 500-ms active pulse. The keyboard remains in a reset mode until the clock and data lines are pulsed or until another command is sent.

Note: The RESET command also resets the bar code micro-controller.

Resend (Hex FE): The system can send this command when it detects an error in any transmission from the keyboard. The RESEND command could be sent following an invalid keyboard transmission and before the system enables the interface allowing the next keyboard output. On receipt of RESEND, the keyboard will retransmit the previous output unless the previous output was a RESEND command. In this case, the keyboard will resend the last byte prior to the RESEND command.

Set Default (Hex F6): The SET DEFAULT command re-initializes the basic default conditions. Specifically the keyboard responds with an ACK, sets the default keytypes, typematic rate and delay, and continues scanning (only if it was previously enabled).

Default Disable (Hex F5): This command is similar to SET DEFAULT except that the keyboard discontinues scanning and awaits further instruction.

Enable (Hex F4): When ENABLE is issued, the keyboard responds with an ACK, erases its output buffer, clears the last typematic key, and starts scanning.

Set Rate/Delay (Hex F3): The system can issue this command along with a "parameter byte" to modify the typematic rate and delay. Bits 6 and 5 of the parameter byte hold the delay value while bits 4, 3, 2, 1, 0 (LSB) hold the rate value. Bit 7 (MSB) is always 0. The delay is equal to 1 plus the binary value of bit 6, 5 multiplied by 250 milliseconds.

The period (interval from one typematic output to the next) is determined by the following equation:

```
Period = ((8 + A) x 2 x B)/2 x 0.00834  Seconds  ± 20%

Where A = Binary value of bits 2, 1, 0 (LSB)
  and B = Binary value of bit 4, 3
```

The typematic rate (characters per second) is 1/period. The following table results:

Bits	Rate	Bits	Rate	Bits	Rate	Bits	Rate
43210	(CPS)	43210	(CPS)	43210	(CPS)	43210	(CPS)
00000	30.0	01000	15.0	10000	7.5	11000	3.7
00001	30.0	01001	13.3	10001	6.7	11001	3.3
00010	24.0	01010	12.0	10010	6.0	11010	3.0
00011	24.0	01011	10.9	10011	5.5	11011	2.7
00100	20.0	01100	10.0	10100	5.0	11100	2.5
00101	20.0	01101	9.2	10101	4.6	11101	2.3
00110	17.1	01110	8.6	10110	4.3	11110	2.1
00111	17.1	01111	8.0	10111	4.0	11111	2.0

The keyboard responds to the command with an ACK, discontinues scanning, and waits for the parameter byte. The keyboard then responds with an ACK, sets delay and rate, and continues scanning (if previously enabled). If another valid command is received in place of the parameter byte, the SET RATE/DELAY command is terminated without affecting the existing delay and rate, scanning is discontinued, and the new command is processed. A subsequent retransmission of the SET RATE/DELAY command and its accompanying parameter byte will restart scanning (if previously enabled).

Read ID (Hex F2): This command requests the keyboard ID information. The keyboard responds with an ACK, discontinues scanning, and outputs the two keyboard ID bytes. The first byte will be a Hex AB and the second byte a Hex 84. This identifies the keyboard as a "P" keyboard to the system.

Select Alternate Scan Codes (Hex F0): This command tells the keyboard to select one of two sets of scan codes. The keyboard acknowledges the command with an ACK, discontinues scanning, clears the keystroke buffer, and clears the last typematic key. The system then sends an option byte. Upon receiving this byte, the keyboard returns an ACK, and selects the scan code set identified by the option byte (shown below):

Value	Action
Hex 01	Tells the keyboard to use scan code Set 1.
Hex 02	Tells the keyboard to use scan code Set 2.
Hex 00	Queries the current scan code set.

After returning the ACK, the keyboard returns a byte indicating the scan code set currently in use. The keyboard then continues scanning (if previously enabled). If another valid command is received in place of the option byte, the SELECT ALTERNATE SCAN CODE command is terminated without changing the scan codes in use, however the keystroke buffer and the last typematic key will have been cleared.

Echo (Hex EE): The ECHO command is used as a diagnostic tool. Upon receipt of the command, the keyboard responds with Hex EE and continues scanning.

Set/Reset Mode Indicators (Hex ED): This command is used to activate or deactivate the NUM LOCK, CAPS LOCK, and/or SCROLL LOCK indicator lights. The indicators can be activated/deactivated in any combination, depending on the contents of the option byte. The bits are defined as follows:

Bit	Description
0	Scroll Lock Indicator
1	Num Lock Indicator
2	Caps Lock Indicator
3 - 7	Reserved — all bits must be 0.

To activate an indicator, the corresponding bit must be set (1). To deactivate an indicator, the corresponding bit must be reset (0). It is up to the system to remember the previous state of an indicator in case its setting does not change when a command sequence is issued to change the state of another indicator.

Bit 1 also sets the internal Num Lock state, which affects the scan codes that are sent for certain keys. Upon receipt of this command, the keyboard sets or resets the Num Lock indicator, and also sets the internal Num Lock state to match the Num Lock bit sent in the option byte.

The keyboard responds to the SET/RESET MODE INDICATORS command with an ACK, discontinues scanning, and waits for the option byte. The keyboard then responds to the option byte with an ACK, sets the indicators appropriately, and continues scanning (if previously enabled). If another command is received in place of the option byte, the SET/RESET MODE INDICATORS command is terminated without affecting the existing indicator states, and the new command is processed.

Note: Immediately after power-on, or the 'Reset' command, the indicators default to the "off" state.

Unsupported Commands

The following system commands will be acknowledged by the keyboard but will not be supported.

- SET KEY TYPES
- SET ALL KEY TYPES

Outbound Commands

The following describes the commands that the keyboard sends to the system, and shows their hexadecimal values.

Command	Hex Value
BAT Completion Code	AA
BAT Failure Code	FC
Key Detection Error/Overrun	FF
Resend	FE
ACK	FA
Break Code Prefix	F0
Echo	EE
Keyboard ID	AB84

Figure 3-2. Commands from the Keyboard

BAT Completion Code (Hex AA): Following satisfactory completion of the BAT, the keyboard sends hex AA. Any other code indicates a failure of the keyboard.

BAT Failure Code (Hex FC): If a BAT failure occurs, the keyboard sends this code, discontinues scanning, and waits for a system response or reset.

Key Detection Error (Hex FF): The keyboard sends a key detection error character (hex FF) if conditions in the keyboard make it impossible to identify a switch closure.

Overrun (Hex FF): An overrun character (hex FF) is placed in the keyboard buffer and replaces the last code when the buffer capacity has been exceeded. The code is sent to the system when it reaches the top of the buffer queue.

Resend (Hex FE): The keyboard issues a RESEND command following the receipt of an invalid input or any input with incorrect parity. If the system has not sent anything to the keyboard, no response is required.

ACK (Hex FA): The keyboard issues an ACK response to any valid input other than ECHO or RESEND. If the keyboard is interrupted during the output of an ACK, it will abort the ACK and process the new command.

Break Code Prefix (Hex F0): This code will be transmitted as the first of 2-byte sequence to indicate the 'break' of a key. The keyboard will begin transmission of the second byte typically within 2 msec after the interface is enabled following the output of 'break code' prefix.

Note: Subsequent key transmissions prior to the interface enable may result in delayed transmission - 10 msec worst case.

Echo (Hex EE): The ECHO command is sent in response to and ECHO command received from the system board.

Keyboard ID (Hex AB84): The keyboard ID consists of two bytes, HEX AB84. The keyboard responds to the READ ID with an ACK, discontinues scanning and outputs the two ID bytes, low byte (Hex AB) first, then high byte (Hex 84). Following the output of the ID bytes, the keyboard commences scanning.

Scan Codes

Since the keyboard is a non-detachable, integral part of the Type 7690, it only supports "P" scan code sets 1 and 2 — the *Extended PC Scan Codes*. Each key is assigned a make and break scan code and, in some cases, an extra set of codes to generate artificial shift states in the system. The typematic scan codes are identical to the make scan code for each key.

Key Number Assignments

The following figure illustrates the key numbers assigned to the Type 7690 keyboard. References are made to these key numbers in the scan code charts on the following pages.

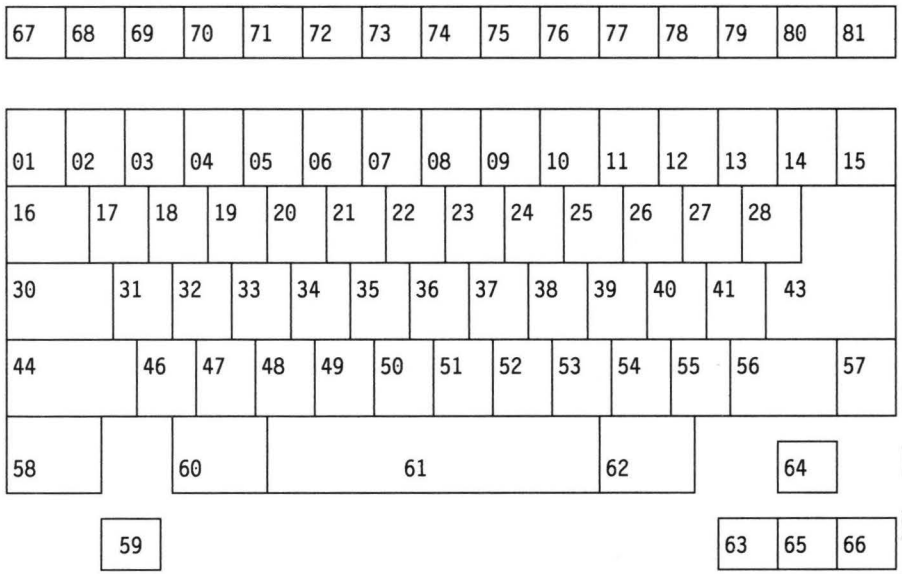


Figure 3-3. Key Number Assignments

The following keys send the codes shown, regardless of the shifted states of the keyboard. Refer to Figure 3-3 on page 3-14 to locate keys by their assigned numbers as listed in the following charts.

Key #	Legend	Make/Break Scan Set 1	Make/Break Scan Set 2
01	`	29/A9	0E/F0 0E
02	1	02/82	16/F0 16
03	2	03/83	1E/F0 1E
04	3	04/84	26/F0 26
05	4	05/85	25/F0 25
06	5	06/86	2E/F0 2E
07	6	07/87	36/F0 36
11	0	0B/8B	45/F0 45
12	-	0C/8C	4E/F0 4E
13	=	0D/8D	55/F0 55
14	\	2B/AB	5D/F0 5D
15	Bksp	0E/8E	66/F0 66
16	Tab	0F/8F	0D/F0 0D
17	Q	10/90	15/F0 15
18	W	11/91	1D/F0 1D
19	E	12/92	24/F0 24
20	R	13/93	2D F0 2D
21	T	14/94	2C/F0 2C
22	Y	15/95	35/F0 35
26	P	19/99	4D/F0 4D
27	[	1A/9A	54/F0 54
28	]	1B/9B	5B/F0 5B
30	Caps Lock	3A/BA	58/F0 58
31	A	1E/9E	1C/F0 1C
32	S	1F/9F	1B/F0 1B
33	D	20/A0	23/F0 23
34	F	21/A1	2B/F0 2B
35	G	22/A2	34/F0 34

Figure 3-4. Scan Codes (Part 1 of 11)

Key #	Legend	Make/Break Scan Set 1	Make/Break Scan Set 2
36	H	23/A3	33/F0 33
40	;	27/A7	4C/F0 4C
41	'	28/A8	52/F0 52
44	Shift	2A/AA	12/F0 12
46	Z	2C/AC	1A/F0 1A
47	X	2D/AD	22/F0 22
48	C	2E/AE	21/F0 21
49	V	2F/AF	2A/F0 2A
50	B	30/B0	32/F0 32
51	N	31/B1	31/F0 31
53	,	33/B3	41/F0 41
56	Shift	36/B6	59/F0 59
60	Alt	38/B8	11/F0 11
61	Space	39/B9	29/F0 29
62	Alt	E0 38/E0 B8	E0 11/E0 F011
70	F3	3D/BD	04/F0 04
71	F4	3E/BE	0C/F0 0C
72	F5	3F/BF	03/F0 03
73	F6	40/C0	0B/F0 0B
74	F7	41/C1	83/F0 83
75	F8	42/C2	0A/F0 0A
76	F9	43/C3	01/F0 01
77	F10	44/C4	09/F0 09
79	Scroll Lock	46/C6	7E/F0 7E

Figure 3-5. Scan Codes (Part 2 of 11)

The remaining keys send a series of codes depending on the state of various shift keys (Ctrl, Alt, and Shift), the state of Num Lock (On or Off), and the state of the Function (Fn) key. Because the base scan code is identical to that of another key, an extra code (hex E0) is added to the base code to make it unique.

The following chart shows the make/break codes associated with the Function (Fn) keys for both scan sets.

Key #	Legend	Base Case Scan Code Make/Break Scan Set 1	Fn Case Scan Code Make/Break Scan Set 1	Base Case Scan Code Make/Break Scan Set 2	Fn Case Scan Code Make/Break Scan Set 2
58	Ctrl	1D/9D		14/F0 14	
	R-Ctrl		E0 1D/E0 9D		E0 14/E0 F0 14
67	Esc	01/81		76/F0 76	
	Sys Req		54/D4		84/F0 84
68	F1	3B/BB		05/F0 05	
	F11		57/D7		78/F0 78
69	F2	3C/BC		06/F0 06	
	F12		58/D8		07/F0 07

Figure 3-6. Scan Codes (Part 3 of 11)

The following charts show the make/break codes that are generated by the Num Lock key, and Num Lock key combinations. Separate charts are provided for scan code sets 1 and 2.

Key #	Legend Scan Set 1	Base Case Make/Break	Fn Case Make/Break (* Note)	Ctrl Down Make
78	NumLock	00	45/C5	
	Pause			E1 1D 45 E1 9D C5

Figure 3-7. Scan Codes (Part 4 of 11)

Key #	Legend Scan Set 2	Base Case Make/Break	Fn Case Make/Break (* Note)	Ctrl Down Make
78	NumLock	00	77/F0 77	
	Pause			E1 14 77 E1 F0 14 F0 77

Figure 3-8. Scan Codes (Part 5 of 11)

*** Note:** In the above charts, the Fn + Num Lock combination sends the required scan code to enable the system Num Lock state.

The following charts show the make/break scan codes associated with the Num Lock key combinations. Separate charts are provided for scan code sets 1 and 2.

Key #	Legend Scan Set 1	Base Case	Fn	NumLock + Shift	NumLock Mode Fn + Shift
		Make/Break	Make/Break	Make/Break	Make/Break
08	& 7	08/88	2A 4B AA/ 2A CB AA	AA 4B 2A/ AA CB 2A	47/C7
09	* 8	09/89	2A 48 AA/ 2A C8 AA	AA 48 2A/ AA C8 2A	48/C8
10	(9	0A/8A	2A 49 AA/ 2A C9 AA	AA 49 2A/ AA C9 2A	49/C9
12	_ -	0C/8C	4A/CA	4A/CA	4A/CA
13	+ =	37/B7	4E/CE	4E/CE	4E/CE
23	U 4	16/96	2A 4B AA/ 2A CB AA	AA 4B 2A/ AA CB 2A	4B/CB
24	I 5	17/97	2A 4C AA/ 2A CC AA	AA 4C 2A/ AA CC 2A	4C/CC
25	O 6	18/98	2A 4D AA/ 2A CD AA	AA 4D 2A/ AA CD 2A	4D/CD
37	J 1	24/A4	2A 4F AA/ 2A CF AA	AA 4F 2A/ AA CF 2A	4F/CF
38	K 2	25/A5	2A 50 AA/ 2A D0 AA	AA 50 2A/ AA D0 2A	50/D0
39	L 3	26/A6	2A 51 AA/ 2A D1 AA	AA 51 2A/ AA D1 2A	51/D1
43	Enter	1C/9C	E0 1C/E0 9C	1C/9C	1C/9C
52	M 0	32/B2	2A 52 AA/ 2A D2 AA	AA 52 2A/ AA D2 2A	52/D2
54	> .	34/B4	2A 53 AA/ 2A D3 AA	AA 53 2A/ AA D3 2A	53/D3
55	? /	35/B5	E0 35/E0 B5	E0 AA E0 35/ E0 2A E0 B5	E0 35/E0 B5
57	* PrtSc	37/B7	37/B7	E0 37/E0 B7	37/B7

Figure 3-9. Scan Codes (Part 6 of 11)

Key #	Legend Scan Set 2	Base Case	Fn	NumLock + Shift	NumLock Mode Fn + Shift
		Make/Break	Make/Break	Make/Break	Make/Break
08	& 7	3D/F0 3D	12 6C F0 12/ 12 F0 6C F0 12	F0 12 6C 12/ F0 12 F0 6C 12	6C/F0 6C
09	* 8	3E/F0 3E	12 75 F0 12/ 12 F0 75 F0 12	F0 12 75 12/ F0 12 F0 75 12	75/F0 75
10	(9	46/F0 46	12 7D F0 12/ 12 F0 7D F0 12	F0 12 7D 12/ F0 12 F0 7D 12	7D/F0 7D
12	_ -	4E/F0 4E	7B/F0 7B	7B/F0 7B	7B/F0 7B
13	+ =	55/F0 55	69/F0 69	69/F0 69	69/F0 69
23	U 4	3C/F0 3C	12 6B F0 12/ 12 F0 6B F0 12	F0 12 6B 12/ F0 12 F0 6B 12	6B/F0 6B
24	I 5	43/F0 43	12 73 F0 12/ 12 F0 73 F0 12	F0 12 73 12/ F0 12 F0 73 12	73/F0 73
25	O 6	44/F0 44	12 74 F0 12/ 12 F0 74 F0 12	F0 12 74 12/ F0 12 F0 74 12	74/F0 74
37	J 1	3B/F0 3B	12 69 F0 12/ 12 F0 69 F0 12	F0 12 69 12/ F0 12 F0 69 12	69/F0 69
38	K 2	42/F0 42	12 72 F0 12/ 12 F0 72 F0 12	F0 12 72 12/ F0 12 F0 72 12	72/F0 72
39	L 3	4B/F0 4B	12 7A F0 12/ 12 F0 7A F0 12	F0 12 7A 12/ F0 12 F0 7A 12	7A/F0 7A
43	Enter	5A/F0 5A	E0 5A/E0 F0 5A	5A/F0 5A	5A/F0 5A
52	M 0	3A/F0 3A	12 70 F0 12/ 12 F0 70 F0 12	F0 12 70 12/ F0 12 F0 70 12	70 /F0 70
54	> .	49/F0 49	12 71 F0 12/ 12 F0 71 F0 12	F0 12 71 12/ F0 12 F0 71 12	71/F0 71
55	? /	4A/F0 4A	E0 4A/E0 F0 4A	E0 F0 12 E0 4A/ E0 F0 4A E0 12	E0 4A/E0 F0 4A
57	* PrtSc	7C/F0 7C	7C/F0 7C	E0 7C/E0 F0 7C	7C/F0 7C

Figure 3-10. Scan Codes (Part 7 of 11)

The following charts show the scan codes associated with all shift modes of the cursor keys. A separate chart is provided for scan code sets 1 and 2.

Key #	Legend Scan Set 1	Shift Mode	Make/Break
63	Left	Base case	E0 4B/E0 CB
	Home	Fn case	E0 47/E0 C7
		NumLock + Shift	E0 4B/E0 CB
		NumLock + Shift + Fn	E0 47/E0 C7
		Shift	E0 AA E0 4B/E0 CB E0 2A *
		Fn + Shift	E0 AA E0 47/E0 C7 E0 2A *
		NumLock	E0 2A E0 4B/E0 CB E0 AA *
		NumLock + Fn	E0 2A E0 47/E0 C7 E0 AA *
64	Up	Base case	E0 48/E0 C8
	PgUp	Fn case	E0 49/E0 C9
		NumLock + Shift	E0 48/E0 C8
		NumLock + Shift + Fn	E0 49/E0 C9
		Shift	E0 AA E0 48/E0 C8 E0 2A *
		Fn + Shift	E0 AA E0 49/E0 C9 E0 2A *
		NumLock	E0 2A E0 48/E0 C8 E0 AA *
		NumLock + Fn	E0 2A E0 49/E0 C9 E0 AA *
65	Down	Base case	E0 50/E0 D0
	PgDn	Fn case	E0 51/E0 D1
		NumLock + Shift	E0 50/E0 D0
		NumLock + Shift + Fn	E0 51/E0 D1
		Shift	E0 AA E0 50/E0 D0 E0 2A *
		Fn + Shift	E0 AA E0 51/E0 D1 E0 2A *
		NumLock	E0 2A E0 50/E0 D0 E0 AA *
		NumLock + Fn	E0 2A E0 51/E0 D1 E0 AA *

Figure 3-11. Scan Codes (Part 8 of 11)

* **Note:** If the left shift key (44) is held down, the AA - 2A shift break and make codes envelop the other scan codes. If the right shift key is held down, B6 - 36 envelops the other scan codes. If both shift keys are held down, both sets of make/break codes envelop the other scan codes.

Key #	Legend Scan Set 1	Shift Mode	Make/Break
66	Right	Base case	E0 4D/E0 CD
	End	Fn case	E0 4F/E0 CF
		NumLock + Shift	E0 4D/E0 CD
		NumLock + Shift + Fn	E0 4F/E0 CF
		Shift	E0 AA E0 4D/E0 CD E0 2A *
		Fn + Shift	E0 AA E0 4F/E0 CF E0 2A *
		NumLock	E0 2A E0 4D/E0 CD E0 AA *
		NumLock + Fn	E0 2A E0 4F/E0 CF E0 AA *
80	Ins	Base case	E0 52/E0 D2
		NumLock + Shift	E0 52/E0 D2
		Shift	E0 AA E0 52/E0 D2 E0 2A *
		NumLock	E0 2A E0 52/E0 D2 E0 AA *
81	Del	Base case	E0 53/E0 D3
		NumLock + Shift	E0 53/E0 D3
		Shift	E0 AA E0 53/E0 D3 E0 2A *
		NumLock	E0 2A E0 53/E0 D3 E0 AA *

Figure 3-12. Scan Codes (Part 9 of 11)

* **Note:** If the left shift key (44) is held down, the AA - 2A shift break and make codes envelop the other scan codes. If the right shift key is held down, B6 - 36 envelops the other scan codes. If both shift keys are held down, both sets of make/break codes envelop the other scan codes.

Key #	Legend Scan Set 2	Shift Mode	Make/Break
63	Left	Base case	E0 6B/E0 F0 6B
	Home	Fn case	E0 6C/E0 F0 6C
		NumLock + Shift	E0 6B/E0 F0 6B
		NumLock + Shift + Fn	E0 6C/E0 F0 6C
		Shift	E0 F0 12 E0 6B/E0 F0 6B E0 12 *
		Fn + Shift	E0 F0 12 E0 6C/E0 F0 6C E0 12 *
		NumLock	E0 12 E0 6B/E0 F0 6B E0 F0 12 *
		NumLock + Fn	E0 12 E0 6C/E0 F0 6C E0 F0 12 *
64	Up	Base case	E0 75/E0 F0 75
	PgUp	Fn case	E0 7D/E0 F0 7D
		NumLock + Shift	E0 75/E0 F0 75
		NumLock + Shift + Fn	E0 7D/E0 F0 7D
		Shift	E0 F0 12 E0 75/E0 F0 75 E0 12 *
		Fn + Shift	E0 F0 12 E0 7D/E0 F0 7D E0 12 *
		NumLock	E0 12 E0 75/E0 F0 75 E0 F0 12 *
		NumLock + Fn	E0 12 E0 7D/E0 F0 7D E0 F0 12 *
65	Down	Base case	E0 72/E0 F0 72
	PgDn	Fn case	E0 7A/E0 F0 7A
		NumLock + Shift	E0 72/E0 F0 72
		NumLock + Shift + Fn	E0 7A/E0 F0 7A
		Shift	E0 F0 12 E0 72/E0 F0 72 E0 12 *
		Fn + Shift	E0 F0 12 E0 7A/E0 F0 7A E0 12 *
		NumLock	E0 12 E0 72/E0 F0 72 E0 F0 12 *
		NumLock + Fn	E0 12 E0 7A/E0 F0 7A E0 F0 12 *

Figure 3-13. Scan Codes (Part 10 of 11)

* **Note:** If the left shift key (44) is held down, the F0 12 - 12 shift break and make codes envelop the other scan codes. If the right shift key is held down, F0 59 - 59 envelops the other scan codes. If both shift keys are held down, both sets of make/break codes envelop the other scan codes.

Key #	Legend Scan Set 2	Shift Mode	Make/Break
66	Right	Base case	E0 74/E0 F0 74
	End	Fn case	E0 69/E0 F0 69
		NumLock + Shift	E0 74/E0 F0 74
		NumLock + Shift + Fn	E0 69/E0 F0 69
		Shift	E0 F0 12 E0 74/E0 F0 74 E0 12 *
		Fn + Shift	E0 F0 12 E0 69/E0 F0 69 E0 12 *
		NumLock	E0 12 E0 74/E0 F0 74 E0 F0 12 *
		NumLock + Fn	E0 12 E0 69/E0 F0 69 E0 F0 12 *
80	Ins	Base case	E0 70/E0 F0 70
		NumLock + Shift	E0 70/E0 F0 70
		Shift	E0 F0 12 E0 70/E0 F0 70 E0 12 *
		NumLock	E0 12 E0 70/E0 F0 70 E0 F0 12 *
81	Del	Base case	E0 71/E0 F0 71
		NumLock + Shift	E0 71/E0 F0 71
		Shift	E0 F0 12 E0 71/E0 F0 71 E0 12 *
		NumLock	E0 12 E0 71/E0 F0 71 F0 12 *

Figure 3-14. Scan Codes (Part 11 of 11)

* **Note:** If the left shift key (44) is held down, the F0 12 - 12 shift break and make codes envelop the other scan codes. If the right shift key is held down, F0 59 - 59 envelops the other scan codes. If both shift keys are held down, both sets of make/break codes envelop the other scan codes.

Bar Code Characters

The following chart shows how the keyboard micro-controller translates ASCII bar code data to keyboard scan codes.

Bar Code Character	Emulated Key No.	Make/Break Scan Set 1	Make/Break Scan Set 2
`	01	29/A9	0E/F0 0E
~		(Shift)	(Shift)
1	02	02/82	16/F0 16
!		(Shift)	(Shift)
2	03	03/83	1E/F0 1E
@		(Shift)	(Shift)
3	04	04/84	26/F0 26
#		(Shift)	(Shift)
4	05	05/85	25/F0 25
$		(Shift)	(Shift)
5	06	06/86	2E/F0 2E
%		(Shift)	(Shift)
6	07	07/87	36/F0 36
^		(Shift)	(Shift)
7	08	08/88	3D/F0 3D
&		(Shift)	(Shift)
8	09	09/89	3E/F0 3E
*		(Shift)	(Shift)
9	10	0A/8A	46/F0 46
(		(Shift)	(Shift)
0	11	0B/8B	45/F0 45
)		(Shift)	(Shift)
-	12	0C/8C	4E/F0 4E
_		(Shift)	(Shift)
=	13	0D/8D	55/F0 55
+		(Shift)	(Shift)
\	14	2B/AB	5D/F0 5D
|		(Shift)	(Shift)
Q	17	10/90	15/F0 15
W	18	11/91	1D/F0 1D
E	19	12/92	24/F0 24
R	20	13/93	2D F0 2D
T	21	14/94	2C/F0 2C
Y	22	15/95	35/F0 35
U	23	16/96	3C/F0 3C
I	24	17/97	43/F0 43
O	25	18/89	44/F0 44
P	26	19/99	4D/F0 4D
[	27	1A/9A	54/F0 54
{		(Shift)	(Shift)

Figure 3-15 (Part 1 of 2). Bar Code Character Scan Code Emulation

Bar Code Character	Emulated Key No.	Make/Break Scan Set 1	Make/Break Scan Set 2
]	28	1B/9B	5B/F0 5B
}		(Shift)	(Shift)
A	31	1E/9E	1C/F0 1C
S	32	1F/9F	1B/F0 1B
D	33	20/A0	23/F0 23
F	34	21/A1	2B/F0 2B
G	35	22/A2	34/F0 34
H	36	23/A3	33/F0 33
J	37	24/A4	3B/F0 3B
K	38	25/A5	42/F0 42
L	39	26/A6	4B/F0 4B
;	40	27/A7	4C/F0 4C
:		(Shift)	(Shift)
'	41	28/A8	52/F0 52
"		(Shift)	(Shift)
Z	46	2C/AC	1A/F0 1A
X	47	2D/AD	22/F0 22
C	48	2E/AE	21/F0 21
V	49	2F/AF	2A/F0 2A
B	50	30/B0	32/F0 32
N	51	31/B1	31/F0 31
M	52	32/B2	3A/F0 3A
,	53	33/B3	41/F0 41
<		(Shift)	(Shift)
.	54	34/B4	49/F0 49
>		(Shift)	(Shift)
/	55	35/B5	4A/F0 4A
?		(Shift)	(Shift)
SPACE	61	39/B9	29/F0 29

Figure 3-15 (Part 2 of 2). Bar Code Character Scan Code Emulation

Note: In the chart above, the (Shift) notation indicates that the base scan code is enveloped with the shift make/break codes to produce the desired character. For example, the keyboard sends a 02/82 (scan set 1) for the '1' character. When the bar code character is '!' the keyboard sends 2A 02/82 AA. If Caps Lock mode is active, or if the user is holding down a shift key when a lower-case bar code character is received, the keyboard compensates so that the system does not receive the shifted scan code.

Keyboard Layout

Only the U.S. keyboard layout is supported by the Type 7690, as shown below.

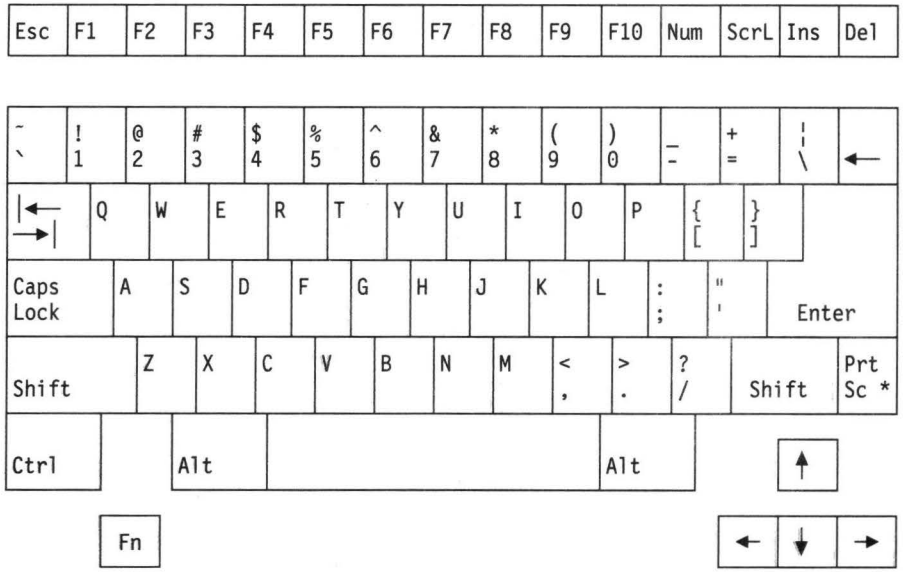


Figure 3-16. Keyboard Layout

Certain key combinations are required to generate standard "P" keyboard scan codes. These key combinations are shown in the following charts.

This Keyboard	"P" Keyboard Function	This Keyboard	"P" Keyboard Function
Ctrl + Num Lock	Pause	Fn + Esc	SysRq
Ctrl + Scroll Lock	Break	Fn + F1	F11
Fn + ←	Home	Fn + F2	F12
Fn + ↑	PgUp	Ctrl	Left Ctrl
Fn + →	End	Fn + Ctrl	Right Ctrl
Fn + ↓	PgDn	Fn + Enter	Numeric keypad Enter

Figure 3-17. Key Combinations Producing "P" Keyboard Functions

The following charts list the keys that generate the "P" keyboard's Keypad characters when pressed with Num Lock on, or when pressed in conjunction with the Fn key.

Key	Num Lock ON	Num Lock OFF
M	Keypad 0	M
J	Keypad 1	J
K	Keypad 2	K
L	Keypad 3	L
U	Keypad 4	U
I	Keypad 5	I
O	Keypad 6	O
7	Keypad 7	7
8	Keypad 8	8
9	Keypad 9	9
-	Keypad -	-
=	Keypad +	=
.	Keypad .	.
/?	Keypad /	/?
* PrtSc	Keypad * PrtSc	* PrtSc

Figure 3-18. Num Locked Keys Producing "P" Keypad Characters

Key	Num Lock OFF
Fn + M	Keypad 0
Fn + J	Keypad 1
Fn + K	Keypad 2
Fn + L	Keypad 3
Fn + U	Keypad 4
Fn + I	Keypad 5
Fn + O	Keypad 6
Fn + 7	Keypad 7
Fn + 8	Keypad 8
Fn + 9	Keypad 9
Fn + -	Keypad -
Fn + =	Keypad +
Fn + .	Keypad .
Fn + / ?	Keypad /
Fn + * PrtSc	Keypad * PrtSc

Figure 3-19. Fn-key Combinations Producing "P" Keypad Characters

Section 4. Bar Code Feature

Description

The Type 7690 supports a bar code wand/card reader feature. The bar code wand is connected to a bar code controller which in turn is connected to the keyboard micro-controller.

The bar code controller is responsible for the actual bar code scanning and the conversion of bar code data to ASCII. The ASCII bar code data is then sent to the keyboard micro-controller where it is converted to the appropriate scan codes, serialized, and merged into the keyboard data stream for transmission to the system board. See "Bar Code Characters" on page 3-24 for scan code information.

A single carriage return (0Dh) follows the bar code data by default. To distinguish bar code data from normal keyboard data, application programs can request that a header and/or trailer be appended to the bar code data. Refer to "Specifying Special Bar Code Data Formats" on page 4-12 for information about bar code headers and trailers.

Electrical Characteristics

The bar code reader uses a single TTL input bit. A high level (1) indicates a dark bar while a low level (0) indicates a light bar. All bar width detection and decode algorithms are performed by the bar code controller.

The bar code wand pin assignments are shown below.

Pin #	Assignment
1	Scanner Synchronization
2	Bar Code Data
3	+LED Read OK
4	Scanner Type Sense (Wand=Float)
5	Scanner Trigger
6	5V Scanner Enable
7	+5V
8	Ground

Figure 4-1. Bar Code Wand Pin Assignments

Note: A presence test is performed during the POST to determine if the bar code controller is present.

Bar Code Controller

The bar code controller is a 40 pin IC combined with a 1Kx8 RAM. It can decode several types of bar codes and output the decoded data over a full-duplex serial port to the keyboard micro-controller.

The narrowest bar must generate a pulse width greater than 150 µSec and the widest bar must generate a pulse width less than 70 MSec. The bar code must have a minimum contrast ratio of 45% and be scanned with a tilt angle between vertical and 45 degrees.

Bar Code Types

The bar code controller can automatically recognize and correctly decode the following bar codes:

- Code 39 (ANSI MH10.8M-1983, MIL-STD-1189)
- Interleaved 2 of 5 Code (ANSI MH10.8M-1983)
- UPC/EAN/JAN Codes
- Standard Codabar (NM7)
- Code 128

Any of these codes can be scanned bidirectionally. Supplemental digits can also be read but the code must be scanned in the forward direction; the supplemental digits must be scanned last. The following supplemental codes can be supported:

- UPC-A 2-Digit Supplemental
- UPC-E 2-Digit Supplemental
- EAN/JAN 13 2-Digit Supplemental
- EAN/JAN 8 2-Digit Supplemental
- UPC-A 5-Digit Supplemental
- UPC-E 5-Digit Supplemental
- EAN/JAN 13 5-Digit Supplemental
- EAN/JAN 8 5-Digit Supplemental

The system does not support supplemental codes by default. The application program must request this option using the protocol described in "Programming Considerations" on page 4-5.

Communications with the Keyboard Micro-controller

The serial link between the bar code controller and the keyboard micro-controller has the following characteristics:

- 2400 BAUD
- 1 Stop Bit
- Odd Parity

The bar code controller pinout information is shown below.

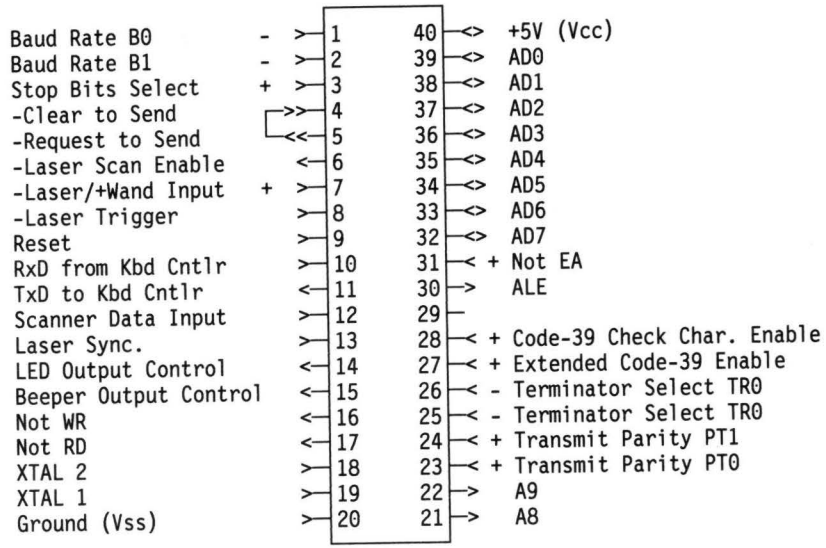


Figure 4-2. Bar Code Controller Pinout

Programming Considerations

Application programs can change the default characteristics of the bar code feature. For example, an application program can:

- Disable support for specific bar code types
- Enable check character verification
- Change decoding options:
  - enable full ASCII conversion for Code 39
  - disable start/stop characters for Codabar
  - enable 2- or 5-digit supplements for UPC/EAN/JAN
- Enable label length checking for Interleaved 2 of 5
- Specify special bar code data formats
- Enable Single Read Mode
- Request status report from controller
- Reset bar code controller
- Provide positive operator feedback for good reads

These programmable bar code functions are supported by the bar code controller and can be enabled or disabled through a special extension of the Interrupt 10h BIOS call, as follows:

Programmable Bar Code Functions

Interrupt 10h can be used with the registers shown below to issue bar code commands.

Input
```
AX = 11BAh
BH = 05h
ES = Segment of command string
DI = Offset of command string
CX = Length of command string
```

Output
```
AH = BAh if the function is supported
```

The function performed depends on the content of the command string, which contains instructions that are passed to the bar code controller through the keyboard controller.

Bar Code Commands

The command string is an "escape sequence" that can consist of several individual commands. It takes the following general form:

```
<boc><esc>-y<n><alpha><n><alpha>. . .<n><capital alpha><eoc>
```

Where:

<boc>	Is beginning of command marker (F0 hex 04 hex). This tells the keyboard controller to open the gateway to the bar code controller.
<esc>	Is the Escape character (1B hex or 27 decimal)
-	Is an ASCII "-" character (2D hex or 47 decimal)
y	Is an ASCII "y" character (79 hex or 121 decimal)
<n>	Is a decimal number, one or two ASCII digits.
<alpha>	Is a letter of the alphabet.
<eoc>	Is the end of command marker (F0 hex 00 hex). This tells the keyboard controller to close the gateway to the bar code controller.

Notes:

1. Embedded spaces are not allowed in an escape sequence.
2. An individual command consists of the subsequence <n><alpha>, where <alpha> is the command operator and <n> is the operand.
3. The last <alpha> in the escape sequence must be in upper case.
4. When concatenating commands in a single command string, only one <esc>-y character sequence is required, and all <alpha>s other than the last must be in lower case.

The following is an example of a valid command string. It enables Code 39 (while disabling all other codes) and enables Code 39 check character verification in a single escape sequence.


```
<boc><esc>-y1f1G<eoc>
```

Descriptions of the individual bar code commands are provided on the following pages.

Disabling Support for Specific Bar Code Types: Application programs can disable or enable support for specific types of bar codes by using the Int 10h function call (described earlier) in conjunction with the following command:

```
<boc><esc>-y<n>F<eoc>
```

Where:

<boc>	Is the beginning of command marker (F0 hex 04 hex)
<esc>	Is the Escape character (1B hex or 27 decimal)
<n>	Can be the sum of any of the following digits: 1 = Code 39 2 = Interleaved 2 of 5 4 = UPC/EAN 8 = Codabar 16 = Code 128
<eoc>	Is the end of command marker (F0 hex 00 hex)

Note: All codes are enabled by default.

Industrial Bar Code Selection: Application programs can disable or enable support for industrial types of bar codes with the following command:

```
<boc><esc>-y<n>C<eoc>
```

Where:

<boc>	Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Can be the sum of any of the following digits:
1 = Code 39
2 = Code 39 with full ASCII conversion
4 = Interleaved 2 of 5
8 = Codabar
16 = Code 128

<eoc> Is the end of command marker (F0 hex 00 hex)

By summing different combinations of these digits, the following selections can be realized:

<n>	Codes Selected
0	No codes selected
1	Code 39 (3 of 9 code)
3	Extended Code 39
4	Interleaved 2 of 5
5	Interleaved 2 of 5 and Code 39
7	Interleaved 2 of 5 and Extended Code 39
8	Codabar
9	Codabar and Code 39
11	Codabar and Extended Code 39
12	Codabar and Interleaved 2 of 5
13	Codabar, Interleaved 2 of 5, and Code 39
15	Codabar, Interleaved 2 of 5, and Extended Code 39
16	Code 128
17	Code 128 and Code 39
19	Code 128 and Extended Code 39
20	Code 128 and Interleaved 2 of 5
21	Code 128, Interleaved 2 of 5, and Code 39
23	Code 128, Interleaved 2 of 5, and Extended Code 39
24	Code 128 and Codabar
25	Code 128, Codabar, and Code 39
27	Code 128, Codabar, and Extended Code 39
28	Code 128, Codabar, and Interleaved 2 of 5
29	Code 128, Codabar, Interleaved 2 of 5, and Code 39
31	Code 128, Codabar, Interleaved 2 of 5, and Extended Code 39

Note: All codes are enabled by default (full ASCII conversion is disabled for Code 39).

UPC/EAN/JAN Code Selection and Decoding Options: Support for UPC/EAN/JAN bar codes and their supplemental digits can be enabled and/or disabled with the following command:

```
<boc><esc>-y<n>U<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Can be the sum of any of the following digits:
1 = UPC/EAN/JAN enabled, no supplements
2 = UPC only option, no supplements
4 = 2-digit supplement option
8 = 5-digit supplement option

<eoc> Is the end of command marker (F0 hex 00 hex)

By summing different combinations of these digits, the following combinations of options can be realized:

<n>	Codes/Options Selected
1	UPC/EAN/JAN enabled, no supplements
2	UPC only enabled, no supplements
5	UPC/EAN/JAN enabled with 2-digit supplements
7	UPC only enabled with 2-digit supplements
9	UPC/EAN/JAN enabled with 5-digit supplements
11	UPC only enabled with 5-digit supplements
13	UPC/EAN/JAN enabled with 2- or 5-digit supplements
15	UPC only enabled with 2- or 5-digit supplements

Note: UPC/EAN/JAN without supplements is the default selection.

Enabling Check Character Verification: When check character verification is enabled, only bar codes containing valid check characters can be read. Application programs can enable or disable check character verification with the following command:

```
<boc><esc>-y<n>G<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Can be the sum of any of the following digits:
1 = Code 39 check character verification
2 = Interleaved 2 of 5 check character verification
8 = Check character transmission

<eoc> Is the end of command marker (F0 hex 00 hex)

Note: Check character verification is disabled by default.

Changing Decoding Options: Certain decoding options can be enabled and/or disabled with the following command:

```
<boc><esc>-y<n>H<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Can be the sum of any of the following digits:
1 = Code 39 full ASCII conversion
2 = Codabar start/stop character suppression
4 = UPC only (for UPC/EAN)
8 = UPC/EAN 2-digit supplements
16 = UPC/EAN 5-digit supplements

<eoc> Is the end of command marker (F0 hex 00 hex)

Note: These options are disabled by default.

Enabling Label Length Checking for Interleaved 2 of 5: To enable/disable label length checking, use the following escape sequence:

```
<boc><esc>-y<n>M<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Can be one of the following three types of label length checking:

- If n=0 then symbols can be of any even length between 4 and 32, inclusive.
- If n≥3 and n≤32 then only symbols of length n will be read. If an odd value is specified, it will be incremented to the next even integer since Interleaved 2 of 5 symbols must have an even number of digits.
- If n=33 then the symbols can be either 6 or 14 digits long.

<eoc> Is the end of command marker (F0 hex 00 hex)

Note: Variable length (n=0) is enabled by default.

Specifying Special Bar Code Data Formats: Application programs can request that header and/or trailer text be appended to the bar code data by using the following commands:

Header Text

```
<boc><esc>-y<n>N<text><eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Is the number of characters (0 to 10) in <text>

<text> Is a sequence of ASCII characters to be used as the header. Blank, Escape, XON, and XOFF characters *are* allowed.

<eoc> Is the end of command marker (F0 hex 00 hex)

Note: No header text is transmitted by default.

Trailer Text

```
<boc><esc>-y<n>0<text><eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Is the number of characters (0 to 10) in <text>

<text> Is a sequence of ASCII characters to be used as the trailer. Blank, Escape, XON, and XOFF characters *are* allowed. A carriage return must always be the last character in the trailer <text>.

<eoc> Is the end of command marker (F0 hex 00 hex)

Note: A carriage return (0Dh) is appended by default.

Enabling Single Read Mode: When enabled, this message pacing protocol allows a single bar code to be read only after a NEXT READ command has been issued (see the following command). This protocol allows the system to process transmissions one at a time, ensuring that no other transmission will occur until the system is ready (signalled by a NEXT READ). Upon enabling this mode, a NEXT READ command is required to allow the first read. Enable Single Read Mode with the following command:

```
<boc><esc>-y<n>J<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Can be one of the following digits:
0 = Disable Single Read Mode
1 = Enable Single Read Mode

<eoc> Is the end of command marker (F0 hex 00 hex)

Note: Single Read Mode is disabled by default.

Next Read: This command is used in conjunction with Single Read Mode to allow a single bar code to be read. See "Enabling Single Read Mode" for more information. Issue the NEXT READ command with the following escape sequence:

```
<boc><esc>-y<n>K<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Can be one of the following digits:
0 = Allow next bar code read
1 = Allow next bar code read

<eoc> Is the end of command marker (F0 hex 00 hex)

Note: Next read not allowed (upon enabling Single Read Mode).

Requesting a Status Report: This command causes a status message to be returned in the form of an escape sequence, followed by a string of characters ending with a carriage return.

To request status, use the following command:

```
<boc><esc>-y<n>$<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<n> Must be 1.

<eoc> Is the end of command marker (F0 hex 00 hex)

Within the returned string, status information is contained in the lower nibble of each byte. The upper nibble is set to the value 3 hex, so that the byte as a whole forms an ASCII character in the range of 30 hex to 3F hex. The parity bit is included.

The following chart shows the status nibble corresponding to each possible status byte transmitted in the string.

ASCII Character	Binary Representation Upper Nibble	Lower Nibble
0	P011	0000
1	P011	0001
2	P011	0010
3	P011	0011
4	P011	0100
5	P011	0101
6	P011	0110
7	P011	0111
8	P011	1000
9	P011	1001

ASCII Character	Binary Representation Upper Nibble	Lower Nibble
:	P011	1010
;	P011	1011
<	P011	1100
=	P011	1101
>	P011	1110
?	P011	1111

The individual characters and the information they contain are shown in the following chart.

Character Number	Character	Status Information
1	esc (escape character) hexadecimal 1B or decimal 27	
2	\ (backslash character) hexadecimal 5C or decimal 92	
3	Lower nibble = upper nibble of software version number	
4	Lower nibble = lower nibble of software version number	
5	Lower nibble contains last code read	*Bit Meaning (0=no, 1=yes)* **0** Code 39 last read **1** Interleaved 2 of 5 last read **2** UPC/EAN/JAN last read **3** Codabar last read
6	Lower nibble contains last code read and remaining status	*Bit Meaning (0=no, 1=yes)* **0** Code 128 last read **1** Not used **2** Last check char. was valid **3** Scanner type (0=moving beam, 1=fixed beam)
7	carriage return	Terminator character

Resetting the Bar Code Controller: This command returns the bar code controller to its initial power up condition. All buffers are cleared and the self test is performed. Upon issuing this command, the bar code reader will be inoperative for approximately one second while the system resets.

To reset the bar code controller, use the following escape sequence:

```
<boc><esc>E<eoc>
```

Where:

<boc> Is the beginning of command marker (F0 hex 04 hex)

<esc> Is the Escape character (1B hex or 27 decimal)

<eoc> Is the end of command marker (F0 hex 00 hex)

Operator Feedback: Operator feedback is the responsibility of the application. The application should specify that the bar code contain a unique header and/or trailer (see "Specifying Special Bar Code Data Formats" on page 4-12) and then monitor the keyboard data stream for this data. When the application detects bar code input, the program should sound a tone on the speaker to indicate a successful read.

Section 5. Touch Panel

Description

The touch panel employed by the Type 7690 is a low resolution pointing device that can resolve a touch point to a single column/row coordinate. A maximum of 79 discreet points can be detected in the horizontal plane, and a maximum of 47 discreet points can be detected in the vertical plane.

The touch panel is based on an infrared technology: it uses light emitting diodes (LEDs) and light sensitive detectors that are mounted behind the black translucent bezel that borders the LCD. Each LED in the horizontal and vertical plane is aligned with a corresponding light detector at the opposite side of the LCD. A touch point is detected when a horizontal and vertical light beam is interrupted.

The smallest object that can be detected by the touch panel is equal to the distance between the outside edges of two adjacent LEDs, as shown below.

Horizontal 0.792 cm (0.312 in.)

Vertical 0.927 cm (0.365 in.)

Touch Panel Registers

The following charts detail the bit assignments for the touch panel registers F300h through F303h.

Bit	I/O	Signal Name
7	O	Driver/Receiver Y-8
6	O	Driver/Receiver Y-4
5	O	Driver/Receiver Y-2
4	O	Driver/Receiver Y-1
3	O	Driver/Receiver X-8
2	O	Driver/Receiver X-4
1	O	Driver/Receiver X-2
0	O	Driver/Receiver X-1

Figure 5-1. F300h Touch Panel Data

Bit	I/O	Signal Name
7	I	Touch Panel Not Installed
6	I	-Sel Data All High
5	I	Keyboard Not Installed
4	I	Reserved
3	I	Scanner Type (1=Wand, 0=Scanner)
2	I	ADC Check (Input over 1/2 scale)
1	I	LCD Power OK (-22V)
0	I	+12V OK (-Battery Power/Fuse Open)

Figure 5-2. F301h Diagnostic Data

Bit	I/O	Signal Name
7	O	Reserved
6	O	Reserved
5	O	Reserved
4	O	Reserved
3	I/O	+ADC Convert Busy/+Start ADC
2	I/O	+Receiver Select
1	I/O	+Driver Enable
0	I/O	+LED Select

Figure 5-3. F302h Touch Panel Controls

Bit	I/O	Signal Name
7	O	Reserved
6	O	Reserved
5	O	+ADC 2.500V
4	O	+ADC 1.250V
3	O	+ADC 0.625V
2	O	+ADC 0.313V
1	I/O	+ADC 0.156V
0	I/O	+ADC 0.078V

Figure 5-4. F303h Touch Panel ADC Output

Touch Panel Device Driver

This section describes the interfaces to the Touch Panel Device Driver (LEDTOUCH.SYS). The device driver provides a standard DOS device read interface as well as an Interrupt 33 (hex) mouse interface for the touch panel. The device name, as known to DOS, is **POINT**.

Device Driver Installation

The device driver is named **LEDTOUCH.SYS**, and is provided on the Customer-Level Diagnostic Diskette, which is included with the IBM 7690 Clinical Workstation *Guide to Operations* (SA12-7007). It requires 4Kb of RAM for program and data, and occupies 7.5Kb of disk space. It is a DOS device driver and must be loaded via the DOS CONFIG.SYS file. Refer to the DOS manual for information about using the CONFIG.SYS file.

The basic installation procedure follows:

1. Copy the device driver (LEDTOUCH.SYS) from the Customer-Level Diagnostic Diskette to the System (boot) diskette.
2. Make sure a CONFIG.SYS file exists on the System diskette, and add the following line to the file:

   DEVICE=LEDTOUCH.SYS
3. The device driver will be loaded by DOS when the system is started (booted) from the System diskette.

The Customer-Level Diagnostic Diskette contains a DOS batch file (LEDINST.BAT) that simplifies the device driver installation for end users. The use of this batch file is documented in the workstation *Guide to Operations*.

DOS Device Programming Interface

The POINT device is an input-only character device. It does not directly support the IOCTL function. The current touch point is returned as an ASCII text string followed by a single carriage return. The carriage return character is removed by DOS. The format of the return string is:

```
0               No touch point returned
1,HHH,VVV       Active touch point returned
```

Figure 5-5. Touch Screen Data Format

Where:

0 Indicates a touch point is not active. The horizontal and vertical coordinates will not be returned in this case.

1 Indicates a touch point is currently active. The horizontal and vertical coordinates will follow, in ASCII, separated by commas.

HHH The horizontal touch coordinate. This will be a value from 001 to 080 and indicates the horizontal character position of the current touch point.

VVV The vertical touch coordinate. This will be a value from 001 to 030 and indicates the vertical line position of the current touch point.

Note: The maximum number of lines is determined by the current video mode.

For example, (in video mode 0):

```
0               A touch point is not active.
1,001,001       The upper left corner.
1,080,025       The lower right corner.
1,080,001       The upper right corner.
1,001,025       The lower left corner.
1,040,012       The screen center.
```

BASIC and BASIC Compiler Programming Interface

The following BASIC program sample demonstrates the DOS Device programming interface.

```
100 OPEN "POINT" FOR INPUT AS #1    ' Open the input file
200 LINE INPUT #1,C$                ' Read a coordinate
300 IF C$="0" GOTO 200              ' No Point if zero
400 X=VAL(MID$(C$,3,3))             ' Get the horiz coord.
500 Y=VAL(MID$(C$,7,3))             ' Get the vert coord
600 LOCATE Y,X,1,1,8                ' Display a cursor
```

Figure 5-6. Touch Panel BASIC Programming Interface

User Interface for LCD Control

When the LEDTOUCH device driver is polling the touch panel for input, the user can change certain characteristics of the LCD by touching different sections of the display while holding down the right-hand Shift key on the keyboard.

Touch Area	Effect on LCD
Left Side	Toggles the display between a black foreground on a white background, and a white foreground on a black background.
Right Side	Controls vertical centering of the display (except when in video mode 11). Touch the right side of the display and slide your finger up to move the active display area up; slide your finger down to move the active display area down. Display data will not move outside the display bounds.

Mouse Emulation Programming Interface

The mouse interface emulates the MicroSoft® mouse driver, which uses software interrupt 33h.

Touch coordinates are returned scaled from 0 to 639 horizontally and 0 to 199 vertically. Mouse buttons are emulated by the following keyboard keys:

Mouse Button	Keyboard Key
Left Button	Left Ctrl Key
Center Button	Left Shift Key
Right Button	Left Alt Key

To call the mouse emulator from an assembler language program, you must first load the AX, BX, CX and DX registers with parameters and then execute a software interrupt 33h.

The following are the functions supported by the mouse interface.

Reset (0)

Input AX = 0

Output AX = Status (-1 = Installed, 0 = Not Installed)

This function resets the driver variables to their initial values. It also returns the necessary information to determine if the driver is installed.

Variable	Value
Cursor Level	-1
Graphics Cursor shape	Arrow
Graphics Cursor Hot Spot	-1,-1
Text Cursor	Inverting Box
Mickey to Pixel Ratio, Horizontal	8
Mickey to Pixel Ratio, Vertical	16

Figure 5-7 (Part 1 of 2). Reset Variable Initialization

MicroSoft is a registered trademark of MicroSoft Corporation

Variable	Value
Min/Max Cursor Position, Horizontal	0/639
Min/Max Cursor Position, Vertical	0/199

Figure 5-7 (Part 2 of 2). Reset Variable Initialization

Show Cursor (1)

Input AX = 1

Output AX = Current Cursor Level

This function increments the cursor level counter and if the counter is zero, the cursor is displayed on the screen. If the cursor is already displayed, additional calls to the function will be ignored.

Hide Cursor (2)

Input AX = 2

Output AX = Current Cursor Level

This function decrements the cursor level counter. If the cursor was visible, this function will hide it. You should use this function before changing any information under the cursor.

All Hide Cursor functions should be accompanied by a corresponding Show Cursor function to properly display the cursor.

Get Touch Position and Button Status (3)

Input AX = 3

Output BX = Button Status
CX = Current Horizontal Cursor Position
DX = Current Vertical Cursor Position

This function is the general purpose polling function. BX returns the status of all three buttons in a single value:

Bit	Button
0	Left
1	Right
2	Middle

Figure 5-8. Button Status

For each button, the bit will be set to 0 if the button is up and to 1 if the button is down. CX and DX return the coordinate of the current cursor position and are guaranteed to be within the range of the logical coordinate system.

Set Touch Cursor Position (4)

Input AX = 4
CX = New Horizontal Cursor Position
DX = New Vertical Cursor Position

Output None

This function resets the current cursor position. The new values must be in the horizontal and vertical ranges of the logical coordinate system.

Get Button Press Information (5)

Input AX = 5
BX = Button (0=Left, 1=Right, 2=Middle)

Output AX = Button Status
BX = Count of Button Presses Since Last Call
CX = Horizontal Cursor Position of Last Button Press
DX = Vertical Cursor Position of Last Button Press

This function returns the information regarding a specific button. The number of button presses since the last call and the cursor coordinates of the last button press are provided. The number of button presses will always be in the range 0 to 32,767 with no detection of overflow. The count is reset after the function call.

The current button status for all buttons is also provided in AX. Refer to Figure 5-8 on page 5-8.

Get Button Release Information (6)

Input AX = 6
BX = Button (0=Left, 1=Right, 2=Middle)

Output AX = Button Status
BX = Count of Button Releases Since Last Call
CX = Horizontal Cursor Position of Last Button Release
DX = Vertical Cursor Position of Last Button Release

This function is similar to "Get Button Press Information (5)" except that it returns information about releases and not presses.

Set Min/Max Horizontal Position (7)

Input AX = 7
CX = Minimum Position
DX = Maximum Position

Output None

This function restricts all future cursor position reports to the specified region. If the touch point is outside the region defined by this call, it will be reported as the closest defined position.

If the minimum value is greater than the maximum value, the two values are interchanged.

This function is identical to Function 8 except that it is used to specify minimum and maximum horizontal coordinates.

Set Min/Max Vertical Position (8)

Input AX = 8
CX = Minimum Position
DX = Maximum Position

Output None

This function restricts all future cursor position reports to the specified region. If the touch point is outside the region defined by this call, it will be reported as the closest defined position.

If the minimum value is greater than the maximum value, the two values are interchanged.

This function is identical to Function 8 except that it is used to specify minimum and maximum horizontal coordinates.

Set Graphics cursor Block (9)

Input AX = 9
BX = Horizontal Cursor Hot Spot
CX = Vertical Cursor Hot Spot
DX = Pointer to screen and cursor mask arrays

Output None

This function sets shape, color and center of the graphic cursor. The shape and color are specified by two arrays of 16 words each. The first array specifies the screen mask and is used to remove or keep current screen content. The second array is used to define new data which is to be combined with the remaining data on the screen, if any. The two arrays must be contiguous in storage with the mask array at the lowest address.
When a graphics cursor is displayed on the screen, the following operations take place.

1. Previously saved screen data is restored if any.
2. The display area under the cursor position is saved.
3. The screen mask array is logically ANDed with the current screen image.
4. The data array is then logically XORed with the remaining screen data.

The logical result of the above operations is as follows:

Original	Mask	Data	Result
X	0	0	0 (Bit Off)
X	0	1	1 (Bit On)
0	1	0	0 (Same as Original)
0	1	1	1 (Same as Original)
1	1	0	1 (Invert Original)
1	1	1	0 (Invert Original)

Figure 5-9. Graphics Cursor Array Results

Notes:

1. This process is only performed when the cursor reporting position is moved. Repeated touches in the same position will not perform the graphics cursor display procedure to improve performance.
2. Whenever the screen content under the graphics cursor position is changed, the cursor should be turned off to prevent "restoring" incorrect data on the new screen.

Cursor Hot Spot: All function calls that refer to the cursor position give the logical coordinates of the cursor's hot spot. The hot spot is specified as the offset from the upper left corner of the bit map (0,0). The hot spot defines a pixel associated with the cursor bit map that is used for the purpose of reporting location. For example, the tip of an arrow cursor or the center of a cross cursor could be defined as the hot spot.

Note: The hot spot does not need to be within the cursor bit map. The default arrow cursor hot spot is one pixel to the left and one pixel up from the upper left corner of the bit map (-1,-1).


```
        dw      0ffffh          ; 1111111111111111
        dw      03fffh          ; 0011111111111111
        dw      03fffh          ; 0011111111111111
        dw      00fffh          ; 0000111111111111
        dw      00fffh          ; 0000111111111111
        dw      003ffh          ; 0000001111111111
        dw      003ffh          ; 0000001111111111
        dw      000ffh          ; 0000000011111111
        dw      0003fh          ; 0000000011111111
        dw      0003fh          ; 0000000011111111
        dw      003ffh          ; 0000001111111111
        dw      033ffh          ; 0011001111111111
        dw      033ffh          ; 0011001111111111
        dw      0c3ffh          ; 1100001111111111
        dw      0c3ffh          ; 1100001111111111
        dw      0ffffh          ; 1111111111111111
;
        dw      00000h          ; 0000000000000000
        dw      08000h          ; 1000000000000000
        dw      0c000h          ; 1100000000000000
        dw      0e000h          ; 1110000000000000
        dw      0f000h          ; 1111000000000000
        dw      0f800h          ; 1111100000000000
        dw      0fc00h          ; 1111110000000000
        dw      0fe00h          ; 1111111000000000
        dw      0ff00h          ; 1111111100000000
        dw      0ff80h          ; 1111111110000000
        dw      0f800h          ; 1111100000000000
        dw      08c00h          ; 1000110000000000
        dw      00c00h          ; 0000110000000000
        dw      00600h          ; 0000011000000000
        dw      00600h          ; 0000011000000000
        dw      00000h          ; 0000000000000000
```

Figure 5-10. Default Graphics Cursor

Set Text Cursor (10)

Input AX = 10 (0Ah)
BX = 0-Software Cursor or 1-Hardware Cursor
CX = Screen Mask or Scan Line Start
DX = Cursor Data or Scan Line Stop

Output None

This function selects the software or hardware text cursor. If the software cursor is selected (BX=0), this function defines the screen mask and cursor data. If the hardware cursor is specified (BX=1),

the hardware is set up with the first and last scan lines defining the cursor.

Software Text Cursor: The cursor appearance is derived by modifying the character and attributes on the screen. Each character is defined by two bytes. The high order byte (bits 8-15) define the attribute while the low order byte (bits 0-7) defines the character code point.

The screen mask is first used to modify the current character by ANDing the mask with the current contents. The cursor is then displayed by XORing the remaining screen data with the cursor data. For example, to use an up-arrow for a text cursor, the following masks would be used:

```
Screen Mask (CX) = 0000h (Current values ignored)
Cursor Data (DX) = 0F18h (Up-arrow = 18h)
```

Figure 5-11. Default Text Cursor

Read Motion Counters (11)

Input AX = 11 (0Bh)

Output CX = Horizontal Counter
DX = Vertical Counter

This function returns the horizontal and vertical relative movement counts since the last request. Positive counts indicate movement to the right or down. Negative counts indicate movement to the left or up.

Set User Defined Subroutine Mask (12)

Input AX = 12 (0Ch)
CX = Event Mask

Bit Condition
0 - Cursor Position Changed
1 - Left Button Pressed
2 - Left Button Released
3 - Right Button Pressed
4 - Right Button Released
5 - Middle Button Pressed
6 - Middle Button Released
7-15 Not Used

DX = User Subroutine Offset
ES = User Subroutine Segment

Output None

This function sets the address of a user subroutine to be called when a specified event occurs. The running program is temporarily interrupted and the specified subroutine is called whenever one or more of the conditions specified by the event mask occur.

The event mask defines the conditions that will cause an interrupt. Each bit in the mask corresponds to a specific test condition. For an event to cause a call to your subroutine, the specified bit must be set to a one (1). More than one event can occur during a single call to the specified subroutine. All event test conditions are automatically disabled by Function 0.

When the user subroutine is called, the following information is passed:

AX = Event ID Word. Bits set as in the Event Mask.
BX = Button State
CX = Horizontal Cursor Position
DX = Vertical Cursor Position
SI = Raw Horizontal Motion Counts
DI = Raw Vertical Motion Counts

Note: The user subroutine must use a FAR RETurn. That is, it must be defined as a PROC FAR.

No-Operation (13)

Input AX = 13 (0Dh)

Output None

This function call is ignored.

No-Operation (14)

Input AX = 14 (0Eh)

Output None

This function call is ignored.

Set Mickey/Pixel Ratio (15)

Input AX = 15 (0Fh)
CX = Horizontal Ratio
DX = Vertical Ratio

Output None

This function sets the relative magnitude of the value returned for touch points in different areas of the screen.

No-Operation (16)

Input AX = 16 (10h)

Output None

This function call is ignored.

No-Operation (17)

Input AX = 17 (11h)

Output None

This function call is ignored.

Set Large Graphics Cursor Block (18)

Input AX = 18 (12h)

Output AX = 0 (Function not supported)

A 0 is returned in AX to indicate this function is not supported.

No-Operation (19)

Input AX = 19 (13h)

Output None

This function call is ignored.

Read Light Pen Position (88)

Input AX = 88 (58h)

Output AH = Touch Active
BX = Pel Column
CX = Raster Line
DH = Character Line
DL = Character Column

This function call is used to return light pen information similar to the Video INT 10H read light pen position call.

Set Mouse Button Following (89)

Input AX = 89 (59h)
BX = Release Buttons
CX = Tracking Buttons

Output None

This function call is used to select which buttons, if any, will be used to indicate the active touch status. The button(s) specified in BX will be activated once when the touch point is removed. The button(s) specified in CX will be activated when touch is in process and will be inactive when the touch is removed.

Bit	Button
0	Left
1	Right
2	Middle

The default is no buttons for release and no buttons for tracking. The data specified by this call is not reset during an Initialize function call. That is, whatever button data is selected by this function will remain active until changed by a subsequent call to this function.

Section 6. System BIOS

System BIOS Usage

The basic input/output system (BIOS) resides in ROM on the system board and provides device-level control for the major I/O devices in the system. Additional ROM modules may be located on adapters to provide device-level control for that adapter. BIOS routines enable the assembler language programmer to perform block or character-level I/O operations without concern for device address and operating characteristics. System services, such as memory size determination, are provided by the BIOS.

The goal of BIOS is to provide an operational interface to the system and relieve the programmer of the concern about the characteristics of hardware devices. The BIOS interface insulates the user from the hardware, allowing new devices to be added to the system, yet retaining the BIOS level interface to the device. In this manner, hardware modifications and enhancements are transparent to user programs.

The IBM Personal Computer *Macro Assembler* manual and the IBM Personal Computer *Disk Operating System (DOS)* manual provide useful programming information related to this section. A description of the BIOS interface is given in this section.

Access to BIOS is through the 8086 software interrupts. The software interrupts hex 10 through 1A each access a different BIOS routine. For example, to determine the amount of memory available in the system,

INT 12H

invokes the BIOS routine for determining memory size and returns the value to the caller.

Parameter Passing

All parameters passed to and from the BIOS routines go through the microprocessor registers. The description of each BIOS function shows the registers used on the Call and the Return. For the memory size example, no parameters are passed. The memory size, in 1K increments, is returned in the AX register.

If a BIOS function has several possible operations, the AH register is used as input to indicate the desired operation. For example, to set the time of day, the following code is required:

```
MOV  AH,1           ;Function is to set time of day
MOV  CX,HIGH_COUNT ;Establish the current time
MOV  DX,LOW_COUNT
INT  1AH            ;Set the time
```

To read the time of day:

```
MOV  AH,0           ;Function is to read time of day
INT  1AH            ;Read the timer
```

Generally, the BIOS routines save all registers except for AX and the flags. Other registers are modified on return only if they are returning a value to the caller. The exact register use is in the description of each BIOS function.

Hardware Interrupts

For the hardware interrupt assignments, see the Software Interrupt Listing table later in this section. Interrupt level 0 corresponds to interrupt vector 8, level 1 to interrupt vector 9, and so forth, including interrupt level 7, which corresponds to interrupt vector 0F.

For information about sharing interrupts, see "Interrupt Sharing" in Section 1.

Software Interrupts

With software interrupt sharing, it is possible for software interrupt routines to "daisy chain" BIOS interrupts hex 10 through 1F, similar to hardware interrupt routines. The interrupt routine must check the function value in AH, and if the value is not in the routine's range of function calls, the interrupt routine transfers control to the next routine in the chain.

Int	Address in Hex	Name
0	0-3	Divide by Zero
1	4-7	Single Step
2	8-B	Non-Maskable
3	C-F	Breakpoint
4	10-13	Overflow
5	14-17	Print Screen
6	18-1B	Reserved
7	1C-1F	Reserved
8	20-23	Timer
9	24-27	Keyboard
A	28-2B	Reserved
B	2C-2F	Communications
C	30-33	Communications
D	34-37	Reserved
E	38-3B	Diskette
F	3C-3F	Printer
10	40-43	Video BIOS
11	44-47	Equipment Check
12	48-4B	Memory
13	4C-4F	Diskette
14	50-53	Communications
15	54-57	System Services
16	58-5B	Keyboard
17	5C-5F	Printer
18	60-63	Resident BASIC
19	64-67	Bootstrap
1A	68-6B	Time of Day
1B	6C-6F	Keyboard Break
1C	70-73	Timer Tick
1D	74-77	Video
1E	78-7B	Diskette Parameters
1F	7C-7F	Video Graphics Characters
40	100-103	Diskette Pointer Save Area
41	104-107	Reserved
42	108-10B	Video
43	10C-10F	Character Graphics Table
46	118-11B	Reserved
4A	128-12B	Reserved
60-67	180-19F	Reserved for User Programs

Figure 6-1. Software Interrupt Listing

Vectors with Special Meanings

Interrupt Hex 1B - Keyboard Break Address: This vector points to the code to be used when the Ctrl and Break keys are pressed on the keyboard. The vector is invoked while responding to the keyboard interrupt, and control is returned through an IRET instruction. The power-on routines initialize this vector to an IRET instruction; nothing occurs when the Ctrl and Break keys are pressed unless the application program sets a different value.

Control may be retained by this routine, with the following considerations. The Break may have occurred during interrupt processing, so that one or more End of Interrupt commands must be sent to the interrupt controller. Also, all I/O devices should be reset in case an operation was underway at that time.

Interrupt Hex 1C - Timer Tick: This vector points to the code to be executed on every timer-tick interrupt. This vector is invoked while responding to the timer interrupt, and control is returned through an IRET instruction. The power-on routines initialize this vector to point to an IRET instruction; nothing occurs unless the application modifies the pointer. It is the responsibility of the application to save and restore all registers that are modified.

Interrupt Hex 1D - Video Parameters: This vector is maintained for compatibility with earlier IBM display adapters. For the current video parameters, see "Alternate Parameter Table" in Section 1.

Interrupt Hex 1E - Diskette Parameters: This vector points to a data region containing the parameters required for the diskette drive. The power-on routines initialize the vector to point to the parameters contained in the BIOS diskette routine. These default parameters represent the specified values for any IBM drives attached to the system. Changing this parameter block may be necessary to reflect the specifications of other drives attached.

Interrupt Hex 1F - Graphics Character Extensions: When operating in graphics modes 4, 5, and 6, the read/write character interface forms the character from the ASCII code point, using a set of dot patterns. The dot patterns for the first 128 code points are contained in BIOS. To gain access to the second 128 code points, this vector must be established to point at a table of up to 1024 bytes, where each code point is represented by 8 bytes of graphics information. At power-on, this vector is initialized to 000:0, and it is the responsibility of the application to change this vector if additional code points are needed.

Interrupt Hex 40 - Reserved: The BIOS routines use interrupt hex 40 to revector the diskette pointer.

Other Read/Write Memory Usage: The BIOS routines use 256 bytes of memory from absolute hex 400 to 4FF. This memory is called the BIOS data area. The routines also use an expandable memory segment called the extended BIOS data area. Location hex 40E and 40F in the BIOS data area points to the extended data area. Both memory segments are defined later in this section.

Memory locations hex 300 to 3FF are used as a stack area during the power-on initialization and during bootstrap when it receives control from power-on. The application can set its own stack area.

Interrupt	Address	Function
20	80-83	DOS Program Terminate
21	84-87	DOS Function Call
22	88-8B	DOS Terminate Address
23	8C-8F	DOS Ctrl Break Exit Address
24	90-93	DOS Irrecoverable Error Vector
25	94-97	DOS Absolute Read
26	98-9B	DOS Absolute Write
27	9C-9F	DOS Terminate, Fix in Storage
28-3F	A0-FF	Reserved for DOS
40-5F	100-17F	Reserved for BIOS
60-67	180-19F	Reserved for User Program Interrupts
68-6F	1A0-1BF	Reserved
70	1C0-1C3	Reserved
71-7F	1E0-1FF	Reserved
80-85	200-217	Reserved for BASIC
86-F0	218-3C3	Used by BASIC Interpreter while BASIC is Running
F1-FF	3C4-3FF	Reserved

Figure 6-2. BASIC and DOS Interrupts

Address	Mode	Function
400-4A0	BIOS	See BIOS Data Area
4A1-4EF		Reserved
4F0-4FF		Reserved as Inter-application Communication Area for any Application
500-5FF		Reserved for DOS and BASIC
500	DOS	Print Screen Status Flag Store 00 = Print Screen Not Active, or Successful Print Screen Operation 01 = Print Screen in Progress FF = Error Encountered during Print Screen Operation
504	DOS	Single Drive Mode Status Byte
510-511	BASIC	BASIC Segment Address Store
512-515	BASIC	Clock Interrupt Vector Segment:Offset Store
516-519	BASIC	Break Key Interrupt Vector Segment:Offset Store
51A-51D	BASIC	Error Interrupt Vector Segment:Offset Store

Figure 6-3. Reserved Memory Locations

Offset	Length	Function
2E	2	Line Number of Current Line being Executed
30	2	Offset into Start of Program Text
4E	1	Character Color in Graphics Mode*
6A	1	Keyboard Buffer Contents 0 = No Characters in Buffer 1 = Characters in Buffer
347	2	Line Number of Last Error
358	2	Offset into Start of Variables (End of Program Text 1-1)

* Set to 1, 2, or 3 to get text in colors 1-3. The default is 3.

Figure 6-4. BASIC Workspace Variables

Starting Address	Function
00000	BIOS Interrupt Vectors
00080	Available Interrupt Vectors
00400	BIOS Data Area
00500	User Read/Write Memory
F0000	Read-Only Memory

Figure 6-5. BIOS Memory Map

BIOS Programming Considerations

Warning: When using an in-circuit emulator in place of the system microprocessor, take care to prevent the request/grant signals between the emulator and the system support gate array from getting out of synchronization, or damage to the gate array will result.

The BIOS code is invoked through software interrupts. The programmer should not "hard code" BIOS addresses into application programs. The internal workings and absolute addresses within BIOS are subject to change without notice.

If an error is reported by the diskette code, reset the drive adapter and retry the operation. A specified number of retries may be needed on diskette reads to ensure that the problem is not due to motor startup or head settling.

When altering I/O-port bit values, the programmer should change only the bits necessary to the current task. When finished, the programmer should restore the original environment. Not following these guidelines may cause incompatibility with present and future applications.

Adapters with System-Accessible ROM Modules: The ROM BIOS provides a means to integrate ROM code on adapters into the system's code. During the POST, interrupt vectors are established for the BIOS calls. After the default vectors are in place, a scan for additional ROM modules takes place. At this point, a ROM routine on the adapter gains control and establishes or intercepts interrupt vectors to hook itself into the system's code.

During POST, the absolute addresses hex C0000 through EFFFF are scanned in 2K increments searching for valid adapter ROM. Addresses hex C0000 through C7FFF are scanned before the video is initialized and hex C8000 through EFFFF are scanned at the end of POST. A valid ROM is defined as follows:

Byte 0 Hex 55
Byte 1 Hex AA

Byte 2 A length indicator representing the number of 512-byte blocks in the ROM (length/512). A checksum is also done to test the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be deemed valid.

When the POST identifies a valid ROM, it does a Far Call to byte 3 of the ROM (which should be executable code). The adapter may now perform its power-on initialization tasks. The feature ROM should return control to the BIOS routines by executing a Far Return.

Interrupt Interface Listing

The following contains the BIOS interrupts and the registers used on the Call and Return.

Interrupt 02H - Non-Maskable Interrupt Routine

This routine attempts to find the storage location containing the bad parity. If found, the segment address is displayed; if not found, four question marks are displayed. An NMI is generated by a system memory or I/O channel memory failure.

Interrupt 05H - Print Screen

This interrupt is invoked to print the screen contents. The cursor position at the time this routine is invoked is saved and restored upon completion. The routine is intended to run with interrupts enabled. If a subsequent Print Screen key is pressed while this routine is printing, it is ignored. The base printer status is checked for Not Busy and Not Out of Paper. An initial status error aborts the print request. Address 50:00 contains the status of the print screen:

```
50:0    = 00    Print screen has not been called, or upon return
                from a call, indicates a successful operation.
        = 01    Print screen is in progress - ignore this request.
        = FF    Error encountered during printing.
```

Interrupt 08H - System Timer

This routine handles the timer interrupt from channel 0 of the timer. The input frequency is 1.19318 MHz and the divisor is 65,536, resulting in approximately 18.2 interrupts every second.

The interrupt handler:

- Maintains a count of interrupts (doubleword at 40:6C) since power-on time, which may be used to establish time of day. If the system has been powered on for 24 hours, the overflow flag at 40:70 is incremented. The day counter word at 40:CE must be updated when the time counter crosses a day boundary.
- Decrements the motor control count (40:40) of the diskette. Upon reaching 0, it turns off the diskette drive motor, and resets the motor running flags.
- Invokes a user routine through interrupt hex 1C at every timer tick. The user must code a routine and place the correct address in the vector table.

Interrupt 09H - Keyboard

This routine is invoked upon the make or break of every keystroke.

For ASCII keys, when a make code is read from port hex 60, the character and scan codes are placed in the keyboard buffer (40:1E for a length of 32 bytes) at the address pointed to by the buffer tail pointer word at 40:1C. The buffer tail pointer is then increased by 2 unless it wraps past the end of the buffer, in which case it is reinitialized to the start of the buffer.

For shift keys, the keyboard flags are updated accordingly on makes or breaks.

For the Ctrl-Alt-Delete sequence, the handler sets the memory-test-complete word at 40:72 to hex 1234 and then jumps to POST. POST checks the memory-test-complete word and does not retest memory if hex 1234 is present.

For the Pause key, the handler loops until a valid ASCII keystroke is pressed.

For the Print Screen key, interrupt 05 is invoked to print the screen.

For a Ctrl-Break sequence, the control break interrupt handler, interrupt hex 1B, is invoked.

For the System Request key, interrupt hex 15 is invoked with (AH) = 85H; for Interrupt Complete, interrupt hex 15 is invoked with (AH) = 91H.

The keyboard intercept is handled through interrupt hex 15, with (AH) = 4FH.

Note: Refer to INT 10H for bar code programming.

Interrupt 10H - Video

Four text and five graphics modes of operation are available. The text modes and three of the graphics modes are standard CGA modes.

(AH) = 00H Set Mode

The AL register contains the mode value; if bit 7 in AL is set, the video buffer is not cleared.

The cursor is not supported in graphics modes.

For the graphics modes 4, 5, 6, and 13, the font is an 8-by-8 character box that is double-scanned to generate the 8-by-16 character. The box size refers to the font supported by BIOS.

The power on default mode is 3.

Remember

The Type 7690 only supports its integral LCD with 2 colors (black and white). Video modes are remapped by the ROM on the interface adapter. Refer to Figure 1-24 on page 1-41 for details.

Mode in Hex	Type	Colors	Alpha Format	Buffer Start	Box Size	No. Pages	PEL Dimensions
0, 1	A/N	16	40-by-25	B8000	8-by-16	8	320-by-400
2, 3	A/N	16	80-by-25	B8000	8-by-16	8	640-by-400
4, 5	APA	2	40-by-25	B8000	8-by-8	1	320-by-200
6	APA	2	80-by-25	B8000	8-by-8	1	640-by-200
11	APA	2	80-by-30	A0000	8-by-16	1	640-by-480
13	APA	256	40-by-25	A0000	8-by-8	1	320-by-200

(AH) = 01H Set Cursor Type

BIOS maintains only one cursor type for all video pages. If an application requires that different cursor types be preserved for different pages, it must maintain the different types itself.

When operating with 400 scan lines, the hardware modifies the cursor type as follows:

Start line = (CH)*2
End line = [(CL)*2-1]

```
(CH) -  Bits 4-0 = Start line for cursor
        Hardware controls the cursor blink
(CL) -  Bits 4-0 = End line for cursor
```

(AH) = 02H Set Cursor Position

```
(DH,DL) - Row,column  (00,00) is upper left
(BH) - Page number (00 for graphics)
```

(AH) = 03H Read Cursor Position

```
(BH) - Page number (00 for graphics)

ON RETURN:
      (DH,DL) - Row,column of cursor for
                requested page
      (CH,CL) - Cursor mode currently set
```

(AH) = 04H Reserved

(AH) = 05H Select Active Display Page

This function is valid only for alphanumeric modes (only those modes support more than one page).

```
(AL) - New page value
       0-7 for modes 0,1
       0-3 for modes 2,3
```

(AH) = 06H Scroll Active Page Up

```
(AL) - Number of lines; input lines blanked
         at bottom of window.
       AL = 0 means blank entire window
(BH) - Attribute to be used on blank line
(CH,CL) - Row,column of upper left corner of scroll
(DH,DL) - Row,column of lower right corner of scroll
```

(AH) = 07H Scroll Active Page Down

```
(AL) - Number of lines, input lines blanked at top
         of window
       AL = 0 means blank entire window
(BH) - Attribute to be used on blank line
(CH,CL) - Row,column of upper left corner of scroll
(DH,DL) - Row,column of lower right corner of scroll
```

(AH) = 08H Read Attribute/Character at Current Cursor Position

```
(BH) - Display page (alpha)

ON RETURN:
     (AH) - Attribute of character read (alpha)
     (AL) - Character read
```

(AH) = 09H Write Attribute/Character at Current Cursor Position

These two functions, (AH) = 09H and 0AH, are similar. The function (AH) = 09 is used for the graphics modes. For the read/write character interface in graphics modes (4, 5, and 6), the characters are formed from a character image maintained in the system ROM, which contains only the first 128 characters. To read or write the second 128 characters, the user must initialize the pointer at interrupt 1F (location 0007C) to point to the table containing the code points for the second 128 characters (128-255). For the graphics modes 11 and 13, 256 graphics characters are supplied in the system ROM.

For the write character interface in graphics mode, the character count in CX produces valid results only for characters on the same row. Continuation to following lines will not produce correct results.

For graphics modes other than mode 13, if bit 7 in BL is set to 1, the color value is exclusive-OR'd with the current contents of the video memory.

```
(AL) - Character to write
(BH) - Display page (alpha)
(BL) - Attribute of character (alpha)/color
         of character (graphics)
(CX) - Count of characters to write
```

Note: The displayed output may be affected by the Color Map function. Refer to "Color Map Function" on page 7-3.

(AH) = 0AH Write Character Only at Current Cursor Position

```
(AL) - Character to write
(BH) - Display page (alpha)
(CX) - Count of characters to write
```

(AH) = 0BH Set Color Palette

This function is used to select the colors to be used in the 320-by-200 graphics modes. The following are the results for modes 4 and 5:

Color ID = 0 selects the background color (0-15)
Color ID = 1 selects the color set to be used

```
0 = green (1) / red (2) / brown (3)
1 = cyan (1) / magenta (2) / white (3)
```

(AH) = 0CH Write Dot

If 640-by-200 or 640-by-480 graphics mode is being used, then a single black or white dot is written. For 320-by-200 modes (modes 4, 5, and 6), a dot may be written using one of the four possible color choices. For mode 13, a dot may be written using one of the 256 possible color choices. In each case, the least significant bits of AL hold the color value to be used. If bit 7 of AL is set to 1, the color value is exclusive-OR'd with the current contents of the dot (except mode 13).

```
(AL) - Color value
(BH) - Display page (alpha)
(CX) - Column number
(DX) - Row number
```

(AH) = 0DH Read Dot

This function reads a dot in the same manner as Write Dot.

```
(BH) - Display page (alpha)
(CX) - Column number
(DX) - Row number

ON RETURN:
(AL) - Dot read
```

(AH) = 0EH Write Teletype to Active Display

This function call is used to provide a teletype-like interface to the video logic. The input character is written to the current cursor position, and then the cursor position is updated. If the cursor leaves the last column of the display field, the column is set to zero and the row is incremented. If the row value exceeds the display field, the cursor is placed on the last row, first column, and the entire screen is scrolled up one line. The attribute for filling the blank line is either the attribute of the previous cursor position (alpha modes) or the color 0 (graphics modes). The screen width is controlled by the previous Mode Set.

```
(AL) - Character to write
(BL) - Foreground color in graphics mode
```

(AH) = 0FH Current Video State

```
ON RETURN:
(AH) - Number of character columns on the screen
(AL) - Mode currently set (see AH=00)
(BH) - Current active display page
```

(AH) = 10H Color Palette Interface

(AL) = 00H Set Individual Register

On the Type 7690, this routine is used only to inhibit bit 3 of the attribute byte when 512 characters are active to provide eight consistent colors. The only value supported is (BX) = 0712H.

```
(BH) - Value to set
(BL) - Register to set
```


(AL) = 03H Toggle Intensify/Blinking Bit

```
(BL) = 00H  Enable intensify
     = 01H  Enable blinking
```

(AL) = 10H Set Individual Color Register

```
(BX) = Color register to set
(DH) = Red value to set
(CH) = Green value to set
(CL) = Blue value to set
```

(AL) = 12H Set Block of Color Registers

The table format is red value, green value, blue value.

```
(ES:DX) = Pointer to a table of color values
  (BX) = First color register to set
  (CX) = Number of color registers to set
```

(AL) = 15H Read Individual Color Register

```
(BX) = Color register to read
ON RETURN:
(DH) = Red value returned
(CH) = Green value returned
(CL) = Blue value returned
```

(AL) = 17H Read Block of Color Registers

The table format is red value, green value, blue value.

```
(ES:DX) = Pointer to a destination table
                for values
  (BX) = First color register to be read
  (CX) = Number of color registers to be read
```

(AL) = 1BH Sum Color Values to Grey Shades

This routine reads the red, green, and blue values found in the color registers, performs a weighted sum (30% red, 59% green, and 11% blue), then writes the result into the red, green, and blue components of the color register. The original data in each color register is not retained; if those values will be needed later, they must be preserved by the calling routine.

```
(BX) = First color register to sum
(CX) = Number of color registers to sum
```

(AH) = 11H Character Generator Load

This function initiates a Mode Set, completely resetting the video environment but maintaining the regen buffer. For a description of the options available for character loads, refer to "RAM-Loadable Fonts" in Section 1.

(AL) = 00H User Alpha Load

BH contains the value hex 10 for normal operation. If (BH) = 0EH, the characters are extended to 16-high by extending the last line of the 14-high character.

```
(ES:BP) = Pointer to user table
   (BH) = Number of bytes per character
   (BL) = Block to load
   (CX) = Count to store
   (DX) = Character offset into table
```

(AL) = 01H Reserved If called, (AL) = 04H is executed.

(AL) = 02H ROM 8-by-8 Double-Dot Font

```
(BL) = Block to load
```


(AL) = 03H Set Block Specifier

This routine is executed after loading a font to make that character font active. This routine is valid in alpha modes only. For more information on block specifier, see "RAM-Loadable Fonts" in Section 1.

When 512 characters are active, a function call with (AX) = 1000H and (BX) = 0712H should be executed to set eight consistent colors.

```
(BL) = Character generator block selects
```

(AL) = 04H ROM 8-by-16 Font

```
(BL) = Block to load
```

(AL) = 10H Reserved If called, (AL) = 00H is executed.
(AL) = 11H Reserved If called, (AL) = 04H is executed.
(AL) = 12H Reserved If called, (AL) = 02H is executed.
(AL) = 14H Reserved If called, (AL) = 04H is executed.

The following routines are designed to be called immediately after a Mode Set. Performing them at any other time will give undetermined results.

(AL) = 20H User Graphics Chars (INT 1FH - 8-by-8)

This function allows the user to set up a pointer to a font table which defines the upper half (characters 128-255) of the graphics characters (modes 4, 5, and 6).

```
(ES:BP) - Pointer to user table
```

(AL) = 21H User Graphics Characters (INT 43H)

This routine allows the user to set up a pointer to a font table used for the modes 11 and 13 graphics characters.

```
(ES:BP) - Pointer to user table
   (CX)  - Points (bytes per character)
   (BL)  - Row specifier
         = 00 - User specified in DL
         = 01  (0EH) - 14
         = 02  (19H) - 25
         = 03  (2BH) - 43
```

(AL) = 22H Reserved If called, (AL) = 24H is executed.

(AL) = 23H ROM 8-by-8 Double-Dot Font

This function loads the ROM 8-by-8 double-dot font in the INT 43H pointer.

```
(BL) = Row specifier
```

(AL) = 24H ROM 8-by-16 Font

This function loads the ROM 8-by-16 font in the INT 43H pointer.

```
(BL) = Row specifier
```


(AL) = 30H Information

```
(CX) = Points
(DL) = Rows

ON RETURN:
(ES:BP) = Pointer to table
```

(BH) = 00H Return Current INT 1FH Pointer

(BH) = 01H Return Current INT 43H Pointer

(BH) = 02H Reserved If called, (BL) = 06H is executed.

(BH) = 03H Return ROM 8-by-8 Font Pointer

(BH) = 04H Return ROM 8-by-8 Font Pointer (Top)

Returns pointer to top half (characters 128-255) of ROM 8-by-8 font.

(BH) = 06H Return ROM Alpha Alternate 8-by-16

Returns pointer to ROM 8-by-16 font. There is no alternate 8-by-16 font.

(AL) = BAH BIOS Extensions for Type 7690

The following are BIOS extensions to support the Type 7690-specific features. Just a reminder: (AH) = 11H.

(BH) = 00H Get LCD Size

```
ON RETURN:
        (AH) = BAH if the function is supported
        (BX) = Offset from scanline 0
        (CX) = Number of active scanlines
        (DX) = Model 30 video hardware status byte
        (DI) = Total number of physical scanlines
```

(BH) = 01H Set LCD Vertical Position

```
        (BL) = Offset from scanline 0

ON RETURN:
        (AH) = BAH if the function is supported
```

(BH) = 02H LCD Normal/Reverse

```
(BL) = 0  Normal video (white on black)
(BL) = 1  Reverse video (black on white)

ON RETURN:
(AH) = BAH if the function is supported
```

(BH) = 03H Backlight Control

```
(BL) = 0  Backlight OFF
(BL) = 1  Backlight ON

ON RETURN:
(AH) = BAH if the function is supported
```

(BH) = 04H Power Control

```
(BL) = 0  Low power mode
(BL) = 1  Normal mode

ON RETURN:
(AH) = BAH if the function is supported
```

(BH) = 05H Write String to Keyboard Controller

```
(ES) = Segment of string
(DI) = Offset of string
(CX) = Length of string

ON RETURN:
(AH) = BAH if the function is supported
```

This function can be used to program the bar code controller. The string is a command sequence that is passed to the bar code controller by the keyboard controller. See "Programmable Bar Code Functions" on page 4-5 for descriptions of the bar code command strings.

(AH) = 12H Alternate Select

(BL) = 20H Select Alternate Print Screen Routine

This loads the BIOS print screen pointer into INT 15H. No alternate print screen routine is supported.

(BL) = 31H Default Palette Loading During Mode Set

The color registers are not altered during Mode Set if Disable Default Palette Loading is selected. The number of color registers loaded depends on the mode selected. Mode 13 loads the first 248 color registers. All other modes load the first 16 registers. Shades of grey are supported for BW monitors, or are enabled by using BIOS call (AH) = 12H, (BL) = 33H.

```
(AL) = 00  Enable default palette loading
     = 01  Disable default palette loading

ON RETURN:
(AL) = 12H  Function supported
```

(BL) = 32H Video

This routine enables and disables the address decode for the video I/O port and regen buffer.

```
(AL) = 00  Enable video
     = 01  Disable video

ON RETURN:
(AL) = 12H  Function supported
```

(BL) = 33H Summing to Gray Shades

When enabled, summing occurs while loading the color palette during Mode Set and Color Palette Interface routines.

```
(AL) = 00  Enable summing
     = 01  Disable summing

ON RETURN:
(AL) = 12H  Function supported
```

(BL) = 35H Display Switch

When the system video adapter and expansion slot video adapter have the same BIOS data areas and hardware capabilities, they are in conflict. If POST detects the conflict, the system video adapter is disabled and the expansion slot video adapter becomes the primary adapter.

This routine allows switching the active display between these two video adapters. The following shows the procedure when initially switching to the system video adapter:

1. Initial Expansion Slot Video Off, (AL) = 00.
2. Initial System Video On, (AL) = 01.

Afterwards, switching displays is done through the sequence:

1. Switch Off Active Video, (AL) = 02
2. Switch On Inactive Video, (AL) = 03

Switching off the active video adapter disables the video function that is active at that time. The Switch State buffer saves the video state information used when that video adapter is reactivated.

Switching on the inactive video adapter enables the video function that was inactive and uses its buffer to retrieve the video information.

All subroutines under display switching return a value of hex 12 to indicate that the function is supported.

(AL) = 00H Initial Feature Video Off

```
(ES:DX) - Pointer to Switch State buffer of 128 bytes
```

(AL) = 01H Initial System Video On

(AL) = 02H Switch Active Video Off

```
(ES:DX) - Pointer to Switch State buffer
```

(AL) = 03H Switch Inactive Video On

```
(ES:DX) - Pointer to Switch State buffer saved earlier
```

(AH) = 13H Write String

CARRIER RETURN, LINE FEED, BACKSPACE, and BELL are treated as commands rather than as printable characters.

```
(ES:BP) = Pointer to string to be written
   (CX) = Character only count
   (DX) = Position to begin string, in cursor terms
   (BH) = Page number
```

(AL) = 00H Write Character String

```
(BL) = Attribute
       String is (CHAR, CHAR, CHAR, ...)
       Cursor not moved
```

(AL) = 01H Write Character String and Move Cursor

```
(BL) = Attribute
       String is (CHAR, CHAR, CHAR, ...)
```

(AL) = 02H Write Character and Attribute Strings

This function is for alpha modes only.

```
String - (CHAR, ATTR, CHAR, ATTR, ...)
Cursor not moved
```

(AL) = 03H Write Character And Attribute Strings

This function is for alpha modes only.

```
STRING - (CHAR, ATTR, CHAR, ATTR, ...)
CURSOR IS MOVED
```

(AH) = 1AH Read/Write Display Combination Code

(AL) = 00H Read Display Combination Code

Display Code	Description
00H	No Display
0BH	Analog Monochrome
0CH	Analog Color

```
ON RETURN:
     (AL) = 1AH - Function supported
     (BL) - Active display code
     (BH) - Alternate display code
```

(AL) = 01H Write Display Combination Code

```
     (BL) - Active display code
     (BH) - Alternate display code

ON RETURN:
     (AL) = 1AH - Function supported
```

(AH) = 1BH Return Functionality and Video State Information

The user buffer contains functionality and video state information as described by the requested implementation type. When the implementation type in BX is set to 0, the buffer size is 64 bytes.

```
     (BX) = Implementation type
  (ES:DI) = User buffer pointer for return of
            functionality/state information

ON RETURN:
     (AL) = 1BH - Function supported
```

(AH) = 1CH - FFH Reserved

The following is the format of the functionality and state information table for the video. The size is 64 bytes, and the offset is shown as the hex value from (DI).

Offset	Size	Description
00	Word	Offset to Static Functionality Table
02	Word	Segment to Static Functionality Table
04	Byte	Video Mode (Refer to (AH) = 00 for Supported Modes)
05	Word	Columns on Screen (No. of Char. Columns)
07	Word	Length of Regen Buffer in Bytes
09	Word	Start Address in Regen Buffer
0B	Word	Cursor Position for 8 Display Pages (Row,Col)
1B	Word	Cursor Mode Setting (Cusor Start/End Value)
1D	Byte	Active Display Page
1E	Word	Controller Address
20	Byte	CRT Mode Set
21	Byte	CRT Palette
22	Byte	Rows on Screen (No. of Char. Lines)
23	Word	Character Height (Scan Lines/Char.)
25	Byte	Display Combination Code (Active)
26	Byte	Display Combination Code (Alternate)
27	Word	No. of Colors Supported for Current Mode
29	Byte	No. of Display Pages Supported for Current Mode
2A	Byte	Scan Lines in Current Mode = 0 - 200 = 1 - 350 = 2 - 400 = 3 - 480
2B - 2C	Byte	Reserved = 0
2D	Byte	Miscellaneous State Information Bit 7,6 - Reserved Bit 5 - Blink Enabled Bit 4 - Reserved = 0 Bit 3 - Default Palette Loading Bit 2 - Monochrome Display Attached Bit 1 - Summing Active Bit 0 - Reserved = 0

Offset	Size	Description
2E - 30	Byte	Reserved
31	Byte	Video Memory Available = 0 - 64K = 1 - 128K = 2 - 192K = 3 - 256K
32	Byte	Save Pointer State Information Bits 7-5 Reserved = 0 Bit 4 Palette Override Active Bit 3 Graphics Font Override Active Bit 2 Alpha Font Override Active Bit 1 Dynamic Save Area Active Bit 0 512 Character Set Active
33 - 3F	Byte	Reserved

The following is the format of the static functionality table pointed to by the start of the functionality and state information table. The table is 16 bytes long. A bit is set to 1 if the function is supported.

	Bit	Function
Byte 0		Video Modes (3 Bytes)
	7	Mode 7
	6	Mode 6
	5	Mode 5
	4	Mode 4
	3	Mode 3
	2	Mode 2
	1	Mode 1
	0	Mode 0
Byte 1	7	Mode F
	6	Mode E
	5	Mode D
	4	Mode C
	3	Mode B
	2	Mode A
	1	Mode 9
	0	Mode 8

	Bit	Function
Byte 2	7-4	Reserved
	3	Mode 13
	2	Mode 12
	1	Mode 11
	0	Mode 10
Bytes 3-6		Reserved
Byte 7		Scan Lines Available in Text Modes
	7-3	Reserved
	2	400
	1	350
	0	200
Byte 8		Character Blocks Available in Text Modes
Byte 9		Max. Number of Active Character Blocks in Text Modes
Byte A		Miscellaneous Functions
	7	Reserved = 0
	6	Color Register, See (AH) = 10
	5	Palette, See (AH) = 10
	4	Reserved = 0
	3	Default Palette Loading, See (AH) = 12
	2	Character Font Loading, See (AH) = 11
	1	Summing
	0	Reserved = 0
Byte B		Miscellaneous Functions
	7-4	Reserved = 0
	3	DCC
	2	Blink Enabled
	1	Reserved = 0
	0	Reserved = 0
Bytes C,D		Reserved
Byte E		Save Pointer Functions
	7-5	Reserved = 0
	4	Palette Override
	3	Graphics Font Override
	2	Alpha Font Override
	1	Dynamic Save Area
	0	512 Character Set
Byte F		Reserved

The following is the format for the SAVE_TBL. All entries are doubleword. For more information, see "Alternate Parameter Table" on page 1-77.

Entry	Description
1	Video Parameter Table Pointer This must point to the video parameter table in BIOS
2	Reserved as all 0's
3	Alpha Mode Auxiliary Font Pointer This is a pointer to a descriptor table used during a mode set to select a user font in A/N mode. The table has the following format: **Size** **Description** Byte Bytes per character Byte Block to load, should be 00 for normal operation Word Count to store, should be hex 100 for normal operation Word Character offset, should be 00 for normal operation DWord Pointer to a font table Byte Displayable rows, if the value is FF, the maximum calculated value will be used; otherwise, this value is used. Byte Consecutive bytes of mode values for which this font description is to be used. The end of this stream is indicated by a byte code of FF.
4	Graphics Mode Auxiliary Pointer This is a pointer to a descriptor table used during a mode set to select a user font in graphics mode. The table has the following format: **Size** **Description** Byte Displayable rows Word Bytes per character DWord Pointer to a font table Byte Consecutive bytes of mode values for which this font description is to be used. The end of this stream is indicated by a byte code of FF.
5 - 7	Reserved as all 0's.

Interrupt 11H - Equipment Determination

This routine determines what optional devices are attached. The EQUIP_FLAG variable is set during the power-on diagnostics, using the following hardware assumptions:

```
        Port 3FA - Interrupt ID register (primary)
             2FA - Interrupt ID register (secondary)
                   Bits 7-3 are always 0
        Port 378 - Output port of printer 1
             278 - Output port of printer 2
             3BC - Output port of printer 3

ON RETURN:
       (AX) - Equipment flag
             Bit 15,14   = Number of printers attached
             Bit 13,12   = Reserved
             Bit 11,10,9 = Number of RS-232C ports attached
             Bit 8       = Reserved
             Bit 7,6     = Number of diskette drives
                           00=1, 01=2 only if bit 0 = 1
             Bit 5,4     = Initial video mode
                           00 - reserved
                           01 - 40-by-25 using color
                           10 - 80-by-25 using color
                           11 - 80-by-25 using BW
             Bit 3       = Reserved
             Bit 2       = Pointing device attached
             Bit 1       = Math coprocessor installed
             Bit 0       = IPL diskette installed
```

Interrupt 12H - Memory Size Determination

This routine returns the amount of RAM in the system as determined by the POST routines.

The following are the memory determination assumptions:

- All installed memory is functional. If the memory test during POST indicates less, that value becomes the default.
- All memory from 0 to 640K must be contiguous.

Note: The memory value returned will be the total system memory minus the 1K block of extended BIOS memory. A 640K machine will return 639K if all the memory is functioning properly.

```
ON RETURN:
        (AX) - Number of contiguous 1K blocks of memory
```


Interrupt 13H - Diskette

For operations requiring the diskette drive motor, the multitasking hook function (INT 15H, AX = 90FDH) is called. This tells the operating system that the BIOS is waiting for the motor to accelerate, allowing the operating system to perform a different task.

Before waiting for an interrupt, BIOS calls Device Busy (INT 15H, AX = 9001H), informing the operating system of the Wait. The complementary Interrupt Complete (INT 15H, AX = 9101H) is called, indicating that the operation is complete.

(AH) = 00H Reset Diskette System

This function issues a hard reset to the controller and then generates a Prepare command. The drive is recalibrated when the next drive operation is initiated.

If an error is reported by the diskette code, the appropriate action is to reset the diskette and then retry the operation.

```
        (DL) - Drive number
             Bit 7 = 0 for diskette (value checked)

ON RETURN:
        (CY) - Set indicates status is nonzero
        (AH) - Status of operation (see Read Status)

       Diskette status at 40:41 = status of operation
```

(AH) = 01H Read Status of Last Operation

```
        (DL) - Drive number
             Bit 7 = 0 for diskette (value checked)

  ON RETURN:
        (CY) - Set indicates status is nonzero
        (AH) - Status of operation
```

(AH)	Error	(AH)	Error
80	Time Out	08	Reserved
40	Seek Failure	06	Media Has Been Changed
20	General Controller Failure	04	Sector Not Found
10	CRC Error	03	Write Protect Error
0C	Unsupported Track, Sectors/Track Combination	02	Bad Address Mark
09	DMA Boundary Error	01	Invalid Function Request
		00	No Error

(AH) = 02H Read Desired Sectors into Memory

The two most significant bits in CL are the two most significant bits of the 10-bit track number.

If an error is reported by the diskette code, the appropriate action is to reset the diskette and then retry the operation.

```
        (DL) - Drive number,
                  Bit 7 = 0 for diskette (value checked)

        (DH) - Head number, (origin of 0, not value checked)
        (CH) - Track number, (origin of 0, not value checked)
        (CL) - Sector number, (origin of 1, not value checked)
        (AL) - Number of sectors (not value checked)
     (ES:BX) - Address of buffer

  ON RETURN:
        (CY) - Set indicates status is nonzero
        (AL) - Number of sectors actually transferred
        (AH) - Status of operation (see Read Status)

       Diskette status at 40:41 = status of operation
```


(AH) = 03H Write Desired Sectors from Memory

The two most significant bits in CL are the two most significant bits of the 10-bit track number. If an error is reported by the diskette code, the appropriate action is to reset the diskette and then retry the operation.

```
        (DL) - Drive number,
                  Bit 7 = 0 for diskette (value checked)
        (DH) - Head number (origin of 0, not value checked)
        (CH) - Track number (origin of 0, not value checked)
        (CL) - Sector number (origin of 1, not value checked)
        (AL) - Number of sectors (not value checked)
     (ES:BX) - Address of buffer

  ON RETURN:
        (CY) - Set indicates status is nonzero
        (AL) - Number of sectors actually transferred
        (AH) - Status of operation (see Read Status)

       Diskette status at 40:41 = status of operation
```

(AH) = 04H Verify Desired Sectors

The two most significant bits in CL are the two most significant bits of the 10-bit track number. If an error is reported by the diskette code, the appropriate action is to reset the diskette and then retry the operation.

```
        (DL) - Drive number,
                  Bit 7 = 0 for diskette (value checked)
        (DH) - Head number (origin of 0, not value checked)
        (CH) - Track number (origin of 0, not value checked)
        (CL) - Sector number (origin of 1, not value checked)
        (AL) - Number of sectors (not value checked)

  ON RETURN:
        (CY) - Set indicates status is nonzero
        (AL) - Number of sectors verified
        (AH) - Status of operation (see Read Status)

       Diskette status at 40:41 = status of operation
```

(AH) = 05H Format Desired Track

When using this function, (ES:BX) points to the buffer containing the collection of desired address fields for the track. Each field has 4 bytes with a format as follows:

Byte 0 Track number
Byte 1 Head number
Byte 2 Sector number
Byte 3 Number of bytes per sector

- 00 = 128
- 01 = 256
- 02 = 512
- 03 = 1024

There must be one entry for every sector on the track. This is used to find the requested sector during read/write access. Before formatting a diskette when there is more than one format, Set Media Type (AH = 18H) must be called. If it is not called, the default is the maximum capacity of the drive.

The two most significant bits in CL are the two most significant bits of the 10-bit track number. If an error is reported by the diskette code, the appropriate action is to reset the diskette and then retry the operation.

```
        (DL) - Drive number,
               Bit 7 = 0 for diskette (value checked)
        (DH) - Head number (origin of 0, not value checked)
        (CH) - Track number (origin of 0, not value checked)
        (AL) - Number of sectors (origin of 1, not value checked)
     (ES:BX) - Address of buffer

  ON RETURN:
        (CY) - Set indicates status is nonzero
        (AH) - Status of operation (see Read Status)

        Diskette status at 40:41 = status of operation
```


(AH) = 06H - 07H Reserved

```
  ON RETURN:
        (CY) - Set indicates error
        (AH) - Status of operation = 01 for invalid command

        Diskette status at 40:41 = status of operation
```

(AH) = 08H Read Drive Parameters

There is a parameter table for each supported media type.

```
        (DL) - Drive number,
               Bit 7 = 0 for diskette (value checked)

  ON RETURN:
     (ES:DI) - Pointer to 11 byte parameter table
               associated with the maximum supported media types
               on the drive in question.
        (CH) - Least significant 8 of 10 bits maximum number
               of tracks (origin of 0)
        (CL) - Bits 7 & 6 - 2 most significant bits of maximum
               tracks
             - Bits 5 thru 0 - maximum sectors per track
               (origin of 1)

        (DH) - Maximum head number
               (origin of 0)
        (DL) - Number of diskette drives installed

        (BH) = 0
        (BL) - Bits 7 through 4 = 0
               Bits 3 through 0 - valid drive type
                 03 = 720 K,  3.5 in, 80 track

        (AX) = 0

    If the drive number is invalid,
    ES,AX,BX,CX,DH,DI = 0 ; DL = number of drives.
          If no drives are present, DL = 0

       Diskette status 40:41 = 0 and (CY) = 0
```

(AH) = 09H - 14H Reserved

```
ON RETURN:
      (CY) - Set indicates error
      (AH) - Status of operation = 01 for invalid command

      Diskette status at 40:41 = status of operation
```

(AH) = 15H Read DASD Type

```
      (DL) - 7-bit drive number, bit 7 = 0 for diskette
               (value checked)

ON RETURN:
      (AH) = 00 - Drive not present
           = 01 - Diskette, no change line available
           = 02 - Diskette, change line available
           = 03 - Reserved

      Diskette status at 40:41 = status of operation
```

(AH) = 16H Disk Change Line Status

```
      (DL) - 7-bit drive number, bit 7 = 0 for diskette
               (value checked)

ON RETURN:
      (CY) - Set if (AH) is not zero
      (AH) = 00 - Disk change line not active
             01 - Invalid drive number
             06 - Disk change line active

      Diskette status at 40:41 = (AH) on return
```


(AH) = 17H Set DASD Type for Format

The 'disk change' line status is checked for all drives supporting the 'disk change' signal. This function is supported for compatibility purposes; however, Set Media Type for Format, (AH) = 18H, is the suggested function to use.

```
      (DL) - 7-bit drive number, bit 7 = 0 for diskette
               (value checked)
      (AL) = 04 - 720K diskette in a 720K diskette drive

ON RETURN:
      (CY) - Set indicates error
      (AH) - Status of operation
           = 01 for invalid request

      Diskette status at 40:41 = status of operation
```

(AH) = 18H Set Media Type For Format

This function is called before issuing the first Format Desired Track command. If the diskette is changed, this function is called again. The diskette must be present.

This function monitors the 'disk change' signal. If the signal is active:

1. The logic attempts to reset the signal to the inactive state.
2. If the attempt succeeds, BIOS sets the correct data rate for formatting.
3. If the attempt fails, BIOS returns the time-out error (hex 80) in AH.

There is one parameter table for each supported medium type.

(DL) - 7-bit drive number, bit 7 = 0 for diskette
(value checked)
(CH) - Least significant 8 of 10 bits, number
of tracks (origin of 0)
(CL) - Bits 7 & 6 - 2 most significant bits of number of
tracks
- Bits 5 through 0 - sectors per track
(origin of 1)

ON RETURN:
(ES:DI) - Pointer to 11-byte parameter table for
this medium type, unchanged if AH is nonzero

(CY) - Set if track and sectors/track is not supported
(AH) - Status of operation = 01 for invalid request

(AH) = 19H - FFH Reserved

ON RETURN:
(CY) - Set indicates error
(AH) - Status of operation
= 01 for invalid command

Diskette status at 40:41 = status of operation

Interrupt 14H - Asynchronous Communications

These routines provide RS-232C support.

(AH) = 00H Initialize the Communications Port

(AL) - Parameters for initialization
(DX) - RS-232C card number (0 based)

7 6 5 Baud Rate	4 3 Parity	2 Stopbit	1 0 Word Length
000 - 110	X0 - None	0 - 1	10 - 7 Bits
001 - 150	01 - Odd	1 - 2	11 - 8 Bits
010 - 300	11 - Even		
011 - 600			
100 - 1200			
101 - 2400			
110 - 4800			
111 - 9600			

ON RETURN:
(AL) - Modem status
Bit 7 = Received line signal detect
Bit 6 = Ring indicate
Bit 5 = Data set ready
Bit 4 = Clear to send
Bit 3 = Delta received line signal detect
Bit 2 = Trailing edge ring indicate
Bit 1 = Delta data set ready
Bit 0 = Delta clear to send

(AH) - Line control status
Bit 7 = Timeout
Bit 6 = Tx shift register empty
Bit 5 = Tx holding register empty
Bit 4 = Break detect
Bit 3 = Framing error
Bit 2 = Parity error
Bit 1 = Overrun error
Bit 0 = Data ready

(AH) = 01H Send Character

```
        (AL) - Character to send
        (DX) - RS-232C card number (0 based)

  ON RETURN:
        (AL) is preserved
        (AH) - Status
               Bit 7 = 1 unable to transmit
                     If bit 7 = 0 (able to transmit),
                     then bits 6 thru 0 are:
               Bit 6 = Tx shift register empty
               Bit 5 = Tx holding register empty
               Bit 4 = Break detect
               Bit 3 = Framing error
               Bit 2 = Parity error
               Bit 1 = Overrun error
               Bit 0 = Data ready
```

(AH) = 02H Receive Character

This routine waits for the character. Bits 1 through 4 have meaning only if bit 7 (Timeout) is not set.

```
        (DX) - RS-232C card number (0 based)

  ON RETURN:
        (AL) - Character received
        (AH) - Line status
               Bit 7 = Timeout
               Bit 4 = Break detect
               Bit 3 = Framing error
               Bit 2 = Parity error
               Bit 1 = Overrun error
```


(AH) = 03H Read Status

```
        (DX) - RS-232C card number (0 based)

  ON RETURN:
        (AL) - Modem status register
               Bit 7 = Received line signal detect
               Bit 6 = Ring indicate
               Bit 5 = Data set ready
               Bit 4 = Clear to send
               Bit 3 = Delta received line signal detect
               Bit 2 = Trailing edge ring indicate
               Bit 1 = Delta data set ready
               Bit 0 = Delta clear to send

        (AH) - Line status register
               Bit 7 = Timeout
               Bit 6 = Tx shift register empty
               Bit 5 = Tx holding register empty
               Bit 4 = Break detect
               Bit 3 = Framing error
               Bit 2 = Parity error
               Bit 1 = Overrun error
               Bit 0 = Data ready

        (DX) - RS-232C card number (0 based)
```

(AH) = 04H Extended Initialize

(DX) - RS-232C card number (0 based)

(AL) - Break
- 00 - No break
- 01 - Break

(BH) - Parity
- 00 - None
- 01 - Odd
- 02 - Even
- 03 - Stick parity odd
- 04 - Stick parity even

(BL) - Stop bit
- 00 - One
- 01 - Two if 6-, 7-, or 8-bit word length
  One and a half if 5-bit word length

(CH) - Word length
- 00 - 5 bits
- 01 - 6 bits
- 02 - 7 bits
- 03 - 8 bits

(CL) - Baud rate
- 00 - 110 Baud
- 01 - 150 Baud
- 02 - 300 Baud
- 03 - 600 Baud
- 04 - 1200 Baud
- 05 - 2400 Baud
- 06 - 4800 Baud
- 07 - 9600 Baud
- 08 - 19200 Baud

ON RETURN:
- (AL) - Modem status register, see (AH)=03
- (AH) - Line status register, see (AH)=03

(AH) = 05H Extended Communications Port Control

(AL) = 00H Read Modem Control Register

ON RETURN:
- (AL) - Modem status register, see (AH)=03
- (AH) - Line status register, see (AH)=03
- (BL) - Modem control register
  - Bits 7-5 Reserved = 0
  - Bit 4 = Loop
  - Bit 3 = OUT 2
  - Bit 2 = OUT 1
  - Bit 1 = Request to Send
  - Bit 0 = Data Terminal Ready

(AL) = 01H Write Modem Control Register

(BL) - Modem control register
- Bits 7-5 Reserved = 0
- Bit 4 = Loop
- Bit 3 = Out 2
- Bit 2 = Out 1
- Bit 1 = Request to Send
- Bit 0 = Data Terminal Ready

ON RETURN:
- (AL) - Modem status register, see (AH)=03
- (AH) - Line status register, see (AH)=03
- (BL) - Modem control register

(AH) = 06F - FFH Reserved

Interrupt 15H - System Services

(AH) = 00 - 4EH Reserved

```
ON RETURN:
      (CY) - Carry flag set
      (AH) = 86 invalid function
```

(AH) = 4FH Keyboard Intercept

Keyboard intercept (keyboard escape) is called asynchronously by the keyboard interrupt 09 routine. This allows for a keystroke to be changed or absorbed. Normally the system returns with the scan code unchanged, but the operating system can redirect an interrupt 15H to its own routine and do one of the following:

- Replace (AL) with a different scan code and return with the carry flag set, effectively changing the keystroke
- Process the keystroke and return with the carry flag cleared, causing the interrupt 09 routine to ignore the keystroke.

```
      (CY) - Set to change keystroke
      (AL) = Scan code

ON RETURN:
      (CY) - Carry flag set
      (AL) = Scan code
```

(AH) = 51H - 7FH Reserved

```
ON RETURN:
      (CY) - Carry flag set
      (AH) = 86H
```

(AH) = 80H Device Open

```
      (BX) = Device ID
      (CX) = Process ID
```

(AH) = 81H Device Close

```
      (BX) = Device ID
      (CX) = Process ID
```


(AH) = 82H Program Termination

```
      (BX) = Device ID
```

(AH) = 83H Event Wait

```
      (AL) = 00  Set interval
           = 01  Cancel
   (ES:BX) - Pointer to a byte in caller's memory
             that will have the most significant
             bit set as soon as possible after the
             interval expires.
   (CX,DX) - Number of microseconds to elapse before
             posting.

ON RETURN:
      (CY) - Cleared if (AL) is not zero
           - Set if function is already busy
```

(AH) = 84H Joystick Support

(DX) = 00H Read Current Switch Settings

```
ON RETURN:
      (CY) - Set if invalid call
      (AL) = Switch settings (bits 7-4)
```

(DX) = 01H Read Resistive Inputs

```
ON RETURN:
      (CY) - Set if invalid call
      (AX) = A(x) value
      (BX) = A(y) value
      (CX) = B(x) value
      (DX) = B(y) value
```

(AH) = 85H System Request Key Pressed

```
      (AL) = 00 - Make of key
           = 01 - Break of key
```

(AH) = 86H Wait

```
   (CX,DX) - Number of microseconds to elapse before
             returning to caller
```

(AH) = 87H - 8FH Reserved

```
ON RETURN:
      (CY) - Carry flag set
      (AH) = 86H
```

(AH) = 90H Device Busy

This function allows the operating system to take control when the system is about to wait for a device.

```
ON RETURN:
      (AL)   Type code (see (AH) = 91H)
```

(AH) = 91H Interrupt Complete

This function is called to tell the operating system that an interrupt has occurred. The type codes for functions 90H and 91H are in the following categories:

00 to 7F Indicates serially reusable devices. The operating system must serialize the access.

80 to BF Indicates reentrant devices; ES:BX is used to distinguish different calls (multiple I/O calls are allowed simultaneously).

C0 to FF Indicates wait-only calls; there are no complementary Posts for these Waits. They are timeout only. Times depend on the type of device.

```
(AL) - Type Code

Type  Description                          Timeout
 00  =  Disk                                Yes
 01  =  Diskette                            Yes
 02  =  Keyboard                            No
 80  =  Network                             No
     ES:BX --> Network Control Block
 FD  =  Diskette motor start                Yes
 FE  =  Printer                             Yes
 FC  =  Reserved                            Yes
```


(AH) = 92H - BFH Reserved

```
      (CY) - Carry flag set
      (AH) = 86H
```

(AH) = C0H Return System Configuration Parameters

```
ON RETURN:
    (ES:BX) = Pointer to system descriptor vector in ROM
       (CY) = Carry flag clear
       (AH) = 0
```

The following is the format of the system descriptor table.

Size	Description
Word	Length of Descriptor in Bytes, Minimum is 8 Bytes
Byte	Model Byte
Byte	Submodel Byte
Byte	BIOS Revision Level
Byte	Feature Information Byte 1 Bit 7 = 1 BIOS uses DMA channel 3 Bit 6 = 0 One interrupt controller Bit 5 = 0 No Real-time clock Bit 4 = 1 Keyboard escape sequence (INT 15H) called in keyboard interrupt (INT 09) Bit 2 = 1 Extended BIOS data area is allocated

(AH) = C1H Return Extended BIOS Data Area Segment Address

```
ON RETURN:
        (CY) = Set on error
        (ES) = Segment to extended BIOS data area
```

(AH) = C2H Pointing Device

After POST, the following default parameters are set:

Package size is set to 3 bytes.
Pointing device is disabled.
Sample rate is set to 100 reports per second.
Resolution is set to 4 counts per mm.
Scaling is set to 1:1.

When the device driver is called, the following information is on the stack (each entry is word length):

Entry	Description
1	Status (High Byte = 0) Low Byte Bit 7 1 = Y data overflow Bit 6 1 = X data overflow Bit 5 Y data, 1 = negative Bit 4 X data, 0 = positive Bits 3,2 Reserved Bit 1 1 = Right button pressed Bit 0 1 = Left button pressed
2	X Data (High Byte = 0) Low Byte - Bit 7 MSB, Bit 0 LSB
3	Y Data (High Byte = 0) Low Byte - Bit 7 MSB, Bit 0 LSB
4	Z Data (High Byte = 0) Low Byte = 0

The following are the return values for all functions of pointing device:

```
ON RETURN:
      (CY) = Set if unsuccessful operation
      (AH) = Status
             00 - No error
             01 - Invalid function call
             02 - Invalid input
             03 - Error
             04 - Reserved
             05 - No Far Call installed
             06 - Reserved
```


(AL) = 00H Enable Pointing Device

```
(BH) = 0  Disable
     = 1  Enable
```

(AL) = 01H Reset Pointing Device

(AL) = 02H Set Sample Rate

```
(BH) - Rate value
       0 -  10 reports/sec
       1 -  20 reports/sec
       2 -  40 reports/sec
       3 -  60 reports/sec
       4 -  80 reports/sec
       5 - 100 reports/sec
       6 - 200 reports/sec
```

(AL) = 03H Set Resolution

```
(BH) - Resolution value
       0 - 1 count /mm
       1 - 2 counts/mm
       2 - 4 counts/mm
       3 - 8 counts/mm
```

(AL) = 04H Read Device Type

```
(BH) = Device ID
```

(AL) = 05H Initialization

```
(BH) - Data package size
       1 - 1 Byte
       2 - 2 Bytes
       3 - 3 Bytes
       4 - 4 Bytes
       5 - 5 Bytes
       6 - 6 Bytes
       7 - 7 Bytes
       8 - 8 Bytes
```

(AL) = 06H Extended Commands

(BH) = 00H Return Status

```
ON RETURN:
      (BL) - Status Byte 1
             Bit 7 = 0 - Reserved
             Bit 6 = 0 - Stream mode
                   = 1 - Remote mode
             Bit 5 = 1 - Pointer enabled
             Bit 4 = 0 - 1:1 scaling
                   = 1 - 2:1 scaling
             Bit 3 = 0 - Reserved
             Bit 2 = 1 - Left button pressed
             Bit 1 = 0 - Reserved
             Bit 0 = 1 - Right button pressed

      (CL) - Status Byte 2
             00 - 1  count/mm
             01 - 2 counts/mm
             02 - 4 counts/mm
             03 - 8 counts/mm

      (DL) - Status Byte 3
             0A -  10 reports/sec
             14 -  20 reports/sec
             2B -  40 reports/sec
             3C -  60 reports/sec
             50 -  80 reports/sec
             64 - 100 reports/sec
             C8 - 200 reports/sec
```

(BH) = 01H Set Scaling to 1:1
(BH) = 02H Set Scaling to 2:1

(AL) = 07H Device Driver Far Call

Setting the segment and offset to all 0's cancels the device driver.

```
(ES) = Segment pointer
(BX) = Offset pointer
```


Interrupt 16H - Keyboard

(AH) = 00H Keyboard Read

The ASCII characters and the scan code are extracted from the buffer (40:1E for a length of 32 bytes). The keyboard buffer pointer (word at 40:1A) is increased by 2 or reinitialized to the start of the buffer if the pointer is already at the end.

This function returns control only upon a keystroke being available; the keystroke is removed from the buffer. If no keystroke is available, Device Busy (INT 15H, AX = 9002H) is called to tell the operating system that a keyboard loop is about to take place, allowing the operating system to perform another task. Eventually, the keyboard interrupt (INT 09) calls Interrupt Complete (INT 15H, AX = 9102H) to Post the operation complete.

```
ON RETURN:
      (AH) - Scan code
      (AL) - ASCII character
```

(AH) = 01H Keystroke Status

The keystroke is not removed from the buffer.

```
ON RETURN:
      (ZF) = Set if no code is available
           = Clear if code is available
      (AL) - ASCII character
      (AH) - Scan code
```

(AH) = 02H Shift Status

The bits in AL are set for the following conditions.

```
ON RETURN:
      (AL) - Shift status
             Bit 7 - Insert locked
             Bit 6 - Caps locked
             Bit 5 - Nums locked
             Bit 4 - Scroll locked
             Bit 3 - Alt key pressed
             Bit 2 - Ctrl key pressed
             Bit 1 - Left shift key pressed
             Bit 0 - Right shift key pressed
```

(AH) = 03H Set Typematic Rate

(AL) = 05H Set Typematic Rate and Delay

If the typematic rate or delay is not within the supported range, the function returns with no action taken.

```
(BH) - Delay value
(BL) - Typematic rate
```

Value in BL	Char/Sec	Value in BL	Char/Sec	Value in BL	Char/Sec
00	30.0	0B	10.9	16	4.3
01	26.7	0C	10.0	17	4.0
02	24.0	0D	9.2	18	3.7
03	21.8	0E	8.6	19	3.3
04	20.0	0F	8.0	1A	3.0
05	18.5	10	7.5	1B	2.7
06	17.1	11	6.7	1C	2.5
07	16.0	12	6.0	1D	2.3
08	15.0	13	5.5	1E	2.1
09	13.3	14	5.0	1F	2.0
0A	12.0	15	4.6		

Value in BH	Delay Value
0	250 ms
1	500 ms
2	750 ms
3	1000 ms

(AH) = 05H Keyboard Write

This function places an ASCII character scan code combination in the keyboard buffer as if that key had been pressed.

```
(CL) - ASCII character
(CH) - Scan code

ON RETURN:
      (AL) = 00  Successful operation
           = 01  Buffer full
```


(AH) = 10H Extended Keyboard Read

The ASCII character and the scan code are extracted from the buffer (40:1E for a length of 32 bytes). The keyboard buffer pointer (word at 40:1A) is increased by 2 or reinitialized to the start of the buffer if the pointer is already at the end.

This function returns control only upon a keystroke being available; the keystroke is removed from the buffer.

```
ON RETURN:
      (AL) - ASCII Character
      (AH) - Scan code
```

(AH) = 11H Extended Keystroke Status

This function does not remove the keystroke from the buffer.

```
ON RETURN:
      (ZF) = Set if no code is available
           = Clear if code is available

If code is available:
      (AL) - ASCII character
      (AH) - Scan code
```

(AH) = 12H Extended Shift Status

The bits in AL and AH are set for the following conditions. Only AX and the flags are changed. All other registers are preserved.

```
ON RETURN:
      (AL) - Shift status
             Bit 7 - Insert locked
             Bit 6 - Caps locked
             Bit 5 - Nums locked
             Bit 4 - Scroll locked
             Bit 3 - Alt key pressed
             Bit 2 - Ctrl key pressed
             Bit 1 - Left shift key pressed
             Bit 0 - Right shift key pressed

      (AH) - Extended shift status
             Bit 7 - SysRq key pressed
             Bit 6 - Caps Lock key pressed
             Bit 5 - Num Lock key pressed
             Bit 4 - Scroll Lock key pressed
             Bit 3 - Right Alt key pressed
             Bit 2 - Right Ctrl key pressed
             Bit 1 - Left Alt key pressed
             Bit 0 - Left Ctrl key pressed
```


Interrupt 17H - Printer

These routines provide printer support. When the printer is busy, BIOS calls Device Busy (INT 15H, AX = 90FEH) to tell the operating system that a time out loop is about to begin.

(AH) = 00H Print Character

```
      (AL) - Character to print
      (DX) - Printer to be used (0,1,2) corresponding
             to actual values in PRINTER_BASE area

ON RETURN:
      (AH) - Status
             Bit 7  - Not busy
             Bit 6  - Acknowledge
             Bit 5  - Out of paper
             Bit 4  - Selected
             Bit 3  - I/O error
             Bit 2,1 - Unused
             Bit 0  - Time out
```

(AH) = 01H Initialize the Printer Port

```
      (DX) - Printer to be used (0, 1, 2) corresponding to
             actual values in PRINTER_BASE area

ON RETURN:
      (AH) - Status - same as function 00
```

(AH) = 02H Read Status

```
      (DX) = Printer to be used (0,1,2) corresponding to
             actual values in PRINTER_BASE area

ON RETURN:
      (AH) - Status - same as function 00
```

(AH) = 03H - FFH Reserved

Interrupt 19H - Bootstrap Loader

Track 0, sector 1 is read into the boot location (segment 0 offset 7C00) and control is transferred there with the following values:

```
(CS) = 00H
(IP) = 7C00H
(DL) = Drive that boot sector was read from
```

If there is a hardware error, control is transferred to the ROM BASIC entry point.

Interrupt 1AH - Time of Day

(AH) = 00H Read System Time Counter

This function causes the timer overflow flag to be reset to 0. Timer counts occur at the rate of 1,193,180/65,536 counts per second, or about 18.2 per second.

```
ON RETURN:
      (CX) = High portion of count
      (DX) = Low portion of count
      (AL) = 0 if timer has not passed 24 hours worth
             of counts since power-on, last system time
             counter read or write
           > 0 if timer has passed 24 hours worth of
             counts since power-on, last system time
             counter read or write
```

(AH) = 01H Set System Time Counter

This function causes timer overflow flag to be reset to 0. Timer counts occur at the rate of 1,193,180/65,536 counts per second, or about 18.2 per second.

```
ON RETURN:
      (CX) - High portion of count
      (DX) - Low portion of count
```

(AH) = 0AH Read System Day Counter

```
ON RETURN:
      (CX) = Count of days since 1-1-1980
```

(AH) = 0BH Set System Day Counter

```
      (CX) = Count of days since 1-1-1980
```

(AH) = 0CH - FFH Reserved

```
ON RETURN:
      (CY) - Set for invalid function request
```

BIOS Data Area and Locations

The IBM BIOS routines use 256 bytes of memory from absolute address hex 400 to 4FF.

Address	Function
40:0	COM1 Port Address (Word)
40:2	COM2 Port Address (Word)
40:4	COM3 Port Address (Word)
40:6	COM4 Port Address (Word)
40:8	LPT1 Port Address (Word)
40:A	LPT2 Port Address (Word)
40:C	LPT3 Port Address (Word)
40:E	Extended BIOS Data Area Segment (Word)
40:10	Equipment Word (Word) 15,14 Number of Printers Attached 13,12 Reserved 11-9 Number of RS-232C Cards Attached 8 Reserved 7,6 Number of Diskette Drives 5,4 Initial Video Mode 00 = Unused 01 = 40-by-25 Color 10 = 80-by-25 Color 11 = 80-by-25 Monochrome 3 Reserved 2 Mouse Present 1 Coprocessor Installed 0 IPL Diskette Installed
40:12	Reserved
40:13	Memory Size in K Bytes (Word)
40:15	Reserved
40:16	BIOS Control Flags
40:17	Keyboard Flags (Byte) Alt and Ctrl bits are set if Alt or Ctrl keys are pressed. 7 Insert Locked 6 Caps Locked 5 Nums Locked 4 Scroll Locked 3 Alt Key Depressed 2 Ctrl Key Depressed 1 Left Shift Key Depressed 0 Right Shift Key Depressed
40:18	Keyboard Flags 1 (Byte) 7 Insert Key Pressed 6 Caps Lock Key Pressed 5 Num Lock Key Pressed 4 Scroll Lock Key Pressed 3 Pause Locked 2 SysRq Key Pressed 1 Left Alt Key Pressed 0 Left Ctrl Key Pressed
40:19	Storage For Alternate Keypad Entry (Byte)
40:1A	Pointer To Buffer Head Within Data Segment 40 (Word)
40:1C	Pointer To Buffer Tail Within Data Segment 40 (Word)
40:1E	Keyboard Buffer (32 Bytes)
40:3E	Drive Recalibration Status (Byte) 7 Working Interrupt Flag Always 0 on Return from Diskette BIOS 3 Recalibrate Drive 3 2 Recalibrate Drive 2 1 Recalibrate Drive 1 0 Recalibrate Drive 0
40:3F	Motor Status (Byte) 7 Write Operation Otherwise Read 3 Drive 3 Motor On 2 Drive 2 Motor On 1 Drive 1 Motor On 0 Drive 0 Motor On
40:40	Motor Off Counter (Byte), Decremented by Timer. When 0, All Drive Motors Turned Off

Address	Function
40:41	Status of Last Diskette Operation (Byte) 80 - Time Out 40 - Seek Failure 20 - General Controller Failure 10 - CRC Error 0C - Unsupported Track, Sectors/Track Combination 09 - DMA Boundary Error 08 - DMA Failure 06 - Media Has Been Changed 04 - Sector Not Found 03 - Write Protect Error 02 - Bad Address Mark 01 - Invalid Function Request 00 - No Error
40:42	Status Returned from Controller (7 Bytes)
40:49	Current CRT Mode (Byte) See Interrupt 10H
40:4A	Number of Columns on Screen (Word)
40:4C	Regen Buffer Length in Bytes (Word)
40:4E	Starting Address Offset of Regen Buffer (Word)
40:50	Cursor Position Page 1 (Word)
40:52	Cursor Position Page 2 (Word)
40:54	Cursor Position Page 3 (Word)
40:56	Cursor Position Page 4 (Word)
40:58	Cursor Position Page 5 (Word)
40:5A	Cursor Position Page 6 (Word)
40:5C	Cursor Position Page 7 (Word)
40:5E	Cursor Position Page 8 (Word)
40:60	Cursor Mode (Word)
40:60	End Line for Cursor
40:61	Start Line for Cursor
40:62	Current Page being Displayed (Byte)
40:63	Base Port Address for Active Display (Word)
40:65	Current Setting of the 3-by-8 Register (Byte) Mirror Image Written to Base Port Address + 4 for Set Mode
40:66	Current Palette Setting Color Card (Byte) Mirror Image Written to Base Port Address + 5
40:67 - 6B	Reserved
40:6C	Timer Counter Low Word,High Word (DWord) Increased Approximately 18 Times per Second

Address	Function
40:70	Timer Overflow (Byte) Not 0 = Timer Counted Past 24 Hours 0 = NOT
40:71	BIOS Break Flag (Byte) Bit 7 - Set if Break Key Pressed
40:72	Reset Flag (Word), If Hex 1234, Then No Need to Test Memory on POST
40:74	Reserved
40:75	Reserved
40:76	Reserved
40:77	Reserved
40:7B	Reserved
40:78	LPT1 Timeout Value (Byte)
40:79	LPT2 Timeout Value (Byte)
40:7A	LPT3 Timeout Value (Byte)
40:7C	COM1 Timeout Value (Byte)
40:7D	COM2 Timeout Value (Byte)
40:7E	COM3 Timeout Value (Byte)
40:7F	COM4 Timeout Value (Byte)
40:80	Start of Keyboard Buffer within Data Segment 40 (Word)
40:82	End of Keyboard Buffer within Data Segment 40 (Word)
40:84	Rows on the Screen (Byte)
40:85	Bytes per Character (Word)
40:87	Mode Options (Byte) = 00
40:88	Reserved
40:89	7-6 Reserved 5 1 - LCD Reverse Video Enable 0 - LCD Reverse Video Disable 4 1 - 8-by-16 Text Font 0 - 8-by-8 Text Font 3 0 - Default Palette Loading Enabled 2 0 - Color Monitor Attached — Disable Color Map Function 1 - Monochrome Attached — Enable Color Map Function 1 Video Summing Enabled 0 Reserved
40:8A	Display Combination Code 0B = BW Analog 0C = Color Analog

Address	Function
40:8B	Last Diskette (Byte) Bits 7,6 Data Rate Selected 00 = 500K bps 01 = Reserved 10 = 250K bps 11 = Reserved Bits 5,4 Step Rate Time Selected 00 = for SRT = 0C 01 = for SRT = 0D 10 = for SRT = 0A 11 = Reserved
40:8C	Reserved
40:8D	Reserved
40:8E	Reserved = 00
40:8F	Reserved
40:90	Media State Drive 0 (Byte) See Below
40:91	Media State Drive 1 (Byte) See Below Bit Description For 40:90 and 40:91 7,6 Data rate 00 - 500K bps 01 - Reserved 10 - 250K bps 11 - Reserved 5 Reserved 4 0 - Media/Drive Unestablished 3 Reserved 2-0 Reserved = 111B
40:93	Reserved
40:94	Track Currently SEEKed to, Drive 0 (Byte)
40:95	Track Currently SEEKed to, Drive 1 (Byte)
40:96	Keyboard Type (Byte) 7 Read ID in Process 6 Last Character was First ID Character 5 Force Num Lock if Rd ID and KBX 4 101/102-key Keyboard Installed 3 Right Alt Key Depressed 2 Right Ctrl Key Depressed 1 Last Code was E0 Hidden Code 0 Last Code was E1 Hidden Code
40:97	Keyboard LED Flags (Byte)
40:98	Pointer to Users Wait Flag (DWord)

Address	Function
40:9C	User Timeout Value Low Word, High Word (DWord) in Microseconds
40:A0	Reserved
40:A1-A3	Reserved
40:A4-A7	Reserved
40:A8-AB	Pointer to Alternate Parameter Table (Video)
40:AC-CD	Reserved
40:CE	Day Counter (Word)
40:D0-EF	Reserved
40:F0-FF	Reserved for User
50:00	Print Screen Status Byte

Extended BIOS Data Area

Power-on Self Test (POST) carves out the highest possible 1K of memory below 640K to be used as the extended data area. The word pointer at 40:0E in the BIOS data area points to the segment. The first byte in the extended BIOS data area is initialized to the length, in K bytes, allocated. The allocation of the data area within the carved segment is:

Offset (Hex)	Function
00	Number of bytes allocated in multiples of K (Byte)
01-21	Reserved
22-2F	Pointing device interface BIOS data area (14 Bytes)
	22 Device Driver Far Call Offset (Word)
	24 Device Driver Far Call Segment (Word)
	26 Pointing Device Flag (1st Byte)
	7 Command in Progress
	6 Resend
	5 Acknowledge
	4 Error
	3 Reserved = 0
	2-0 Index Count
	27 Pointing Device Flag (2nd Byte)
	7 Device Driver Far Call flag
	6-3 Reserved
	2-0 Package Size
	28 - 2F Reserved

ROM Tables

The following tables are located in ROM.

Asynchronous Baud Rate Initialization Table

Offset (Hex)	Size	Function
0	Word	Init value for 110 Baud
2	Word	Init value for 150 Baud
4	Word	Init value for 300 Baud
6	Word	Init value for 600 Baud
8	Word	Init value for 1200 Baud
A	Word	Init value for 2400 Baud
C	Word	Init value for 4800 Baud
E	Word	Init value for 9600 Baud

Diskette Parameter Table

Offset (Hex)	Size	Function
0	Byte	First specify byte
1	Byte	Second specify byte
2	Byte	Number of timer ticks to wait prior to turning diskette motor off
3	Byte	Number of bytes/sector = 0 128 bytes/sector = 1 256 bytes/sector = 2 512 bytes/sector = 3 1024 bytes/sector
4	Byte	Sectors/track
5	Byte	Gap length
6	Byte	Data length
7	Byte	Gap length for format
8	Byte	Fill byte for format
9	Byte	Head settle time in ms
A	Byte	Motor startup time in 1/8 seconds

Model Bytes

The model byte is located at F000:FFFE in ROM. Use the read system configuration parameters (INT 15H, AH = C0H) to find the model and submodel byte. For the Type 7690, the model byte is hex FA and the submodel is 00.

Section 7. Programming Tips and Techniques

Color Mapping Considerations

Since the LCD is a black and white display, application programs for the Type 7690 should only use black and white display attributes. The Type 7690 automatically remaps color video modes to their black and white equivalents to maintain a level of compatibility with existing color applications.

When referring to sections in this manual pertaining to video modes, keep in mind that the modes are remapped as follows:

Mode Requested	Mode Enabled
0	0
1	0
2	2
3	2
4	4
5	4
6	6
11	11
13	13

Figure 7-1. Video Mode Map

Note: Even though video mode 13 is supported, its implementation is not recommended due to the color limitations of the LCD.

Selecting Display Attributes

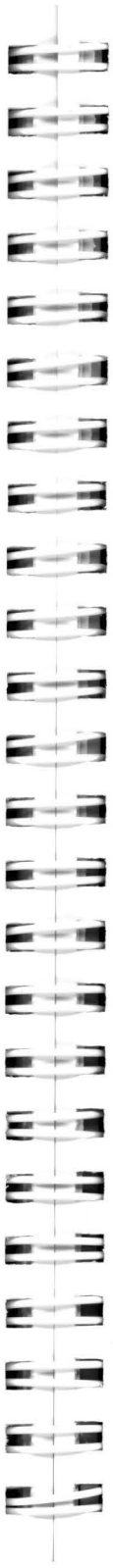
There are three methods for selecting display attributes on the Type 7690:

1. *Attribute selection for the Type 7690 only*

   If the application is intended only to run on the Type 7690, choose only black and white attributes (color 0 for black, color 7 for white).

2. *Attribute selection based on video modes*

   If the video mode is 0, 2, 4, or 6 use black and white attributes. If the video mode is 1, 3, or 5 use any of the color attribute combinations.



   This is the preferred method for applications that must run on the Type 7690 *and* a color system.

3. *Strategic attribute selection*

   The LCD interface is designed so that all EVEN colors appear black and all ODD colors appear white on the Type 7690's LCD. A strategic color application would use opposing foreground and background colors (ODD on EVEN or EVEN on ODD) that produce pleasing color combinations on a color system while producing visible characters on the Type 7690's LCD.

Even Colors		Odd Colors	
00	Black	01	Blue
02	Green	03	Cyan
04	Red	05	Magenta
06	Brown	07	White
08	Gray	09	Light Blue
0A	Light Green	0B	Light Cyan
0C	Light Red	0D	Light Magenta
0E	Light Yellow	0F	Bright White

Figure 7-2. Color Listing

Color Map Function

The Type 7690 BIOS contains a color map function that can be enabled or disabled by an application program. Refer to "BIOS Data Area and Locations" on page 6-60 (Address 40:89). This function is enabled by default. When enabled, the color map function ensures that text written through Int 10H (AH=09), with different foreground and background color combinations, is visible on the LCD.

If the application program specifies (through BIOS) an odd foreground color on an odd background color, the color map function changes the foreground to an even color. If the application specifies an even foreground color on an even background color, the color map function changes the foreground to an odd color. For example, if the application specifies white on blue, color map changes the colors to brown on blue (which would appear as black on white on the LCD).

Note: Applications that write to the screen directly (bypassing BIOS) cannot take advantage of the color map function.

Sensing a Power Loss Condition

When primary power is lost while the system is on, the battery pack is automatically activated. The Type 7690's diskette drive and display are not operational while the system is being powered by the battery pack. Therefore, application programs should monitor the system for the loss of primary power so that the appropriate actions can be taken.

The following figure shows how an application can test the power register on the I/O channel.

```
MOV     DX,0F301h       ; I/O address index
IN      AL,DX           ; Read power register
TEST    AL,1            ; Test for Power On
JNZ     POWER_ON        ; Power is ON
JZ      POWER_OFF       ; Power if OFF
```

Figure 7-3. Sample Assembler Program to Sense Loss of Primary Power

Diskette Drive

Since the diskette drive is not operational under battery power, the application should provide for unlimited retries during diskette I/O operations when primary power is lost.

Keyboard

Although the keyboard is operational under battery power, it's usefulness is questionable since the LCD will not be operational. Applications should inhibit keyboard input when primary power is lost. One exception might be to provide the user with a hot-key logoff feature to safely end the application.

Blanking the LCD

To extend the life of the LCD assembly, application programs must turn off the LCD backlight after five minutes of system inactivity. This can be done by placing the following code in a timer routine.

```
MOV     AX,11BAh        ; Extended BIOS function calls
MOV     BH,03h          ; Backlight Control
MOV     BL,0            ; Turn backlight OFF
INT     10h             ; BIOS Video Call
```

Figure 7-4. Sample Assembler Program to Blank the LCD

To turn the LCD back on, use the following code:

```
MOV     AX,11BAh        ; Extended BIOS function calls
MOV     BH,03h          ; Backlight Control
MOV     BL,1            ; Turn backlight ON
INT     10h             ; BIOS Video Call
```

Figure 7-5. Sample Assembler Program to Turn On the LCD

Power On Indicator

When the LCD backlight is turned off using the Backlight Control function call, as described above, the system ensures that at least the Scroll Lock indicator light is illuminated. This serves to remind the user that the machine is on.

If any of the keyboard indicator lights are on when the function call is issued, their states are not altered. If all indicator lights are off when the function call is issued, the Scroll Lock light is turned on. The indicator lights are returned to their original states when the LCD backlight is turned back on.

Hot Key LCD Blanking

Application programmers should consider providing the user with a method of turning the LCD backlight off before the timed five minute interval expires.

Index

Special Characters

A

B

C

D

V

W

Numerics

Federal Communications Commission (FCC) Statement

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Instructions to User: If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the computer with respect to the receiver.
- Move the computer away from the receiver.
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

Properly shielded and grounded cables and connectors must be used for connection to peripherals in order to meet FCC emission limits. Proper cables are available from IBM authorized dealers. IBM is not responsible for any radio or television interference caused by using other than recommended cables or by unauthorized modifications to this equipment. It is the responsibility of the user to correct such interference.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:

How to Identify and Resolve Radio-TV Interference Problems

This booklet is available from the following:

Consumer Assistance and
Small Business Division
Room 254
1919 M St. NW
Washington, DC 20554
Tele (202) 632-7000

FOB Public Contact Branch
Room 725
1919 M St. NW
Washington, DC 20554
Tele (202) 634-1940